DEVELOPMENT OF A MACHINE LEARNING PLATFORM FOR ANALYSIS OF MITOCHONDRIAL FEATURES IN LIVE-CELL IMAGES

A Thesis Submitted to
the Graduate School of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Yalçın TARKAN

December 2022

İZMİR

ACKNOWLEDGEMENTS

First of all, I would like to thank my wife Figen for encouraging me to return to academy after all these years and support all the way by taking the heavy lift while I was struggling with all the work in the company, courses and research all together in the meantime, most of which collided with the global pandemic changing all our routines dramatically. I am also grateful to my son Kaan who supported me with the innovative ideas and feedback when I most needed in his study and research field.

I am delighted to learn a lot of inspiring and envisioning things in microscopy and microbiology thanks to Prof.Dr.Serdar Özçelik as the domain expert, as my advisor and a relentless motivator all along this learning journey. I owe him so much more than thanks for the invaluable information he conveyed enabling me to study, research and learn a lot in this amazing area of science that I was completely unaware of before. I think it is my duty to say thank you to the research Dr.Özge Tüncel and Dr.Iroda Saydullaeva in Nanobiolab Research Group IYTE for working hard to provide images my thesis research mainly based on. Last but not least, I want to thank my supervisor, guiding light and pillar of support Assoc. Prof. Dr. Tuğkan Tuğlular for his trust, always on help and facilitations. His outstanding support and guidance are the primary reason that my study and research in this challenging thesis has concluded. I learned a lot from him and for that I am deeply grateful too.

ABSTRACT

DEVELOPMENT OF A MACHINE LEARNING PLATFORM FOR ANALYSIS OF MITOCHONDRIAL FEATURES IN LIVE-CELL IMAGES

It is a laborious and error-prone manual process to mark the organelles in 2D and 3D images of living cells and identify the behavioral feedback to stimulations under measured conditions. This manual process can be simplified by being largely automated with machine learning techniques. We created a machine learning-based software platform named MitoML, which extracts sub-cellular structures, specifically mitochondria, and helps to identify the effects of external factors or changes under natural conditions. We investigate appropriate machine learning techniques for these objectives. Image processing and segmentation techniques with neural networks, enable researchers to carry out experiments with much better accuracy and a larger scale by automatically segmenting and counting the mitochondria, calculate the energy potentials based on region brightness. This way, analysis of mitochondria feedback in healthy and cancer cells under various conditions, such as nanomedicine and different treatment therapies, can be performed using MitoML. As a result of our work, we proposed a cascaded neural network architecture that can identify and count mitochondria, quantify its energy levels in fluorescence and other microscopy images, fast and at a standard reliable accuracy. Our test results outperformed the classical image processing algorithms provided in segmentation tools and software for medical image segmentation which was taken as a base line. Achieved accuracy rates 93.4% and %89.6 according to Dice and IoU metrics respectively are also better than any other related work encountered during the research. The proposed method can be improved to cover other sub-cellular structures relieving the researchers from non-standardized and laborious manual work which is prone to human error.

ÖZET

CANLI HÜCRE GÖRÜNTÜLERİNDE MİTOKONDRİ ÖZNİTELİKLERİNİN ANALİZİ İÇİN MAKİNE ÖĞRENMESİ PLATFORMU GELİŞTİRİLMESİ

Canlı hücrelerdeki organelleri 2 ve 3 boyutlu görüntülerinde işaretlemek ve izlenen koşullar altında uyarılara verdikleri davranışsal geri bildirimleri ölçmek zahmetli ve hataya açık bir manuel işlemdir. Bu manuel süreç, makine öğrenimi teknikleriyle büyük ölçüde otomatik hale getirilerek basitleştirilebilir. Bu çalışmada, özellikle mitokondri olmak üzere, hücre altı yapıları belirleyen, doğal koşullar altında ya da dış faktörlerden kaynaklanan değişikliklerin etkilerini izlemeye yardımcı olan MitoML adlı makine öğrenimi tabanlı bir yazılım platformu oluşturduk. Bu hedefler için uygun makine öğrenimi tekniklerini araştırarak alternatif çözümler üzerinde çalışmalar yaptık. Yapay sinir ağları ile görüntü işleme ve bölütleme teknikleri, mitokondrileri otomatik olarak bölerek ve sayarak, araştırmacıların büyük ölçekte deneyler yapmalarını, ölçüm işlemlerini yüksek doğrulukta gerçekleştirmelerini ve bölge parlaklığına dayalı enerji potansiyellerini hesaplamalarını sağlar. MitoML kullanılarak, sağlıklı ve kanser hücrelerinde, nano tıp ve farklı tedavi terapileri gibi çeşitli koşullar altında mitokondri geri bildiriminin analizi gerçekleştirilebilir. Çalışmamızın bir sonucu olarak, mitokondriyi tanımlayabilen ve sayabilen, floresan ve diğer mikroskopi görüntülerinde enerji seviyelerini hızlı ve standart güvenilir bir doğrulukla ölçebilen, kademeli bir sinir ağı mimarisi önerdik. Test sonuçlarımız, temel alınan tıbbi görüntü segmentasyonu için kullanılan araçlardaki klasik görüntü işleme algoritmalarından daha iyi performans gösterdi. Ayrıca, Dice ve IoU metriklerine göre sırasıyla %93,4 ve %89,6 doğruluk oranlarına erişerek, araştırma sırasında karşılaşılan diğer çalışmaların sonuçlarından daha iyi skorlar da elde etmeyi başardık. Önerilen yöntem, araştırmacıları, insan hatasına açık, standart güvenilirliği olmayan ve zahmetli diğer manuel işlemlerden, diğer hücre altı yapılar için de kurtaracak şekilde geliştirilebilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Digital transformation is raging everywhere, researchers devise new ways of solving problems or finding faster and better ways of doing ordinary tasks. Computer vision, especially image segmentation and object detection with machine learning (ML) techniques being used at an increasing rate, have been finding such beneficial novel approaches in the biomedical and healthcare domain lately. Development in machine learning algorithms and substantial rise in success rates of segmentation and image processing via ML results in widespread exposure in automation of tasks especially in microscopy, pathology, screening and imaging areas of healthcare. We see a highly increasing number of research on using artificial intelligence (AI) in efforts of making sense of x-ray, MRI, tissue and cell images. The sub-cellular cosmos is deep and much more complex than it is commonly thought. Investigation of sub-cellular structures like mitochondria, nucleus, lysosomes, ribosomes and their status changes, reactions to different situations take key place in understanding diseases and support to help find cures.

There is a lot of research around the world monitoring/testing the reactions of such sub-cellular structures under certain conditions in labs. Tests are hoped to be helpful for diagnosis and drug development by researching the reactions of mitochondria with different interventions like ray, radiation, or chemical substance exposures. This process produces vast amounts of data, as well as static snapshot images, recordings of a period during the tests may result in millions if not billions of image frames. All these images must be processed, marked, regions drawn, labeled, and analyzed by the researchers. Laboratory works, getting images of the tissues, cells and sub-cellular structures, grouping, segmenting and identification of desired patterns require significant effort.

Researchers already utilize various automation tools making use of legacy image processing techniques, but the benefits of these tools are not enough to process large batches of images in a short time. There is some software used in such processes which semi automate the process like image processing tools. There are also some image editors which assist in finding contours and drawing the borders. Despite all, the number of processable images is not remarkably high even with such assisting tools. Utilization of these tools and techniques still requires trained eyes and experience as well as too much dedicated time. Moreover, with such tools, the process is still human dependent and error prone just like any other human task.

Instead of putting all the workload on lab researchers, software systems making use of machine learning algorithms may be trained to do the job for the researchers. The success rate may be even better. Such a system provides reliably standard accuracy rates independent of the researcher's ability and much higher scalability processing enormous amounts of data.

In recent years, as well as other areas, many machine learning applications have been developed to process microscopic images too. There are even special machine learning models developed solely for biomedical image segmentation. Most well-known of them U-Net [1] was first introduced at Freiburg University for biomedical image segmentation in 2015. It is used in many previous works to provide automated or semi-automated segmentation in microscopic images.

The aim of this thesis is to devise a platform to assist researchers for segmentation of sub-cellular structures especially the mitochondria, which are energy producing double membrane-bound organelles found in most eukaryotic cells. We propose MitoML to enable researchers to analyze many times more images at a reliable and standard quality automatically. Despite the recent advancements in image processing with ML, since the microscopic images are much more indistinguishable and inter-object contrasts are low with respect to daily life photos, it is not an easy task to detect and segment the cells and sub-cellular structures is not an easy task. There are many research on cell segmentation, making use of ML and DL techniques. Moreover, it is also possible to find some commercial tools that make cell segmentation of microscopic images at some level of success. However, the sub-cellular organelle segmentation remains mostly uncharted territory. Mitochondria segmentation, labeling and energy potential monitoring is far from being common.

Making use of ML in microscopic image processing obviously is not an authentic or pioneer idea even for mitochondria or sub-cellular structures. However, the training process of ML models requires large datasets to reach satisfactory achievement levels for even the segmentation of objects in daily life photos. For instance, the popular image training set ImageNet (https://www.image-net.org) is composed of 14 million hand-annotated images. Places, scene recognition database of MIT (http://places.csail.mit.edu/index.html) has 2.5 million images in 205 categories. MNIST data set (Modified National Institute of Standards and Technology database http://yann.lecun.com/exdb/mnist/) which contains only writings of 10 digits, contains more than 60.000 handwritten single digit training images and 10.000 test images. On the other hand, acquiring tens of thousands of manually segmented images in microscopy for training is beyond reach.

The U-Net is a popular Convolutional Neural Networks (CNNs) model used in biomedical image segmentation known for its success with relatively smaller training datasets, we researched its abilities in segmentation of mitochondria by using pre-processing (like AHE & CLAHE) and post-processing (like watershed) methods, different loss functions and other ML hyper parameters like dropout and learning rate ratios. We also explored using Generative Adversarial Networks (GANs) to support finding self-supervised or semi-supervised solutions to cope with small training data availability. Primary work in this thesis focused on using two cascaded CNNs, both being U-Net but each employed for a different task, and therefore, configured with different parameters.

Outline is this thesis is as follows: Chapter 2 contains fundamentals, explanation of terms and terminology used. Chapter 3 presents the literature search and findings of related work. In Chapter 4, proposed methods and works carried out to solve the problem are described. An evaluation of the results is presented in chapter 5. Conclusion and future work are described in Chapter 6 followed by references and appendix sections.

CHAPTER 2

FUNDAMENTALS

2.1. Neural Networks

Neural Networks, also called Artificial Neural Networks (ANNs) occupy a significant role in machine learning. They aim to model neuron networks in the biological brains of humans and animals. A neural network is composed of a set of connected nodes called artificial neurons. By reflecting the behavior of biological brains, they are used to recognize patterns and help solve problems in the fields of machine learning and deep learning [2]. Some categories of neural networks in terms of training and operating methods can be listed as follows.

Supervised Learning [37]: It is a machine learning approach where models are trained using labeled or annotated data sets to classify and find patterns in order to learn to predict the outcomes with unseen similar data. During the training phase data is fed into the model until the neurons and parameters are adjusted to fit all manual annotations as a part of the validation process.

Unsupervised (self-supervised) Learning [37]: This learning method uses only data without any tags or labels by making use of patterns and by building an internal representation of the domain its exploring. They are generally used in clustering, classification and dimensionality reduction. These types of algorithms aimed to discover hidden patterns or data groupings without the need for human help or intervention.

Semi Supervised Learning [40]: It is a concept between supervised and unsupervised learning. The algorithms make use of a small set of manually annotated data and a large set of unlabeled data are called semi supervised learning algorithms.

Reinforcement Learning [41]: The main purpose in this learning method is to maximize the notion of cumulative reward while taking actions in an environment. An intelligent agent focuses on finding a balance between exploration of uncharted territory and exploitation of current knowledge by taking different actions each time and learning to make better choices based on metrics behaving as the reward-penalty system.

Transfer Learning [41]: Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem like making use of the knowledge gained while learning to classify cell nucleus in segmentation of mitochondria or other sub cellular structures.

Contrastive Learning [50]: It is an approach to formulate the task of finding similar and dissimilar things for a deep learning model. It is mostly utilized in feature space to facilitate model training and the point features are derived from the backbone network.

2.2. Neural Network Models

Convolutional Neural Networks (CNNs) [3][4][6]: CNNs are a class of artificial neural networks. CNNs are made of multilayer perceptrons. They are a fully connected version of feed forward artificial neural networks. Counter-intuitively, most convolutional neural networks are not invariant to translation, due to the down sampling operation they apply to the input [15]. CNN based neural network systems are widely used in medical classification tasks as well as common image processing and segmentation areas. CNN is an excellent feature extractor, therefore utilizing it to classify medical images can avoid complicated and expensive feature engineering. CNNs were established to overcome the problem related to the implementation of FFNNs (Feed Forward Neural Networks). CNNs can start making use of the input modality's two-dimensional spatial limitations while somehow decreasing the quantity of system parameters in the training phase. Convolution, pooling, and fully connected layers constitute CNN as three primary layers. These layers are engaged with certain spatial activities [9][10].

