AN AUTOMATED FRAMEWORK TO SEGMENT OVERLAPPING CELLS FROM CONFLUENT CANCEROUS CELL LINES

A project submitted in partial fulfillment of the requirement for the degree of **Master of Computer Application** in the Department of Computer Science and Engineering.

Of

Jadavpur University

By

Subhadeep Saha

Registration No. – 154242 of 2020-2021

Examination Roll No. – MCA2360029

Under the Guidance of

Prof. Debotosh Bhattacharjee

Department of Computer Science & Engineering

Jadavpur University, Kolkata-700032

India

2023

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

To Whom It May Concern

I hereby recommend that the project entitled "An Automated Framework to Segment Overlapping Cells from Confluent Cancerous Cell Lines" has been carried out by Subhadeep Saha (University Registration No.: 154242 of 2020-2021, Examination Roll No. - MCA2360029) under my guidance and supervision may be accepted in partial fulfillment of the requirement for the Degree of Master of Computer Application in Department of Computer Science and Engineering, Jadavpur University.

Prof. Debotosh Bhattacharjee (Project Supervisor)

Department of Computer Science & Engineering

Jadavpur University, Kolkata-700032

Prof. Nandini Mukhopadhyay

Head, Department of Computer Science & Engineering

Jadavpur University, Kolkata-700032

Prof. Ardhendu Ghoshal

Dean, Faculty of Engineering & Technology

Jadavpur University, Kolkata-700032

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Certificate of Approval

This is to certify that the project entitled "An Automated Framework to Segment Overlapping Cells from Confluent Cancerous Cell Lines" is a bonafide record of work carried out by Subhadeep Saha in partial fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval, all the undersigned do not necessarily endorse or approve any statement made, the opinion expressed, or the conclusion drawn there in but approve the project only for the purpose it has been submitted.

Signature of Examiner 1

Date:

Signature of Examiner 1

Date:

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Declaration of Originality & Compliance of Academic Ethics

I hereby declare that this project entitled "An Automated Framework to Segment Overlapping Cells from Confluent Cancerous Cell Lines" contains a literature survey and original research work by the undersigned candidate as part of his Master of Computer Application studies. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Subhadeep Saha

Registration No. : 154242 of 2020-2021

Examination Roll No. : MCA2360029

Project Title: An Automated Framework to Segment Overlapping Cells from Confluent Cancerous Cell Lines

---

Signature with Date

Acknowledgement

The writing of the project and the related work has been a protracted process that has involved the contributions of many people from the very beginning to the creation of the final project.

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this project. Firstly I would like to thank my project supervisor, **Dr. Debotosh Bhattacharjee,** for his invaluable guidance, support, and encouragement throughout my research work. His expertise, insightful comments, and constructive feedback have been instrumental in shaping this project. His motivation always gave me the required input and momentum to continue my work, without which the project work would not have taken its current shape. I feel deeply honored that I got this opportunity to work with him.

I am also thankful to all the faculty members of the Department of Computer Science & Engineering of Jadavpur University for their continuous support.

Last but not least, I would like to thank all my batch mates of Master of Computer Application at Jadavpur University for staying by my side and helping me whenever needed.

Name: Subhadeep Saha

Registration No. : 154242 of 2020-2021

Examination Roll No. : MCA2360029

ABSTRACT

In this project, we have segment cancer overlapping cell nuclei from microscopy images. This deep learning segmentation model works with two stages, coarse segmentation, and fine segmentation. For coarse segmentation, I use Mask R-CNN, a deep learning instance segmentation technique for pixel-level segmentation, and for fine segmentation, I use the Euclidean distance transform method to generate the marker and then use the watershed segmentation method. Maximum nuclei are segmented in the coarse segmentation part, but some complex overlapping nuclei cannot be segmented well. For this, I use the fine segmentation stage. In fine segmentation for each clump nucleus, we use the Euclidean distance transform method to find the seed points for watershed segmentation. Nuclei that cannot be segmented by Mask R-CNN, by fine-segmenting those overlapping nuclei.

For this experiment on microscopy images, I used the Kaggle 2018 Data Science Bowl dataset, which was taken using various magnifications and modalities. After comparing this method with other popular deep learning methods, this method archives a dice score of 83.3%, AJI of 70.03%, Precision of 89.80%, and Recall of 87.20%, respectively.

Table of Contents

1. INTRODUCTION

In the past few years, medical treatments have required computer technology to detect many types of diseases like cancer, Alzheimer, etc., and to cure all of these complex diseases, we need to understand the drug effect on the cells of the patient, which we have done using cell segmentation.

Cell segmentation divides a microscopic image domain into segments representing distinct cell instances. In many biomedical studies, it is an essential step. Because it is a common prerequisite for many quantitative data analysis pipelines, single nucleus segmentation is a frequent challenge in microscopy image processing. Accurate segmentation is crucial for tracking individual cells, extracting features, and categorizing cellular phenotypes.

The role of Cell segmentation is one of the most crucial steps in the medical field for detecting diseases like cancer, heart disease, chronic obstructive pulmonary disease, Alzheimer's disease, and diabetes. Anatomical data can be more accurately interpreted thanks to cell segmentation, which isolates only the necessary portions. A common first step in image analysis procedures intended to extract important biological information is the identification of cell nuclei. Examples of research studies where the nucleus is a trustworthy reference compartment for identifying single cells in microscopy images include counting cells, following moving populations, localizing proteins, categorizing phenotypes, and profiling treatments.

Segmenting the cell nuclei in microscope pictures is crucial in interpreting imaging data for biological and biomedical applications. Separating the nuclei in microscope images is crucial in the study of disease. This technique can extract individual nuclei's size, density, and pleomorphism, among other morphometric and visual properties. By examining these traits, it can now categorize phenotypes and profile therapies in various biological applications, such as cancer diagnostics, medication development, and cell biology research. It is well known that breast cancer is one of the most common malignant tumors in women globally [1]. The most accurate method for diagnosing, evaluating, and predicting the prognosis of a condition is still microscopic inspection of H&E stained tissue [2]. Accurate cell picture segmentation is required for several downstream studies, such as tracking, cell classification, and feature extraction. The traits that can be retrieved can provide crucial information on biological activities like cell division, cell death, and cell migration. It can help in the development of new therapies and treatments.

How it helps for cancer diagnosis: Cancer is a diverse and complex group of diseases that can invade and spread to different body parts due to abnormal

cell growth and proliferation. It is a significant problem for global health and one of the main causes of death.

In India, 32% of people will develop cancer at some point. With about 0.3 million deaths per year, cancer is one of the most common diseases in India that contributes to the highest mortality rate [5]. Changes in a person's habits, such as an increase in tobacco use, a decline in dietary habits, a lack of activities, and many others, increase their risk of contracting this disease. The recent convergence of medical and engineering advancements has increased the likelihood of cancer cures. The chances of beating cancer are largely dependent on early detection and diagnosis.

Identifying and isolating specific cells or cell regions within an image is known as cell segmentation. It is a crucial step in analyzing cancer images because it allows extracting quantitative features to help with cancer diagnosis and prognosis. We can extract the morphological and textural characteristics of the cells, such as their size, shape, texture, and spatial distribution, by segmenting the cells. These characteristics can be used to categorize various types of cancer, grade the tumor, establish its stage, and forecast the prognosis for the patient.

X-rays, computed tomography (CT), magnetic resonance imaging (MRI), positron emission tomography (PET), and ultrasound are a few of the medical imaging methods that can be used to diagnose cancer. The type, location, and cancer stage determine the best imaging modality. Each imaging technique has advantages and disadvantages. For instance, the detection and staging of cancers of the chest, abdomen, and pelvis are frequently performed using CT and MRI, whereas the detection and follow-up of tumor response to therapy are frequently performed using PET.

However, accurate segmentation of cancer cells remains a significant challenge due to several factors, regardless of the imaging technique used. First off, cancer cells can appear in clusters or overlap with healthy cells. They can also have a variety of sizes, shapes, and textures. Second, poor image quality can make distinguishing between cancer and healthy cells or tissue structures difficult. This is due to factors like noise, blur, and low contrast. Thirdly, it is not easy to integrate the data and extract valuable features because different imaging modalities have different resolutions and contrast mechanisms.

Numerous cell segmentation algorithms have been put forth in the literature to address these issues. These algorithms can be roughly divided into two groups: supervised algorithms and unsupervised algorithms. The learning of a model that can segment cells in new images is done by supervised algorithms using manually labeled training data. On the other hand,

unsupervised algorithms use mathematical or statistical techniques to segment cells based on image features without training data.

Deep learning is one of the most popular supervised algorithms for cell segmentation. Neural networks are used in the machine learning subfield of deep learning to learn data representations. Deep learning has recently demonstrated promising outcomes in several image analysis tasks, including cell segmentation. For instance, nuclei segmentation in histology images has been accomplished using the Mask R-CNN[39] deep learning algorithm.

Unsupervised learning, specifically clustering algorithms, is another promising method for cell segmentation. Based on their features, clustering algorithms group related objects together. Clustering algorithms can be used to group pixels or areas of images that are part of the same cell in the context of cell segmentation. For instance, the watershed algorithm has accomplished cell segmentation in microscopy images.

Hence, it should be noted that cell segmentation is an essential step in the analysis of cancer image data. It makes extracting quantitative features that can help with cancer diagnosis, treatment, and prognosis possible. Due to the heterogeneity of cancer cells, the complexity of imaging modalities, and the variation in image quality, the accurate segmentation of cancer cells continues to be a significant challenge. Nevertheless, improvements in deep learning and clustering algorithms have produced encouraging outcomes in cell segmentation, giving hope for future cancer diagnosis and treatment improvements.

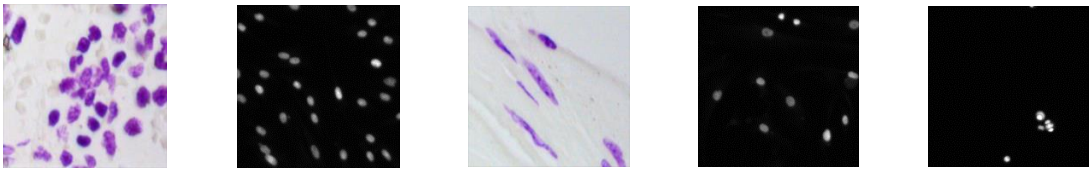


Fig-1.1: Some examples of microscopy cell images

In microscopy images, cell segmentation distinguishes individual cells from complex cell clusters, tissues, or other biological structures. Cell segmentation isolates key characteristics of individual cells, including their shape, size, texture, and fluorescence intensity. Then, various quantitative analyses are performed using these features, including cell counting, tracking, and profiling. Investigating various biological processes like cell division, differentiation, migration, and how different medications or illnesses affect cellular behavior can all be done using this information.

For example, in cancer diagnosis, cell segmentation can assist in locating abnormal cells and quantifying their attributes, such as size and shape, which can be used to categorize cancer types and stages of the disease. Also, it can be used in drug discovery to determine how drugs affect specific cells and evaluate their potential as therapeutic agents. It can be used in tissue engineering to monitor cell behavior and enhance tissue growth. Additionally, precise cell segmentation is essential in various clinical contexts, where it's critical to examine the traits and behaviors of individual cells to create potent therapies.

Nevertheless, getting precisely accurate segmentations of a microscope picture at the instance level can be challenging for several reasons [3][4].

Firstly, the image quality of the acquired image can vary greatly depending on the imaging equipment employed. When the image quality is poor, there is low contrast, noise, and blur; it can be exceedingly challenging to differentiate individual cells from the backdrop accurately.

Secondly, due to cell overlapping, it can be challenging to separate cells from tissue samples accurately. Accurate segmentation is further complicated by the potential occurrence of imprecise or partially occluded cell borders.

Thirdly, the amount of nucleus occlusion, absorption, and groups can influence the determination of morphological traits and cause over- or under-segmentation in cells with complicated shapes.

Fourthly, the images are displayed in a variety of pathological circumstances. Nuclei's appearance, magnitude, and density change depending on the disease subtype due to nuclei expansion, margination, and conspicuous nucleoli. We, therefore, need a strategy that generalizes these variances and difficulties.

The use of microscopic image techniques has expanded rapidly in recent years, and numerous algorithms are now available for studying the segmentation of nuclear images. Image segmentation can be done using algorithms such as thresholding and morphological operations, watershed segmentation, object detection, and instance segmentation using deep learning. As a result, numerous complex algorithms have been proposed to achieve the accuracy required for medical image segmentation.

Now we give an overview of how cell segmentation works from start to end of the process (Shown in fig-1.2),

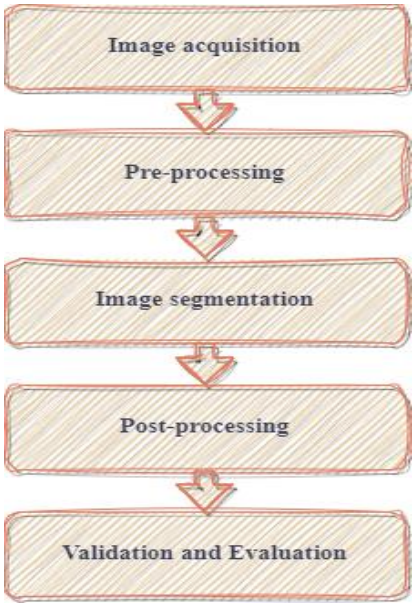


Fig-1.2: Structure of cell segmentation process

Image acquisition: The acquisition of images containing the target cells or tissues is the first step in nuclei segmentation. Depending on the application and the type of cells being imaged, this can be done using various imaging techniques, such as bright field, fluorescence, or confocal microscopy.

Pre-processing: Preprocessing procedures are frequently used to improve the quality of the images and lower noise after they have been acquired. Image filtering, contrast enhancement, and image normalization are common preprocessing methods.

Image segmentation: The main task of nuclei segmentation is to recognize and distinguish individual nuclei from the image's background. Different image segmentation algorithms can accomplish this, such as thresholding, region-based, edge-based, and machine learning-based methods. Setting a pixel intensity value as a threshold and classifying pixels as foreground (nuclei) or background based on whether they are above or below that threshold is a straightforward and frequently used technique called thresholding. To create nuclei regions using region-based methods, connected regions of pixels with similar intensity values are grouped. Edge-based techniques concentrate on locating the edges or boundaries of nuclei using gradients in intensity or other features. Machine learning-based methods train a model using labeled training data to segment nuclei in new images automatically.

Post-processing: Then Nuclei segmentation is followed by post-processing techniques to polish the data and eliminate false positives or negatives. This can involve filtering based on size, shape, or other features to eliminate noise or artefacts and morphological operations like erosion or dilation to smooth out or fill in gaps in the segmented nuclei.

Validation and Evaluation: Following nuclei segmentation, it's critical to validate and assess the precision of the segmentation findings. To evaluate the accuracy and robustness of the segmentation algorithm, it is possible to compare the segmented nuclei with ground truth data, such as manually annotated images, and calculate metrics like precision, recall, F1 score, or Jaccard index.

The objective for this project is to detect individual nuclei overlapping in a given microscopy image accurately. Due to the occlusion, the nuclei's variable shapes and sizes, and the image's noise, this is a difficult task. In cancer research, segmenting overlapping nuclei can offer crucial details for cancer diagnosis, prognosis, and treatment.

To study how drugs affect cell growth and division, segmentation of overlapping nuclei is crucial in the drug discovery process. When cells are treated with medications that affect their proliferation or result in abnormal growth, overlapping nuclei may develop.

In developmental biology, segmenting overlapping nuclei is crucial for understanding how cells grow and differentiate during embryonic development. During cell division and tissue differentiation, overlapping nuclei can occur, and their precise segmentation can shed light on the mechanisms that control cell proliferation and differentiation.

So, this automated segmentation of overlapping nuclei can produce quantitative cell proliferation measurements, including the number of cells, their growth rate, and the nuclei's size and shape. These measurements can be used to screen and assess potential drugs for their effectiveness and toxicity.

2. RELATED STUDIES

In the last few years, medical image segmentation has been boosted quite a bit. As a result, many research studies have been done on cell segmentation, and many of them outperformed. We will now discuss some previous notable nuclei segmentation techniques from microscopic images.

