الجمهورية الجزائرية الديمقراطية الشعبية

**MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH**

**YAHIA FARES UNIVERSITY OF MEDEA**
**FACULTY OF SCIENCE**

**DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE**



# Graduation Project

**Submitted for graduation in Computer science master**
**Specialty: Systems Engineering and Web Technology**

# TITLE

## MALARIA DETECTION IN HUMAN CELLS USING DEEP LEARNING

**Proposed and Directed by:**
**Mr. YAZID BOUNAB**
**Mr. ALI GAGUI**
**Mr. AHMED CHERIF MAZARI**

**Prepared and Presented by:**
**Mr. Benaferi Mourad**
**Mr. Benabderrahmane Youcef**

Academicyear2019-2020

## Abstract

This master's report presents an automatic identification of Malaria infected cells in human blood samples using deep learning methods. We used a dataset of 27,558 images divided equally into infected and uninfected cells. Various machine learning and deep learning techniques used to solve this task. However, practical performance has not proven sufficient yet. In this respect, we built a system that can discriminate the infected cells from the uninfected ones. We proposed three different deep learning architectures, convolutional neural network (CNN), AutoEncoder and transfer learning using pretrained model (VGG16). The best result went to the CNN model which showed superiority over the AutoEncoder and transfer learning model (VGG16) in all performance indicators such as sensitivity (99.78%), specificity (99.95%) precision (99.87%) and F1 score (99,87%).

**Key words:**
Deep learning, CNN, AutoEncoder, VGG16.

**الملخص**
تعرض هته المذكرة الكشف عن إصابة الخلايا بالملاريا في عينات دم الإنسان بالستعمال طرق التعلم العميق. استخدمنا مجموعة بينات من 27.558 صورة مقسمة بالتساوي الى خلايا مصابة و غير مصابة. مختلف تقنيّات التعلم الآلي و التعلم العميق إستخدمت لحل هته المهمة. مع ذلك, لم يثبت الأداء العملي جدارته بعد. و في هذا الصدد قمنا ببناء نظام يمكنه التمييز بين الخلايا المصابة و الغير مصابة. اقترحنا ثلاث بنى مختلفة للتعلم العميق. الشبكة العصبية التلافيفيةِ CNN, التشفير التلقائي AutoEncoder و نقل التعلم باستخدام نموذج مسبق التدريب VGG16. افضل نتيجة كانت لنموذج الشبكة العصبية التلافيفية CNN, الذي أظهر تفوقا على نموذج التشفير التلقائي و نقل التعلم في جميع مؤشرات الأداء متمثلة في الحساسية بقيمة 99.78 , النوعية ب 99.95 , الدقة ب 99.87 و الب .99.87

**الكلمات المفتاحية :**
التعلم العميق, VGG16, AutoEncoder, CNN.

## Résumé

Ce rapport de master présente une automatique identification de la maladie de Malaria des cellules infectées dans des échantillons de sang humain à l'aide de méthodes d'apprentissage profondeur. Nous avons utilisé une base de données de 27,558 images divisées également en cellules infectées et non infectées. Divers techniques de l'apprentissage automatique et l'apprentissage en profondeur utilisées pour résoudre ce problème. Cependant, les performances pratiques ne se sont pas encore avérées suffisantes. A cet égard, nous avons construit un système qui peut distinguer les cellules infectées des non infectées. Nous avons proposé trois différentes architectures d'apprentissage en profondeur, le réseau neuronal convolutif (CNN), l'AutoEncoder et l'apprentissage par transfert en utilisant un modèle Pré-entraîné (VGG16). Le meilleur résultat est allé au modèle CNN qui à montrer une supériorité sur l'Autoencoder et le modèle pré-entrainé VGG16 dans tous les indicateurs de performance tels que, la sensibilité (99.78%), la spécificité (99,95%), la précision (99.87%) et le F1-Score (99,87%).



**Les mots clés :**

Apprentissage automatique, CNN, AutoEncoder, VGG16.

# ACKNOWLEDGEMENTS

First and foremost, praises and thanks to god, the almighty for his showers of blessing throughout our research work to complete the research successfully.

we would like to express our deep and sincere gratitude and say a special thanks to our supervisors **Mr. Yazid Bounab, Mr ALI Gagui, Mr AHMED CHERIF MAZARI** for their consistent support, overall insight and guidance during the running of this project and acknowledging that it was a great privilege and honor to work and study under their guidance.

we also extend our sincere thanks to the members of discussion committee for their time and insightful comments.

Without forgetting to thanks and extend our full gratitude to the friend and brother **Mounir Messadi** from the bottom of our heart for his help and big constant support, as he did not hesitate to help us in any way he could do.

we also things all peoples who didn't hesitate to help us weather with benefit information and ideas or with constant encouragement and planting a spirit of optimist and perseverance in us.

# Dedication

This thesis is dedicated to;

My father “Ahmed” who enlightens me the path of success.

My mother “Nadia”, my life and happiness who supported me in my studies and encouraged me to release my dreams.

My brothers “Ghanou” and “Tarak” the fruits of my life.

My sisters “Loubna” and “Marwa” the flowers of the world around me.

My classmate “Sarah” who encouraged me and supported me.

My partner “Youcef” with who i shared the sadness and the happiness.

My dear friends “Menouar” and “Rahim” that I love.

My cousin “Rafik” childhood friend and brother.

To all my friends, my family and everyone who knows me and helped me in this thesis.

Mourad.

# Dedication

This thesis is dedicated to;

My father “Yagoub” who helped me throughout my academic career to success.

My mother “Houria”, who gave me the encouragement and made my life much easier.

My brother “Ghanou” who helped me to improve myself, and “Aissa” who supported me morally.

My sister “Sabah” who I love a lot.

My close friend “Safa” who gave me moral support.

My partner “Mourad”, my brother and my best friend who shared the good and bad with me in my journey.

To all my friends, my family and everyone who knows me and helped me in this thesis.

Youcef.

# Table of Contents

**General Introduction** ..... 1

**Chapter 1: images and malaria detection in human cells** ..... 3

**Introduction** ..... 4

**1-Images** ..... 4

**1.1-Basic image definition:** ..... 4

**2.Digital image definition :** ..... 5

**2.1. Different types of digital images:** ..... 5

**3.pixels** ..... 5

**3.1 The binary image** ..... 6

**3.2 Grayscale image :** ..... 6

**3.3 Color image:** ..... 7

**4. Image type :** ..... 7

**5. Image resolution:** ..... 9

**5.1 Spatial Resolution:** ..... 9

**5.2 Gray level Resolution:** ..... 9

**6. Image Segmentation** ..... 10

**6.1 Definition:** ..... 10

**6.2 Filter:** ..... 11

**6.3 smoothing images:** ..... 11

**6.4. Image noise:** ..... 12

**6.5. convolution:** ..... 12

**6.6. The contour:** ..... 13

**7. medical imaging:** ..... 14

**Introduction:** ..... 14

**7.1. Medical imaging technologies:** ..... 14

**7.2. Medical image segmentation:** ..... 16

**8.Computer vision** ..... 16

**9. Malaria images:** ..... 16

**9.1. Malaria detection:** ..... 19

**Conclusion:** ..... 20

**Chapter 2: State of the art of Malaria detection using deep learning.** ..... 21

**Introduction** ..... 22

**1.Machine learning** ..... 22

**2.Deep learning** ..... 22

**3.Difference between deep learning and machine learning:** ..... 23

**4.Deep learning techniques:** 23
**4.1. Artificial Neural Network(ANN):** 23
**4.2. Deep Neural Networks(DNN):** 24
**4.3. Deep Belief Networks:** 24
**4.4. Recurrent Neural Networks:** 24
**4.5. Long Short-Term Memory (LSTM):** 25
**4.6. AutoEncoder** 26
**4.7. Convolutional Neural Networks (CNN)** 26
**5. Network Architectures** 30
**5.1. VGG:** 31
**5.2. ResNet:** 33
**5.3. AlexNet:** 33
**5.4. GoogleNEt Architecture:** 34
**6. Data Processing:** 34
**6.1 Data pre-processing:** 34
**6.2. Normalization:** 35
**6.3. Data augmentation:** 35
**7. Feature extraction:** 36
**8. Model Evaluation:** 36
**9. Application of deep learning:** 37
**10. Previous Studies:** 38
**Conclusion** 41
**Chapter 3: proposed approach base on DL for Malaria Detection.** 42
**Introduction:** 43
**1.Methodology:** 43
**2. System architecture:** 45
**3.Proposed approaches:** 45
**3.1 Convolutional Neural Network CNN:** 45
**3.2 VGG16 model:** 49
**3.3 AutoEncoder model:** 55
**Conclusion:** 58
**Chapter 4: implementation and evaluation** 59
**Introduction:** 60
**1.Dataset:** 60
**2.Environment:** 61
**2.1 Hardware used** 62

**3.Data preprocessing:** ..... 62
**4. Data augmentation:** ..... 63
**5. Splitting the data:** ..... 63
**6. Train, validation and test:** ..... 64
**7. Over-fitting and under-fitting:** ..... 64
**8. Network Architecture:** ..... 64
**8.1 VGG-16 implementation:** ..... 64
**8.2 CNN implementation:** ..... 71
**8.3 AutoEncoder implementation:** ..... 75
**9. Discussion:** ..... 79
**10. Application:** ..... 81
**Conclusion:** ..... 82
**General Conclusion** ..... 83

**Figure List:**

Figure 1. 1 Normal Image(Visuals, 2019)..... 4
Figure 1. 2 Binary Image(Alofi Kanter, 2015)..... 6
Figure 1. 3 a grayscale image(W.carter, 2017) ..... 7
Figure 1. 4 color image(Padmavathi & Thangadurai, 2016)..... 7
Figure 1. 5 image resolution (David Coleman, 2020)..... 9
Figure 1. 6 gray-level resolution(Talley J Lambert, 2014) ..... 10
Figure 1. 7 Mean filter application(Cornelisse, 2018) ..... 11
Figure 1. 8 median filter (Mark Nixon & Alberto Aguado, 2002)..... 12
Figure 1. 9 different contours types ..... 13
Figure 1. 10 Typical IVUS image(Chen, 2014). ..... 15
Figure 1. 11 Soft X-ray tomography and Iron x-ray fluorescence map ..... 16
Figure 1. 12 A-blood infection (Kim et al., 2010) B- splenic rupture(Kim et al., 2010) ..... 17
Figure 1. 13 Five different human malaria Plasmodium species and their life stages in thin blood film (Source: K. Silamut and CDC). ..... 18
Figure 1. 14 malaria Blood smears Image sources: Microscopist, Wendy Saravagy. Illustrations of blood smears. Microscopist examining blood smears. ..... 19

Figure 2. 1 deep learning basics(Parashu Ram Pal, 2019) ..... 22
Figure 2. 2 the difference between deep learning and machine learning(Hammami, 2019). .. 23
Figure 2. 3 schematic diagram of an RNN (Murad & Pyun, 2017) ..... 25
Figure 2. 4 LSTM architecture (Yan, 2016)..... 25
Figure 2. 5 AutoEncoder architecture (Matthew Stewart & Stewart, 2019)..... 26
Figure 2. 6 typical CNN architecture (Camacho, 2018)..... 26
Figure 2. 7 Types of pooling (Tammina, 2019) ..... 27
Figure 2. 8 fully connected layer (Tammina, 2019)..... 28
Figure 2. 9 Dropout example (Budhiraja, 2016) ..... 28
Figure 2. 10 Batch Normalization(I. R, 2018) ..... 29
Figure 2. 11 sigmoid function(Sai Bharadwaj Reddy & Sujitha Juliet, 2019)..... 29
Figure 2. 12 softmax application..... 30
Figure 2. 13 VGG16 and the VGG19 architectures. ..... 33
Figure 2. 14 ResNet architecture. ..... 33
Figure 2. 15 AlexNet Architecture..... 33
Figure 2. 16 GoogleNet architecture. ..... 34
Figure 2. 17 data preprocessing step in deep learning ..... 34
Figure 2. 18 biological data normalization. ..... 35
Figure 2. 19 data augmentation basics(Avatar & B, 2020). ..... 36
Figure 2. 20 deep learning application. ..... 37
Figure 2. 21 West Africa shows cases of malaria ("world malaria report," 2016). ..... 38

Figure 3. 1 steps of our work..... 44
Figure 3. 2 The summary of our architecture and the total parameters and the trainable parameters from them..... 48
Figure 3. 3 the architecture of our proposed Convolutional Neural network model. ..... 49
Figure 3. 4 The VGG16 network summary without Fully Connected layer. ..... 50
Figure 3. 5 the proposed vgg16 architecture summary. ..... 52

Figure 3. 6  the frozen and trainable layers in our proposed model. ..... 53
Figure 3. 7 our VGG16 proposed model. ..... 54
Figure 3. 8 the summary of our architecture and the total parameters and the trainable parameters from them. ..... 57
Figure 3. 9 our proposed AutoEncoder architecture. ..... 58

Figure 4. 1 Samples drawn from NIH (National Library of Medicine) Malaria dataset which are malaria infected parasite red blood cells. ..... 60
Figure 4. 2 Samples drawn from NIH (National Library of Medicine) Malaria dataset which are malaria uninfected red blood cells. ..... 61
Figure 4. 3 samples from the processing data. ..... 62
Figure 4. 4 example of data augmentation in training set ..... 63
Figure 4. 5 the two losses and accuracies of the second case with data augmentation. ..... 65
Figure 4. 6 the training and validation accuracy. ..... 66
Figure 4. 7 the training and validation loss. ..... 66
Figure 4. 8 The model starts training and logs the losses and accuracies ..... 68
Figure 4. 9 the training and validation accuracy. ..... 68
Figure 4. 10 the training and validation loss. ..... 69
Figure 4. 11 The model starts training and capture the losses and accuracies. ..... 72
Figure 4. 12 the training and validation accuracy. ..... 73
Figure 4. 13 The training and validation loss. ..... 73
Figure 4. 14 The model start training and capture the two losses (both loss and val_loss) and the tow accuracy (acc and val_acc). ..... 76
Figure 4. 15 the training and validation accuracy. ..... 77
Figure 4. 16 the training and validation loss. ..... 77
Figure 4. 17 the representation of results. ..... 80
Figure 4. 18 comparing performance ..... 81
Figure 4. 19 test application. ..... 82
Figure 4. 20 execution result. ..... 82

**Table List:**

Table 1. 1 description and uses of different Image Types (Singh, 2019)..... 8

Table 2. 1 testing results of(Dong et al., 2017)..... 39
Table 2. 2 Sensitivity, Specificity, and MCC: A) SVM on Features and CNN on B) Patches C) Focus Stack..... 39
Table 2. 3 the results of the two ten-fold cross-validations(Liang et al., 2016)..... 40
Table 2. 4 model performance..... 40
Table 2. 5 Performance evaluation of VGG16-SVM & VGG19-SVM..... 41
Table 2. 6 4-fold cross-validation results for ANN classifier. ..... 41
Table 2. 7 The performance of the five classifiers. ..... 41

Table 4. 1 organization of training, testing and validation in the VGG16, CNN, AutoEncoder proposed models. ..... 63
Table 4. 2 the training and validation accuracy with an input size (125x125x3) with and without using data augmentation..... 69
Table 4. 3 distribution of images in test dataset. ..... 70
Table 4. 4 performance of the tested model ..... 70
Table 4. 5 the confusion matrix of tested model. ..... 70
Table 4. 6 the confusion matrix of the tested model without using data augmentation. ..... 71
Table 4. 7 the sensitivity and specificity of the tested model in the two cases. ..... 71
Table 4. 8 the training and validation accuracy..... 74
Table 4. 9 distribution of images in test dataset. ..... 74
Table 4. 10 performance of the tested model ..... 74
Table 4. 11 the confusion matrix of tested model. ..... 75
Table 4. 12 the sensitivity and specificity of the tested model ..... 75
Table 4. 13 the training and validation accuracy..... 78
Table 4. 14 distribution of images in test dataset. ..... 78
Table 4. 15 performance of the tested model. ..... 78
Table 4. 16 the confusion matrix of tested model. ..... 79
Table 4. 17 the sensitivity and specificity of our model. ..... 79
Table 4. 18 the results of the three proposed model..... 79
Table 4. 19 comparing results. ..... 80

# General Introduction

The last decade world has seen tremendous technological development in many disciplines. Artificial intelligence has proven its effectiveness that it will be dominant in all other domains such as biomedical engineering, where the biomedical engineering field tends to design and develop biomedical prototypes as medical tools that help for giving a distinctive diagnostic. The detection of any disease in its early stage is very crucial to supply the right treatment as results save human lives. Hence, doctors and physicians indeed need such devices.

Malaria as one of the lethal diseases that the world still facing until now. Each year, Malaria takes a truly enormous number of lives in third world countries.

According to the report released by the World Health Organization (WHO), in 2018 there were 228 million infections with this disease, 405,000 from them were death (WHO, 2020). Usually, the detection of such disease is human-based detection which consists of a set of putting a blood sample of human under the lances of a microscope to check it whether it has an infection by Malaria virus or not. However, this classical method is very costly in terms of time and human resource. Malaria virus has many types such as oval, vivax, falciparum, malarie and anopheles. It is worth to mention that this process is not only limited by filtering out the infected samples, it also identifies the instance of Malaria types within blood sample. By knowing the exact type of virus doctors will provide the patient with the right vaccine. Hence, Malaria detection should be directed to automation, that can facilitate the diagnosis and treatment of the case. Artificial Intelligence (AI) is a technique of automation that can be used for detecting Malaria disease with a low-cost, and effectiveness. Thus, Malaria detection could benefit from automation using deep learning.

To automate the malaria diagnosis with Artificial Intelligence, the system requires a dataset which contains a set of images with certain resolution. These images are fully composed of pixels arranged in rows and columns. Where these images are an ocular or other representation of a real object, a graphic or a picture that contains all information needed to detect the case.

Malaria images belong to the medical field, there are many technologies in medical imaging used to identify malaria infections such as X-ray images, Magnetic Resonance Imaging (MRI) images and CT-Scan images(Vannier & Haller, 1998). the development of these technologies has helped doctors to make easy and less life threatening therapeutic decisions, medical imaging techniques have many advantages, especially the pathological visualization entity itself, allowing a better understanding and orientation of the diagnosis.

In the direction of developing automated Malaria diagnosis, a large number of works have been done using machine learning and deep learning methods which gave an increasing attention for its great capability in building automated diagnosis system for Malaria, these methods operate on different dataset. Support Vector Machine (SVM) and Naïve bayes classifier were utilized in contrast to the supervised learning, unsupervised learning(Devi, Roy, Singha, Sheikh, & Laskar, 2018), for instance, K-nearest Neighbor (KNN) has also been proposed for Malaria infection identification(Malihi, Ansari-Asl, & Behbahani, 2013), Deep Neural Network (DNN) with set of layers was designed to detect Malaria infected cells. almost all of these methods

mentioned has not proven sufficient yet in all performance whether in accuracy, sensitivity, specificity, precision and F1-Score.

The goal of our work is to develop a system for Malaria detection using deep learning to achieve better result than the existing systems. So, we propose a novel approaches based on Convolutional Neural Network (CNN), AutoEncoder and pretrained model Visual Geometry Group (VGG16).

Therefore, our thesis subdivided as follows:

The first chapter presents the fundamental notion of images with the problematic of malaria disease.

First and foremost, it contains the different components of images as well as the techniques that can be used for segmentation in medical imaging.
The second chapter contains the different types of Deep Learning, how they work and their applications and different architectures.

Therefore, Malaria definition and the state of the art of detecting malaria in human cells.

The third chapter presents the fundamental notion of our proposed approaches, their components and functions used in the three models we proposed.

The fourth chapter presents our implementation results of the three proposed approaches.

First and foremost, it is divided into two parts: the first part being concerning the data such as preprocessing, data augmentation and data splitting into sets which are validation, training and test.

While the second part concerning the model as training, compiling and finally testing the model and getting results.

# Chapter 1: images and malaria detection in human cells

# Introduction

Images are an ocular or other representation of a real object, a graphic or a picture that contains all information needed to produce a live working copy (see disk image, executable image and image copy).

Images are one of the many important solutions that man uses for transmuting and delivering knowledge and information since the beginning of humanity, an image alone can bring together an immense amount of information.

A system image processing mainly consists of the following functions: image acquisition, pre-treatment for noise reduction, analyze the image to find the information included in it.

In recent years, image treatment has been developed, different techniques and software provided to process digital photo are proof certain (Quellec et al., 2017)

In this chapter, we are going to present images, basic concepts on images processing citing the different types of images and presenting image segmentation in the use of treatment especially in the medical field, as well as the Malaria detection in human cells, and therefore a number of definition and concepts will be displayed.

# 1-Images

### 1.1-Basic image definition:

An image (from Latin: *imago*) is an artifact that depicts visual perception, such as a photograph or other two-dimensional picture, particularly one that resembles a subject (usually a physical object). In the context of signal processing, an image is a distributed amplitude of color(s).(Chakravorty, 2018)

An image defines a reproduction or exact analogies representation of a real scene (a person or any other object) in a two-dimensional form is called an *image*.

An image is nothing but, an array or a matrix of multiple pixels that are properly arranged in columns and rows. Figure 1.1 is an example of a Normal image.



*Figure 1. 1 Normal Image*(Visuals, 2019)

## 2.Digital image definition :

The digital image is a numeric representation of a two-dimensional image like design, icon… converted from its original analog state by digital converters, these images have a finite set of digital values, called pixels. Digital images can be created directly using computer programs like three-dimensional (3D) modeling.

The process of digitizing an image includes sampling and quantizing.

**a) Sampling:** is the measurements that are taken at a separate location within the image

**b) Quantizing:** is the process of assigning discrete values to the measurement taken in the sample.

### 2.1. Different types of digital images:

There are different types of digital images, it may be of vector or raster type

**a) Vector images**

a vector image -named a line mode image- is composed by individual mathematical geometry, primitive geometry defines for a different characteristic like there form, position or colors.

These modifications can lead to different transformation such as overwriting, enlargement and without losing the initial quality of the image.

**b) Raster image**

a raster image (bitmap) it's an image build by a matrix or network table, each case has one color (colored point). it has a point of a juxtaposition of colors gathered in an image, these colored points named pixel in a numeric image.

The advantages of these modes are:

-simplify memory storage, it enabled the coding sequence value of the matrix.

-make the treatment easy by primitive algorithms at the level of pixel.

The disadvantages of the representation of this type:

-the importance of memory space to guide large images or good quality images.

## 3.pixels

A pixel is a word framed from the picture element and is the smallest unit in a digital image. the abbreviation of a pixel is “picture element” Multiple pixels arranged in rows and columns from an image, and thus, an image is fully composed of pixels.

The pixel’s position in a matrix (line number and column number) and the numeric value present the color or its gray level it can be presented on.

### 3.1 The binary image

the binary system is composed specifically of zeros and ones (0 and 1) and containing only two colors (white and black).
The black color is represented by ‘0’ and the white color is presented by ‘1’. Each pixel in this type of image will only have either of the two 0 or 1, presenting black or white exclusively.

Figure 1.2 is an example of a Binary image

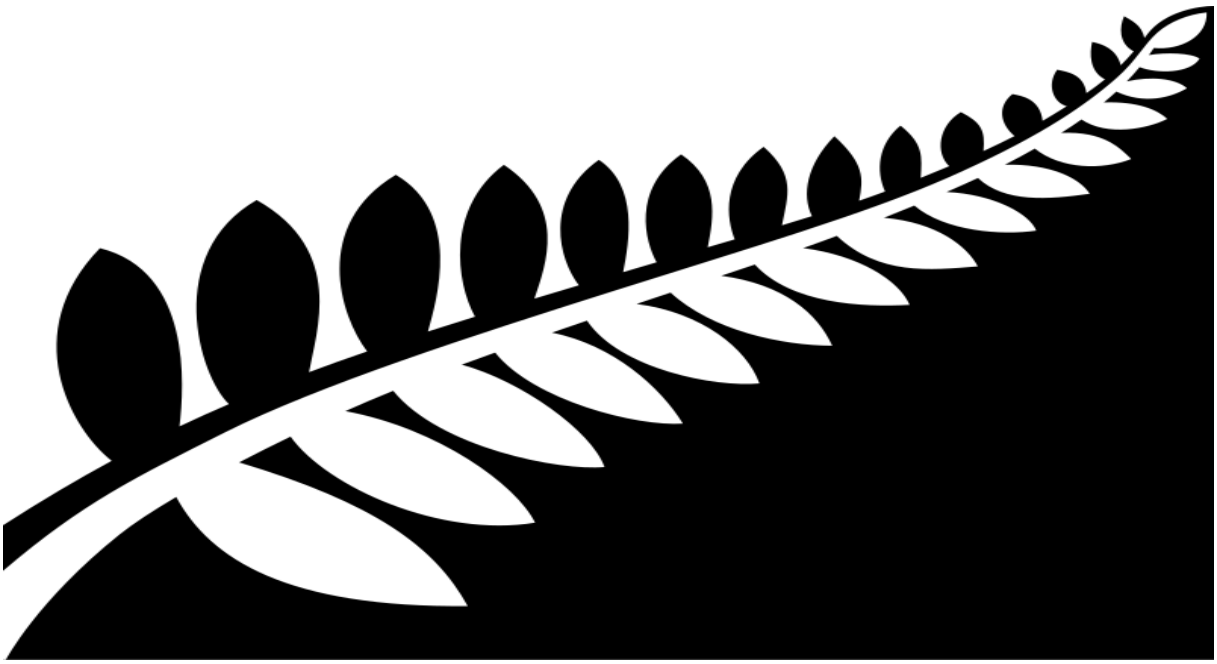


*Figure 1. 2 Binary Image*(Alofi Kanter, 2015)

### 3.2 Grayscale image:

It is a special type of image, often regarded as uncolored, referring to the degradation of shades between white and black.

the most commonly used format is 8 bit and the range of this shade varies from 0 to 255.

’0’ stands for black (same as the previous cases mentioned) while ’255’ stands for white and in-between form the different shades. Figure 1.3 is an example of a grayscale image

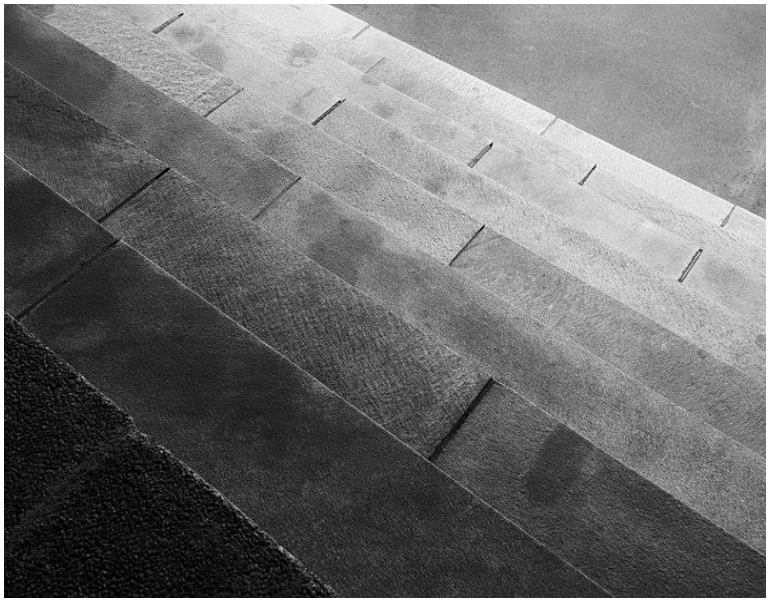


*Figure 1. 3 a grayscale image(W.carter, 2017)*

**3.3 Color image:**

In a color image, each pixel has color information and is composed of three channels which are most commonly regarded as RGB mainly representing Red, Green, and Blue colors.

Each of these channels needs 8bit for storage and hence, it would become 24bits for each pixel. therefore, RGB together constitutes a pixel in a color image.

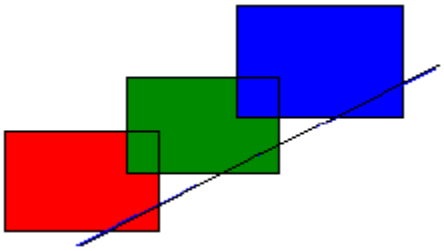


*Figure 1. 4 color image(Padmavathi & Thangadurai, 2016)*

**4. Image type :**

The following are some of the most widely used image formats, which are explained in Table1

- *JPEG*: Joint Photographic Experts Group
- *JPEG2000*: New JPEG format developed in 2000
- *TIFF*: Tagged Image File Format
- *GIF*: Graphics Interchange Format
- *BMP*: Bitmap
- *PNG*: Portable Network Graphics
- *WebP*: Format developed by Google
- *SVG*: Scalable Vector Graphics

| Image Format | Description | Use |
|---|---|---|
| **JPEG** | Lossy compression of raw images | Photographs and Paintings |
| **JPEG2000** | Optimized form of JPEG ; better compression ratio; both lossless and lossy compression | Surveillance |
| **TIFF** | Lossless compression ; can be stored and retrieved without losing information | Document storage |
| **GIF** | Bitmap image format; supports animation; lossless compression. | Gaming and animation |
| **BMP** | Independent of display device; lacks of compression. | In windows |
| **PNG** | Lossless data compression; supports different color spaces | Image transfer over the internet |
| **WebP** | Lossless and lossy compression; small size; but comparable image quality with JPEG. | Stickers in messaging apps |
| **SVG** | For interactivity and animation; behaviors and image defined in XML format; they can by searched; indexed; and compressed. | Web site development |

*Table 1. 1 description and uses of different Image Types (Singh, 2019)*

## 5. Image resolution:

image resolution can be defined in multiple ways, whereas in general it refers to the number of pixels in an image (sometimes called as pixel resolution). For example, an image that is 1920 pixels wide and 1440 pixels high (1920*1440) contains a total of 2.764,800 pixels and the equivalent of 2.7 megapixels.

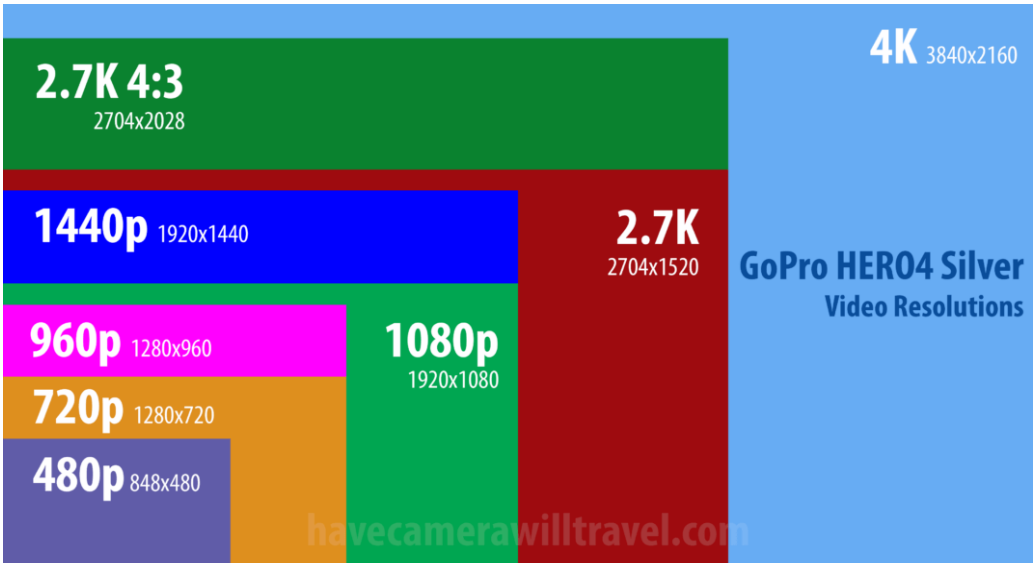



*Figure 1. 5 image resolution (David Coleman, 2020)*

### 5.1 Spatial Resolution:

Spatial resolution states that the clarity of an image cannot be determined by the pixel resolution because the number of pixels in an image does not matter.

Spatial resolution refers to the smallest discernible detail in an image or the number of independent pixels' value per inch.

