**CLASSIFICATION OF MALIGNANT OR BENIGN TYPE OF SKIN CANCER USING DEEP LEARNING APPROACH**

A blue and red logo

Description automatically generated with low confidence

***SUMIT RAJ***

***SUDIPTA JAHARLAL GIRI***

***SHADMAN TABRAIZ***

***PUJA DEY***

**DEPARTMENT OF COMPUTER SCIENCE AND  ENGINEERING**

May, 2022

**CLASSIFICATION OF MALIGNANT OR BENIGN TYPE OF SKIN CANCER USING DEEP LEARNING APPROACH**

***Report submitted to***

***Department of Computer Science and Engineering***

***Dr. B. C. Roy Engineering College, Durgapur, WB***

***for the partial fulfilment of the requirement to award the degree***

***of***

**Bachelor of Technology**

**in**

**Computer Science and Engineering**

***by***

***Sumit Raj*** *(12000118017)*

***Sudipta Jaharlal Giri*** *(12000118019)*

***Shadman Tabraiz*** *(12000118042)*

***Puja Dey****(12000118071)*

A blue and red logo

Description automatically generated with low confidence

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**DR. B.C. ROY ENGINEERING COLLEGE, DURGAPUR, WB**

A blue and red logo

Description automatically generated with low confidence

**DECLARATION**

We, the undersigned, hereby declare that our B.Tech. final year Project entitled “**Classification of Malignant or Benign type of Skin Cancer using Deep Learning approach**” is original and is our own contribution. To the best of our knowledge, the work has not been submitted to any other Institute for the award of any degree or diploma. We declare that we have not indulged in any form of plagiarism to carry out this project and/or writing this project report. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing in the text of the report and giving their details in the references. Finally, we undertake the total responsibility of this work at any stage here after.

Signature of the Students

--------------------------------

***Sumit Raj (****12000118017)*

--------------------------------

***Sudipta Jaharlal Giri (****12000118019)*

--------------------------------

***Shadman Tabraiz (****12000118042.)*

--------------------------------

***Puja Dey (****12000118071)*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Dr. B.C. ROY ENGINEERING COLLEGE, DURGAPUR, WB.**

A blue and red logo

Description automatically generated with low confidence

**RECOMMENDATION**

This is to recommend that the work undertaken in this report entitled, “**Classification of Malignant or Benign type of Skin Cancer using Deep Learning approach**” has been carried out by “**Sumit Raj, Sudipta Jaharlal Giri, Shadman Tabraiz, Puja Dey**” under my/our supervision and guidance during the academic year 2021-22. This may be accepted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prof. Sabbir Reza Tarafdar**

Assistant Professor

Department of CSE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Prof.** **Dr. Chandan Koner**

Head of the CSE Department

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Dr. B.C. ROY ENGINEERING COLLEGE, DURGAPUR, WB.**

A blue and red logo

Description automatically generated with low confidence

**APPROVAL**

This is to certify that, **Sumit Raj**, **Sudipta Jaharlal Giri**, **Shadman Tabraiz** and **Puja Dey** students in the Department of Computer Science & Engineering, worked on the project entitled " **Classification of Malignant or Benign type of Skin Cancer using Deep Learning approach** ".

I hereby recommend that the report prepared by them may be accepted in partial fulfilment of the requirement of the Degree of Bachelors of Technology in the Department of Computer Science and Engineering, Dr. B.C. Roy Engineering College, Durgapur.

**Board of Examiners**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Prof. Sabbir Reza Tarafdar**

**(Supervisor)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr. Chandan Koner**

**(HOD, CSE)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Co-ordinator**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Dr. B.C. ROY ENGINEERING COLLEGE, DURGAPUR, WB.**

A blue and red logo

Description automatically generated with low confidence

**ACKNOWLEDGEMENT**

With great pleasure and thanks, we express our gratitude to our esteemed instructor **Prof. Sabbir** **Reza Tarafdar** for his inspiring direction, constructive criticism, and important suggestions during the project work.

We would like to express our heartfelt gratitude to Prof. **Dr. Chandan Koner**, Head of Department, for encouraging and allowing us to present the project " **Classification of Malignant or Benign type of Skin Cancer using Deep Learning approach** " at our department premises for partial fulfilment of the requirements leading to the award of the B.Tech, Degree.

Furthermore, we are grateful to our parents, friends, and well-wishers for their direct or indirect assistance in conducting our study and continuing our work.

Also, we would like to recognise the support we obtained from a variety of online sites, publications, and articles, particularly the efforts of our fellow scholars.

***Sumit Raj***

***Sudipta J Giri***

***Shadman Tabraiz***

***Puja Dey***

**ABSTRACT**

Deep learning has had a huge influence on several domains of technology in recent years. Computer vision, or the capacity for computers to perceive pictures and videos on their own, is one of the trendiest subjects in this business. Self-driving cars, biometrics, and facial recognition all rely on computer vision to work. At the core of computer vision is image processing.

Image processing can be defined as the technical analysis of an image using complex algorithms. Here, the image is used as the input, where the useful information returns as the output. The implementation of image processing techniques has had a massive impact on many tech organizations, for example: the digital image can be made available in any desired format (improved image, X-Ray, photo negative, etc.)

Biomedical image processing is a very broad field; it covers biomedical signal gathering, image forming, picture processing, and image display to medical diagnosis based on features extracted from images.

Skin cancer is one of the most prevalent kinds of human cancer in the medical field. It is often diagnosed visually, beginning with a clinical screening and perhaps followed by a dermoscopic study, a biopsy, and histological testing. Machine learning is constantly being utilized to estimate the accuracy of diagnosing various medical conditions more effectively. Many innovative ways have been devised to expedite the operation while maintaining the best percentage of accuracy.

In this thesis study, we investigated several architectures for more successfully detecting skin cancer with image processing, which is a subset of the deep learning concept under machine learning. The dataset contains almost 3000+ images of the patients having skin diseases classified into two classes, malignant and benign.

**Keywords**: Machine Learning, Skin cancer, Image Processing, Convolutional Neural Network, Malignant, Benign, ResNet50, VGG, Xception, CNN architecture.

**Nomenclature**

The next list describes several symbols & abbreviations that will be later used within the body of the document

AVG - Average

CNN - Convolutional Neural Network

DCNN - Deep Convolutional Neural Network

DNN - Deep Neural Network

NN - Neural Network

ReLU - Rectified Linear Unit

ResNet - Residual Network

RGB - Red Green and Blue

SF - Sigmoid Function

VGG - Visual Geometry Group

Xception Extreme Inception

**List of Figures**

* Figure 1: The Architecture of CNN 11
* Figure 2: Input Layer 12
* Figure 3: Dense Layer 13
* Figure 4: Convolution Layer 14
* Figure 5: Pooling Operation 15
* Figure 6: Normalization Layer 16
* Figure 7: Batch Normalization 16
* Figure 8: Fully Connected Layer . 17
* Figure 9: Output Layer 17
* Figure 10: Proposed System 18
* Figure 11: Example of used data set 19
* Figure 12: The architecture of CNN model 23
* Figure 13: Build Architecture of CNN model 24
* Figure 14:CNN Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs) 25
* Figure 15: Output of original test data for CNN dataset 26
* Figure 16: Output of predicted data setwith CNN architecture 26
* Figure17: ResNet50 Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs) 30
* Figure 18: Output of original test data 31
* Figure 19: ResNet50 prediction on the same test data 31
* Figure 20: Vgg16 32
* Figure 21: VGG16 Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs) 33
* Figure 22: Output of original test data 34
* Figure 23: VGG16 prediction on the same test data 35
* Figure 24: Original Xception 36
* Figure 25: Modified Xception 36
* Figure 26: Xception Model Accuracy(accuracy vs epochs) Model Loss (loss vs epochs) 37
* Figure 27: Output of original test data 38
* Figure  28: Xception prediction on the same test data 39

**List of Tables**

* Table 1: Result after every 5 epochs in CNN model 24
* Table 2: Outputs per 5 epochs in ResNet50 Architecture 29
* Table 3: Outputs per 5 epochs in VGG16 Architecture 34
* Table 4: Outputs per 5 epochs in Xception Architecture 38
* Table 5: comparative analysis of Accuracy and Loss of different CNN architectures that we have applied. 39

**CONTENTS**

Title Page i

Declaration iii

Recommendation by the Supervisor iv

Approval v

Acknowledgement vi

Abstract … 1

Nomenclature… 2

List of Figures 3

List of Tables 4

Table of Contents … 5

Chapter 1 Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7

* 1. Machine Learning
  2. Deep learning
  3. Medical Image Processing
  4. Biomedical Image Processing
  5. Skin Cancer

|  |
| --- |
| Chapter 2 Literature Survey \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9 |
|  |

|  |
| --- |
| Chapter 3 Theoretical Study \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11  3.1 CNN Background Study  3.1.1 Input Layer  3.1.2 Dense Layer  3.1.3 Convolution Layer  3.1.4 ReLU  3.1.5 Pooling Layer  3.1.6 Normalization Layer  3.1.7 Fully Connected Layer  3.1.8 Output Layer  3.2 Flowchart  3.3 Dataset Description  3.4 Pre-processing Image  3.5 Library and Parameter required |
|  |

Chapter 4 Implementation and Result\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_23

4.1 CNN

4.2 Resnet50

4.3 VGG16

4.4 Xception

4.5 Comparison of CNN Architecture

Chapter 5 Conclusion and Future Scope\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_41

* Bibliography

**CHAPTER 1 INTRODUCTION**

**Objective**

Converting the digital images into a dataset for building training and testing models. Applying various Convolutional Neural Network (CNN) models to predict the types of a Skin Cancer (Malignant and Benign). Analyzing reasons why performance varies between models having tested with the same dataset.

