Faculty of Engineering

Computer and Communications Program

Graduation Project submitted in partial fulfillment of the B. Sc. Degree.

Medical Image Diagnosis and Disease Detection Web Application using Deep Learning Technology

Supervised by:

Prof.Dr.Nour Ismail

Presented by:

Mohamed Fawzy

Merna Adel

Amr Sherief

Yasser Mohamed

Abdulrahman Mohamed

Acknowledgments

To begin with, we would like to thank God for the support given throughout our graduation project. We'd also like to thank our spectacular supervisor, Dr.Nour Ismail for his endless support, constant follow up, and academic help given throughout our journey in working on our graduation project. Finally, we would like to thank our families for their constant love, support, and encouragement throughout the time we spent working too. Due to you all we were able to reach our goal of that interesting journey.

Thank you.

Abstract

The vast evolving of machine learning technologies had been noticeably obvious during the last few years in which it is hard to find an application nowadays that doesn't include machine learning models as part of its back-end design. However, as part of the wide machine learning family, deep learning showed a great success in different real world applications providing methods of neural network, and image processing or detection which opened our eyes to use that kind of technology as the back bone of our web application back-end.

Our main idea was composed of implementing a web application using up-to-date web development tools that provides services for clients that includes disease diagnoses using image processing in which a client provides as an input some visual data which is processed using deep learning models to give as an output a full diagnoses of what the client (patient in our case) is suffering from along with the diagnoses accuracy percentage.

Table Of Contents

Chapter 1: Introduction

1.1. Overview

Deep learning, which is a subfield of machine learning (ML), has seen a dramatic revival in the past few years. Such revival is mainly driven by the huge increases in the computational powers of computers and machines, and also the availability of massive new datasets which weren't available before. The field has witnessed fast paced advances in the ability of machines to understand and manipulate data, including images.

Health care and medicine were one of the major fields which benefited greatly from such rise because of the huge volume of data being generated and the high increase in performance and capabilities of the medical machines.

The most common models of deep learning used widely in the medical field are trained using supervised learning, in which datasets are labeled. The term labeled dataset refers to the case in which datasets are composed of input data points (e.g. skin lesion images) and corresponding output data labels (e.g. benign or malignant).

1.2. Deep learning for Computer Vision

Some of the greatest successes of deep learning have been in the field of computer vision (CV). Computer Vision focuses on understanding both images and videos, and deals with the task of object detection, classification and segmentation which are of major importance in the medical field if employed in determining whether a radiograph of a patient contains malignant tumors or diseases indication.
Many studies have demonstrated promising results in complex diagnostics in dermatology, radiology, ophthalmology, and pathology. Deep-learning systems

can be used as a second opinion to physicians and highlight concerning areas in images. It's worth noting that different and various deep learning models have achieved physician-level accuracies at different variety of segmentation and diagnostic tasks including diabetic retinopathy, breast lesion detection in mammograms, spinal analysis with magnetic resonance imaging and cardiovascular risk. A single deep learning model has even been shown to be effective at diagnosis across medical modalities (e.g., radiology and ophthalmology).

Chapter 2 Bones Abnormality Classification from X Ray Images Using Deep Learning.

2.1. Problem Statement

2.1.1. Explanation

We introduce MURA [1], a large dataset of musculoskeletal radiographs containing 40,005 images from 14,863 studies. Where each image is manually labeled by radiologists as either normal or abnormal. Large, high quality datasets have The state of the art for this work is implemented The state of the art for this work is implemented played a critical role in driving progress of fields with deep learning methods.

Musculoskeletal conditions affect more than 1.7 billion people worldwide, and are the most common cause of severe, long term pain and disability, with 30 million emergency department visits annually and increasing. This work is done to help in the healthcare field. A good classification model maybe a way to fasten the construction of tools that help in early diagnosis and alert on patient‘s hands, even far isolated patients, where few do doctors can reach.

The MURA abnormality detection task is a binary classification task, where the input is an upper extremity radiograph study and the expected output is a binary

label y ∈ {0, 1} indicating whether the study is normal or abnormal, respectively. There are many algorithms in this field that approach this problem, but of course there is a significant difference between the shallow and deep methods in machine learning.

2.1.2. State of the Art

The state of the art for this work was implemented by team of different departments in Stanford University and the work in one of our approach is based upon their assumptions.

2.1.3. Prior Work

There are many algorithms and tools the help in the task of diseases detection, many approaches have been made over the past years like classic machine learning as K-Nearest Neighbors (KNN) , Decision Trees and Support Vector Machine (SVM), the another type is using deep learning like Convolution Neural Network (CNN) and they had been prove to get good result.

The lack of quality and scarcity of open data is a common thing in the solutions to the medical field problems using deep learning, because many hospitals save the records of the patients, however don't publish them because of privacy issues. Team of different departments in Stanford University worked in collecting this data and then manually labeled data by radiologists as normal or abnormal. They have trained their data using 196-layer DenseNet to detect and localize abnormalities. They compared their model and radiologists on the Cohen's kappa statistic.

The function Cohen Kappa Score computes Cohen's kappa statistic. This measure is intended to compare labels by different human annotators, not a classifier versus a ground truth. The kappa score is a number between -1 and 1. Scores above .8 are generally considered good agreement zero or lower means no agreement (practically random labels).

$$Kappa = (OA - AC) / (1 - AC)$$

OA: observed agreement AC: Agreement by chance

2.2. Dataset

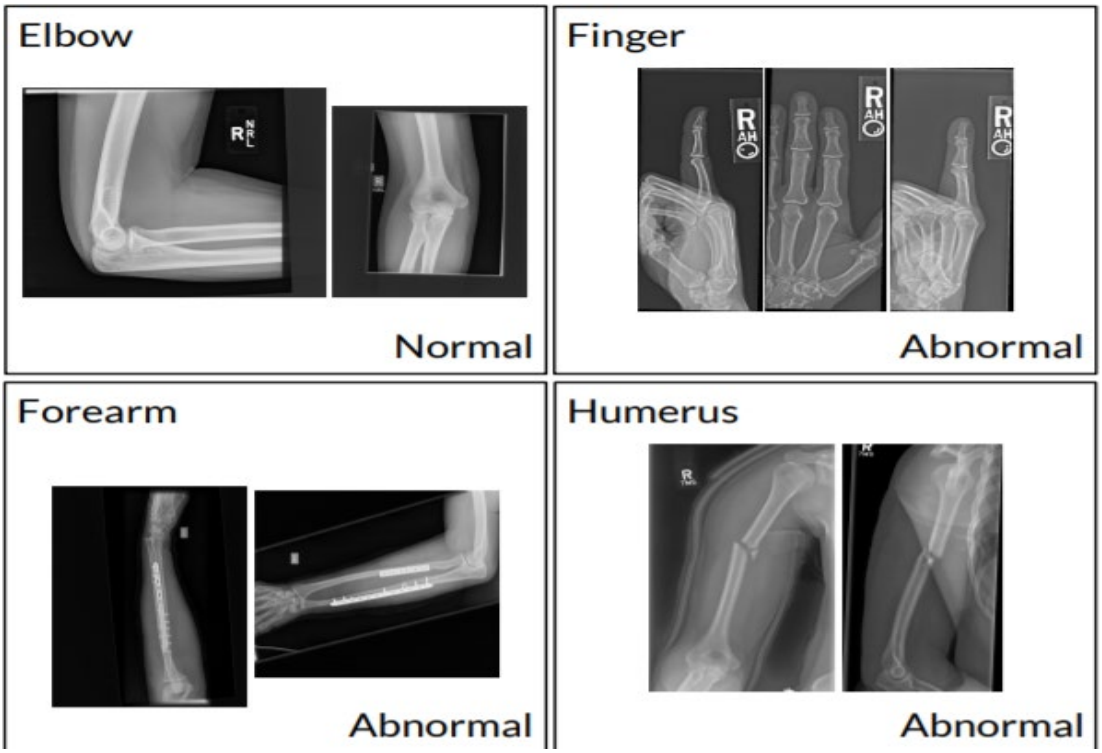


Figure 2.1: Examples of dataset

The MURA dataset contains 14,863 musculoskeletal studies of the upper extremity, where each study contains one or more views and is manually labeled

by radiologists as either normal or abnormal. These examples in (Figure) show a normal elbow study (left), an abnormal finger study with degenerative changes (middle left), an abnormal forearm study (middle right) demonstrating operative plate and screw fixation of radial and ulnar fractures, and an abnormal humerus study with a fracture (right).

2.2.1. Data collection

Data collected from Stanford Hospital. We assembled a dataset of musculoskeletal radiographs consisting of 14,863 studies from 12,173 patients, with a total of 40,005 multi-view radiographic images. Each belongs to one of seven standard upper extremity radiographic study types: elbow, finger, forearm, hand, humerus, shoulder, and wrist. Figure summarizes the distribution of normal and abnormal studies.

Study	Train		Validation		Total
	Normal	Abnormal	Normal	Abnormal	
Elbow	1094	660	92	66	1912
Finger	1280	655	92	83	2110
Hand	1497	521	101	66	2185
Humerus	321	271	68	67	727
Forearm	590	287	69	64	1010
Shoulder	1364	1457	99	95	3015
Wrist	2134	1326	140	97	3697
Total No. of Studies	8280	5177	661	538	14656

Figure 2.2: Distribution of normal and abnormal studies.

2.2.2. Data Labels

Each study was manually labeled as normal or abnormal by board certified radiologists from the Stanford Hospital at the time of clinical radiographic interpretation in the diagnostic radiology environment between 2001 and 2012.

2.2.3. Metadata

The metadata of those images are contained in Excel sheet contain the path of each patient study and which the study is positive or negative. The first five samples are available in Figure 1.3

	images_paths	labels
0	MURA-v1.1/train/XR_SHOULDER/patient00001/study...	1
1	MURA-v1.1/train/XR_SHOULDER/patient00002/study...	1
2	MURA-v1.1/train/XR_SHOULDER/patient00003/study...	1
3	MURA-v1.1/train/XR_SHOULDER/patient00004/study...	1
4	MURA-v1.1/train/XR_SHOULDER/patient00005/study...	1

Figure 2.3: Meta Data of the dataset

2.2.4. Techniques In Deep Learning

In deep Learning we have many approaches to work with our data in our problem we work with two approaches. The first one is Transfer Learning, another one is to run our model from scratch. Now we will illustrates the details of the two approaches.

2.2.4.1. Transfer Learning [2]

In transfer learning we have several approaches:

a) First using pre-trained model as a feature extractor and then using them to train our new dataset then replace the final layers with the required output for the new data and with fine tuning and backpropagations techniques to update parameters of the final layers.

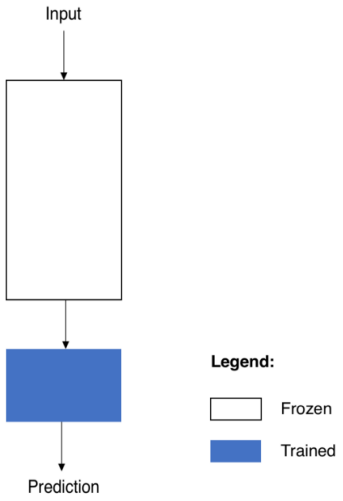


Figure 2.4: Architecture of a model based on CNN as feature extractor with CNN classifier.

b) The second approach that we can use the pre-trained as a feature extractor but in this case the pre-trained will be in the model without updating parameters of the pre-trained model or we can choose some layers in the pre-trained to update during the fit the whole model we choose the second one if the new data is so far different to the data that was used to train the pre-trained model.

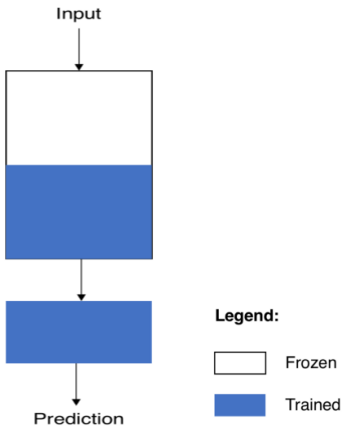


Figure 2.5: Architecture of a model based on CNN as feature extractor in the model with CNN classifier.

2.2.4.2 Run from scratch

In this approach we train all the model layers and update the parameters of the model to fit our data but this techniques need much time and high computational power and we will illustrate the implementation of the model later.

Figure 2.6: Architecture of a model based on run from scratch

2.2.5 Data augmentation

This techniques helps to increase number of images to train the model. This is done by doing random transformation on the available data like cropping rotating, zooming and flipping, this will help us to reduce the model overfitting.

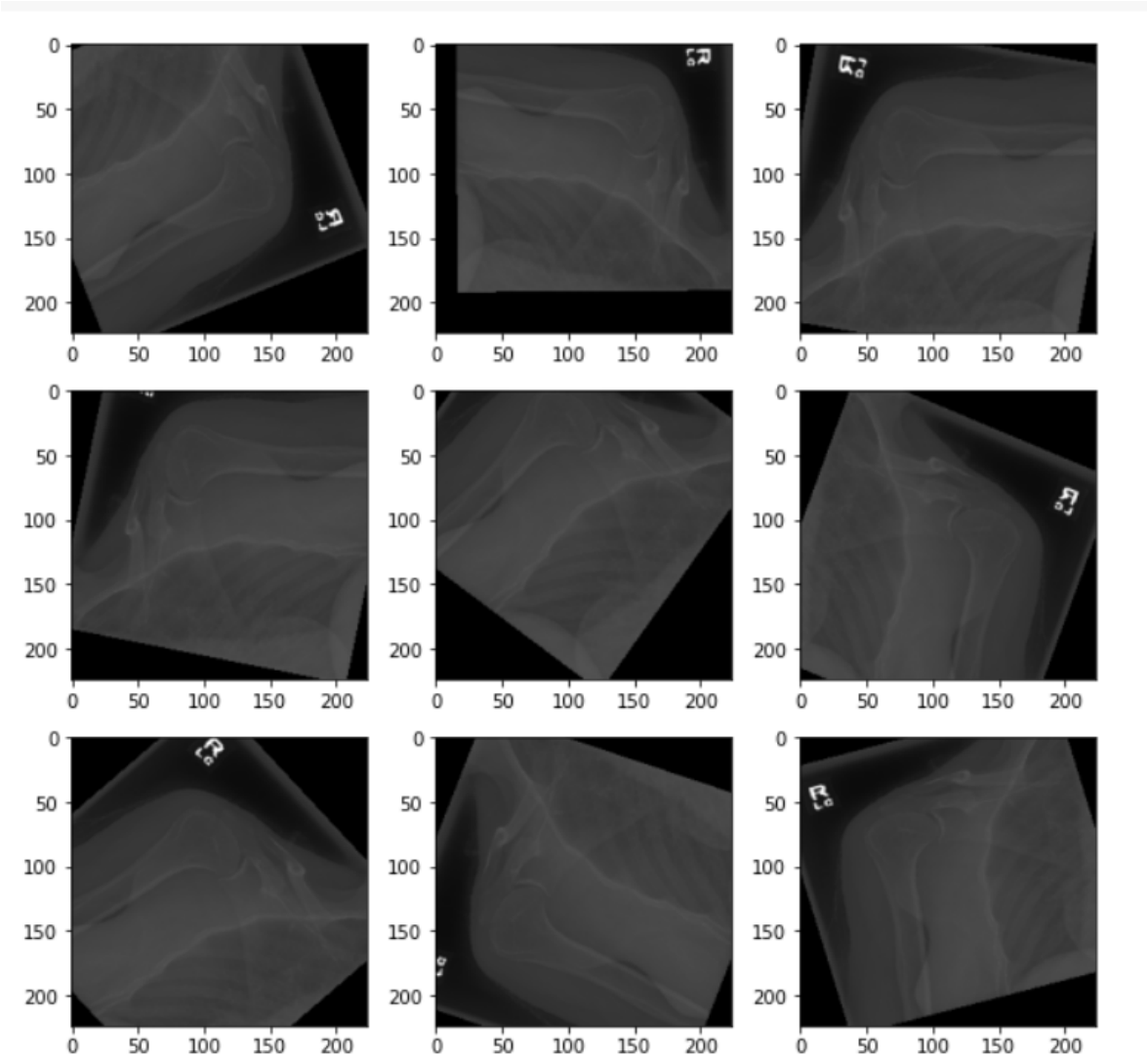


Figure 2.7: Data Augmentation on shoulder image.

2.2.6 Data Exploration

In this section we will illustrate the number of positive and negative images in each bone type in train set and validation set**.**

Table 2.1: Train Dataset

Bone	Number of positive images	Number of negative images	Total number of images
Elbow	2006	2925	4931
Finger	1968	3138	5106
Hand	1484	4059	5543
Humerus	599	673	1272
Forearm	661	1164	1825
Shoulder	4168	4211	8379
Wrist	3987	5765	9752
Total number of images			36808

Table 2.2: Validation Dataset

Bone	Number of positive images	Number of negative images	Total number of images
Elbow	230	235	465
Finger	247	214	461
Hand	189	271	460
Humerus	140	148	288
Forearm	151	150	301
Shoulder	278	285	563

Wrist	**295**	**364**	**659**
Total number of images			**3197**

For test dataset we split the validation dataset two parts one for validation data and another data for test data with ratio 1/10 of the validation data for the test data.

2.3. System Architecture

We use in our problem MobileNet as a pre-trained model on ImageNet in our Transfer Learning approach. It has good result in our training dataset and it was faster in real training prediction time than DenseNet Model.

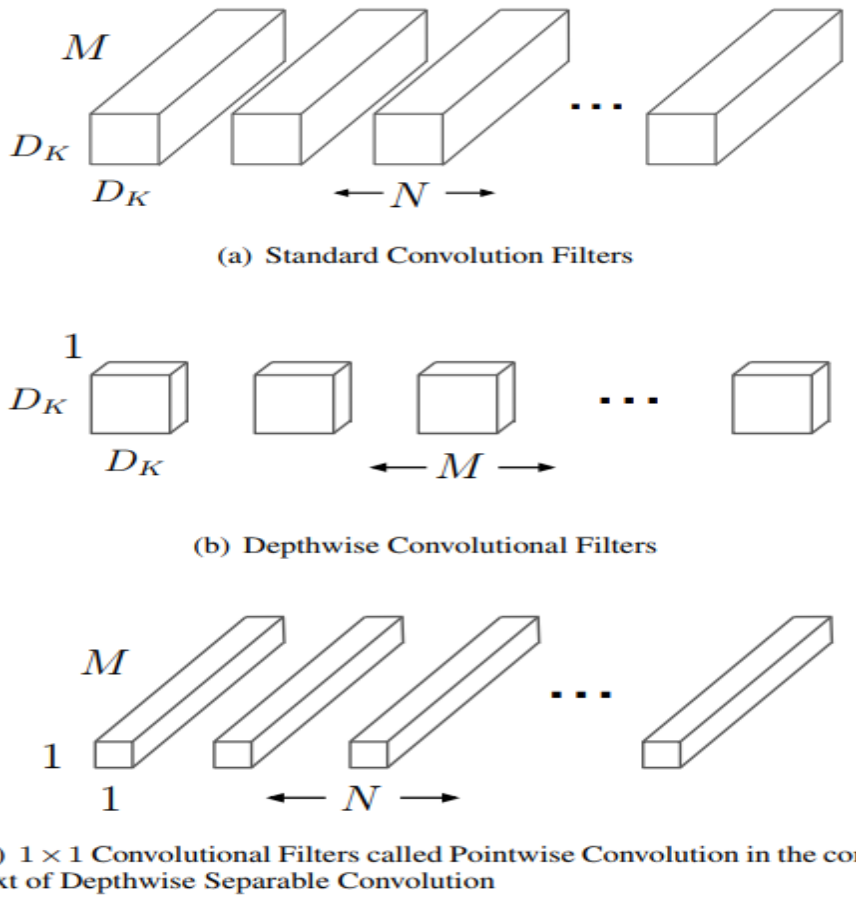
2.3.1. MobileNet Description

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard et al, 2017. The source paper for Mobilenet [3] **.**

2.3.2. Why Using MobileNet?

It uses depth wise separable convolutions which basically means it performs a single convolution on each color channel rather than combining all three and flattening it. Or as the authors of the paper explain clearly: " For MobileNets the depth wise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depth wise

convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depth wise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size." In Figure 2.8: Difference between different optimizers explains the difference between point wise and depth wise convolution.



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2.8: Difference between pointwise and depth wise convolutions

2.3.3. MobileNet Architecture

MobileNet is having 30 layers with:

1. Convolutional layer with stride 2.
2. Depth wise layer.

3. Pointwise layer that doubles the number of channels.

4. Depth wise layer with stride 2.

5. Pointwise layer that doubles the number of channels.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 2.9: Full architecture of MobileNet

2.3.4. Implementation of the Model

2.3.4.1 Implementation of Pre-Trained Model (MobileNet) as Feature Extractor in Model.

1- We remove the last 3 layers.

2-Adding Global Average Pooling layer [4] instead of fully connected layer. The Global Average Pooling mechanism Computes the mean value for each feature map and supplies it to a Dense layers.

3- Adding Three Dense layers with RELU activation function and the last one with SIGMOID function.

4-freezing layers that we do not want to update their parameters and leave about 17 layers to update their parameters.

2.3.4.2 Implementation of Pre-Trained Model as Feature Extractor Preprocessor

1- Using MobileNet as Feature extractor and remove classifier layers from the model.

2- Prepare classifier model that contains two dense layers with dropout layers with rate equal to 0.3.

3- Final layer is dense layer with SIGMOID activation function.

2.3.5 Implementation of Scratch Model.

We use (VGG-16) as a scratch model for our problem. In the next Figure the full architecture of VGG-16 model and scratch model.

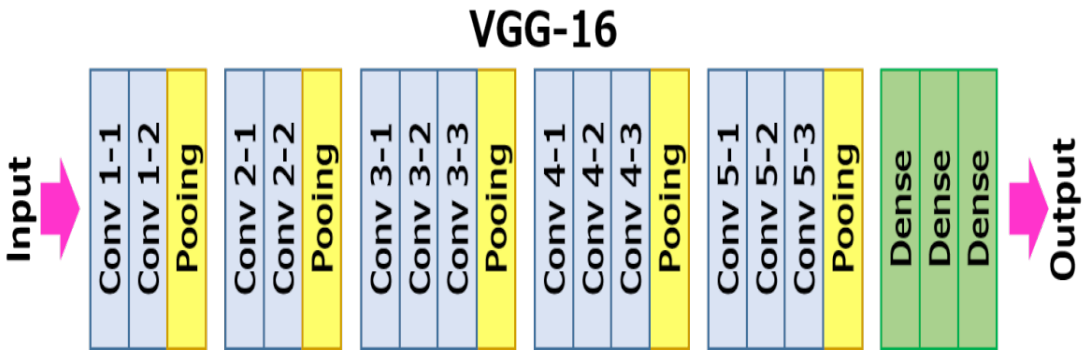


Figure 2.10: Full architecture of VGG-16

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 32)	896
conv2d_2 (Conv2D)	(None, 224, 224, 32)	9248
batch_normalization_1 (Batch	(None, 224, 224, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 32)	0
dropout_1 (Dropout)	(None, 112, 112, 32)	0
conv2d_3 (Conv2D)	(None, 112, 112, 64)	18496
conv2d_4 (Conv2D)	(None, 112, 112, 64)	36928
batch_normalization_2 (Batch	(None, 112, 112, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 56, 56, 64)	0
dropout_2 (Dropout)	(None, 56, 56, 64)	0
conv2d_5 (Conv2D)	(None, 56, 56, 128)	73856
conv2d_6 (Conv2D)	(None, 56, 56, 128)	147584
batch_normalization_3 (Batch	(None, 56, 56, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 128)	0
dropout_3 (Dropout)	(None, 28, 28, 128)	0
flatten_1 (Flatten)	(None, 100352)	0
dense_1 (Dense)	(None, 1024)	102761472
batch_normalization_4 (Batch	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 128)	131200
batch_normalization_5 (Batch	(None, 128)	512
dropout_5 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Figure 2.11: Full architecture of scratch model

2.3.6. Optimization Algorithm

Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. We use Adam [5] Optimizer in backward function to update our weights of features. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

The intuition behind the Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta.

To update the parameter:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t.$$

Equation 2.1: Update parameters of the model

M(t) and V(t) **are values of the first moment which is the Mean and the second moment which is the uncentered variance of the gradients respectively.**

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}.$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

Equation 2.2: First and second order of momentum

According to Kingma et al., 2014, the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

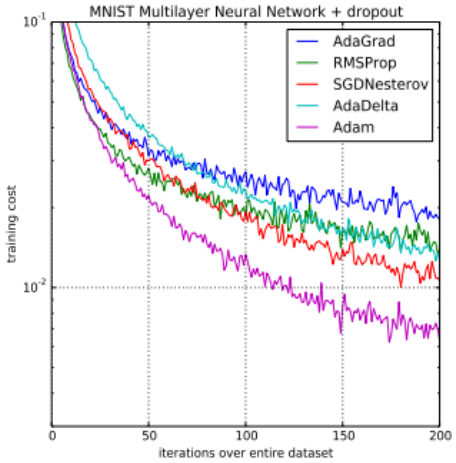


Figure 2.12: Difference between different optimizers

2.3.7. Loss Function [6]

As part of the optimization algorithm, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a loss function that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.

The purpose of loss functions is to compute the quantity that a model should seek to minimize during training. Neural network models learn a mapping from inputs to outputs from examples and the choice of loss function must match the framing of the specific predictive modeling problem, such as classification or regression.

Binary classification are those predictive modeling problems where examples are assigned one of two labels. The problem is often framed as predicting a value of 0 or 1 for the first or second class and is often implemented as predicting the probability of the example belonging to class value 1.

We use Binary Cross-Entropy loss. Cross-entropy is the default loss function to use for binary classification problems. Mathematically, it is the preferred loss function under the inference framework of maximum likelihood. It is the loss

function to be evaluated first and only changed if you have a good reason. Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect cross-entropy value is 0.

