APPLICATION OF CONVOLUTIONAL NEURAL NETWORK FOR LEUKOCYTE QUANTIFICATION FROM A SMARTPHONE BASED MICROFLUIDIC BIOSENSOR

BY,

HARSHITHA GOVINDARAJU

A thesis submitted to the School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Biomedical Engineering

Written under the direction of

Dr. Umer Hassan

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January 2021

ABSTRACT OF THE THESIS

APPLICATION OF CONVOLUTIONAL NEURAL NETWORK FOR LEUKOCYTE QUANTIFICATION FROM A SMARTPHONE BASED MICROFLUIDIC BIOSENSOR

By, HARSHITHA GOVINDARAJU

Thesis Director: Dr. Umer Hassan

Advancements in computer vision methodologies and machine learning in the medical domain have played a major role in diagnostics and clinical pathology. Cell quantification from whole blood can aid in detecting and managing infections, cardiovascular diseases and biomarker detection which in turn helps in understanding the immunological and genetic disorders, cancers, etc. Developing a point-of-care solution for this will accelerate the therapy timeline and increase the accessibility across the world. Our lab has previously developed a smartphone based microfluidic biosensor for capturing the microscopic images of various components of the blood cells. Using this design, in this study, a deep learning-based cell quantification from the captured images is investigated and the cell counts are predicted using a convolutional neural network architecture. The proposed methodology was evaluated on a dataset varying in numbers, clarity, smartphones, fluorophores and cell numbers. This model was then integrated into an Application Programming Interface (API) to predict the cell counts from an image using the trained model. Our results showed successful prediction of cell

counts from a smartphone captured image in cross-validation with $R^2 = 0.99$ for N=33. This helps in eliminating the need for manual pre-processing of an image and morphological methods for cell counting which is a user-skill based approach. This proposed Deep Learning based cell quantification has shown agility and more automated process when compared to the benchmark techniques.

ACKNOWLEDGEMENTS

I would like to begin by thanking Dr. Umer Hassan for being a great mentor and for believing in me and giving me the opportunity to work under his guidance. His guidance proved invaluable through the course of my thesis project and for my future career.

I am thankful to Dr. Nada M. Boustany and Dr. Jay C. Sy for being a part of my master's thesis committee, whose feedback will be immensely helpful. Special thanks to Ahsan M. Sami for developing the backbone of this research and for patiently teaching me various techniques of the research. I would like to extend my thanks to my lab members Brandon Ashley and Priya Parikh for their timely help.

Additionally, I would like to thank Lawrence Stromberg, Dr. Joseph Freeman and all the faculty of BME for their advice, support and kindness they've shown me through the course of my master's degree.

I would also like to thank my dearest friends for their constant motivation. And finally, thanks to my parents for having dreamt this for me and for their endless love and unwavering support. Without you all, I would not be writing this today.

Table of Contents

ABSTRACT OF THE THESIS II

ACKNOWLEDGEMENTS IV

TABLE OF CONTENTS V

LIST OF FIGURES VII

INTRODUCTION 1

1.1 Cell imaging 1

1.2 Point of care testing 3

1.3 Smartphone based microfluidic biosensor 4

1.4 Artificial intelligence and smartphone 5

1.5 Current smartphone setup 6

1.6 Statement of problem 6

1.7 Specific aims 6

1.8 Organization of the thesis 7

DEVICE SETUP AND DATA COLLECTION 8

2.1 Device setup 8

2.2 Imaging the green fluorescent microbeads 9

2.3 Imaging the red fluorescent beads 9

2.4 Imaging leukocytes 10

2.5 Dataset samples 10

2.6 Device specifications 13

PREPARING THE DATASET 14

3.1 Introduction to ImageJ 14

3.2 Cropping 17

3.3 Data augmentation 18

3.4 Image degradations 19

NEURAL NETWORK ARCHITECTURES 20

4.1 ARTIFICIAL NEURAL NETWORK 20

4.2 CONVOLUTIONAL NEURAL NETWORK 22

4.2.1 Convolution 24

4.2.2 Rectifier Linear Unit 27

4.2.3 Pooling 28

4.2.4 Regularization 28

4.2.5 Flattening and Fully Connected Layer (FCN) 29

4.2.6 Analysis metrics 30

RESULTS 31

5.1 STAGE I: STANDARD TESTING 31

5.2 STAGE II: CROSS VALIDATION 37

5.3 STAGE III: RED BEADS TESTING 40

5.4 ANALYSIS OF RESULTS 40

DEPLOYMENT OF THE DEEP LEARNING MODEL INTO AN APPLICATION PROGRAMMING INTERFACE 43

CONCLUSION AND FUTURE SCOPE 45

REFERENCES 46

LIST OF FIGURES

Figure 1: 3D CAD Model of the smartphone based fluorescent microscope. 8

Figure 2 : Two models of smartphones employed for fluorescent image acquisition (A) Samsung Galaxy S9+ (B) Nokia Lumia 1020. 11

Figure 3: (A) Sample green microbeads image acquired from the smartphone based fluorescent microscope (B) Sample red microbeads image acquired from the smartphone based fluorescent microscope (C) Sample human leukocyte image acquired from the smartphone based fluorescent microscope. 12

Figure 4: Sample images of the whole microfluidic sample from (A) Green microbeads (B) Human Leukocyte (C) Red microbeads 13

Figure 5: Overview of the ImageJ cell counting methodology. 15

Figure 6: Correlation plot of cell concentration count and ImageJ count. 17

Figure 7: Cropped images after undergoing slicing process. 18

Figure 8: Neurons and its connections in an Artificial Neural Network (ANN) architecture 20

Figure 9: Two architectures of CNN investigated in this study (A) Architecture 1 (B) Architecture 2. 24

Figure 10:Input image convolved with vertical line detector filter. 25

Figure 11: Input image convolved with horizontal line detector filter. 25

Figure 12: Input image convolved with Sobel filter. 25

Figure 13: Input image convolved with Sobel filter. 26

Figure 14: Feature maps stacked together after convolution. 27

Figure 15: ReLU function graph. 27

Figure 16: Preview of the Dropout mechanism. 29

Figure 17: Training and validation plots at learning rate=0.0001. 32

Figure 18: Correlation plot between the GT count from Image J and the predicted count from CNN model in figure 9A. 32
Figure 19: Bland Altman analysis of the GT count and predicted count from CNN. . 33
Figure 20: Correlation plot for ground truth count and prediction count for architecture in figure 9B. 33
Figure 21: Training and validation loss curves for different learning rate hyperparameters. 34
Figure 22: The correlation plots for varying batch size=8 and 32 given the LR=0.001. 35
Figure 23: The correlation plots for varying batch size= 8 and 32 given the LR=0.0001. 36
Figure 24: Images collected from Samsung S9+. (A) Regular image. (B) Image with Gaussian Blur, sigma=3. (C) Image applied with binary mask 38
Figure 25: Images collected from Nokia Lumia 1020. 38
Figure 26: Correlation plot of GT count and predicted count. 39
Figure 27: Bland Altman Analysis for GT count and predicted count from cross validation. 39
Figure 28: Correlation plot of GT count and prediction count from binary masked red beads images. 40
Figure 29: Histogram plot of difference between ground truth and predicted count from standard testing results 41
Figure 30: Histogram of the difference between ground truth and predicted count from cross-validation testing 41
Figure 31: Preview of the web application. 43

CHAPTER 1

Introduction

1.1 Cell imaging

Cell quantification is an important aspect in diagnostics and clinical investigations. Cell counts especially aids in curating general pathology and clinical pathology. The complete blood count (CBC) and leukocyte differential count (LDC) are the most commonly requested and performed laboratory tests. The results can be used to detect disorders, infections, immune system diseases and nutritional status of the patient. Methodologies to count cells constantly work towards abridging the time of analysis and accuracy. With the evolution of microscopes, it helped researchers to count different components of blood from high resolution images. Cell counting process can be broadly classified into manual, semi-automated and fully automated process. Manual and semi-automated process results are variable to subjective assessments. It involves a highly trained laboratorian to manually count from the sample slides under the microscope. However, manual cell counting process can become tedious and error prone as it is subjected to the user's acuity like hemocytometer. Although in some cases, it is still established as the gold standard. Fully automated machines come with very little sample preparation complexity and with advancements in image processing and computer vision, it led to software that helped researchers count different components of blood by applying mathematical morphology without the manual “clicker”. Other popular cell counting methodologies like flow cytometer implements signal-based detection that involves using an external system for computation of the counts from the sample. The process here can be prone to technical as well as human errors.

With the rise of various improved and advanced imaging techniques, there is abundance in cell image data. The quality in these images can bring out the minute details in cell profiling and aid in drawing pathology results. For years now, computer research scientists have come out with powerful software that can help in quantification. But they are specific to quantifying only a particular component in a cell. Devices like Vi-CELL BLU, Vi-CELL MetaFLEX and Multisizer 4e Particle Size Analyzer from Beckman Coulter are fully automated machines that aid in cell counting, sizing, viability analysis and cell culture monitoring systems. But these systems can only be employed in laboratory settings. The size of this machine is also one of the drawbacks when it comes to point of care applications. Their image processing softwares are specific to the device collecting the data. This limits the ability to employ it with other devices. Invitrogen™ Countess™ is another tool used in image cytometry. The image processing software used in this device is very specifically applicable only for that device. A software application can only perform tasks it is designed for a particular type of cell. So, there is a need for developing an adaptable solution like Artificial Intelligent systems, which can aid better in cell quantification. There is also an issue of memory storage when it comes to these devices. In other devices, it again varies with the markers or labels that are used to image the cell components for counting. There is also a huge gap in the amount of microscopic image data and utilization of these images to come up with powerful deep learning models to predict the cell quantification by learning the relevant features necessary for the task from the input data. Although learning based algorithms were introduced to the biomedical science field years ago, it is not until recently the graph has taken off in terms of implementation. This is a result of improvement in imaging techniques and high quantity of microscopic images. The deep learning implementation can be either specific or broad depending on the modelling

process which can be time and again improvised. It follows a simple concept that it learns from its predicted error difference. When the model has learnt to predict better, it can then be deployed. Using this aspect, we can make the model "learn" any task depending on the requirement. This again provides a broad application in image cytometry. But as we move towards improving the quality of images and cell analysis, taking the size of the imaging device into account itself has a key role towards advancements.

1.2 Point of care testing

Point of care testing solutions integrate biosensors, assays, microfluidic chips, bioanalytical platforms and lab-on-chip technologies to provide rapid diagnosis and aid in accelerating the prevention and treatment plans. As infectious diseases continue to attack global health with newer or re-emerging virulent, the need for testing kits that can be employed both in laboratory settings and at patient's vicinity as a portable equipment is growing. Although the pharma industry is setting high standards in coming out with powerful drugs to combat these ailments, the gap between diagnosis and treatment plan still remains largely unaddressed. Mostly in developing countries with high population and limited resources, it is challenging to set up laboratories with skilled workers and meet the growing demand for disease diagnosis. Such scenarios are where we need to employ these diagnostic kits. This has led to the demand for coming up with faster and efficient POC devices.[1][2]

1.3 Smartphone based microfluidic biosensor

As we understand how important time is for disease diagnosis, point-of-care testing in such scenarios has to be lightweight and easy to operate to employ it in a daily setting. Commonly used biosensors on the other hand are heavy, have a complex design setup and are expensive which limits their ability for point of care testing for the common. Lot of efforts have been made to slim down the microfluidic lab-on-chip device designs that can integrate biochemical sensors, microfluidics and micro-electronic mechanical systems (MEMS) [3][4][5]. But the data collected from these models would require an additional processing unit that can collect the data, have a separate memory for data storage, display the data and results, analyze and transfer the data to different mediums for further analysis. A smartphone on the other hand can provide a single hand solution to these challenges and due to this, a lot of microfluidic researcher's focus has shifted towards integrating developing smartphone based analytical biosensors. [6]

Smartphones intend to replace the traditional phones into miniature computers that have their own multicore processing unit with a variety of built-in sensors that makes it user friendly. In addition to this they provide large data storage capabilities, battery power, external ports for multi device connectivity, smart interfaces and internet connectivity. With time, the smartphone technology is only going to advance and hence extending its capabilities for analytical biosensing. This technology can be seen implemented as an electrochemical analyzer of Uric Acid from a whole blood sample with an enzymatic test strip. The results from these analyses can aid in characterizing many diseases related to gout, kidney and heart.[7]. Another research showed an ECG signal processor for monitoring heart diseases [8]. Implementation of smartphone in monitoring Urinary Tract Infections using colorimetric reaction pads. [9] Detecting microbial contamination in meat meant for consumption through a smartphone-based biosensor

[10], food evaluation [11] Smartphone for testing biomarkers from liquid biopsy [12], mobile health [13]. All these researches have two things in common- 1) Point of Care Testing and 2) Integration of Smartphone based biosensor.

1.4 Artificial intelligence and smartphone

With release versions of smartphone devices, there is constant development in improving the camera device and clarity of the images taken from smartphones. This acts as a valuable feature in using it as a biosensor for researchers. Further the smartphone can be embedded with additional sensors for application specific purposes [6]. Area of mobile data science has bloomed along with the number of smartphone users. With mobile applications having permissions to user information, it has allowed mobile application developers to come up with AI Powered Smart Applications that serve a lot of human activities on a day-to-day basis. AI based modeling on mobile applications allows smartphone apps to make predictions by understanding the user's needs. This is done by learning the user's data and their usage patterns. This also enforces right suggestion of resources and hence personalizing the product for usage.[14] This intelligent technology when applied in the field of point of care testing can prove to be an effective system for early, smart detection and in turn help in accelerating the treatment plans. Open-sourced collection of workflows like TensorFlow aid in developing, training models using Python or JavaScript languages and deploy them on cloud, on-prem, web applications or device applications. This allows the researchers to also employ high-level Keras API to build the application specific to the Biomedical Task [15].

