COMPARATIVE ANALYSIS OF DEEP LEARNING AND GRAPH CUT ALGORITHMS FOR CELL IMAGE SEGMENTATION

by

Ghazal Reshad

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in

The Faculty of Science

Computer Science

University of Ontario Institute of Technology

August 2020

THESIS EXAMINATION INFORMATION

Submitted by: **Ghazal Reshad**

Master of Science in Computer Science

Thesis title: Comparative Analysis of Deep Learning and Graph Cut Algorithms for Cell Image Segmentation

An oral defense of this thesis took place on August 26, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Faisal Qureshi
Research Supervisor	Dr. Mehran Ebrahimi
Examining Committee Member	Dr. Kourosh Davoudi
Thesis Examiner	Dr. Jing Ren, Faculty of Engineering and Applied Science

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Image segmentation is a commonly used technique in digital image processing with many applications in the area of computer vision and medical image analysis. The goal of image segmentation is to partition an image into multiple regions, normally based on the characteristics of pixels in a given image. Image segmentation could involve separating the foreground from background in an image, or clustering image regions based on similarities in intensity, color, or shape.

In this thesis, we consider the problem of cell image segmentation and evaluate the performance of two major techniques on a dataset of cell image sequences. First, we apply a traditional segmentation algorithm based on the so-called graph cut that addresses the segmentation problem using an energy minimization scheme defined on a weighted graph. Second, we use modern techniques based on deep neural networks, namely U-Net and LSTM that have a time-consuming training and a relatively quick testing phase.

Performance of each technique will be analyzed qualitatively and quantitatively based on various standard measures and will be compared statistically.

Keywords:

Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Contribution

I have compared two different techniques applied to medical image datasets for image segmentation. The methods have been quantitatively and qualitatively compared utilizing different measures.

In chapter 3 each technique have been explained in detail. I was responsible for data pre-processing, and implementation of the Graph Cut model.

Acknowledgements

I would like to sincerely thank my supervisor Prof. Mehran Ebrahimi for his guidance, advice, and his patience during this project. He believed in me and gave me the opportunity to be part of the medical image processing community.

I would also like to thank all the members of the Computer Science program, especially those in the Imaging Lab. You have all welcomed me and guided me throughout my studies and I am grateful for all the support and kindness that I have received.

Lastly, and most importantly, I would like to thank my family without whom this journey would have not been possible. You have been with me throughout all the ups and downs and supported me no matter what!

Contents

Abstract iii

Declaration iv

Contribution v

Acknowledgements vi

1 Introduction **1**

1.1 Image Segmentation 1

1.2 Preliminary Material 2

1.2.1 Definition 2

1.2.2 Hard Segmentation and Partial-Volume Effects 4

1.2.3 Continuous or Discrete Segmentation 6

1.2.4 Interactions 6

1.2.5 Validation 6

1.2.6 Thresholding 7

1.2.7 Multi-label image classification 8

2 Background **10**

2.1 Deep Learning 12

2.1.1 Neural Networks 12

2.1.2 Optimization 16

2.1.3 Backpropagation 18

2.1.4 The Convolution Operation 21
2.1.5 Rectified Linear Unit (ReLu) 23
2.1.6 Max Pooling 23
2.1.7 Long Short-Term Memory 24
2.2 Graph Cut Segmentation 28

3 Methodology **34**
3.1 Data 35
3.2 Graph Cut Cell Segmentation 36
3.3 U-Net 39
3.4 LSTM-UNet 42
3.5 Software 45

4 Results **47**

5 Conclusion **72**

Bibliography **74**

List of Tables

4.1 Jaccard and Dice indices for LSTM-Unet, graph cut, and U-Net on Flou-N2DH-SIM+ dataset. Table on the left shows result for LSTM-Unet and the table on the middle displays result for the same series of images resulted by graph cut and the table on the right illustrate the results for U-Net. . 58

4.2 Comparing the performance of each technique using Jaccard Index 58

4.3 Comparing the performance of each technique using Dice Index 59

4.4 Detailed comparison of LSTM-Unet and graph cut performance. 59

4.5 Sensitivity and Specificity for both LSTM-Unet and graph cut on Flou-N2DH-SIM+ dataset. Table on the left shows result for LSTM-Unet and the table on the right displays result for the same series of images resulted by graph cut. 60

4.6 Comparing the performance of each technique using Sensitivity and Specificity 60

4.7 Comparing the performance of each technique using Sensitivity and Specificity for PhC-C2DH-U373 dataset 61

4.8 Jaccard and Dice indices for LSTM-Unet, graph cut, and U-Net on PhC-C2DH-U373 dataset. Table on the left shows result for LSTM-Unet and the table on the middle displays result for the same series of images resulted by graph cut and the table on the right illustrate the results for U-Net. . 67

4.9 Comparing the performance of each technique using Jaccard Index for PhC-C2DH-U373 dataset 67
4.10 Sensitivity and Specificity for both LSTM-Unet and graph cut on PhC-C2DH-U373 dataset. Table on the left shows result for LSTM-Unet and the table on the right displays result for the same series of images resulted by graph cut. 71
4.11 Comparing the performance of each technique using Dice Index for PhC-C2DH-U373 dataset 71

List of Figures

1.1 Examples of image segmentation [25, 21] 3

1.2 A histogram showing three apparent classes [44]. 8

2.1 Schematic overview of backpropagation in a simple computational graph. During forward pass, vectors x and y are inputs to a node that performs some fixed computation on its inputs producing vector z. Note that we can compute the Jacobian matrices $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ for the node at this stage. The output z flows further to the graph where at the end we calculate a loss using a differentiable scalar-valued function $\mathcal{L}$. The backward pass proceeds in the reverse order, effectively calculating the gradient of the loss with respect to all the elements int the graph using the chain rule. The gradient of the loss with respect to the vector z is calculated $\frac{\partial \mathcal{L}}{\partial z}$ and gets multiplied with the local gradients calculated during the forward pass to find the global gradient of the loss with respect to the inputs $\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial \mathcal{L}}{\partial z}$ and $\frac{\partial \mathcal{L}}{\partial y} = \frac{\partial z}{\partial y}\frac{\partial \mathcal{L}}{\partial z}$. In neural networks, each node contains parameters, where th gradient with respect to each parameter tells us how they should be changed to minimize the loss [40]. 19

2.2 Graph of **ReLu** function 22

2.3 Recurrent Neural Networks loops [1] 25

2.4 An unrolled recurrent neural network [1] 25

2.5 RNN with short term dependencies [1] 26

2.6 RNN with long term dependencies [1] 26
2.7 The repeating module in a standard RNN contains a single layer [1]. . . . 27
2.8 The repeating module in an LSTM contains four interacting layers. Each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation while a line forking denote its content being copied and the copies going to different locations [1]. 27
2.9 The graph corresponding to an arbitrary 2×3 image. 28
2.10 A random cut of graph 29
2.11 Graph construction in Greig et. al. [24]. Edge costs are reflected by thickness. 31

3.1 Initializing the graph cut using the given human-selected foreground/background labels. The small rectangle corresponds to a foreground selection and the bigger rectangle corresponds to a background selection. 38
3.2 U-Net architecture. The number of channels is denoted under the box. The arrows on top of the box represent copied feature maps [47]. 40
3.3 The LSTM-Unet network architecture. The down sampling path (left) consists of a LSTM layer followed by a convolutional layer with ReLU activation, the output is then down-sampled using max pooling and passed to the next layer. The up-sampling path (right) consists of a concatenation of the input from the lower layer with the parallel layer from the down-sampling path followed by two convolutional layers with ReLU activations [4]. 43

4.1 Series of continuous of original images from Flou-N2DH-SIM+ 48
4.2 Series of continuous of ground truth images 49

4.3 Series of continuous images segmented by LSTM-Unet 50

4.4 Same series of images as in Figure 4.3 segmented by graph cut 51

4.5 Series of continuous images segmented by U-Net 52

4.6 The case where graph cut out performs LSTM-Unet. (a) is the original image. (b) is the mask provided from ISBI Cell Tracking Challenge. (c) shows the result of LSTM-Unet and (d) is the output of graph cut. . . . 54

4.7 Displaying the difference images segmented by LSTM-Unet. (a) and (b) are the Ground Truth mask. (c) and (d) are the segmented images by LSTM-Unet, and (e) and (f) are the difference images of the result from ground truth. 55

4.8 Displaying the difference images segmented by graph cut. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by graph cut, and (e) and (f) is the difference images of the result from ground truth. 56

4.9 Displaying the difference images segmented by Unet. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by U-Net, and (e) and (f) is the difference images of the result from ground truth. . 57

4.10 PhC-C2DH-U373 dataset original images 62

4.11 PhC-C2DH-U373 dataset ground truth images 63

4.12 PhC-C2DH-U373 dataset segmented by LSTM-Unet 64

4.13 PhC-C2DH-U373 dataset segmented by graph cut 65

4.14 PhC-C2DH-U373 dataset segmented by U-Net 66

4.15 Displaying the difference images segmented by LSTM-Unet. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by LSTM-Unet, and (e) and (f) is the difference images of the result from ground truth for PhC-C2DH-U373 dataset. 68

4.16 Displaying the difference images segmented by graph cut. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by graph cut, and (e) and (f) is the difference images of the result from ground truth for PhC-C2DH-U373 dataset. 69
4.17 Displaying the difference images segmented by U-Net. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by U-Net, and (e) and (f) is the difference images of the result from ground truth for PhC-C2DH-U373 dataset. 70

Chapter 1

Introduction

1.1 Image Segmentation

Image segmentation is a fundamental problem in computer vision. In recent years, the field of image segmentation has grown and today we can see its emergence in medical imaging where segmentation and tracking has been introduced on the medical image databases to boost the diagnosis process. The objective of image segmentation is to segment or better said to partition an image into several non-overlapping regions that are deemed meaningful according to some objective criterion. Image segmentation has been a long studied problem. Medical imaging has many significant applications in the prospect of analysis and diagnostics. There is a natural need for automatic segmentation of medical images where the rate of making mistakes by human is increasing [46]. An instance of medical image segmentation application can be extracting cell images, where the goal is to first be able to detect cell objects with their true boundary being detected and segmented out from the rest of the image. A possible application would be to count the number of cells in the image which can be critical to researchers in different biological fields as well as the biomedical engineers. Another application could be to track and detect the cells in a sequence of images. Furthermore, we can apply segmentation

to recognize and analyze the various part of the cells. In this thesis, our goal is to apply different techniques from *Classical Image Segmentation methods* to modern *Deep Learning algorithms* to report a statistical analysis of the performance of each module on different datasets. Continuing in this chapter we will introduce some preliminary material regarding image segmentation. Chapter 2 will provide background information on deep learning model and graph cut relating to other parts of this thesis. Chapter 3 will be an in-depth explanation of methods that has been used and also information regarding the software and technologies for the purpose of the thesis and explaining about datasets. Chapter 4 will cover the results. Finally, in Chapter 5 we will conclude the purpose of the thesis.

1.2 Preliminary Material

1.2.1 Definition

An image is a collection of measurements in two-dimensional (2-D) or three-dimensional (3-D) space. In medical imaging, these measurements or image intensities can be radiation absorption in X-ray imaging, acoustic pressure in ultra-sound, or computed tomography (CT) scans. If more than one measurement is made the image is called a vector or multichannel image. Images may be acquired in a continuous domain or in discrete. In 2-D discrete images, the location of each measurement is called a pixel, and in 3-D image, it is called a voxel. For simplicity we use 'pixel' for both 2-D and 3-D cases.

Classically, image segmentation is defined as the partitioning of an image into non overlapping, constituent regions (Figure 1.1) that are homogeneous with respect to some characteristics such as intensity or texture [26, 41]. If the domain of an image is represented by Ω, then the segmentation problem is to determine the subsets $S_k \subset \Omega$, such that their union is the entire domain . Thus, the set that make up a segmentation must satisfy

(a) Original image

(b) Segmentation result

(c) Original image

(d) Segmentation result

(e) Original image

(f) Segmentation result

(g) Original image

(h) Segmentation result

Figure 1.1: Examples of image segmentation [25, 21]

$$\Omega = \bigcup_{k=1}^{K} S_k \tag{1.1}$$

where $S_k \cap S_j = \phi$ for $k \neq j$. Ideally, a segmentation method finds the subsets that correspond to distinct regions of interest in a given image.

When the constraint that regions be connected is removed, then the sets S_k are called *pixel classification*, and the sets themselves are called *classes*. Pixel classification, rather than classical segmentation, is often a desirable goal when dealing with medical images, particularly when disconnected regions belonging to the same tissue class require identification. Determination of total number of classes K in pixel classification can be a difficult problem [32]. Often, the value of K is assumed to be known based on prior knowledge of the anatomy being considered. For example, in the segmentation of magnetic-resonance (MR) brain images, it is common to assume that $K = 3$, corresponding to gray-matter, white-matter and cerebrospinal-fluid tissue classes [45].

Labelling is the process of assigning a meaningful designation to each region or class that can be performed separately from segmentation. It maps the numerical index k of set of S_k to an anatomical designation. In medical imaging, the labels are often visually obvious and can be determined on inspection by a physician or technician. Computer-automated labelling is desirable when labels are not obvious in automated processing systems. For example in digital mammography, in which the image is segmented into distinct regions and the regions are subsequently labelled as healthy or tumorous tissue.

1.2.2 Hard Segmentation and Partial-Volume Effects

In medical imaging where multiple tissues contribute to a single pixel or voxel resulting in a blurring of intensity across boundaries is called the partial volume-effects. There are times when it is difficult to precisely determine the boundaries of the two objects. A *hard segmentation* forces a decision of whether a pixel is inside or outside the object. Soft segmentations on the other hand, retain more information from the original image by

allowing for uncertainty in the location of object boundaries.Thus, partial volume effects can cause boundaries to be blurred or fuzzy across significant portions of an image.

In pixel classification methods, the notation of soft segmentation stems from the generalization of a set *characteristic function*. A characteristic function is simply an indicator function denoting whether a pixel is inside or outside its corresponding set. For a location $x \in \Omega$, the characteristic function of the set S_k is defined as

$$f_k(x) = \begin{cases} 1, & if \quad x \in S_k \\ 0, & otherwise. \end{cases} \tag{1.2}$$

Characteristic functions can be generalized to *membership functions* [58], which can be real-valued instead of binary. Membership functions $m_k(x)$ satisfy the following constraints

$$0 \leq m_k(x) \leq 1 \quad for\ all\ x \in \Omega, 1 \leq k \leq K. \tag{1.3}$$

$$\sum_{k=1}^{K} m_k(x) = 1 \quad for\ all\ x \in \Omega. \tag{1.4}$$

The value of membership function $m_k(x)$ can be interpreted as the contribution of class k to location x. Thus, whether membership values are greater than zero for two or more classes, those classes are overlapping. Conversely, if the membership function is unity for some x and k, then class k is the only contributing class at location x. Membership functions can be derived by using fuzzy clustering and classifier algorithms [43, 28] or statistical algorithms in which case the membership functions are probability functions [35, 55] or they can be computed as estimates of partial-volume fractions [14].

1.2.3 Continuous or Discrete Segmentation

Nearly all medical images used for image segmentation are presented as discrete samples on a uniform grid. Segmentation methods typically operate on the same discrete grid as the image. However, certain methods such as deformable models are capable of operating in the continuous spatial domain, thereby providing the potential for subpixel accuracy in delineating structures. Subpixel accuracy is desirable particularly when the resolution of the image is on the same order of magnitude as the structure of interest.

1.2.4 Interactions

The trade-off between manual interaction and performance is an important consideration in any segmentation application. Manual interaction can improve accuracy by incorporating the prior knowledge of an operator. For large-population studies, however, this can be laborious and time-consuming. The type of interaction required by segmentation methods can range from completely manual delineation of an anatomical structure to the selection of a seed for a region growing algorithm [38]. The differences in these types of interaction are the amounts of time and effort required, as well as the amounts of training required by the operators or experts. Methods that rely on manual interaction can also be vulnerable to reliability issues. However, even the automated segmentation methods typically require some interaction for specifying some initial parameters, whose values can significantly affect performance [16].

1.2.5 Validation

To quantify the performance of a segmentation method, validation experiments are necessary. Validation is typically performed with one of two different types of truth models. The most straightforward approach to validation is to compare the automated segmentation with manually obtained segmentation [56]. This approach, besides suffering from

the drawbacks outlined above, does not guarantee a perfect truth model, because an operatorś performance can also be flawed. The other common approach to validating segmentation methods is through the use of physical phantoms [33] or computational phantoms [15]. Physical phantoms provide an accurate depiction of anatomy. Computational phantoms can represent anatomy realistically, but usually simulate the image acquisition process by using simplified methods.

Once a truth model is available, a figure of merit must be defined for quantifying accuracy or precision [10]. The choice of the figure of merit is dependent on the application and can be based on region information, such as the number of pixels misclassified, or boundary information, such as distance to boundary. A survey on this topic has been provided [59].

1.2.6 Thresholding

Thresholding is one of the most common methods used for image segmentation. The success of this method is dependent on the intensity values of pixels in the image. The foreground image in this case is classified by comparing it through a threshold value with the background image that classifies it as a foreground image if there is a difference in the intensity values. Additional operations are needed to eliminate noise from the image and to acquire more effective results in the process of segmentation.

Thresholding approaches segment images by creating a binary partitioning of the image intensities. Figure 1.2 shows histogram of an image that includes three apparent classes. A thresholding procedure attempts to determine an intensity value, called the threshold, which separates the desired classes. The segmentation is then achieved by grouping all pixels with intensity greater than the threshold into one class, and all other pixels into another class. Two potential thresholds are shown in Figure 1.2 at the valleys of the histogram.

Thresholding is a simple yet often effective means for obtaining a segmentation in

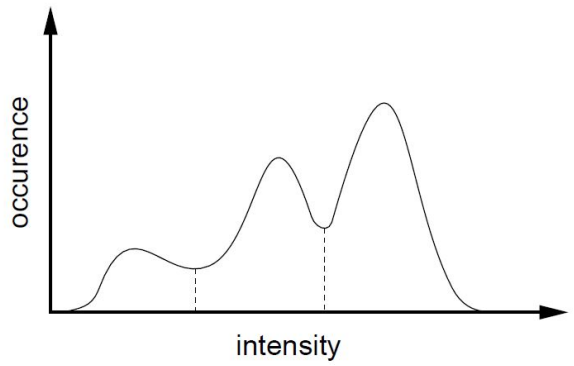


Figure 1.2: A histogram showing three apparent classes [44].

images where different structures have contrasting intensities or other quantifiable features. The partitioning is usually generated interactively, although automated methods do exist [50]. For images, interactive methods can be based on visual assessment of the resulting segmentation since the thresholding is implementable in real-time.

Thresholding is often used as an initial step in a sequence of image processing operations. Thresholding typically does not take into account the spatial characteristics of an image. This causes it to be sensitive to noise and intensity inhomogeneities. Variations of classical thresholding have been proposed for medical image segmentation that incorporate information based on local intensities and connectivity [34].

1.2.7 Multi-label image classification

Multi-label image classification aims to detect different objects in images. In single-label image classification there is only one category of objects of interest to be recognized so these problems can turn into binary classification where there can exist multiple number of the same object (foreground) being separated from the background. In comparison multi-label image classification is more complicated than single-label one. The labels of each image might be different and the number of labels per image is not fixed. To approach such problem one has to find a correlation between labels. With the development

of machine learning and deep learning technologies, many solutions [12, 13, 20, 54, 60] have been proposed to learn the label correlation and have achieved promising performance on different benchmarks [57].

In this thesis, we are focused on single-label image classification, specifically we are interested in cell segmentation and the techniques that are based on binary segmentation where we have two classes, namely foreground and background. The significance of cell segmentation lies in that cells are the base of each biological mechanism and organ. In order to make the diagnosis automated in the medical field one needs to be able to detect cells in a given image in order to make further inspection.

However note, it is possible to transfer the binary classification to multi-label classification but depending on the problem and number of instances and also the methods that are being used one must acknowledge the challenges. For example by expanding graph cut algorithms into detecting more than one object as foreground, the increasing number of connections and memory constraints must be considered.

