1.INTRODUCTION

1.1.MOTIVATION

Our motivation was to explore the field of Deep Learning, and we have observed that the detection of diseases when using image processing can help rural as well as urban people. Skin diseases can be fatal if not treated in earlier stages, so we had this idea of collecting 3 most common skin cancers and use it to predict the type of cancer. To understand how Deep Neural Networks work, concepts such as Transfer Learning which can help us fine-tune our model by pre-trained weights.

1.2.PROBLEM DEFINITION

There are many types of skin diseases and skin cancers in which some of them can be fatal. Skin cancers must be treated in the earlier stages or it can cost one's life. Many people neglect the skin lesions which can cost their life. In rural areas, where the hospitals are not well equipped, struggle to detect the diseases in early stages. So the need for automatic skin disease prediction is increasing for the patients and as well as the dermatologist.

The available diagnosis procedure consists of long laboratory procedures but this system will enable users to predict the skin disease using image processing. With a predictor system, a patient or the dermatologist can check whether the lesion is malignant or not easily, so that cancer can be treated in the early stages. We collected images of three most common skin cancers and the prediction system is used to predict these 3 skin cancers, namely Melanoma, Squamous cell carcinoma (SCC), Basal cell carcinoma (BCC). Early detection of Melanoma can potentially improve survival rate, yet nearly 30,000 people die yearly in the US alone. People

often neglect their health conditions in the early stages and run to the hospitals when things get worse. This can be avoided by helping people understand or diagnose the disease. Skin cancers

do not give any pain most of the time, but they are clearly visible, as the cancer is nothing but the abnormal growth of skin cells.

1.3.CASE STUDY

Our Case Study for this project mainly focuses on the issues that are faced in the dermatological industry, the project finally tries to present a solution. This study has been conducted by researching the internet from trusted websites, and we have formatted them in a Q&A format for better understanding and readability.

1.How common is Skin cancer?

WHO says —The incidence of both non-melanoma and melanoma skin cancers has been increasing over the past decades. Currently, between 2 and 3 million non-melanoma skin cancers and 132,000 melanoma skin cancers occur globally each year. One in every three cancers diagnosed is skin cancer and, according to Skin Cancer Foundation Statistics, one in every five Americans will develop skin cancer in their lifetime. As ozone levels are depleted, the atmosphere loses more and more of its protective filter function and more solar UV radiation reaches the Earth's surface. It is estimated that a 10 per cent decrease in ozone levels will result in an additional 300,000 non-melanoma and 4,500 melanoma skin cancer cases. The global incidence of melanoma continues to increase – however, the main factors that predispose to the development of melanoma seem to be connected with recreational exposure to the sun and a history of sunburn. These factors lie within each individual's own responsibility.

Deaths- Currently, between 2 and 3 million non-melanoma skin cancers and 132,000 melanoma skin cancers occur globally each year. About 6,850 people die from melanoma each year. For other, less common types of skin cancer, about 4,630 people die every year.The vast majority of skin cancer deaths are from melanoma.Nearly 20 Americans die from **melanoma** every day. In 2019, it is estimated that 7,230 deaths will be attributed to melanoma — 4,740 men

and 2,490 women. Research indicates that men diagnosed with melanoma between the ages of 15 and 39

were 55 percent more likely to die from melanoma than females diagnosed with melanoma in the same age group.18 People with SCC have a higher risk of death from any cause than the general population. An estimated 4,420 deaths from skin cancers other than melanoma and NMSC are expected to occur in the United States in 2019.

Survival Rates - Basal cell and squamous cell carcinomas, the two most common forms of skin cancer, are highly curable if detected early and treated properly. The five-year survival rate for people whose melanoma is detected and treated before it spreads to the lymph nodes is 99 percent. Five-year survival rates for regional and distant stage melanomas are 64 percent and 23 percent, respectively.

2.How fast can Melanoma Kill a person?

The ACS reports that "the five-year relative survival rate for melanoma is 92 percent. Eighty-four percent of cases are diagnosed at a localized stage, for which the five-year survival rate is 98 percent."Melanoma can grow very quickly. It can become life-threatening in as little as six weeks and, if untreated, it can spread to other parts of the body.

3.Can you have skin cancer for years without knowing?

According to healthline.com, certain types of skin cancer can be diagnosed initially just by visual inspection — though a biopsy is necessary to confirm the diagnosis. But other cancers can form and grow undetected for 10 years or more, as one study found, making diagnosis and treatment that much more difficult.

4.How is skin cancer detected?

According to skincancer.net, a skin biopsy is needed to diagnose skin cancer. Your doctor removes a sample of skin tissue, which is sent to a laboratory. In the laboratory, a pathologist

studies the sample under a microscope. The pathologist looks for abnormal cells that indicate cancer.

5.Is Skin Cancer lesion always visible?

According to cancercenter.com, Skin cancer may initially appear as a nodule, rash or irregular patch on the surface of the skin. These spots may be raised and may ooze or bleed easily. As the cancer grows, the size or shape of the visible skin mass may change and the cancer may grow into deeper layers of the skin.

6.How long would it take for the biopsy results?

According to cancerresearchuk.org, It takes about 2 to 3 weeks to get the results of your biopsy. You usually go back to your GP or skin specialist (dermatologist) for these. You need treatment for the area if the skin sample contains any cancerous cells.

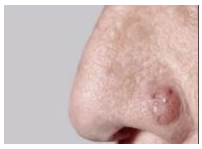
1.3.1.STUDY CONCLUSIONS

As mentioned previously, our aim is to create a classifier that can accurately classify a skin cancer lesion. From the following case study questions,Q1 tells us the need for this classifier, tells us how common skin cancer is, and joining it with Q2 we can say it's very much curable if detected at an early stage. However Q3 talks about how, many people do not take a skin lesion seriously, and whether it is malignant or not, it goes unnoticed and this can only get worse. For a skin cancer problem, we can easily use an image classifier approach, because Q4 gives us the evidence that it's the only type of cancer that can be detected visually and Q5 says skin cancers always start with a smaller diametric lesion. So in this case the best solution would be for people themselves to see if a lesion is cancerous or not and what type it is. From Q6 we can make out that these results would take as long as 2 to 3 weeks, but instead if a machine can classify its most probable type and further test it for that type might reduce the time.

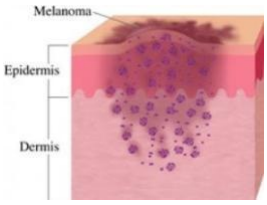
1.3.2.TYPES OF SKIN CANCERS

BASAL CELL CARCINOMA :

- Basal cell carcinoma, or BCC, is a cancer of the basal cells at the bottom of the epidermis.
- Occasionally some BCCs are aggressive, and, if left to grow, may spread into the deeper layers of the skin and sometimes to the bones. This can make treatment difficult.

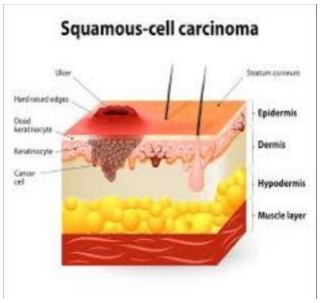


MELANOMA :



- This is a less common type of skin cancer. Melanoma behaves differently to BCC and SCC.
- It can grow quickly and needs to be treated early. If melanoma is not removed the cells can grow deeper into the layers of the skin.
- If melanoma cells get into the blood or lymphatic system, they can travel to other parts of the body.

SQUAMOUS CELL CARCINOMA :



- Squamous cell carcinoma, or SCC, is a cancer of the cells in the outer layer of the skin.
- It is the second most common type of skin cancer.
- Usually, SCCs are slow-growing. They only spread to other parts of the body if they are left untreated for a long time.

- Occasionally, they can behave more aggressively and spread at an earlier stage.

1.4.OBJECTIVE

1. **Primary Objective:** Our primary objective is to classify the skin cancers to their types by building the best model using Convolutional Neural Network and Transfer Learning. To deploy it as a web application which allows the patient to upload a picture of the skin lesion or the infected area and by clicking the Predict Button which would input to the prototype and the further processing and analysis is done on this input image to give the final result.

2. **Second Objective:** Our second objective is to learn how Convolutional Neural Networks work on sample data like this, to learn and carefully observe other architectures such as VGG16/19, InceptionV2/V3, ResNet50/101 etc which have excelled in Imagenet throughout the years. Study the results with and without transfer learning, learn how to deploy a Deep Learning model into a web application using Node.js and Tensorflow.js .

2. LITERATURE REVIEW AND PROPOSED WORK

2.1. LITERATURE REVIEW :

1. **(Uzma Bano Ansari et al., 2017):** Proposed a technique to classify a melanoma lesion only. A very careful study on skin cancers and their anotomy, performed three preprocessing techniques --- Gray scale conversion, noise removal, and image enhancement. Used Support Vector Machine Classifier, the image is segmented and then fed into the fit function for training, as melanoma is tested based on the shape. Used a very small sample size, which cannot identify the possibilities of real world images.

2. ***(Xin He et al., 2019):** Computer-Aided Clinical Skin Disease Diagnosis using CNN and Object Detection Models, gives us the influence of object detection technique in this approach for training, which can increase the accuracy and decrease the computation cost by removing unwanted background learning. Used an Ensemble learning approach to get the final output. Used two datasets Skin-10(which contains images belonging to 10 classes of skin diseases, with a total of 10,218 images) and Skin-100(which contains 100 classes of common skin diseases with a total of 19,807 images). The best accuracy achieved is 79.01% for dataset Skin-10 and 53.54% for dataset Skin-100, the reason we believe for the low accuracy is because of the class imbalance after checking the dataset they mentioned, the model was sensitive towards the high volume class, which may have led to the poor performance. Emphasizes on the importance of object detection and ensemble learning methods.

3. **(Enakshi Jana et al., 2017):** Research on Skin Cancer Cell Detection using Image Processing, gives the role of segmentation, features extraction in image processing, proposes a technique to remove unwanted features in the area of interest, such as hair.

4. **(Pravin S. Ambad et al.,2016):** A Image analysis System to Detect Skin Diseases, workflow comprises basic steps. Used a two level classifier, which is a great idea, the first classifier detects if a lesion is normal or defected and further if its infected, the second classifier classifies whether the lesion is a Melanoma, Psoriasis, Dermo. Two way classifier is a good approach.

2.2. EXISTING WORK AND DISADVANTAGES

There are some existing systems which scan the skin lesion by using Image processing and other algorithms. Most of the methodologies try to extract features using few methods and tools, while in CNN, the model itself would look for these features and extract them after every fully connected layer. Using pre-trained models would also help us in attaining better accuracy.

Disadvantages :

As it is observed, image recognition is very complex and challenging one affecting a variety of parameters such as intensity, skin tones, body parts, color and size. The existing systems are confined to one skin disease or skin cancer. They predict when the skin lesion is malignant or not without actually classifying the image.

2.3. PROPOSED WORK

The proposed work is intended to classify the most harmful three types of skin cancers. The proposed work uses different architectures to define the best architecture, where best architecture is evaluated by its performance, also using transfer learning to train the model with pre-trained weights of imagenet dataset. The further reading of this documentation would show the results of transfer learning are better than simply training the model with no weights. The proposed work uses several augmentation

methods to generate more images, to go with the principal, more images, better results. ResNet101 with imagenet pretrained weights has given us the best accuracy performance,

than any previously published work, with training accuracy of 0.9992 and validation accuracy of 0.9563. The further sections would explain the work in a detailed manner.