**1.1 Machine Learning**

Machine learning is the use of algorithms to examine, learn, and draw conclusions or predictions about data. Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

**1.2 Deep Learning**

Deep learning is a subfield of machine learning that operates on the neural network notion of the brain. Artificial neural networks are deep learning models that are based on a collection of linked units known as artificial neurons or neurons, with each connection between neurons capable of transmitting a signal from one neuron to another. CNN is a deep learning variant that creates a model that takes an image as input and analyses it to extract features from it by identifying patterns in it.

**1.3 Medical Image Processing**

Medical imaging is the process of producing visible images for the structures of the inner body for scientific and medical cause such as medical studies and treatments.

The image processing in the medical field creates data bank of the functions of all organs in order to make anomalies recognition much easier.

**1.4 Biomedical Image Processing**

Biomedical imaging concentrates on the capture of images for both diagnostic and therapeutic purposes. Biomedical imaging technologies utilize either x-rays (CT scans), sound (ultrasound), magnetism (MRI), radioactive pharmaceuticals (nuclear medicine: SPECT, PET) or light (endoscopy, OCT) to assess the current condition of an organ or tissue and can monitor a patient over time over time for diagnostic and treatment evaluation.

Biomedical image processing is similar in concept to biomedical signal processing in multiple dimensions. It includes the analysis, enhancement and display of images captured via x-ray, ultrasound, MRI, nuclear medicine and optical imaging technologies.

**1.5 Skin Cancer**

Skin cancer is the world's nineteenth most frequent cancer. Over the last year, there were roughly 300,000 new cases. Non-melanoma skin cancer is the fifth most frequent malignancy in both men and women, with over 1 million cases in 2018.

It is usually treated visually, beginning with a medical exam and potentially followed by dermoscopic study, a biopsy, and histopathology. CNN use advanced algorithms to learn characteristics from massive amounts of health care data and then applies the resulting insights to aid in diagnosis. In the previous ten years, the use of CNN has continued to rise.

In this thesis study, we investigated several architectures for more successfully detecting skin cancer with image processing, which is a subset of the deep learning concept under machine learning. The dataset contains almost 3000+ images of the patients having skin diseases classified into two classes, malignant and benign.

**CHAPTER 2**

**LITERATURE REVIEW**

1. In today's advanced world of research and technology, skin cancer is regarded as one of the most venturesome cancers to be discovered in the human body. Detecting skin cancer at an early stage is critical and essential. With the advancement of medical science and technology, researchers and scholars have developed several efficient methods for detecting skin cancer. We examined a significant number of works and related papers to become acquainted with the current systems and works linked to our objective. Current methods utilise manual, semi-automatic, or completely automatic boundary identification algorithms to subdivide skin lesions in the input picture. Shape, colour, texture, and brightness are the features employed in various articles to accomplish skin lesion dissection.
2. Tanzila Saba et al. proposed with DCNN an unique automated system for detecting and recognizing skin lesions. They took three steps. The first step was to boost the contrast of the photos with first local laplacian filtering and HSV colour conversion. The second stage was to extract the lesion border using the colour CNN methodology's XOR function. The next stage was to extract the features using InceptionV3 and transfer learning in order to feature fusion utilising the hamming distance approach. They presented a feature selection strategy based on entropy control for the gathering of the most discriminating kinds. Different datasets were used to vary their suggested system. Using their suggested approach, they attained great accuracy.
3. J. Kawahara, A. Bentaieb, G. Hamarneh proposed (Published 13 April 2016 Computer Science 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)) Diagnosing an unknown skin lesion is the first step to determine appropriate treatment. They demonstrated that a linear classifier, trained on features extracted from a convolutional neural network pretrained on natural images, distinguishes among up to ten skin lesions with a higher accuracy than previously published state-of-the-art results on the same dataset. Further, in contrast to competing works, our approach requires no lesion segmentations nor complex preprocessing. They gained consistent additional improvements to accuracy using a per image normalization, a fully convolutional network to extract multi-scale features, and by pooling over an augmented feature space. Compared to state-of-the-art, our proposed approach achieves a favourable accuracy of 85.8% over 5-classes (compared to 75.1%) with noticeable improvements in accuracy for underrepresented classes (e.g., 60% compared to 15.6%). Over the entire 10-class dataset of 1300 images captured from a standard (non-dermoscopic) camera, our method achieves an accuracy of 81.8% outperforming the 67% accuracy previously reported.
4. P. Dwivedi, Understanding and coding a resnet in keras, Mar. 2019. [Online].Available: https ://towardsdatascience .com/understanding- and- coding - a-resnet-in-keras-446d784d33/.ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

**CHAPTER 3**

**THEORETICAL STUDY**

**3.1 CNN Background Study**

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

When we perceive a picture, our brain analyses a massive quantity of information. Each neuron has its own receptive field and is linked to other neurons so that it covers the full visual field. In the biological vision system, each neuron responds to inputs only in the confined region of the visual field known as the receptive field; similarly, each neuron in a CNN analyses data only in its receptive field. The layers are designed such that simpler patterns (lines, curves, etc.) are detected first, followed by more complicated patterns (faces, objects, etc.). One may give computers sight by utilizing a CNN.

CNN uses a filter to remove critical information. Combining a feature map with a convolution filter results in a modified feature map. To extract the most usable information from a certain activity, the kernels are modified based on learned parameters. Convolutional networks discover the finest components of a task intuitively. One input layer, one or more hidden layers, and one output layer comprise a Neural Network. These layers can be dense, convolutional, pooling, normalizing, rectified linear unit layers, completely linked layers, and so forth. Convolutional Neural Network layers must have at least one convolutional layer.



Figure 1: The Architecture of CNN

**3.1.1 Input Layer**

Input Layer in CNN works based on only image data. These images have three dimensions, [width x height x depth] which is a matrix of pixel values. For example, for an input value of (width=32, height=32, depth=3) or [32x32x3] where depth stands for Red-green, and blue channels or RGB channels. Also, the input layer has to be divisible many times by 2. To work with an input layer, at first, the three-dimensional matrix has to be reshaped into a single column. For example, if there is an image of dimension 32x32=1024, it needs to be converted into 1024x1 before passing it into the input. For `x' training examples the dimension of input will be (1024,x)



Figure 2: Input Layer

**3.1.2 Dense Layer**

Dense implements the operation: output = activation (dot (input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, the kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use bias is True). These are all attributes of Dense.

In a neural network, the dense layer is the non-linear layer. These layers use the same formulae as linear layers, but the difference is that dense layers send the final output via a non-linear function known as the Activation function. One of the remarkable qualities of dense layers is their ability to simulate any mathematical function. Although there are certain limits, for example, the output will always be the same for every given input vector. The dense layer can neither provide various replies on the same input nor identify time recurrence.

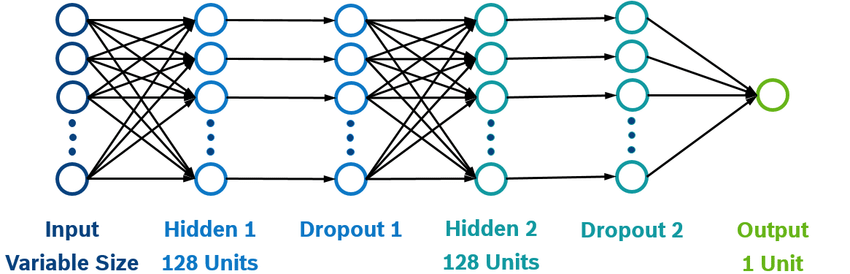


Figure 3: Dense Layer

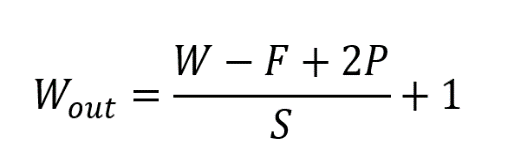
**3.1.3 Convolution Layer**

The convolution layer is the core building block of the CNN. It carries the main portion of the network’s computational load.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride.

If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula number (1) :

 ………. (1)

This will yield an output volume of size Wout x Wout x Dout.

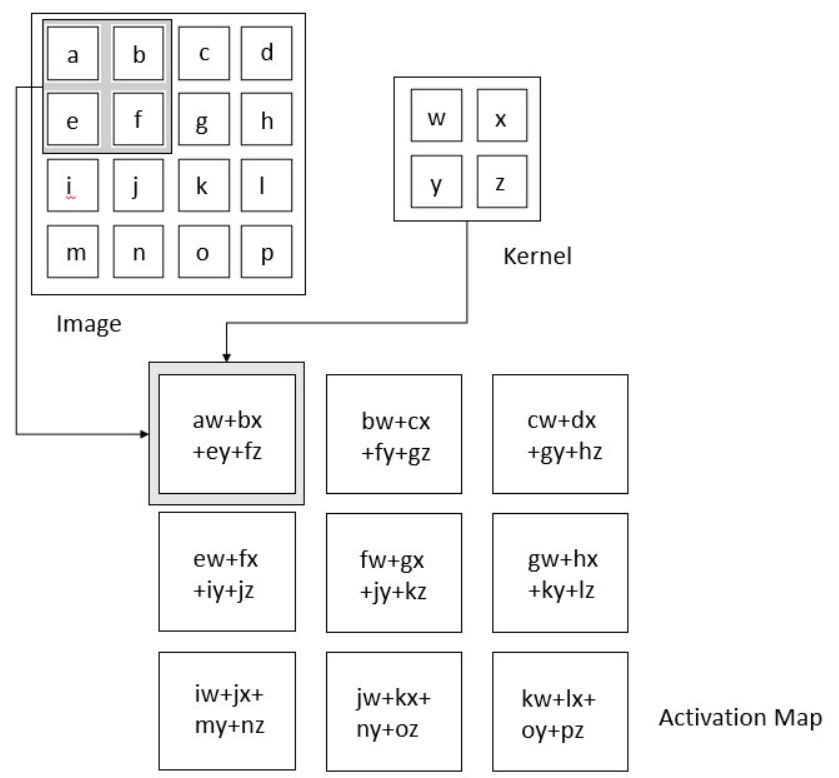
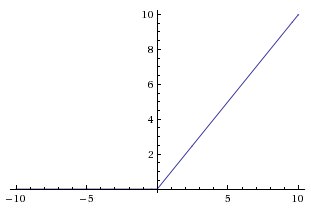