Approaches to segment nuclei on watershed segmentation, morphological operations like erosion, dilation, opening and closing color-based thresholding, and variations of active contours were used before the invention of deep learning. Otsu's method[6] is a well-known thresholding technique. Thresholding is a fundamental segmentation algorithm establishing a pixel value threshold to separate the foreground from the background. Another fundamental algorithm is watershed segmentation, which divides areas of an image according to their local minima or maxima. Active contour models, also called snakes, are a category of algorithm that repeatedly alters a contour to match the boundary of an object in an image. To enhance the image quality, various pre-processing techniques, such as contrast enhancement and deblurring, were combined with these techniques. In addition, several post-processing methods, including hole filling, noise removal, graph cuts, etc., were used to polish the segmentation algorithms' outputs. However, these methods frequently give inaccurate results due to differences in the morphologies of the nuclei in different organs and tissue types and inter- and intra-nuclei color variations in dense, sparse nuclei. Deep learning techniques currently outperform predetermined algorithms for segmentation. Based on annotations of nuclear and non-nuclear pixels, machine learning techniques produce data-driven algorithms trained under supervision.

A marker-controlled watershed based on mathematical morphology was presented by Yang et al. [7] to extract nuclear instances with less over-segmentation. Marker-controlled watershed segmentation has several drawbacks, including sensitivity to marker placement and high computational. Box-free instance segmentation is an illustrative class of algorithms that segments nuclei instances directly without using bounding boxes. For instance, Chen et al. [8] explicitly used the nuclear appearance and contour information to present a deep contour-aware network (DCAN). They first used complementary appearance and contour information to perform semantic segmentation and contour extraction within a unified multi-task learning

framework. They then combine the contours with the semantic segmentation results to distinguish between touching and clustered nuclei. In the MICCAI nuclei segmentation challenge, DCAN came in the first place. However, when the boundaries between the touching nuclei are unclear, it frequently results in over-segmentation. Boundary-Enhanced Segmentation Network (BESNet), proposed by Oda et al. [9], has two decoding paths that can be used to restore the resolution and an encoding path similar to Unet. One decoding strategy aims to improve nuclear boundaries, which can then be used to boost the quality of the other decoding strategy's segmentation of all the nuclei. BESNet has two drawbacks: first, it learns complementary information in nuclei branches but ignores the potential advantages of nuclei for contour, and second, it has a high computational complexity. Zhou et al. [10] proposed the Contour-aware Informative Aggregation Network (CIA-Net) by bi-directionally aggregating task-specific features, which benefits spatial and texture dependencies between nuclei and contour. Hover-Net, developed by Graham et al. [11], uses the instance-rich information encoded in the vertical and horizontal distances between nuclear pixels and their centers of mass to segment and classify nuclear data. By combining a multi-scale dense unit, a spatial aware network (Spa-Net) was proposed by Koohbanani et al. [12] to predict the position information of each nucleus. A clustering technique separates the final nuclei instances by grouping the predicted location ordinates.

Other than that, In 2012, novel contour-based "minimum-model" cell detection and segmentation approaches were proposed for the nuclei segmentation task. These approaches used a priori knowledge to detect contours independent of their shape and showed promising segmentation results [13]. A CNN model from microscopic images was used to propose the Nuclei membrane segmentation method 2012 [14]. UNet was proposed by Ronneberger et al. [15] in 2015, and this model was used to segment medical images, achieving state-of-the-art performance. An improved version of the U-Net deep learning model with recurrent, residual modules instead of forwarding convolutional layers was proposed for medical image segmentation problems in 2019 by Md Zahangir Alom and Chris Yakopcic [16]. The model outperformed U-Net and Seg-Net in tests using various medical imaging modalities, including those for lung segmentation, skin cancer segmentation, and segmentation of retinal blood vessels.

Lugagne et al. [17] created a deep learning-based image analysis pipeline to perform segmentation, tracking, and lineage reconstruction in 2020. To

track cells and reconstruct lineages, they also use machine learning. In 2020, Moshkov et al. [18] proposed a method for cell segmentation on microscopy images by combining two techniques: semantic segmentation based on the U-Net and instance segmentation based on the Mask R-CNN models. White blood cell nuclei can be automatically segmented and classified using a CNN model in 2020, according to a method proposed by Banik et al. [19]. This method aids in the diagnosis of blood-related diseases. They divide the WBC's nuclei using color space conversion and the K-means algorithm, localize the WBC based on the nuclei's locations, and then employ a convolutional neural network (CNN) model-based classification technique. Feixiao Long proposed a microscopy cell nuclei segmentation with enhanced U-Net in 2020 [20]. It has been demonstrated to have comparative performance with other well-known U-Net-like structures, at least in the cell nuclei segmentation task. A method for segmenting the nuclei in cell images was put forth by Rautaray et al. in 2020 [21]. They employ a fully convolutional neural network architecture specifically designed for encoder-decoder applications. The method for label-free live cell segmentation proposed by Edlund et al. in 2021 [22] involves training a convolutional neural network and assessing the segmentation accuracy of the model using a set of benchmarks. In 2022, Noah F. Greenwald et al. [23] proposed a method for cell segmentation of tissue images using the TissueNet dataset for issues like using a deep-learning-enabled segmentation algorithm to identify each cell's boundary in an image precisely.

All the state-of-the-art mentioned above is given in below table 2.1 with a method, dataset used, result, and limitation.

Table 2.1:

Author	Method	Dataset	Result	Limitation
Lugagne et al. [17]	U-Net Architechture with DeLTA Algorithm	The E. coli strains image	Segmentation Error rate – 0.2% Tracking Error rate – 3.07%	Cell tracking is not promising
Moshkov et al. [18]	U-Net, along with Mask R-CNN	Kaggle's Data Science Bowl 2018	mAP(Mean Average Precision) – 0.644	Accuracy is not great
Banik et al. [19]	convolutional neural network (CNN) model by combining	BCCD, ALL-IDB2, JTSC and CellaVision	Average accuracy 96%	High Processing Time

	the concept of fusing the features of the first and last convolutional layers			
Feixiao Long [20]	An enhanced, light-weighted U-Net (called U-Net+) with a modified encoded branch	2018 Kaggle cell nuclei segmentation competition	Avg IOU – 0.566 Avg precision – 63.4	The complexity of the model
Rautaray et al. [21]	Fully Convolutional neural network with residual block and custom encoder-decoder	Broad Bioimage Benchmark Collection, BBBC038v1 Dataset	F1 Score – 0.9736 Iou – 0.9486	Huge training time
Edlund et al. [22]	Two state-of-the-art CNN-based instance segmentation models	LIVECell Dataset	Avg Precision – 80%	Annotations in some regions where cell boundaries are not readily visible
Z. Zeng et al. [52]	Ric-Unet (Network based on U-Net)	H&E stained histology images	F1 Score – 0.83 Aggregated Jaccard index – 0.56	Complexity of model
Fernandez et al. [53]	Based on U-Net architecture	Microscopic images of T-Cells	F1 Score – 0.7347 Precision – 0.8113 Recall – 0.6713	Training time is very high
Caicedo et al. [54]	Based on U-Net architecture	Broad Bioimage Benchmark Collection with	F1 Score – 0.899 Jaccard index – 0.91	Require high computing power

		accession number BBBC039		
Yang et al. [7]	Segmentation by threshold selection	Nuclei fluorescence image	Segmentation accuracy- 98.8% Cell division tracking accuracy- 97.4% Cell tracking accuracy- 97.6%	It is not effective on overlapping nuclei
Chen et al. [8]	Fully Convolution network with deep contour-aware network	Dataset of gland segmentation challenge, MICCAI 2015	DSC Score – 0.812	The boundaries between the touching nuclei are unclear
Oda et al. [9]	U-Net architecture with two decoding part	histopathological images	DSC Score – 0.74 F1 Score – 0.7952	a high computational complexity
Zhou et al. [10]	Fully Convolution network with one densely connective encoder and two decoders	MoNuSeg dataset of the 2018 MICCAI challenge	F1 Score – 0.8458 AJI - 0.6306	The complexity of the model
Graham et al. [11]	Deep neural Network with 50 residual layers along with a dense layer	Colorectal nuclear segmentation and phenotypes (CoNSeP) dataset, consisting of 41 H&E stained image	DSC Score – 0.853 AJI – 0.571	High Computational cost
Koohbanani et al. [12]	Mixture Density Networks (MDNs)	Colorectal cancer (CRC) dataset	F1 Score – 0.832 Precision – 0.788 Recall – 0.882	Huge training time
Wienert et al. [13]	Novel contour optimization method and an	H&E stained tissue	Precision – 0.908 Recall – 0.859	No promising result for

	optional cluster separation			overlapping cells
Gambardella et al. [14]	DNN architecture consists of a succession of convolutional, max-pooling, and fully connected layers	ISBI 2012 EM Segmentation Challenge Dataset	Rand error [$\cdot 10^{-3}$] – 48 Warping error [$\cdot 10^{-6}$] – 434 Pixel error [$\cdot 10^{-3}$] - 60	misplaced borders
Ronneberger et al. [15]	U-Net Architecture	ISBI 2012 EM Segmentation Challenge Dataset	IOU on PhC-U373 data – 0.920 IOU on DIC-HeLa data – 0.775	Require high computing power
Alom et al. [16]	RU-Net and R2U-Net model based on U-Net	DRIVE Dataset, consisting of 40 color retina images, STARE Dataset, Contain 20 color images	Accuracy in DRIVE data – 0.9613 Accuracy in STARE data – 0.9712	Complexity of Model

3. DETAILED BACKGROUND OF DEEP LEARNING

In this part of the project, we introduce deep learning from the basics and also talk about the neural network, which is the main concept of deep learning.

3.1. History of neural network

The term "neural networks" naturally assumes it has something to do with the human brain. The inspiration for this idea came directly from neuroscience at the outset. Deep learning has evolved and becomes a distinct field of study over time. It incorporates methods from probability theory, linear algebra, and machine learning. The best way to describe artificial neural networks is as parallel-processing-based computational models [24] with specific characteristics like the ability to adapt, generalize, cluster, or organize data.

The first artificial neuron, a mathematical representation of a neuron cell, was created in 1943, marking the beginning of neural networks. McCulloch and Pitts [25] presented this model, which is a binary classifier. They showed that a simple model of a neuron can be used to perform logical operations. Weights were in charge of dictating how the model behaved, and they had to be manually set by a human operator. They also created a physical model of the neuron using electrical circuits as part of their research. Although it had no application in real life, it encouraged other researchers to continue their work. F. Rosenblatt [26] published the Perceptron neuron model in 1957. This model could learn weights to represent given data. Additionally, Rosenblatt demonstrated that his learning algorithm always discovered the ideal weight vector when one existed.

Neural networks, however, lost popularity in the 1960s and 1970s due to computing power restrictions and a lack of theoretical understanding. The advent of backpropagation, a method for training multi-layer neural networks, in the 1980s led to a resurgence in interest in neural networks. Neural networks were extensively used in the 1990s in many different industries, including speech recognition, image processing, and control systems. However, the rise of support vector machines (SVMs), regarded as a more trustworthy and understandable substitute, caused their popularity to decline again in the early 2000s.

Recently, neural networks have come back thanks to the availability of large amounts of data and advances in computing power, which have enabled the training of deep neural networks with many layers. Deep learning involves deep neural networks and has shown impressive results in various tasks, including image and speech recognition, natural language processing, and game playing. The use of neural networks in modern technology spans various fields, such as self-driving cars, medical diagnosis, and financial forecasting.

3.2. Computer vision and machine learning

Computer vision is the study of visual data and is primarily concerned with sophisticated image processing methods. The main objective is to make computers capable of viewing, processing, analyzing, and comprehending digital photos or videos. Many industries, including healthcare, finance, transportation, and entertainment, stand to benefit from the convergence of computer vision and machine learning. Computers can use algorithms provided by machine learning to carry out tasks like time series analysis, speech recognition, and medical image analysis. Machine learning methods are being used to complete computer vision tasks as labeled data is becoming more and more readily available online. Object detection, image classification, and segmentation are just a few of the computer vision tasks where deep learning, a type of machine learning, is currently state-of-the-art. Deep learning is a machine learning algorithm comprising several layers using input data to extract useful features for a task. As an illustration, the first layers of the deep learning architecture extract simple features when the input is an image, whereas higher levels extract high-level and more abstract features. These capabilities enable tasks like image segmentation, object detection, and classification. Many deep learning-based algorithms and architectures have been developed over the years. One such deep learning architecture is the Convolutional Neural Network (CNN), initially created for image classification and swiftly modified for image segmentation.

3.3. Artificial neural networks (ANNs) and Convolution neural networks (CNNs)

The biological behavior of neurons served as a model for artificial neural networks (ANNs). A neuron can be modeled as a function that retrieves an output after receiving inputs (from the axons of other neurons). The inputs are first weighted based on a set of weights $\{\omega\}$. The processing unit will then

add these weighted inputs and put them through an activation function. Therefore, the final result is provided by,

$$y = f\left(\sum_i \omega_i x_i + b\right)$$

Where the non-linear activation function f(.) is indicated, ω_i stands for the weight, x_i for the neuron's input, and b for the bias term, which is a threshold.

An artificial neural network can be created by fusing neurons (fig 3.1).

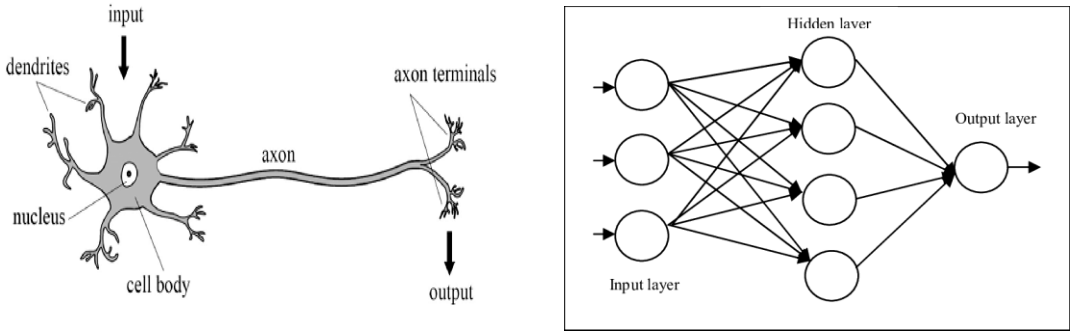


Fig-3.1: (a) Example of a biological neuron, reference [55] (b) Architecture of ANNs, reference [56].

An input vector is fed into a neural network and is processed along one or more hidden layers. A group of neurons are present in each hidden layer but are not connected to the neurons in the layer above or below them. The weight identifies the strength of the connection between two neurons. The information spreads sequentially from the input layer to the final layer, the output layer. The network's output in a classification task corresponds to the probabilities of each class.

For example, let's say there is a neural network (Shown in fig 2.2) for the dataset of diabetic and non-diabetic persons where eight input features are present, which are Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age and by input these feature our artificial neural network predicts that one person is diabetic or not.

