

LYME DISEASE DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

A PROJECT REPORT

Submitted by

Varun Shukla (20BAI10292)
Deepika S. Srivastava (20BAI10319)
Gargi Choudhary (20BAI10358)
Janvi Bhalani (20BAI10368)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING



VIT[®]
BHOPAL
www.vitbhopal.ac.in

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTRIKALAN, SEHORE

MADHYA PRADESH - 466114

**VIT BHOPAL UNIVERSITY, KOTHRIKALAN,
SEHORE**

MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled “LYME DISEASE DETECTION USING NEURAL NETWORKS ” is the bonafide work of “Varun Shukla (20BAI10292), Deepika S. Srivastava (20BAI10319), Gargi Choudhary (20BAI10358), Janvi Bhalani (20BAI10368)” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR

Dr S Sountharajan

Program Chair, B.Tech AIML

School of Computing Science and Engineering

PROJECT GUIDE

Dr. N. Pazhaniraja

Teaching Fellow

School of Computing Science and Engineering

VIT Bhopal University

The Project Exhibition II Examination is held on 23rd April, 2022.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. S. Poonkuntran, Professor and Dean, School of Computer Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Dr. N. Pazhaniraja, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF FIGURES AND GRAPHS

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1(a)	Working of Convulational Neural Network	11
2(a)	Layers of CNN	14
3(a) , 3(b)	Sample Image	19, 19
3(c)	Types of Activation functions	21
4(a)	Types of Functions in CNN model	24
5,6	Implementation Screenshots	27

LIST OF GRAPHS

GRAPH NO.	TOPIC	PAGE NO.
5(a)	Training and Validation AUC	33
5(b)	Training and Validation Loss	33

ABSTRACT

Over the past four decades, Lyme disease has remained a virulent and pervasive illness, persisting all over the world. Recent increases in illness in many countries has sparked a renewed interest in improved Lyme diagnostics. While current standards of diagnosis are acceptable for the late stages of the disease, it remains difficult to accurately diagnose early forms of the illness. In addition, current diagnostic methods tend to be relatively expensive and require a large degree of laboratory-based analysis. A deep learning model has the potential to differentiate between a simple rash and that caused due to Lyme disease.

We propose a simple deep learning model that uses Convolutional Neural Network to predict the presence or absence of the disease using different optimizing and activation functions.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	List of Figures and Graphs	4
	Abstract	5
1	CHAPTER-1: PROJECT DESCRIPTION AND OUTLINE 1.1 Introduction 1.2 Motivation for the work 1.3 Introduction to the model 1.4 Problem Statement 1.5 Objective	9- 11
2	CHAPTER-2: RELATED WORK INVESTIGATION 2.1 Introduction 2.2 Core area of the project 2.3 Existing methods for disease detection	12 - 15

3	<p>CHAPTER-3:</p> <p>REQUIREMENT ARTIFACTS</p> <p>3.1 Introduction</p> <p>3.2 Hardware and Software requirements</p> <p>3.3 Specific Project requirements</p> <p>3.3.1 Data requirement</p> <p>3.3.2 Functions requirement</p> <p>3.3.3 Performance requirement</p> <p>3.4 Summary</p>	16 - 22
4	<p>CHAPTER-4:</p> <p>DESIGN METHODOLOGY AND ITS NOVELTY</p> <p>4.1 Methodology and goal</p> <p>4.2 Novelty of the project</p> <p>4.3 Functional modules design and analysis</p>	23 - 25
5	<p>CHAPTER-5:</p> <p>TECHNICAL IMPLEMENTATION & ANALYSIS</p> <p>5.1 Outline</p> <p>5.2 Technical coding and code solutions</p> <p>5.3 Prototype</p> <p>5.4 Test and validation</p> <p>5.5 Performance Analysis</p> <p>5.6 Model Enhancement Method</p>	26 - 33

6	CHAPTER-6: PROJECT OUTCOME AND APPLICABILITY 6.1 Outline 6.2 Key implementations outlines of the System 6.3 Significant project outcomes 6.4 Project applicability on Real-world applications	34 - 36
7	CHAPTER-7: CONCLUSIONS AND RECOMMENDATION 7.1 Limitation/Constraints of the System 7.2 Future Enhancements 7.3 Inference	37
	References	38 - 39

CHAPTER 1

PROJECT DESCRIPTION AND OUTLINE

1.1 INTRODUCTION

Lyme disease is a rapidly growing illness that remains poorly understood within the medical community. It is an infectious vector-borne disease, which can manifest itself in most cases with erythema migrans (EM) skin lesions. The current approaches for tick identification involves molecular methods like gene sequencing in a laboratory, which happens to be time consuming and expensive. Recent studies show that convolutional neural networks (CNNs) perform well to identify skin lesions from images. The main objective of this study is to extensively analyze the effectiveness of CNNs for diagnosing Lyme disease from images and to find out the best architecture considering the limited resources. According to some studies, this deep learning model competes on par with expert dermatologists for disease diagnosis.

In this project we attempt to design a model with an accuracy above 80% to detect the Lyme's disease.

1.2 MOTIVATION FOR THE WORK

Lyme's disease is a disease transmitted to humans through the bite of infected blacklegged ticks. It is a skin disease which hasn't been much talked or researched about. Since its infection can be seen by the naked eyes, making a model which can detect the presence of the disease is the motivation behind this project.

This model can be used by doctors to confirm their diagnosis.

1.3 INTRODUCTION TO THE MODEL

- **What are neural networks?**

Image Recognition Systems are more efficient when they are based on neural networks. The neural network architecture refers to the elements that are connected to make a network that is used for image recognition. The inspiration behind the working of neural networks is the human brain. Its working is based on the idea of how neurons pass signals around the human brain to process input into an output.