Figure 4: Convolution Layer

**3.1.4 ReLU**

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as f(x)=max(0,x) .Graphically it looks like this-



**3.1.5 Pooling Layer**

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

There are several pooling functions such as the average of the rectangular neighborhood, the L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

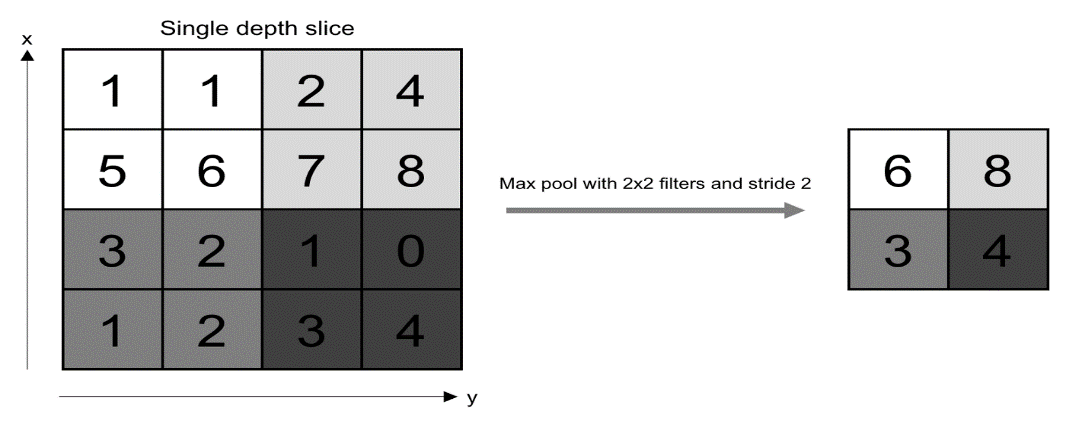
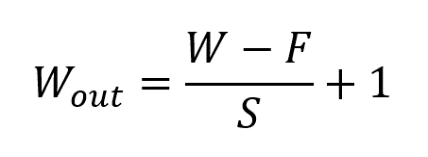


Figure 5: Pooling Operation

If we have an activation map of size W x W x D, a pooling kernel of spatial size F, and stride S, then the size of output volume can be determined by the following formula:

………. (2)

In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

**3.1.6 Normalization Layer**

The normalization layer basically normalizes the activation outputs of the previous layer which means it puts in a transformation that upholds the mean activation close to zero and the activation standard derivation close to 1. Batch normalization is mostly used here although there are many other alternatives like weight normalization, layer normalization, instance normalization, group normalization, spectral normalization, etc.

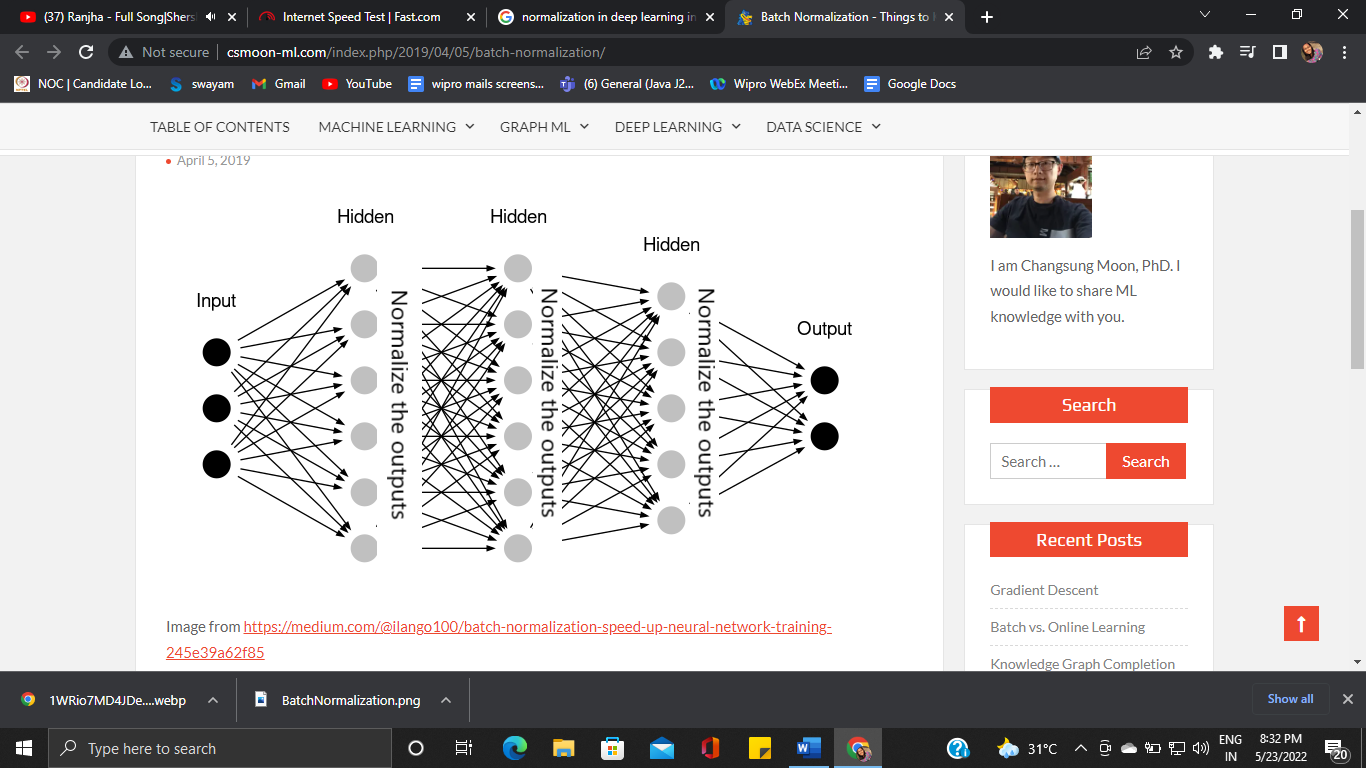


Figure 6: Normalization Layer

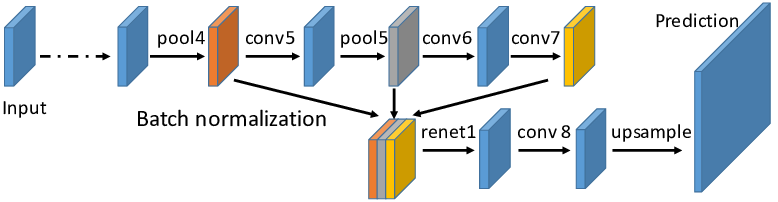


Figure 7: Batch Normalization

**3.1.7 Fully Connected Layer**

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect.

The FC layer helps to map the representation between the input and the output.

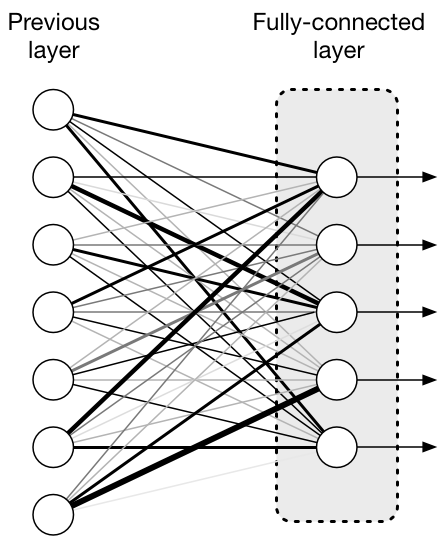


Figure 8 : Fully Connected Layer

**3.1.8 Output Layer**

The output layer in an artificial neural network is the last layer of neurons that produces given outputs for the program. Though they are made much like other artificial neurons in the neural network, output layer neurons may be built or observed in a different way, given that they are the last “actor” nodes on the network.

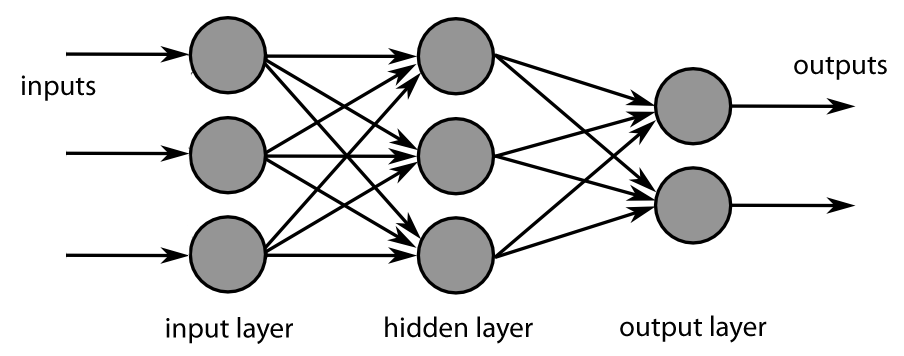
****

Figure 9: Output Layer

**3.2 Flowchart**

Collect Data

Preprocessing

Split Data

Train Data

Test Data

Split Data

Train Data

Validation Data

Model Processing (CNN, VGG16, ResNet50, Xception)

Accuracy Checking

Figure 10: Proposed System

**3.3 Dataset Description**

The dataset we used for our thesis is a secondary dataset. We collected it from kaggle. The dataset is known as “Skin Cancer: Malignant vs Benign". The dataset contains a balanced dataset of images of benign skin moles and malignant skin moles. It consists of total 4000+ pictures. The dataset is divided into two main folders named \test"(1000+ pictures) and \train"(1000+ pictures) with each in the folder having two more sub folders named \malignant" and \benign". In each of the folders above have the images with a unique id of each image.

