Project Report

on

Early Stage Skin Cancer Detection

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

Session 2022-23
in

CSE

By
AMRIT ANAND
2100320100021

Under the guidance of

DR. KADAMBRI AGARWAL

ABES ENGINEERING COLLEGE, GHAZIABAD

AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)

Project Report

on

Early Stage Skin Cancer Detection

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE (font size 20)

Session 2021-22
in

CSE

By
AMRIT ANAND
2100320100021

Under the guidance of

DR. KADAMBRI AGARWAL

ABES ENGINEERING COLLEGE, GHAZIABAD (font 16)

AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)

STUDENT'S DECLARATION

I / We hereby declare that the work being presented in this report entitled "EARLY STAGE SKIN CANCER DETECTION" is an authentic record of my / our own work carried out under the supervision of Dr. /Mr. /Ms. "KADAMBRI AGARWAL"

The matter embodied in this report has not been submitted by me / us for the award of any other degree.

Dated: 17/02/2023

Signature of students(s)
Amrit Anand
Department: CSE

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Signature of Supervisor
(Name)
(Designation)
(Computer Science & Engineering Department)

CERTIFICATE

This is to certify that Project Report entitled “EARLY STAGE SKIN CANCER DETECTION” which is submitted by Amrit Anand in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, formerly Uttar Pradesh Technical University is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Supervisor **Project Coordinator**

HOD’s Signature

Date

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Dr Kadambri Agarwal, Department of Computer Science & Engineering, ABESEC Ghaziabad for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Professor (Dr.) Divya Mishra, Head, Department of Computer Science & Engineering, ABESEC Ghaziabad for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:
Name :
Roll No.:
Date :

Signature:
Name :
Roll No.:
Date :

ABSTRACT

Skin cancer is a life-threatening disease that affects millions of people worldwide, with its various forms, presenting unique challenges for early detection and treatment.
Melanoma, the most unpredictable form of skin cancer, is known to be the most aggressive.
Early detection is critical in treating skin cancer, and it has been shown that detecting melanoma cancer at an early stage increases the likelihood of successful treatment. However, accurately diagnosing skin cancer can be a challenging task for medical professionals.
In this study, we present a method that uses image pre-processing and Sequential CNN as well as various pre-trained models like VGG16 ResNet50 to aid in early stage skin cancer detection. Our approach involves processing the patient's skin lesion image by removing noise and irrelevant background using various filtering and transformation techniques. The preprocessed image is then analyzed by our model, which classifies it as either a normal skin lesion (benign) or a melanoma cancer lesion (malignant) with a high degree of accuracy.
The proposed method has the potential to revolutionize the way dermatologists diagnose and treat skin cancer by providing a easy to use and cost-effective alternative to traditional diagnostic methods.
In conclusion, we used various machine learning techniques for the diagnosis of skin cancer. The proposed method can help medical professionals to accurately detect and diagnose skin cancer, thereby improving patient outcomes and reducing the overall burden of this disease on individuals and society.

TABLE OF CONTENTS

Page

CHAPTER 1

INTRODUCTION

Skin cancer is a life-threatening disease that affects millions of people worldwide Early detection is critical in treating skin cancer, and it has been shown that detecting melanoma cancer at an early stage increases the likelihood of successful treatment.

1. Problem Introduction

1.1. Motivation

At some point, we have all likely wondered if a mole or skin rash was a cause for concern. According to Google, there are over ten billion yearly searches related to skin conditions, and that seemingly normal mole could potentially be an early sign of skin cancer. Skin cancer is now considered one of the most dangerous forms of cancer for humans and is categorized into various types, including Melanoma, Basal cell carcinoma, and Squamous cell carcinoma. While most skin cancers do not spread to other organs, Melanoma and Merkel cell carcinoma have a higher chance of spreading. Detecting Melanoma cancer early can increase the chances of successful treatment. The 21st century has witnessed significant developments in technology, leading to the emergence of Telemedicine, a modern technology that supports remote health care. Dermatology, which relies heavily on visual examination, is a suitable field for Telemedicine. An app that utilizes AI and the high-resolution cameras on smartphones can provide users with a range of possible diagnoses for skin conditions, making it a more reliable alternative to self-diagnosing.