- **The use of CNN:**

Convolutional Neural Network is a part of deep learning and is a very efficient and effective way of recognizing images.

The underlying architecture of a CNN model includes several layers, the first being the input layer and the last being the output layer. In between lies the convolutional layer followed by the pooling layers and convolutional layers.

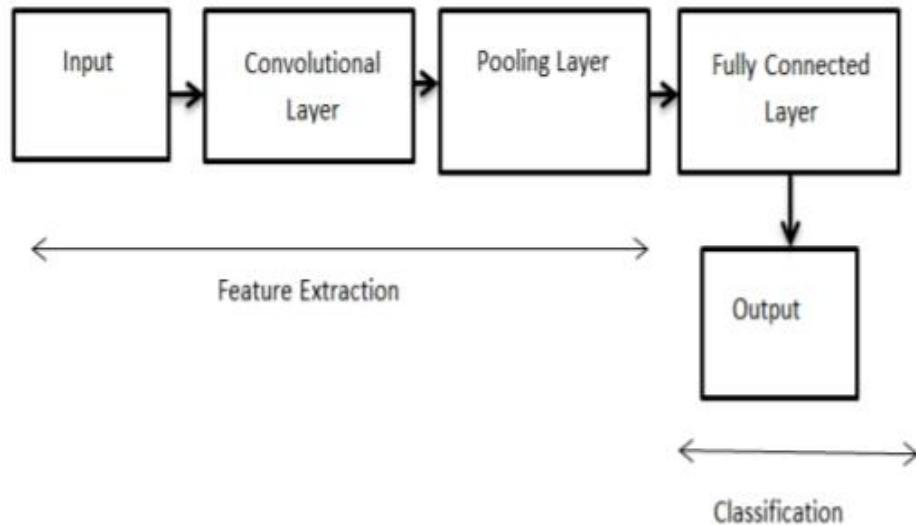


Fig. 1(a)

- **The use of ANN**

ANN is a model in machine learning which consists of a large number of artificial neurons connected to each other. The structure resembles that of axons in the human brain. This type of architecture consists of input, hidden and output layers. The networks are composed of many interconnected neurons working in unison to achieve a specific goal.

1.4 PROBLEM STATEMENT

To use machine learning algorithms to accurately predict Lyme Positive or Lyme Negative from scanned images of the affected area. To do this, we need to assume that all images containing spots/affected area have the same characteristics, and we can conclude that images

with these characteristics are of Lyme's disease or not. However, this hypothesis is ideal and does not always hold true in practice.

1.5 OBJECTIVE

- To design a system that recognizes disease in image using neural networks.
- To overcome the issues of inaccuracy presented in many image recognition systems by developing a more efficient system that would use efficient technology for recognizing affected areas from the images.
- To honour the usefulness of neural network technology.

CHAPTER - 2

RELATED WORK INVESTIGATION

2.1 OUTLINE

Lyme disease is a vector borne disease, which is spread by ticks. The most common sign of the infection is an expanding red rash, known as erythema migrans, which appears at the site of the tick bite about a week afterwards. During the early stage of illness, Lyme disease may be diagnosed clinically in patients who present with an EM rash. The diagnostic delays are usually associated with false negative test results or with positive test results that are dismissed by their clinicians as “false positives”.

2.2 CORE AREA OF THE PROJECT

This project revolves around the detection of Lyme Disease using Convolutional Neural Networks (CNN).

CONVOLUTIONAL NEURAL NETWORK

CNN is an artificial neural network mostly used for disease detection. It is a class of deep neural networks. The Convolutional neural networks are regularized versions of multilayer perceptron (MLP). They were developed based on the working of the neurons of the animal visual cortex. CNN are composed of multiple layers of artificial neurons. Each layer extracts features and pass it on to the next layer which detects more complex features. The final layer will give us

an output which would have the prediction of whether the input image prediction is positive or negative.

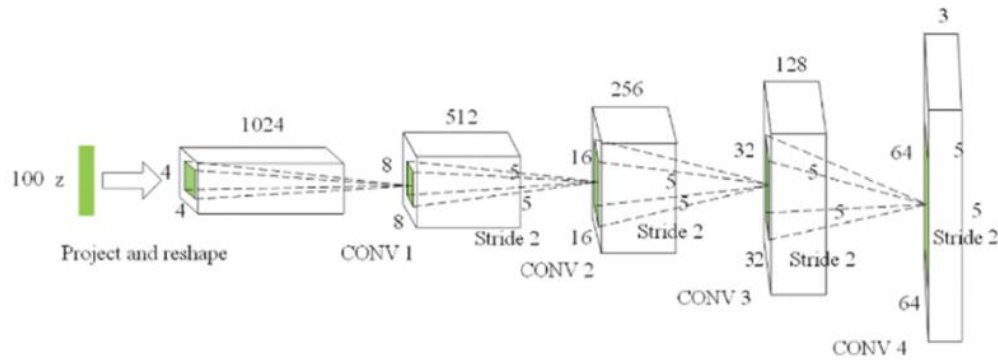


Fig. 2(a)

CNN consists of a lot of layers. These layers when used repeatedly, lead to a formation of a Deep Neural Network. The fundamental types of layers used to build a CNN are:

1. Input

This layer holds the uncooked pixel values of photograph and convert it to grayscale pics using 28x28 matrix of pixels.

2. Convolutional Layer

This layer gets the effects of the neuron layer that is linked to the enter regions. The wide variety of filters to be used in this layer is described here. Each filter may additionally be a 5x5 window that slider over the input records and receives the pixel with the most intensity as the output.

3. Rectified Linear Unit (ReLU) Layer

This layer applies an thing smart activation function on the picture records and makes use of again propagation techniques. ReLU function is utilized in order to preserve the equal values of the pixels and not being changed by means of the returned propagation.