Below is the example of used data set:

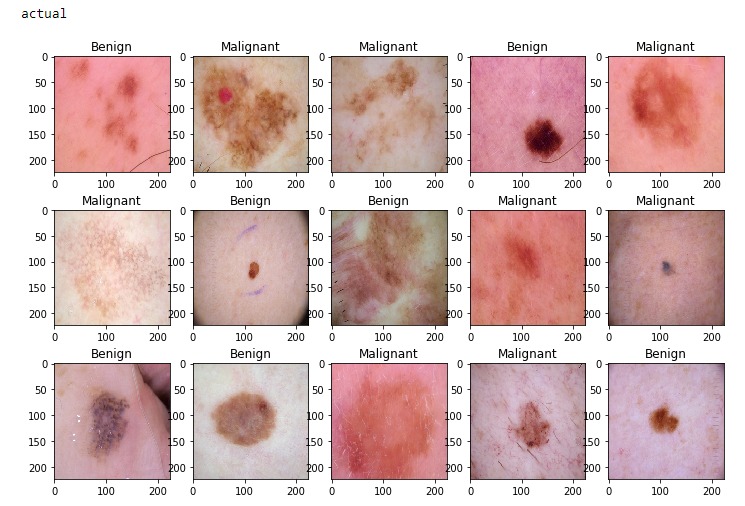


Figure 11: Example of used data set.

**3.4 Pre-processing Image**

* All of the photos in our collection have the same resolution of 224x224 pixels.
* We first read all of the photos as a list of arrays, and then transformed them to RGB images.
* After that, we changed the array list to Numpy Arrays with the data type "uint8" for each of them.
* We assigned a 0 to benign images and a 1 to malignant images.

* Furthermore, we concatenated all benign and malignant photos from the test and train folders into two distinct arrays known as the test and train set.
* As part of the normalization procedure, we divided the train and test arrays by 255 such that each value remained within the same measurement range of 0 to 1, with higher values being closer to 1 would represent malignant and closer to 0 would represent benign.
* The input shape of the input images from the dataset is now 224X224X3

**3.5 Library and Parameters required**

A general description of the Parameters and Library that we used to compile the models is given below:

**TFLearn** – Deep learning library featuring a higher-level API for TensorFlow used to create layers of our CNN

**Keras** - Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

**numpy** – To process the image matrices

**matplotlib** – To display the result of our predictive outcome.

**open-cv** – To process the image like converting them to grayscale and etc.

**os** – To access the file system to read the image from the train and test directory from our machines

**random** – To shuffle the data to overcome the biasing • matplotlib – To display the result of our predictive outcome.

**tensorflow** – Just to use the tensorboard to compare the loss and adam curve our result data or obtained log.

**Seaborn**- statistical data visualization. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics

**glob** - glob (short for global) is used to return all file paths that match a specific pattern. We can use glob to search for a specific file pattern, or perhaps more usefully, search for files where the filename matches a certain pattern by using wildcard characters.

**itertools** - Itertools is a Python module that is part of the Python 3 standard libraries. It lets us perform memory and computation efficient tasks on iterators.

**Activation of Fully Connected Layer**

Activating the fully connected layers may result in overfitting of the whole model, which implies that it will perform well for known examples but poorly for unexpected ones. Only when the fully connected network is false may the picture size be specified. Otherwise, it will be transformed to the fixed size by the models.

**Weight Selection**

We trained the model from scratch so that it could learn on its own, and then compared each model to choose the best model for this project. Thus, weight was set to null.

**Pooling Layer selection**

There are many types of pooling layers available but among them max pooling and average pooling are the most common and most widely used.

If we need to discover patterns throughout the entire picture, average pooling is a preferable alternative since it takes the average value of the pixel from the dened matrix. However, because we are identifying diseases from a patient's body section that is only a subset of the entire picture, we are applying max pooling.

**Dropout Selection**

Dropout is a regularisation technique used in CNN to avoid the dataset from   overflowing during the training phase. In our model, the dropout value was set at 0.25. That is, it will disregard 25% of the neurons at random when training.

**Activation Function Selection**

In our proposed system we only used two types of activation functions, the Sigmoid Function and the Relu Function.

**Sigmoid Function**

In a sigmoid function, the more negative the input is, the closer to 0 the output will be. In a same way, the more positive the number is, the closer to 1 the output will be. The range will vary from 0 to 1.

**SoftMax Function**

SoftMax activation function is a type of sigmoid function which can be used in classification problems. The *SoftMax function* transforms the outputs of the each class into between 0 and 1.

**Relu Function**

In a Relu function, what it does is, it converts the value to a scale started from 0 to positive innity. Means, if the input value is less than or equal to 0, then the output will be 0. Furthermore, if the input is greater than 0, the output will be the value itself.

**Flatten Layer Selection**

After employing the pooling layer, we must apply an flatten layer to attenuate the entire network. Attening converts the whole pooling feature map matrix into a single column. This is then forwarded to the neural network for further processing.

**Adam**

Adaptive Moment Estimation (Adam) is a method for calculating adaptive learning rates for each parameter. Adam additionally keeps track of the exponentially decaying average of previous gradients. Adam outperforms other adaptive learning-method algorithms because it converges very quickly and the Model’s learning speed is fast and efficient, and it also solves every problem found in other optimization methods, such as learning rate loss or at a very low value, lower convergence, or extreme variance in parameter updates. All of these strategies result in the fluctuation of the loss function.

**CHAPTER 4**

**IMPLEMENTATION AND RESULTS**

Each part has demonstrated the test accuracy after each 5 epochs till 50 epochs with their loss, accuracy, val-loss, and val-accuracy. Then, an extensive graphical overview (Accuracy vs Epochs curve and Loss vs Epochs curve) of each model of the test result was presented, along with some sample test images from our dataset, and the predicted results of our used architectures were compared to demonstrate an extensive comparative analysis.