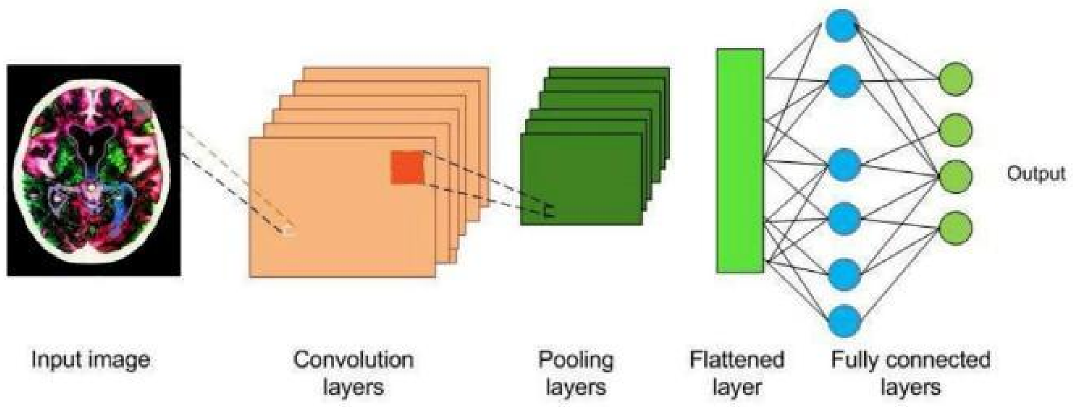


Figure 1. Building blocks of a CNN. Source [12]

Recurrent Neural Networks (RNNs): They are a class of neural network models that enables the output from some nodes to affect subsequent input to the same nodes recurrently which lets them show temporal dynamic behaviors. They are derived from Feed Forward Neural Networks and evolved to Long Short Term Memory (LSTM) based models. This makes them applicable to tasks such as unsegmented, connected handwriting recognition [16] or speech recognition [17][18].

Region-Based Convolutional Neural Networks (RCNNs) [4]: Region-based Convolutional Neural Networks are a family of machine learning models for computer vision and specifically object detection developed by Girshick et al., from UC Berkeley in 2014 [44].

Feed Forward Neural Networks (FFNNs) [5]: FFNNs and their fully connected forms, multilayer perceptrons, are made up of layers classified as input, hidden and output layers. Models consist of one input and one output layer and there may be none to many hidden layers. Layer sizes and connection between nodes may vary. Data is fed from input to output making classification on the output processing the input via nodes and weights.

Deep Convolutional Neural Network (DCNN) [8]: Deep Convolutional Neural Networks (DCNNs) have recently shown state of the art performance in high level vision tasks, such as image classification and object detection. This work brings together methods from DCNNs and probabilistic graphical models for addressing the task of pixel-level classification (also called "semantic image segmentation").

Autoencoders (AE): An autoencoder is a type of unsupervised learning neural network used to learn codings of unlabeled data [19] Autoencoders made up of input, output layers and symmetric hidden layers between them. They learn the representation of the input data in an unsupervised approach and stores it in hidden layers.

Generative Adversarial Networks (GANs): GANs are a class of machine learning frameworks designed by Ian Goodfellow et al. In 2014 [20]. There are 2 distinct engine networks that work, one generative and the other discriminative, against each other. While the generative engine in the model generates data aiming to produce indifferentiable results from the real, the discriminative engine network gets generated data and tries to detect if it is real or fake. Although originally proposed as a form of unsupervised learning, GANs have also proved useful for semi-supervised learning [21], fully supervised learning [22], and reinforcement learning [23].

Projective Adversarial Network (PAN): Three-dimensional medical image segmentation has always been a problem to be solved. Khosravan et al. [11] proposed a new segmentation network PAN to capture 3D semantics in an efficient and computationally efficient way. PAN integrates high-level 3D information through 2D projection, without relying on 3D images or enhancing the complexity of segmentation.

Support Vector Machines (SVMs): These are mature algorithms of supervised learning methods based on statistical learning frameworks, which are also known as Support Vector Networks. They are mainly used for linear and non-linear classification and regression analysis, developed by Vladimir Vapnik and Corrinna Cortes [24].

Markov Random Field (MRF): It is a stochastic process in the theory of probability that first appeared in the early 20th century and became a common tool in various problems as well as statistical mechanics. It is used in image processing beginning from date to the early 70s [25].

2.3. Neural Network Hyper Parameters

Neural network success heavily depends on hyper parameter setting. Setting hyper parameters in training and testing still is a standalone research area [31]. Hyper parameter definitions and roles, their detailed comprehension is a subject of deep learning literature. The following definitions are taken from [32][33][42].

Epoch: Epoch is the number of passes completed in the training process that the complete training dataset has been processed. One pass of the neural network model on all the training data set is called one epoch.

Batch Size: It represents the amount of data that will be used in each iteration in neural network trainings. If batch size is equal to all dataset, it is called batch-mode, if it is greater than one and less than the whole dataset, it is called mini-batch, and if it is one, it is called stochastic-mode.

Optimization Algorithm: Optimization Algorithm is the function chosen to be used in training to find the minimum and the maximum results during iterations. Adam, RMSProp and (Stochastic Gradient Descent) SGD variations are some of the commonly used alternatives.

Dropout: It is a technique in neural network training to reduce over fitting (learning the training data but failing on different data sets) and improve generalization by random removal of the connections of neurons in layers at a given rate.

Learning Rate: Learning Rate is a sensitive tuning parameter of optimization algorithms which can be defined as the step size while trying to find the minima in the function. Untuned learning rate may result in stucking in local minimas when too low or missing the minima point and wandering around when too high.

Pooling: The function of pooling is to reduce the size of the feature map so that we have fewer parameters in the network. There are variations like average pooling, max pooling, S3Pooling etc.

Padding: Number of pixels added to the outer frame of an image when it is being processed by the kernel of a CNN is called padding. It works by extending the area of which a CNN processes an image. Adding padding to an image processed by a CNN allows for a more accurate analysis of images by providing more space for the filter to cover in the images.

2.4. Evaluation Metrics

Semantic segmentation requires metrics to be evaluated to see how successful the task is done like every prediction task based on a learned model. There are many evaluation metrics [34][35].

Pixel Accuracy: It is probably the most basic conceptual approach. It is the percentage of pixels in the image classified correctly.

Intersection-Over-Union (IoU): It is one of the most used metrics in semantic segmentation, also known as the Jaccard Index. It is defined as the area of the intersection divided by the union area.

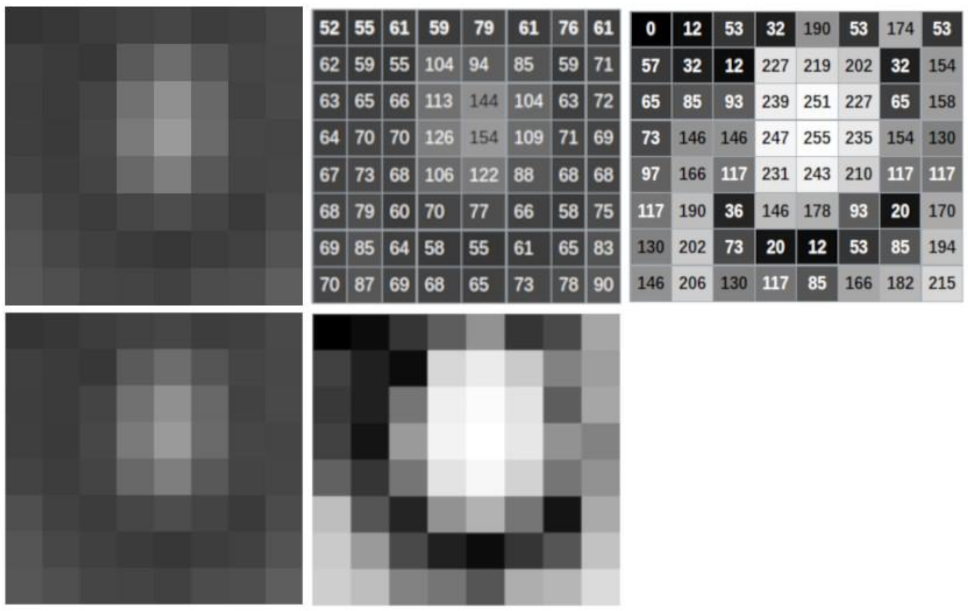
$$\frac{IoU = |\, A \cap B \,|}{|\, A \cup B \,|}$$

Dice Coefficient (Sørensen–Dice coefficient or F1 Score): The Dice coefficient can be formulated as the area of the intersection of A and B multiplied by 2, and then divided by the sum of the total area A and B have.

$$\frac{Dice = 2\ x\ |A \cap B|}{|A| + |B|}$$

2.5. Other Related Terms

Histogram Equalization (HE): It is a common technique in image processing to increase the contrast by spreading out highly populated values in histogram. Figure 2 shows a sample 8-bit grayscale 8x8 image on the left and its density map in the middle with its original values, right most shows the equalized values. Equalized histogram enhances the contrast as seen below:



52	55	61	59	79	61	76	61
62	59	55	104	94	85	59	71
63	65	66	113	144	104	63	72
64	70	70	126	154	109	71	69
67	73	68	106	122	88	68	68
68	79	60	70	77	66	58	75
69	85	64	58	55	61	65	83
70	87	69	68	65	73	78	90

0	12	53	32	190	53	174	53
57	32	12	227	219	202	32	154
65	85	93	239	251	227	65	158
73	146	146	247	255	235	154	130
97	166	117	231	243	210	117	117
117	190	36	146	178	93	20	170
130	202	73	20	12	53	85	194
146	206	130	117	85	166	182	215

Figure 2. Histogram equalization of a sample 8-bit grayscale 8x8 image, its pixel values and the image before and after the equalization [28]

Adaptive Histogram Equalization (AHE): It is a contrast improvement technique in images, using multiple histograms each corresponding to a different section of the underlying image [28].

Contrast Limited Adaptive Histogram Equalization (CLAHE): Contrast Limited AHE (CLAHE) is an enhanced version of the adaptive histogram equalization. In constant or near constant regions of an image's histogram becomes highly concentrated. This makes adaptive histogram equalization (AHE) tends to over amplify the contrast, causing noise to be amplified in these regions. CLAHE aims to solve this problem of noise amplification by limiting the contrast by regions [28].

Watershed: It is a method used for image segmentation using morphological operations like opening and dilation. It is based on finding a sure background and foreground. The remaining unknown area is "flooded" until reaching the known areas. It is especially useful for detecting touching or overlapping objects [45].

Atlas-based auto segmentation (ABAS): It is a traditional method for organ contouring. Areas like medical image segmentation where low contrast, fuzzy contours, similar intensities with adjacent objects of interest etc. It is an approach making use of an atlas (a reference image) for segmentation [27]. Several factors like the size of the dataset used to create the atlas, approaches for image registration can affect segmentation performance.

CHAPTER 3

RELATED WORK

Our proposed system combines various approaches for pixel-level classification also known as “semantic image segmentation”, which has been heavily researched in recent years. There are many research for nucleus and cell segmentation in the medical image segmentation research area. However, systems developed for sub-cellular structures and organelles are remarkably less explored. Our cascaded CNN (U-Net) model with supportive techniques aimed to be used especially for mitochondria segmentation but suitable for development to be used to identify cells and other cellular structures with additional training.

There are different approaches and structures that are being used in various research.

3.1. Classical Machine Learning Methods

Linear Regression, Support vector machine (SVM), Random Forest (RF) and Markov Random Field (MRF) are among segmentation techniques that have been studied rigorously in the past decade. We reviewed Support Vector Machines, Random Forest, Linear Regression, Markov Random Field segmentation and Atlas-based approaches which were quite popular before modern advances in deep learning methods.

Although the SVM was invented in the nineteenth century, it is still popular in data analysis and machine learning areas today. Compared to other regression algorithms, such as logistic regression and neural network, the SVM provides a cleaner and sometimes more powerful way for learning complex non-linear functions.

The SVMs are supervised machine learning techniques that make a non-probabilistic binary classifier by assigning new examples to one class or the other. More specifically, the kernel support vector machines (SVMs) are a nonlinear classifier where the representations are built from pre‑specified filters in contrast to the deep learning paradigm where the representations are directly learned from data [42].

Random Forests (RFs) are an ensemble learning method used to build predictive models by combining decisions from a sequence of base models. They use multiple learning models to obtain better predictions. A random forest model makes use of a forest of uncorrelated random decision trees to reach the best prediction result. A RF classifier uses a series of their branch-like classification trees that map the features of a sample to a class output. Uncorrelated decision trees with random feature combinations help to reduce the variance in the data to reduce the risk of overfitting.

Linear regression is one of the most common and well-known methods in machine learning. Although it is a fairly simple framework, it is still a basis for many other techniques. The model is determined by linear functions and unknown parameters of those functions estimated from the data in linear regression.

Prior to modern advances in deep learning methods, atlas-based and deformable model segmentations were also quite popular for medical images. Some atlas-based segmentation methods have more accurate segmentation results than those from deep learning-based methods (98.0 % vs. 94.0 % for mandible) [43].

Atlas-based segmentation does not belong to general machine learning algorithms but is a specific method for segmentation with high performance. In summary it can be described as follows: An atlas is a mapping A, where n-dimensional spatial coordinates to labels are L. Atlas can itself be considered as a special type of a kind of label image. An atlas is usually generated by manual segmentation, which can be expressed as a mapping, M. For image segmentation of M based on atlas, each point in an image has a corresponding equivalent in the other [43].

Markov Random Field (MRF) is another segmentation method within the classical machine learning concepts. MRF is a kind of conditional probability model, a stochastic process that calculates the probability of a pixel, based on its neighboring pixels and using the local features of the image. It is a powerful segmentation method since it connects prior contextual information and provides spatial continuity.

3.2. Other Related Segmentation Methods

Recent architectures for image segmentation most commonly use CNN to assign class labels to patches of the image. CNN was first introduced by Lecun et al. [2] and has become a dominant network architecture in computer vision and image analysis. In recent years, the supervised framework of machine learning has achieved remarkable progress with more accurate segmentation outcomes. Especially, deep convolutional neural networks (DCNNs) have reached the state-of-the-art performance for the semantic segmentation of images of several types.