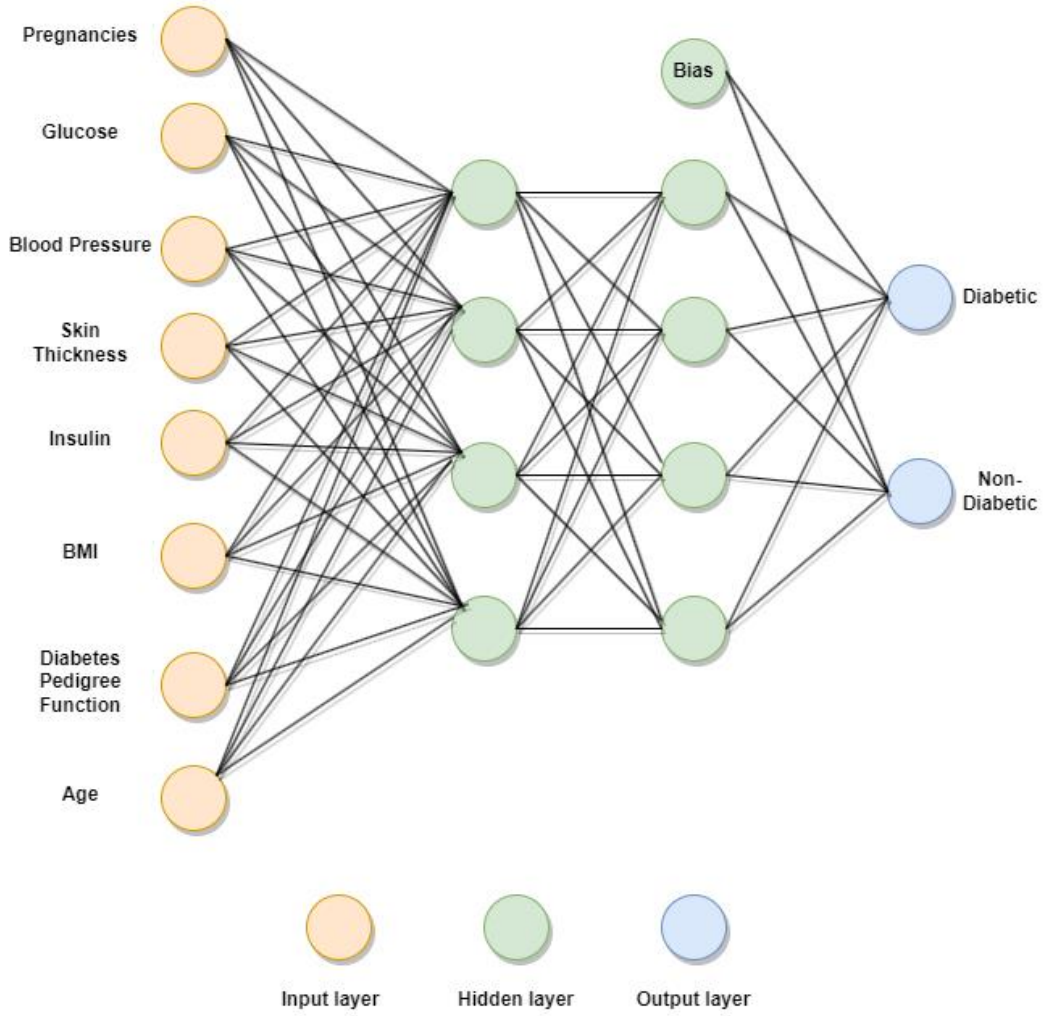


Fig-3.2: Neural network example of a diabetic and non-diabetic dataset

After passing the input vector forward through the neural network (fig-2.2), the class with the highest probability will be the neural network's output.

The input for the prior illustration was an eight-dimensional vector. If an image is used as the input, what will happen? If the image has a low resolution, such as 16 x 16 x 3, each neuron in the first fully connected layer would have (16 x 16 x 3 + 1) = 769 weights. A completely connected network might be adequate for an image of this size. However, if we give a Red-Green-Blue (RGB) image as an input, the number of weights for each neuron would be (512 x 512 x 3 + 1) = 786433. This is quite a high number that would produce an expensive neural network in computation and overfitting. As a result, CNNs were created to address this issue and were based on the concept of neural networks.

CNN has the benefit of being able to be used with structured data. The network receives data from a multidimensional array. A grayscale input image is represented as a two-dimensional array where each number corresponds to a single pixel. A color image with each pixel described by a vector of color values is represented as a three-dimensional array. Additionally, one-dimensional sequences like written text or sound signals can be used to train a convolutional neural network. A convolutional network's neurons are arranged in a grid and can consider the relationships between nearby data samples.

Each neuron scans a section of the previous layer to account for individual pixels in an image that is not entirely independent. The CNN architecture is based on how the brain functions. Each neuron in the brain responds to stimuli in a specific visual cortex area known as its receptive field [27]. Similarly, the region of the image that a CNN neuron is focused on is referred to as its receptive field. The neurons in a CNN are arranged into various feature maps in each layer's multiple layers. The weight parameters shared by the neurons in the same feature map are identical. Using this weight sharing, the number of parameters that can be learned can be decreased. Therefore, side-by-side neurons in a feature map that observe a region of size (n, m, d) and share the same weights only need to learn n x m x d+1 parameters, regardless of the size of the input image.

In most cases, they must learn 3 x 3 x d + 1 or 5 x 5 x d + 1 number of parameters, where d denotes the depth of the input image or the quantity of feature maps in the final layer, and 1 denotes the bias term. For example, if a 3 x 3 or 5 x 5 filter is used for the convolution layer, then for a 3 x 3 filter size, the learnable parameter will be (3 x 3 x 3 + 1) = 28, and for 5 x 5 filter size, the learnable parameters will be (5 x 5 x 3 + 1) = 76. The filter or kernel of the convolution operation is another name for the weight parameters. Next, we will discuss the convolution layer, the main part of CNNs.

3.4. Convolution Layer

The core of a CNN's architecture is its convolutional layer. It is a layer of neurons that uses convolution to calculate their inner potential. A convolutional layer's neurons are arranged in a grid in a fixed position. These neurons' grid is known as a feature map. A kernel provides the weights of the connections between them. One convolutional layer may contain multiple feature maps that operate concurrently on the same data set. The kernel of each feature map is used.

A filter will first convolve the input image, sliding the kernel across the image and computing dot products at each location. The filter is applied to each pixel in the input image, and the dot product of each pixel is calculated, returning one value that is added to the filtered image. An illustration of the convolution operation is shown in Fig-3.3; take note that the input image is 6 x 6, and the filtered output image is 4 x 4 in size. This is because the pixel in the left top corner cannot be positioned with the kernel's center on top of it. The original image can be padded by values to fix this issue. The original image can be padded by values to fix this issue. When padding is zero, for example, pixels outside the original image are set to zero [28]. The calculation for the number of pixels to pad in each direction is,

$$p = \frac{f-1}{2}$$

Where p is the number of pixels to pad in each direction, and f x f is the size of the convolutional kernel. In other words, increasing the size of an image by p zeros changes it from (h, w) to (h + 2p, w + 2p).

The stride is yet another vital idea. The stride in the example shown in Fig-3.3 is one because the filter shifts one pixel at a time during the convolution operation. The stride measures how many units or pixels the filter shifts when convolved with an image or a feature map. If an image of size (h, w, d) is convolved with N filters of size f x f x d, assuming the image is padded with zeros by the formula shown above this, and assuming the stride of the convolution operation is s, then the size of output filtered image will be,

$$H = \frac{h+2p-f}{s} + 1, W = \frac{w+2p-f}{s} + 1, \text{D} = \text{N}$$

Where (H, W) represents the size of the filtered image(s) and D represents how many filtered images there are. Each filter will produce a filtered image in the convolutional layer. The dot product of the filter and a small volume of the filtered image yield each number in the filtered image.

Although convolution is a linear operation, most problems have non-linear input–output mappings as their solutions. Therefore, the filtered image(s) are subjected to non-linear activation functions. The network will be able to learn non-linear decision boundaries as a result. Without activation functions, the network's input-to-output relationship would be linear. In the next section, we will discuss the activation functions.

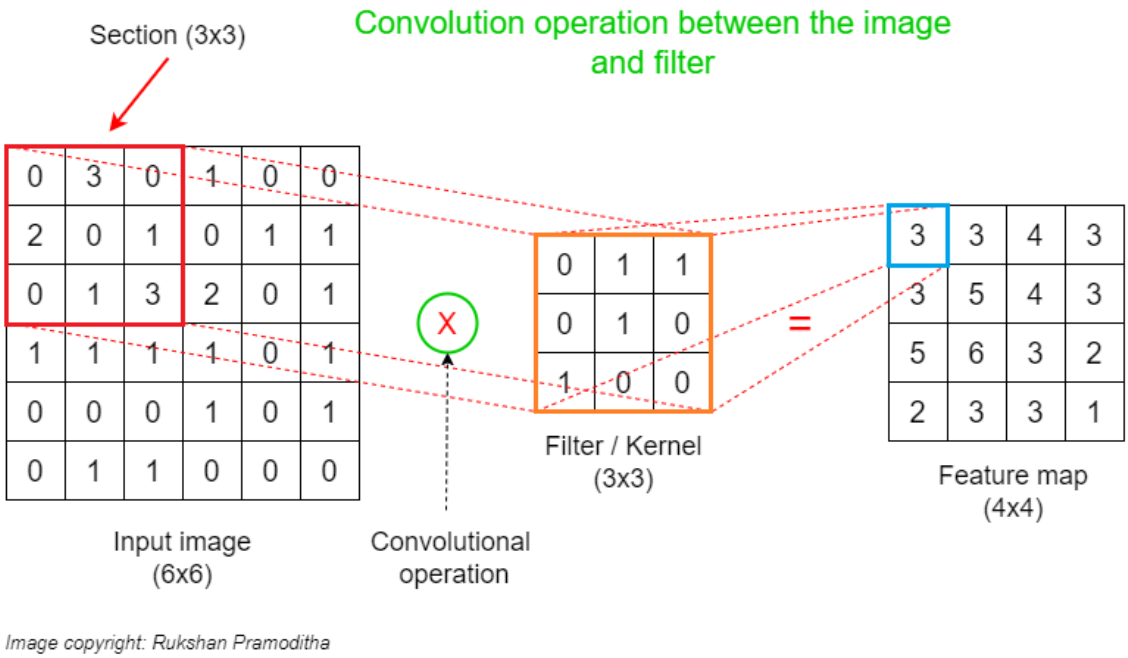
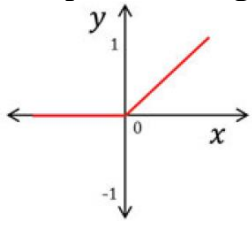


Fig-3.3: An illustration of how a convolution layer works, Reference [57]

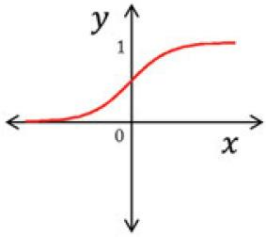
3.5. Activation Functions

The activation function [29], which introduces non-linearity into the network and enables it to model intricate relationships between inputs and outputs, is one of the crucial elements of a CNN. Sigmoid, tanh, ReLU (Rectified Linear Unit), Leaky ReLU, and ELU (Exponential Linear Unit) are some activation function types used in CNNs. Although sigmoid and tanh functions were frequently used in the early stages of neural networks, they have some disadvantages, including slow convergence and vanishing gradients. Due to its efficiency and simplicity, ReLU is currently the most popular activation function.

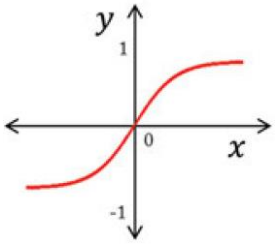
ReLU (rectified Linear Unit): ReLU is defined as $f(x) = \max(0, x)$, where x is the input to the function. Some advantages are that it does not saturate in the positive region ($x > 0$), is computationally effective, and converges more quickly than sigmoid and Tanh. And some disadvantages are that the output is not zero centered, it may lead to dead neurons in the negative region (i.e., $x<0$), which will never activate or update. The graphical plot is,



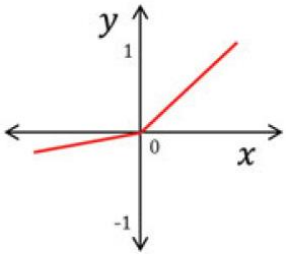
Sigmoid: Sigmoid is defined as, $(x) = \frac{1}{1+ e^{-x}}$, and has been used in the past for binary classification problems. Some advantages are Squashes the number to the range [0, 1], interpreted as a neuron's "firing rate" reaching saturation. And some disadvantages are that the gradients can be "killed" by saturated neurons or units, results are not zero-centered, and the computational cost of exponential is high. The graphical plot is,



Hyperbolic Tangent (Tanh): Tanh is defined as, $\tanh(x) = \frac{e^{x}- e^{-x}}{e^{x}+ e^{-x}}$, it is a sigmoidal function that maps its input to a range between -1 and 1. Some advantages are that it is zero-centered, and some disadvantages are When saturated 'kills' the gradients. The graphical plot is,

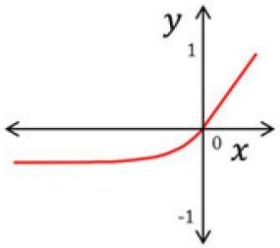


Leaky Rectified Linear Unit (LeakyReLU): LeakyReLU is defined as f(x) = max(ax, x), where x is the input to the function, and a is a small positive constant (usually around 0.01). Some advantages are that it does not saturate, is efficiently computed, and does not lead to dead neurons. And some disadvantages are that compared to ReLU; there isn't much of a performance improvement. The graphical plot is,



Exponential Linear Unit (ELU): ELU is defined as,

$elu(x) = \begin{cases} x, & if\ x \geq 0 \\ \alpha(e^x - 1), & if\ x < 0 \end{cases}$, Where α is a hyperparameter that controls the function's output when x is negative. Some advantage is that Mean activations are pushed toward zero. And some disadvantages are required an expensive and time-consuming exponential calculation and does not set the values' centers at 0. The graphical plot is,



3.6. Fully Connected Layer

Fully connected layers (FC layers), or dense layers in neural networks, are layers in which every neuron in one layer is connected to every neuron in the layer below it. In other words, all the neurons before the current layer contribute information. A fully connected layer's function is to learn non-linear feature combinations from the preceding layer. For instance, each unit of the first FCL will have h x w x d + 1 weight/connections if the final output volume of the convolutional and pooling layers has the dimensions (h, w, d) (Shown in fig-3.4).

The FC layer typically receives the output of the preceding layer as an input, flattened into a vector. The input vector is then subjected to weights and biases the FC layer applies, enabling it to learn and represent more intricate patterns and relationships. Depending on the particular task and network architecture, a fully connected layer's number of neurons can change. Multiple FC layers may occasionally be stacked on top of one another to learn increasingly complex features. Deep learning models frequently employ fully connected layers for tasks like speech recognition, natural language processing, and image classification. However, because so many weights and biases need to be learned, they can also be costly computationally and prone to overfitting.

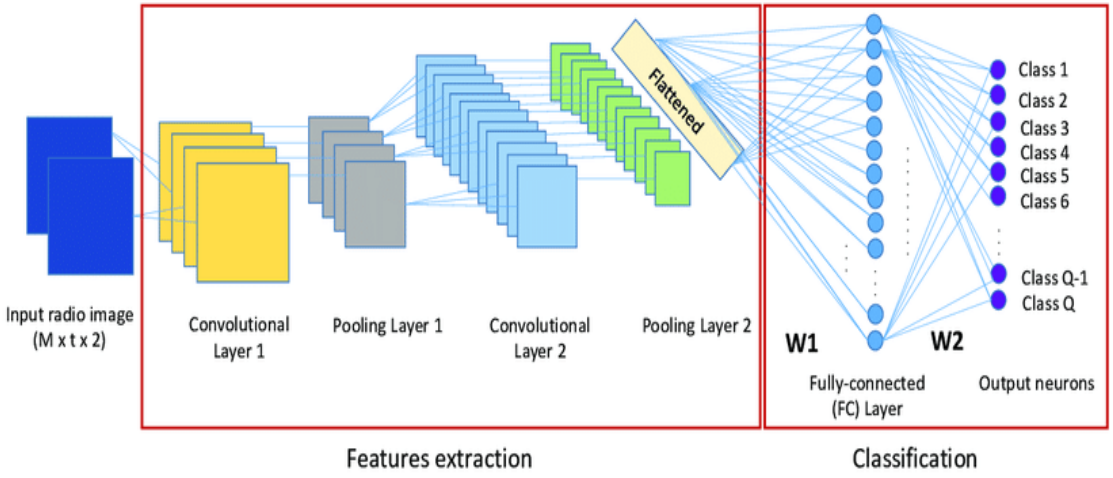


Fig-3.4: An illustration of a fully connected layer after the pooling layer, Reference [58]

3.7. Pooling Layer

A pooling layer is a neural network layer with no learnable parameters. During the pooling operation, the feature map is down-sampled (its height and width are decreased). This operation not only allows for a reduction in the amount of computation required but also for acquiring a feature representation invariant to small input translations. The pooling layer primarily introduces invariance to minute translations in the input image. The feature map's resolution decreases, and the pooling layer decreases sensitivity to slight input variations. By lowering the number of parameters in the network, pooling layers also aid in reducing the computational complexity of CNNs. The size of the output activation map for a pooled region of size f x f and stride s is given by,

$$H = \frac{h-f+s}{s}, W = \frac{w-f+s}{s}$$

Where (h, w) and (H, W) are the input feature map's height and width and the output obtained after applying the pooling operation to this feature map. The pooling layer generates an output of size (H, W, d) for an input volume of size (h, w, d), where the relationship between (H, W) and (h, w) is shown in the previous equation.