**4.1 CNN**

Below figure describes the CNN model

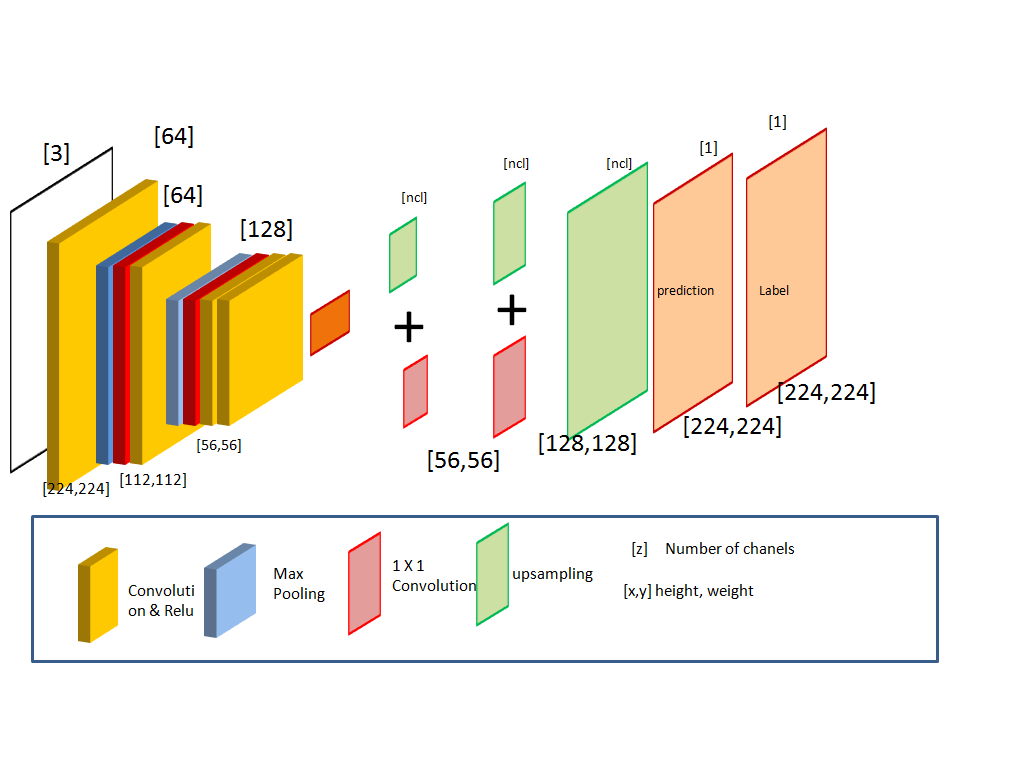
****

Figure 12: The architecture of CNN model

**The summary of CNN model:**

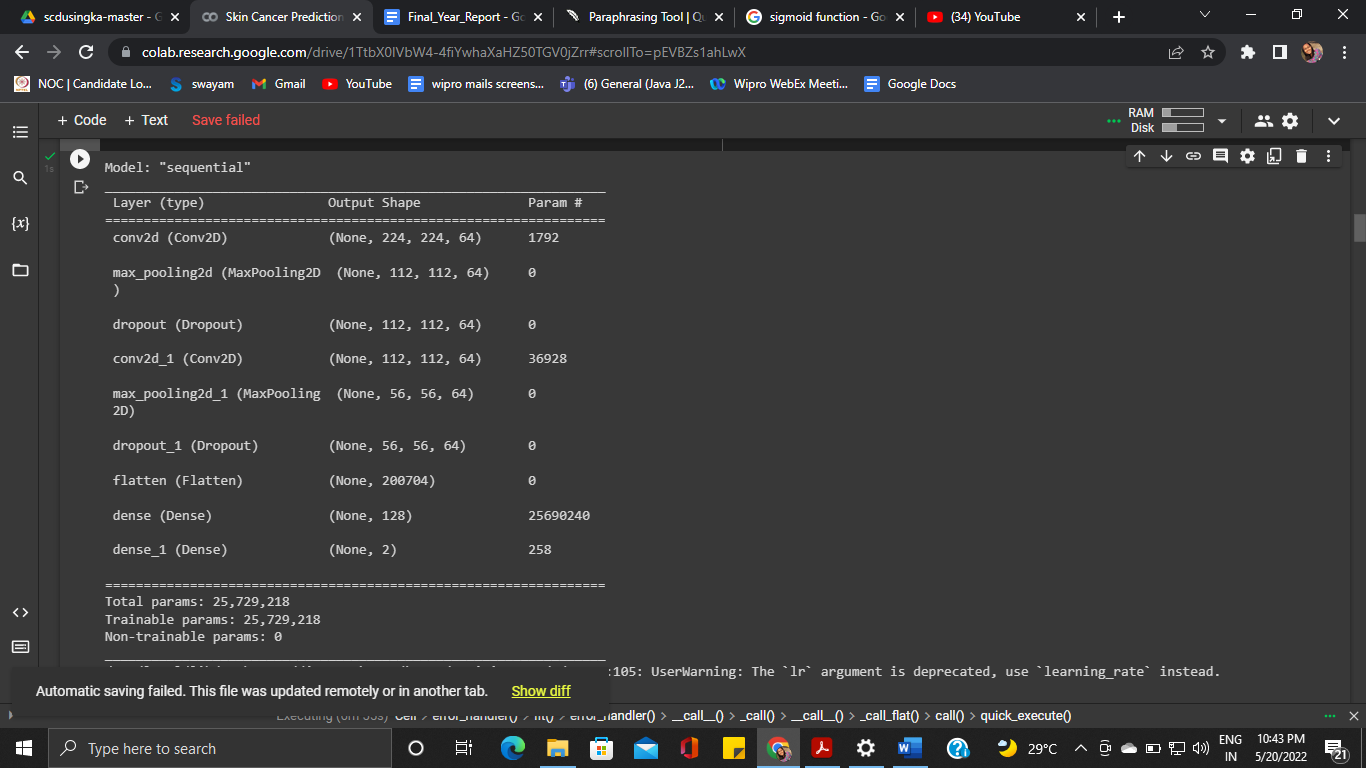
****

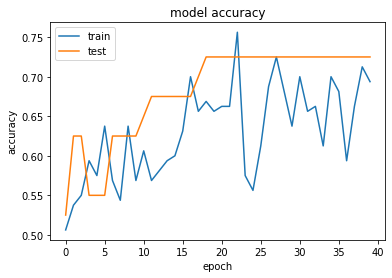
Figure 13: Build Architecture of CNN model

**Result Calculated after every 5 epochs:**

Table 1: Result after every 5 epochs in CNN model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of epochs** | **loss** | **accuracy** | **val\_loss** | **val\_accuracy** |
| **1** | 1.0668 | 0.5063 | 0.6987 | 0.5250 |
| **5** | 0.9449 | 0.5750 | 0.6829 | 0.5500 |
| **10** | 0.8588 | 0.5688 | 0.6245 | 0.6250 |
| **15** | 0.8199 | 0.6000 | 0.6057 | 0.6750 |
| **20** | 0.7857 | 0.6562 | 0.5926 | 0.7250 |
| **25** | 0.8511 | 0.5562 | 0.5858 | 0.7250 |
| **30** | 0.8243 | 0.6375 | 0.5866 | 0.7050 |
| **35** | 0.7406 | 0.7000 | 0.5877 | 0.7250 |
| **40** | 0.7645 | 0.6938 | 0.5864 | 0.7250 |

**The graph for CNN Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs)**

****

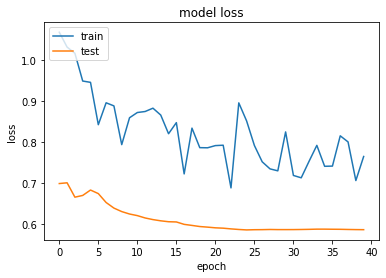
****

Figure 14:CNN Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs)

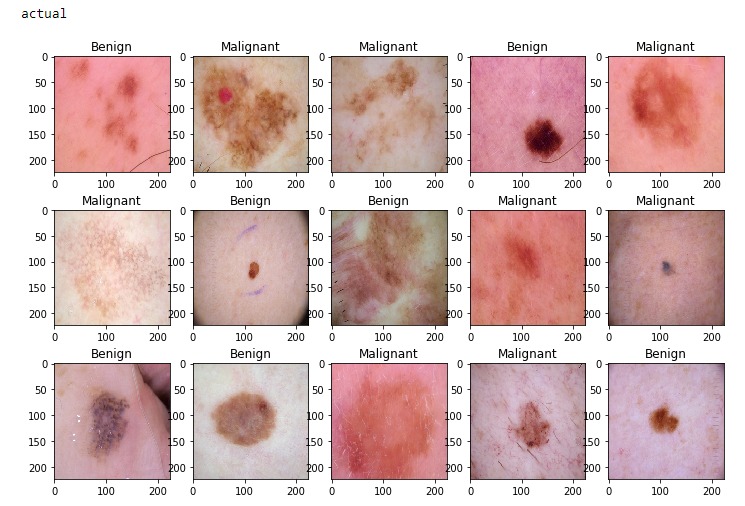
****

Figure 15: Output of original test datafor CNN dataset

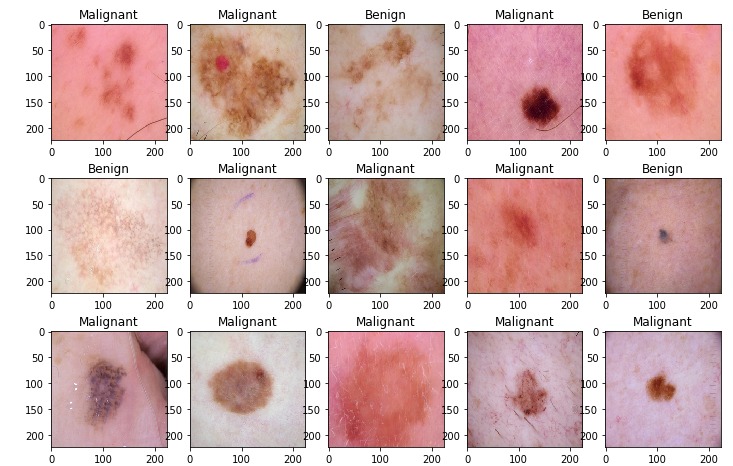
****

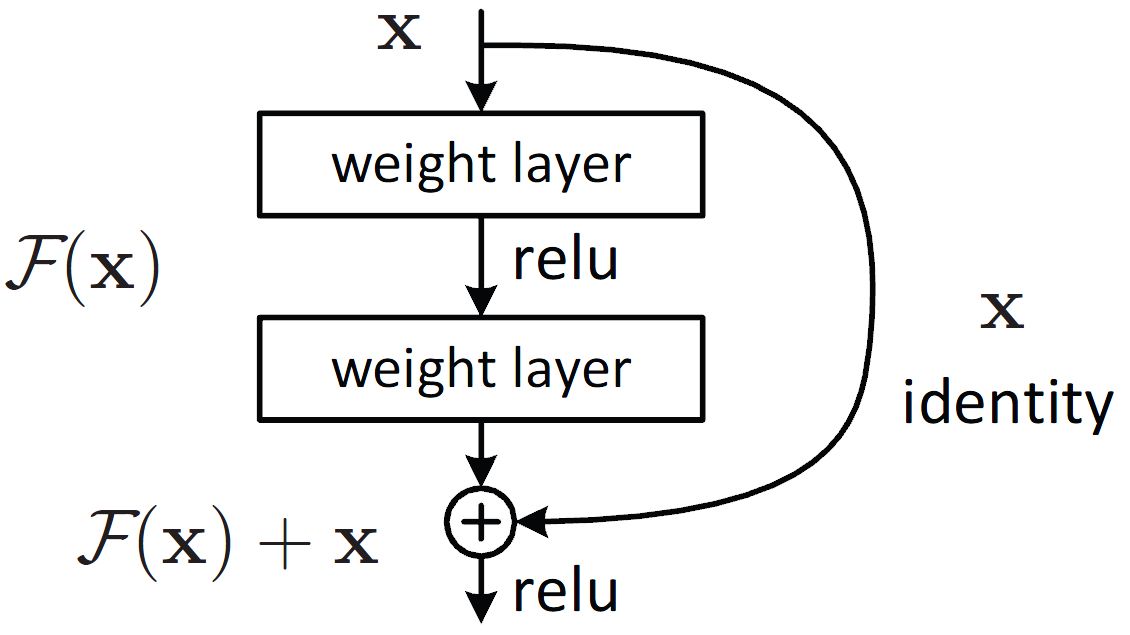
Figure 16: Output of predicted data setwith CNN architecture

**4.2 Resnet50**

**ResNet Concept**

ResNet stands for Residual Network. It is an innovative neural network that was first introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 computer vision research paper titled ‘Deep Residual Learning for Image Recognition’.

It is almost related to other neural networks that have convolution, pooling, activation and fully connected layers stacked one over another. There are some identity connections between the layers.The network learns many features at the end of its layers.



In the above figure, the very first thing we can notice is that there is a direct connection that skips some layers of the model. This connection is called ’skip connection’ and is the heart of residual blocks. The output is not the same due to this skip connection. Without the skip connection, input ‘X gets multiplied by the weights of the layer followed by adding a bias term.

Then comes the activation function, f() and we get the output as H(x).