As the spatial resolution, it's for clarification so for different devices, different measures have been applied to measure it.

how many details an image contains, is usually given by the number of pixels per inch (PPI), a spatial resolution of output devices, such as monitors or printers, is usually given in dots per inch (DPI) and a Lines per inch (LPI) is usually used in laser printers. (Reilly & Frey, 1996) .

### 5.2 Gray level Resolution:

Gray level resolution refers to the predictable or deterministic change in the shades or levels of gray in an image and it’s equal to the number of bits per pixel.

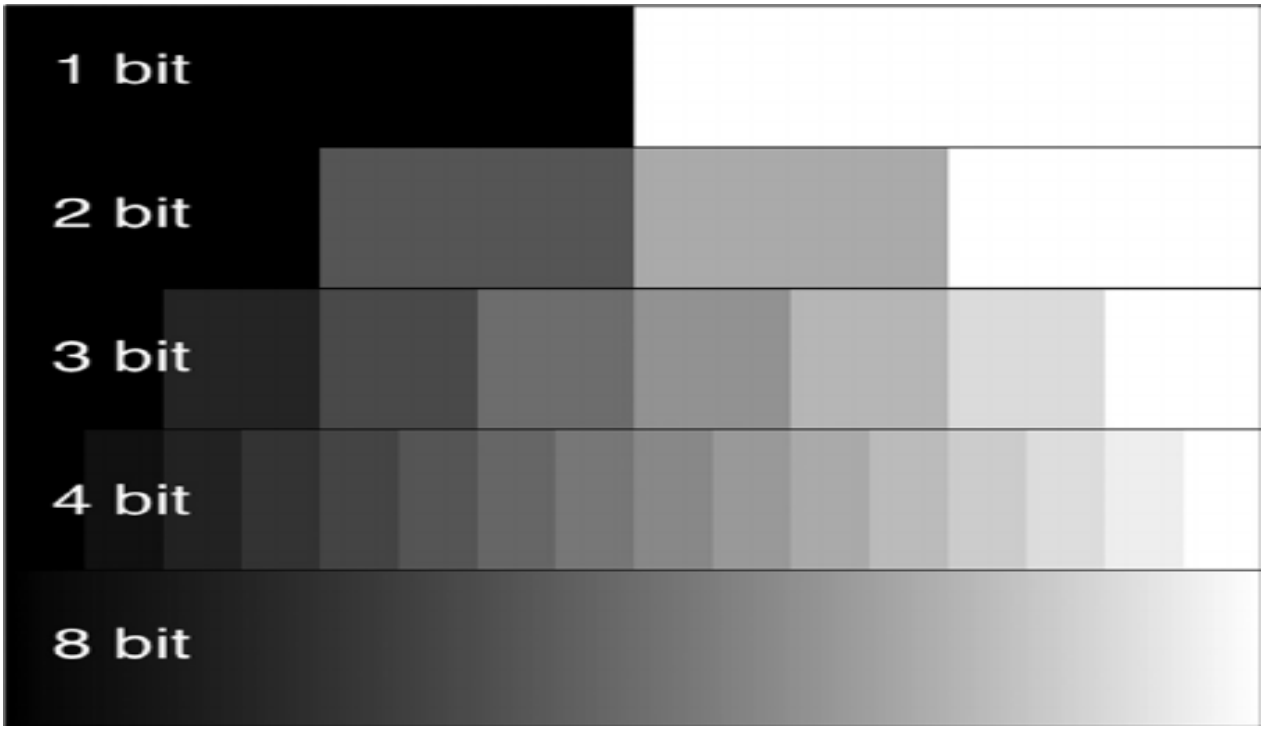



*Figure 1. 6 gray-level resolution(Talley J Lambert, 2014)*

## 6. Image Segmentation

### 6.1 Definition:

Image segmentation is an operation that aims to gather pixels between one pixel and another according to predefined criteria, it is generally the first step of the indispensable process of interpretation of the scene. pixels are also gathered in regions composed with a partition of an image, it is also called binarization.

The segmentation is the main step of image treatment. there are many techniques of segmentation, which can be regrouped on four principle classes:

1- thresholding: the thresholding method is merged with other methods(Suzuki & Toriwaki, 1991).

2- The region's growing method extends thresholding by combining it with connectivity. This method needs seed for each region and has the same problem of thresholding for determining a suitable threshold for homogeneity(Balafar, Ramli, Saripan, & Mashohor, 2010)

3- statistical models

4- active control models and clustering.

#### *6.1.1. Region approach:*

In region approach, the affinity of the related point is known as a context, related points with similar properties: gray intensity, color, texture, will be combined in the same set.

### *6.1.2 bored approach:*

This approach is a none-context technique that ignores rapports that may exist between region and image. We regroup pixels according to a global interlude, it understands the techniques of edge-based detection, thus, edge-based does not always lead directly to the researched partition. So it needs to use algorithms of closed edge-based.

### 6.2 Filter:

The filter is represented by a matrix with dimension and coefficients that determine the filter properties. The multiplication of the image matrix with filter make a new matrix correspond to the treated image.

### 6.3 smoothing images:

Smoothing image, it is a filtering operation to eliminate image noise. there are filters used to smooth images, these filters are:

### *6.3.1 linear filter:*

the linear filters used to convert a set of input data to a set of output data according to convolution operation.

a) **Medium filter**: the medium filter is a filter used to average the value of the neighboring pixels. The result of this filter is a more blurred image.
b) **Gaussian filter:** The Gaussian filter depends on the standard deviation of the image(distribution) and assumes the mean is zero (we can define a mean different from zero as well). Gaussian filters do not take care of the edges.(Singh, 2019)
c) **Mean Filter**: Mean filter is the simplest low pass linear filter.

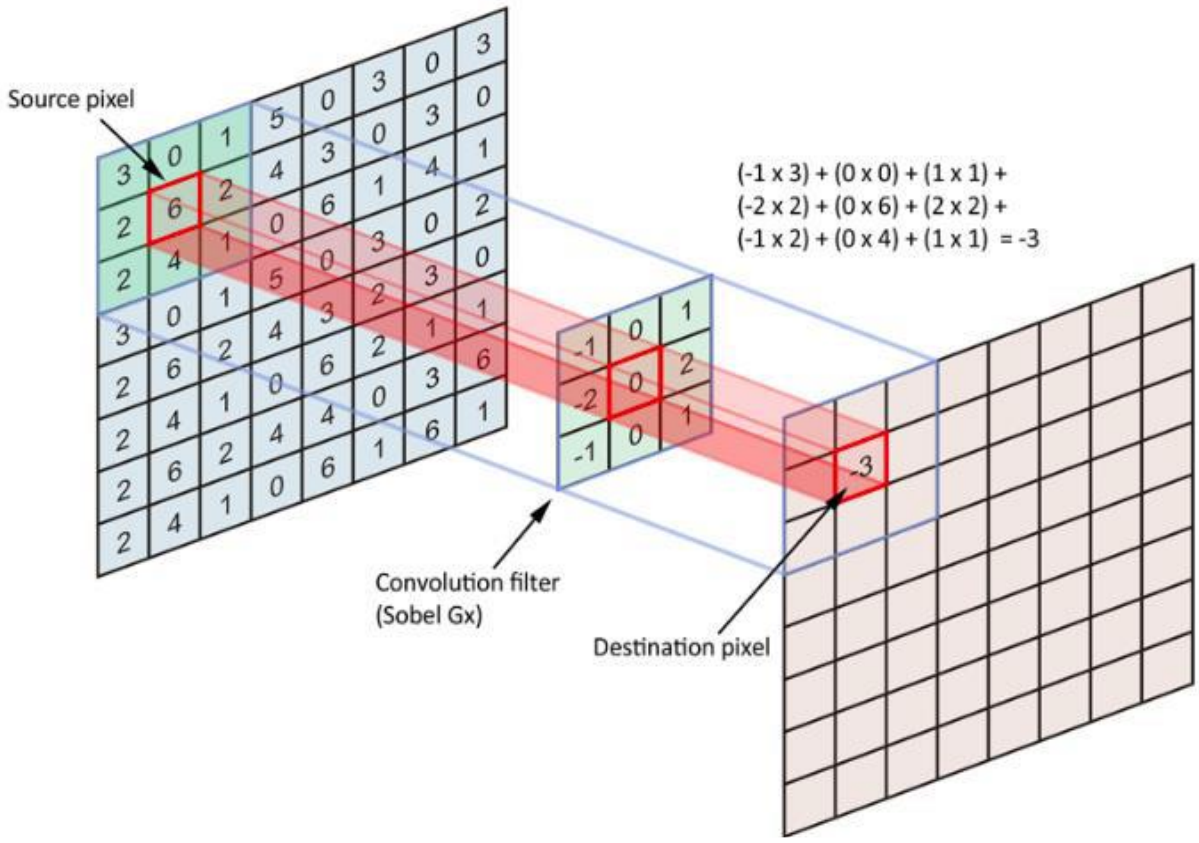



*Figure 1. 7 Mean filter application(Cornelisse, 2018)*

### *6.3.2 nonlinear filter:*

**a) Median filter:** The median filter is one of the most basic image-smoothing filters. It's a nonlinear filter that removes black-and-white noise present in an image by finding the median using neighboring pixels.(Singh, 2019). This filter is used to attenuate isolated pixels, of a very different value from their neighborhood. This filter does not introduce blurring on the contours of particles unlike smoothing. Figure 1.8 shows the median filter application(Mark Nixon & Alberto Aguado, 2002).

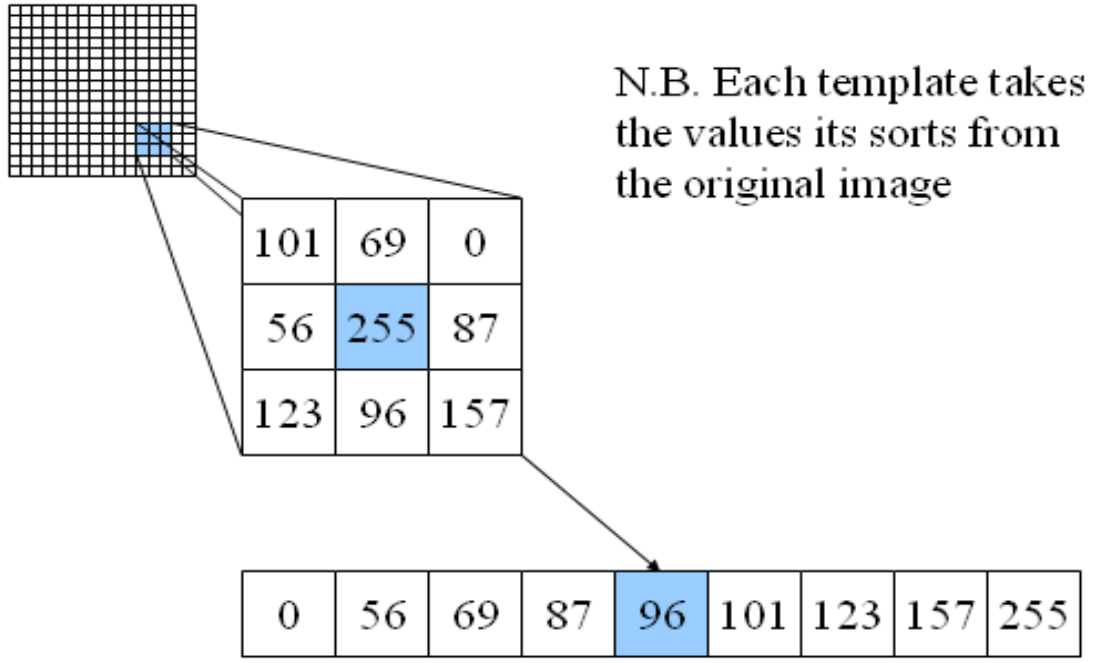



*Figure 1. 8 median filter (Mark Nixon & Alberto Aguado, 2002).*

Working of 3*3 median filter; in the filtering process the center pixel value 255 is replaced by 96 which is the median value of the 8 neighborhood pixels.

### 6.4. Image noise:

image noise is a signal interference and variation of brightness or color information. Noise is any kind of deviation from what's expected at the pixel level, there are several reasons why noise forms, for example, some noise forms because of how light arrives at the sensor while some is generated from converting the charger of each pixel into digital information.

### 6.5. convolution:

Convolution is applying a matrix named a 'kernel' or 'filter' to an image and that can reduce the size of it.to keep the same size of an image we can add several padding layers.

Convolution is used for extracting features from an image like shape and edge as well as other applications.

There are two types of convolution:

**Valid convolution:**

Without padding operation, whereas if the matrix size is (n*n) and the size of the filter is (f*f) and the Padding value is equal to zero (0). So, the new matrix size will be:

**(n*n) * (f*f) = (n-f+1) * (n-f +1)**

S is stride which is the number of pixels shifts over the input matrix. If the stride value is over than one (1). The new matrix size will be:

**(n*n) * (f*f) = (((n-f) /s) +1) * (((n-f) / s) +1)**

**same convolution:**

with the use of padding operation, the matrix stays the same after the convolution

(n + 2p – f +1) = n

So, from that we can find the padding value to keep the same matrix size where:

**2p –f + 1 = n-n**

**2p –f +1 = 0**

**P = (f – 1)/2**

### 6.6. The contour:

Contour detection is like a curve joining all the continuous points, having colors and intensity. The contours are a helpful tool for shape analysis, detecting object and object recognition. The contours can find the edges of images which can reduce the quality of data as well as eliminate some information.

There are three contours type:

- (Step): the outline is sharp (ideal contours).

-Ramp: the contours are more blurred

-roof: it is a line on a uniform background

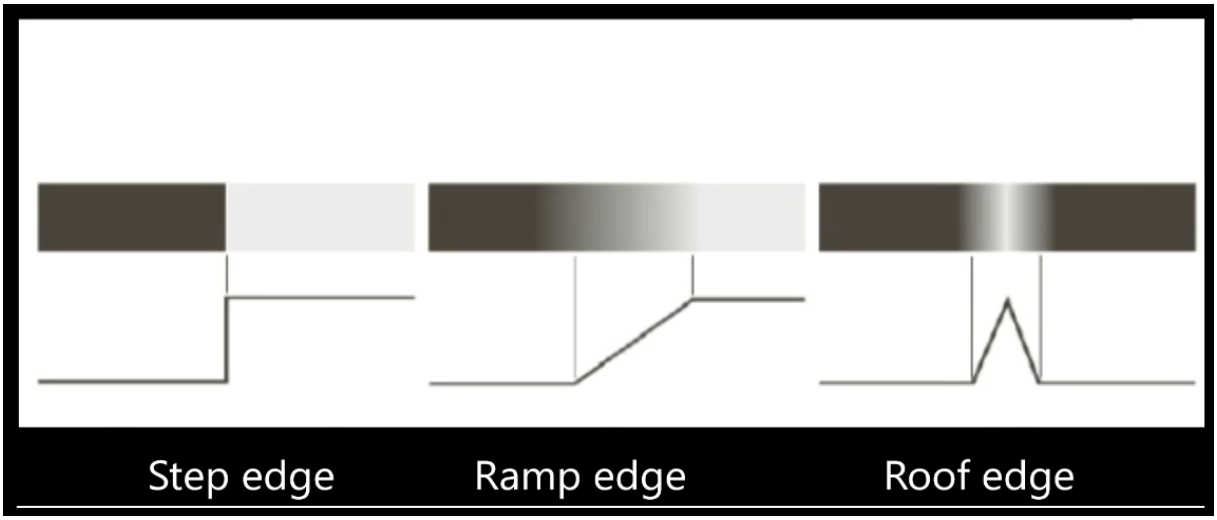



*Figure 1. 9 different contours types*

# 7. medical imaging:

**Introduction:**

Medical imaging technologists and radiographers acquire images, high-quality images of the human anatomy to diagnose and help manage patient's conditions without any surgery or intrusive exams.

The development of technology in medical imaging has helped doctors to make easy and less life threatening therapeutic decisions, medical imaging techniques have many advantages, especially the pathological visualization entity itself, allowing a better understanding and orientation of the diagnosis.

## 7.1. Medical imaging technologies:

### *7.1.1 history of medical imaging:*

The first Medical imaging was elaborated by Wilhelm Conrad Rontgen (1845-1923) called X-ray. Working with an early cathode ray tube called a Crooke's tube, he noticed that the invisible rays were able to penetrate some solids (like human flesh) better than others (like bone or metal(Bradley, 2015).

After that there was a lot of development like X-ray tomography that was introduced in the 1940s, allowing "tomograms" or slices to be obtained through tissues without the over- or under-lying tissue's being seen. (Bradley, 2015)

In 1972 the British radiologists Allan Mc Kornack (1924-1998) and Godfrey N. Hounsfield (1918), developed a scanner with digital image processing.

Magnetic resonance imaging MRI, with high-resolution sectional images that require no X-rays at all, MRI began in the 1980s, the technology depicts the function and structure of organs and tissues with unprecedented quality.

### *7.1.2. Imaging modalities:*

There are many techniques of medical imaging processing used like magnetic resonance imaging (MRI), X-ray, digital mammography and others on (Vannier & Haller, 1998). For clinical usage, images are now almost collected and stored in a digital representation(Chang & Teng, 2007).

We can classify medical imaging into two categories, functional and structural imaging. the functional or physiological imaging is a detection technique or measurement of changes in metabolic information like variation of blood flow, regional chemical composition and cell visualization.

Structural imaging is a body visualization that gathers scanner, radiography, and MRI.

**X-Ray:**

x-ray is a snapshot of a specific parts of the body whether it's the chest or the abdomen or a specific joint like the knee or the thigh.

a beam of x-rays (highly charged electrons) that is passed through the body

-the high-density object will appear white (example: bone)

-the low-density objects will appear black (example: lung)

**CT Scan:**

It is a Series of cross-sectional X-rays taken from a 360-degree view around the patient, which helps create slice by slice maps of the patient, it also helps for looking to bone soft tissue and vessels all at the same time without separating them.

-The high-density object will appear white while the low-density objects will appear black

Intravascular **Ultrasound** (IVUS)**:**

high-frequency waves are bounced off of tissues in the human body, used to create a cross-sectional image of a specific part of the body, it is most effective in visualizing body cavities, tissues, and blood vessels. A growing number of cardiologists think that new information yielded by IVUS can make a significant difference in how the patient is treated and can provide more accurate information which will reduce complications and incidence of heart diseases. Figure-1.10, generated in real-time, yields information that is not possible with routine imaging methods or even non-invasive Multislice CT scans. (Chen, 2014)

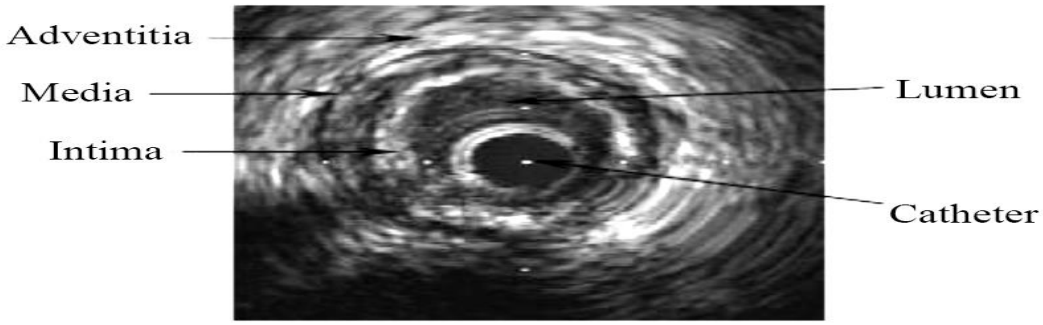



*Figure 1. 10 Typical IVUS image(Chen, 2014).*

**magnetic resonance imaging (MRI):**

Magnetic resonance imaging, invented in 1970, using for human body visualization and can be adapted to image brain, spinal cord and muscles. Clinical MRI is based on the hydrogen nucleus due to its abundance in the human body and its magnetic resonance sensitivity(Balafar et al., 2010).

For image formation, a large static magnetic field is used to perturb magnetically moments of the proton that exist in the hydrogen nucleus from their equilibrium and observing how perturbed moments relaxes back to their equilibrium. Naturally, the protons are oriented randomly. But in the existence of a static magnetic field, they line up with the field and the net magnetization of protons tends toward the direction of the field. In the existence of enough energy, it is possible to make the net magnetization zero. In the relaxation process, and the induced electronic signal is recorded. The strength and duration of the signal depend on three quantities:

1. $\rho$ (proton density)
2. spin-lattice relaxation time: the time which describes how fast the net magnetization takes to relax back to its equilibrium (T1).
3. spin-spin relaxation time: with this time, magnetization components decrease to zero (T2).(Balafar et al., 2010)

### 7.2. Medical image segmentation :

Medical image segmentation is the isolation of scenes into the component regions, and there are Segmentation techniques used to extract an attribute from an image by partition it into multiple regions.

Segmentation is achieved by subjective or objective methods that associate and aggregate locally homogeneous image elements based on some measurable
feature (or features) such as intensity, texture, gradient or many others. (Vannier & Haller, 1998)
segmentation techniques are very important in discovering the region of interest.

## 8.Computer vision

Computer vision is an important technique to provide the required information in diagnosis and treatment. the first thing we need to consider for medical images is to use image processing techniques including image enhancement, color image processing, wavelet and multiresolution image processing, image compression, morphological image processing and image segmentation. Both image enhancement and segmentation can be considered as low-level computer vision. For mid-level computer vision, computers can organize the knowledge (information) acquired in the low-level vision to make some useful decisions. For high-level vision, computers must further provide some "thinking" capabilities like humans do(C. H. Chen, 2014).

## 9. Malaria images:

Malaria is an infectious disease in the human body, specifically in blood or liver. Different medical imaging methods are used to identify malaria infections like X-ray, ultrasound images, RMI and abdominal computed tomography (CT).

X-ray Malaria image:

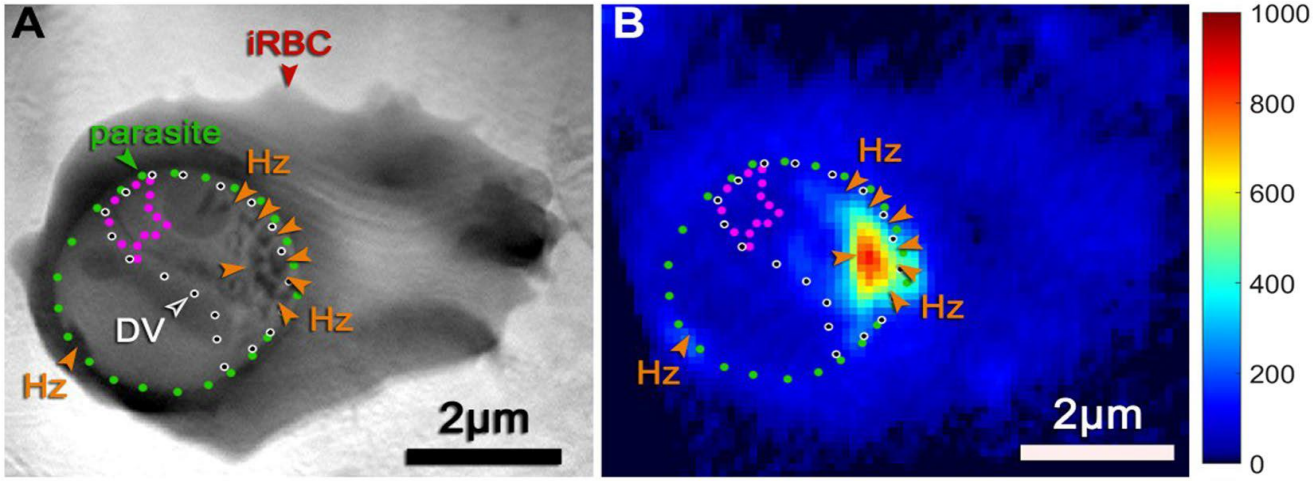



*Figure 1. 11 Soft X-ray tomography and Iron x-ray fluorescence map*

(**A**) Soft X-ray tomography projection image of the infected cell.
The dotted lines delineate the parasite (green), its digestive vacuole (black-on-white), and the area of the digestive vacuole free
from hemozoin fluorescence signal (magenta). Some of the hemozoin crystals are indicated by orange arrows.
(B) Iron x-ray fluorescence map of the same cell aligned to its x-ray tomography dataset.(Kapishnikov et al., 2017)

Abdominal computed tomography (CT):

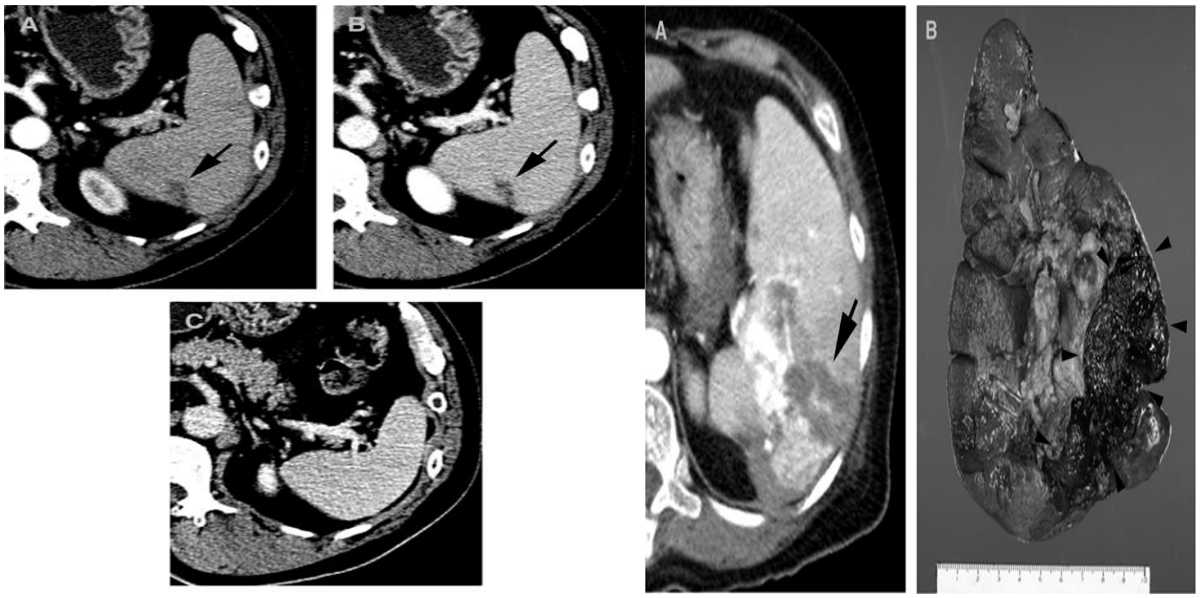



*Figure 1. 12 A-blood infection (Kim et al., 2010) B- splenic rupture(Kim et al., 2010)*

Several types of Malaria parasite taken by microscope. five (5) Plasmodium species cause malaria in humans: *Plasmodium falciparum, Plasmodium vivax, Plasmodium malaria, Plasmodium ovale*, and Plasmodium knowlesi. (Poostchi, Silamut, Maude, Jaeger, & Thoma, 2018). During 48 hours each parasite species goes through several steps ordered as follows: ring stage, trophozoite stage, schizont stage, and gametocyte stage.
figure 1.13 shows the different stages of each species.

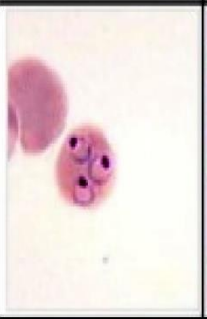
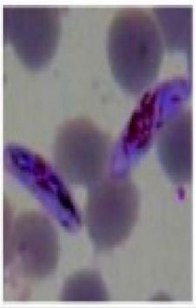
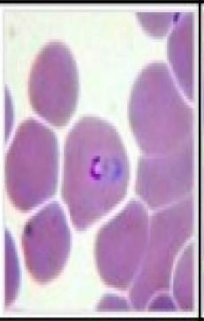
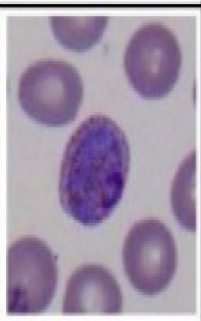
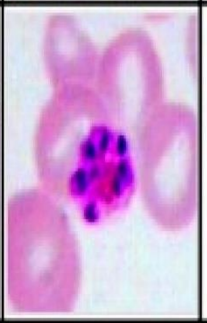
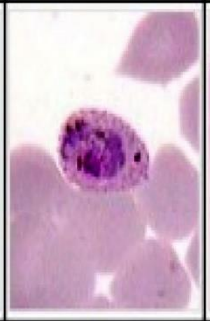
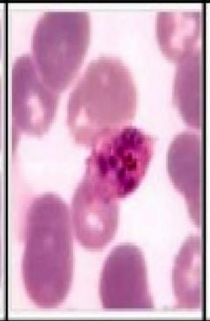
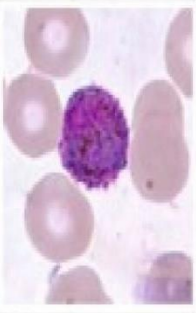
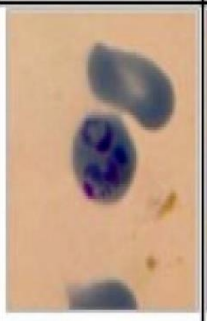
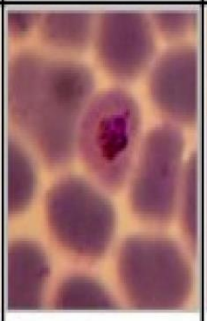
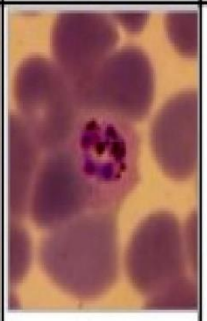
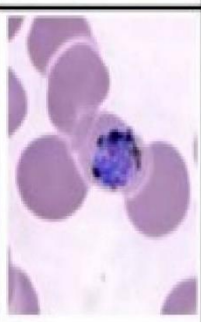
| Stages / Species | Ring | Trophozoite | Schizont | Gametocyte | |
|---|---|---|---|---|---|
| *P.falciparum* |  |  |  |  | • *Parasitised red cells (pRBCs) not anlarged.*<br>• *RBCs cantaining mature trophozoites sequestered in deep vessels.*<br>• *Total parasite bimass = circulating parasites + sequestered parasites.* |
| *P.vivax* |  |  |  |  | • *Parasites prefer young red cells.*<br>• *pRBCs enlarged.*<br>• *Trophozoites are amoebold in shape.*<br>• *all stages present in peripheral blood.* |
| *P.malariae* |  |  |  |  | • *Parasites prefer old red cells.*<br>• *pRBCs not enlarged.*<br>• *Trophozoites tend to have a band shape.*<br>• *all stages present in peripheral blood.* |
| *P.ovale* |  |  |  |  | • *pBRCs slightly enlarged an have an oval shape, with tifted ends.*<br>• *All stages present in peripheral blood.* |
| *P.knowlesi* |  |  |  |  | • *pRBCs not enlarged.*<br>• *Trophozoites, pigment spreads inside cytoplasm, like P.malariae, band form may be seen.*<br>• *Multiple invasion & high parasitaemia can be seen like P. falciparum.*<br>• *all stages present in peripheral blood.* |

*Figure 1. 13 Five different human malaria Plasmodium species and their life stages in thin blood film (Source: K. Silamut and CDC).*

### 9.1. Malaria detection:

Detecting Malaria is not the easiest action to do with humans compared to a large number of cells to be examined, because specialists manually count how many cells are infected and how many are uninfected.

There are different methods used in the past where high-quality microscopy services are not available such as polymerase chain reaction (PCR) and rapid diagnostic tests (RDT).