Chapter 2

Background

Image segmentation is a fundamental problem in computer vision as well as medical imaging. The objective of image segmentation is to segment an image into several non-overlapping regions that are deemed meaningful according to some objective criterion. Image segmentation has been a long-studied problem. Since the first image segmentation approach being published over 40 years ago, see for instance [39], thousands of algorithms have been proposed [18, 37, 42, 11, 36], and they can be very different using different mathematical models or according to different application goals. Several common approaches have appeared in the recent literature on medical image segmentation. In this thesis, we review both conventional and modern methods, provide an overview of their implementation, and discuss their advantages and disadvantages. Although each technique is described separately, multiple techniques are often used in conjunction for solving different segmentation problems.

Most of the image segmentation methods that we will describe can be posed as optimization problems where the desired segmentation minimizes some energy or cost function defined by the particular application. In probabilistic methods, this is equivalent to maximizing a likelihood or a *posteriori probability*. Given the image $\mathbf{y}$, we desire the segmentation output image $\hat{\mathbf{x}}$ such that

$$\hat{\mathbf{x}} = \arg \min_{x} \varepsilon(x, y) \tag{2.1}$$

where ε, the energy function, depends on the observed image y and a segmentation x. Defining an appropriate ε is a major difficulty in designing segmentation algorithms because of the wide variety of image properties that can be used, such as intensity, edges, and texture. In addition to information derived from image, prior knowledge can also be incorporated to further improve performance. The advantage of posing a segmentation as an optimization problem is that it precisely defines what is desirable in the segmentation. It is clear that for different applications, different energy functions are necessary.

Several general surveys on image segmentation exist in the literature [26, 41]. Additional surveys on image segmentation specifically for medical images have also appeared [51].

In the process of segmentation of a medical image, the details required by the segmentation process are highly dependent on clinical application of the problem [61]. The purpose of segmentation is to improve the process of visualization to handle the detection process more effectively and efficiently. Through the process of segmentation one can analyze, diagnose, quantify, monitor and plan the navigation of a disease.

Segmentation of medical image could be challenging [52]. The problem of uncertainty arises when there is noise in the image which makes the classification of an image difficult [5]. The reason is that intensity values of pixels are altered to the noise in the image. This alternation in the intensity values of pixels disturbs uniformity in the intensity range of image. Noise can be present in the image because of motion and blurring effect. The problem of partial volume averaging causes the issue of inconsistency in the intensity values of image pixels. In order to handle this uncertainty in the medical image diagnosis systems image segmentation is playing a vital role [27].

2.1 Deep Learning

2.1.1 Neural Networks

Artificial neural networks are popular techniques that simulate the mechanism of learning in biological organisms [3]. These networks are computing systems inspired by a biological mechanism which contain many computation units referred to as neurons. An artificial neural network in its simplest form is a differentiable function $\mathcal{F} : X \to Y$ that transforms an input set X to the desired output set Y. The function $\mathcal{F}$, also called a model, is a composition of many simple functions known as neurons each doing a linear transformation on their input using their parameters known as weights, followed by a non-linearity. The search space of function $\mathcal{F}$ and the intermediate parameters of it neurons are determined by optimizing the model with respect to a differentiable loss function using some derivative-based optimization technique. The process of optimizing a model is called *training* where the model parameters are adjusted using a finite set of input-output pairs called *training set*. Once the model is trained, it can be used as *inference* where it maps any unseen input from set X to an output from set Y. This ability to compute functions of unseen inputs by training over a finite training set is referred to as *model generalization*.

The goal of this Section is to define background material needed to follow algorithms being used on Chapter 3.

Machine learning is a subfield of computer science that is concerned with building algorithms which, to be useful, rely on a collection of examples of some phenomenon. These examples can come from nature, be handcrafted by humans or generated by another algorithm. It can also be represented as a set of methods and technologies to allow computers to learn from experience [23]. One solution to machine learning is to have machines understand the world in terms of a hierarchy of concepts. With this approach, the machine can learn complicated tasks by building them from simpler ones in a deep

hierarchy of concepts. This approach to machine learning is called *Deep Learning* [23].

Several machine learning techniques require hard-coded knowledge about the world in a formal language. Difficulties faces using this approach suggested systems to acquire the ability to learn the hard-coded knowledge on their own, by extracting patterns from raw data. Many machine learning tasks can be solved by representing the right set of features to extract for that task, then providing these features to a simple machine learning algorithm. For many task, however, it is not always easy to figure out what features should be extracted. One solution to this problem is to use machine learning to discover not only the mapping from representation to output but also the representation itself. One of the main challenge of this approach is that sometimes it is not easy to extract high-level, abstract features from raw data. *Deep learning* solves this central problem by introducing representations that are expressed in terms of other, simpler representations. Deep learning enables the computer to build complex concepts out of simpler concepts.

Supervised learning is a class of learning problems that can be formulated as a machine performing a mapping $f : X \to Y$, from a vector space of all possible inputs X to the vector space of all possible outputs Y where the output is known in advance and supplied by supervision. Given a training set of n examples of input-output pairs $\{(x_1, y_1), ...(x_n, y_n)\} \in X \times Y$, where y_i can be generated by a known function $y_i = f(x_i)$ the job of learning algorithm is approximate the true function f with a *hypothesis* function $h : X \to Y$. One example of supervised learning is *classification* problems where the input needs to be mapped to a category of IDs using a learned function $h(X)$. For example in a binary classification task of face detection, X is set of input images and $Y = \{0, 1\}$ is a set of labels with 1 indicating a match and 0 otherwise. The output of the hypothesis function is a probability value in the interval $[0, 1]$ indicating the probability of a face matching the target. Another common supervised learning is *regression* task where the output $Y = \mathbb{R}^m$ is a set of real-valued targets. For example, in a learning algorithm that

estimates the age of a person from an image, the input is an image and the hypothesis function outputs a real-valued number estimating the age.

The learning procedure consists of finding a hypothesis function h from a *hypothesis space* $\mathcal{H}$ using a training set, where $\mathcal{H}$ is a space of functions $f : X \to Y$ the algorithm will search through. More precisely, let $\{(x_1, y_1), ...(x_n, y_n)\} \sim p_{data}$ (p_{data} is the probability distribution of data) be the training set of n independent and identically distributed (iid) examples taken from data distribution p_{data} and $f : X \to Y$ be the true mapping from an input set X to Y. We consider a scalar-valued loss function $L(\hat{y}_i, y_i)$ that measures the disagreement between the true label y_i and the predicted value $(\hat{y})_i = h(x_i)$ for some $h \in \mathcal{H}$ and we define $\mathbb{E}$ to be the expected value of a given random variable. Our objective is to estimate h using

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim p_{data}} [L(h(x), y)]. \tag{2.2}$$

In practice the expectation is taken over the training set meaning we seek to find a function h^* that minimizes the expected loss over the training set. Once the function h^* is learned we can use it to map samples from X to Y. We say the model can **generalize** if it performs accurately on novel unseen samples after being trained using the training data set.

One example of supervised learning algorithm is *logistic regression* which is used in binary classification problems. They hypothesis is defined as a *logistic function*, also known as the sigmoid function that measures the conditional probability of *true* label given an input X.

$$h_\theta(X) = \frac{1}{1 + \exp^{-\theta^T} X} = Pr(Y = 1|X; \theta), \tag{2.3}$$

where θ is a vector of model parameters and the sigmoid function outputs the probability of the model predicting 1. The $Pr(Y = 1|X; \theta)$ is the conditional probability of event $Y = 1$ given the event X over θ. The probability of the model predicting 0 is then given

by

$$Pr(Y = 0|X; \theta) = 1 - h_\theta(X), \tag{2.4}$$

we can write the probability of $Pr(Y|X; \theta), Y \in \{0, 1\}$ as a Bernoulli distribution

$$Pr(Y|X; \theta) = h_\theta(X)^Y (1 - h_\theta(X))^{(1-Y)}. \tag{2.5}$$

The *maximum likelihood* is a common approach used to estimate the model where we define the likelihood function over all (x_i, y_i) samples in the training set as

$$\mathcal{L}(X; \theta) = Pr(Y|X; \theta) = \prod_i h_\theta(x_i)^{y_i} (1 - h_\theta(x_i))^{(1-y_i)}. \tag{2.6}$$

It is a common practice to take the logarithm of the likelihood function. The loss is defined as minimizing the negative log likelihood of the above equation over the training set

$$\ell(X; \theta) = -\sigma_i y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i)). \tag{2.7}$$

The minimization is performed by finding the gradient of the log-likelihood function with respect to model parameters in a gradient-based algorithm.

Many supervised learning problems can be solved using the above formulation. A neural network classification, for example, can also use maximum likelihood to estimate model's parameters. The function space on which the hypothesis is defined is what makes models different; In general there is a tradeoff between complex hypotheses that fits the training data well and simpler hypotheses that may generalize better [49]; this is known as *bias-variance tradeoff* in supervised learning. Once the hypothesis and the scalar-valued loss functions are selected, the problem of supervised learning reduces to an optimization problem to estimate the model parameters.

2.1.2 Optimization

Most deep learning algorithms involve optimization of some sort. Optimization is referred to a task of minimization or maximization of some function $\ell(\theta) : A \to \mathbb{R}$ for some set A to the set of real numbers by altering θ. Normally, the optimization problems are phrased as minimizing $\ell(\theta)$ where we seek an element $\theta^* \in A$ that satisfies $\ell(\theta^*) \leq \ell(\theta)$ for all $\theta \in A$. In case of maximization we may alter the algorithm as minimizing $-\ell(\theta)$. The function $\ell(\theta)$ is called an *objective function*; when the optimization is performed by minimization, the function may also be referred to as the *cost function* or the *loss function*.

For example the maximum likelihood estimation for supervised learning discussed earlier in this Section $\theta^* = \arg\max_{\theta \in \Theta} \mathcal{L}(X; \theta)$ with $\mathcal{L}$ being the likelihood function, can be solved by defining an objective function given by the log likelihood

$$\ell(\theta) = \log \mathcal{L}(X; \theta) = \mathbb{E}_{x \sim p_{data}} \log p_{model}(x; \theta), \tag{2.8}$$

where p_{model} and p_{data} are the model and data distributions respectively and we maximize $\ell(\theta)$ subject to $\theta \in \Theta$, or as we saw earlier minimize $-\ell(\theta)$. Sometimes we can obtain this analytically by solving $\nabla_\theta \ell(\theta) = 0$ for θ where ∇ is the gradient operator. However, this requires the closed-form solution for the equation which may not exist. Other times we can solve this using *derivative-based* optimization methods.

Derivative-based optimization. These methods are based on the assumption that the objective function is smooth and differentiable. First order derivative-based optimization methods compute the gradient of the objective function $\nabla_\theta \ell(\theta)$ with respect to its parameters θ. The gradient is a vector of partial derivatives that gives the direction in which ℓ increases most rapidly along every dimension of θ. The gradient vector then can be used as a search direction. A very simple first order derivative-based optimization is *gradient ascent*. The idea is to take small steps in the objective function landscape in the direction of its gradient using an iterative process.

$$\theta^{t+1} = \theta^t + \alpha \nabla_\theta \ell(\theta) \tag{2.9}$$

where α is a small positive *scalar* controlling the step-size, in the context of machine learning also known as the *learning rate*. Normally optimization by minimization is preferred, where we take steps in the opposite direction of the gradient effectively performing *gradient descent*.During the optimization a training set $\{x_1, ...x_n\} \sim p_{data}$ is being used to approximate the model parameters and p_{data} is the training data distribution.

$$\theta^{t+1} = \theta^t - \alpha \nabla_\theta \mathbb{E}_{x \sim p_{data}} \left[\ell(x; \theta) \right] \tag{2.10}$$

where the expression is taken over the entire training set. This can be computationally very expensive with a large dataset where we need to evaluate the loss for every training example in order to perform one step of gradient descent. To resolve this problem, *stochastic gradient descent* (SGD) [31] algorithm is proposed that calculates the gradient over a small subset of the training set

$$\theta^{t+1} = \theta^t - \alpha \nabla_\theta \left[\frac{1}{m} \sum_{x_i \in \mathbb{S}} \ell(x_i; \theta) \right] \tag{2.11}$$

where $\mathbb{S}$ is a subset of training examples $\{x_1, ...x_m\} \sim p_{data}$ randomly selected for each iteration of gradient descent and is called *minibatch*. The typical size of a minibatch is between 1 and 128 [30]. The idea behind SGD is that we can perform many approximate updates instead of one exact gradient update. Each update only approximately takes a step toward the objective function's minimum, and this is why the algorithm is called "stochastic". However, this process can converge much faster than the regular gradient descent. This optimization algorithm is sometimes called *minibatch gradient descent* [6].

Optimization methods that only use gradients, such as gradient descent are called *first-order optimization algorithms*. In comparison, *second-order optimization methods* that leverage second derivatives information in an iterative updating optimization can

reach the critical point much faster then first-order algorithms. For example, Newton's method in optimization is an iterative method to find the roots of a derivative of a twice-differentiable function. It is based on a second-order Taylor series expansion to approximate a function $f(\mathrm{x})$ near a point $\mathrm{x}^{(0)}$ [23].

$f(\mathrm{x}) \approx f(\mathrm{x}^{(0)}) + (\mathrm{x} - \mathrm{x}^{(0)})^T \nabla_{\mathrm{x}} f(\mathrm{x}^{(0)}) + \frac{1}{2}(\mathrm{x} - \mathrm{x}^{(0)})^T \mathbf{H}(f)(\mathrm{x}^{(0)})(\mathrm{x} - \mathrm{x}^{(0)})$, where x is a multi-dimensional input array and $\mathbf{H}(f)$ is the Hessian matrix of second-order partial derivatives of f with respect to every input dimension. Solving the above equation for the critical point x^* of the function we obtain

$$\mathrm{x}^* = \mathrm{x}^{(0)} - \left[\mathbf{H}(f)(\mathrm{x}^{(0)})\right]^{-1} \nabla_{\mathrm{x}} f(\mathrm{x}^t). \tag{2.12}$$

We can solve the optimization problem recursively.

$$\mathrm{x}^{t+1} = \mathrm{x}^t - \gamma \left[\mathbf{H}(f)(\mathrm{x}^t)\right]^{-1} \nabla_{\mathrm{x}} f(\mathrm{x}^t), \tag{2.13}$$

where γ is a small step size similar to the learning rate in the gradient descent algorithm. This approach, despite having a useful property of reaching the critical point much faster, may also converge to saddle points or local maximum which is a harmful property for minimization problems. Another problem with this method is that it requires to find an inverse of a Hessian matrix which can be computationally expensive when the input dimension is large. Many second-order derivative methods are introduced in the literature, that fix converging to saddle points or problems with computing the Hessian matrix. However, second-order methods still remain difficult to scale to large networks [23].

2.1.3 Backpropagation

In the stochastic gradient descent algorithm discussed earlier, we need to compute the gradient of the loss with respect to model's parameters to minimize it. Computing the gradient using analytical expression is straightforward, however evaluating such expression for every parameter in model that contains thousands or even millions of parameters

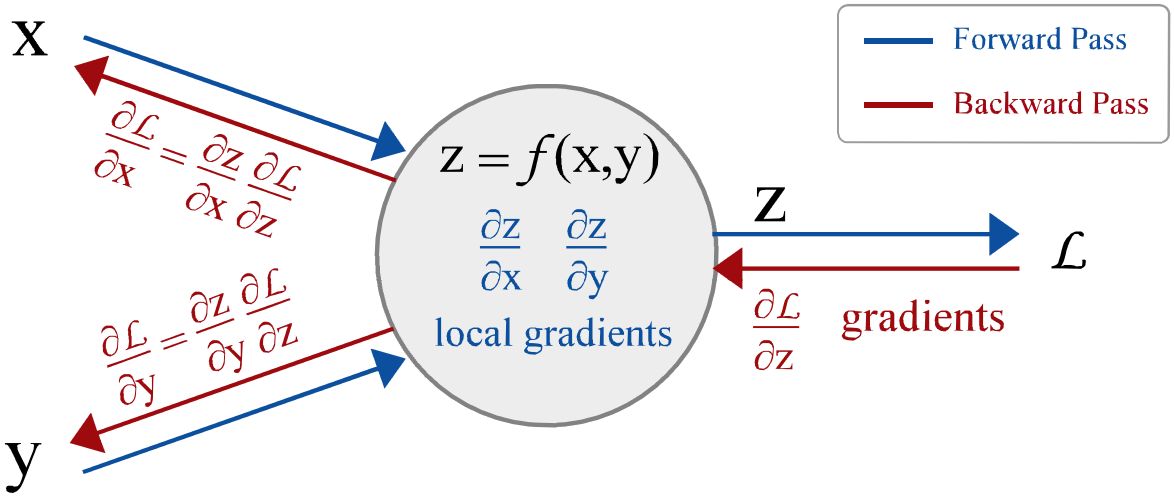


Figure 2.1: Schematic overview of backpropagation in a simple computational graph. During forward pass, vectors x and y are inputs to a node that performs some fixed computation on its inputs producing vector z. Note that we can compute the Jacobian matrices $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ for the node at this stage. The output z flows further to the graph where at the end we calculate a loss using a differentiable scalar-valued function $\mathcal{L}$. The backward pass proceeds in the reverse order, effectively calculating the gradient of the loss with respect to all the elements int the graph using the chain rule. The gradient of the loss with respect to the vector z is calculated $\frac{\partial \mathcal{L}}{\partial z}$ and gets multiplied with the local gradients calculated during the forward pass to find the global gradient of the loss with respect to the inputs $\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial z}{\partial x}\frac{\partial \mathcal{L}}{\partial z}$ and $\frac{\partial \mathcal{L}}{\partial y} = \frac{\partial z}{\partial y}\frac{\partial \mathcal{L}}{\partial z}$. In neural networks, each node contains parameters, where th gradient with respect to each parameter tells us how they should be changed to minimize the loss [40].

is computationally expensive. Using chain rule of calculus, one can see that different elements of a gradient with respect to model's parameters contain many common subexpressions. The *backpropagation* algorithm or simply *backprop* [48], is a recursive application of the chain rule that avoids re-computing these subexpressions to compute the gradient efficiently. The idea is based on formalizing the model as a function mapping from input to output in a directed acyclic graph (DAG) called the *computational graph*. In a computational graph, we use nodes to indicate differentiable transformations (operation) performed on some input variables (scalar, vector, matrix, or tensor). A node may contain its own variables and always produces one or more outputs which then flows to other nodes. The graph may be evaluated in a *forward pass* or *backward pass*.

In the forward pass, we take an input (batch of data in neural network application) and forward the graph by evaluating each operation in the graph recursively. Each node in the graph has a known differentiable operation, and during the forward pass, the Jacobian of the output of the node with respect to its inputs (and its local variables) are evaluated and stored locally. In the backward pass, the gradient of the loss with respect to the output of the graph is calculated and gets passed to the nodes in the graph in reverse order. The gradient of the loss with respect to each node's inputs (and local variables) is evaluated using the chain rule of calculus by multiplying the gradient coming from the next node with the local gradients stored locally during the forward pass. The result is passed to previous nodes to recursively evaluate the gradient with respect to every variable in the graph.

Figure 2.1 shows a schematic overview of backpropagation for a single node in a computational graph. In a neural network application, the inputs to each node are commonly tensors generated by transformations applied on the network input from the previous nodes. The local variable of the nodes are the network parameters we try to find. The gradient with respect to each parameter tells us how they should be changed to minimize the loss. In practice, deep learning software frameworks, use backpropagation

to evaluate the gradients where we design the graph and the intermediate operations and the backward pass is performed implicitly by the framework during optimization.

2.1.4 The Convolution Operation

In its most general form, convolution is an operation on two functions of a real-valued argument. To motivate the definition of convolution, we start with examples of two functions we might use. Suppose we are tracking the location of a spaceship with a laser sensor. Our laser sensor provides a single output $x(t)$, the position of the spaceship at time t. Both x and t are real valued, that is, we can get a different reading from the laser sensor at any instant in time.

