



**Strathmore**  
UNIVERSITY

Strathmore University  
**SU+ @ Strathmore**  
University Library

---

**Electronic Theses and Dissertations**

2018

# Circulating tumor identification using neural networks for monitoring Cancer progression

Stephen O. Oketch  
*Faculty of Information Technology (FIT)*  
*Strathmore University*

Follow this and additional works at <https://su-plus.strathmore.edu/handle/11071/5999>

## Recommended Citation

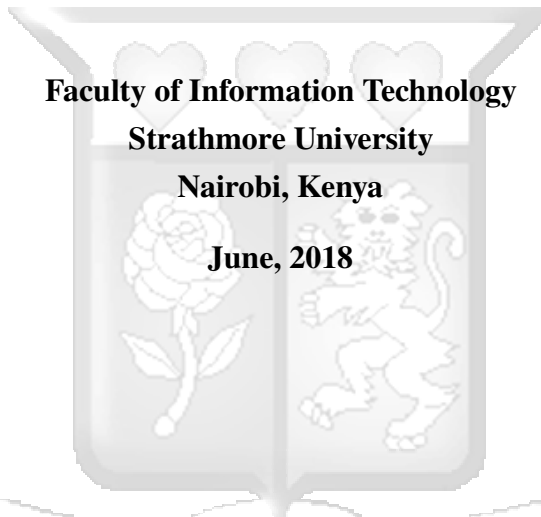
Oketch, S. O. (2018). *Circulating tumor identification using neural networks for monitoring Cancer progression* (Thesis). Strathmore University. Retrieved from <https://su-plus.strathmore.edu/handle/11071/5999>

This Thesis - Open Access is brought to you for free and open access by DSpace @Strathmore University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DSpace @Strathmore University. For more information, please contact [librarian@strathmore.edu](mailto:librarian@strathmore.edu)

**CIRCULATING TUMOR IDENTIFICATION USING NEURAL NETWORKS FOR  
MONITORING CANCER PROGRESSION**

**Stephen Oketch Obonyo  
096573**

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in  
Information Technology at Strathmore University.



**Faculty of Information Technology  
Strathmore University  
Nairobi, Kenya  
June, 2018**

This thesis is available for Library use on the understanding that it is copyright material and that  
no quotation from the thesis may be published without proper acknowledgement. .

## Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself. No part of this thesis may be reproduced without the permission of the author and Strathmore University.

.....

.....

.....



## Approval

The thesis of Stephen Oketch Obonyo was reviewed and approved by the following

Dr. Joseph Orero  
Senior Lecturer, Faculty of Information Technology  
Strathmore University

## Abstract

Cancer is the third most killer disease in Kenya after infectious and cardiovascular diseases. It contributes to a significant portion of annual national deaths, led by breast and prostate cancer. Existing cancer treatment methods vary from patient to another based on the type and stage of tumor development. The treatment modalities such as surgery, chemotherapy and radiation have been successful when the disease is detected early and constantly monitored. Ineffective treatment method and development of complications such as cancer relapse must be monitored as they are likely to cause more deaths. Detection of circulating tumor cells (CTC's) is a pivotal monitoring method which involves identification of cancer related substances known as tumor markers. These are often excreted by primary tumors into patient's blood. The presence, absence or number of CTC's can be used to evaluate patient's disease progression and determine the effectiveness of current treatment option. This research work proposed an adaptive learning-based, computational model to help in cancer monitoring. It identifies and enumerates CTC's based on the auto-learned features from stained CTC images using deep learning methodology. The 3.0% error rate model, without human intervention, automatically learned the best set of representative features from labelled samples. The representations were used in enumerating and identifying CTC's given a new test example.

**Keywords:** CTC Detection, CTC Enumeration, Neural Networks, Deep Learning



# Table of Contents

	<b>Page</b>
<b>Declaration</b>	<b>i</b>
<b>Approval</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>x</b>
<b>Definition of Terms</b>	<b>xiii</b>
<b>Acknowledgment</b>	<b>xiv</b>
<b>Dedication</b>	<b>xv</b>
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem statement . . . . .	2
1.2.1 Objectives . . . . .	3
1.2.1.1 General Objective . . . . .	3
1.2.1.2 Specific Objectives . . . . .	3
1.3 Research Questions . . . . .	3
1.4 Justification . . . . .	4
1.5 Scope and Limitation . . . . .	4
<b>Chapter 2: Literature Review</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Cancer Treatment Monitoring . . . . .	5
2.2.1 Cancer Monitoring Methods . . . . .	5
2.2.1.1 Conventional Cancer Monitoring . . . . .	5
2.2.1.2 Biopsy . . . . .	8
2.2.1.3 Liquid Biopsy . . . . .	9
2.2.1.4 Diffusion-Weighted MRI (DW-MRI) . . . . .	13
2.2.1.5 Mobile Application Symptom Monitoring . . . . .	14
2.2.1.6 Cancer Spread Prediction Using Deep Learning . . . . .	16
2.2.1.7 Limitation of the Current Monitoring Approaches . . . . .	16
2.3 Application of Machine Learning in Cancer Monitoring . . . . .	17
2.3.1 Machine Learning . . . . .	17
2.3.2 Machine Learning Application in Cancer Monitoring . . . . .	17
2.3.3 Machine Learning Algorithms . . . . .	19

2.3.3.1	Feed Forward Neural Networks . . . . .	19
2.3.3.2	Convolutional Neural Network (ConvNet) . . . . .	21
2.3.3.3	K-nearest Neighbors (kNN) . . . . .	25
2.3.3.4	Naive Bayes . . . . .	25
2.3.3.5	Support Vector Machine . . . . .	26
2.3.3.6	Decision Tree . . . . .	27
2.4	Feature Selection from CTC Images . . . . .	28
2.4.1	Manual CTC Feature Extraction . . . . .	29
2.4.2	Automated Feature Extraction . . . . .	29
2.5	Conceptual Framework . . . . .	29
<b>Chapter 3:</b>	<b>Methodology</b>	<b>31</b>
3.1	Introduction . . . . .	31
3.2	Software Development Methodology . . . . .	31
3.3	Research Design . . . . .	32
3.3.1	Research Framework . . . . .	32
3.3.2	Data Acquisition . . . . .	33
3.3.3	Image Preprocessing . . . . .	34
3.3.4	Data Splitting . . . . .	34
3.3.5	Feature Extraction . . . . .	35
3.3.6	Neural Network/ConvNet Architecture . . . . .	35
3.3.6.1	Loading Data . . . . .	36
3.3.6.2	Convolution, Activation and Pooling . . . . .	36
3.3.6.3	Fully Connected Layers . . . . .	37
3.3.6.4	Training . . . . .	37
3.3.7	Simple Classical Models . . . . .	37
3.3.7.1	Model Testing and Performance Evaluation . . . . .	37
3.4	Research Quality, Validity and Reliability . . . . .	38
3.5	Ethics Considerations . . . . .	38
<b>Chapter 4:</b>	<b>System Analysis, Design and Architecture</b>	<b>39</b>
4.1	Introduction . . . . .	39
4.2	Requirement Analysis . . . . .	39
4.2.1	Functional Requirements . . . . .	39
4.2.2	Non-Functional Requirement . . . . .	39
4.2.2.1	Ease of re-training . . . . .	39
4.2.2.2	Robust generalization . . . . .	39

4.2.2.3	Security . . . . .	39
4.2.2.4	Persistent Weight Storage . . . . .	40
4.2.2.5	Adaptation . . . . .	40
4.3	System Design and Architecture . . . . .	40
4.3.1	Partial Domain Model . . . . .	40
4.3.2	Use Case Diagram . . . . .	41
4.3.3	Use Case Basic Flows . . . . .	42
4.3.4	Data Flow Diagram . . . . .	43
4.3.4.1	Context Diagram . . . . .	44
4.3.4.2	Level 0 DFD . . . . .	44
4.3.5	Sequence Diagram . . . . .	45
4.3.6	Class Diagram . . . . .	46
<b>Chapter 5:</b>	<b>System Implementation and Testing</b>	<b>48</b>
5.1	Introduction . . . . .	48
5.2	Hardware & Software Environment . . . . .	48
5.3	Training Dataset Generation and Preprocessing . . . . .	48
5.4	Training Machine Learning Models . . . . .	52
5.4.1	Simple Classical Models . . . . .	52
5.4.1.1	Support Vector Machines (SVM) . . . . .	52
5.4.1.2	Random Forest . . . . .	53
5.4.1.3	Multinomial Naive Bayes . . . . .	53
5.5	Deep Learning Models . . . . .	53
5.6	Multilayer Perceptron (MLP) . . . . .	53
5.7	Convolutional Neural Network (ConvNet) . . . . .	54
5.8	Capsule Neural Networks (CapsNet) . . . . .	56
5.9	Convolutional Neural Network for Regression . . . . .	57
5.10	Model Evaluation . . . . .	58
5.11	Model Selection . . . . .	59
5.12	Testing New Instance . . . . .	59
<b>Chapter 6:</b>	<b>Discussions</b>	<b>61</b>
6.1	Introduction . . . . .	61
6.2	Model Selection . . . . .	61
6.2.1	CTC Classification Models . . . . .	61
6.2.2	CTC Enumeration Model . . . . .	63
6.3	Research Findings . . . . .	63

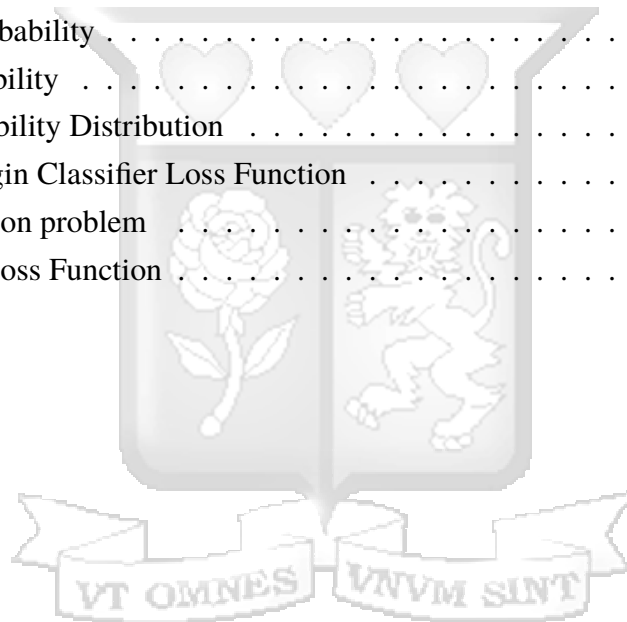
6.3.1	Cancer Monitoring Methods . . . . .	63
6.3.2	Machine Learning Models for Cancer Monitoring . . . . .	63
6.3.3	Extracting Features from CTC's . . . . .	64
6.3.4	Developing and testing the model . . . . .	64
6.4	Model Validation . . . . .	65
6.5	Conclusion . . . . .	65
<b>Chapter 7: Conclusions, Recommendations and Future Work</b>		<b>66</b>
7.1	Conclusion . . . . .	66
7.2	Challenges . . . . .	66
7.3	Recommendations . . . . .	67
7.4	Future Work . . . . .	67
<b>References</b>		<b>69</b>





# List of Equations

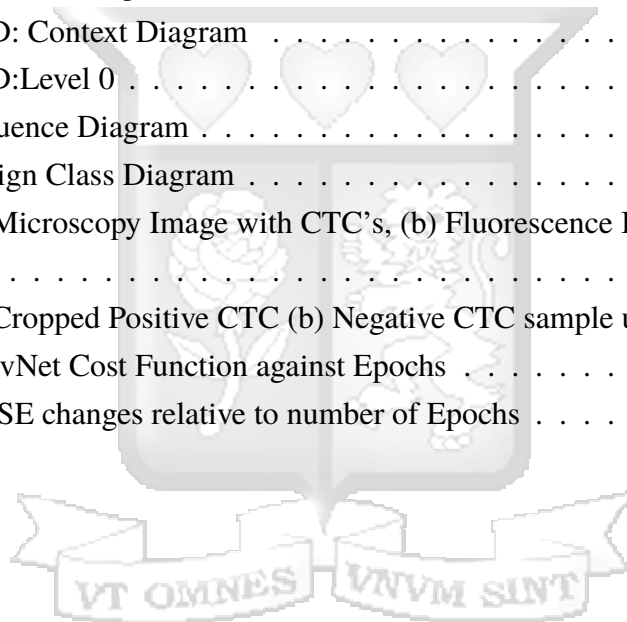
1	Forward Propagation . . . . .	20
2	Network Sigmoid Activation . . . . .	20
3	Network Prediction . . . . .	20
4	Mean Squared Loss Function . . . . .	21
5	Convex Optimization . . . . .	21
7	Partial Derivatives . . . . .	21
8	Softmax Network Activation . . . . .	24
9	Manhattan Distance . . . . .	25
10	Euclidean Distance . . . . .	25
11	Conditional Probability . . . . .	25
12	Posterior Probability . . . . .	26
13	Gaussian Probability Distribution . . . . .	26
14	Maximum Margin Classifier Loss Function . . . . .	27
15	SVM optimization problem . . . . .	27
18	Decision Tree Loss Function . . . . .	28



## List of Figures

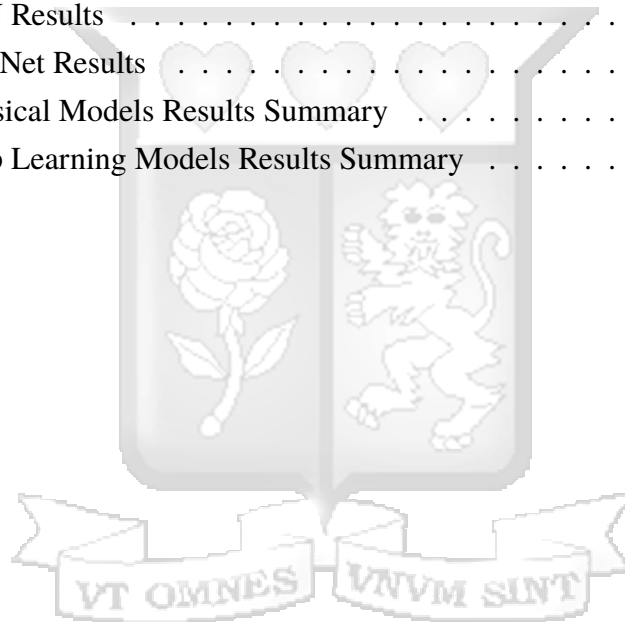
Figure 2.2.1	A cross-sectional CT scan sample. Adapted from ( <i>Diagnosis: CT Scan</i> , 2007)) . . . . .	6
Figure 2.2.2	PET/CT scan samples of Esophageal Cancer Patient . Adapted from (Akiyama et al., 2015)) . . . . .	7
Figure 2.2.3	MRI sample from Breast Cancer Patient. (Adapted from (U. Patel, Patel, Cobb, Benkers, & Vermeulen, 2014)) . . . . .	8
Figure 2.2.4	Biopsy; using a special needle to extract tissue from body. (Adapted from ( <i>Liquid Biopsy Predicts Colon Cancer Recurrence</i> , 2016)) . . . . .	9
Figure 2.2.5	Liquid Biopsy; Detecting Calculating DNA(ctDNA) through Blood Test (Adapted from (Crowley, Di Nicolantonio, Loupakis, & Bardelli, 2013)) . . . . .	10
Figure 2.2.6	Liquid Biopsy Processes.(Adapted from ( <i>Liquid Biopsy Predicts Colon Cancer Recurrence</i> , 2016)) . . . . .	10
Figure 2.2.7	CTC prepping. Magnetic Force Pull Localization (Adapted from ( <i>CELLSEARCH System Overview CellSearch</i> , 2018)) . . . . .	11
Figure 2.2.8	CTC Liquid Biopsy Using CellSearch System (Adapted from (A. S. Patel et al., 2011)) . . . . .	12
Figure 2.2.9	Change in average cell diffusion for an effective therapy (Adapted from Schneider et al. (2014)) . . . . .	13
Figure 2.2.10	How radiation can result a change with restricted diffusion and low ADC values. (Adapted from (U. Patel et al., 2014)) . . . . .	14
Figure 2.2.11	Objects, entities and processes involved in mobile phone symptom monitoring (Adapted form Kearney et al. (2009)) . . . . .	15
Figure 2.3.1	Past 5-year studies trend proposing use of machine learning in cancer management . . . . .	19
Figure 2.3.2	Feed Forward Neural Network Architecture . . . . .	20
Figure 2.3.3	CNN Architecture . . . . .	22
Figure 2.3.4	CNN Local Receptive Field Connection (Adapted from Nielsen (2015)) . . . . .	22
Figure 2.3.5	ConvNet Max Pooling (Adapted from Karn (2016) ) . . . . .	23
Figure 2.3.6	Pooling Applied to all feature maps (Adapted from Karn (2016) . . . . .	24
Figure 2.3.7	Maximum Margin Classifier . . . . .	26
Figure 2.3.8	Decision Tree . . . . .	27
Figure 2.4.1	General Feature Selection Process . . . . .	28
Figure 2.5.1	Concept Framework . . . . .	30

Figure 3.2.1	Scrum Agile Development Methodology . . . . .	32
Figure 3.3.1	Research Framework; preprocessing, Feature generation, Training& validation . . . . .	33
Figure 3.3.2	(a) A microscopy image with CTC's, (b) Fluorescence image with CTC locations in 3.3.2 (a) . . . . .	34
Figure 3.3.3	Convolutional. Adapted from Nielsen (2015) . . . . .	35
Figure 3.3.4	The ConvNet Architecture Adapted from Mao, Yin, and Schober (2016) . . . . .	36
Figure 3.3.5	A 2 * 2 Max Pooling. Maximum value in 2 * 2 sub-region(Adapted from Deshpande (2016)) . . . . .	37
Figure 4.3.1	A Partial Domain Model . . . . .	41
Figure 4.3.2	Use Case Diagram . . . . .	42
Figure 4.3.3	DFD: Context Diagram . . . . .	44
Figure 4.3.4	DFD:Level 0 . . . . .	45
Figure 4.3.5	Sequence Diagram . . . . .	46
Figure 4.3.6	Design Class Diagram . . . . .	47
Figure 5.3.1	(a) Microscopy Image with CTC's, (b) Fluorescence Image with CTC's location(s) . . . . .	49
Figure 5.3.2	(a) Cropped Positive CTC (b) Negative CTC sample used in training . . . . .	52
Figure 5.7.1	ConvNet Cost Function against Epochs . . . . .	56
Figure 5.9.1	RMSE changes relative to number of Epochs . . . . .	57



## List of Tables

Table 2.2.1	A summary of results mobile phones to monitoring . . . . .	16
Table 2.3.1	Confusion Matrix based on a study by Scholtens et al. (2012) . . . . .	18
Table 5.2.1	Hardware & Software Specification . . . . .	48
Table 5.3.1	A summary of Training and testing dataset dimensions . . . . .	51
Table 5.4.1	Linear SVM Results Summary . . . . .	52
Table 5.4.2	Non Linear SVM Results Summary . . . . .	53
Table 5.4.3	Random Forest Results Summary . . . . .	53
Table 5.4.4	Multinomial Naive Bayes Results Summary . . . . .	53
Table 5.6.1	Multi Layer Perceptron Results . . . . .	54
Table 5.7.1	CNN Results . . . . .	55
Table 5.8.1	CapsNet Results . . . . .	56
Table 6.2.1	Classical Models Results Summary . . . . .	62
Table 6.2.2	Deep Learning Models Results Summary . . . . .	62



## List of Abbreviations

ANN - Artificial Neural Network

ADC- Apparent Diffusion Coefficient (ADC).

BC - Breast Cancer

CapsNet - Capsule Neural Network

CT - Computer Tomography

ConveNet - Convolutional Neural Network

ctDNA - Circulating Deoxyribonucleic acid

CNN - Convolutional Neural Network

CTC's - Circulating Tumor Cells

cfDNA - circulating free DNA

DL - Deep Learning

DFD - Data flow diagram

DCD - Design class diagrams

DNA - Deoxyribonucleic acid

FDA - Food and Drug Administration

GD - Gradient Descent

ML - Machine Learning

Mobile Apps - Mobile Applications

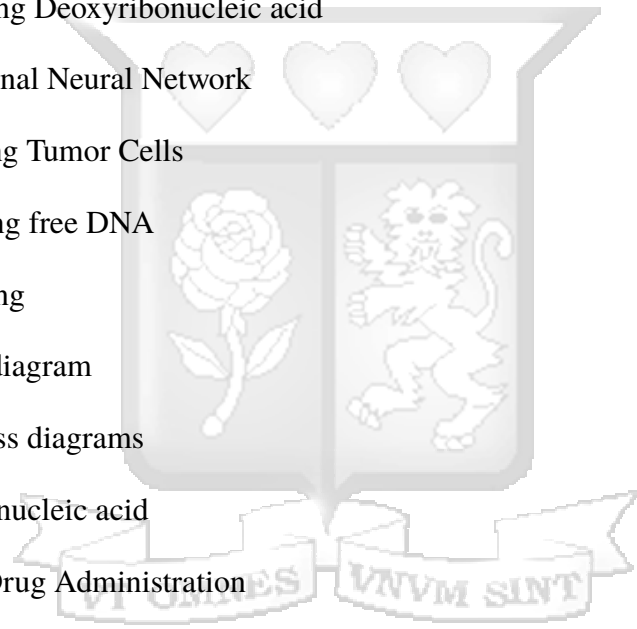
MRI - Magnetic Resonance Imaging

MNIST Dataset - Modified National Institute of Standards and Technology database

NN - Neural Networks

RELU' - Rectified Linear Units

RMSE - Root Mean Squared Error



RGB - Red, Green, Blue

RMSE - Root Mean Squared Error

RNA - Ribonucleic acid

SVM - Support Vector Machines



## Definition of Terms

**ImageNet competition-** Competition in which different industry and academic teams compete to achieve highest accuracy in object detection and classification Deng et al. (2012)

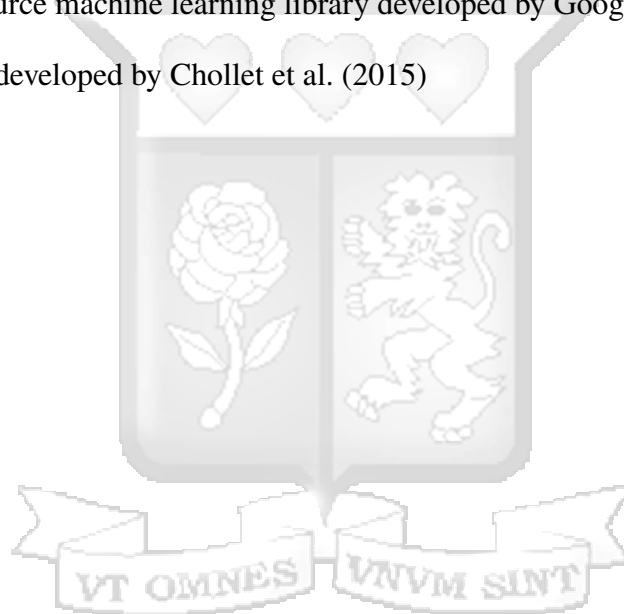
**MNIST-** A Database of handwritten digits

**Cost Function-** Error value to be minimized

**Convex Function-** A mathematical function which if the line segment between any two points on the graph of the function lies above or on the graph, in a Euclidean space of at least two dimensions

**Tensorflow-** Open Source machine learning library developed by Google

**Keras-** Deep Library developed by Chollet et al. (2015)



## Acknowledgment

I would like to thank Selina Olwanda and Anastasia Omodin'g of Strathmore Medical Center, Catherine Nyongesa of Texas Cancer Treatment Center. Their expertise input in this research work was pivotal. I would also like to thank my supervisor, Dr. Joseph Orero, for his immeasurable guidance and contribution to this research endeavor. The Faculty of Information Technology staff never-ending support is also acknowledged. Above all, I thank God the Almighty for granted strength and ability.





## Dedication

I dedicate this work to all my family members; my mother, sister and brothers. Their emotional support during this time was apt.



# Chapter 1: Introduction

## 1.1 Background

Cancer is a disease occurring as a result of genetic mutation or abnormal changes in the genes responsible for regulating the growth of body cells. The cells gain ability to keep dividing without control, producing more cells just like it and forming a tumor (Ferlay, Héry, Autier, & Sankaranarayanan, 2010). Globally, this disease has been attributed the indiscriminate leader in human deaths.