2.4. TECHNOLOGICAL REVIEW

PYTHON

For implementing the project, we are making use of python language as the main language. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The following libraries are used for the project in python:

KERAS

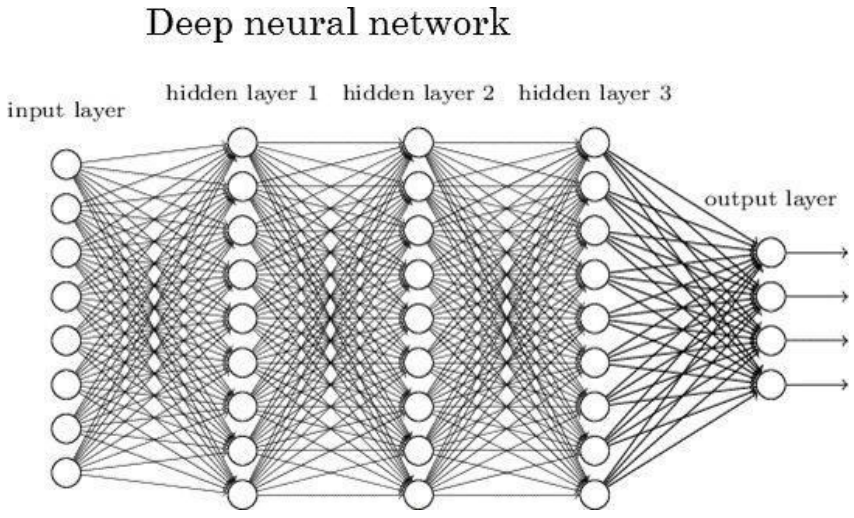
Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

TENSORFLOW

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. TensorFlow provides stable Python and C APIs, and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift.

DEEP LEARNING

Deep learning techniques utilizes the image appearance pattern around the tumour. The appearance pattern includes information on local contrast, nearby tissues, boundary sharpness, and etc. Such information is different from but as powerful as the diagnostic features.

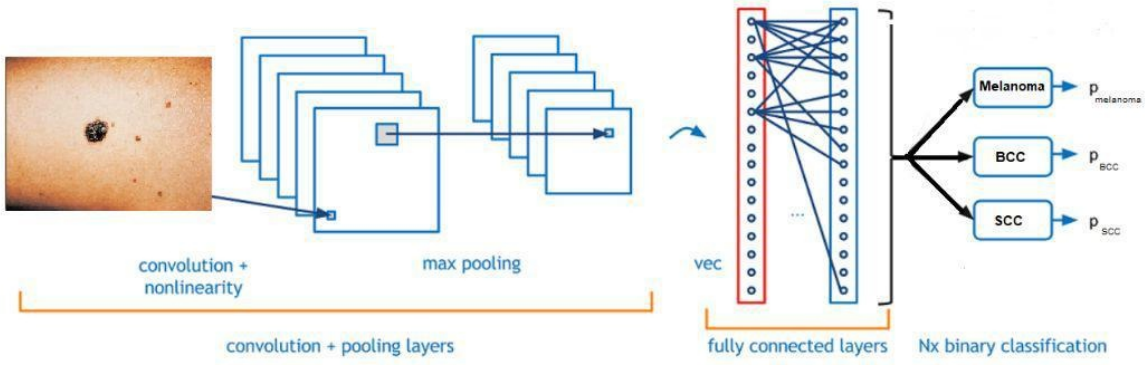


They are not affected by the size of the tumour, because they are computed from the entire image patch which includes both the tumour and its surrounding tissues.

CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (CNN) is an algorithm in deep learning which consists of a combination of convolutional and pooling layers in sequence and then followed by fully connected layers at the end as like a multilayer neural network. CNN is a class of algorithms which is motivated to take advantage of any 2D structure in data. Hence CNN is one of the preferred algorithms for image classification.

CNN also proves promising for tasks related to natural language processing. CNN leverages the local feature of an image to achieve higher accuracy for classification. CNN is one of the neural networks which is easy to train and has less number of hyperparameters as compared to ordinary fully connected neural networks.



TRANSFER LEARNING

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems. In computer vision, transfer learning is usually expressed through the use of pre-trained models. A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem similar to the one that we want to solve. Accordingly, due to the computational cost of training such models, it is common practice to import and use models from published literature (e.g. VGG, Inception, ResNet).

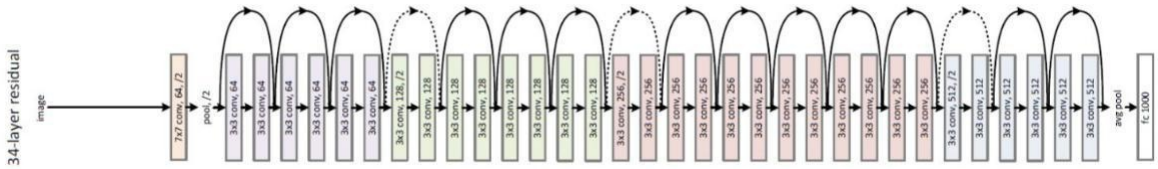
IMAGE NET

The rise in popularity and use of deep learning neural network techniques can be traced back to the innovations in the application of convolutional neural networks to image classification tasks. Some of the most important innovations have sprung from submissions by academics and industry leaders to the ImageNet Large Scale Visual Recognition Challenge, or ILSVRC, is an annual competition that uses subsets from the ImageNet dataset and is designed to foster the development and benchmarking of state-of-the-art algorithms. The ImageNet dataset is a very large collection of human annotated photographs designed by academics for developing computer vision algorithms. The ILSVRC tasks have led to milestone model architectures and techniques in the intersection of computer vision and deep learning.

The reason we have mentioned IMAGENET is because the pre-trained models we used for transfer learning are trained on Imagenet dataset, which has over 14,197,122 images belonging to 1000 classes. In this project we'll be using the architectures that have won the ILSVRC throughout these years, to see how their pre-trained weights would help us in achieving our objective to classify the skin cancer types. Since there is no similarity between our dataset and the images belonging to ImageNet we'll be using the transfer learning method, where we'll train the model with the top layers frozen and get the best out of this.

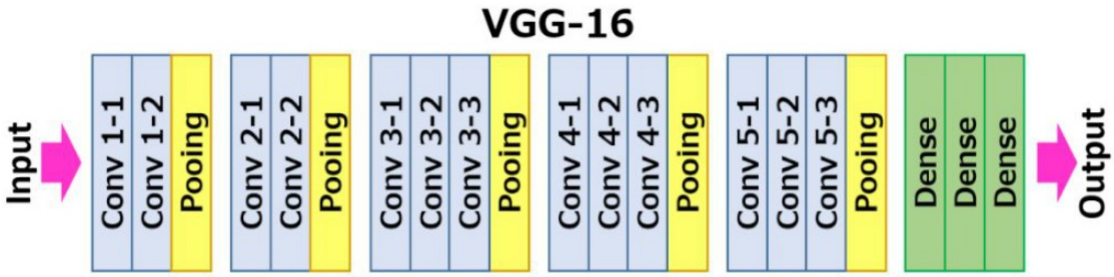
Residual Networks(ResNet - 50, 101)

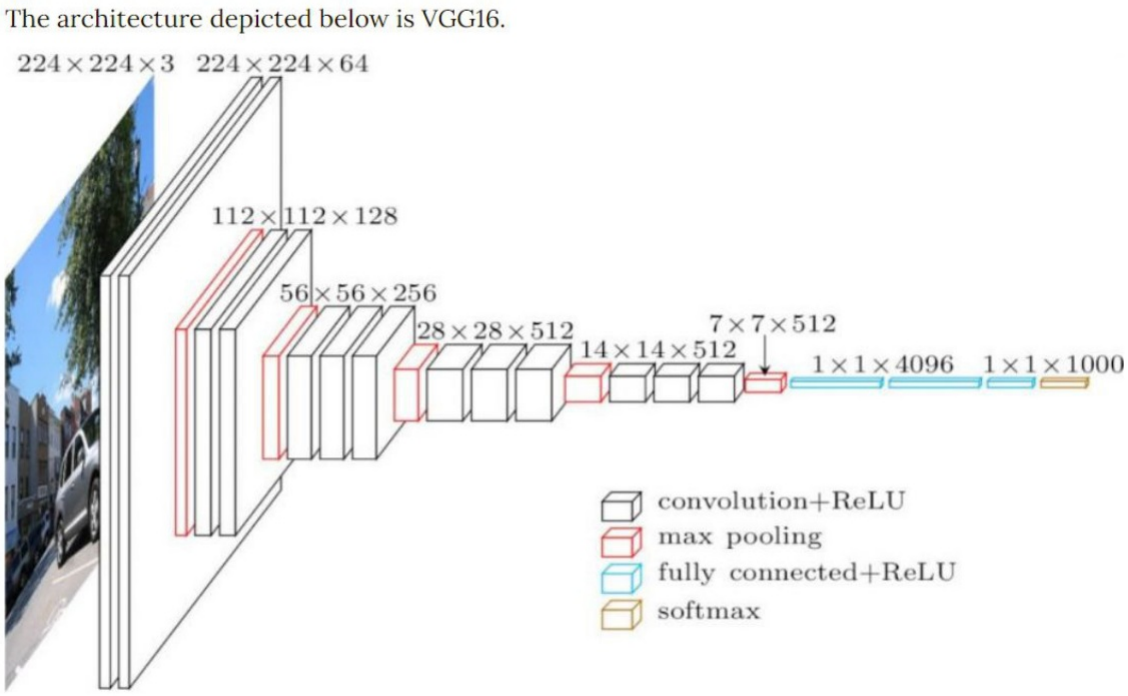
A Residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. ResNet-N is a deep residual network that is N layers deep. It is a subclass of convolutional neural networks, with ResNet most popularly used for image classification. We can load a pre-trained 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. As a result, the network has learned rich feature representations for a wide range of images.



Visual Geometry Group(VGG16)

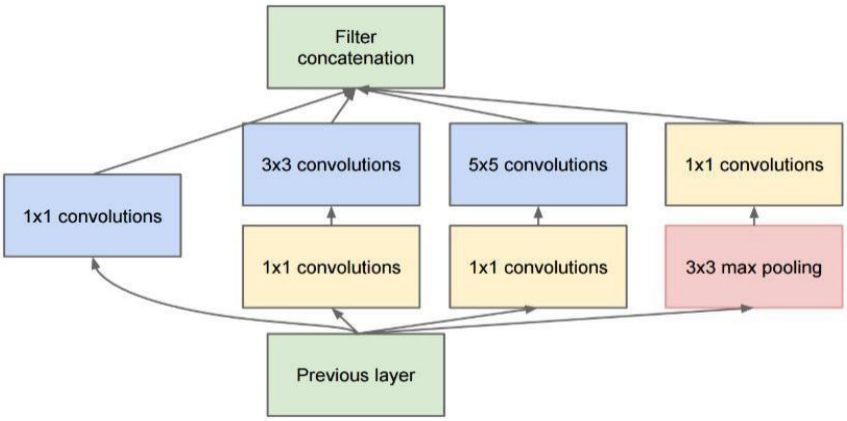
VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to ILSVRC-2014. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth.



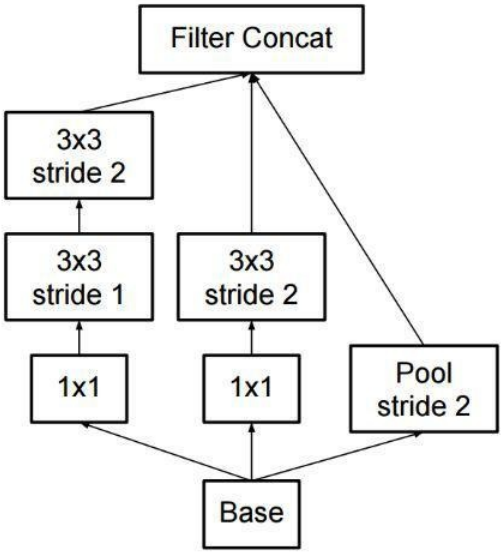


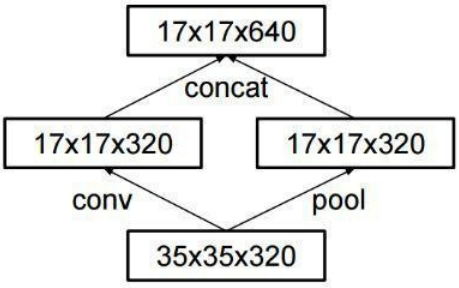
INCEPTIONV2

Inception Layer is a combination of all those layers namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer with their output filter banks concatenated into a single output vector forming the input of the next stage.