Now suppose that our laser sensor is somewhat noisy. To obtain a less noisy estimate of the spaceship's position, we would like to average several measurements. Of course, more recent measurements are more relevant, so we will want this to be a weighted average that gives more weight to recent measurements. We can do this with a weighting function $w(a)$, where a is the age of measurement. If we apply such a weighted average operation at every moment, we obtain a new function s providing a smoothed estimate of the position of the spaceship

$$s(t) = \int x(a)w(t-a)\ da. \tag{2.14}$$

This operation is called *convolution*. The convolution operation is typically denoted with an asterisk

$$s(t) = (x * w)(t). \tag{2.15}$$

In convolutional network terminology, the first argument (in this example, the function x) to the convolution is often referred to as the *input*, and the second argument (in this example, the function w) as the *kernel*. The output is sometimes referred to as the *feature map*.

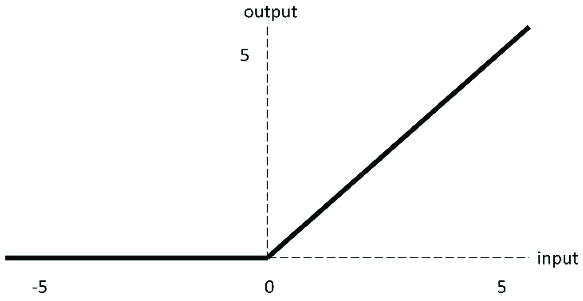


Figure 2.2: Graph of **ReLu** function

In machine learning applications, the input is usually a multidimensional array of data, and the kernel is usually a multidimensional array of parameters that are adapted by the learning algorithm. We will refer to these multidimensional arrays as tensors. Because each element of the input and kernel must be explicitly stored separately, we usually assume that these functions are zero everywhere but in the finite set of points for which we store the values. This means that in practice, we can implement the infinite summation as a summation over a finite number of array elements.

Finally, we often use convolutions over more than one axis at a time. For example, if we use a two-dimensional image I as our input, we also want to use a two-dimensional kernel K

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{2.16}$$

Convolution is commutative, meaning we can equivalently write

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{2.17}$$

Usually the latter formula is more straightforward to implement in a machine learning library, because there is less variation in the range of values of m and n.

2.1.5 Rectified Linear Unit (ReLu)

The rectified linear unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive values x it returns that value back. It can be written as $f(x) = \max(0, x)$. Graphically it looks like Figure 2.2.

2.1.6 Max Pooling

A typical layer of a convolutional network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activation. In the second stage, each linear activation is run through a nonlinear activation function, such as the ReLu. This stage is sometimes called the *detector stage*. In the third stage, we use a *pooling function* to modify the output of the layer further.

A pooling function replaces the output of the network at a certain location with a summary statistic of the nearby outputs. For example, the *max pooling* can be considered as an operation which reports the maximum output within a rectangular neighbourhood. Other popular pooling functions include the average of a rectangular neighbourhood, the L^2 norm of a rectangular neighbourhood, or a weighted average based on the distance from the central pixel.

In all cases, pooling helps to make representation approximately *invarient* to small translations of the input. Invariance to translation means that if we translate the input image by a small amount, the values of most of the pooled outputs do not change. *Invariance to local translation can be a useful property if we care more about whether some feature is present than exactly where it is.*. For example, when determining whether an image contains a face, we just need not to know the location of the eyes with pixel-perfect accuracy, we just need to know that there is an eye on the left side of the face and an eye on the right side of the face.

2.1.7 Long Short-Term Memory

In the Section we will be explaining about a powerful recurrent neural networks which later on will be associated with the implementation of one the techniques being used in this thesis.

Recurrent networks can in principle use their feedback connections to store representations of recent input events in form of activations ("short-term memory", as opposed to "long-term memory" embodied by slowly changing weights). This is potentially significant for many applications, including speech processing, and music composition. Hochreiter et. al in [29] presents *"Long Short-Term Memory"* (LSTM), a novel recurrent network architecture in conjunction with an appropriate gradient-based learning algorithm. LSTM is designed to overcome the back-flow error problems of recurrent neural nets. It can learn to bridge time intervals in excess of 100 steps even in case of noisy, incompressible input sequences, without loss of short time lag capabilities. This is achieved by an efficient, gradient-based algorithm for an architecture enforcing *constant* (thus neither exploding nor vanishing) error flow through internal states of special units (provided the gradient computation is truncated at certain architecture-specific points —this does not affect long-term error flow though) [29].

Humans don't start their thinking from scratch every second. As you read this thesis, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

In Figure 2.3, a chunk of neural network, A, looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next. These loops make recurrent neural networks seem kind of mysterious. However, if we think a bit more, it turns out they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Figure 2.4 illustrates what happens if

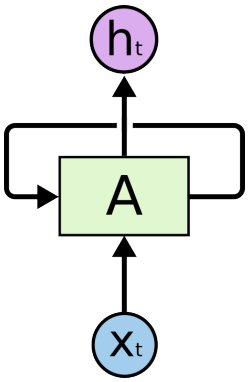


Figure 2.3: Recurrent Neural Networks loops [1]

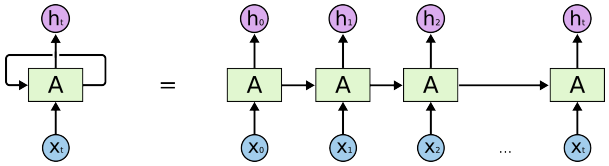


Figure 2.4: An unrolled recurrent neural network [1]

we unroll the loop.

The chain-like nature reveals that recurrent neural networks are related to sequences and lists. They are the natural architecture of neural network to use for such data.

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. The obstacle is that could RNNs do this task? Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the *sky*,"we don't need any further context, it's pretty obvious that the next word is going to be sky. In such cases, where the gap between relevant information and the place that it's needed is small, RNNs can learn to use the past information. See Figure 2.5.

But there also cases where we need more context. Consider trying ot predict the last word in the text "I grew up in France ... I speak fluent *French*". Recent information suggests that the next word is probably the name of a language, but if we want to narrow

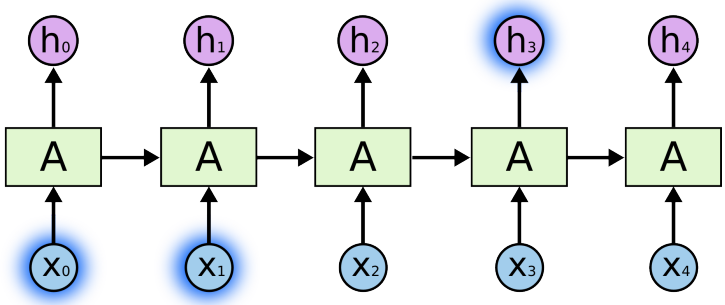


Figure 2.5: RNN with short term dependencies [1]

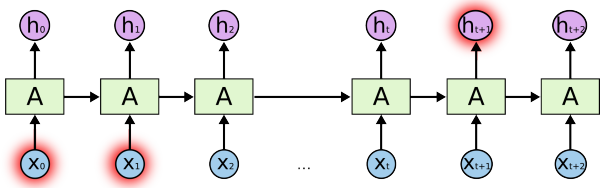


Figure 2.6: RNN with long term dependencies [1]

down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large. Unfortunately, as the gap grows, RNNs become unable to learn to connect the information.

In theory RNNs are absolutely capable of handling such "long-term dependencies". See Figure 2.6. A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them. That's where long short term memory networks (LSTMs), a special kind of RNN, capable of learning long-term dependencies come into picture. LSTMs are explicitly designed to avoid the long-term dependency, as to remembering information for longer periods of time.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer, that is an activation function ranging from $(-1, 1)$ which maps the negative inputs to negative and the zero inputs are mapped near zero. LSTMs also

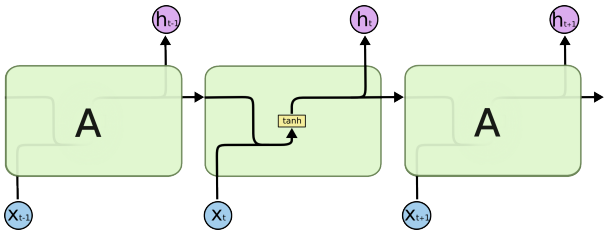


Figure 2.7: The repeating module in a standard RNN contains a single layer [1].

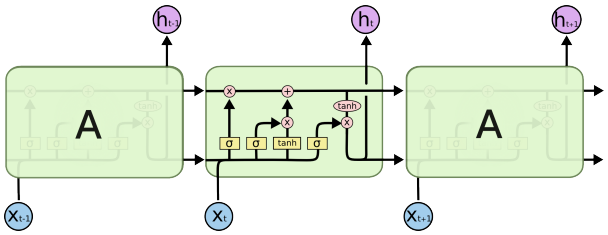


Figure 2.8: The repeating module in an LSTM contains four interacting layers. Each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation while a line forking denote its content being copied and the copies going to different locations [1].

have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. The core idea behind LSTMs is the cell state, the horizontal line running through the top of the Figure 2.7 and Figure 2.8. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called *gates*. Gates are a way to optionally let information through. They are composed out of a sigmoid (defined earlier in this Chapter) neural net layer and a point-wise multiplication operation.

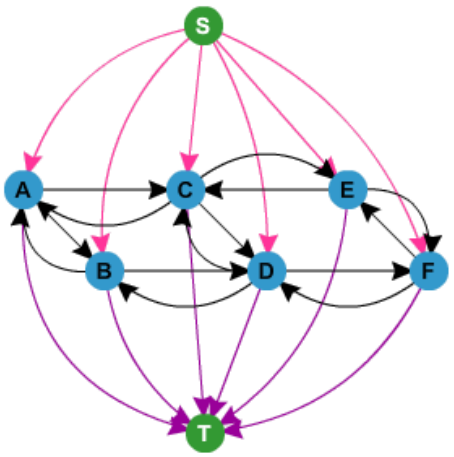


Figure 2.9: The graph corresponding to an arbitrary 2×3 image.

2.2 Graph Cut Segmentation

First, some terminology will be introduced. A graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined as a set of nodes or vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ connecting "neighbouring"nodes. For simplicity, we mainly concentrate on undirected graphs where each pair of connected nodes is described by a single edge $e = \{p, q\} \in \mathcal{E}$. The terminologies can be expanded to directed graphs as well.

In computer vision, the nodes of graphs consist of image pixels or voxels. se There are also two typically designated terminal nodes S (*source*) and T (*sink*) that represent "object"and "background"labels. Typically, neighbouring pixels are interconnected by edges in a regular grid-like fashion (usually *4–neighbourhood or 8–neighbourhood*). Edges between pixels are called *n–links* where n stands for "neighbour". Note that a neighbouring system can be arbitrary and may include diagonal or any other kind on n –links. Another types of edges, called *t–links*, are used to connect pixels to terminals. All graph edges $e \in \mathcal{E}$ including n–links and t–links are assigned some nonnegative weight (cost) w_e. See Figure 2.9.

An s–t cut is a subset of edges $\mathcal{C} \subset \mathcal{E}$ such that the terminals S and T become completely separated on the included graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \backslash \mathcal{C} \rangle$. Note that a cut divides the

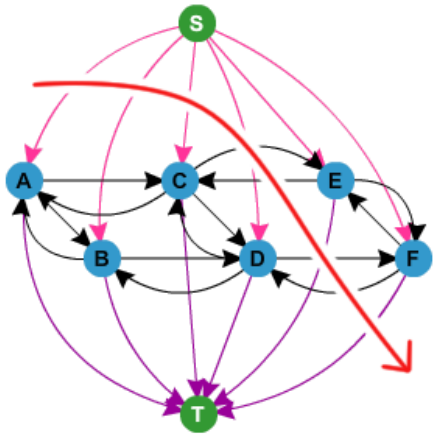


Figure 2.10: A random cut of graph

nodes between the terminals. Any cut corresponds to some binary partitioning of an underlying image into "object"and "background"segments. See Figure 2.10.

The goal is to compute the best cut that would give an "optimal"segmentation. In optimization the cost of a cut is defined as the sum of the costs of edges that is part of the cut

$$|\mathcal{C}| = \sum_{e \in \mathcal{C}} w_e. \tag{2.18}$$

Note that the included n–links are located at the segmentation boundary. Thus, their total cost represents the cost of segmentation with a desirable balance of boundary and regional properties. Numerically, the technique is based on a well-know optimization fact that a globally minimum s–t can be computed efficiently in low-order polynomial time [17, 22]. The corresponding algorithm works on any graphs. Therefore, the proposed graph cut segmentation method is not restricted to 2D images and computes globally optimal segmentation on volumes of any dimensions.

Note that a fast implementation of graph cut algorithms can be an issue in practice. The most straight forward implementations of the standard graph cut algorithms, e.g. max-flow [17] or push-relabell [22], can be slow. The experiments in Boykov and Kol-

mogorov [7] compare several well-known "tuned"versions of these standard algorithms in the context of graph based methods in vision.

Segmentation Energy. Consider an arbitrary set of data elements (pixels or voxels) $\mathcal{P}$ and some neighbourhood system represented by a set $\mathcal{N}$ of all (unordered) pairs $\{p, q\}$ of neighbouring elements in $\mathcal{P}$. For example, $\mathcal{P}$ can contain pixles (or voxels) in a 2D (or 3D) grid and $\mathcal{N}$ can contain an unordered pairs of neighbouring pixels (voxels) under a standard 4–(or 8–) neighbourhood system. Let $A = (A_1, ..., A_p, ..., A_{|\mathcal{P}|})$ be a binary vector whose components A_p specify *assignments* to pixels p in $\mathcal{P}$. Each A_p can be either "object"or "background". Vector A defines a segmentation. Then, the soft constraints imposed on boundary and region properties of A are described by the cost function

$$E(A) = \lambda \cdot R(A) + B(A) \tag{2.19}$$

where

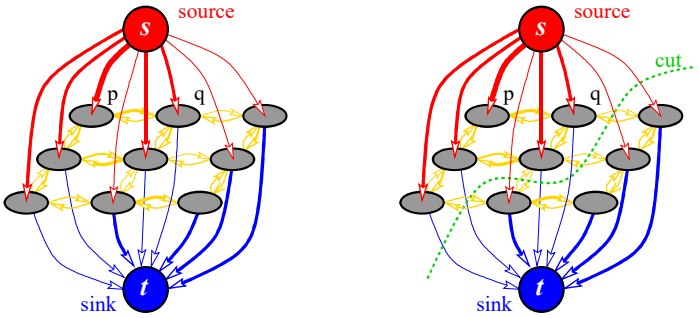
$$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p) \ (regional\ term) \tag{2.20}$$

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{p,q} \cdot \delta_{A_p \neq A_q} \ (boundary\ term) \tag{2.21}$$

and

$$\delta_{A_p \neq A_q} = \begin{cases} 1, & if \quad A_p \neq A_q \\ 0, & if \quad A_p = A_q. \end{cases} \tag{2.22}$$

The coefficient $\lambda \geq 0$ in equation (2) specifies a relative importance of the region properties term $R(A)$ versus the boundary properties term $B(A)$. The regional term $R(A)$ assumes that the individual penalties for assigning pixel p to "object"and "background", correspondingly the $R_p(\text{"}object\text{"})$ and $R_p(\text{"}background\text{"})$, are given. The regional term can be computed as



(a) A graph $\mathcal{G}$ (b) A cut on $\mathcal{G}$

Figure 2.11: Graph construction in Greig et. al. [24]. Edge costs are reflected by thickness.

$$R(A) = \sum -\log h(A_p) \tag{2.23}$$

where $h(\cdot)$ is the likelihood of the observed pixel.

We will continue this section with an example which was used early in computer vision, *Binary Image Restoration* [9] which used directed graphs.

The earliest use of graph cuts for energy minimization in the vision is due to Greig. et. al. [24]. They consider the problem of binary image restoration. Given a binary image corrupted by noise, the task is to restore the original image. This problem can be formulated as a simple optimization over binary variables corresponding to image pixels. In particular, Greig.et.al. builds a graph shown in Figure 2.11(a) where non-terminal nodes $p \in \mathcal{P}$ represent pixels while terminals s and t represent two possible intensity values. To be specific, source s will represent intensity 0 and sink t will represent intensity 1. Assume that $I(p)$ is the observed intensity at pixel p. Let $D_p(l)$ be a fixed penalty for assigning to pixel p some 'restored intensity' label $l \in \{0, 1\}$ ($D_p(l)$ is computed as the regional term explained earlier). Naturally, if $I(p) = 0$ then $D_p(0)$ should be smaller then $D_p(1)$ and weight of (p, t) is $D_p(0)$. Even though t-link weights should be non-negative, restriction $D_p \geq 0$ for data penalties is not essential.

Now there need to be regularizing constraints added which help to remove image noise. Such constraints enforce spacial coherence between neighbouring pixels by minimizing discontinuities between them. In particular, we create n-links between neighbouring pixels using any (e.g. 4- or 8-) neighbourhood system. The weight of this n-links is set to a smoothing parameter $\lambda > 0$ that encourages minimum cut to serve as few n-links as possible.

Remember that a cut $\mathcal{C}$ is a binary partitioning of the nodes into subsets $\mathcal{S}$ and $\mathcal{T}$. A cut can be interpreted as binary labeling f that assigns labels $f_p \in \{0, 1\}$ to image pixels: if $p \in \mathcal{S}$ then $f_p = 0$ and if $p \in \mathcal{T}$ then $f_{\mathrm{p}} = 1$. Obviously, there is a one-to-one correspondence between cuts and binary labellings of pixels. Each labeling f gives a possible image restoration result.

Consider the cost of an arbitrary cut $\mathcal{C} = \{\mathcal{S}, \mathcal{T}\}$. This cost includes weights of two types of edges: visited t-links and visited n-links. Note that a cut severs exactly one t-link per pixel; it must sever t-link (p, t) if pixel p is in the source component $p \in \mathcal{S}$ or t-link (s, p) if pixel p is in the sink component $p \in \mathcal{T}$. Therefore, each pixel p contributes either $D_p(0)$ or $D_p(1)$ towards the t-link part of the cut cost, depending on the label f_p assigned to this pixel by the cut. The cut cost also includes weights of visited n-links $(p, q) \in \mathcal{N}$, where $\mathcal{N}$ is the set of ordered pairs of connected nodes. Therefore,

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\substack{(p,q) \in \mathcal{N} \\ p \in \mathcal{S}, q \in \mathcal{T}}} w(p, q). \tag{2.24}$$

The cost of each $\mathcal{E}$ defines the 'energy' of the corresponding labeling f

$$E(f) := |\mathcal{C}| = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \cdot \sum_{(p,q) \in \mathcal{N}} \mathcal{I}(f_p = 0, f_q = 1) \tag{2.25}$$

where $\mathcal{I}(\cdot)$ is the identity function giving 1 if its argument is true and 0 otherwise. Stated simply, the first term says that pixel labels f_p should agree with the observed data while the second term penalized discontinuities between neighbouring pixels. Obviously,

minimum cut gives labeling f minimizing energy.

Note that parameter λ weights the relative importance of the data constraints and the regularizing constraints. Note that if λ is very small, optimal labeling assigns each pixel p a label f_p that minimizes its own data cost $D_p(f_p)$. In this case, each pixel chooses its own label independently from the other pixels. If λ is big, then all pixels must choose one label that has a smaller average data cost. For intermediate values of λ, optimal labeling f should correspond to a balanced solution with compact spatially coherent clusters of pixels who generally like the same label. Noise pixels, or outliers, should confirm to their neighbours.

In general, graph construction as in Figure 2.11 can be used for other binary 'labelling' problems. Suppose we are given a penalty $D_p(l)$ that pixel p incurs when assigned label $l \in \mathcal{L} = \{0, 1\}$ and we need to find a spatially coherent binary labeling of the whole image.

Chapter 3

Methodology

Over the past few decades, the medical imaging techniques, such as computed tomography (CT) has been used for early detection, diagnosis and treatment of diseases. Up until recent years, the clinical detection has been performed mostly by human experts such as radiologist and physicians. This method has its own downsides with wide variations in pathology the risk of potential fatigue of human experts increases. Now days researchers have begun to benefit from computer-assisted interventions. Although the rate of progress in medical image analysis has not been rapid as that in medical imaging technologies, the situation is improving with the introduction of machine learning techniques, especially with the rise of deep learning algorithms.