A study by Fitzmaurice et al. (2015), showed that cancer caused more than 8 million deaths in 2013 all over the world. Based on this, it was ranked the second most killer disease just after cardiovascular. Its burden has been felt from first to third world countries. This has been exacerbated by unique geographical, cultural and demographic factors such as aging population, increased predisposition to cancer causing conditions, among others, smoking, being overweight and physical inactivity.

The burden has been more vibrant in the developing countries. Studies show that they have contributed to 57% of cancer cases accounting for 65% related deaths worldwide (Torre et al., 2015). In Kenya, cancer is the third most killer disease after the infectious and cardiovascular diseases (Ministry of Health and Sanitation, 2011). The disease contributed to 7% of total national mortality annually. Based on the Ministry of Health report, this number is dominated by breast and prostate cancer related deaths in women and men respectively.

Most of the cancer types are treatable if detected early, followed by an effective treatment. Traditional radiography, magnetic resonance imaging (MRI), Computer Tomography (CT) and ultrasound are the most common detection modalities. Today, cancer treatment methods vary widely among different patients based on stage and type of tumor. For one medical surgery or radiation would apply, chemotherapy for another, and hormone therapy for others.

Over the years, chemotherapy has been a core component of most cancer treatment. It has been successful in treatment of colon, breast and pancreatic (Kearney et al., 2009). Other modalities such as surgery also have special use cases. Though they have been successful, these treatment methods are not always guaranteed to work for all the patients, all the time. Aside from this unique characteristic, they also possess dire side effects. If these are not closely monitored and attended, they can be gravely life threatening. Often, such effects act as an indicator of the disease progression.

Monitoring a cancer patient undergoing treatment is important just as diagnosis process. The subject requires follow up and regular disease progression evaluation. Dickinson, Hall,

Sinclair, Bond, and Murchie (2014), suggested that this can be achieved through various ways. Among the options they had proposed include regular face-face consultations, laboratory testing, normal medical checkup, detection of disease recurrence and monitoring severity of the symptoms associated with the treatment method. Some of these avenues are being used today to monitor cancer patients. In addition, due to proliferation of mobile computing and ubiquity of Internet, Mobile applications are being embraced by some medical facilities to monitor patients (Ekeland, Bowes, & Flottorp, 2010). Though this is not a standard routine such techniques have significantly complemented the conventional approaches and proved convenient for most patients.

Ekeland et al. (2010), argued that current monitoring methods are unsustainable due to exponential increase in the number of cancer patients, limited human and capital resources required. There is a need for more automated methods not only to complement the existing routines but also aimed at increasing access level to monitoring service. Furthermore, the new approach should require very limited human expertise to match inadequacy of medically trained professional persons in a developing country, Kenya. Moreover, the approach should be convenient and positioned to minimize the core issues associated with the current methods.

## **1.2 Problem statement**

Making regular phone calls, use of mobile applications and adherence to standard medical procedures are the contemporary common cancer monitoring practices. Making phone calls or using Mobile Applications are majorly used to monitor and report the severity of patient symptoms and general disease progression. On the other hand, standard medical procedures are regular routines set by the treatment facility to continuously assess the subject response to treatment through avenues such as physical examination, biopsy and liquid biopsy.

Biopsy is an invasive medical process which can be used to detect the presence of cancer related cells such as Circulating Tumor Cells (CTC's) in a raw body tissue. In contrast, liquid biopsy can be invasive or non-invasive. It involves detection of the cancer cells, tumor markers, contained body fluid such as blood, sweat or saliva. The fluid is stained using special medical reagents and equipment. Biopsies play key role in monitoring cancer treatment response and determining the effectiveness of a treatment option. In addition, these processes are key in assessing the cancer recurrence and progression (Crowley et al., 2013).

The named approaches have been successful in monitoring cancer patients albeit some challenges. Biopsy and liquid biopsy relies on human expert to interpret the final results; mostly from stained image sample. The sample analysis or interpretation using medical device such as a microscope demands lot of human expertise, time and resource input. Following this characterization, the researcher argued that the number of patients who can access the service is directly

proportional to the human effort. This relationship further critically determines the number of patients who can access monitoring service at any given time. Similarly, making daily phone calls is resource intensive as substantial time, money and human capital is also required.

Tumors excretes substances called Circulating Tumors Cells (CTC's) into the patient blood. They can be detected through biopsy process. During treatment, the presence, absence or the number CTC's in body fluid; blood, can be used as a progress metric indicator. This relationship can be harnessed to evaluate how a patient is responding to treatment hence forming a basis for treatment response evaluation. It can also determine the effectivenesses of treatment mode a subject has been enrolled in.

This research work proposed an automated CTC monitoring model. It identifies and enumerates CTC's. The model learns not only to identify the absence or presence of these tumor markers from stained image sample but also quantify the number of the cells present. Deep learning methodology was preferred in model development process as this approach gave the ability to learn the intricate low level, representative features over on a large dimension space. This Artificial Neural Network based approach learned the best representations used to discriminate and enumerate CTC's.

Proposed approach was projected to make it possible for a patient to know if a their condition is worsening or improving given the current treatment method. This was argued would be possible within a shorter periods of time as it automated manual biopsies.

## **1.2.1 Objectives**

### **1.2.1.1 General Objective**

To develop a learning model to identify and enumerate circulating tumor cells from a stained image sample to monitor cancer progression.

### **1.2.1.2 Specific Objectives**

- i). To review current cancer monitoring methods during treatment
- ii). To review how machine learning models have been used in monitoring cancer patients
- iii). To analyze methods of extracting features from circulating tumor images for detection and enumeration
- iv). To develop and test the model

## **1.3 Research Questions**

- i). What are the current cancer monitoring methods in use during treatment?
- ii). Which machine learning models have been used in monitoring cancer patients?

- iii). How do we extract features from circulating tumor image to be used detection and enumeration?
- iv). How do we develop and test the model?

## **1.4 Justification**

After successful diagnosis with cancer, a patient is subjected to treatment method based on their biological uniqueness, stage of illness and type of cancer. During therapy session, the patient must be constantly monitored to assess the effectiveness of current treatment method and avert possible deaths arising from treatment complications. This can be achieved by identifying and enumerating CTC's in blood or other body fluids. Current standard medical processes are very resource intensive; unique devices are required, skilled labour, and time to analyze and present results.

Cancer treatment monitoring is expensive and average patients in Kenya can't afford it. Biopsy for instance costs Kshs.35, 000 on average. This research work proposed a less expensive, learning-based, adaptive computational model which can identify circulating tumor cells in a stained CTC image sample. It automates the result analysis processes hence limited human expertise required. Consequently this was projected to lower the total cost of the service thereby increasing mass access. Patients will be able to access such services locally; or possibly regionally.

## **1.5 Scope and Limitation**

Identification and enumeration of tumor cells was based on blood as the fluid sample. The image staining process was created through established standard medical procedures. All other widely accepted and recommended medical procedures were adhered to. In addition, this work only focused on those patients enrolled in a particular treatment option; but not those unaware of their disease status or awaiting the status confirmation.

## Chapter 2: Literature Review

### 2.1 Introduction

This chapter covers significance of cancer monitoring during treatment and methods currently employed. It further investigates use of machine learning models in cancer treatment and monitoring and concludes by presenting a conceptual framework for this research work.

### 2.2 Cancer Treatment Monitoring

Following a successful cancer diagnosis, follow up care or monitoring is a crucial process. Firstly, it helps to maintain good health of the patient. Secondly, these activities help track general disease progression. Thirdly, managing side effects related to treatment method and finally, it helps to detect the recurrence of disease early hence necessary medical interventions can be sought in time.

Today, the most common cancer treatment methods include surgery, radiotherapy, chemotherapy, immunotherapy and combined modality therapy. These named options have related toxic effects to the subject. Such, among others, include death due to ineffective treatment acute and sub-acute toxic, and chronic effects. World Health Organization (1979), proposed standard format which can be used in reporting these effects. The guideline included the severity of symptoms, organs affected and response to therapy.

Severity of symptoms or extreme side effects and response to therapy are the key components of cancer monitoring as they are indicative of improving or deteriorating patient condition. The treatment option failure can be detected then consequently altered in time. This early intervention can reduce deaths related to treatment or modality complications.

#### 2.2.1 Cancer Monitoring Methods

##### 2.2.1.1 Conventional Cancer Monitoring

Standard cancer monitoring approaches which are widely accepted by the medical research community are in existence. Such include computer tomography (CT) uses X-rays to generate images of the body organs within 15 minutes. Based on *Computed Tomography (CT)* (2010), during the process an X-ray machine spins around the whole patient's body making a slice of images at every instant. This gives more information than a static conventional X-ray machine. Figure 2.2.1 shows a sample of CT with labeled anatomical structures. Its made up of cross-sectional area of the abdomen and capture from a patient lying on the back. A tumor is visible in

the head of pancreas with the arrow indicating superior mesenteric artery.

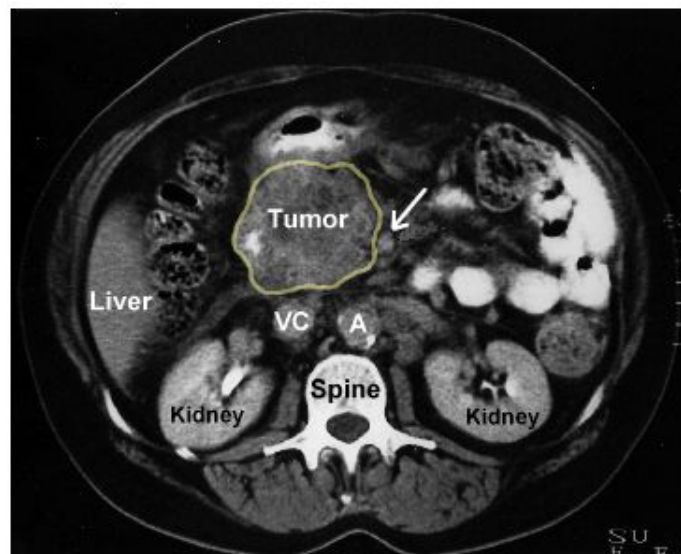


Figure 2.2.1: A cross-sectional CT scan sample. Adapted from (*Diagnosis: CT Scan*, 2007))

A CT scan is used to determine whether the tumor is growing bigger or smaller. This often is possible even before emergence of the symptoms or a physical medical examination (Egan, 2000). During this process, bone scan, radiograph of the bones or CT scan of the abdomen; in breast cancer, are some of the common procedures. In an event that the results of these procedures are suspected, then other approaches such as Positron emission tomography (PET/CT) can be used to assert the output obtained. PET/CT has the ability to detect abnormal lymph nodes and tumor distant recurrence (Graham et al., 2014).

*PET / CT Scan* (2010), reported that PET/CT scan is a type of nuclear medicine imaging which measures the metabolic activity of cells in body organs. Aside from the assertion characteristic, it can also be used among the patients who are perceived to have a specific physical or biological conditions affecting heart and brain, or in those with a unique type of cancer. The success of this method is attributed to ability to generate body organ images in relation to key biochemical reactions thus detecting variations which indicate patient disease state. This forms the basis of progress monitoring. Sample images obtained from this procedure have been represented by schematic 2.2.2. Figure 2.2.2 a; scan showing accumulation within the patient esophagus b; same subject scan after 4 sessions of chemotherapy, c; deposition on left axillary lymph node and d; the scan after 4 sessions of treatment showing tumors vanishing.

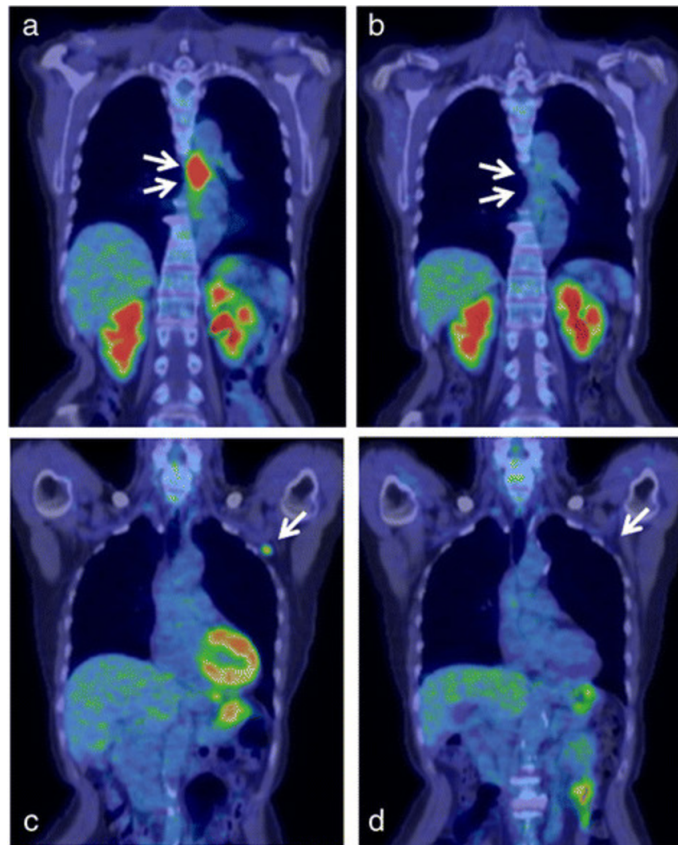


Figure 2.2.2: PET/CT scan samples of Esophageal Cancer Patient . Adapted from (Akiyama et al., 2015))

Mammogram, ultrasound and Magnetic Resonance Imaging (MRI) are also used in place of CT or (PET/CT) (Egan, 2000). MRI uses an intense magnetic fields during the examination process. As a result, the subject will be advised accordingly by the radiologist based on their unique conditional factors such as pregnancy, skin tattoos, pacemaker and history of implants (*Prepare for Magnetic Resonance Imaging (MRI)*, 2010). The process takes forty- five to sixty minutes per body part. During this time, magnets are open on both ends and the patient is expected to lay still and also hold their breath occasionally; up to thirty seconds. Radiologist can communicate with the patient during the session to instruct about any discomfort and other related concerns. Figure 2.2.3, shows a sample result of MRI for a breast cancer patient. In A; MRI scan showing an area of radiation in treated breast cancer patient, B; the same subject 4 years after treatment, cystic spaces forming in radiation areas and C; another scan showing no enhancement; with the arrow, thus exclusion of tumor recurrence.



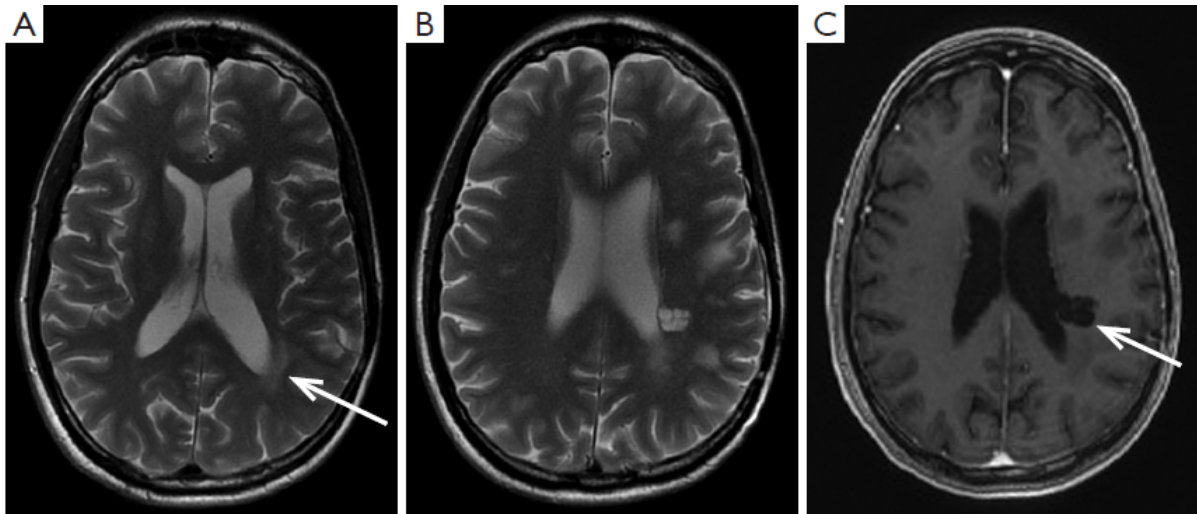


Figure 2.2.3: MRI sample from Breast Cancer Patient. (Adapted from (U. Patel et al., 2014))

Despite the success of these conventional imaging approaches, they have a limitations. First, they can't tell if the detected abnormality is a cancer tumor or just a normal scar related tissue. Secondly, they are pivoted around clinical morphology metric where tumor size, shape or color is measured before, during and after therapy. Lastly, the output of these techniques are often obtained late during treatment.

### 2.2.1.2 Biopsy

Biopsy is a technique use to diagnose and monitor cancer disease. Many argue that it is a sure way of diagnosing cancer. The process involves removal of physical body tissue for examination followed by imaging using special medical devices such as microscope. The cells are extracted from the sample using a needle and examined for tumor. The procedure vary among patients based on the type and location of the tumor. Apart from diagnosis, it has potential for monitoring of the treatment response through analysis of genes related to tumors. Crowley et al. (2013), reported that biopsy can be used in determining the disease progression and patient response to treatment method. Figure 2.2.4 depicts tissue extraction through biopsy process.

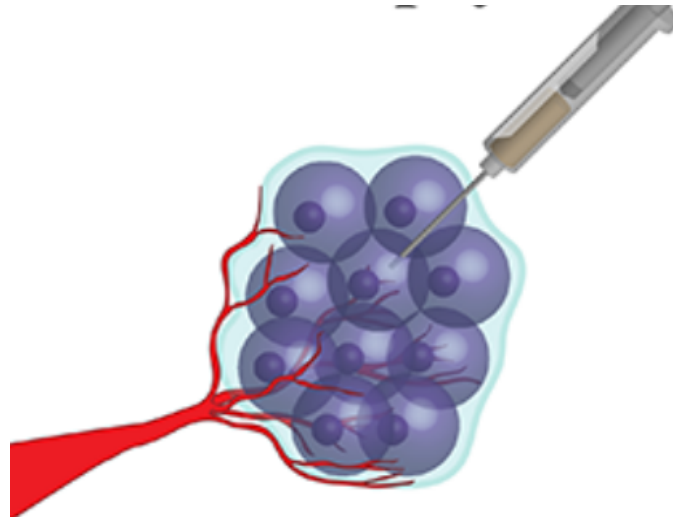


Figure 2.2.4: Biopsy; using a special needle to extract tissue from body. (Adapted from (*Liquid Biopsy Predicts Colon Cancer Recurrence*, 2016))

Biopsy process has some challenges associated with it. Firstly, it is invasive. The process is painful and can cause pain to the patient thus discomfort during the procedure. Secondly, surgical complications exacerbate the chances of cancer spread to other parts of the body. Third, the procedure requires localized sampling of tissue (*Liquid Biopsy Predicts Colon Cancer Recurrence*, 2016). These limitations have significantly constrained its adoption and as a result other non-invasive options such as liquid biopsy have gotten much attention and favored by the medical community.

### 2.2.1.3 Liquid Biopsy

Liquid biopsy is a non-invasive approach used to detect and monitor cancer. It involves use of simple blood tests to detect cancer agents such as freely circulating DNA (ctDNA) and Circulating Tumor Cells in blood. The latter and former have proven to bear a direct correlation with the presence of tumor cells. In addition to blood, saliva, urine, plasma, cerebrospinal and seminal plasma are other biological fluid variations which can be used (Di Meo, Bartlett, Cheng, Pasic, & Yousef, 2017). With contrast to biopsy and conventional imaging methods which are dependent on cell tissue and images respectively, liquid biopsy is a tumor marker monitoring method. A blood samples can be used to determine disease and tumor progression (*NCI Dictionary of Cancer Terms*, 2018). This method overcomes some of the limitations of tissue-based biopsy by allowing for the screening biomarkers which are secreted by tumor cells into the blood.

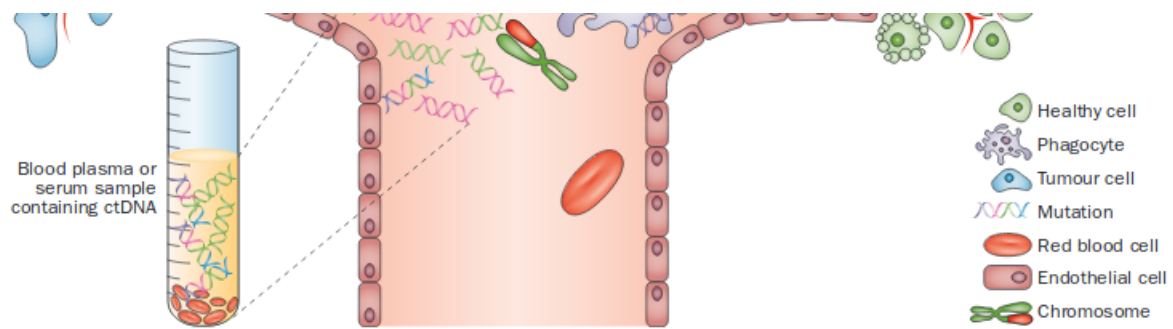


Figure 2.2.5: Liquid Biopsy; Detecting Calculating DNA(ctDNA) through Blood Test (Adapted from (Crowley et al., 2013))

Egan (2000), defines tumor marker, or biomarkers, as a substance which is produced by either cancerous or non cancerous cells. Varying types of cancer such as ovarian, prostate and breast cancer releases biomarkers into the blood. A blood test can be used to detect these markers' antibodies through blood sampling as depicted by figure 2.2.5 above. Di Meo et al. (2017), reported a higher number of CTC's in renal cell carcinoma patients and found to correlate with the metastasis, disease spread. Other similar studies have also recorded a similar concluding that CTC enumeration is an early indicator of cancer recurrence and overall mortality.

The general liquid biopsy process is composed of collection of blood sample, preparation of collected sample, detection using imaging device and finally; data analysis and interpretation. These processes are diagrammatically illustrated by figure 2.2.6 below.

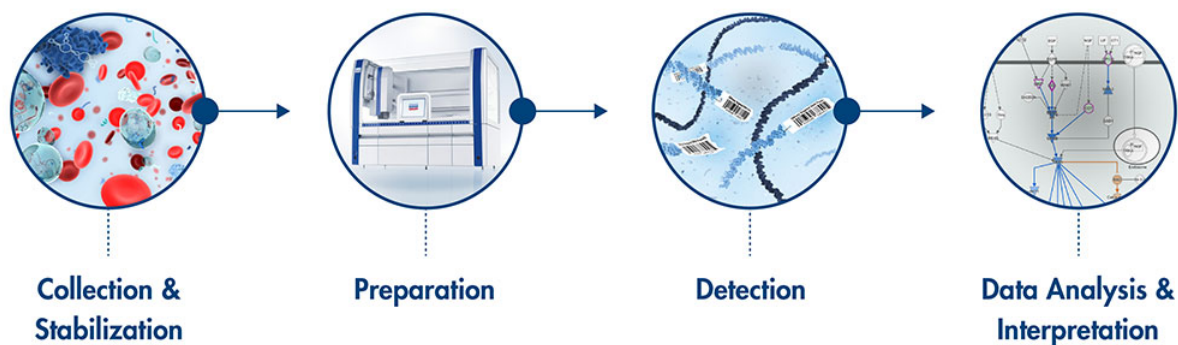


Figure 2.2.6: Liquid Biopsy Processes.(Adapted from (*Liquid Biopsy Predicts Colon Cancer Recurrence*, 2016))

DNA based Liquid biopsy requires 3 ml of blood fluid in case of for plasma preparation. This should be carried out within 5 - 6 hours of withdrawal. In this context, the collection is done using

tube cleansed with anticoagulant such as ethylenediaminetetraacetic acid. Cells are thereafter extracted through centrifugation followed by plasma removal after successful blood clot. The DNA related to tumors are then extracted using the recommended medical kits such as Maxwell RSC (MR), ccfDNA Plasma, Nucleic Acid Isolation, MagNA Pure Compact (MPC), QIAamp Circulating Nucleid Acid (QCNA). Currently, there is no agreed extraction method but each have got associated sensitivity and accuracy issues (Pérez-Barríos et al., 2016).