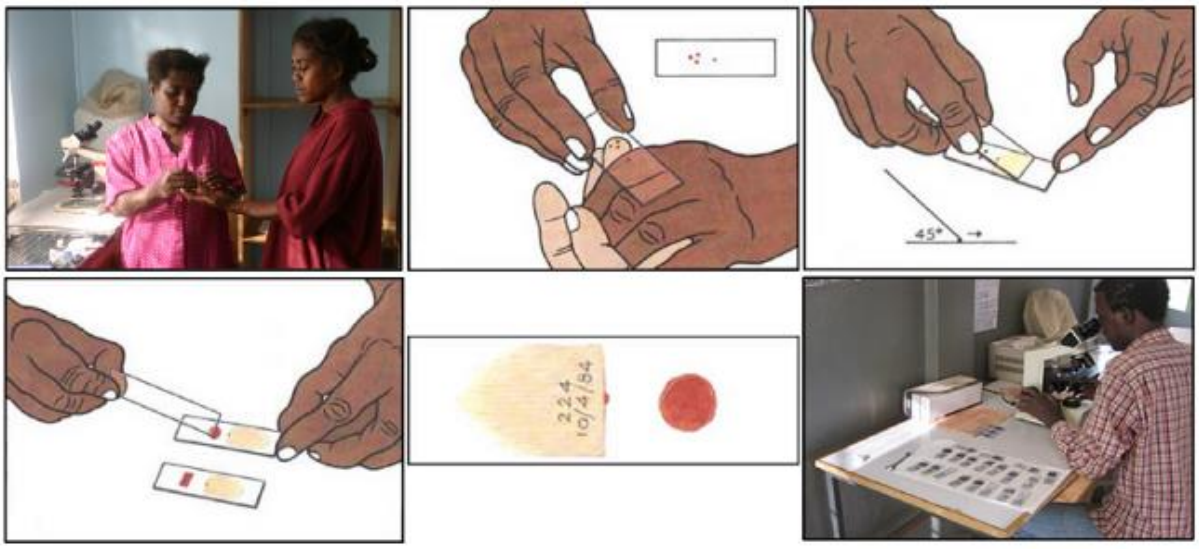



*Figure 1. 14 malaria Blood smears* Image sources: Microscopist, Wendy Saravagy. Illustrations of blood smears. Microscopist examining blood smears.

Rajaraman and al.., in his paper explains:

Thick blood smears assist in detecting the presence of parasites while thin blood smears assist in identifying the species of the parasite causing the infection
(Centers for Disease Control and Prevention, 2012). The diagnostic accuracy heavily depends on human expertise and can be adversely impacted by the inter-observer variability and the liability imposed by large-scale diagnoses in disease-endemic/resource-constrained regions (Mitiku, Mengistu & Gelaw,2003). Alternative techniques such as polymerase chain reaction (PCR) and rapid diagnostic tests (RDT) are used; however, PCR analysis is limited in its performance (Hommelsheim et al., 2014) and RDTs are less cost-effective in
disease-endemic regions (Hawkes, Katsuva & Masumbuko, 2009). (Rajaraman et al., 2018).

Hence, Malaria detection should be directed to automation, that can facilitate the diagnosis and treatment of the case. Artificial intelligence (AI) is a technique of automation that can be used for detecting Malaria disease with a low-cost, and effective. Thus, Malaria detection could benefit from automation using deep learning.

## Conclusion:

This chapter presents the fundamental notion of images, digital images and medical imaging with the problematic of malaria disease.
First and foremost, it contains the different components of images as well as the techniques that can be used for segmentation in medical imaging.
Furthermore, this project entertains different deep learning techniques for classifying Malaria infection; whereas the following chapters will present and discuss the methods of classification and other neural networks.

# Chapter 2: State of the art of Malaria detection using deep learning.

## Introduction

According to the considerable number of data that needs to be analyzed by specialists, a human can commit multiple errors as well as waste an important amount of time to diagnose the designated case in the pathological field.

the need prompted us to use deep learning for pathological field related diagnosis to detect disease with a low cost, high efficiency and minimal time.

In this chapter, we aim to present machine learning, deep learning, and its diverse applications, as well as Malaria detection (previously mentioned).

## 1.Machine learning

Machine learning is a subfield of computer science (CS) and artificial intelligence (AI) that deals with the construction and study of systems that can learn from data, rather than follow only explicitly programmed instructions(Preethi, 2014).

Machine learning gives computers the ability to learn on their own without any human interaction to do programming, design nor development of algorithms necessary.

## 2.Deep learning

Deep learning is a subset of machine learning inspired by the structure of the human brain, this structure is based on artificial neural network (ANN). ANN constantly receives learning algorithms and continuously growing amounts of data to increase the efficiency of training processes.

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved state-of-the-art speech recognition, visual object recognition, object detection and many other domains(Lecun, Bengio, & Hinton, 2015). Figure 2.1 shows an example of the deep basics.

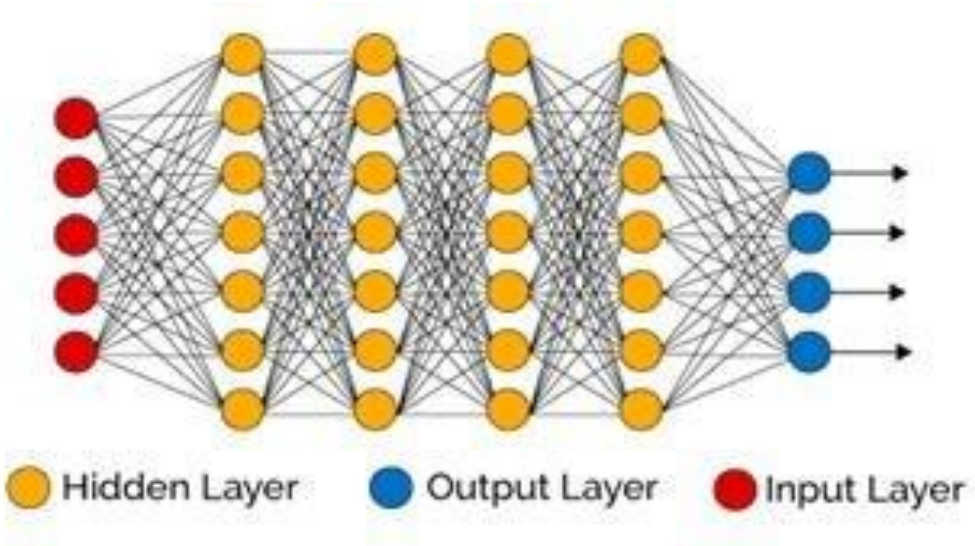



*Figure 2. 1 deep learning basics*(Parashu Ram Pal, 2019)

## 3.Difference between deep learning and machine learning:

As previously mentioned; deep learning is a substance of machine learning branch that is used in the extraction of knowledge from a large data array loaded into the machine. Users correct errors made by machines to eliminate negative overtraining effects frequently appearing in deep learning.

In machine learning, users manually label features and provide a machine with training data to help the system make correct decisions.

The differences between machine learning and deep learning consist of:
-time and mathematical operation is greater in DL compared with ML.
-ML uses structured data whereas DL uses unstructured data.
-DL requires an immense amount of data, on the other hand ML needs a relatively less number of data which afterwards becomes unnecessary.

-The output of ML is purely an amount information while output of DL consists of information, image, audio, video or even text.

-Unlike ML, DL needs high performance hardware.
-ML requires features that are accurately identified by users while DL creates new features by itself.

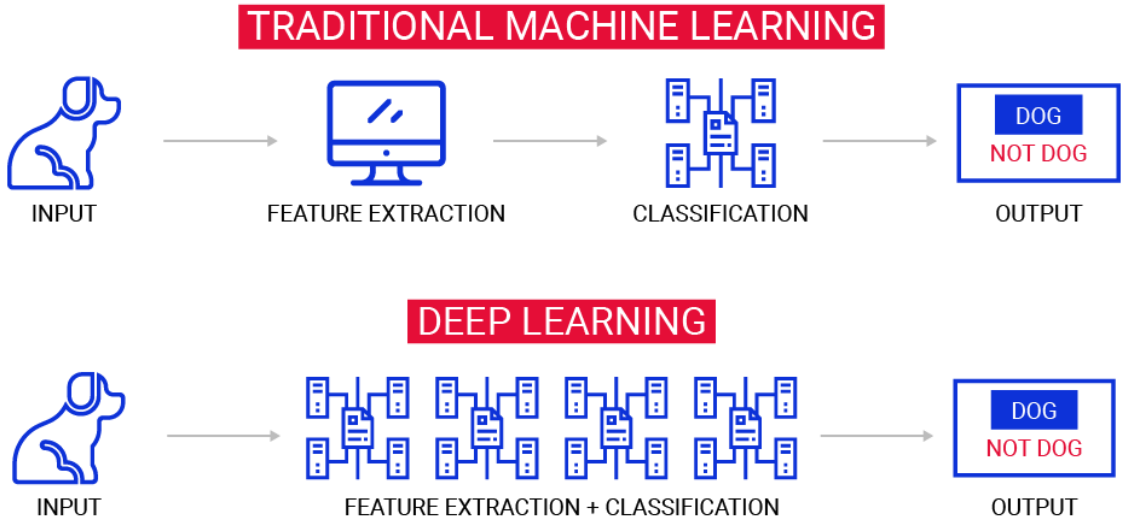



*Figure 2. 2 the difference between deep learning and machine learning(Hammami, 2019).*

## 4.Deep learning techniques:

There are many techniques in deep learning that have been applied to solve problems. From these techniques we mention:

### 4.1. Artificial Neural Network(ANN):

Artificial neural systems or neural networks are cellular systems that can acquire, store and utilize experimental knowledge(Nelson & Wang, 1992). ANN is based on a collection of connected neurons where each connection between neurons can transmit a signal to another neuron.

### 4.2. Deep Neural Networks(DNN):

DNN is an Artificial neural network that contains a lot of layers between the input and the output layer, where neurons receive as input the neuron activations from the previous layer, and perform a simple computation (Montavon, Samek, & Müller, 2018).

### 4.3. Deep Belief Networks:

(DBN) is a class of deep neural networks that are composed of a multiple stochastic layers and latent variables. The latent variables typically have binary values and are often called hidden units or feature detectors. The top two layers have undirected, symmetric connections between them and form an associative memory. The lower layers receive top-down, directed connections from the layer above. The states of the units in the lowest layer represent a data vector. (Geoffrey E. Hinton (2009).)

### 4.4. Recurrent Neural Networks:

An RNN is a neural network architecture that contains cyclic connections, which enable it to learn the temporal dynamics of sequential data. A hidden layer in an RNN contains multiple nodes.

each node has a function for generating the currently hidden state **ht** and output **yt** by using its current input **xt** and the previously hidden state **ht−1** according to the following equations: (Murad & Pyun, 2017)

$$h_t = \mathcal{F}(W_h h_{t-1} + U_h x_t + b_h) \quad (1)$$

$$y_t = \mathcal{F}(W_y h_t + b_y), \quad (2)$$

**-wh:** the weight for the hidden-to-hidden recurrent connection

**-uh:** is the input-to-hidden connection

**-wy:** is hidden to the output connection

**-bh:** bias terms for the hidden

**-by:** is the output states

**-F:** is the activation function for each node

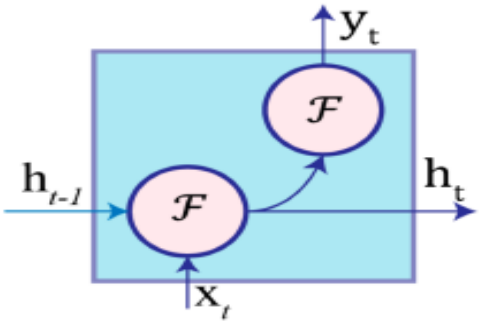



*Figure 2. 3 schematic diagram of an RNN (Murad & Pyun, 2017)*

### 4.5. Long Short-Term Memory (LSTM):

LSTM is an artificial Recurrent Neural Network (RNN) architecture used in deep learning so that large neural network structures can be trained.
LSTM leads to many more successful runs and learns much faster. LSTM also solves complex, artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms(Hochreiter, 1997)

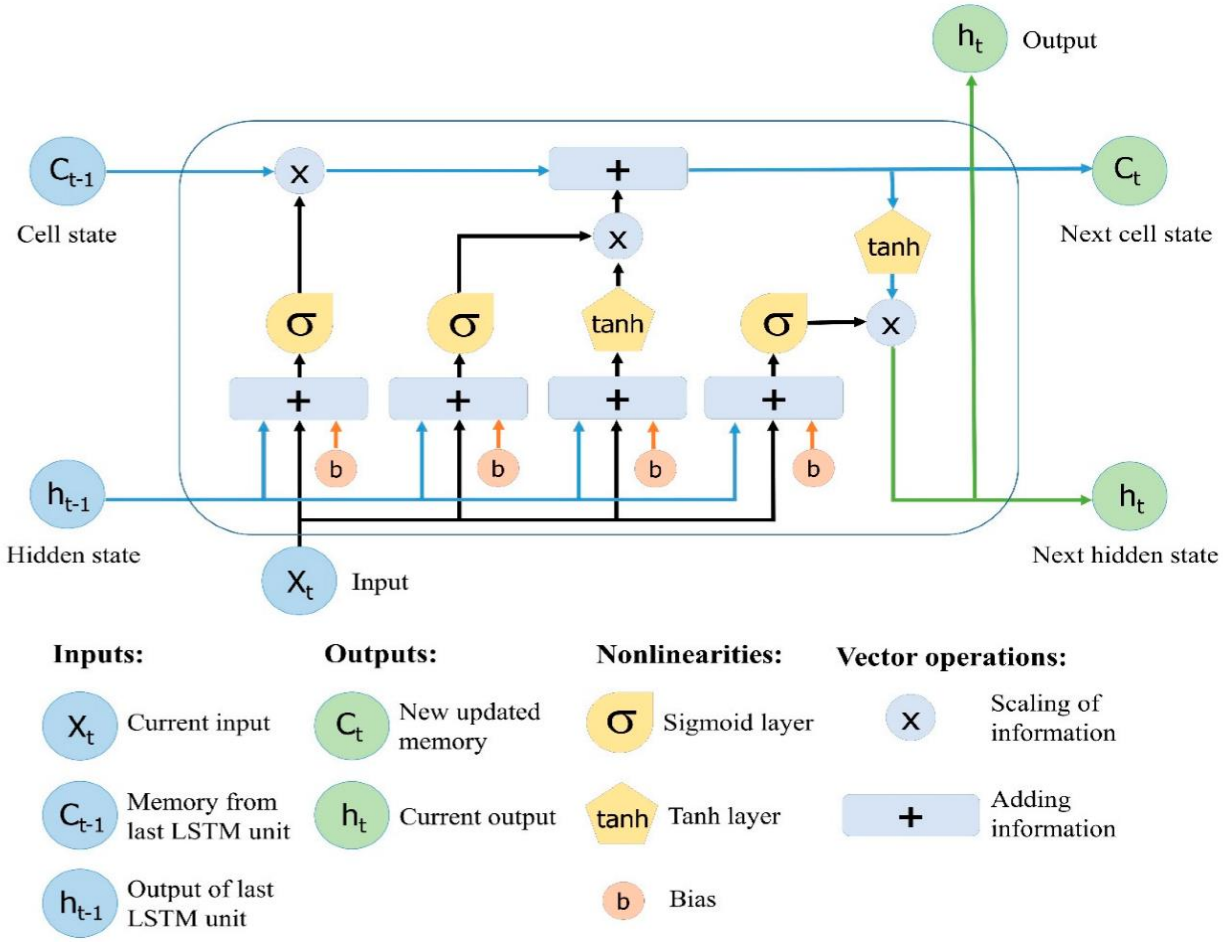



*Figure 2. 4 LSTM architecture* (Yan, 2016)

### 4.6. AutoEncoder

AutoEncoder is an artificial neural network for unsupervised learning. Autoencoder extract output data to reconstruct input data and compare it with original input data. After numerous times of iterations, the value of cost function reaches its optimality, which means that the reconstructed input data is able to approximate the original input data to a maximum extent. (M. Chen, Shi, Zhang, Wu, & Guizani, 2017). Figure 2.5 shows auto-encoder architecture.

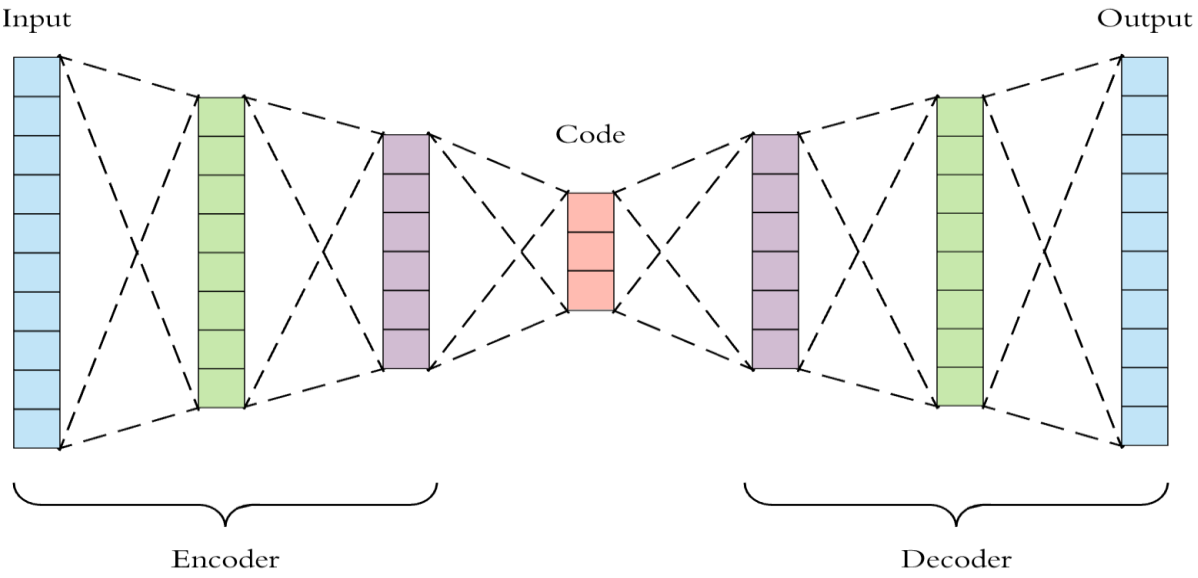



*Figure 2. 5 AutoEncoder architecture* (Matthew Stewart & Stewart, 2019)

The Encoder and Decoder are fully-connected feedforward neural networks, and the Code is a single layer of an ANN with the user's dimensionality. The number of nodes in the code layer is a *hyper parameter* that should be settled before training the Autoencoder.

### 4.7. Convolutional Neural Networks (CNN)

A convolutional neural network (CNN) is a part of a deep neural network that contains one or more convolutional layers that automatically recognize, divide, and bind images and classifications as well as specialize in processing data that contains a network like topology such as an image.

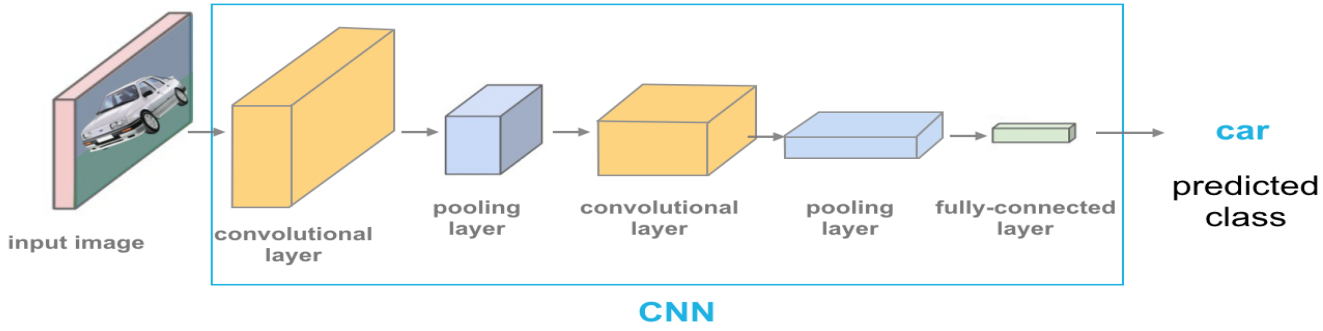



*Figure 2. 6 typical CNN architecture* (Camacho, 2018)

### *4.7.1. Convolutional Layer:*

A convolutional layer is the most important component of a CNN that summarizes the presence of features in an input image and contains a set of filters which parameters must be learned. Using a filter to determine the presence of attributes followed by several filters that can be used to extract different attributes. The size of these filters is considerably smaller than the input size of the image.
The convolutional layer contains also an activation function for the filters.

### *4.7.2. Pooling Layer:*

Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map(Gonzalez, 2007) nonetheless, it is a new layer added after applying activation function such as (ReLU) to the output of features maps by convolutional layer. Figure 2.7 shows the pooling types.

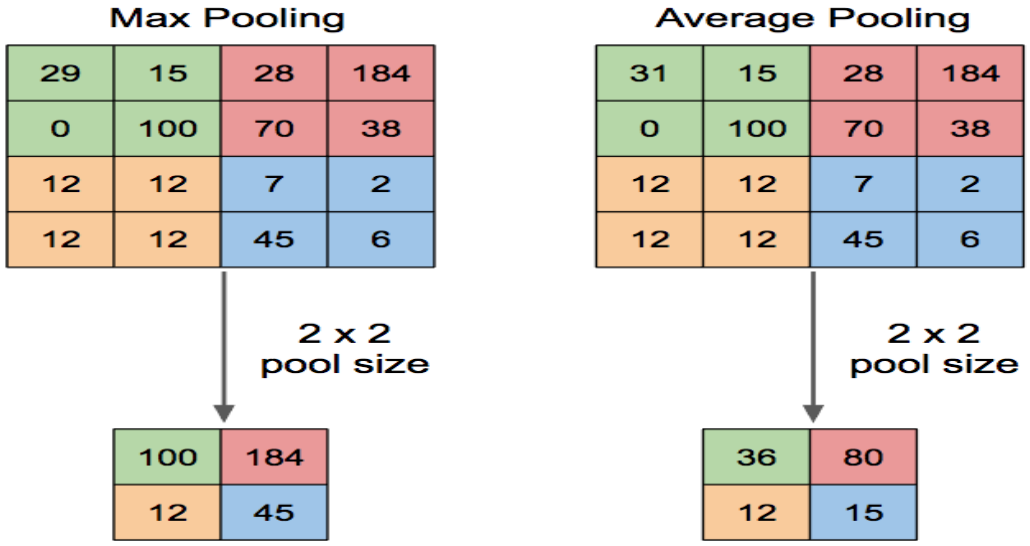



*Figure 2. 7 Types of pooling (Tammina, 2019)*

### *4.7.3. Rectified Linear Unit Layer (ReLU):*

A rectified linear unit (ReLU) is a layer that applies the non-saturating activation function
$f(x) = \max\{0, x\}$
Compared to other functions the usage of ReLU is preferable because CNNs with ReLU are trained without making a significant difference to generalization
accuracy.(Romanuke, 2017).

### *4.7.4. Fully connected layer:*

The fully-connected layer comes after several convolutional and max-pooling layers and gives high-level reasoning in the neural network. Each neuron in a fully connected layer has connections to all activations in the previous layer.

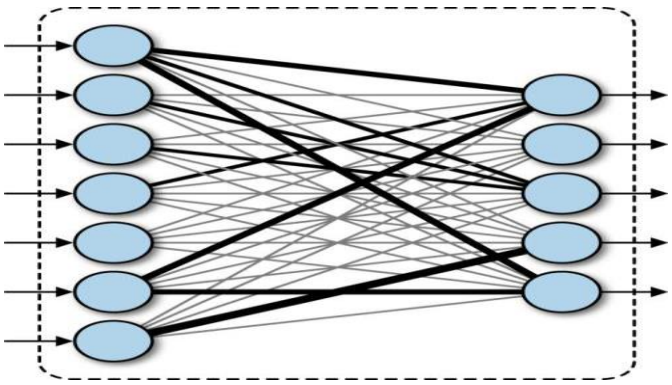


*Figure 2. 8 fully connected layer (Tammina, 2019)*

### *4.7.5. Dropout Layer:*

Dropout is a layer used only during the training whilst on hold during testing time, it's a method used for reducing the overfitting.

The idea of dropout is simplistic in nature. This layer "drops out" a random set of activations in that layer by setting them to zero.

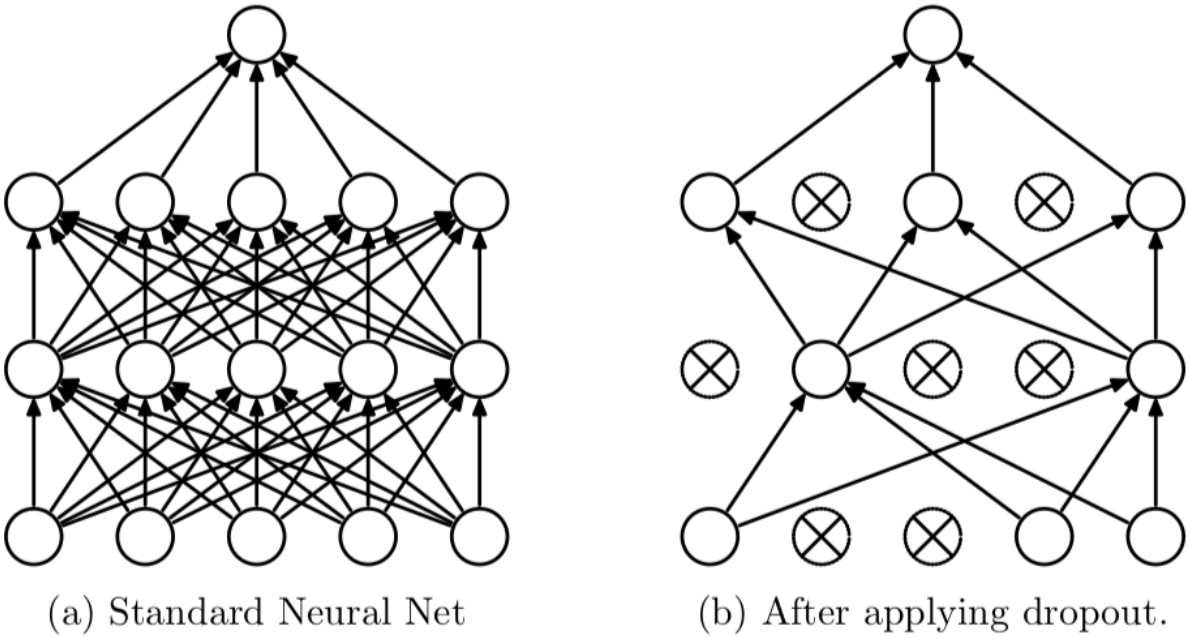



*Figure 2. 9 Dropout example* (Budhiraja, 2016)

### *4.7.6. Batch Normalization Layer:*

Batch Normalization adds normalization layers between each existing layer. is a technique to normalize activations in intermediate layers of deep neural networks. Its tendency to improve accuracy and speed up training have established BN as a favorite technique in deep learning(Bjorck, Gomes, Selman, & Weinberger, 2018).

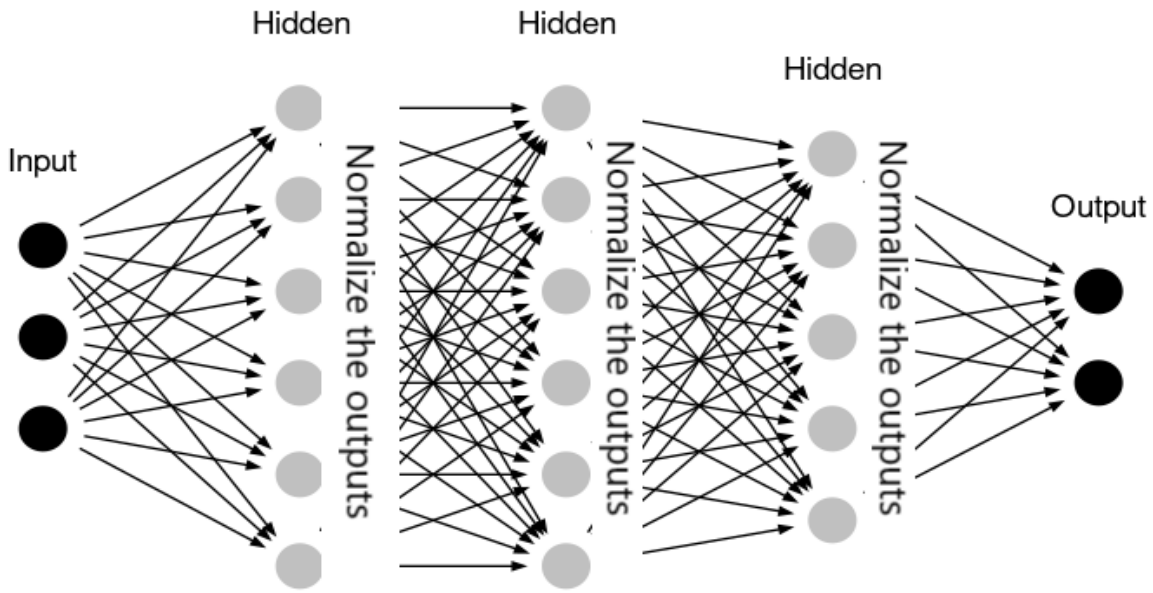



*Figure 2. 10 Batch Normalization*(I. R, 2018)

***4.7.7. sigmoid:***

Sigmoid is a non-linear function that produces a new representation of the original data when it is used by each neuron in a multi-layer neural network. This function
performs well with binary classification.

The mathematical representation of sigmoid activation is shown in (1), where z is the dot product of each neuron value with the weights(Sai Bharadwaj Reddy & Sujitha Juliet, 2019). Figure 2.11 shows the sigmoid function.

$$Q(z)= \frac{1}{1+e^{-z}} \text{----------(1)}$$

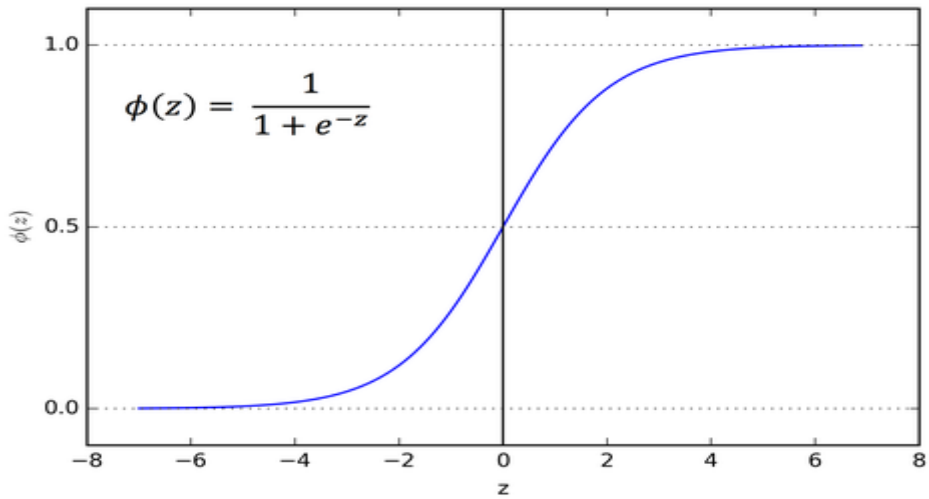



*Figure 2. 11 sigmoid function*(Sai Bharadwaj Reddy & Sujitha Juliet, 2019)

***4.7.8. Softmax:***

A Softmax function is a type of squashing function. Squashing functions limit the output of the function into the range 0 to 1. This allows the output to be interpreted directly as a probability.

Similarly, softmax functions are multi-class sigmoids, meaning they are used in determining the probability of multiple classes at once. Since the outputs of a softmax function can be interpreted as a probability (i.e. They must sum to 1), a softmax layer is typically the final layer used in neural network functions. It is important to note that a softmax layer must have the same number of nodes as the output layer.(Jason Brownlee, 2016).