Researchers have approached medical data using various techniques and algorithms. The first methods were based on *foreground/background* separation utilizing mathematical representation of image data such as *graphs*. This has led to a wide field of study namely *Graph Cut Segmentation* [8, 9].

In applying machine learning, finding or learning informative features that well describe the regularities or patterns inherent in data plays a pivotal role in various tasks in medical image analysis. Conventionally, meaningful or task-related features were designed mostly by human experts on the basis of their knowledge about the target domains,

making it challenging for non-experts to exploit machine learning techniques for their own studies. In the meantime, there have been efforts to learn sparse representation based on predefined dictionaries, possibly learned from training samples. Sparse representation is motivated by the principle of parsimony in many areas of science; that is, the simplest explanation of a given observation should be preferred over the complicated ones. Sparsity-including penalization and dictionary learning have demonstrated the validity of this approach for feature representation and feature selection in medical image analysis (DL in Medical). It should be noted that sparse representation or dictionary learning methods described in the literature still find informative pattern or regularities inherent in data with a shallow architecture, thus limiting their representation power. However, deep learning has overcome this obstacle by incorporating the feature engineering step into a learning step. That being said, instead of extracting features using human experts, deep learning only needs a set of example input data with minimum prepossessing, if necessary, and then discovers the informative representations in a self-taught manner. Therefore, the burden of data engineering has shifted from humans to computers, allowing non-experts in machine learning to effectively use deep learning for their own research and/or applications, especially in medical image analysis.

3.1 Data

The purpose of this thesis is to compare the two most successful neural networks architecture introduced for medical image segmentation with graph based techniques. Our focus will be on studying how each algorithm is applied and explain them each one in detail. We will also discuss the performance of each network on datasets from **International Symposium on Biomedical Imaging** (ISBI) challenge [2]. The datasets consist of 2D time-lapse video sequences of fluorescent counterstained nuclei or cells moving on top or immersed in a substrate, along with 2D bright field, phase contrast, and Differential

Contrast (DIC) microscopy videos of cells moving on a flat substrate. In this thesis, we use the Flou-N2DH-SIM+ and PhC-C2DH-U373 datasets. The Flou-N2DH-SIM+ is gathered using Zeiss Axiovert 100s with microscope with Micromax 1300-YHS camera, pixels size of 0.125×0.125 microns and 29 minute time steps. The PhC-C2DH-U373 is taken using Nikon microscope with 0.65×0.65 microns pixel size and 15 minutes time steps [2].

3.2 Graph Cut Cell Segmentation

Graph cut works by representing image pixels as nodes and constructing a connection (directed edges) between nodes, given each connection a weight. The algorithm works by introducing two extra nodes, namely *Source s* and *Sink t*. All nodes then will be connected to the source and the sink given them an arbitrary high value weight. The goal of graph cut is to be able to separate the nodes belonging to source (foreground) from the ones in the sink (background) by creating a cut $\mathcal{C}$ with minimum cost. By this definition graph cut can be considered as a binary optimization approach. In fact, underlying min-cut/max-flow algorithms are inherently binary techniques. Graph cut in this thesis is based of the general case above using an initial human-interacted seed (Figure 3.1) for background and foreground. The seed will be then used as an input for a naive Bayesian classifier to be trained on.

Bayesian classifier is a conditional probability model, given a problem instance to be classified, represented by a vector $\mathrm{x} = (x_1, ..., x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $Pr(C_k|x_1, ..., x_n)$ for each of K possible outcomes or classes C_k. In other words, a Bayes classifier is probabilistic classifier based on applying Bayes' theorem for conditional probabilities. The assumption is that all features are independent and unrelated to each other (this is the 'naive' part). Bayes classifiers can be trained very efficiently. The classifier is constructed by

multiplying the individual conditional probabilities from each input feature vector to get the total probability of a class. Then the class with highest probability is selected [53]. The classifier is built using the Gaussian probability distribution. This means that each input data has its individual mean and covariance that are computed from a set of training data. At the end the probability for each data point is computed and the ones with highest probability are selected. The initial seeding which includes samples of foreground and background pixels, goes through Bayesian classifier so that after training it will assign each pixel a probability to be in class of foreground or background.

We require human input for our initial seeding. The reason being that graph cut is sensitive to how the background and foreground are being selected. If the areas are not correctly marked then the result will not be accurate. We believe given the visual selection by human results in less error compared to automated selection.

Given that we have trained a Bayes classifier [53] on foreground and background pixels, we can compute the probabilities $PF(I_p)$ and $PB(I_p)$, which are the probability of pixel belonging to the class of foreground pixels or background pixels. Here I_i is the intensity vector of pixels i. We have used the *4-neighbour* structure where each pixel is connected to the pixels directly above, below, left, and right.

In addition to the pixel nodes, we also have the two special nodes source and the sink. In our model all the pixels are connected to the source and the sink. To build the graph

- Every pixel node has an incoming edge from the source node,
- Every pixel node has an outgoing edge to the sink node,
- Every pixel node has one incoming and one outgoing edge to each of its neighbours.

We can now create a model for the edge weights as follows

(a) Original Image (b) Initial human-interacted seed

Figure 3.1: Initializing the graph cut using the given human-selected foreground/background labels. The small rectangle corresponds to a foreground selection and the bigger rectangle corresponds to a background selection.

$$
\begin{aligned}
w_{sp} &= \frac{PF(I_p)}{PF(I_p) + PB(I_p)} \\
w_{pt} &= \frac{PB(I_p)}{PF(I_p) + PB(I_p)} \\
w_{pq} &= \kappa e^{\frac{-|I_p - I_q|^2}{\sigma}}
\end{aligned}
\tag{3.1}
$$

where w_{si} is the edge weight from the source to the pixel i, w_{it} is the edge weight from pixel i to the sink, and w_{ij} is the edge weight between pixel i and pixel j. Also we have set $\kappa = 2$ and $\sigma = e^{-9}$ [53].

With this model, each pixel is connected to the foreground and background (source and sink) with weights equal to a normalized probability of belonging to that class. The w_{ij} describes the pixel similarity between neighbours, similar pixels have weight close to κ, dissimilar close to 0. The parameter σ determines how fast the values decay towards zero with increasing dissimilarity. In the end we apply the max-flow/min-cut algorithm using the PyMaxflow package in Python and construct a result based on maximum flow that can go through nodes and edges of our graph. The foundation of PyMaxflow package is based on an improved version of Ford-Fulkerson algorithm [19] which is based on augmented path methods to find the maximum flow on a given graph. A typical augmentation path algorithm works by carrying distribution of flow f among the edges from source s to sink t in a graph $\mathcal{G}$ using a *residual graph* $\mathcal{G}_f$. The residual graph $\mathcal{G}_f$ has

the same structure as the graph $\mathcal{G}$ but the capacity of an edge in $\mathcal{G}_f$ shows the capacity of the same edge in $\mathcal{G}$ given the amount of flow already in the edge. At the beginning, the amount of flow from source to the sink is zero ($f = 0$) and the edges of $\mathcal{G}_0$ have the same capacity as edges in the original $\mathcal{G}$. At each new iteration, the algorithm finds the shortest path from $s \to t$ the doesn't exceed the maximum capacity of each edge. If such a path exists, it gets augmented by modifying the capacities of the edges to the maximum possible flow. Based on the result of max-flow each pixel will be assigned to one of the classes of background or foreground (or $\{0, 1\}$ accordingly) and then they will be mapped to pixels in the original image.

3.3 U-Net

There is a belief about a successful training of a deep neural network that it requires many thousands of training samples. The challenge in medical image analysis is the low number of training samples which makes it difficult for researchers to apply the state-of-the-art neural networks such as *ImageNet* to medical image databases. Here we are going to study a successful convolutional networks for biomedical image segmentation, called **U-Net** [47]. The contribution of this method is that it is and end to end network which consists of a encoder path and an decoder path. Since it's introduction this architecture has become a benchmark for medical image segmentation [47].

U-Net was developed by Olaf Ronnebrger et. al. for Bio Medical image segmentation [47]. The architecture contains two paths. First path is the contraction path (also called as the *Encoder*) which is used to capture the context in the image. The encoder is just a traditional stack of *Convolutional* and *Max Pooling* layers. The second path is the symmetric expanding path (also called as the *Decoder*) which is used to enable precise localization using *Transposed convolutions*. Thus it is an end-to-end fully convolutional network (FCN). It only contains Convolutional layers and does not contain any *Dense*

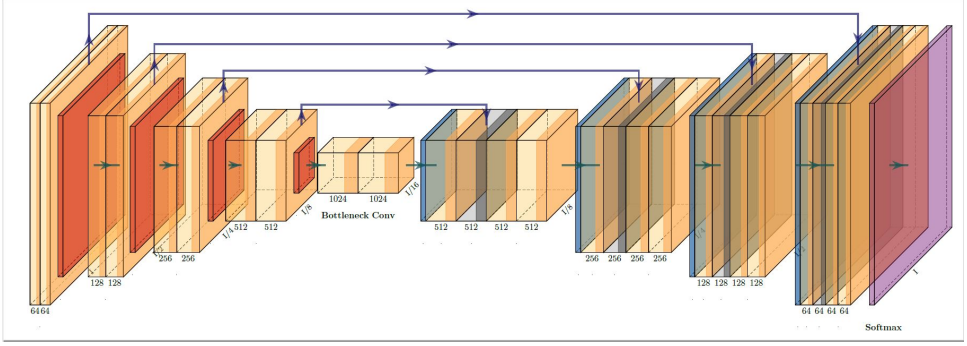


Figure 3.2: U-Net architecture. The number of channels is denoted under the box. The arrows on top of the box represent copied feature maps [47].

layers because of which it can accept image of any size.

Most of the operations are convolutions followed by a non-linear activation function (ReLU). It consists of a 3×3 convolution which only the valid part of it is used, meaning that for a 3×3 convolution theres is a 1-pixel border loss. This allows later to process large images in individual tiles.

The next operation in the network is the max pooling operation that reduces the x-y size of the feature map. The max pooling acts on each channel separately and propagates the maximum activation from each 2×2 window to the next feature map. After each max pooling operations the number of channels increases by a factor of 2. The sequence of convolution and max pooling operations result in spatial contraction where we gradually increase the *What* and at the same time decrease the *Where*. At the end of contraction path all the features maps create a single feature vector. The architecture includes an expansion path to create a high-resolution segmentation map. This path consists of up-convolution and concatenation with the corresponding high-resolution features from the contracting path. The up convolution uses a kernel to map each feature vector to the 2x2 output pixel window again followed by a non-linear activation function. The output segmentation map has two channels. One for *foreground class* and one for the *background class*. In total network has 23 convolutional layers.

To allow a seamless tiling of the output segmentation map, it is important to select the input tile size such that all 2×2 max pooling operation are applied to a layer with an even x and y size.

The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent. The energy function is computed by a pixel-wise softmax over the final feature map combined with the cross entropy loss function. The softmax is defined as $p_k(\mathrm{x}) = \exp(a_k(\mathrm{x}))/(\sum_{k'=1}^{k} \exp(a_{k'}(\mathrm{x})))$ where $a_k(\mathrm{x})$ denotes the activation in feature channel k at the pixel position $\mathrm{x} \in \Omega$ with $\Omega \subset \mathbb{Z}^2$ subset of all positive integers. k is the number of classes and $p_k(\mathrm{x})$ is the approximated maximum-function, i.e. $p_k(\mathrm{x}) \approx 1$ for the k that has the maximum activation $a_k(\mathrm{x})$ and $p_k(\mathrm{x}) \approx 0$ for all other k. The cross entropy then penalizes at each position the deviation of $p_{\ell(\mathrm{x})}(\mathrm{x})$ from using

$$E = \sum_{\mathrm{x} \in \Omega} w(\mathrm{x}) \log(p_{\ell(\mathrm{x})}(\mathrm{x})) \tag{3.2}$$

where $\ell : \Omega \to 1, ..., K$ is the true label of each pixel and $w : \Omega \to \mathbb{R}$ is a weight map that we introduce to give some pixels more importance in the training.

We pre-compute the weight map for each ground truth segmentation to compensate the different frequency of pixels from a certain class in the training data set, and to force network to learn the small separation borders that we introduce between touching cells.

The separation is computed using morphological operations. The weight map is then computed as

$$w(\mathrm{x}) = w_c(\mathrm{x}) + w_0 \cdot \exp\Big(- \frac{\big(d_1(\mathrm{x}) + d_2(\mathrm{x})\big)^2}{2\sigma^2} \Big) \tag{3.3}$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the class frequencies, $d_1 : \Omega \to \mathbb{R}$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \to \mathbb{R}$ the distance to the border of the second nearest cell.

3.4 LSTM-UNet

Live cell microscopy sequences exhibit complex spatial structures, and complicated temporal behaviour, making their analysis a challenging task. Considering cell segmentation problem, which plays a significant role in the analysis, the spatial properties of data can be captured using Convolutional Neural Networks (CNNs). Recent approaches show promising segmentation results using convolutional encoder-decoders such as U-Net. Nevertheless, these methods are limited by their inability to incorporate temporal information, that can facilitate segmentation of individual touching cells or of cells that are partially visible. In order to exploit cell dynamics [4] has proposed a novel segmentation architecture which integrates convolutional Long Short Term Memory with the U-Net [47]. The network's unique architechture allows it to capture multi-scale, compact, spatio-temporal encoding in the LSTM memory units.

The proposed network incorporates LSTM blocks into the U-Net architecture. This combination, as suggested here, is shown to be powerful. The U-Net architecture, built as an encoder-decoder with skip connections, enables to extract meaningful descriptors at multiple image scales. However, this alone does not account for the cell specific dynamics that can significantly support the segmentation. The introduction of LSTM blocks into the network allows considering past cell appearances at multiple scales by holding their compact representation in the LSTM memory units. The authors [4] have proposed the incorporation of LSTM layers in every scale of the encoder section of the U-Net. Applying the LSTM on multiple scales is essential for cell microscopy sequences (since the frame to frame differences might be at different scales, depending on cells' dynamics. Moreover, the microscopy sequence can be of arbitrary length, making the use of bi-directional LSTMs computationally impractical). The network is fully convolutional and, therefore, can be used with any image size during both training and testing. Figure 3.3 illustrates the network architecture.

In the paper [4], authors address individual cells' segmentation from microscopy se-

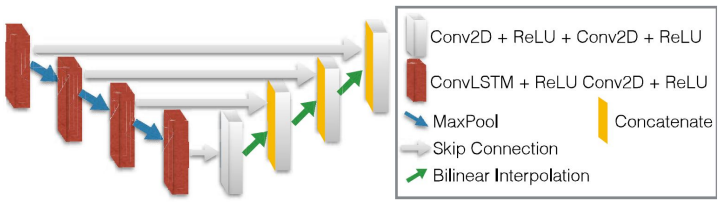


Figure 3.3: The LSTM-Unet network architecture. The down sampling path (left) consists of a LSTM layer followed by a convolutional layer with ReLU activation, the output is then down-sampled using max pooling and passed to the next layer. The up-sampling path (right) consists of a concatenation of the input from the lower layer with the parallel layer from the down-sampling path followed by two convolutional layers with ReLU activations [4].

quences. The main challenge in this type of problem is not only foreground-background classification but also the separation rotation of adjacent cells. They adopt the weighted distance loss. The loss is designed to enhance individual cells' delineation by a partitioning of the d dimensional (2 or 3) image domain $\Omega \in \mathbb{R}^d$ into two classes: foreground and background, such that pixels which are near the boundaries of two adjacent cells are given higher importance. We set $\mathcal{C} = \{0, 1\}$ to denote these classes, respectively. Let $\{I_t\}_{t=1}^{T}$ be input image sequence of length T, where $I_t : \Omega \to \mathbb{R}$ is a grayscale image. The network is composed of two sections of L blocks each, the encoder recurrent block $E_{\theta_l}^{\{l\}}(.)$ and the decoder block $D_{\theta_l}^{\{l\}}(.)$ where θ_l are the networks parameters. The input to the LSTM encoder layer $l \in [0, ..., L-1]$ at time $t \in T$ includes the down-sampled output of the previous layer, the output of the current layer at the previous time-step and the LSTM memory cell. These three inputs are denoted as $x_t^{\{l\}}, h_{t-1}^{\{l\}}, c_{t-1}^{\{l\}}$ respectively. Their formal definition is

$$(h_t^{\{l\}}, c_t^{\{l\}}) = E_{\theta_l}^{\{l\}}(x_t^{\{l\}}, h_{t-1}^{\{l\}}, c_{t-1}^{\{l\}}) \tag{3.4}$$

where,

$$x_t^{\{l\}} = \begin{cases} I_t, & l = 0 \\ MaxPool(h_t^{\{l-1\}}), & 0 < l < L. \end{cases} \tag{3.5}$$

The inputs to the decoder layers $l \in [L, ..., 2L-1]$ are the up-sampled output of the previous layer and the output of the corresponding layer from the encoder denoted by $y_t^{\{l\}}$ and $h_t^{\{2L-1-l\}}$ respectively. The decoder outputs is denoted as $z_t^{\{l\}}$. Formally,

$$y_t^{\{l\}} = \begin{cases} h_t^{\{l-1\}}, & l = L \\ UpSmaple(z_t^{\{l-1\}}), & L < l < 2L-1 \end{cases} \tag{3.6}$$

$$z_t^{\{l\}} = D_{\theta_l}(y_t^{\{l\}}, h_t^{\{2L-1-l\}}). \tag{3.7}$$

They defined a network f_Θ with parameters Θ as the composition of L encoder blocks followed by L decoder blocks, and denote $\Theta := \{\theta_l\}_{l=0}^{2L-1}$. Note that the encoder blocks, $E_{\theta_l}^{\{l\}}$, encode high-level *spatio-temporal* features at multiple scales and decoder blocks, $D_t^{\{l\}}$, refines that information into a full scale segmentation map.

$$o_t = f_\Theta = z_t^{\{2L-1\}}. \tag{3.8}$$

The final output is set as a $|\mathcal{C}|$-dimensional feature vector corresponding to each input pixel $v \in \Omega$. The segmentation is then defined as the pixel label probabilities using softmax equation:

$$Pr(c|o_t(\mathrm{v})) = \frac{\exp\{[o_t(\mathrm{v})]_c\}}{\sum_{c' \in \mathcal{C}} \exp\{[o_t(v)]_{c'}\}}. \tag{3.9}$$

The final segmentation is defined as

$$\Gamma_t = \arg_{c \in C}\max\ Pr(c|o_t(\mathrm{v})). \tag{3.10}$$

Each connected component of the foreground class is given a unique label and is considered an individual cell.

During the training phase the network is presented with a full sequence and manual annotations $I_t, \Gamma_{t\,t=1}^{T}$, where $\Gamma_t : \Omega \rightarrow [0, 1]$ are the ground truth (GT) labels. The loss is defined using the distance weighted cross-entropy loss as proposed in the original U-Net paper. The loss imposes the separation of cells by introducing an exponential penalty factor which is proportional to the distance of a pixel from its nearest and second nearest cells' pixels. Consequently, pixels that are located between two adjacent cells are significant importance whereas pixels further away from the cells have a minor effect on the loss.