On the other hand, CTC Liquid biopsy detects and enumerate cells of epithelial origin encompassing CD45-, EpCAM+, and cytokeratins 8, 18+, or 19 using 7.5 ml of blood. There are three major processes involved; collection of blood sample, preparation of the collected content and sample analysis using immunomagnetic and florescence imaging. During the preparation phase, blood is placed on a special tube and then centrifuged to extract plasma. The resulting sample is then placed in the autoprep subsystem. This section contains the ferrofluid to isolate the epithelial cells. At this stage CTC are delinked form other solid blood components. This is then followed circulating cell tumor differentiation through staining using cytokeratin antibodies associated with epithelial cells.

The antibody stain is then used to identify CD45 marker. Consequently, another DNA stain is added in the sample to enable conspicuous visualization of nuclei of circulating tumor and leukocyte. This marks the end of sample preparation phase. This is then followed by last process; the analysis. During this phase, the cells are placed in a magnetic cartridge which generates magnetic fields to centrally locate the cells as shown in figure 2.2.7.

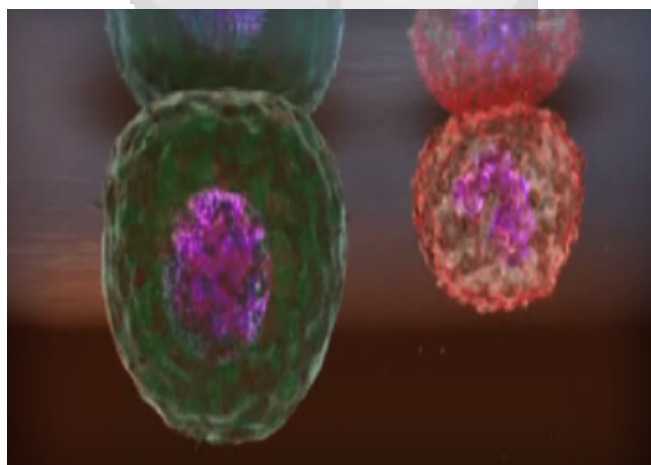


Figure 2.2.7: CTC prepping. Magnetic Force Pull Localization (Adapted from (*CELLSEARCH System Overview CellSearch*, 2018))

The stained CTC's are then placed on CellSearch CellTracks Analyzer II module for analysis. The system displays suspected CTC cells which are positive for the tested antibodies. The cell

counts forms the basis of disease progression estimation and in addition informing if the current therapy or the treatment option is effective or not. CTC liquid biopsy process is represented using figure 2.2.8.

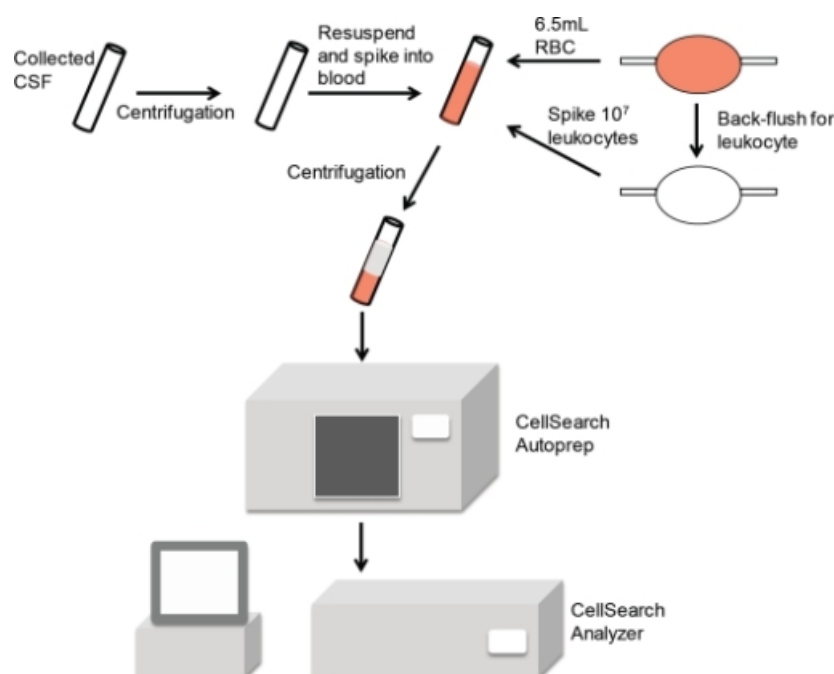


Figure 2.2.8: CTC Liquid Biopsy Using CellSearch System (Adapted from (A. S. Patel et al., 2011))

Detection of tumor related DNA (ctDNA) in blood can be used as clinical metric of measuring the patient performance to treatment aside from helping in early diagnosis. Crowley et al. (2013), demonstrated that this non-invasive approach have a greater potential. Circulating biomarkers such as proteins, RNA and DNA can be used in detection, monitoring and specialize treatment of cancer by understanding the gene mutation. Clinical utility and cost effectiveness are also possible substantial gain from the approach. According to Diehl et al. (2008), monitoring tumor-specific DNA in plasma among colorectal cancer patients made it possible to detect the disease recurrence with over 99% accuracy. The same pattern have also been reported in other similar parallel research work contextualizing breast and lung cancer.

The down side of this technique though is; approximately 1 CTC in a hundred million cells. As a result of this, the identification and enumerations requires a device having very high specificity and sensitivity (Nelson, 2010). Following this rarity constraint, the Food and Drug Administration (FDA) has clinically approved only CellSearch as the mode of CTC identification and enumeration of CTC's within the medical sector. It has been validated to monitor patient with colorectal, breast,, and prostate cancer patients. Due to specificity and sensitivity characteristics, CTC have

not been fully embraced by the medical community in playing key role in cancer management (Diehl et al., 2008).

#### 2.2.1.4 Diffusion-Weighted MRI (DW-MRI)

Current cancer monitoring processes based on morphology and clinical laboratory tests are often obtained late during treatment, Focusing on the extent of tumor size or shape. Introduced by Thoeny and Ross (2010), DW-MRI is a functional imaging tumor marker which can provide an early quantitative treatment response metric for a cancer patient. It is sensitive to biological changes occurring at the cell level during therapy hence allowing visualization and quantification of water molecules mobility in the observed tissue (Schneider et al., 2014).

This approach not only make it possible to predict response to the anti-cancer treatment in adjuvant, non-adjuvant or metastatic setting but also to detects metastases and makes it possible to personalize treatment procedures for different patients. Treatment modality such as chemotherapy and radiology can lead to loss of cell membrane structure. This effect can be quantified by detecting an increase in average diffusion value of the tumor cell. This is diagrammatically represented by figure 2.2.9.

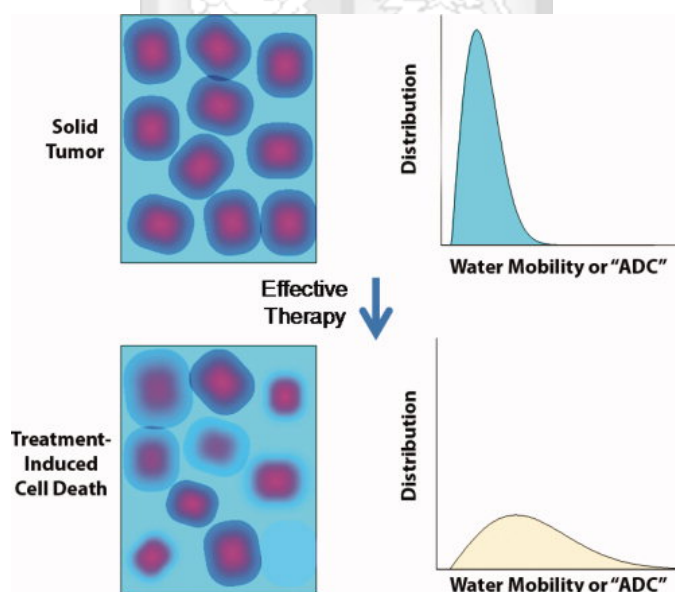


Figure 2.2.9: Change in average cell diffusion for an effective therapy (Adapted from Schneider et al. (2014))

Effectiveness of treatment option for most tumor types can be detected early using apparent diffusion coefficient (ADC). Schneider et al. (2014), showed that malignant cancer cells are known to have lower ADC's in comparison to healthier ones. Higher initial values during pre-treatment

was an indicator for poor therapy result. On the other hand, increase in ADC's over time during treatment was good response indicator (Cui, Zhang, Sun, Tang, & Shen, 2008). The same trend have been recorded in breast (Theilmann et al., 2004), prostate (Røe, Kakar, Seierstad, Ree, & Olsen, 2011) and colorectal (Wybranski et al., 2011) cancer subjects.

According to U. Patel et al. (2014), when the diffusion of water particles is lowered, this is often referred to as restriction, otherwise, facilitation when they increase. In addition, this study also echoed the fact that state of diffusion can be used characterize the disease progression. Figure 2.2.10 depicts the diffusion concept at a low level.

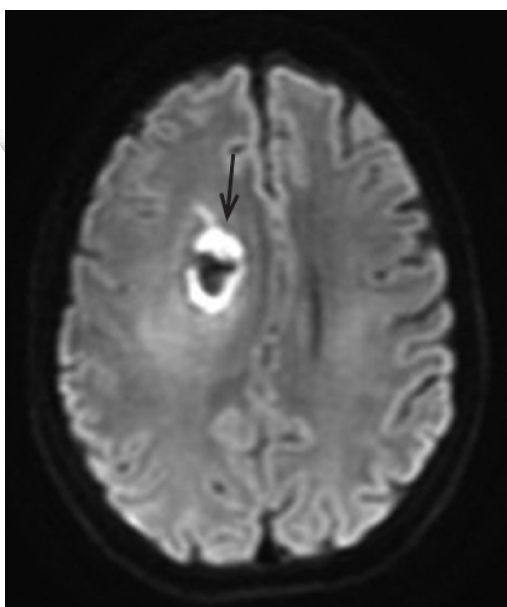


Figure 2.2.10: How radiation can result a change with restricted diffusion and low ADC values. (Adapted from (U. Patel et al., 2014))

### 2.2.1.5 Mobile Application Symptom Monitoring

Aside from aforementioned mentioned monitoring techniques, other studies have argued that mobile applications can also be effective monitoring tool. They can help track symptom severity related to treatment method regularly. The toxic symptoms associated with a treatment method such as nausea, vomiting, fatigue, bone pains, mucositis, hand-foot syndrome, diarrhoea, loss of weight and appetite can be reported and conveniently managed. Otherwise, if indicators are not managed they can be unbearable and gravely life threatening.

Kearney et al. (2009), evaluated the impact of using mobile applications in remote monitoring symptom severity among the patients suffering from lung, breast and colorectal cancer. The study subjects used mobile application to report side effects. From their analysis, it is arguable that use of mobile phones can be used to effectively enhance the management of symptoms among

cancer patients undergoing chemotherapy and other forms of treatment. The technique provided an ideal and accurate avenue of symptom tracking, recording and communication. User input was used to measure the extent of severity upon receipt. Further, the application automatically created personalized medical report for each subject. Information computed and presented to the patient included precise instructions to manage the symptoms, diet and health advice among others.

Figure 2.2.11 shows the application logic and the related entities in the system.

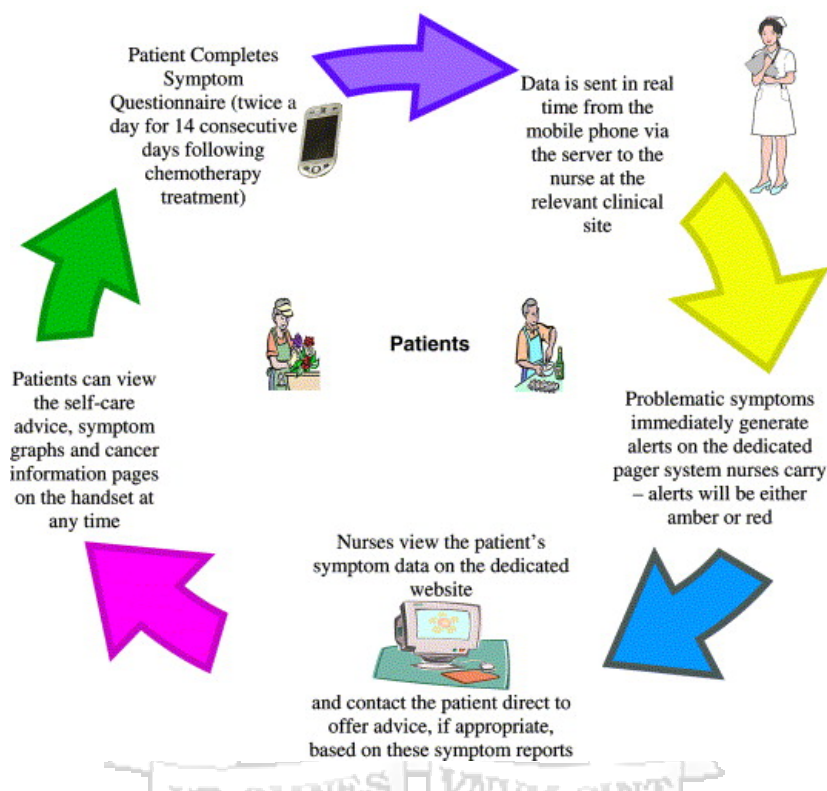


Figure 2.2.11: Objects, entities and processes involved in mobile phone symptom monitoring (Adapted from Kearney et al. (2009))

In another related study done by Maguire et al. (2008), a risk model was developed based on the symptoms assessment input from the user. It alerted medical expert about the severity and life threatening symptoms. This proved very convenient. Further, the application allowed patients to fill electronic copy of symptom questionnaire, take their temperature thermometer and remotely sent the value. The study concluded that patients felt confident about being constantly cared as a result of prompt response. A summary of this study is shown in table 2.2.1. The symbols ++ means strong effect, +; some effect and n/a implying not available



<b>Results</b>	<b>Strength and direction of evidence</b>
Patient provide communication	++
Monitor Treatment response	+
Detect Unrecognized problems	++
Changes to patient health behavior	n/a
Changes to patient management	+
Improved patient satisfaction	n/a
Improved health outcomes	++
Strong and effective quality improvement	n/a

Table 2.2.1: A summary of results mobile phones to monitoring

### 2.2.1.6 Cancer Spread Prediction Using Deep Learning

Deep learning is a machine learning methodology. It is composed of Neural Network model layers stacked on top of each other. The method learns input features automatically. According to LeCun, Bengio, and Hinton (2015), deep learning models have dramatically improved the state-of-the-art visual object recognition and detection. They reported that these models performance is pivoted around local receptive fields, pooling, and weight sharing principles associated with backpropagation as the learning rule.

Cancer response detection through conventional imaging, biopsy, liquid biopsy or mobile phone symptom monitoring involves physical examination of image sample. These processes are currently laborious and time consuming. Wang, Khosla, Gargeya, Irshad, and Beck (2016), proposed a model to automate the image analysis process. Their deep learning model can automatically extract features from the image and predict whether the cancer has spread to other parts of the body or not. The model recorded Area Under Receiver Operating Curve (ROC) of **0.925**. This level of performance was complemented with the human expertise, **0.966** accuracy, further pushing the model recognition levels up to **0.995**. Deep learning algorithms can significantly fasten the cancer monitoring process through automation of manual image analysis.

### 2.2.1.7 Limitation of the Current Monitoring Approaches

Conventional imaging or morphological based monitoring techniques often takes place late during treatment where the reduction in size of the tumor need to be measured. Additionally, they are costly and cannot be done on a regular basis to monitor the response. Diffusion-weighted approach,

though can detect the effectiveness of treatment approach in advance, can be very resource intensive in terms of expertise, cost and devices required. Mobile phone symptom monitoring on the other hand are very apt in collecting and relaying first hand information from the patient correctly and giving alert information based on the risk level. Limitation of this option is attributed to inability to detect the changes in patient health behavior, level of patient satisfaction and quality improvement. Moreover, conventional imaging procedures; aside from being expensive, demands that the patient be physically present for the relevant medical procedure. This can be a huge hindrance to the patients who are living in rural areas who have to travel long distances to the regional medical centers.

## **2.3 Application of Machine Learning in Cancer Monitoring**

### **2.3.1 Machine Learning**

Machine learning is a sub-field of Artificial Intelligence concerned with algorithms which allow computers to learn. An algorithm is given set of data and infers information about the properties of data by approximating the latent function. Learned function can make predictions on the new instances. A learning process can be supervised or unsupervised. The former technique involves learning from a training data having corresponding labels, the latter on the other hand, involves learning from data with no predefined labels. In unsupervised learning, the task is to find a hidden pattern such as set of clusters in training dataset.

### **2.3.2 Machine Learning Application in Cancer Monitoring**

Machine Learning models have been used in cancer monitoring in varying dimensions. The models have been used in detection of circulating tumors. Cytokeratins, Cytokeratin, Apoptotic and debris are common circulating tumors. They can be extracted through various methods such as fictionalized and structured medical wire (FSMW), Epithelial Cell Adhesion Molecule (EpCAM), density gradient centrifugation and membrane filtration (Chausovsky et al., 1999).

Automated analysis and enumeration of CTS's in blood have been achieved by use of different machine learning algorithms. Svensson, Krusekopf, Lücke, and Thilo Figge (2014), used Naive Bayes classifier and Generative Mixture Model to detect tumors in blood. In the study, cells were collected using fictionalized medical wire and thereafter stained. The intensity of RGB (Red, Green, Blue) values of the resulting sample was then used as the input features.

In another study done by Scholtens et al. (2012), Random Forest and Decision Trees were used to discriminate type of tumor cell in blood. The target classes were labeled as CTC, Apoptotic CTC, CTC debris, Leukocytes and Debris. Table 2.3.1 is a confusion matrix summary.

	<b>CTC</b>	<b>CTC Apoptic</b>	<b>CTC Depris</b>	<b>Leukocytes</b>	<b>Debris</b>
<b>CTC</b>	76 (83)	14 (15)	1(1)	1(1)	0
<b>CTC Apoptic</b>	14 (18)	53 (66)	9(11)	0	4(5)
<b>CTC Debris</b>	20 (1)	87 (6)	1356(91)	7(10)	26(2)
<b>Leukocytes</b>	3 (10)	2 (10)	0	1270(94)	60(5)
<b>Debris</b>	43 (1)	84 (2)	192(4)	190(4)	4177(89)

Table 2.3.1: Confusion Matrix based on a study by Scholtens et al. (2012)

Mao et al. (2016), also proposed another machine learning approach to detecting CTC's in blood. In their research, a variant of Artificial Neural Network known as Convolutional Neural Network (ConvNet) was used. The model automatically learned the features used to classify sample as either having CTC or not. They proved that automated feature discovery gave much better results than the hand crafted ones. In a parallel research work by Wang et al. (2016), a similar learning approach was used in detection of tumor recurrence yielding state of the art results.

Kourou, Exarchos, Exarchos, Karamouzis, and Fotiadis (2015), reviewed machine learning algorithms which have been used in monitoring cancer progression. They argued that the a number of research work have employed the models in prediction of cancer susceptibility, projection of disease recurrence and estimating probability of survival. On the other hand, other studies focused on cancer susceptibility. Learning models were experimented and proposed for such use cases in predicting the risk associated with developing tumor. Further, machine learning recurrence models predicting the spread of cancer to other parts of the body based on clinical, imaging and genomic data were have also been proposed by the research community.

Kourou et al. (2015), summarized the distribution of past 5-year medical research which are based on machine learning as a core tool solution. The study enlisted Decision Trees, Support Vector Machines (SVM), Naive Bayes and Artificial Neural Network (ANN) as best performing learning algorithms. Figure 2.3.1, depicts this trend.



Figure 2.3.1: Past 5-year studies trend proposing use of machine learning in cancer management

## 2.3.3 Machine Learning Algorithms

### 2.3.3.1 Feed Forward Neural Networks

Feed Forward Neural Networks architecture is composed of input, neurons, activation functions, multiple layers and weights. Input data is fed into the network through the input layer. Each layer learns some specific abstract feature about the input data and then transfers that information to the next layer. Figure 2.3.2 captures the Network architecture.

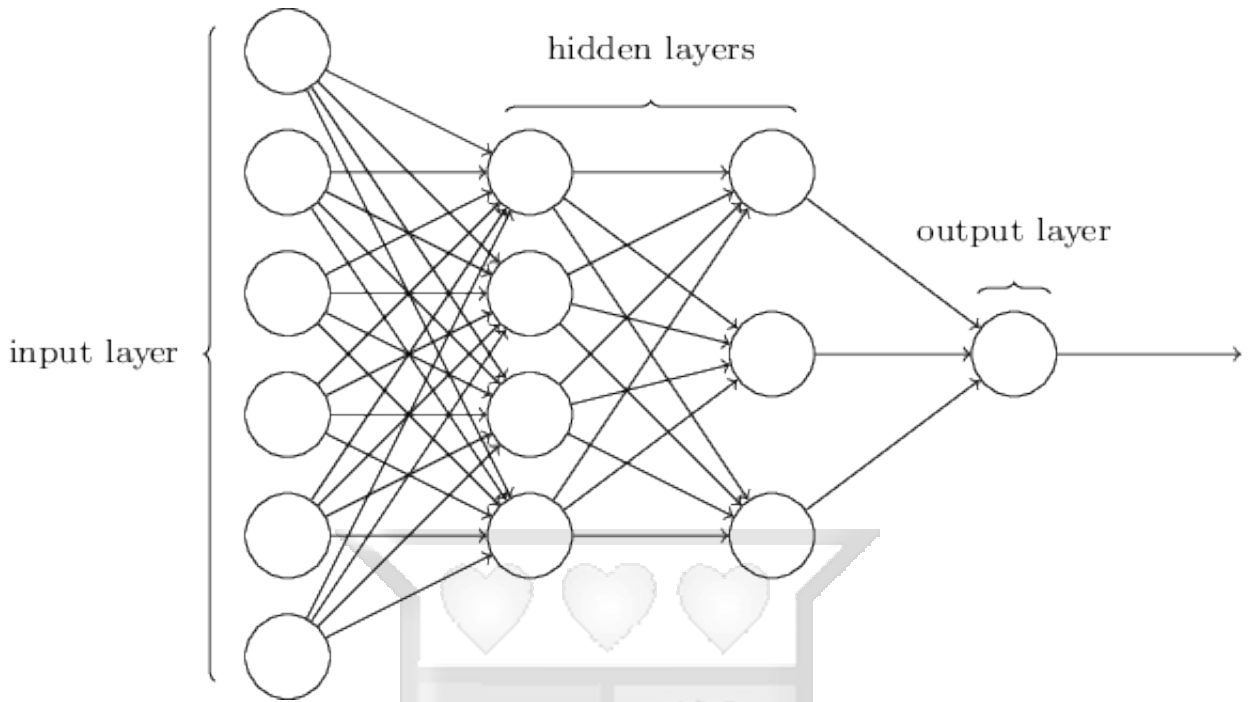


Figure 2.3.2: Feed Forward Neural Network Architecture

During forward propagation output value from input layer is weighted as

$$z = \vec{W} \cdot \vec{X} + \vec{b} \quad (1)$$

where  $z$  is the activity, and  $\vec{W}$  the weight values matrix.  $z$  is activated from one layer to the other using an activation function such as sigmoid function,  $\sigma$ , given as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2)$$

Predicted output value is expressed as

$$\hat{y} = \sum_1^n \sigma(z) \quad (3)$$

where  $z$  is the output of the last layer

The learning process is based on forward propagation to the final layer, leading to a predicted value  $\hat{y}$ , followed by a backward propagation of the error to all the previous layers. As the error

is being propagated backwards, the weights and biases in the network are being adjusted using gradient descent in order to get the optimal values. The Error,  $E$  is calculated based on the equation 4

$$E = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \quad (4)$$

where  $\hat{y}$  is the predicted value and  $y$  is the actual values summed over all the training data samples  $m$ . Weight and bias optimization task can be formulated follows

$$\min_{w,b} \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \quad (5)$$

Update rule is based on the following 7 below

$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} E \quad (6)$$

$$b_i := w_i - \alpha \frac{\partial}{\partial b_i} E \quad (7)$$

Weight and bias update process is repeated for a number of epochs for each training set instance. At the end of iteration, the biases and weights can be used to weight the new input and predict most probable target label in classification or continuous value in regression (LeCun et al., 2015).

### 2.3.3.2 Convolutional Neural Network (ConvNet)

ConvNet or CNN differs from feed forward network in that they do not have layer to layer connection and the input features are automatically learned. The working principles of CNN is based on the operations such as convolution, activation, pooling and classification (fully connected layers).