In CNNs, various pooling layer types are employed, including maximum pooling, average pooling, global maximum pooling, global average pooling, etc. The choice of pooling layer depends on the particular task at hand and the network architecture, and each type has advantages and disadvantages of its own. The max and average pooling operations are the most popular [30].

A window is slid across the feature map by max pooling, and the output corresponds to the window's maximum value at each position. The output will have a halved height and width because this operation is typically applied to non-overlapping regions of the feature map, so the window size is typically 2 x 2, and the stride is typically 2. A max pooling operation is demonstrated in Figure 2.5 using a window size of 2 x 2 and a stride of 2.

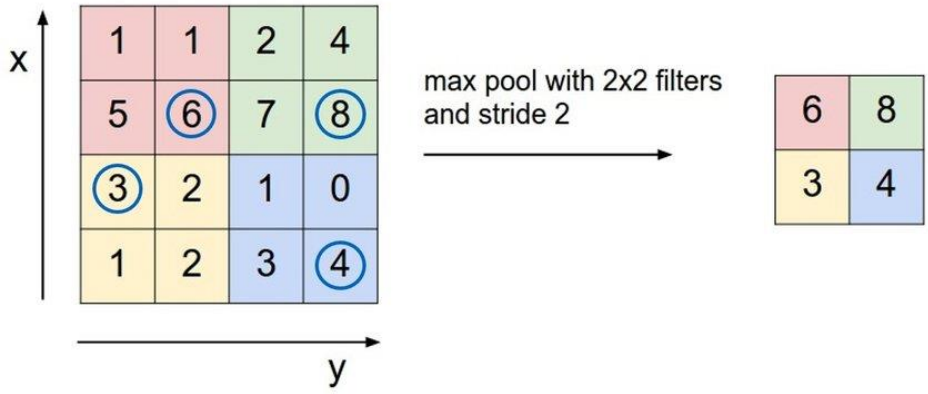


Fig-3.5: An illustration of max pooling operation with window size 2x2 and a stride of 2, Reference [59]

In summary, pooling is especially useful for image classification tasks where you only need to find the presence of a specific object in an image but don't care exactly where it is. In addition, pooling filters with longer strides and smaller outputs than convolutional filters contribute to the network's effectiveness and speed up training. In other words, location invariance can significantly increase the network's statistical effectiveness.

3.8. Training of CNNs

The main goal of training a convolution neural network is to optimize the neurons' weights. Optimizing the model's parameters during training is necessary for CNN to classify the input images correctly. To update the model parameters using an optimization algorithm, the training process involves feeding the network a large set of labeled images, propagating the inputs forward through the network, computing the loss function, and backpropagating the error through the network. The objective of training a CNN is to reduce the loss function, which measures the discrepancy between the predicted and produced outputs. The loss function is calculated by contrasting the predicted output with the ground truth labels. To minimize the loss, the parameters are updated in the opposite direction of the gradient using

the backpropagation algorithm, which is used to compute the gradient of the loss function concerning the model parameters.

How the parameters will be updated will be decided by the optimizer. Stochastic Gradient Descent (SGD), SGD with momentum, SGD with Nesterov momentum, Adaptive Gradient (AdaGrad), Adaptive Delta (AdaDelta), RMSprop, and ADAptive Moment Estimation (ADAM) are examples of frequently used optimizers. Fig-2.6 shows graphically the training loss during the iteration of training.

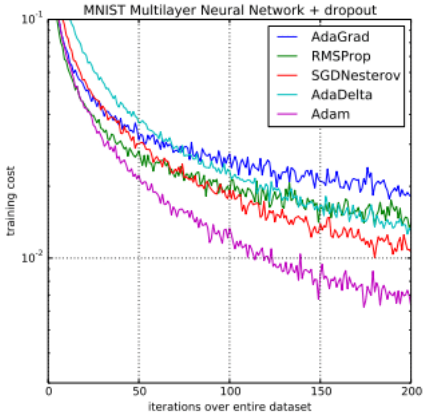


Fig-3.6: An illustration of training loss during training with different optimizers, Reference [31]

The most popular optimizer in deep learning is the ADAM one.

Despite this, you can train a deep-learning model using the abovementioned optimizers. As an illustration, the gradient algorithm can be used to minimize the loss function,

$$\omega_i(t+1) = \omega_i(t) + \Delta\omega_i(t), \Delta\omega_i(t) = -\eta \frac{\partial L}{\partial \omega_i}|_{\omega(t)}$$

where the learning rate is indicated by η, $\omega_i(t+1)$ represents the weight ω_i at iteration t + 1, and $\omega_i(t)$ represents the weight ω_i at iteration t. This algorithm specifies how to modify the weights to reduce the loss function. The backpropagation algorithm is widely used to calculate the gradient $\Delta\omega_i(t)$, which is shown in the above equation. The input is first forwarded through the network to produce the corresponding output. The loss value can be determined by contrasting this output with the corresponding ground truth data. This loss value is considered as the backpropagation algorithm moves backward (from the last layers to the first layers) and employs the chain rule to determine the contribution of each parameter to the loss. Since the weights

are updated in each iteration following the optimizer selected to solve the problem, the backpropagation algorithm enables the calculation of $\Delta\omega_i(t)$.

Weight initialization: The network's weights need to be given some values. As a result, a loss value can be calculated in the first iteration after forward passing the input through the network; this loss value will initially be high because the weights were initialized at random. It has been demonstrated that specific initializations can increase the stability of the training process. As a result, several initialization techniques for weights have been suggested, including Xavier initialization, Gaussian random initialization, and uniform random initialization.

Training, validation, and testing of CNNs: In creating any machine learning model, including Convolutional Neural Networks (CNNs), training, validation, and testing are crucial. The process of tuning a CNN's parameters—namely, its weights and biases—on a training dataset is called training. Testing assesses the model's final performance on a dataset that hasn't been used before, while validation involves using a different dataset to gauge the model's performance during training.

The CNN gains the ability to recognize patterns in the input data that are pertinent to the task at hand (for example, classifying images) during training. The error between the predicted output and the actual output is propagated backward through the network in a process known as backpropagation, and the weights and biases are updated accordingly using an optimization algorithm like stochastic gradient descent.

Typically, validation is done once the network has seen the entire training dataset or after each training epoch. The model's generalization performance, or how well it can generalize to new data, is evaluated using the validation dataset. The training can be stopped, or the model can be changed to prevent overfitting if the model is overfitting to the training data (i.e., performing well on the training data but poorly on the validation data).

After training, a different testing dataset assesses the model's final performance. This dataset shouldn't have been used for training or validation and should accurately represent the data the model will encounter in the real world. An estimation of the model's propensity to perform well in the real world is given by how well the model performed on the testing dataset.

3.9. Some CNN Architecture for Image Segmentation

The convolutional neural network cannot be used directly for image segmentation. The task of image classification is one where CNNs are frequently used. There are several ways to change the network to be

appropriate for segmentation. SegNet, sliding windows, and fully convolutional networks are among the successful ones.

Sliding windows: The sliding window was one of the earliest segmentation techniques using CNNs. Each pixel is assigned a classification based on evaluating the surrounding, closely cropped area [32]. Accurately estimating the size of the cropped part is difficult. The amount of contextual information and computational requirements must be balanced. The computation takes too long if the cropped area is too large. If the area is too small, not enough information is contained there. The sliding window method is less effective and computationally expensive than the current methods.

Fully Convolution Network: An end-to-end convolutional network architecture for segmentation is introduced by the Fully Convolutional Network (FCN) [33]. The network output in this architecture has the same dimension as an input because there are no pooling layers. The model directly generates the labeled image. The final layer contains a 1-1 convolution that assigns a one-hot classification vector to each pixel.

A successful concept for creating a neural network for segmentation is end-to-end learning. The simplicity of the model is a benefit of this strategy. The entire segmentation pipeline is replaced by one method. A higher probability of an error results from more computation steps, and this probability increases. On the other hand, the end-to-end architecture is only appropriate if there is sufficient data to train it properly.

SegNet: This architecture was developed by a team led by V. Badrinarayanan[34] to lower FCN's computational costs and increase the network's generalization capacity. It is a member of a group of end-to-end architectures. It employs an up-sampling neural network layer, which performs the pooling operation in reverse. This layer can expand the feature map's dimension without any learnable variables.

The SegNet topology is reminiscent of the auto-encoder neural network type. The encoder and decoder are the two symmetric components that make up SegNet. The encoder component makes use of a typical CNN layer structure. Convolutional layers come first, then pooling layers. Convolutional and upsampling layers are interlaced in the decoder portion to increase a feature map's dimensions to the original image.

3.10. Some Case Studies of contemporary convolutional networks:

This subsection will briefly introduce some well-known examples of modern convolutional networks. They all participated in the Large Scale Visual Recognition Competition (ILSVRC), which has been going on since

2010 [35]. It ranks among the most significant difficulties in this area. Classifying digital photos into tens of thousands of categories is the competition's task. The previously discussed architectures are still used for other computer vision tasks.

AlexNet: This network won the 2012 ILSVRC competition [36]. It was the first network to demonstrate that deep learning can be as effective as conventional image analysis methods. The network's relatively straightforward architecture is depicted in Fig. 2.7. Three fully-connected layers come after the first five max-pooling convolutional layers. Convolutional and pooling layers have typically been used in CNNs until now.

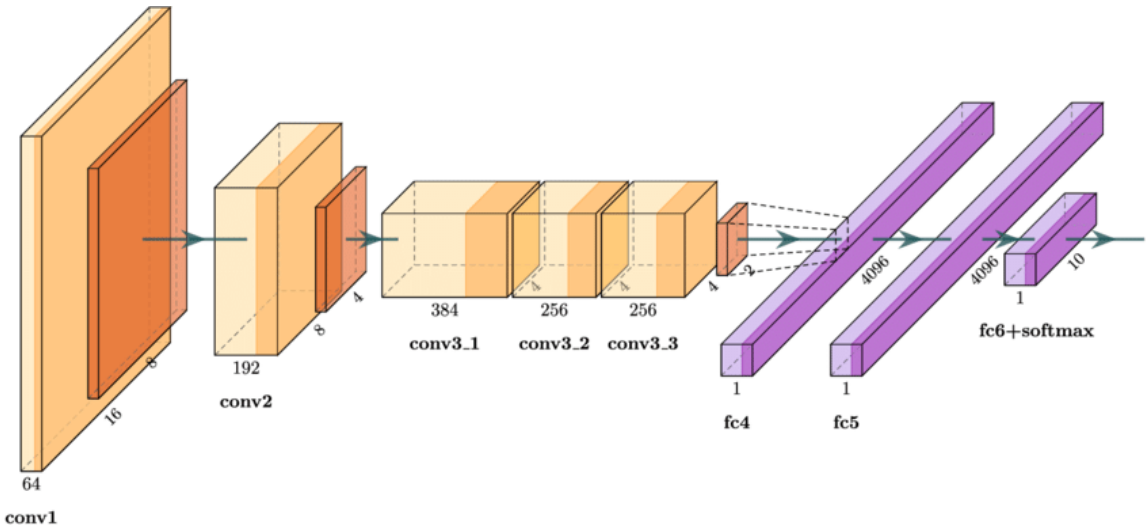


Fig-3.7: An illustration of AlexNet architecture, Reference [60]

GoogLeNet: This network, created by Google [37], won the ILSVRC-2014 with a 93.3% accuracy rate. The high complexity of this network architecture is a defining feature. It introduced a brand-new construction element called the Inception Module. The architecture of the network is shown in fig-2.8. The module is made up of several tiny convolutional networks that are computed concurrently. The computation is successful as a result of sharing some computational phases. A concatenation of the module's outputs constitutes its final response. The authors provided evidence that this strategy produces excellent outcomes. The Inception module enables complex feature recognition without deepening the network.

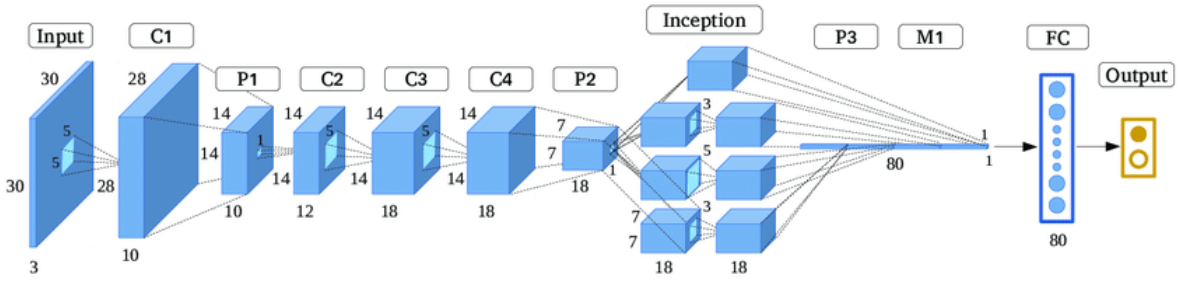


Fig-3.7: An illustration of GoogLeNet architecture, Reference [61]

ResNet: The ResNet achieved 96.4% accuracy in ILSVRC in 2016. It is a Microsoft [38] product with 152 layers stacked in leftover blocks. The residual block schema is depicted in Figure 2.8. This block consists of a collection of layers connected by a skip. The block layers can learn an identity function thanks to this connection. As a result, if a layer in the block does not provide any new information, it can be skipped. This method enables the training of extremely deep networks. It avoids serious network-specific issues, such as high variance and vanishing gradient.

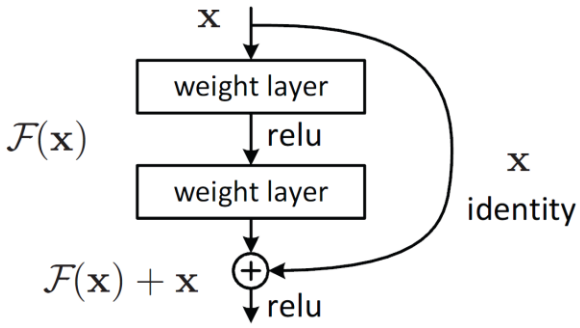


Fig-3.8: An illustration of the residual block, Reference [62]

In recent years, the U-Net and the Mask R-CNN are the two architectures that have been most frequently used to address the issue of cell nuclei segmentation. In summary, because U-Net is built for semantic segmentation, it cannot distinguish different instances of the same object. Mask R-CNN is intended for segmentation, but because of its more complex architecture than the U-Net, it typically requires more computing power. On the one hand, the U-Net-based approaches need additional techniques to distinguish touching nuclei and resolve the issue as an instance segmentation issue.

4. METHODOLOGY FOR SEGMENTATION

As previously stated, this project's main goal is to segment the overlapping cells of microscopy images properly. This section explains the methodology used to segment the overlapping cells accurately and then gives a brief overview of transfer learning and Mask-RCNN[39]. We have done this segmentation process of overlapping cells in two stages which we will discuss in further sub-sections.

4.1. Transfer Learning

Transfer learning is a deep learning technique that avoids training a model from scratch by using pre-trained models as a starting point for new tasks. It has grown in popularity recently due to its capacity to reduce training time and enhance model performance.

The pre-trained models have learned features that are helpful for various tasks because they have been trained on large datasets. The pre-trained model's upper layers detect more complex features like shapes, textures, and patterns, while the lower layers detect more fundamental features like edges, lines, and curves. We can take advantage of these learned features by using a pre-trained model as a starting point and train a model to carry out a new task with less training data and in less time.

The two primary types of transfer learning are feature extraction and fine-tuning. By changing the weights of all the layers in the model, including the pre-trained layers, we can train a previously trained model on a new task during fine-tuning. This method performs well when the new task is similar to the previously trained task. In contrast, feature extraction entails training a new classifier on top of the extracted features and using the pre-trained layers as a fixed feature extractor. This method performs well when the pre-trained model has acquired features helpful for the new task.