The function of softmax function is:

Y = softmax (XW + b) ----------**(1)**

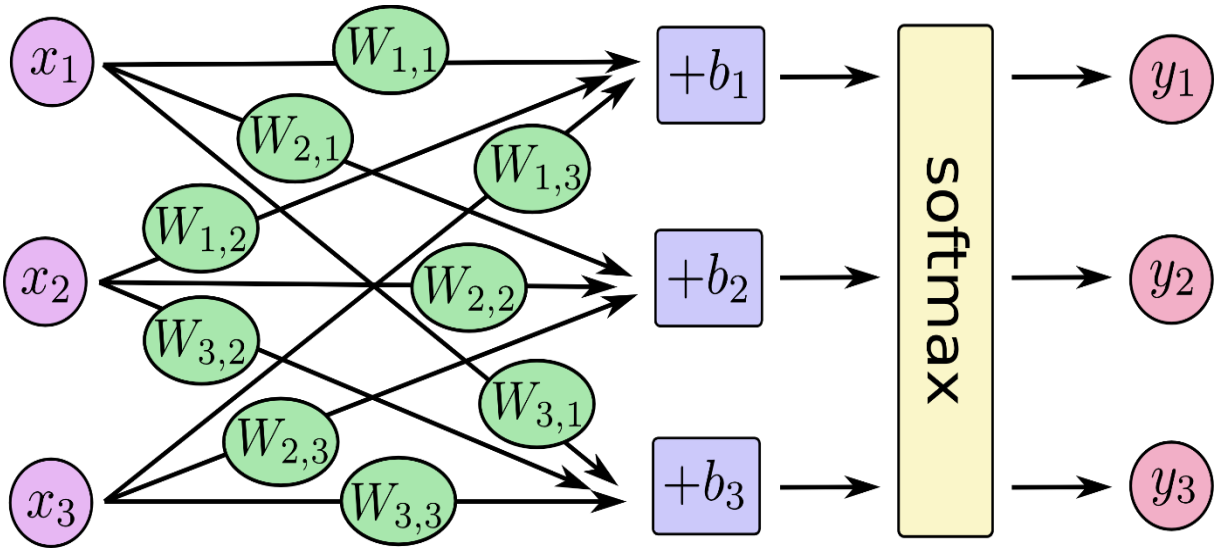



*Figure 2. 12 softmax application*

***4.7.9. Adam Optimization:***

Adam is an adaptive algorithm for optimization that has been directed to the training data in neural networks.
Adam is combining the advantages of two recently popular methods which are: AdaGrad (Duchi, Bartlett, & Wainwright, 2012)which works well with sparse gradients, and RMSprop (Tieleman & Hinton, 2012), which works well in on-line and non-stationary settings(Kingma & Ba, 2015)
Adam optimization is the most supporting algorithms for which concerns model performance improvements by reducing the cost function. Some notable advantages of Adam are:

The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to a diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters(Kingma & Ba, 2015).

## 5. Network Architectures

Each type of network is meant for a specific input modality, which in turn encodes assumptions about the structure of the data. The given architecture will work on a specific problem depending on the match between the structure of data and the assumptions of the network architecture.

### 5.1. VGG:

VGG visual geometry group network is a type of convolutional neural network that is used widely for image classification and has proven to be a very effective method. (Muhammad, Wang, Chattha, & Ali, 2018)
The oxford team of researcher's who developed this model and made the structure and weights available online. (Sai Bharadwaj Reddy & Sujitha Juliet, 2019)
The number after the name represents the number of layers that the architecture is composed of. like VGG-16 and VGG-19. VGG-19 has nineteen (19) layers while VGG-16 has sixteen (16) layers, thirteen (13) of them are convolutional layers followed by three (3) fully connected layers.
In VGG16, 224x224 RGB images are passed through 5 blocks of convolutional layers where each block is composed of increasing numbers of 3x3 filters. The stride is fixed to 1 while the convolutional layer inputs are padded such that the spatial resolution is preserved after convolution (i.e. the padding is 1 pixel for 33 filters). The blocks are separated by max-pooling layers. Max-pooling is performed over 22 windows with stride 2. The 5 blocks of convolutional layers are followed by three fully-connected (FC) layers. The final layer is a soft-max layer that outputs class probabilities.(Tindall, Luong, & Saad, 2015)
Figure 2.13 shows the VGG-16 and the VGG-19 architectures.

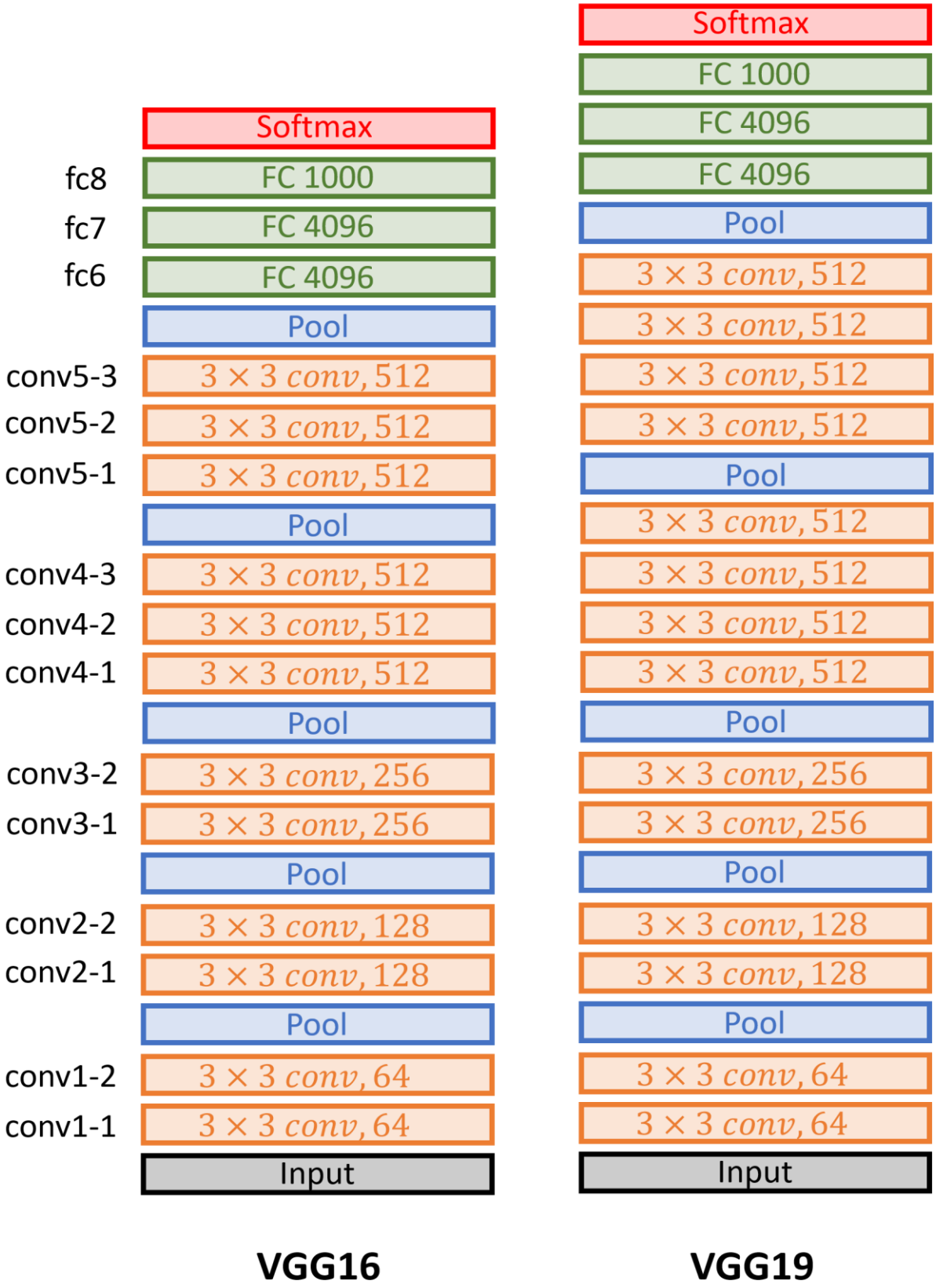

Softmax
FC 1000
FC 4096
FC 4096
Pool
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
Pool
3 × 3 conv, 512
3 × 3 conv, 512
3 × 3 conv, 512
Pool
3 × 3 conv, 256
3 × 3 conv, 256
Pool
3 × 3 conv, 128
3 × 3 conv, 128
Pool
3 × 3 conv, 64
3 × 3 conv, 64
Input
fc8
fc7
fc6
conv5-3
conv5-2
conv5-1
conv4-3
conv4-2
conv4-1
conv3-2
conv3-1
conv2-2
conv2-1
conv1-2
conv1-1
VGG16
VGG19

*Figure 2. 13 VGG16 and the VGG19 architectures.*

**5.2. ResNet:**

ResNet stands for Residual Network, ResNet uses skip connection to propagate information over layers allowing Data collectors to build deeper networks. The number after the name represents the number of layers that the model contains, like ResNet-50 that is composed of 50 layers.

Figure 2.14 represents ResNet-50 architecture.

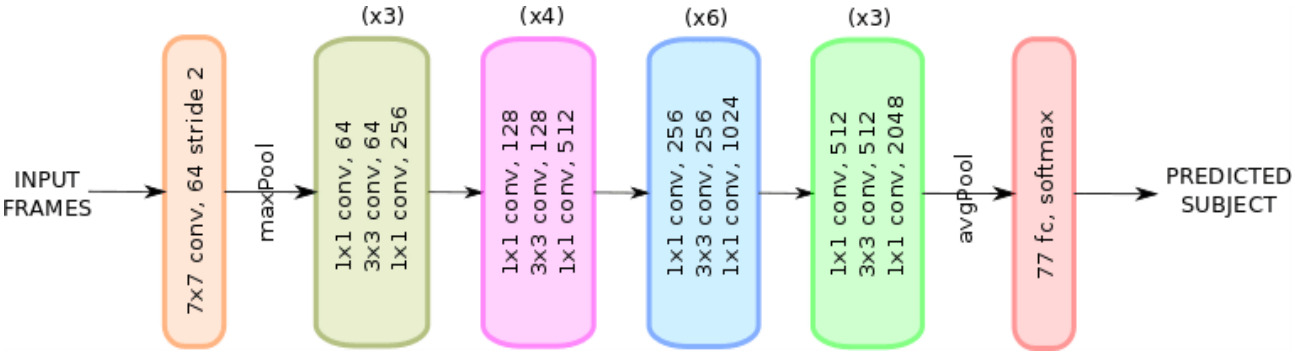



*Figure 2. 14 ResNet architecture.*

So there are (1+ 16 * 3 =49 Convolutional layers in figure 2.14). Then, a fully connected layer (FC) and the softmax classifier are added to ResNet-50 to fit the class labels of fault diagnosis dataset.(Wen, Li, & Gao, 2019).

**5.3. AlexNet:**

AlexNet is a Convolutional Neural Network that was developed by Alex Krizhevsky(Alex Krizhevsky, Sutskever, & Geoffrey E. Hinton, 2012). The AlexNet architecture contains eight learned layers, five convolutional layers, and three fully connected layers. The last fully-connected layer connects to 1000 classes and the rest of the network is considered as a feature extractor. AlexNet can yield 4096-dimensional feature vector for each image, which contains the activations of the hidden layer immediately prior the output layer (Almisreb, Jamil, & Din, 2018). Figure 2.15 shows AlexNet architecture.

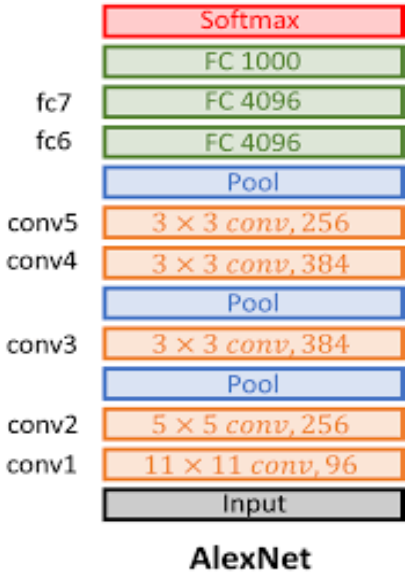



*Figure 2. 15 AlexNet Architecture*

### 5.4. GoogleNEt Architecture:

GoogleNet is type of pretrained convolutional neural network with depth of 22 layers It has two convolutional layers, two pooling layers and nine inception modules, and a fully connected layer. Again, an inception module has six convolution layers and a pooling layer. The module consists of filters of sizes 1x1, 3x3 and 5x5 in its architecture(Yang et al., 2018). Figure 2.16 shows GoogleNet architecture.

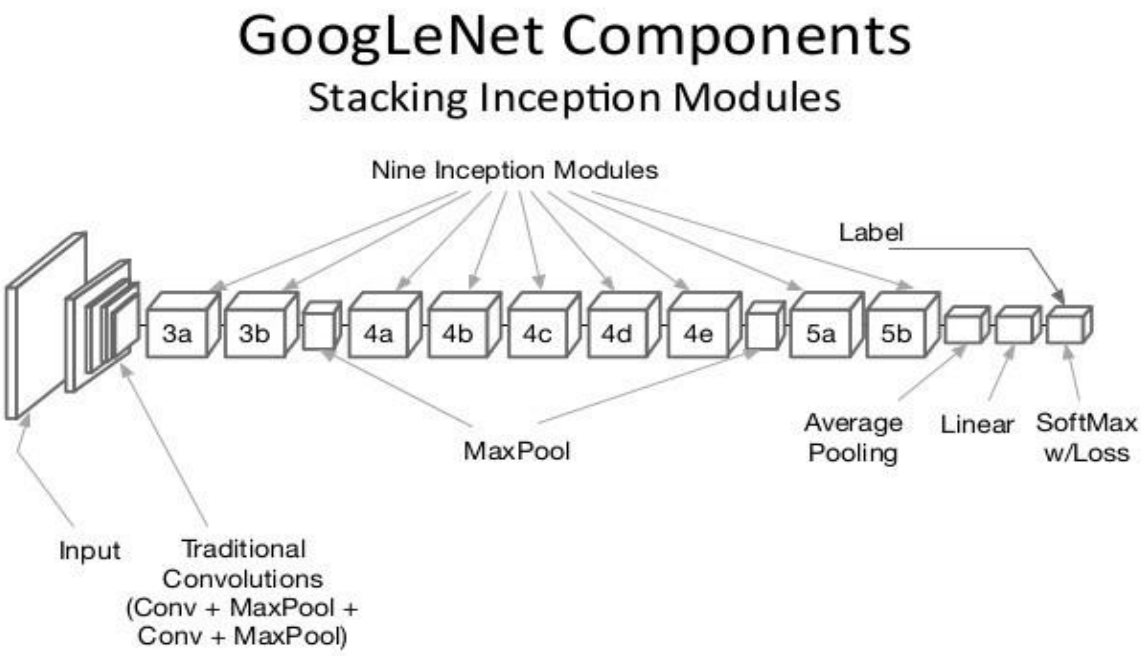



*Figure 2. 16 GoogleNet architecture.*

## 6. Data Processing:

### 6.1 Data pre-processing:

Data pre-processing is a data mining technique and the most important step of machine learning that involves transforming raw data into an understandable format before feeding it to the algorithm. Figure 2.17 shows the data processing step in deep learning.

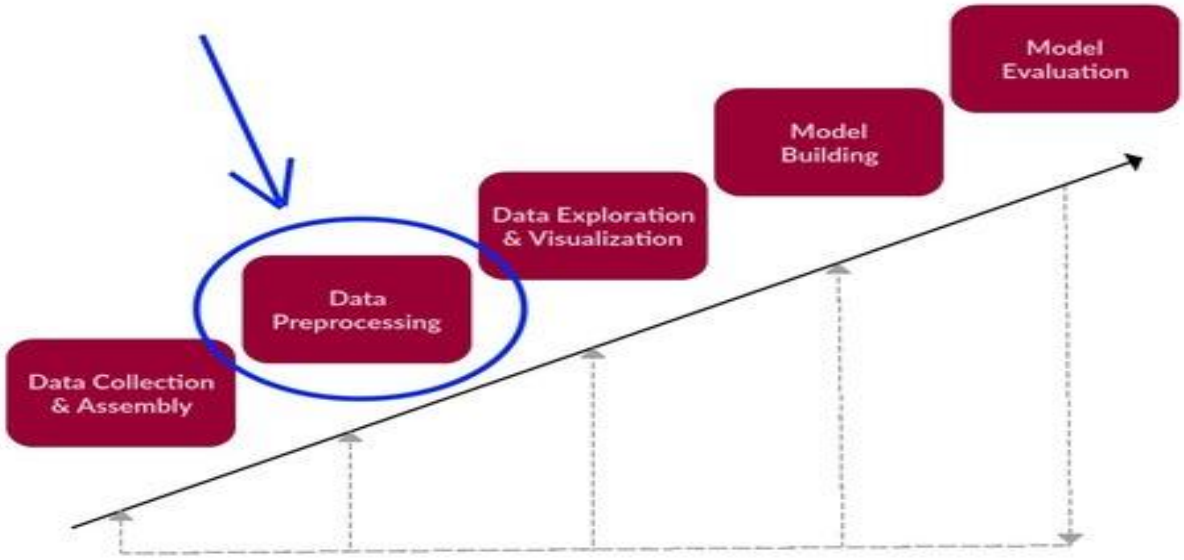



*Figure 2. 17 data preprocessing step in deep learning*

### 6.2. Normalization:

Data normalization is a technique that is used for data preparation in machine learning as well as deep learning. Normalizing data reduce the number of training epochs required to train deep networks and accelerate the training of deep learning neural networks.(Brownlee, 2019)
Figure 2.18 shows normalizing biological data.

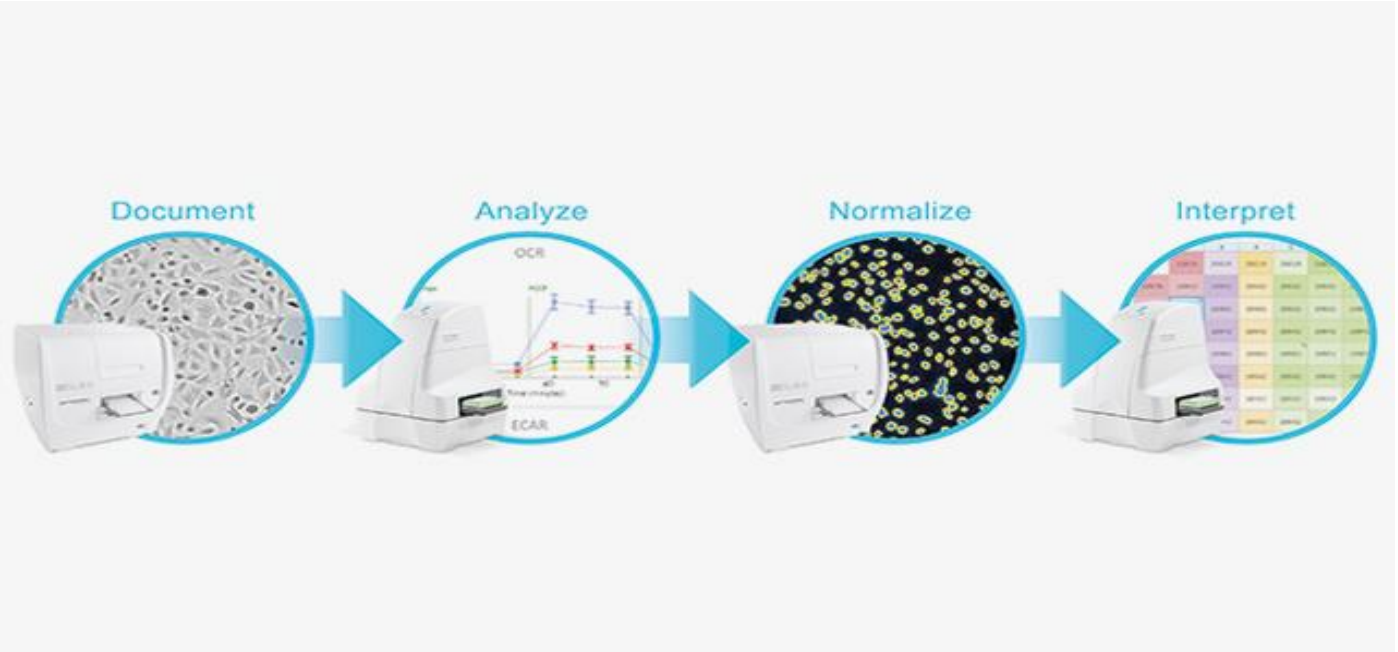



*Figure 2. 18 biological data normalization.*

### 6.3. Data augmentation:

Data augmentation is a strategy used for image classification as well as creating modified versions of images in the dataset. Even when the data is of lower quality, algorithms can perform better, as long as useful data can be extracted by the model from the original data set. For example, text-to-speech and text-based models have improved significantly due to the release of a trillion-word corpus by Google. (Brannon, 2009). Figure 2.19 presents the data augmentation basics and some of its content.

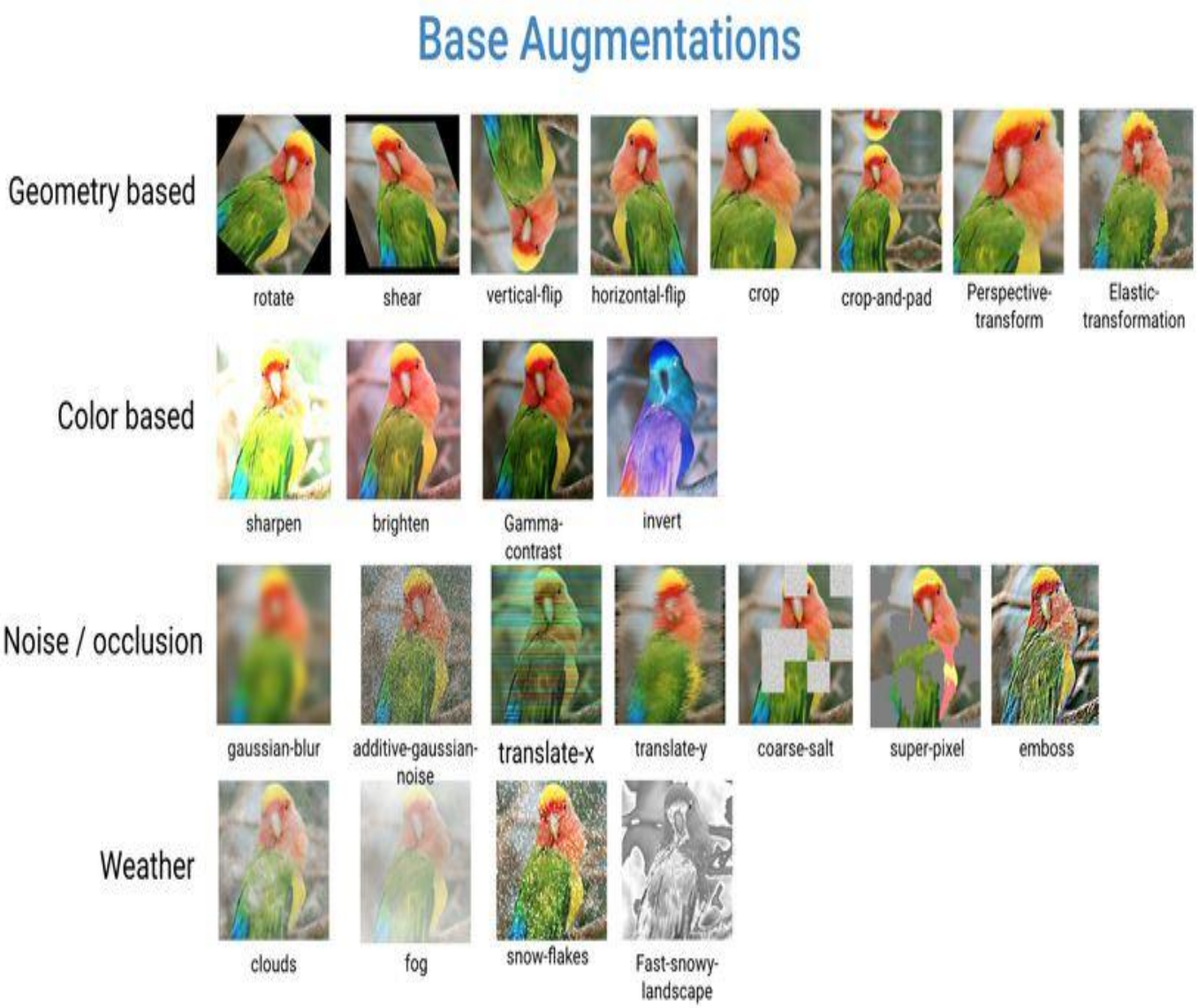



*Figure 2. 19 data augmentation basics*(Avatar & B, 2020).

## 7. Feature extraction:

Feature extraction techniques are applied to the sub-images just obtained after the pre-processing stage. This phase is of high importance to the extraction of useful data aimed for correct classification. Two different techniques have been proposed for the classification block, a convolutional neural network previously mentioned and the second is support vector machine (SVM).

Support Vector Machines (SVM) are a powerful machine learning method for both regression and classification problems. (Huang, Wang, & Chung, 2019).

## 8. Model Evaluation:

Evaluation of the model following the training is an important aspect that should be considered in every machine learning pipeline. Once operational, precision verification is advised.

There are several criteria that can be used to evaluate the model, for example (precision, F-measure, sensitivity, specificity etc…).

**Accuracy** = (TP+TN) / (TP+FN+FP+TN)

**Sensitivity** OR **Recall** =TP / (TP+FN)

**Specificity**=TN / (TN+FP)

**Precision**=TP/ (TP+FP)

**F1-Score**= 2* (Precision * Recall) / (Precision + Recall)

* **TP**: the number of parasites which has been detected parasitic correctly by classifier.

* **FP**: the number of parasites which has been detected parasitic wrongly by classifier.

* **TN**: the number of non-parasites which has been detected non-parasitic correctly by classifier.

* **FN**: the number of non-parasites which has been detected non-parasitic wrongly by classifier.

## 9. Application of deep learning:

There is a variety of application in deep learning such as speech recognition, image recognition and natural language processing (NLP).
Convolutional Neural Networks (CNN) gives satisfying results in image recognition problems, whereas Long Short-Term Memory (LSTM) perform well in speech recognition (Sundermeyer, Schlüter, & Ney, 2012). and others are mentioned in the following Figure 2.20.

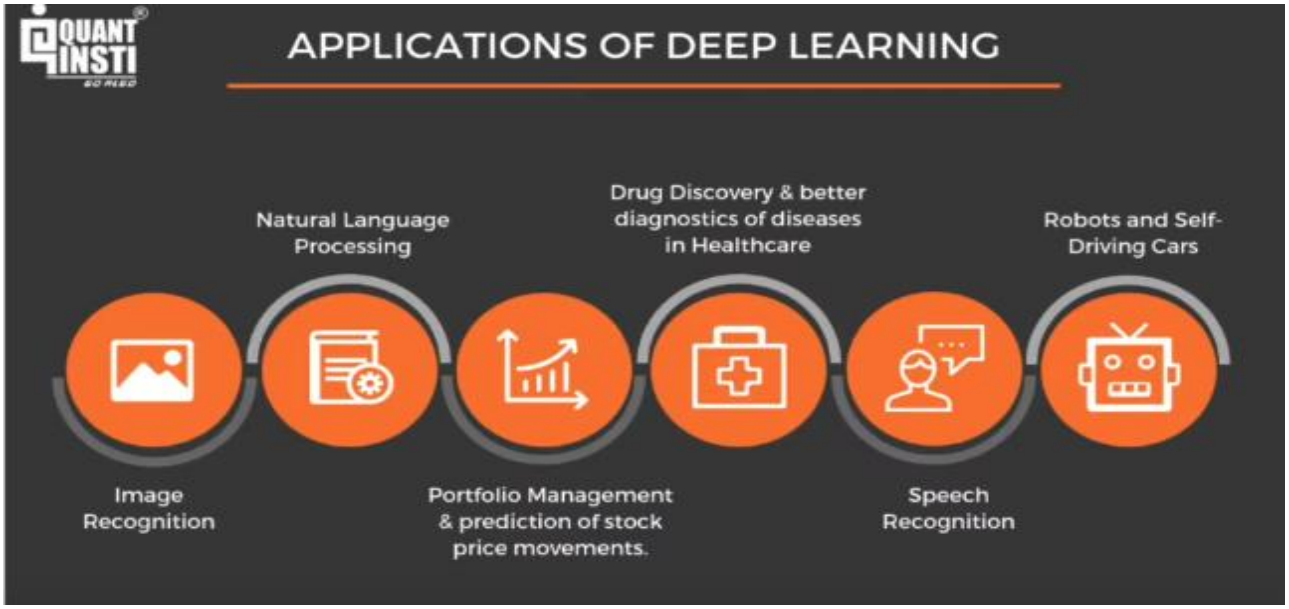



*Figure 2. 20 deep learning application.*

## 10. Previous Studies:

To improve the diagnosis of Malaria, researchers have conducted a considerate amount of research in order to use artificial intelligence.

Errors, mistakes and uncertainties are inevitable when using microscopes, therefore operators and actuators must be highly experienced. Intelligence systems have become an important aspect that needs to be used we use to the growth and survival of the healthcare sector.

Several related works have shown that most of these cases occurred in the African region and remote areas. (Poostchi, Silamut, Maude, Jaeger, & Thoma, 2018) .

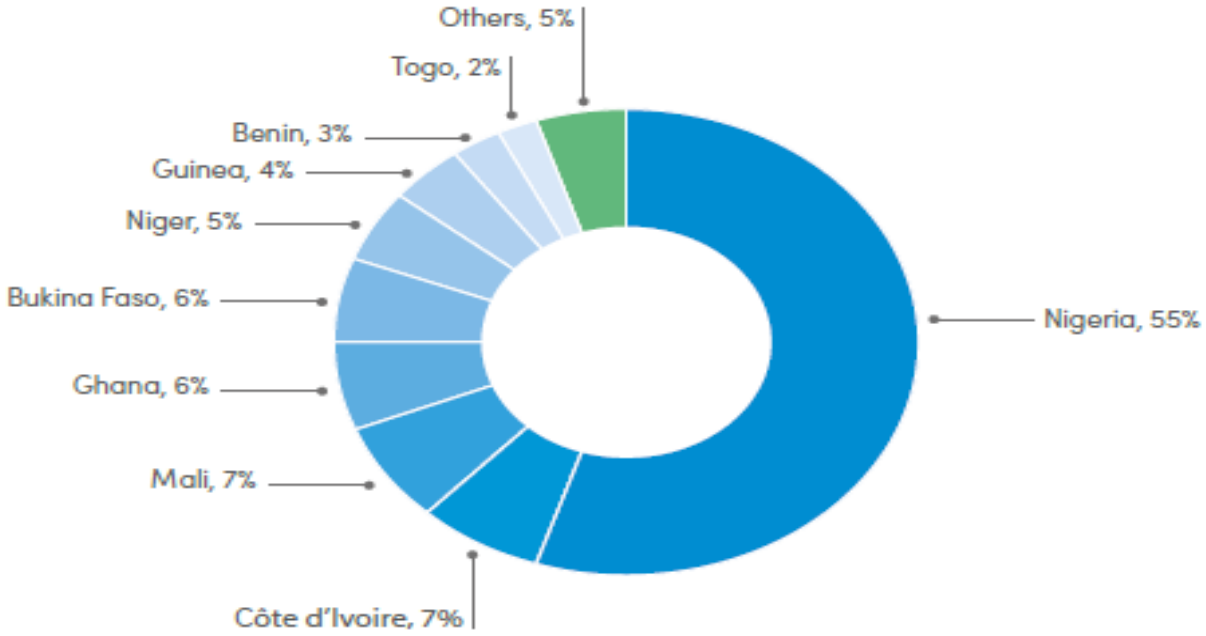



*Figure 2. 21 West Africa shows cases of malaria ("world malaria report," 2016).*

However, among communities and within, there are efforts made for the control of the spread of Malaria. In its work, (Elmardi et al., 2009) surveillance and adaptive strategies for environmental management have been reported with a high level of performance. These strategies reduce the mortality and Malaria cases by 70% to 95% within 3-5 years.

However, using deep learning has become one of the effective ways to diagnose the case of Malaria.

At present, researchers across the world have begun to apply DL tools and obtain promising results in a wide variety of medical image
analyses/understanding tasks(Rajaraman et al., 2018). Previous studies reveal important ways to detect Malaria using deep learning methods.