1.5 Current smartphone setup

As we understand the potential of the smartphone being a biological sensor, a research shows a modular microscopic smartphone based fluorescent microscope for imaging and quantification of multiple fluorescent microparticles [16]. This device is designed for multiple fluorescent markers and offers multiple magnifications. The setup has the flexibility to work with any smartphone. This sensor also showed to image microfluidic chips along with sample on slides/cover clips.

1.6 Statement of problem

Although the device is designed to image and include large types of cells, this study is focused on the dataset from green fluorescent microbeads, red fluorescent microbeads and human leukocytes. In recent years, advancements in clinical biochemistry and integration of electronics, optics and data collections and processing has led to abundance of data. Utilizing this data to train a deep learning model, integrate this model into the POC device to further aid in the prediction of the disease could potentially advance disease diagnosis. This is the primary aim of this research- to apply CNN, a predefined deep learning model, to predict the cell counts from the Smartphone captured images of Leukocyte particles. This ultimately enables a user independent software that can perform powerful cell quantification.

1.7 Specific aims

1. Collection of datasets with different cell concentration, design of the setup, smartphone setup
2. Creating the dataset for training. Labelling the images using ImageJ

3. Data augmentation by cropping, binary mask, blurring,
4. CNN optimization with different neural networks, learning rate, batch size, epochs
5. Evaluating of the results with test images
6. Comparison of predicted cell counts against different ground truths
7. Deploying the trained model into web applications
8. Testing the web application with a new acquired image
9. Investigating the AI integrated device and its implementation

1.8 Organization of the thesis

In chapter 2 we have highlighted the design and working of the modular smartphone based fluorescent microscope. In chapter 3 we discuss the data collection, preprocessing and labelling required for training the neural network. In chapter 4 we discuss the intuition behind the neural network and convolutional neural network. In chapter 5 we discuss the results obtained by training and testing the images acquired from smartphone based fluorescent microscope using the convolutional neural network. In chapter 6 we discuss a practical approach to this research. In chapter 7 we have summarized our findings and highlighted the future scope of this research.

CHAPTER 2

Device Setup and Data Collection

2.1 Device setup

The device primarily consists of two units- top and bottom. The top portion consists of lens holder and slot for high pass filter. The lens holder has detachable lens which makes the device flexible to use with different magnification levels. Two lenses are used in this study. One with a focal length of 10mm and other with a focal length of 15mm. The top portion acted as the platform for the smartphone for imaging. The top portion also consisted of 4 screws at each corner of the device which when rotated can aid in varying the image focus and setting the right imaging angle. The bottom portion consists of the cavity to place the microfluidic chip with the captured leukocytes or the imaging subject and slots for LED's and band pass filter. A cover shield is placed on top of the chip and affixed to the bottom portion. This section of the device minimized the leakage of light reflecting from the interiors of the cavity. This device was designed to primarily image microparticles using fluorescence microscopy. Two main fluorophores were used in this study to label the microparticles.

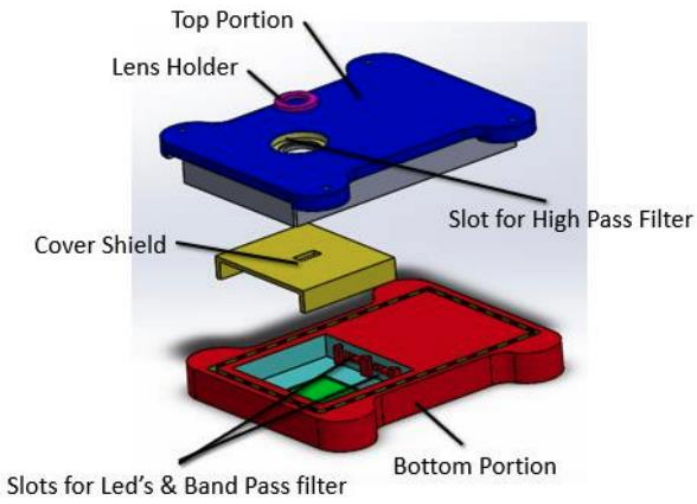


Figure 1: 3D CAD Model of the smartphone based fluorescent microscope [16].

2.2 Imaging the green fluorescent microbeads

The beads used in this study were of the size 8.3µm to mimic the size of the human leukocytes. Three blue LED's are used to excite the fluorophores with a bandpass filter's specification of wavelength of 470nm and a bandwidth of ~40nm that is placed in the bottom portion of the device. A long pass filter with a cut-off value of 500nm is also used while imaging the green fluorescent microbeads. For setting up the sample to image, 1X PBS buffer was used to produce different concentrations of the beads for imaging in turn producing dataset with varying cell count. Then 2µl sample was taken from each concentration to pour on the glass slide and place it in the imaging cavity of the device in the bottom portion. The lens specifications used in this study was focal length=15mm.

2.3 Imaging the red fluorescent beads

The mean diameter of the acquired red fluorescent beads were 10µm. In this case, green LED's were used to excite the particles. When working with the red fluorophores, the device was configured to use a bandpass filter with 535nm center wavelength and a bandwidth of 50nm. Also, a long pass filter with a cutoff value of 593nm was used. For imaging, 1X PBS buffer was used to produce different concentrations of the beads for imaging in turn producing dataset with varying cell count. Then 2µl sample was taken from each concentration to pour on the glass slide and place it in the imaging cavity of the device in the bottom portion. The lens specifications used in this study was focal length=15mm.

2.4 Imaging leukocytes

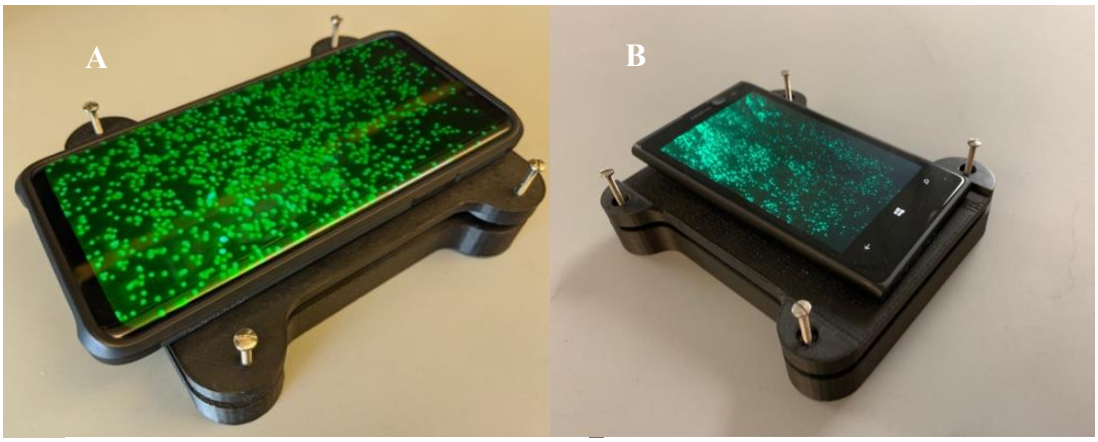
To understand the point of care application of the device and its functionality in understanding disease quantification from biological sample was key to this study. The red blood cells sample from peripheral human blood was lysed using RBC lysis buffer media from ThermoFisher. To image the isolated leukocytes using the device, they were required to be treated with green nuclear stain. For this process, a stock solution for the nuclear stain was prepared by adding 3µl of SYTO 16, (ThermoFisher Scientific, Catalogue Number: S7578), in 1 ml of 1X PBS. The two solutions were then mixed with a ratio of 1:1 in a 1.5ml Eppendorf tube and were incubated in dark for 15minutes. As it is key to create different cell count images for this study, this was achieved by using 1X PBS buffer to create different concentrations of fluorescent leukocyte samples for imaging. To image it using the device, 2µl sample was poured onto the glass slide and placed inside the imaging cavity of the bottom portion of the device. As the leukocytes were activated with green fluorescence, blue LED's were used to excite the sample and then imaged by the desired Smartphone along with a long pass filter of cutoff value 500nm and lens with a focal length of 15mm.

2.5 Dataset samples

Magnification is an important aspect when imaging a sample required for pathological analysis. The current device system integrates this important part of microscopy by allowing the user to choose between multiple magnification settings [16]. This is achieved by simply changing the lenses in the lens holder section of the top portion. In some cases, the depth of the sample used to image using the device can cause focusing issues. The cells at the bottom layer of the sample can get out of focus and may lead to imaging issues. As cell counts are an important part in cell quantification process, it is

essential to make sure that all the cells are imaged without blurring. To justify this issue, the samples were spread out on the glass slide. Furthermore, the height of the microfluidic sample is normally 60µm and there is no major depth difference between the cells and hence known to not cause any focusing issues. And more importantly, the Z-stage design of the setup enables the user to image any out of focus particles.

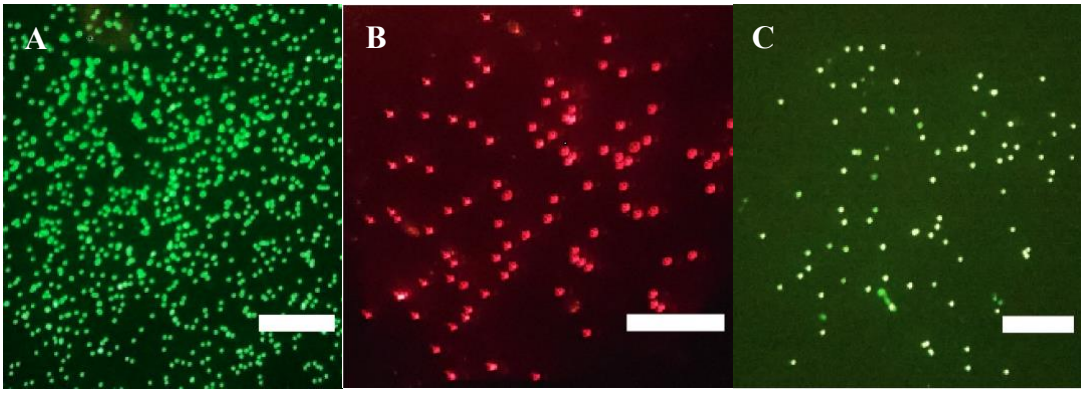
The dataset includes imaged samples from two devices- Nokia Lumia 1020 and Samsung Galaxy S9+. Both devices used in this study have different camera specifications and are on the far end of each other in terms of imaging clarity. This study includes dataset from each of the devices and observes the model's performance.



Samsung Galaxy S9+ Nokia Lumia 1020

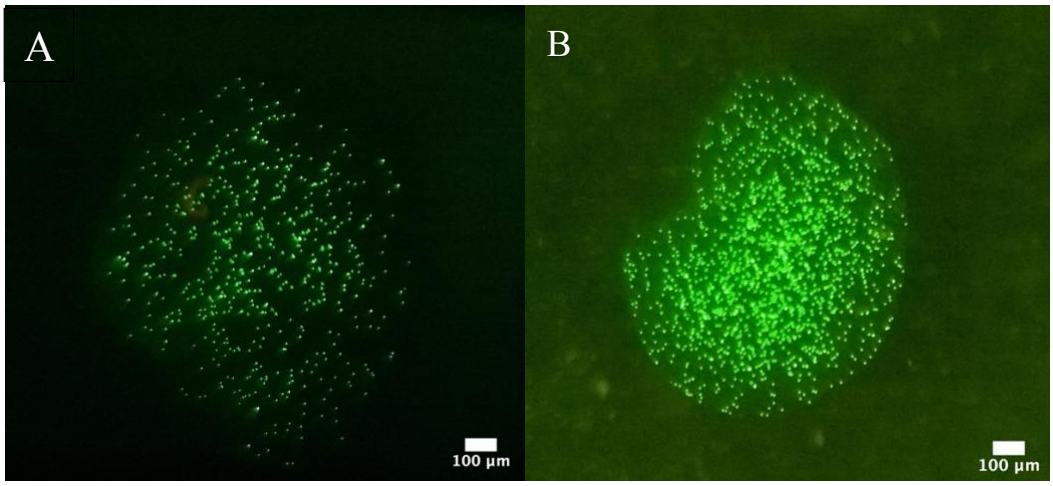
Figure 2 : Two models of smartphones employed for fluorescent image acquisition (A) Samsung Galaxy S9+ (B) Nokia Lumia 1020 [28].

The samples collected for green and red fluorescent microparticles, leukocytes from Samsung Galaxy S9+ are shown below in Figure 3.