The dataset is divided into three sets for training: training, validation, and testing. The validation set is used to adjust the hyper parameters and avoid overfitting, the training set is used to train the model, and the test set is used to assess the model's final performance. The model's performance is checked using the validation set, and the hyper parameters—such as learning rate, batch size, and optimizer—are changed to get the best results. Overfitting can happen when a model is trained excessively well on the training set but performs poorly on the validation and test sets. Techniques like regularization and early stopping can be used to prevent overfitting.

In conclusion, transfer learning is a powerful method that can be applied to shorten training times and enhance model performance. We can train a new model on a new task with less training data and in less time by utilizing the learned features of a pre-trained model. The similarity between the pre-trained and new tasks determines whether to use feature extraction or fine-tuning. The dataset is divided into three training, validation, and testing sets. The validation set is used to adjust the hyper parameters and avoid overfitting.

For this project, we use transfer learning from the pre-trained weight of the MS-COCO dataset[41] which is a good starting point for training a new deep-learning model. The Microsoft Common Objects in Context (MS-COCO) dataset[41] is frequently used in computer vision and image recognition tasks. More than 2.5 million labeled object instances are spread across over 330,000 images in over 80 object categories. Due to the dataset's wide range of object sizes, shapes, and occlusions, it is not easy to analyze.

4.2. Mask R CNN

A framework for segmenting instances called Mask R-CNN[39] detects objects while creating segmentation masks for every instance. It improves on Faster R-CNN[42] and Fast R-CNN[43], its forerunners: Fast R-CNN uses features extracted from candidate bounding boxes to perform classification and bounding-box regression; Faster R-CNN adds a previous stage known as a region proposal network; and Mask R-CNN additionally produces an object mask.

Mask R-CNN is an effective and adaptable deep learning architecture for object detection and instance segmentation tasks. It is a well-liked option for transfer learning in these tasks due to its capacity to utilize pre-trained weights on the MS COCO dataset.

The backbone network and the mask-head network are the two main components of the architecture. The backbone network, usually a pre-trained convolutional neural network (CNN) like ResNet, Mobilenet, or Inception, performs feature extraction. The mask-head network is a small network that outputs a binary mask for each object proposal using the features from the backbone network. In the next section, we will discuss the backbone.

4.2.1. Backbone

Mask RCNN uses a convolutional backbone architecture to compute feature maps from an input image. The core element of Mask R-CNN, which

is in charge of feature extraction and representation learning, is known as the backbone, and it is extremely important to the model's overall performance. A pre-trained convolutional neural network (CNN) trained on a sizable dataset, such as ImageNet, is typically used as the feature extractor in Mask R-CNN.

In most cases, the backbone comprises several convolutional and pooling layers that gradually reduce the spatial resolution of the input image while increasing the number of channels, or features, extracted from the image. A feature map with high-level semantic information about the image is the backbone's output, and the model's subsequent layers use it to carry out object detection and segmentation.

Different types of backbone architecture are present for Mask R-CNN, such as resNet, Mobilenet, Inception, etc. But resnet family architecture is the most commonly used architecture for Mask RCNN, introduced in 2016 by He et al. [44] with feature pyramid network (FPN).

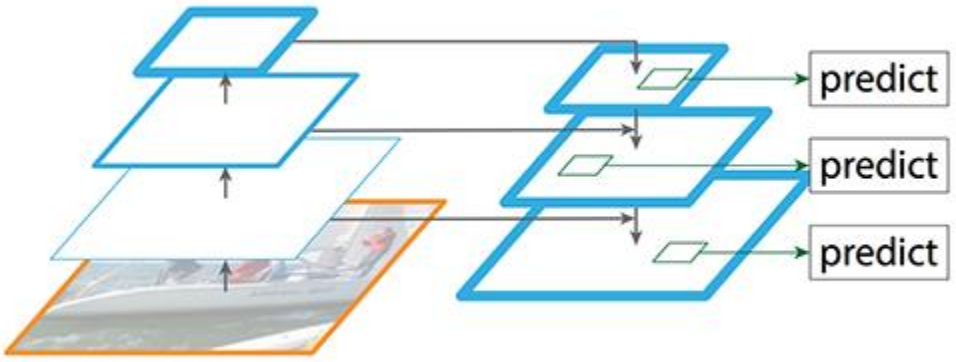


Fig-4.1: FPN Architecture. Reference [45]

A bottom-up pathway, a top-down pathway, and lateral connections comprise the FPN. The convolutional backbone's feedforward computation, which divides the CNN into stages based on the output size of each layer, is the bottom-up pathway. It samples the final output of each stage as the feature map for that scale, resulting in a hierarchical feature pyramid. Figure 4.1 depicts the qualitative architecture and path directions of the FPN. Create feature maps C2, C3, C4, and C5 in ResNet using conv2 through conv5 with their respective strides of 4, 8, 16, and 32 pixels, corresponding to a scaling step of 2. With a scaling step of 2, the top-down pathway interpolates the semantically more robust high-level feature maps to create higher-resolution low-level features. They are improved by 1x1 convolutional layers connected laterally to the corresponding levels of the bottom-up feature pyramid. The 1x1 convolution is applied to the coarsest feature map in the bottom-up

pathway to start the pathway, and the interpolation is repeated until a feature pyramid (P2, P3, P4, P5) is produced.

Mobilenet architecture (Shown in fig-4.2) is also used as the backbone network for various computer vision tasks performed by Mask RCNN. The foundation of MobileNets is a set of depth-separable convolutional layers. A depthwise convolution and a pointwise convolution make up each depthwise separable convolution layer. A MobileNet has 28 layers if depthwise and pointwise convolutions are counted separately.

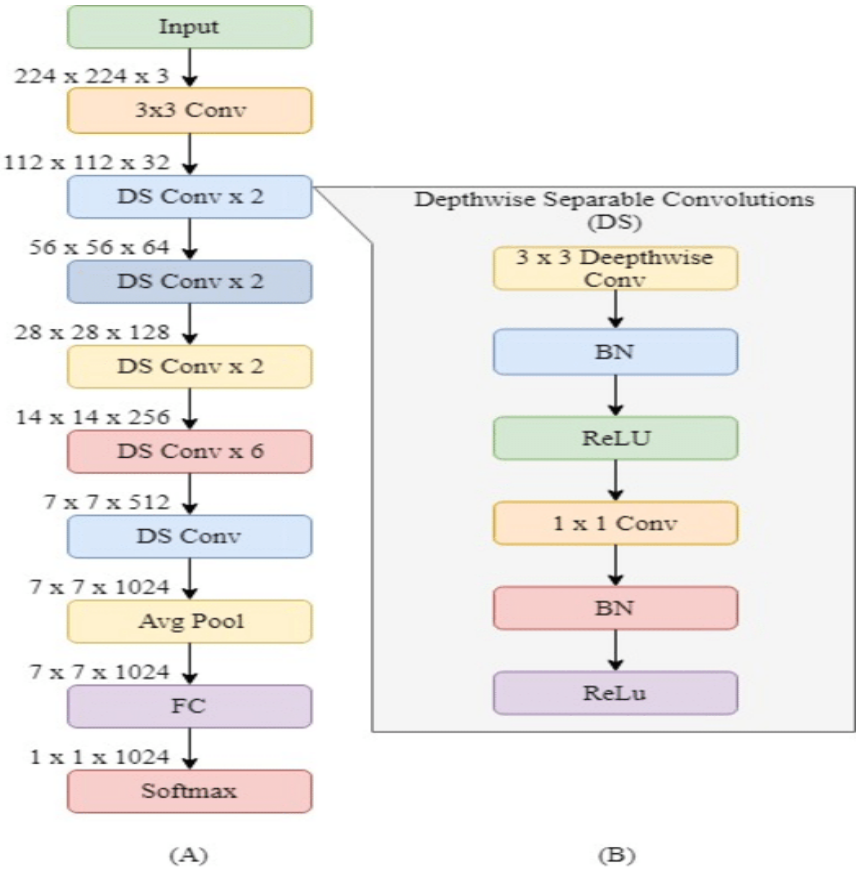


Fig-4.2: (a) An illustration of Mobilenet architecture. (b) Explanation of DS Layer. Reference [63]

4.2.2. Region Proposal Network

Region Proposal Network (RPN) is used to extract the region of interest (rois), which was introduced in Faster RCNN [42]. The RPN is a CNN that slides a mini-network over the feature map, where two sibling 1x1 convolutional layers for box-regression and classification are followed by a n x n convolutional layer that takes a spatial window of the same size. Regions

of interest (RoIs) or candidate bounding boxes can be found on each pyramid level.

In Fast RCNN, a Roi with w width and h height can be assigned to a level P_k using,

$$k = [k_0 + \log_2 \frac{\sqrt{wh}}{224}]$$

Where 224 is the canonical ImageNet pre-training size, for Mask RCNN, $k_0 = 4$.

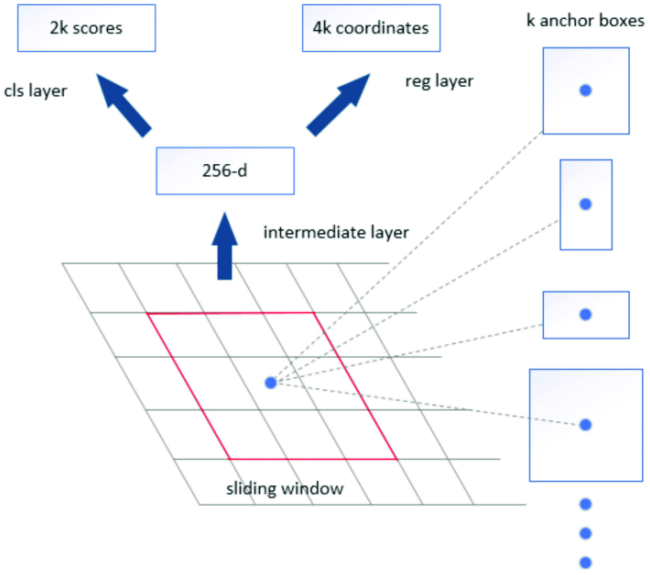


Fig-4.2: Region Proposal Network (RPN). Reference [46]

The Region Proposal Network produces a set of rectangular proposal boxes with object scores from an image of any size as input. A small network was moved onto the convolutional map of the most recent shared convolutional output to generate regional proposals. This network accepts as input a n x n spatial window of a convolutional feature map. Convolution mapping produced a 512-dimensional feature vector from a 3 x 3 sliding window. The architecture of the region proposal network used in this work is shown in Figure 4.2. Multiple regional proposals were generated at each sliding window's location and corresponded to different scales and aspect ratios on the current sliding window. As a result of the use of three scales and three aspect ratios, k=9 regional proposals were produced for each slide location. The suggestions are also known as anchors. The two fully connected layers of the bounding box regression and the bounding box classification layer were then filled with the regional proposals.

4.2.3. ROI Align

In the detection and segmentation-based tasks, Region of Interest Align, or RoIAlign, is an operation for extracting a small feature map from each RoI. As a result, the extracted features are correctly aligned with the input, and the harsh quantization of the RoI Pool is removed. RoIAlign employs bilinear interpolation to determine the precise values of the input features at four regularly sampled locations in each RoI bin without any quantization of the RoI boundaries or bins (using x/16 instead of [x/16]), and the result is then aggregated (using max or average) [39].

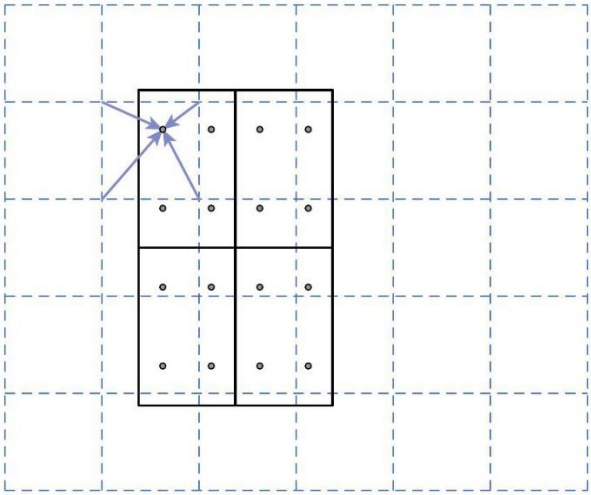


Fig-4.3: RoIAlign: The dashed grid represents a feature map, the solid lines of RoI (with 2x2 bins in this example), and the dots of the four sampling points in each bin. Reference [39]

4.2.4. Loss Function

The multi-task loss function of Mask R-CNN combines the loss of classification, localization, and segmentation mask.

$$L = L_{cls} + L_{box} + L_{mask}$$

where L_{cls} and L_{box} are the same as in Faster R-CNN [42].

L_{cls} is the classification loss, which is a log loss function defined as,

$$L_{cls} = \frac{1}{N_{cls}} \sum_{i} -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)$$

Where the sum is over the number of anchors, N_{cls} is the mini-batch size, p_i^* is the ground truth label of whether anchor i is an object and p_i is the predicted probability of anchor I being an object.

L_{box} is the bounding box loss which sums over the co-ordinates of all bounding boxes, which is defined as,

$$L_{box} = \sum_{i} L_1^{smooth}(t_i^u - v_i)$$

Where u is the class label, t_i is the predicted bounding box, and v_i is the ground truth bounding box. L_1^{smooth} is the smooth L1 loss function (shown in fig-4.4), which is defined as,

$$L_1^{smooth} = \begin{cases} 0.5x^2, & if\ |x| < 1 \\ |x| - 0.5, & Otherwise \end{cases}$$

L_{mask} is the mask loss which is the average binary cross-entropy loss, and that is defined as,

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} [y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log 1 - \hat{y}_{ij}^k]$$

where y_{ij} is the label of a cell (i, j) in the true mask for the region of size m x m, $\hat{y}_{ij}^k$ is the predicted value of the same cell in the mask learned for the ground-truth class k and y_{ij} is the label of a pixel (i, j) in the ground truth mask for the size m x m region.

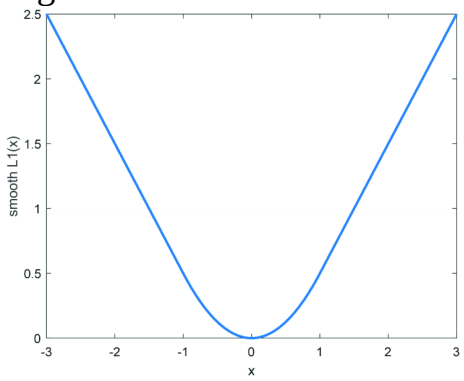


Fig-4.4: Curve of the smooth L1 loss. Reference [47]

4.3. Methodology

This section describes the methodology for nuclei segmentation, which is used for this project work. The entire flowchart of the proposed work is shown in Fig 4.5. In the following section, each of the components of the network is explained elaborately. For the first stage of the methodology, we discuss coarse segmentation, which is done by Mask R-CNN [39], and in the second

stage of the methodology, we talk about fine segmentation of nuclei which is done by a Euclidean distance transform [40] and watershed segmentation. The coarse segmentation map generated by Mask R-CNN successfully segments most of the overlapping cells except for some complex overlapping cells sent to the second phase for fine segmentation. We have done the second phase with a novel technique for marker generation to be used by the Euclidean distance transform for fine segmentation of the complex cells not segmented in the first phase.