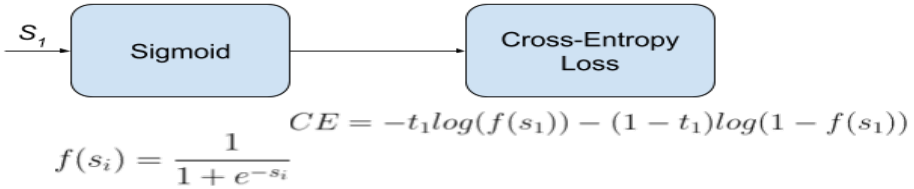


Figure 2.13: Binary cross entropy

The function requires that the output layer is configured with a single node and a sigmoid activation in order to predict the probability for class 1.

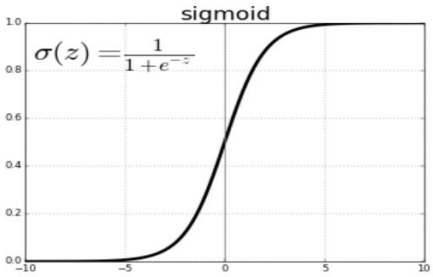


Figure 2.14: Sigmoid function

2.3.8 Callbacks [7]

A callback is a set of functions to be applied at given stages of the training procedure. You can use callbacks to get a view on internal states and statistics of the model during training. We use different types of Callback in feature extractor model we use ModelCheckPoint, EarlyStopping and LearningRateScheduler. In feature extractor in the model we use only Model Check Point and LearningRateScheduler and the same in the scratch model.

2.3.8.1 EarlyStopping

Overfitting is a nightmare for Machine Learning practitioners. One way to avoid overfitting is to terminate the process early. The EarlyStopping function has

various metrics/arguments that you can modify to set up when the training process should stop.

2.3.8.2 ModelCheckpoint

This callback saves the model after every epoch.

2.3.8.3 LearningRateScheduler

This one is pretty straightforward: it adjusts the learning rate over time using a schedule that you already write beforehand. This function returns the desired learning rate (output) based on the current epoch (epoch index as input).

2.3.9 Hyperparameters

In this section we decide Starting Learning Rate, Batch Size and steps of training in the next tables we illustrate Hyperparameters values.

Table 2.3:Hyperparameters of first approach in Transfer Learning

Hyperparameters	values
Starting Learning Rate	0.001
Batch Size	128
steps of training	288
Number of epochs to reduce LR	4
Factor of reducing Learning Rate	(LR*0.1)

Table 2.4: Hyperparameters of second approach in Transfer Learning

Hyperparameters	values
Starting Learning Rate	0.001

Batch Size	128
steps of training	288
Number of epochs to reduce LR	8
Factor of reducing Learning Rate	(LR*0.1)

Table 2.4: Hyperparameters of scratch model

Hyperparameters	values
Starting Learning Rate	0.001
Batch Size	128
steps of training	288
Number of epochs to reduce LR	12
Factor of reducing Learning Rate	(LR*0.1)

2.4. Results

2.4.1 Metrics

List of metrics to be evaluated by the model during training and testing. Each of this can be a string (name of a built-in function).

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.

$$\text{Accuracy} = TP+TN/TP+FP+FN+TN$$

(Equation 2.3: Compute Accuracy)

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy,

especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different.

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

(Equation 2.4: Compute F1 Score)

2.4.2 Result of Feature Extractor Model followed by classifier model.

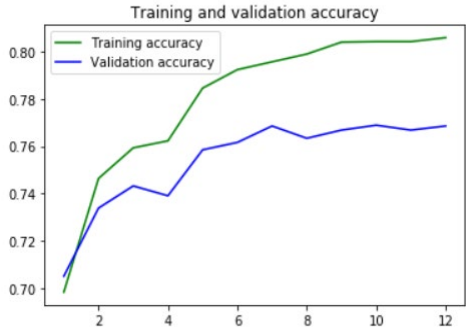


Figure 2.15: training and validation Accuracy

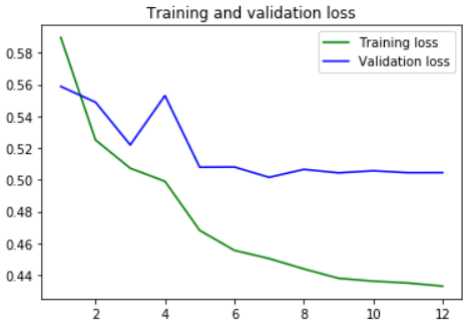


Figure 2.16:training and validation loss

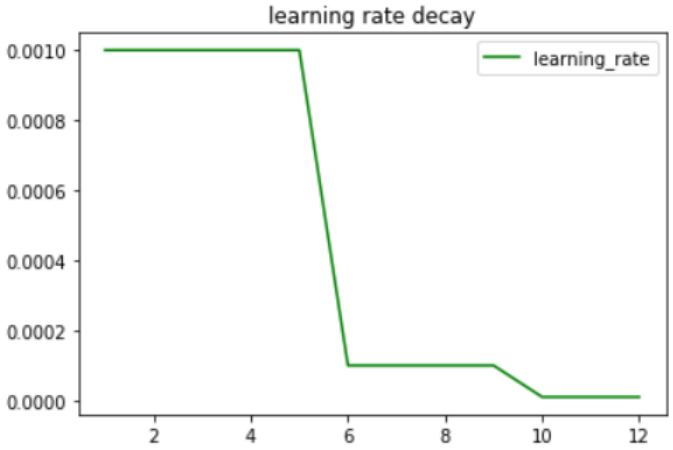


Figure 2.17: Learning Rate Decay

Accuracy of the Model : 0.76 **F1-Score of the Model : 0.74**

2.4.3 Result of Pre-Trained Model (MobileNet) as Feature Extractor in Model.

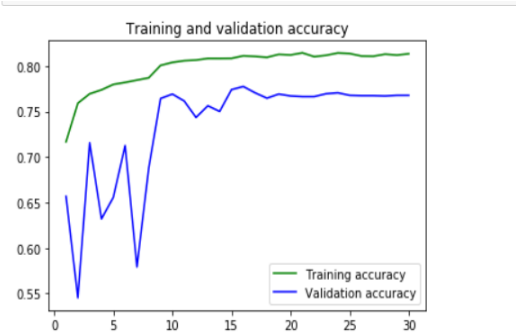


Figure 2.18: Training and Validation Accuracy

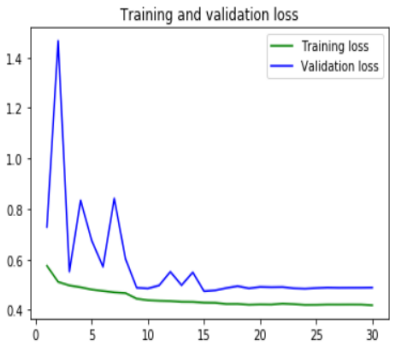


Figure 2.19: Training and Validation Loss

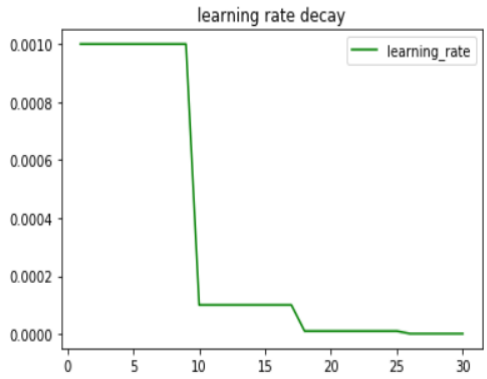


Figure 2.20: Learning Rate Decay

Accuracy of the Model: 0.77 **F1-Score of the Model: 0.76**

Chapter 3 Skin Cancer Classification From Dermatoscopic Images Using Deep Learning

A skin lesion is a part of the skin that has an abnormal growth or appearance compared to the skin around it. Two categories of skin lesions exist: primary and secondary. Primary skin lesions are abnormal skin conditions present at birth or acquired over a person's lifetime. Secondary skin lesions are the result of irritated or manipulated primary skin lesions. For example, if someone scratches a mole until it bleeds, the resulting lesion, a crust, is now a secondary skin lesion.

3.1. Overview

Dermatologic diseases are the fourth most common cause of all human illnesses, with approximately one-third of the world's population being affected by at least one skin condition. There are more than 3000 known skin diseases that vary in severity and symptoms, from self-limiting infections and benign tumours, to chronic inflammatory diseases and malignant neoplasms that cause significant morbidity and severely affect the quality of life. Skin conditions can affect any individual at any age and are somewhat more frequent in men and the elderly. It is estimated that every individual experiences a skin disease at least once during their lifetime.

3.1.1 What is Dermatology?

Dermatology involves the study, research, diagnosis, and management of any health conditions that may affect the skin, fat hair, nails, and membranes. A dermatologist is the health professional who specializes in this area of healthcare.

The skin is the largest organ of the body, which acts as a barrier to protect the internal organs from injury and bacteria. It is also a good indicator of the overall health of the body, making the field of dermatology important in the diagnosis and management of many health conditions

3.1.2. How does a dermatologist detect skin diseases?

Medical imaging can show itself as a high value resource, as dermatology has an extensive list of illness that it has to treat. In addition, the field has developed its own vocabulary to describe these lesions. However, verbal descriptions have their limitations and a good picture can replace successfully many sentences of description and is not susceptible to the bias of the message carrier. 62 This way, the analysis from medical images is similar to the analysis with the naked eye and so the same techniques and implications can be applied. Thus skin cancer can often be detected through naked eye and medical photography.

3.1.3 What are skin lesions?

A skin lesion is a part of the skin that has an abnormal growth or appearance compared to the skin around it.

Two categories of skin lesions exist: primary and secondary. Primary skin lesions are abnormal skin conditions present at birth or acquired over a person’s lifetime.

Secondary skin lesions are the result of irritated or manipulated primary skin lesions. For example, if someone scratches a mole until it bleeds, the resulting lesion, a crust, is now a secondary skin lesion.

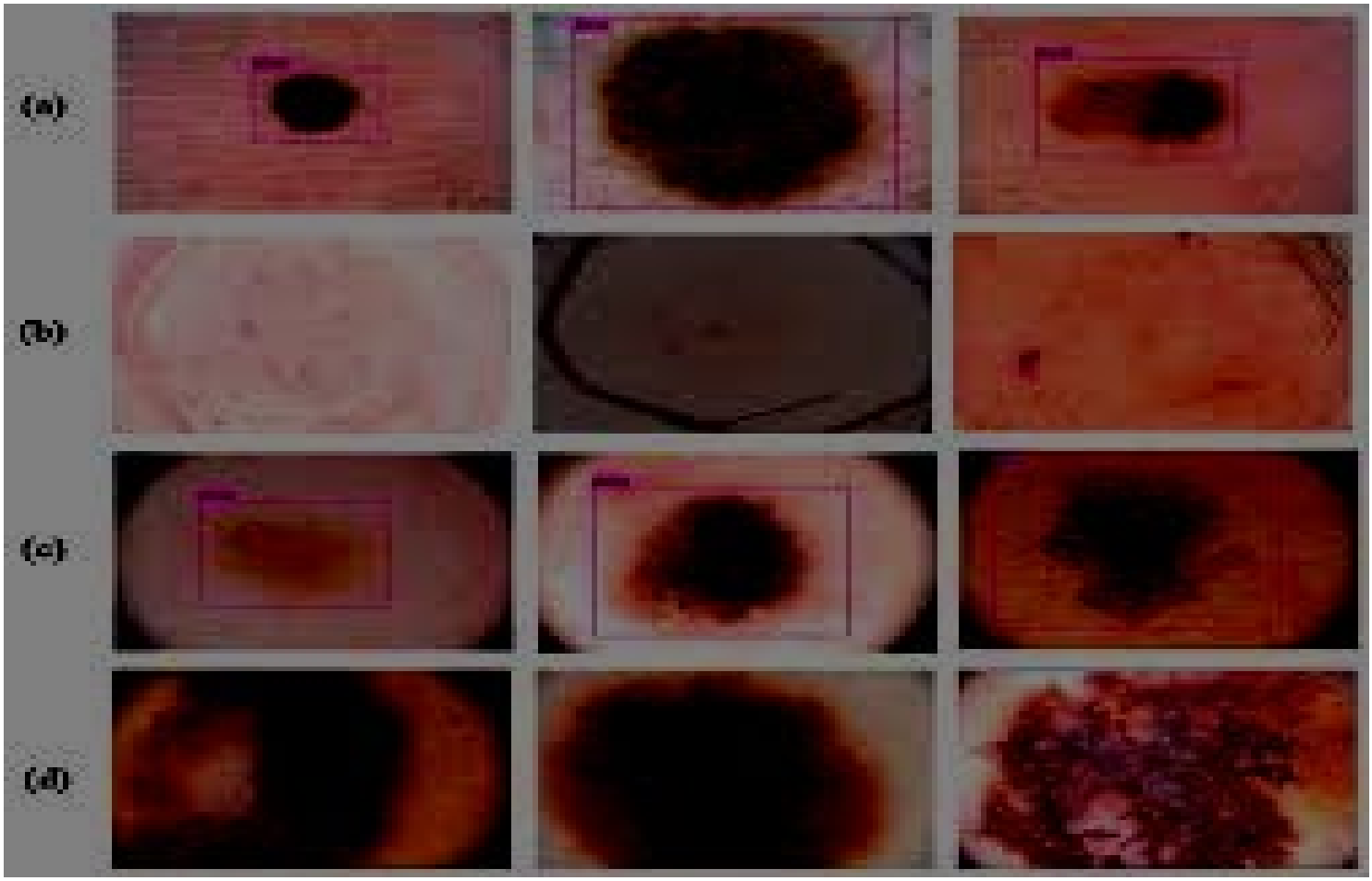


Figure 3.1 Dataset

3.1.4. What causes Skin Lesions?

- **Acne**

Figure 3.2 Acne

- Commonly located on the face, neck, shoulders, chest, and upper back
- Breakouts on the skin composed of blackheads, whiteheads, pimples, or deep, painful cysts and nodules
- May leave scars or darken the skin if untreated

- **Herpes simplex**

Figure 3.3 Herpes Simplex

- The viruses HSV-1 and HSV-2 cause oral and genital lesions
- These painful blisters occur alone or in clusters and weep clear yellow fluid and then crust over
- Signs also include mild flu-like symptoms such as fever, fatigue, swollen lymph nodes, headache, body aches, and decreased appetite

- Blisters may reoccur in response to stress, mensturation, illness, or sun exposure

- **Actinic keratosis**

Figure 3.4 Actinic keratosis

- Typically less than 2 cm, or about the size of a pencil eraser
- Thick, scaly, or crusty skin patch
- Appears on parts of the body that receive a lot of sun exposure (hands, arms, face, scalp, and neck)
- Usually pink in color but can have a brown, tan, or gray base

- **Allergic eczema**

Figure 3.5 Allergic Eczema

- May resemble a burn
- Often found on hands and forearms
- Skin is itchy, red, scaly, or raw
- Blisters that weep, ooze, or become crusty

- **Impetigo**

Figure 3.6 Impetigo

- Common in babies and children
- Rash is often located in the area around the mouth, chin, and nose

- Irritating rash and fluid-filled blisters that pop easily and form a honey-colored crust

- **Contact Dermatitis**

Figure 3.7 Contact Dermatitis

- Appears hours to days after contact with an allergen
- Rash has visible borders and appears where your skin touched the irritating substance
- Skin is itchy, red, scaly, or raw
- Blisters that weep, ooze, or become crusty

- **Psoriasis**

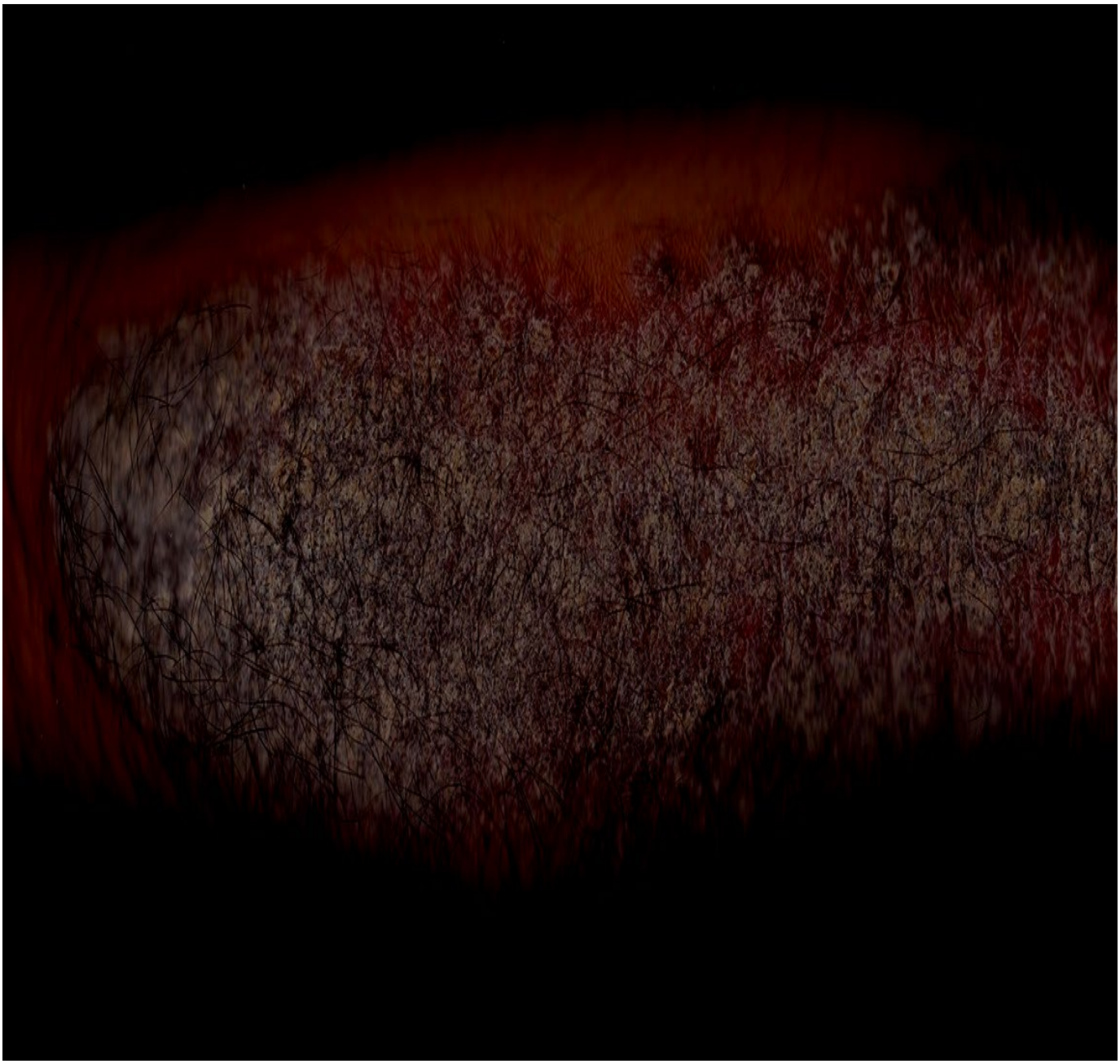


Figure 3.8 Psoriasis

- Scaly, silvery, sharply defined skin patches
- Commonly located on the scalp, elbows, knees, and lower back
- May be itchy or asymptomatic

- **Shingles**

Figure 3.9 Shingles

- Very painful rash that may burn, tingle, or itch, even if there are no blisters present
- Rash comprising clusters of fluid-filled blisters that break easily and weep fluid
- Rash emerges in a linear stripe pattern that appears most commonly on the torso, but may occur on other parts of the body, including the face
- Rash may be accompanied by low fever, chills, headache, or fatigue

- **Cellulitis**

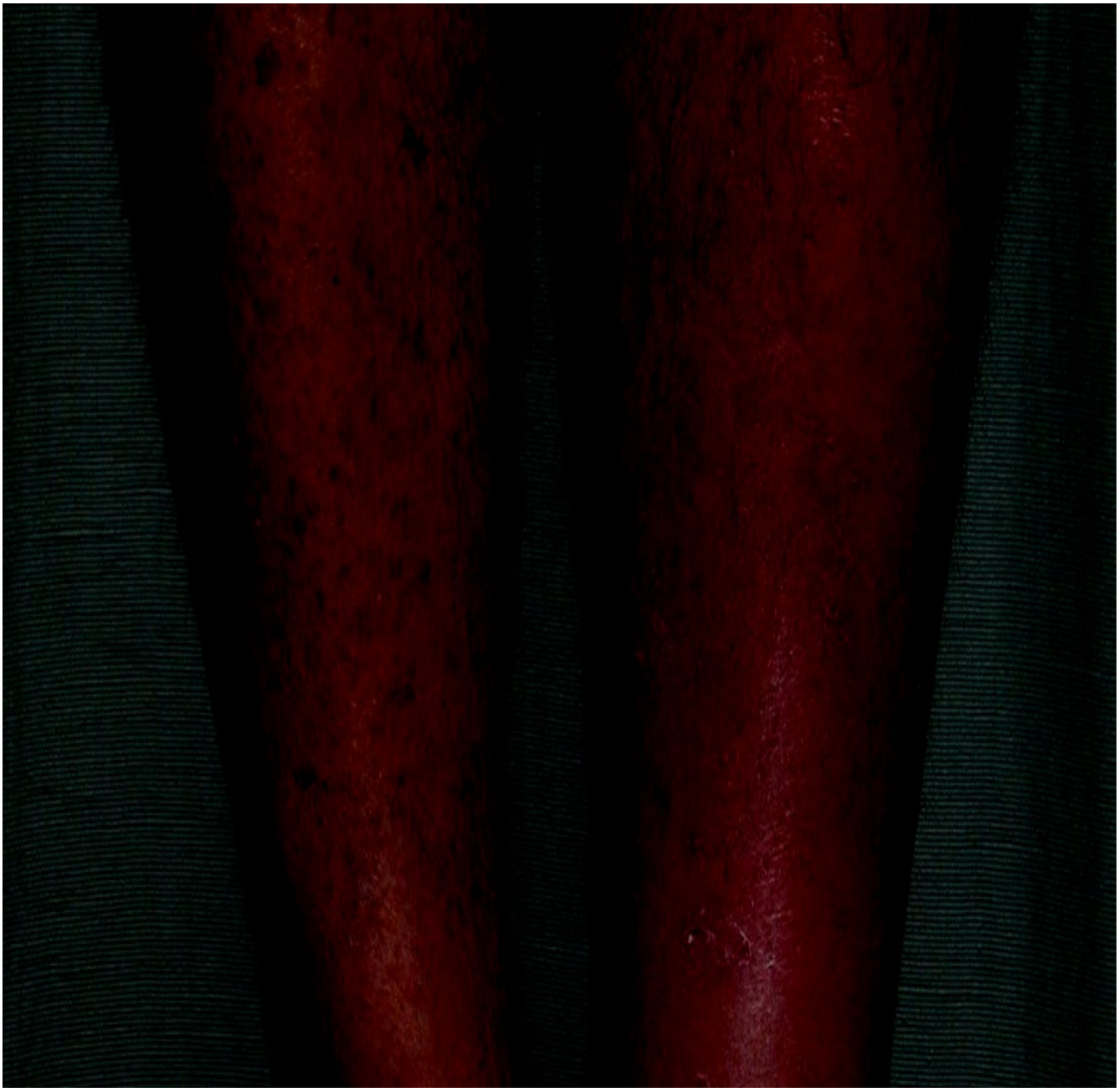


Figure 3.10 Cellulitis

This condition is considered a medical emergency. Urgent care may be required.