4. Nadam Function

Nadam is an extension of the Adam algorithm that incorporates Nesterov momentum and can result in better performance of the optimization algorithm.

5. Pooling Layer

Down-sampling operation along the spatial dimensions (width, height), resulting in volume is utilized in this layer.

6. Fully Connected Layer

This layer is used to compute the score instructions that potential which class has the maximum score corresponding to the enter digits. The category label with the largest likelihood is chosen as the ultimate classification from the network and proven in the output.

2.3 EXISTING METHODS

Lab tests to identify antibodies to the bacteria can help confirm or rule out the diagnosis. These tests are most reliable a few weeks after an infection, after your body has had time to develop antibodies. They include:

- **Enzyme-linked immunosorbent assay (ELISA) test.** The test used most often to detect Lyme disease, ELISA detects antibodies to *B. burgdorferi*. But because it can sometimes provide false-positive results, it's not used as the sole basis for diagnosis.

This test might not be positive during the early stage of Lyme disease, but the rash is distinctive enough to make the diagnosis without further testing in people who live in areas infested with ticks that transmit Lyme disease.

- **Western blot test.** If the ELISA test is positive, this test is usually done to confirm the diagnosis. In this two-step approach, the Western blot detects antibodies to several proteins of *B. burgdorferi*.

BLOOD TESTS

Perfect blood tests do not exist. Even with the best tests, some tests will not detect a patient who has Lyme disease (ie, not sensitive enough) or the tests will falsely come back positive in a person who doesn't have Lyme disease (ie, the specificity is poor).

Direct Detection Tests

- **Culture:** The only way of knowing this for sure is if a person is still infected with a living spirochete is if the organism can be cultured. Unfortunately culturing Lyme disease is nearly impossible after the initial infection because the spirochete doesn't stay in the blood or spinal fluid very long.
- **PCR:** This test looks for evidence of the DNA of *Borrelia burgdorferi* in the blood or spinal fluid. This test however is not very sensitive for Lyme disease because the genetic material of the spirochete doesn't stay in the blood or spinal fluid very long
- **Antigen detection:** This test looks for pieces of the protein of *Borrelia burgdorferi*. Tests that use this technique have focused on the urine primarily. The assumption is that if pieces of the protein are still present, then the organism has been there recently. This is still considered a relatively experimental test.

CHAPTER 3

REQUIREMENT ARTIFACTS

3.1 INTRODUCTION

To develop a Lyme disease detection model, we require certain hardware and software components. These components help in acquiring the textual image in the form of an input and further determine whether the image is that of a simple rash or that caused by Lyme disease.

The code for this particular model can be executed on different IDE's like Python 3.7, Anaconda 3, or Google Collab. These IDE's have in-built libraries and requires no prior setup, which helps the users.

The CNN model has proved to be efficient in the determination model. This is because of the different functions it is associated with. In this model, activation functions, optimization functions and loss functions are proved to be efficient.

Activation functions are used to learn and approximate any kind of continuous and complex relationship between variables of the network. The most commonly used activation functions are ReLU, Softmax, tanH and sigmoid functions.

Optimizers are algorithms or methods used to change the attributes of the neural network in order to reduce the losses. Attributes may include learning rates and weights.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

SOFTWARE:

- **python 3.7**

Python is broadly utilized universally and is a high-level programming language. It enables one to express ideas in fewer lines of code. Python is a programming language that gives you a chance to work rapidly and coordinate frameworks more effectively.

- **Anaconda3**

Anaconda is used as IDE all through the implementation of this project. Anaconda is a free and open-source appropriation of the Python and R programming for logical figuring such as statistics science, AI applications, instruction of large-scale information, prescient investigation, etc.

- **Google Collab**

Google collab allows developers to write and execute Python codes through their browsers. It is a hosted Jupyter notebook that requires no setup. There are many advantages of using google collab in this model.

- ✓ Sharing
- ✓ Versioning
- ✓ Code snippets
- ✓ Price
- ✓ Forms for non-technical users
- ✓ Pre-installed libraries
- ✓ Free GPU and TPU use

HARDWARE:

- Graphical Processor Unit (GPU)
- Memory
- Storage
- Field Programmable Gate Array (FPGA)
- Scanning tools

3.3 DATA REQUIREMENT

The data contains images of the EM (Erythema Migrans) also known as the "Bull's Eye Rash" It is one of the most prominent symptoms of Lyme disease. Also in the data contains several other types of rashes which may be often confused with EM rash by doctors and most of the medical field.



Fig. 3(a)

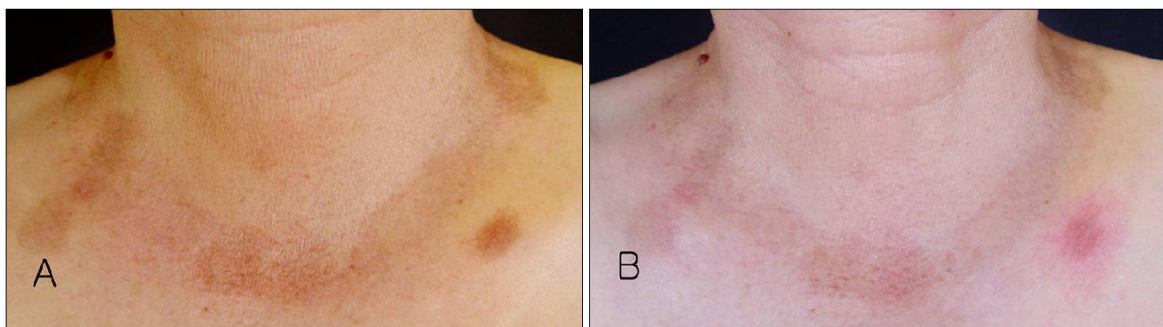


Fig. 3(b)

The dataset is divided into Lyme_Negative_By_Disease, Lyme_Positive_By_Disease, Train, Validation.