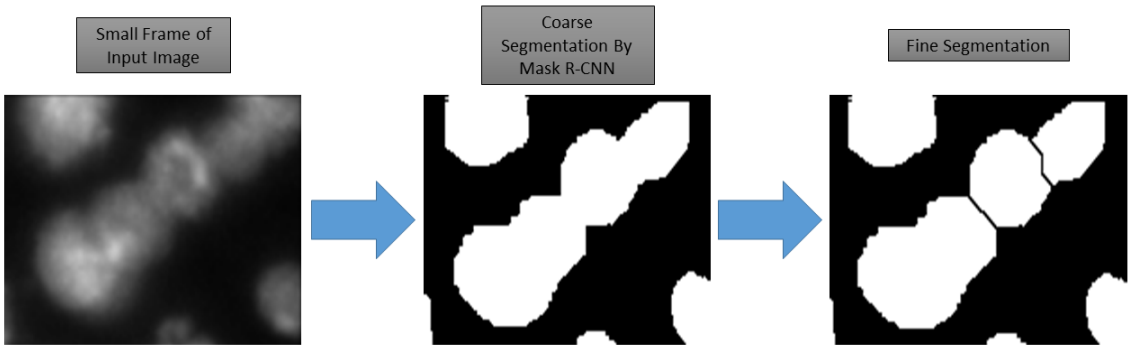


Fig-4.5: Workflow of this methodology

4.3.1. Coarse Segmentation

ResNet-101 [39] is utilized as the Mask R-CNN [39] backbone network to produce feature maps from training images, as shown in Figure 4.6. As a result, the ResNet-101 network processes the input image, which has first been downsized to 512*512 pixels, and the network extracts feature maps from the image. A convolutional layer with a kernel size of 7x7 is used by ResNet-101 at level 1, followed by a batch normalization layer (BN) and a ReLu activation function. The remaining blocks, composed of two convolutional layers, a BN layer, and a ReLu activation function, are stacked at each stage. The number of learnable kernels increases by powers of two for the remaining blocks, with the first convolutional layer's kernel size being 1x1, and the second's kernel size being 3x3. The network learns basic cell features in the first few layers, like subsequent residual blocks, edge detection, and advanced cell features. A region proposal network (RPN) routes feature

maps created from the backbone network. RPN uses a sliding window method to search for features on his map to create a set of anchors. Bounding boxes of various sizes and aspect ratios are positioned all over the feature map as anchors. Set the size of the anchor box to be small because the core is typically not very large so that the model can extract the feature map more precisely. Then, regardless of size or aspect ratio, ROI Align extracts a fixed-size feature map from each traversed candidate bounding box. Mask R-CNN then predicts the ground truth bounding box offsets of object instances and proposals by using a fully connected network (FCN) on each proposal. A more precise bounding box results from applying these offsets to the bounding box of the initial proposal. Using classification headers, the bounding box's content is divided into categories. The classification header directs This feature map to several predefined classes (core and background). To produce a binary mask for each bounding box that identifies which pixels inside the bounding box belong to the object of interest, the mask head processes the feature map for each bounding box and applies four convolution layers. binary classification) to divide images into core and non-core classes based on some threshold.

The learning rate for this Mask R-CNN model is 0.001 for training the network head and 0.0001 for training all network layers. Since the nucleus is typically not large, we set the anchor box sizes small so the model can extract the feature map more precisely. The obtained loss error values have also been considered when maximizing the number of epochs. The proposed methodology uses two types of losses: classification loss and mask loss. To improve instance segmentation accuracy between the actual mask and the predicted mask, the classification loss, a cross-entropy loss, is used to optimize the accuracy of object detection, along with the mask loss, a binary cross-entropy loss.

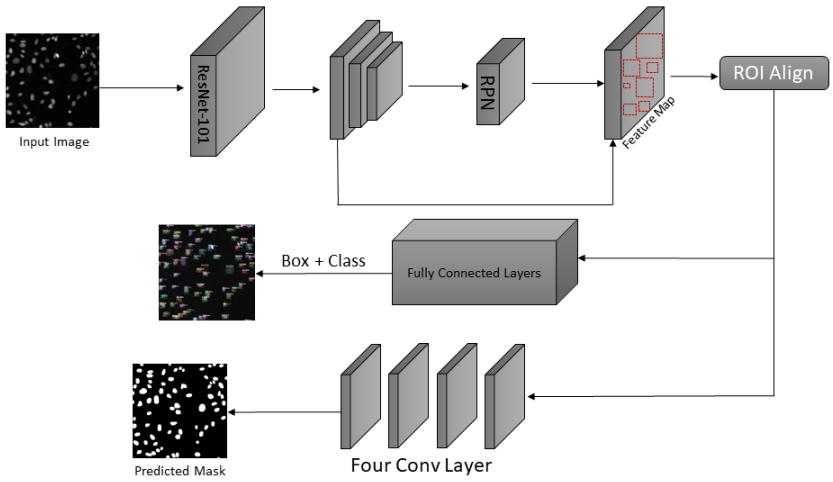


Fig-4.6: An illustration of the coarse segmentation by Mask RCNN

Except for a few complex cells incorrectly segmented during this phase, the segmentation map created after coarse segmentation separates most of the overlapping cells. As a result, after area-based thresholding, which segments the complex overlapping nuclei that were not segmented in the first phase, these complex nuclei clumps were sent to the second phase for fine segmentation.

4.3.2. Fine Segmentation

In this phase of this nuclei segmentation work, we will fine-segment the overlapping nuclei from the predicted binary mask by Mask R-CNN, which cannot segment very complex overlapping nuclei.

To begin with, we use Binary distance transform (BDT) [40] on the binarized images, which are generated from the coarse segmentation part.

Binary distance transform (BDT) is the distance transform that refers to the transformation of a binary image, let b_i to a distance map λ where the value of λ for each pixel corresponds to the distance between that pixel and the closest background.

Let B be the set of foreground pixels and $\overline{B}$ be the set of background pixels and $\subset b_i$. Then the BDT of the set B can be written as $\{(p, \lambda_B(p)),\ \forall p \in B\}$ on $\mathbb{R}^2$, where $\lambda_B(p) :\ \mathbb{R}^2 \rightarrow \mathbb{R}$ is the BDT value at p, that is given in below equation,

$$\lambda_B(p) = \min\{dis(p,q), \forall\ q \in\ \overline{B}$$

Here for distance metric, we use Euclidean distance, and dis(g, b) is the Euclidean distance between p and q, and Euclidean distance is measured by the given below formula,

$$dis(p,q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where (x_1, y_1) and (x_2, y_2) are the coordinates of p and q respectively.

Now, A point $p \in b_i$ is said to be a local maxima point in distance transform field λ, if $\lambda(p) > \lambda(q)$ $where\ q \in H^*(p)$ where $H^*(p)$ is the 8-adjacent neighbors of p excluding p.

But local maxima will generate more than one marker point, leading to over-segmentation; for this problem, we need to extend the maxima transform. Extended maxima determine the regional maxima of the H-maxima transform. H-maxima will delete any maxima in λ that has a depth below or equal to a set threshold μ. This H-maxima transformation reconstructs λ by the given below equation,

$$M^{\mu}_{max}(\lambda) = R^{\sigma}_{\lambda}(\lambda - \alpha)$$

Here reconstruction and dilation are represented by R and σ. The extended maxima transform is given below,

$$E^{\mu}_{max}(\lambda) = R_{max}(M^{\mu}_{max}(\lambda))$$

Where, R_{max} is the regional maxima of the H-maxima transform.

After getting the H-maxima point, we use this point as a marker point and fine segment by watershed segmentation [64].

Therefore, in the coarse segmentation stage done by Mask RCNN, some complex cell structure may not be segmented well, so we may be unable to differentiate all overlapping nuclei. To avoid this mistake, we fine-segment those complex overlapping nuclei by the Euclidean distance transform method, making the segmentation method very accurate. The next chapter will evaluate this segmentation method and compare this results with popular cell segmentation methods.

5. EXPERIMENTAL RESULTS AND DISCUSSION

This project section will contain the results and discuss the cell segmentation method.

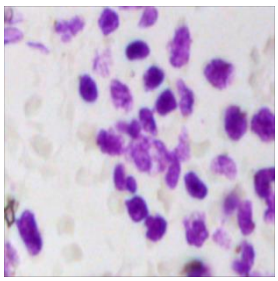
5.1. Computational Environment

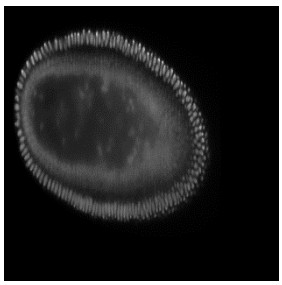
Tensorflow [48] and Keras [49], two open-source deep learning libraries, are the foundation for all deep learning implementations. A high-level Tensorflow application programming interface (API) is called Keras. These frameworks compute backpropagation and automatically update the parameters. On Tensorboard, a set of visualization tools for TensorFlow programs, the validation loss, and other metrics (validation accuracy, validation F1 score, etc.) were tracked. Additionally, every experiment was run in Python 3.5 on a workstation with an Intel Xeon Processor, 128 GB of Ram, and NVIDIA Quadro P5000 GPU of 16GB power.

5.2. Database Preparation

We used the Kaggle 2018 Data Science Bowl dataset for this project. The dataset included 670 microscopic images with associated ground truth using various magnifications and modalities (bright field and fluorescence microscopy), as shown in fig-5.1. Experts in the field created the binary mask for each of the images. The nuclei in each image in the dataset were numerous. The images varied in imaging modality, cell type, and magnification. We randomly divided the dataset into three sections for training, validation, and testing. Training, validation, and testing are each given a weight of 70%, 20%, and 10%, respectively. Therefore we trained this model with 469 images, used 134 images for validation, and for testing purposes, we used 67 images.

Original Image





Ground truth

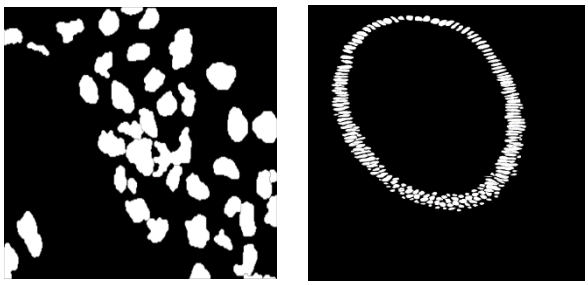


Fig-5.1: Examples of the dataset images

5.3. Data Pre-Processing

The dataset has undergone two preprocessing steps. Various augmentation techniques were used to increase the number of training samples in the dataset and resize all input images to 512*512 pixels.

If the training dataset is too small, a technique known as data augmentation is frequently used to enhance the neural network's performance. The objective is to produce fresh data with the same probability distribution as test samples. Additionally, this procedure is used to expand a set of features in a training dataset by creating samples with new features. The majority of augmentation techniques rely on giving existing samples random modifications.

To improve training datasets, many image transformation techniques are used. We must select those that are appropriate for the task at hand. We decided to add all these transformations to the training dataset because cells don't have a fixed orientation, their shapes vary, and convolution isn't equivariant to basic geometric transformations like rotation, mirroring, and size changes. We augment the dataset using the image data generator (from the Python library).

5.4. Mask R-CNN Training Configuration

We use Mask R-CNN with ResNet-101 as the backbone to train this model in the coarse segmentation stage. We used 10 epochs to train the head layer only, and after that, we trained the model with another 20 epochs with all layers. We use 2 batch sizes and steps per epoch at the total trainable image divided by batch number. We set detection min confidence with 0.5 and set Region Proposal Network (RPN) anchor size (8, 16, 32, 64, 128) smaller than the standard usage size because the nuclei size is typically small. We set the learning rate at 0.001 when training the head layer and 0.0001 at the all-layer training part.

5.5. Training and Validation Loss

We track the training and validation loss of the model using TensorBoard. In fig-5.2, we show the overall training and validation loss which is calculated by class loss, box loss, and mask loss, i.e., $L = L_{cls} + L_{box} + L_{mask}$.

In fig-5.3, we show the class training and validation losses which are training class loss and validation class loss.

In fig-5.4, we show the box training and validation losses which are training box loss and validation box loss.

In fig-5.5, we show the mask training and validation losses which are training mask loss and validation mask loss.

And for all figures last 20 epochs, training and validation losses will be shown in the graph.