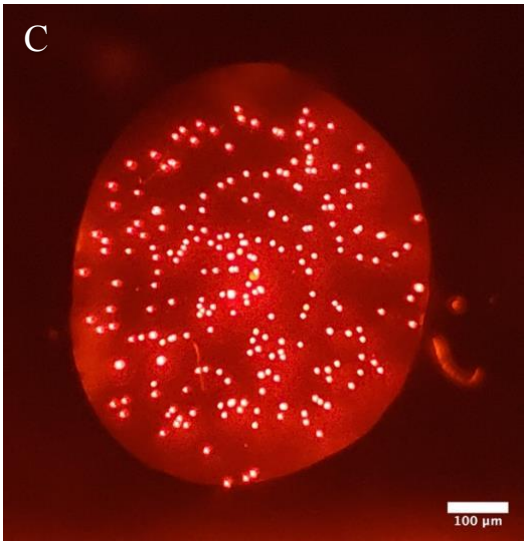


Green microbeads Red microbeads Human leukocytes

Figure 3: (A) Sample green microbeads image acquired from the smartphone based fluorescent microscope (B) Sample red microbeads image acquired from the smartphone based fluorescent microscope (C) Sample human leukocyte image acquired from the smartphone based fluorescent microscope (scale bar = 100µm).



Green microbeads Human leukocytes



Red microbeads

Figure 4: Sample images of the whole microfluidic sample from (A) Green microbeads (B) Human Leukocyte (C) Red microbeads (sample bar=100µm)

2.6 Device specifications

Device Material		3D printed with Onyx, Markforged	
Dimensions		120mm(W) x 140mm(L) x 22.7mm(H)	
Lens		10mm, 15mm focal length	
Resolution		3.9µm for Nokia Lumia 1020, 6.2µm for Samsung galaxy S9+ [16]	
Filters	Green Beads	Red Beads	Leukocytes
Long pass with cutoff value attached to lens	500nm	593nm	500nm

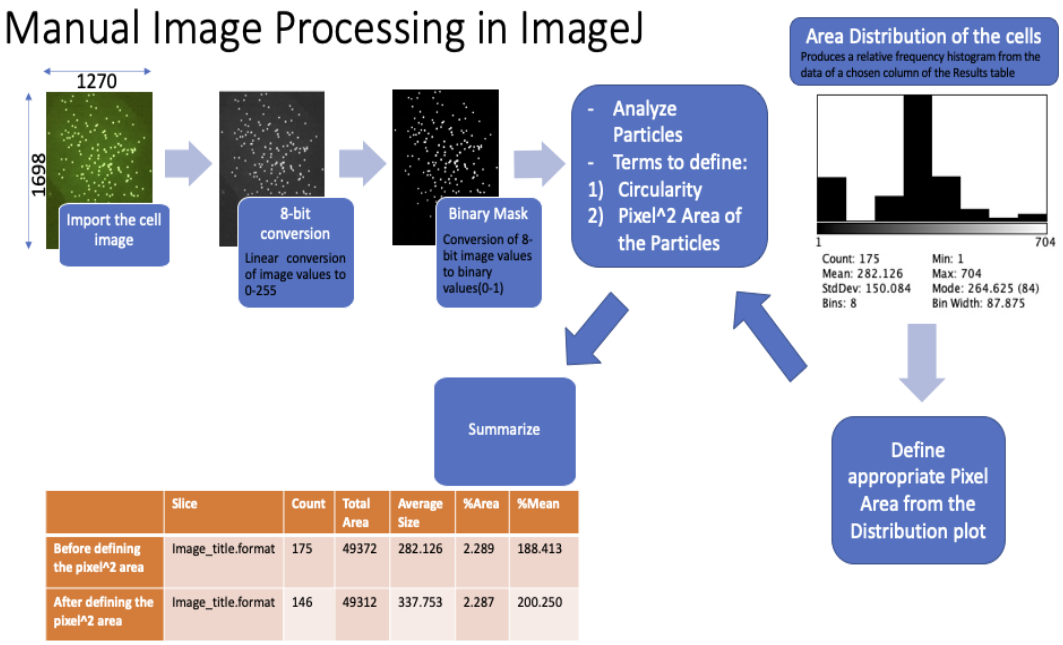
CHAPTER 3

Preparing the dataset

3.1 Introduction to ImageJ

The images acquired from the smartphone devices are generally in JPG, PNG, JPEG formats. The image file size ranges from 25KB to 15MB. The resolution of the images ranges from 100pixels ~ 5000pixels. The image processing software used to annotate the data collected is ImageJ, Software version- 1.52q. It is java-based image processing software which is open sourced and in the public domain. The author is Wayne Rasband, National Institute of Mental Health, MD. Since this application supports different image formats, ability to work with different resolutions of images as it is compatible with over 150 different biological image formats as called Bio-Formats [17]. Due to open-source licensing, a large number of plugins are available which is useful for image pre-processing and annotation. It even has the flexibility for the user to create a program to automate the process of working with images. Hence, this trait becomes utilitarian when there are a large number of datasets to preprocess and annotate.

From our aim, for cell quantification, it is important to get the cell counts from the captured image to understand the particle concentration in the sample. The workflow to get the cell count from image using ImageJ is as shown in Figure 5:



	Slice	Count	Total Area	Average Size	%Area	%Mean
Before defining the pixel^2 area	Image_title.format	175	49372	282.126	2.289	188.413
After defining the pixel^2 area	Image_title.format	146	49312	337.753	2.287	200.250

Figure 5: Overview of the ImageJ cell counting methodology.

1. The image captured from the smartphone setup is loaded onto the ImageJ software.
2. An 8-bit conversion of the image is done to linearly scale the image from min-max to 0-255.
3. Then the image undergoes the process of "thresholding" where it automatically sets a threshold value and converts the values above the set threshold to 1 and below the threshold to 0. Hence converting the image to a binary image.
4. The binary image is then set for calculating the cell counts from the image by using "Analyze Particles". This process works by scanning the image until it finds the edge of the cell/particle in the image. The wand tool outlines the edge. An inbuilt "Measure" command measures the outline until the scanning process reaches the end of the image. Depending on the selection type, the measure command calculates and displays area statistics, line lengths, point coordinates

and/or angles. In this case, since the cell/particles are the highlights of quantification, the area of the cell occupying the image is produced. The "Analyze Particles" command can also be configured to count by giving a preset value of area and circularity.

5. After the selections are made and results of the image are obtained, we can observe two tables- Results and Summary. The results table contains the area of each cell and the summary table specifies the total cell/particle count in the image, total area occupied by the cells in the image, the average size of the cells/particles and the %area of the cells covering the whole image. This insight is helpful in annotating each cell image.
6. From the "Area Distribution" graph in Figure 5, we observe that particles with pixel^2 area= 1 are also counted as a cell. From the summary table, when we have the average size of the cell>> min pixel^2 area of the cell, we can thus conclude that these particles are "noise" in the image and do not account for the cell count. We then understand this criterion and define a particular range of pixel^2 areas to be considered as a cell/particle to count.
7. After the setting the values in for "Analyze Particles" we obtain cell count from the image.

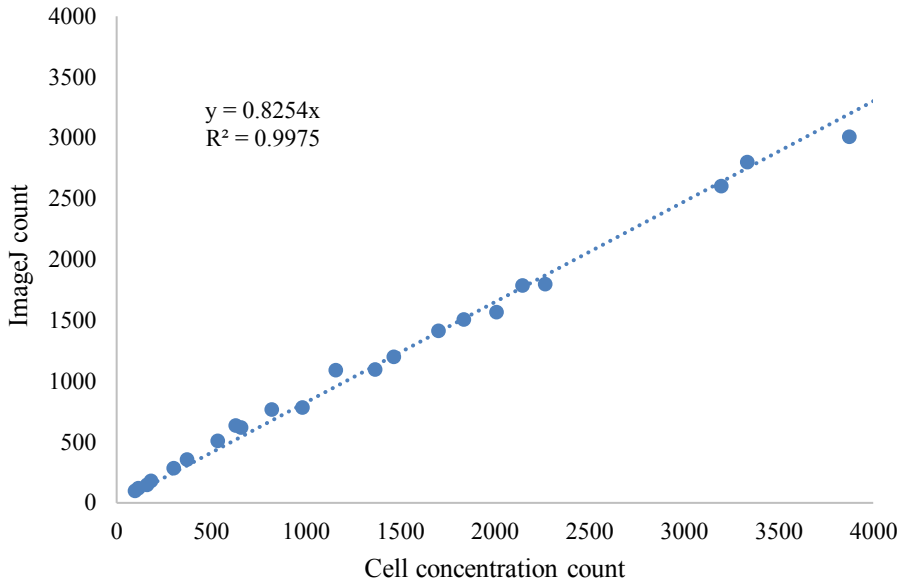


Figure 6: Correlation plot of cell concentration count and ImageJ count.

3.2 Cropping

With a high-resolution smartphone, the image captured ranges from 1000 px ~ 3000px. Each pixel is fed to each neuron. Feeding this to a neural network, it will have to work with around approx. $10^6 \sim 30^6$ neurons in the first layer. Having an inadequately large number of neurons in training layers can cause overfitting. But learning every feature in the image is important for the cell counting regression task. Although the convolution process employed in this study can downsize this, it will still end up losing important features along the way. To address this bottleneck, in this study we have cropped the whole image to smaller images which is then rescaled to a standard size of 128X128. If an image of size 848X1134 is considered, it is cropped to 8 sections widthwise and 11 sections height wise. This results in 88 smaller images with dimensions 106X103. These images are then rescaled to 128X128 using bicubic interpolation to feed the CNN architecture. This computer vison methodology acts as an advantage in data augmentation technique which is a method to increase the training set for better model

prediction. The Figure 6 below shows the example of the cropping pipeline implemented for this application.

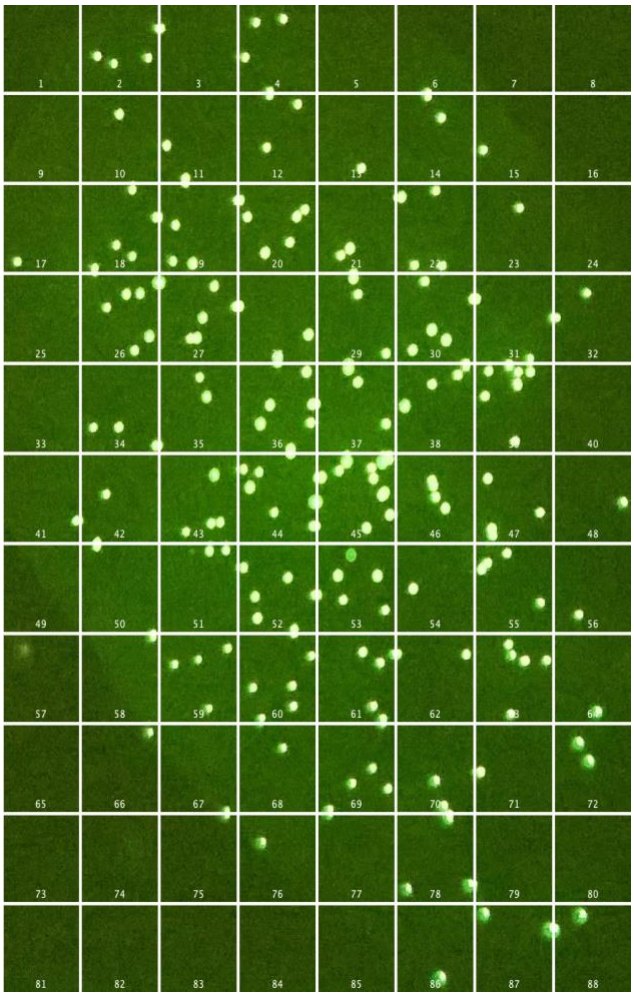


Figure 7: Cropped images after undergoing slicing process.

3.3 Data augmentation

The novel part of the study is adding binary cell image data along with actual image data in the training. This not only accounts in the process of data augmentation but also training the model with binary cell images. Advantages of including the cell mask is that when the model is set for predicting cell images with different fluorophores, the images can then be converted to a binary image for the prediction in the deployment stage. This result is discussed in Chapter 5.3.

3.4 Image degradations

To observe the model's performance on degraded images like blurring, noise and distortions as these effects can be expected in the practical setting. Image sensors of camera which are responsible for converting light and color spectrum into electrical signals that get coded into image data tends to have photon counting noise particularly in low light situations [18]. We modeled the gaussian noise with sigma=3 and included this dataset in cross validation process as shown in Figure 24-B.

This study seeks to observe the model's performance when these images are added to the training and testing how it would affect the performance of CNN.

CHAPTER 4

Neural Network Architectures

4.1 Artificial Neural Network

The computation model of an artificial neural network was originally derived from a biological neuron and its functionality to perform complex tasks in an animal brain.[19] This primary idea has now given a pathway to many intelligent systems. Increase in the computational capability, high processing power and large quantity of open-source data has all contributed to ANN's ability to outperform other Machine Learning techniques. It typically consists of an input layer of neurons, a couple of hidden layers and a final layer of neurons.