- Caused by bacteria or fungi entering through a crack or cut in the skin
- Red, painful, swollen skin with or without oozing that spreads quickly

- Hot and tender to the touch
- Fever, chills, and red streaking from the rash might be a sign of serious infection requiring medical attention

- **Scabies**

Figure 3.11 Scabies

- Symptoms may take four to six weeks to appear
- Extremely itchy rash may be pimply, made up of tiny blisters, or scaly
- Raised, white or flesh-toned lines

- **Boils**

Figure 3.12 Boils

- Bacterial or fungal infection of a hair follicle or oil gland
- Can appear anywhere on the body, but are most common on the face, neck, armpit, and buttock
- Red, painful, raised bump with a yellow or white center
- May rupture and weep fluid

- **Bullae**

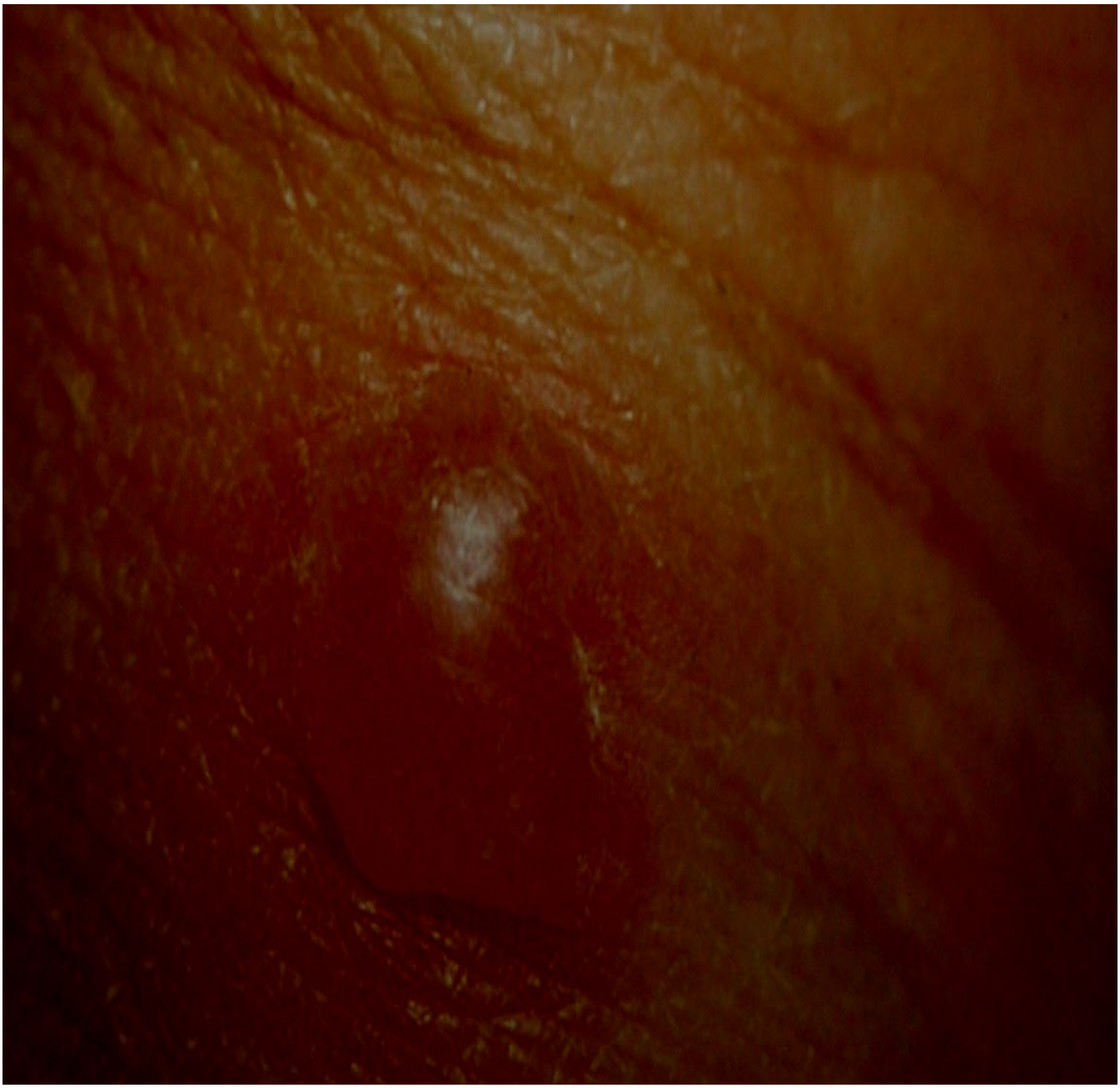


Figure 3.13 Bullae

- Clear, watery, fluid-filled blister that is greater than 1 cm in size
- Can be caused by friction, contact dermatitis, and other skin disorders
- If clear liquid turns milky, there might be an infection

- **Blister**

Figure 3.14 Blister

- Characterized by watery, clear, fluid-filled area on the skin
- May be smaller than 1 cm (vesicle) or larger than 1 cm (bulla) and occur alone or in groups
- Can be found anywhere on the body

- **Rash**

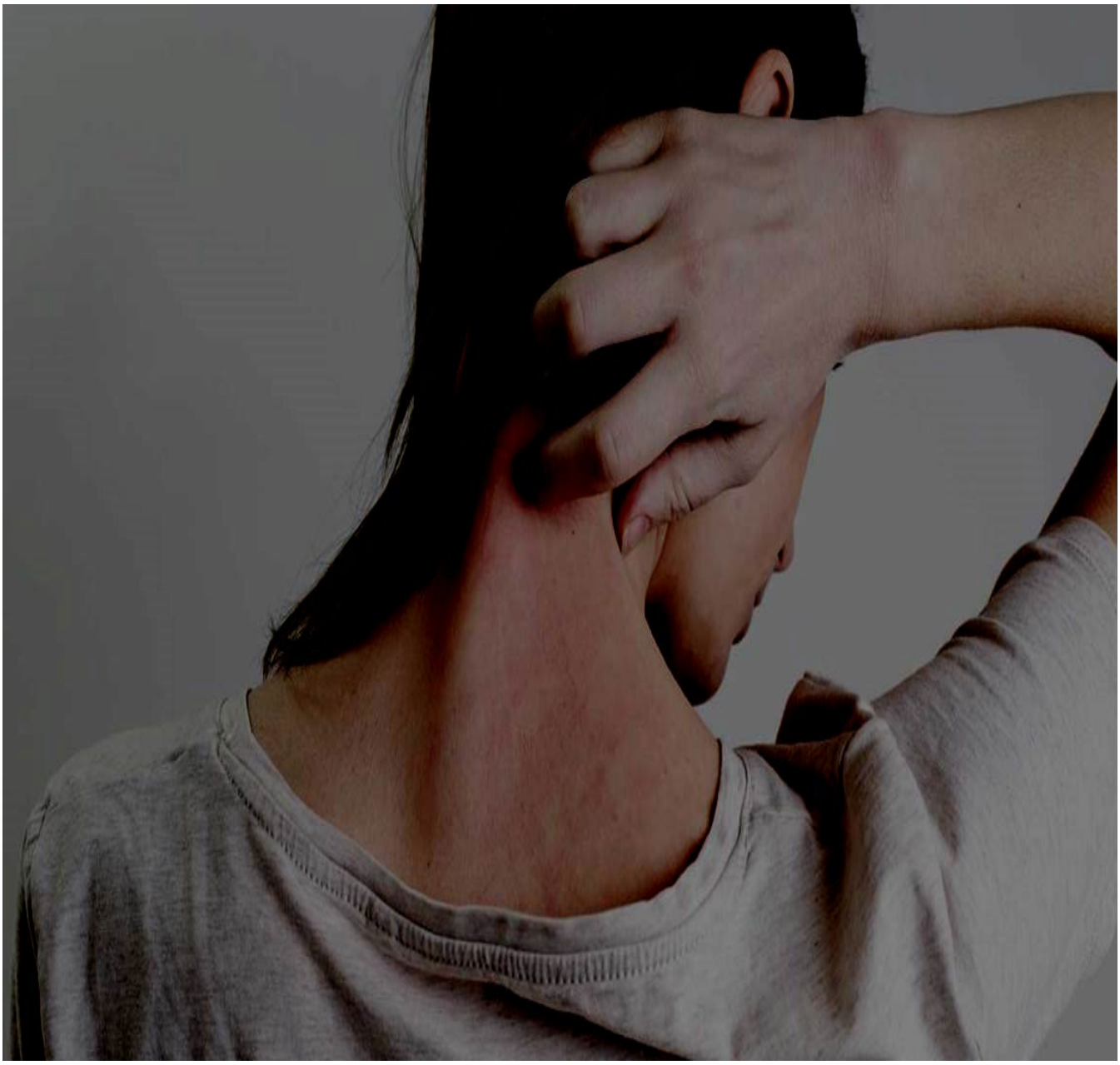


Figure 3.15 Rash

This condition is considered a medical emergency. Urgent care may be required.

- Defined as a noticeable change in the color or texture of the skin
- May be caused by many things, including insect bites, allergic reactions, medication side effects, fungal skin infection, bacterial skin infection, infectious disease, or autoimmune disease
- Many rash symptoms can be managed at home, but severe rashes, especially those seen in combination with other symptoms such as fever, pain, dizziness, vomiting, or difficulty breathing, may require urgent medical treatment

- **Hives**

Figure 3.16 Hives

- Itchy, raised welts that occur after exposure to an allergen
- Red, warm, and mildly painful to the touch
- Can be small, round, and ring-shaped or large and randomly shaped

- **Wart**

Figure 3.17 Wart

- Caused by many different types of a virus called human papillomavirus (HPV)
- May be found on the skin or mucous membranes
- May occur singly or in groups

- Contagious and may be passed to others

3.1.5 Skin Cancer

3.1.5.1 History of Skin Cancer

The first descriptions of cancer are documented in Egyptian papyri dating 2500 BC. Hippocrates described nonulcer and ulcer forming tumors, which he named carcinos (Greek for crab) as these tumors had finger-like projections resembling a crab. Celsus later used the Latin term for crab, cancer, to refer to tumors. The term oncology comes from oncos (Greek for swelling), which Galen used to describe tumors. In the 19th century, scientific oncology proliferated with the introduction of the modern microscope. Laennac made the first description of melanoma in 1804, Jacob of basal cell carcinoma in 1827, and Bowen of squamous cell carcinoma in situ in 1912. Virchow, the founder of cellular pathology, provided the basics for modern cancer pathology as he demonstrated that cancer cells derived from other cells. He also hypothesized that chronic radiation caused cancer. A group of experiments performed by dermatologists, physicists and other medical scientists between the 19th and 20th century demonstrated that skin cancers were induced by UV radiation and that skin pigmentation had a protective effect from radiation. They also demonstrated that certain chemicals could potentiate skin damage and skin cancer formation, while other chemicals could block radiation and protect the skin. It was in 1928 that the first sunscreen became available. Although it was known that UV radiation could cause skin cancer, it was not until the 1940s that its damage to nucleic acids became evident. Furthermore, in the 1970s oncogenes and tumor suppressor genes were discovered as important gene families related to cancer development. Since the 16th century, cancers were treated with surgery as long as the tumor was localized. Mohs revolutionized skin cancer surgery in the 1930s, which method remains vital to the treatment of skin cancers today. Medical noninvasive treatment options became possible due to innovations in molecular biology in the 1960s. Photodynamic therapy and lasers were also added to the armamentarium for the treatment of actinic keratosis and cutaneous cancers. In 2012, vismodegib was approved for the treatment of metastasized or advanced basal cell carcinomas. Thanks to these discoveries, advancements in technology and the rapid evolution of molecular biology, today we have a greater comprehension of

the pathophysiology of skin cancers and are able to prevent, appropriately diagnose and treat cutaneous malignancies.

3.1.5.2 Skin Cancer Types

Basal cell carcinoma (BCC)
This is the most common type of skin cancer.

- BCC frequently develops in people who have fair skin. People who have skin of color also get this skin cancer.
- BCCs often look like a flesh-colored round growth, pearl-like bump, or a pinkish patch of skin.
- BCCs usually develop after years of frequent sun exposure or indoor tanning.
- BCCs are common on the head, neck, and arms; however, they can form anywhere on the body, including the chest, abdomen, and legs.
- Early diagnosis and treatment for BCC are important. BCC can grow deep. Allowed to grow, it can penetrate the nerves and bones, causing damage and disfigurement

Figure 3.15 Basal cell carcinoma

Squamous cell carcinoma (SCC) of the skin

SCC is the second most common type of skin cancer.

- People who have light skin are most likely to develop SCC. This skin cancer also develops in people who have darker skin.
- SCC often looks like a red firm bump, scaly patch, or a sore that heals and then re-opens.
- SCC tends to form on skin that gets frequent sun exposure, such as the rim of the ear, face, neck, arms, chest, and back.
- SCC can grow deep into the skin, causing damage and disfigurement.
- Early diagnosis and treatment can prevent SCC from growing deep and spreading to other areas of the body.

Figure 3.16 Squamous cell carcinoma (SCC) of the skin

Melanoma

Melanoma is often called "the most serious skin cancer" because it has a tendency to spread.

- Melanoma can develop within a mole that you already have on your skin or appear suddenly as a dark spot on the skin that looks different from the rest.
- Early diagnosis and treatment are crucial.
- Knowing the ABCDE warning signs of melanoma can help you find an early melanoma.

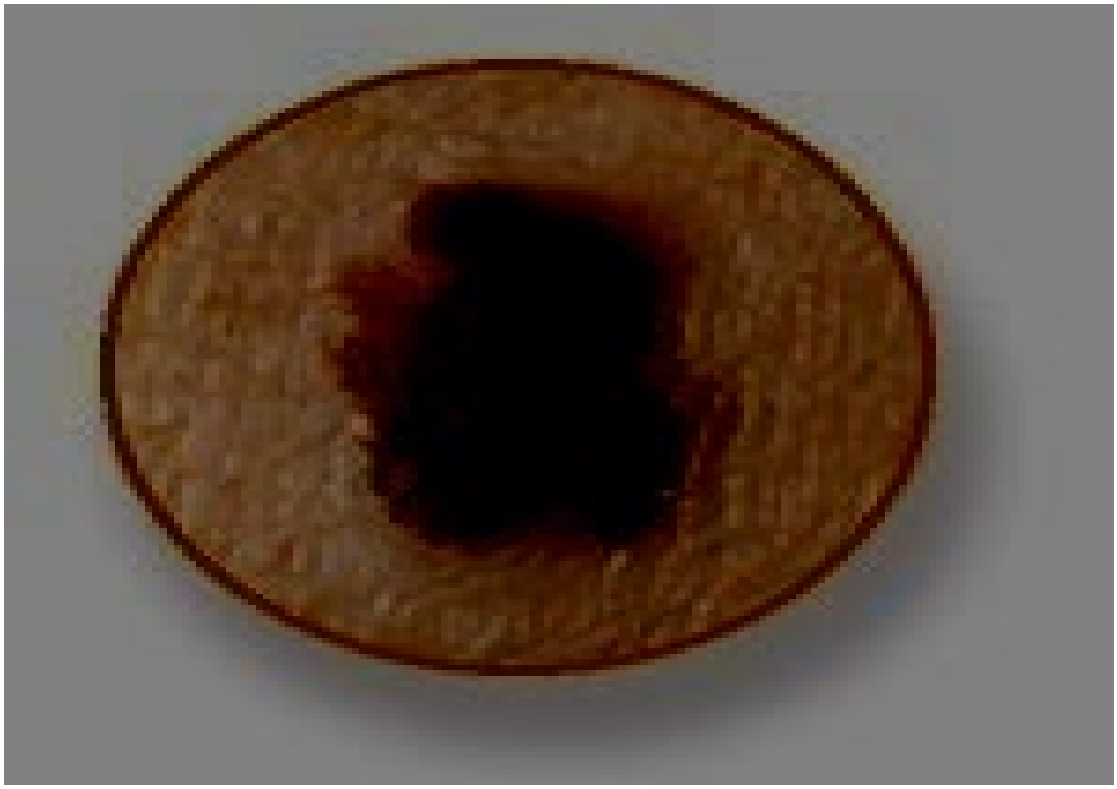


Figure 3.17 Melanoma

3.1.5.3 Skin Cancer Causes

Figure 3.18 Skin Cancer Causes

Skin cancer occurs when errors (mutations) occur in the DNA of skin cells. The mutations cause the cells to grow out of control and form a mass of cancer cells.

Cells involved in skin cancer:

Skin cancer begins in your skin's top layer — the epidermis. The epidermis is a thin layer that provides a protective cover of skin cells that your body continually sheds. The epidermis contains three main types of cells:

- Squamous cells lie just below the outer surface and function as the skin's inner lining.
- Basal cells, which produce new skin cells, sit beneath the squamous cells.
- Melanocytes — which produce melanin, the pigment that gives skin its normal color — are located in the lower part of your epidermis. Melanocytes produce more melanin when you're in the sun to help protect the deeper layers of your skin.

3.2. Problem Statement

3.2.1 Explanation

The most common human malignancy, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions.

3.3. Dataset

3.3.1 Introduction

This the **HAM10000 ("Human Against Machine with 10000 training images")** dataset. It consists of 10015 dermatoscopic images which are released as a training set for academic machine learning purposes and are publicly available through the ISIC archive. This benchmark dataset can be used for machine learning and for comparisons with human experts.

3.3.2 Lesions Available in the Dataset

It has 7 different classes of skin cancer which are listed below:

1. Melanocytic nevi
2. Melanoma
3. Benign keratosis-like lesions
4. Basal cell carcinoma
5. Actinic keratoses
6. Vascular lesions
7. Dermatofibroma

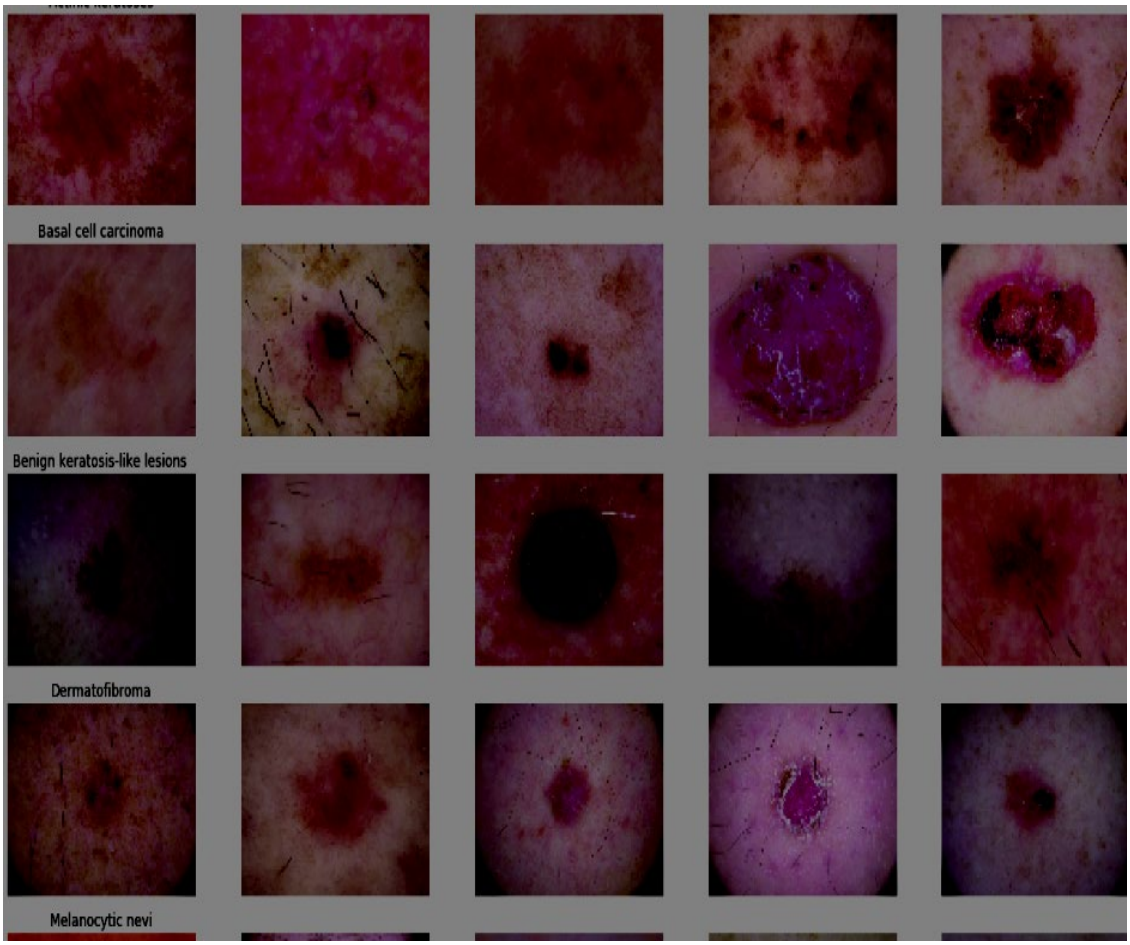


Figure 3.19 Dataset Classes

3.3.3. System Architecture and Preprocessing

3.3.4 Importing Essential Libraries

This is a collection of the most important Python libraries for Machine Learning.

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
import seaborn as sns
from PIL import Image
np.random.seed(123)
from sklearn.preprocessing import label_binarize
from sklearn.metrics import confusion_matrix
import itertools

import keras
from keras.utils.np_utils import to_categorical # used for converting labels to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras import backend as K
import itertools
from keras.layers.normalization import BatchNormalization
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import pydicom
import os
import matplotlib.pyplot as plt
import collections
from tqdm import tqdm_notebook as tqdm
from datetime import datetime
```

Figure 3.20 Essential Libraries

3.3.5 Making Dictionary of images and labels

In this step I have made the image path dictionary by joining the folder path from base directory base_skin_dir and merge the images in jpg format from both the folders HAM10000_images_part1.zip and HAM10000_images_part2.zip

```
base_skin_dir = os.path.join('..', 'input')

# Merging images from both folders HAM10000_images_part1.zip and HAM10000_images_part2.zip into one dictionary

imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x
                     for x in glob(os.path.join(base_skin_dir, '*', '*.jpg'))}

# This dictionary is useful for displaying more human-friendly labels later on

lesion_type_dict = {
    'nv': 'Melanocytic nevi',
    'mel': 'Melanoma',
    'bkl': 'Benign keratosis-like lesions ',
    'bcc': 'Basal cell carcinoma',
    'akiec': 'Actinic keratoses',
    'vasc': 'Vascular lesions',
    'df': 'Dermatofibroma'
}
```

Figure 3.21 Dictionary of images

3.3.6 Reading and Processing Data

In this step we have read the csv by joining the path of image folder which is the base folder where all the images are placed named base_skin_dir. After that we made some new columns which is easily understood for later reference such as we have made column path which contains the image_id, cell_type which contains the short name of lesion type and at last we have made the categorical column cell_type_idx in which we have categorize the lesion type in to codes from 0 to 6.

```
skin_df = pd.read_csv(os.path.join(base_skin_dir, 'HAM10000_metadata.csv'))

# Creating New Columns for better readability

skin_df['path'] = skin_df['image_id'].map(imageid_path_dict.get)
skin_df['cell_type'] = skin_df['dx'].map(lesion_type_dict.get)
skin_df['cell_type_idx'] = pd.Categorical(skin_df['cell_type']).codes
```