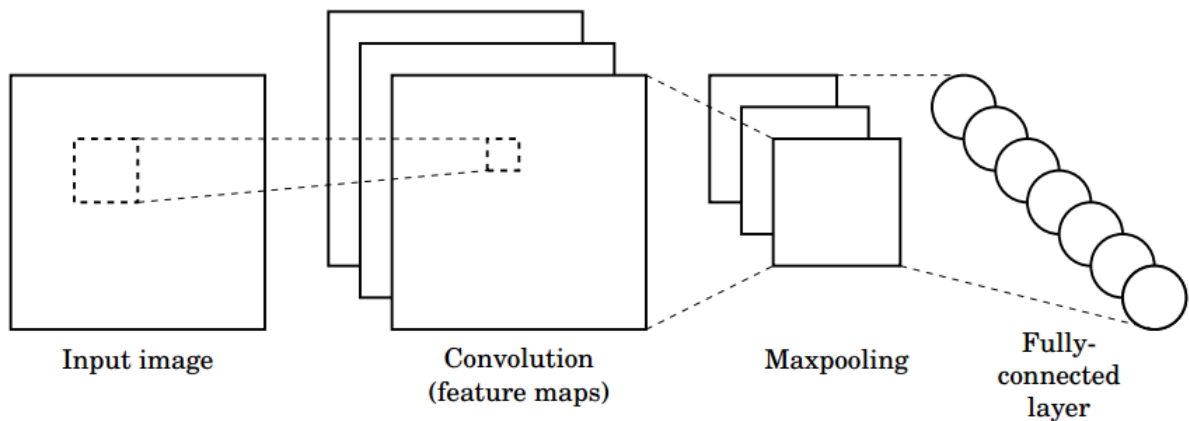


Figure 2.3.3: CNN Architecture

The convolution step extracts features from the input such as image. This process is achieved by defining a weight matrix or the kernel of size  $m * m$  which is slid across the image input from left to right, computing dot matrix in each position. This results in a 2-dimensional in  $K$  set of feature maps. Feature depth, stride and padding values are hyper parameter. Due to exponential number of features in images, each neuron is usually connected to a small area of the input know as the local receptive filed. The local receptive fields process makes connections in a small, localized regions on input data. Each neuron in the first hidden layer will be connected to a small region of input neurons. Figure 2.3.4 is a high level depiction of this concept. It shows that the hidden neuron will be connected to  $m * m$  area in the previous layer. Each of the neurons will have same weight and bias. Sharing weights and biases greatly reduces the number of parameters involved in learning.

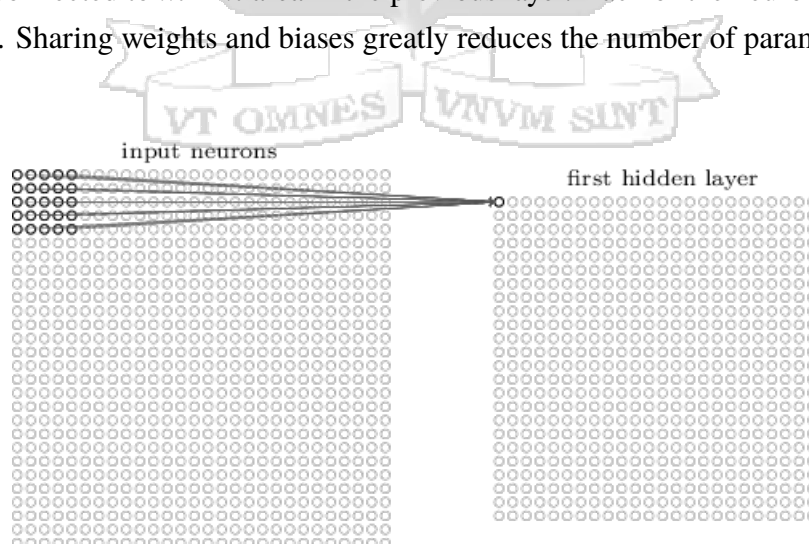


Figure 2.3.4: CNN Local Receptive Field Connection (Adapted from Nielsen (2015))

The pooling layer is usually inserted between successive Convolutional layer. Its role is to

reduce amount of learning features and computation cost. Pooling is applied to each and every feature map resizing them spatially. The feature map is downsampled and the most important features retained. Max, sum, average and L2 norm are the most common pooling methods Nielsen (2015). The process is depicted in figure 2.3.5

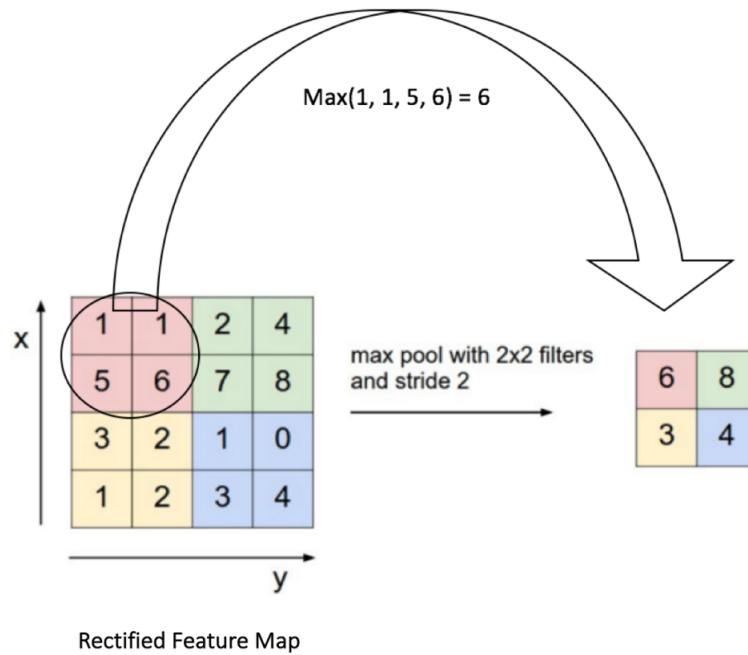


Figure 2.3.5: ConvNet Max Pooling (Adapted from Karn (2016) )

Rectified Linear Unit (RELU) is applied to each feature map to remove non negative values and replacing them with 0. RELU introduces the non-linearity enabling capturing of intricate patterns in the data. It is defined by  $f(x) = \max(0, x)$ . Pooling process applied to all the rectified feature maps results in downsampled feature maps as shown in figure 2.3.6.



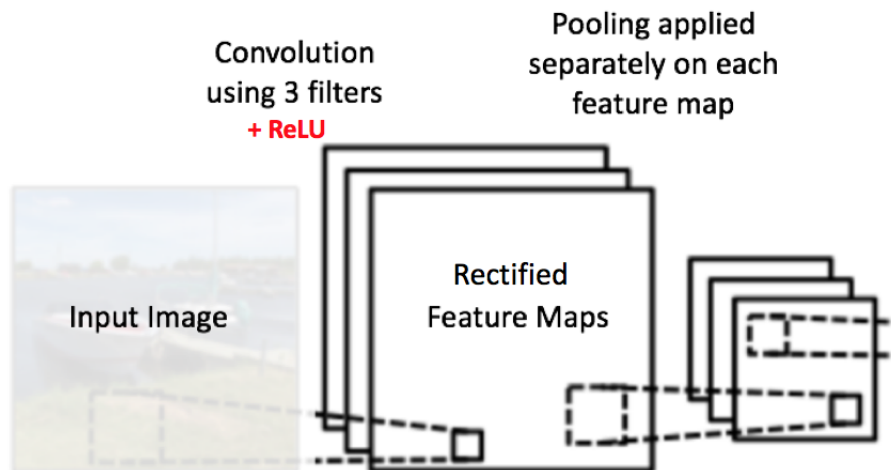


Figure 2.3.6: Pooling Applied to all feature maps (Adapted from Karn (2016))

The fully connected layer (FC) is the same as the feed forward network layers with all the neurons output from the last Convolutional layer. This layer is activated by Softmax function defined by equation 8 for classification tasks. FC layer allows for classification of the current training instance into one of the  $C$  classes. Weights or the kernel is adjusted using backpropagation algorithm.

$$\frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \quad \text{for } j = 1, \dots, K \quad (8)$$

According to LeCun et al. (2015), deep learning methods such as CNN have dramatically improved the state-of-the-art in visual object recognition and object detection. They argue that deep learning can discover intricate structure in large data sets using local receptive fields, pooling, and weight sharing and learning through the back propagation. CNN success in image detection and recognition have been attributed to automatic feature discovery and translation invariance.

In the ImageNet competition, deep Convolutional Neural Network was trained to classify the 1.2 million high-resolution images into the 1000 different classes. On the test data, they achieved top-1 and top-5 error rates of 37.5% and 17.0% which is a much better record than the previous learning approaches (Krizhevsky, Sutskever, & Hinton, 2012). Same performance trend have been noted in other parallel similar studies such as Havaei et al. (2017) where such model was used automate brain tumor segmentation.

### 2.3.3.3 K-nearest Neighbors (kNN)

k-Nearest neighbor classification assigns the majority class of the  $k$  nearest neighbors to a test document. kNN is a case based learning method which can be used in creating classification or regression models. The  $k$  nearest neighbors are determined based on some distance metric measure. Two mostly used distance metrics include Manhattan and Euclidean distance measure which are formulated by the following equations.

Given feature  $X = \{x_1, x_2, x_3, \dots, x_n\}$  and the labels  $y = \{y_1, y_2, y_3, \dots, y_n\}$ , the distance function  $d(X, y)$  can be expressed in one of the following ways.

i) Manhattan Distance

$$d_A(x, y) = \sum_{i=1}^N |x_i - y_i| \quad (9)$$

ii) Euclidean Distance

$$d_A(x, y) = \sum_{i=1}^N \sqrt{|x_i^2 - y_i^2|} \quad (10)$$

### 2.3.3.4 Naive Bayes

Bayesian classifier is a probabilistic model based on Bayes theorem. It models the underlying word features in different classes and then make prediction based on the posterior probability of the document's class. Given data sample  $X$ , hypothesis  $H$ , learning process involves determining  $P(H|X)$ . This is the probability that hypothesis  $H$  holds given the observed data sample  $X$ .  $P(H)$  is the prior probability,  $P(X)$ , is the probability that sample data is observed.  $P(X|H)$  is the posterior probability of observing the sample  $X$ , given that the hypothesis holds. Using the Bayesian theorem, the likelihood of a new data sample  $X_i$  can be formulated as shown in equation

11

$$P(H|X) = \frac{P(X|H) * P(H)}{P(X)} \quad (11)$$

Assuming hypothesis hold for the fact that  $X$  can belong to any of the predefined classes

$C_1, C_2, \dots, C_n, P(X|H)$  is given by :

$$P(X|C_i) = \prod_{k=1}^n P(X_k|C_i) \quad (12)$$

For continuous value prediction,  $P(X_k|C_i)$  is usually computed based on Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x - \mu)^2}{2\sigma^2}} \quad (13)$$

### 2.3.3.5 Support Vector Machine

A support Vector Machine (SVM) is a large-margin classifier. It is a vector space based machine learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data, possibly discounting some points as outliers or noise. This is depicted by figure 2.3.7.

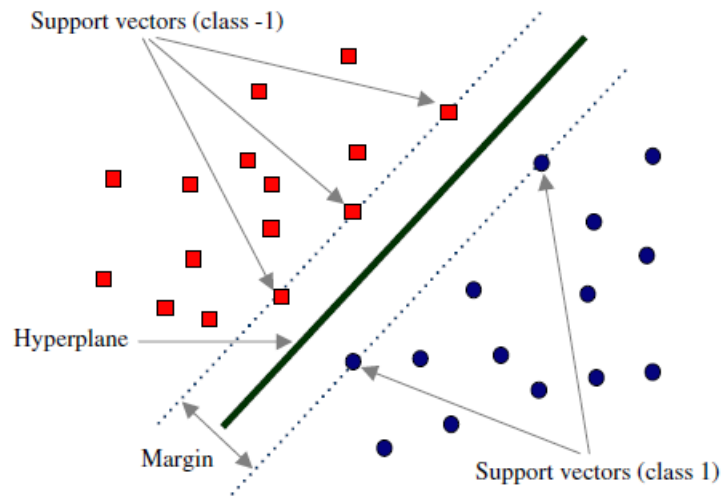


Figure 2.3.7: Maximum Margin Classifier

Support vector machines are not necessarily better than other machine learning methods, but they perform at the state-of-the-art level and have much current theoretical and empirical appeal. SVM's have been used in document and image classification, segmentation of proteins in medical science and hand written characters recognition. SVM uses hinge loss to achieve maximum-margin

separation property. The loss is defined by equation 14

$$C(x, y, f(x)) = (1 - y * f(x))_+ \quad (14)$$

where  $C$  is the loss function,  $x$  the sample,  $y$  is the true label,  $f(x)$  the predicted label. The objective function can be simplified as

$$\min_w ||w||^2 + \sum_{i=1}^n (1 - y_i \langle x_i, w \rangle)_+ \quad (15)$$

### 2.3.3.6 Decision Tree

Decision tree is a supervised learning algorithm used in classification and regression. Training data is split into distinct sets based on most informative feature attribute. Each leaf of the tree corresponds to predefined target in classification task. Root node and internal nodes represents a condition of the attribute upon which the tree splits into. Figure 2.3.8 depicts the tree structure.

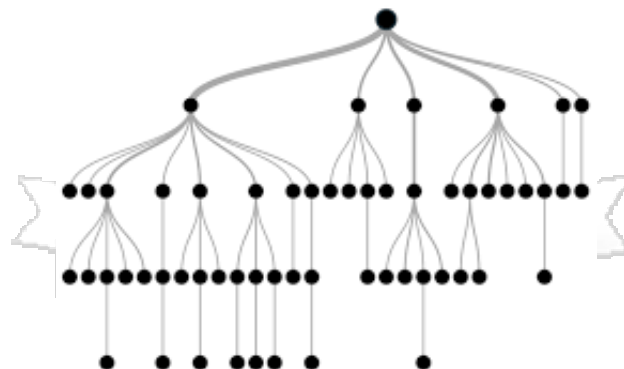


Figure 2.3.8: Decision Tree

The internal node splitting is a recursive process. Knowing when to stop splitting is based on the minimum number of samples in a leaf or maximum depth of the tree. Deciding on the attribute to split on can be determined by measure of impurity by using gini index, classification error or

entropy.

$$Gini \text{ Index} = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2 \quad (16)$$

$$Entropy = - \sum_{i=1}^n p_i \log_2(p_i) \quad (17)$$

$$Classification \text{ Error} = 1 - \max(p_i) \quad (18)$$

## 2.4 Feature Selection from CTC Images

Feature engineering or selection is process of defining set of attributes which best describes a target output. It generally encompasses selection of a subset of features which act as input to a machine learning algorithm. A general feature selection process involves selection of the candidate set which is then evaluated based on some metric such as accuracy and then repeated until the best performing set is obtained. This process is schematically represented by figure 2.4.1.

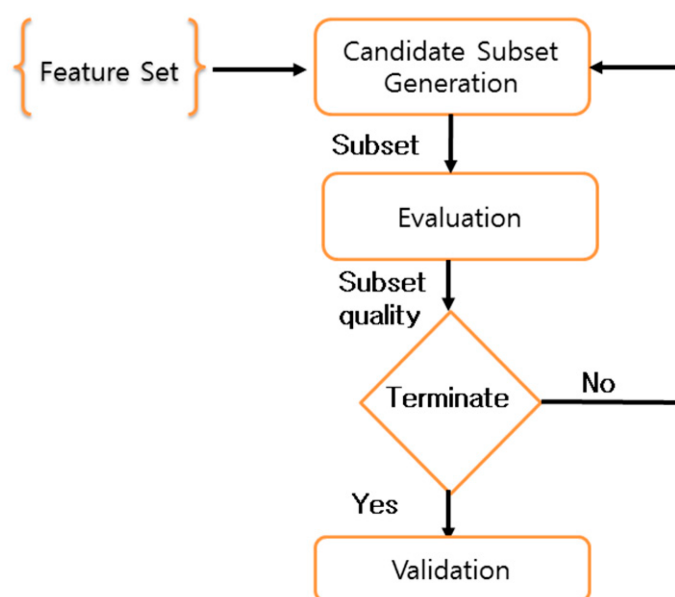


Figure 2.4.1: General Feature Selection Process

Feature selection process improves performance of the model and enables learning algorithms to train much faster by reducing the dimensionality of data space. Often, the practice also contributes to control of overfitting through bias-variance trade off.

### 2.4.1 Manual CTC Feature Extraction

Svensson et al. (2014) and Ciuarte, Marita, and Buiga (2015) presented histogram based method for identification of most informative features. This was based on image analysis regarding RGB or Hue, Saturation, Value (HSV) values. A histogram was then created to numerically visualize the latter and former distributions. This enumeration helped to create of features which are input to a learning algorithm. In another similar parallel study by Scholtens et al. (2012), they proposed different approach. Their technique involved extracting morphological patterns; perimeter, circularity, texture characteristics; contrast mean, entropy mean, quantitative metrics; total intensity, standard deviation and correlation; gradient,  $R^2$ . This unique characterizations recorded and tabulated forming the feature set.

### 2.4.2 Automated Feature Extraction

Manual identification of best representative features from an image sample is quite time consuming and quite error prone. Best feature set in a sample can be learned automatically during the training process. Mao et al. (2016), used deep ConvNet to achieve this. In their work they showed that best features can be learned using the neural network architecture. The network, Convolutional Neural Network (ConvNet), is a variation of Artificial Neural Network which has the ability to learn best set of features during training from input instance which can be used in classification, regression or clustering.

## 2.5 Conceptual Framework

Following our literature the researcher proposed a conceptual framework as depicted by figure 2.5.1. Blood sample drawn from a patient is prepared by following standard medical procedure in place such as liquid biopsy preparation. The sample is then stained from which features are automatically extracted. A model is then trained based on these representations to discriminate if the sample has circulating tumor cells or not and also quantify the number based on the same set. This process is iterated with other data samples from different subjects and finally used with a new test instance to make a prediction.

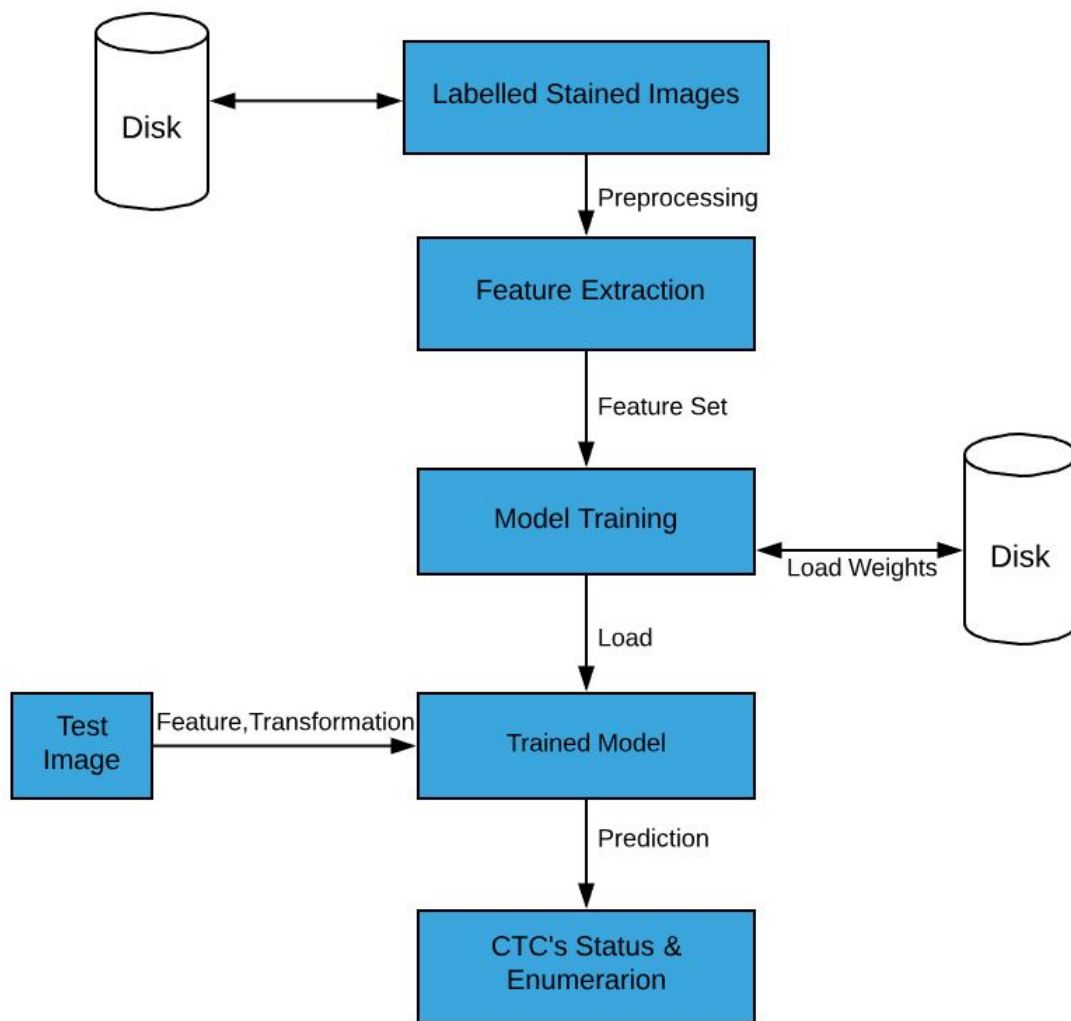


Figure 2.5.1: Concept Framework

## Chapter 3: Methodology

### 3.1 Introduction

This section describes research design for achieving the stated objectives in chapter 1. It also highlights the research framework, data acquisition and pre-processing, data splitting, feature extraction, model architecture and training process. It concludes by highlighting the research quality and validation with a summary of possible ethical concerns of this research work.

### 3.2 Software Development Methodology

Software development methodology which was preferred is scrum. It is an iterative and incremental development methodology which allowed researcher to deliver working software over a regularly scheduled intervals. This framework emphasizes on constant iteration, communication and improvement. The approach made it easy to quickly build a prototype and iterate the development process without constraint by a static development plan. Iteration is essential aspect in the model creation process since different set of models, parameters and hyper-parameters must be experimented and recorded. The set combination giving best result was selected in building the final model prototype. Further, image database was to be updated regularly by the client and model must be similarly updated to reflect this change. Several sprints will were created, the performance evaluated in every iteration as represented the figure 3.2.1.





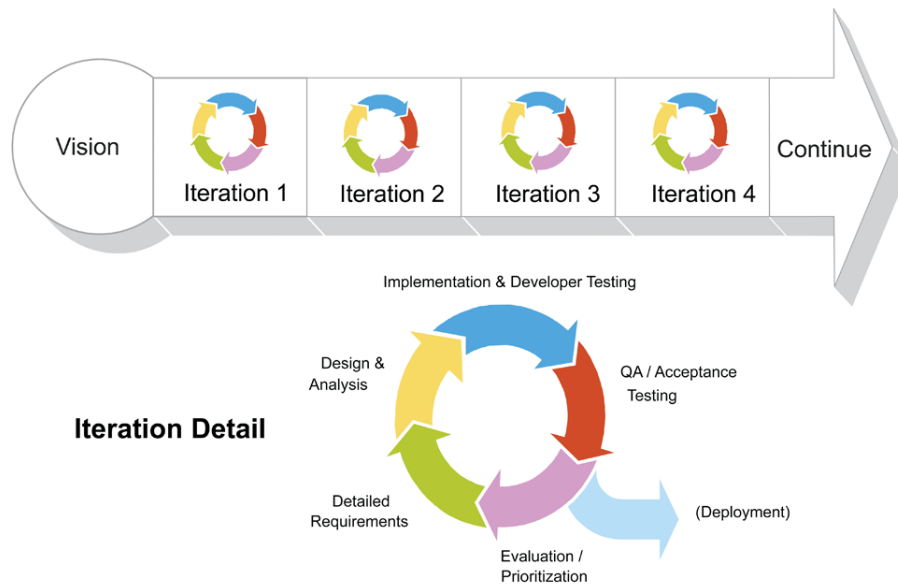


Figure 3.2.1: Scrum Agile Development Methodology

### 3.3 Research Design

CTC's are detected by analyzing the image data. These images were generated through a medical staining procedure in order to make the cancer cells or the tumor markers visible under the imaging device such as microscope. Making decision whether the stained sample has CTC or not and an approximate number may be quite daunting and time consuming. Often, many medical experts even differed at times; obtaining different result from the same sample with same imaging devices. The following sections will describe: i) a general research framework, ii) data acquisition, iii) data pre-processing, iv) data splitting; subsets for training and evaluation and iv) feature extraction and model creation.