(Bibin, Nair, & Punitha, 2017) with a dataset of 4100 cells, training and tests were randomized. The authors used Deep Belief Network (DBN) toward Malaria parasite detection in peripheral blood smear images and proposed 6-layer of DBN, in the classification of the dataset mentioned before the authors reported 96,35 % accuracy, 89,66% F-score, 97,60 of sensitivity and 95,92 of specificity.
(Kanaa, Tchiotsop, Ele, Tonye, & Belon, 2016) proposed a new methodology by using a two-tone adaptive median filter and sauvola segmentation to quantify the parasitemia in Plasmodium falciparum-infected blood smears.

In this method, the analysis was performed using supervised classification on two databases of 100X and 40X microscopic images originating from real-time
collection of patients' blood of some Cameroon's laboratories and their result was better on 40X magnified database with accuracy at 96.71%, a sensitivity at 81.58% and a specificity at 97.11%.

(Imran Razzak & Informatics, 2015) used a Rao's method and Bounding Box for parasite segmentation and used Artificial Neural Network ANN as a classifier for malaria classification. In another study (Dong et al., 2017) compared the performance of Support Vector Machine SVM and three types of Convolutional Neural Network CNN including the LeNet, AlexNet and GoogLeNet toward classifying parasitized and uninfected cells. The author split the data randomly into train/test sets and 25% of the training images were randomly selected to validate the models. Their result reported that all these deep convolution neural networks achieved classification accuracies of over 95% higher than the accuracy of about 92% attainable by using the support vector machine method. table 2.1 shows the testing results of the authors.

| | | **Ground Truth** | | |
|---|---|---|---|---|
| | | **Positive** | **Negative** | **Accuracy** |
| **LeNet-5** | Positive | 493 | 25 | 96.18% |
| | Negative | 24 | 741 | |
| **AlexNet** | Positive | 502 | 39 | 95.79% |
| | Negative | 15 | 727 | |
| **GoogleNet** | Positive | 503 | 10 | 98.13% |
| | Negative | 14 | 756 | |
| **SVM** | Positive | 500 | 90 | 91.66% |
| | Negative | 17 | 676 | |

*Table 2. 1 testing results of*(Dong et al., 2017)

(Gopakumar, Swetha, Sai Siva, & Sai Subrahmanyam, 2018) used for the detection of a custom-designed Convolutional Neural Network CNN operating on a focus stack of images. The authors used a customized portable slide scanner and off-the-shelf components for data acquisition and proposed a two-level segmentation strategy and demonstrated sensitivity and specificity of 97.06% and 98.50% respectively. Table 2.2 shows the performance of proposed algorithms by the authors comparing with other methods.

| **Metric** | **Method – A** | **Method – B** | **Method – C** |
|---|---|---|---|
| **Sensitivity** | 92.95% | 96.64% | 97.06% |
| **Specificity** | 93.82% | 98.27% | 98.50% |
| **MCC** | 0.4430 | 0.7036 | 0.7305 |

*Table 2. 2 Sensitivity, Specificity, and MCC: A) SVM on Features and CNN on B) Patches C) Focus Stack*

(Liang, Powell, Ersoy, & Poostchi, 2016) proposed 16-layer Convolutional Neural Network CNN to automatically classify single cells in thin blood smears on standard microscope slides as either infected or uninfected. The authors compared the proposed model with the pre-trained models AlexNet and Support Vector Machine SVM classifier. The study reported that the custom model was more accurate than the pre-trained model with accuracy at 97.37%. Table 2.3 shows the performance proposed model and transfer learning.

| **Measure** | **CNN Model** | **Transfer Learning** |
|---|---|---|
| **Accuracy** | 97.37% | 91.99% |
| **Sensitivity** | 96.99% | 89.00% |
| **Specificity** | 97.75% | 94.98% |
| **Precision** | 97.73% | 95.12% |
| **F1 Score** | 97.36% | 90.24% |
| ***Matthews correlation coefficient*** | 94.75% | 85.25% |

*Table 2. 3 the results of the two ten-fold cross-validations(Liang et al., 2016).*

The work done by (Sai Bharadwaj Reddy & Sujitha Juliet, 2019) uses transfer learning with ResNet50 for malaria cell-image classification. The authors proposed 2 layers, pre-trained ResNet layer and dense layer and as input for the model used RGB image with a size of 224*224 pixels. The accuracy and loss performed by the authors of both training and validation data are tabulated in Table 2.4.

| **Metrics** | **Value** |
|---|---|
| **Training accuracy** | 95.91% |
| **Validation accuracy** | 95.4% |
| **Training loss** | 0.1134 |
| **Validation loss** | 0.1301 |

*Table 2. 4 model performance.*

(Vijayalakshmi & Rajesh Kanna, 2019) used transfer learning approach by unifying the existing Visual Geometry Group (VGG) network and Support Vector Machine (SVM). This consolidation was implemented by using the "Train top layers and freeze out the rest of the layers" strategy, the frozen layers were replaced with SVM. The authors compared the two VGG architecture, the obtained accuracy of VGG without SVM was 87.84%, 91.47% for VGG16, VGG19 respectively.
Tables 2.5 indicate details performance evaluation, sensitivity, specificity, precision, F-score and the accuracy obtained for VGG16-SVM and VGG19-SVM.

| Method | Sensitivity | Specificity | Precision | F-Score | Accuracy |
|---|---|---|---|---|---|
| **VGG16-SVM** | 89.80% | 88.81% | 84.47% | 87.05% | 89.21% |
| **VGG19-SVM** | 93.44% | 92.92% | 89.95% | 91.66% | 93.13% |

*Table 2. 5 Performance evaluation of VGG16-SVM & VGG19-SVM*

(Vishnu, 2016) performed cross-validation and grid-search for finding the Support Vector Machine SVM parameters, the trained SVM is used to classify infected or uninfected cells. The authors got accuracy at 95.5% in the result.

(Devi et al., 2018) proposed a technique where features extraction based on the histogram and for classification the authors used four classifiers which are Artificial neural network (ANN), naïve Bayes, Support Vector Machine SVM, K nearest Neighbor K-NN. The authors compared the result obtained by these classifiers and concluded that ANN provides a higher classification rate. Table 2.6 shows 4-fold cross-validation results for the ANN classifier.

| Exp | Sn (%) | Sp (%) | Acc (%) | F-Score (%) |
|---|---|---|---|---|
| **Subset1** | 92 | 96.88 | 96.31 | 85.16 |
| **Subset2** | 96 | 96.88 | 96.77 | 87.12 |
| **Subset3** | 88 | 95.34 | 94.50 | 78.52 |
| **Subset4** | 96 | 97.93 | 979.71 | 90.52 |
| **Average** | 93 | 96.75 | 96.32 | 85.32 |

*Table 2. 6 4-fold cross-validation results for ANN classifier.*

In (Malihi et al., 2013) work used five classifiers **support** vector machines (SVM), nearest mean (NM), K nearest neighbors (KNN), 1-NN, and Fisher. In this research K-nearest, neighbors' classifier had the best accuracy, which was 91%. Table 2.7 shows the performance of the five classifiers.

| ACC | Classifier | | | | |
|---|---|---|---|---|---|
| | **SVM** | **K-NN** | **I-NN** | **NM** | **Fisher** |
| **The research method** | 90% | 91% | 87% | 70% | 84% |

*Table 2. 7 The performance of the five classifiers.*

(R & Jaeger.) proposed 13-layers convolutional Neural Network (CNN) model for improving Malaria cell classification, the dataset was randomly split into 70% for training and 30% for validation with a size of 32*32 pixels for each image. The model achieved as a result of 98.61% classification accuracy.

(Mohanty, Pattanaik, & Swarnkar, 2019)compared the Auto encoder and Self organizing Maps deep learning techniques for automatic malaria identification from blood smear images and his results reveal that the deep-learning-based Auto encoder technique is better than the Self-organizing Maps technique in terms of accuracy of 87.5%.

## Conclusion

This chapter presents the fundamental notion of two concepts:
Deep Learning and Malaria detection in human cells.
First and foremost, it contains the different types of Deep Learning, how they work and their applications and different architectures.
Therefore, Malaria Definition, their techniques that can be used to detect malaria in human cells.

# Chapter 3: proposed approach base on DL for Malaria Detection.

## Introduction:

Based on the prior studies of Malaria detection using deep learning. We propose a novel architecture for Malaria detection in human cell image. AutoEncoder, Convolutional Neural Network (CNN) as one of the architectures of deep neural network techniques gives impressive results in many domains of computer vision. However, most of the time, the use of pretrained model like (VGG16, ResNet50, AlexNet and GoogleNet) give high performance. In the next section, we'll be going in details to present our proposed approach and compare between them based on results.

## 1.Methodology:

According to the earlier studies mentioned in the previous chapter. Our objective is to build a model that can overcome the existing models in terms of accuracy and precision.

First and foremost, we start our models by importing the dataset which is obtained from the official website of (National Library of Medicine, 2019).

The dataset has two folders for the infected and uninfected cells.

For the second step, the dataset should be ready to enter to the input. Data preprocessing is one of the necessary steps to apply before entering the dataset to training by cleaning it, ignore the duplicated images and others techniques which can help in reducing under fitting and over fitting.

For us, for the infected folder we do some preprocessing to the images and we concentrate on the infection itself by coloring it and letting the rest of the cell space with Blanc.

And for the uninfected folder, we color all cell spaces with Blanc.

Next, we resize the images, and we put it in the shape of (125x125x3) where this shape is very suitable for obtaining good results due to the large number of parameters that are processed by each model.

Finally, we shuffle the two folders of cells infected and uninfected. And we split the dataset into three sets training, testing and validation.

After that and for the last step, we enter this input of dataset to each model, Convolutional Neural Network (CNN), pretrained model (VGG16) and AutoEncoder model and train each model by this shuffled dataset.

To illustrate more Figure 3.1 shows steps of our work from the importing data to getting results.

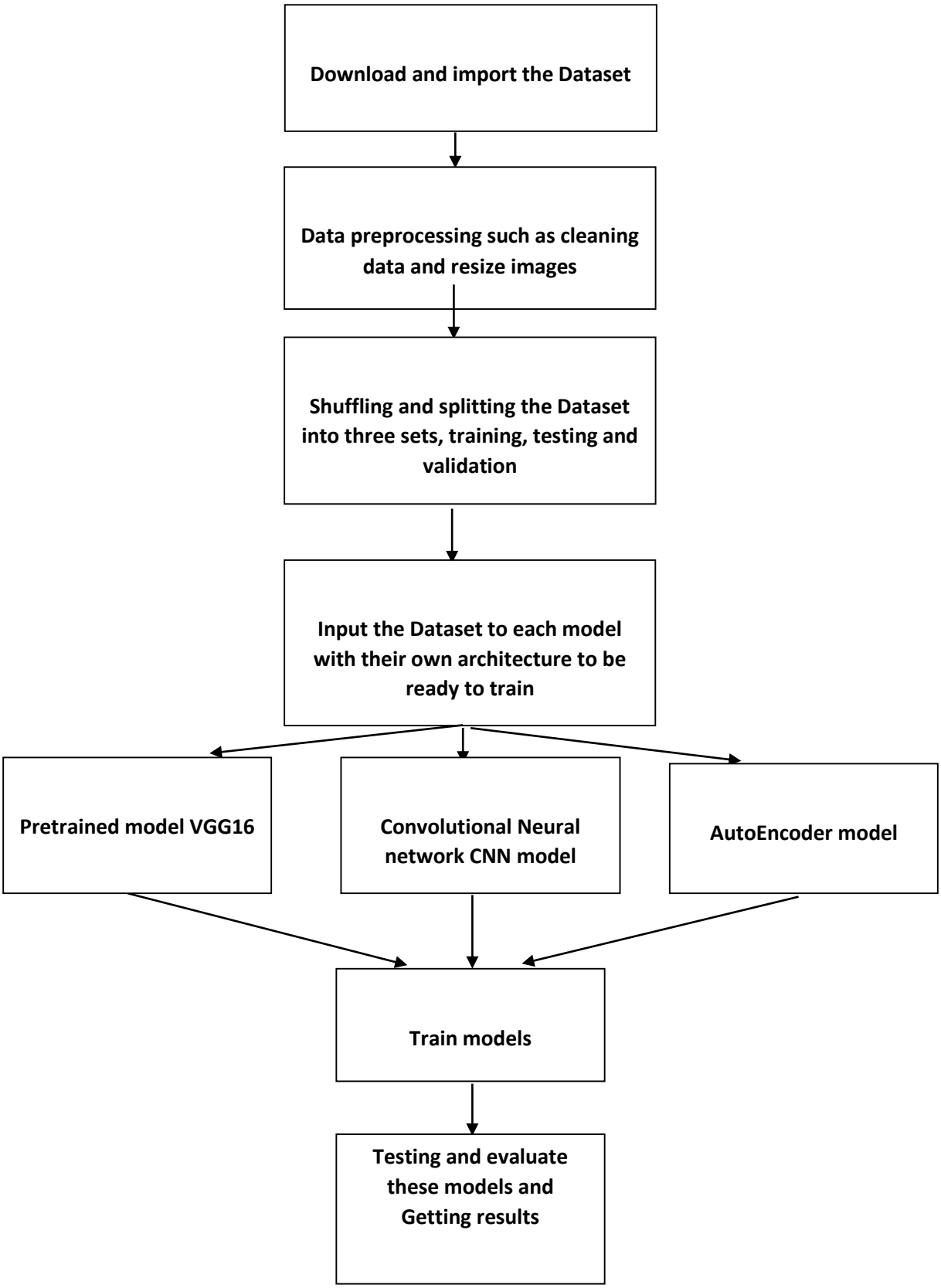



*Figure 3. 1 steps of our work*

## 2. System architecture:

the system architectures contain three proposed models which are Convolutional Neural Network CNN, AutoEncoder and pretrained model Visual Geometry Group (VGG16) by applying some modifications to their architecture.

Each model from the proposed approaches contains set of layers, in each layer has lot of parameters that we manipulate to get best model for us.

For the first proposed model of the Convolutional Neural Network CNN we create six (6) convolutional layers, two of them are not followed by anything, two are followed by two MaxPooling layers and two are followed by AvgPooling layers. And three Fully connected layers and one for classification with sigmoid activation function.

For the second proposed model which is based on the pretrained model Visual Geometry Group VGG16 where we do some transformation on the architecture and we add some layers and froze others to get the best classification for our work.

The third proposed model of the AutoEncoder, we use five convolutional layers, each layer from the two first convolutional layers is followed by MaxPooling layer which is the encoder part. And each layer from the two last layers is followed by UpSampling layer which is the decoder part, and for the convolutional layer in the middle is not followed by anything. After that we use three Fully Connected layers and one dense layer for the classification with sigmoid activation function.

We will see the reason of choosing those architectures and those type of parameters with all details in the following points.

## 3.Proposed approaches:

### 3.1 Convolutional Neural Network CNN:

As mentioned in the previous chapter. Convolutional neural network (CNN) is a part of a deep neural network that contains one or more convolutional layers that automatically recognize, divide, and bind images and classifications as well as specialize in processing data that contains a network like topology such as an image, contains also MaxPooling, AveragePooling, Dropout, Batch Normalization and fully connected layers with the activation functions.
In this proposed approach, our input is in shape (W, H, C), where W is the width, H is the height, where both height and width represent the resolution of images, and C is the number of channels. In addition, because we used colored images (RGB images) so the channel should be equal to three C=3. So the input is an image with shape of (125,125,3).

We start our model with an image of shape (125x125), and we use 16 filters of shape (3x3) with no padding (padding=valid) and stride s=1, we also use kernel_initializer with seed =2 to seed the random generator and give as output an image with shape of (123,123,16).

After the input layer we used the ReLU activation function, it is use in almost all the convolutional neural network and is the most use activation function in the world for the reason that ReLU is half rectified. ReLU replace the negative values by zero and keep the

same values when equal or above of zero that reduces the computational cost for training of the network.

We use this activation function for all the network convolutional, maxpooling and dense layers except for the output layer.

After the activation function we use dropout layer to drop randomly (0.2) of the output of the convolutional layer. The values of the dropout layers are between zero and one (0-1). And we put the value of (0.2) in order not to drop large number of nodes which gave us desired results.

The reason for using dropout layers is to prevent Neural Networks from overfitting(Garbin, Zhu, & Marques, 2020).

For all the network, both convolutional and maxpooling layers we use valid padding which means that we do not keep the same shape of the image between layers and do not use padding.

Let it be known that there are two kind of padding valid and same, for us, the reason for choosing valid padding is to reduce the dimension of the input image size and therefore reduce the number of nodes in the output.

For the second convolutional layer, we use (32) filter with shape of (4x4) and we kept the same value of strides, padding and kernel_initializer with a ReLU activation function and dropout layer which reduce the size to (60,60,32) followed by maxpooling operation with shape of (2x2) and stride=2 which also reduce the size from (60,60,32) to (30,30,32).

After that, we use the third convolutional layers, however, we raise the number of filter to (64) with shape of(3x3) while we keep the same value of padding and stride. This reduces the width and height while increases the depth into (14,14,64).
We use also fourth convolutional layer followed by maxpooling operation, for the convolutional layer we use (128) filter of size (3x3), stride=1 and the same padding and kernel_initializer that reduce the size to (12,12,128). And for the maxpooling layer we use shape of (2x2), stride=2 and padding=Valid which reduces the size into (6,6,128).
Similar to the fourth convolutional layers, we create fifth convolutional layers with the change of number of filter to 256 and stride =2 that gives us a result of (2,2,256) followed by the same AveragePooling layers in the fourth layer which reduces the size into (1,1,256).
Finally, we put the last convolutional layer with 512 filter of shape (1x1), stride =2, padding =valid and the same kernel-initializer followed by AveragePooling layer of shape (1x1) and the same stride and padding.
We choose AveragePooling for the last layer with shape of (1x1) for the simple reason of using (2x2) as shape have no meaning with a vector with one dimension (1D). so we use AveragePooling in order not to lose important information.
The reason of choosing these small filters mentioned before (1x1, 3x3 and 4x4) is because it allows to build more expressive features while using less parameters. for three stacked convolutional layers with 3x3 filter has the same view as one convolutional layer with 7x7 filter, and less parameters, we have: an image with size (32x32)

(3*3*32) *3 = 864…… (1)
(7*7*32) =1568……. (2)
And the dimension will be (26x26) for all the first and the second filter.

After all these convolutional and maxpooling layers we use three fully connected layers with an input of the first fully connected layers which is equal to the product of the last pooling layer dimension which was (1,1,512) in which also means (512) input. First and foremost, we used a flatten to convert the values from two dimension 2D matrix of features to one dimension 1D and which is a vector because fully connected layers accepts only vectors. Just after we put three Dense layers. The three dense layers have respectively 128, 64 and 16 nodes, we choose these numbers with power of two. Because they perform well in the neural network and help the network to learn faster. And for the last dense layer contains one (1) node which is the output that do the classification and indicates if the input image is infected or uninfected.
We use for prediction of output classes a sigmoid activation function instead of softmax activation function. We choose this activation function because we have binary classification.

For optimization we use Adaptive Moment Estimation (Adam) optimization technique because it's faster to converge into minimum and gives better results than Stochastic Gradient Descent (SGD).
Figure 3.2 shows the summary of our architecture and the total parameters, trainable and non-trainable parameters from them.

Model: "model"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_2 (InputLayer) | [(None, 125, 125, 3)] | 0 |
| conv2d_5 (Conv2D) | (None, 123, 123, 16) | 448 |
| activation_5 (Activation) | (None, 123, 123, 16) | 0 |
| dropout_4 (Dropout) | (None, 123, 123, 16) | 0 |
| conv2d_6 (Conv2D) | (None, 60, 60, 32) | 8224 |
| activation_6 (Activation) | (None, 60, 60, 32) | 0 |
| max_pooling2d_2 (MaxPooling2 | (None, 30, 30, 32) | 0 |
| dropout_5 (Dropout) | (None, 30, 30, 32) | 0 |
| conv2d_7 (Conv2D) | (None, 14, 14, 64) | 18496 |
| activation_7 (Activation) | (None, 14, 14, 64) | 0 |
| dropout_6 (Dropout) | (None, 14, 14, 64) | 0 |
| conv2d_8 (Conv2D) | (None, 12, 12, 128) | 73856 |
| activation_8 (Activation) | (None, 12, 12, 128) | 0 |
| max_pooling2d_3 (MaxPooling2 | (None, 6, 6, 128) | 0 |

| | | |
|---|---|---|
| dropout_7 (Dropout) | (None, 6, 6, 128) | 0 |
| conv2d_9 (Conv2D) | (None, 2, 2, 256) | 295168 |
| activation_9 (Activation) | (None, 2, 2, 256) | 0 |
| average_pooling2d_1 (Average | (None, 1, 1, 256) | 0 |
| dropout_8 (Dropout) | (None, 1, 1, 256) | 0 |
| conv2d_10 (Conv2D) | (None, 1, 1, 512) | 131584 |
| activation_10 (Activation) | (None, 1, 1, 512) | 0 |
| average_pooling2d_2 (Average | (None, 1, 1, 512) | 0 |
| dropout_9 (Dropout) | (None, 1, 1, 512) | 0 |
| flatten (Flatten) | (None, 512) | 0 |
| dense (Dense) | (None, 128) | 65664 |
| activation_11 (Activation) | (None, 128) | 0 |
| dropout_10 (Dropout) | (None, 128) | 0 |
| dense_1 (Dense) | (None, 64) | 8256 |
| activation_12 (Activation) | (None, 64) | 0 |
| dropout_11 (Dropout) | (None, 64) | 0 |
| dense_2 (Dense) | (None, 16) | 1040 |
| activation_13 (Activation) | (None, 16) | 0 |
| dropout_12 (Dropout) | (None, 16) | 0 |
| dense_3 (Dense) | (None, 1) | 17 |
| activation_14 (Activation) | (None, 1) | 0 |

Total params: 602,753
Trainable params: 602,753
Non-trainable params: 0

*Figure 3. 2 The summary of our architecture and the total parameters and the trainable parameters from them.*

To show our proposed approach, we illustrate it with drawing representation of all the convolutional, MaxPooling, AveragePooling and Fully connected layers with filter size and number of channels that every layer contains of. Figure 3.3 presents this model.

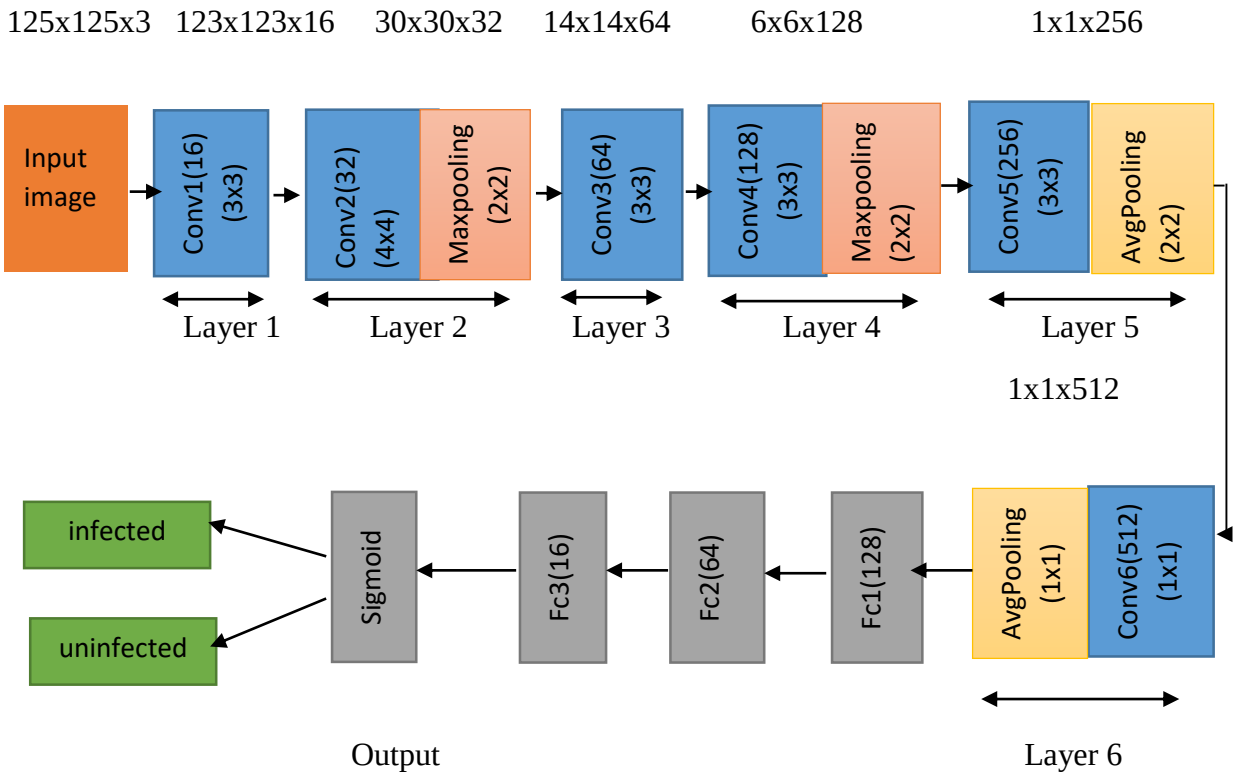



*Figure 3. 3 the architecture of our proposed Convolutional Neural network model.*

**3.2 VGG16 model:**

As mentioned in the previous chapter, VGG16 is a pre-trained model with an input (224x224) RGB images are passed through 5 blocks of convolutional layers. The 5 blocks of convolutional layers are followed by three fully-connected (FC) layers and the final layer is a soft-max layer that outputs class probabilities.(Tindall et al., 2015)
So, we apply certain modifications to the VGG16 architecture and we change some functions that the VGG16 use.

Firstly, as input we use an image size of (125,125,3) and we instantiate the VGG16 network with weights pre-trained on ImageNet and leaving off the Fully Connected layer head.

After leaving off the Fully Connected layer, the VGG16 does not do any sort of classification like the Figure 3.4 shows.

Model: "vgg16"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_21 (InputLayer) | [(None, 125, 125, 3)] | 0 |
| block1_conv1 (Conv2D) | (None, 125, 125, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 125, 125, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 62, 62, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 62, 62, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 62, 62, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 31, 31, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 31, 31, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 31, 31, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 31, 31, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 15, 15, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 15, 15, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 15, 15, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 15, 15, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 7, 7, 512) | 2359808 |
| block5_conv2 (Conv2D) | (None, 7, 7, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, 7, 7, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, 3, 3, 512) | 0 |

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

*Figure 3. 4 The VGG16 network summary without Fully Connected layer.*

From there, we construct a new fully-connected layer head consisting of two dense layer with dropout rate equal to (0.2) after each one, the first and the second dense layers contains 512 and 256 nodes respectively.

A fully connected layer occupies most of the parameters. So, we use dropout to prevent Neural Networks from overfitting(Garbin et al., 2020). Because the dropout drops nodes randomly and that has the effect of reducing the capacity or thinning the network during training.

And we chose the dense layers with values of 512 and 256 for the two dense layers with shape of power of two in place of 4096 for the first and second and 1000 node for the third dense layers, because they perform well in the neural network and helps the network to learn faster and minimize number of parameters and nodes to do binary classification for the output.

For the output, we create a dense layer with one (1) output and sigmoid activation function in place of softmax activation function that the VGG16 works with. We choose this activation function because we have binary classification and the sigmoid function performs well in two-class logistic regression, whereas the softmax function is used for the multiclass logistic regression. Figure 3.5 shows our proposed architecture.

Model: "model"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | [(None, 125, 125, 3)] | 0 |
| block1_conv1 (Conv2D) | (None, 125, 125, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 125, 125, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 62, 62, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 62, 62, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 62, 62, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 31, 31, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 31, 31, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 31, 31, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 31, 31, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 15, 15, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 15, 15, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, 15, 15, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, 15, 15, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 7, 7, 512) | 2359808 |

| | | |
|---|---|---|
| block5_conv2 (Conv2D) | (None, 7, 7, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, 7, 7, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, 3, 3, 512) | 0 |
| flatten (Flatten) | (None, 4608) | 0 |
| dense (Dense) | (None, 512) | 2359808 |
| dropout (Dropout) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 256) | 131328 |
| dropout_1 (Dropout) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 1) | 257 |

Total params: 17,206,081
Trainable params: 16,945,921
Non-trainable params: 260,160

*Figure 3. 5 the proposed vgg16 architecture summary.*

17 million trainable params is entirely too many for a network that is already trained on a better computer than we will have access to– especially at the lower-level features. therefore, we ignore updating those and focus on the classification element by looping over all layers in the base model and freeze them so they will not be updated during the first training process such that only the ('block3_conv1','block4_conv1' and 'block5_conv1') and Fully connected layer heads will be trained. And hence from block3 to the last layer of the VGG16 architectures the model will update layers. The result is a Network that has many trainable parameters which should yield better accuracy. Figure 3.6 shows the frozen and trainable layers of our model.

| Proposed model |
|---|
| block1_conv1 False |
| block1_conv2 False |
| block1_pool False |
| block2_conv1 False |
| block2_conv2 False |
| block2_pool False |

| |
|---|
| block3_conv1 True |
| block3_conv2 True |
| block3_conv3 True |
| block3_pool True |
| block4_conv1 True |
| block4_conv2 True |
| block4_conv3 True |
| block4_pool True |
| block5_conv1 True |
| block5_conv2 True |
| block5_conv3 True |
| block5_pool True |
| flatten_1 True |
| dense_3 True |
| dropout_2 True |
| dense_4 True |
| dropout_3 True |
| dense_5 True |

*Figure 3. 6  the frozen and trainable layers in our proposed model.*

To illustrate more, we plan our proposed approach in the following Figure 3.7 that shows our VGG16 proposed model with all trainable layers and frozen from them and the modification that we add.

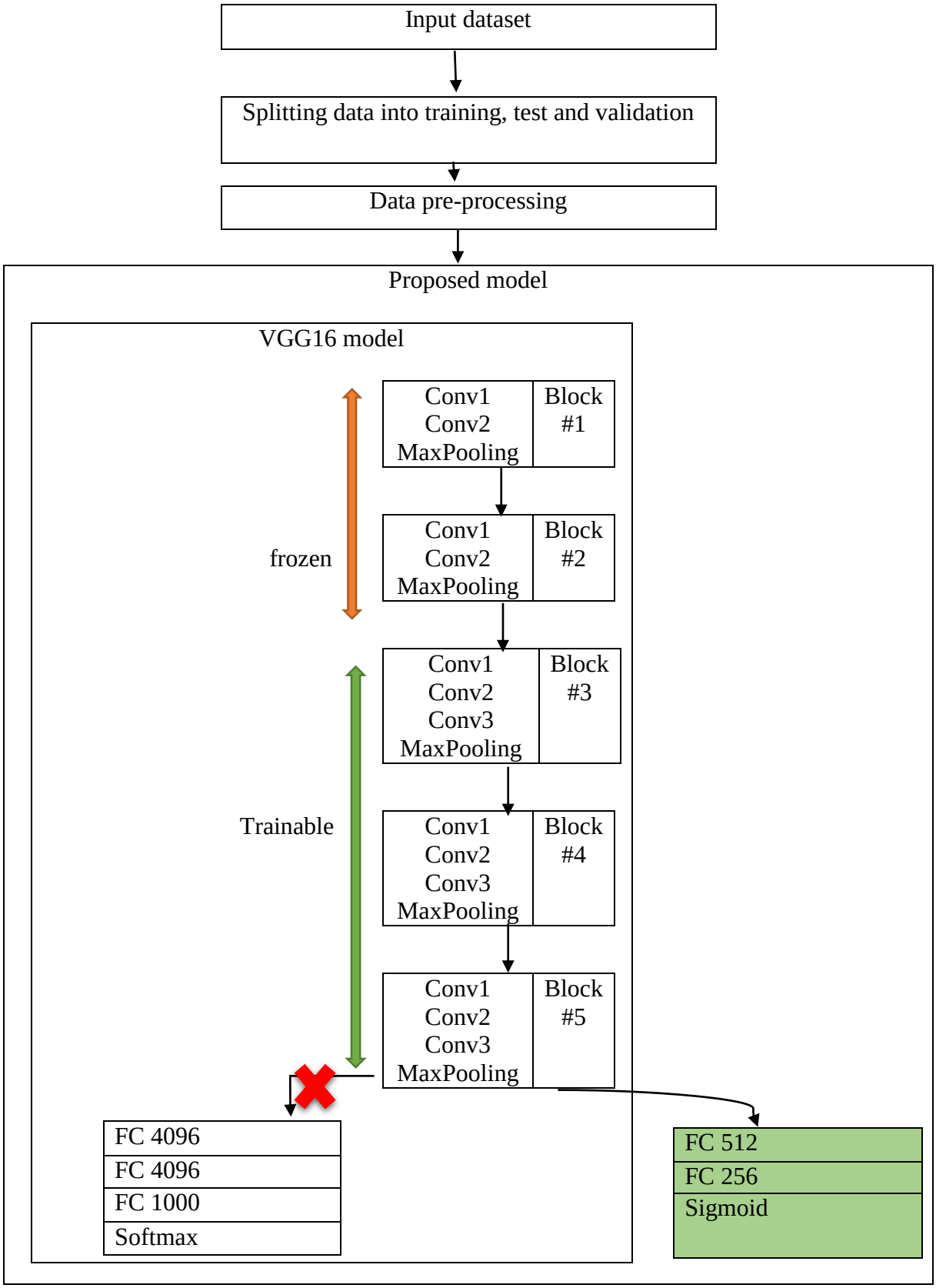



*Figure 3. 7 our VGG16 proposed model.*

### 3.3 AutoEncoder model:

Like the Convolutional Neural Network CNN proposed approach, in this model, our input is in (W, H, C) shape, where W is the width, H is the height and C is the number of channels. In addition, because we use colored images (RGB images) so the channel should be equal to three C=3. So the input is an image with shape of (125,125,3).