H(x)=f (wx + b) or H(x)=f(x) ……… (3)

Now with the introduction of a new skip connection technique, the output is H(x) is changed to

H(x)=f(x)+x ………. (4)

But the dimension of the input may be varying from that of the output which might happen with a convolutional layer or pooling layers. Hence, this problem can be handled with these two approaches:

* Zero is padded with the skip connection to increase its dimensions.
* 1×1 convolutional layers are added to the input to match the dimensions. In such a case, the output is:

H(x)=f(x)+w1.x ………. (5 )

Here an additional parameter w1 is added whereas no additional parameter is added when using the first approach.

These skip connections technique in ResNet solves the problem of vanishing gradient in deep CNNs by allowing alternate shortcut path for the gradient to flow through. Also, the skip connection helps if any layer hurts the performance of architecture, then it will be skipped by regularization.

**ResNet50**

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224x224.

**Applying ResNet50 in our Dataset**

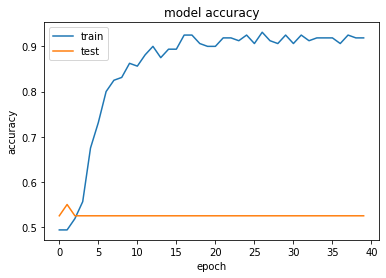
We utilised ResNet50 architecture to test the accuracy of our dataset after training our model with simple CNN. We retained all of the parameters in the CNN model at their previous values. We modified a few settings inside the ResNet50 architecture to meet our requirements.

For example, we set the include-top property to true. Most earlier research in this field have employed a few of CNN architectures, such as ResNet50, with their weight's value set to 'imagenet' since they used a pre-trained model.

However, we set the parameter to 'None' since we intended to train our model from start and then test its accuracy using the ResNet architecture. Another modification we did was to alter the pooling operation from "MaxPooling" to "pooling" and use an avg value (average). This implies that instead of picking the largest value as we did in the MaxPooling operation, it will take the average value of a particular matrix of pixels. We built the model after defining it by using the optimizer 'Adam' and the loss function 'binary-cross entropy.' Finally, we used 20% of the training dataset as a validation dataset using the model, and then we displayed the graph of accuracy vs epoch and loss vs epoch to obtain a visual depiction of our model's accuracy and loss. In our dataset, we achieved 82.424 percent accuracy using the ResNet50 architecture. Finally, we saved the model to a json and a h5 file before deleting it to train our dataset with a new architecture. The table below shows the loss, accuracy, validation loss, and validation accuracy per 5 epochs of ResNet50.It is clearly shown that the accuracy was rising up over time and the loss was degrading.

Table 2: Outputs per 5 epochs in ResNet50 Architecture

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. of Epochs | loss | accuracy | val\_loss | val\_accuracy |
| 1 | 0.9510 | 0.4938 | 0.6928 | 0.5250 |
| 5 | 0.6586 | 0.6750 | 0.6928 | 0.5250 |
| 10 | 0.4599 | 0.8625 | 0.6936 | 0.5250 |
| 15 | 0.3818 | 0.8938 | 0.6944 | 0.5250 |
| 20 | 0.3557 | 0.9000 | 0.6962 | 0.5250 |
| 25 | 0.3235 | 0.9250 | 0.6998 | 0.5250 |
| 30 | 0.3069 | 0.9250 | 0.7041 | 0.5250 |
| 35 | 0.3231 | 0.9187 | 0.7090 | 0.5250 |
| 40 | 0.3064 | 0.9187 | 0.7175 | 0.5250 |



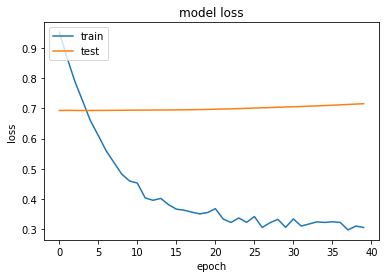


Figure17: ResNet50 Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs)

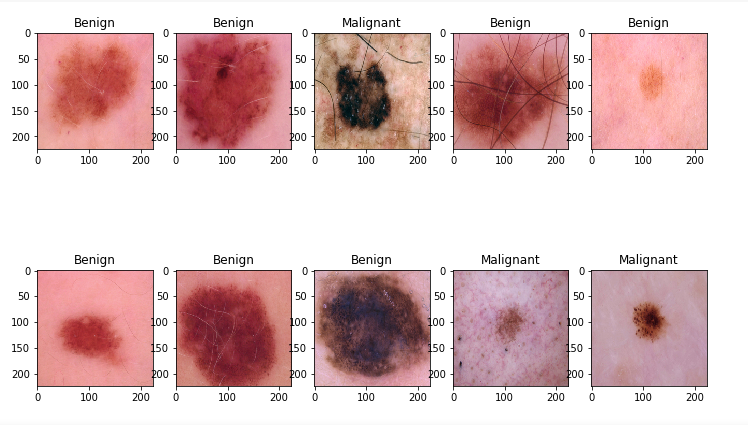


Figure 18: Output of original test data

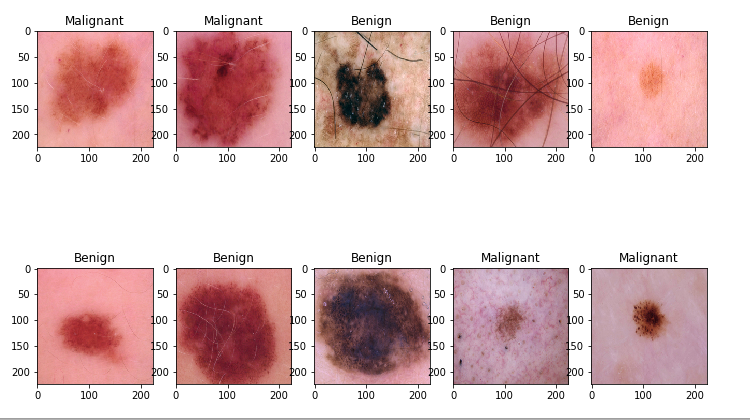


Figure 19: ResNet50 prediction on the same test data

**4.3 VGG16**

VGG16 is a variant of VGG model with 16 convolution layers and we have explored the VGG16 architecture in depth.

VGGNet-16 consists of 16 convolutional layers and is very appealing because of its very uniform Architecture.

**16 Layers of VGG16**

1.Convolution using 64 filters

2.Convolution using 64 filters + Max pooling

3.Convolution using 128 filters

4. Convolution using 128 filters + Max pooling

5. Convolution using 256 filters

6. Convolution using 256 filters

7. Convolution using 256 filters + Max pooling

8. Convolution using 512 filters

9. Convolution using 512 filters

10. Convolution using 512 filters+Max pooling

11. Convolution using 512 filters

12. Convolution using 512 filters

13. Convolution using 512 filters+Max pooling

14. Fully connected with 4096 nodes

15. Fully connected with 4096 nodes

16. Output layer with Softmax activation with 1000 nodes.

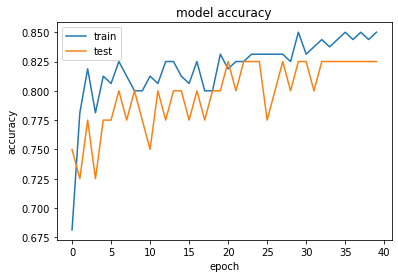


Figure 20: Vgg16

**Applying VGG16 in Our Dataset**

After applying ResNet50, we trained our model with VGG16 architecture to measure the accuracy of our dataset. We pass the same parameters in the model and started training it. After training our model with VGG16, we got about 84.242% accuracy from our dataset. The table above shows the loss, accuracy*,* validation loss and validation accuracy per 5 epochs of VGG16.It is clearly shown that the accuracy was rising up over time and the loss was degrading.

Figure shows the accuracy and loss of VGG16 throughout the training process.



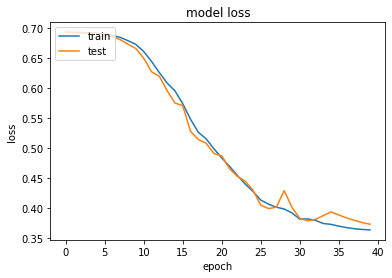


Figure 21: VGG16 Model Accuracy (accuracy vs epochs) and Model Loss (loss vs epochs)

Table 3: Outputs per 5 epochs in VGG16 Architecture

| No of epoches | loss | accuracy | val\_loss | val\_accuracy |
| --- | --- | --- | --- | --- |
| 1 | 0.6931 | 0.6812 | 0.6929 | 0.7500 |
| 5 | 0.6911 | 0.8125 | 0.6904 | 0.7750 |
| 10 | 0.6728 | 0.8000 | 0.6731 | 0.8000 |
| 15 | 0.5955 | 0.8125 | 0.5752 | 0.8000 |
| 20 | 0.4990 | 0.8313 | 0.4911 | 0.8000 |
| 25 | 0.4281 | 0.8313 | 0.4302 | 0.8250 |
| 30 | 0.3923 | 0.8500 | 0.4016 | 0.8250 |
| 35 | 0.3733 | 0.8438 | 0.3940 | 0.8250 |
| 40 | 0.3640 | 0.8500 | 0.3736 | 0.8250 |