3.5 Software

The implementation of the methods presented in this thesis are written in Python. Python was designed with simple readability in mind in order to reduce the complexity required to maintain and update existing code. The following python packages have been used in this project:

- **NumPy** is the library created for the purpose of scientific computing in Python. Its designed is highly similar to MATLAB, containing a large collection of mathematical functions and operations that can be applied to data stored as multi-dimensional arrays.
  https://numpy.org/

- **PyMaxflow** is a Python library for graph construction and maxflow computation (commonly known as graph cut). The core of this library is the C++ implementation by Vladimir Kolmogorov. Besides the wrapper to the C++ library, PyMaxflow offers

  – NumPy integration,

  – methods for the construction of common graph layouts in computer vision and

graphics,

- implementation of algorithms for fast energy minimization which use the maxflow method: the alpha-beta-swap and alpha-expansion.

https://pypi.org/project/PyMaxflow/

- **PyTorch** is ab open source deep learning library that is syntactically similar to NumPy. Like many existing machine learning libraries, PyTorch supports GPU acceleration through Nvidia's CUDA, Nvidia's parallel programming model, where data is stored in multidimensional arrays called tensor. The LSTM-Unet implementation is based on PyTorch.
  https://pytorch.org/

- **TensorRT** is built on CUDA and enables you to optimize inference for all deep learning frameworks, leveraging libraries, development tools and technologies in CUDA-X for artificial intelligence, autonomous machines, high-performance computing, and graphics. It has been used for LSTM-Unet development.
  https://developer.nvidia.com/tensorrt

- **Keras** is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Unet is imeplemented using this library.
  https://keras.io/

- **LSTM-Unet**. The source can be found at: https://github.com/arbellea/LSTM-UNet/

Chapter 4

Results

The goal of this thesis was to present most recent and to our knowledge the most accurate medical image segmentation algorithm using neural networks, introduced as *LSTM-Unet* and explore a comparison between conventional methods and most recent ones. In this thesis we explored a promising conventional method that has been used widely in the fields of computer vision and medical imaging, namely *Graph Cut*. Both techniques come with their unique capabilities and implementation as well as their drawbacks. In this Chapter we will be discussing a comparison between these two approaches and support our discoveries by illustrating the results of each one.

The LSTM-Unet having the characteristics of LSTM neural networks is equipped with the ability to preserve information from previous input entry and make a decision based on the stored data which helps to reach a more accurate segmentation on series of continuous images that is the case for most medical datasets. Figure 4.3 illustrates this ability.

There could be times when accessing a device with GPU can be a problem and also the challenge with medical data is that it may not have always have the labels or ground truth needed to train a neural network with. Graph cut 4.4 algorithm can be at advantage in a sense that is does not need to be fed with ground truth, as long as there is

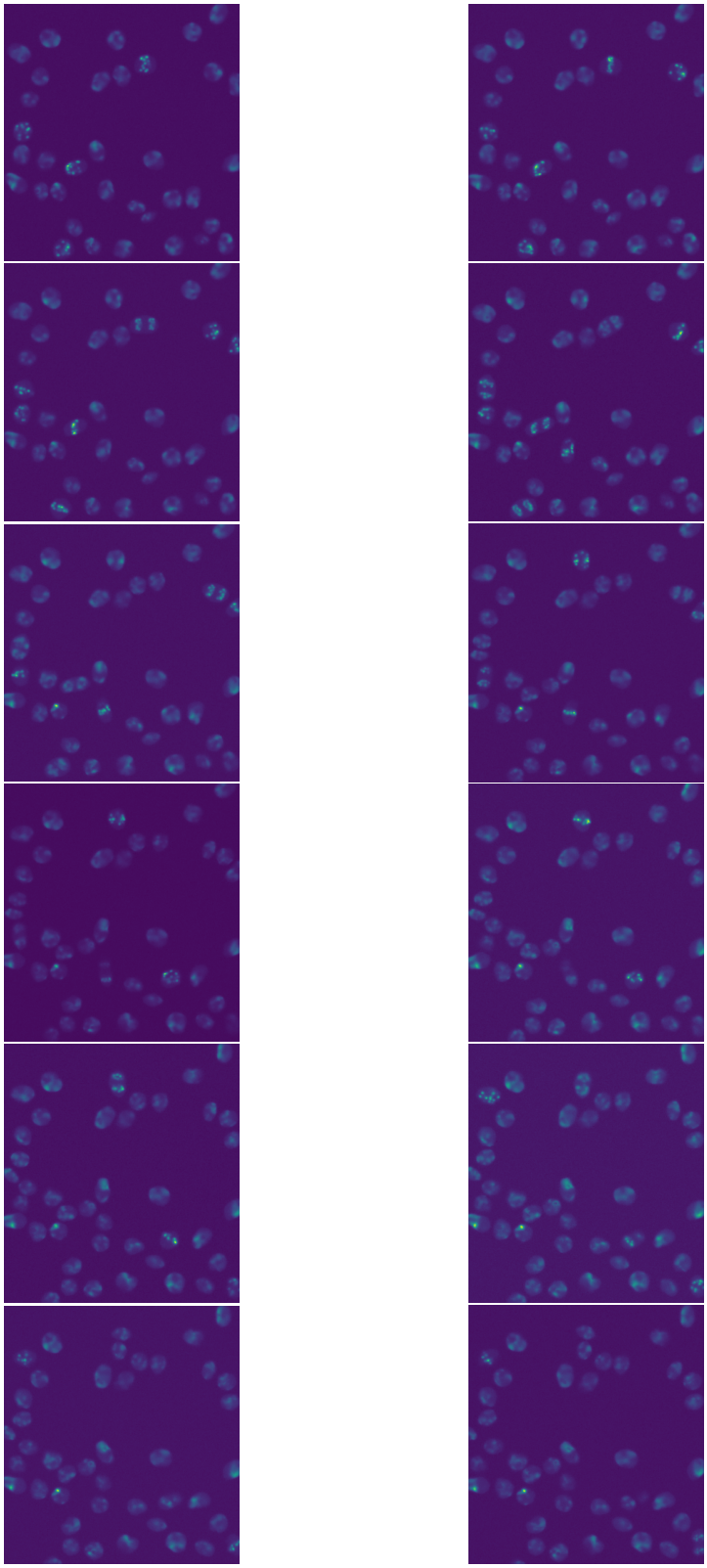


Figure 4.1: Series of continuous of original images from Flou-N2DH-SIM+

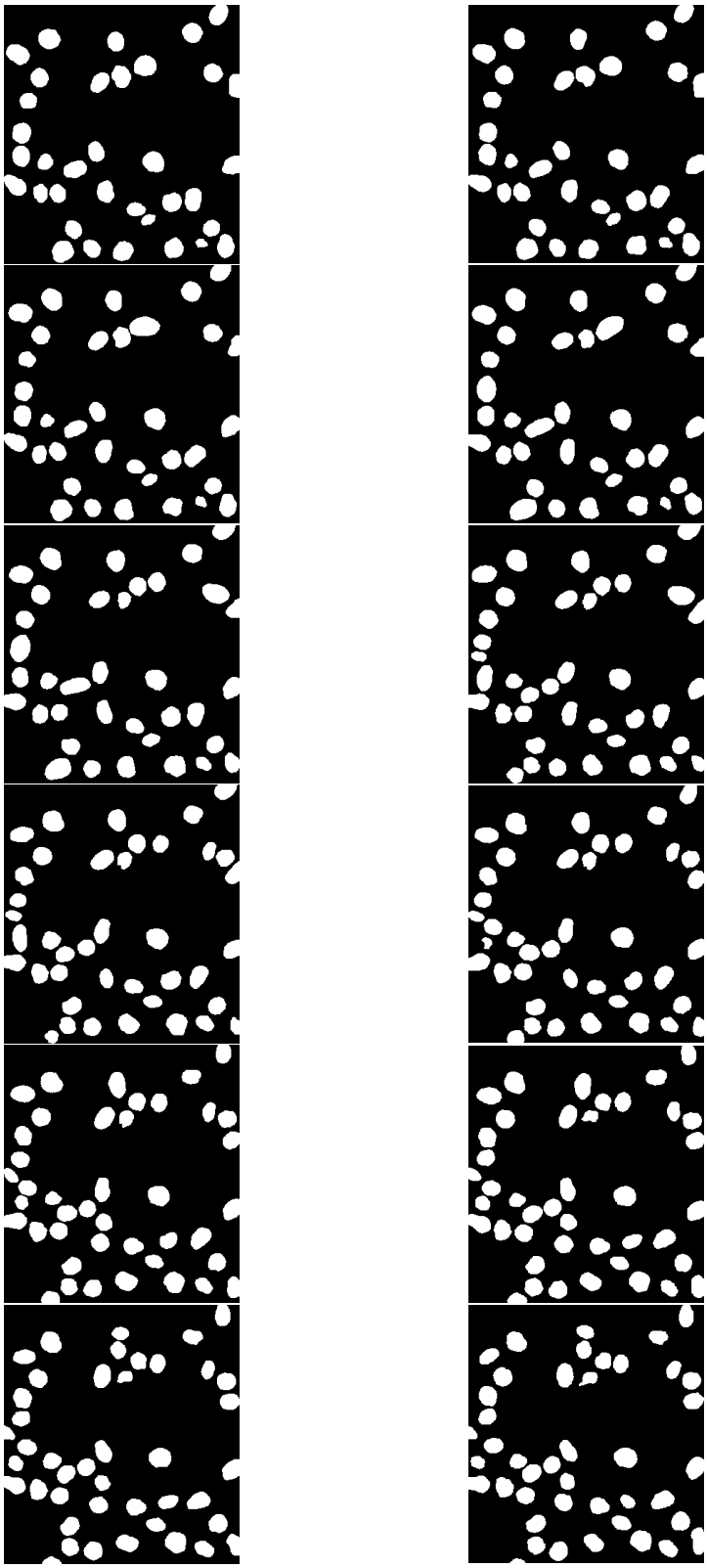


Figure 4.2: Series of continuous of ground truth images

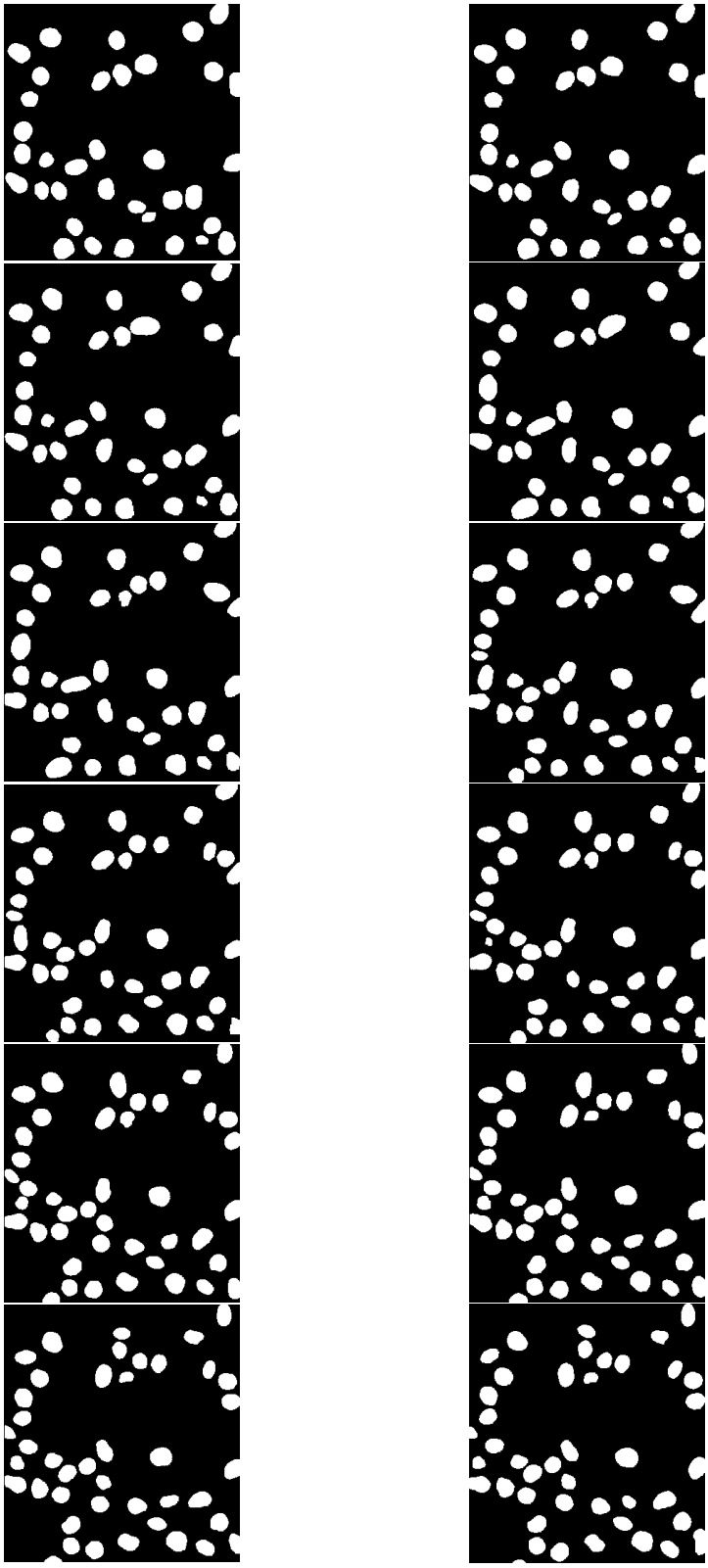


Figure 4.3: Series of continuous images segmented by LSTM-Unet

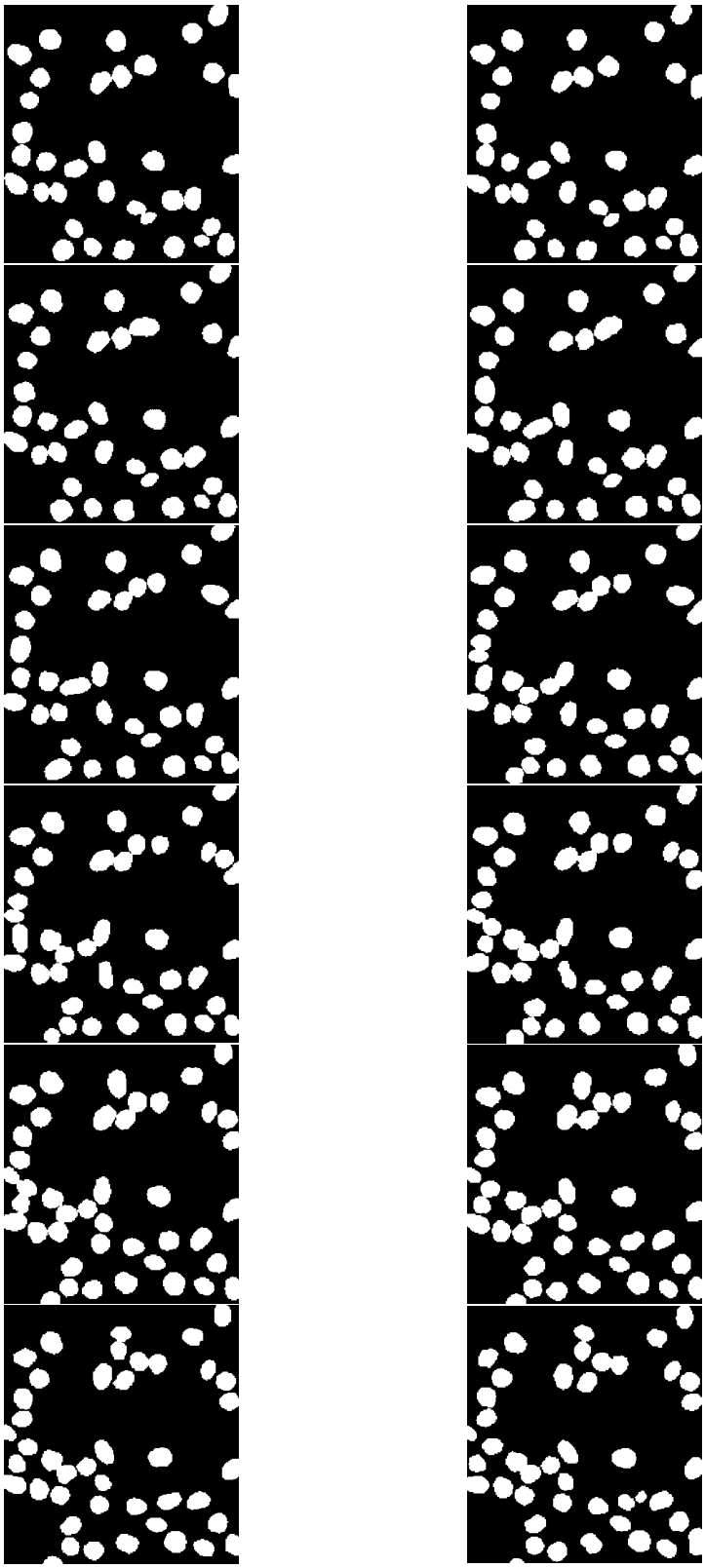


Figure 4.4: Same series of images as in Figure 4.3 segmented by graph cut

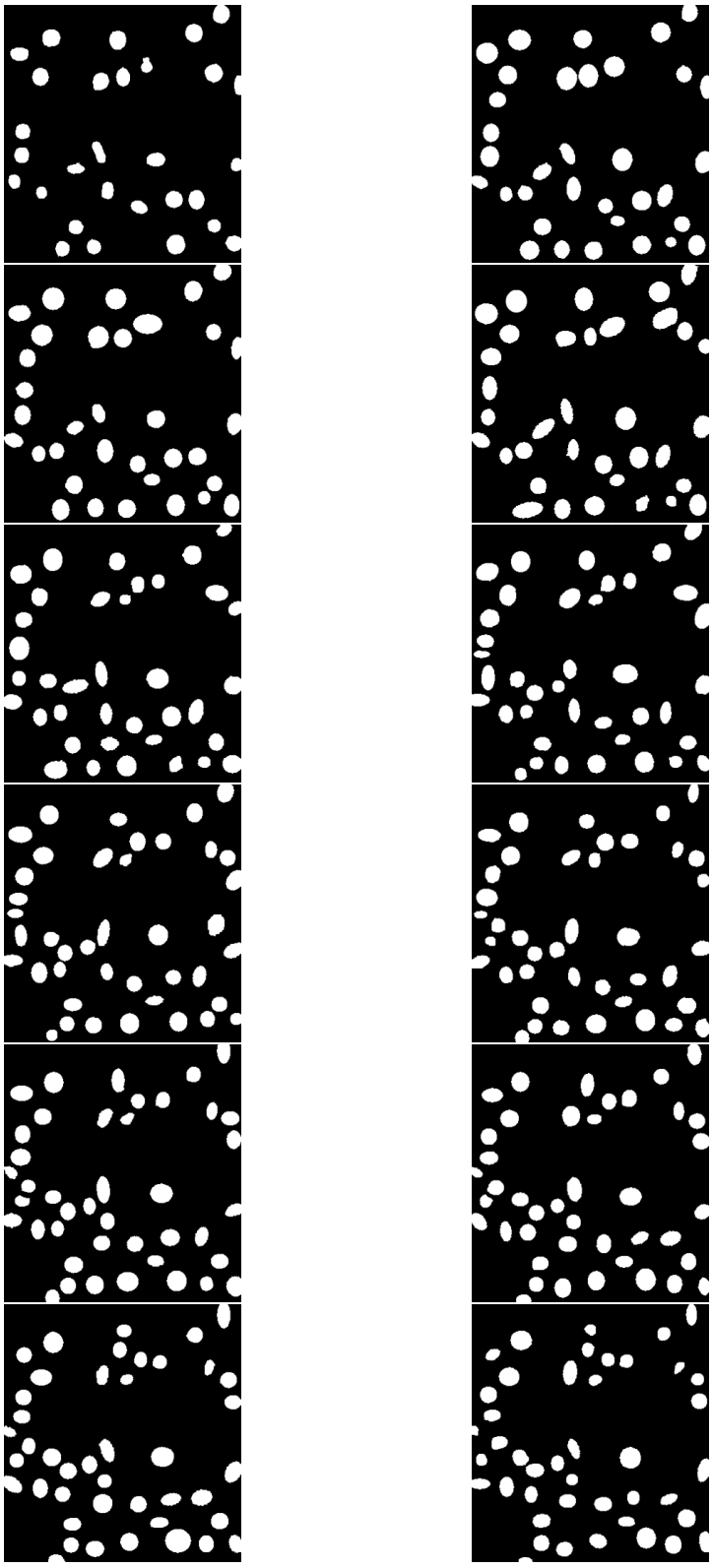


Figure 4.5: Series of continuous images segmented by U-Net

a capability to distinguish foreground/background objects by user to initialize the graph construction and it only needs CPU included device which is something that is always accessible. Table 4.1 displays how graph cut algorithm is accurate and able to extract salient part of the image as LSTM-Unet.