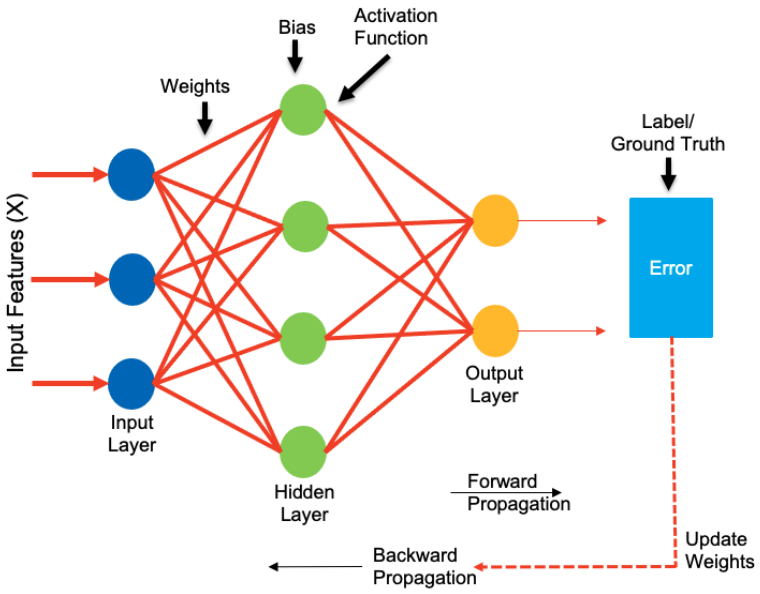


Figure 8: Neurons and its connections in an Artificial Neural Network (ANN) architecture

The features from the input are passed through the input neurons that is connected to a hidden layer through *channels*. These channels have a numerical value assigned to them called weights(W) and these weights get multiplied to the corresponding incoming

input features(X). Their sum is then sent to the neurons in the hidden layer. Each of the neurons in the hidden layer is associated with a term called *bias* which is then added to the input sum. This is then passed through the activation function which decides if a particular neuron is activated or not. And only the activated neuron transmits data to the next layer over the channels. In this way, the data is propagated across the network. The process till here is known as forward propagation.

$$Y = \sum_1^i WiXi + \text{bias}$$

$$Z = \text{Act}(Y)$$

In the output layer, the neuron with the highest value gets fired and determines the output value. The values computed till this stage are probable or predicted. These predicted values are then compared with the actual output values and the error is computed. The error function decides how far away is the predicted value from the actual value. This information is then passed back to the network through the process known as backpropagation. The weights are now adjusted, and the loss is again measured. This process of forward propagation and backward propagation continues until the neurons are now able to predict closely to the actual output value. The weight update process can be explained by this formula [20]:

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta(y_j - \hat{y}_j)\, x_i$$

$w_{i,j}$ is the connection weight between the i^{th} input neuron and the j^{th} output neuron
X_i is the i^{th} input value of the current training session
$\hat{Y}_j$ is the output of the j^{th} output neuron for the current training session
$Y_{j\,i}$ is the target output of the j^{th} output neuron for the current training session
η is the learning rate.

They considerably have more diverse applications to a problem. Convolutional Neural Networks (CNN) in addition is a specialized architecture for computer vision tasks. They have similar functionality as ANN but with a most important layer called

convolution layer. Convolution is a mathematical cross-correlation operation of two functions that slides one over the other after one is shifted and measures the integral of the product of the two. In this layer, it does not allow every pixel in the image to be connected to the first layer but every neuron in the subsequent layer is connected to the first layer. This architecture enables us to concentrate on the low-level features in the first layers and gather them to high level features in the subsequent layers. CNN also overcomes the bottleneck of implementing ANN for computer vision tasks. An image of 100X100 has 10,000 pixels and when fed into a first layer of artificial neural network with 1000 neurons, this limits its ability to capture all the right features/weights for activating and carrying the right information to the next layers in the architecture which results to a total of 10-100 million connections. Convolution operation can be done on 2D images which is also advantageous over artificial neural networks that work with 1D input where the images have to be flattened for computations.

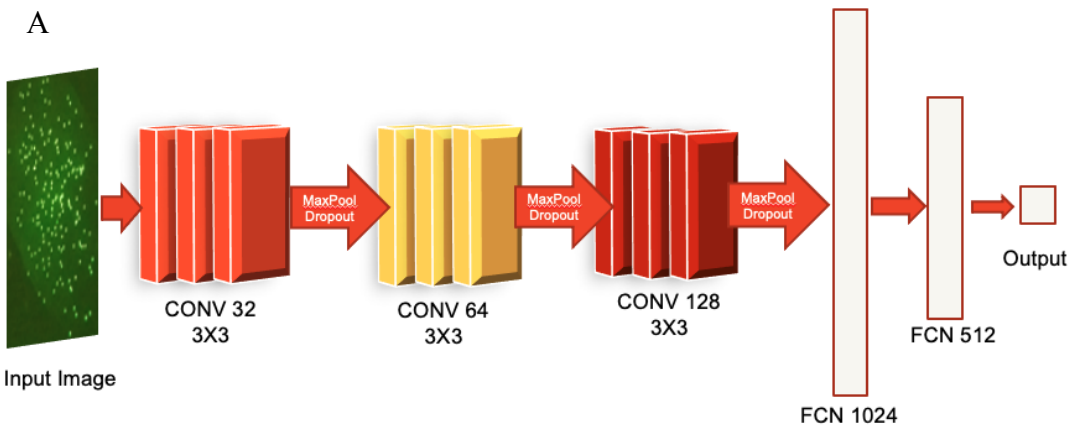
4.2 Convolutional Neural Network

CNN is popularly used for image recognition, segmentation, classification, localization, detection and other natural language processing tasks. This has led to its use in diverse areas such as visual recognition tasks, smart surveillance, self-driving cars, robotics, sports and recreation, drones and in health and medicine. Some of the popular CNN architectures include AlexNet[21], VGGNet[22], GoogleLeNet [23], ResNet[24] and UNet [25]. These popular networks are now more advanced and have given rise to many other networks for different applications. In the medical imaging field, CNN is applied in the reconstruction pipelines of MR Imaging, radiotherapy, PET-MRI attenuation correction, radiomics, theragnostic [26][27]. It has also popularly been used for segmentation tasks in medical imaging [28] which has an essential role in diagnosis and

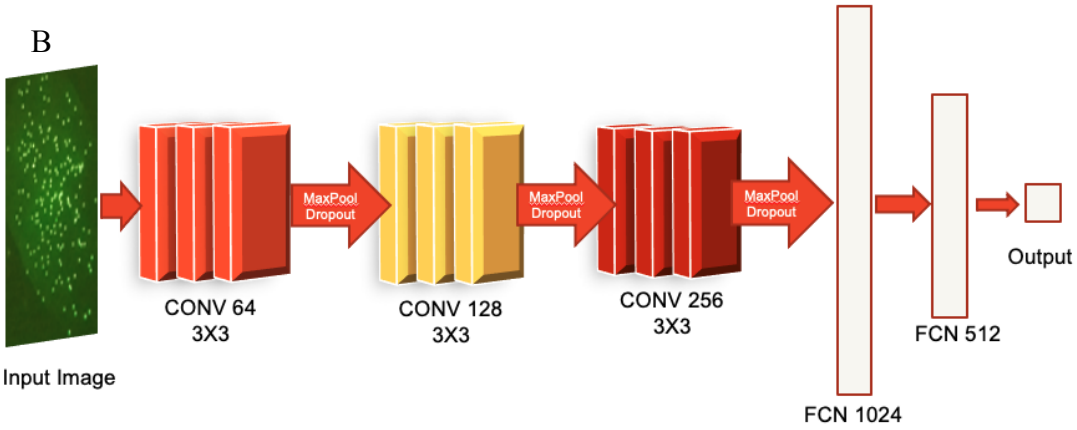
pre-surgery evaluations by extracting the region of interests (ROI). This has also played a key role in detecting anomalies from the medical images where the gold standard is set by expert radiologists [29]. From this we observe how popular and effective CNN is for computer vision task and therefore it was chosen for this study.

Since cell counting is an important task in the field of disease prognosis, a lot of research in recent years has implemented CNN models with slight modifications w.r.t the task in hand. A CNNCS framework for cell detection and localization which introduces compressed based encoding along with the popular AlexNet architecture [30]. A U-CNN framework proposed for small object counting by the density map obtained from the model output [31]. Two variations of Fully Convolutional Regression Networks (FCRN) framework for microscopy cell counting [32].

This research focuses on implementing a simple CNN model for Cell Counting task to further deploy this in an Application Programmable Interface (API). The main aim is to get optimum results with low computation/model complexity so that this system can be employed in a limited resource setting for Point of Care diagnostic applications.
The two popular CNN architectures that were investigated for the study are depicted in the Figure 9 below:



Architecture 1



Architecture 2

Figure 9: Two architectures of CNN investigated in this study (A) Architecture 1 (B) Architecture 2.

4.2.1 Convolution

When our cell image was multiplied with a 3X3 filter (convolution kernel) for vertical line detection, we observe that the resulting images are highlighted in the areas that are most similar to the filter and in this case vertical white lines get enhanced while the rest is blurred as shown in Figure 10. Similarly, when the image was multiplied with a filter for horizontal line detection, we observe only the horizontal white lines enhanced and rest of the image is blurred out as shown in Figure 11. We see more prominent observations when vertical and horizontal Sobel filters are applied as shown in Figure 12 and 13.

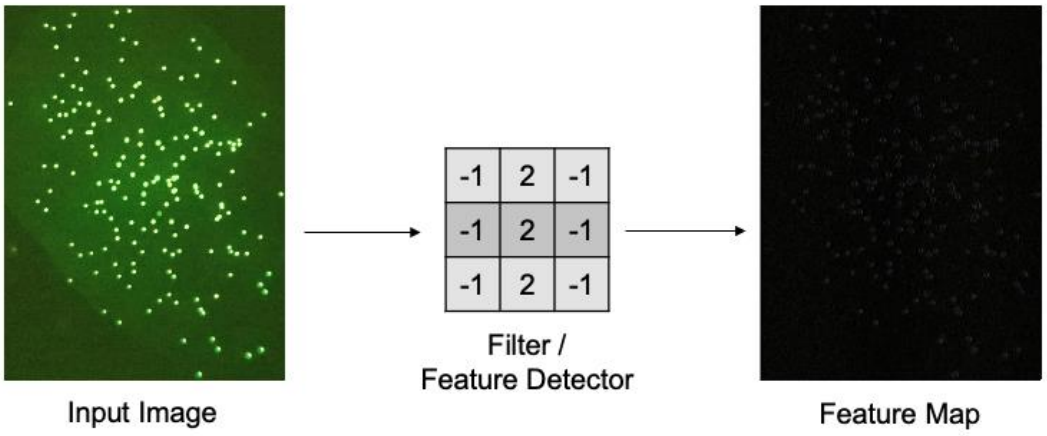


Figure 10:Input image convolved with vertical line detector filter.

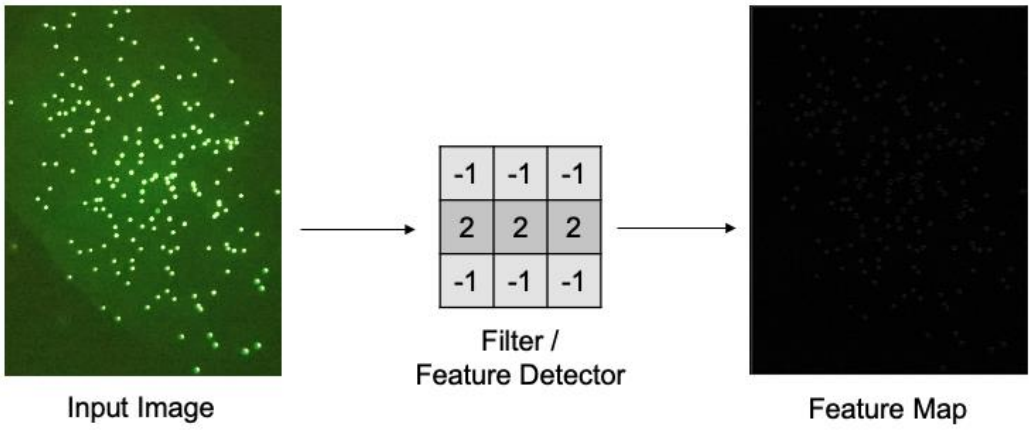


Figure 11: Input image convolved with horizontal line detector filter.

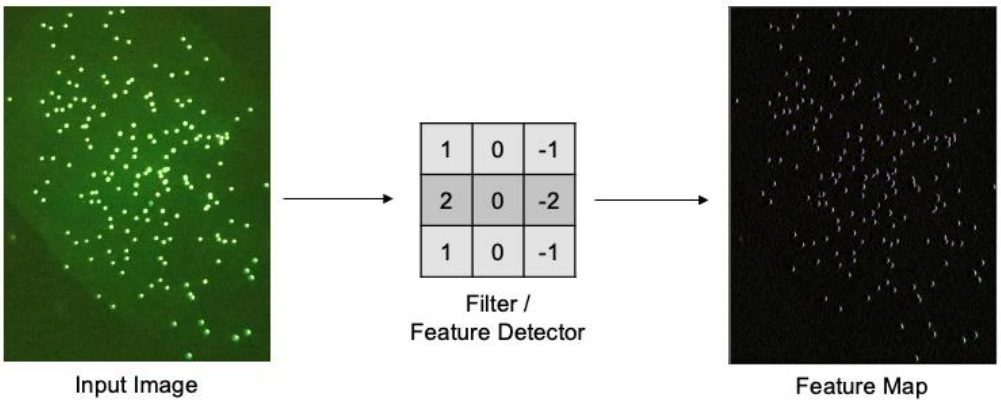


Figure 12: Input image convolved with Sobel filter.