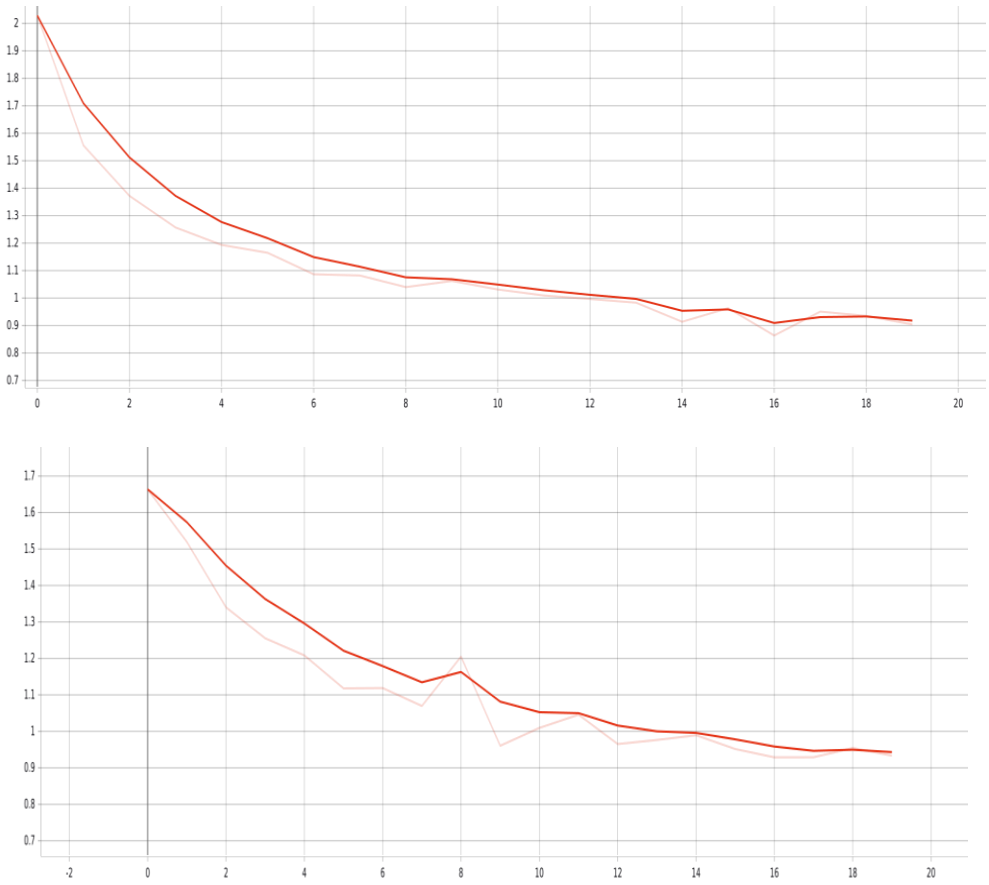


Fig-5.2: Overall training and validation loss

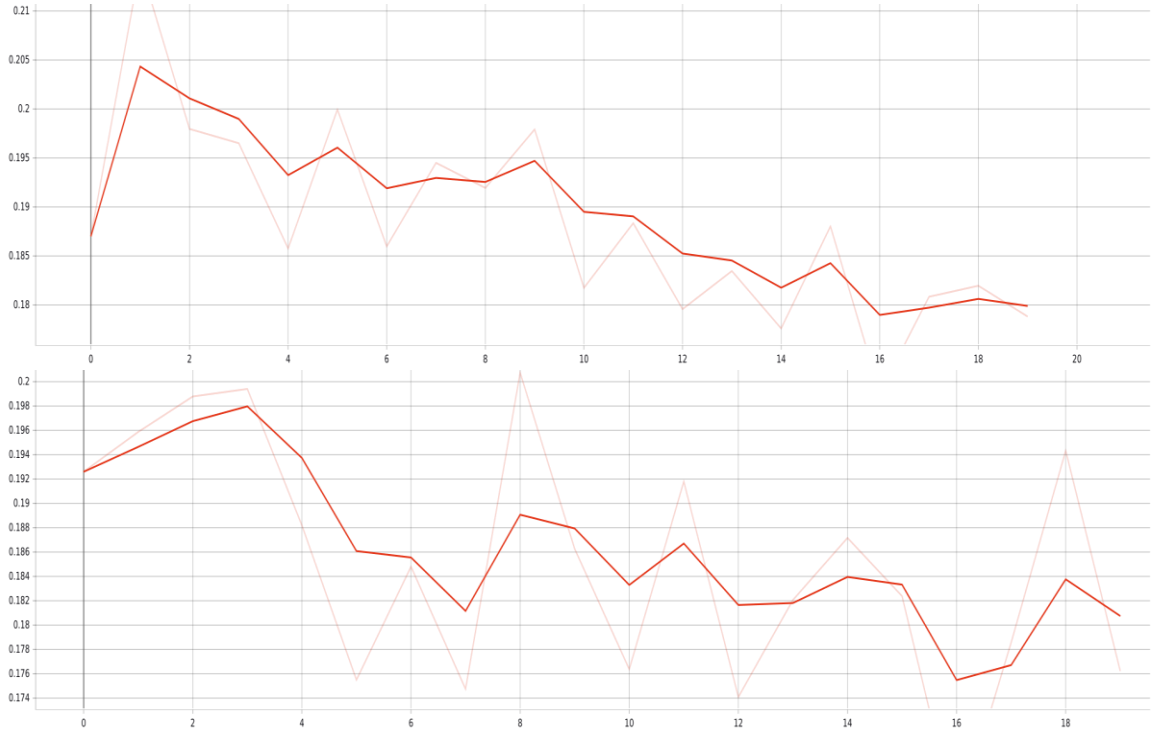


Fig-5.3: Class loss of training and validation

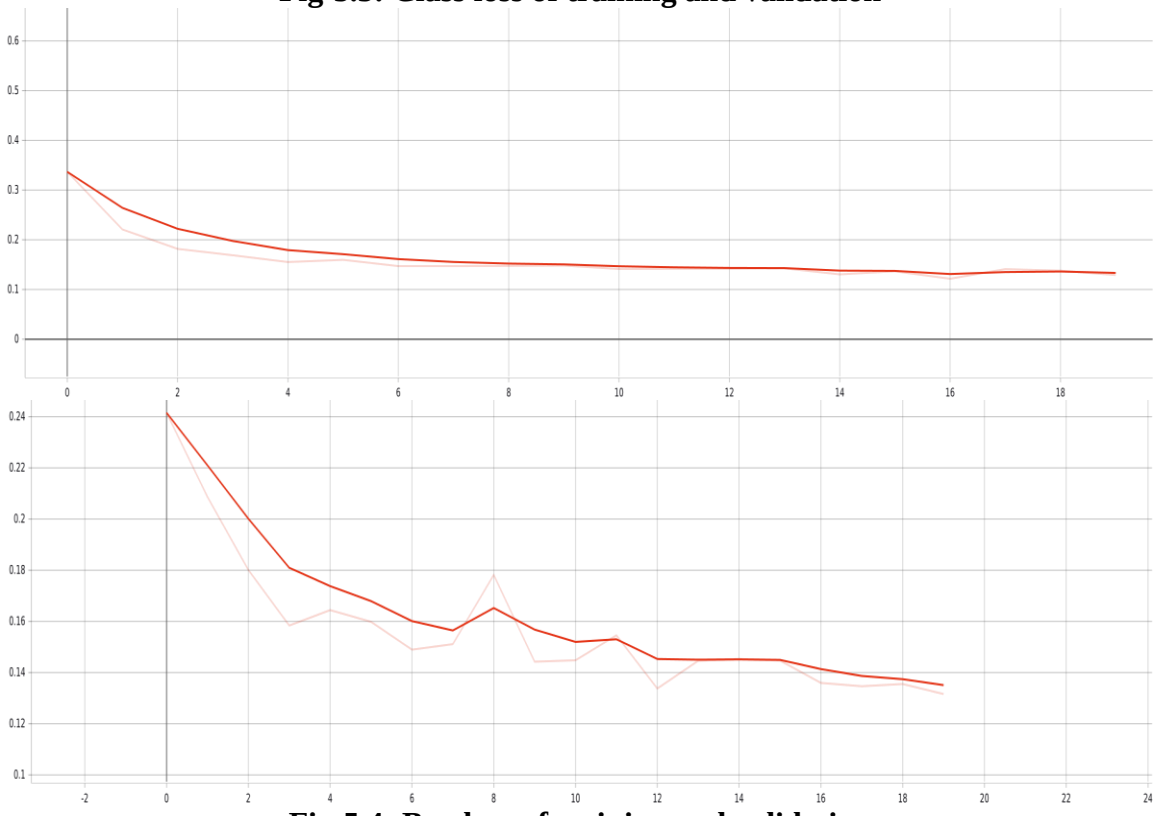


Fig-5.4: Box loss of training and validation

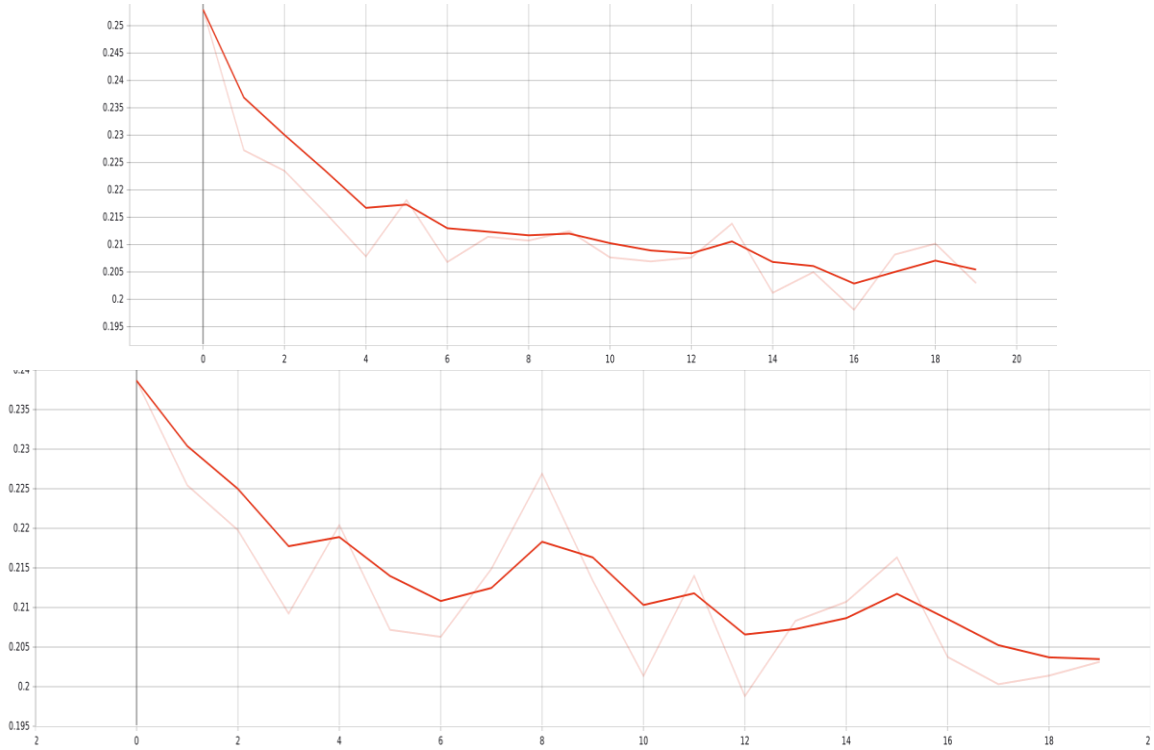


Fig-5.5: Mask loss of training and validation

5.6. Quantitative and Qualitative Evaluation

5.6.1. Quantitative Evaluation

Dice Score (DSC) [50], Aggregated Jaccard Index (AJI) [51], Precision (Pre) [50], and Recall (Rec) [50] have all been used to assess the effectiveness of this suggested method for segmenting nuclei. I have contrasted the effectiveness of the suggested framework in Table 1 with two standard deep learning methods, Mask R-CNN [39] and U-Net [15]. Because they are so well-liked in nuclei segmentation, these two deep-learning techniques were used for comparison. The performance of U-Net compared to the other methods is shown in Table 5.1 as being the lowest. Compared to the suggested method, Mask R-CNN performs better than U-Net but still receives poor scores. This is primarily because most deep learning techniques succeed at segmenting simple overlapping clumps but fall short of complex clumps. Thus, in this work, we had a second phase for segmenting those clumps that Mask R-CNN did not segment in addition to the coarse segmentation phase, which primarily used Mask R-CNN. Each cell in a clump received a marker from the novel marker generation step suggested in work, which the marker-

controlled watershed algorithm used to segment the cells that Mask R-CNN had missed. Thus, this suggested approach performs better than both SOTA deep learning approaches. Additionally, this method demonstrates the framework to be reliable and consistent, which maintains a trade-off between Precision and Recall.

Table 5.1. Quantitative Evaluation of Cell segmentation results. The best results are in the Bold.

Dataset	Models	Segmentation Performance			
		DSC↑	AJI↑	Pre↑	Rec↑
Kaggle 2018 Data Science Bowl	U-net [15]	0.786	0.533	0.831	0.775
	Mask R-CNN [39]	0.829	0.651	0.857	0.808
	Proposed Method	**0.883**	**0.703**	**0.898**	**0.872**

Figure 5.6 further illustrates the Dice Score (DSC) determined for each sample in the test set using the methods listed in Table 1. It is crystal clear from Fig. 5.6 that this proposed method (PM) outperforms the other deep learning models for most of the test images.

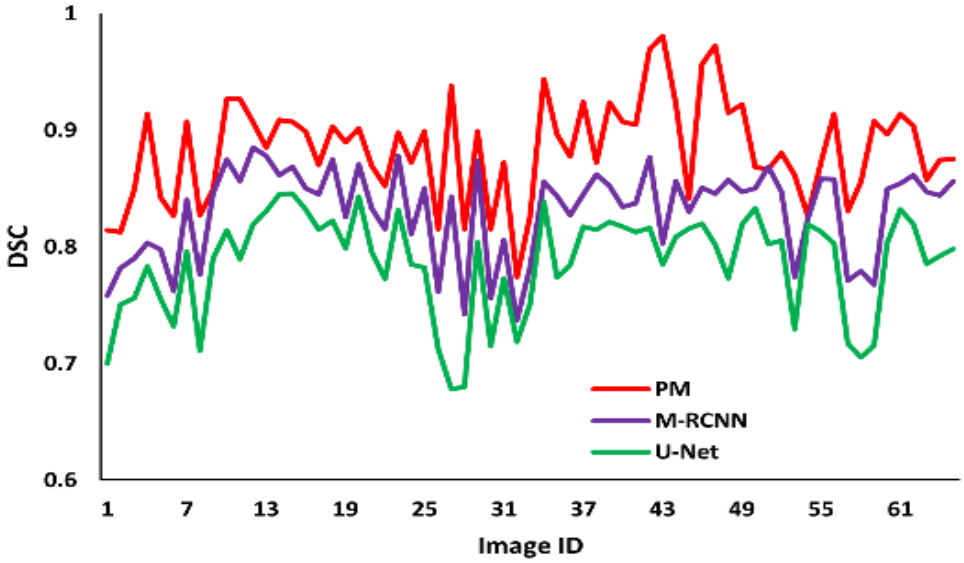
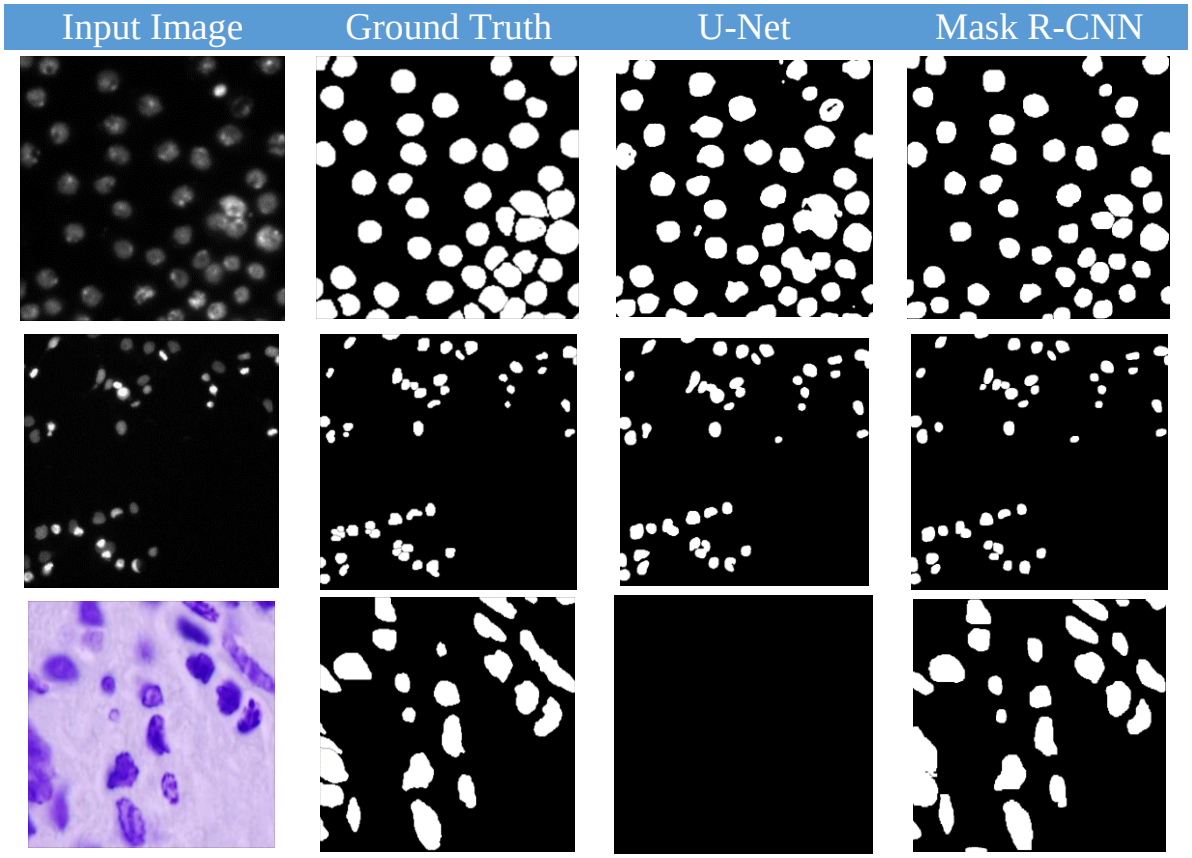


Fig-5.6: Comparative analysis of the method using Scatter Plot in terms of DSC

5.6.2. Qualitative Evaluation

In this part of the evaluation, we will show the visual difference between different cell segmentation methods with the cell segmentation which is done at the coarse segmentation stage by Mask R-CNN. We will compare visually predicted cell segmentation by U-Net architecture, Mask R-CNN with ResNet-101 architecture as the backbone, and ground truth. In fig-5.7, the first column is the input image with 512x512 resolution; the second column is the ground truth which is given in the dataset, the third column is predicted cell segmentation by U-Net architecture, and the last column is predicted cell segmentation by Mask R-CNN with ResNet-101 architecture. We can see that the nuclei boundaries are very accurate in the last column, i.e., by Mask R-CNN concerning the ground truth image, whereas in the third column, images are more inaccurate concerning the ground truth. In the last column, the maximum nuclei are also separated from each other (those very complex overlapping cells which Mask-RCNN does not separate will goes in the fine segmentation stage), which can tell that Mask R-CNN can segment the overlapping cells very accurately rather than U-Net architecture which is present in the third column.