Further, Lyme_Negative_By_Disease consists of images of Drug Rashes, Pityriasis Rosea Rash and rashes caused by Ring Worms, whereas in Lyme_Positive_By_Disease consists of EM Rashes.

MODULES

- **Numpy**

Numpy is for cropping, padding, flipping, scaling and rotating images.

- **Pandas**

It has features that are used for exploring, cleaning, transforming and visualizing data.

- **Matplotlib**

It consists of several plots like line plot, bar plot, etc. through which we can visualise the various types of data.

- **Tensorflow**

It helps in easy model building by easing the process of acquiring data, training models, serving predictions and refining future results.

- **OpenCV**

It helps in loading images from a specified file. It also helps in doing complex tasks like identification, classification, etc.

3.4 FUNCTIONS REQUIREMENT

- **ACTIVATION FUNCTION**

Activation functions are an important part of any neural network. They are fundamentally used for determining the output of deep learning models, its accuracy and performance efficiency of the training model that can design or divide a huge scale neural network.

Activation function must be efficient, and it should reduce the computation time because the neural network sometimes trained on millions of data points.

- **OPTIMIZATION FUNCTIONS**

Optimization functions are used during backward propagation to update the attributes of neural networks. It also ensures that only that amount of data is provided to the network as much necessary.

The different types of optimization functions are: Adagrad, Adadelata, RmsProp, Adam, etc.

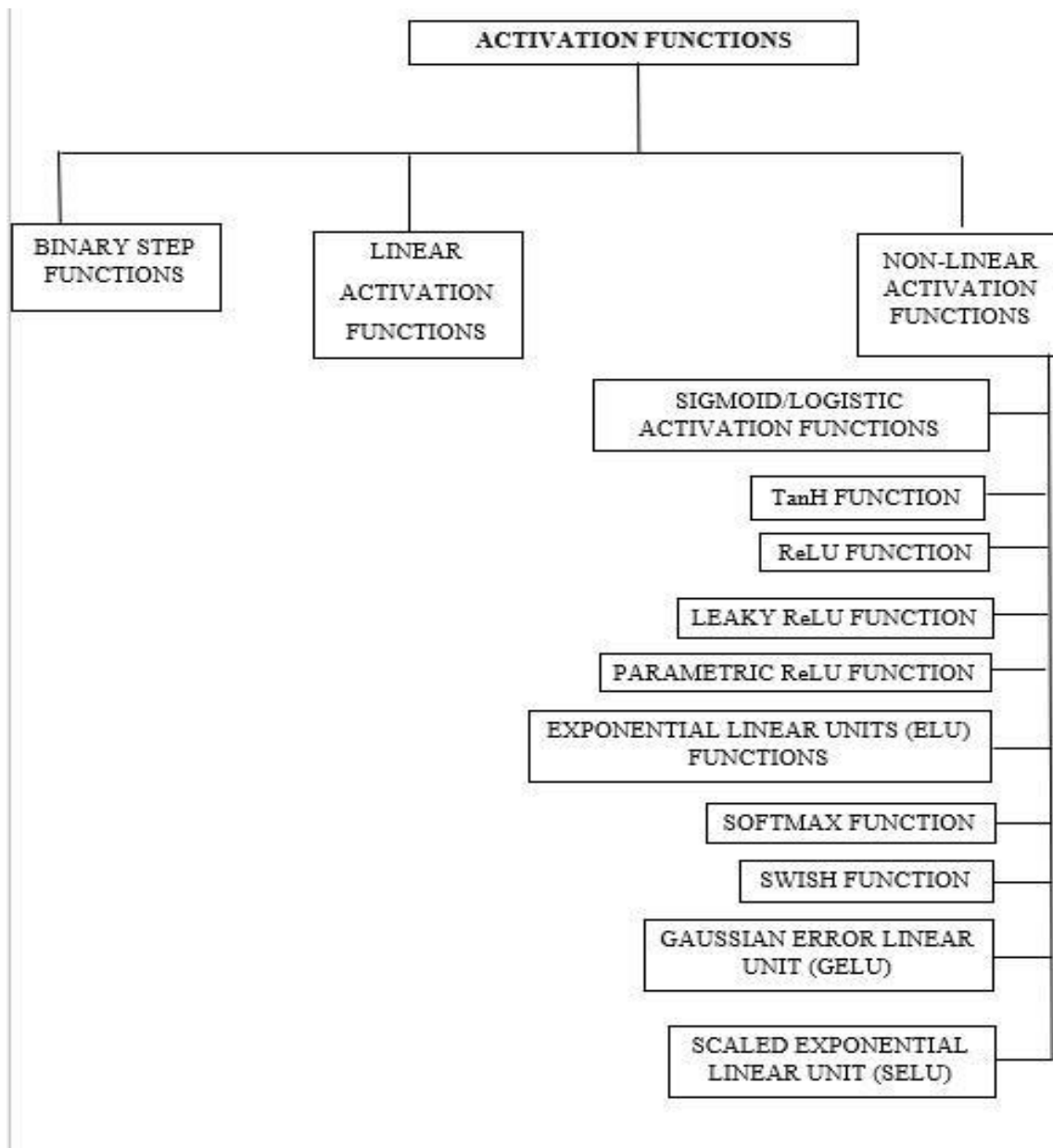


Fig. 3(c)

● LOSS FUNCTION

The loss function in a neural network quantifies the difference between the expected outcome and the outcome produced by our model.

3.5 PERFORMANCE REQUIREMENT

The performance of the model can be tested based on certain indications of correctness and functionality.