The baseline of our comparison is the original U-Net Which has been introduced in detail in Chapter 3 (Figure 4.5). Next we will illustrate the performance of each technique on Flou-N2DH-SIM+ dataset. Figure 4.1 and Figure 4.2 illustrate some sample of original images and ground truth from Flou-N2DH-SIM+ dataset. The dataset consists of total of 65 images with ground truth that has been used to train LSTM-Unet and graph cut. The test dataset includes 30 samples of these images.

As we can see from Table 4.1 and 4.2, displaying only 20 instances of our test set, both graph cut and LSTM-Unet give promising results. However, it can be understood from the quantitative measure that graph cut is consistent in its performance and results are close to each other. Also, there can be cases (Figure 4.6) where graph cut outperforms the LSTM-Unet.

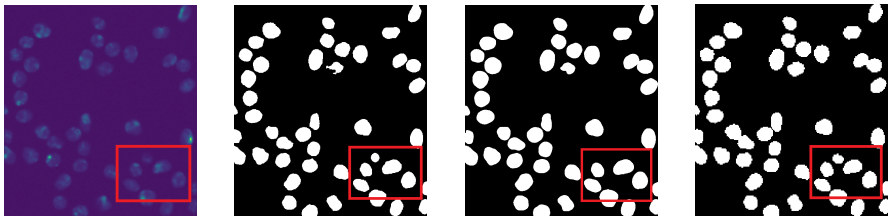
Each method was evaluated using the scheme proposed in the online version of the Cell Tracking Challenge [2]. Specifically, SEG for segmentation. The SEG measure *mean Jaccard index*, (Equation 4.1) of a pair of ground truth label X and its corresponding segmentation Y is defined as

$$Jaccard = \frac{|X \bigcap Y|}{|X \bigcup Y|}. \tag{4.1}$$

In addition to Jaccard Index we have also computed another measure, *Dice coefficient*, to measure the segmentation accuracy of each methods. Dice index, given two sets X and Y is described as

$$DSC = \frac{2|X \bigcap Y|}{|X| + |Y|} \tag{4.2}$$

where X is the ground truth image and Y is the segmented image by each technique



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

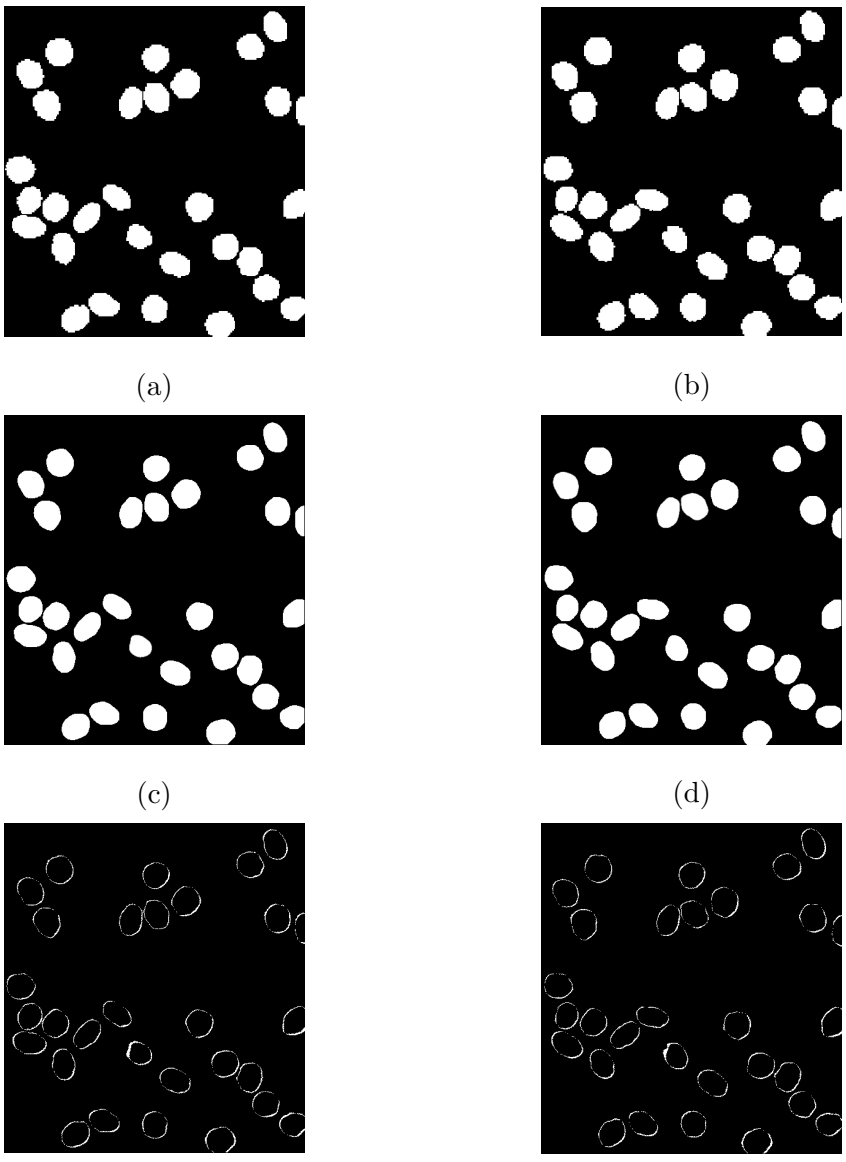
Figure 4.6: The case where graph cut out performs LSTM-Unet. (a) is the original image. (b) is the mask provided from ISBI Cell Tracking Challenge. (c) shows the result of LSTM-Unet and (d) is the output of graph cut.

(Table 4.1).

Furthermore, we have additionally evaluated *sensitivity*, *specificity*, and *accuracy* for both LSTM-Unet and graph cut. Sensitivity (also called the true positive rate) measures the proportion of actual positives that are correctly identified. Specificity (also called the true negative rate) is defined as the proportion of the actual negatives that are correctly identified. Accuracy is used as a statistical measure of how well a test correctly identifies or excludes a condition. In other words accuracy is the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. Table 4.5 displays 20 instances of our test set for Flou-N2DH-SIM+ dataset.

We have also presented the differences of resulting image segmentation for each method from the provided ground truth. See Figures 4.7, 4.8, and 4.9 respectively for LSTM-Unet, graph cut, and U-Net.

In addition to Flou-N2DH-SIM+ dataset we have also evaluated each method on another set of dataset, PhC-C2DH-U373. This dataset includes 115 images of sizes 520×696 with their ground truth that has been used to training each model on. Figures 4.10 and Figure 4.11 display some examples of original images and their corresponding ground truth of this dataset. The segmentation results from each technique are shown in Figures 4.12 and 4.13 for LSTM-Unet and graph cut.

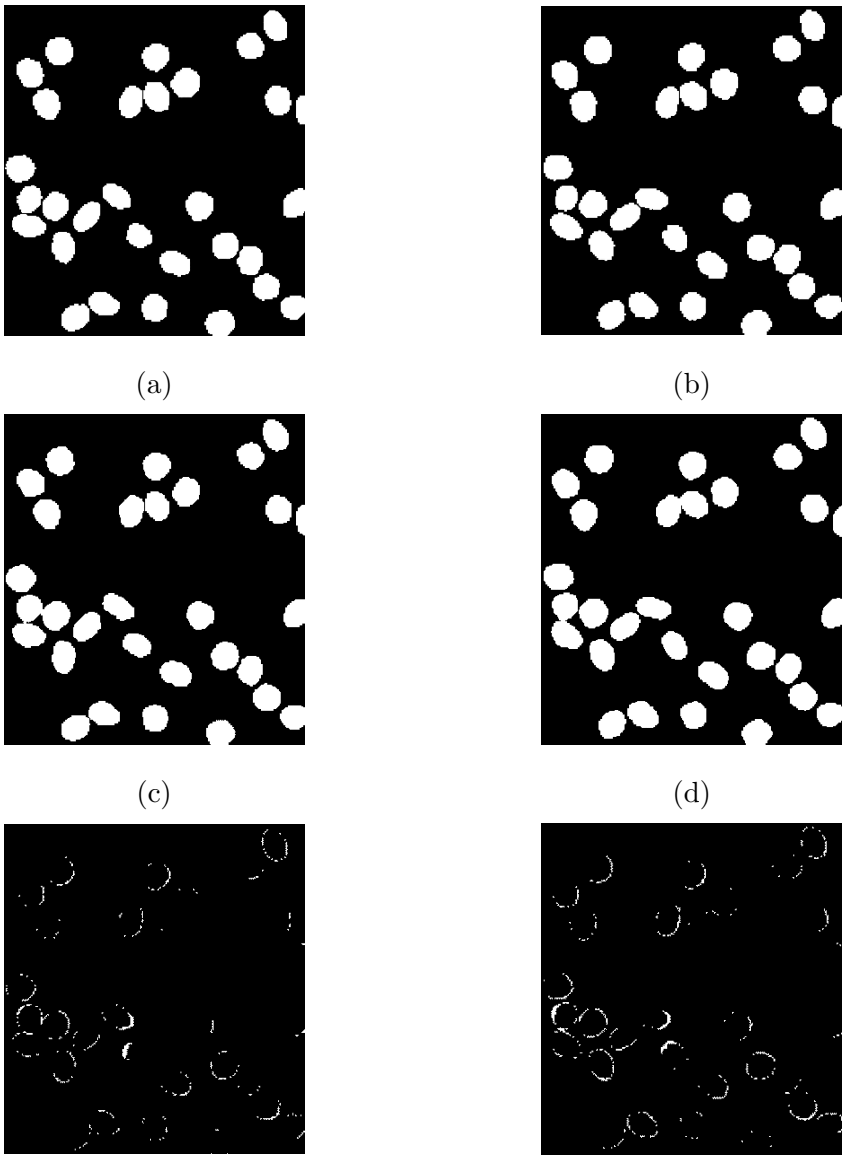


(a) (b)

(c) (d)

(e) (f)

Figure 4.7: Displaying the difference images segmented by LSTM-Unet. (a) and (b) are the Ground Truth mask. (c) and (d) are the segmented images by LSTM-Unet, and (e) and (f) are the difference images of the result from ground truth.

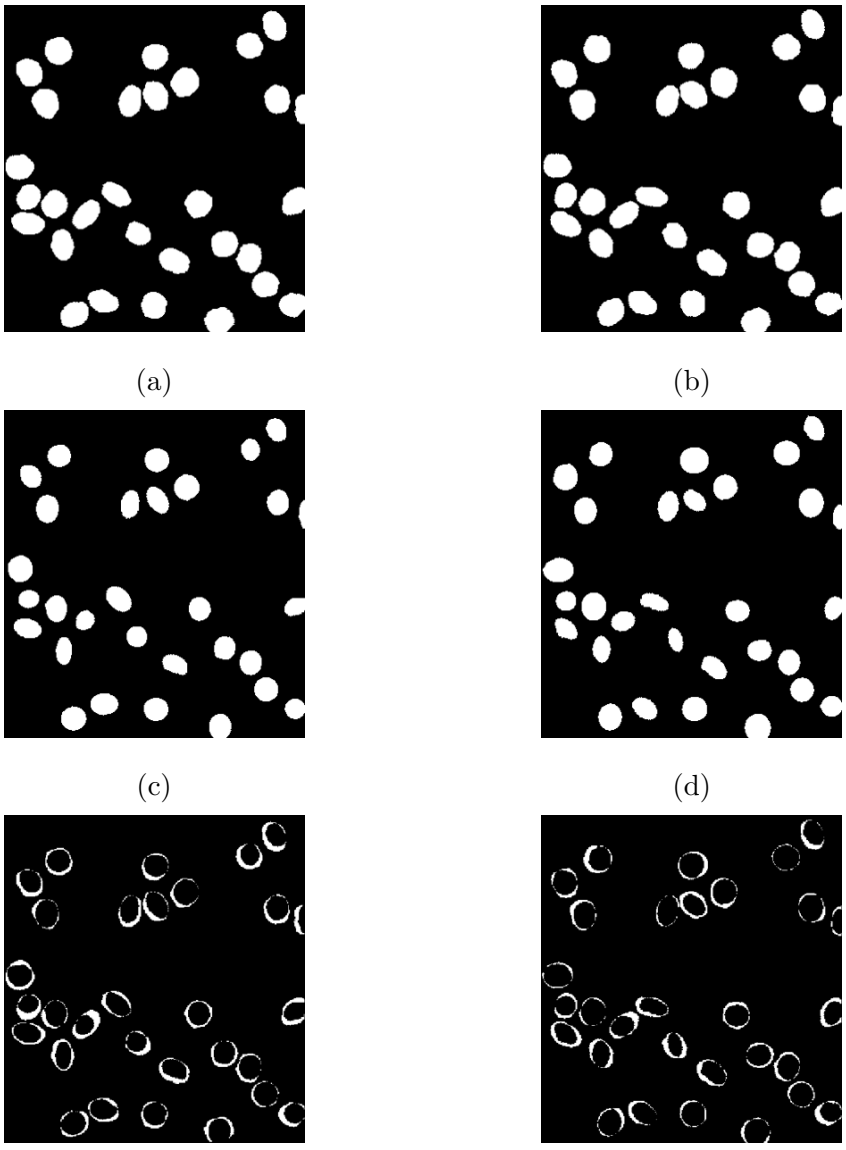


(a) (b)

(c) (d)

(e) (f)

Figure 4.8: Displaying the difference images segmented by graph cut. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by graph cut, and (e) and (f) is the difference images of the result from ground truth.



(a) (b)

(c) (d)

(e) (f)

Figure 4.9: Displaying the difference images segmented by Unet. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by U-Net, and (e) and (f) is the difference images of the result from ground truth.

Image ID	Jaccard Index	Dice Index
0	0.96	0.93
1	0.96	0.93
2	0.96	0.93
3	0.96	0.93
4	0.96	0.93
5	0.96	0.93
6	0.96	0.93
7	0.96	0.93
8	0.96	0.93
9	0.96	0.93
10	0.96	0.93
11	0.96	0.93
12	0.96	0.93
13	0.96	0.93
14	0.96	0.93
15	0.96	0.92
16	0.96	0.92
17	0.96	0.92
18	0.95	0.92
19	0.95	0.92

Image ID	Jaccard Index	Dice Index
0	0.97	0.95
1	0.97	0.95
2	0.97	0.95
3	0.97	0.95
4	0.97	0.95
5	0.97	0.95
6	0.97	0.95
7	0.97	0.95
8	0.97	0.95
9	0.97	0.95
10	0.97	0.95
11	0.97	0.95
12	0.97	0.95
13	0.97	0.95
14	0.97	0.95
15	0.97	0.95
16	0.97	0.95
17	0.96	0.94
18	0.96	0.94
19	0.95	0.94

Image ID	Jaccard Index	Dice Index
0	0.72	0.69
1	0.72	0.69
2	0.65	0.63
3	0.66	0.63
4	0.64	0.63
5	0.64	0.73
6	0.63	0.62
7	0.63	0.62
8	0.64	0.63
9	0.64	0.63
10	0.65	0.63
11	0.65	0.63
12	0.66	0.63
13	0.79	0.77
14	0.66	0.63
15	0.66	0.63
16	0.66	0.63
17	0.64	0.63
18	0.65	0.63
19	0.65	0.63

Table 4.1: Jaccard and Dice indices for LSTM-Unet, graph cut, and U-Net on Flou-N2DH-SIM+ dataset. Table on the left shows result for LSTM-Unet and the table on the middle displays result for the same series of images resulted by graph cut and the table on the right illustrate the results for U-Net.

Method	Jaccard Average	Standard Deviation	Range
LSTM-Unet	0.95	0.04	0.82 - 1.08
Graph Cut	0.97	0.01	0.94 - 0.99
U-Net	0.70	0.08	0.46 - 0.94

Table 4.2: Comparing the performance of each technique using Jaccard Index

Method	Dice Average	Standard Deviation	Range
LSTM-Unet	0.93	0.04	0.91 - 0.94
Graph Cut	0.95	0.02	0.88 - 1.00
U-Net	0.69	0.06	0.45 - 0.90

Table 4.3: Comparing the performance of each technique using Dice Index

Method	num of Epochs	Training Time	Testing Time	Memory Usage	Device
LSMT-Unet	1000000	14 days	0.03 hr	0.8 GB	TITAN XP
Graph Cut	-	-	1 hr	0.45 GB	CPU

Table 4.4: Detailed comparison of LSTM-Unet and graph cut performance.

Table 4.8 gives details on Jaccard and Dice measure for PhC-C2DH-U373 for 12 test images. Table 4.10 displays sensitivity and specificity measures evaluated for the same set of images. In Tables 4.9 and 4.11 we present a comparison on LSTM-Unet and graph cut regarding the result of segmentation for PhC-C2DH-U373 test images.

We can understand from the Table 4.1 that how each technique is performing regarding the segmentation as we defined Jaccard and Dice to be our segmentation measures. Take away from this result, is that both LSTM-Unet and garph cut perform with higher measures than U-Net implying that they can perform with higher accuracy regarding the segmentation task. Between graph cut and LSTM-Unet, graph cut still performs with a slightly higher accuracy. Also, it is implied form Tables 4.2 and 4.3 that in terms of consistency graph cut outperforms both the LSTM-Unet and U-Net models by having the standard deviation of 0.01 and also a closer range, where range is computed as $(Average - 3 \cdot SD, Average + 3 \cdot SD)$. LSTM-Unet is the second best and U-Net shows lack of consistency as it has a wide range.

In terms of sensitivity and specificity both graph cut and LSTM-Unet evaluate to the same average, implying that both methods can accurately choose the correct class

Image ID	Sensitivity	Specificity	Accuracy
0	0.97	0.97	0.97
1	0.97	0.97	0.97
2	0.98	0.97	0.97
3	0.98	0.96	0.97
4	0.97	0.96	0.97
5	0.97	0.97	0.97
6	0.99	0.97	0.97
7	0.99	0.97	0.97
8	0.98	0.98	0.97
9	0.99	0.98	0.97
10	0.98	0.97	0.97
11	0.98	0.97	0.97
12	0.99	0.97	0.97
13	0.99	0.97	0.97
14	0.98	0.97	0.97
15	0.99	0.97	0.97
16	0.99	0.97	0.97
17	0.98	0.97	0.97
18	0.99	0.97	0.97
19	0.98	0.97	0.97

Image ID	Sensitivity	Specificity	Accuracy
0	0.99	0.98	0.98
1	0.97	0.99	0.98
2	0.97	0.99	0.98
3	0.98	0.98	0.98
4	0.99	0.98	0.98
5	0.98	0.99	0.99
6	0.96	0.99	0.99
7	0.97	0.99	0.98
8	0.99	0.98	0.98
9	0.99	0.97	0.98
10	0.99	0.98	0.98
11	0.99	0.98	0.98
12	0.98	0.98	0.98
13	0.98	0.98	0.98
14	0.97	0.98	0.98
15	0.96	0.98	0.98
16	0.98	0.98	0.98
17	0.97	0.98	0.98
18	0.97	0.98	0.98
19	0.98	0.97	0.98

Table 4.5: Sensitivity and Specificity for both LSTM-Unet and graph cut on Flou-N2DH-SIM+ dataset. Table on the left shows result for LSTM-Unet and the table on the right displays result for the same series of images resulted by graph cut.

Method	Sensitivity Average	Specificity Average	Accuracy
LSTM-Unet	0.98	0.97	0.97
Graph Cut	0.98	0.98	0.98

Table 4.6: Comparing the performance of each technique using Sensitivity and Specificity

for pixels in the image, saying that they can assign 98 percent of the pixels belonging to class of foreground have been correctly identified as the foreground. Sometimes they even get really close to 100 percent accuracy which is the ideal condition. The same can be concluded for specificity where now the percentage shows that how much of the pixels belonging to background class have been assigned to their correct class. Both models seem to be consistent in their performance. Their consistency is visualized in Figures 4.7 and 4.8. We can understand that the difference image being calculated for each pair of result and ground truth in each model result in a similar image.