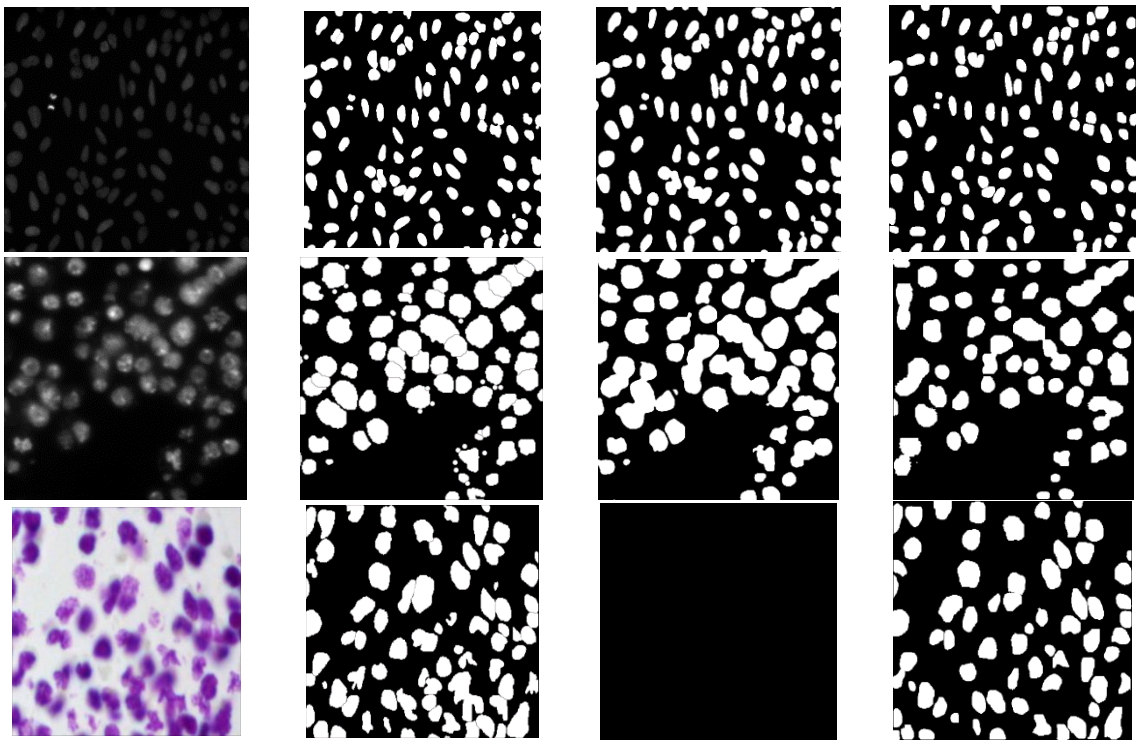


Fig-5.7: Visual difference between different segmentation methods

We can clearly see that the third and last row U-Net model does not predict any nuclei in that type of purple image, which is a big blunder since it is a very important process for medical treatments. But Mask R-CNN predicts very well in that type of image also. But in the fifth row, the input image has very complex overlapping nuclei, which we can see in the ground truth, But Mask RCNN struggles to segment all overlapping nuclei. In this case, we must apply the fine segment stage, shown in fig-5.8.

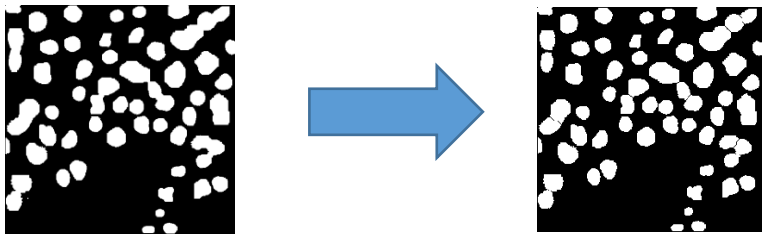


Fig-5.8: After fine segmentation by Marker Controlled Watershed of complex overlapped nuclei

6. CONCLUSION

Deep learning has exploded in the past ten years, greatly benefiting cellular biophysics. High-resolution single-cell detection in dense cell populations is made possible by instance segmentation frameworks. However, because nuclei are frequently clumped together without clearly delineating boundaries, creating an automated system for nuclei segmentation is very difficult. Modern techniques generally succeed in segmenting nuclei with slight overlap but fall short for complex clumps. In this project, we investigated Mask R-CNN's ability to find overlapping cells from microscopy images. And with that, we also added an extra stage to detect complex overlapping cells by using a marker-controlled watershed. An extensive experiment has been performed on the Kaggle 2018 Data Science Bowl dataset to evaluate the robustness of the proposed framework. After measuring the model's performance, we discovered that Mask R-CNN is reasonably adept at maintaining high mAP (>85%) at initial IoU values and generally does not produce false positives. The evaluation metrics obtained by the given method show the framework to be robust and consistent. In conclusion, given a small dataset, Mask R-CNN achieves respectable instance segmentation at both low and high cell densities. It is obvious that more training data of elongated and blurred cells and physically touching cells are required if a more robust model is desired. According to my findings, a much more reliable detector can be developed with advancements like image augmentations and a larger dataset. The next stage in developing an experimental pipeline is to track cancer cells in microscopy videos using statistical inference or CNN architectures.

REFERENCE

[1] R. L. Siegel, K. D. Miller, and A. Jemal, "Cancer statistics, 2019," CA. Cancer J. Clin., vol. 69, no. 1, pp. 7–34, Jan. 2019, doi: 10.3322/CAAC.21551.

[2] [3] E. A. Rakha et al., "Breast cancer prognostic classification in the molecular era: The role of histological grade," Breast Cancer Res., vol. 12, no. 4, pp. 1–12, Aug. 2010, doi: 10.1186/BCR2607/TABLES/2.

[3] H. Irshad, L. Montaser-Kouhsari, G. Waltz, O. Bucur, J. Nowak, F. Dong, N. W. Knoblauch, and A. H. Beck, "Crowdsourcing image annotation for nucleus detection and segmentation in computational pathology: evaluating experts, automated methods, and the crowd," in Pacific symposium on biocomputing Co-chairs. World Scientific, 2014, pp. 294–305.

[4] N. Kumar, R. Verma, D. Anand, Y. Zhou, O. Onder, E. Tsougenis, H. Chen, P. Heng, J. Li, Z. Hu et al., "A multi-organ nucleus segmentation challenge." IEEE Transactions on medical imaging, 2019.

[5] Ali I., Wani W. A., Saleem K. Cancer scenario in India with future perspectives. Cancer Therapy. 2011;8:56–70. [Google Scholer]

[6] Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms." IEEE Trans. Sys. Man. Cyber. 9(1) : 62 -66

[7] X. Yang, H. Li, and X. Zhou, "Nuclei segmentation using marker controlled watershed, tracking using mean-shift, and Kalman filter in timelapse microscopy." IEEE Transactions on Circuits & Systems, vol. 53, no. 11, pp. 2405–2414, 2006.

[8] H. Chen, X. Qi, L. Yu, Q. Dou, J. Qin, and P.-A. Heng, "Dcan: Deep contour-aware networks for object instance segmentation from histology images," Medical image analysis, vol. 36, pp. 135–146, 2017.

[9] H. Oda, H. R. Roth, K. Chiba, J. Sokoli´c, T. Kitasaka, M. Oda, A. Hinoki, H. Uchida, J. A. Schnabel, and K. Mori, "Besnet: boundary-enhanced segmentation of cells in histopathological images," in International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2018, pp. 228–236.

[10] Y. Zhou, O. F. Onder, Q. Dou, E. Tsougenis, H. Chen, and P.-A. Heng, "Cia-net: Robust nuclei instance segmentation with contour-aware information aggregation," in International Conference on Information Processing in Medical Imaging. Springer, 2019, pp. 682–693.

[11] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y.W. Tsang, J. T. Kwak, and N. Rajpoot, "Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images," Medical Image Analysis, vol. 58, p. 101563, 2019.

[12] N. A. Koohbanani, M. Jahanifar, A. Gooya, and N. M. Rajpoot, "Nuclear instance segmentation using a proposal-free spatially aware deep learning

framework." in International Conference on Medical Image Computing and Computer-Assisted Intervention, 2019, pp. 622–630.
[13] Wienert S, Heim D, Saeger K, Stenzinger A, Beil M, Hufnagl P, et al. Detection and segmentation of cell nuclei in virtual microscopy images: a minimum-model approach. Sci Rep. 2012;2(1):1–7. https://doi.org/10.1038/srep00503.
[14] Ciresan D, Giusti A, Gambardella L, Schmidhuber J. Deep neural networks segment neuronal membranes in electron microscopy images. Adv Neural Inf Proces Syst. 2012;25:2843–51.
[15] Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: international conference on medical image computing and computer-assisted intervention, pp. 234–241 (2015). Springer.
[16] Alom MZ, Yakopcic C, Hasan M, Taha TM, Asari VK. Recurrent residual u-net for medical image segmentation. J Med Imaging. 2019;6(1):014006. https://doi.org/10.1117/1.JMI.6.1.014006.
[17] Lugagne J-B, Lin H, Dunlop MJ (2020) DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. PLoS Comput Biol 16(4): e1007673. https://doi.org/10.1371/journal.pcbi.1007673
[18] Moshkov, N., Mathe, B., Kertesz-Farkas, A. et al. Test-time augmentation for deep learning-based cell segmentation on microscopy images. Sci Rep **10**, 5068 (2020). https://doi.org/10.1038/s41598-020-61808-3
[19] Partha Pratim Banik, Rappy Saha, Ki-Doo Kim, An Automatic Nucleus Segmentation and CNN Model based Classification Method of White Blood Cell, Expert Systems with Applications, Volume 149, 2020, 113211, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.113211.
[20] Long, F. Microscopy cell nuclei segmentation with enhanced U-Net. BMC Bioinformatics **21**, 8 (2020). https://doi.org/10.1186/s12859-019-3332-1
[21] Rautaray, S. S., Dey, S., Pandey, M. and Gourisaria, M. K. (2020). Nuclei Segmentation in Cell Images using Fully Convolutional Neural Networks. International Journal on Emerging Technologies, 11(3): 731–737.
[22] Edlund, C., Jackson, T.R., Khalid, N. et al. LIVECell—A large-scale dataset for label-free live cell segmentation. Nat Methods **18**, 1038–1045 (2021). https://doi.org/10.1038/s41592-021-01249-6.
[23] Greenwald, N.F., Miller, G., Moen, E. et al. Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. Nat Biotechnol 40, 555–565 (2022). https://doi.org/10.1038/s41587-021-01094-0
[24] KRÖSE, Ben; KROSE, Ben; SMAGT, Patrick van der; SMAGT, Patrick. An introduction to Neural Networks. 1993.
[25] MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The Bulletin of mathematical biophysics. 1943, vol. 5, no. 4, pp. 115–133. ISSN 1522-9602. Available from DOI: 10.1007/BF02478259.

[26] ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. Psychological Review. 1958, pp. 65–386.
[27] J. Blumberg and G. Kreiman, "How cortical neurons help us see: visual recognition in the human brain," The Journal of clinical investigation, vol. 120, no. 9, pp. 3054–3063, 2010.
[28] R. Szeliski, Computer vision: algorithms and applications. Springer Science & Business Media, 2010.
[29] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," arXiv preprint arXiv:1811.03378, 2018.
[30] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," Neural computation, vol. 29, no. 9, pp. 2352–2449, 2017.
[31] Adam: A Method for Stochastic Optimization, 2017, https://doi.org/10.48550/arXiv.1412.6980.
[32] TORRALBA, A.; MURPHY, K. P.; FREEMAN, W. T. Sharing features efficient boosting procedures for multiclass object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Washington, DC, 2004, vol. 2, pp. 762–769.
[33] LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully Convolutional Networks for Semantic Segmentation. CoRR. 2014, vol. abs/1411.4038. Available from arXiv: 1411.4038.
[34] BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. CoRR. 2015, vol. abs/1511.00561. Available from arXiv: 1511.00561.
[35] RUSSAKOVSKY, Olga et al. ImageNet Large Scale Visual Recognition Challenge. CoRR. 2014, vol. abs/1409.0575. Available from arXiv: 1409.0575.
[36] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (eds.). Advances in Neural Information Processing Systems 25. Curran Associates, Inc., 2012, pp. 1097–1105. Available also from: http : / / papers . nips . cc / paper /4824-imagenet-classification-with deep-convolutionalneural-networks.pdf.
[37] SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott E.; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent; RABINOVICH, Andrew. Going Deeper with Convolutions. CoRR. 2014, vol. abs/1409.4842. Available from arXiv: 1409.4842.
[38] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. CoRR. 2015, vol. abs/1512.03385. Available from arXiv: 1512.03385.
[39] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 2, pp. 386–397, Mar. 2017, doi:

10.1109/TPAMI.2018.2844175.
[40] Ricardo Fabbri, Luciano Da F. Costa, Julio C. Torelli, and Odemir M. Bruno. 2008. 2D Euclidean distance transform algorithms: A comparative survey. ACM Comput. Surv. 40, 1, Article 2 (February 2008), 44 pages. https://doi.org/10.1145/1322432.1322434
[41] Lin, TY. et al. (2014). Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48
[42] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. ArXiv, 2016.
[43] R. Girshick. Fast r-CNN. ArXiv, 2015.
[44] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. IEEE Conference on CVPR, 2016.
[45] T.Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. ArXiv, 2017, https://doi.org/10.48550/arXiv.1612.03144.
[46] Xia, Denan & Chen, Peng & Wang, Bing & Zhang, Jun & Xie, Chengjun. (2018). Insect Detection and Classification Based on an Improved Convolutional Neural Network. Sensors. 18. 4169. 10.3390/s18124169.
[47] Chen, Zhong & Zhang, Ting & Ouyang, Chao. (2018). End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images. Remote Sensing. 10. 139. 10.3390/rs10010139.
[48] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
[49] F. Chollet et al., "Keras," https://keras.io, 2015, accessed: 2019-02-08.
[50] K. Roy, D. Banik, D. Bhattacharjee, O. Krejcar, and C. Kollmann, "LwMLA-NET: A Lightweight Multi-Level Attention-Based NETwork for Segmentation of COVID-19 Lungs Abnormalities From CT Images," IEEE Trans. Instrum. Meas., vol. 71, 2022, doi 10.1109/TIM.2022.3161690.
[51] N. Kumar, R. Verma, S. Sharma, S. Bhargava, A. Vahadane, and A. Sethi, "A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology," IEEE Trans. Med. Imaging, vol. 36, no. 7, pp. 1550–1560, Jul. 2017, doi: 10.1109/TMI.2017.2677499.
[52] Z. Zeng, W. Xie, Y. Zhang, and Y. Lu, "RIC-Unet: An improved neural network based on Unet for nuclei segmentation in histology images," IEEE Access, 2019.
[53] F. A. Guerrero-Pena, P. D. M. Fernandez, T. I. Ren, M. Yui, E. Rothenberg, and A. Cunha, "Multiclass weighted loss for instance segmentation of cluttered cells," in 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE, 2018, pp. 2451–2455.
[54] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W. Karhohs, M. Broisin, M. Csaba, C. Mc Quin, S. Singh, F. Theis, et al., "Evaluation of deep learning strategies for nucleus segmentation in fluorescence images," BioRxiv, p.

335216, 2019.
[55] Neves, Cláudia & Gonzalez, Ignacio & Leander, John & Karoumi, Raid. (2018). A New Approach to Damage Detection in Bridges Using Machine Learning. Lecture Notes in Civil Engineering. 73-84. 10.1007/978-3-319-67443-8_5.
[56] Vui, Chang & Soon, Gan & On, Chin & Alfred, Rayner & Anthony, Patricia. (2013). A review of stock market prediction with Artificial neural network (ANN). Proceedings - 2013 IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2013. 477-482. 10.1109/ICCSCE.2013.6720012.
[57] Rukshan Pramoditha, Convolutional Neural Network (CNN) Architecture Explained in Plain English Using Simple Diagrams, 2022, in Towards Data Science, Available: https://towardsdatascience.com/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f
[58] Njima, Wafa & Ahriz, Iness & Zayani, Rafik & Terre, M. & Bouallegue, Ridha. (2019). Deep CNN for Indoor Localization in IoT-Sensor Systems. Sensors. 19. 3127. 10.3390/s19143127.
[59] Musiol, Martin. (2016). Speeding up Deep Learning Computational Aspects of Machine Learning. Available: https://www.researchgate.net/publication/308414212
[60] Strisciuglio, Nicola & Lopez Antequera, Manuel & Petkov, Nicolai. (2020). Enhanced robustness of convolutional networks with a push–pull inhibition layer. Neural Computing and Applications. 32. 1-15. 10.1007/s00521-020-04751-8.
[61] Guo, Zhiling & Chen, Qi & Wu, Guangming & Xu, Yongwei & Shibasaki, Ryosuke & Shao, Xiaowei. (2017). Village Building Identification Based on Ensemble Convolutional Neural Networks. Sensors. 17. 2487. 10.3390/s17112487.
[62] Xiao, Yuelei, and Xing Xiao. 2019. "An Intrusion Detection System Based on a Simplified Residual Network" Information 10, no. 11: 356. https://doi.org/10.3390/info10110356
[63] Phiphitphatphaisit, Sirawan & Surinta, Olarik. (2020). Food Image Classification with Improved MobileNet Architecture and Data Augmentation.
[64] Chen, Qing, Xiaoli Yang, and Emil M. Petriu. "Watershed segmentation for binary images with different distance transforms." In Proceedings of the 3rd IEEE international workshop on haptic, audio and visual environments and their applications, vol. 2, pp. 111-116. 2004.