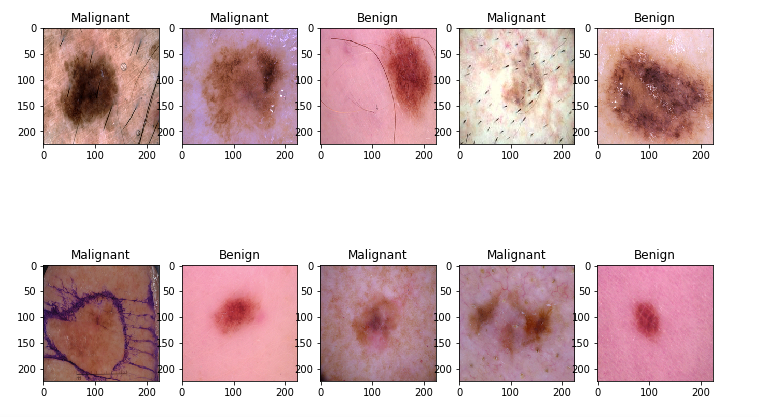


Figure 22: Output of original test data

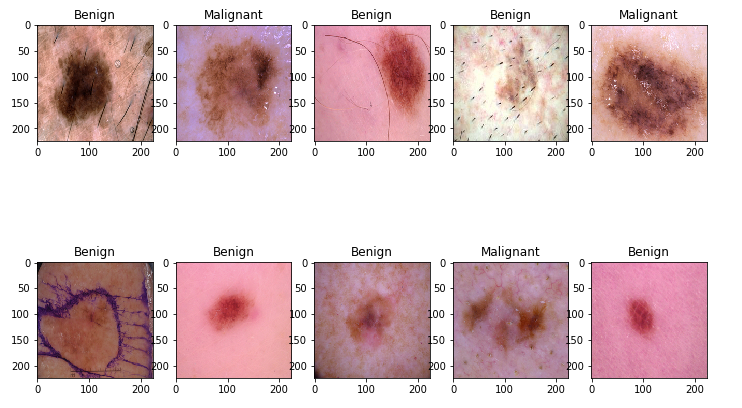


Figure 23: VGG16 prediction on the same test data

**4.4 Xception**

**Xception Concept**

Xception stands for Extreme version of Inception, is reviewed. With a modified depthwise separable convolution, it is even better than Inception-v3.

If a layer has five channels then there will be 5 nxn spatial convolution. Xception is considered better than Inception v3 on various datasets. There are two types of depthwise separable convolution, one is original depthwise separable convolution and modified depthwise separable convolution. Original depthwise separable convolution is the depthwise convolution followed by a pointwise convolution. Pointwise convolution is the 1x1 convolution to change the dimension. Depthwise separable convolution does not perform convolution across all its channels like conventional approach. For this reason, this is a convenient approach since the runtime gets optimized and the model shows better accuracy also the model gets lighter. On the other hand, modified depthwise separable convolution is the pointwise convolution followed by depthwise convolution. This modification is inspired by 1x1 convolution done by Inception v3 before performing nxn spatial convolution. Besides, to show the mean accuracy prediction when multiple objects appears in a single image, xception is the most fruitful approach.

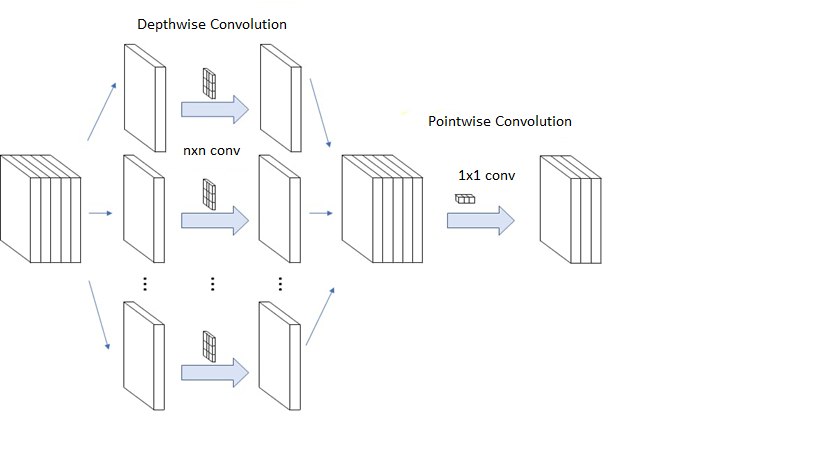


Figure 24: Original Xception

Below is the figure showing the modified depthwise separable convolution in xception

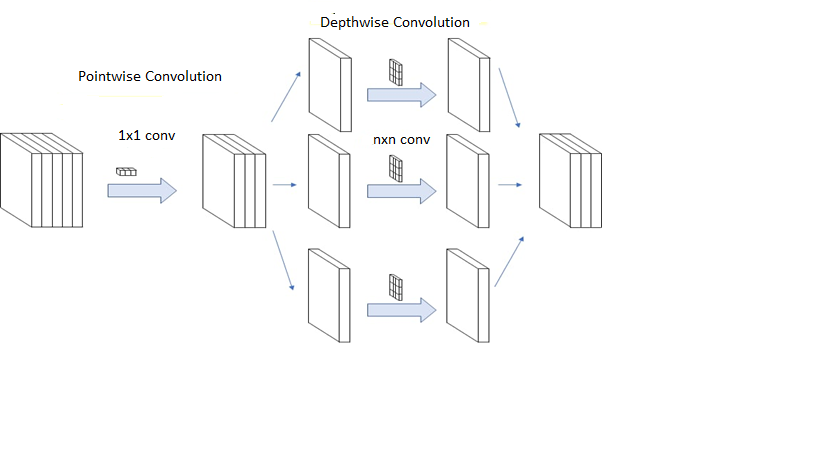
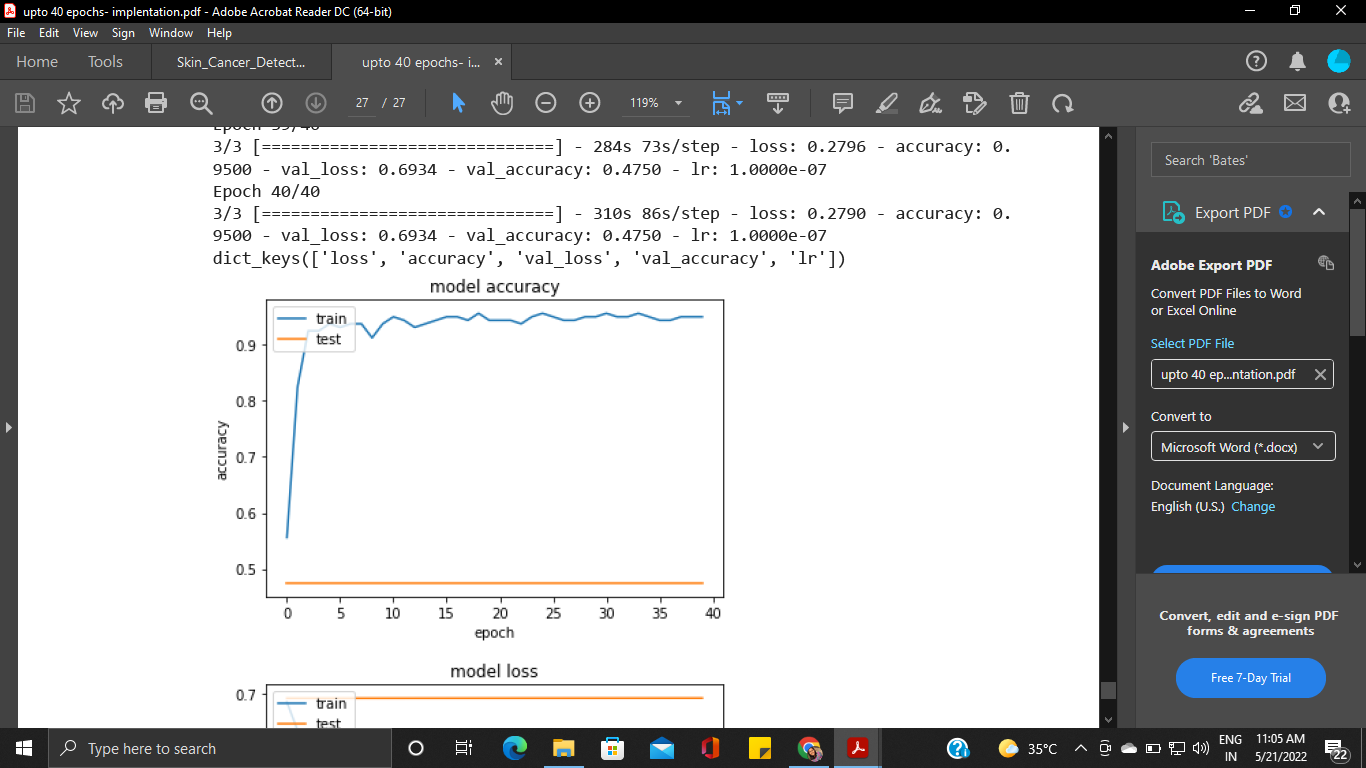


Figure 25: Modified Xception

.

**Applying Xception in Our Dataset**

After applying VGG16, we trained our model with Xception architecture to measure the accuracy of our dataset. We did all the process exactly as we did with other architectures by keeping all other parameter’s value exactly the same. After training our model with Xception, we got about 94% accuracy from our dataset. The table below shows the loss,accuracy, validation loss and validation accuracy per 5 epochs of Xception.It is clearly shown that the accuracy was rising up over time and and the loss was degrading.