For the second dataset, the results have slightly dropped. See Table 4.8. It can be derived from the computed measures that graph cut and LSTM-Unet are sensitive to the input data and the changes in the images' texture can lead to a different performance. Still both models exceed the 80 percent accuracy rate for segmentation and they are consistent regarding the results by looking at Tables 4.9 and 4.11.

In case of sensitivity and specificity between LSMT-Unet and graph cut by interpreting the numbers in Tables 4.10 and 4.7 it is understood that both techniques are comparable in terms of accuracy. This is also visible in Figures 4.15 and 4.16.

Method	Sensitivity Average	Specificity Average	Accuracy
LSTM-Unet	0.95	0.97	0.97
Graph Cut	0.95	0.99	0.98

Table 4.7: Comparing the performance of each technique using Sensitivity and Specificity for PhC-C2DH-U373 dataset

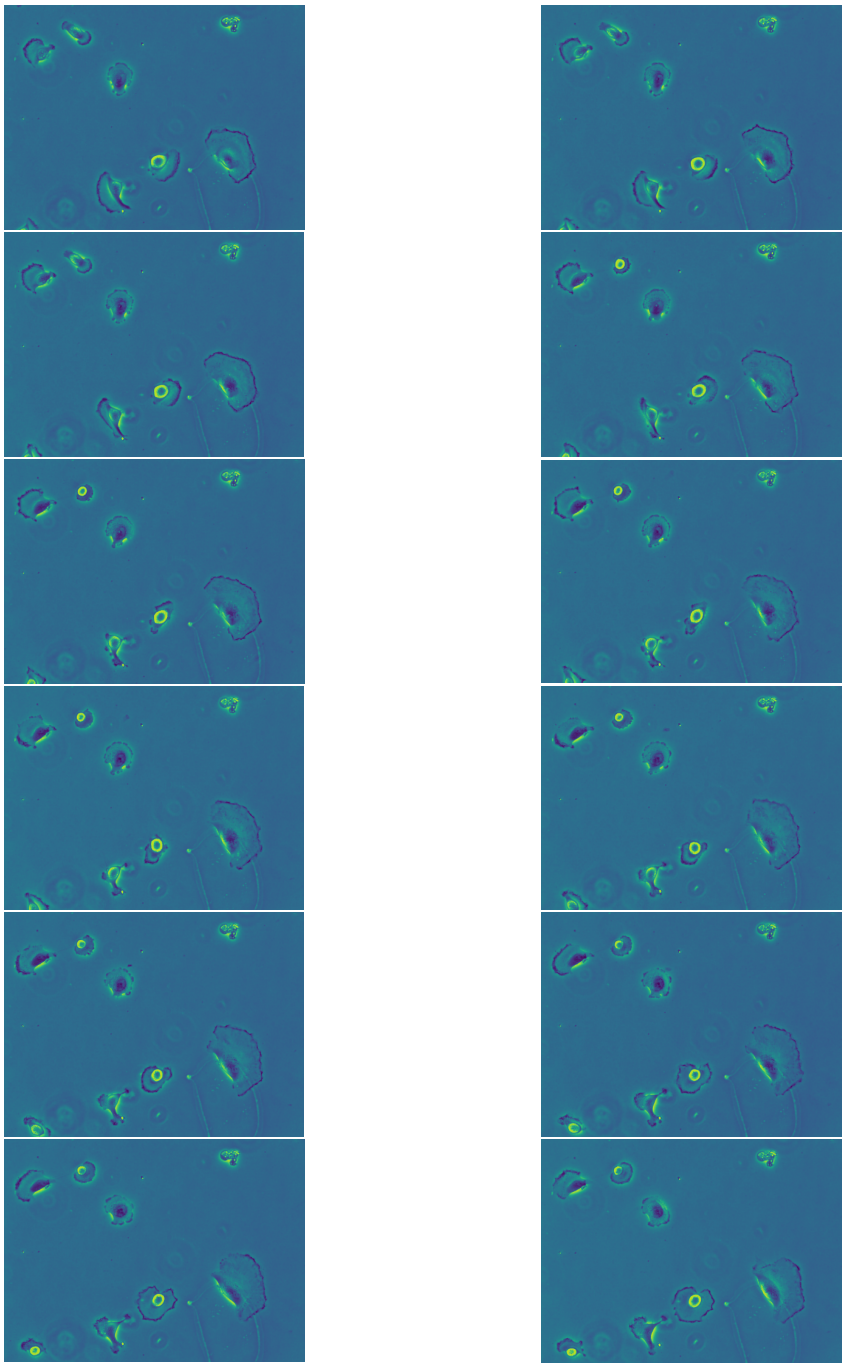


Figure 4.10: PhC-C2DH-U373 dataset original images

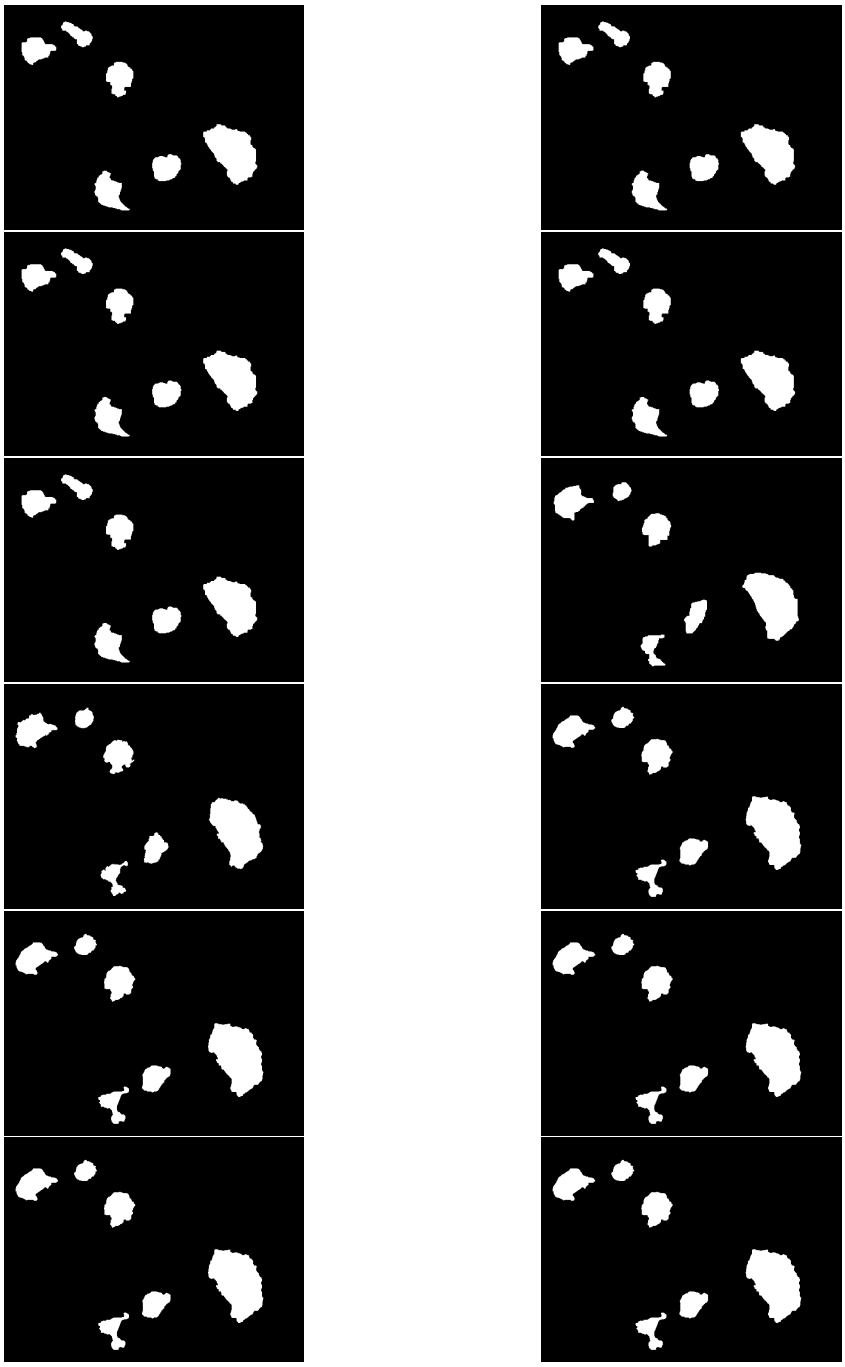


Figure 4.11: PhC-C2DH-U373 dataset ground truth images

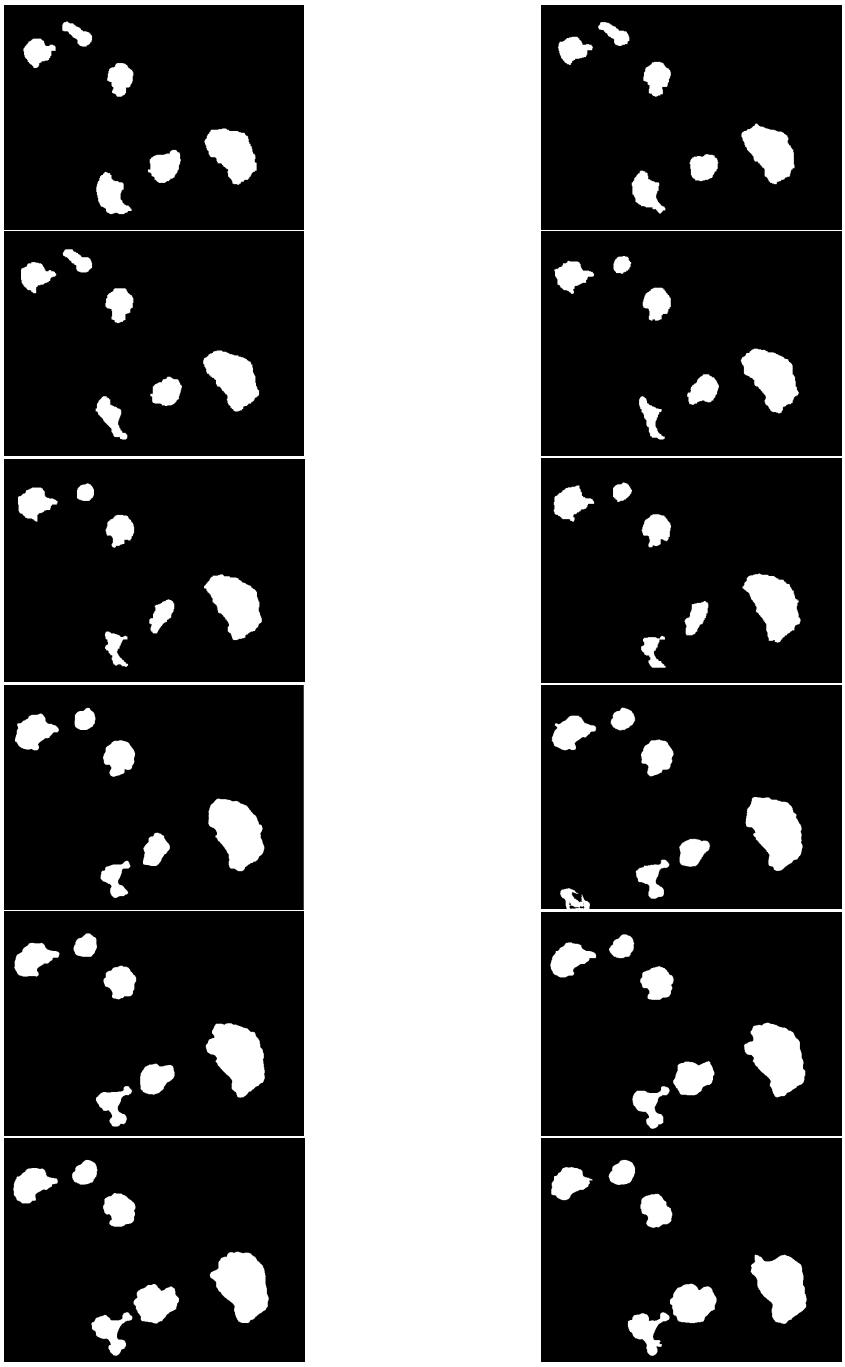


Figure 4.12: PhC-C2DH-U373 dataset segmented by LSTM-Unet

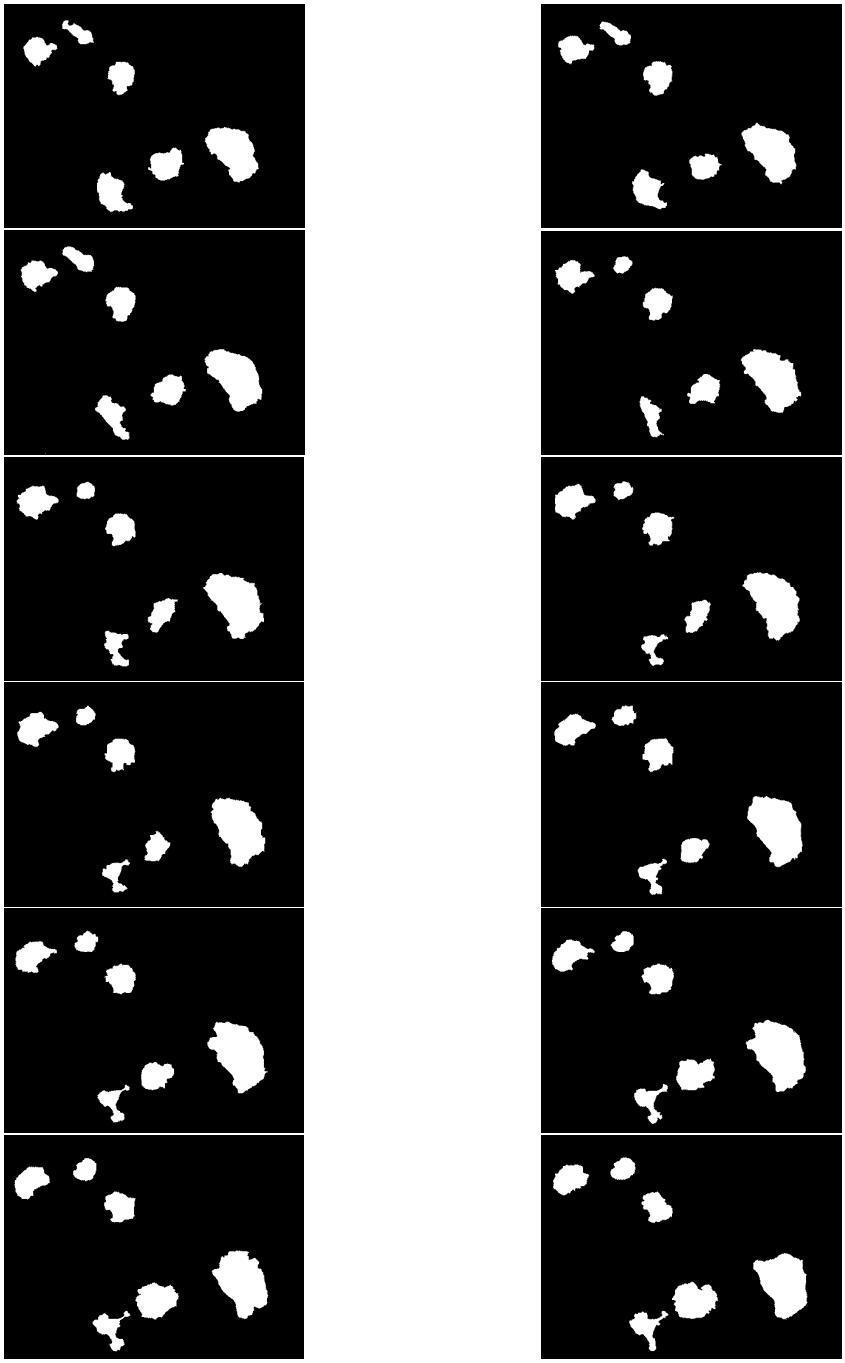


Figure 4.13: PhC-C2DH-U373 dataset segmented by graph cut

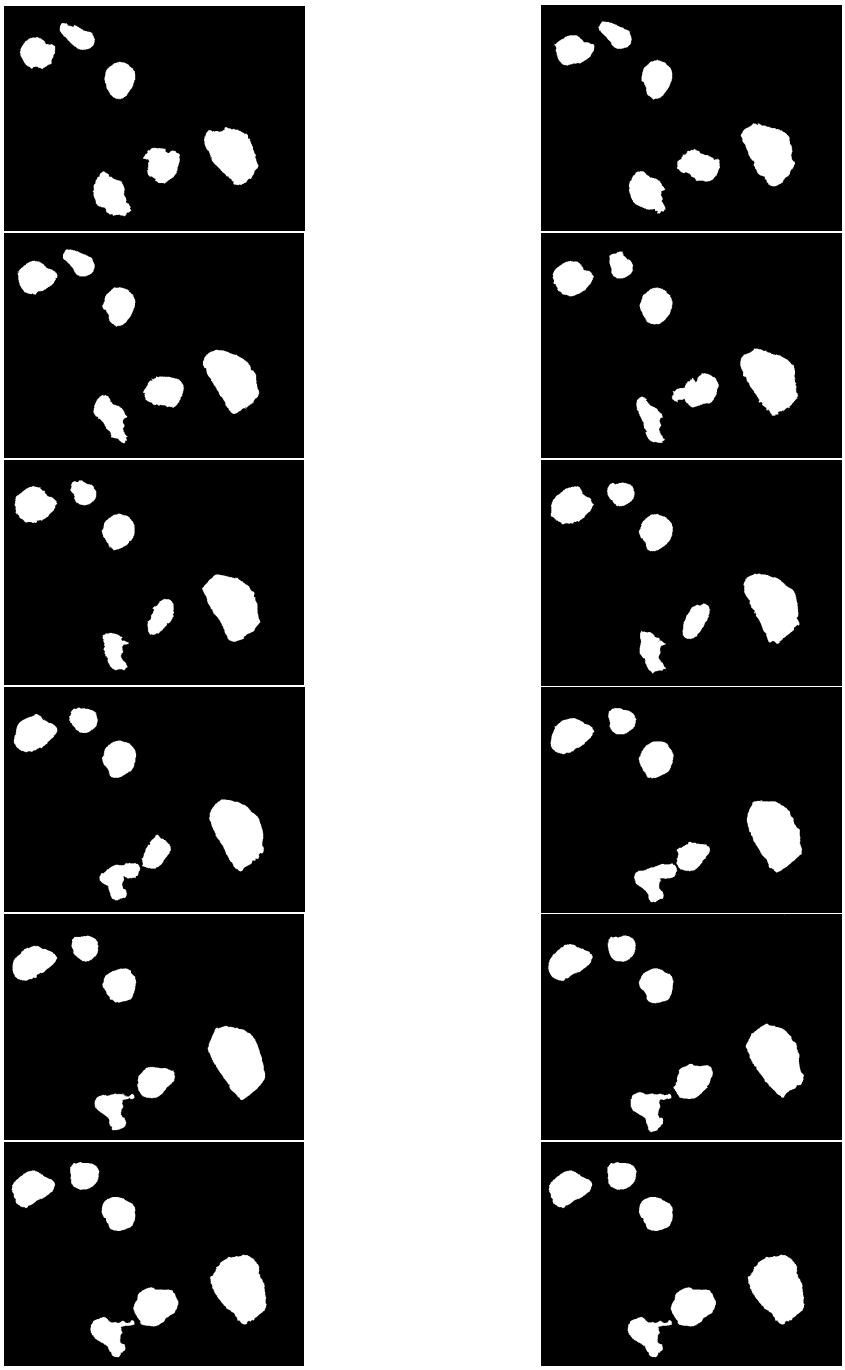


Figure 4.14: PhC-C2DH-U373 dataset segmented by U-Net

Image ID	Jaccard Index	Dice Index
0	0.87	0.77
1	0.91	0.83
2	0.87	0.77
3	0.81	0.68
4	0.76	0.61
5	0.87	0.78
6	0.9	0.82
7	0.91	0.83
8	0.87	0.77
9	0.83	0.71
10	0.79	0.66
11	0.74	0.59

Image ID	Jaccard Index	Dice Index
0	0.88	0.79
1	0.94	0.88
2	0.89	0.8
3	0.83	0.71
4	0.78	0.64
5	0.93	0.87
6	0.93	0.87
7	0.93	0.87
8	0.92	0.84
9	0.88	0.78
10	0.82	0.7
11	0.76	0.61

Image ID	Jaccard Index	Dice Index
0	0.86	0.76
1	0.85	0.74
2	0.85	0.74
3	0.84	0.73
4	0.86	0.75
5	0.86	0.76
6	0.85	0.73
7	0.85	0.74
8	0.87	0.77
9	0.85	0.74
10	0.85	0.73
11	0.82	0.69

Table 4.8: Jaccard and Dice indices for LSTM-Unet, graph cut, and U-Net on PhC-C2DH-U373 dataset. Table on the left shows result for LSTM-Unet and the table on the middle displays result for the same series of images resulted by graph cut and the table on the right illustrate the results for U-Net.