The change to inception v2 was that they replaced the 5x5 convolutions by two successive 3x3 convolutions and applied pooling:


Filter Concat
3x3
stride 2
3x3
stride 1
3x3
stride 2
1x1
1x1
Pool
stride 2
Base


17x17x640
concat
17x17x320
17x17x320
conv
pool
35x35x320

FRAMEWORKS

NODE.JS

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting, i.e., running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

Node.js can generate dynamic page content. It can create, open, read, write, delete, and close files on the server. It can collect form data and can add, delete, modify data in the database.

TENSORFLOW.JS

TensorFlow. js is a JavaScript library developed by Google for training and using machine learning (ML) models in the browser. It's a companion library to TensorFlow, a popular ML library for Python. TensorFlow.js also includes a Layers API, which is a higher level library for building machine learning models that uses Core, as well as tools for automatically porting

TensorFlow SavedModels and Keras hdf5 models. With TensorFlow.js we can import an existing, pre-trained model for inference and we can re-train an imported model.

EXPRESS.JS

Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js. It provides a robust set of features for web and mobile applications.

Express provides a thin layer of fundamental web application features, without obscuring Node.js features.

jQUERY

jQuery is a fast, small, and feature-rich JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on the website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

3.REQUIREMENT ANALYSIS

3.1. PLATFORM REQUIREMENT

The supported Operating Systems for client include:

- Windows XP onwards
- Linux any version

Windows and Linux are two of the operating systems that will support comparative website. The comparative website will be tested on both Linux and windows.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1. Software Requirement

The Software Requirements in this project include:

- Python 3.4(and above)
- A Windows 10 OS, as Python 3.4 does not run on Windows XP anymore.
- All the necessary modules to be installed.

3.2.2. Hardware Requirement

Hardware Requirement for project development includes:

- 4 GB Ram

- 40 GB Hard Disk Minimum
- Intel Xeon/AMD Ryzen 3 1200 and above

3.3. FLOW CHART

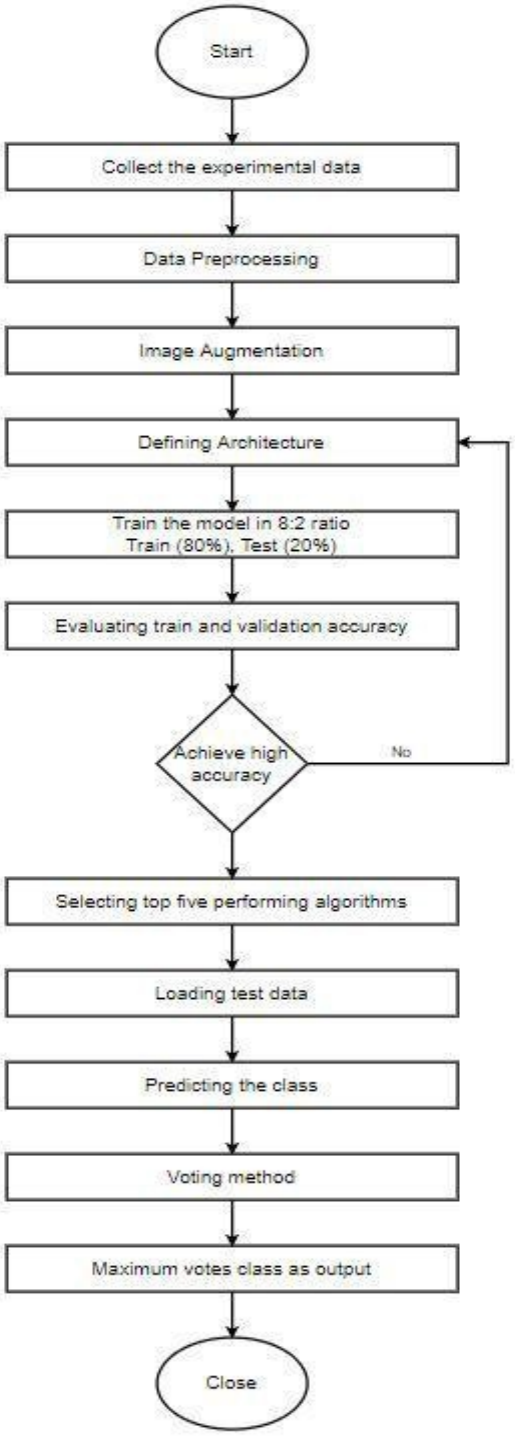


Fig 3.1

3.4. SYSTEM ARCHITECTURE

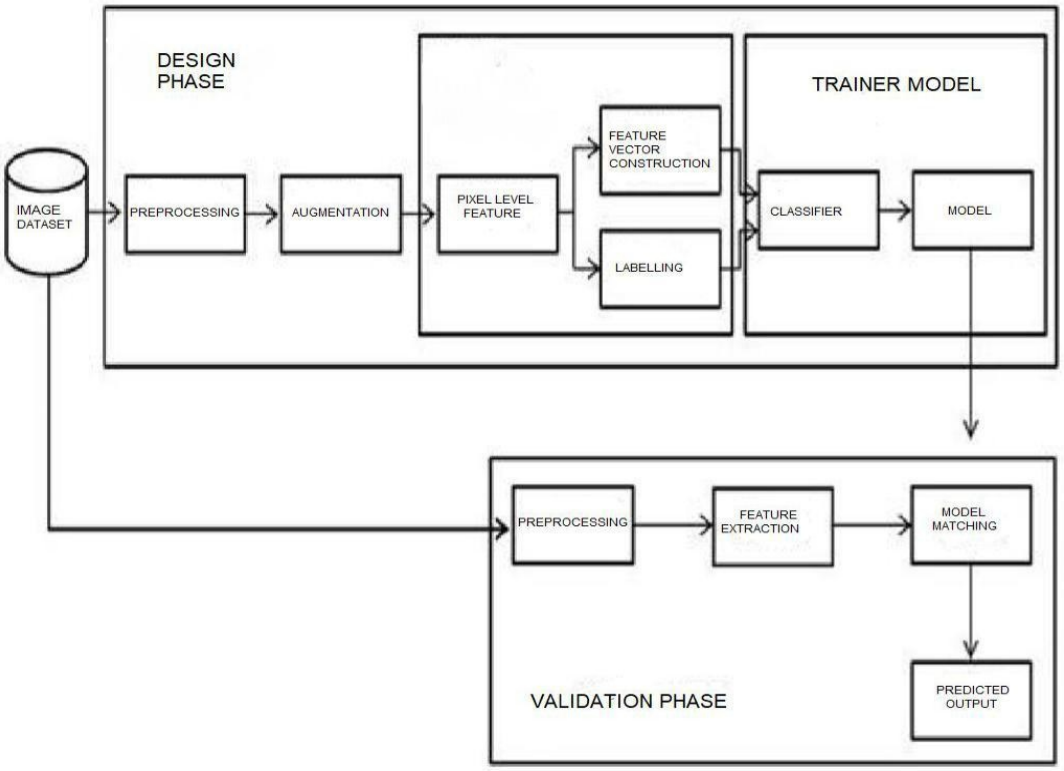


Fig 3.2

Description :

The above diagram explains the overall working of the system for detecting the skin cancer. The image dataset is preprocessed and augmented so that the features of each skin cancer is detected which is used for classification. A model is selected to predict the class of the image. The top 5 performing algorithms are selected and the test data is loaded to make the predictions. Voting method takes place and the class with maximum votes comes as the output. The accuracy of the prediction is displayed to the user.

4.DESIGN

4.1.INTRODUCTION

Design is the most important step in the development phase. Once the software requirements have been analyzed and specified the software design involves three technical activities: design, coding, implementation and testing that are required to build and verify the software. The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer requirements into finished software or a system. Software design is a process through which requirements are translated into a presentation of software.

4.2.UML DIAGRAMS

4.2.1 DEPLOYMENT DIAGRAM

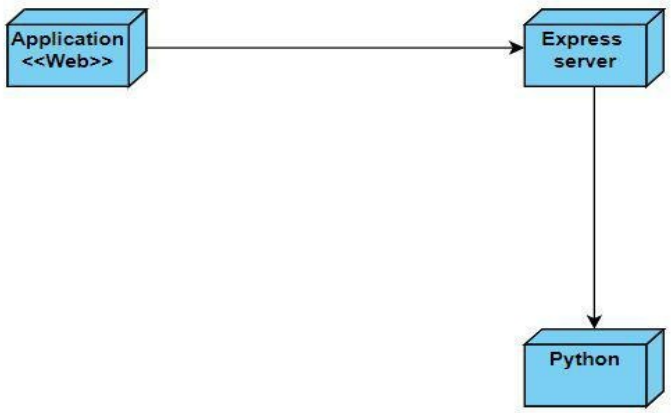


Fig 4.1

4.2.2. USECASE DIAGRAM

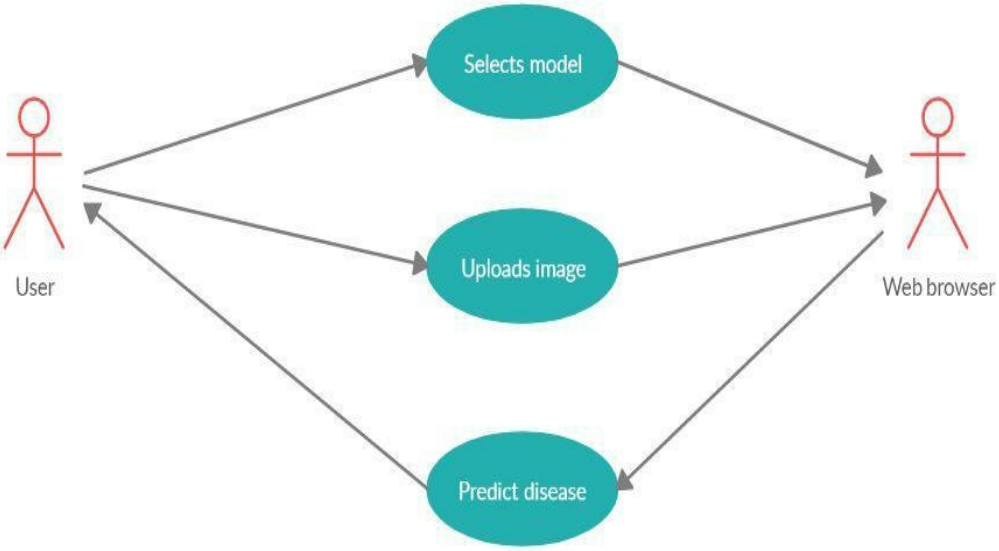


Fig 4.2

- As shown in the above figure, after opening the browser the user selects the model which processes the image and displays the result.
- The user uploads the image and the uploaded image is displayed on the screen indicating that the upload is successful.
- The user should click the predict button and the results are displayed to the user showing the image belongs to which class.