#### 3.3.1 Research Framework

Building CTC detection and enumeration model was comprised of four phases. First, the image preprocessing phase; the images reshaped and regions of interest identified and extracted. Secondly, feature extraction; the informative features were extracted from input image for training. Thirdly, the training phase; learning algorithm figuring the best set of features and finally, the testing phase; the trained model tested and its performance evaluated on the held out data set. These processes have been summarized figure 3.3.1.

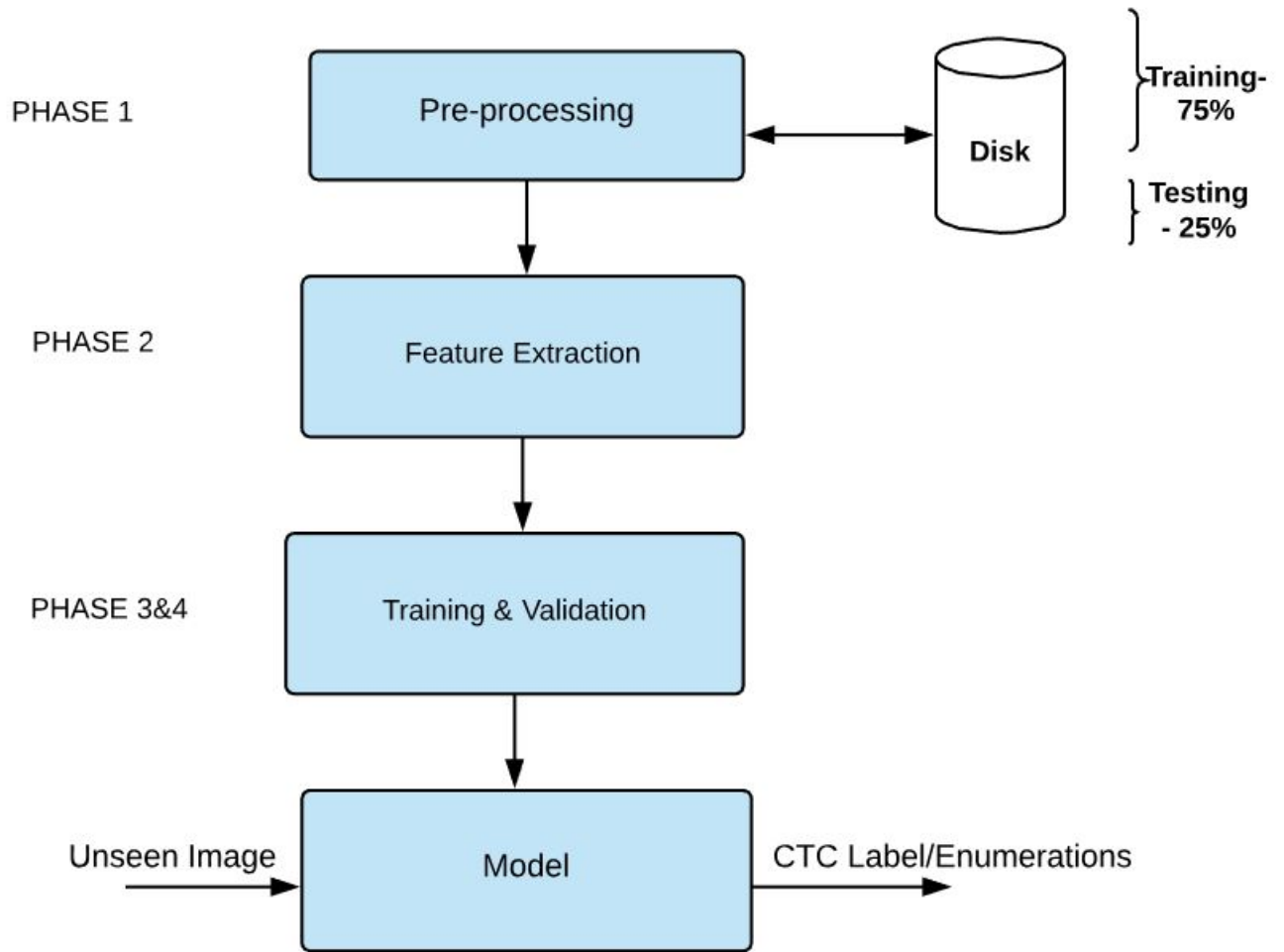
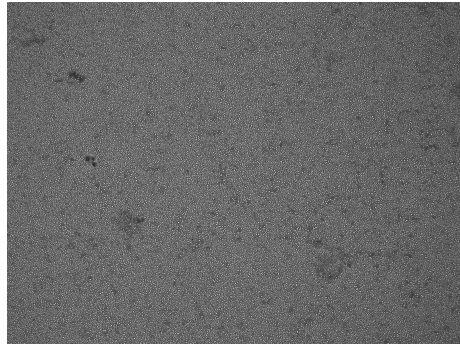


Figure 3.3.1: Research Framework; preprocessing, Feature generation, Training& validation

### 3.3.2 Data Acquisition

Data used in this researcher work was secondary data. It was provided by researcher named who is a PhD candidate at Missouri University of Science and Technology. Figure 3.3.2, depicts a sample microscopy image and the corresponding florescence image. The latter is used to locate the exact location of the CTC's in the former. The image sets bear varying inherent format and shape. The dataset provided is made up of 189 distinct images which were be used to extract the region of interest to generate both the positive (with CTC) and negative (without CTC) training sets. A single microscopy image had one or more positive/negative training sample.



(a) A microscopy image



(b) Fluorescence image depicting location of CTC's in 3.3.2 (a)

Figure 3.3.2: (a) A microscopy image with CTC's, (b) Fluorescence image with CTC locations in 3.3.2 (a)

From a total of 189 provided images, 1904 training set composed of negative and positive samples were generated. Half of the number, 952, were positive samples and other half negative. Out of this, 80% of the dataset, 1523, was used for training the algorithm and remaining 20%, 383, used for testing the trained model.

### 3.3.3 Image Preprocessing

This process involved getting the image data ready for training the model. First, all the microscopy images were converted to a one channel image instead of the 3 constituted by the RGB intensity values. Secondly, all the fluorescence images were converted to gray scale and then thresholded. Thresholding helped to accurately locate the circulating tumors before being mapped on the corresponding microscopy image. Thirdly, regions of interest were be cropped composing the positive and negative samples. Lastly, all the images will be formatted to same height,  $x$  and width denoted by  $y$ .

### 3.3.4 Data Splitting

Following the image preprocessing step, positive and negative sample instances were stacked forming matrix data structure. This was be persistently stored and then loaded later for training. Before initiation of the latter process, the dataset was shuffled and then split randomly into two subsets; training and testing. Shuffling reinforced randomness hence reducing bias during training. The training set, composed of the 75% of the dataset, was be used for training the model while testing set, the remaining 25%, was used to evaluate the model performance. Both sets had feature vectors and corresponding labels hence a supervised learning task.

### 3.3.5 Feature Extraction

Features were extracted manually, for instance raw pixel intensities, then fed into a learning algorithm or learned during training process. In this research work, these two approaches to feature extraction were experimented and the best performing method opted.

Convolutional Neural Network (CNN) architecture have the ability to learn the best representative sets automatically. According to Nielsen (2015), one or more feature maps can be created in a Convolutional layer. In this layer, a kernel size of a  $m * m$  was defined and passed across the entire image to create the feature maps. This is called convolution. Every feature map had its own set of shared weights and bias. These maps acted as the input to next layer of the network. A high level feature generation process is captured by figure 3.3.3.

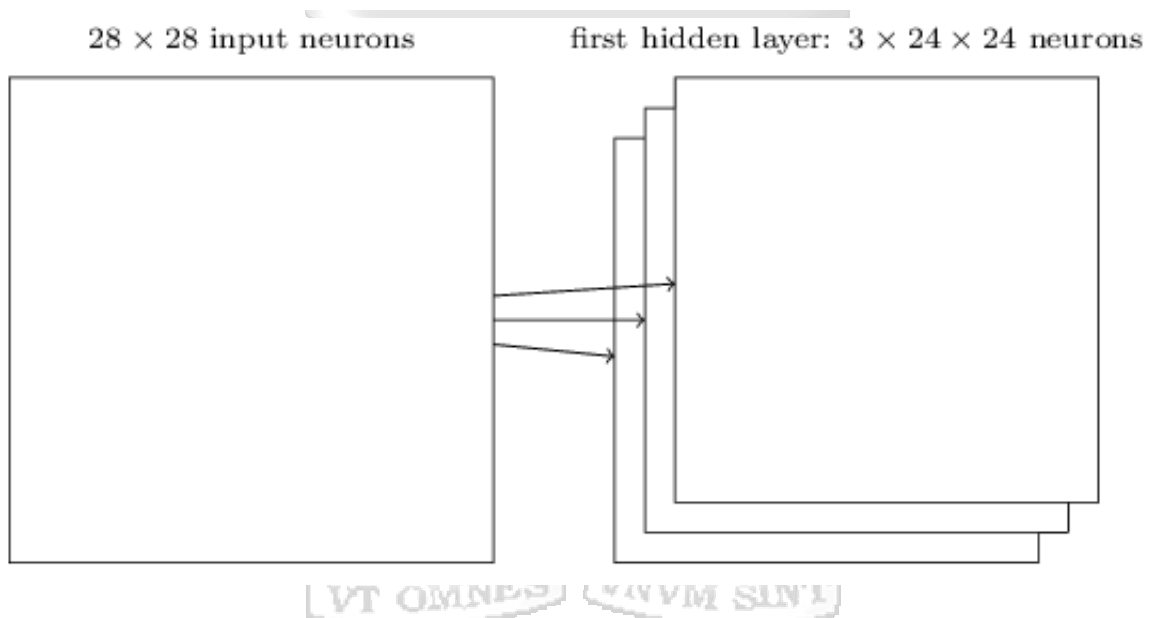


Figure 3.3.3: Convolutional. Adapted from Nielsen (2015)

### 3.3.6 Neural Network/ConvNet Architecture

The network model proposed in this study is shown in figure 3.3.4 and adapted from Mao et al. (2016). It is composed of convolution-pooling-convolution-pooling and two fully connected layers (FC).

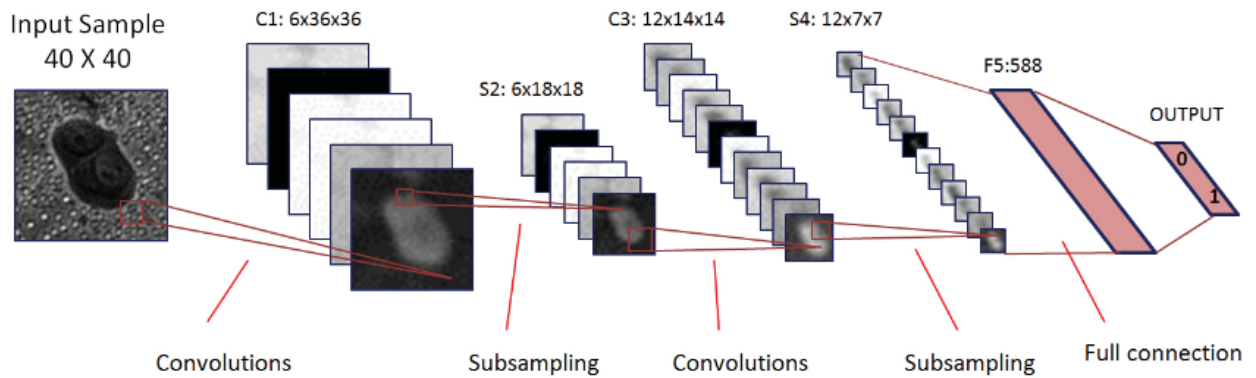


Figure 3.3.4: The ConvNet Architecture Adapted from Mao et al. (2016)

### 3.3.6.1 Loading Data

The training images were loaded into matrix  $X$  variable and training targets or labels loaded into matrix or vector  $y$ . The data was loaded in batches and fed during training. Weight initialization was done with a small amount of noise to prevent gradient from exploding or vanishing

### 3.3.6.2 Convolution, Activation and Pooling

Based on the design architecture shown in figure 3.3.4, the input image volume of shape  $l * w * d$  where  $l$  is the length,  $w$  image width and  $d$  the number of channels was fed into the network. A filter matrix or kernel of shape  $5 * 5$  was defined and used to convolve the input based on the some padding and striding value parameters. This resulted in a set of  $K$  feature maps which were the representative of low level learned features. Every convolution step was followed by activation using Rectified Linear Units (RELU) defined by equation 3.2.1. RELU introduced nonlinearity and also handled gradient diminishing issue as the error is back propagated. Pooling sampled out set of values based on specified criteria. The max pooling used to downsampled values in a  $2 * 2$  subregion. This process have been is illustrated by figure 3.3.5. In the subsequent layers, the same processes were repeated and followed by two fully connected layers.

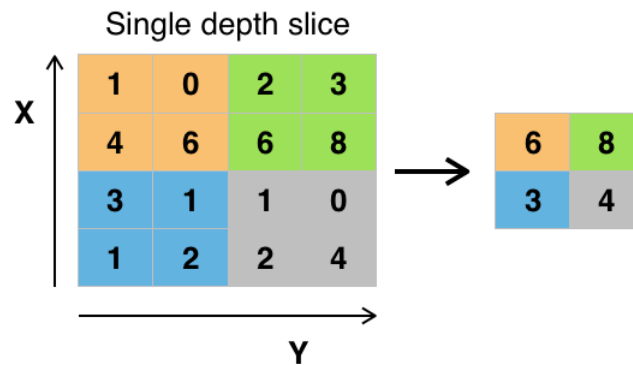


Figure 3.3.5: A 2 \* 2 Max Pooling. Maximum value in 2 \* 2 sub-region(Adapted from Deshpande (2016))

### 3.3.6.3 Fully Connected Layers

Output of the last convolution, feature maps, were unrolled to a densely connected layer of neurons. This enabled classification or regression based on the loss function defined. The FC layer was followed by a dropout layer which helped to reduce overfitting by inducing noise into the network through arbitrary turning on and off network nodes. This technique was proposed by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014), and have significantly increased the generalization ability of the neural network models. Softmax activation and Root Mean Squared were used to create probability distribution and predict continuous value respectively.

### 3.3.6.4 Training

The error output from the final layer was be used to train the model using backpropagation with stochastic gradient descent. This adjusted the weights and biases in a way that it reduced the cost function, logistic loss, to make the correct prediction.

### 3.3.7 Simple Classical Models

Apart from the neural network model, ConvNet, other simple classic models such as Support Vector Machines, Naive Bayes and Decision Trees were experimented based on the same feature set. Dataset shape, however, was be formatted to accordingly match algorithms' unique cases

#### 3.3.7.1 Model Testing and Performance Evaluation

The learned model was tested on the held out data set; the testing set. Its performance was evaluated using confusion matrix; precision and recall for classification and root mean squared

error to enumeration of CTC cells. Based on the learned weights and biases, the test image input was forward passed into the learned model to detect the features upon which the new prediction was made. The predicted value corresponded to 'yes' or 'no' or number  $x$ . A 'yes' meant that the sample input contained the circulating tumor cells and vice versa,  $x$  is the enumerated number of CTC's.

### 3.4 Research Quality, Validity and Reliability

This research model had to be evaluated and validated to ensure that the produced results are reproducible and stable. Quality, validity and reliability in this work was aimed to be achieved through various ways. Firstly, more than one model was developed and the produced results compared. Secondly, best learning recommended techniques such as dropout and regularization will were employed. These practices helped to reduce the model complexity hence a better generalization on unseen instances. Finally, all the models' performance were be benchmarked against standard machine learning metric measures. These have been formulated by equations 21 below.

$$Precision = \frac{TP}{TP + FP} \quad (19)$$

$$Recall = \frac{TP}{TP + FN} \quad (20)$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (21)$$

**where**

TP - True Positive

FN - False Negative

TN - True Negative

FP - False Positive

### 3.5 Ethics Considerations

Due to privacy concerns and existing patient data acquisition regulations and policies, the researcher sought for permission from relevant authorities and ethics bodies to gain a rightful access to any data and information used in this work. Although, real identity of an individual was not an input in this research context, user privacy control mechanisms in place were valued.

# Chapter 4: System Analysis, Design and Architecture

## 4.1 Introduction

In this section key system requirements are presented; both functional and non-functional. It also discusses system design and architecture based on conceptual models, data flow, use case, sequence and design class diagrams.

## 4.2 Requirement Analysis

### 4.2.1 Functional Requirements

- i). System should accept stained image input from the user
- ii). The system should transform the image and extract set of features
- iii). The system should detect CTC contained in the uploaded sample
- iv). The system should enumerate number of CTC contained in the sample instance
- v). The system should display the response whether the current sample contains CTC or not and quantify the number.

### 4.2.2 Non-Functional Requirement

#### 4.2.2.1 Ease of re-training

The system should be easy to retrain and adjust the weights. The addition of new dataset samples requires tuning of the model weights to account for the new instance variation. This ensures that the performance of the algorithm is improved.

#### 4.2.2.2 Robust generalization

The system should be able to generalize the new instances with no complexity and a reasonable bias-variance trade off. A complex model based system is often termed as overfitting and hardly generalize well on new test cases.

#### 4.2.2.3 Security

System should be secure prevent any unauthorized changes to the model parameters or low level system functionalities. Modifications made to the model must be authorized by the system to guarantee reliable system behavior.



#### 4.2.2.4 Persistent Weight Storage

The system should persistently store the model weights to avoid re-training every time a prediction is made. This will guarantee short time response on new test cases sample detection and enumeration of CTC's. Other parameter values should also be stores in the same way to allow for model update on new train instances.

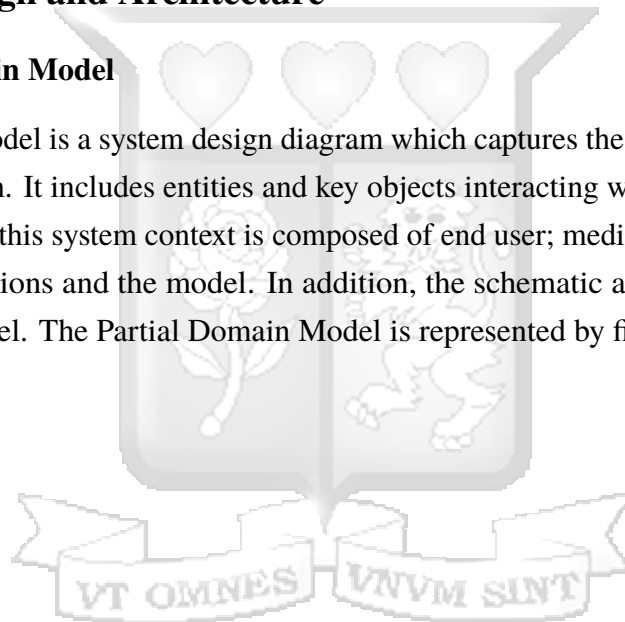
#### 4.2.2.5 Adaptation

The system should adapt gradually; constantly improving based on different exposure to different training dataset. This ensures relevant response is given based on the unseen input instances.

### 4.3 System Design and Architecture

#### 4.3.1 Partial Domain Model

Partial Domain Model is a system design diagram which captures the most important concepts of the proposed system. It includes entities and key objects interacting with the system. Instances of latter and former in this system context is composed of end user; medical expert, stained image, feature set representations and the model. In addition, the schematic also indicate the possible attributes at a high level. The Partial Domain Model is represented by figure 4.3.1.



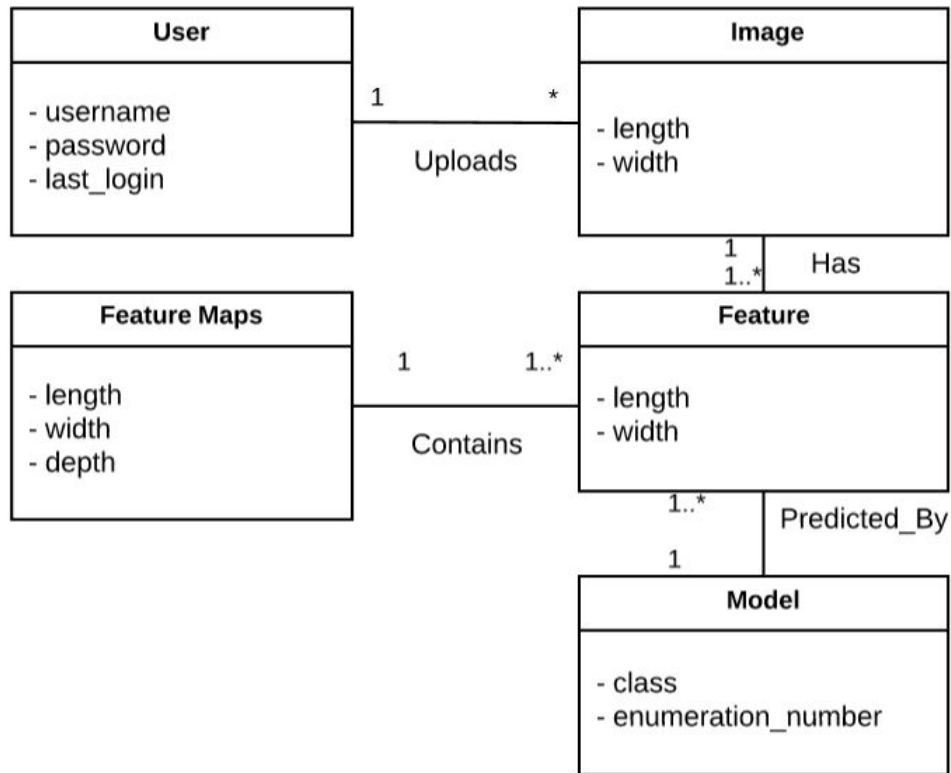


Figure 4.3.1: A Partial Domain Model

### 4.3.2 Use Case Diagram

Use case diagram is used to represent the actors interacting with the system. The primary actor is the medical expert who will be responsible for uploading the input image into the system. The actor is will also interact with authenticate and generate use cases as well as the label process. System Use Case Diagram have been represented in figure 4.3.2.

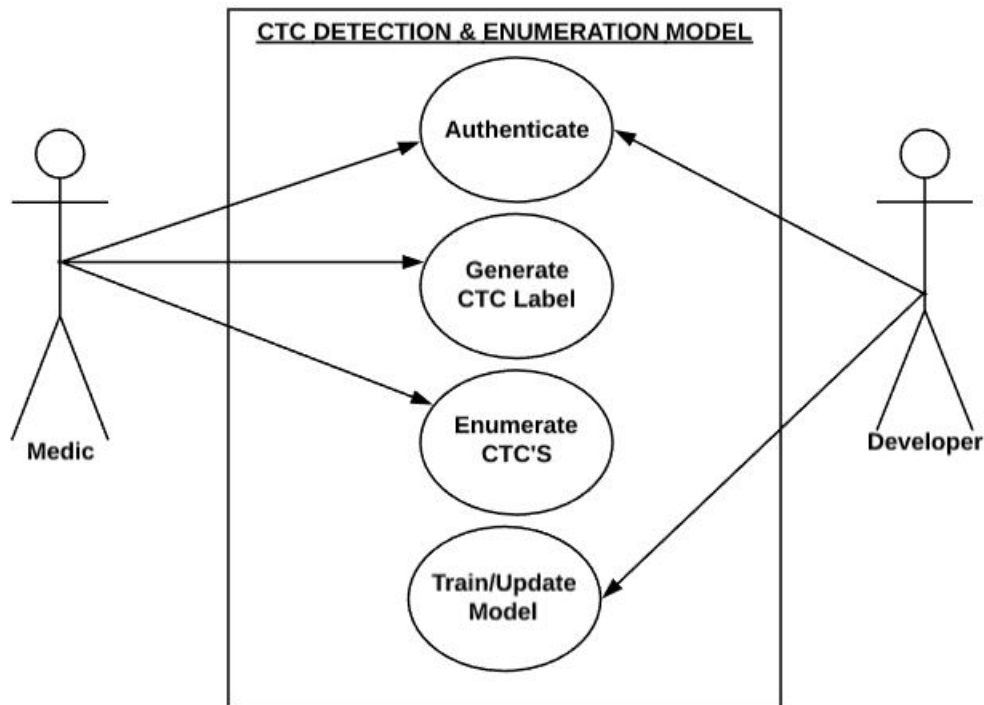


Figure 4.3.2: Use Case Diagram

### 4.3.3 Use Case Basic Flows

This subsection highlights the success scenarios, primary actors, preconditions and post conditions of different uses represented by figure 4.3.2.