- To test individual units of the system, **unit testing** is used. It focusses on image acquisition, segmentation, feature extraction, etc. It helps in identifying bugs in the code, thus making it easy to fix the code.
- When individual units of the system are combined and tested as a group, it leads to **integration testing**. The main aim of such testing is to expose faults in the interaction of the integrated units.
- **Validation testing** determines whether the developmental process meets the specified requirements. It helps identifying defects in the system.
- In order to ensure that the systems graphical user interface meets the specified specifications, and it is user friendly, we perform **GUI testing**.

3.6 SUMMARY

For any model to run smoothly it requires certain software's and hardware's. For this particular model, we require Python 3.7, Anaconda 3 and Google Collab as the software's and Graphical Processor Unit (GPU), Memory, Storage, Field Programmable Gate Array (FPGA) and scanning tools as the main hardware components.

The dataset consisting of certain images that helps in differentiating between a simple rash like that caused due to certain drugs compared to the EM Rash.

Certain modules are imported which helps in easy detection.

Our model works based on Convolutional Neural Networks, thus, using certain activation, optimization, and loss functions.

To test the performance of the model, we can perform a series of test known as unit testing, integration testing, validation testing and GUI testing. These tests have a common motive of exposing faults, so that users can rectify them accordingly.

CHAPTER - 4

DESIGN METHODOLOGY AND ITS NOVELTY

4.1 METHODOLOGY AND GOAL

This project revolves around different algorithms under Neural Networks.

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

The primary goal of this project is to predict the result according to the given image i.e. if the disease is detected or not. We have trained the model using Convolutional Neural Network.

4.2 NOVELTY OF THE PROJECT

Based on our research on the topic image recognition specifically there is no high accuracy model present for detecting Lyme's disease. The disease is quite uncommon amongst the lot but it does exist and happens to a good number of population every year. So making a model to detect this disease will ease the work of doctors with their diagnosis.

4.3 FUNTIONAL MODULES DESIGN AND ANALYSIS

4.3.1. CNN Model

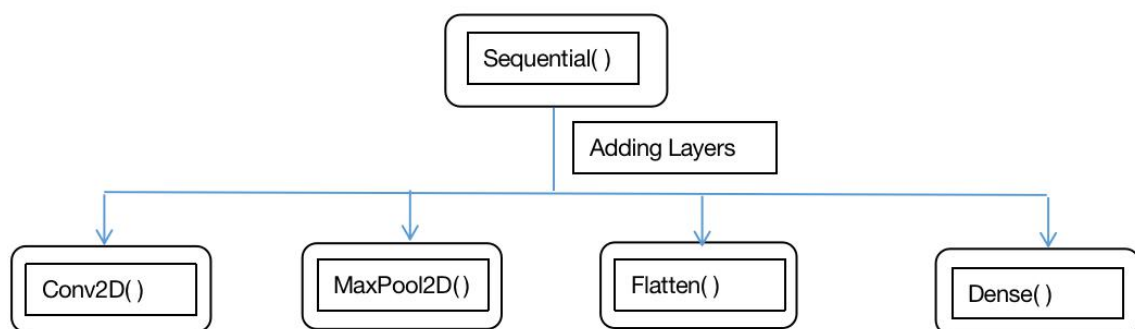


Fig. 4(a)

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

We add layers like Conv2d, MaxPool2D, Flatten and Dense.

Let us understand what these layers do :

1. **Conv2D()** - this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

Mandatory Conv2D parameter is the numbers of filters that convolutional layers will learn from.

It is an integer value and also determines the number of output filters in the convolution.

kernel_size

This parameter determines the dimensions of the kernel. Common dimensions include 1×1 , 3×3 , 5×5 , and 7×7 which can be passed as (1, 1), (3, 3), (5, 5), or (7, 7) tuples.

It is an integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window.

This parameter must be an odd integer.

padding

The padding parameter of the Keras Conv2D class can take one of two values: 'valid' or 'same'.

Setting the value to "valid" parameter means that the input volume is not zero-padded and the spatial dimensions are allowed to reduce via the natural application of convolution.

Activation

The activation parameter to the Conv2D class is simply a convenience parameter which allows you to supply a string, which specifies the name of the activation function you want to apply after performing the convolution.

2. `MaxPool2D()` - Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.
3. `Flatten()` - `flatten` function flattens the multi-dimensional input tensors into a single dimension, so you can model your input layer and build your neural network model, then pass those data into every single neuron of the model effectively.
4. `Dense()` - A Dense layer feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer. It's the most basic layer in neural networks.

CHAPTER 5

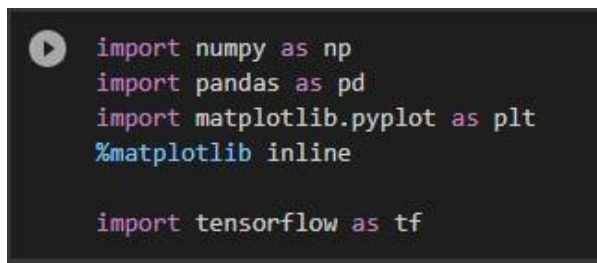
TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 OUTLINE

In this section of the report we will discuss about the coding aspect of the project. We have implemented the CNN code in Python Programming Language. We have also analyzed the accuracy and loss of the trained data using graphical representations.

5.2 TECHNICAL CODING AND CODE SOLUTIONS

Step 1: Importing libraries

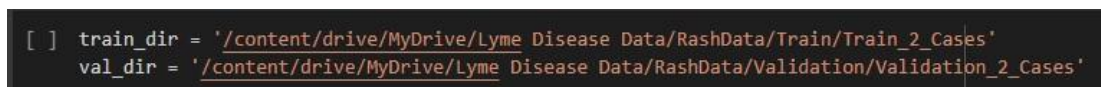


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf
```

Fig. 5(a)


Step 2: Loading the training and validation dataset directories.