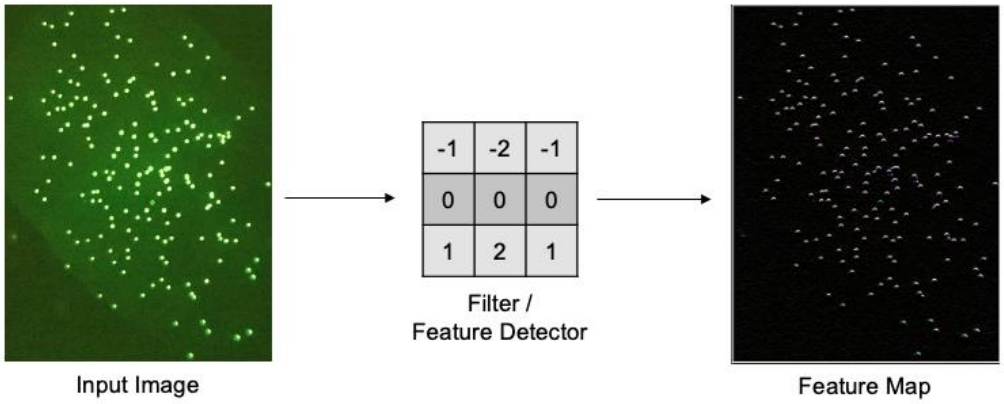


Figure 13: Input image convolved with Sobel filter.

With every convolution layer, neurons use these weights (highlighted region of the image) after convolution and ignore everything else in their receptive field. In this way a feature map is produced by using a same filter for a layer of neurons. For one particular feature map, all the neurons share the same weights and bias values. Similarly, for different feature maps, the neurons will have their respective weights and bias values. So, in the convolution layer, the computations take place in such a way that first it applies multiple filters to its incoming input. Due to its architecture, it can now detect multiple features irrespective of where it is present in the input. Once the neural network learns a feature or pattern from the convolved image in a particular location, it can learn the same feature or patterns in any other location. In Figure 14, we can see how multiple features are mapped together, and this undergoes next processes in the CNN for further prediction computations.

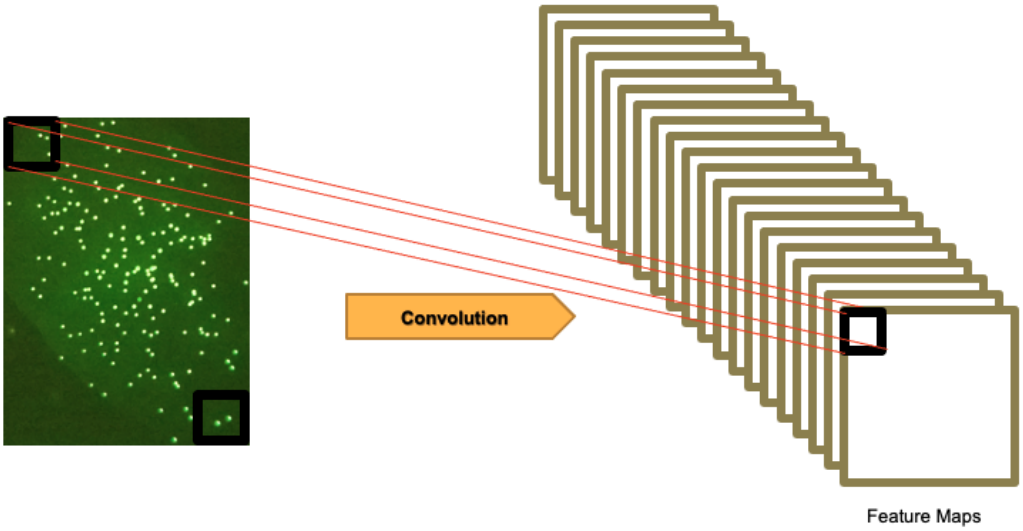


Figure 14: Feature maps stacked together after convolution.

4.2.2 Rectifier Linear Unit

The activation function used in this study is the ReLU Activation Function. ReLU is known for its faster computations.

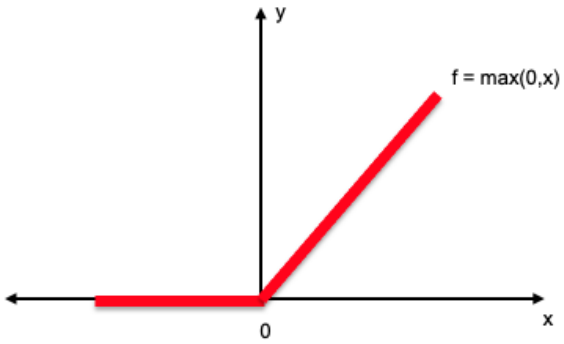


Figure 15: ReLU function graph.

This function reduces the non-linearity in the images. It basically sets all the negative values in the input to zero and every other positive value to as it is. Image input are

generally non-linear. After the feature maps are extracted during convolution process, there is a possibility of some negative values coming up in the computations. These values are made to zero. And sometimes when neurons are constantly set to zero or not activated for a long time, they become inactive and they are also known as *dying ReLUs*. For this problem, *leaky ReLU* is implemented as they are shown to outperform strict ReLU activation functions [33].

4.2.3 Pooling

The objective of pooling is to subsample the input image which reduces the dimension of the image. This reduces the network load and only important features are carried through the network. The number of parameters to train reduces thereby reducing the computational load and memory usage. This helps in avoiding the problem of over-fitting where the model performs very well in prediction task with training set of data but fails to keep up when fed with testing set of data which usually happen when the model is too complex for the problem with many features to train.

In this study, MaxPooling is used with a pooling kernel of 2X2. The pooling layer neurons have no weights. When this function is applied, the kernel computes over the image matrix and picks the highest value from the matrix it is striding over. And hence at the end of this operation, the dimensionality of the input matrix is reduced with only prominent features left to train from.

4.2.4 Regularization

Regularization is a technique to prevent the overfitting of data. Images are complex set of data with different channels. When they are fit into a neural network model with

millions of parameters, it tends to fit variety of complex input data that is fed to it. Although it performs well in training showing low loss, with testing dataset it may sometimes fails. To avoid this in convolution neural network, in this study we implement the technique of Dropout [34]. In this a neuron is dropped out temporarily along with its network connection randomly during training. This reduces inter dependency of every neuron to other neurons and the network eventually works better by avoiding the problem of overfitting.

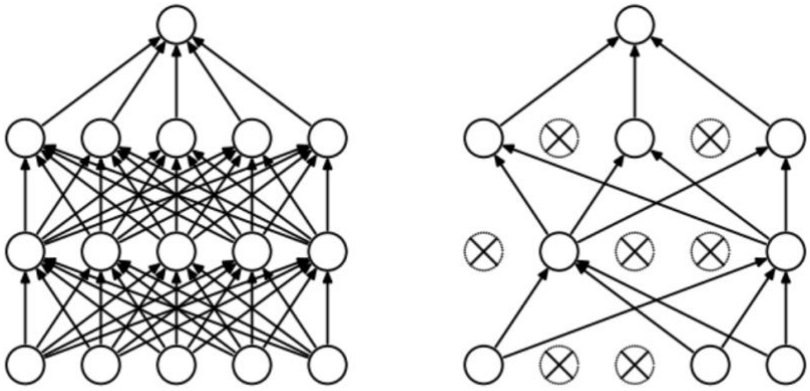


Figure 16: Preview of the Dropout mechanism.

4.2.5 Flattening and Fully Connected Layer (FCN)

After the feature maps are reduced, there are eventually flattened to a vector that is the input to the fully connected layer. The fully connected layer now combines all the features into attributes for better prediction. At this stage all the important features are condensed, and the model is set for the regression task.

4.2.6 Analysis metrics

In order to test the model's performance, the loss calculated using the Mean Square Error (MSE) regression loss function. It is calculated by the formula

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

CHAPTER 5

Results

To test the application of CNN on the smartphone acquired images, different kinds of datasets were divided into different stages of testing.

5.1 Stage I: Standard testing

CNN are known to down perform with low quality image data. To first test the performance, a set of 23 high quality images were chosen. Although ImageJ requires a lot of manual preprocessing and skill to extract the appropriate count from biomedical image, it is still powerful in labelling the cells. Hence in our study, we have used ImageJ count as the Ground Truth (GT) count for the supervised learning task.

In this testing the green fluorescent images from Samsung Galaxy S9+ were considered. The training set also included mask images of the experimental image data for data augmentation purposes. 23 images were split into 17 training images and 6 testing images (approx. 74% - 26% split ratio) with varying cell counts from 3600~90. After slicing the training and testing images, there were 5032 images in the training set and 1779 images in the testing set. The 5032 images in the training set were augmented with mask images totaling up to 10064 images in the training batch. In compilation, Adam compiler was used with learning rate=0.0001 initially as previous studies showed that these hyperparameters worked well with small object counting [32].

The model was trained with monitored MSE, 10% validation dataset that was allotted from the training and batch size set to 16. The model was then used to predict the cell count from the test images and correlation plot was drawn to compare the GT count and

the predicted count. We can observe the plot in Figure 18 with an R^2 value of 0.99 which shows a good correlation.

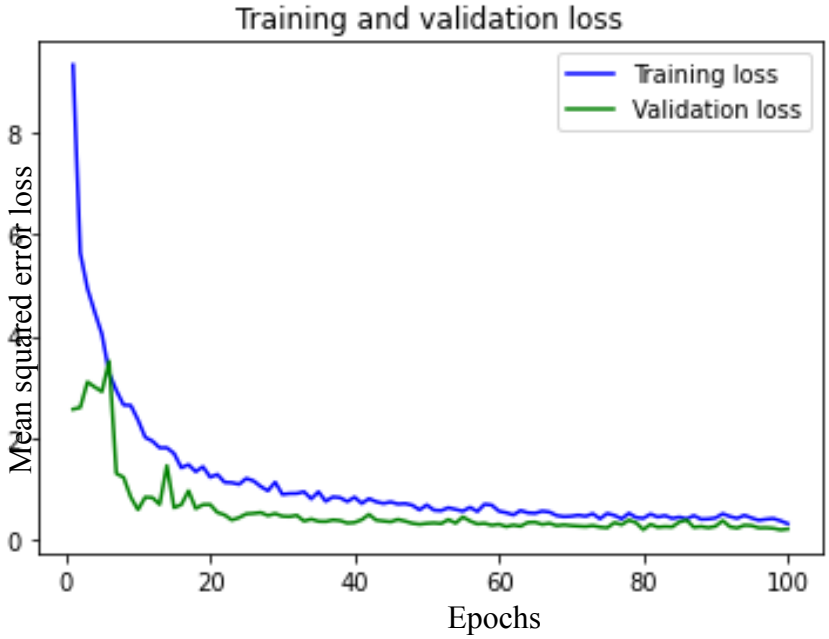


Figure 17: Training and validation plots at learning rate=0.0001.

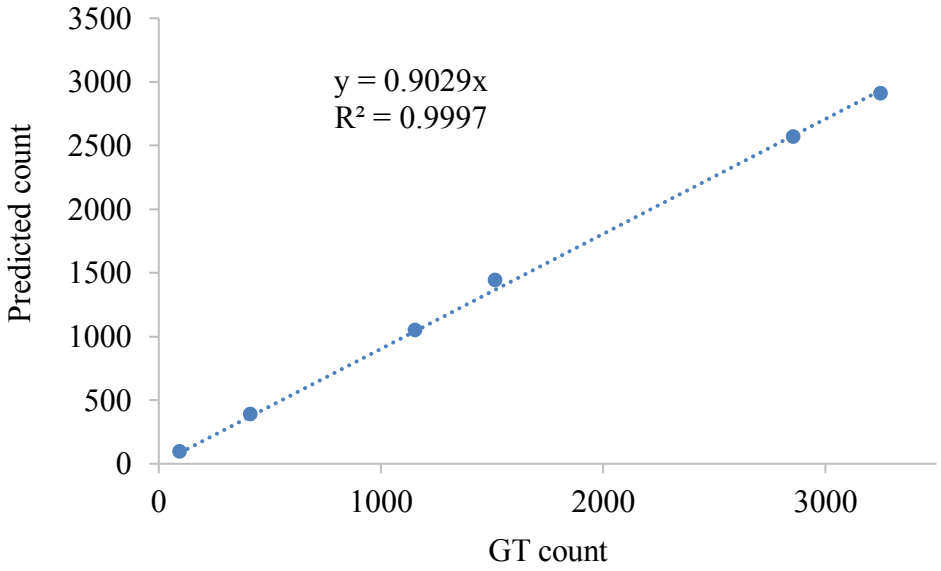


Figure 18: Correlation plot between the GT count from Image J and the predicted count from CNN model in figure 9A.

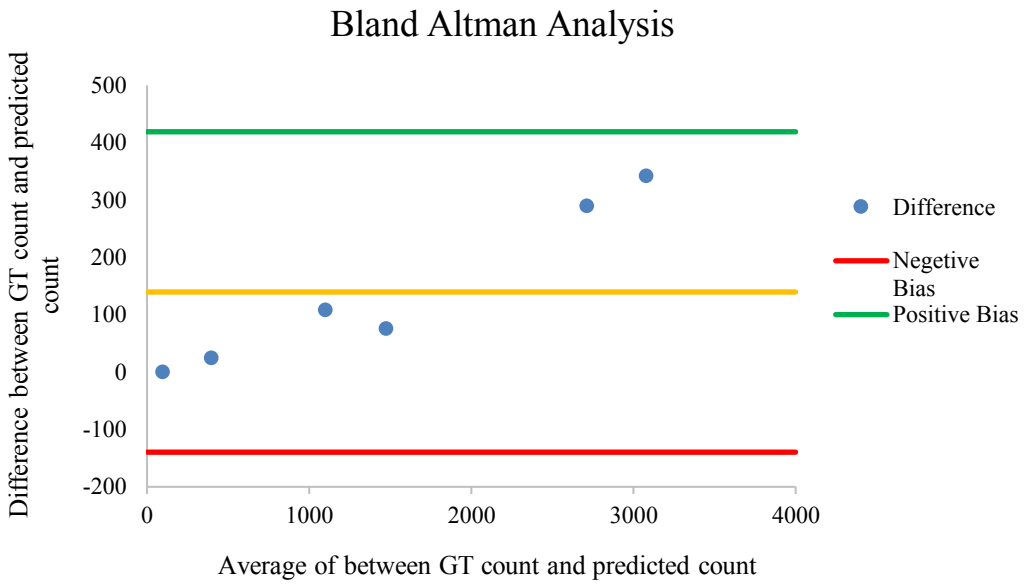


Figure 19: Bland Altman analysis of the GT count and predicted count from CNN.