Literature search reveals that various neural network architectures are used for many different organs problems and disease identification, such as lung disease, skin cancer, brain tumor, liver lesion etc. In addition to early network models like VGG, ResNet, and Inception versions, Deep Convolutional Neural Network (DCNN), Feed Forward Neural Networks (FFNNs), Region-Based Convolutional Neural Networks (RCNNs) and Projective Adversarial Network (PAN) architectures are widely researched.

Another important segmentation model is Mask Region Based Convolutional Neural Network (Mask R-CNN). Faster R-CNN is also a popular target detection framework, and Mask R-CNN extends it to an instance segmentation framework [4].

One of the most researched CNN for image segmentation in medical imaging is U-Net [1]. It is the most suitable segmentation model for medical image segmentation because of its ability to simultaneously combine low-level information which is required to improve accuracy and high-level information to extract complex features.

A typical U-Net architecture [1] is composed of u-shaped encoder and decoder layers as seen in Figure 3 below. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations explained in the legend section of the figure

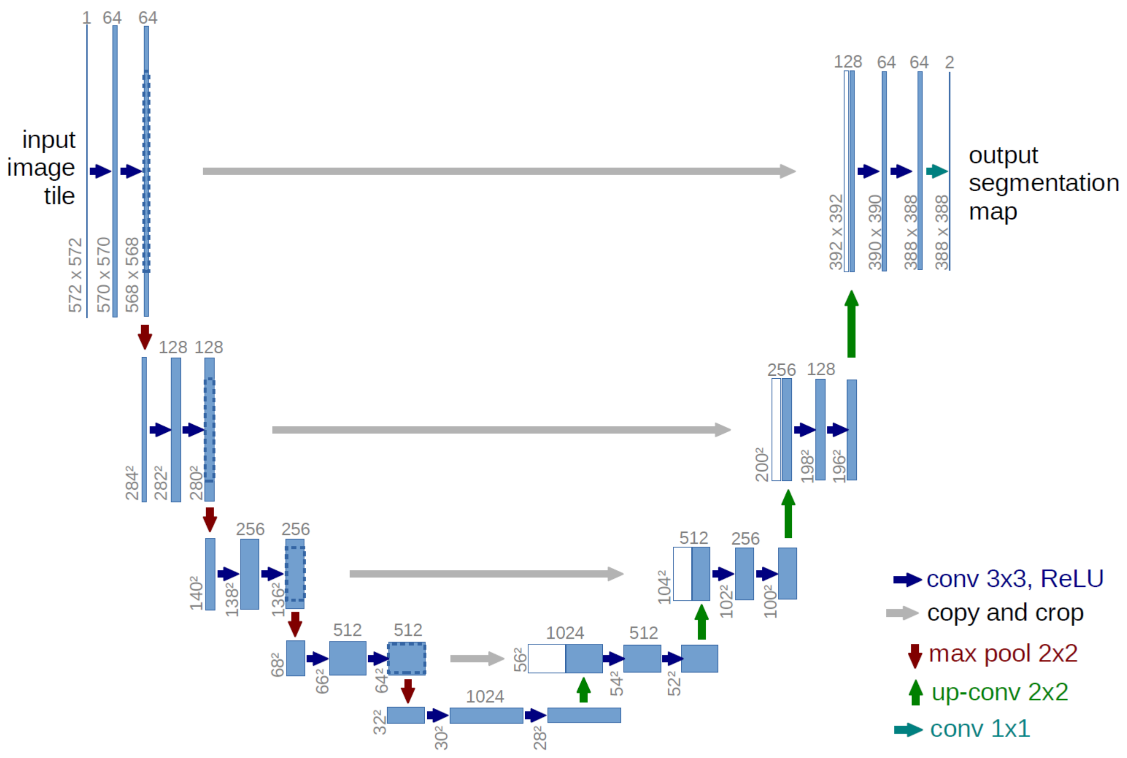


Figure 3. Encoder and decoder layers of a typical U-Net architecture are shown in the figure with the arrows representing the processes taken.

U-Net is an architecture for semantic segmentation, composed of an encoding path and a decoding path. The encoding path follows the typical architecture of a convolutional network [1]. It consists of the repeated application of two 3×3 unpadded convolutions, each followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for down-sampling. Every step in the decoding path consists of an up-sampling of the feature map followed by a 2×2 upconvolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the encoding path, and two 3×3 convolutions, each followed by a ReLU. 1×1 convolution is used to map component feature vector to the number of classes required in output at the final layer. In total the network has 23 convolutional layers [13].

Variations of U-Net are studied in many research including dual models, cascaded or 3D forms of the original model and so on. On the other hand, cascaded neural networks are used in many areas such as face detection, image dehazing, protein identification and object detection. The purpose of using two or more networks consecutively is usually separating the tasks and optimizing each model to a specific task. Region proposal and mask generation are some of the many such tasks.

One of the earliest research on automatic mitochondria segmentation was carried out in Middle East Technical University (METU) 2012 [46]. This study was for the detection and segmentation of mitochondria in Electron Microscopy (EM) images. It was not based on ML or DL but relied on approximate elliptical shape and double membrane boundary of mitochondria and the results were refined using active contours. Since the elliptical shape assumption is quite restrictive and double membrane boundaries cannot be seen in fluorescence microscopy, this study did not contain much valuable information for our work to inherit.

Another early research in this area was "Segmentation of mitochondria in electronic microscopy images using algebraic curves" [51]. In this 2013 study, curve fitting and random forest classifiers were used to segment mitochondria in EM images. This work stated that it achieved 82.49% Dice value in mouse neuropil cells dataset.

A very similar platform MitoSegNet [47] proposed in 2020. It uses a modified single U-Net model for the segmentation of mitochondria. Drop out layers removed from the U-Net model, batch normalization layers added instead. Accuracy comparisons were with three segmentation methods, namely Gaussian, Hessian, and Laplacian, ImageJ/Fiji provided. Only the Dice metric was used for evaluation, but two other morphological accuracy metrics were used as a different approach. According to this study MitoSegNet outperformed all non-deep learning segmentation methods [47].

A very new study which was published recently contains a deep learning solution for instance segmentation of mitochondria in EM images [48]. This study proposed a platform called MitoNet based on the following approach: Mitochondria are extremely pleomorphic organelles. Automatically annotating each one accurately and precisely in any 2D or volume EM image is an unsolved computational challenge [48]. This related work experimented with different deep learning models but achieved their best results with the adopted Panoptic-DeepLab [49] model which is an encoder decoder model very much like U-Net. One of the most outstanding parts of this study was that the dataset used for training was made up of approximately 1.5 million images. Different accuracy rates were achieved for different cell types reaching up to 94% and 90.2% according to Dice and IoU metrics in Fly Brain cells. However, top results in HeLa cells were 79.1% and 72.8% according to Dice and IoU metrics respectively [49].

Another study presented in 2021 proposed a contrastive learning based framework for mitochondria segmentation task in EM images [50]. This framework used a 3D U-Net with Watershed post processing for backbone segmentation to generate a mask of regions. They used a contrastive learning block to improve the cross entropy loss function which penalizes the predicted probability of all pixels without any distinction [50].

Mitometer [51] study has the most similar problem definition with our research since it works on fluorescence microscopy images to segment mitochondria in in-vivo cells. But this study is not based on ML or DL models dissimilarly. Study proposed a signature algorithm to provide a fast and unbiased segmentation claiming that the DL based methods are prone to biased results since they are trained with human annotated images.

CHAPTER 4

PROPOSED METHOD

Although there are more recent alternative architectures proposed without the need for post-processing like Region-Based Convolutional Neural Network (RCNN) [3], we found a cascaded U-Net [1] architecture more plausible since it is heavily used in medical image segmentation and successful with smaller data sets more importantly.

Our approach is made up of using two cascaded U-Net models, making use of a signature noise reduction, Contrast Limited Adaptive Histogram Equalization (CLAHE) and image size manipulation in preprocessing, Watershed in post processing to conclude. Data augmentation techniques were used to enrich the training data. Alternative pre-processing and post-processing techniques and various combinations of the CNN hyper parameters were tried to reach the presented results.

We used the classical segmentation methods in image segmentation as a baseline to assess the performance of our method in comparison with commonly used publicly available tools and techniques. Swift algorithm [30] used by OpenCV to find contours is a long established classical but widely used successful method. However, it is not very effective segmenting the close-proximity objects touching or overlapping. Cells or mitochondrial networks may be observed in such situations in their nature. Watershed algorithm comes in handy in such cases since it is substantially a better bet segmenting the objects in the image. So, we compared the results of the proposed system to the results of Suzuki and Watershed combined which yields better results rather than using either of them alone.

To evaluate the success, we used two metrics Dice and IoU together. Although both are commonly used to assess the results of object segmentation tasks in image processing, either may be better depending on the usage case and characteristics of the images and objects. Use of two evaluation metrics (Dice & IoU) together is preferred due to the fact that the Dice metric tends to measure a score in close proximity to the mean and the IoU metric tends to further penalize single instances of bad segmentation in contrast.

Although we experimented with many different approaches and techniques until reaching the proposed method, the architecture that we decided to work with has following phases and steps:

The training phase was carried out with two different datasets. The first dataset, in-vivo HeLa cells images, was used to train both U-Net models to provide the best segmentation results for cell segmentation and tracking. We aimed to construct base models with cell images since this dataset was bigger, contained annotated silver truth and ground truth images, and more importantly it was publicly available. To make the dataset bigger and to provide better generalization, we made use of data augmentation techniques in this dataset too. All images were pre-processed for noise reduction and contrast increase.

Second dataset was the mitochondria images and time series videos captured in IZTECH Nanobiolab. Our trained U-Net models were fine-tuned with this second dataset to adapt the models for mitochondria segmentation. In this episode of training, the same data augmentation techniques were used, and the same pre-processing methods were applied to the original images.

Both training episodes, first with cell images and latter with mitochondria images, experimented many times with various parameter alternative combinations to reach better results according to the evaluation metrics.

Each dataset was divided into training, validation and test groups as 20%, 20% and 60% respectively. Training and validation groups were used to construct model results. The remaining 60% was used in test phases.

During the test phase, all images were first resized to 512x512 pixels and pre-processed with the same noise reduction and histogram equalization techniques. Color images were converted to grayscale before fed into the first U-Net model responsible for segmenting foreground and background to reveal the regions of interest. Outputs of the first U-Net were processed by the second U-Net model to detect mitochondria instances. Watershed algorithm applied to the output to finalize the segmentation details. The Suzuki algorithm used to draw the borders onto both the original images. Computed the potential of each region based on brightness of image at its original form prior to any processing. An information file generated containing the region count, total region area and potential for each processed image.

Added value other than segmentation:

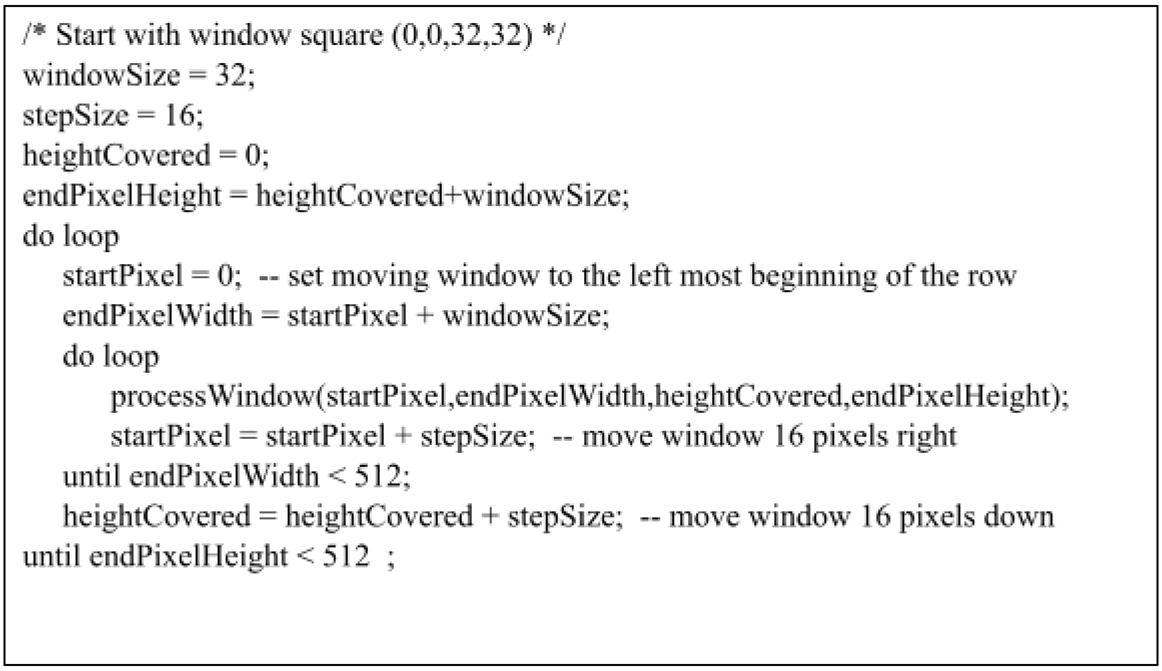
Mito ML also provides the number of objects (mitochondria) detected, total size of mitochondria network and mitochondria potential based on the brightness intensity of the segmented region.

Time series videos can automatically be split into frames and mitochondria count, size and potential changes can be tracked.