```
[ ] train_dir = '/content/drive/MyDrive/Lyme Disease Data/RashData/Train/Train_2_Cases'
    val_dir = '/content/drive/MyDrive/Lyme Disease Data/RashData/Validation/Validation_2_Cases'
```

Fig. 5(b)

Step 3: Preprocessing of data



```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    validation_split=0.5,
    seed=42,
    subset='training',
    batch_size=batch_size,
    image_size=(img_height, img_width)
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    validation_split=0.5,
    seed=42,
    subset='validation',
    batch_size=batch_size,
    image_size=(img_height, img_width)
)
```

Fig. 5(c)

```

        batch_size=batch_size,
        image_size=(img_height, img_width)
    )

    test_ds = tf.keras.preprocessing.image_dataset_from_directory(
        val_dir,
        batch_size=batch_size,
        image_size=(img_height, img_width),
        seed=42
    )

```

Fig. 5(d)

Step 4: Visualizing dataset

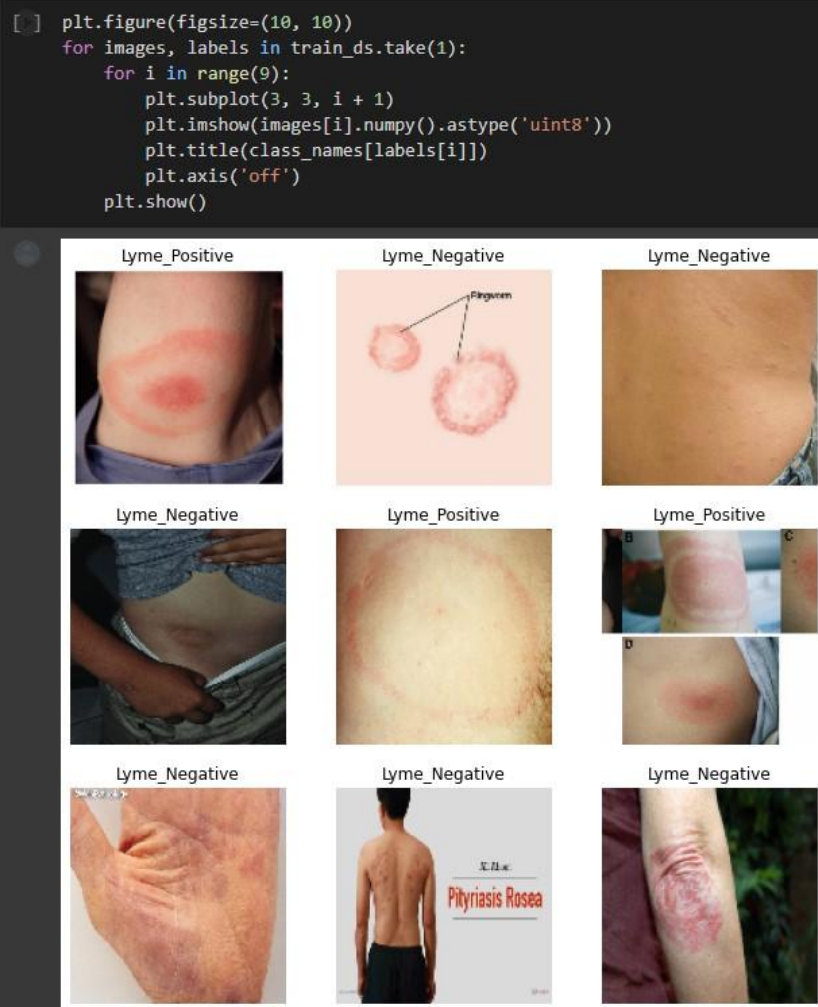


Fig. 5(e)

Step 5: Modeling

```
▶ AUTOTUNE = tf.data.experimental.AUTOTUNE

train_ds = train_ds.shuffle(1000).cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Fig. 5(f)

Step 6: Creating CNN model

```
[ ] inputs = tf.keras.Input(shape=(img_height, img_width, 3))
x = tf.keras.layers.experimental.preprocessing.Rescaling(1./255)(inputs)

x = tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Conv2D(128, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Conv2D(256, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)
```

Fig. 5(g)

```

x = tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu')(x)
x = tf.keras.layers.MaxPooling2D()(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Flatten()(x)

x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Dense(256, activation='relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)

x = tf.keras.layers.Dense(128, activation='relu')(x)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs, outputs)

```

Fig. 5(h)

Step 7: Compilation of model

```

[ ] model.compile(optimizer="Nadam", loss='binary_crossentropy', metrics=['accuracy', tf.keras.metrics.AUC(name='auc')])

```

Fig. 5(i)

Step 8: Defining epochs

Epochs - An epoch in machine learning means one complete pass of the training dataset through the algorithm.

```

▶ epochs = 200

history = model.fit(
    train_ds,
    validation_data=val_ds,
    batch_size=batch_size,
    epochs=epochs,
    callbacks=[tf.keras.callbacks.ReduceLROnPlateau(patience=100)],
    verbose=1
)

```

Fig. 5(j)

5.3 PROTOTYPE

<https://colab.research.google.com/drive/18PP18aQIMLiNkh88m5l-8XBZjcmfJl1je?usp=sharing>