4.2.3. SEQUENCE DIAGRAM

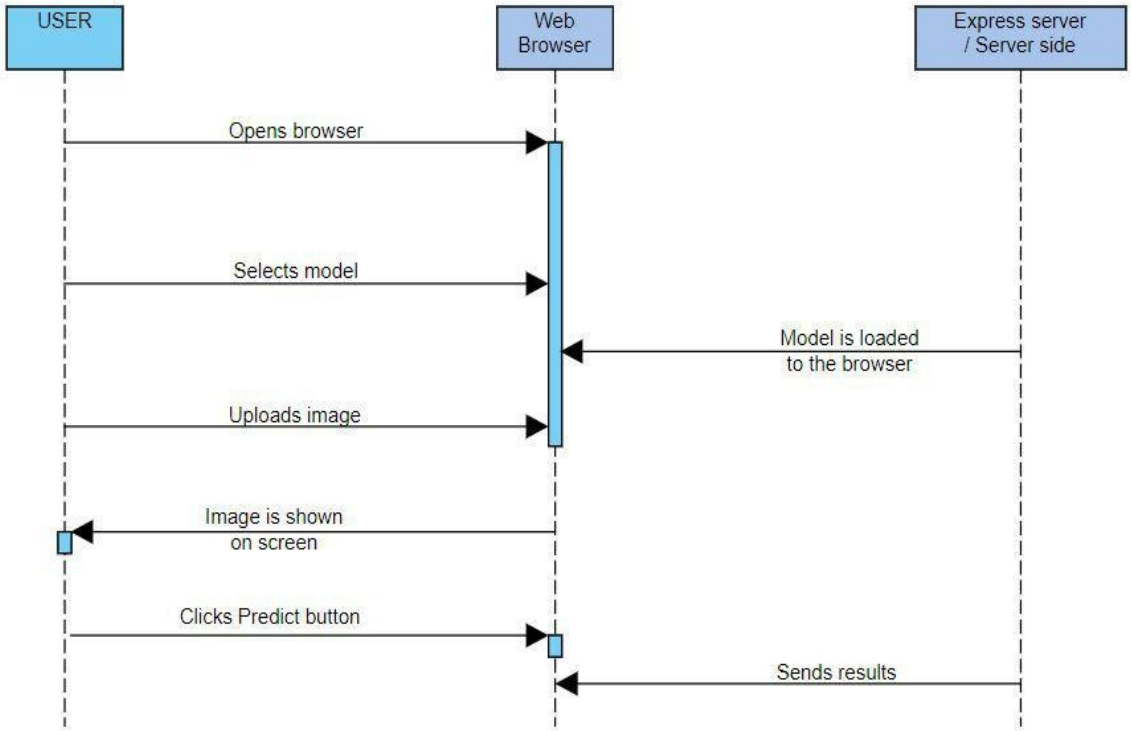


Fig 4.3

- The user opens the browser and a html page is displayed where he can upload the image to check the lesion.

- The user selects a model and the respective model is loaded to the browser.
- The user uploads an image and as a response the web browser displays the uploaded image indicating the upload is successful.
- After the image is uploaded the user clicks the predict button, in return the results will be displayed to the user.

5. IMPLEMENTATION AND RESULTS

5.1. INTRODUCTION

1. To classify the skin cancers and build the best model using classified images as a training set. The characteristics of skin cancer highly influence the class attribute or high value dependency.
2. Our second objective is to learn how Convolutional Neural Networks work on sample data like this, to learn and carefully observe other architectures such as VGG16/19, InceptionV2/V3, ResNet50/101 etc which have excelled in Imagenet throughout the years. We also want to see how Machine Learning algorithms would perform with image classification -- We use general ML models and Ensemble methods., which are explained in detail further.

5.2.IMPLEMENTATION

Steps involved in Back End development.

5.2.1 Data Analysis

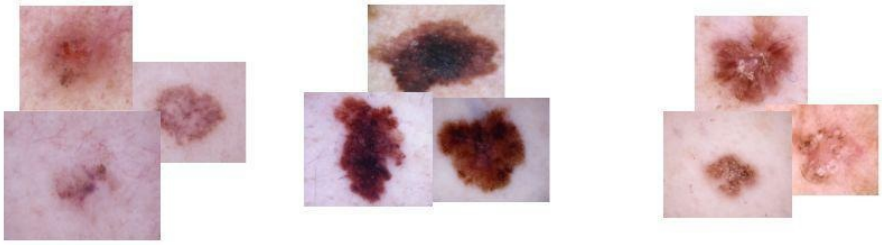
1. Data Collection :

- The data has been collected from various sources - HAM 10000 dataset, ISIC Dataset, Web scraping, Social media, other sites.
- We have collected Dermoscopic images, Precancerous and Cancerous images , which covers all the characteristics of each type of skin cancer.

- Dermoscopic images are collected for the hospitals and dermatologists by which the predictor system can help them better in identifying the disease.

Dataset Sources -

1. HAM 10000 dataset : The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. The images required for the project, that is, BCC, Melanoma, SCC images are filtered from the dataset of 10015 dermatoscopic images. The required images are filtered with the help of metadata provided along with the HAM10000 dataset.



(a)BCC (b) Melanoma (c) SCC

Dermatoscopic images of

BCC, Melanoma, SCC collected from the HAM10000 Dataset.

2. Web Scraping : Web scraping is an automated method used to extract large amounts of data from websites. The data on the websites are unstructured. Web scraping helps collect these unstructured data and store it in a structured form.

Tools used for web scraping - Selenium

Selenium : Selenium is a Web Browser Automation Tool. It allows you to open a browser and perform tasks, in our case, searching for specific information on the web pages. The required web pages are filtered and upto 1000 images are retrieved for each class.

3. Social Media : We have collected images from Instagram by using hashtags, only the necessary images focussing the skin lesions are cropped and edited to add these images to the dataset. The unnecessary images are deleted from the dataset.

4. ISIC Dataset : The International Skin Imaging Collaboration: Melanoma Project is an academia and industry partnership designed to facilitate the application of digital skin imaging to help reduce melanoma mortality. We have collected 587 BCC, 603 Melanoma, 225 SCC images from this dataset. ISIC has developed and is expanding an open source public access archive of skin images to test and validate the proposed standards. This ISIC archive serves as a public resource of images for teaching and for the development and testing of automated diagnostic systems.

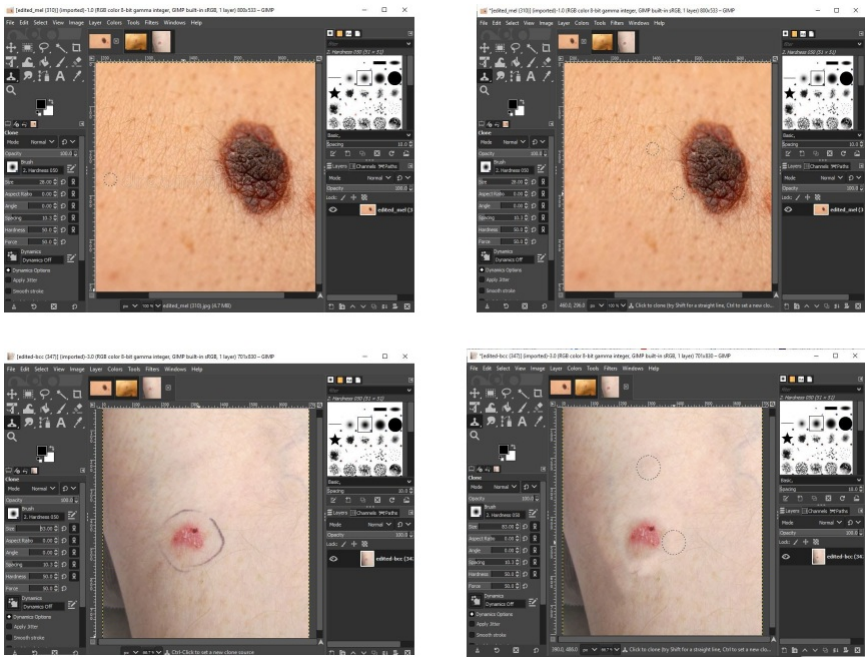
2. Data Cleaning :

- Data cleaning is done by removing the unwanted images and cropping the images so that the lesion is focused.
- After data collection, the images are uploaded to the google drive in different folders, that is, HAM 10000, Web, Social media.
- Each and every image is manually checked and the images of each folder are divided into three segments, namely, Original, Edit, Crop.
- Original - The images which require no changes.
  Edit - The images which require watermark, markings removal.
  Crop - The images which are to be cropped to focus the lesion.
- The above changes are made to the images and these are stored as our target dataset.
- Some images contained watermarks and clinical markings on lesions which might confuse the model, these pictures have been either avoided or edited.

Tool used -

GIMP : GIMP is a free and open-source raster graphics editor used for image retouching and editing, free-form drawing, converting between different image formats, and more specialized tasks.

We used gimp to edit watermarks, sketch markings, etc from the images obtained from web scraping and social media.



5.2.2 Generating ImageData

Since our dataset is imbalance, which means our classes have different number of images, for melanoma we have found many images, the data available was huge since its a very common, but we found less number of images for Squamous and Basal Cell Carcinoma, for Melanoma it were 1500+ images and for SCC 700+ and for BCC 900+ images. With class imbalanced data, there is a greater chance that the model might overfit to the class belonging the highest number of classes, or be sensitive to only the class that has more images. To avoid this vulnerability to overfitting we have used ImageDataGenerator class, that generates images with specified image augmentations.

IMAGE AUGMENTATION

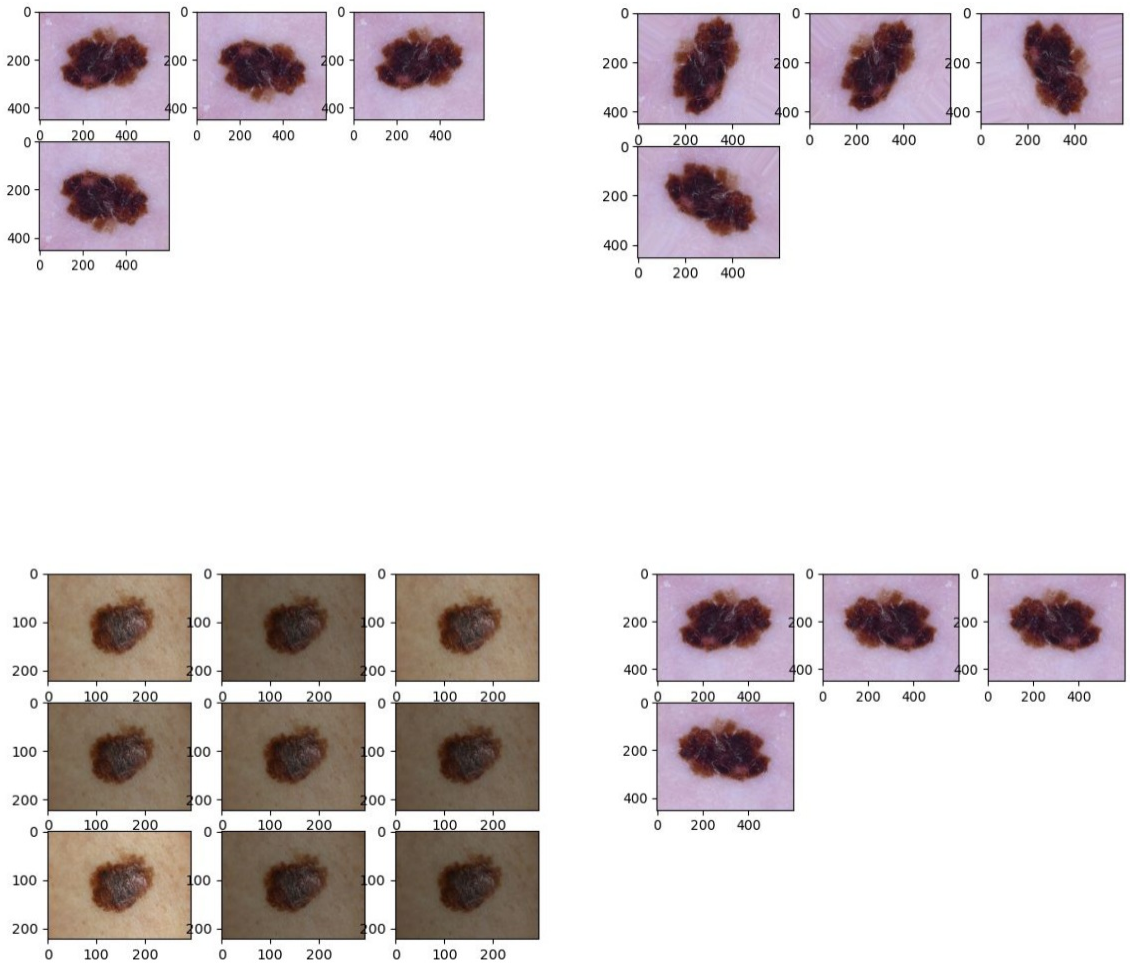
Image augmentation is a technique that is used to artificially expand the data-set. This is helpful when we are given a data-set with very few data samples. In the case of Deep Learning, this situation is bad as the model tends to over-fit when we train it on a limited number of data samples.