In this stage we also tested out the convolution model by changing the convolution filters to 64X128X256 as shown in Figure 9B. The correlation plots for the results are as shown below in Figure 20 with R^2 value of 0.99.

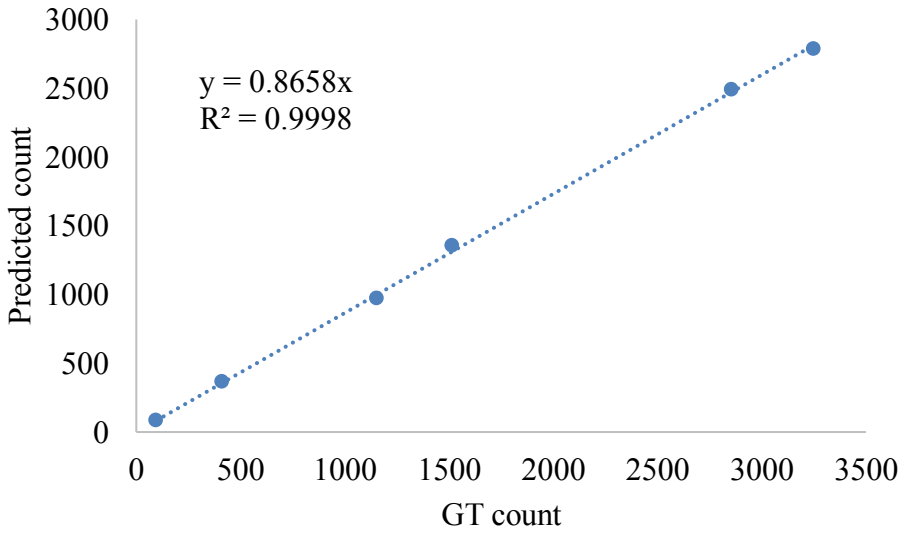


Figure 20: Correlation plot for ground truth count and prediction count for architecture in figure 9B.

We further studied to improve the model and observe the changes when hyperparameters like Learning Rate (LR) and Batch Size (BS) were tweaked. The graph of the MSE loss over the epochs are as shown below in Figure 21 for different learning rate parameters.

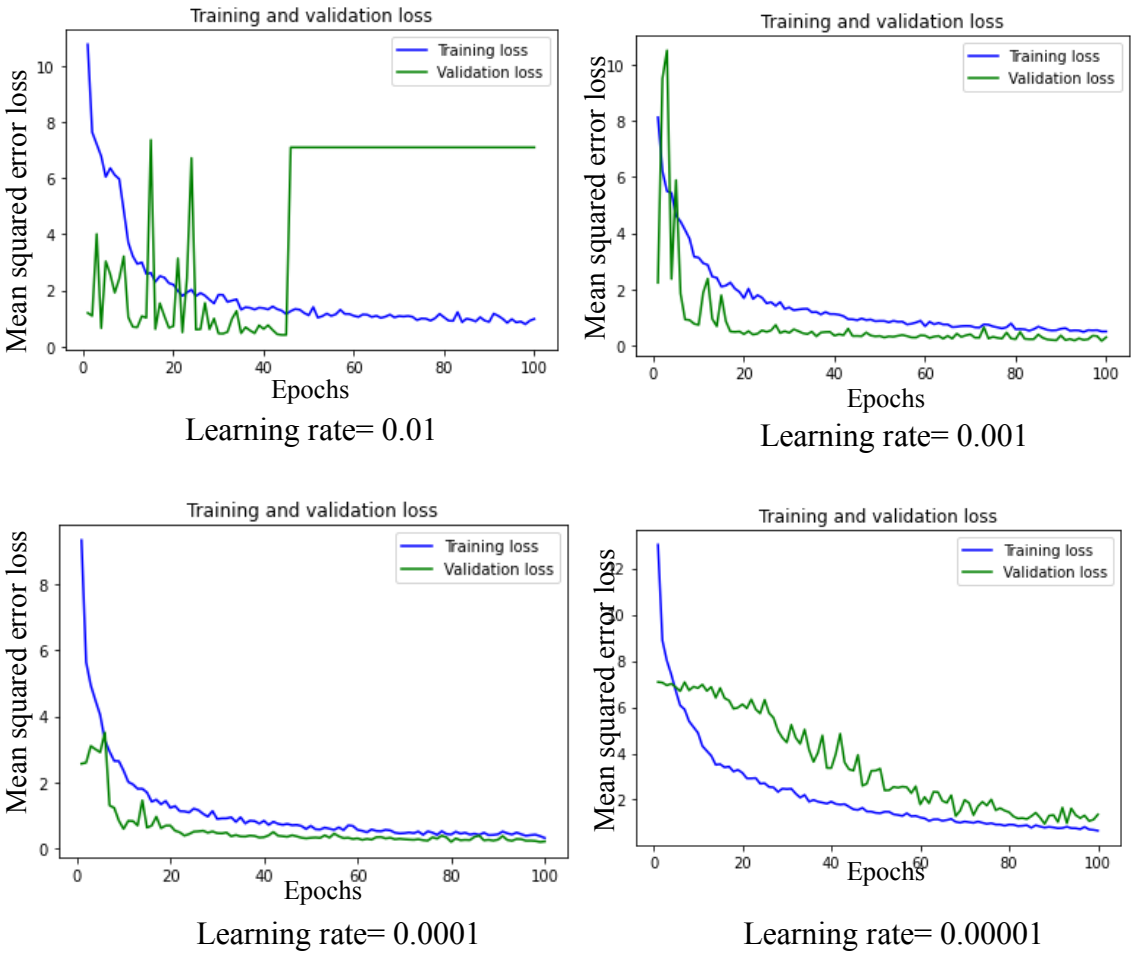


Figure 21: Training and validation loss curves for different learning rate hyperparameters.

We observe that for LR= 0.01, the model is over fitting, although the training loss is decreasing over epochs, the validation loss is increasing. For LR= 0.001 and 0.0001, the training and validation loss curve are within the acceptable bounds. For LR=

0.00001, we again see that the model is over fitting. The reason is that, in this case the LR is too small and it takes too long to converge at the global minimum.[35]

After finding suitable LR for this application, we studied how the tweaking the batch size would affect the results. After testing with default batch size=16, we tested for BS=8 and BS 32 and the following observations were made in correlations plots depicted in Figure 22 and 23.

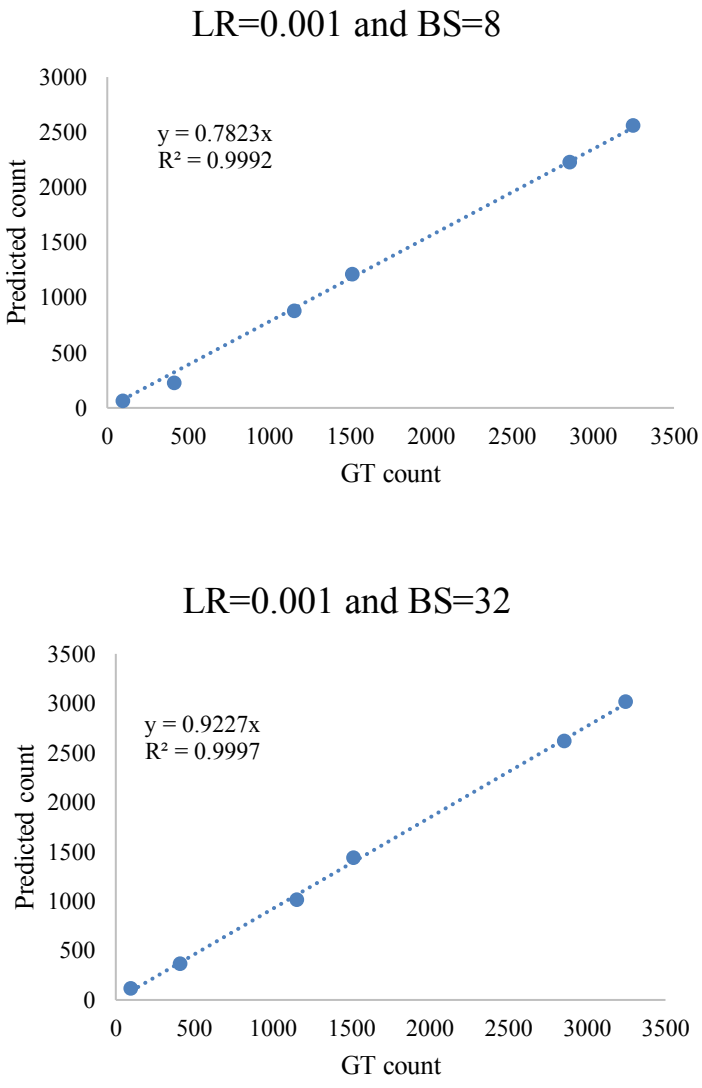


Figure 22: The correlation plots for varying batch size=8 and 32 given the LR=0.001.

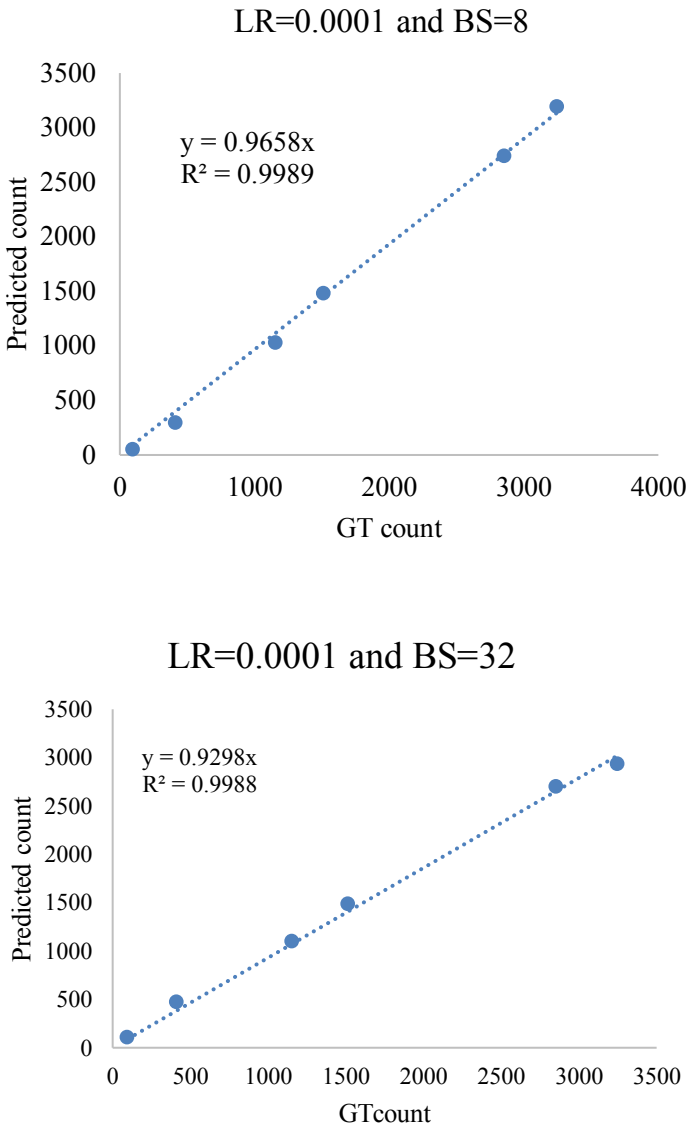


Figure 23: The correlation plots for varying batch size= 8 and 32 given the LR=0.0001.

From the above correlation plots we observe that LR=0.0001 is able to make the model fit more accurately when compared to other rates. When BS=8, we have a better y intercept value for LR=0.0001 when compared to LR=0.001 and also is able to predict higher cell count ranges better. But a BS=32 for LR=0.001 is giving better predictions than BS=8. However, for BS=8, the computation time is longer and for high amount of

data, it takes longer time to train and can become a bottleneck in real time applications. If the model had to only be trained once before deployment and there is room for long computation time in a given application, then lower BS and low LR can be implemented. For dynamic training purposes and since our idea of application has to be implemented in real time, requiring low computation power to successfully predict cell count from the images was the priority and LR value of 0.0001 and BS of 16 was kept standard for further testing.

5.2 Stage II: Cross validation

Although the dataset considered for the testing was based on random selection and training set contained different levels of cell counts, cross validation was performed to validate the results and stability of the CNN model. With one test, we can only concur that the model was able to perform well only when it was trained on the dataset that was chosen. For the cross validation, we employed a method where we randomized the selection of train and test data and until all the sample images were placed in the test set at least once by the randomization algorithm, the validation process continued. In such case, there were cases in which the sample fell in the test set more than once and counts were predicted more than once. The average of the those predicted counts were considered for evaluation. In this particular study, a total of 11 randomization took place to ensure all the data fell in the test set at least once. As a result, 11 models were produced.