Signature Noise Reduction algorithm:

Main idea is to neutralize the pixels with values under a threshold value, assuming those are noise occurred in image capture. Chosen threshold is 1/e (1 over Euler's number) which is approximately 0.367879. Our study showed that applying this technique through a clipping window which moves around the whole image provides remarkably better results compared to applying it to the whole image at once. Our chosen window size was 32 and step size was 16 in 512x512 pixels. So, the algorithm steps were like:

Algorithm 1:



```
/* Start with window square (0,0,32,32) */
windowSize = 32;
stepSize = 16;
heightCovered = 0;
endPixelHeight = heightCovered+windowSize;
do loop
    startPixel = 0;  -- set moving window to the left most beginning of the row
    endPixelWidth = startPixel + windowSize;
    do loop
        processWindow(startPixel,endPixelWidth,heightCovered,endPixelHeight);
        startPixel = startPixel + stepSize;  -- move window 16 pixels right
    until endPixelWidth < 512;
    heightCovered = heightCovered + stepSize;  -- move window 16 pixels down
until endPixelHeight < 512  ;
```

Below figure illustrates the moving behavior of Algorithm 1 given above. Moving window process through the signature noise reduction algorithm. The 32x32 pixel window was moved with 16-pixel steps to the right and then 16 pixels down iteratively until all the image regions are covered to treat each region adaptively.

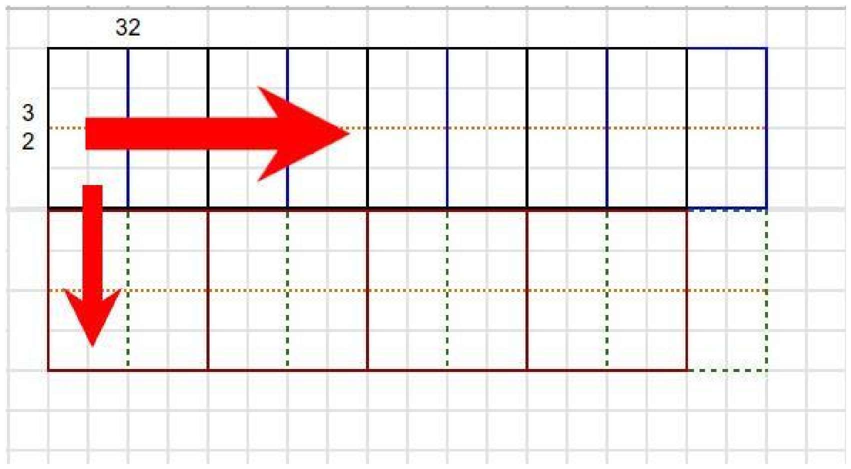


Figure 4. Movement of the chosen size (32x32 pixel) processing window in algorithm 1.

The function called "processWindow" in algorithm1 executes two steps for given window.

processWindow:

```
find the brightest pixel (which has the maximum value) in window
reset all pixels with value under ( max_value / e ) to zero
```

Since the algorithm assumes that the region of interests have bright areas on the dark background. On a light background and dark region of interest images, all values were reversed by subtracting 254 from all values and taking absolute values without sign before processing starts and after it finishes. Since the grayscale images have 0-254 values in each pixel, this operation made black pixels white and vice versa first, after processing completed reverse operation restored the original image only without cleansed noise.

This operation itself reduced the noise, increased contrast and sharpened the edges. It increased the success rate of the detection and segmentation with a small sacrifice of precision. Except for pre-processing there was a need to enrich the limited size dataset to be able to do proper training. We overcame this hurdle by data augmentation and transfer learning. We used HeLa cell images to train our models and use them as a base for secondary training with a limited number of mitochondria images.

It is possible to use pretrained U-Net architecture on single cell segmentation by adapting to new tasks based on a small set of annotated samples [26]. This approach proposes that it is possible to train U-Net models with annotated cell images and then use them for new tasks by fine-tuning with a few annotated images. We tried to test this approach by training models first using HeLa cell timeseries images and then various mitochondria networks of single and multiple cell images. Both episodes of training required the trial of different loss functions to be able to get the best convergence.

There are many different loss function algorithms used in deep learning. A loss function is responsible for comparing the predicted and targeted output values to tell how successfully the model fits the data. In the training phase, the aim is to minimize this loss to achieve the best results in modelling. We tried a set of various loss functions to assess the effects. At the end, results of combinations from three of them, namely softmax_cross_entropy, sigmoid_focal_crossentropy and categorical_crossentropy, were comparable. Detailed results are presented in Chapter 5.

Since the categorical focal loss function tends to provide better predictions without the need for remarkably high certainty, it was an advantage to be chosen (for example instead of binary cross-entropy) due to the nature of images with uncertain forms and locations of the cells. The second loss function we alternatively worked with was the softmax cross-entropy, since it is generally the primary loss function chosen to train deep neural networks.

$$Focal\,Loss = \sum_{i=1}^{i=n} \left(i - p_i\right)\gamma . \left(p_i\right)$$

Binary cross entropy compares each of the predicted classes to the actual class if it matches, resulting in 0 for false predictions and 1 for the correct ones. It then calculates the score that penalizes the probabilities based on the distance from the expected value. The distance here shows how far or close the predicted value is from the actual value.

$$Binary\,cross\,entropy = -\left(y.\log log\,(p) + (1 - y).\log log\,(1 - p)\right)$$

$$Softcross\,entropy = -\sum_{c=1}^{M} y_{o,c}.\ \log log\left(p_{o,c}\right)$$

In the training phase of the neural network model, the loss function is tried to be minimized by modifying the weights in each epoch. The optimizer function reduces the overall loss calculated by the chosen loss function and improves accuracy by modifying the attributes of the neural network, like weights and sometimes the learning rate. Adam, which is presented by Djederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 paper [14] is a popular optimizer function in deep learning training.

In our research, we used Adam optimizer with the learning rate 0.0001. This learning rate was chosen after the trials with varying values between 0.00001 and 0.001. Networks were trained for 100 epochs with a batch size of 16.

Drop out is a powerful tool to prevent over-fitting. We tested different “Drop Out” rates for different layers of the two U-Net models during hyper parameter tuning. As the result of these tests, we chose 0.1 value for the selected layers of the first model but decided to use 0 for the second U-Net model.

As a result of parameter tuning with many variations, a mean accuracy success rate achieved up to 93.4% and 89.6 according to Dice and IoU metrics, respectively.

These results were calculated based on the manually segmented ground truth images vs. machine-generated segmentation comparisons. Below image in Figure 5, is an example of a snapshot from a 3D time series video, original image frame on the left, and auto-segmented output image on the right. The original image visible on the left is the mitochondrial networks of a few cells with different potentials. It is fed to the second U-Net model by masking the ROIs (regions of interest) detected by the first U-Net. The high potential regions segmented in the output of the second model (consisting of pixels whose count value is above the threshold value) were determined by the Watershed algorithm and their borders were drawn. In the rendered image on the right, it is seen that all high potential regions have been identified and marked within the red border lines.

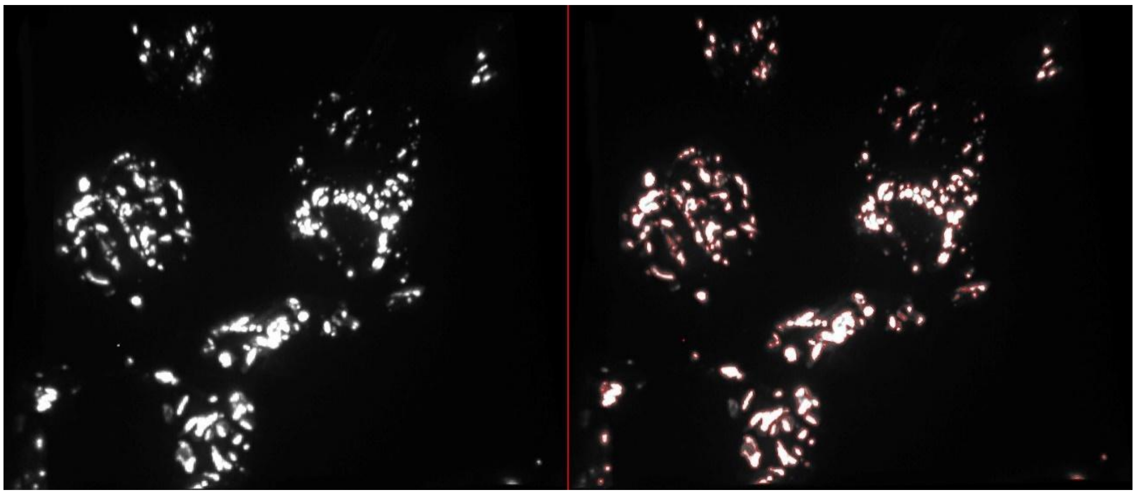


Figure 5. Mitochondrial networks of a few in-vivo cells, original form on the left and MitoML output on the right. Image is a snapshot of 3D video captured in IZTECH Nanobiolab.

Below images in Figure 6, show an input image with a completely different structure. Bright background and darker objects with different shapes were not familiar for the trained model. The model was only able to detect and segment light structures on darker background initially due to the previous training data. After making some adjustments and fine tuning, opposite structured images were able to be processed like above. Original on the left and the segmented on the right

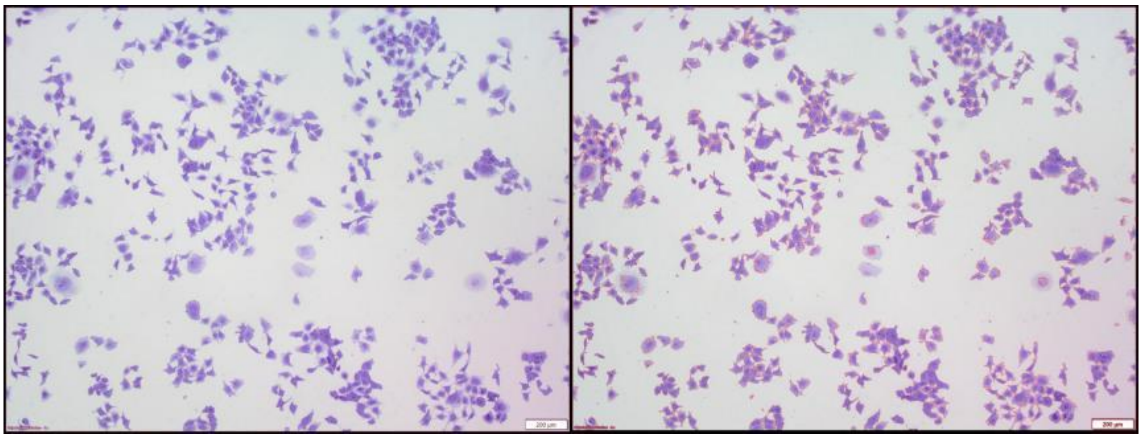


Figure 6. Original on the left, captured as a still tiff image in IZTECH Nanobiolab, output of our system is on the right.

An example from the HeLa cell images used in initial training is presented below in Figure 7. Despite its lack of contrast and ambiguous structure and remarkably close locations of objects of interest, MitoML successfully segmented, marked the center and drew contours of the cells in most of the cases.

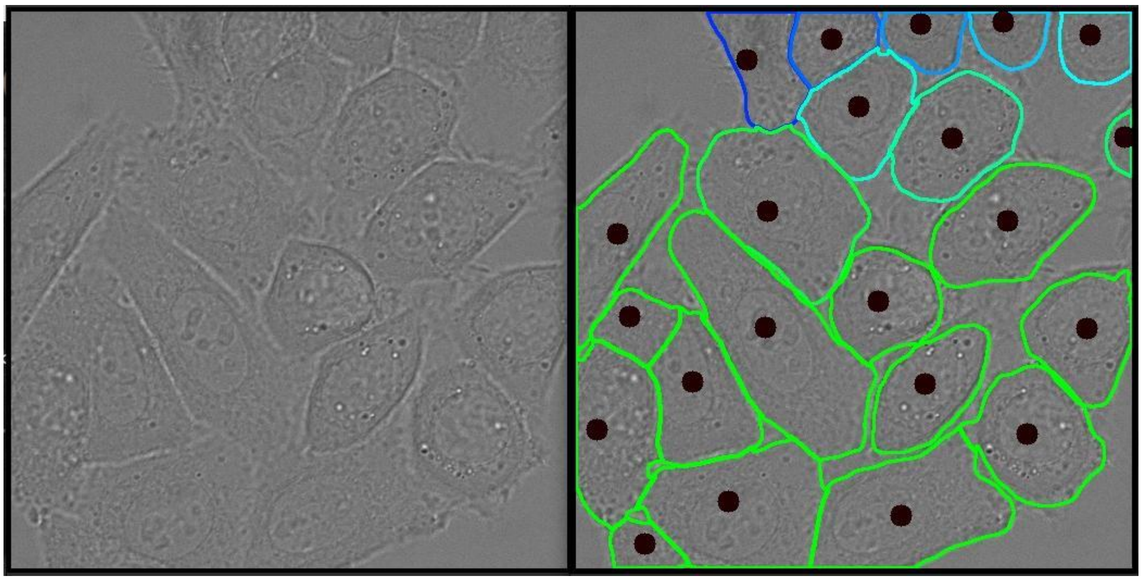


Figure 7. Original microscopic image on the left and the output of the platform is on the right. Raw images are gathered from public datasets.

Two MitoML outputs are given below in Figure 8, to demonstrate an intermediate step achieved in training with HeLa cells. Two snapshots from a time series video of in-vivo cells, hyper parameter tuning allowed us to determine even the smallest indentations and track relocations and divisions. On the other hand, this extreme sensitivity came with the false detection of non-existing objects like the one seen as number 6 on the right image.