Image augmentation parameters that are generally used to increase the data sample count are zoom, shear, rotation, preprocessing_function and so on. Usage of these parameters results in generation of images having these attributes during training of Deep Learning models. Image samples generated using image augmentation, in general results in an increase of existing data samples set by nearly 3x to 4x times.

```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
import os

datagen1 = ImageDataGenerator(brightness_range=[0.3,1.0])

datagen2 = ImageDataGenerator(zoom_range=[0.5,1.0])

datagen3 = ImageDataGenerator(horizontal_flip = True)

datagen4 = ImageDataGenerator(rotation_range = 90)

datagen5 = ImageDataGenerator(vertical_flip = True)

```

We have used custom parameter values and used only the ones that would be helpful to the data we are dealing with, and did not use the ones that would distort the image too much by further removing the lesion features/information. Using this tool from keras, we have generated 3552 images for each class, and total dataset image files to 10,656 images.

The image below can explain the output of using image augmentation.



5.2.3 Model Training

Model training for deep learning is the most tiring job, since it takes a lot of time to train, to overcome this we have trained our models on Google Colab which allows users to use their integrated GPUs and TPUs for free, with almost 12GB of RAM. We started off by building a simple CNN architecture to train on our data.

- **ReLU :** ReLU stands for rectified linear unit, and is a type of activation function. ReLU is the most commonly used activation function in neural networks, especially in CNNs. ReLu is used because it is simple, fast, and empirically it seems to work well. We learned that training a deep network with ReLu tended to converge much more quickly and reliably than training a deep network with sigmoid activation.
- **SoftMax :** Softmax is often used in neural networks, to map the non-normalized output of a network to a probability distribution over predicted output classes. The standard softmax function is defined by the formula. The softmax activation is normally applied to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh, or another activation function. The reason why softmax is useful is because it converts the output of the last layer in your neural network into what is essentially a probability distribution.
- **Adam Optimizer :** Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.
- **SGD Optimizer :** Stochastic gradient descent is an iterative method for optimizing an objective function with suitable smoothness properties. It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient by an estimate. Especially in big data applications this reduces the computational burden, achieving faster iterations in trade for a slightly lower convergence rate.

- **Google Collab :** Google Colab is a free cloud service. We can develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.
- **VS Code :** Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes embedded Git and support for debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is highly customizable, allowing users to change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

2. CNN-basic

A typical CNN design begins with feature extraction and finishes with classification. Feature extraction is performed by alternating convolution layers with subsampling layers. Classification is performed with dense layers followed by a final softmax layer. For image classification, this architecture performs better than an entirely fully connected feed forward neural network. A basic CNN architecture contains the following.

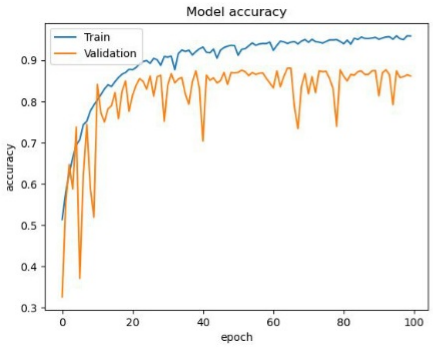
- **Filters** is the number of desired feature maps.
- **Kernel_size** is the size of the convolution kernel. A single number 5 means a 5x5 convolution.
- **Padding** is either 'same' or 'valid'. Leaving this blank results in padding='valid'.

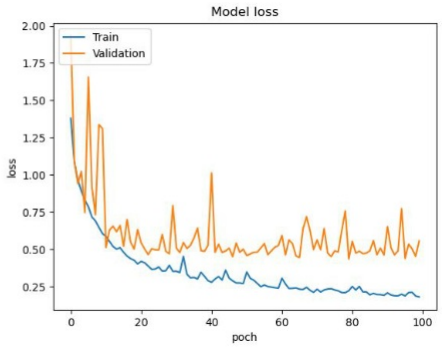
  If padding is 'valid' then the size of the new layer maps is reduced by kernel_size-1. For example, if you perform a 5x5 convolution on a 28x28 image (map) with padding='valid', then the next layer has maps of size 24x24. If padding is 'same', then the size isn't reduced.
- **Activation** is applied during forward propagation. Leaving this blank results in no activation.

We used ‘ReLu’ activation for every layer and Softmax in the end, we also used a drop out of 0.4(40%) to generalize data, thereby avoiding overfitting. Our architecture takes in an input_size of (28,28,1) ‘grayscale’ image and consists of

- Two Convolutional Layers with feature map 32x32 and kernel_size 3x3, with activation ‘ReLu’ and one Convolutional Layer with feature map 32x32, kernal_size 5x5 with stride 2.
- Two Convolutional Layers with feature map 64x64 and kernel_size 3x3, with activation ‘ReLu’ and one Convolutional Layer with feature map 64x64, kernal_size 5x5 with stride 2.
- Flatten layer followed by a fully connected layer, dense - 128, a dropout of 0.4, dense -3(Number of classes present).

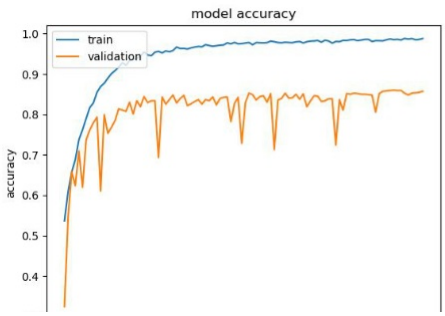
The following model is trained with batch_size = 32 and epochs = 100, gave us training accuracy of 0.97 and on validation gave 0.82.4. The learning curves are shown below,

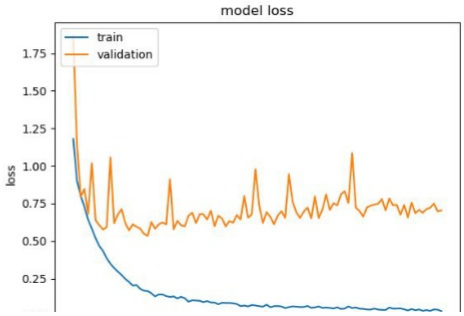




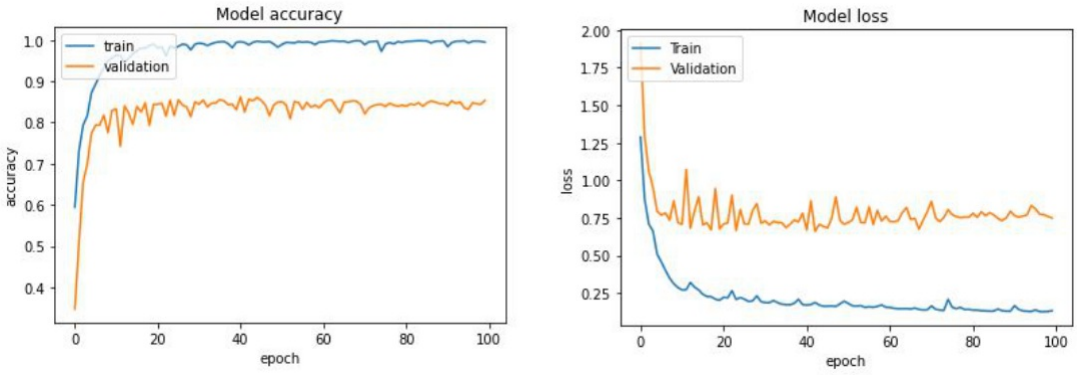
From the curves we see that the model trained too well on data and it over fitted. And the accuracy of validation has ups and downs. Next we tried to change the architecture a bit.

We added BatchNormalization after every layer and kernal_regularizer, bais_regularizer in every layer to fine tune the model. The results are as follows,





We see the fluctuations on validation accuracy and loss, to smoothen these fluctuations, we added two dropouts, one with 0.4 after two layers, and 0.5 before the final fully connected layer. This gave us better results, we used kernel regularizer=l2(0.001), bias regularized = l2(0.001). The accuracy after 100 epoches was 0.8647 and on training accuracy it was 0.9947.



But these results are not the best, we next trained with ILSVRC Models that were trained on the imagenet dataset. We used VGG16, InceptionResNetV2 and ResNet101. We try to train our data without imagenet weights and solely on their architectures to see the results and then use transfer learning method to fine tune the model and train it with imagenet pretrained weights.

2. Without Transfer Learning:

We have used ResNet101, VGG, and InceptionResNetV2 without transfer learning, which means weights as None. We created a pickle object file of our image data ready in our drive to avoid extracting data everytime we ran.

```
import pickle
X = pickle.load(open("/content/drive/My Drive/Skin Cancer DATA/equal/X_equal.pickle", "rb"))
y = pickle.load(open("/content/drive/My Drive/Skin Cancer DATA/equal/y_equal.pickle", "rb"))
```

1. **ResNet101- Without Weights:**

   We used keras.applications to use ResNet101 architecture, from the documentation provided by keras, the input images were extracted and resized to 224x224x3, since most of the imagenet models use this image format.

```
base_model = applications.ResNet101(weights= None, include_top=False, input_shape= (224,224,3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
predictions = Dense(3, activation= 'softmax')(x)
model = Model(inputs = base_model.input, outputs = predictions)
```

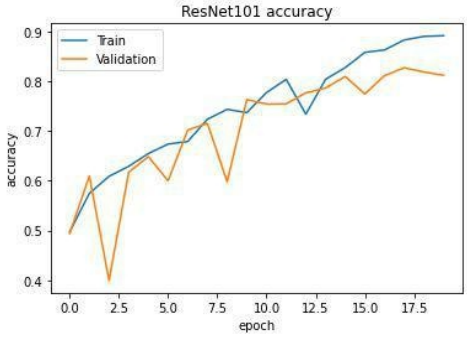
We used Stochastic gradient descent with learning_rate of 0.01, momentum of 0.9.

```
from keras.optimizers import SGD, Adam
sgd = SGD(lr=0.01, momentum=0.9, nesterov=False)
model.compile(optimizer= sgd, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

With ResNet's 101 layers, the Trainable params: 42,558,979 , Non-trainable params: 105,344. We trained the model with Google Colab with Google Compute Engine backend integrated GPU, which saved us a lot of time. With batch_size of 32, epochs 20, and validation_slipt 0.2, the training accuracy, validation accuracy, with their losses

```
Epoch 20/20
8516/8516 [==============================] - 153s 18ms/step - loss: 0.3045 - accuracy: 0.8911 - val_loss: 0.5634 - val_accuracy: 0.8116
```

TRAINING ACCURACY = **0.891** and VALIDATION ACCURACY = **0.8116** , which is an acceptable result, but the difference must be of a smaller value to be called an optimal model. The Learning curves also show the evidence of a little overfitting.



2. InceptionResNetV2 - Without Weights:

We used keras.applications to use InceptionResNetV2 architecture, from the documentation provided by keras, the input images were extracted and resized to 224x224x3, since most of the imagenet models use this image format.

```
base_model1 = applications.InceptionResNetV2(weights= None, include_top=False, input_shape= (224,224,3))

x = base_model1.output
x = Flatten()(x)
x = Dropout(0.4)(x)
predictions = Dense(3, activation= 'softmax')(x)
model = Model(inputs = base_model1.input, outputs = predictions)
```

We used Adam optimizer with a learning_rate of 0.0001

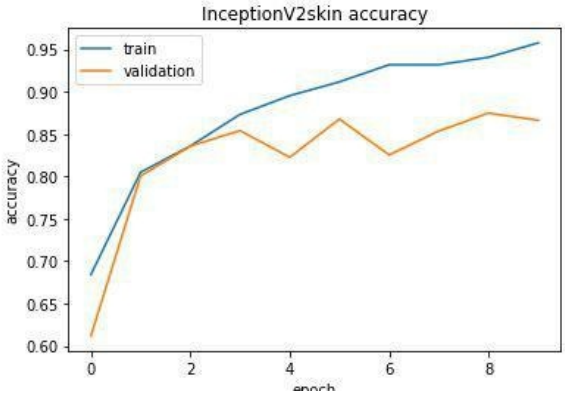
```
from tensorflow.keras.optimizers import SGD,Adam
adam = Adam(lr=0.0001)
model.compile(optimizer= adam, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

With InceptionV2's Architecture, Total params: 54,451,939, Trainable params: 54,391,395, Non-trainable params: 60,544. We trained the model with Google Colab with Google Compute Engine backend integrated GPU. With batch_size of 32, epochs 10, and validation_slipt 0.2, the training accuracy, validation accuracy, with their losses are as follows

```
Epoch 10/10
267/267 [==============================] - 97s 362ms/step - loss: 0.1210 - accuracy: 0.9575 - val_loss: 0.4921 - val_accuracy: 0.8661
```

TRAINING ACCURACY = **0.9575** and VALIDATION ACCURACY = **0.8661** , which is an acceptable result, but the difference must be of a smaller value to be called an optimal model.

