# Implement Deep Learning Techniques To Detect Malaria Using Ibm Cloud

# CHAPTER -1

# INTRODUCTION

## 1.1.PROJECT DESCRIPTION :-

Malaria is a deadly, infectious mosquito-borne disease caused by Plasmodium parasites. These deadly parasites can live in your body for over a year without any problems! Thus, a delay in the right treatment can lead to complications and even death. Hence early and effective testing and detection of malaria can save lives. Convolutional neural networks (CNN) have the ability to automatically extract features and learn filters. A Machine Learning solution for the detection of malaria requires manual input of the parameters — for example, size, colour, the morphology of the cells, whereas implementing a Convolutional Neural Network (CNN) algorithm would greatly speed up prediction time while mirroring (or even exceeding) the accuracy of clinicians. In this project, we will be building a deep learning model that can detect and classify malaria disease. A web application is integrated into the model, from where user can upload an x-ray image and see the analysed results on User Interface

## 1.2.Technical Architecture :-

_*Fig-1.1- Technical Architecture*

# CHAPTER-2

## 2 . AIM AND SCOPE OF THE PRESENT INVESTIGATION

**2.1.Aim:** The main aim of this malaria detection is to address the challenges in the existing system by automating the process of malaria detection using ibm cloud and image processing.

**2.2.Scope of the project:** By the end of this project you will:

- know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks
- Gain a broad understanding of image data.
- Work with Sequential type of modeling
- Work with Keras capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

# CHAPTER-3

-

## 3.EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED

### 3.1.Pre requisites :-

To complete this project you should have the following software  and packages .

1. **Anaconda Navigator :**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform,  package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

*Fig- 3.1. Packages used in project*

To build Deep learning models you must require the following packages

2. **Tensor flow:** TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

3. **Keras :** Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

1. Flask: Web frame work used for building Web applications.

**3.2: Project Flow:**

- User interacts with User interface to upload image
- Uploaded image is analysed by the model which is integrated
- Once model analyses the uploaded image, the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
    - Collect the dataset or Create the dataset
- Data Preprocessing.
    - Import the ImageDataGenerator library
    - Configure ImageDataGenerator class
    - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
    - Import the model building Libraries
    - Initializing the model
    - Adding Input Layer
    - Adding Hidden Layer
    - Adding Output Layer
    - Configure the Learning Process
    - Training and testing the model
    - Optimize the Model
    - Save the Model
- Application Building
    - Create an HTML file
    - Build Python Code

**3.3. Project Structure:**

*Fig-3.2.Project Structure*

## 3.4.Dataset Collection

Artificial Intelligence is a data hunger technology, it depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Convolutional

Neural Networks, as it deals with images, we need training and testing data set. It is the actual data set used to train the model for performing various actions. In this activity lets focus of gathering the dataset

### 3.4.1:Download The Dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.
The dataset used for this project was obtained from Kaggle. Please refer to this link to download the dataset
The dataset contains two classes of images :

- Infected (Parasitized)
- Uninfected

## 3.5.Image Preprocessing

Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library.
- Configure ImageDataGenerator Class.
- Applying ImageDataGenerator functionality to the trainset and test set.

Note: The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data.

### 3.5.1:Import The ImageDataGenerator Library

mage data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

Let us import the ImageDataGenerator class from keras

***Fig-3.3:Import Image Data Generator Library***

### 3.5.2:Configure ImageDataGenerator Class

There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- Image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed.

***Fig-3.4:Image Data Generator Class***

validation_split: Optional float between 0 and 1, fraction of data to reserve for validation.

### 3.5.3:Apply ImageDataGenerator Functionality To Trainset And Testset

Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code
For Trainingset using flow_from_directory function.

This function will return batches of images from the subdirectories Infected and uninfected, together with labels 0 and 1 (0 corresponding to Infected and 1 corresponding to uninfected).

**Arguments:**

- **directory**: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- **batch_size**: Size of the batches of data. Default: 32.
- **target_size**: Size to resize images to after they are read from disk.
- **class_mode**:
  - 'int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).
  - 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).
  - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
  - None (no labels).

**Apply for trainset :**

***Fig-3.5:Apply for trainset***

We can see that there are 22047 images belonging to 2 classes (Infected and uninfected)

**Apply for Testset:**

***Fig-3.6:Apply for testset***

We can see that there are 5510 images belonging to 2 classes (Infected and uninfected)

## 3.6.Model Building

**We are using Deep neural networks for building the model**

This activity includes the following steps

- Import the model building Libraries
- Initializing the model
- Adding CNN Layers
- Adding Hidden Layer
- Adding Output Layer
- Configure the Learning Process
- Training and testing the model
- Saving the model

### 3.6.1:Import Libraries And Initialize The Model

Importing libraries is a very crucial step in our deep learning model building process. We have to define how our model will look and that requires.

***Fig-3.7: Import Libraries And Initialize The Model***

**Initialize the model:**

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.