1.2. Project Objective

- To build binary class CNN model using transfer learning with VGG16 and a Sequential CNN model.
- To build an application for users to check their moles, if they are skin cancer ornot

2. Related Previous Work

Ref	Description	Detection Accuracy	Scope for improvement
[1]	• Two layered cnn is used for melanoma classification. • Illumination Correction, Mask Generation, Gaussian Filter is applied to remove noise issues. • Cropping and rotation is used for data augmentation	• With 81% accuracy	• Deep layered network may be used for increased results. • Training will get improved by collecting more data • Cancer detection can be implemented to localize the position of cancer part
[2]	• Feature extraction through normalized symmetrical GLCM. Texture features are extracted from each of the four classes. • Multi-Class Support vector machine used for classification purpose. It classifies the given data set into one of the four skin cancer classes.	• 81.43%. accurate	• GLCM works only on gray level image matrix to capture most feature such as contrast, mean, energy, homogeneity
[3]	• AlexNet used for feature extraction • SVM worked as a classifier.	• 94.17% accuracy for BCC • 95.1% accuracy for SCC • 98.9% accuracy for AK	• No further improvements could be found
[7]	• Applied many pre-processing techniques, Dull Razor Filter to remove hair, obtaining the grayscale of the image, contrast enhancement and median filter to remove noise • The infected area was segmented from the rest of the skin according to its binarized gray level by applying Maximum Entropy. Thresholding • GLCM implemented to extract features from processed image	• 86.66% accurate	• System should detect other skin cancer classes too

CHAPTER 2

LITERATURE SURVEY

1. Dataset

1.1. Dataset Description

The dataset used in this project is titled "Skin Cancer: Malignant vs Benign" and was created by Fanconic on Kaggle[8]. The dataset consists of 3297 images of skin lesions, which were collected from patients with both malignant and benign skin cancer diagnoses.

Each image is labeled with a corresponding binary classification indicating whether the skin lesion is malignant or benign.

The images are of varying sizes and have different levels of color saturation and contrast. To help address this variability, each image waspreprocessed to a standard 224x224 pixel size. In addition, each image was RGB-normalized and centered on zero mean.

It is worth noting that this dataset is a publicly available resource for skin cancer research, but it is important to acknowledge that there may be some limitations to its use. For example, the dataset is relatively small and may not be fully representative of the diversity of skin cancer cases in general. It is also possible that the dataset may not generalize well to otherdatasets, particularly those collected using different imaging technologies or from different populations.

1.2. Dataset Distribution

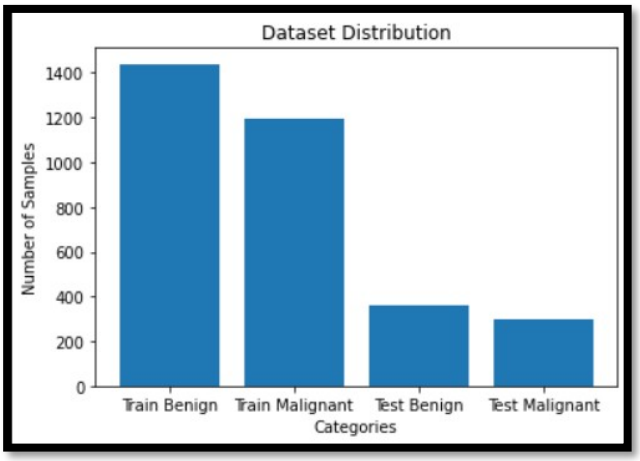

Fig 5: Dataset Distribution

1.3. Images from Dataset

Fig 1: 97

Fig 2: 77

Fig 3: 120

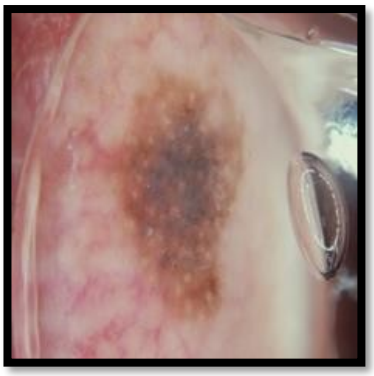

Fig 4: 119

2. Python Modules Used

2.1. Pandas, for data manipulation and analysis.

2.2. Glob module provides a convenient way to search for files that match a specified pattern. It allows you to find files based on their names or file types, and can be useful for working with large sets of data files.