The Learning curves also show the evidence of overfitting. With the difference being more than ResNet's.



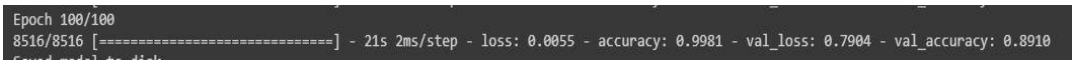
3. VGG16 - Without Weights:

We used keras.applications to use VGG16 architecture, from the documentation provided by keras, the input images were extracted and resized to 224x224x3, since most of the imagenet models use this image format. We used Adam optimizer with a learning_rate of 0.0001

```
from keras.optimizers import SGD, Adam
opt =Adam(learning_rate=0.0001)
model.compile(optimizer=opt, loss="sparse_categorical_crossentropy", metrics=['accuracy'])
```

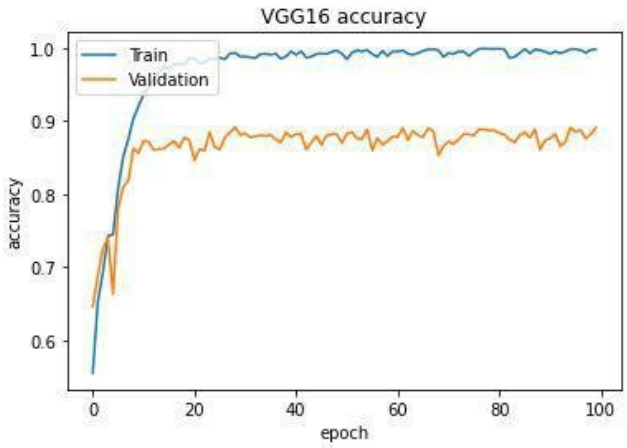
With VGG16's Architecture, Total params: 3,588,003, Trainable params: 3,588,003, Non-trainable params: 0. We trained the model with Google Colab with Google Compute Engine backend integrated GPU. With batch_size of 32,

epochs 100, and validation_slipt 0.2, the training accuracy, validation accuracy, with their losses are as follows



TRAINING ACCURACY = **0.9981** and VALIDATION ACCURACY = **0.8918** , which is an acceptable result, but the difference must be of a smaller value to be called an optimal model.

The Learning curves also show the evidence of overfitting. With the difference being more than InceptionV2's.



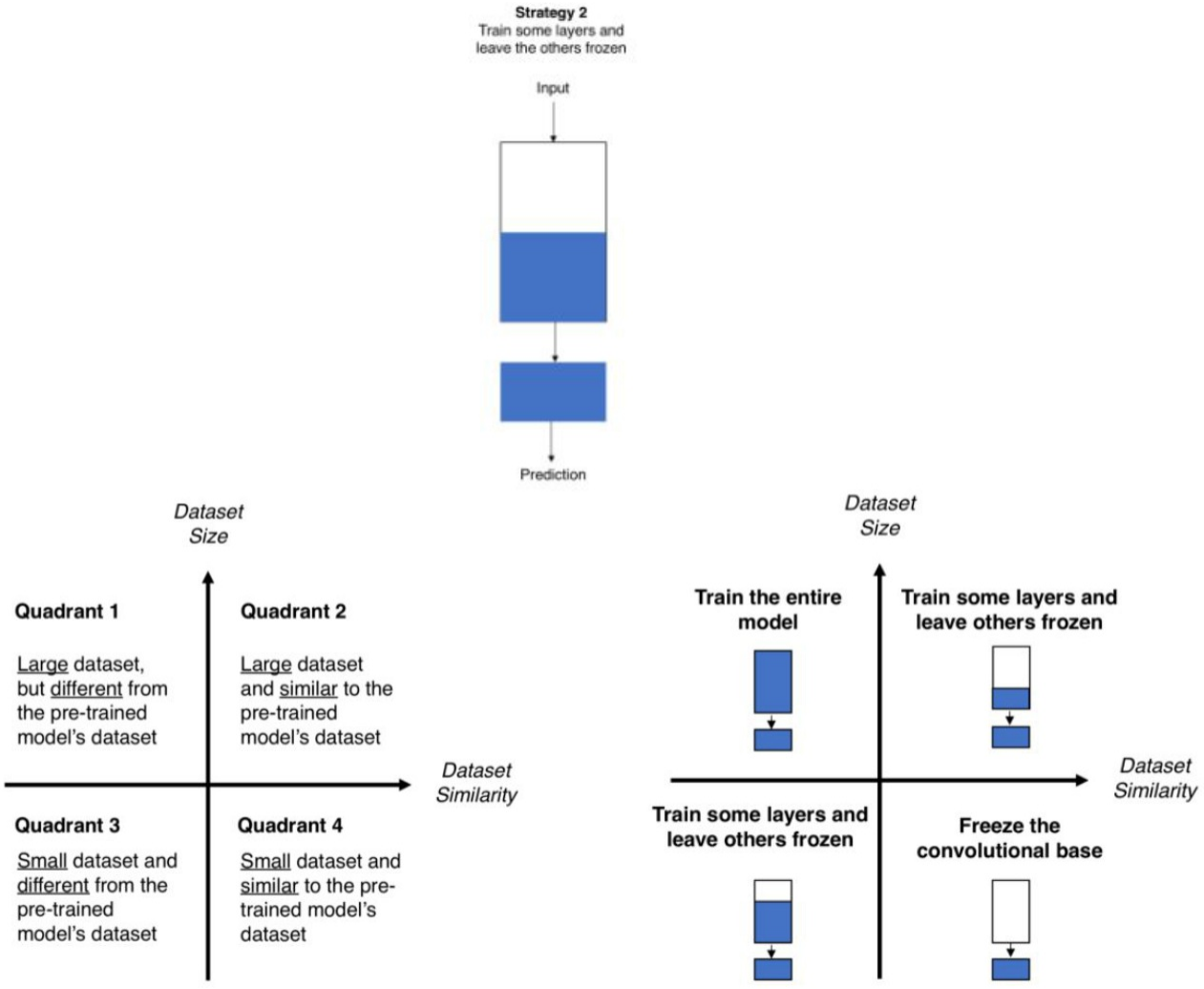
With the results of imagenet models without imagenet pre-trained weights, we see that the model's performance hasn't been satisfying. We then tried to train these models with imagenet trained weights, excluding the top layer. With the Transfer Learning quadrant principle.

3. With Transfer Learning:

With transfer learning, instead of starting the learning process from scratch, we start from patterns that have been learned when solving a different problem. This way we leverage previous learnings and avoid starting from scratch. There are basically three types of transfer learning --

1. Train the entire model
2. Train some layers and leave the others frozen
3. Freeze the convolutional base.

Out of these 3 types, we are going to pick the second way, this strategy is used when we have a small dataset, as 10k images is small compared to the 14million images of imagenet.



We see that there are no similarities between imagenet data and our dataset, so we conclude that our dataset is dissimilar to that of imagenet, taking into consideration the size, Quadrant 3 strategy suits best to our problem.

1. **VGG16 - With ImageNet Weights:**

We used keras.applications to use VGG16 architecture like mentioned before, from the documentation provided by keras, the input images were extracted and resized to 224x224x3, and the weights parameter is set to 'imagenet', and include_top as False, since most of the imagenet models use this image format.

```
vgg_model = applications.VGG16(weights='imagenet',
                               include_top=False,
                               input_shape=(224, 224, 3))


layer_dict = dict([(layer.name, layer) for layer in vgg_model.layers])

x = layer_dict['block2_pool'].output

x = Conv2D(filters=64, kernel_size=(3, 3), activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(3, activation='softmax')(x)
```

We used Adam optimizer with learning_rate of 0.0001, many articles have suggested its better to use a smaller learning rate during transfer learning.

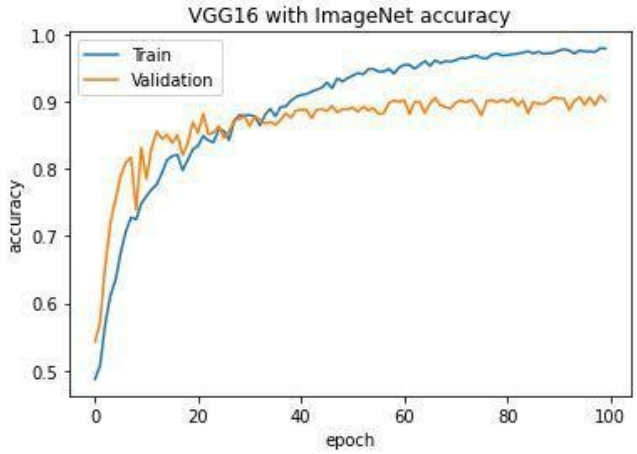
```
from keras.optimizers import Adam
adam = Adam(learning_rate = 0.0001)

custom_model.compile(loss='sparse_categorical_crossentropy',
                     optimizer=adam,
                     metrics=['accuracy'])
```

With VGG16's Architecture, Total params: 12,278,915, Trainable params: 12,278,915, Non-trainable params: 0. We trained the model with Google Colab with Google Compute Engine backend integrated GPU. With batch_size of 32, epochs 100, and validation_slipt 0.2, the training accuracy, validation accuracy, with their losses are as follows

```
8516/8516 [==============================] - 37s 4ms/step - loss: 0.0533 - accuracy: 0.9785 - val_loss: 0.6426 - val_accuracy: 0.9009
```

TRAINING ACCURACY = **0.9785** and VALIDATION ACCURACY = **0.9009** ,which is an acceptable result, but the difference value can be accepted but we cannot certainly call it an optimal model.

The Learning curves also do not show much of the evidence of overfitting. Compared with VGG16 without pre-trained weights.



We can see a slight deviation between epochs 25 and 35. but then the model at validation and training seemed to go smoothly.

2. InceptionResNetV2 - With ImageNet Weights:

We used keras.applications to use InceptionResNetV2 architecture like mentioned before, from the documentation provided by keras, the input images were extracted and resized to 224x224x3, and the weights parameter is set to ‘imagenet’, and include_top as False, since most of the imagenet models use this image format.

```
base_model1 = applications.InceptionResNetV2(
    weights= 'imagenet',
    include_top=False,
    input_shape= (224,224,3))

x = base_model1.output
x = Flatten()(x)
x = Dropout(0.4)(x)
predictions = Dense(3, activation= 'softmax')(x)
model = Model(inputs = base_model1.input, outputs = predictions)
```

We used Adam optimizer with learning_rate of 0.0001, many articles have suggested it's better to use a smaller learning rate during transfer learning.