```
# Now lets see the sample of tile_df to look on newly made columns
skin_df.head(50)
```

	lesion_id	image_id	dx	dx_type	age	sex	localization	path	cell_type	cell_type_idx
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0027419.jpg	Benign keratosis-like lesions	2
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0025030.jpg	Benign keratosis-like lesions	2
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0026769.jpg	Benign keratosis-like lesions	2
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0025661.jpg	Benign keratosis-like lesions	2
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	../input/ham10000_images_part_2/ISIC_0031633.jpg	Benign keratosis-like lesions	2
5	HAM_0001466	ISIC_0027850	bkl	histo	75.0	male	ear	../input/HAM10000_images_part_1/ISIC_0027850.jpg	Benign keratosis-like lesions	2
6	HAM_0002761	ISIC_0029176	bkl	histo	60.0	male	face	../input/HAM10000_images_part_1/ISIC_0029176.jpg	Benign keratosis-like lesions	2

Figure 3.22 Processing Data

3.3.7 Data Cleaning

In this step we check for Missing values and datatype of each field

Figure 3.23 Data Cleaning

As it is evident from the above that only age has null values which is 57 so we will fill the null values by their mean.

3.4 Exploratory data analysis (EDA)

In this we will explore different features of the dataset, their distributions and actual counts

Plot to see distribution of 7 different classes of cell type

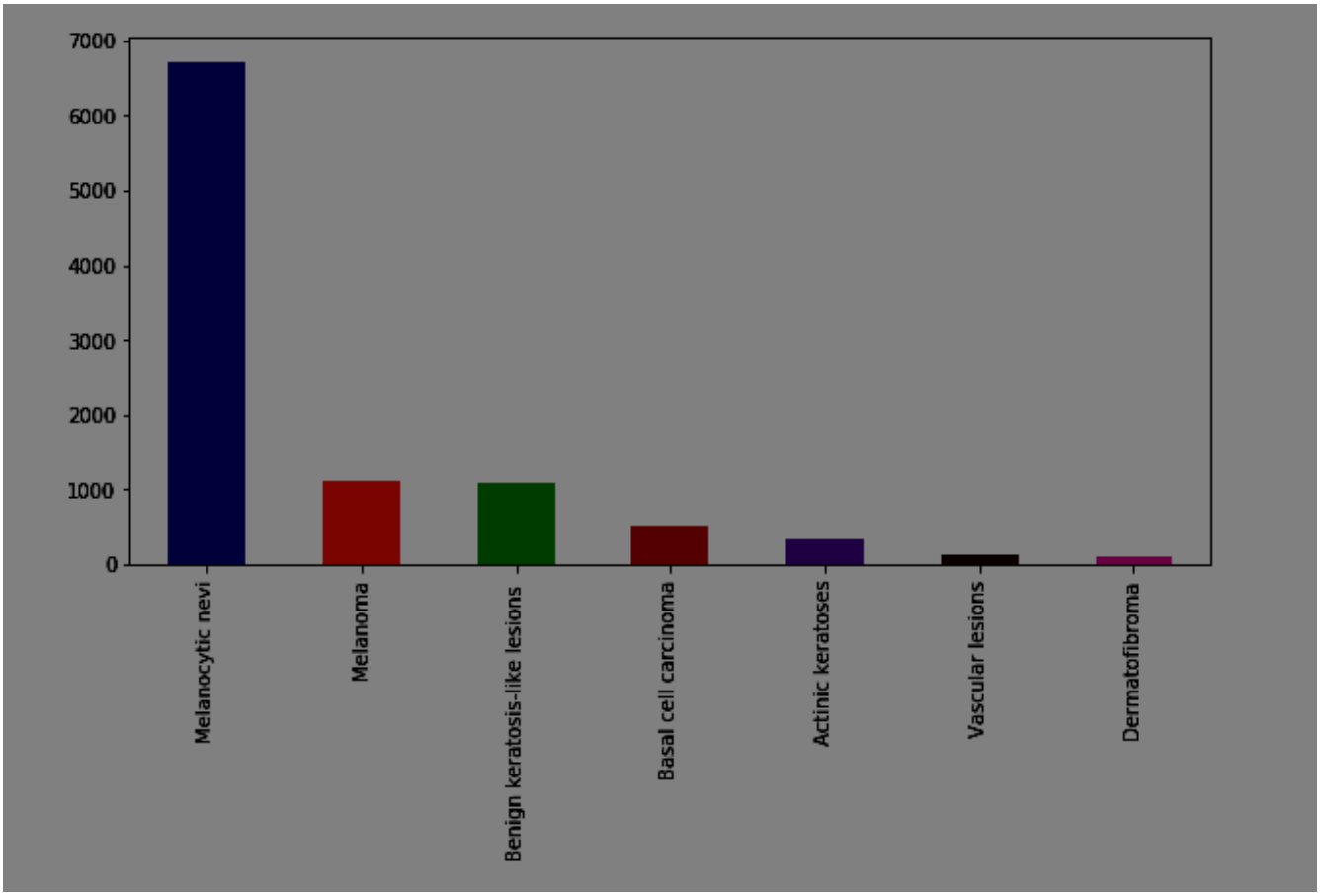


Figure 3.24 Distribution of 7 different classes

Its seems from the above plot that in this dataset cell type Melanecytic nevi has very large number of instances in comparison to other cell types

Plotting of Technical Validation field (ground truth) which is dx_type to see the distribution of its 4 categories which are listed below:
1. Histopathology (Histo): Histopathologic diagnoses of excised lesions have been performed by specialized dermatopathologists.
2. Confocal: Reflectance confocal microscopy is an in-vivo imaging technique with a resolution at near-cellular level , and some facial benign with a grey-world assumption of all training-set images in Lab-color space before and after manual histogram changes.
3. Follow-up**:** If nevi monitored by digital dermatoscopy did not show any changes during 3 follow-up visits or 1.5 years biologists accepted this as evidence of biologic benignity. Only nevi, but no other benign diagnoses were labeled with this type of ground-truth because dermatologists usually do not

monitor dermatofibromas, seborrheic keratoses, or-vascular-lesions.
4. Consensus**:** For typical benign cases without histopathology or followup biologists provide an expert-consensus rating of authors PT and HK. They applied the consensus label only if both authors independently gave the same unequivocal benign diagnosis. Lesions with this type of groundtruth were usually photographed for educational reasons and did not need further follow-up or biopsy for confirmation.

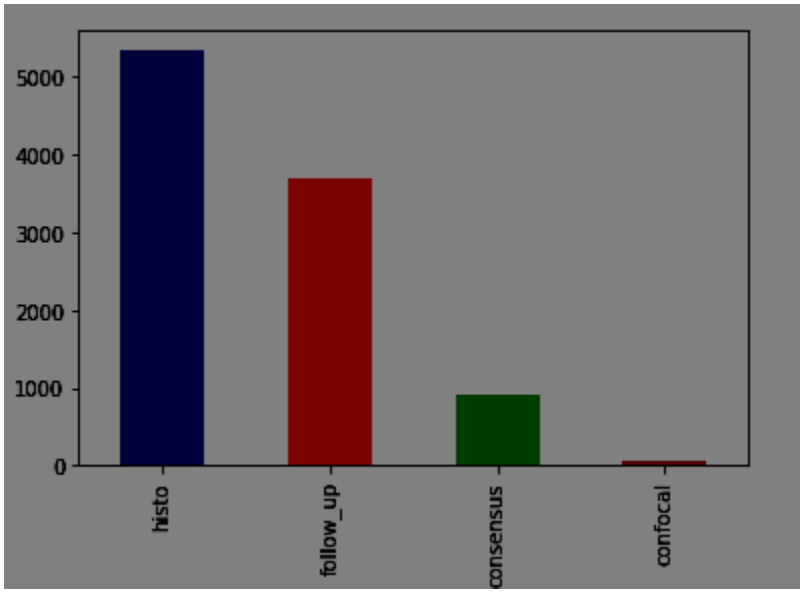


Figure 3.25 Distribution of 7 different classes

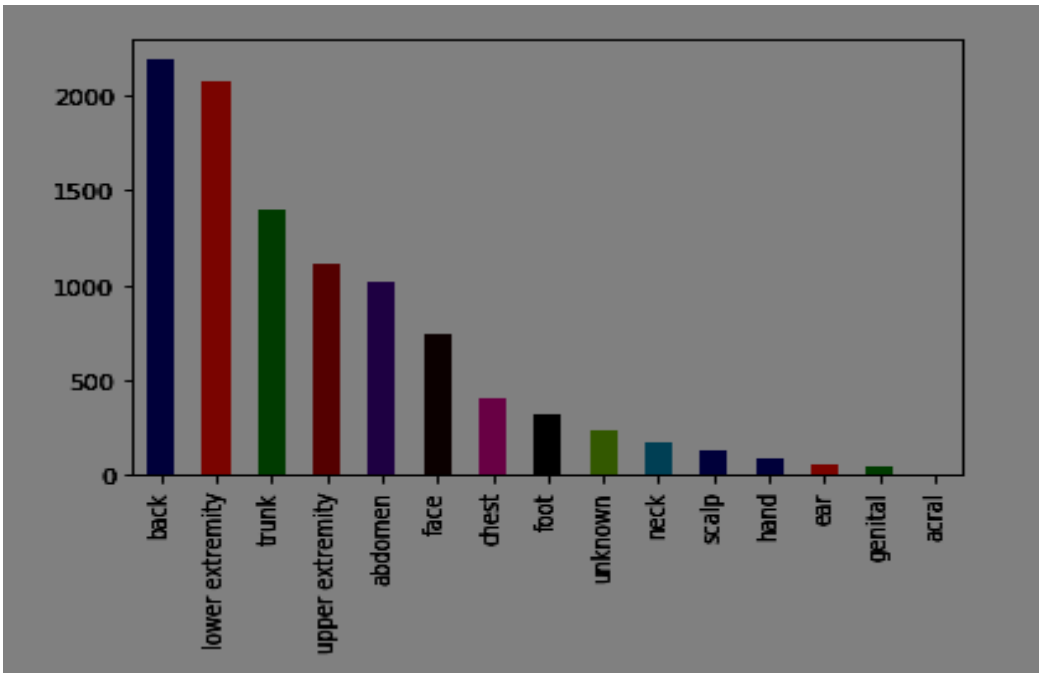


Figure 3.26 Distribution of 7 different classes

It seems back , lower extremity, trunk and upper extremity are heavily compromised regions of skin cancer.

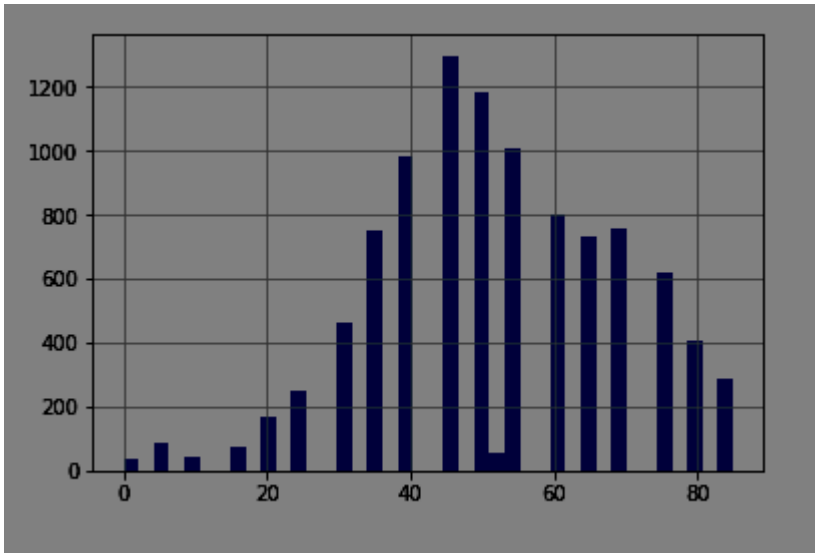


Figure 3.27 Distribution of 7 different classes ages

It seems that there are larger instances of patients having age from 30 to 60

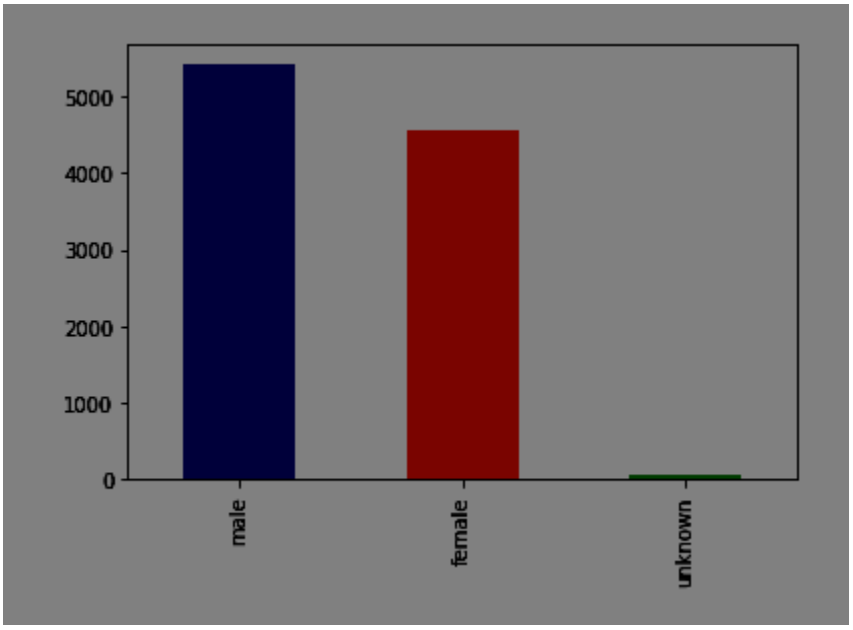


Figure 3.28 Distribution of 7 different classes gender

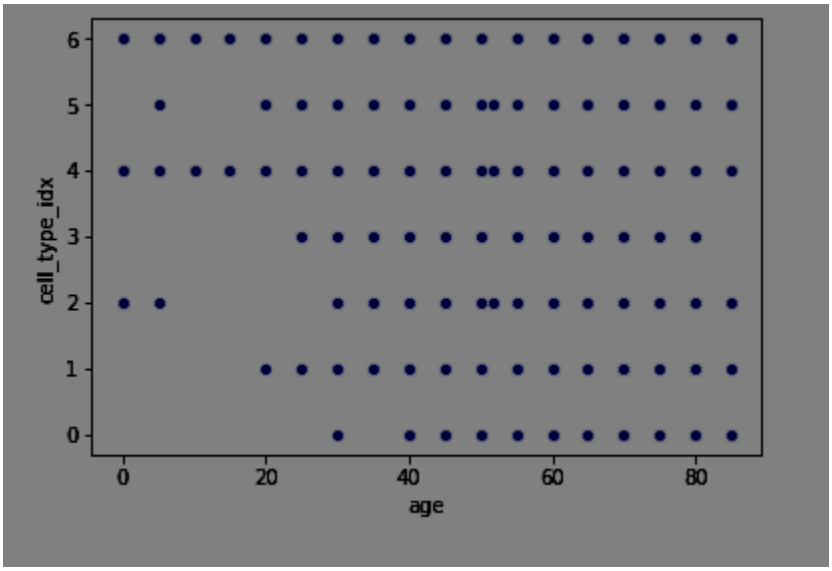


Figure 3.29 Distribution of 7 different classes

It seems that skin cancer types 0, 1, 3 and 5 which are Melanocytic nevi, dermatofibroma, Basal cell carcinoma and vascular lesions are not much prevalent below the age of 20 years

3.4.1 Loading & Resizing of images

In this step images will be loaded into the column named image from the image path from the image folder. We also resize the images as the original dimension of images are 450 x 600 x3 which TensorFlow can't handle, so that's why we resize it into 100 x 75. As this step resize all the 10015 images dimensions into 100x 75 so be patient it will take some time.

	lesion_id	image_id	dx	dx_type	age	sex	localization	path	cell_type	cell_type_idx	image
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0027419.jpg	Benign keratosis-like lesions	2	[[[191, 153, 194], [191, 154, 198], [189, 152,...
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0025030.jpg	Benign keratosis-like lesions	2	[[[23, 13, 22], [25, 15, 24], [24, 16, 31], [3...
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0026769.jpg	Benign keratosis-like lesions	2	[[[186, 126, 136], [190, 136, 149], [195, 137,...
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	../input/HAM10000_images_part_1/ISIC_0025661.jpg	Benign keratosis-like lesions	2	[[[24, 11, 18], [27, 13, 26], [39, 23, 33], [6...
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	../input/ham10000_images_part_2/ISIC_0031633.jpg	Benign keratosis-like lesions	2	[[[136, 91, 114], [148, 104, 127], [161, 117, ...

Figure 3.30 Resizing & loading

As we can see image column has been added in its color format code

Most interesting part it's always better to see sample of images below we will show images of each cancer type

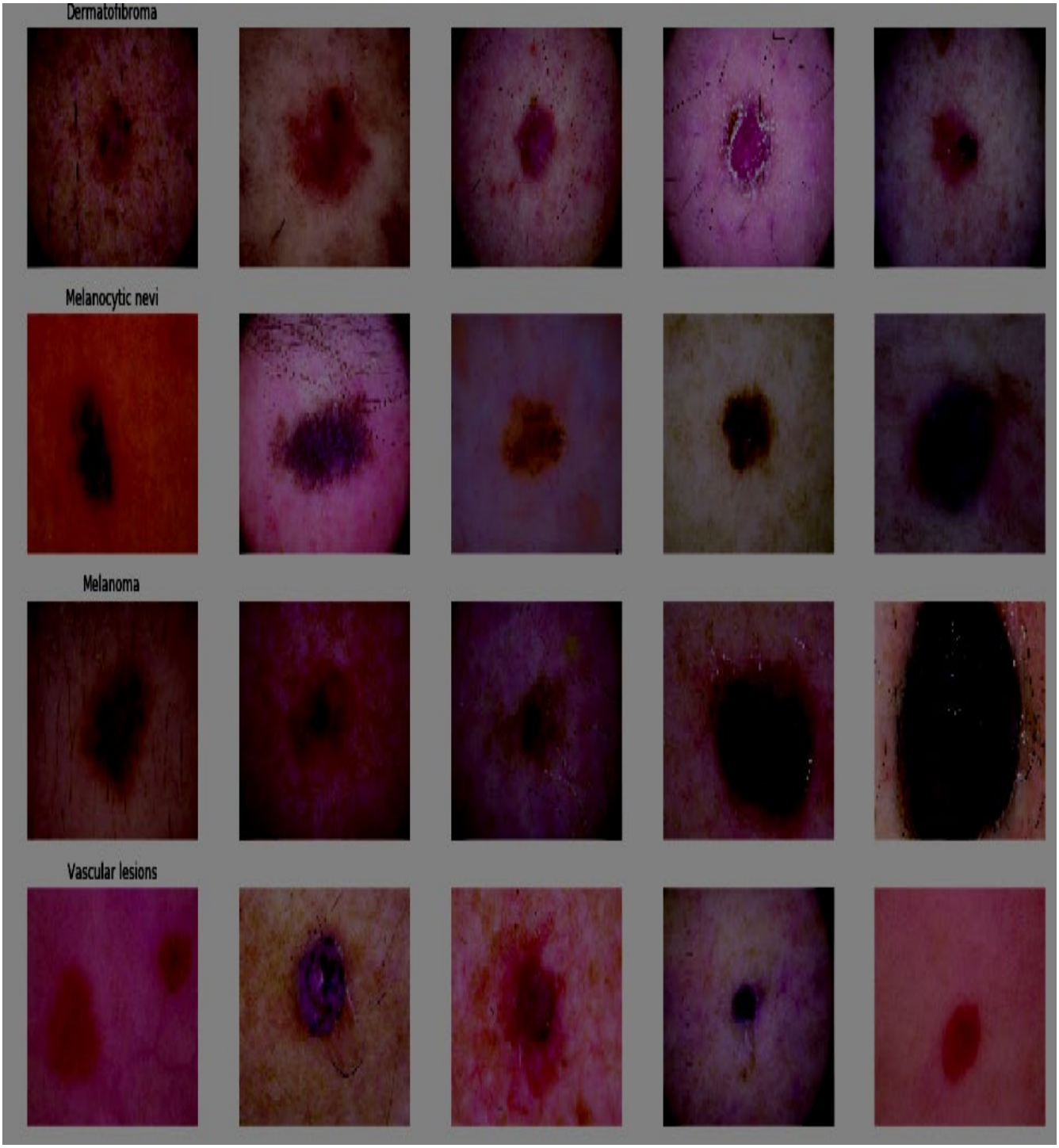


Figure 3.31 Resizing & loading

3.4.2 Train Test Split

In this step we have spited the dataset into training and testing set of 80:20 ratio

3.4.3 Normalization

I choose to normalize the x_train, x_test by substracting from theor mean values and then dividing by their standard deviation.

3.4.4 Label Encoding

Labels are 7 different classes of skin cancer types from 0 to 6. We need to encode these labels to one hot vectors

3.4.5 Train validation split

I choose to split the train set in two parts : a small fraction (10%) became the validation set which the model is evaluated and the rest (90%) is used to train the model.

```
x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train, test_size = 0.1, random_state = 2)



x_train = x_train.reshape(x_train.shape[0], *(75, 100, 3))
x_test = x_test.reshape(x_test.shape[0], *(75, 100, 3))
x_validate = x_validate.reshape(x_validate.shape[0], *(75, 100, 3))
```

Figure 3.32 Train Validation Split

3.5 Model Building

3.5.1 What is CNN?

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics**.** The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Input Image

In the figure, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). . This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

Convolution Layer — The Kernel

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our **5x5x1 input image, I**. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow. We have selected **K as a 3x3x1 matrix.**

Kernel/Filter, K = 1 0 1

0 1 0

1 0 1

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.

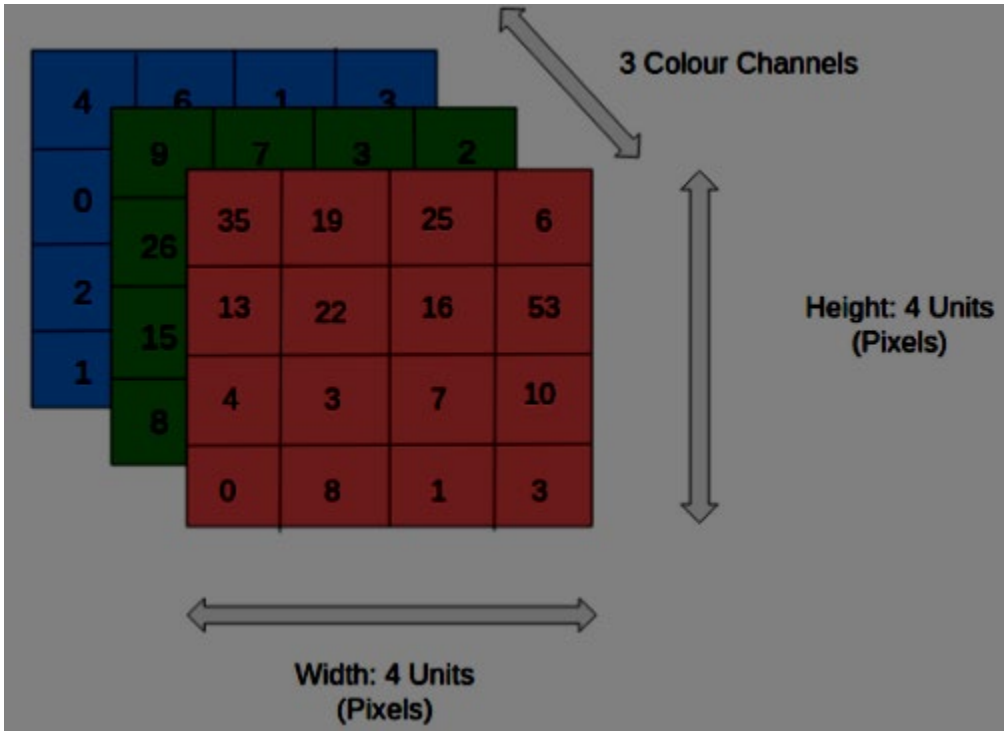


Figure 3.33 Filtering

3.5.2 Model Building (CNN)

I used the Keras Sequential API, where you have just to add one layer at a time, starting from the input.
The first is the convolutional (Conv2D) layer. It is like a set of learnable filters. I choosed to set 32 filters for the two firsts conv2D layers and 64 filters for the two last ones. Each filter transforms a part of the image (defined by the kernel size) using the kernel filter. The kernel filter matrix is applied on the whole image. Filters can be seen as a transformation of the image.

The CNN can isolate features that are useful everywhere from these transformed images (feature maps).
The second important layer in CNN is the pooling (MaxPool2D) layer. This layer simply acts as a downsampling filter. It looks at the 2 neighboring pixels and picks the maximal value. These are used to reduce computational cost, and to some extent also reduce overfitting. We have to choose the pooling size (i.e the area size pooled each time) more the pooling dimension is high, more the

downsampling is important.Combining convolutional and pooling layers, CNN are able to combine local features and learn more global features of the image.
Dropout is a regularization method, where a proportion of nodes in the layer are randomly ignored (setting their wieghts to zero) for each training sample. This drops randomly a propotion of the network and forces the network to learn features in a distributed way. This technique also improves generalization and reduces the overfitting.'relu' is the rectifier (activation function max(0,x). The rectifier activation function is used to add non linearity to the network.
The Flatten layer is use to convert the final feature maps into a one single 1D vector. This flattening step is needed so that you can make use of fully connected layers after some convolutional/maxpool layers. It combines all the found local features of the previous convolutional layers.

In the end i used the features in two fully-connected (Dense) layers which is just artificial an neural networks (ANN) classifier. In the last layer(Dense(10,activation="softmax")) the net outputs distribution of probability of each class.

```
# Set the CNN model
# my CNN architechture is In -> [[Conv2D->relu]*2 -> MaxPool2D -> Dropout]*2 -> Flatten -> Dense -> Dropout -> Out
input_shape = (75, 100, 3)
num_classes = 7

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding = 'Same',input_shape=input_shape))
model.add(Conv2D(32,kernel_size=(3, 3), activation='relu',padding = 'Same',))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.35))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

Figure 3.34 Model Architecture

3.5.3 Setting Optimizer & Annealing

Once our layers are added to the model, we need to set up a score function, a loss function and an optimisation algorithm. We define the loss function to measure how poorly our model performs on images with known labels. It is the error rate between the oberved labels and the predicted ones. We use a specific form for categorical classifications (>2 classes) called the "categorical_crossentropy". The most important function is the optimizer. This function will iteratively improve parameters (filters kernel values, weights and bias of neurons ...) in order to minimise the loss. I choosed Adam optimizer because it combines the advantages of two other extensions of stochastic gradient descent. Specifically:

1. Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).
2. Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

Adam realizes the benefits of both AdaGrad and RMSProp.
Adam is a popular algorithm in the field of deep learning because it achieves good results fast.
The metric function "accuracy" is used is to evaluate the performance our model. This metric function is similar to the loss function, except that the results from the metric evaluation are not used when training the model (only for evaluation).

```
# Define the optimizer
optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

```
# Compile the model
model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])
```

```
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

In order to make the optimizer converge faster and closest to the global minimum of the loss function, i used an annealing method of the learning rate (LR).

The LR is the step by which the optimizer walks through the 'loss landscape'. The higher LR, the bigger are the steps and the quicker is the convergence. However the sampling is very poor with an high LR and the optimizer could probably fall into a local minima.
Its better to have a decreasing learning rate during the training to reach efficiently the global minimum of the loss function.
To keep the advantage of the fast computation time with a high LR, i decreased the LR dynamically every X steps (epochs) depending if it is necessary (when accuracy is not improved).
With the ReduceLROnPlateau function from Keras.callbacks, i choose to reduce the LR by half if the accuracy is not improved after 3 epochs.