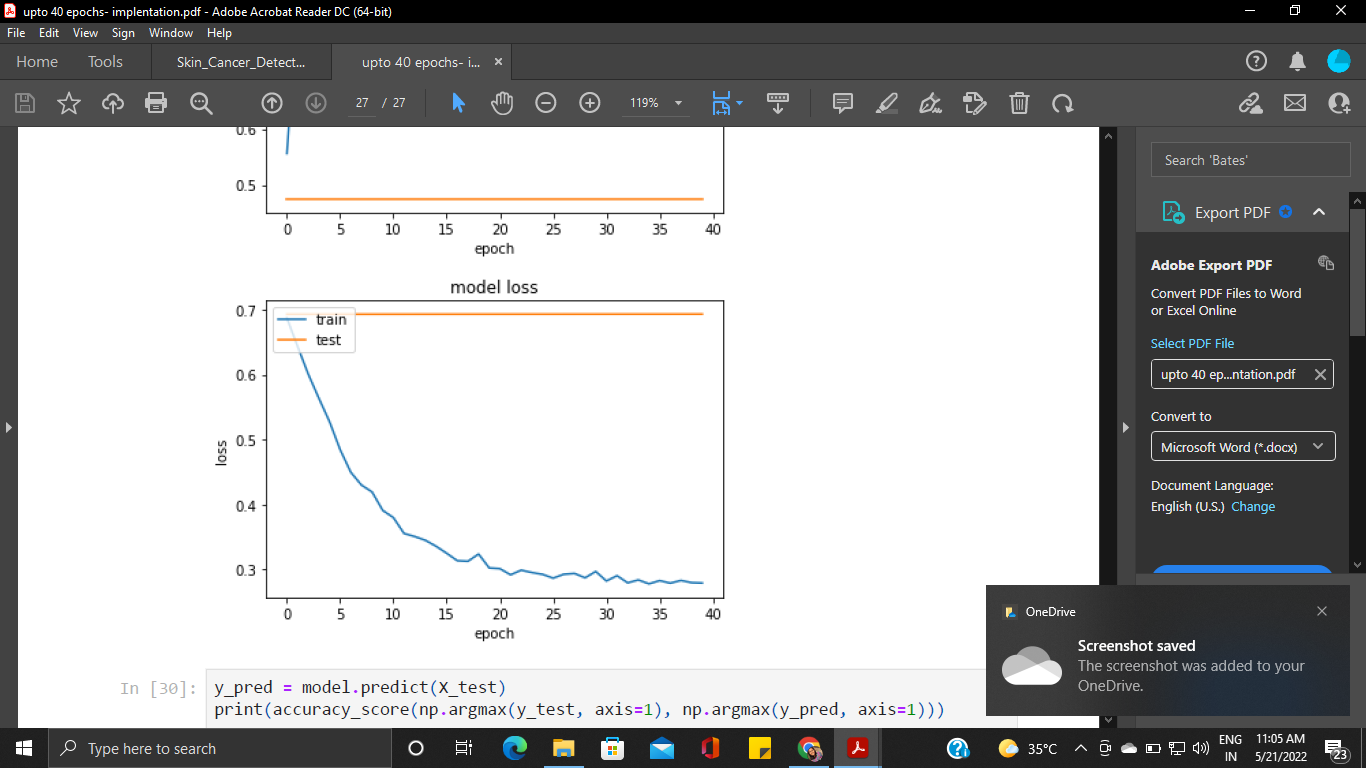
****

Figure 26: Xception Model Accuracy(accuracy vs epochs) Model Loss (loss vs epochs)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No of epochs** | **loss** | **accuray** | **val\_loss** | **val\_accuracy** |
| 1 | 0.6872 | 0.5562 | 0.6932 | 0.4750 |
| 5 | 0.5278 | 0.9375 | 0.6932 | 0.4750 |
| 10 | 0.3909 | 0.9375 | 0.6932 | 0.4750 |
| 15 | 0.3358 | 0.9438 | 0.6932 | 0.4750 |
| 20 | 0.3023 | 0.9438 | 0.6932 | 0.4750 |
| 25 | 0.2924 | 0.9563 | 0.6933 | 0.4750 |
| 30 | 0.2969 | 0.9500 | 0.6933 | 0.4750 |
| 35 | 0.2778 | 0.9500 | 0.6933 | 0.4750 |
| 40 | 0.2790 | 0.9500 | 0.6934 | 0.4750 |

**Table 4: Outputs per 5 epochs in Xception Architecture**

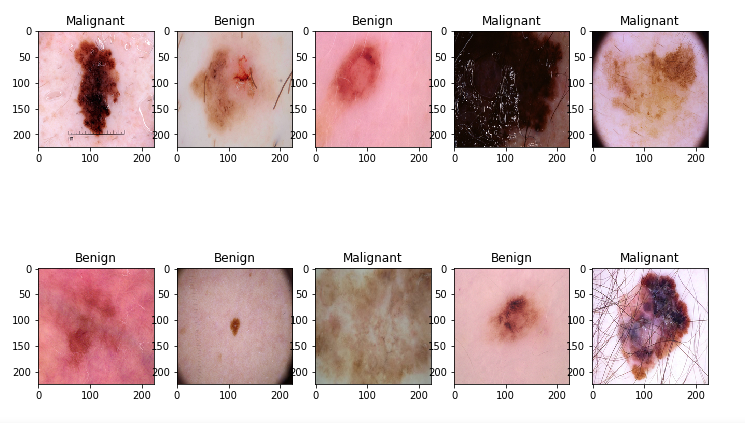
****

Figure 27: Output of original test data

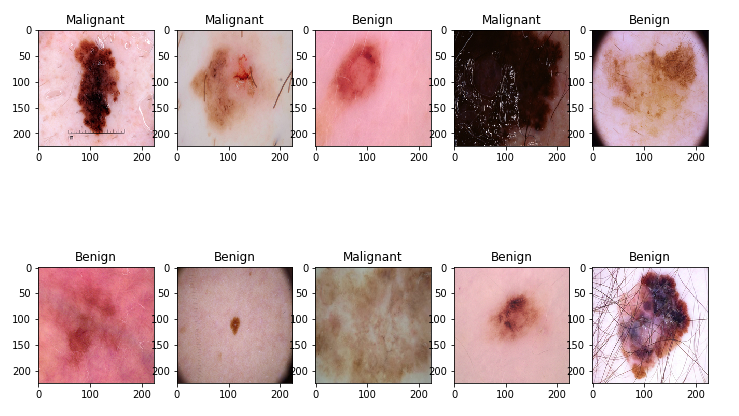
****

Figure 28: Xception prediction on the same test data

* 1. **Comparison of CNN Architectures**

The table below shows the comperative analysis of Accuracy and Loss of different CNN architectures that we have applied.

Table 5: comperative analysis of Accuracy and Loss of different CNN architectures that we have applied.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Name** | **loss** | **acurracy** | **val\_loss** | **val\_acurracy** |
| CNN | 0.7645 | 0.6938 | 0.5864 | 0.7250 |
| ResNet50 | 0.3064 | 0.9187 | 0.7175 | 0.5250 |
| VGG16 | 0.3640 | 0.8500 | 0.3736 | 0.8250 |
| Xception | 0.2790 | 0.9500 | 0.6934 | 0.4750 |

As we have tried to compare 4 different  architectures , a huge number of variance in prediction is witnessed on those architectures.

Out of those four architectures, the worst accuracy we got was from CNN

It was only 69%. Also the highest accuracy we got was from Xception which was 95%, ResNet50  gave a accuracy of 91% and VGG16 gave a accuracy of 85%  The only drawback VGG architectures have is that they are pretty much slower in terms of training and evaluating.Xception is one of the latest additions in CNN architectures. This architecture delivered the best accuracy out of all the architectures we have used on our dataset.There are certain reasons behind this. This architecture was built with the concept of previously dened architectures for example ResNet.This model uses an modied depthwise convolutional block which starts with a point-wise convolution. Furthermore, there is no non linearity between the blocks.

**CHAPTER 5**

**CONCLUSION AND**

**FUTURE SCOPE**

During the study time, we discovered a few of thesis works in this field. However, the majority of the effort was done only on CNN. Comparing CNN architectures was our concept, and we achieved a decent outcome. Other research works assisted us in training our model with CNN, however we had to train our model with different CNN architectures on our own. Our model might be improved by modifying the parameters, such as increasing the number of epochs, decreasing batch size, adjusting the value of dropout, and so on, which would obviously take longer.The best architecture that we found for our dataset was the Xception.

This architecture gave us about almost 95 % accuracy for our dataset. Due to low

power computer and other factors like, dataset image quality, image pre-processing,getting less accuracy can arise. Another concern that may occur that CNN takes a lot of time to process and train image data. Moreover, we had a dataset with having similar type of images, however when we will test it for real life implementation, we may get degraded accuracy than what we got now.

**Future Possibilities**

Skin cancer detection utilising CNN and related designs has a lot of room for improvement. The world is evolving at a breakneck pace, and so is technology, as well as the entire computer industry. Everything is becoming more efficient and quicker. Thus, with a better picture classification system, we may be able to enhance our detection accuracy and assist clinicians in detecting skin cancer in all patients sooner.

Keras also allows users to develop and build their own CNN models based on their own requirements.

Thus, by learning all the possibilities of all the parameters that have been used in our model, we can dene our own CNN architecture which will overcome all the obstacles to detect skin cancer more eciently. Moreover, we are still trying to gure out a best process to combine all the architectures together and then show the combined result, which is also known as the ensembling process.

We are working to complete two types of ensembling, one is, to average the predicted output of each models and then predict the output of an test image with mostly predicted value from all the architectures. The second one is, training one model after another using the predicted output from the previous model as the input of the next model and then nish all the algorithms in this process. Another important work that we are trying to do from our work is to convert our dataset into CSV le, means into numerical values and apply machine learning algorithms furthermore to determine the accuracy of our proposed system with a higher rate of accuracy.

**Bibliography**

[1] <http://towardsdatascience.com/introduction-to-resnets-c0a830a288a4/>

[2] <https://machinethink.net/blog/mobilenet-v2/>

[3] <https://machinethink.net/blog/compressing-deep-neural-nets/>

[4] <https://developer.nvidia.com/discover/convolutional-neural-network>

[5] <https://engmrk.com/vgg16-implementation-using-keras/>

[6] [https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras- 446d7ff84d33/](https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-%20%20%20446d7ff84d33/)

[7] <https://www.wcrf.org/dietandcancer/cancer-trends/skin-cancer-statistics>

[8] <https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568>

[9] <https://www.kaggle.com/datasets/fanconic/skin-cancer-malignant-vs-benign>

[10] <https://ieeexplore.ieee.org/abstract/document/9340143>