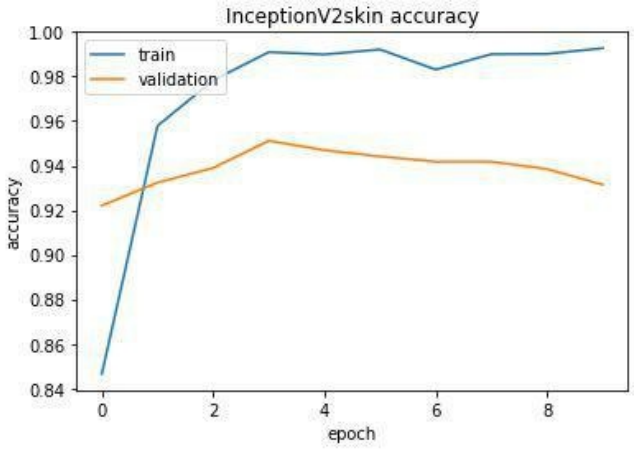
```
from tensorflow.keras.optimizers import SGD,Adam
adam = Adam(lr=0.0001)
model.compile(optimizer= adam, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

With InceptionResNetV2's Architecture, Total params: 54,451,939, Trainable params: 54,391,395, Non-trainable params: 60,544. We trained the model with Google Colab with Google Compute Engine backend integrated GPU. With batch_size of 32, epochs 10, and validation_slipt 0.2, the training accuracy, validation accuracy, with their losses are as follows

```
Epoch 10/10
267/267 [==============================] - 372s 1s/step - loss: 0.0232 - accuracy: 0.9927 - val_loss: 0.3728 - val_accuracy: 0.9314
Using TensorFlow backend.
```

TRAINING ACCURACY = **0.9927** and VALIDATION ACCURACY = **0.9314** ,which is an acceptable result, but the difference value can be accepted but we cannot certainly call it an optimal model. This has been the best result so far, and compared with the InceptionV2 model without using any pre-trained weights, we can say the transfer learning method outperformed.

The Learning curves also do not show much of the evidence of overfitting. Compared with InceptionV2 without pre-trained weights. This plotting might look sloppy, but Colab, automatically reduces the Y-axis coordinates to give a closer view.



3. ResNet101 - With ImageNet Weights:

We used keras.applications to use ResNet101 architecture like mentioned before, from the documentation provided by keras, the input images were extracted and resized to 224x224x3, and the weights parameter is set to 'imagenet', and include_top as False, as we are using the second strategy of transfer learning, since most of the imagenet models use this image format.

```
base_model = applications.ResNet101(weights= 'imagenet', include_top=False, input_shape= (224,224,3))
```

```
Cloning into 'resnet_utils.py'...
fatal: repository 'https://github.com/tensorflow/models/blob/master/research/slim/nets/resnet_utils.py/' not found
Downloading data from https://github.com/keras-team/keras-applications/releases/download/resnet/resnet101_weights_tf_dim_ordering_tf_kernels_notop.h5
171450368/171446536 [==============================] - 7s 0us/step
```

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.4)(x)
predictions = Dense(3, activation= 'softmax')(x)
model = Model(inputs = base_model.input, outputs = predictions)
```

We used Adam optimizer with learning_rate of 0.0001, many articles have suggested it's better to use a smaller learning rate during transfer learning.

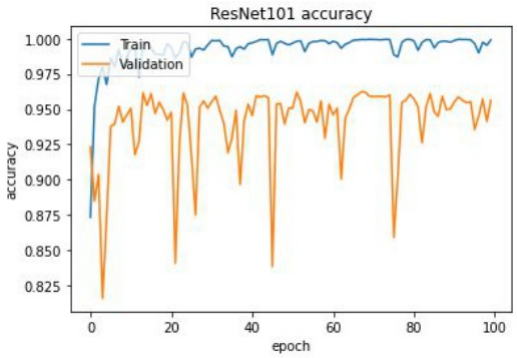
```
from keras.optimizers import SGD, Adam
adam = Adam(lr=0.0001)
model.compile(optimizer= adam, loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

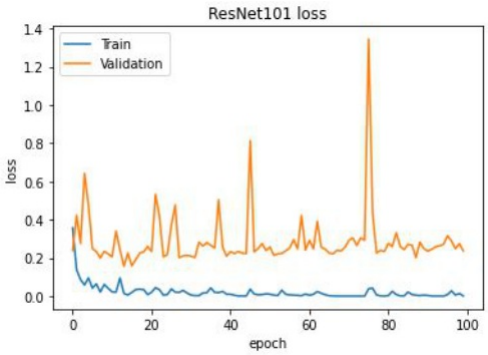
With ResNet101's Architecture,Total params: 42,664,323, Trainable params: 42,558,979, Non-trainable params: 105,344. We trained the model with Google Colab with Google Compute Engine backend integrated GPU. With batch_size of 32, epochs 100, and validation_slipt 0.2, the training accuracy, validation accuracy, with their losses are as follows

```
Epoch 100/100
8516/8516 [==============================] - 160s 19ms/step - loss: 0.0017 - accuracy: 0.9992 - val_loss: 0.2373 - val_accuracy: 0.9563
```

TRAINING ACCURACY = **0.9992** and VALIDATION ACCURACY = **0.9563** ,which is an acceptable result and almost an optimal model,but the difference value is accepted. This has been the best result so far with training and testing accuracies having very less difference, which means that the model has trained well on the training data of 8516 images, generalized well without overfitting and it can 95.63% accurately predicts to new data, and compared with the ResNet101 model without using any pre-trained weights, we can say the transfer learning method outperformed the traditional approaches.

The Learning curves also do not show much of the evidence of overfitting. Compared with ResNet101 without pre-trained weights. This plotting might look sloppy, but Colab, automatically reduces the Y-axis coordinates to give a closer view.





5.2.4 FrontEnd development

- We used Node.js and Express server as a middleware for our project's deployment. As there is a way to deploy models using Node.js and writing wrapped javascript code in jQuery we can deploy our deep learning models on web apps.
- Tensorflow.js allows the browser to load the trained model, and allows us to use the tf methods in the jQuery.
- Before we can deploy the saved model of ResNet101 which has been trained by pretrained weights, we must convert it into a Tensorflow.js model, the present model is saved as a keras model and thus we are required to change the model to tfjs.
- We can do this by downloading tensorflow.js and using the command tensorflow js converter.

```
E:\Skin Cancer Classification\project>tensorflowjs_converter --input_format keras finalResNet101Skinimg.h5 models/tfjs/resnet
```

```
tensorflowjs_converter --input_format keras \
                       path/to/my_model.h5 \
                       path/to/tfjs_target_dir
```

- We must write our package.json file for the packages we need for this project and save it in the local-server folder.

skin_web > local-server > {} package.json >

```
{
  "name": "tensorflowjs",
  "version": "1.0.0",
  "dependencies": {
    "express": "latest"
  }
}
```

- By opening our terminal from VS Code or command prompt, we must enter npm install.’npm’ stands for Node Package Manager.
- We can see the folders of the packages written in package.json installed.

- We must create a new javaScript file called as server.js, which enables the server and import express and port number to the localhost to run in by entering in terminal

```
local-server>node ./server.js
```

```
const express = require("express");
const app = express();

app.use(function(req, res, next){
    console.log(`${new Date()} - ${req.method} request for ${req.url}`);
    next();

});
app.use(express.static("../static"));

app.listen(81, function(){
    console.log("Serving static on 81")

});
```

- The terminal would print “Serving static on 81”, where 81 is the unused port. We then opened our html file with the localhost and port id. “localhost:81/skin.html”
- We must copy our tensorflow.js model to the static folder. The static folder contained our html file and two javascript files that are used for the function calling from the html.
- We created a cancer_classes.js, to store a dictionary variable with our class names and their specified values.

```
skin_web > static > JS cancer_classes.js > CANCER_CLASSES
1
2   const CANCER_CLASSES ={
3     0: 'Basal Cell Carcinoma',
4     1: 'Melanoma',
5     2: 'Squamous Cell Carcinoma',
6
7   }
```

- The second file is predict.js which holds the server side script that is going to execute when an event occurs in the html static page.

- We used bootstrap framework for our styling, the whole web page is now divided as rows and columns which would be easier to specify the sections of every element in the browser page.
- The html file contains mainly four divisions ,Ids as follows
    1. model-selector
    2. image-selector
    3. predict-button
    4. prediction-list
- The model-selector allows the user to select the model, it is loaded by using loadModel method which returns a promise response, into a model variable, we write this code as an async function, and we use await for the response. To let the user know that model is loading and has successfully loaded we make use of a loading bar.

```
let model
async function loadModel(name) {

    $(".progress-bar").show();
    model = await tf.loadModel(saveLocation + '/model.json');
    $(".progress-bar").hide();
}
```

- The image-selector would enable the user to upload an image file when he clicks the upload button.

```
$("#image-selector").change(function () {
    let reader = new FileReader();
    reader.onload = function () {
        let dataURL = reader.result;
        $("#selected-image").attr("src", dataURL);
        $("#prediction-list").empty();
    }
    let file = $("#image-selector").prop("files")[0];
    reader.readAsDataURL(file);
});
```

- The predict-button when clicked would show the predictions in the prediction-list section.

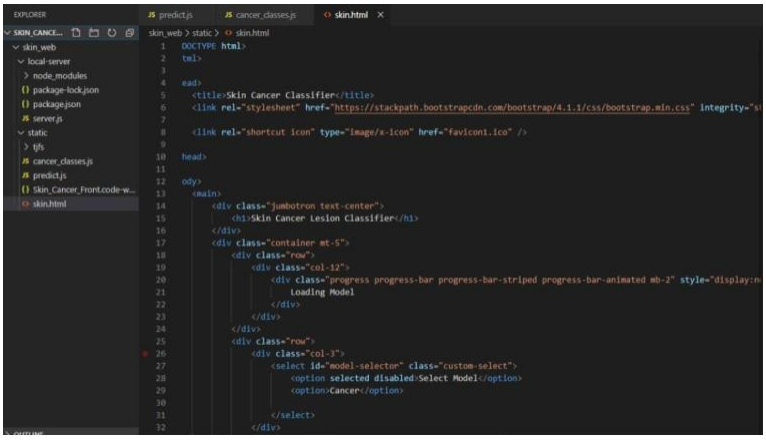
```
let predictions
$("#predict-button").click(async function () {
    let image = $("#selected-image").get(0);
    let modelName = $("#model-selector").val();
    let tensor = tf.fromPixels(image)
    .resizeNearestNeighbor([224,224])
    .toFloat();
    let offset = tf.scalar(127.5);
    tensor = tensor.sub(offset)
    .div(offset)
    .expandDims();
    predictions = await model.predict(tensor).data();
    let top5 = Array.from(predictions)
        .map(function (p, i) {
            return {
                probability: p,
                className: CANCER_CLASSES[i]
            };
        })