As we mentioned before, the AutoEncoder model has two parts, the encoder and decoder.

For the encoder part, we start our proposed model with a convolutional layer with eight (8) filters of shape (3x3) which reduces the size of the image to be (123,123,8)
The size of images is reduced because of using valid padding according to this equation:
**n = (((n + 2p – f) / s) +1) …… (1)**
where s = 1, p = 0, f = 3 and n = 125
n = ((125 + (2) x (0) – 3) / 1) +1)
and therefor;
n = (122 + 1)
n = 123
followed by a 2x2 Max pooling layer which also reduced its size to (61,61,8).
In this case the images size scale down to half in case of using stride value equal to two (2) with applying equation (1).
where s = 2, p = 0, f = 3 and n = 123
n = ((123 + (2) x (0) – 3) / 2) +1)
and therefor;
n = (60 + 1)
n = 61
After the first pooling layer, we use second convolutional layer with (16) filters of shape (3x3) followed by a 2x2 Max pooling layer. This layer reduces the size of image to (29,29,16) because the convolutional layer scales down the size to (59,59,16) and the 2x2 MaxPooling layer to (29,29,16).
And we completed the encoder section with last convolutional layer by changing the number of filters to (32) with (3x3) shape. This reduces the width and height while increases the depth to (27,27,32).
And for the decoder part we start the decoding process with a convolutional layer which is the same as the last convolutional layer in the encoder section. Containing (32) filters of shape (3x3) that also reduce the size of image to (25,25,32).
After this convolutional layer we use 2x2 UpSampling layer which increases the size of image.
UpSampling layers repeat only the rows and columns of the inbound data by specific size. For this case, we have (25 x 25) input data and we want to up simple it by two on rows and two on columns, first we repeat the rows and then repeat the columns to be (50,50,32).

We also use another convolutional layer with (16) filters of shape(3x3) which reduce the size of images to (48x48x16) followed by 2x2 UpSampling layer which increases the width and height of image and decrease the depth, the dimension of image became (96,96,16).

After the encoder and decoder section containing convolutional, maxpooling and UpSampling layers, we use three fully connected layers. Firstly, we use a flatten to convert the values from two dimension 2D matrix of features into a vector that can be fed into a fully connected neural network classifier. The input of the first fully connected layer is equal to the product of the last layer dimension which was (96,96,16) that also means:

96 x 96 x 16=147,456.
So, the input is a vector of 147,456 features.

After the flatten layer we put three Dense layers. The first one has 128 nodes from the output of the flatten layer with ReLU activation function followed by a dropout rate of (0.2). the second and third Dense layers have respectively 64 and 16 nodes and each one of them are followed by ReLU activation function and dropout rate of (0.2).
And we use sigmoid activation function for prediction of the output classes in the last Dense layer which contains one (1) node that indicate if the image was infected or uninfected.

Either in the encoder or decoder part, in every convolutional layer we apply ReLU activation function because ReLU substantially reduces the computational cost for the network's training. Advantages of the ReLU function include fast and efficient computation, scale invariant, and sparsity, and often higher test set accuracy as well(Pradhan & Sameen, 2018.).

Nonetheless, for all the convolutional layers we use valid padding. The reason of choosing valid padding is to reduce the dimension of the input image size in the encoder section and increase it in the decoder section therefore. So, we do not want to preserve the same dimension of the image.

The reason of choosing these small filters mentioned before (3x3) is because it allows to build more expressive features while using less parameters. And with smaller filters size, we get a lower weights and more layers.
In the fully connected layers we apply dropout layers after each dense layer with a rate of (0.2), hence, the reason of choosing this value is to drop 0.2 from the neurons and maintain the majority to reduce the overfitting and beware of failing in the underfitting problem, because dropout is one of the solution that can regularize the over fitting problem. Figure 3.8 shows the summary of our architecture and the total parameters and the trainable parameters.

Model: "model"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | [(None, 125, 125, 3)] | 0 |
| conv2d (Conv2D) | (None, 123, 123, 8) | 224 |
| max_pooling2d (MaxPooling2D) | (None, 61, 61, 8) | 0 |
| conv2d_1 (Conv2D) | (None, 59, 59, 16) | 1168 |
| max_pooling2d_1 (MaxPooling2 | (None, 29, 29, 16) | 0 |
| conv2d_2 (Conv2D) | (None, 27, 27, 32) | 4640 |
| conv2d_3 (Conv2D) | (None, 25, 25, 32) | 9248 |
| up_sampling2d (UpSampling2D) | (None, 50, 50, 32) | 0 |
| conv2d_4 (Conv2D) | (None, 48, 48, 16) | 4624 |

```
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 96, 96, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 96, 96, 16)        0
_________________________________________________________________
flatten (Flatten)            (None, 147456)            0
_________________________________________________________________
dense (Dense)                (None, 128)               18874496
_________________________________________________________________
activation (Activation)      (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
activation_1 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 16)                1040
_________________________________________________________________
activation_2 (Activation)    (None, 16)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 18,903,713
Trainable params: 18,903,713
Non-trainable params: 0
_________________________________________________________________
```

*Figure 3. 8 the summary of our architecture and the total parameters and the trainable parameters from them.*

And figure 3.9 shows our AutoEncoder architecture.

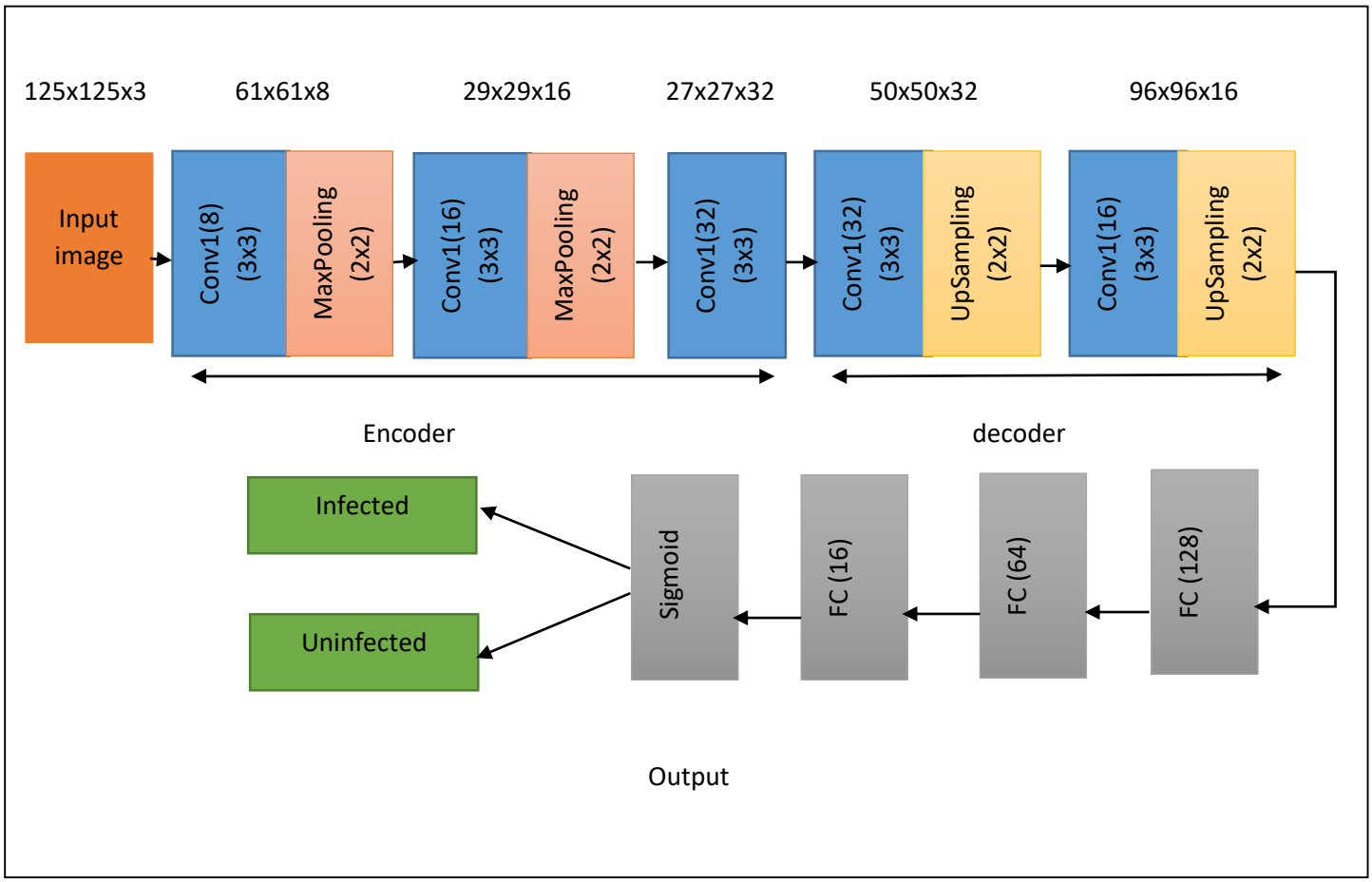



*Figure 3. 9 our proposed AutoEncoder architecture.*

## Conclusion:

This chapter presents the fundamental notion of our proposed approaches, their components and functions used in the three models we proposed. Convolutional Neural Network (CNN) by applying six layers, AutoEncoder with five layers and the pre-trained VGG16 proposed model with applying some modification to their architecture.

And thus, the reason for using these particular type of parameters, functions and size and other not.
In the following chapter we present the results of implementation of the three approaches as well as discuss this result.

# Chapter 4: implementation and evaluation

## Introduction:

In this chapter we will present and discuss the different approaches we implemented to detect malaria in human cells, starting by describing the Dataset, preprocessing the data, data augmentation and the environment and tools we used in our models architecture with our modifications and improvement that have been added, compiling, training and validating the model, and last but not least, a conclusion following a discussion of results.

## 1.Dataset:

The dataset contains an amount of 27,558 images balancing between infected and uninfected cells that are divided equally and accordingly. This data is obtained from the official website of NLM(National Library of Medicine.).

Figure 4.1 and figure 4.2 shows some samples from the dataset containing parasitized and uninfected segmented red blood cells respectively.

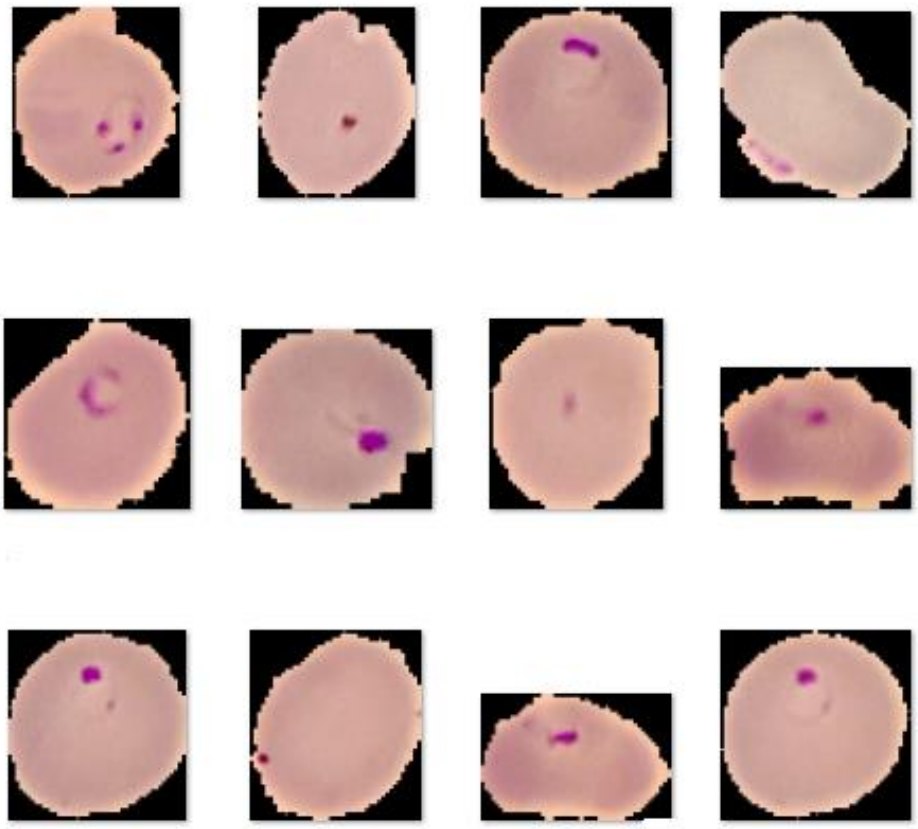


*Figure 4. 1 Samples drawn from NIH (National Library of Medicine) Malaria dataset which are malaria infected parasite red blood cells.*

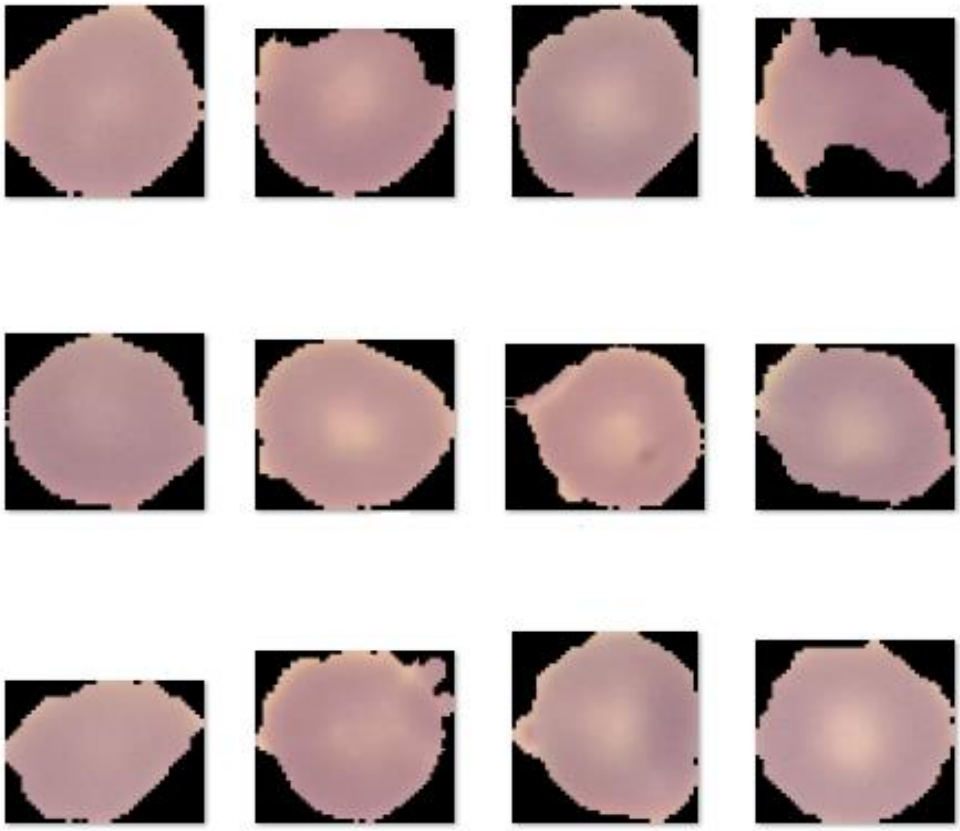


*Figure 4. 2 Samples drawn from NIH (National Library of Medicine) Malaria dataset which are malaria uninfected red blood cells.*

## 2.Environment:

We used Python programming language in our experiment. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.(Menczer, Fortunato, & Davis, 2020)
To conduct our experiment, we used the following modules:

**Numpy:** linear algebra used for general scientific computing, reshaping the vectors into matrices, normalization, randomize splitting data and other mathematical operations.

**Pandas:** data processing used to create DataFrame tasked to read the dataset.

**Matplotlib:** used for visualization.

**Tensorflow:** tensorflow 1.13.1 was used, yet it is worth mentioning that Keras is rather recommended instead of tensorflow for its increased simplicity.

**OS:** The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory.

**Glob:** the glob module is used to retrieve files/pathnames matching a specified pattern.

**OpenCV:** OpenCV is used for all sorts of image and video analysis such as reading an image, changing the color and dimensioning of images.

**sklearn:** used for evaluation and calculating the confusion matrix, precision and recall. And used also to shuffle the infected parasitized and uninfected data together and split it into testing and training set.

**Keras:** deep learning framework was used for building the convolutional neural network. Necessary libraries were imported from Keras to train the model.

### 2.1 Hardware used

The thesis uses free GPUs from Kaggle platform of Google which provide 30 hours of execution time per week. As for the components Kaggle are:
-GPU Memory with 15.9 GB.
-CPU
-RAM with 13 GB
-Disk with 4.9 GB of storage.

## 3.Data preprocessing:

First, we uploaded the data using python code and shuffled it to regroup the infected and uninfected cells. After that we tackled one of the important tasks in the data preprocessing which is to resize Malaria images as the images were of various sizes, the images were resized to 125 by 125 Pixels, this image size struck a balance between providing a high enough resolution for Malaria detection by the model and efficient training. All images were normalized with the OpenCV library. Figure 4.3 shows samples from the processing data.

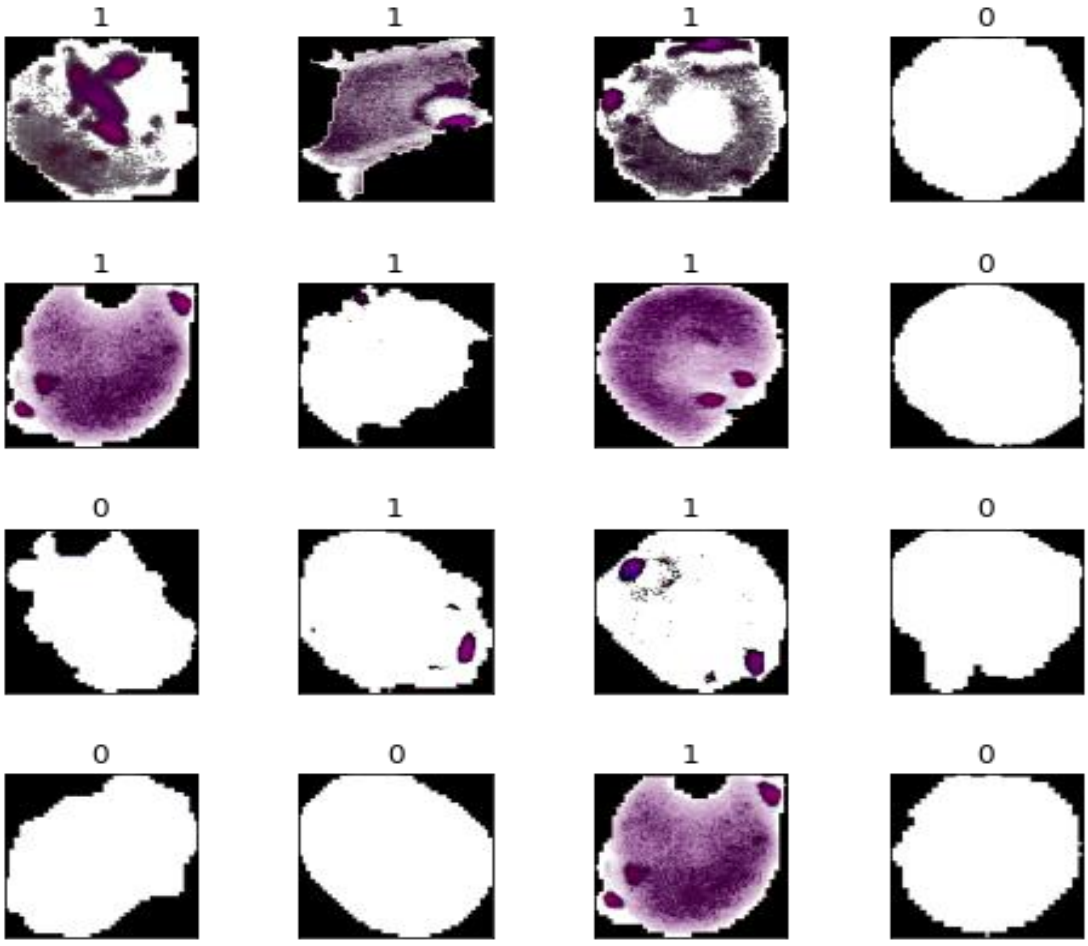



*Figure 4. 3 samples from the processing data.*

This figure shows that we have removed the color from uninfected cells and for the parasitized infected cells we concentrate to the infection and colored it to distinguish the infection.

## 4. Data augmentation:

Data augmentation is having large dataset without collecting new data. However, the performance of the model improved by augmenting the data. We utilized eight augmentation strategies to generate new training sets in the vgg16 model, (Rotation, zoom_range, rotation_range, horizontal flip, width_shift_range, height_shift_range, shear_range, fill_mode). Figure 4.4 shows an example of data augmentation in training set.

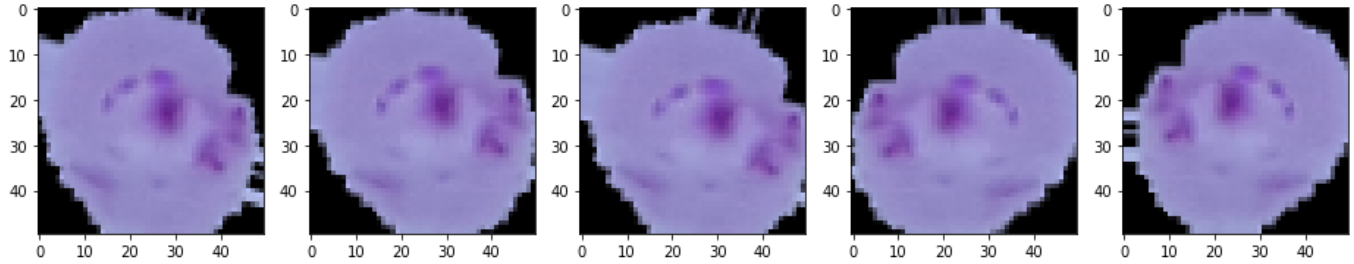


*Figure 4. 4 example of data augmentation in training set*

## 5. Splitting the data:

Dataset divided into three parts (training, test and validation) in the vgg16 model and contains subfolders (parasitized infected and uninfected). Same for vgg16, in the CNN and AutoEncoder model the dataset is constructed of three parts nonetheless (training and testing and validation). There are 27,558 images (PNG) divided to two categories as follows. Table 4.1 shows the organization in the VGG16, CNN and AutoEncoder proposed models.

| **Categories** | **Number-of training images** | **Number-of testing images** | **Number-of validation images** | **Image size** |
|---|---|---|---|---|
| **Parasitized infected** | 7699 | 4146 | 1934 | 125 x 125 pixels |
| **Uninfected** | 7733 | 4122 | 1924 | 125 x 125 pixels |
| **Total** | 15432 | 8268 | 3858 | |

*Table 4. 1 organization of training, testing and validation in the VGG16, CNN, AutoEncoder proposed models.*

For the CNN, AutoEncoder and the vgg16 models we organized the dataset into three sets (training, test and validation) as shown in table 4.1. Therefore, we took (0.3) from the data to the testing set that is equal to (8268) testing images. From the rest of the data we put (0.2) for the validation part which is equal to (3858) images and (15,432) images for the training.

## 6. Train, validation and test:

Training the model used train dataset. In the case of neural networks, the model learns its weights and biases.
The model uses validation dataset for evaluation after every set of predictions as well as helps the model tune its hyper-parameters.
The test dataset is used to evaluate the model after it has been completely trained.

## 7. Over-fitting and under-fitting:

Over-fitting occurs when the model captures the noise of the data. Intuitively, it fits the data too well, or in other words it is too dependent on the data used for training.
Conversely, under-fitting occurs when the model cannot capture the underlying trend of the data, or intuitively it does not fit the data well enough.
Over-fitting and under-fitting both result in poor predictions in a new dataset.

## 8. Network Architecture:

### 8.1 VGG-16 implementation:

the input images were in shape (C, H, W) where H is the height, W is the width, and C is the number of channels. In addition, because our images are colored, and therefore we have three channels: C= 3.
So, the input image size was (125,125,3), and we did some preprocessing to the data before passing it to the training.
After that we Developed the networks based on VGG network model like we mentioned in previous chapter and for optimization we used RMSprop optimization technique. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster(Gandhi, 2018). Finally, for the number of epochs we used 20 epochs to get the best model no less no more.

#### *8.1.1 training the model:*

The model created will be trained on three blocks which are (‘block3’, ‘block4’ and ‘block5’) where each block contains three convolutional layers followed by a MaxPooling layer. The rea son for choosing these blocks over the others is because we focused our attention mainly on th e blocks containing three convolutional layers. After that, there are the fully connected layers

The model is ready for training; during the training, the model will iterate over epochs of the t raining set. Which means gradients will be computed and updates and will be made into the w eights of the network automatically.
An epoch is one of the iterations from all of the training set. Training is usually run until the lo ss converges to a constant.
To save the model with best result, we added one of the features of training. This feature is im plemented via the callback feature of Keras. Call back is a set of functions that will be applied at given stages of training procedure like end of an epoch of training. This is useful because it is needed to do early stops before the model start an over-fitting after a certain number of epoc hs.
And for output classes coding them was considered as follows: {‘parasitized infected’:1, ‘uninfected’:0}.
Here, the input images are labeled and therefore, output neurons are two which means that network is being trained to classify the images into two classes: parasitized infected and uninfected.

#### 8.1.1.1 With data augmentation:

Figure 4.5 shows training the model with data augmentation on 10 epochs

```
Epoch 1/10
100/100 [==============================] - 16s 155ms/step - loss: 0.2148 - accurac
y: 0.9106 - val_loss: 0.0751 - val_accuracy: 0.9754
Epoch 2/10
100/100 [==============================] - 15s 153ms/step - loss: 0.0797 - accurac
y: 0.9800 - val_loss: 0.0602 - val_accuracy: 0.9808
Epoch 3/10
100/100 [==============================] - 16s 157ms/step - loss: 0.0601 - accurac
y: 0.9849 - val_loss: 0.0706 - val_accuracy: 0.9790
Epoch 4/10
100/100 [==============================] - 15s 152ms/step - loss: 0.0617 - accurac
y: 0.9830 - val_loss: 0.0473 - val_accuracy: 0.9871
Epoch 5/10
100/100 [==============================] - 16s 155ms/step - loss: 0.0438 - accurac
y: 0.9884 - val_loss: 0.0487 - val_accuracy: 0.9884
Epoch 6/10
100/100 [==============================] - 16s 156ms/step - loss: 0.0646 - accurac
y: 0.9822 - val_loss: 0.0628 - val_accuracy: 0.9830
Epoch 7/10
100/100 [==============================] - 15s 155ms/step - loss: 0.0558 - accurac
y: 0.9847 - val_loss: 0.0386 - val_accuracy: 0.9897
Epoch 8/10
100/100 [==============================] - 15s 150ms/step - loss: 0.0445 - accurac
y: 0.9887 - val_loss: 0.0375 - val_accuracy: 0.9911
Epoch 9/10
100/100 [==============================] - 15s 154ms/step - loss: 0.0513 - accurac
y: 0.9866 - val_loss: 0.0414 - val_accuracy: 0.9884
Epoch 10/10
100/100 [==============================] - 16s 158ms/step - loss: 0.0372 - accurac
y: 0.9906 - val_loss: 0.0403 - val_accuracy: 0.9897
```

*Figure 4. 5 the two losses and accuracies of the second case with data augmentation.*

After that we drew the training and validation losses and accuracies using matplotlib library, figure 4.6 shows the training and validation accuracy and figure 4.7 shows the training and validation loss.

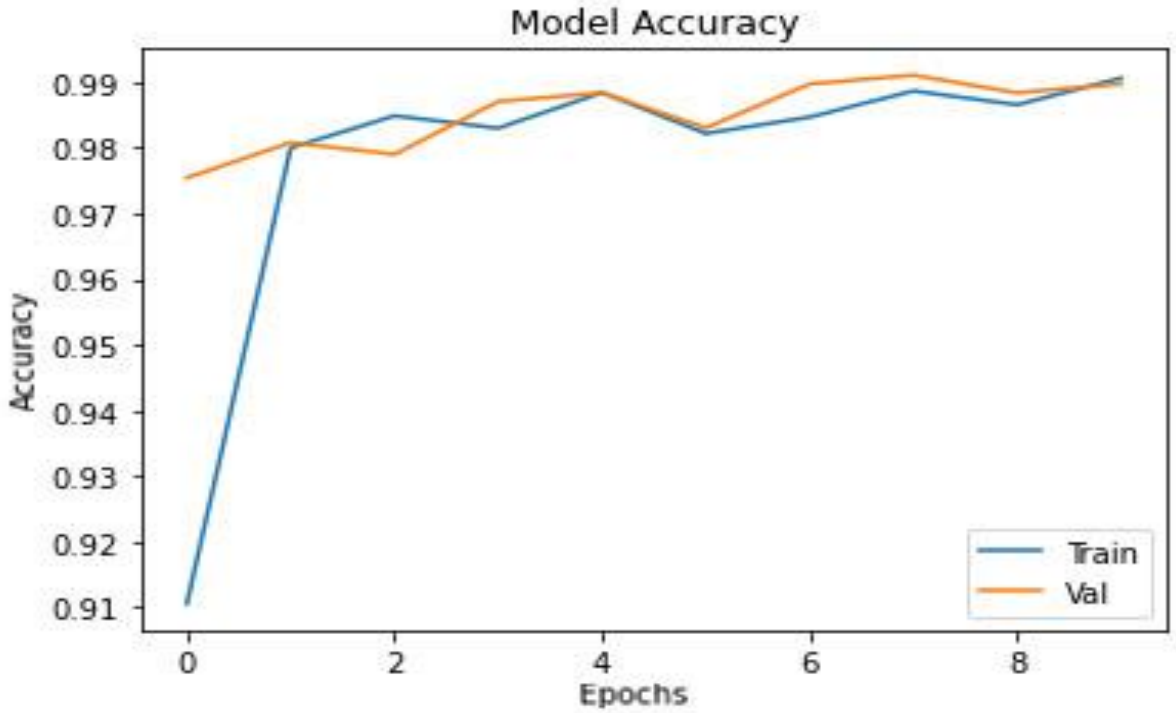



*Figure 4. 6 the training and validation accuracy.*

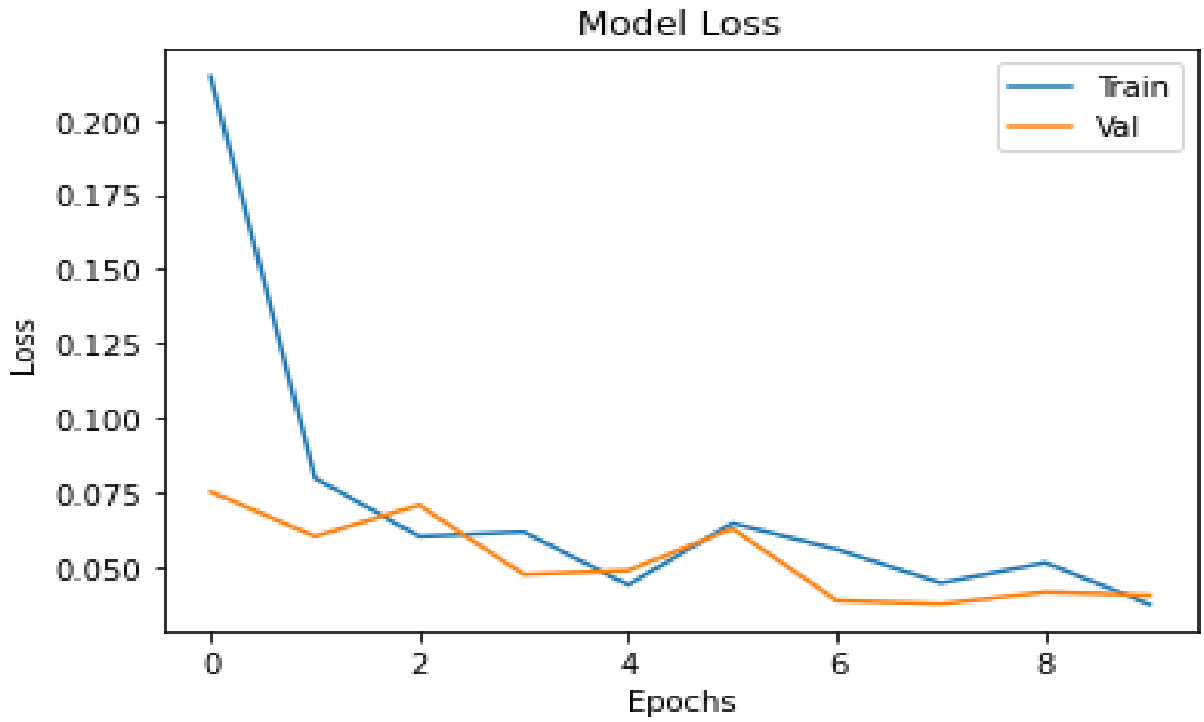



*Figure 4. 7 the training and validation loss.*

From figure 4.6 and figure 4.7 we can notice that both training and validation accuracy increase but not in the good way because they are unstable. And for training and validation loss there is an increase and decrease in their value when we use data augmentation.