In cross validation method, for dataset, we chose the images from the standard testing, binary mask of those images and images applied with gaussian blur for data augmentation purposes as shown in Figure 24. Some of the images acquired from

Nokia Lumia were also considered to analyze the model's behavior with low-quality images as shown in Figure 25.

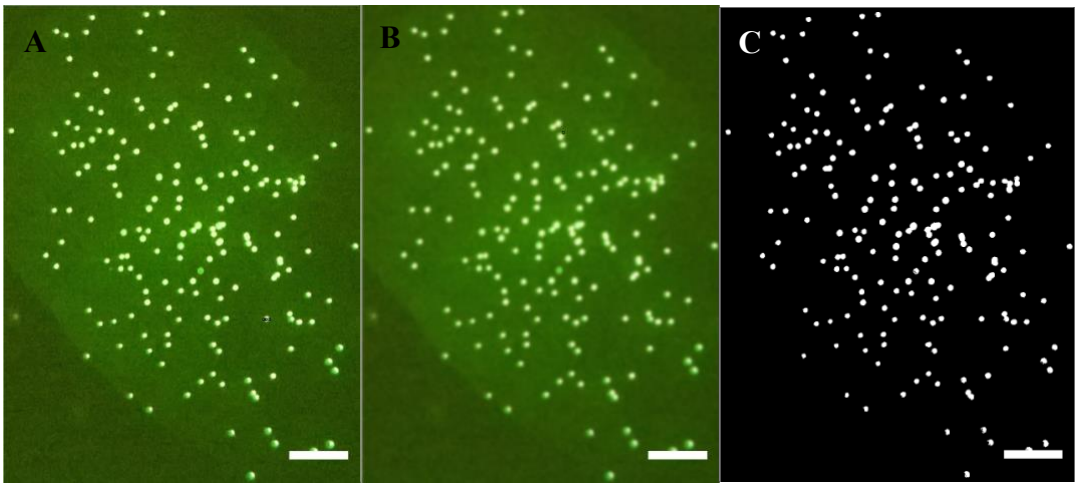


Figure 24: Images collected from Samsung S9+. (A) Regular image. (B) Image with Gaussian Blur, sigma=3. (C) Image applied with binary mask (scale bar = 100µm).

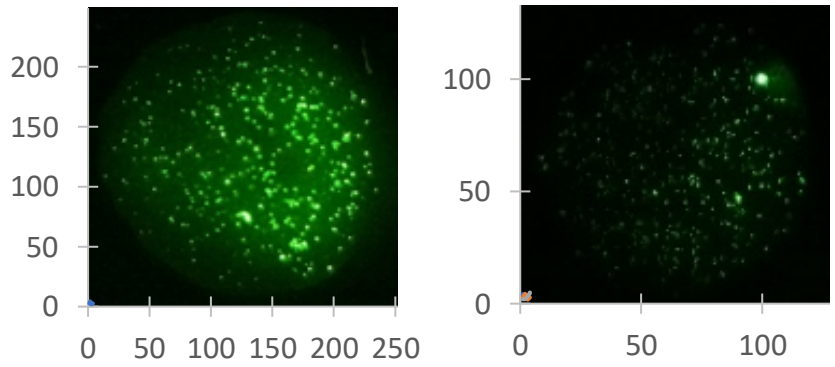


Figure 25: Images collected from Nokia Lumia 1020.

A total of 33 images were considered for the process and the analysis was repeated until all 33 images were in the test set at least once. The correlation plot in Figure 26 gives an R^2 value of 0.99 and Figure 27 depicts the Bland Altman analysis.

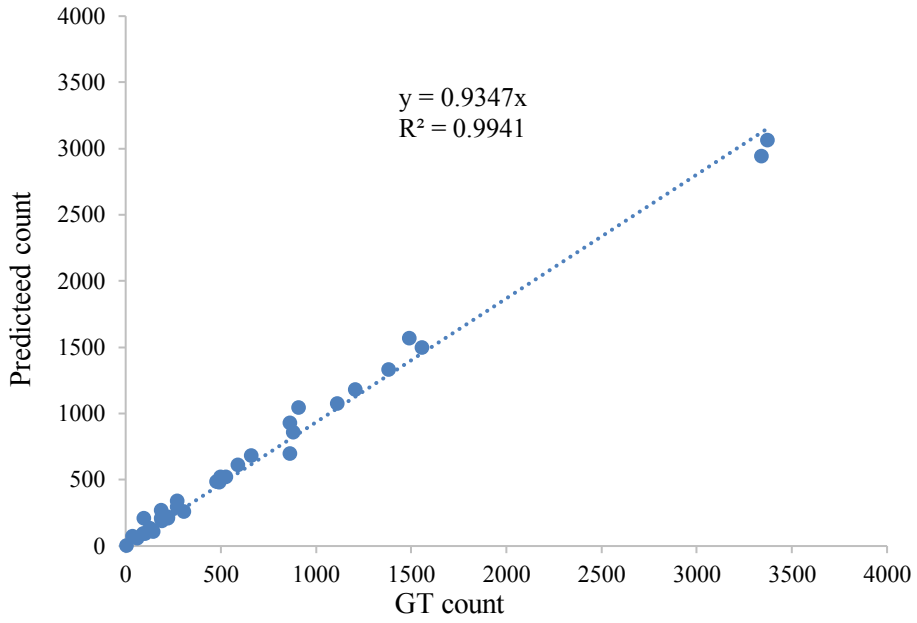


Figure 26: Correlation plot of GT count and predicted count.

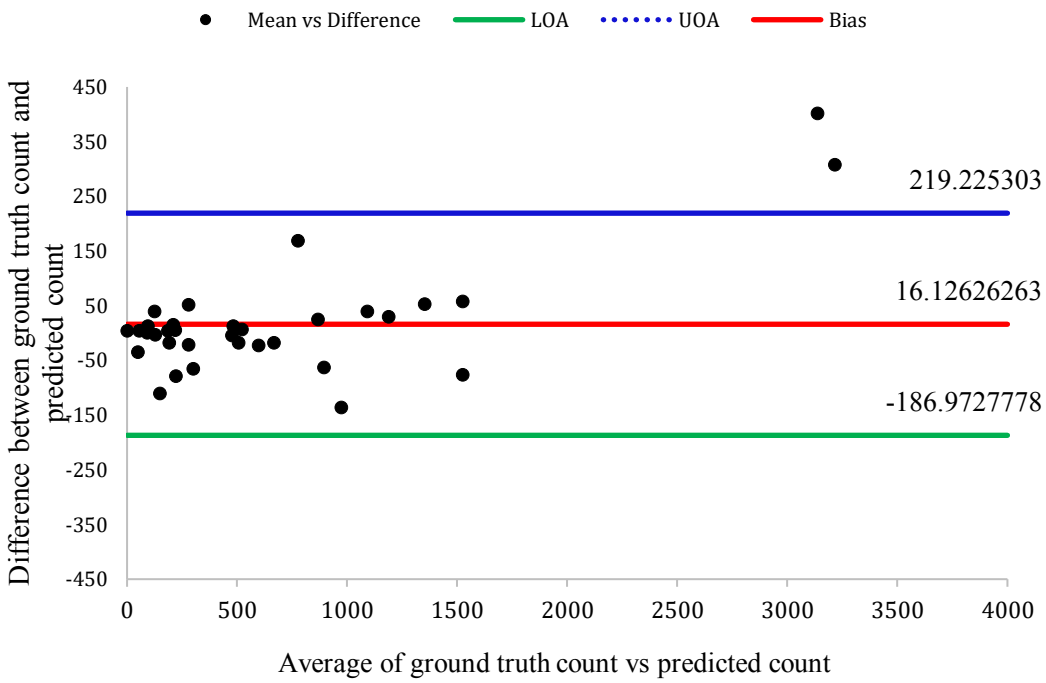


Figure 27: Bland Altman Analysis for GT count and predicted count from cross validation.

5.3 Stage III: Red beads testing

From the results we could concur that when the model was trained on particular kind of fluorescent images, it was able to predict the count from the same kind of fluorescent images. The aim is to standardize the device and implement AI pipeline and obtain particle quantification from the images captured by the smartphone. To test this, we made use of a trained model from the cross-validation result and used these weights to predict the cell count from red beads image after binary mask was applied to it. As the training set contained masked images from green fluorescent beads image, the cross-correlation result for 7 test images was R^2= 0.99.

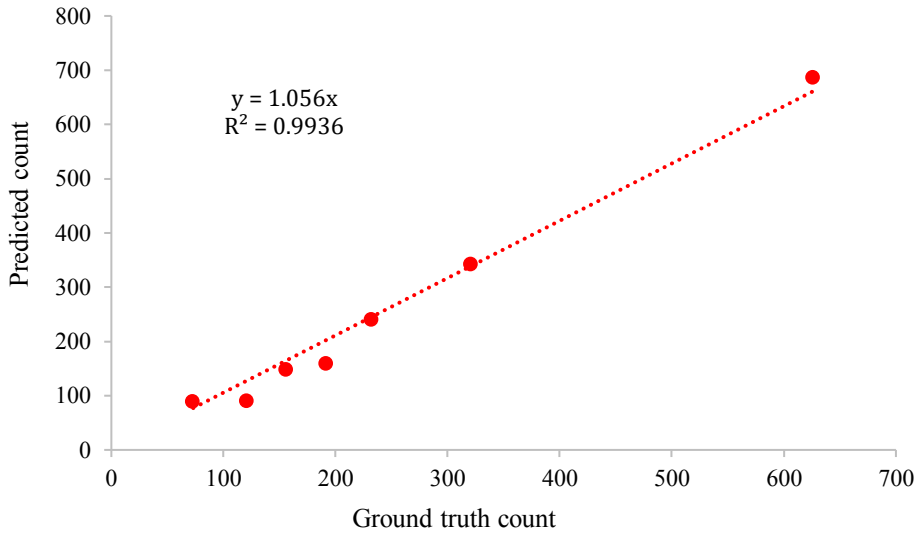


Figure 28: Correlation plot of GT count and prediction count from binary masked red beads images.

5.4 Analysis of Results

From the correlation plot, we have the R^2 value of 0.99 for all the testing. By analyzing the errors bars of the cross-validation results and standard testing results (Lr=0.0001 and BS=8) we notice that the predicted values can be above and below the ground truth.

One of the differences in standard testing and cross-validation testing is the number of datasets used. Cross validation in addition was implemented with blur images.

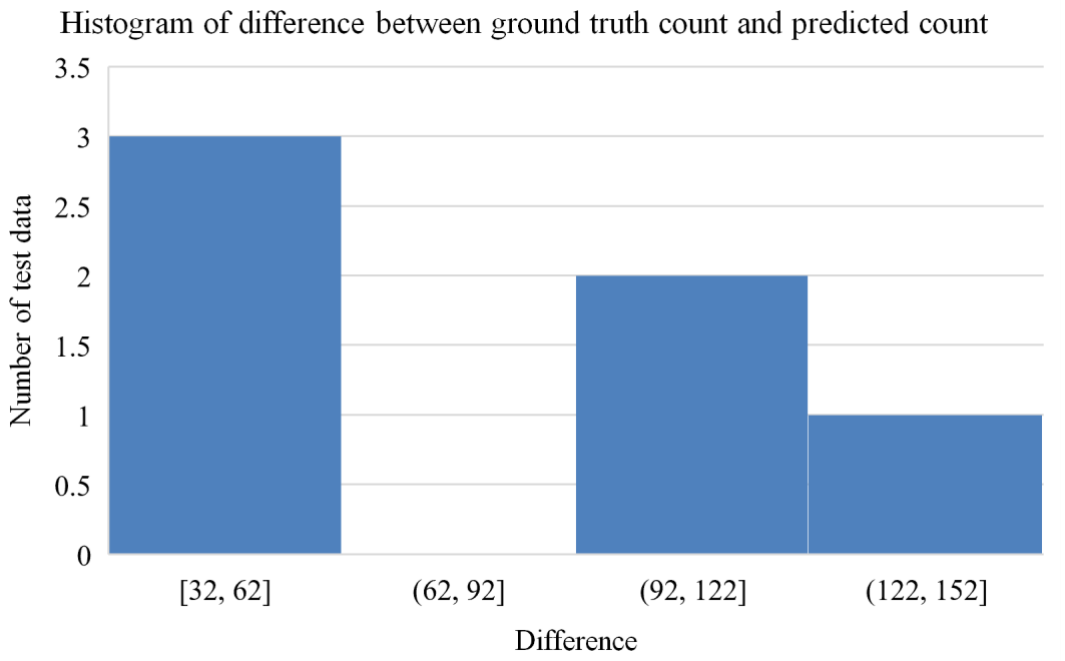


Figure 29: Histogram plot of difference between ground truth and predicted count from standard testing results