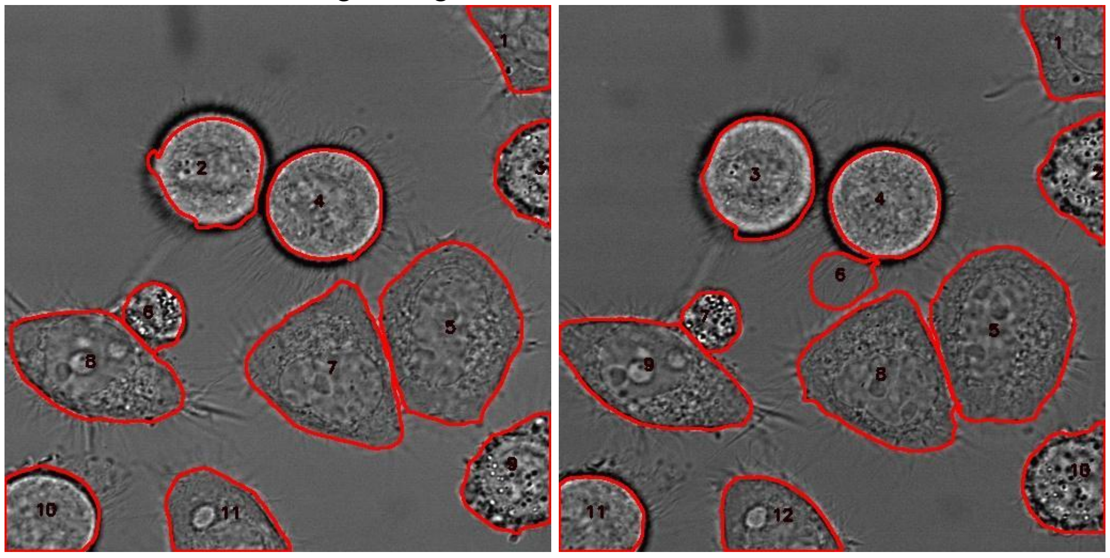
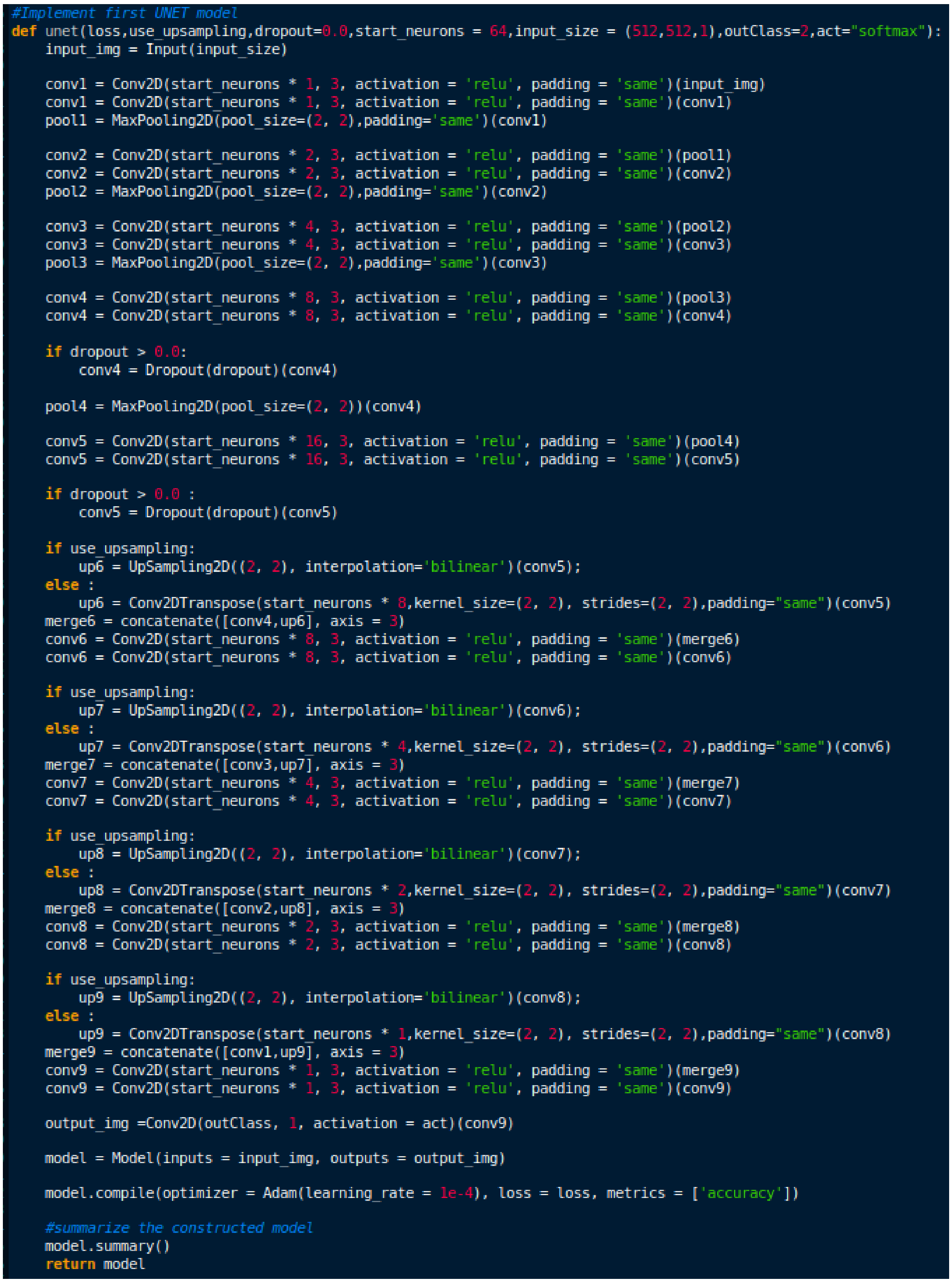


Figure 8. Two frames of an in-vivo HeLa cell video processed by MitoML.

Below code snippet shown in Figure 9, is one version of the first U-Net (the one used for region proposal) implementation in our code. As well as the static convolution layers and pooling, up sampling and dropout applications were bound to the parameters. Optimizer and activation functions, learning rate, loss, input size and neurons of all layers were tested with different combinations to reach the optimal results.



```python
#Implement first UNET model
def unet(loss,use_upsampling,dropout=0.0,start_neurons = 64,input_size = (512,512,1),outClass=2,act="softmax"):
    input_img = Input(input_size)

    conv1 = Conv2D(start_neurons * 1, 3, activation = 'relu', padding = 'same')(input_img)
    conv1 = Conv2D(start_neurons * 1, 3, activation = 'relu', padding = 'same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2),padding='same')(conv1)

    conv2 = Conv2D(start_neurons * 2, 3, activation = 'relu', padding = 'same')(pool1)
    conv2 = Conv2D(start_neurons * 2, 3, activation = 'relu', padding = 'same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2),padding='same')(conv2)

    conv3 = Conv2D(start_neurons * 4, 3, activation = 'relu', padding = 'same')(pool2)
    conv3 = Conv2D(start_neurons * 4, 3, activation = 'relu', padding = 'same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2),padding='same')(conv3)

    conv4 = Conv2D(start_neurons * 8, 3, activation = 'relu', padding = 'same')(pool3)
    conv4 = Conv2D(start_neurons * 8, 3, activation = 'relu', padding = 'same')(conv4)

    if dropout > 0.0:
        conv4 = Dropout(dropout)(conv4)

    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = Conv2D(start_neurons * 16, 3, activation = 'relu', padding = 'same')(pool4)
    conv5 = Conv2D(start_neurons * 16, 3, activation = 'relu', padding = 'same')(conv5)

    if dropout > 0.0 :
        conv5 = Dropout(dropout)(conv5)

    if use_upsampling:
        up6 = UpSampling2D((2, 2), interpolation='bilinear')(conv5);
    else :
        up6 = Conv2DTranspose(start_neurons * 8,kernel_size=(2, 2), strides=(2, 2),padding="same")(conv5)
    merge6 = concatenate([conv4,up6], axis = 3)
    conv6 = Conv2D(start_neurons * 8, 3, activation = 'relu', padding = 'same')(merge6)
    conv6 = Conv2D(start_neurons * 8, 3, activation = 'relu', padding = 'same')(conv6)

    if use_upsampling:
        up7 = UpSampling2D((2, 2), interpolation='bilinear')(conv6);
    else :
        up7 = Conv2DTranspose(start_neurons * 4,kernel_size=(2, 2), strides=(2, 2),padding="same")(conv6)
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(start_neurons * 4, 3, activation = 'relu', padding = 'same')(merge7)
    conv7 = Conv2D(start_neurons * 4, 3, activation = 'relu', padding = 'same')(conv7)

    if use_upsampling:
        up8 = UpSampling2D((2, 2), interpolation='bilinear')(conv7);
    else :
        up8 = Conv2DTranspose(start_neurons * 2,kernel_size=(2, 2), strides=(2, 2),padding="same")(conv7)
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(start_neurons * 2, 3, activation = 'relu', padding = 'same')(merge8)
    conv8 = Conv2D(start_neurons * 2, 3, activation = 'relu', padding = 'same')(conv8)

    if use_upsampling:
        up9 = UpSampling2D((2, 2), interpolation='bilinear')(conv8);
    else :
        up9 = Conv2DTranspose(start_neurons * 1,kernel_size=(2, 2), strides=(2, 2),padding="same")(conv8)
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(start_neurons * 1, 3, activation = 'relu', padding = 'same')(merge9)
    conv9 = Conv2D(start_neurons * 1, 3, activation = 'relu', padding = 'same')(conv9)

    output_img =Conv2D(outClass, 1, activation = act)(conv9)

    model = Model(inputs = input_img, outputs = output_img)

    model.compile(optimizer = Adam(learning_rate = 1e-4), loss = loss, metrics = ['accuracy'])

    #summarize the constructed model
    model.summary()
    return model
```

Figure 9. U-Net implementation snippet from MitoML source code.

4.1 Data Collection

The author of this thesis claims no right of the data used and acknowledges that they do not have any legal claim to this data. Ownership of the data belongs to the original owners after the completion of this thesis. Dataset used in this work were gathered from public sources including publicly available live HeLa cell images and the microscopy images acquired by the researchers in IZTECH Nanobiolab.

During training, validation and test phases basically two data sets are used. The first one is the HeLa cell images from the cell tracking challenge which publicly publishes the data set for scientific usage purposes at website http://celltrackingchallenge.net/datasets. These contain grayscale in-vivo HeLa cell snapshots taken with Zeiss LSM 510 Meta during 10 min time.

The latter, cell and mitochondria images both grayscale and color taken in Nanobiolab of IZTECH using Olympus SC50 under various conditions. Some images were direct output of microscopy in raw tiff format with 512x512 pixels and 2560x1920 pixels and some are extracted from timelapse videos recorded by splitting 0.4 second frames of the video and adjusting the changing brightness and the contrast.

4.2 Data Augmentation and Transformation

The performance and the accuracy of deep learning models depend on the dataset size and quality. However, it is hard to acquire such a number of medical images. Therefore, we made use of two enhancement methods; former is training the model on cell segmentation and fine tune it with mitochondria images in hand and the latter is enlarging the dataset and increasing the variations by using data augmentation techniques like zooming, rotating in x, y, z axis, mirroring, shearing and changing brightness.

Histogram equalization techniques (specifically Adaptive Histogram Equalization (AHE) and Contrast Limited Adaptive Histogram Equalization (CLAHE)) were tried in the pre-processing phase. Although histogram equalization is a common and useful technique, a more advanced approach was required in challenging images like microscopic images of cells and sub-cellular structures. AHE is, on the other hand, differentiates from the ordinary histogram equalization in the way that it computes several histograms, each of which corresponds to a different section of the image, and uses these histograms to adjust the lightness effectively. We used the enhanced form of AHE alongside our signature noise reduction algorithm.

Our custom noise reduction method was used through the elimination of the low pixels with the brightness lower than 1/e of the max value of each image, making grayscale pixels normalized between 0-254 by dividing and reshaping the arrays that form the image. This method makes it substantially easier to differentiate between background pixels and cell pixels. Finally, all the arrays were resized to 512x512 to preserve the original image format.

Both silver truth and ground truth annotations are used for training, test, and validation. Ground truth images are the ones segmented and annotated manually by lab researchers whereas silver truth ones are the publicly available outputs of open cell segmentation competitions like cell tracking challenge network, and handpicked successful segmentation results of conventional algorithms like Watershed and Suzuki algorithms, which are also used by OpenCV finding contours and segmentation.

CHAPTER 5

EVALUATION

Experiments, results, discussion, and threats to validity are explained in this section.

5.1. Experiments and Results

Computer Hardware specifications are listed in Table 1. Same hardware used for data augmentation, image processing, training, validation and test processes.

Table 1. Computer Hardware Specifications

CPU	Intel® Coffee Lake Core™ i7-9750H 12MB L3; 2.6GHz > 4.5GHz
GPU	nVIDIA® GeForce® GTX1050 DX12 3GB GDDR5
RAM	16GB DDR4L 1.2V 2666MHz SODIMM
Disk	256GB SAMSUNG PM981a M.2 SSD PCIe 3.0 x4

Software platforms and tools are given in Table 2.

Table 2. Software Platform and Tools

Ubuntu 20.04	Linux Operating System
Python 3.8	Programming Language all implementations are done
PyCharm IDE	Editor and Integrated Development Environment
CUDA 11	CUDA is a GPU computing SDK for nVidia graphic cards
Tensorflow	Open-source ML platform provided by Google
Keras	API library used as an interface for Tensorflow
NumPy	Mathematical functions library for scientific computing
OpenCV	Open-source computer vision library used for image processing
Sckit-Image	Open-source image processing library

Developing a machine learning platform requires a large amount of software. It was our purpose to develop this platform with open-source software. The software environment used was Ubuntu Linux 20.04, Python 3.8, PyCharm IDE, Tensorflow, Keras, NumPy, OpenCV and Sckit-Image. Python 3.8 and PyCharm IDE were chosen as the primary development platform. Tensorflow and Keras libraries have been used to perform model development, training, validation, testing and other deep learning tasks.

The most important criteria for choosing Tensorflow can be listed as that it is widely used in deep-learning processes and has capabilities suitable for this study's goals. It was also preferable that it was quite easy to find examples and research on machine learning and deep learning applied using Tensorflow. Keras was used since it provides easy access to Tensorflow features in Python, enabling better and more efficient use of features. The Sckit-Image and OpenCV image processing libraries were other tools used for image processing tasks, including pre-processing and post-processing of images.