Now, will initialize our model.

**3.7:Add CNN Layers**

In Deep Learning, Convolutional Neural Network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. Here we are using CNN as we are working with images.

- For information regarding CNN Layers refer to the link
- Link: https://victorzhou.com/blog/intro-to-cnns-part-1/
- We are adding a convolution layer with activation function as “relu” and
- with a small filter size (3,3) and number of filters (16) followed by a max pooling layer.
- Maxpool layer is used to downsample the input.
- Dropout layer is used to deactivate the neurons randomly.Dropout(0.2) indicates that 20 % of the neurons are deactivated.
- Flatten layer flattens the input. Does not affect the batch size.

***Fig-3.8:After Adding CNN Layers***

**3.8:Add Dense Layers**

The dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.

***Fig-3.9:Add Dense Layers***

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

***Fig-3.10:After Adding Dense Layers***

## 3.9:Configure Learning Process

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. Loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation process.
- Optimization is an important process which optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
- Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process

## 3.10:Train The Model

Now ,let us train our model with our image dataset. 80% of the image dataset is set for training and 20% for testing. Proper training gives higher accurate results.

**fit_generator** functions used to train a deep learning neural network
**Arguments:**

**steps_per_epoch** : it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your train dataset divided by the batch size.

**Epochs** : an integer and number of epochs we want to train our model for.

**validation_data** can be either:
- an inputs and targets list

- a generator
- an inputs, targets, and sample_weights list which can be used to evaluate
the loss and metrics for any model after any epoch has ended.

**validation_steps** :only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your test dataset divided by the validation batch size.

*Fig-3.12:Train the model*

### 3.11:Save The Model

The model is saved with .h5 extension as follows
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

*Fig-3.13:Save the model*

## 3.12:Test The Model

Once your traing is done, then lets test the model

Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data.
Load the saved model using load_model

***Fig-3.14.:Import load model***

Taking an image as input and checking the results

***Fig-3.15 :Import image***

By using the model we are predicting the output for the given input image

***Fig-3.16:Given input image***

If the output is 1 then Uninfected will be printed else Infected will e printed

## 3.13:- <u>Application Building</u> :-

In this section, let'sbuilding a web application that is integrated to the model and built.AUI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This sectionhas the following tasks

1. BuildingHTML Pages
2. Building server side script

### 3.13.1: <u>Building Html Pages</u> :-

Let's create our HTML structure.

The **HyperText Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by , written using angle brackets. Tags such as directly introduce content into the page. Other tags such as surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium, former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the element, in collaboration with javascript.

Now create a form with the class “credit-form.” Then divide our form in two sections. The first section is the form header where will have our form title, and the second is the form body where will have all the form elements and buttons.

***Fig-3.17:.Base Html Code***

*Fig-3.18:.Index Html Code*

*Fig-3.19:Css Code*

*Fig-3.20:Js code*

## 6.2: BuildPython source code :-

3.

*Fig-3.21:Build Python Code*

**3.13.2: Run the Application :-**

1.
2.
3.
4.

5.

Click on the predict button from the top right corner enter the inputs, click on the submit button,and see the result/prediction on theweb.

***Fig-3.22:.After Run the Application***

Now Enter the URL, localhost:5000 on the browser, you will redirect to cred.html page. Let'slook our cred page.

# CHAPTER-4

## 4.Results and Discussion, performance and analysis

_

**4.1:OUTPUT :-**

**After Enter the URL, localhost:5000 on the browser, you will redirect to cred.html page.**

Let’s look our cred page.

And get the out put like this.

***Fig-4.1:Output***

## 4.2:Result:-

***Fig-4.3: Not Infected with Pneumonia***

# CHAPTER-5

## 5. SUMMARY AND CONCLUSION

### 5.1:summary:-

2. Malaria parasites can be identified by examining under the microscope a drop of the patients blood,spread out as a blood smear on a microscope slide.prior to examination,specimen is stained to give the parasites a distinctive appearance.

3. Inorder to control malaria reduction of morbidity and mortality and eliminate it interruption of the transmission cycle,it is essential to identify and treat infected individuals early in the course of the illness.

### 5.2:CONCLUSION :-

1. **Malaria detection by itself is not an easy procedure and the availability of the right personnel across the globe is also a serious concern.**

2. **We looked at easy to build open-source techniques leveraging AI which can give us state-of-the-art accuracy in detecting malaria thus enabling AI for social good.**

**Let’s hope for more adoption of open-source AI capabilities across healthcare making it cheaper and accessible for everyone across the world**

### 5.3:REFERENCES :-

1. Refer the link below to download anaconda navigator.

   Link : https://youtu.be/1ra4zH2G4o0

1. Please refer to the link given below to download the data set and to know about the dataset:- https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria

2. How to install the required libraries visit the below video link to Install the necessary Packages https://youtu.be/akj3_wTploU

3. To create the flask application visit the link below

   https://youtu.be/lj4I_CvBnt0