Method	Jaccard Average	Standard Deviation	Range
LSTM-Unet	0.84	0.06	0.66 - 1.02
Graph Cut	0.87	0.06	0.69 - 1.05
U-Net	0.85	0.01	0.82 - 0.89

Table 4.9: Comparing the performance of each technique using Jaccard Index for PhC-C2DH-U373 dataset

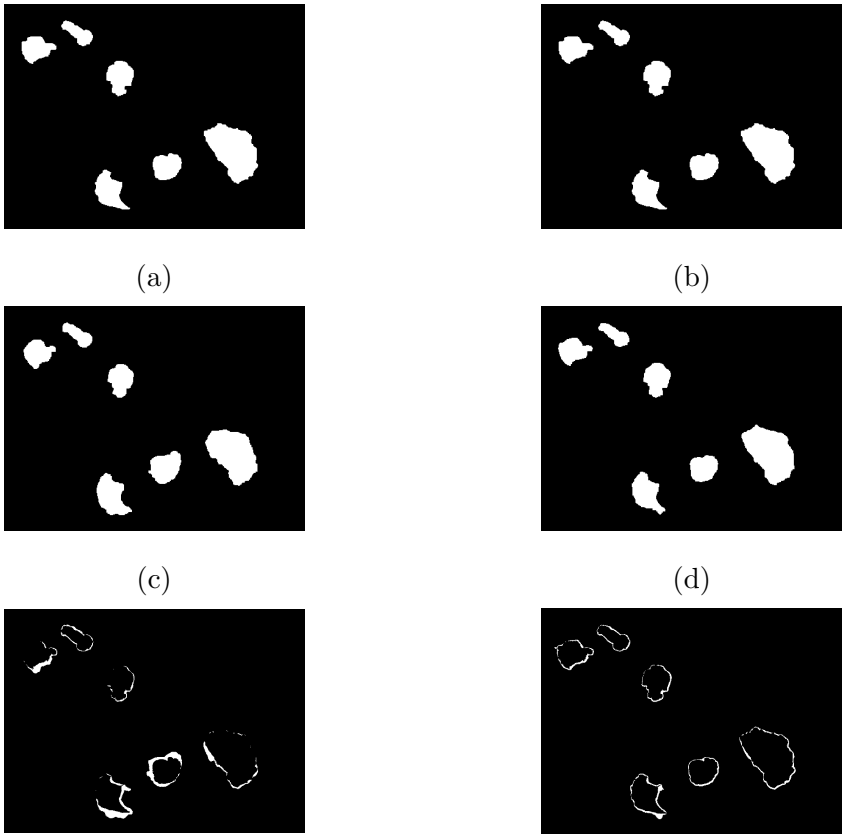
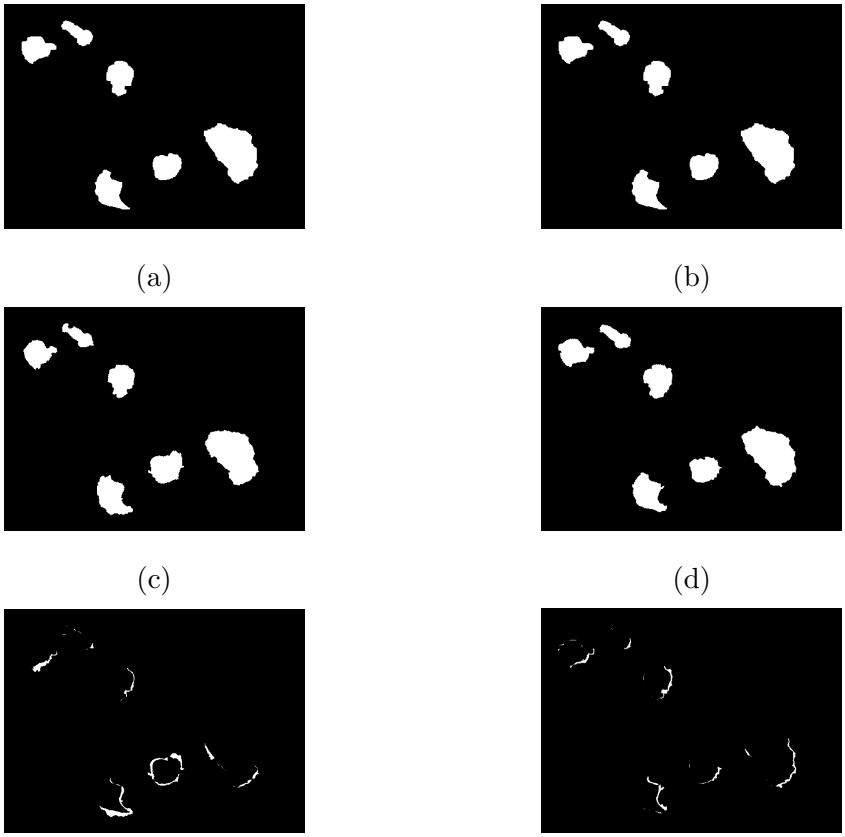


Figure 4.15: Displaying the difference images segmented by LSTM-Unet. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by LSTM-Unet, and (e) and (f) is the difference images of the result from ground truth for PhC-C2DH-U373 dataset.

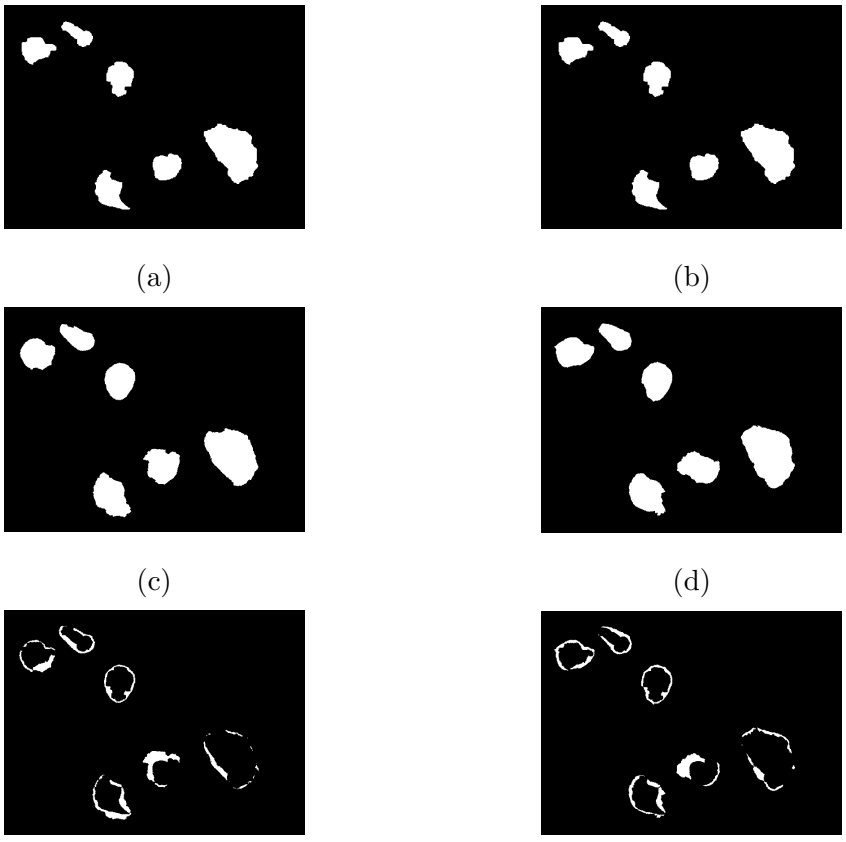


(a) (b)

(c) (d)

(e) (f)

Figure 4.16: Displaying the difference images segmented by graph cut. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by graph cut, and (e) and (f) is the difference images of the result from ground truth for PhC-C2DH-U373 dataset.



(a) (b)

(c) (d)

(e) (f)

Figure 4.17: Displaying the difference images segmented by U-Net. (a) and (b) are the Ground Truth masks. (c) and (d) are the segmented images by U-Net, and (e) and (f) is the difference images of the result from ground truth for PhC-C2DH-U373 dataset.

Image ID	Sensitivity	Specificity	Accuracy
0	0.97	0.98	0.98
1	0.99	0.98	0.98
2	0.97	0.98	0.98
3	0.9	0.97	0.97
4	0.84	0.97	0.96
5	0.99	0.98	0.98
6	0.99	0.98	0.98
7	0.99	0.98	0.98
8	0.98	0.98	0.98
9	0.98	0.97	0.97
10	0.94	0.96	0.96
11	0.88	0.96	0.95

Image ID	Sensitivity	Specificity	Accuracy
0	0.95	0.99	0.98
1	0.95	0.99	0.99
2	0.93	0.99	0.98
3	0.96	0.99	0.98
4	0.95	0.98	0.97
5	0.96	0.99	0.99
6	0.93	0.99	0.99
7	0.93	0.99	0.99
8	0.95	0.99	0.99
9	0.96	0.98	0.98
10	0.97	0.98	0.97
11	0.97	0.97	0.96

Table 4.10: Sensitivity and Specificity for both LSTM-Unet and graph cut on PhC-C2DH-U373 dataset. Table on the left shows result for LSTM-Unet and the table on the right displays result for the same series of images resulted by graph cut.

Method	Dice Average	Standard Deviation	Range
LSTM-Unet	0.74	0.08	0.49 - 0.98
Graph Cut	0.78	0.09	0.51 - 1.05
U-Net	0.74	0.02	0.68 - 0.8

Table 4.11: Comparing the performance of each technique using Dice Index for PhC-C2DH-U373 dataset

Chapter 5

Conclusion

In this thesis we explored two different techniques in depth in addition to the state-of-the-art algorithm, *U-Net*, and analyzed the performance of each of the approaches. Both qualitative and quantitative reports illustrate that graph cut and LSTM-Unet execute with higher accuracy than the original U-Net, graph cut with 97 percent accuracy rate, LSTM-Unet with 95 percent accuracy rate, and U-Net with 70 percent accuracy rate.

Even though, two out of the three methods were based on deep learning and neural networks module, and also the input of each method varies from the others (a sequence of frames for LSMT-Unet, and single frames for graph cut and U-Net) we were able to apply a comparison with a conventional technique using graph theory and optimization to reach the same and in some cases higher accuracy rate.

The take away from our analysis is that not all segmentation problem needs to be solved using deep learning and neural networks. Depending on the task and the dataset being used one might think of other techniques such as conventional ones which require less devices and training time.

One significant conclusion from our results can be that although the input settings for each method varies but at the end all of our techniques are based on optimization and in comparison graph cut method without training or using the ground truth images

can solve the cell segmentation problem as accurate and even in some cases with higher accuracy rate than deep modules.

In conclusion, we suggest that when approaching a segmentation technique, utilizing conventional techniques such as graph cut may be a suitable approach for some applications. Depending on the problem definition and dataset at hand with the available devices, deep learning may not always be the right solution. The use of GPUs can make the process faster but one also should consider the time required to implement the neural networks and also the time they need to be trained. Sometimes, it may take up to weeks to train the network but still don't get the satisfactory outputs and need to start training with different hyper parameters which makes the process even longer but with graph cut, there is no training needed. All that is important is what parameters and normalizers to consider depending on the problem and how to construct the graph to reach the desired solution. Moreover, the other advantages of graph cut can be their independence from the ground truth data. While in order to train a network there should be a ground truth to train the networks and optimize the weights based on.

Note that both graph cut and U-Net are implemented so that at each step during their performance they are given one input image and they perform on the singular frame and output the resulting segmentation (one can think of it as a *for loop* in programming) whereas LSTM-Unet is given a sequence of images all at once and for each prediction is able to refer to the previous frame and gather some helpful information and forward them to the current step. Given this ability it may have overcome the sequence cell segmentation task to some extends. This may not be the case for every segmentation problem and this being said, depending on the problem definition being mindful of the conventional techniques such as graph cut could make the process of segmentation less complex.

Bibliography

[1] Understanding lstm networks, 2015.

[2] International Symposium on Biomedical Imaging (ISBI) cell tracking challenge, 2019.

[3] Charu C Aggarwal. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[4] A Arbelle and TR Raviv. Microscopy cell segmentation via convolutional lstm networks. corr. *arXiv preprint arXiv:1895.11247*, 2018.

[5] Wolfgang Birkfellner. Image processing in clinical practice. *Applied Medical Image Processing: A Basic Course*, page 45, 2016.

[6] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[7] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[8] Yuri Boykov and Olga Veksler. Graph cuts in vision and graphics: Theories and applications. In *Handbook of mathematical models in computer vision*, pages 79–96. Springer, 2006.

[9] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[10] Vikram Chalana and Yongmin Kim. A methodology for evaluation of boundary detection algorithms on medical images. *IEEE Transactions on medical imaging*, 16(5):642–652, 1997.

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[12] Shang-Fu Chen, Yi-Chen Chen, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. Order-free rnn with visual attention for multi-label classification. *arXiv preprint arXiv:1707.05495*, 2017.

[13] Tianshui Chen, Zhouxia Wang, Guanbin Li, and Liang Lin. Recurrent attentional reinforcement learning for multi-label image recognition. *arXiv preprint arXiv:1712.07465*, 2017.

[14] Hwan Soo Choi, David R Haynor, and Yongmin Kim. Partial volume tissue classification of multichannel magnetic resonance images-a mixel model. *IEEE Transactions on Medical Imaging*, 10(3):395–407, 1991.

[15] D Louis Collins, Alex P Zijdenbos, Vasken Kollokian, John G Sled, Noor J Kabani, Colin J Holmes, and Alan C Evans. Design and construction of a realistic digital brain phantom. *IEEE transactions on medical imaging*, 17(3):463–468, 1998.

[16] JW Davenport, JC Bezdek, and RJ Hathaway. Parameter estimation for finite mixture distributions. *Computers & Mathematics with Applications*, 15(10):819–828, 1988.

[17] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2015.

[18] King-Sun Fu and JK Mui. A survey on image segmentation. 13(1):3–16, 1981.

[19] DR Fulkerson and LR Ford. *Flows in networks*. Princeton University Press, 1962.

[20] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.

[21] Yossi Gandelsman, Assaf Shocher, and Michal Irani. double-dip: Unsupervised image decomposition via coupled deep-image-priors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 6, page 2, 2019.

[22] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[24] Dorothy M Greig, Bruce T Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society: Series B (Methodological)*, 51(2):271–279, 1989.

[25] T. Nguyen J. Zheng H. Li, J. Cai. A benchmark for semantic image segmentation. In *ICME*, 2013.

[26] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[27] Bing Song He, Feng Zhu, and Yong Gang Shi. Medical image segmentation. In *Advanced Materials Research*, volume 760, pages 1590–1593. Trans Tech Publ, 2013.

[28] R Craig Herndon, Jack L Lancaster, Arthur W Toga, and Peter T Fox. Quantification of white matter and gray matter volumes from t1 parametric images using fuzzy classifiers. *Journal of Magnetic Resonance Imaging*, 6(3):425–435, 1996.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[31] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[32] David A Langan, James W Modestino, and Jun Zhang. Cluster validation for unsupervised stochastic model-based image segmentation. *IEEE Transactions on Image Processing*, 7(2):180–195, 1998.

[33] Tianhu Lei and Wilfred Sewchand. Statistical approach to X-ray CT imaging and its applications in image analysis. ii. a new stochastic model-based image segmentation technique for x-ray ct image. *IEEE Transactions on Medical Imaging*, 11(1):62–69, 1992.

[34] Huai-Dong Li, Maria Kallergi, Laurence P Clarke, Vijay K Jain, and Robert A Clark. Markov random field for tumor detection in digital mammography. *IEEE transactions on medical imaging*, 14(3):565–576, 1995.

[35] Zhengrong Liang. Tissue classification and segmentation of MR images. *IEEE Engineering in Medicine and Biology Magazine*, 12(1):81–85, 1993.

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[37] Luca Lucchese and Sanjit K Mitra. Colour image segmentation: a state-of-the-art survey. *Proceedings-Indian National Science Academy Part A*, 67(2):207–222, 2001.

[38] Andrew Mehnert and Paul Jackway. An improved seeded region growing algorithm. *Pattern Recognition Letters*, 18(10):1065–1071, 1997.

[39] John L Muerle. Experimental evaluation of techniques for automatic segmentation of objects in a complex scene. *Pictorial pattern recognition*, pages 3–13, 1968.

[40] Kamyar Nazeri Naeini. Structure guided image restoration a deep learning approach (master's thesis). 2019.

[41] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.

[42] Bo Peng, Lei Zhang, and David Zhang. A survey of graph theoretical approaches to image segmentation. *Pattern recognition*, 46(3):1020–1038, 2013.

[43] Dzung L Pham and Jerry L Prince. An adaptive fuzzy c-means algorithm for image segmentation in the presence of intensity inhomogeneities. *Pattern recognition letters*, 20(1):57–68, 1999.

[44] Dzung L Pham, Chenyang Xu, and Jerry L Prince. Current methods in medical image segmentation. *Annual review of biomedical engineering*, 2(1):315–337, 2000.

[45] Jagath C Rajapakse, Jay N Giedd, and Judith L Rapoport. Statistical approach to segmentation of single-channel cerebral MR images. *IEEE transactions on medical imaging*, 16(2):176–186, 1997.

[46] Maryam Rastgarpour and J Shanbehzadeh. Application of AI techniques in medical image segmentation and novel categorization of available methods and. In *Tools, Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 Vol I, IMECS 2011, March 16-18, 2011, Hong Kong*. Citeseer, 2011.

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

[48] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[49] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach (global 3rd edition). *Essex: Pearson*, 2016.

[50] Prasanna K Sahoo, SAKC Soltani, and Andrew KC Wong. A survey of thresholding techniques. *Computer vision, graphics, and image processing*, 41(2):233–260, 1988.

[51] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[52] Vibhakar Shrimali, RS Anand, and Vinod Kumar. Current trends in segmentation of medical ultrasound b-mode images: A review. *IETE technical review*, 26(1):8–17, 2009.

[53] Jan Erik Solem. *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. " O'Reilly Media, Inc.", 2012.

[54] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[55] William M Wells, W Eric L Grimson, Ron Kikinis, and Ferenc A Jolesz. Adaptive segmentation of MRI data. *IEEE transactions on medical imaging*, 15(4):429–442, 1996.

[56] Peter Wust, Johanna Gellermann, Jürgen Beier, Susan Wegner, Wolfgang Tilly, Jens Tröger, Detlev Stalling, H Oswald, HC Hege, P Deuflhard, et al. Evaluation of segmentation algorithms for generation of patient models in radiofrequency hyperthermia. *Physics in Medicine & Biology*, 43(11):3295, 1998.

[57] Z. Yan, W. Liu, S. Wen, and Y. Yang. Multi-label image classification by feature attention network. *IEEE Access*, 7:98005–98013, 2019.

[58] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[59] Yu Jin Zhang. A survey on evaluation methods for image segmentation. *Pattern recognition*, 29(8):1335–1346, 1996.

[60] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5513–5522, 2017.

[61] Tranos Zuva, Oludayo O Olugbara, Sunday O Ojo, and Seleman M Ngwira. Image segmentation, available techniques, developments and open issues. *Canadian Journal on Image Processing and Computer Vision*, 2(3):20–29, 2011.