A classical approach based on the Suzuki and Watershed algorithm used as a baseline for comparison. To research the best possible outcomes many alternative configurations of the proposed model are tested. Hyper parameters, optimization and loss functions, and pre-processing methods are tested against each other in various combinations. The results of best performing four configurations and the baseline configuration with Suzuki and Watershed are listed as follows:

Table 3. Configuration and results comparison

Classical Methods	**Preprocessing:**	Noise Reduction + CLAHE
Suzuki-Watershed	**R1mean (IoU)**	73.50%
	R2mean (Dice)	86.50%
Config 1:	**Preprocessing:**	Adaptive Histogram Equalization
	Loss:	softmax_cross_entropy - sigmoid_focal_crossentropy
	R1mean (IoU)	74.10%
	R2mean (Dice)	88.40%
Config 2:	**Preprocessing:**	Noise Reduction + AHE
	Loss:	categorical_crossentropy
	R1mean (IoU)	88.20%
	R2mean (Dice)	92.00%
Config 3:	**Preprocessing:**	Noise Reduction + CLAHE
	Loss:	softmax_cross_entropy - categorical_crossentropy
	R1mean (IoU)	88.40%
	R2mean (Dice)	92.10%

cont. on the next page

cont. of Table 3

Config 4:	**Preprocessing:**	Noise Reduction + CLAHE
	Loss:	categorical_crossentropy - softmax_cross_entropy
	R1mean (IoU)	89.60%
	R2mean (Dice)	93.40%

All other combinations were ignored due to incomparably bad results. Following observations directed the selection of best performing configurations:

- All trials without up sampling yielded worse results compared to models using up sampling.
- DropOut ratio = 0.1 yielded best results compared to alternative values tried 0 - 0.05 - 0.15 - 0.2.
- **Adam** outperformed **RMSProp** and **SGD** as the optimization function in all combinations.
- Learning Rate 0.0001 (1e-4) was the best in terms of convergence and not overfitting among other values 0.001 (1e-3) and 0.00001 (1e-5).
- All combinations for 1st and 2nd U-Net models with loss functions **binary_cross_entropy** - **categorical_cross_entropy** - **softmax_cross_entropy** - **sigmoid_cross_entropy.**
- No preprocessing and preprocessing with only histogram equalization configurations yielded worse results.

Individual results of each image in the test dataset can be found in APPENDIX A section for each configuration.

MitoML outputs segmented region count, region area sizes and total potential as an additional info for each input image. An output file generated for each individual input image containing information like below:

Region Count: 23 - Region Areas (in Pixels): [203, 6, 115, 85, 50, 22, 285, 19, 25, 19, 11, 63, 106, 110, 175, 120, 213, 54, 75, 183, 208, 130, 36] - Region Potential: 216,314 (Region Potential value is the sum of light intensity the pixel values between 0 and 254.)

As shown in Table 3, configuration 4 yielded the best results with mean accuracy of 89.60% and 93.40% according to IoU and Dice metrics, respectively. In this configuration, we used noise reduction and CLAHE in the pre-processing phase. Categorical cross entropy is used as the loss function in the first U-Net model. On the other hand, softmax cross entropy was chosen for the second U-Net model since it yielded better results. We used 0.1 as the drop out ratio and Adam optimizer with learning rate 1-e4 for this configuration just like all other alternatives.

5.2. Discussion

In this study, the principal aim was to develop a platform for lab researchers to help them easily quantify and analyze effects of their test designs by measuring the status of mitochondrial networks. This process involves capturing images or videos to be split into time series frames later. In these captured images, segmenting and labeling the mitochondria are done manually. Alternatively, sometimes this process may be semi-automated by using conventional image processing tools. As the last step, the regions are counted and region potentials are measured based on pixel brightness values using image processing tools. When a video is in question, calculating the effects and change in time requires iteration of these steps for each video frame and comparison of individual outcomes too. Obviously, this procedure requires trained eyes and a huge amount of time. Moreover, it is always error prone and human dependent in any case even with expensive proprietary tools.

There are many academic and commercial work in progress to ease the process and increase the quality of the output in processing of microscopic images, segmentation of objects in all kinds of medical images including, x-ray images, mammography, MRI, tissues, cellular and even sub-cellular images. On the other hand, commercial tools and academic research mostly focuses on cell segmentation, cell tracking and only a small portion is on nucleus segmentation, but it is not common to find intelligent systems trained for sub-cellular structures.

Apart from other objects, mitochondrial networks may have many various forms, may change structure depending on the cell state or external environment. As well as their variable network structure, mitochondria may lose their potential and fade away at different speeds during test leaving varying footprints in time series images or videos. Different approaches in microscopy, using various hardware, lenses, dyes and lasers or lights to illuminate objects in interest leads to the need for an adaptive system that can carry out the same task for all these conditions and their different output images.

Specific neural network structures are being trained and used for medical image segmentation and interpretation but in general all needs a remarkably high number of training images with all situations to train so that the resulting model can successfully identify all possible test images later. Since this process, for almost all models, heavily depends on supervised learning, it requires manual data processing to provide this number of images for training which is a real big deal. This was a major problem for this study too. Using GANs to generate additional data to be used in training to enrich the dataset was one of the ways tried but it also requires remarkable time and effort to design, implement and run such a system. Besides, results of such a system must be validated before being used as an input for our original study, which is another line of work and research itself.

Using data augmentation techniques like we did, is quite common in such cases to enrich the datasets, generating possible different alternatives that may be encountered during real life situations. They usually provide performance at better intercepting the same objects and structures in different situations rather than identifying different objects or morphologies of them. So, using data augmentation techniques like zooming, stretching, rotating and changing contrast, brightness etc. has provided some variance. So related works usually made use of similar techniques to enrich their limited data similarly. But it is not enough to replace the need for bigger datasets naturally.

Besides using data augmentation techniques, making use of previously trained neural networks on cell segmentation and fine tuning the models to perform better for mitochondrial networks helped achieving better results with fewer data. This approach was also included in our work to achieve better results with less data. One of the reasons we chose U-Net architecture was its known success with fewer images in training too. There is more research on “few shot learning” due to this widespread problem but it is neither studied nor researched in this scope.

Since this area is rather less explored and more challenging, it required searching for better ways to achieve beneficial results and testing different approaches. We achieved the presented results, after testing many structures with many configuration parameter combinations. At the end of this research, using two U-net architecture cascaded to each other with separate roles so with different optimum hyper parameters, led to better segmentation results compared to publicly available common tools and techniques. Pre-processing steps tried and chosen as effective played a significant role in this success. Without proper noise reduction and contrast adjustment techniques applied the results were not near these levels.

Since the neural networks have many hyper parameters, affecting the behavior in training and evaluation radically, selection of optimization algorithms, loss functions, learning rate, momentum, decay, drop out, up sampling, pooling, padding, batch size, epoch and all the combinations to find the best scenario was a tiresome task took considerable time and effort.

As a part of this work, a post phase implementation was carried out to make the platform more beneficial for the potential users by helping them to analyze the results of their tests. Calculating and presenting the region counts, borders and potentials of each region was added to the platform even though these are not a part of deep learning research but have a vital role making the research output practically usable.

When we inspected the results, we saw that the accuracy loss is mainly caused by false negatives. There are almost no false positives at all. Mitochondria instances with low potential values may be missed by the system when a very high potential mitochondrial area is in close proximity. These results of the system seems to be improved more by finding a solution to this accuracy loss.

5.3. Threats to Validity

Since the annotated data was limited in variety, we used silver truth images in both training and validation. The rest of the training was based on self-supervised approaches. Given that neural networks require rich datasets; this shortcoming may pose a threat to the data reliability of the results.

On the other hand, ground truth images, the most trustworthy base of CNN's training, eventually an output of some human workers. Since we are not certain about the ability of these people, it may be a risk parameter for validity if they are not dexterous at manually annotating mitochondria in cell images. Considering there may be a few hundred to a few thousand mitochondria in a single cell, it is obviously an error prone task for any human.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This thesis focused on analysis of mitochondria in live cells with the help of machine learning. We scrutinized similar attempts, related work, designed a scaffold structure composed of two U-net CNNs, trained and tested with various configurations until reaching the success level that makes the platform usable in daily life.

Separately configured two U-net networks perform very well when used in a cascaded form for region proposal and segmentation supported by CLAHE pre-processing and watershed post-processing methods. Using pre-trained models, making use of transfer learning and reinforcement learning helped to reach achieved results with considerably small data set in hand. In any case, limitations on the variety of data that can be acquired and the power of the processing hardware platform restricted the tried parameter and model structure alternatives.

These results can be improved by repeating the work with larger and richer data sets and a more powerful hardware platform in relation to this within the extended period. We strongly believe that success metrics can be increased with better hyper parameter tuning with larger and richer datasets. Instead of starting with cell tracking models and evolving them to be used on mitochondria network detection, training the models from the scratch with directly mitochondria images both 2D and 3D may yield much better results for the purpose of this thesis.

To improve the achievement of this study mode, it would be also a good idea to train our CNN structure with much more detailed but in vitro EM images hoping to teach the neural networks better understanding of the cell organelles.

It is also possible to conduct extensive tests on in vivo cells under controlled conditions to monitor the reactions of healthy and ill cells. Annotate certain reactions to let the platform learn how to identify cell types by monitoring the mitochondria changes. This way MitoML platform can be enhanced to be used for diagnosis purposes.

Initial purpose and the scope of this thesis is to devise and develop a platform to identify segment and quantify mitochondria in live cells, but it can be improved to be used to work on tissue or other biomedical images.

REFERENCES

1. Olaf Ronneberger, Philipp Fischer, Thomas Brox U-Net: Convolutional Networks for Biomedical Image Segmentation 2015 https://arxiv.org/abs/1505.04597v1

2. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015; https://doi.org/10.1038/nature14539

3. Girshick, Ross & Donahue, Jeff & Darrell, Trevor & Malik, Jitendra Region-Based Convolutional Networks for Accurate Object Detection and Segmentation 2015 doi: 10.1109/TPAMI.2015.2437384

4. Qing et al. Deep convolutional neural network based medical image classification for disease diagnosis 2019 Journal of big data doi: 10.1186/s40537-019-0276-2

5. Ilonen, J., Kamarainen, JK. & Lampinen, J. Differential Evolution Training Algorithm for Feed-Forward Neural Networks Neural Processing Letters 17, 93–105 (2003). doi: 10.1023/A:1022995128597

6. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989, 1, 541–551

7. Afia Zafar et.al A Comparison of Pooling Methods for Convolutional Neural Networks Appl. Sci. 2022, 12, 8643. doi: 10.3390/app12178643

8. Liang-Chieh Chen et. al Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs https://arxiv.org/abs/1412.7062v4

9. Traore, B.B.; Kamsu-Foguem, B.; Tangara F. Deep convolutional neural network for image recognition Ecol. Inform. 2018, 48, 257–268

10. Islam, M.S.; Foysal, F.A.; Neehal, N.; Karim, E.; Hossain, S.A. A CNN based classification approach for recognizing traditional bengali games InceptB Procedia Comput. Sci. 2018, 143, 595–602

11. Khosravan, N.; Mortazi, A.; Wallace, M.; Bagci, U. Pan Projective adversarial network for medical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Shenzhen, China, 13–18 October 2019; pp. 68–76.

12. D.R.Sarvamangala, Raghavendra V. Kulkarni Convolutional neural networks in medical image understanding doi: 10.1007/s12065-020-00540-3

13. Li, C., Fan, Y. & Cai, X. PyConvU-Net: a lightweight and multiscale network for biomedical image segmentation 2021 doi: 10.1186/s12859-020-03943-2

14. Diederik P. Kingma, Jimmy Ba Adam: A Method for Stochastic Optimization doi: 10.48550/arXiv.1412.6980

15. Mouton, Coenraad; Myburgh, Johannes C.; Davel, Marelie H. Gerber, Aurona (ed.). Stride and Translation Invariance in CNNs 2020. Springer International Publishing. 1342: 267–281. arXiv:2103.10097. doi:10.1007/978-3-030-66151-9_17

16. Graves, Alex; Liwicki, Marcus; Fernandez, Santiago; Bertolami, Roman; Bunke, Horst; Schmidhuber, Jürgen 2009. "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition" doi:10.1109/tpami.2008.137. PMID 19299860. S2CID 14635907

17. Sak, Haşim; Senior, Andrew; Beaufays, Françoise Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling 2014 doi: 10.48550/arXiv.1402.1128

18. Li, Xiangang; Wu, Xihong Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. 2014 arXiv:1410.4281 [cs.CL].

19. Kramer, Mark A. Nonlinear principal component analysis using auto associative neural networks 1991. AIChE Journal. 37 (2): 233–243. doi:10.1002/aic.690370209.

20. Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua. Generative Adversarial Nets. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680

21. Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi. Improved Techniques for Training GANs. 2016 arXiv:1606.03498 [cs.LG].

22. Isola, Phillip; Zhu, Jun-Yan; Zhou, Tinghui; Efros, Alexei Image-to-Image Translation with Conditional Adversarial Nets 2017. Computer Vision and Pattern Recognition.

23. Ho, Jonathon; Ermon, Stefano (2016). "Generative Adversarial Imitation Learning". Advances in Neural Information Processing Systems. 29: 4565–4573. arXiv:1606.03476. Bibcode:2016arXiv160603476H.

24. Cortes, Corinna; Vapnik, Vladimir Support-vector networks 2017. Machine Learning. 20 (3): 273–297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018. S2CID 206787478.

25. Zoltan Kato, Markov Random Fields in Image Segmentation, 2012. doi: 10.1561/2000000035

26. Al-kofahi, Yosuef & Zaltsman, Alla & Graves, Robert & Marshall, Will & Rusu, Mirabela. (2018). A deep learning-based algorithm for 2-D cell segmentation in microscopy images. BMC Bioinformatics. 19. 10.1186/s12859-018-2375-z.