#### 8.1.1.2 without data augmentation:

Figure 4.8 shows that the two losses (both loss and val_loss) are decreasing and the tow accuracy (acc and val_acc) are increasing. So this indicates the modeling is trained in a good way.

Train on 15432 samples, validate on 3858 samples
Epoch 1/20
386/386 [==============================] - 25s 64ms/step - loss: 0.7210 - accuracy : 0.8258 - val_loss: 0.1535 - val_accuracy: 0.9355
Epoch 2/20
386/386 [==============================] - 24s 61ms/step - loss: 0.1888 - accuracy : 0.9274 - val_loss: 0.0969 - val_accuracy: 0.9683
Epoch 3/20
386/386 [==============================] - 24s 61ms/step - loss: 0.1169 - accuracy : 0.9590 - val_loss: 0.0731 - val_accuracy: 0.9760
Epoch 4/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0830 - accuracy : 0.9719 - val_loss: 0.0609 - val_accuracy: 0.9819
Epoch 5/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0669 - accuracy : 0.9797 - val_loss: 0.0495 - val_accuracy: 0.9838
Epoch 6/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0541 - accuracy : 0.9828 - val_loss: 0.0469 - val_accuracy: 0.9861
Epoch 7/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0460 - accuracy : 0.9858 - val_loss: 0.0473 - val_accuracy: 0.9874
Epoch 8/20
386/386 [==============================] - 24s 62ms/step - loss: 0.0388 - accuracy : 0.9879 - val_loss: 0.0452 - val_accuracy: 0.9870
Epoch 9/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0350 - accuracy : 0.9888 - val_loss: 0.0388 - val_accuracy: 0.9893
Epoch 10/20
386/386 [==============================] - 24s 62ms/step - loss: 0.0294 - accuracy : 0.9915 - val_loss: 0.0393 - val_accuracy: 0.9900
Epoch 11/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0281 - accuracy : 0.9913 - val_loss: 0.0423 - val_accuracy: 0.9896
Epoch 12/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0263 - accuracy : 0.9917 - val_loss: 0.0401 - val_accuracy: 0.9906
Epoch 13/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0212 - accuracy : 0.9924 - val_loss: 0.0383 - val_accuracy: 0.9909
Epoch 14/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0194 - accuracy : 0.9937 - val_loss: 0.0420 - val_accuracy: 0.9913
Epoch 15/20

```
386/386 [==============================] - 24s 61ms/step - loss: 0.0182 - accuracy
: 0.9939 - val_loss: 0.0371 - val_accuracy: 0.9916
Epoch 16/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0168 - accuracy
: 0.9938 - val_loss: 0.0343 - val_accuracy: 0.9916
Epoch 17/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0150 - accuracy
: 0.9948 - val_loss: 0.0325 - val_accuracy: 0.9913
Epoch 18/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0121 - accuracy
: 0.9954 - val_loss: 0.0347 - val_accuracy: 0.9916
Epoch 19/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0114 - accuracy
: 0.9959 - val_loss: 0.0327 - val_accuracy: 0.9932
Epoch 20/20
386/386 [==============================] - 24s 61ms/step - loss: 0.0112 - accuracy
: 0.9973 - val_loss: 0.0424 - val_accuracy: 0.9916
```

*Figure 4. 8 The model starts training and logs the losses and accuracies*

To capture the training model status, we used Matplotlib to draw plot the training and validation loss side by side, as well as the training and validation Accuracy.
Figure 4.9 shows the training and validation accuracy and Figure 4.10 shows the training and validation loss.

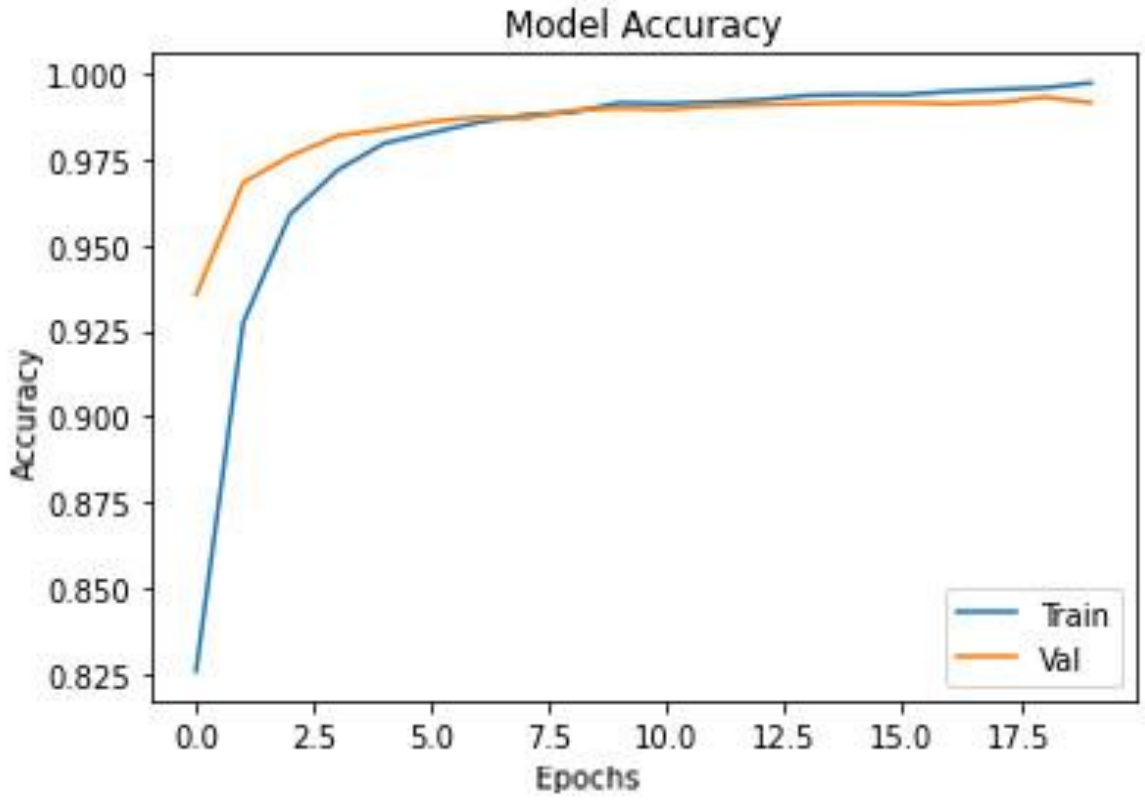



*Figure 4. 9 the training and validation accuracy.*

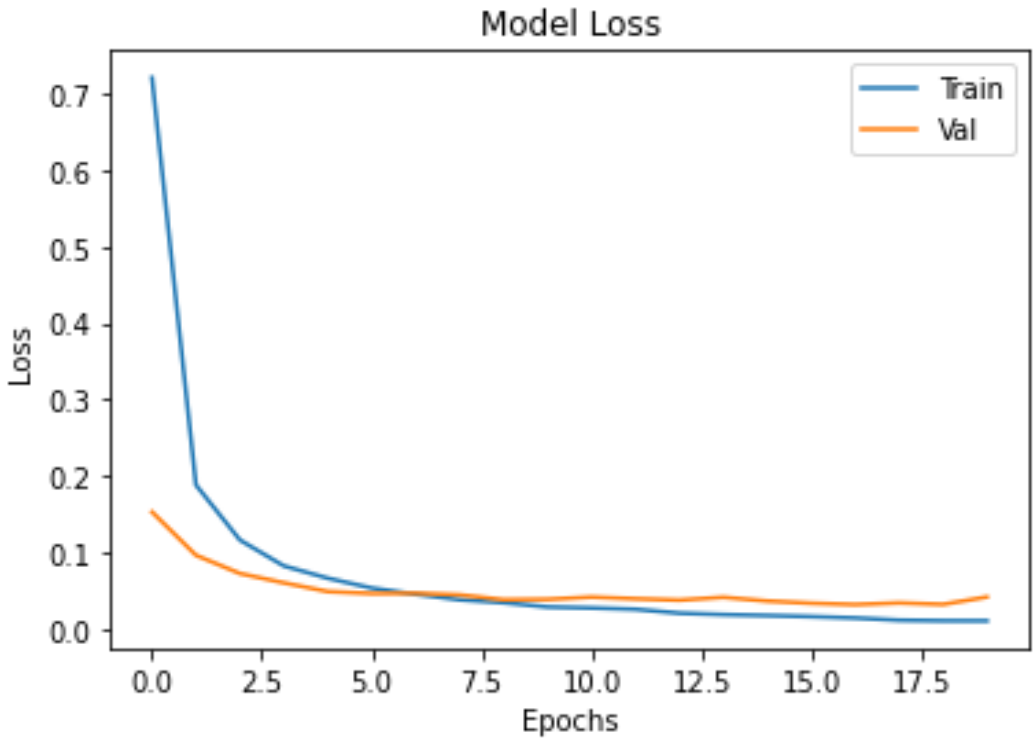



*Figure 4. 10 the training and validation loss.*

From figure 4.9 we can learn that both training and validation accuracy are increasing and for the next figure 4.10 we can notice that the training and validation loss are decreasing with each epoch which indicates that the model is training well.

### *8.1.2 Results:*

Table 4.2 shows the training and validation accuracy for both cases with and without using data augmentation.

| **Accuracy** | **Validation** | **Training** |
|---|---|---|
| **With data augmentation** | 96.92% | 97.12% |
| **Without data augmentation** | 99.12% | 99.74% |

*Table 4. 2 the training and validation accuracy with an input size (125x125x3) with and without using data augmentation.*

From table 4.2 we conclude that we got better result without using data augmentation.

### *8.1.3 Model evaluation*

**Data Set for testing the model**

The testing dataset contains 8268 Malaria images, in (PNG) format, they are images of two cla ssifications of Malaria (parasitized infected, uninfected), distributed as in the following table 4 .3 that shows the first case:

| Categories | Number of testing images | Image size |
|---|---|---|
| **Parasitized infected** | 4146 | 125x125 pixels |
| **Uninfected** | 4122 | 125x125 pixels |

*Table 4. 3 distribution of images in test dataset.*

**Testing the model**

Just After we trained and validated the model, we moved to the test phase where the test set ha d 8268 samples of Malaria images whereas 4146 images of them were infected and while the r emaining 4122 were uninfected.

Testing the model is done through loading the test images and predicting their classes using the model. predict_classes () function, probabilities of each image belonging to a specific class were calculated, by the following classification: (0:'uninfected', 1:'parasitized infected').

The result of our tested model such as accuracy, precision, recall, F1 Score are shown in the following Table 4.4.

| | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| **Performance with Data augmentation** | 0.9747 | 0.9751 | 0.9747 | 0.9747 |
| **Performance without data augmentation** | 0.9903 | 0.9904 | 0.9903 | 0.9903 |

*Table 4. 4 performance of the tested model*

Table 4.5 shows the confusion matrix of the tested model with using data augmentation.

| | Uninfected:0 | Parasitized infected:1 |
|---|---|---|
| **Uninfected:0** | 3958 | 164 |
| **Parasitized infected:1** | 45 | 4101 |

*Table 4. 5 the confusion matrix of tested model.*

And table 4.6 shows confusion matrix of the tested model without using data augmentation.

| | **Uninfected:0** | **Parasitized infected:1** |
|---|---|---|
| **Uninfected:0** | 4105 | 17 |
| **Parasitized infected:1** | 63 | 4083 |

*Table 4. 6 the confusion matrix of the tested model without using data augmentation.*

So, based on the two confusion matrices we calculated our tested model sensitivity and specifi city with and without using data augmentation.
The results are shown in the following Table 4.7.

| **performance** | **With using data augmentation** | **Without using data augmentation** |
|---|---|---|
| **sensitivity** | 0.9891 | 0.9848 |
| **specificity** | 0.9602 | 0.9958 |

*Table 4. 7 the sensitivity and specificity of the tested model in the two cases.*

**Comparing the results**
Based on the obtained results like we mentioned before such as accuracy of training and valid ation, precision, sensitivity and specificity, we conclude that the model whilst without using d ata augmentation gives better results. This indicates that using data augmentation is not necess arily the appropriate solution, since it spoils the clarity of the image and in some cases creates similar images, which causes the network to preserve instead of train, and therefore we get un wanted results such as over-fitting and under-fitting.

## 8.2 CNN implementation:

The same with the vgg16 model our input images were (125,125,3) and for optimization we used Adaptive Moment Estimation (Adam) optimization technique which give us good results by using 13 epochs, and that proved to be enough to get the best model, eventually, and after this number, the model fell in over-fitting.

### *8.2.1 Training the model:*

The model contains six layers and three fully connected layers, two layers contain just convolu tional layers, three layers contain convolutional layers followed by MaxPooling layers and for the last layer contains convolutional layers followed by AveragePooling layer.
The model is ready to train. An epoch is one-iteration over all of the training set. Training is u sually run until the loss converges to a constant.
And for output classes coding it was represented as the following: {‘parasitized infected’:1, ‘uninfected’:0}.

Here, the input images are labeled therefore, output neurons are two which means that network is being trained to classify the images into two classes: parasitized infected and uninfected, same as the vgg16 model. Figure 4.11 shows that the two losses (both loss and val_loss) are decreasing and the tow accuracy (acc and val_acc) is increasing. So this indicates the modeling is trained in an acceptable way.

```
Epoch 1/13
386/386 [==============================] - 5s 13ms/step - loss: 0.4438 - accuracy:
0.7991 - val_loss: 0.1977 - val_accuracy: 0.9446
Epoch 2/13
386/386 [==============================] - 4s 12ms/step - loss: 0.1600 - accuracy:
0.9558 - val_loss: 0.0597 - val_accuracy: 0.9845
Epoch 3/13
386/386 [==============================] - 5s 12ms/step - loss: 0.0623 - accuracy:
0.9823 - val_loss: 0.0767 - val_accuracy: 0.9786
Epoch 4/13
386/386 [==============================] - 4s 12ms/step - loss: 0.0322 - accuracy:
0.9916 - val_loss: 0.0183 - val_accuracy: 0.9964
Epoch 5/13
386/386 [==============================] - 5s 12ms/step - loss: 0.0234 - accuracy:
0.9935 - val_loss: 0.0158 - val_accuracy: 0.9958
Epoch 6/13
386/386 [==============================] - 4s 11ms/step - loss: 0.0205 - accuracy:
0.9941 - val_loss: 0.0127 - val_accuracy: 0.9977
Epoch 7/13
386/386 [==============================] - 4s 12ms/step - loss: 0.0138 - accuracy:
0.9958 - val_loss: 0.0124 - val_accuracy: 0.9964
Epoch 8/13
386/386 [==============================] - 5s 12ms/step - loss: 0.0138 - accuracy:
0.9959 - val_loss: 0.0175 - val_accuracy: 0.9951
Epoch 9/13
386/386 [==============================] - 4s 12ms/step - loss: 0.0115 - accuracy:
0.9968 - val_loss: 0.0131 - val_accuracy: 0.9977
Epoch 10/13
386/386 [==============================] - 5s 12ms/step - loss: 0.0129 - accuracy:
0.9961 - val_loss: 0.0238 - val_accuracy: 0.9981
Epoch 11/13
386/386 [==============================] - 5s 12ms/step - loss: 0.0139 - accuracy:
0.9968 - val_loss: 0.0233 - val_accuracy: 0.9951
Epoch 12/13
386/386 [==============================] - 5s 12ms/step - loss: 0.0123 - accuracy:
0.9970 - val_loss: 0.0060 - val_accuracy: 0.9994
Epoch 13/13
386/386 [==============================] - 4s 12ms/step - loss: 0.0082 - accuracy:
0.9976 - val_loss: 0.0089 - val_accuracy: 0.9984
```

*Figure 4. 11 The model starts training and capture the losses and accuracies.*

Figure 4.12 shows the training and validation Accuracy. And Figure 4.13 shows the training and validation loss.

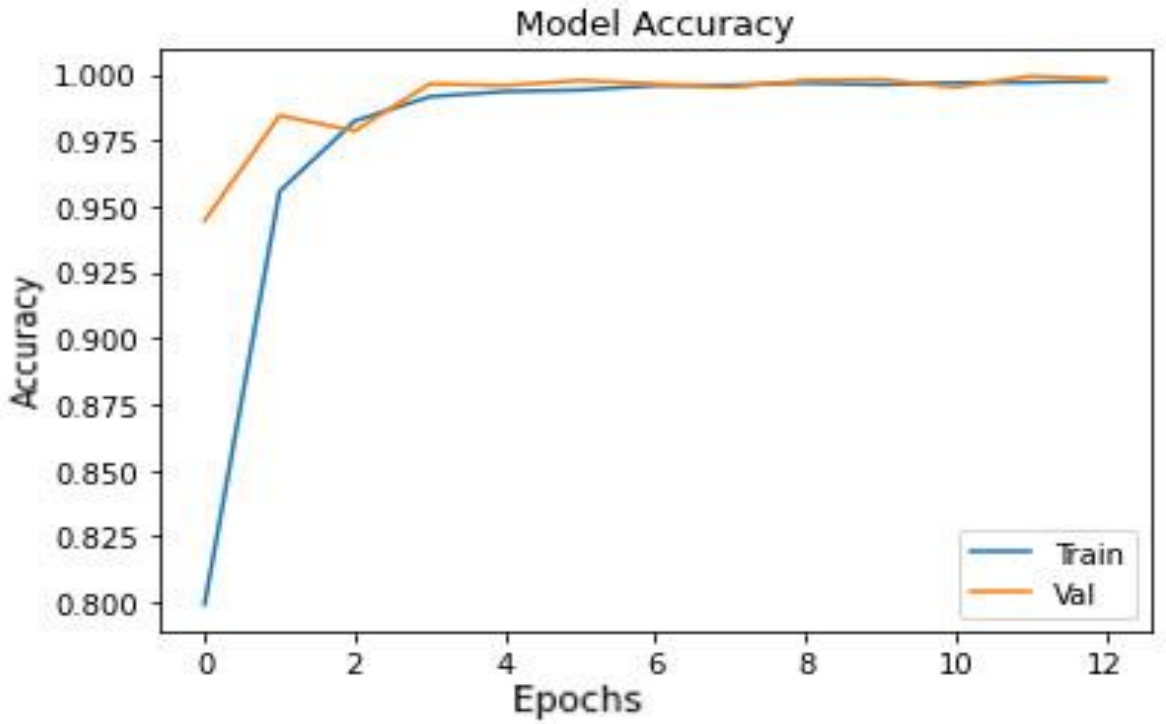



*Figure 4. 12 the training and validation accuracy.*

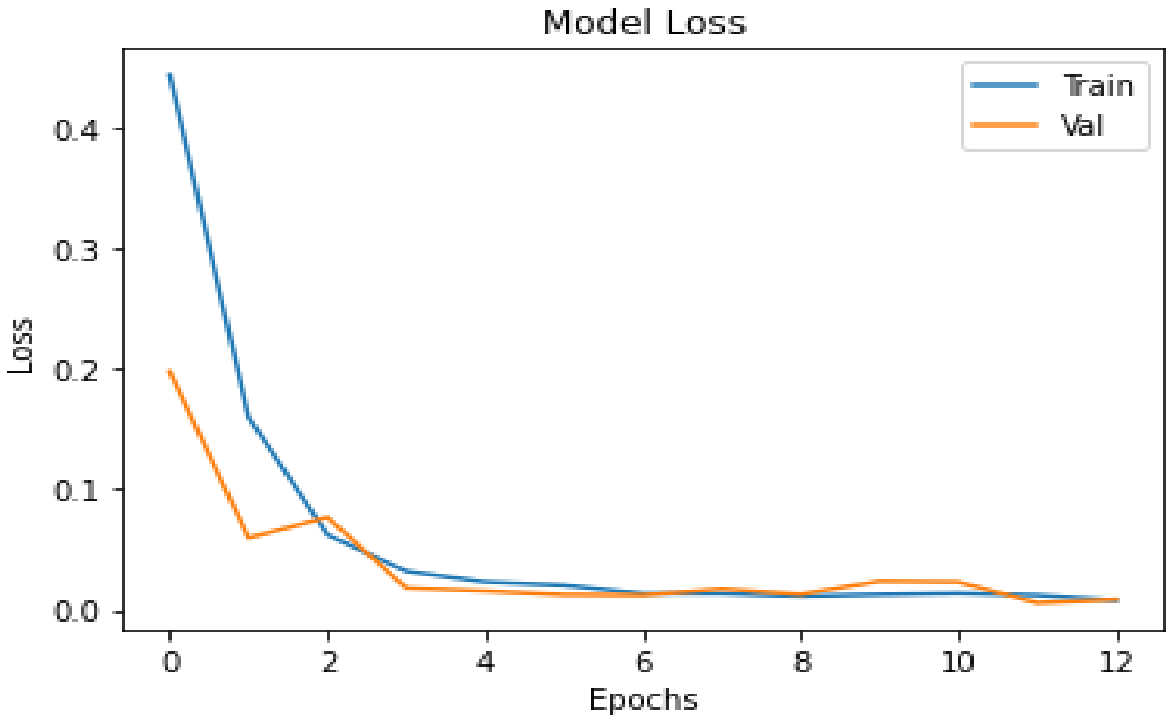



*Figure 4. 13 The training and validation loss.*

Figure 4.12and Figure 4.13showed the learning curve of the network during training. It is noticeable that the validation loss and training loss were decreasing with the number of

iterations, which is desirable; it proved that the model is functional. It also showed the learning curve of the network during training. It is seen that the validation accuracy and training accuracy were increasing, Table 4.8 shows the training and validation accuracy.

| | **Validation** | **Training** |
|---|---|---|
| **Accuracy** | 99.92% | 99.95% |

*Table 4. 8 the training and validation accuracy.*

### *8.2.2 Model evaluation:*

**Data Set for testing the model**

Testing dataset contains 8268 of Malaria images, in (PNG) format, they are images of two classifications of Malaria (parasitized infected, uninfected), distributed as in the following Table 4.9:

| **Categories** | **Number of testing images** | **Image size** |
|---|---|---|
| Parasitized infected | 4146 | 125x125 pixels |
| Uninfected | 4122 | 125x125 pixels |

*Table 4. 9 distribution of images in test dataset.*

**Testing the model**

After training and evaluating the model on the validation, the network is tested using 8268 Malaria images, among them, 4146 parasitized infected and 4122 uninfected.

Testing the model is done through loading the test images and predicting their classes using the model. predict_classes () function, probabilities of each image belonging to a specific class were calculated, by the following classification: (0:'uninfected', 1:'parasitized infected').

The result of our tested model such as accuracy, precision, recall, F1 Score are shown in the following Table 4.10.

| | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|---|
| **Basic Model CNN** | 0.9987 | 0.9987 | 0.9987 | 0.9987 |

*Table 4. 10 performance of the tested model*

Table 4.11 shows confusion matrix of the tested model.

| | **Uninfected:0** | **Parasitized infected:1** |
|---|---|---|
| **Uninfected:0** | 4120 | 2 |
| **Parasitized infected:1** | 9 | 4137 |

*Table 4. 11 the confusion matrix of tested model.*

And for the sensitivity and specificity of the tested model are shown in Table 4.12.

| | |
|---|---|
| **sensitivity** | 0.9978 |
| **specificity** | 0.9995 |

*Table 4. 12 the sensitivity and specificity of the tested model*

**8.3 AutoEncoder implementation:**

in this model, our input is an image with shape (C, H, W). Hence, the image size is (125, 125, 3), and for optimization we used Adaptive Moment Estimation (Adam) optimization technique which give us desired results. Unlike the CNN model, in this model we used 10 epochs that were enough to get the best model with the best results.

***8.3.1 Training the model:***

This model contains two parts, encoder and decoder parts. For the encoder section we created three layers, each layer of the first two layers contains convolutional layer followed by a 2x2 maxpooling layer, and the third layer contains just a convolutional layer.

For the decoder section we created two layers, each one contains convolutional layer followed by a 2x2 UpSampling layer. The UpSampling layers are used in the AutoEncoder model where we multiply the size of image by two (2).

The model is ready to train. An epoch is one-iteration over all of the training set. And for the o utput classes coding them was represented as the following: {‘parasitized infected’:1, ‘uninfec ted’:0}.

Here, the input images are labeled therefore, the output neurons are two which means that the network is being trained to classify the images into two classes: parasitized infected and uninfected.

Figure 4.14 shows that the two losses (both loss and val_loss) are decreasing and the tow accuracy (acc and val_acc) are increasing. So this indicates the modeling is trained in a good way.

```
Train on 12345 samples, validate on 3087 samples
Epoch 1/10
12345/12345 [==============================] - 10s 807us/sample - loss: 0.7743 -
accuracy: 0.8408 - val_loss: 0.1301 - val_accuracy: 0.9686
Epoch 2/10
12345/12345 [==============================] - 5s 442us/sample - loss: 0.1048 - a
ccuracy: 0.9741 - val_loss: 0.0995 - val_accuracy: 0.9835

Epoch 3/10
12345/12345 [==============================] - 5s 413us/sample - loss: 0.0754 - a
ccuracy: 0.9823 - val_loss: 0.1867 - val_accuracy: 0.9501
Epoch 4/10
12345/12345 [==============================] - 5s 414us/sample - loss: 0.0608 - a
ccuracy: 0.9858 - val_loss: 0.0746 - val_accuracy: 0.9809
Epoch 5/10
12345/12345 [==============================] - 5s 402us/sample - loss: 0.0530 - a
ccuracy: 0.9878 - val_loss: 0.0616 - val_accuracy: 0.9864
Epoch 6/10
12345/12345 [==============================] - 5s 400us/sample - loss: 0.0422 - a
ccuracy: 0.9904 - val_loss: 0.0832 - val_accuracy: 0.9815
Epoch 7/10
12345/12345 [==============================] - 5s 402us/sample - loss: 0.0528 - a
ccuracy: 0.9868 - val_loss: 0.0701 - val_accuracy: 0.9838
Epoch 8/10
12345/12345 [==============================] - 5s 400us/sample - loss: 0.0362 - a
ccuracy: 0.9904 - val_loss: 0.0769 - val_accuracy: 0.9851
Epoch 9/10
12345/12345 [==============================] - 5s 401us/sample - loss: 0.0381 - a
ccuracy: 0.9906 - val_loss: 0.0622 - val_accuracy: 0.9854
Epoch 10/10
12345/12345 [==============================] - 5s 391us/sample - loss: 0.0294 - a
ccuracy: 0.9933 - val_loss: 0.0706 - val_accuracy: 0.9883
```

*Figure 4. 14 The model start training and capture the two losses (both loss and val_loss) and the tow accuracy (acc and val_acc).*

After training the model we used matplotlib library to plot the result of both training and validation loss and accuracy. Figure 4.15 shows the training and validation Accuracy and Figure 4.16 shows the training and validation Accuracy.