Data Augmentation

It is the optional step. In order to avoid overfitting problem, we need to expand artificially our HAM 10000 dataset. We can make your existing dataset even larger. The idea is to alter the training data with small transformations to reproduce the variations

Approaches that alter the training data in ways that change the array representation while keeping the label the same are known as data augmentation techniques. Some popular augmentations people use are grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more.
By applying just a couple of these transformations to our training data, we can easily double or triple the number of training examples and create a very robust model.

```
# With data augmentation to prevent overfitting

datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=False,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(x_train)
```

Figure 3.35 Data Augmentation

For the data augmentation, I choose to: Randomly rotate some training images by 10 degrees Randomly Zoom by 10% some training images randomly shift images horizontally by 10% of the width randomly shift images vertically by 10% of the height once our model is ready, we fit the training dataset

3.5.4 Fitting the model

In this step finally I fit the model into x_train, y_train. In this step I have chosen batch size of 10 and 50 epochs as small as your batch size will be more efficiently your model will train and I have chosen 50 epochs to give the model sufficient epochs to train.

```
# Fit the model
epochs = 50
batch_size = 10
history = model.fit_generator(datagen.flow(x_train,y_train, batch_size=batch_size),
                              epochs = epochs, validation_data = (x_validate,y_validate),
                              verbose = 1, steps_per_epoch=x_train.shape[0] // batch_size
                              ,callbacks=[learning_rate_reduction]
                              )
```

Figure 3.36 Fit the model

3.6 Model Evaluation

```
766/766 [==============================] - 18s 23ms/step - loss: 0.6493 - acc: 0.7604 - val_loss: 0.6410 - val_acc: 0.7570
Epoch 43/50
766/766 [==============================] - 18s 23ms/step - loss: 0.6407 - acc: 0.7654 - val_loss: 0.6419 - val_acc: 0.7582
Epoch 44/50
766/766 [==============================] - 18s 23ms/step - loss: 0.6460 - acc: 0.7560 - val_loss: 0.6415 - val_acc: 0.7594
Epoch 45/50
766/766 [==============================] - 17s 23ms/step - loss: 0.6459 - acc: 0.7557 - val_loss: 0.6424 - val_acc: 0.7594
Epoch 46/50
766/766 [==============================] - 19s 24ms/step - loss: 0.6411 - acc: 0.7627 - val_loss: 0.6425 - val_acc: 0.7559
Epoch 47/50
766/766 [==============================] - 17s 23ms/step - loss: 0.6421 - acc: 0.7637 - val_loss: 0.6419 - val_acc: 0.7570
Epoch 48/50
766/766 [==============================] - 17s 22ms/step - loss: 0.6526 - acc: 0.7584 - val_loss: 0.6412 - val_acc: 0.7570
Epoch 49/50
766/766 [==============================] - 18s 24ms/step - loss: 0.6371 - acc: 0.7659 - val_loss: 0.6396 - val_acc: 0.7559
Epoch 50/50
766/766 [==============================] - 17s 23ms/step - loss: 0.6450 - acc: 0.7632 - val_loss: 0.6406 - val_acc: 0.7617
```

Figure 3.37 Model Training

3.6.1 Testing and validation accuracy

In this step we will check the testing accuracy and validation accuracy of our model,plot confusion matrix and also check the missclassified images count of each type.

```
loss, accuracy = model.evaluate(x_test, y_test, verbose=1)
loss_v, accuracy_v = model.evaluate(x_validate, y_validate, verbose=1)
print("Validation: accuracy = %f  ;  loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f  ;  loss = %f" % (accuracy, loss))
model.save("model.h5")
```

```
1503/1503 [==============================] - 0s 294us/step
852/852 [==============================] - 0s 275us/step
Validation: accuracy = 0.761737  ;  loss_v = 0.640614
Test: accuracy = 0.779774  ;  loss = 0.612470
```

Figure 3.38 Testing Validation Accuracy

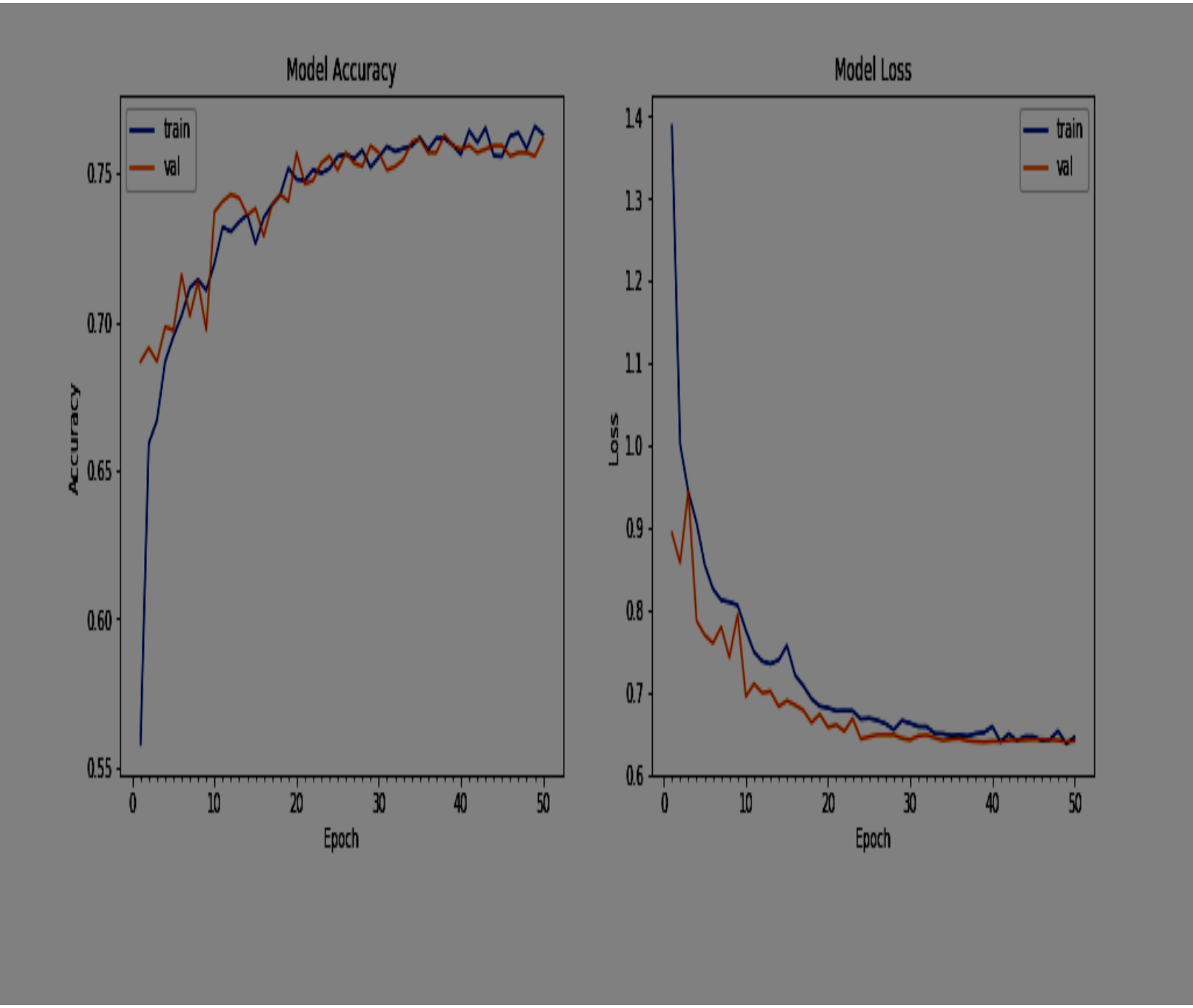


Figure 3.39 Model Accuracy

3.6.2 Confusion matrix

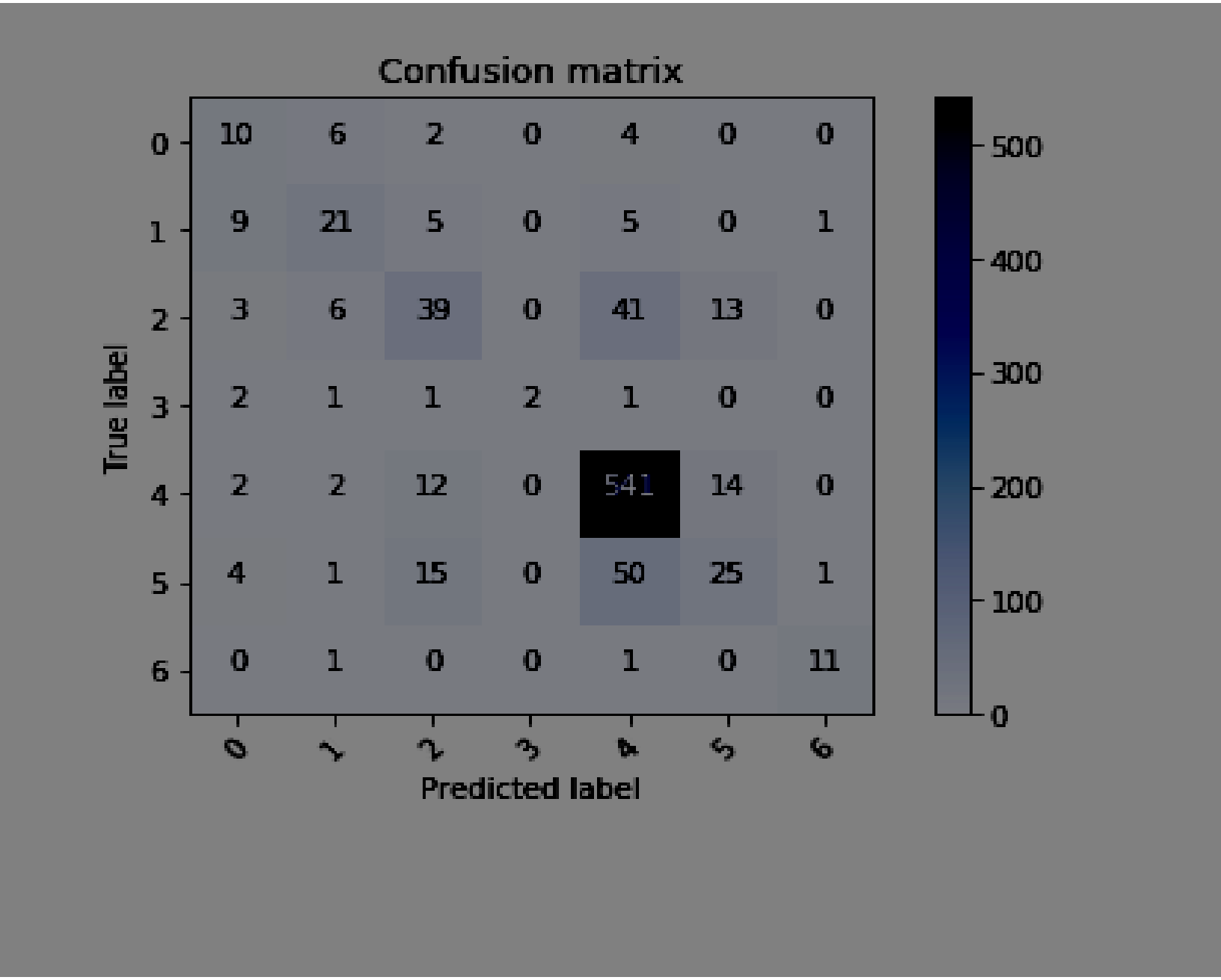


Figure 3.40 Confusion Matrix

3.7. Conclusion and Future Work

3.7.1 Conclusion

It seems our model has maximum number of incorrect predictions for Basal cell carcinoma which has code 3, then second most missclassified type is Vascular lesions code 5 then Melanocytic nevi code 0 where as Actinic keratoses code 4 has least misclassified type.

3.7.2 Proposed Future Work

We can also further tune our model to easily achieve the accuracy above 80% and I think still this model is efficient in comparison to detection with human eyes having 77.0344% accuracy

Chapter 4. Another Try in Skin Cancer Classification From Dermatoscopic Images Using Deep Learning

4.1 Augmentation & Transfer learning

4.1.1 Augmentation used on dataset

Rotation range=180,

Width shift range=10%,

Height shift range=10%,

Zoom range=10%,

Horizontal flip,

Vertical flip

4.1.2 Transfer learning

Is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.

It is a deep learning technique that enables developers to harness a neural network used for one task and apply it to another domain

The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

The biggest benefit of transfer learning shows when the target data set is relatively small. In many of these cases, the model may be prone to overfitting, and data augmentation may not always solve the overall problem.

4.1.3 Fine-Tuning of a Pre-Trained Network

Is common technique in transfer learning. Fine tuning means taking weights of a trained neural network and use it as initialization for a new model being trained on data from the same domain (often e.g. images). It can be implemented as following:

The common practice is to truncate the last layer (softmax layer) of the pre-trained network and replace it with our new softmax layer that are relevant to our own problem. For example, pre-trained network on ImageNet comes with a softmax layer with 1000 categories.

If our task is a classification on 10 categories, the new softmax layer of the network will be of 10 categories instead of 1000 categories. We then run back propagation on the network to fine-tune the pre-trained weights. Make sure cross validation is performed so that the network will be able to generalize well.

4.2 Model: InceptionResNetV2

InceptionResNetV2, ResNet50, and Xception are distinguished as the top three convnets, providing state-of-the-art classification accuracies of
96.17%, 94.81%, and 93.57%, respectively
The InceptionResNetV2 network is constructed by integrating the two most successful deep CNNs, ResNet and Inception

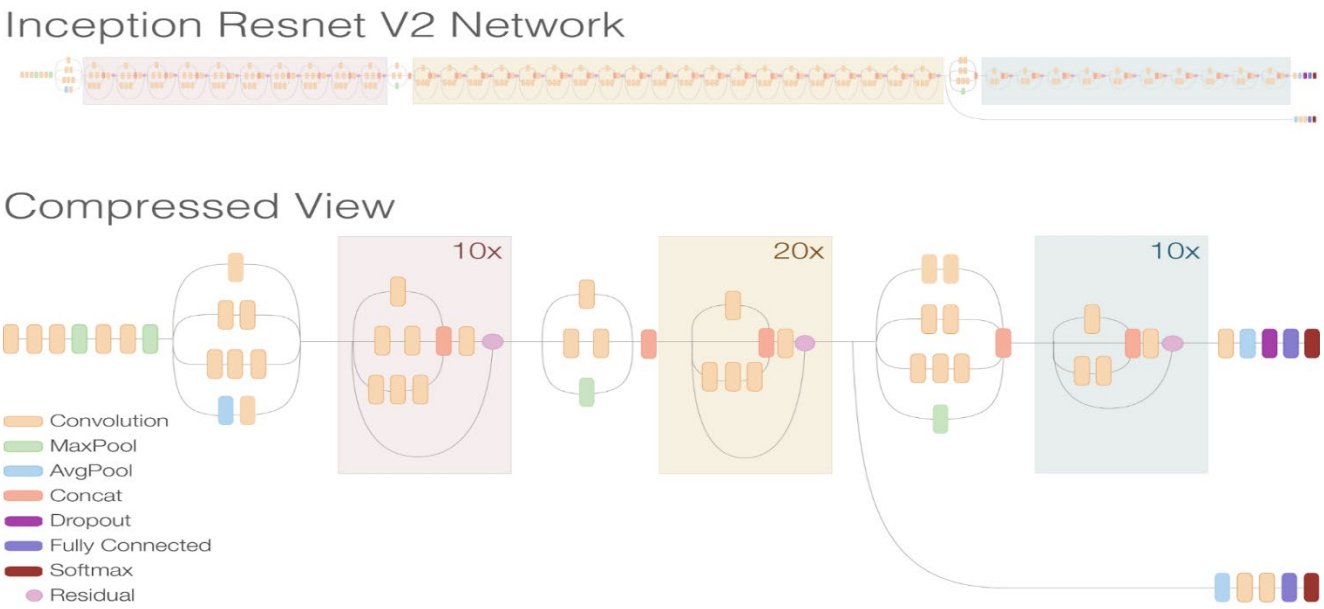


Figure 4.1: Modified structure and compressed view of Inception ResNet v2 network

Detailed view:

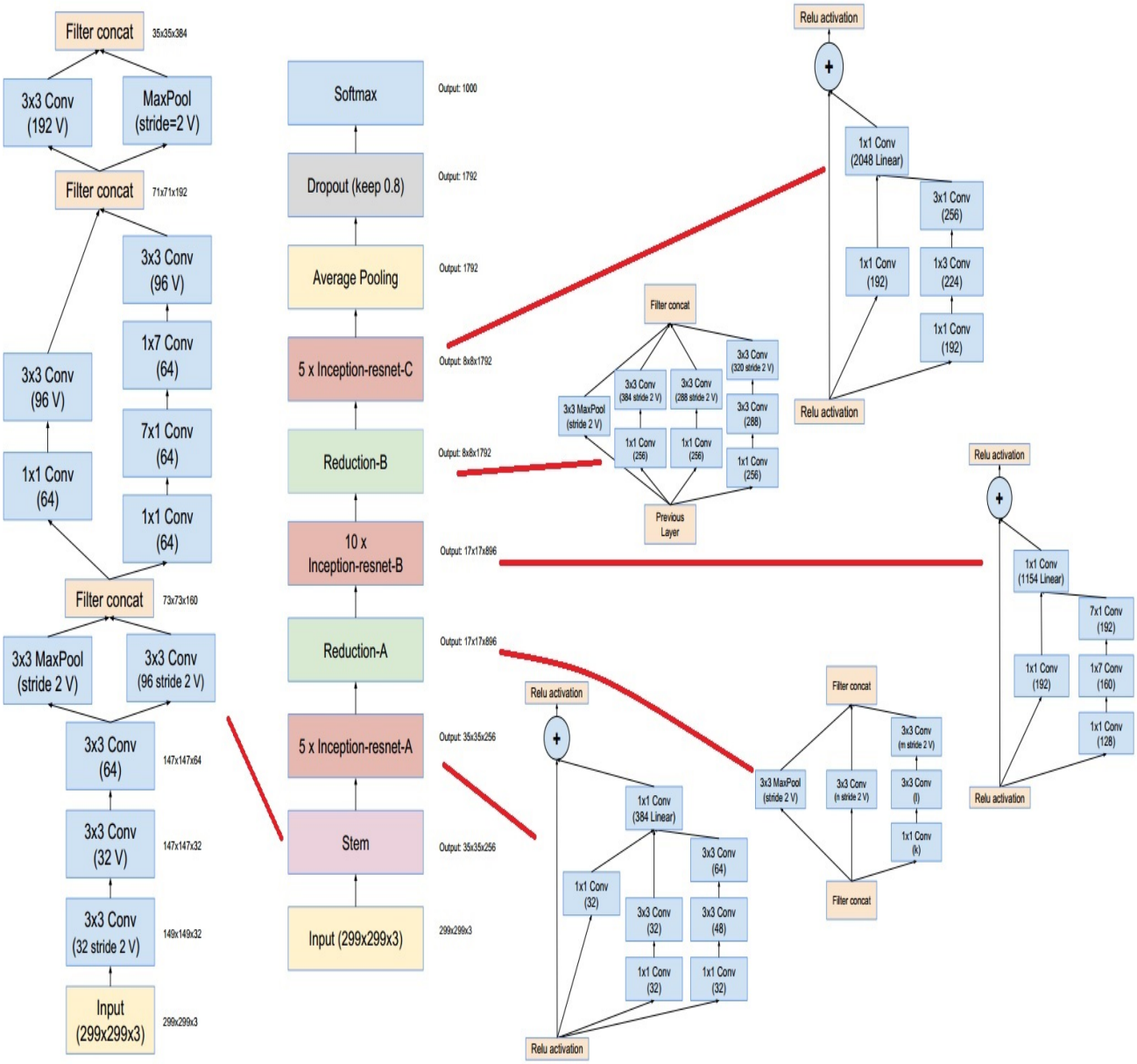


Figure 4.2: Detail view of structure of Inception ResNet v2 network

Introduce residual connections that add the output of the convolution operation of the inception module, to the input. For residual addition to work, the input and output after convolution must have the same dimensions. Hence, we use 1x1

convolutions after the original convolutions, to match the depth sizes (Depth is increased after convolution).

The "**stem**" here, refers to the initial set of operations performed before introducing the Inception blocks.

4.2.1 Inception modules.

There is three main inception modules, named A,B and C. They look very similar to their Inception v2 (or v3) counterparts.

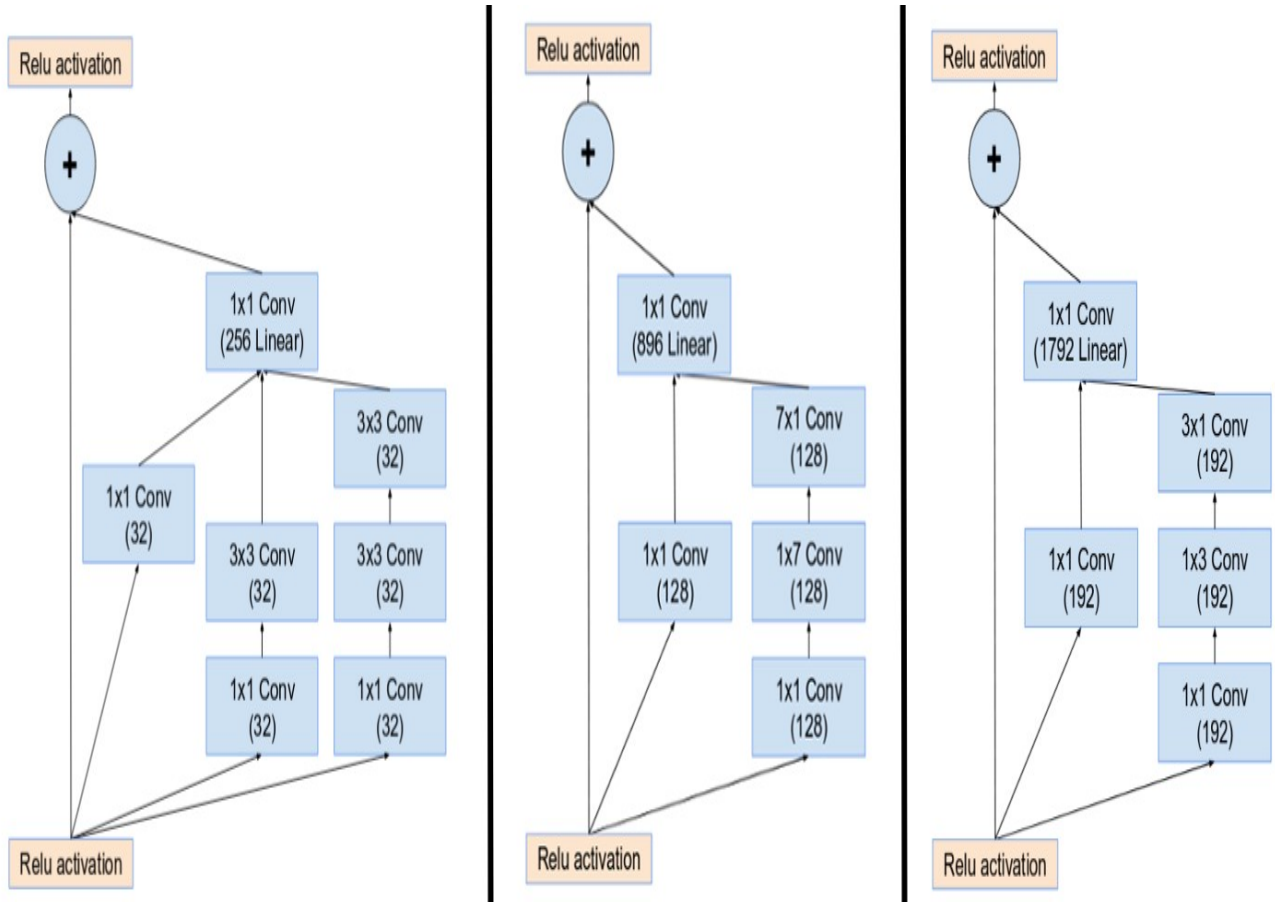


Figure 4.3: The three main inception modules in Inception Resnet v2

4.2.2 Reduction Blocks

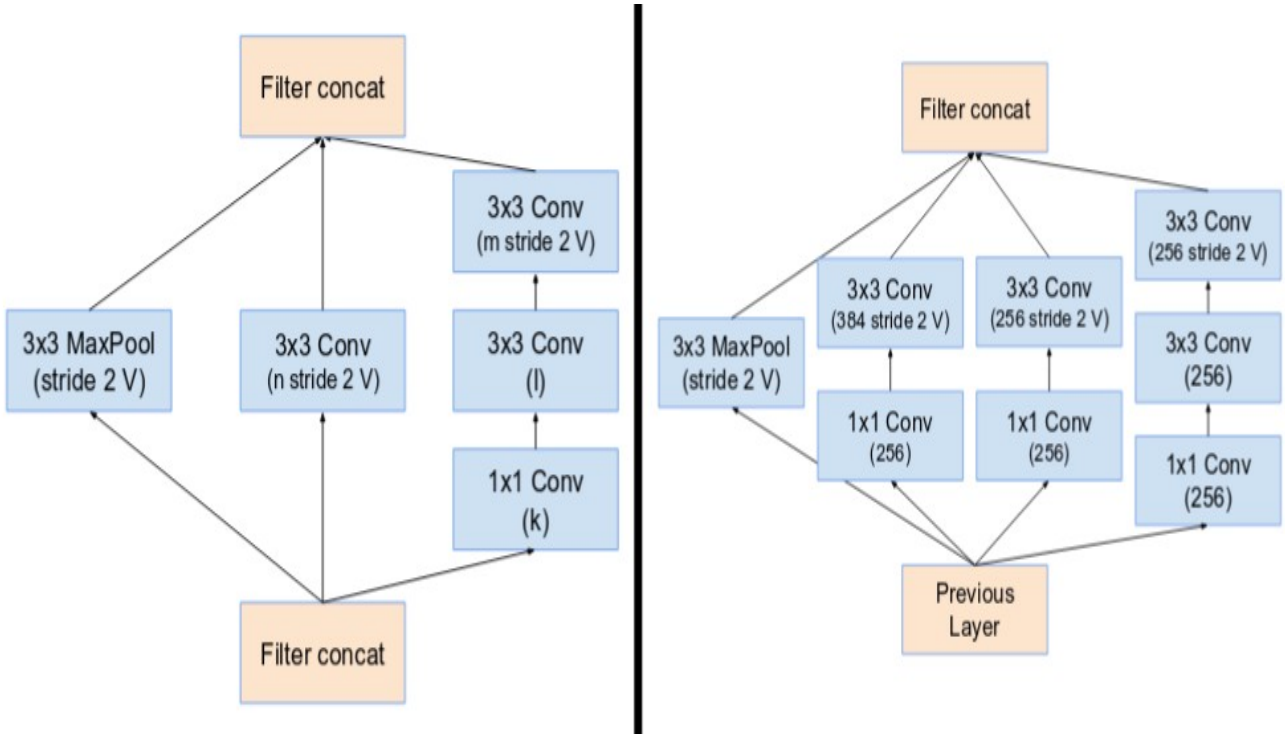


Figure 4.4: Reduction Block A (35x35 to 17x17 size reduction) and Reduction Block B (17x17 to 8x8 size reduction)

4.2.3 Activation scaling

Networks with residual units deeper in the architecture caused the network to “die” if the number of filters exceeded 1000. Meaning that the last layer before the average pooling started to produce only zeros after a few tens of thousands of iterations. This could not be prevented, neither by lowering the learning rate, nor by adding an extra batch-normalization to this layer. Hence, to increase stability, the authors scaled the residual activations by a value around 0.1 to 0.3.

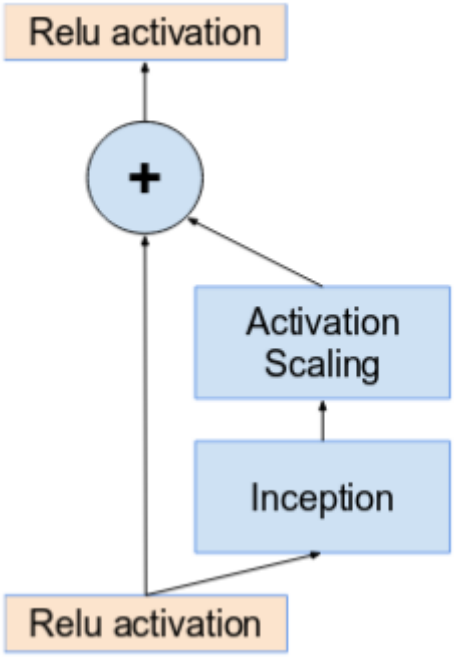


Figure 4.5: The general schema for scaling combined Inception Resnet modules.