27. Bach Cuadra, M., Duay, V., Thiran, JP. (2015). Atlas-based Segmentation. In: Paragios, N., Duncan, J., Ayache, N. (eds) Handbook of Biomedical Imaging. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-09749-7_12

28. S. M. Pizer, E. P. Amburn, J. D. Austin, et al.: Adaptive Histogram Equalization and Its Variations. Computer Vision, Graphics, and Image Processing 39 (1987) 355-368

29. Gonzalez, Rafael C. (2018). Digital image processing. Richard E. Woods (4th ed.). New York, NY: Pearson. pp. 138–140. ISBN 978-1-292-22304-9. OCLC 991765590

30. Satoshi Suzuki, KeiichiA be, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing, 1985, Pages 32-46, https://doi.org/10.1016/0734-189X(85)90016-7

31. Leslie N. Smith, A disciplined approach to neural network hyper-parameters 2018 doi: 10.48550/arXiv.1803.09820

32. Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning, MIT press Cambridge, 2016.

33. Genevieve B Orr and Klaus-Robert Müller. Neural networks: tricks of the trade. Springer, 2003.

34. Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena Martinez V, Garcia-Rodriguez J. A review on deep learning techniques applied to semantic segmentation. arXiv website. arxiv.org/abs/1704.06857

35. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF, eds. Medical image computing and computer-assisted intervention: MICCAI 2015. Germany. Springer, 2015:234–241

36. Erickson BJ, Korfiatis P, Akkus Z, Kline TL. Machine learning for medical imaging. RadioGraphics 2017; doi: 10.1148/rg.2017160130

37. Flach P. Machine learning: the art and science of algorithms that make sense of data. Cambridge, England: Cambridge University Press, 2012.

38. Sebastian Ruder An overview of gradient descent optimization algorithms (2016) arXiv:1609.04747v2 [cs.LG]

39. Y. LeCun, Y. Bengio, and T. B. Laboratories, Convolutional Networks for Images, Speech, and Time-Series, The handbook of brain theory and neural networks. editor / M.A. Arbib. MIT Press, 1995, pp. 138-142

40. Xiaojin Zhu, Andrew B. Goldberg, Introduction to Semi-Supervised Learning Synthesis Lectures on Artificial Intelligence and Machine Learning. University of Wisconsin, Madison. 2009 doi: 10.2200/S00196ED1V01Y200906AIM006

41. Lisa Torrey and Jude Shavlik. University of Wisconsin, Madison WI, USA "Transfer learning." Handbook of research on Machine learning applications and trends: algorithms, methods, and techniques 2010

42. J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: NIPS 2011, 2011, pp. 2546–2554.

43. Hyunseok Seo, Lei Xing et. al Stanford University, Stanford, CA, 94305, USA. Machine Learning Techniques for Biomedical Image Segmentation: An Overview

of Technical Aspects and Introduction to State‐of‐Art Applications doi: 10.1002/mp.13649

44. R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.

45. Meyer and S. Beucher, Morphological segmentation, J.Vis. Commun. Image Represent., vol. 1, pp. 21–46, Sept.1990.

46. Mumcuoğlu, Ü. E., Tasel, S. F., Perkins, G., Martone, M. E., & Gurcan, M. N. (2012). Computerized detection and segmentation of mitochondria on electron microscope images. JOURNAL OF MICROSCOPY, 248–265. https://hdl.handle.net/11511/32292

47. Christian A. Fischer, Laura Besora-Casals, Stéphane G. Rolland, Simon Haeussler, Kritarth Singh, Michael Duchen, Barbara Conradt, Carsten Marr, MitoSegNet: Easy-to-use Deep Learning Segmentation for Analyzing Mitochondrial Morphology, iScience,Volume 23, Issue 10,2020, doi : 10.1016/j.isci.2020.101601.

48. Ryan Conrad, Kedar Narayan Instance segmentation of mitochondria in electron microscopy images with a generalist deep learning model, 2022.Cold Spring Harbor Laboratory doi: 10.1101/2022.03.17.484806

49. Cheng, B. et al. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2020.

50. Zhili Li, Xuejin Chen, Jie Zhao and Zhiwei Xiong Contrastive Learning for Mitochondria Segmentation arXiv:2109.12363v1 [cs.CV] 25 Sep 2021

51. Seyedhosseini M, Ellisman MH, Tasdizen T. SEGMENTATION OF MITOCHONDRIA IN ELECTRON MICROSCOPY IMAGES USING ALGEBRAIC CURVES. Proc IEEE Int Symp Biomed Imaging. 2013;2013:860-863. doi: 10.1109/ISBI.2013.6556611.

52. Lefebvre, A.E.Y.T., Ma, D., Kessenbrock, K. et al. Author Correction: Automated segmentation and tracking of mitochondria in live-cell time-lapse images. Nat Methods 19, 770 (2022). https://doi.org/10.1038/s41592-022-01506

APPENDIX A

DATASET RESULTS

Table 4. Individual test results of configuration 1

Individual File Results	IoU	Dice
./Mito01/results03/mask000.tif	71.10%	87.10%
./Mito01/results03/mask001.tif	71.90%	86.80%
./Mito01/results03/mask002.tif	73.60%	87.40%
./Mito01/results03/mask003.tif	72.70%	86.20%
./Mito01/results03/mask004.tif	74.20%	86.70%
./Mito01/results03/mask005.tif	69.20%	85.10%
./Mito01/results03/mask006.tif	71.60%	87.50%
./Mito01/results03/mask007.tif	69.70%	85.30%
./Mito01/results03/mask008.tif	72.30%	86.90%
./Mito01/results03/mask009.tif	70.50%	85.40%
./Mito01/results03/mask010.tif	65.50%	84.70%
./Mito01/results03/mask011.tif	68.50%	85.30%
./Mito01/results03/mask012.tif	71.00%	87.60%
./Mito01/results03/mask013.tif	65.80%	85.30%
./Mito01/results03/mask014.tif	72.30%	86.60%
./Mito01/results03/mask015.tif	69.50%	86.30%
./Mito01/results03/mask016.tif	70.80%	87.10%
./Mito01/results03/mask017.tif	70.80%	86.10%
./Mito01/results03/mask018.tif	70.30%	87.60%
./Mito01/results03/mask019.tif	69.20%	86.60%
./Mito01/results03/mask020.tif	70.30%	87.60%
./Mito01/results03/mask021.tif	65.00%	84.50%
./Mito01/results03/mask022.tif	70.50%	86.80%
./Mito01/results03/mask023.tif	71.40%	87.60%
./Mito01/results03/mask024.tif	70.50%	87.60%
./Mito01/results03/mask025.tif	73.20%	88.50%
./Mito01/results03/mask026.tif	74.20%	87.70%
./Mito01/results03/mask027.tif	77.20%	88.30%
./Mito01/results03/mask028.tif	77.00%	89.60%
./Mito01/results03/mask029.tif	76.20%	88.40%
./Mito01/results03/mask030.tif	76.40%	88.60%

cont. on the next page

cont. of Table 4

./Mito01/results03/mask031.tif	75.40%	88.70%
./Mito01/results03/mask032.tif	74.20%	88.60%
./Mito01/results03/mask033.tif	76.10%	88.50%
./Mito01/results03/mask034.tif	77.20%	88.70%
./Mito01/results03/mask035.tif	77.60%	88.60%
./Mito01/results03/mask036.tif	72.80%	87.40%
./Mito01/results03/mask037.tif	77.40%	89.40%
./Mito01/results03/mask038.tif	78.10%	89.20%
./Mito01/results03/mask039.tif	75.70%	89.20%
./Mito01/results03/mask040.tif	74.50%	88.00%
./Mito01/results03/mask041.tif	76.50%	88.90%
./Mito01/results03/mask042.tif	77.80%	89.50%
./Mito01/results03/mask043.tif	76.60%	88.50%
./Mito01/results03/mask044.tif	76.00%	88.80%
./Mito01/results03/mask045.tif	78.10%	89.30%
./Mito01/results03/mask046.tif	78.00%	89.20%
./Mito01/results03/mask047.tif	77.40%	89.30%
./Mito01/results03/mask048.tif	76.90%	88.40%
./Mito01/results03/mask049.tif	78.50%	88.90%
./Mito01/results03/mask050.tif	77.80%	89.00%
./Mito01/results03/mask051.tif	75.10%	88.40%
./Mito01/results03/mask052.tif	76.40%	88.90%
./Mito01/results03/mask053.tif	78.50%	89.00%
./Mito01/results03/mask054.tif	79.50%	88.10%
./Mito01/results03/mask055.tif	78.00%	89.30%
./Mito01/results03/mask056.tif	78.20%	88.20%
./Mito01/results03/mask057.tif	75.40%	88.40%
./Mito01/results03/mask058.tif	76.40%	87.90%
./Mito01/results03/mask059.tif	76.10%	87.30%
./Mito01/results03/mask060.tif	73.90%	87.10%
./Mito01/results03/mask061.tif	74.90%	87.50%
./Mito01/results03/mask062.tif	70.40%	88.10%
./Mito01/results03/mask063.tif	74.60%	88.90%
./Mito01/results03/mask064.tif	74.20%	89.00%
./Mito01/results03/mask065.tif	75.10%	89.20%
./Mito01/results03/mask066.tif	72.40%	89.80%
./Mito01/results03/mask067.tif	73.70%	89.40%
./Mito01/results03/mask068.tif	74.50%	90.70%

cont. on the next page

cont. of Table 4

./Mito01/results03/mask069.tif	74.30%	90.30%
./Mito01/results03/mask070.tif	75.10%	91.10%
./Mito01/results03/mask071.tif	73.10%	91.30%
./Mito01/results03/mask072.tif	74.50%	91.30%
./Mito01/results03/mask073.tif	74.50%	90.80%
./Mito01/results03/mask074.tif	74.40%	90.70%
./Mito01/results03/mask075.tif	75.10%	91.10%
./Mito01/results03/mask076.tif	73.00%	90.60%
./Mito01/results03/mask077.tif	76.10%	90.80%
./Mito01/results03/mask078.tif	75.50%	90.00%
./Mito01/results03/mask079.tif	75.20%	90.30%
./Mito01/results03/mask080.tif	76.80%	90.00%
./Mito01/results03/mask081.tif	78.00%	91.00%
./Mito01/results03/mask082.tif	76.20%	91.30%
./Mito01/results03/mask083.tif	73.90%	90.40%

Table 5. Individual test results of configuration 2

Individual File Results	IoU	Dice
./Mito01/results03/mask000.tif	71.10%	87.10%
./Mito01/results03/mask001.tif	71.90%	86.80%
./Mito01/results03/mask002.tif	73.60%	87.40%
./Mito01/results03/mask003.tif	72.70%	86.20%
./Mito01/results03/mask004.tif	74.20%	86.70%
./Mito01/results03/mask005.tif	69.20%	85.10%
./Mito01/results03/mask006.tif	71.60%	87.50%
./Mito01/results03/mask007.tif	69.70%	85.30%
./Mito01/results03/mask008.tif	72.30%	86.90%
./Mito01/results03/mask009.tif	70.50%	85.40%
./Mito01/results03/mask010.tif	65.50%	84.70%
./Mito01/results03/mask011.tif	68.50%	85.30%
./Mito01/results03/mask012.tif	71.00%	87.60%
./Mito01/results03/mask013.tif	65.80%	85.30%
./Mito01/results03/mask014.tif	72.30%	86.60%
./Mito01/results03/mask015.tif	69.50%	86.30%
./Mito01/results03/mask016.tif	70.80%	87.10%
./Mito01/results03/mask017.tif	70.80%	86.10%
./Mito01/results03/mask018.tif	70.30%	87.60%

cont. on the next page

cont. of Table 5

./Mito01/results01/mask019.tif	86.40%	92.10%
./Mito01/results01/mask020.tif	84.70%	91.50%
./Mito01/results01/mask021.tif	85.10%	90.70%
./Mito01/results01/mask022.tif	85.30%	91.60%
./Mito01/results01/mask023.tif	86.50%	91.10%
./Mito01/results01/mask024.tif	86.80%	91.20%
./Mito01/results01/mask025.tif	87.70%	92.20%
./Mito01/results01/mask026.tif	87.20%	91.80%
./Mito01/results01/mask027.tif	88.30%	91.70%
./Mito01/results01/mask028.tif	88.10%	91.90%
./Mito01/results01/mask029.tif	87.10%	91.70%
./Mito01/results01/mask030.tif	91.80%	92.60%
./Mito01/results01/mask031.tif	92.40%	92.40%
./Mito01/results01/mask032.tif	90.20%	92.40%
./Mito01/results01/mask033.tif	91.00%	91.70%
./Mito01/results01/mask034.tif	88.50%	91.90%
./Mito01/results01/mask035.tif	90.30%	91.90%
./Mito01/results01/mask036.tif	88.40%	92.00%
./Mito01/results01/mask037.tif	90.30%	92.60%
./Mito01/results01/mask038.tif	89.90%	92.30%
./Mito01/results01/mask039.tif	91.60%	92.10%
./Mito01/results01/mask040.tif	90.40%	91.80%
./Mito01/results01/mask041.tif	91.50%	92.30%
./Mito01/results01/mask042.tif	91.90%	92.30%
./Mito01/results01/mask043.tif	91.40%	91.90%
./Mito01/results01/mask044.tif	91.20%	92.00%
./Mito01/results01/mask045.tif	90.70%	92.00%
./Mito01/results01/mask046.tif	91.50%	92.30%
./Mito01/results01/mask047.tif	93.00%	92.00%
./Mito01/results01/mask048.tif	91.60%	92.10%
./Mito01/results01/mask049.tif	91.20%	92.50%
./Mito01/results01/mask050.tif	91.50%	92.40%
./Mito01/results01/mask051.tif	91.10%	92.30%
./Mito01/results01/mask052.tif	89.10%	92.10%
./Mito01/results01/mask053.tif	90.70%	92.40%
./Mito01/results01/mask054.tif	90.80%	91.30%
./Mito01/results01/mask055.tif	91.80%	91.50%
./Mito01/results01/mask056.tif	92.10%	91.60%