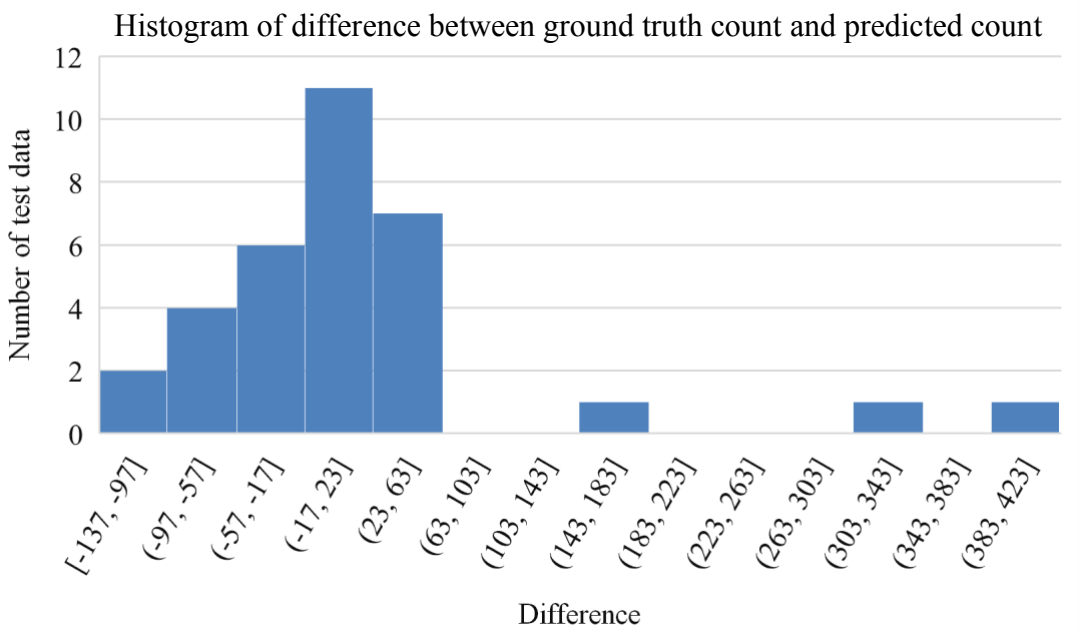


Figure 30: Histogram of the difference between ground truth and predicted count from cross-validation testing

The number of images used in standard testing is 23 images and the number of images used in cross-validation is 33 images along with gaussian blurred images of sigma=3. We observe from the cross-validation results that the prediction range is between -137 to 423 which implies that for a particular test image the predicted count was 137 higher than the ground truth count and 423 lower than ground truth count. Also, from the correlation plots of red beads testing we see observe that there is approximately 5% overcounting. From dataset preparation, the images are sliced to feed the neural network along with the ground truth count. ImageJ calculates our ground truth by analyzing the particles based on thresholding. This rises a possibility of counting a particular cell twice if it was sliced into two images or present on the boundary of slicing. Thus, we propose to improve it by labeling the center coordinate of a cell in an image and obtaining the total count by the no of coordinates in the image. The possibility of a single coordinate getting sliced is low, this can ensure to overcome the over counted cell problem while slicing the image. Overall, we see an improvement in the accuracy when the datasets used for training was increased in cross validation compared to standard testing. Hence, by improving data augmentation techniques and adding more images to train the model, the prediction can be improved thus improving the accuracy.

CHAPTER 6

Deployment of the Deep Learning Model into an Application Programming Interface

The device key aspect is to see its performance in real time applications as a point of care diagnostic sensor. As discussed in Chapter 1, we saw how mobile phones are flexible to be integrated with advance APIs. To show a demonstration of this, we developed a Web based API that is deployed on the Google Cloud Platform (GCP).

Flask is popular web application framework and is a lightweight Web Server Gateway Interface (WSGI). It depends on the Jinja template engine and Werkzeug WSGI toolkit. Since the deep learning computations were done using tools like TensorFlow, Keras and Python, hence Flask framework for deployment of the Keras model was most suited.

HTML assisted with Cascading Style Sheets (CSS) were used to develop the front-end API.

Link: Cell Count App

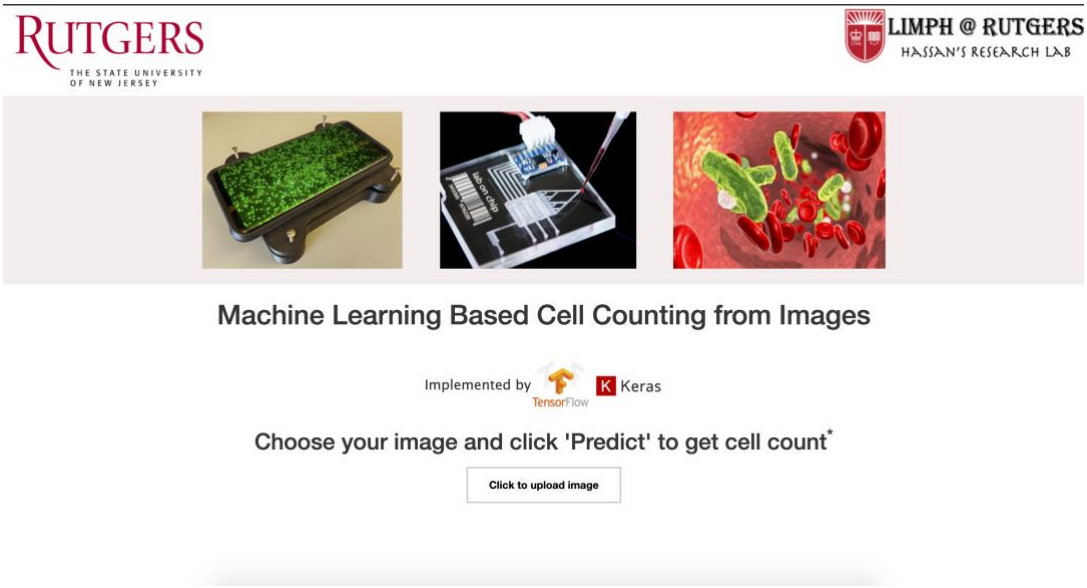


Figure 31: Preview of the web application.

Since the deployment is made on a cloud-based platform and published to a larger audience, this can aid in monitoring the user data and uploads leading to a bigger medical data science project. The uploads that are new data can again be used in training the model to obtain better prediction as Deep Learning models work better when trained on a larger set of data. In this case, a progressive validation is done to monitor the model's performance rather than using predefined set of training and testing images. This dynamic model has the potential to improve the field of point of care diagnostics with its user-friendly device and quantification methodology which is powered by AI.

CAPTER 7

Conclusion and Future Scope

We have presented a methodology for applying a convolutional neural network framework to predict the cell count from an image acquired from smartphone based fluorescent microscopy. This study makes use of the novel design and presents an Artificial Intelligence pipeline to integrate it with. We tested two different convolutional frameworks, and both provided satisfactory results in predicting the cell count from an image. To further improve the model, we worked by tweaking the dependent hyperparameters and found satisfactory range of values to work with for a better model's performance.

To complete the lifecycle of the AI implementation, the trained Keras model weights were deployed on an application programmable interface which was a web application running on google cloud platform. Further, the scope is to develop a dynamic pipeline in which new data used for prediction on the web applications can further be used to train and update the model. To implement it in a low resource setting, we aim to further develop a mobile application which can be installed in the smartphone device that captures the image and use the weights calculated during the training to evaluate the cell count from the acquired image. This has the added advantage to use the application without the web interface. This technology thus shows the potential of an AI integrated Point-of-Care diagnostic biosensor.

References

1. Kumar, S., Kumar, S., Ali, M.A., Anand, P., Agrawal, V.V., John, R., Maji, S. and Malhotra, B.D. (2013), Microfluidic-integrated biosensors: Prospects for point-of-care diagnostics. Biotechnology Journal, 8: 1267-1279.
2. Urdea, M., Penny, L., Olmsted, S. et al. Requirements for high impact diagnostics in the developing world. Nature 444, 73–79 (2006).
3. Foudeh AM, Fatanat Didar T, Veres T, Tabrizian M. Microfluidic designs and techniques using lab-on-a-chip devices for pathogen detection for point-of-care diagnostics. Lab Chip. 2012 Sep 21;12(18):3249-66. doi: 10.1039/c2lc40630f. Epub 2012 Aug 2. PMID: 22859057.
4. Daw, R., Finkelstein, J. Lab on a chip. Nature 442, 367 (2006).
5. C. D. Chin, V. Linder and S. K. Sia, Lab-on-a-chip devices for global health: past studies and future opportunities, Lab Chip, 2006, 7(1), 41–57
6. Huang, X., Xu, D., Chen, J., Liu, J., Li, Y., Song, J., . . . Guo, J. (2018). Smartphone-based analytical biosensors. Analyst, 143(22), 5339-5351. doi:10.1039/C8AN01269E
7. Guo, J. (2016). Uric Acid Monitoring with a Smartphone as the Electrochemical Analyzer. Analytical Chemistry, 88(24), 11986-11989. doi: 10.1021/acs.analchem.6b04345
8. S. K. Jain and B. Bhaumik, "An Energy Efficient ECG Signal Processor Detecting Cardiovascular Diseases on Smartphone," in IEEE Transactions on Biomedical Circuits and Systems, vol. 11, no. 2, pp. 314-323, April 2017, doi: 10.1109/TBCAS.2016.2592382.
9. Karlsen H, Dong T. Smartphone-Based Rapid Screening of Urinary Biomarkers. IEEE Trans Biomed Circuits Syst. 2017 Apr;11(2):455-463. doi: 10.1109/TBCAS.2016.2633508. Epub 2017 Mar 15. PMID: 28320676.
10. Liang, P. S., Park, T. S., & Yoon, J. Y. (2014). Rapid and reagentless detection of microbial contamination within meat utilizing a smartphone-based biosensor. Scientific reports, 4, 5953.
11. Yanli Lu, Zhenghan Shi, Qingjun Liu, Smartphone-based biosensors for portable food evaluation, Current Opinion in Food Science,Volume 28,2019, Pages 74-81, ISSN 2214-7993.
12. Junjie Liu, Zhaoxin Geng, Zhiyuan Fan, Jian Liu, Hongda Chen, Point-of-care testing based on smartphone: The current state-of-the-art (2017–2018), Biosensors and Bioelectronics, Volume 132, 2019, Pages 17-37, ISSN 0956-5663.
13. Aldo Roda, Elisa Michelini, Martina Zangheri, Massimo Di Fusco, Donato Calabria, Patrizia Simoni, Smartphone-based biosensors: A critical review and perspectives, TrAC Trends in Analytical Chemistry, Volume 79, 2016, Pages 317-325, ISSN 0165-9936.
14. Sarker, I.H., Hoque, M.M., Uddin, M.K. et al. Mobile Data Science and Intelligent Apps: Concepts, AI-Based Modeling and Research Directions. Mobile Network Appl (2020).
15. Pang B, Nijkamp E, Wu YN. Deep Learning with TensorFlow: A Review. Journal of Educational and Behavioral Statistics. 2020;45(2):227-248. doi:10.3102/1076998619872761
16. M. A. Sami, K. Wagner, P. Parikh and U. Hassan, "Smartphone Based Microfluidic Biosensor for Leukocyte Quantification at the Point-of-Care," 2019 IEEE Healthcare Innovations and Point of Care Technologies, (HI-POCT), Bethesda, MD, USA, 2019, pp. 119-122, doi: 10.1109/HI-POCT45284.2019.8962697

17. Retrieved from https://docs.openmicroscopy.org/bio-formats/6.0.1/supported-formats.html
18. Charles Boncelet, Chapter 7 - Image Noise Models, Editor(s): Al Bovik, The Essential Guide to Image Processing, Academic Press, 2009, Pages 143-167, ISBN 9780123744579
19. McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943).
20. GeÌ•ron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.
21. A. Krizhevsky, I. Sutskever and G. Hinton, "Imagenet classification with deep convolutional neural networks", Proc. Neural Inf. Process. Syst., 2012.
22. Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556.
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.
24. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.
25. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI.
26. Gong, E., Pauly, J.M., Wintermark, M. and Zaharchuk, G. (2018), Deep learning enables reduced gadolinium dose for contrast-enhanced brain MRI. J. Magn. Reson. Imaging, 48: 330-340. https://doi.org/10.1002/jmri.25970
27. Alexander Selvikvåg Lundervold, Arvid Lundervold, an overview of deep learning in medical imaging focusing on MRI, Zeitschrift für Medizinische Physik, Volume 29, Issue 2, 2019, Pages 102-127, ISSN 0939-3889.
28. Akkus, Z., Galimzianova, A., Hoogi, A. et al. Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions. J Digit Imaging 30, 449–459 (2017).
29. Sydney, R.C., Centre, C.M., Institute, S.C., & Hbku (2019). Deep Learning for Anomaly Detection: A Survey.
30. Yao Xue and Nilanjan Ray. Cell detection with deep convolutional neural network and compressed sensing. ArXiv, abs/1708.03307, 2017.
31. Tan R., Zhang J., Chen P., Wang B., Xia Y. (2018) Cells Counting with Convolutional Neural Network. In: Huang DS., Gromiha M., Han K., Hussain A. (eds) Intelligent Computing Methodologies. ICIC 2018. Lecture Notes in Computer Science, vol 10956. Springer, Cham.
32. Weidi Xie, J. Alison Noble & Andrew Zisserman (2018) Microscopy cell counting and detection with fully convolutional regression networks, Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 6:3, 283-292, DOI: 10.1080/21681163.2016.1149104
33. Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. ArXiv, abs/1505.00853.
34. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. ArXiv, abs/1207.0580.
35. Ruder, S. (2016). An overview of gradient descent optimization algorithms. ArXiv, abs/1609.04747.