4.3 Metrics

Accuracy & Loss

Training:

Accuracy is 92.02%, Loss is 25.42%

Validation:

Accuracy is 91.15%, Loss is 27.11%

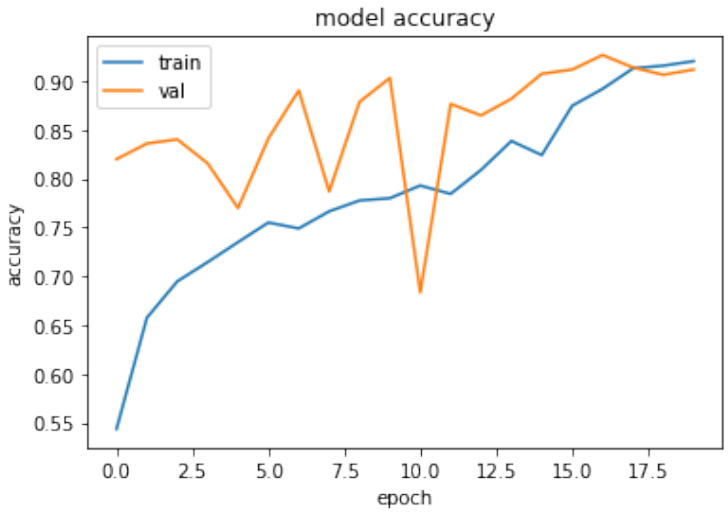


Figure 4.6: Accuracy of the model

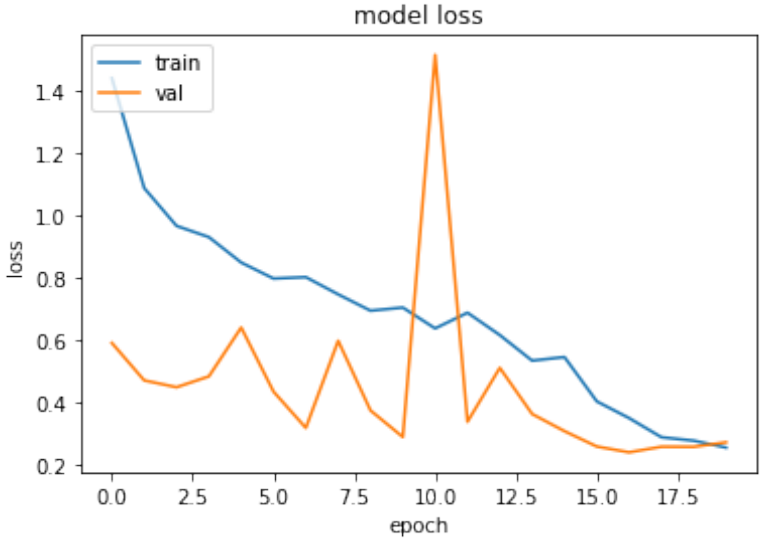


Figure 4.7: Loss of the model

Chapter 5 Diagnosing Pneumonia With Machine Learning Using Chest X-Ray Images

5.1. Pneumonia

Pneumonia is "a severe form of an acute lower respiratory infection that specifically affects the lungs" and is typically caused by bacteria, viruses or fungi. The lungs reaction to these foreign microbes is to cause an inflammatory response causing the bronchioles and alveoli to fill with fluid and become solid

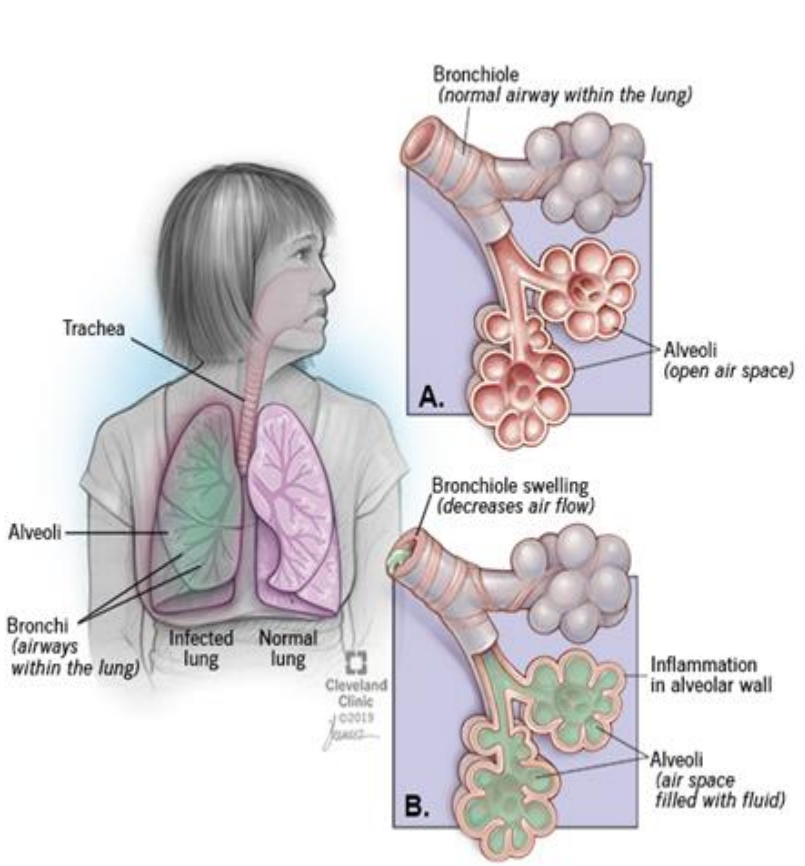


Figure 5.1: Difference between infected lung and normal lung

Clinically Relevant Anatomy

The lungs are responsible for the gaseous exchange of carbon dioxide and oxygen and consist of bronchi, which divide into bronchioles that end in alveoli. The capillaries (small blood vessels) that are found in abundance in the lungs are responsible for gaseous exchange. The exchange of these two gases is known as respiration:

- On inhalation oxygen entering the lung where crosses into the bloodstream, via the capillaries, for distribution around the body
- Carbon dioxide, a waste product of cell metabolism, enters the lungs from the body in the bloodstream and crosses over into the lungs where it is then exhaled into the atmosphere. Moving out of the lungs.

During a Pneumonia infection, the alveoli of one or both lungs fill up with pus or fluid. This increases the work of breathing, and thus gaseous exchange cannot occur as it normally would.

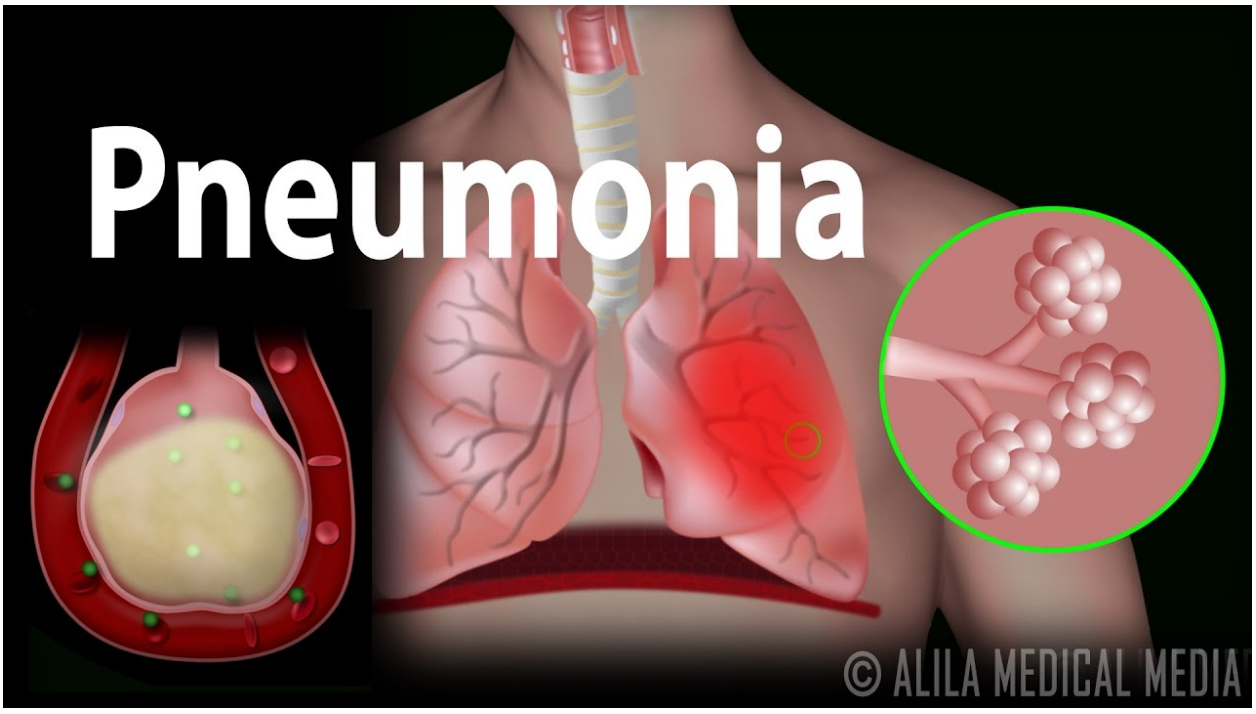


Figure 5.2: Lung during pneumonia infection

The Body's Defense against Pneumonia

The body has several defense mechanisms against the agents that can cause Pneumonia:

- Coughing
- Mucociliary escalator - lines the airway that assists the movement of bacteria out of the airways and away from the lungs
- Macrophages

If these mechanisms fail and a microbe is successful in colonising the alveoli they then multiple and quickly move over into the lung tissue activating an inflammatory response; the result is Pneumonia.

5.1.1 Causes of Pneumonia

Pneumonia is caused by a number of infectious agents, including viruses, bacteria and fungi. When these germs enter the lungs, they can overwhelm the immune system and invade nearby lung tissues, which are very sensitive. Once infected, the air sacs in the lungs become inflamed, causing coughing, fever, chills, and breathing problems.

You can get pneumonia as a complication of viral infections such as COVID-19 or the flu, or even a common cold. But bacteria, fungi, and other microorganisms can also cause it.

5.1.2 Types of pneumonia

- viral pneumonia – most commonly caused by the respiratory syncytial virus (RSV) and sometimes influenza type A or B; viruses are a common cause of pneumonia in young children
- aspiration pneumonia – caused by breathing in harmful substances, such as smoke or a chemical
- hospital-acquired pneumonia – pneumonia that develops in hospital while being treated for another condition or having an operation; people in intensive care on breathing machines are particularly at risk of developing ventilator-associated pneumonia

5.1.3 Pneumonia Transmission

Pneumonia is spread when droplets of fluid containing the pneumonia bacteria or virus are launched in the air when someone coughs or sneezes and then inhaled by others. You can also get pneumonia from touching an object previously touched by the person with pneumonia (transferring the germs) or touching a tissue used by the infected person and then touching your mouth or nose.

Who is most at risk for getting pneumonia?

People who have an increased risk of pneumonia include:

- People over the age of 65 and infants under age 2. The weakening immune system of people makes them less able to fight off illnesses
- People with a health-caused weakened immune system. Examples include:
    - People who are receiving chemotherapy
    - Transplanted organ recipients
- People who have health conditions that affect the lungs or heart

- People who are in the hospital. In particular, people in the ICU or anyone recovering who spends a large amounts of time lying on their backs. This position allows fluids, mucus or germs to settle in the lungs. People who need ventilators to breathe are at even greater risk since they have a difficult time coughing up germs that could cause a lung infection.
- People who smoke or drink alcohol. Smoking damages lung tissue and long-term alcohol abuse weakens the immune system.

5.1.4 Diagnosing pneumonia

A general practitioner (GP) may be able to diagnose pneumonia by asking about your symptoms and examining your chest. Further tests may be needed in some cases.

You may need a chest X-ray or other tests, such as a sputum (mucus) test or blood tests, if your symptoms have not improved within 48 hours of starting treatment. Pneumonia can be difficult to diagnose because it shares many symptoms with other conditions, such as the common cold, bronchitis and asthma.

5.1.5 Diagnosing pneumonia with machine learning

Build a deep learning model to automatically identify whether a patient is suffering from pneumonia or not by looking at chest X-ray images. The algorithm had to be very accurate because lives of people is at stake.

5.2.Dataset

https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Dataset is 5856 images split into 3 folders each folder has Pneumonia labeled images and Normal labeled images

-Train folder has 5216 images split into 1340 Normal images and 3875 Pneumonia images

-Validation folder has 16 images split into 8 each

-Test folder has 624 images split into 234 Normal images and 390 Pneumonia images

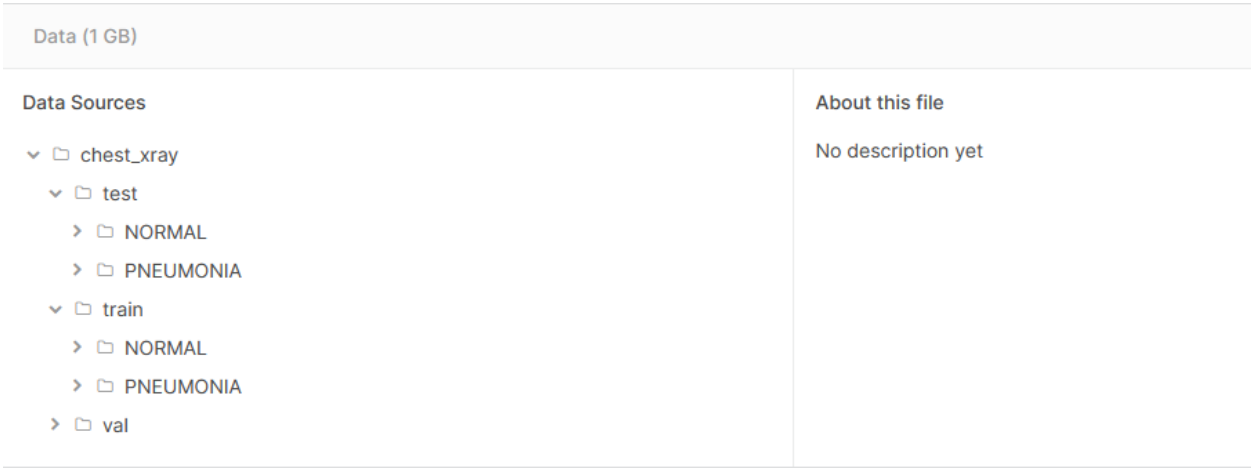


Figure 5.3: hierarchy of the dataset folders

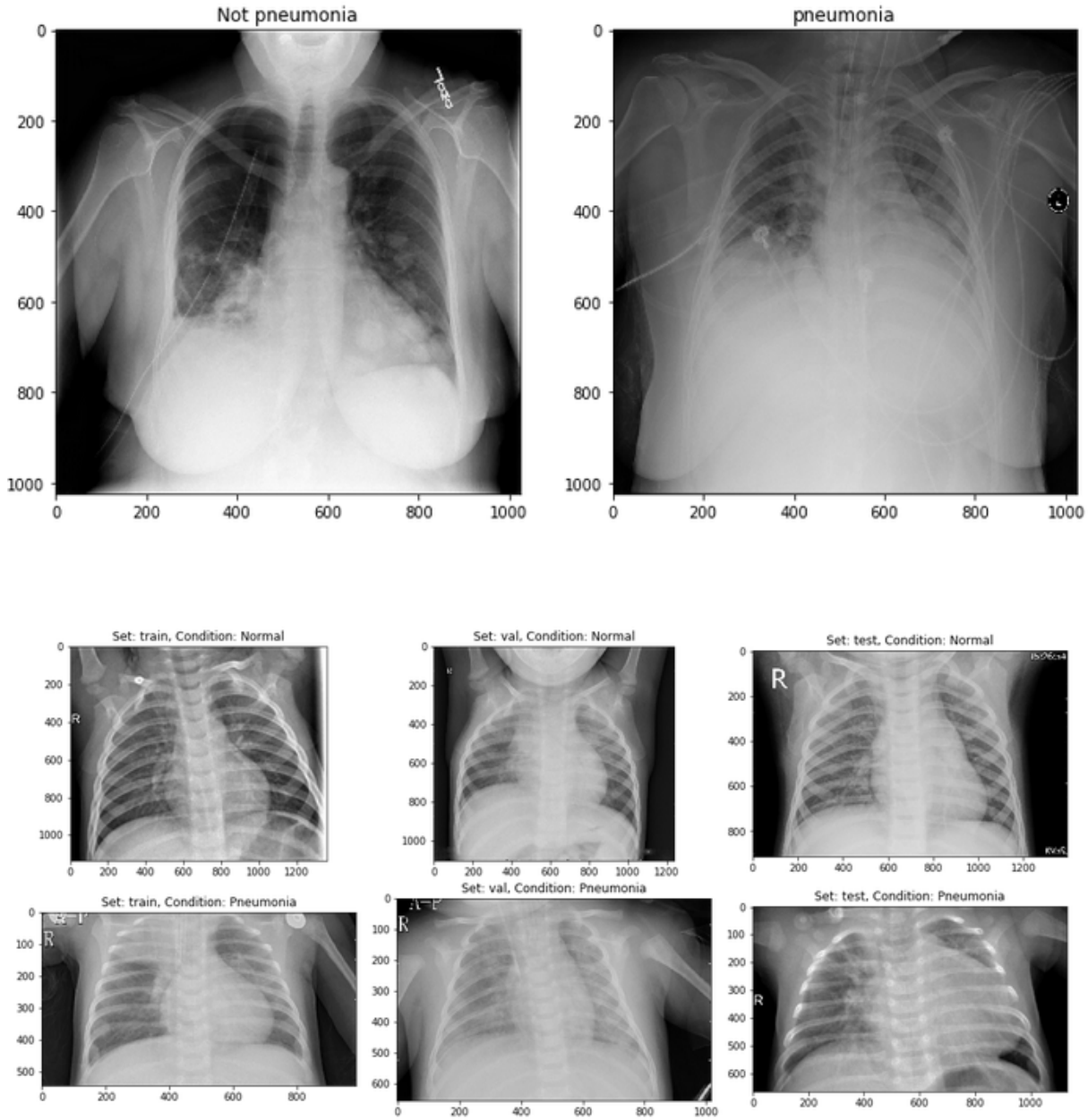


Figure 5.4: Sample X-ray images. Images were resized to 150 by 150

5.3. Processing the data using Augmentation

Image 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. Augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images. Transforms include a range of operations from the field of image manipulation, such as shifts, flips, zooms, and much more the practice of data augmentation is an effective way to increase the size of the training set. Augmenting the training examples allow the network to “see” more diversified, but still representative, data points during training.

Augmentation used in model:

Zoom range=30%,

Horizontal flip

5.4. Model structure

Model is a standard convolutional neural network which consists of 5 convolutional blocks each consists of convolutional layer, max pooling and batch normalization Batch normalization used to put the data on the same scale to increase performance of training and bring stability to the model dropouts are used as a regularization method to avoid overfitting in the model

Dense layers were used for classification with Relu as the activation function except for the last layer where Sigmoid was used for the binary classification .Adam was used as the optimizer for optimizing weights and binary cross entropy was used for the loss

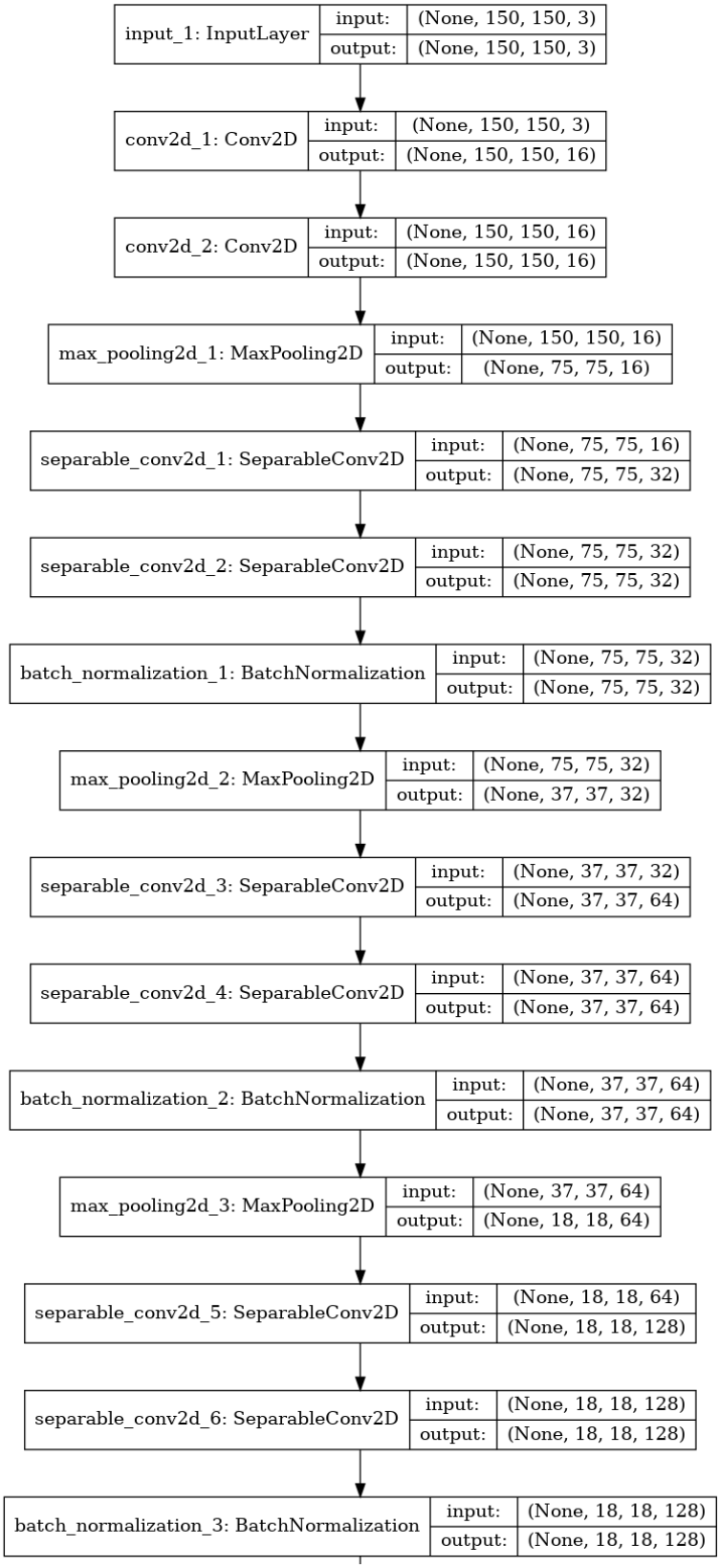


Figure 5.5a: Model structure

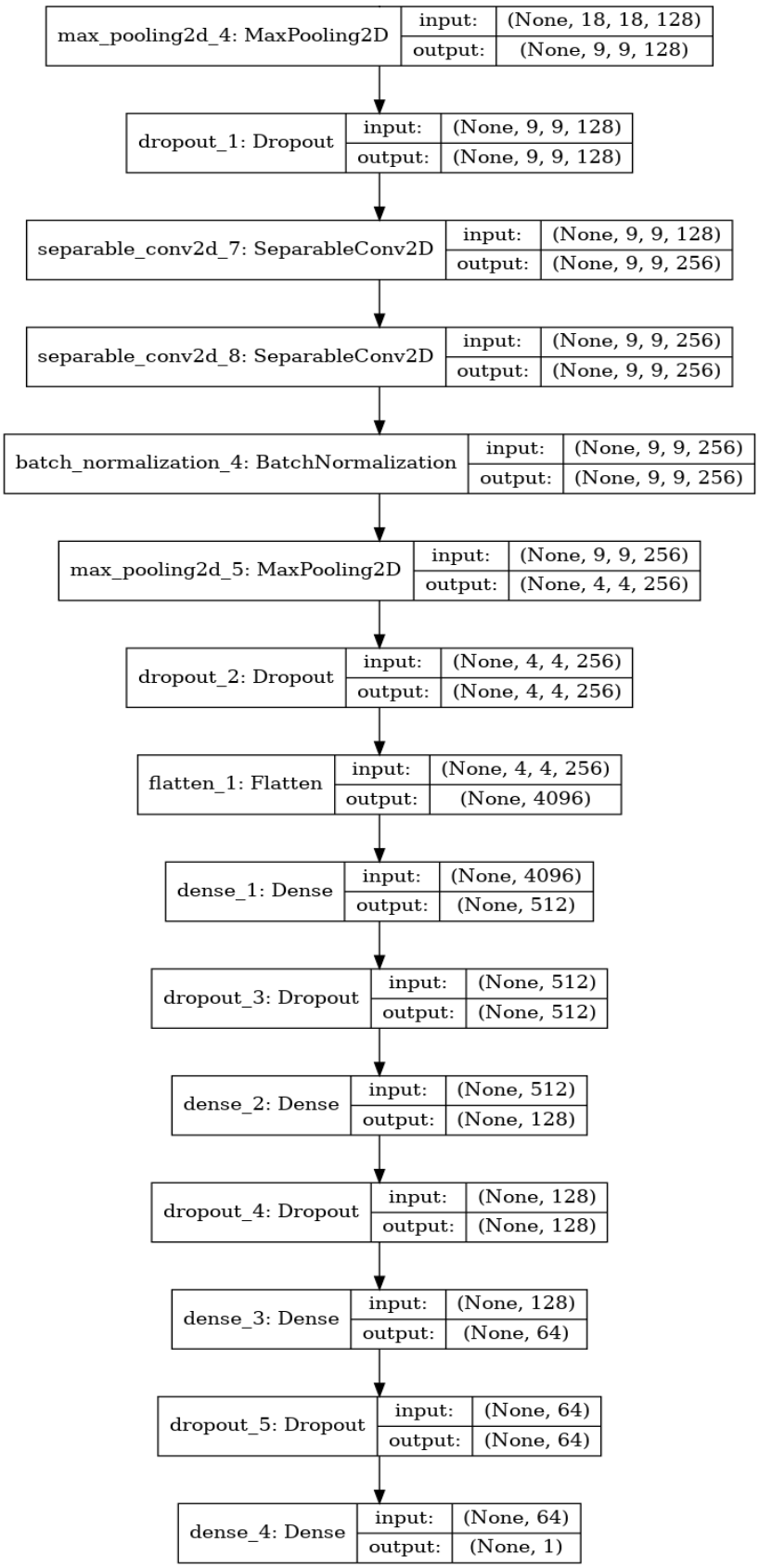


Figure 5.5b: Model structure

5.5. Metrics

Training:

Accuracy is 95.32%, Loss is 13.38%

Validation:

Accuracy is 90.03%, Loss is 32.68%

```
161/163 [============================>.] - ETA: 1s - loss: 0.1375 - accuracy: 0.9507
162/163 [============================>.] - ETA: 0s - loss: 0.1380 - accuracy: 0.9504
163/163 [==============================] - 98s 603ms/step - loss: 0.1375 - accuracy: 0.9507 - val_loss: 0.2388 - val_accuracy: 0.8716
Epoch 10/10
  1/163 [..............................] - ETA: 14s - loss: 0.0595 - accuracy: 0.9688
  2/163 [..............................] - ETA: 14s - loss: 0.1001 - accuracy: 0.9531
  3/163 [..............................] - ETA: 14s - loss: 0.0885 - accuracy: 0.9583
157/163 [===========================>..] - ETA: 3s - loss: 0.1332 - accuracy: 0.9534
158/163 [============================>.] - ETA: 2s - loss: 0.1327 - accuracy: 0.9535
159/163 [============================>.] - ETA: 2s - loss: 0.1323 - accuracy: 0.9536
160/163 [============================>.] - ETA: 1s - loss: 0.1334 - accuracy: 0.9535
161/163 [============================>.] - ETA: 1s - loss: 0.1332 - accuracy: 0.9536
162/163 [============================>.] - ETA: 0s - loss: 0.1338 - accuracy: 0.9535
163/163 [==============================] - 98s 599ms/step - loss: 0.1338 - accuracy: 0.9532 - val_loss: 0.3268 - val_accuracy: 0.9003
```

Figure 5.6: Evaluation of the model

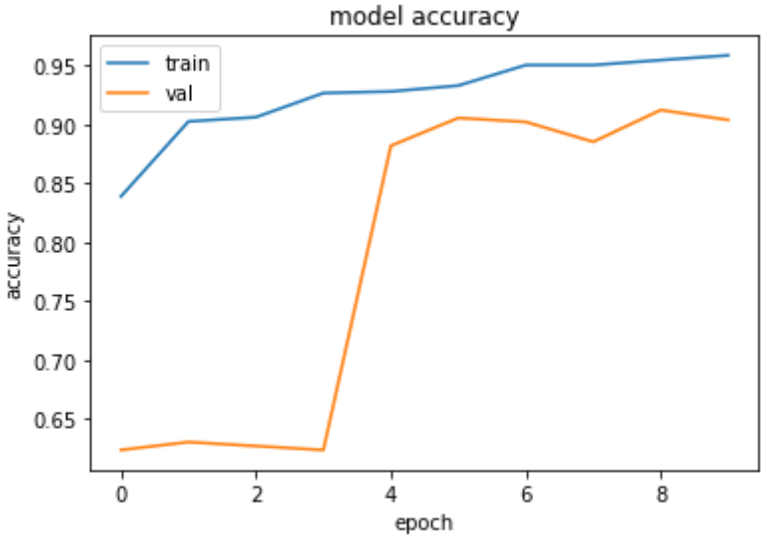


Figure 5.7: Accuracy of the model

Loss

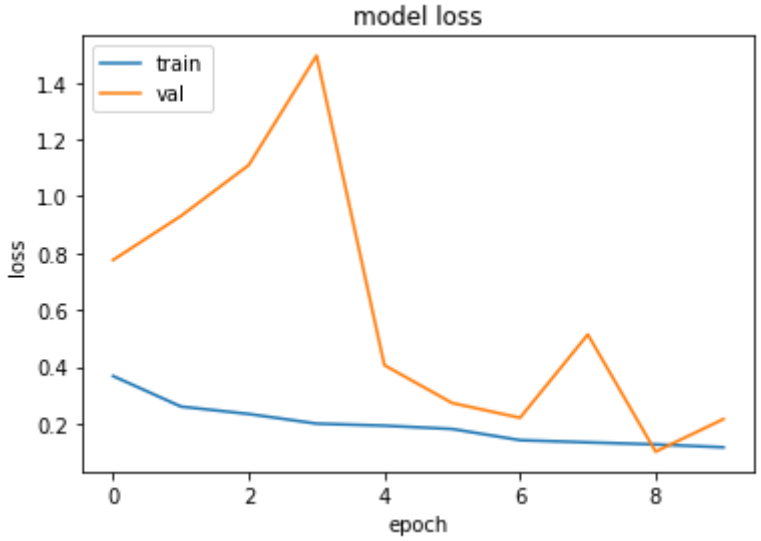


Figure 5.8: Loss of the model

Test accuracy is 90.06%

Predictions

```
> def preparepic(filepath):
    img_size = 150
    img_array = cv2.imread(filepath)
    new_array = cv2.resize(img_array, (img_size, img_size))
    return new_array.reshape(-1, img_size, img_size, 3)
dict = ["NORMAL", "PNEUMONIA"]
prediction = model.predict([preparepic("../input/chest-xray-pneumonia/chest_xray/test/NORMAL/IM-0021-0001.jpeg")])
print(dict[int(prediction[0][0])])
NORMAL
```

```
> def preparepic(filepath):
    img_size = 150
    img_array = cv2.imread(filepath)
    new_array = cv2.resize(img_array, (img_size, img_size))
    return new_array.reshape(-1, img_size, img_size, 3)
dict = ["NORMAL", "PNEUMONIA"]
prediction = model.predict([preparepic("../input/chest-xray-pneumonia/chest_xray/test/NORMAL/IM-0013-0001.jpeg")])
print(dict[int(prediction[0][0])])
NORMAL
```

Figure 5.9: Sample prediction code

Chapter 6. Disease Detection Web Application (Dr.Tech)

6.1. Implementation Details

First and foremost, we had to agree on the basic tools that will be used for the user interface implementation, and as a result of our discussion we agreed to use JavaScript language as it is the latest, and most popular used language for web development in the current time frame. Moreover, concerning the front-end of our website, we had to choose between React or Angular frameworks so we chose React, and began to construct our design plan. As discussed above, our website depends on deep learning technology using well-known models as the backbone of its back-end to add the artificial intelligence part to our website by providing large quantities of train data presented as high definition images of infected areas that the patients suffer from. The website begins to learn from input data by image processing, and classifying different cases for each disease aiming to increase the accuracy of our models as much as possible. The point or the general concept of machine learning technologies is training our models as much as possible by applying large amounts of data in order to decrease the error percentage to reach the well-known standards or even aim for better.

6.2. Front-End Design

After analyzing a lot of similar medical websites (i.e. vezzeta, Egyptian website for reserving doctor appointments) we had a vision for how will the website will

appear to the user ,and how we can design it to be suitable for all clients. We didn't want to complicate the design to facilitate using the website ,providing an easy approach to reach or navigate through all the pages of our website with ease ,and describing each part of each page with simple labels or sentences. Our website composes of exactly three main pages as We didn't think that we would need to implement more pages that would appear as just stuffing unnecessary visualization components that wouldn't hold any main functionality or even secondary ones therefore it was more than sufficient to implement only 3 main pages to represent the UI (or user interface) for our website.

However, our main challenge wasn't the front-end of the website but it was mainly understanding the complex deep learning models used as the back-end of our website for image processing, that was our main goal in which to have a deep understanding for how those models work , not just classifying different diseases but also risk levels for each disease ,and how exactly is image processing done in details. We will now discuss each part of our written code for the front-end of the website followed by the back-end code that will be discussed separately later in another section.

6.2.1. Homepage

As mentioned before , we agreed on using JavaScript language for implementing our pages through React framework which is one of the newly created frameworks for web development ,and it is specifically used by software developers for implementing front-ends of web applications along with other frameworks such as Angular. Our homepage is composed of three main components: a navigation bar, a carrousel , and a footer ,along with an intermediate body with 3 hyperlinks that summarizes the procedures that should be done by a client to use the service available. Considering our first component, it is hard to find any website nowadays

that doesn't have a navigation bar as part of its page components. It has been used for many years till today even if there were slight design changes that differ a website from another, it still holds the same functionality ,and same concept of making it easier for the client to move around or surf the website easily. The navigation bar is composed of a number of hyperlinks that allows the client to move from one page to another. We added a home hyperlink to return to the homepage, a sign up hyperlink so that the client can create a personal account to hold his/her medical data ,and previous diagnosis ,and a log in hyperlink for regular users.

The second component of the first page is a simple carrousel composed of three slides. The first slide provides a simple description for our website including what service it provides exactly. The second slide is an encouraging slide for a new visitor to create an account which is a well-known strategy used in many websites to catch the user's attention ,and encourage them to create a personal account followed by offering additional services with additional charges but as we explained we are just trying to present our work professionally as possible ,and we are not concerned about using the website in practical life ,focusing more on the work done in the implementation itself as applying our work practically requires a medical approval. Therefore, focusing on marketing strategies in how we can attract as many users as possible was not our goal or main concern. The third slide is composed of a simple quote to add some visual effects to the carrousel.

Finally, the third and last component of the homepage is a bottom-positioned navigation bar (or a footer) that holds some general information about the website similar to what we see in nowadays websites such as the copyright , contact information , and links from well-known social media sites such as Facebook , Instagram , and twitter. A figure is given below showing the homepage which is a screenshot taken of our devices screen showing the results after executing the code. Note that all the code is written from scratch ,and nothing was copied from the internet , just the general code used to implement some components to save time in writing the code. Figure 6.1 below shows our index or homepage.

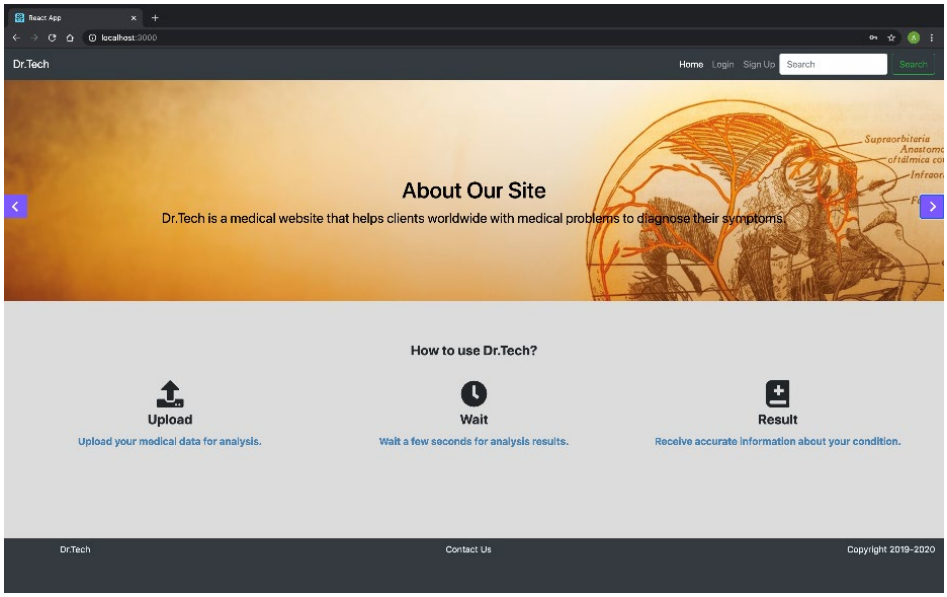
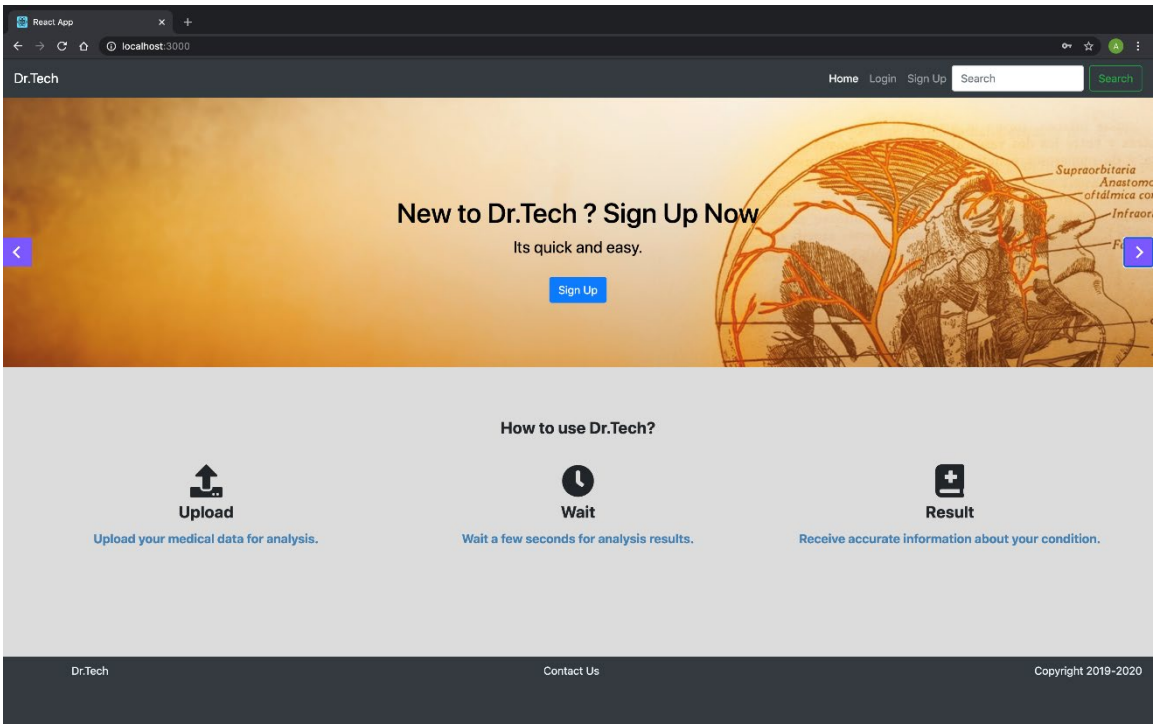


Figure 6.1 Homepage



6.2.2. Forms

In order for the client to create a new personal account or even log in to an already existing one , a form page had to be provided in order to allow that, therefore we designed our second page to be a dynamically changed one according to the client's choice whether to sign up creating a new account or to log in an existing one. In fact we can say that each form either the log in or the sign up can be seen as two different pages but we thought that it is unnecessary to describe it that way as we want it to appear to the user as a single dynamically changing page between sign up ,and log in therefore we didn't describe our website previously to be composed of 4 main pages instead of three.

We also implemented all the essential validation properties of both forms as there must be certain rules or policies when creating a new account or log in for a current one like for example the well-known format of an email address in which no email can be accepted unless it is in that format or leaving a recommended field empty that is crucial as part of the personal information needed to be provided by the client.

A rising question can pop considering implementing the forms mentioned above, why did we even need to create such forms ? given that our main goal was not to focus on such components as we could have simply allowed access to any client whether he/she is a regular client or a new one. The answer is pretty simple , we aimed to present our website as professionally as possible ,and didn’t want it to appear less than any other website practically used with no general components missing that should be found in any website nowadays. Figure 6.2 and 6.3 below shows the UI for the sign up ,and log in forms.

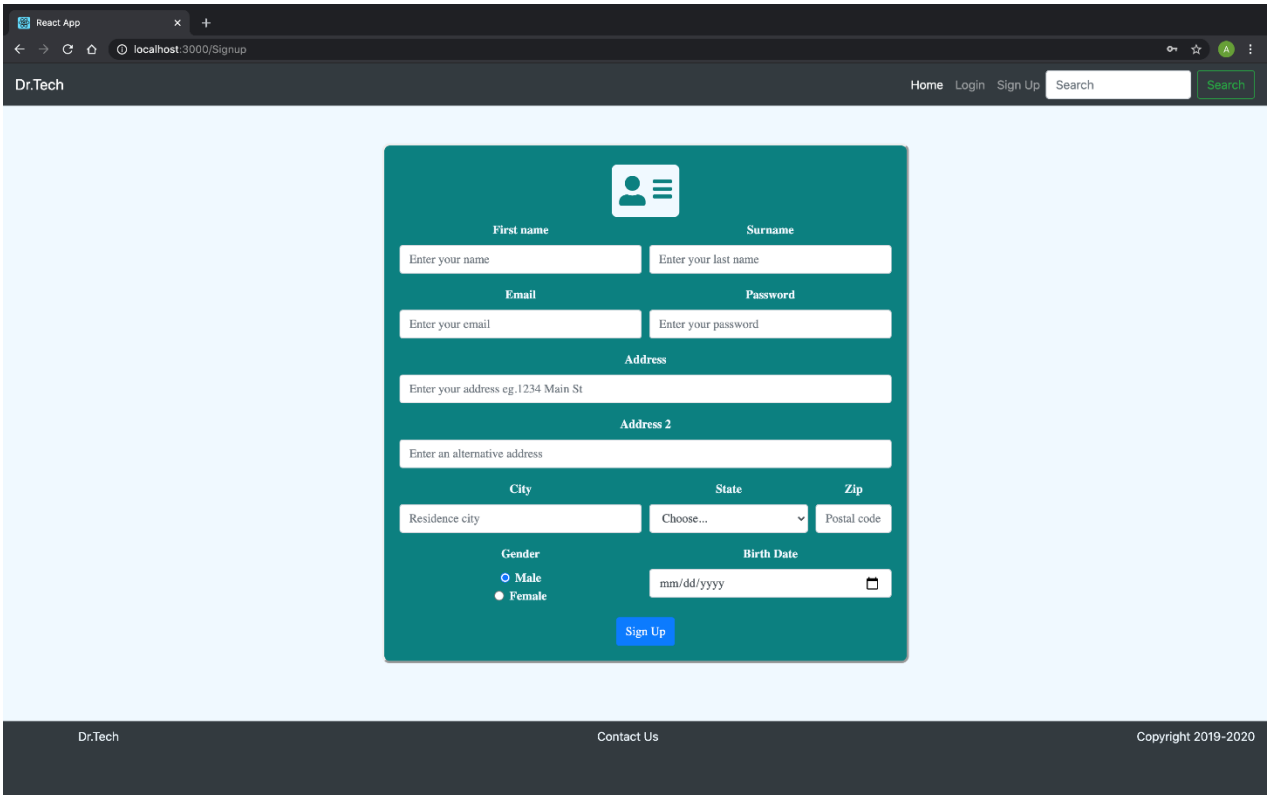


Figure 6.2 Sign Up

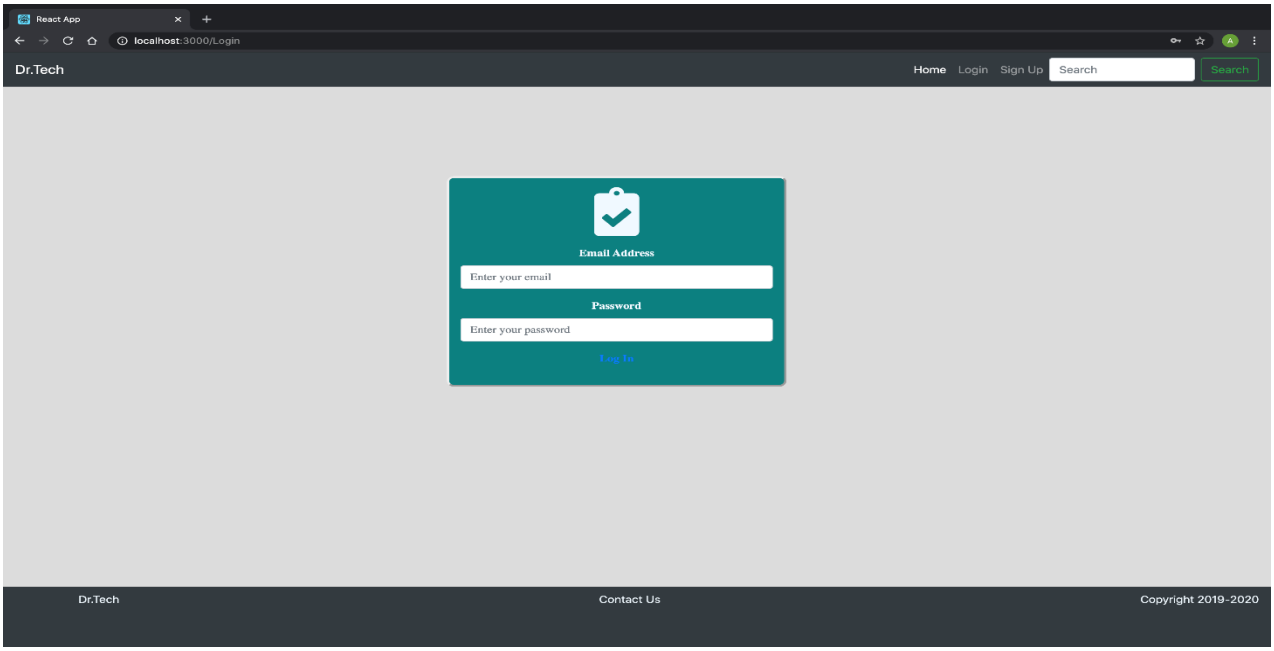


Figure 6.3 log In

6.2.3 Client page

For the third and main page of our web application , we implemented a page that facilitates for the client to upload his/her data represented as high definition medical images for processing and providing the right diagnoses of what he/she is suffering from exactly. The page simply composes of an upload data area or drop box for the client to upload the data then comes the image processing phase done by the deep learning model depending on the type of the photo as we have more than one model for more than one disease, and finally the result is given to the client (patient) describing his/her medical case ,and what they suffer from. A screenshot taken for the web page is provided below in figure 6.4 for the client web page.

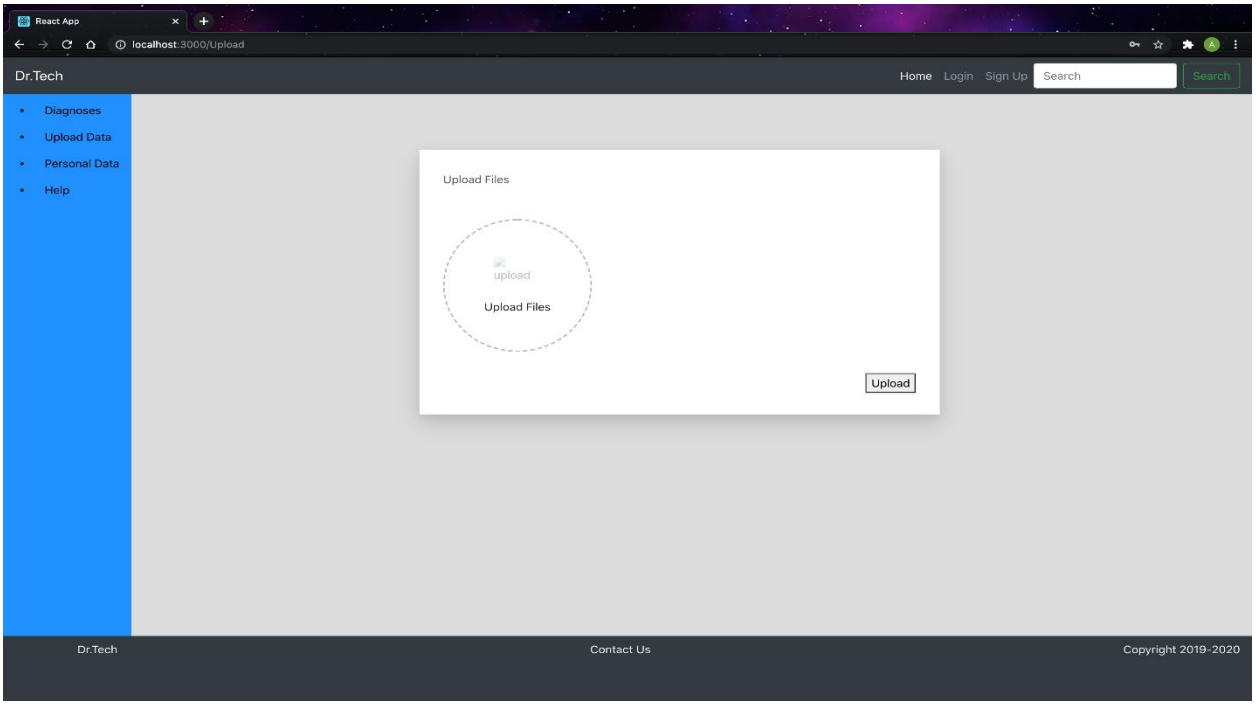


Figure 6.4 Client Page

6.3. Back-End Design

After discussing how we implemented our Front-End for the web application , we will now discuss in details our back-end design. Considering the size of the data to be stored by an API end point as part of the back-end design in which HTTP requests is sent to server in order to store new data , update existing one , or even delete unwanted data, we didn't find the need to store our data in a separate database created specially for our data as we didn't see that we needed such huge space for it ,added to that, our main goal as we discussed is to understand the deep learning models itself.

However, we still needed to store a amount of data such as the personal account information of clients ,and the medical images provided by each client for analysis. We had to look for a substitute to save these data even if it is not as crucial as making sure that our models work perfectly giving the highest accuracy percentage possible. That was when we decided to use JSON Placeholder.

6.3.1. JSON Placeholder

What is JSON Placeholder ? JSONPlaceholder is a free online REST API that you can use whenever we want to store , create , modify , and delete data without the need to create a new database from scratch. In fact, some developers use it's services to get some fake data ,and it had proven to be great for tutorials , testing new libraries ,and sharing code examples.