**Use case :** Enumerate CTC

**Primary Actor:** Medical professional

**Pre Conditions:** Image input must have width and height of 40 \* 40

**Post Conditions:** Generation of CTC number in an image sample

**Basic Flow:**

- 1) The user uploads the stained image sample
- 2) Batches of similar size each 40 by 40 are generated from the input instance
- 3) Each batch pixels is normalized by diving each pixel value with highest pixel value in image;  
255
- 4) Representative features are extracted from the normalized image
- 5) Model weights are loaded from the persistent storage into disk
- 6) Image features are weighted by the model

- 7) The model makes predictions and shows the number of CTC's in the sample instance to the user

**Extensions**

If 2 fails;

- a) Reshape the batch instance to height and width 40 and 40 respectively or
- b) Discard the sample

CTC detection use case flow is functionally similar to the represented CTC enumeration process. The only difference is that, the task involved is classification instead of regression.

**Use case :** Model Update

**Primary Actor:** Developer

**Pre Conditions:** Model weights and new image samples must exist. The weights stored persistently

**Post Conditions:** All the weight values must be updated

**Basic Flow:**

- 1) The developer loads model the weights into memory
- 2) The model is bootstrapped using the loaded weights
- 3) The new stained image samples are loaded into memory
- 4) Batches of similar size each 40 by 40 are generated from the new samples
- 5) Each batch pixels is normalized by diving each pixel value with highest pixel value in image; 255
- 6) Representative features are extracted from the normalized image
- 7) The features are used to make predictions and weights adjusted using gradient descent learning rule
- 8) The new weights are persistently stored

**Extensions**

- a) If 1 fails, initialize the weights randomly
- b) If 3 fails, terminate the weight update process and notify the user about the error.
- c) If 4 fails, reshape the batch instance to height and width 40 and 40 respectively or discard the sample

#### 4.3.4 Data Flow Diagram

Data flow diagram (DFD) depicts how data flows in the system. It captures the external entities interacting with the system such as the user. It also shows all the processes involved in the system for instance training, model controlling and prediction. Finally, the diagram depicts data stores interaction with the system processes and entities. The DFD diagram is shown in figure 4.3.4.

#### 4.3.4.1 Context Diagram

Context diagram captures high level data flow between the major processes alongside inputs and outputs involved. This schematic is shown by figure 4.3.3

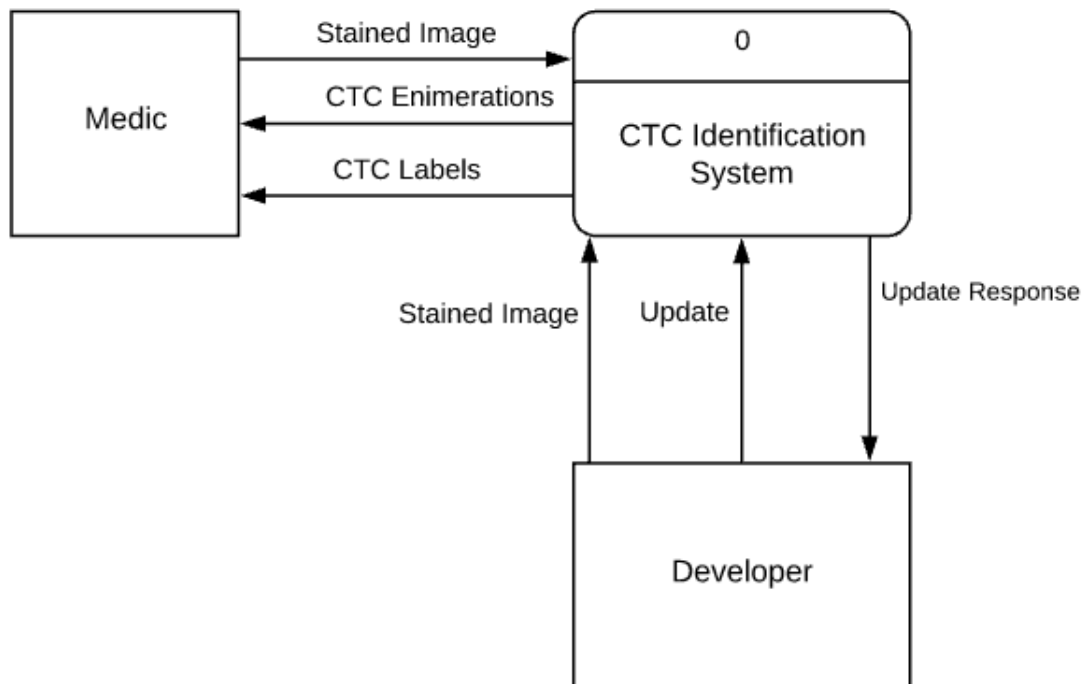


Figure 4.3.3: DFD: Context Diagram

#### 4.3.4.2 Level 0 DFD

This DFD shows succinctly shows all the processes, data stores and data flow with all the interactions involved between. Inputs and outputs are also shown. The level 0 DFD captured by figure 4.3.4.

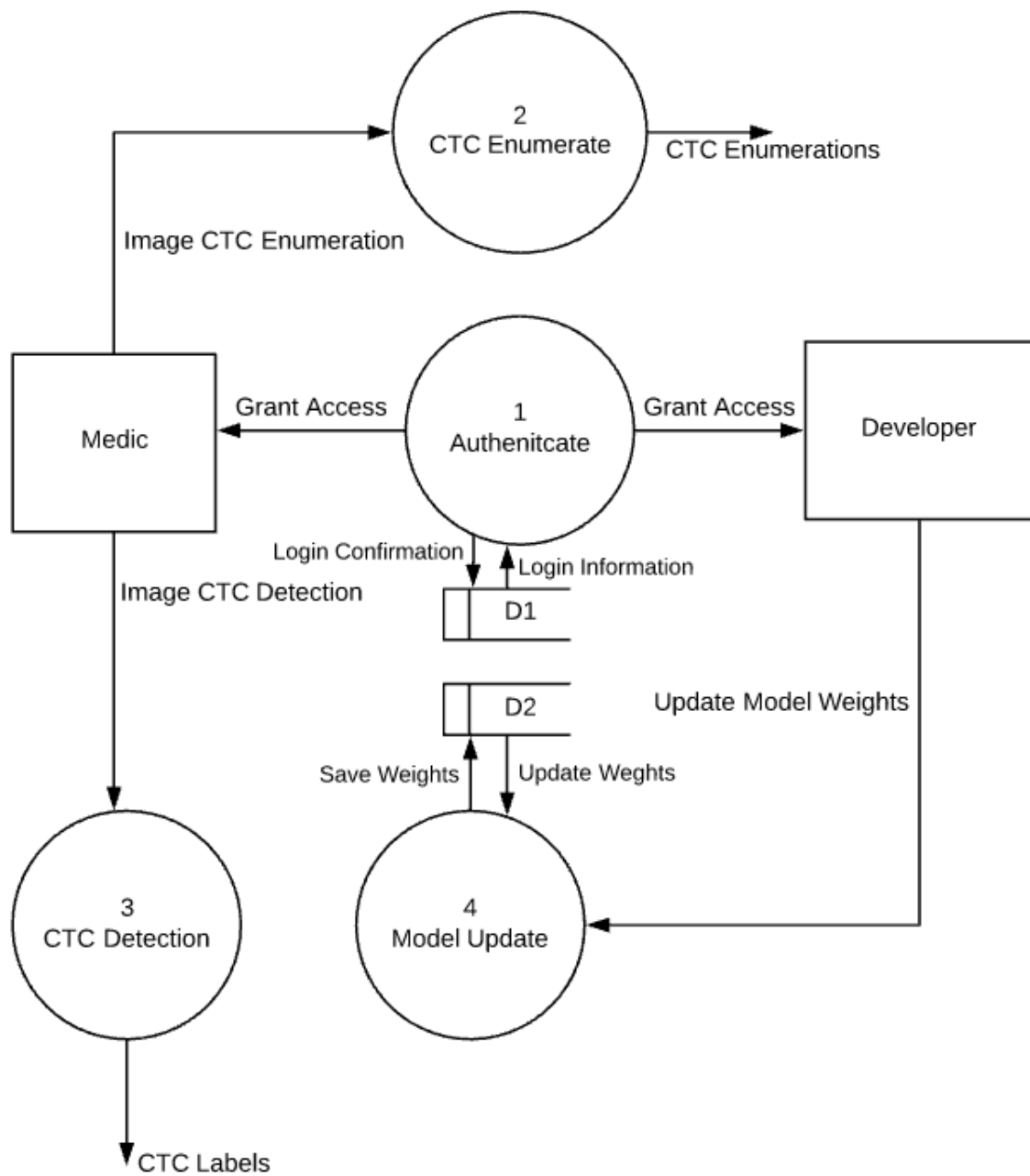


Figure 4.3.4: DFD:Level 0

### 4.3.5 Sequence Diagram

Sequence diagram is used to depict the inter-object interaction in the system. The proposed system will contain seven objects namely; the user, train\_model, load\_image, preprocessor, train\_model, save\_model and model predict. Each of these will encapsulate their properties and methods. The former are used as variables in design classes whereas the latter accomplishes specific behavior

based on the object scope. Method parameters have also been indicated by the sequence diagram in figure 4.3.5.

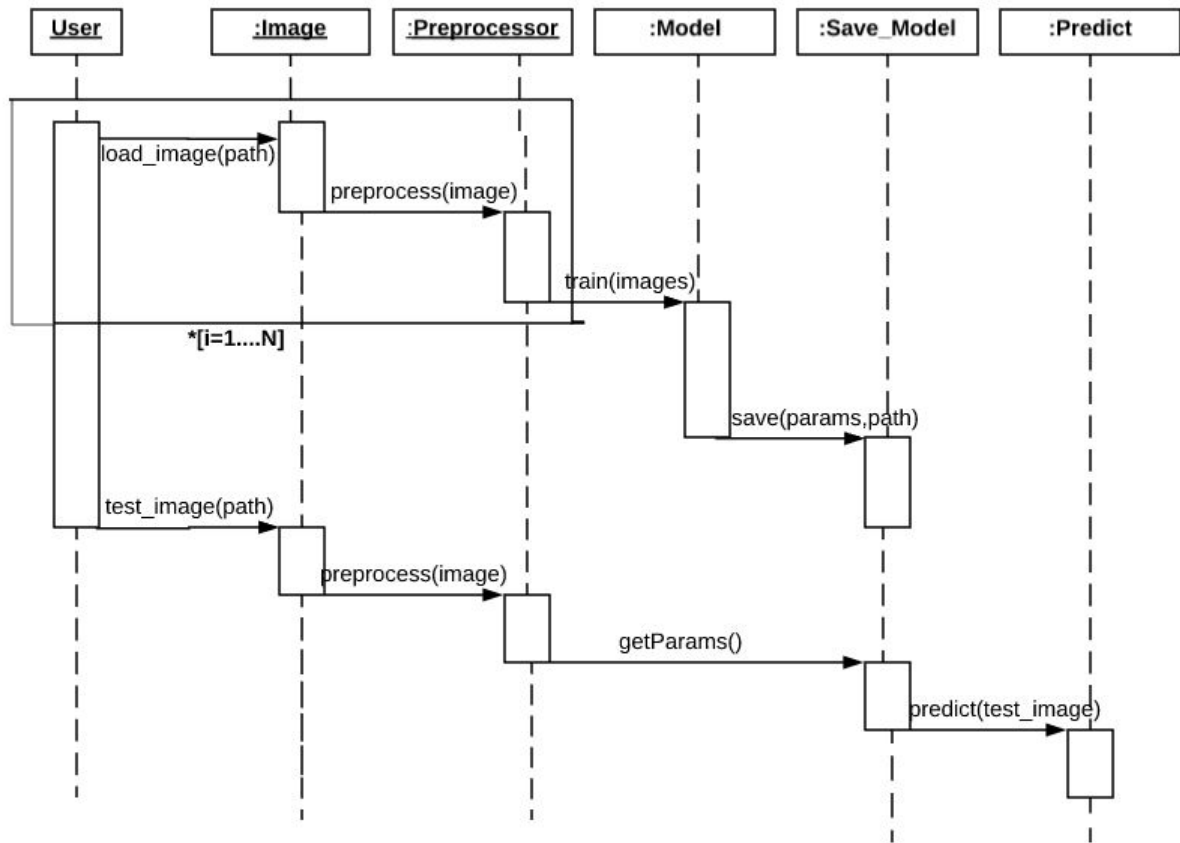


Figure 4.3.5: Sequence Diagram

### 4.3.6 Class Diagram

Class diagram also known as the Design class diagrams (DCD) represents object classes. A class can be implemented in wide range of objected oriented programming language. Every class have a unique name, set of attributes and methods. Every object depicted in the sequence diagram have a corresponding class since objects are instances of a class. DCD shown in figure 4.3.6 further depicts the relationship among the classes in the system.

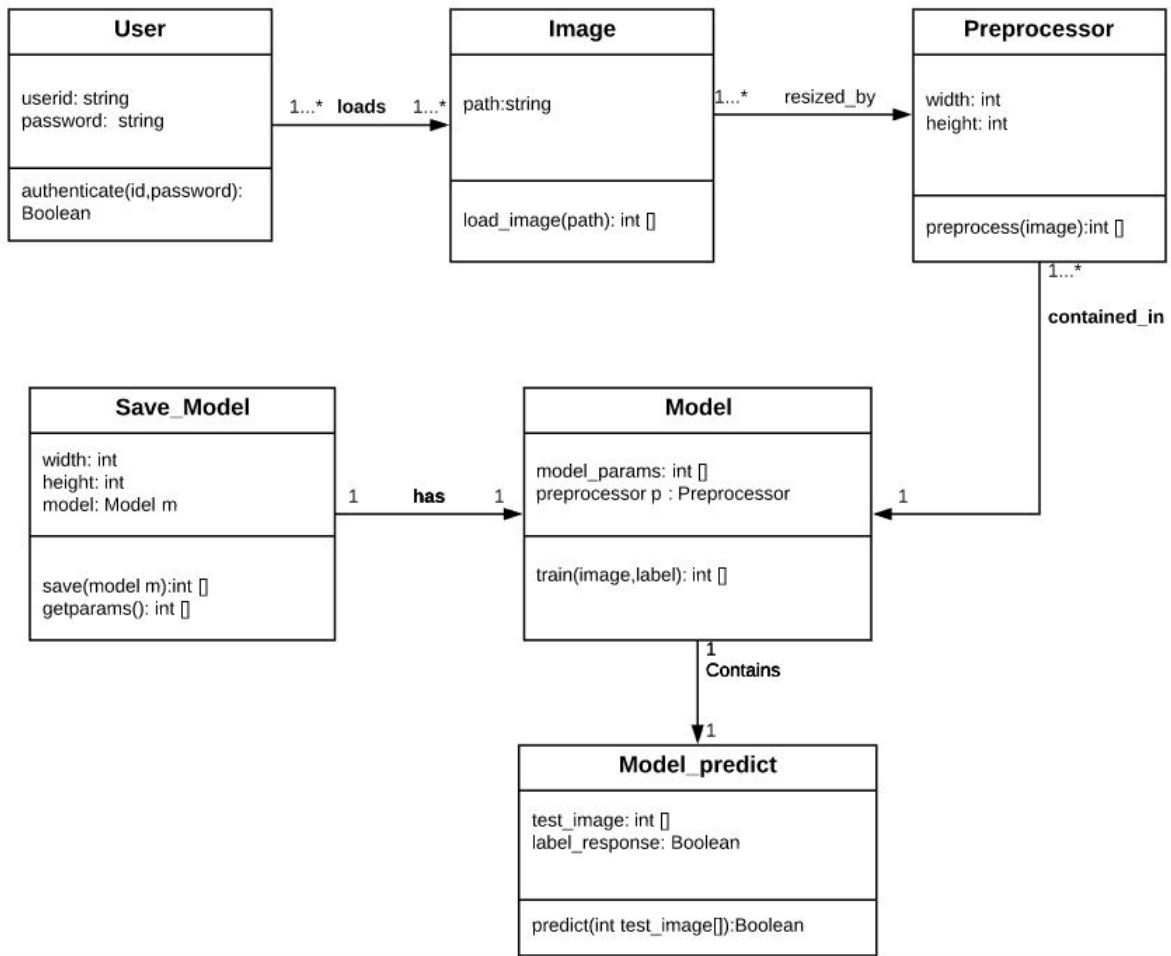


Figure 4.3.6: Design Class Diagram





# Chapter 5: System Implementation and Testing

## 5.1 Introduction

This section describes the carried out in model creation process. Firstly, it highlights specific software and hardware configuration settings, the data preprocessing steps, positive and negative samples generation, training and testing data set generation. Secondly, it describes different set of the models and learning algorithms experimented in tandem with the performance based on precision, accuracy and recall. It concludes by describing the model selection process based on the experiments carried out. The selected model is then used to build final a prototype for this research work.

## 5.2 Hardware & Software Environment

The model experiments were carried out in Google Cloud platform. A cloud environment was preferred due to the high demand for computing resources; memory and processing. Python was the core development language. Machine learning and computer vision extension libraries used include: Scikit-learn (Pedregosa et al., 2011), Keras (Chollet et al., 2015), OpenCV (Bradski & Kaehler, 2000), Numpy (Walt, Colbert, & Varoquaux, 2011) and Tensorflow (Abadi et al., 2016). The hardware and software development environment specifications have been summarized in the table 5.2.1.

Platform	Application	Specification
Software (Python 2.7)	Keras	2.1.1
	Tensorflow	1.4.0
	Numpy	1.14.1
	OpenCV	3.3.0
Hardware Google Cloud)	RAM	8GB
	vCPU's	7

Table 5.2.1: Hardware & Software Specification

## 5.3 Training Dataset Generation and Preprocessing

From a total of 189 provided images, 1904 training set composed of negative and positive samples were generated. Half of the number, 952, were positive samples and other half negative. Out of this, 80% of the dataset, 1523, was used for training the algorithm and remaining 20%, 383, used for testing the trained model. Figure 5.3.1, shows two types of images. Image 5.3.1: (a) is a

microscopy image containing Calculating Tumor Cells (CTC's). Figure 5.3.1: (b) on the other hand depicts an image which is showing actual location of CTC's in the corresponding microscopy (5.3.1:(a))

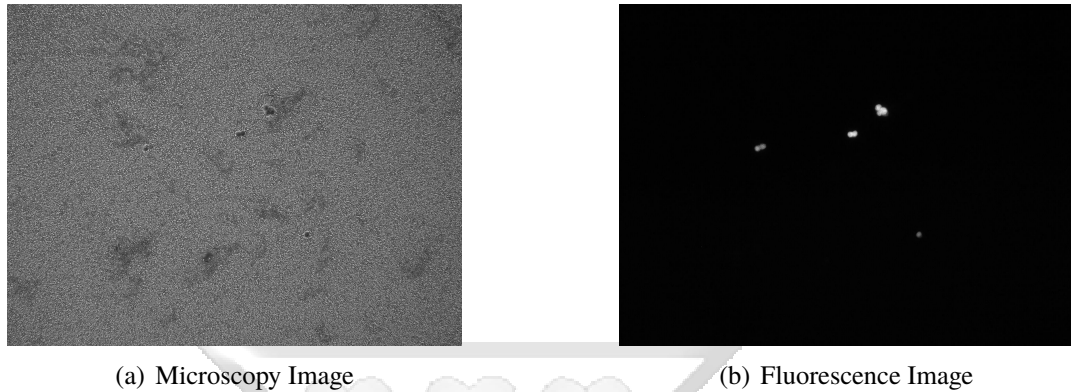


Figure 5.3.1: (a) Microscopy Image with CTC's, (b) Fluorescence Image with CTC's location(s)

The fluorescence image was converted to a gray scale with a single channel. A simple python function which accepts the image name and the path invoking this OpenCV's functionality was used to achieve this. The code listing 1 depicts this.

```
// Convert Image passed to gray scale
def read_image_bw(name=None,path=None):
    image = cv2.imread(path+name)
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Listing 1: RGB to gray scale

Conversion to gray scale was then followed by thresholding. This is image analysis method which allows grouping of image pixels into unique classes. In this case, binary thresholding was applied where the input image pixels are converted to either of the two predefined values, black and white (Gonzalez & Woods, 2002). This is achieved by converting pixel intensity values less than 240 to 0 and 255 otherwise. This approach allowed for a precise location of the CTC's in fluorescence image. Code implementing this functionality is as shown in listing 2.

```

def read_image(name=None):
    image = cv2.imread(HOME+name)
    grey_scaled = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, threshold_image = cv2.threshold(grey_scaled,
                                       240,255,
                                       cv2.THRESH_BINARY)

    return threshold_image

```

Listing 2: Binary Thresholding

Every fluorescent image is thresholded using the same function and  $x, y$  coordinates of white pixels, 255, are located. These locations are mapped to corresponding location on gray scaled microscopy image. This process is then followed by cropping out a positive 40 by 40 and a negative, also 40 by 40, samples. The negative set is generated by randomly generating  $x, y$  coordinates and cropping it out. The two samples are stacked vertically on separate matrices. This process is repeated for all other images appending each new pair to its corresponding matrix data structure.

The resulting positive and negative matrices had 3 dimensions; batch size, length and width. Length and width dimensions defines a sample image size in terms of pixels. On the other hand, batch, also known as the depth, indicates total the number of stacked images in the data structure. Negative and positive samples generation and stacking processes can be summarized programatically by code listing 3.

```

def crop_coordinates():
    pos = np.empty((0,40,40),dtype=float)
    neg = np.empty((0,40,40),dtype=float)
    store_xy = defaultdict(list)
    all_coordinates = get_all_coordinates()
    for img,xy in all_coordinates.items():
        for loc in list(xy[-1]):
            ctc_pos,ctc_neg = generate_negpos_sample(int(loc[1]),
                                                    int(loc[0]),
                                                    read_image_bw(xy[0]))
            if ctc_pos.shape ==(40,40) and ctc_neg.shape ==(40,40):
                afeature_pos = np.expand_dims(ctc_pos,axis=0)
                afeature_neg = np.expand_dims(ctc_neg,axis=0)
                pos = np.append(pos,afeature_pos,axis=0)
                neg = np.append(neg,afeature_neg,axis=0)
    return pos,neg

```

Listing 3: Image Cropping Algorithm

A corresponding set of labels or targets were also generated for each set where label 1 assigned

positive set and 0 label for the negative. Aside from class generation, the actual number of CTC's were enumerated. These categorical classes were one hot encoded to retain the meaning of absence or the presence of CTC. The positive and negative sets were concatenated, and feature matrix then scaled by dividing all the pixel values by the maximum intensity value, 255. This squashed all the values to a range between 0 and 1 hence standardizing the values. Scaling is a standard machine learning feature engineering process which have been proved to increase the model generalization by reducing noise in the data. One hot encoded labels were also concatenated to form another matrix.

The resulting data was shuffled, then split into two portions, training and testing set, in a 80 : 20 fashion. Each of the set persistently stored on the disk for futher analysis and experimentation. The code listing 4 shows the implementation of this aspect.