5.4 TEST AND VALIDATION

```
[ ] plt.figure(figsize=(20, 10))

epochs_range = range(1, epochs + 1)
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_auc = history.history['auc']
val_auc = history.history['val_auc']

plt.subplot(1, 2, 1)
plt.plot(epochs_range, train_loss, label="Training Loss")
plt.plot(epochs_range, val_loss, label="Validation Loss")

plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training and Validation Loss")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs_range, train_auc, label="Training AUC", color='b')
plt.plot(epochs_range, val_auc, label="Validation AUC", color='r')

plt.xlabel("Epoch")
plt.ylabel("AUC")
plt.title("Training and Validation AUC")
plt.legend()

plt.show()
```

Fig. 5(k)

Evaluating loss and Area under curve

```
[ ] np.argmin(val_loss)

13

[ ] np.argmax(val_auc)

119

[ ] model.evaluate(test_ds)

3/3 [=====] - 10s 140ms/step - loss: 2.4587 - accuracy: 0.6782 - auc: 0.7282
[2.458721160888672, 0.6781609058380127, 0.7282134890556335]
```

Fig. 5(l)

Importing cv2 library for taking image inputs and predicting the results

```
[ ] import cv2
    from google.colab.patches import cv2_imshow

[ ] label_dict = {0:'Lyme_Negative', 1:'Lyme_Positive'}

[ ] # Prediction on external image...

    img = cv2.imread(r'/content/drive/MyDrive/download.jpg')
    img = cv2.resize(img, (128,128))
    img_final = np.reshape(img, (1,img_height, img_width,3))
```

Fig. 5(m)

Final output

```
[ ] model.predict(img_final)

    array([[0.07009801]], dtype=float32)

[ ] if model.predict(img_final) > 0.7:
    prediction=1
    else:
    prediction=0

[ ] img_pred = label_dict[prediction]
    cv2.putText(img, "Prediction: " + img_pred, (10,15), cv2.FONT_HERSHEY_TRIPLEX, 0.2, color = (0,0,230))
    cv2.putText(img, "Prediction: " + img_pred, (20,410), cv2.FONT_HERSHEY_DUPLEX, 1.3, color = (0,0,230))
    cv2_imshow(img)
```

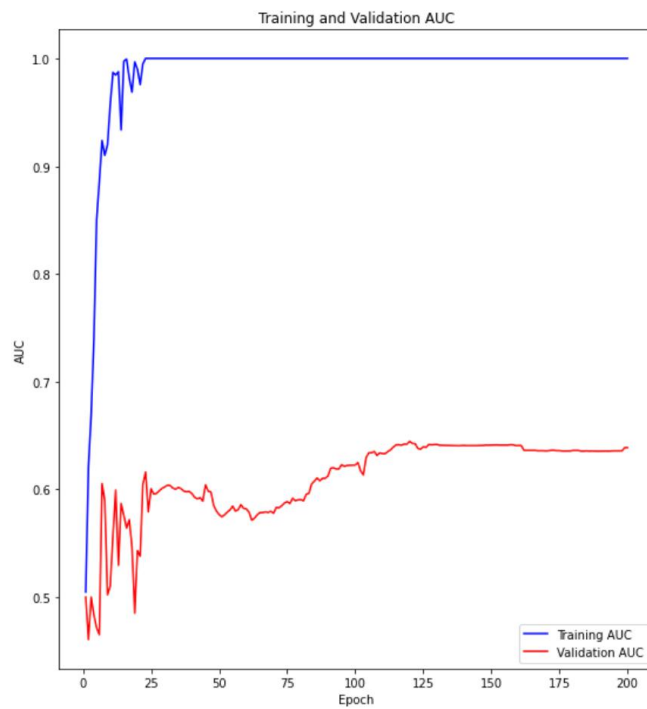
Fig. 5(n)



Fig. 5(o)

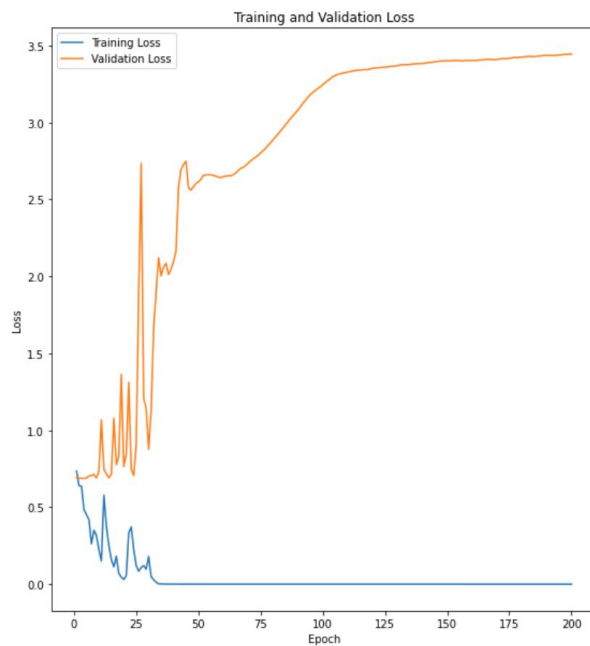
5.5 PERFORMANCE ANALYSIS

Training and Validation Area under curve



Graph. 5(a)

Training and validation loss



Graph 5(b)

5.6 MODEL ENHANCEMENT METHOD

We have noted the performance of our model by changing hyperparameters like batch sizes, optimizer, validation set size etc and concluded that the following are performing better in our model:

Batch_size=32

Validation set size = 50

Optimizer = Nadam

Gradient descent is an optimization algorithm that follows the negative gradient of an objective function in order to locate the minimum of the function.

A limitation of gradient descent is that the progress of the search can slow down if the gradient becomes flat or large curvature. Momentum can be added to gradient descent that incorporates some inertia to updates. This can be further improved by incorporating the gradient of the projected new position rather than the current position, called Nesterov's Accelerated Gradient (NAG) or Nesterov momentum.