cont. on the next page

cont. of Table 5

./Mito01/results01/mask057.tif	90.00%	92.00%
./Mito01/results01/mask058.tif	90.90%	91.80%
./Mito01/results01/mask059.tif	88.90%	91.00%
./Mito01/results01/mask060.tif	88.10%	91.30%
./Mito01/results01/mask061.tif	88.30%	91.50%
./Mito01/results01/mask062.tif	85.50%	92.20%
./Mito01/results01/mask063.tif	90.40%	92.60%
./Mito01/results01/mask064.tif	90.10%	92.70%
./Mito01/results01/mask065.tif	88.50%	92.40%
./Mito01/results01/mask066.tif	88.10%	92.90%
./Mito01/results01/mask067.tif	90.80%	93.20%
./Mito01/results01/mask068.tif	86.50%	93.40%
./Mito01/results01/mask069.tif	88.40%	93.40%
./Mito01/results01/mask070.tif	87.80%	93.70%
./Mito01/results01/mask071.tif	88.60%	93.70%
./Mito01/results01/mask072.tif	88.90%	94.00%
./Mito01/results01/mask073.tif	89.50%	93.80%
./Mito01/results01/mask074.tif	90.90%	93.80%
./Mito01/results01/mask075.tif	91.90%	93.90%
./Mito01/results01/mask076.tif	88.20%	93.60%
./Mito01/results01/mask077.tif	89.70%	93.60%
./Mito01/results01/mask078.tif	88.50%	93.20%
./Mito01/results01/mask079.tif	87.60%	93.30%
./Mito01/results01/mask080.tif	91.30%	93.30%
./Mito01/results01/mask081.tif	88.70%	93.60%
./Mito01/results01/mask082.tif	88.80%	93.90%
./Mito01/results01/mask083.tif	88.80%	93.30%

Table 6. Individual test results of configuration 3

Individual File Results	IoU	Dice
./Mito01/results02/mask000.tif	86.10%	90.40%
./Mito01/results02/mask001.tif	86.70%	90.60%
./Mito01/results02/mask002.tif	85.10%	90.60%
./Mito01/results02/mask003.tif	85.60%	90.70%
./Mito01/results02/mask004.tif	81.00%	90.70%
./Mito01/results02/mask005.tif	82.90%	89.90%
./Mito01/results02/mask006.tif	86.90%	91.50%

cont. on the next page

cont. of Table 6

./Mito01/results02/mask007.tif	83.00%	90.70%
./Mito01/results02/mask008.tif	81.60%	90.50%
./Mito01/results02/mask009.tif	80.50%	90.30%
./Mito01/results02/mask010.tif	80.50%	89.80%
./Mito01/results02/mask011.tif	80.00%	90.10%
./Mito01/results02/mask012.tif	86.00%	91.40%
./Mito01/results02/mask013.tif	80.40%	90.70%
./Mito01/results02/mask014.tif	82.50%	91.00%
./Mito01/results02/mask015.tif	87.60%	91.20%
./Mito01/results02/mask016.tif	88.60%	91.30%
./Mito01/results02/mask017.tif	87.10%	91.70%
./Mito01/results02/mask018.tif	82.40%	90.70%
./Mito01/results02/mask019.tif	86.70%	92.20%
./Mito01/results02/mask020.tif	85.40%	91.50%
./Mito01/results02/mask021.tif	85.40%	91.40%
./Mito01/results02/mask022.tif	86.30%	91.70%
./Mito01/results02/mask023.tif	87.60%	91.30%
./Mito01/results02/mask024.tif	86.90%	91.80%
./Mito01/results02/mask025.tif	88.30%	91.80%
./Mito01/results02/mask026.tif	87.70%	91.90%
./Mito01/results02/mask027.tif	89.00%	91.80%
./Mito01/results02/mask028.tif	88.10%	92.70%
./Mito01/results02/mask029.tif	87.90%	91.80%
./Mito01/results02/mask030.tif	92.70%	92.20%
./Mito01/results02/mask031.tif	92.10%	92.80%
./Mito01/results02/mask032.tif	89.60%	93.10%
./Mito01/results02/mask033.tif	90.10%	92.60%
./Mito01/results02/mask034.tif	88.20%	92.30%
./Mito01/results02/mask035.tif	90.20%	92.00%
./Mito01/results02/mask036.tif	90.10%	91.60%
./Mito01/results02/mask037.tif	90.80%	92.60%
./Mito01/results02/mask038.tif	90.60%	91.90%
./Mito01/results02/mask039.tif	90.40%	92.60%
./Mito01/results02/mask040.tif	91.40%	92.40%
./Mito01/results02/mask041.tif	91.90%	92.30%
./Mito01/results02/mask042.tif	92.50%	92.40%
./Mito01/results02/mask043.tif	91.50%	92.00%
./Mito01/results02/mask044.tif	92.50%	92.10%

cont. on the next page

cont. of Table 6

./Mito01/results02/mask045.tif	90.40%	92.50%
./Mito01/results02/mask046.tif	91.60%	92.30%
./Mito01/results02/mask047.tif	92.30%	92.70%
./Mito01/results02/mask048.tif	91.30%	92.50%
./Mito01/results02/mask049.tif	91.70%	92.50%
./Mito01/results02/mask050.tif	92.20%	92.50%
./Mito01/results02/mask051.tif	91.80%	92.40%
./Mito01/results02/mask052.tif	89.60%	92.20%
./Mito01/results02/mask053.tif	91.10%	92.50%
./Mito01/results02/mask054.tif	90.50%	91.80%
./Mito01/results02/mask055.tif	92.10%	92.10%
./Mito01/results02/mask056.tif	91.40%	92.10%
./Mito01/results02/mask057.tif	89.40%	92.10%
./Mito01/results02/mask058.tif	91.10%	91.80%
./Mito01/results02/mask059.tif	88.50%	91.60%
./Mito01/results02/mask060.tif	88.70%	91.30%
./Mito01/results02/mask061.tif	88.80%	91.60%
./Mito01/results02/mask062.tif	85.90%	92.30%
./Mito01/results02/mask063.tif	90.50%	92.70%
./Mito01/results02/mask064.tif	89.90%	92.50%
./Mito01/results02/mask065.tif	89.10%	92.50%
./Mito01/results02/mask066.tif	88.20%	93.00%
./Mito01/results02/mask067.tif	91.20%	93.00%
./Mito01/results02/mask068.tif	87.10%	93.50%
./Mito01/results02/mask069.tif	88.70%	93.40%
./Mito01/results02/mask070.tif	88.00%	93.80%
./Mito01/results02/mask071.tif	88.80%	93.70%
./Mito01/results02/mask072.tif	89.50%	94.10%
./Mito01/results02/mask073.tif	90.30%	93.70%
./Mito01/results02/mask074.tif	91.70%	93.90%
./Mito01/results02/mask075.tif	91.00%	93.70%
./Mito01/results02/mask076.tif	89.10%	93.50%
./Mito01/results02/mask077.tif	89.30%	93.80%
./Mito01/results02/mask078.tif	87.90%	93.00%
./Mito01/results02/mask079.tif	86.30%	93.20%
./Mito01/results02/mask080.tif	91.80%	93.70%
./Mito01/results02/mask081.tif	89.00%	93.50%
./Mito01/results02/mask082.tif	88.30%	93.70%
./Mito01/results02/mask083.tif	89.00%	93.30%

Table 7. Individual test results of configuration 4

Individual File Results	IoU	Dice
./Mito01/results03/mask000.tif	84.90%	92.00%
./Mito01/results03/mask001.tif	86.30%	91.20%
./Mito01/results03/mask002.tif	85.40%	90.80%
./Mito01/results03/mask003.tif	85.60%	90.80%
./Mito01/results03/mask004.tif	80.60%	90.80%
./Mito01/results03/mask005.tif	82.20%	90.80%
./Mito01/results03/mask006.tif	85.80%	91.50%
./Mito01/results03/mask007.tif	83.00%	90.30%
./Mito01/results03/mask008.tif	81.40%	91.00%
./Mito01/results03/mask009.tif	80.60%	90.20%
./Mito01/results03/mask010.tif	80.20%	89.90%
./Mito01/results03/mask011.tif	79.90%	90.50%
./Mito01/results03/mask012.tif	86.10%	91.50%
./Mito01/results03/mask013.tif	80.30%	90.40%
./Mito01/results03/mask014.tif	82.60%	91.20%
./Mito01/results03/mask015.tif	87.60%	91.40%
./Mito01/results03/mask016.tif	89.10%	91.50%
./Mito01/results03/mask017.tif	86.90%	91.80%
./Mito01/results03/mask018.tif	87.30%	91.70%
./Mito01/results03/mask019.tif	87.40%	92.40%
./Mito01/results03/mask020.tif	85.70%	91.60%
./Mito01/results03/mask021.tif	88.60%	91.70%
./Mito01/results03/mask022.tif	86.50%	91.80%
./Mito01/results03/mask023.tif	87.50%	92.00%
./Mito01/results03/mask024.tif	86.80%	91.90%
./Mito01/results03/mask025.tif	88.50%	92.10%
./Mito01/results03/mask026.tif	87.70%	92.30%
./Mito01/results03/mask027.tif	88.70%	92.50%
./Mito01/results03/mask028.tif	88.20%	92.50%
./Mito01/results03/mask029.tif	87.90%	92.00%
./Mito01/results03/mask030.tif	92.80%	93.40%
./Mito01/results03/mask031.tif	91.00%	93.20%
./Mito01/results03/mask032.tif	89.70%	92.90%
./Mito01/results03/mask033.tif	89.40%	92.70%
./Mito01/results03/mask034.tif	87.80%	92.40%
./Mito01/results03/mask035.tif	88.90%	92.50%
./Mito01/results03/mask036.tif	90.00%	91.90%

cont. on the next page

cont. of Table 7

./Mito01/results03/mask037.tif	90.90%	92.70%
./Mito01/results03/mask038.tif	90.60%	92.00%
./Mito01/results03/mask039.tif	90.90%	92.60%
./Mito01/results03/mask040.tif	91.20%	92.50%
./Mito01/results03/mask041.tif	92.00%	92.90%
./Mito01/results03/mask042.tif	92.30%	92.50%
./Mito01/results03/mask043.tif	91.80%	92.10%
./Mito01/results03/mask044.tif	92.40%	92.20%
./Mito01/results03/mask045.tif	90.60%	92.60%
./Mito01/results03/mask046.tif	91.60%	92.40%
./Mito01/results03/mask047.tif	92.40%	92.50%
./Mito01/results03/mask048.tif	91.50%	92.60%
./Mito01/results03/mask049.tif	91.60%	94.50%
./Mito01/results03/mask050.tif	92.20%	92.60%
./Mito01/results03/mask051.tif	91.80%	92.50%
./Mito01/results03/mask052.tif	89.70%	92.30%
./Mito01/results03/mask053.tif	91.40%	92.70%
./Mito01/results03/mask054.tif	90.50%	92.20%
./Mito01/results03/mask055.tif	90.30%	93.00%
./Mito01/results03/mask056.tif	91.40%	92.30%
./Mito01/results03/mask057.tif	88.50%	92.80%
./Mito01/results03/mask058.tif	90.80%	92.30%
./Mito01/results03/mask059.tif	88.70%	92.30%
./Mito01/results03/mask060.tif	88.50%	91.50%
./Mito01/results03/mask061.tif	88.60%	92.10%
./Mito01/results03/mask062.tif	86.50%	92.50%
./Mito01/results03/mask063.tif	91.20%	92.90%
./Mito01/results03/mask064.tif	90.30%	92.90%
./Mito01/results03/mask065.tif	88.90%	92.60%
./Mito01/results03/mask066.tif	89.60%	93.40%
./Mito01/results03/mask067.tif	90.80%	93.50%
./Mito01/results03/mask068.tif	86.90%	93.50%
./Mito01/results03/mask069.tif	88.50%	93.50%
./Mito01/results03/mask070.tif	87.90%	93.90%
./Mito01/results03/mask071.tif	88.90%	93.80%
./Mito01/results03/mask072.tif	89.20%	94.30%
./Mito01/results03/mask073.tif	90.30%	93.70%
./Mito01/results03/mask074.tif	91.30%	94.10%

cont. on the next page

cont. of Table 7

./Mito01/results03/mask075.tif	91.00%	93.80%
./Mito01/results03/mask076.tif	88.30%	93.70%
./Mito01/results03/mask077.tif	89.10%	93.90%
./Mito01/results03/mask078.tif	88.20%	93.10%
./Mito01/results03/mask079.tif	87.10%	93.50%
./Mito01/results03/mask080.tif	91.70%	93.80%
./Mito01/results03/mask081.tif	88.20%	93.80%
./Mito01/results03/mask082.tif	88.60%	93.90%
./Mito01/results03/mask083.tif	88.20%	93.60%
./Mito01/results03/mask075.tif	91.00%	93.80%
./Mito01/results03/mask076.tif	88.30%	93.70%
./Mito01/results03/mask077.tif	89.10%	93.90%
./Mito01/results03/mask078.tif	88.20%	93.10%
./Mito01/results03/mask079.tif	87.10%	93.50%
./Mito01/results03/mask080.tif	91.70%	93.80%
./Mito01/results03/mask081.tif	88.20%	93.80%
./Mito01/results03/mask082.tif	88.60%	93.90%