```

positive, negative = crop_coordinates()
train_set = np.append(positive,negative,axis=0)/255
labels = np_utils.to_categorical(np.array([1]*
                                         positive.shape[0]
                                         +[0]*negative.shape[0]))
X_train, X_test, y_train, y_test = train_test_split(train_set,
                                                    labels,
                                                    test_size=0.20,
                                                    random_state=42)

outfile = "./train_set_final.npz"
np.savez(outfile,
         X_train=X_train,
         X_test=X_test,
         y_train=y_train,
         y_test=y_test)

```

Listing 4: Cropping, Positive/Negative Generation, Matrix Persistent Storage

Training feature set, training label, testing feature set and testing labels dimensions have been summarized by table 5.3.1.

Dataset	Dimension
X_train	(1523, 40, 40)
y_train	(1523, 2)
X_test	(381, 40, 40)
y_test	(381, 2)

Table 5.3.1: A summary of Training and testing dataset dimensions

Figure 5.3.2, shows a sample of a positive image and a negative image cropped and used in

training. In addition, negative CTC's contained 0 CTC cells whereas positive CTC contained  $x$  CTC cells.

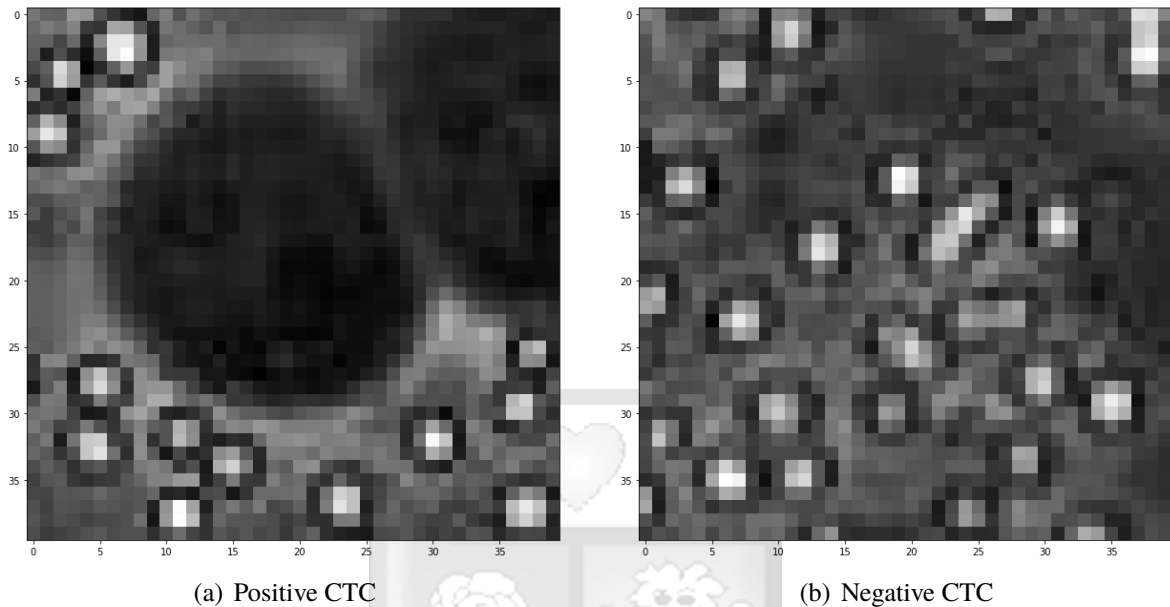


Figure 5.3.2: (a) Cropped Positive CTC (b) Negative CTC sample used in training

## 5.4 Training Machine Learning Models

### 5.4.1 Simple Classical Models

#### 5.4.1.1 Support Vector Machines (SVM)

Two support vector machine models were created using Scikit-learn API; linear and nonlinear. For the latter radial basis function (RBF) kernel was used. The training and testing feature set of dimension  $[batch * length * width]$  was reshaped to  $[1510, 1600]$ . The models results are presented in table 5.4.1 and 5.4.2.

Metric	Value (%)
Accuracy	90.74
Precision	90.00
Recall	91.00

Table 5.4.1: Linear SVM Results Summary

Metric	Value (%)
Accuracy	88.63
Precision	100.00
Recall	77.00

Table 5.4.2: Non Linear SVM Results Summary

#### 5.4.1.2 Random Forest

Random forest is a collection of Decision Trees. The trees vote together to give the best set of results. This model was developed using Scikit-learn API and the result presented by table 5.4.3.

Metric	Value (%)
Accuracy	88.15
Precision	96.00
Recall	81.00

Table 5.4.3: Random Forest Results Summary

#### 5.4.1.3 Multinomial Naive Bayes

Naive Bayes is a probabilistic model based on prior probabilities and maximization of the likelihood. It was experimented using Scikit-learn API. Obtained results recorded table 5.4.4 below.

Metric	Value (%)
Accuracy	89.94
Precision	100.00
Recall	80.00

Table 5.4.4: Multinomial Naive Bayes Results Summary

## 5.5 Deep Learning Models

### 5.6 Multilayer Perceptron (MLP)

A four-layer MLP developed using Keras Application programming interface (API). The training dataset of shape  $[batches, length, width]$  was reshaped resulting in a  $1510 * 1600$  shaped matrix. The first dimension represents the number of samples and next dimension representing the number of features unrolled across  $x$  axis. The network was trained for 500 epochs with back propagation and batch size of 200 in every iteration. Different activation functions were used on the hidden layer. code snippet 5 captures the low level network implementation details using Keras API. The whole code have been appended on appendix. MLP performance is recorded in table 5.6.

```

def mlp_model(num_pixels,num_classes):
    model = Sequential()
    model.add(Dense(num_pixels,
        input_dim=num_pixels,
        kernel_initializer='normal',
        activation='relu'))
    model.add(Dense(num_pixels,
        input_dim=num_pixels,
        kernel_initializer='normal',
        activation='relu'))
    model.add(Dense(num_classes,
        kernel_initializer='normal',
        activation='softmax'))
    model.compile(loss='categorical_crossentropy',
        optimizer='sgd',
        metrics=['accuracy'])
    return model

# Learning
model = mlp_model(NUM_PIXELS,NUM_CLASSES)
model.fit(X_train_mlp, y_train, epochs=500,batch_size=200,verbose=2)

#Evaluation of the model
scores = model.evaluate(X_test_mlp, y_test, verpos)

```

Listing 5: MLP Model Code Sample

The MLP model results are summarized by table 5.6 below.

Metric	Value (%)
Accuracy	92.85
Precision	94.65
Recall	50.42

Table 5.6.1: Multi Layer Perceptron Results

## 5.7 Convolutional Neural Network (ConvNet)

ConvNet is a variant of neural network. It was created using Keras API. In the first convolutional layer, a 40 by 40 input image was fed into the network with a kernel of size  $5 * 5$  and  $1 * 1$  stride size. This was followed by a  $2 * 2$  max pooling layer. In the next convolutional layer, a kernel of size  $5 * 5$  was used and activated by rectified linear units (RELU's). Two fully connected layers

then succeeded, first one being activated by RELU and the second being by softmax functions. The latter function enables generation of a probability distribution. Model was trained for 500 epochs. The code listing 6 shows this process.

```
def cnn_model(input_shape=(40,40,1)):
    model = Sequential()
    model.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1),
        activation='relu',
        input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(12, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(588, activation='relu'))
    model.add(Dense(NUM_CLASSES, activation='softmax'))
    model.compile(loss='categorical_crossentropy', \
        optimizer='adam', metrics=['accuracy'])
    return model

# build the model
model = cnn_model()
model.fit(X_train_cnn, y_train, validation_data=(X_test_cnn, y_test),
    epochs=200, batch_size=200, verbose=2)
```

Listing 6: CNN Model Code

Metric	Value (%)
Accuracy	98.42
Precision	97.86
Recall	49.19

Table 5.7.1: CNN Results

ConveNet error value against the number of iterations



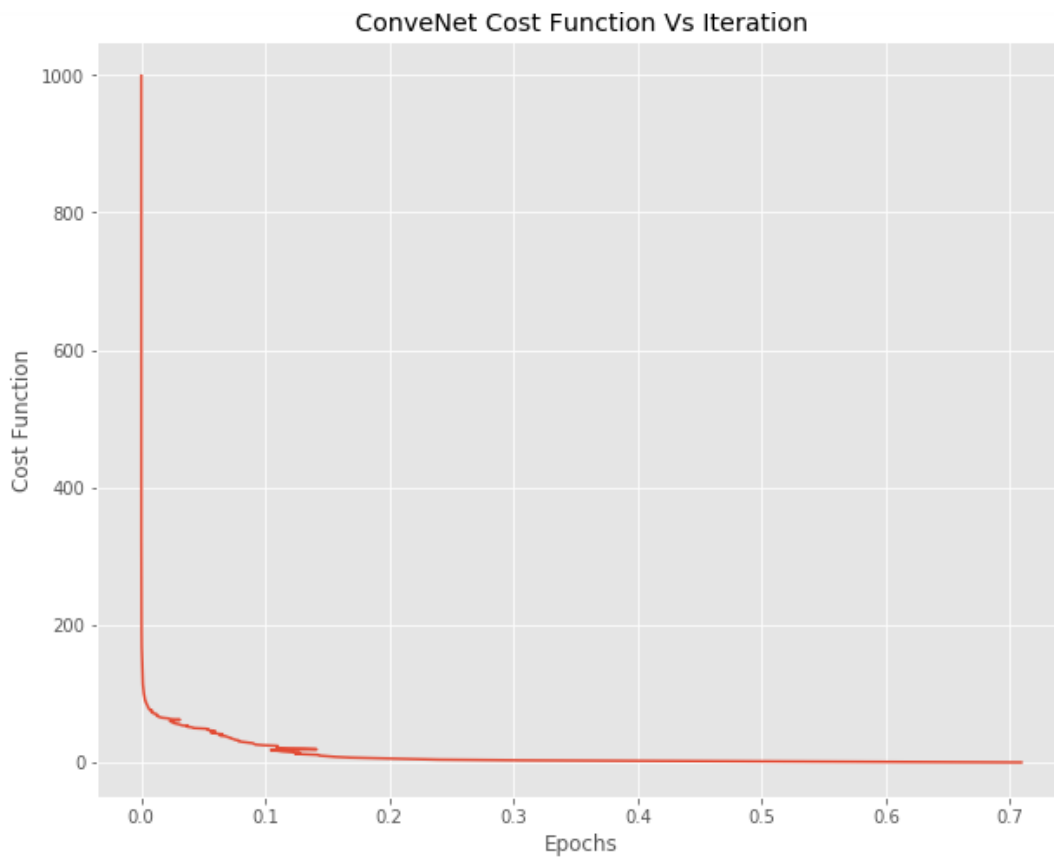


Figure 5.7.1: ConvNet Cost Function against Epochs

## 5.8 Capsule Neural Networks (CapsNet)

This was another deep learning model first proposed by Sabour, Frosst, and Hinton (2017), to address issues with ConvNet. It was trained for 500 epochs. The model performance is presented in table 5.8.1.

Metric	Value (%)
Accuracy	96.82
Precision	95.72
Recall	48.90

Table 5.8.1: CapsNet Results

## 5.9 Convolutional Neural Network for Regression

Following best set of performance obtained from ConvNet, we modified the network architecture to output continuous value instead of categorical. This allowed the researcher to predict the number of CTC's in a given sample. The output layer was modified to produce a continuous value and loss computation changed to compute root mean squared (RMSE) instead of categorical cross entropy. RELU is the only activation function which was used in the hidden layers. Update of the network weights and bias remained same as that of ConvNet for classification. RMSE was used since this is a regression task. Figure 5.9.1, captures the changes in error values over epochs. Code listing 7 shows a low level implementation of this.

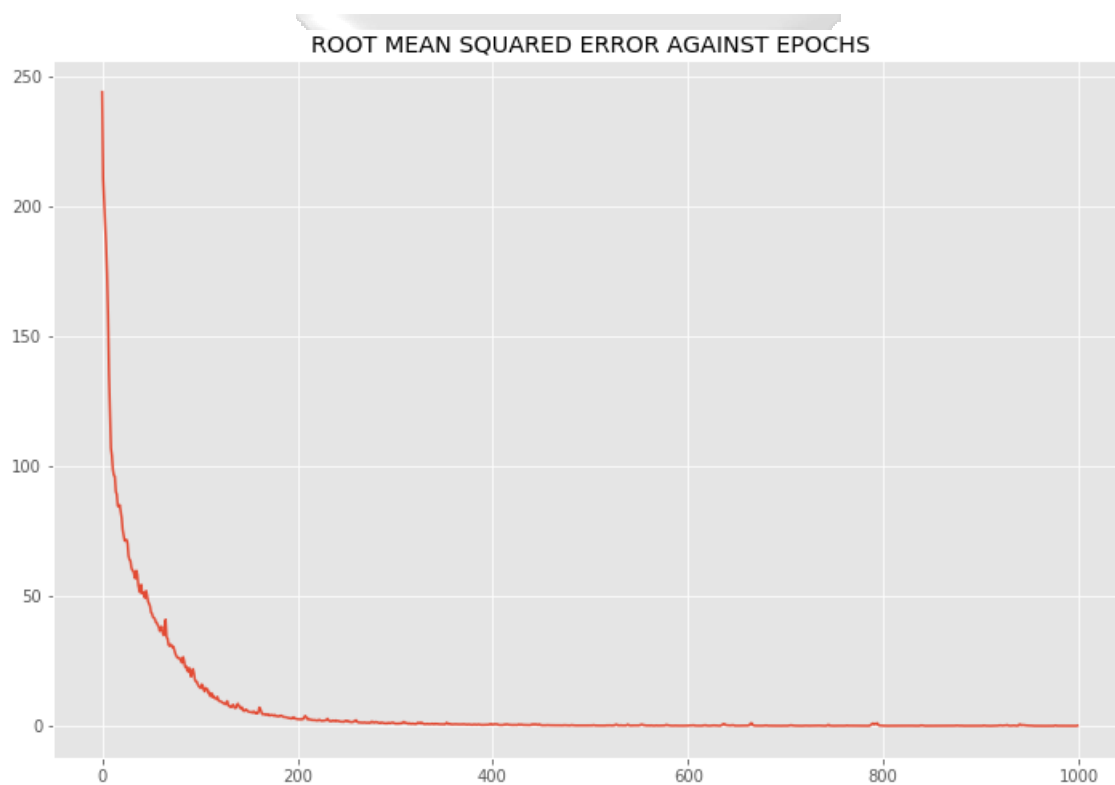


Figure 5.9.1: RMSE changes relative to number of Epochs

```

def cnn_model(input_shape=(40,40,1)):
    model = Sequential()
    model.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1),
    activation='relu',
    input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(12, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(588, activation='relu'))
    model.add(Dense(1))
    model.compile(loss="mean_squared_error", optimizer='adam',
    metrics=['mse'])
    return model

```

Listing 7: ConvNet Regression Code

## 5.10 Model Evaluation

The performance metric of every detection, classification, model was based on accuracy, recall and precision. A custom function was created to implement this functionality. It accepts the predicted and actual labels then computes the metrics based on the formulation presented at the end of chapter 3. Code listing 8 shows this computation. On the other hand, the regression model for CTC enumeration was evaluated using RMSE expressed mathematically as:  $\sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})$ . The formula sums the difference between predicted ( $\hat{y}^{(i)}$ ) and actual values ( $y^{(i)}$ ). The resulting error values is back propagated and the network weights and bias updated using gradient descent rule (backpropagation).

```

def confusion_matrix(actual,predicted):
    tn,fp,fn,tp = 0,0,0,0
    for a,p in zip(actual,predicted):
        if np.argmax(a) == 1:
            if np.argmax(a) == np.argmax(p):
                tp+=1 # truly positive actual-predicted
            else:
                fp+=1
        else:
            if np.argmax(a) == np.argmax(p):
                tn+=1
            else:
                fn+=1 #falsely negative
    return tn,fp,fn,tp

```

Listing 8: Confusion Matrix

## 5.11 Model Selection

Following the experiments, best performing classification model was selected and further modified to numerically quantify the number of CTC's. Detection or the classification models performance were benchmarked on precision, accuracy and recall. The best one was chosen based on these metrics and pixel level stability. Two best models which exhibited these properties were; ConvNet and CapsNet. With reference to these, the latter had precision of 97.86% while the former had 95.72% and an accuracy of 98.42% and 96.82% respectively. ConvNet was selected for creating the final detection and enumeration model for this research work.

## 5.12 Testing New Instance

The ConvNet accepts  $N$  batches of new testing instances with size  $40 * 40$ , and then make a prediction. Before enumeration or detection task is performed, the input image is reshaped and transformed to conform with the original input height and width training set dimensions. These processes were accomplished by code listing 9.

```

def generate_test_ctc(test_image):
    vector = []
    splittedh = np.split(test_image,30)
    for split in splittedh:
        splittedv = np.split(split,40,axis=1)
        for vec in splittedv:
            vector.append(vec)
    return np.expand_dims(np.array(vector)/255,axis=3)