Another limitation of gradient descent is that a single step size (learning rate) is used for all input variables. Extensions to gradient descent like the Adaptive Movement Estimation (Adam) algorithm that uses a separate step size for each input variable but may result in a step size that rapidly decreases to very small values.

Nesterov-accelerated Adaptive Moment Estimation, or the **Nadam**, is an extension of the Adam algorithm that incorporates Nesterov momentum and can result in better performance of the optimization algorithm.

CHAPTER 6

PROJECT OUTCOME AND APPLICABILITY

6.1 OUTLINE

A disease recognition model has numerous practical applications in healthcare, medical education, medical research, emergency diagnosis etc. The use of CNN helps in diagnosing the disease and could help in initial medical emergency care. This helps in regions and places with fewer medical services.

6.2 PROJECT APPLICABILITY ON REAL WORLD APPLICATIONS

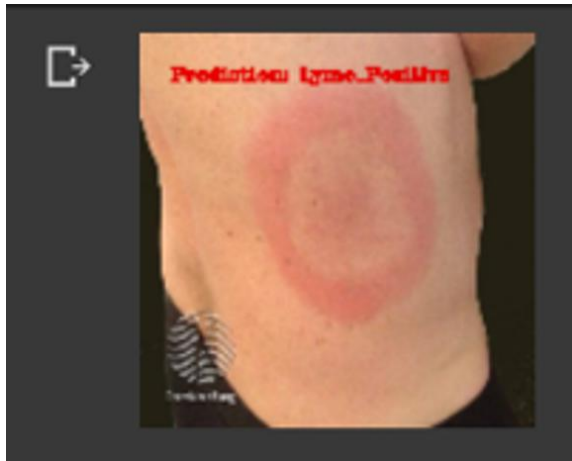
- In hospitals
- Emergency medical diagnosis
- Medical Schools
- Application in medical research after future development of project.

6.3 SIGNIFICANT PROJECT OUTCOMES

We created an accurate disease testing model that can interpret picture of any rash and differentiates it from lyme disease that causes a characteristic skin rash that is erythema migrans effectively.

The use of CNN, increases the level of accuracy as there are multiple convolutional layers and dense layers in our model.

For example,



[0.8794235]

Fig. 6. (a)

6.4 KEY IMPLEMENTATIONS OUTLINES OF THE SYSTEM

- **IN HOSPITALS:**

The main aim of using automated system to diagnose the disease is to aid the physician treating the patients at the hospitals. this can also be used by medical interns at the hospital. We can even use this for faster diagnosis so that the antibiotics could be provided as soon as possible.

- **IN MEDICAL SCHOOL**

Medical schools can use this tool for demo. This model can take images as input, and give a diagnosis as binary output that classifies into 2 classe that is 'Lyme Positive' and 'Lyme Negative'. By further adding some functions we can also show the accuracy by which the model is predicting the disease.

- **IN PLACES WHICH LACK HEALTHCARE SERVICES**

There are still many places where proper medical care is not available. In such places automated diagnosis tool can be used to detect Lyme disease in its first

stage so that it does not advance to third stage that is late disseminated which can have risk of brain disorders, arthritis, short-term memory loss, numbness in arms, legs, hands or feet and so on.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATION

7.1 CONSTRAINTS OF THE SYSTEM

1. This system may not give the perfect accuracy under any of the models.
2. Sometimes what might look legible to us might not be understood by the system and so it may give some garbage value as the output.
3. Unavailability of proper datasets imposed problems during the project and will also raise issues if we wish to use a dataset of some other disease images or any other skin image.
4. All kinds of images of rashes cannot be detected.

7.2 FUTURE ENHANCEMENTS

1. Create a model with accuracy greater than 98%.
2. Train the model for more data.
3. We can deploy the model and create a website for the bigger use.
4. Firstly work on recognizing between a simple rash or any other rash and lyme disease.

7.3 INFERENCE

We have developed a model that detects the presence of Lyme's. We have experienced how the trained model can differentiate between a simple rash and that caused due to Lyme's. The project can be taken further by creating an interface that not only takes in images as inputs but also displays the output as Lyme's positive or Lyme's negative. Although the project still fails to give a 100% accuracy. There is not just one model that can give us an accurate result. Different models have different pros and cons.

REFERENCES

1. <https://ieeexplore.ieee.org/document/9091175>
2. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7601730/>
2. <https://www.ijert.org/medical-image-classification-and-cancer-detection-using-deep-convolutional-neural-networks>
4. <https://www.kaggle.com/datasets/sshikamaru/lyme-disease-rashes>
5. <https://www.kaggle.com/datasets/sshikamaru/lyme-disease-rashes>
6. <https://www.mayoclinic.org/diseases-conditions/lyme-disease/diagnosis-treatment/drc-20374655#:~:text=Enzyme%2Dlinked%20immunosorbent%20assay%20>
7. <https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.mayoclinic.org/diseases-conditions/lyme-disease/diagnosis-treatment/drc-20374655%23:~:text%3DEnzyme%252Dlinked%2520immunosorbent%2520assay%2520>
8. <https://arxiv.org/pdf/2009.11931.pdf>
9. <https://www.hopkinsmedicine.org/news/newsroom/news-releases/ai-and-deep-learning-can-analyze-rash-selfies-for-better-lyme-disease-detection>
10. <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0260622>
11. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
12. <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>

13. <https://machinelearningmastery.com/gradient-descent-optimization-with-nadam-from-scratch/#:~:text=Nesterov%2Daccelerated%20Adaptive%20Moment%20Estimation,optimization%20with%20Nadam%20from%20scratch.>
14. <https://keras.io/api/optimizers/Nadam/>
15. <https://towardsdatascience.com/full-review-on-optimizing-neural-network-training-with-optimizer-9c1acc4dbe78>