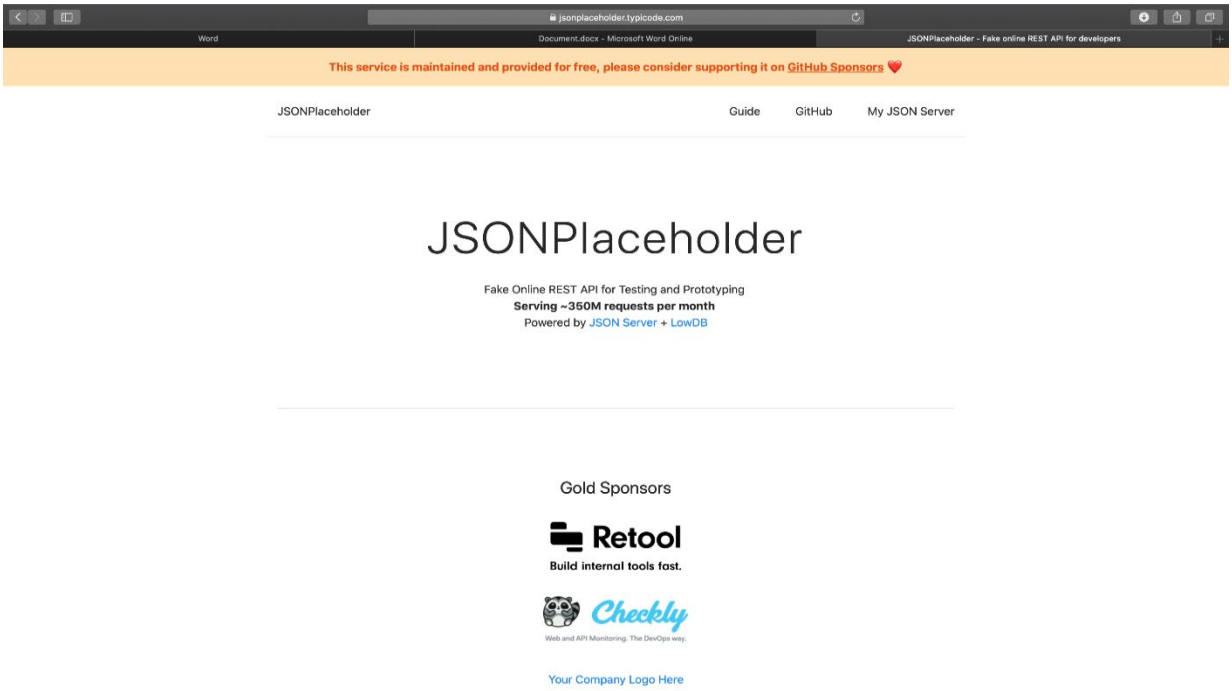
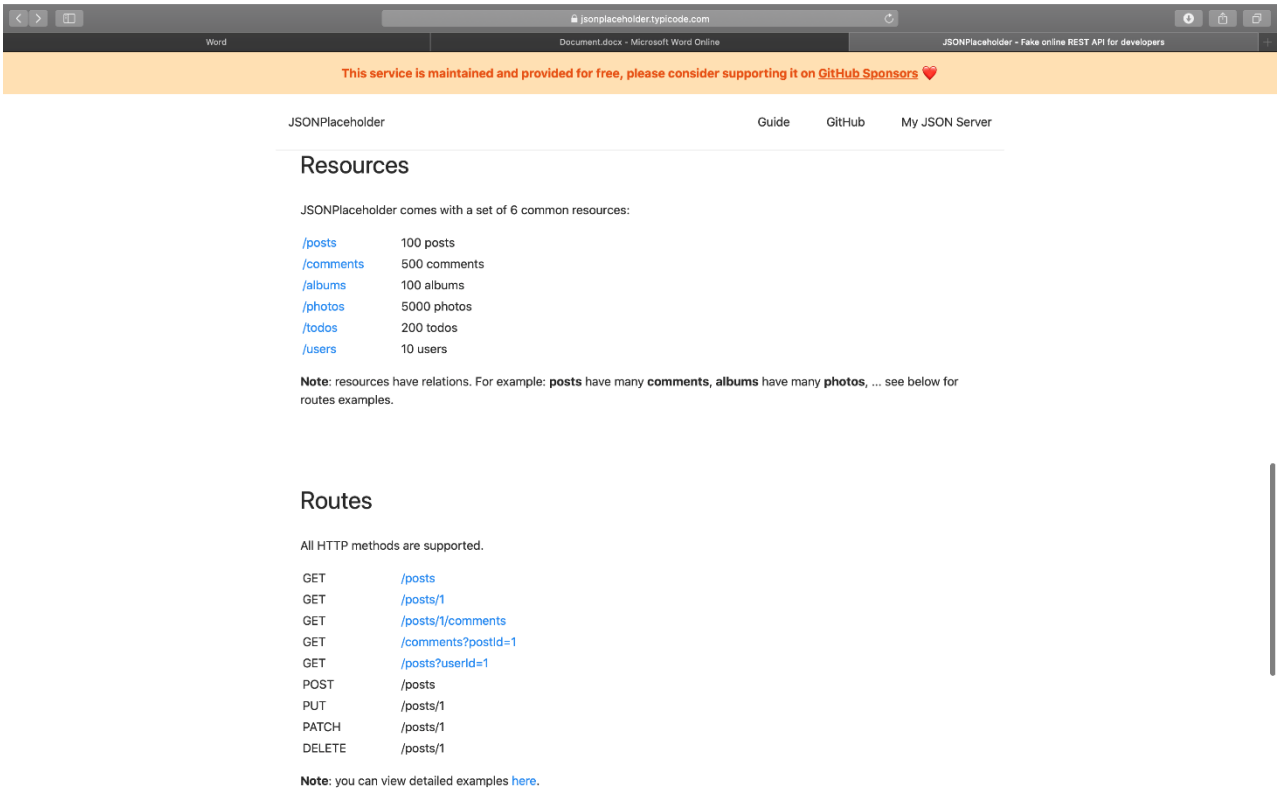


Figure 6.5: JSONPlaceholder

JSON Placeholder comes with a set of 6 common resources in which all HTTP methods are supported.



The next figure will be showing how a fake JSON server can be used to store an application data to be used whenever is needed to using My JSON Server online service ,and a simple github repo.

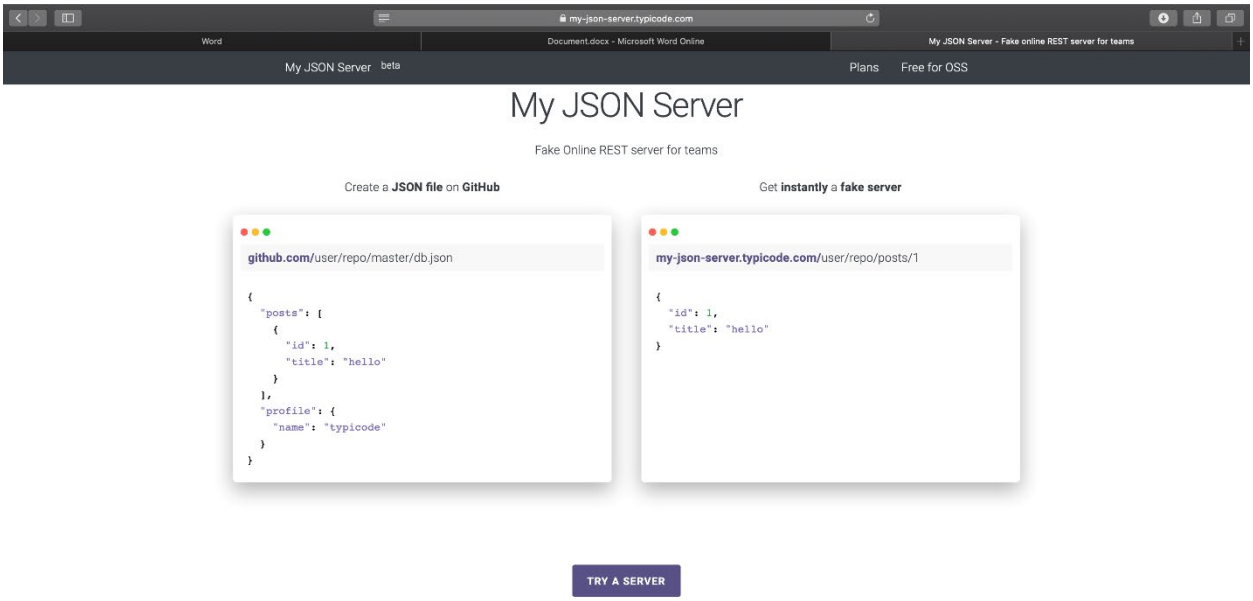


Figure 6.6: MY JSON Server

To explain more, using other databases for applications such as mongoDB or MySQL wasn't necessary for our application as we thought it would be a waste of time to use such services in order to store our data as it won't be much of a benefit for us therefore as shown above we used JSON Placeholder as a substitution which was more than enough.

6.3.2. Code Snippets

As we mentioned we used React framework (or JavaScript library) to implement our Front-End. Moreover, our Back-End was designed mainly using python programming language or to be more specific Flask which is a micro web framework written in python which was used as the link between our Front-End as a UI created by React ,and Back-End as our deep learning models.

Now we will provide some code snippets of our web application beginning with the Front-End. The editor used for writing our code was Visual Studio Code. Figure 2.1 shows the App.js file.

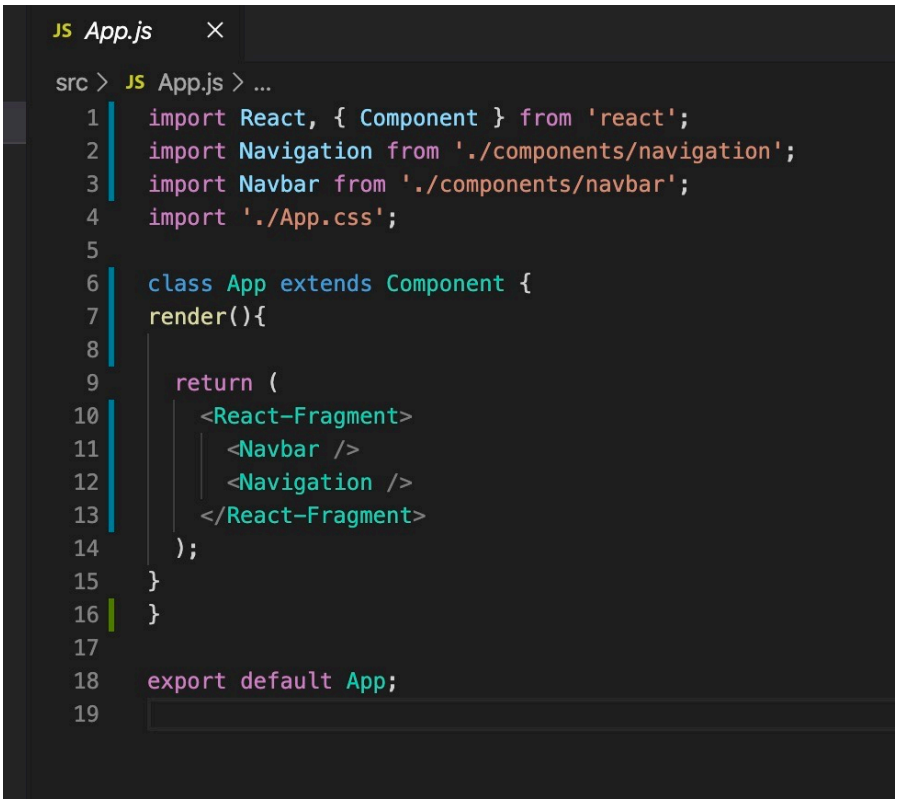


Figure 6.7App.js

In React we express any web page element as a component therefore for example we can say that a navigation bar is a component or Slider (aka carrousel) is a component in a React application. So we can say that each web page is composed of a number or a set of components that together create it. App.js file is the main file rendered by the React-Dom which is the element responsible for bringing its components to the GUI or UI that is seen by a client.

It is composed of two main components: the Navbar which is the navigation bar ,and a component called avigation which is responsible for the switching or real change from one page to another ,and as we examine these components closely we will understand more how all the other components viewed in our home page such as the carrousel , the footer ,and the hyperlinks are implemented.

Now we will take a closer look at the navigation component to understand its code details. Figure 6.8 shows the navigation component code.

Figure 6.8 Navigation Component Code

In that component, we imported another built-in component called the react-router-dom which is an element that is responsible for handling navigation through a react application ,and to be more precise there are actually two main components that handle it which are Route and Switch components imported from react-router-dom.

Each route component represents a link to a specific page ,whether it is the sign up or login form page , the home page , or the client personal page in which all those routes are included inside one switch ,and depending on a certain action taken , mainly a button press, it switches to the correct ,and wanted link in the website. As a final note considering the navigation component , we linked our slider component with the path of our homepage in order to appear each time we return to it.

To show how we linked our code with JSON Placeholder ,and provide a general view of how our data is stored we will now discuss one of the two forms codes which will be the Sign Up form code ,and how we implemented the link between it ,and JSON Placeholder Back-End services. Figure 6.9 shows the Sign Up form code.

signform.jsx ●

src > component > signform.jsx > Sign > handleSubmit

```jsx
import React, { Component } from 'react';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import {faAddressCard} from '@fortawesome/free-solid-svg-icons';
import Joi from 'joi-browser';
import Input from './input';
import axios from 'axios';
class Sign extends Component {
  state =
  {
    new_account: {firstname : "" , surname: "" , email: "" , password: "" , address1: "" , address2: "" , city: "" , zip: ""},
    errors: {}
  };

  schema =
  {
    firstname: Joi.string().required().label("First name"),
    surname: Joi.string().required().label("Surname"),
    email: Joi.string().required().label("Email"),
    password: Joi.string().required().label("Password"),
    address1: Joi.string().required().label("Address"),
    address2: Joi.string().required().label("Address1"),
    city: Joi.string().required().label("City"),
    zip: Joi.number().required().label("Zip")
  };

  handleSubmit = e =>
  {
      e.preventDefault();
      const errors = this.validate();
      this.setState({errors: errors || {}})
      if(errors) return;
      window.location.href = 'http//localhost:3000/Confirm';
      //call the server
      const account = {...this.state.new_account};
      const {data: post} = await axios.post('https://jsonplaceholder.typicode.com/users',account);
  }

  handleChange = ({currentTarget: input}) =>
  {
      const errors = {...this.state.errors};
      const errorMessage = this.validateProperty(input);
      if(errorMessage) errors[input.name] = errorMessage;

      const new_account = {...this.state.new_account};
      new_account[input.name] = input.value;
      this.setState({ new_account });
  }

  validate = () =>
  {
      const options = {abortEarly: false};
      const { error } = Joi.validate(this.state.new_account,this.schema,options);
      if(!error) return null;
      const errors = {};
```

Ln 35, Col 9

Figure 6.9: Sign Up Form

Most of the Sign Up Form code is just the simple general code for any well-known form in a typical website , composed of an input field for each required value such as the first name , last name , email address , and so on. The most important part that we really want to discuss is passing the data for every new account that is created in the website to the My JSON server that we discussed earlier. We didn't find that to be complicated as we simply imported the library that is responsible for that link which is axios after installing it of course ,and added the required lines of code the method or the function responsible for handling the submission of the form which is handleSubmit().

After making sure that there are no validations errors in the form, we clone the data from the new_account object to another object called account which prevents altering the original state in the form ,and preserving it then passing this account object to post its data values in the My JSON server. Similarly, we can use GET HTTP method in the login in form to check the input data with the present one for verification purposes.

Another important note that need to be cleared out is that we used joi library which is the most powerful validation JavaScript library used nowadays for validating data in web application components such as forms ,and similar other components. The upcoming figures will just show the rest of the Sign Up form code.

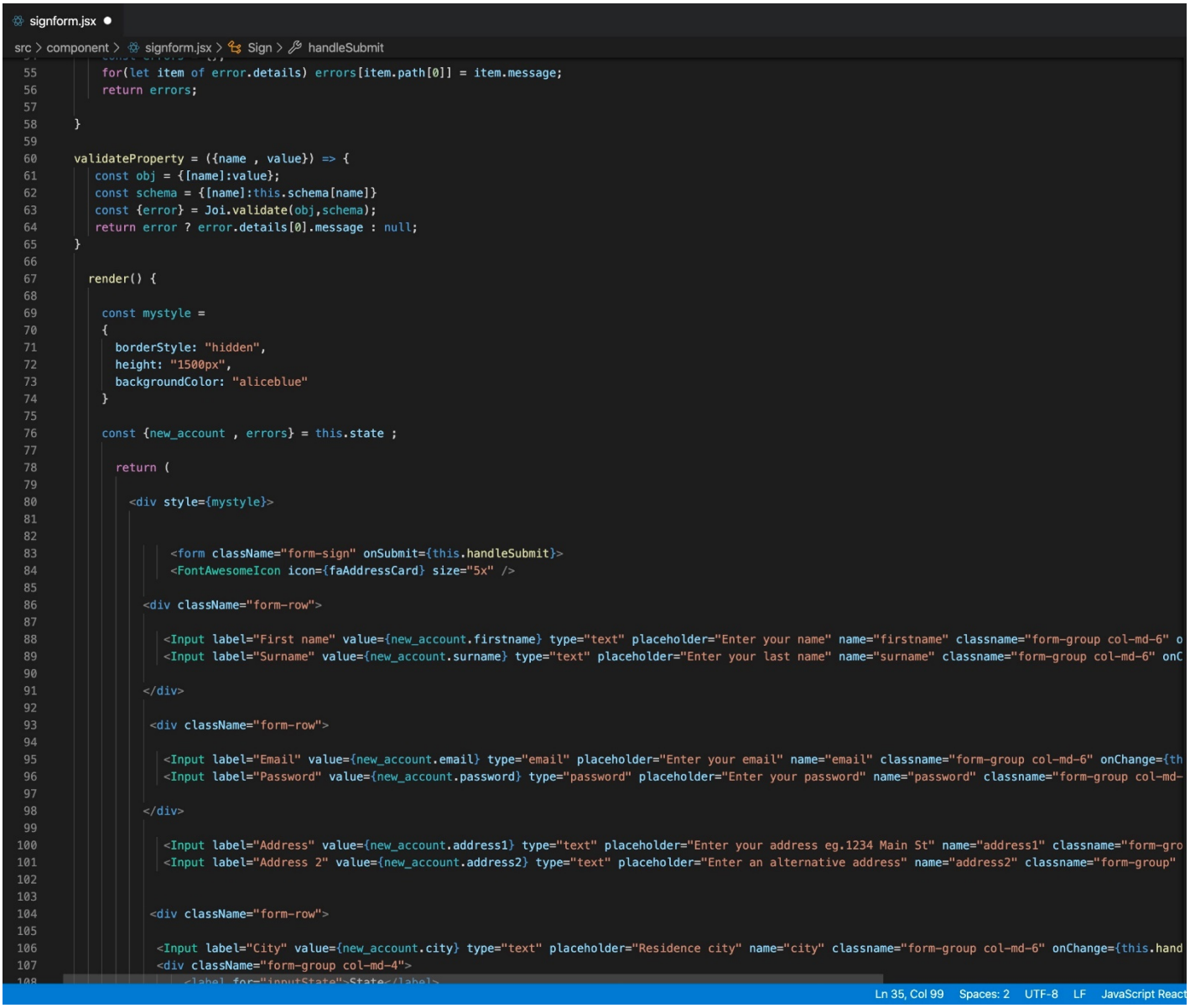


Figure 6.10: Sign Up Form (a)

signform.jsx ●

src > component > signform.jsx > Sign > handleSubmit

```
                <div className="form-group col-md-4">
                  <label for="inputState">State</label>
                  <select id="inputState" className="form-control">
                    <option selected>Choose...</option>
                    <option>...</option>
                  </select>
                </div>
                <Input label="Zip" value={new_account.zip} type="text" placeholder="Postal code" name="zip" classname="form-group col-md-2" onChange={this.handleCha
              </div>

    <div className="form-row">
      <div className="form-group col-md-6">
        <label>Gender</label>
      <div className="form-check">
    <input className="form-check-input" type="radio" name="exampleRadios" id="exampleRadios1" value="option1"></input>
    <label className="form-check-label" for="exampleRadios1">
      Male
    </label>
  </div>
  <div className="form-check">
    <input className="form-check-input" type="radio" name="exampleRadios" id="exampleRadios2" value="option2"></input>
    <label className="form-check-label">
      Female
    </label>
  </div>
      </div>
      <div className="form-group col-md-6">
        <label>Birth Date</label>
        <input type="date" className="form-control"></input>
      </div>
    </div>
    <button type="submit" className="btn btn-primary">Sign Up</button>
  </form>

  </div>

            );
        }
  }

  export default Sign;
```

Figure 6.11: Sign Up Form (b)

Furthermore, we were sure to implement each part by a separate method in which most of our components that requires validation ,and updating changes were designed in which a method for each part separately handles its functionality. As an example, the validation method along with the joi library handles the validation part, the handleChange() method shown above is responsible for updating the values of the user's account depending on any changes that happens to the values ,and so on.

Of course there are several other components that were implemented in the application but we thought the above code snippets is more than enough to visualize the image that we want to create to whomever is interested in such applications while reading this book ,and we didn’t find huge differences between the discussed components ,and the rest of the application components that needed to be pointed out.

In the upcoming we will provide some other snippets of the written code beginning by the CSS (or cascaded style sheet) written code for adding style to the website followed by the flask , python code used to link the previous work with the deep learning technology which is the base of our web application. Figures 6.12 and 6.13 shows CSS written code.

signform.jsx ● # App.css ×

src > # App.css > .button:disabled

```css
.form-sign
{
  border-style: outset;
  border-radius: 10px;
  width: 100%;
  text-align: center;
  padding: 20px;
  font-family: serif;
  font-weight: bold;
  margin-top: 8%;
  max-width: 700px;
  position: absolute;
  left: 30%;
  background-color: teal;
  color: aliceblue;
}

.form-login
{
  border-style: outset;
  border-radius: 10px;
  width: 100%;
  text-align: center;
  font-family: serif;
  font-weight: bold;
  margin-top: 12%;
  padding: 15px;
  max-width: 450px;
  position: absolute;
  left: 35%;
  background-color: teal;
  color: aliceblue;
}

.division
{
  border-style: outset;
  border-radius: 10px;
  margin-top: 50px;
  width: 35%;
  text-align: center;
  padding: 15px;
  font-family: serif;

}

.Slider
{
  border-style: hidden;
  width: 100%;
  height: 50%;
  padding-top: 200px;
  background-image: url('./images/13966.jpg');

```

signform.jsx ● # App.css ×

src > # App.css > .form-login

```css
}

.item
{
  text-align: center;
  font-style: serif;
  color: black;
  font-size: x-large;
}

.divisionStyle
{
  border-style: hidden;
  height: 900px;
  background-color: gainsboro;
}

.sidebar
{
  height: 100%;
  width: fit-content;
  position: fixed;
  z-index: 1;
  top: 0;
  left: 0;
  background-color: dodgerblue;
  overflow-x: hidden;
  padding-top: 60px;
  transition: 0.5s;
}

.diag
{
  right: 10px;
  margin-top: 55px;
  border-style: solid;
  width: fit-content;
}

.help
{
  margin-top: 50px;
  border-style: hidden;
  text-align: center;
  font-style: serif;
  font-size: x-large;
  font-weight: bold;
  padding: 20px;
}

.paragraph
{
  font-style: serif;
```

Figure 6.12 and Figure 6.13 (App.css)

6.4. Deep Learning Models

To be represented as the final part of our article, it is the time now to talk about the coding link between our models ,and the web application itself. As mentioned above, flask web framework was used to link our work together ,and by link we mean combining the models python code with the react framework code so that they can communicate with each other reaching total synchronization for the whole application. In the next ,and last figure , Figure 6.14, we will provide the flask python code for the models.

Figure 6.14: App.py

```
@app.route("/upload-image", methods=["GET", "POST"])
def upload_image():

    if request.method == "POST":

        if request.files:

            image = request.files["image"]
            filepath=os.path.join(app.config["IMAGE_UPLOADS"], image.filename)
            print(filepath)
            image.save(filepath)
            print("Image saved")
            print(filepath)
            result=predictbone(filepath)
            print(result)
            return render_template("index.html",result=result)

    return render_template("index.html")

if __name__ == '__main__':
    app.run(host="0.0.0.0",port=5000,threaded=False)
```

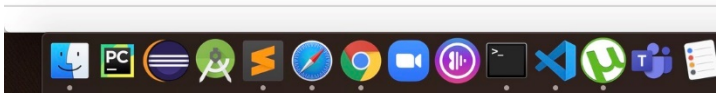


Figure 6.15: App.py

By reading the above code, it is simple to understand its purpose but to make sure that everything is clear ,and understandable ,the following code is composed of the following: first, some basic libraries or files that is required is imported such as sys ,and math etc. Secondly, some flask utilities are prepared ,and the input data image is acquired to get prepared for the image processing by the models. Each model is represented by a separate method , for example a method for skin disease prediction which is predictskin() ,predictbone() (for bone problems) ,and predictpneumonia().

At the beginning, a print message is popped to indicate that the model is loaded at port 5000 which is the default port for flask back-end host ,and then a preparepic() method is called to prepare the input image for processing as mentioned above. A method called upload_image() is implemented to allow the user to input medical images which is linked to our client page that we discussed earlier. Finally, the models prediction is presented to the user as the output of the diagnosis.

Chapter 7. Conclusion

We were successfully able to implement our work effectively, and linking our whole project together, bringing up the medical web application that we aimed to implement. However, there is room for improving its functionality and services in the near future by developing it more.

Chapter 8. References

8.1. References of Chapter 2

[1] ”MURA paper: https://arxiv.org/pdf/1712.06957.pdf”

[2] ”Article about Transfer Learning: “https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/”

[3] “Paper of MobileNet: https://arxiv.org/pdf/1704.04861.pdf ”

[4]“Tutorial on using Global Average Pooling: https://principlesofdeeplearning.com/index.php/a-tutorial-on-global-average-pooling/ ”

[5] ”Paper of Adam optimizer: https://arxiv.org/abs/1412.6980v9 ”

[6]”Tutorial of Loss function: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks”

[7]”Tutorial of Callbacks: https://www.kdnuggets.com/2019/08/keras-callbacks-explained-three-minutes.html”

8.2 References of Chapter 4

[1] “https://arxiv.org/abs/1803.10417”

[2] “https://en.wikipedia.org/wiki/Transfer_learning”

[3]“https://medium.com/starschema-blog/transfer-learning-the-dos-and-donts-165729d66625”

[4] “https://www.tensorflow.org/tutorials/images/transfer_learning”

[5] “https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html”

[6] “https://d2l.ai/chapter_computer-vision/fine-tuning.html”

[7] “https://arxiv.org/abs/1602.07261”

[8] “http://yeephycho.github.io/blog_img/Inception_ResNet_v2_raw.jpg”

[9]“https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202”

[10] “https://ai.googleblog.com/2016/08/improving-inception-and-image.html”

8.3. References of Chapter 5

[1] “https://www.who.int/news-room/fact-sheets/detail/pneumonia”

[2] “https://www.everydayhealth.com/pneumonia/guide/”

[3] “https://www.nhs.uk/conditions/pneumonia/”

[4] “https://www.physio-pedia.com/Pneumonia”

[5] “https://my.clevelandclinic.org/health/diseases/4471-pneumonia”

[6] “https://www.webmd.com/lung/covid-and-pneumonia#1”

[7]“https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/”