def read_image_bw(name=None):
    image = cv2.imread(name)
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

```

Listing 9: Transforming Test Instance



# Chapter 6: Discussions

## 6.1 Introduction

Main objective of this study is to develop a machine learning model of for identifying circulating tumor cells. Several classification models were experimented in chapter 5 and their performance benchmarked against precision, accuracy and recall. ConvNet model was selected as the best performer. Based on the same network architecture, the activation functions and loss function were modified so that it can produce a continuous value. This modification quantified the number of CTC in a sample image. In this section we discuss the process of model selection, validation and address how research questions have been answered.

## 6.2 Model Selection

### 6.2.1 CTC Classification Models

Several machine learning algorithms exist each having best set of problem it can address based on type and amount of data available. These algorithms majorly falls into two major categories; the classical and deep learning models. Knowing the best algorithm or model to use for a particular problem solely depends on experimentation, amount of data available and hyper-parameter tuning and feature engineering approach.

The classical machine learning algorithms are simple models which often depend on manual feature selection. On the other hand, deep learning models automatically learns the best informative features during the training process without or with little human intervention.

Manually generated features such as RGB histogram or pixel intensity though works well, suffers from high dimensionality. The number of weights, biases and hyper-parameters grows exponentially over time hence algorithmically expensive. This feature generation approach also work with the assumption that best set of attributes to represent an instance feature is known in advance which may not be the case. Instead of representing the features manually, such representations can be learned automatically. This is the core theoretical underpinning of deep learning.

Deep learning adoption was precipitated by Krizhevsky et al. (2012) work. In study, output obtained halved error value for object recognition during ImageNet competition. Convolutional Neural Network model architecture outperformed all the classical simple learning models which had been used before on the same dataset. The model with deep network layers, had ability to learn features automatically during the training process. During the process, each filter is tasked

at learning a specific representation in the image such as curves or edges.

Following this, other researchers have extended the original ConvNet model leading to unprecedented success in areas such as real time object detection (Redmon, Divvala, Girshick, & Farhadi, 2016), (Ren, He, Girshick, & Sun, 2015), face detection (Koch, Zemel, & Salakhutdinov, 2015), transfer learning (Pan & Yang, 2010) and large scale object detection architectures (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016).

In this research work, two set of learning models were experimented; classical and deep learning. The former set of models developed included : Support Vector Machine; linear and non linear, random forest and Naive Bayes. The performance of these models have been summarized by table 6.2.1.

<b>Model</b>	<b>Accuracy (%)</b>	<b>Precision (%)</b>
Linear SVM	90.74	90.00
Non-Linear SVM	88.00	100.00
Multinomial Naive Bayes	89.16	96.60
Random Forest	89.94	100.00
Multilayer Perceptron	92.85	94.65

Table 6.2.1: Classical Models Results Summary

From table 6.2.1, the best accuracy metric on test set was obtained by linear support Vector machine, 90.74%. On the other hand, best precision metric was achieved by multinomial Naive Bayes and non linear Support Vector Machine, 100.00% on the same dataset

On the same testing and training set, deep learning models were developed. These are; Convolutional Neural Networks (ConvNet) and Capsule Neural networks (CapsNet). The latter was first proposed by Sabour et al. (2017), aimed at addressing the ConvNet issues by introducing a dynamic routing algorithm and lower level capsules sending their outputs to a higher level capsules. The network proved better at classifying highly overlapping MNIST digits.

The results of these two deep learning models have been summarized in table in 6.2.2

<b>Deep Learning Model</b>	<b>Accuracy (%)</b>	<b>Precision (%)</b>
ConvNets	98.42	97.86
CapsNet	96.82	95.72

Table 6.2.2: Deep Learning Models Results Summary

From the table 6.2.2, ConvNet outperforms capsule neural network both in terms of accuracy and precision with a margin of 2.02% and 2.14% respectively. This is quite a significant error margin even though CapsNet have a much better pixel intensity stability as compared to ConvNet.

Nevertheless, CapsNet still outperformed all the classical machine learning models such as MLP and SVM. The difference in CapsNet and ConvNet accuracy and precision performance is capped at 3.97% and 1.07% respectively.

Deep learning models are capable of learning the best set of representatives in a high dimension space for discrimination of predefined classes. The filters or weights are updated through back propagation and minimization of cross entropy error. This is done over a number of epochs until optimal set of weights which minimizes the error is achieved. Following the result performance comparison and key desirable abilities of deep learning models, this work settled on ConvNet to build the final model.

### **6.2.2 CTC Enumeration Model**

ConvNet was selected as the best classification model. The same network architecture was modified to output a continuous value on the final layer. This was achieved by having one output node in the final layer of the network. Further, all the hidden layers activations were based on RELU function expressed as  $f(x) = \max(0, x)$ . The function produces the value itself as the output and 0 otherwise. Furthermore, the loss function was also modified to Root Mean Squared Error (RMSE) in order to get a continuous value output. Network weights and biases were updated using backpropagation rule.

## **6.3 Research Findings**

### **6.3.1 Cancer Monitoring Methods**

Cancer monitoring or management is just as essential as the diagnosis process. In chapter 2 of this work we discussed in depth the need for cancer monitoring and further reviewed some of the current methods which have been employed in cancer management. These included mammography, biopsy, liquid biopsy and the diffusion weighting schemes. We also presented how information technology have impacted on the monitoring through mobile phone, real time symptom reporting and image analysis automation. This approach have significantly contributed to a better cancer care among the patients and more subject control. Finally, we highlighted some of the issues attributed to the current monitoring techniques as compared to the proposed method.

### **6.3.2 Machine Learning Models for Cancer Monitoring**

In chapter 2, we reviewed set of machine learning models which have been applied in cancer management. From the literature, researcher found out that both regression and classification



models have been applied in monitoring of the disease. Though most of these methodologies have not been formally embraced by the medical fraternity, there is a potential drift towards in this direction due to surge in number of victims relative to human capital required. Deep learning models were also following the suite.

The typical machine learning use cases revolved around prediction of cancer recurrence, susceptibility, disease relapse and recovery. Further, more studies also reviewed the most machine learning methodologies which have been widely used in the process this processes. Support Vector Machines, Perceptron, Naive Bayes, Multilayer Perceptron, and  $k$ -Nearest Neighbors topped the list. Moreover, a summary of machine learning methods adoption in medical research over the years was also presented in the same the referenced chapter.

### **6.3.3 Extracting Features from CTC's**

From the literature review, there exist two techniques of extracting feature sets from an image sample. These are; automated or manual. Classical machine learning algorithms such as SVM's, ANN and MLP requires manual feature engineering. Most informative features describing an instance must be identified and selected before the commencement of training process. These are then used to discriminate the classes during classification or forecast the continuous value in regression, based on the learning problem at hand. On the other hand, in automated approach the best set of descriptive features are learned during the training process. Deep learning algorithms such as Capsule Neural Network or Convolutional Neural Networks operates in this fashion. The two methods to feature engineering were experimented. For manual, the pixel intensity values were used. In the automated context, these attributes were learned during the training session filters updated accordingly. The filters were optimized and iterated over multiple epochs to minimize defined convex cost function.

### **6.3.4 Developing and testing the model**

In this study, multiple machine learning algorithms were used to develop different kinds of models. The best one was selected and used to build the final model. We have developed model based on automated feature discovery through the deep learning approach. Further, the model developed was tested to ascertain the performance. The performance was benchmarked on precision, recall and accuracy metrics. The results were presented in chapter 5.

## 6.4 Model Validation

The model selected in this research was validated using standard approaches laid for machine learning algorithms. The best performing model was selected. Since this was a supervised learning, evaluating performance on accuracy, recall and precision was ideal. Each of these metrics have been presented in the previous sections. Accuracy, is the ratio of true positive to the sum of true positive and false positive, precision on the other hand can be defined as the ratio of true positive to the sum of true positive and false positive, recall is the ratio of true positive to the sum of true positive and false negative.

Different sets of classical models were experimented. Each have its own theoretical underpinnings and varying theory of learning. Support Vector Machine for example is based on hinge loss and maximization of the margin of separation, Naive Bayes; a probabilistic model based on maximization of the likelihood of the posterior probabilities conditioned on the prior values, Decision Tree; a hierarchical model based on entropy, Artificial Neural Networks; mimics the working of human biological neurons having multiple hidden layers and activations with ability to approximate any hidden function and  $k$ NN; mimicking the distance between  $k$  neighbors.

Though the model's results were not exactly the same, the results seemed to converge. Theoretically, many studies have suggested that, when there is a lot of data then the algorithm being used does not make a big difference. The larger dataset size, the better the model's performance despite the theoretical framework.

## 6.5 Conclusion

In this section we have discussed set of models which were created and how they were selected. Maximum margin classifiers, Entropy Based models and Neural Networks performance were discussed. These algorithms have different theory of learning and functional data mappings. The best performing algorithm was selected based on machine learning standard metric values and further modified to create the enumeration model.

## Chapter 7: Conclusions, Recommendations and Future Work

### 7.1 Conclusion

The main goal of this research work was to develop a machine learning tool which can help to monitor disease progression in cancer patients. When a subject is undergoing treatment, the effectivenesses of the treatment methodology can be determined by detecting the presence or absence of Circulating Tumor Cells (CTC's) indicators. Additionally, the enumeration of CTC's number serves the purpose more succinctly.

The researcher segmented the main goal into sub-goals by reviewing the literature on cancer monitoring practices, application of machine tools in cancer treatment and management, set of learning algorithms and feature engineering techniques which have given promising results among the research community.

Multiple set of classical and deep learning models were developed and their performance benchmarked against a metrics such as precision, accuracy and recall. Not only did this help the researcher select the best model but also validate the model. Different machine learning methods have different learning theoretical underpinnings. As a result, the array of experiments carried out helped to select the best model in tandem with a feature extraction technique.

Following the experiments, Convolutional Neural Network (ConvNet), a deep learning based model, outperformed most of the classical models in classification of CTC's. Based on this record, the network architecture was modified to further enumerate the actual number of CTC's in a sample instance. In this light, not only can we discriminate CTC samples from non CTC ones but also enumerate the possible number of cells. The final project's prototype inherited these characteristics.

### 7.2 Challenges

Despite the ability to achieve the main goal outlined for this research endeavor, quite significant challenges were encountered; though did not affect the project to greater extent. Firstly, there was a major lack of enough medical expertise input. Some hospitals and medics professionals request to be involved failed. Most of them were unwilling despite the multiple intervention attempts.

Secondly, medical doctors and radiologists we interacted with were not willing to share patient information due to privacy and health policies concerns. Thirdly, it was quite of a challenge replicating the liquid biopsy process, to get new stained image sample for new real world test instance, using a remote mobile device. The process is heavily dependent on the the already established medical procedures. The success would have further helped us to validate the proposed

and built model.

Lastly, getting right amount of data. The model in this study was developed on 1904 training instances. Theoretically, most machine learning tasks have proven to require a lot of data in order to generalize well by lowering the test error. The model developed by the researcher had a 97% accuracy. Consequently, this was accompanied with false positive and false negative classes. Reduction of the latter and former requires either increasing the amount of training data, advanced hyper-parameter tuning or both.

### **7.3 Recommendations**

The researcher have achieved set objectives of this study albeit 3.0% error rate model. To lower this error value and build a more consumable general model, we lay out following set of recommendations

- 1). This kind of research work should be done in close partnership with a medical facility such a local hospital. This makes it easier to further validate the model to assert the performance on test set is reflected in the real world setting.
- 2). To facilitate ease of adoption of this tool, the project should be carried out liaising with medics or the end users. This would help set a faster pace for the adoption since they'll be using something they're already aware of.
- 3). The hospital setting, further, would ensure more test images can be obtained easily to improve the model's performance. The algorithm accuracy can be surely increased by an increased exposure to new labeled instances from new patient subjects.
- 4). The medical community engagement, in addition, would make it easy to acquire expertise input early from the inception of the project and also address looming social, ethical, privacy concerns common in the industry sector.

### **7.4 Future Work**

The is still so much to be done. In the future, the researcher will collect more data to help reduce the error rate of the current model and to make it generalize much better on unseen instances. This will make model to scale horizontally enabling a significant number of people to access it as a service. Consequently, this amount of data is likely to result in big data context problem. Working on such datasets will require experimentation with more parallel enabled models and resource computing platforms. The researcher is also intending to automate the generation of test image process through by use of mobile devices to which can transmit that data to a remote server hosting the built model. This will make it possible for a huge number of people to gain

access to the model as a service (MaaS). The patient will have the ability to initiate the monitoring process even without having any technical expertise. Moreover, this will make it easier to extend the model's ability to allow people to screen themselves at home and results relayed remotely. The relevant medical centers will receive the input from user and carry out further analysis and possibly relay back the result to the user via same channel.



## References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others (2016). Tensorflow: A system for large-scale machine learning. In *OsdI* (Vol. 16, pp. 265–283).
- Akiyama, Y., Iwaya, T., Shioi, Y., Endo, F., Ishida, K., Kashiwaba, M., ... Sasaki, A. (2015, 12). Successfully treated advanced esophageal cancer with left axillary lymph node metastasis and synchronous right breast cancer: a case report. , 1.
- Bradski, G., & Kaehler, A. (2000). Opencv. *Dr. Dobb's journal of software tools*, 3.
- CELLSEARCH System Overview CellSearch*. (2018, Apr). Retrieved from <https://www.cellsearchctc.com/product-systems-overview/cellsearch-system-overview> ([Online; accessed 30. Apr. 2018])
- Chausovsky, G., Luchansky, M., Figer, A., Shapira, J., Gottfried, M., Novis, B., ... Klein, A. (1999). Expression of cytokeratin 20 in the blood of patients with disseminated carcinoma of the pancreas, colon, stomach, and lung. *Cancer*, 86(11), 2398–2405.
- Chollet, F., et al. (2015). *Keras*.
- Ciurte, A., Marita, T., & Buiga, R. (2015). Circulating tumor cells classification and characterization in dark field microscopic images of unstained blood. In *Intelligent computer communication and processing (iccp), 2015 ieee international conference on* (pp. 367–374).
- Computed Tomography (CT)*. (2010, Apr). Retrieved from <https://radiology.ucsf.edu/patient-care/services/ct#accordion-how-is-ct-different-from-a-regular-xray> ([Online; accessed 29. Apr. 2018])
- Crowley, E., Di Nicolantonio, F., Loupakis, F., & Bardelli, A. (2013). Liquid biopsy: monitoring cancer-genetics in the blood. *Nature reviews Clinical oncology*, 10(8), 472–484.
- Cui, Y., Zhang, X.-P., Sun, Y.-S., Tang, L., & Shen, L. (2008). Apparent diffusion coefficient: potential imaging biomarker for prediction and early detection of response to chemotherapy in hepatic metastases. *Radiology*, 248(3), 894–900.
- Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., & Fei-Fei, L. (2012). Imagenet large scale visual recognition competition 2012 (ilsvrc2012). *Google Scholar*.
- Deshpande, A. (2016). *A beginner's guide to understanding convolutional neural networks part 2 – adit deshpande – cs undergrad at ucla ('19)*. <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>. ((Accessed on 09/13/2017))
- Diagnosis: CT Scan*. (2007, Dec). Retrieved from <http://pathology.jhu.edu/pc/slides/ctscan.html> ([Online; accessed 29. Apr. 2018])

- Dickinson, R., Hall, S., Sinclair, J. E., Bond, C., & Murchie, P. (2014). Using technology to deliver cancer follow-up: a systematic review. *BMC cancer*, *14*(1), 311.
- Diehl, F., Schmidt, K., Choti, M. A., Romans, K., Goodman, S., Li, M., ... others (2008). Circulating mutant dna to assess tumor dynamics. *Nature medicine*, *14*(9), 985–990.
- Di Meo, A., Bartlett, J., Cheng, Y., Pasic, M. D., & Yousef, G. M. (2017). Liquid biopsy: a step forward towards precision medicine in urologic malignancies. *Molecular cancer*, *16*(1), 80.
- Egan, T. K. (2000). Monitoring patients undergoing cancer therapy. *Laboratory Medicine*, *31*(12), 666–671.
- Ekeland, A. G., Bowes, A., & Flottorp, S. (2010). Effectiveness of telemedicine: a systematic review of reviews. *International journal of medical informatics*, *79*(11), 736–771.
- Ferlay, J., Héry, C., Autier, P., & Sankaranarayanan, R. (2010). Global burden of breast cancer. In *Breast cancer epidemiology* (pp. 1–19). Springer.
- Fitzmaurice, C., Dicker, D., Pain, A., Hamavid, H., Moradi-Lakeh, M., MacIntyre, M. F., ... others (2015). The global burden of cancer 2013. *JAMA oncology*, *1*(4), 505–527.
- Gonzalez, R. C., & Woods, R. E. (2002). Thresholding. *Digital Image Processing*, 595–611.
- Graham, L. J., Shupe, M. P., Schneble, E. J., Flynt, F. L., Clemenshaw, M. N., Kirkpatrick, A. D., ... others (2014). Current approaches and challenges in monitoring treatment responses in breast cancer. *Journal of Cancer*, *5*(1), 58.
- Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., ... Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. *Medical image analysis*, *35*, 18–31.
- Karn, U. (2016, 11). *An intuitive explanation of convolutional neural networks*. <http://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/2>. ((Accessed on 09/14/2017))
- Kearney, N., McCann, L., Norrie, J., Taylor, L., Gray, P., McGee-Lennon, M., ... Maguire, R. (2009). Evaluation of a mobile phone-based, advanced symptom management system (asym©) in the management of chemotherapy-related toxicity. *Supportive Care in Cancer*, *17*(4), 437–444.
- Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *Icml deep learning workshop* (Vol. 2).
- Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, *13*, 8–17.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep con-

- volutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Liquid Biopsy Predicts Colon Cancer Recurrence*. (2016, Jul). Retrieved from <https://www.genengnews.com/gen-news-highlights/liquid-biopsy-predicts-colon-cancer-recurrence/81252922> ([Online; accessed 29. Apr. 2018])
- Maguire, R., Miller, M., McCann, L. A., Taylor, L., Kearney, N., Sage, M., & Norrie, J. (2008). Managing chemotherapy symptoms via mobile phones. *Nursing Times*, 104(22), 28–29.
- Mao, Y., Yin, Z., & Schober, J. (2016). A deep convolutional neural network trained on representative samples for circulating tumor cell detection. In *Applications of computer vision (wacv), 2016 ieee winter conference on* (pp. 1–6).
- Ministry of Health and Sanitation. (2011). *National Cancer Control Strategy* [report].
- NCI Dictionary of Cancer Terms*. (2018, Apr). Retrieved from <https://www.cancer.gov/publications/dictionaries/cancer-terms/def/liquid-biopsy> ([Online; accessed 29. Apr. 2018])
- Nelson, N. J. (2010). *Circulating tumor cells: will they be clinically useful?* Oxford University Press.
- Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press USA.
- Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345–1359.
- Patel, A. S., Allen, J. E., Dicker, D. T., Peters, K. L., Sheehan, J. M., Glantz, M. J., & El-Deiry, W. S. (2011). Identification and enumeration of circulating tumor cells in the cerebrospinal fluid of breast cancer patients with central nervous system metastases. *Oncotarget*, 2(10), 752.
- Patel, U., Patel, A., Cobb, C., Benkers, T., & Vermeulen, S. (2014). The management of brain necrosis as a result of srs treatment for intra-cranial tumors. *Translational Cancer Research*, 3(4), 373–382.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.
- Pérez-Barrios, C., Nieto-Alcolado, I., Torrente, M., Jiménez-Sánchez, C., Calvo, V., Gutierrez-Sanz, L., ... Romero, A. (2016). Comparison of methods for circulating cell-free dna isolation using blood from cancer patients: impact on biomarker testing. *Translational lung cancer research*, 5(6), 665.
- PET/CT Scan*. (2010, Apr). Retrieved from <https://radiology.ucsf.edu/patient-care/>



- services/pet-ct ([Online; accessed 29. Apr. 2018])
- Prepare for Magnetic Resonance Imaging (MRI). (2010, Apr). Retrieved from <https://radiology.ucsf.edu/patient-care/prepare/mri> ([Online; accessed 29. Apr. 2018])
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).
- Røe, K., Kakar, M., Seierstad, T., Ree, A. H., & Olsen, D. R. (2011). Early prediction of response to radiotherapy and androgen-deprivation therapy in prostate cancer by repeated functional mri: a preclinical study. *Radiation oncology*, 6(1), 65.
- Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3859–3869).
- Schneider, M. J., Cyran, C. C., Nikolaou, K., Hirner, H., Reiser, M. F., & Dietrich, O. (2014). Monitoring early response to anti-angiogenic therapy: diffusion-weighted magnetic resonance imaging and volume measurements in colon carcinoma xenografts. *PloS one*, 9(9), e106970.
- Scholtens, T. M., Schreuder, F., Ligthart, S. T., Swennenhuis, J. F., Greve, J., & Terstappen, L. W. (2012). Automated identification of circulating tumor cells by image cytometry. *Cytometry Part A*, 81(2), 138–148.
- Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1), 1929–1958.
- Svensson, C.-M., Krusekopf, S., Lücke, J., & Thilo Figge, M. (2014). Automated detection of circulating tumor cells with naive bayesian classifiers. *Cytometry Part A*, 85(6), 501–511.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818–2826).
- Theilmann, R. J., Borders, R., Trouard, T. P., Xia, G., Outwater, E., Ranger-Moore, J., ... Stopeck, A. (2004). Changes in water mobility measured by diffusion mri predict response of metastatic breast cancer to chemotherapy. *Neoplasia*, 6(6), 831–837.
- Thoeny, H. C., & Ross, B. D. (2010). Predicting and monitoring cancer treatment response with diffusion-weighted mri. *Journal of Magnetic Resonance Imaging*, 32(1), 2–16.

- Torre, L. A., Bray, F., Siegel, R. L., Ferlay, J., Lortet-Tieulent, J., & Jemal, A. (2015). Global cancer statistics, 2012. *CA: a cancer journal for clinicians*, 65(2), 87–108.
- Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22–30.
- Wang, D., Khosla, A., Gargeya, R., Irshad, H., & Beck, A. H. (2016). Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*.
- World Health Organization. (1979). Who handbook for reporting results of cancer treatment.
- Wybranski, C., Zeile, M., Löwenthal, D., Fischbach, F., Pech, M., Röhl, F.-W., ... Dudeck, O. (2011). Value of diffusion weighted mr imaging as an early surrogate parameter for evaluation of tumor response to high-dose-rate brachytherapy of colorectal liver metastases. *Radiation Oncology*, 6(1), 43.



# Appendix



## Appendix C: Scripts

### Cropping Algorithm

Developed cropping algorithm. The function accepts three parameters. The  $x$  coordinate and  $y$  coordinate of CTC locations crops a positive  $20 * 20$  location in the contrast image and also generate a negative sample which is not overlapping with previous sample on the same image instance.

```
def generate_negpos_sample(x_center,y_center,positive_image):
    y_cooridnates = [y_center]
    x_cooridnates = [x_center]
    index = 1
    while index<21:
        point_y_up = y_center+index
        point_y_down = y_center-index
        y_cooridnates.append(point_y_up)
        y_cooridnates.append(point_y_down)
        point_x_left = x_center-index
        point_x_right = x_center+index
        x_cooridnates.append(point_x_left)
        x_cooridnates.append(point_x_right)
        index+=1
    set_positive_y_cooridnates = set(y_cooridnates)
    set_positive_x_cooridnates = set(x_cooridnates)
    terminate = False
    while not terminate:
        ax_coordinate = random.choice(list(range(1,1601-21)))
        ay_coordinate = random.choice(list(range(1,1201-21)))
        if len(set_positive_x_cooridnates.intersection(\
            set([ax_coordinate])))>0 or \
            len(set_positive_y_cooridnates.intersection(\
            set([ay_coordinate])))>0:
            continue
        else:
            terminate = True
    return positive_image[y_center-20:y_center+20:\
        ,x_center-20:x_center+20],\
        positive_image[ay_coordinate-20:ay_coordinate+20:\
        ,ax_coordinate-20:ax_coordinate+20]
```

Listing 10: Procedure to crop positive and negative samples

## Generation of Positive and Negative Samples

The function is called to generate positive and negative sample using the the cropping algorithm presented in the previous section. Positive and negative samples are are then stacked on top of each other forming a 3 dimensional matrix;  $x, y, z$  space where  $x$  is the batch dimension,  $y$  image width and  $z$  image height. The final matrix is then persistently stored on the disk and re-used in all other model experiments.

```
def crop_coordinates():
    pos = np.empty((0,40,40),dtype=float)
    neg = np.empty((0,40,40),dtype=float)
    store_xy = defaultdict(list)
    all_coordinates = get_all_coordinates()
    for img,xy in all_coordinates.items():
        for loc in list(xy[-1]):
            ctc_pos,ctc_neg = generate_negpos_sample(int(loc[1]),
                                                    int(loc[0]),
                                                    read_image_bw(xy[0]))
            if ctc_pos.shape ==(40,40) and ctc_neg.shape ==(40,40):
                afeature_pos = np.expand_dims(ctc_pos,axis=0)
                afeature_neg = np.expand_dims(ctc_neg,axis=0)
                pos = np.append(pos,afeature_pos,axis=0)
                neg = np.append(neg,afeature_neg,axis=0)
    return pos,neg
```

Listing 11: Code to generate positive and negative samples Code

## Helper Functions

Helper procedures were created to load, threshold and pre process image.

```
from __future__ import print_function,division
import numpy as np
import argparse
import cv2
import random
import sys
import os
from collections import defaultdict
import pickle
import keras
seed = 7
np.random.seed(seed)
CTC = os.listdir('CTC/')
HOME = "CTC/"
# load all images
def imagectcs():
    imgs = []
    for aimage in CTC:
        if not aimage.endswith('Red.jpg'):
            img = aimage.split('.')[0]
            img = img.split('_')[0].split('-')[-1]
            for aimg in CTC:
                if aimg.endswith('Red.jpg'):
                    imglabel = aimg.split(' ')
                    imglabel = imglabel[1].split('_')[0]
                    imglabel = imglabel.split('-')[1]
                    if imglabel == img:
                        imgs.append((aimage,aimg))
    return imgs
```

Listing 12: Helper Functions

```

def get_image_id(aimg):
    imglabel = aimg.split(' ')
    imglabel = imglabel[1].split('_')[0]
    return imglabel.split('-')[1]

def read_image(name=None):
    image = cv2.imread(HOME+name)
    grey_scaled = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, threshold_image = cv2.threshold(grey_scaled,240,255,
                                       cv2.THRESH_BINARY)

    return threshold_image

def read_image_bw(name=None):
    image = cv2.imread(HOME+name)
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

def get_ctc(ctcimg):
    coordinates = np.argwhere(ctcimg==255)
    if len(list(coordinates))>=1:
        return coordinates
    return None

def get_sample_image():
    index = random_image()
    return index,read_image(all_images[index][1])

def get_all_coordinates():
    store_xy = defaultdict(list)
    for img in all_images:
        ctc_location = get_ctc(read_image(img[1]))
        if ctc_location is None:
            continue
        store_xy[img[1]].append(img[0])
        store_xy[img[1]].append(ctc_location)
    return store_xy

```

Listing 13: Helper Functions

## Multi Layer Perceptron Model

```
from __future__ import print_function,division
import numpy as np
import keras
from keras.datasets import mnist
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
from keras import backend as K
import matplotlib.pyplot as plt
seed = 7
np.random.seed(seed)
outfile = "train_set_final.npz"
list_files = np.load(outfile)
X_train, X_test, y_train, y_test = list_files['X_train'], \
                                   list_files['X_test'], \
                                   list_files['y_train'], \
                                   list_files['y_test'] \
X_train_mlp = X_train.reshape(X_train.shape[0], \
                               NUM_PIXELS).astype('float32')/255
X_test_mlp = X_test.reshape(X_test.shape[0], \
                             NUM_PIXELS).astype('float32')/255

def confusion_matrix(actual,predicted):
    tn,fp,fn,tp = 0,0,0,0
    for a,p in zip(actual,predicted):
        if np.argmax(a) == 1:
            if np.argmax(a) == np.argmax(p):
                tp+=1 # truly positive actual-predicted
            else:
                fp+=1
        else:
            if np.argmax(a) == np.argmax(p):
                tn+=1
            else:
                fn+=1 #falsely negative
    return tn,fp,fn,tp
```



```

def mlp_model(num_pixels,num_classes):
    model = Sequential()
    model.add(Dense(num_pixels, input_dim=num_pixels,\\
kernel_initializer='normal', activation='relu'))
    model.add(Dense(num_pixels, input_dim=num_pixels, \\
kernel_initializer='normal', activation='relu'))
    model.add(Dense(num_classes, kernel_initializer='normal', \\
activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='sgd',
metrics=['accuracy'])
    return model

```

```

model = mlp_model(NUM_PIXELS,NUM_CLASSES)
model.fit(X_train_mlp, y_train,epochs=500, batch_size=200, verbose=2)
scores = model.evaluate(X_test_mlp, y_test, verbose=2)
print("Baseline Error: %.2f%%" % (100-scores[1]*100))
tn,fp,fn,tp = confusion_matrix(y_test,model.predict(X_test_mlp))
precision = tp/(tp+fp)
recall = tp/(tp+tn)
accuracy = (tp+tn)/(tn+fp+fn+tp)
print(accuracy*100,precision*100,recall*100)

```



## Simple Models (SVM, NB Decision Trees)

```
from __future__ import print_function,division
import numpy as np
from keras.datasets import mnist
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
from keras import backend as K
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC,SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from keras.utils.np_utils import to_categorical
from sklearn.metrics import classification_report

np.random.seed(seed)
outfile = "train_set_final.npz"
list_files = np.load(outfile)
X_train, X_test, y_train, y_test = list_files['X_train'],
                                   list_files['X_test'],
                                   list_files['y_train'],
                                   list_files['y_test']
NUM_PIXELS = X_train.shape[1]*X_train.shape[2]
y_train = np.argmax(y_train,axis=1)
y_test = np.argmax(y_test,axis=1)
X_train_sm = X_train.reshape(X_train.shape[0],
                             NUM_PIXELS).
                             astype('float32')/255
X_test_sm = X_test.reshape(X_test.shape[0],
                           NUM_PIXELS).
                           astype('float32')/25

//Linear SVM
svm = LinearSVC().fit(X_train_sm,y_train)
accuracy_score(svm.predict(X_test_sm),y_test)
print(classification_report(y_test,svm.predict(X_test_sm)))
// Non Linear SVM
svm_non_linear = SVC(kernel='rbf').fit(X_train_sm,y_train)
accuracy_score(svm_non_linear.predict(X_test_sm),y_test)
print(classification_report(y_test,svm_non_linear.predict(X_test_sm)))

//random forest
rf = RandomForestClassifier().fit(X_train_sm,y_train)
accuracy_score(rf.predict(X_test_sm),y_test)
print(classification_report(y_test,rf.predict(X_test_sm)))
```

## ConvNet Classification Model

```
import numpy as np
import cv2
import random
import sys
import os
import keras
from keras.datasets import mnist
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
from sklearn.model_selection import train_test_split

outfile = "train_set_final.npz"
list_files = np.load(outfile)
X_train, X_test, y_train, y_test = list_files['X_train'], \
    list_files['X_test'], \
    list_files['y_train'], \
    list_files['y_test']

NUM_CLASSES = y_test.shape[1]
NUM_PIXELS = X_train.shape[1]*X_train.shape[2]
X_train_cnn = X_train.reshape(X_train.shape[0],40, 40, 1).astype('float32')/255
X_test_cnn = X_test.reshape(X_test.shape[0],40, 40,1).astype('float32')/255
def confusion_matrix(actual,predicted):
    tn,fp,fn,tp = 0,0,0,0
    for a,p in zip(actual,predicted):
        if np.argmax(a) == 1:
            if np.argmax(a) == np.argmax(p):
                tp+=1 # truly positive actual-predicted
            else:
                fp+=1
        else:
            if np.argmax(a) == np.argmax(p):
                tn+=1
            else:
                fn+=1 #falsely negative
    return tn,fp,fn,tp
```

```

def cnn_model(input_shape=(40,40,1)):
    model = Sequential()
    model.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1),
    activation='relu',
    input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(12, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(588, activation='relu'))
    model.add(Dense(NUM_CLASSES, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',\
    metrics=['accuracy'])
    return model

```

```

model = cnn_model()
model.fit(X_train_cnn, y_train, epochs=500, batch_size=200, verbose=2)
scores = model.evaluate(X_test_cnn, y_test, verbose=2)
print("Test Error: %.2f%%" % (100-scores[1]*100))
predictions = model.predict(X_test_cnn)
predictionscnn = model.predict(X_test_cnn)
tn,fp,fn,tp = confusion_matrix(y_test,predictionscnn)
precision = tp/(tp+fp)
recall = tp/(tp+tn)
accuracy = (tp+tn)/(tn+fp+fn+tp)
f1 = (precision*recall)/(precision+recall)
print(precision,recall,accuracy,2*f1)

```