    $("#prediction-list").empty();
    top5.forEach(function (p) {
        $("#prediction-list").append(`<li>${p.className}: ${p.probability.toFixed(4)}</li>`);
    });
});
```

This is the whole description of the frontend, the frontend screens are explained further with screenshots in testing and validation section. Section - 6.

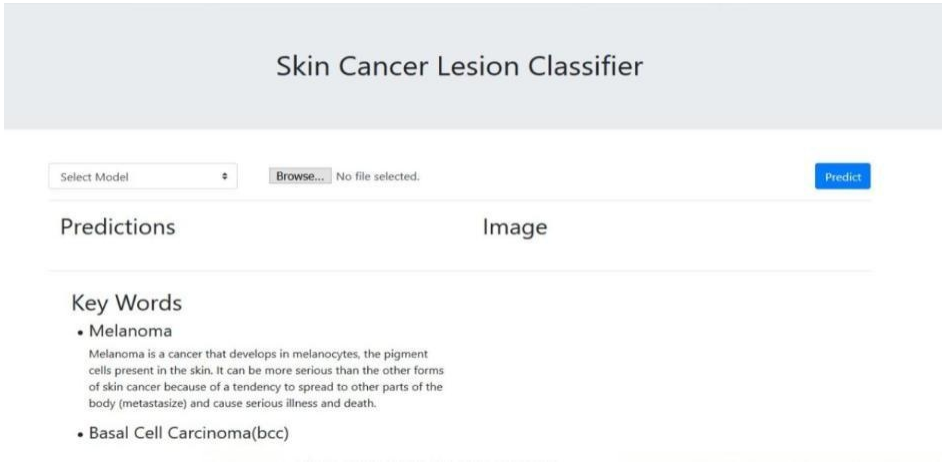
6.TESTING AND VALIDATION

6.1 UNIT TESTING

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In our web application, we can visually check only one module and that is the home page.

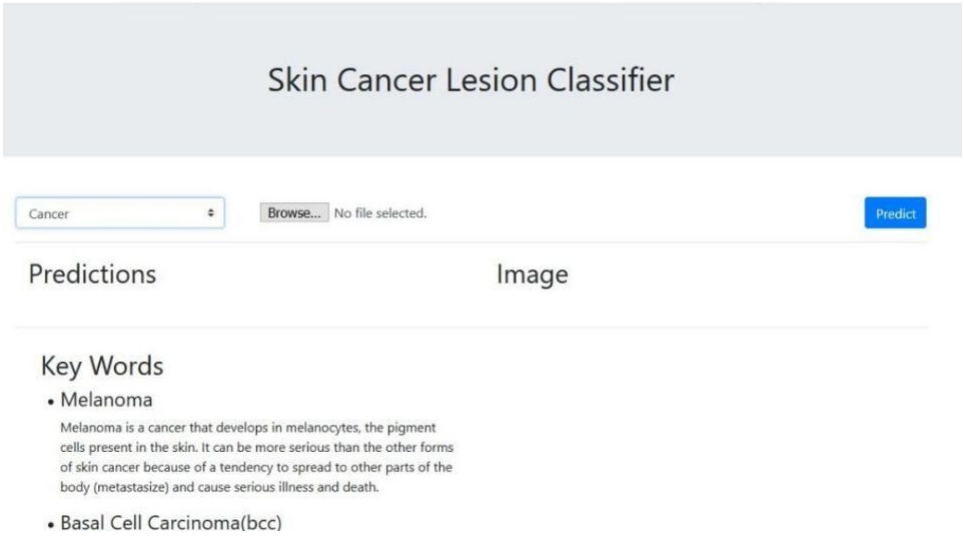


In the above screen unit testing is being performed and after running the code, it is successfully executed and the following screen is displayed to the user.


Skin Cancer Lesion Classifier
Select Model
Browse...
No file selected.
Predict
Predictions
Image
Key Words
• Melanoma
Melanoma is a cancer that develops in melanocytes, the pigment cells present in the skin. It can be more serious than the other forms of skin cancer because of a tendency to spread to other parts of the body (metastasize) and cause serious illness and death.
• Basal Cell Carcinoma(bcc)

6.2 FUNCTIONAL TESTING

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. In image processing application, there is only one fault,i.e., if the file is not selected.

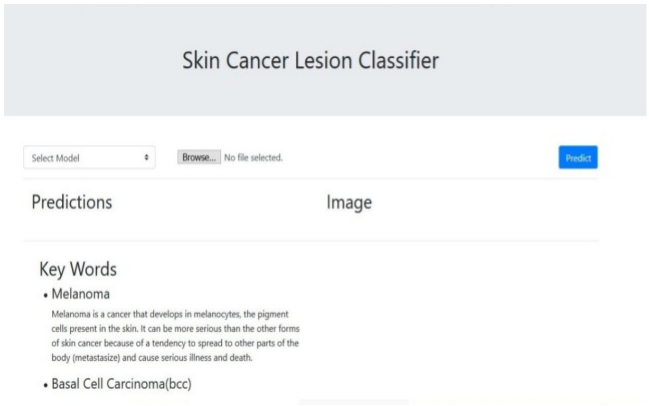


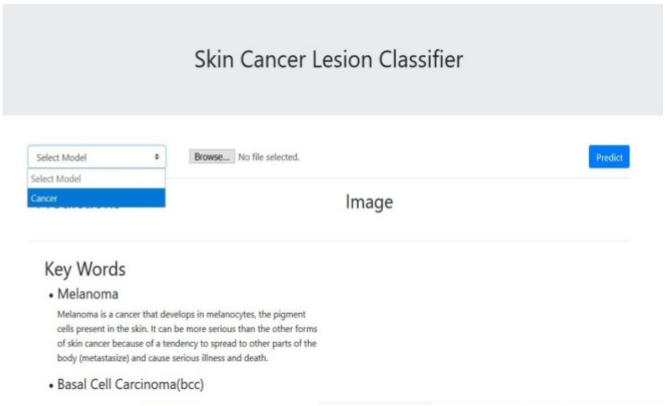
6.3 SYSTEM TESTING

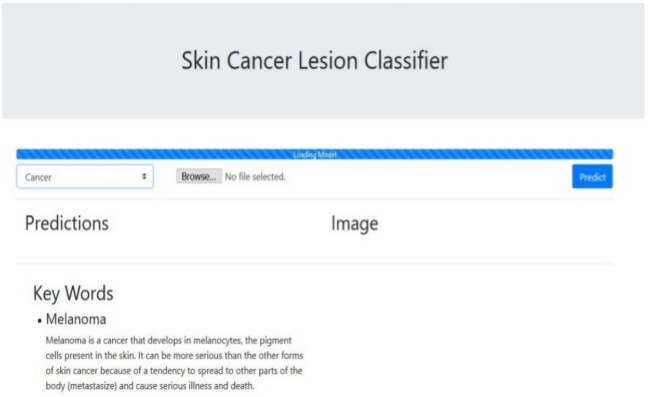
System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

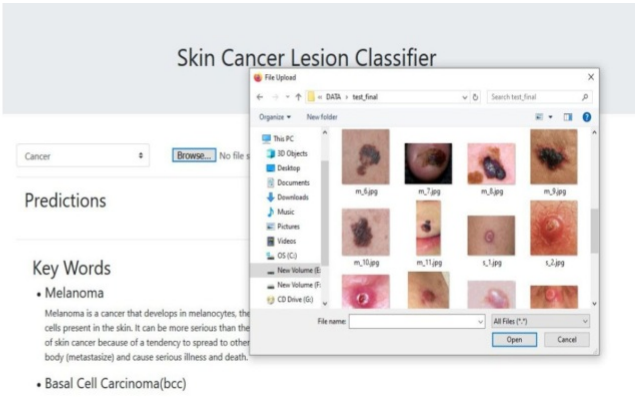
Consider the following test case,

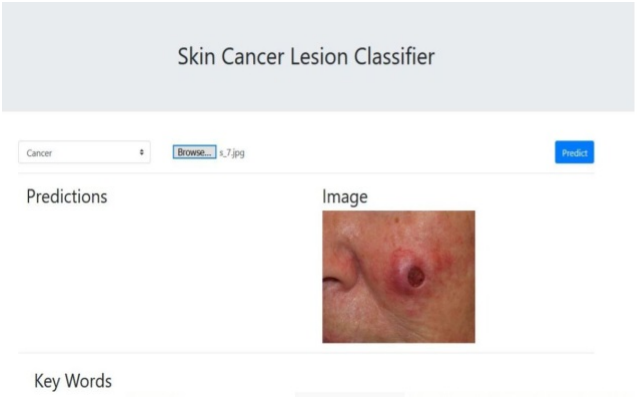
Test case : Let us consider any image of skin lesion and start the image processing to find out the class of the skin cancer

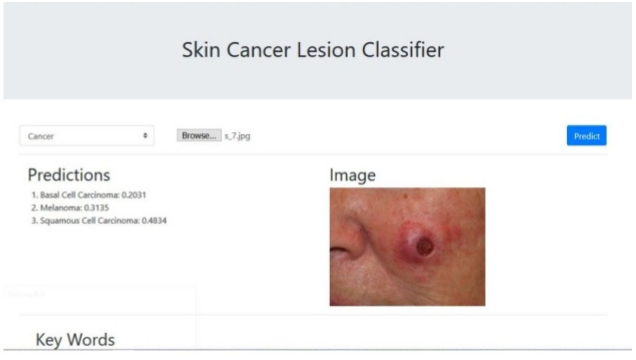












7.CONCLUSION

The project's key goal has been to predict an image of skin lesion to its type with highest accuracy, which has been made possible by the Transfer Learning approach. Several architectures have been trained with different learning rates, epochs and batch sizes, however ResNet101 architecture with Imagenet weights has given us the best accuracy to identify the type of a skin cancer lesion ever to be published or recorded, which is 95.63% with training accuracy of 99.92%, we don't see the model overfitting in this case. Also an ensemble approach which has been said to give better results is implemented by using basic voting mechanism which was written in Python. We have tried to deploy this model as a web application, but we got few errors with the express server and tensorflow.js version. Our second goal of understanding how these deep neural networks work and knowing how to implement them and fine tune them to get better results is achieved.

7.1.FUTURE SCOPE

- Since the project identifies the cancer lesion type, this can be used by the dermatologists and patients as well.
- Dermatologists, before sending the clinical image for the biopsy test can run the lesion through the model and based on the results, they can focus on validating if the lesion belongs to the class that the model has specified. This would cut down the delay of 2 to 3 weeks for the biopsy results.
- If the model predicts inaccurately in certain conditions then the model can be trained with the wrongly classified images to better learn the features it missed in the first learning.

- For better reliability on the model, because we cannot solely trust a machine for final prediction and take the result of the machine as a final answer, we can find the performance of dermatologists and the machine, by providing the dermatologists and the model to classify a set of images and validate them with their predictions and evaluate the performance of the model over an experienced doctor.Machines can find and remember things we humans cannot, so there is a chance that our model may excel in this test.

8.REFERENCES

[1] Uzma Bano Ansari, Tanuja Sarode, Skin Cancer Detection Using Image Processing, International Research Journal of Engineering and Technology (IRJET), Apr - 2017, Mumbai, India.

[2] Xin He, Shihao Wang, Shaohuai Shi, Zhenheng Tang, Computer-Aided Clinical Skin Disease Diagnosis Using CNN and Object Detection Models, arXiv:1911.08705v1 [eess.IV] ,Nov 2019, China.

[3] Enakshi Jana, Dr.Ravi Subban, S. Saraswathi, Research on Skin Cancer Cell Detection using Image Processing, IEEE-International Conference on Computational Intelligence and Computing Research (ICCIC), Dec-2017, India.

[4] Li-sheng Wei, Quan Gan, Tao Ji, Skin Disease Recognition Method Based on Image Color and Texture Features, Aug 2018, China.

[5]Skin Cancers, https://www.who.int/uv/faq/skincancer/en/index1.html

[6] Skin Cancer: Symptoms, Causes and Treatment
https://health.usnews.com/conditions/cancer/skin-cancer

[7]How long can you have cancer without knowing it? - healthline
https://www.healthline.com/health/how-long-can-you-have-cancer-without-knowing

[8]How is Skin cancer diagnosed? https://skincancer.net/diagnosis/

[9]Test to diagnose | Skin Cancer
https://www.cancerresearchuk.org/about-cancer/skin-cancer/getting-diagnosed/tests-diagnose

[10]What are symptoms and signs of Skin cancer?

https://www.cancercenter.com/cancer-types/skin-cancer/symptoms

[11]How is Skin Cancer Diagnosed - https://skincancer.net/diagnosis/

[12]Skin Cancer - https://www.cancercenter.com/cancer-types/skin-cancer

[13]Tests to diagnose Skin Cancer -

https://www.cancerresearchuk.org/about-cancer/skin-cancer/getting-diagnosed/tests-diagnose

[14]A Comprehensive Guide to Convolutional Neural Networks -

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[15] Transfer Learning from pre-trained models|Towards Data Science -

https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 [16]Imagenet tree view - http://image-net.org/explore

[17]An Overview of ResNet and its variants - https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[18] CNN Architecture Series -- VGG-16 with Implementation -- https://medium.com/datadriveninvestor/cnn-architecture-series-vgg-16-with-implementation-part-i -bca79e7db415

[19] A simple guide to the versions of inception networks -

https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[20]Run JavaScript Everywhere - https://nodejs.dev/

[21]Tensorflow.js | Machine learning for JavaScript Developers -

https://www.tensorflow.org/js

[22]Express/Node Introduction -

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

[23]jQuery Introduction - https://www.w3schools.com/jquery/jquery_intro.asp

[24]The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions -

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T

[25]ISIC Archive - https://isic-archive.com/

[26]What selenium? Introduction to Selenium -

https://www.journaldev.com/25395/what-is-selenium-introduction-to-selenium

[27]How to configure Image augmentation in keras? - https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-de ep-learning-neural-networks/

[28]Keras Optimizers - https://keras.io/api/optimizers/

[29]Transfer Learning from pre-trained models - https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

[30]Tensoflow.js Deep Learning with javascript - DeepLizard -

https://www.youtube.com/playlist?list=PLZbbT5o_s2xr83l8w44N_g3pygvajLrJ-