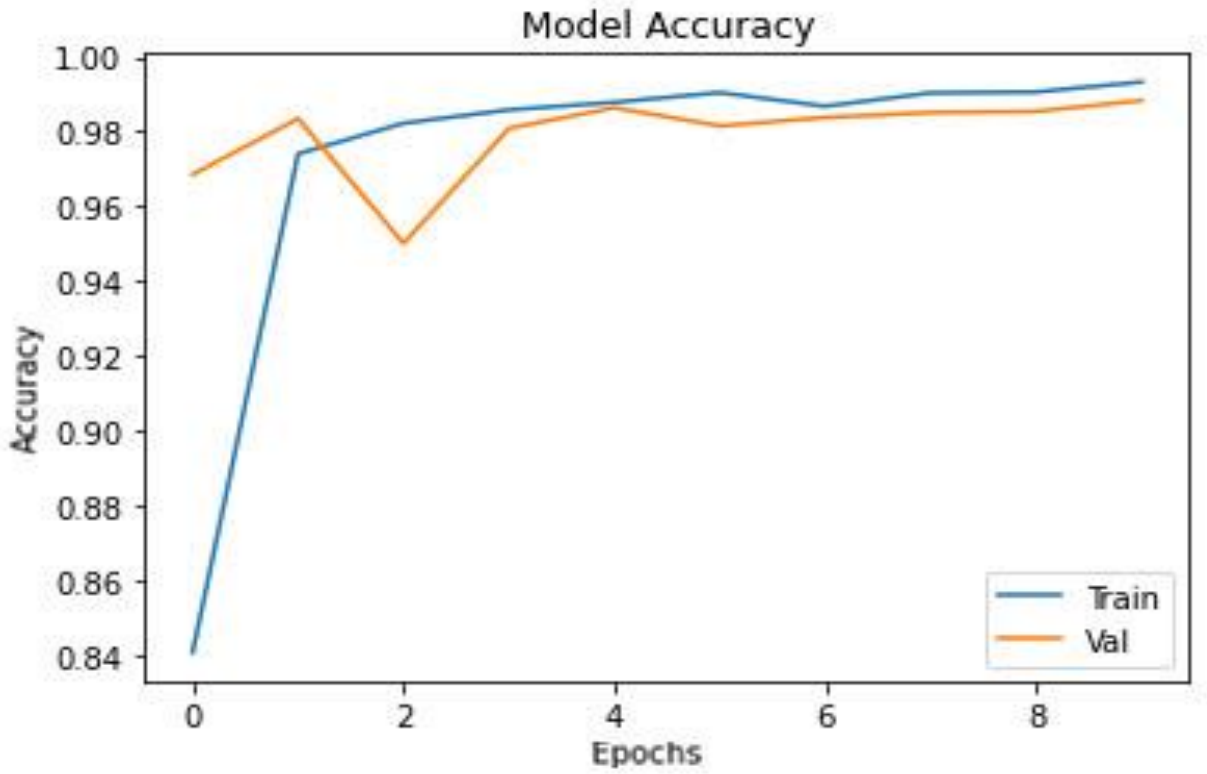



*Figure 4. 15 the training and validation accuracy.*

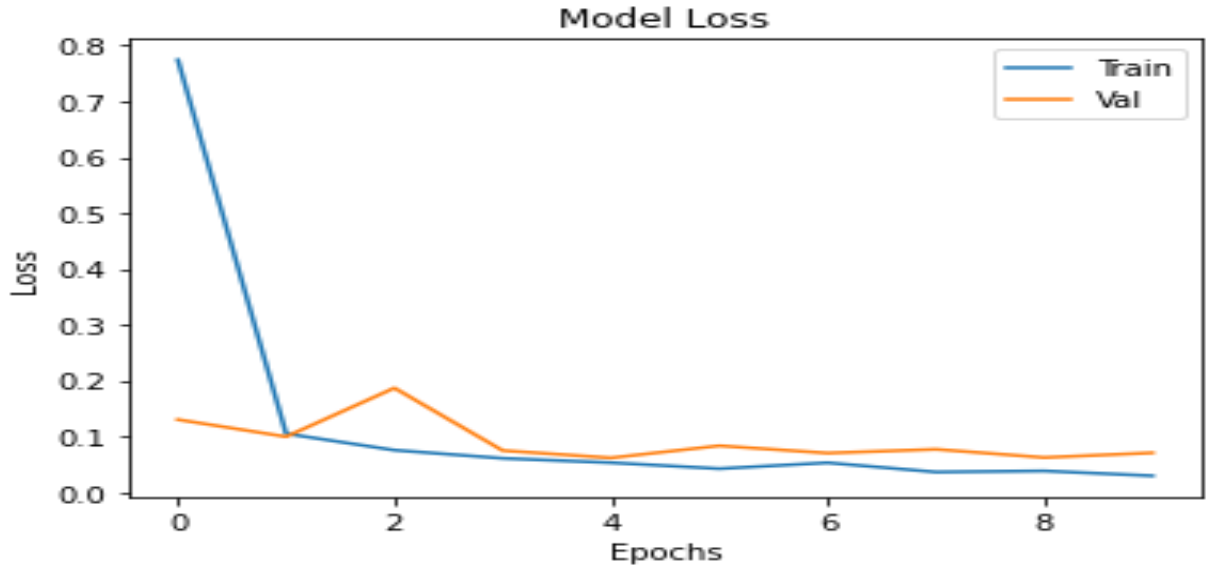



*Figure 4. 16 the training and validation loss.*

Figure 4.15 and Figure 4.16 showed the learning curve of the network during training. It is seen that the validation loss and training loss were decreasing with the number of epochs and that they are about equal, it proved that the model works. It is also worth of mentioning that it showed the learning curve of the network during training. It is seen that the validation accuracy and training accuracy were increasing and are about equal, Table 4.13 shows the training and validation accuracy.

| | **Validation** | **Training** |
|---|---|---|
| **Accuracy** | 99.04% | 99.47% |

*Table 4. 13 the training and validation accuracy.*

### *8.3.2 Model evaluation:*

**Data Set for testing the model**

Testing dataset contains 8268 of Malaria images, in (PNG) format, they are images of two classes of Malaria (parasitized infected, uninfected), distributed as in the following Table 4.14.

| **Categories** | **Number of testing images** | **Image size** |
|---|---|---|
| Parasitized infected | 4146 | 125x125 pixels |
| Uninfected | 4122 | 125x125 pixels |

*Table 4. 14 distribution of images in test dataset.*

**Testing the model**

After training and validating our model, we tested it using 8268 Malaria images, 4146 of them were parasitized infected and 4122 were uninfected. Testing the model is done through loading the test images and predicting their classes using the model. predict_classes () function and probabilities of each image belonging to a specific class were calculated, by the following classification: (0:'uninfected', 1:'parasitized infected').

The result of our tested model such as accuracy, precision, recall, F1 Score are shown in the following Table 4.15.

| | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|---|
| **Basic Model CNN** | 0.9896 | 0.9896 | 0.9896 | 0.9896 |

*Table 4. 15 performance of the tested model.*

Table 4.16 shows confusion matrix of the tested model.

| | **Uninfected:0** | **Parasitized infected:1** |
|---|---|---|
| **Uninfected:0** | 4097 | 25 |
| **Parasitized infected:1** | 61 | 4085 |

*Table 4. 16 the confusion matrix of tested model.*

After that we computed both the sensitivity and specificity of our model through the confusion matrix. Table 4.17 shows the sensitivity and specificity.

| | |
|---|---|
| sensitivity | 0.9852 |
| specificity | 0.9939 |

*Table 4. 17 the sensitivity and specificity of our model.*

## 9. Discussion:

After the execution of the three models and obtaining results, we create Table 4.18 which presents the results in all performance whether in accuracy, sensitivity, specificity, precision and F1-Score.

| | **VGG16 with data augmentation** | **VGG16 without data augmentation** | **CNN** | **AutoEncoder** |
|---|---|---|---|---|
| **Accuracy** | 97,47% | 99,03% | 99,87% | 98,96% |
| **Sensitivity** | 98,91% | 98,48% | 99,78% | 98,52% |
| **Specificity** | 96,02% | 99,58% | 99,95% | 99,39% |
| **Precision** | 97,51% | 99,04% | 99,87% | 98,96% |
| **F1-Score** | 97,47% | 99,03% | 99,87% | 98,96% |

*Table 4. 18 the results of the three proposed model.*

The results in table 4.18 show that the CNN model achieves better results comparing with pre-trained VGG16 and the AutoEncoder models. The CNN model has an accuracy of 99,87%, and the sensitivity, specificity, precision and F1-Score all reach the 99% level.

And for the VGG16 and AutoEncoder models, we can achieve better results by adding some l ayers to the architecture or modify some activation functions such as ReLU, Sigmoid and Soft max. In some case the use of other type of optimization technique such as Adam, RMSprop, S GD or others can also augment result in all performance.
Without forgetting the hardware which can also increase results when using better equipment.

Figure 4.17 shows the representation of results with chart.

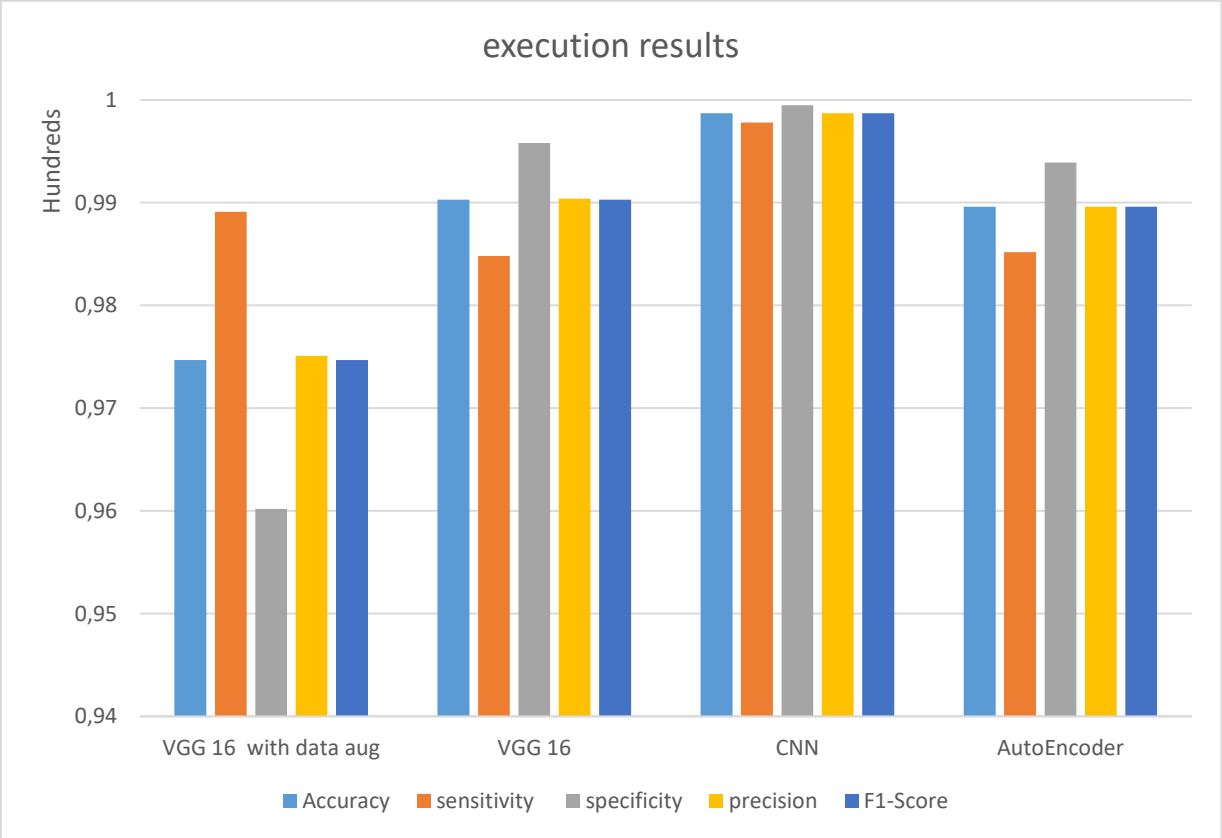



*Figure 4. 17 the representation of results.*

From this figure we can notice that the CNN model shows superiority over the two other model AutoEncoder or pretrained model VGG16 in all performance. This what prove CNN specialization's in the field of images.

**Comparing results with previous studies:**

From the previous works mention in chapter2, we choose those that performed well and compared them with our work. Table 4.19 shows the comparing results.

| Performances/ works | Accuracy | Precision | F1-Score | Sensitivity | Specificity |
|---|---|---|---|---|---|
| (Devi et al., 2018) **ANN** | 96.32% | / | 85.32% | 93% | 96.75% |
| (Vijayalakshmi & Rajesh Kanna, 2019) **VGG19** | 93.13% | 89.95% | 91.66% | 93.44% | 92.92% |
| (Liang et al., 2016) **CNN** | 97.37% | 97.73% | 97.36% | 96.99% | 97.75% |
| (Bibin et al., 2017) **DBN** | 96.35% | / | 89.66% | 97.60% | 95.92% |
| (Gopakumar et al., 2018) **CNN** | / | / | / | 97.06% | 98.50% |
| **Proposed CNN** | 99.87% | 99.87% | 99.87% | 99.78% | 99.95% |
| **Proposed VGG16** | 99.03% | 99.04% | 99.03% | 98.48% | 99.58% |
| **Proposed AutoEncoder** | 98.96% | 98.96% | 98.96% | 98.52% | 99.39% |

*Table 4. 19 comparing results.*

From table 4.19 we can notice that our work gives better results than the existing in all performance such as accuracy, precision, F1-Score, Sensitivity and specificity. Figure 4.18 shows the comparing performance.

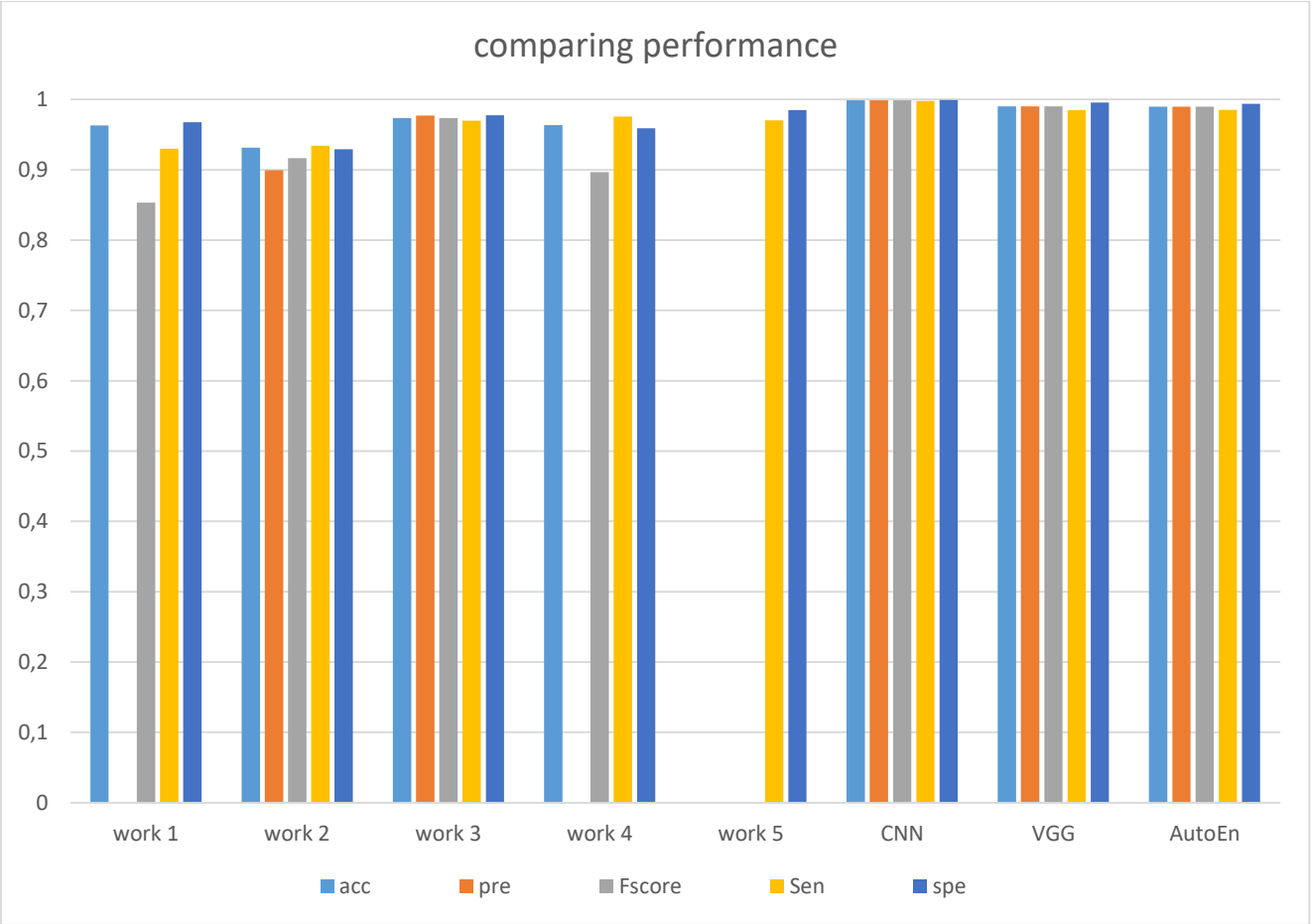



*Figure 4. 18 comparing performance*

This figure shows that our proposed CNN model approaching the 100% which shows superiority over all the previous works mention and over the AutoEncoder or pretrained model VGG16 in all performance. This what allow us to say that this model is considered as sufficient solution to discover this serious disease.

## 10. Application:

We create an application where users can choose an image and the algorithm he wants test with from the three proposed approach, and we classify his input. The purpose of this application is to evaluate our models.
We create this application using HTML, CSS, JAVASCRIPT module for front end, and Python with Flask framework for the backend side.
- Html (Hypertext Markup Language): aside from creating the web pages.
- CSS (Cascading Style Sheet): for styling and making responsive layouts.
- JavaScript: to make the web site interactive.
- Flask: Flask is a micro web framework written in python allows us rapid and secure development for web applications.

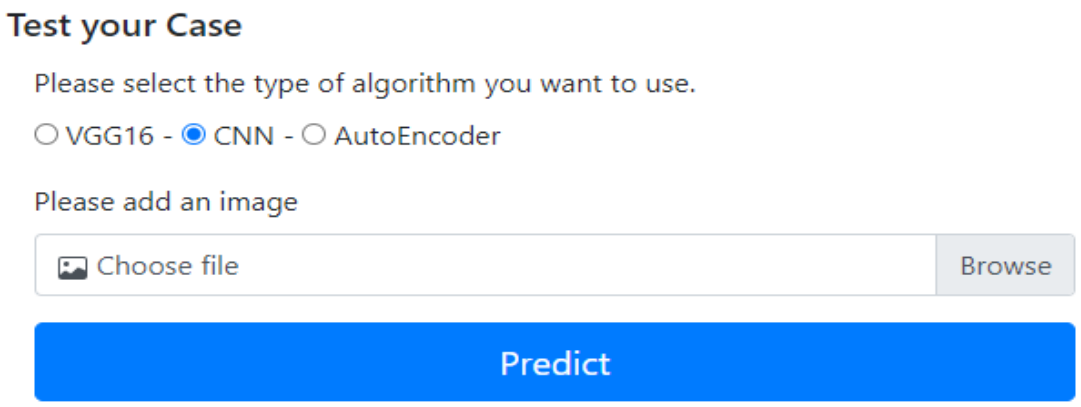



*Figure 4. 19 test application.*

After choosing the model with user and the image he wants to predict, the execution result of predicting the infection with malaria is mention in figure 4.19.




*Figure 4. 20 execution result.*

## Conclusion:

This chapter presents our implementation results of the three proposed approaches.

First and foremost, it is divided into two parts: the first part being concerning the data such as preprocessing, data augmentation and data splitting into sets which are validation, training and test.

While the second part concerning the model as training, compiling and finally testing the model and getting results.

# General Conclusion

We developed a deep learning model that can accurately identify the existence or absence of the Malaria virus in an image taken from a human cell. Consequently, reducing the mortality rate of this serious disease. This model can be embedded in an electronic device for mass production.

We used three different deep learning approaches for Malaria Identification Convolutional Neural Network (CNN), AutoEncoder and Visual Geometry Group (VGG16). The first model gave 99.87% accuracy while the second model gave 98.96%, the third model accuracy of 99.03%. Henceforward, for this task, the CNN model produces the best results compared to the other models.

All the above confirm to us those deep learning methods demonstrated their superiority in the medical field compared to the other high-performing algorithms without using data augmentation because with using it the models gives low performance. Therefore, it is safe to assume that deep learning is suitable solution for the diagnosis of malaria infected cells and we can be sure that this project will detect the disease in the early stage.

We conclude that the performance of deep learning architecture varies according to the task and in particular, the data itself. We demonstrated the use of the CNN model of deep learning algorithms can significantly improve the efficiency and accuracy of Malaria diagnosis.

In future work, we want to test our models with different datasets in the medical field, one of the important disease we want to detect with those models is Corona virus.

We hope to increase the amount of RAM on Kaggle so that the images can expand in size larger than 125x125, which increases the models performances.

## References


Alex Krizhevsky, Sutskever, I., & Geoffrey E. Hinton. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*.

Almisreb, A. A., Jamil, N., & Din, N. M. (2018). Utilizing AlexNet Deep Transfer Learning for Ear Recognition. *Proceedings - 2018 4th International Conference on Information Retrieval and Knowledge Management: Diving into Data Sciences, CAMP 2018*, 8–12. https://doi.org/10.1109/INFRKM.2018.8464769

Alofi Kanter. (2015). Silver Fern (Black & White). Retrieved May 25, 2020, from 10 août 2015 Traced from the the design gallery website: https://fr.m.wikipedia.org/wiki/Fichier:NZ_flag_design_Silver_Fern_(Black_%26_White)_by_Alofi_Kanter.svg

Avatar, & B, N. B. H. (2020). base augmentation. Retrieved September 21, 2020, from 2020 website: https://viblo.asia/p/tang-cuong-du-lieu-trong-deep-learning-oOVlYe4nl8W

Balafar, M. A., Ramli, A. R., Saripan, M. I., & Mashohor, S. (2010). Review of brain MRI image segmentation methods. *Artificial Intelligence Review*, *33*(3), 261–274. https://doi.org/10.1007/s10462-010-9155-0

Bibin, D., Nair, M. S., & Punitha, P. (2017). Malaria Parasite Detection from Peripheral Blood Smear Images Using Deep Belief Networks. *IEEE Access*, *5*, 9099–9108. https://doi.org/10.1109/ACCESS.2017.2705642

Bjorck, J., Gomes, C., Selman, B., & Weinberger, K. Q. (2018). *Understanding Batch Normalization - Google 搜索*. (NeurIPS). Retrieved from https://www.google.ca/search?q=Understanding+Batch+Normalization&oq=Understanding+Batch+Normalization&aqs=chrome..69i57j69i61j69i60j0.588j0j7&client=ubuntu&sourceid=chrome&ie=UTF-8

Bradley, W. G. (2015). of Medical Imaging1 History. *Proceedings of the American Philosophical Society*, *152*(3), 349–361.

Brannon, B. (2009). *Learning from Text at Web Scale*.

Brownlee, J. (2019). Machine Learning Mastery. Retrieved March 26, 2020, from https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

Budhiraja, A. (2016). Dropout in (Deep) Machine learning. Retrieved May 9, 2020, from Dec 15, 2016 website: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

Camacho, C. (2018). Convolutional Neural Networks. Retrieved September 6, 2020, from 2018 website: https://cezannec.github.io/Convolutional_Neural_Networks/

Chakravorty, P. (2018). What Is a Signal? [Lecture Notes]. *IEEE Signal Processing Magazine*, *35*(August), 175–177. https://doi.org/10.1109/MSP.2018.2832195

Chang, P. L., & Teng, W. G. (2007). Exploiting the self-organizing map for medical image segmentation. *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 281–286. https://doi.org/10.1109/CBMS.2007.48

Chen, C. H. (2014). *an Introduction To Computer Vision in Medical Imaging*. 1–16. https://doi.org/10.1142/9789814460941_0001

Chen, M., Shi, X., Zhang, Y., Wu, D., & Guizani, M. (2017). Deep Features Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network. *IEEE Transactions on Big Data*, *7790*(c), 1–1. https://doi.org/10.1109/tbdata.2017.2717439

Devi, S. S., Roy, A., Singha, J., Sheikh, S. A., & Laskar, R. H. (2018). Malaria infected erythrocyte classification based on a hybrid classifier using microscopic images of thin

blood smear. *Multimedia Tools and Applications*, *77*(1), 631–660. https://doi.org/10.1007/s11042-016-4264-7

Dong, Y., Jiang, Z., Shen, H., David Pan, W., Williams, L. A., Reddy, V. V. B., … Bryan, A. W. (2017). Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. *2017 IEEE EMBS International Conference on Biomedical and Health Informatics, BHI 2017*, 101–104. https://doi.org/10.1109/BHI.2017.7897215

Duchi, J. C., Bartlett, P. L., & Wainwright, M. J. (2012). Randomized smoothing for (parallel) stochastic optimization. *Proceedings of the IEEE Conference on Decision and Control*, *12*, 5442–5444. https://doi.org/10.1109/CDC.2012.6426698

Elmardi, K. A., Malik, E. M., Abdelgadir, T., Ali, S. H., Elsyed, A. H., Mudather, M. A., … Adam, I. (2009). *Feasibility and acceptability of home-based management of malaria strategy adapted to Sudan ' s conditions using artemisinin-based combination therapy and rapid diagnostic test*. *8*. https://doi.org/10.1186/1475-2875-8-39

Gandhi, R. (2018). A Look at Gradient Descent and RMSprop Optimizers. Retrieved April 22, 2020, from Jun 19, 2018 website: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*(19), 12777–12815. https://doi.org/10.1007/s11042-019-08453-9

Geoffrey E. Hinton (2009). (n.d.). Deep belief networks. *Scholarpedia, 4(5):5947*. https://doi.org/10.4249/scholarpedia.5947

Gonzalez, T. F. (2007). Handbook of approximation algorithms and metaheuristics. *Handbook of Approximation Algorithms and Metaheuristics*, 1–1432. https://doi.org/10.1201/9781420010749

Gopakumar, G. P., Swetha, M., Sai Siva, G., & Sai Subrahmanyam, G. R. K. (2018). Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner. *Journal of Biophotonics*, *11*(3). https://doi.org/10.1002/jbio.201700003

Hochreiter, S. (1997). *Long Short-Term Memory*. *1780*, 1735–1780.

Huang, H. H., Wang, Z., & Chung, W. (2019). Efficient parameter selection for support vector machines. *Enterprise Information Systems*, *13*(6), 916–932. https://doi.org/10.1080/17517575.2019.1592233

Imran Razzak, M., & Informatics, H. (2015). Automatic Detection and Classification of Malarial Parasite. *International Journal of Biometrics and Bioinformatics (IJBB)*, (9), 1.

Jason Brownlee. (2016). Machine Learning Mastery. Retrieved March 26, 2020, from https://machinelearningmastery.com/crash-course-convolutional-neural-networks/

Kanaa, T., Tchiotsop, D., Ele, P., Tonye, E., & Belon, P. (2016). Automated Diagnosis of Malaria in Tropical Areas Using 40X Microscopic Images of Blood Smears. *International Journal of Biometrics and Bioinformatics*, *10*(2), 12–23.

Kapishnikov, S., Grolimund, D., Schneider, G., Pereiro, E., McNally, J. G., Als-Nielsen, J., & Leiserowitz, L. (2017). Unraveling heme detoxification in the malaria parasite by in situ correlative X-ray fluorescence microscopy and soft X-ray tomography. *Scientific Reports*, *7*(1), 1–12. https://doi.org/10.1038/s41598-017-06650-w

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liang, Z., Powell, A., Ersoy, I., & Poostchi, M. (2016). CNN-based image analysis for

malaria diagnosis. *Ieeexplore.Ieee.Org*, 8–11. Retrieved from https://ieeexplore.ieee.org/abstract/document/7822567/

Malihi, L., Ansari-Asl, K., & Behbahani, A. (2013). Malaria parasite detection in giemsa-stained blood cell images. *Iranian Conference on Machine Vision and Image Processing, MVIP*, (December 2015), 360–365. https://doi.org/10.1109/IranianMVIP.2013.6780011

Mark Nixon & Alberto Aguado. (2002). Computer Vision Demonstration. Retrieved September 5, 2020, from 2002 website: https://www.southampton.ac.uk/~msn/book/new_demo/median/

Matthew Stewart, P. R., & Stewart, M. (2019). Comprehensive Introduction to Autoencoders. Retrieved May 9, 2020, from Apr 14, 2019 website: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

Menczer, F., Fortunato, S., & Davis, C. A. (2020). Python Tutorial. *A First Course in Network Science*, 221–237. https://doi.org/10.1017/9781108653947.010

Mohanty, I., Pattanaik, P. A., & Swarnkar, T. (2019). Automatic detection of malaria parasites using unsupervised techniques. *Lecture Notes in Computational Vision and Biomechanics*, *30*, 41–49. https://doi.org/10.1007/978-3-030-00665-5_5

Montavon, G., Samek, W., & Müller, K. R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing: A Review Journal*, *73*, 1–15. https://doi.org/10.1016/j.dsp.2017.10.011

Muhammad, U., Wang, W., Chattha, S. P., & Ali, S. (2018). Pre-trained VGGNet Architecture for Remote-Sensing Image Scene Classification. *Proceedings - International Conference on Pattern Recognition*, *2018-Augus*(January 2019), 1622–1627. https://doi.org/10.1109/ICPR.2018.8545591

Murad, A., & Pyun, J. Y. (2017). Deep recurrent neural networks for human activity recognition. *Sensors (Switzerland)*, *17*(11). https://doi.org/10.3390/s17112556

National Library of Medicine. (2019). Malaria DataSets. Retrieved March 29, 2020, from https://lhncbc.nlm.nih.gov/publication/pub9932

Nelson, D., & Wang, J. (1992). Introduction to artificial neural systems. *Neurocomputing*, *4*(6), 328–330. https://doi.org/10.1016/0925-2312(92)90018-k

Parashu Ram Pal. (2019). Deep Learning Neural Network. Retrieved September 20, 2020, from 2019 website: https://www.researchgate.net/figure/Deep-Learning-Neural-Network_fig2_333506815

Poostchi, M., Silamut, K., Maude, R. J., Jaeger, S., & Thoma, G. (2018). Image analysis and machine learning for detecting malaria. *Translational Research*, *194*, 36–55. https://doi.org/10.1016/j.trsl.2017.12.004

Pradhan, B., & Sameen, M. I. (2018). *Manifestation of SVM-Based Recti fi ed Linear Unit ( ReLU ) Kernel Function in Landslide Modelling*. 185–195.

Preethi, D. (2014). Performance Evaluation of IoT Result for Machine Learning. *Transactions on Engineering and Sciences*, *2*(11), 1–4. Retrieved from https://pdfs.semanticscholar.org/a1c7/10e8a967893b50fe1e9cdff61055d3564796.pdf

Quellec, J. Le, Harman, J., Defrasne, C., Duquesnoy, F., Quellec, J. Le, Harman, J., … Duquesnoy, F. (2017). *DStretch ® et l ' amélioration des images numériques : applications à l ' archéologie des images rupestres To cite this version : HAL Id : halshs-00935630 DStretch et l ' amélioration des images numériques : applications à l ' archéologie des images rupes*.

R, I. (2018). Batch Normalization — Speed up Neural Network Training. Retrieved May 9, 2020, from May 29, 2018 website: https://medium.com/@ilango100/batch-normalization-speed-up-neural-network-training-245e39a62f85

R, S. K., & Jaeger, S. (n.d.). *Visualizing Deep Learning Activations for Improved Malaria Cell Classification*. 1–8.

Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., … Thoma, G. R. (2018). Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, *2018*(4), 1–17. https://doi.org/10.7717/peerj.4568

Reilly, J. M., & Frey, F. S. (1996). Recommendations for the Evaluation of Digital Images Produced from Photographic, Microphotographic, and Various Paper Formats. *Scanning*.

Romanuke, V. (2017). Appropriate Number and Allocation of RELUS in Convolutional Neural Networks. *Research Bulletin of the National Technical University of Ukraine "Kyiv Politechnic Institute," 0*(1), 69–78. https://doi.org/10.20535/1810-0546.2017.1.88156

Sai Bharadwaj Reddy, A., & Sujitha Juliet, D. (2019). Transfer learning with RESNET-50 for malaria cell-image classification. *Proceedings of the 2019 IEEE International Conference on Communication and Signal Processing, ICCSP 2019*, 945–949. https://doi.org/10.1109/ICCSP.2019.8697909

Singh, H. (2019). *Practical Machine Learning and Image Processing For Facial Recognition, Object Detection, and Pattern Recognition Using Python-Himanshu Singh*. https://doi.org/10.1007/978-1-4842-4149-3

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. *13th Annual Conference of the International Speech Communication Association 2012, INTERSPEECH 2012*, *1*, 194–197.

Suzuki, H., & Toriwaki, J. ichiro. (1991). Automatic segmentation of head mri images by knowledge guided thresholding. *Computerized Medical Imaging and Graphics*, *15*(4), 233–240. https://doi.org/10.1016/0895-6111(91)90081-6

Tindall, L., Luong, C., & Saad, A. (2015). *Plankton classification using vgg16 network*.

Vannier, M. W., & Haller, J. W. (1998). Biomedical image segmentation. *IEEE International Conference on Image Processing*, *2*, 20–24. https://doi.org/10.1201/9781315372273

Vijayalakshmi, A., & Rajesh Kanna, B. (2019). Deep learning approach to detect malaria from microscopic images. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-019-7162-y

Vishnu, M. (2016). *A Com mparison n of Featu ure Selec ction Met thods for r Machin ne Learni ing Base matic Mal larial Cel ll Recogn nition in Wholesli ide Imag ges*. 216–219.

Visuals, M. (2019). Foto de archivo. Retrieved September 20, 2020, from 2019 website: https://es.123rf.com/photo_94886848_joven-en-estudio-mirando-a-cámara.html

Wen, L., Li, X., & Gao, L. (2019). A transfer convolutional neural network for fault diagnosis based on ResNet-50. *Neural Computing and Applications*, *0123456789*. https://doi.org/10.1007/s00521-019-04097-w

WHO. (2020). World Health Organization. Retrieved August 12, 2020, from 14 January 2020 website: https://www.who.int/news-room/fact-sheets/detail/malaria

Yan, S. (2016). Understanding LSTM and Its Diagrams. Retrieved May 9, 2020, from Mar 13, 2016 website: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

Yang, Y., Yan, L. F., Zhang, X., Han, Y., Nan, H. Y., Hu, Y. C., … Wang, W. (2018). Glioma grading on conventional MR images: A deep learning study with transfer learning. *Frontiers in Neuroscience*, *12*(NOV), 1–10. https://doi.org/10.3389/fnins.2018.00804