2.3. Scikit-learn,for machine learning. Provides wide range of algorithms for classification and other tasks, as well as tools for feature selection, and model evaluation.

2.4. TensorFlow for building and training machine learning models. It provides a wide range of tools and APIs for building and deploying neural networks, as well as support for distributed computing and GPU acceleration.

2.5. Matplotlib is a Python library for creating visualizations and plots. It is used for creating line charts, scatter plots, bar charts, and other types of visualizations, as well as support for customizing the appearance of plots.

2.6. The os module provides a way to interact with the operating system from Python. It allows you to perform operations such as creating and deleting files, navigating directories, and setting environment variables.

2.7. Keras is high-level neural networks API. It also provides pre-trained models.

3. Image Data Augmentation

Data augmentation is a technique used to artificially increase the size of a training dataset by generating additional training examples from existing ones. This is particularly useful when working with convolutional neural networks because they are capable of learning from image data, and data augmentation can be used to create variations of images that allow the network to learn more general features and improve its ability to recognize new images.

Using Keras, you can fit models with augmented image data using the ImageDataGenerator class. The library enables automatic implementation of data augmentation during model training. The ImageDataGenerator class is used to achieve this.

4. Convolutional Neural Network

CNN is a type of artificial neural network that is specifically designed for image recognition and processing tasks. They are used to learn features from the input images and then classify them into different categories based on the features learned. CNNs are preferred over ANNs as ANNs require very high computational power and their accuracy is quite low as compared to CNNs.

The key idea behind CNNs is to use a sequence of convolutional layers and pooling layers to extract and process features from the input images. After several convolutional and pooling layers, the features extracted from the image are flattened into a one-dimensional vector, which is then passed through one or more fully connected layers to perform the final classification or regression task.

The main advantages of CNNs are their ability to automatically learn and extract relevant features from the input images, and their ability to generalize well to new

images. They have been used in wide range of applications such as object recognition, image segmentation, and even NLP.

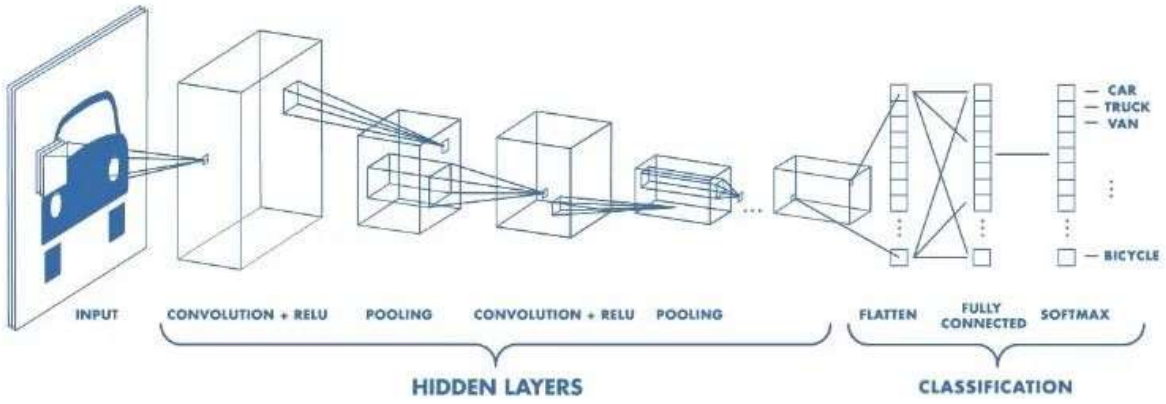

Fig 6: Architecture of CNN

In a Convolutional Neural Network (CNN), there are several types of layers used for processing image data:

1. Convolutional Layers: These layers are used to extract features from the input image by applying a set of filters or kernels. Each filter slides over the input image, and a convolution operation is performed to produce a feature map that highlights the presence of the filter's learned features. Multiple filters are used to produce multiple feature maps, which are then fed to the next layer.
2. Pooling Layers: These layers are used to reduce the size of the feature maps by summarizing a small region of the map (e.g. taking the maximum value). This downsampling process reduces the amount of computation required in subsequent layers and can also help to reduce overfitting.
3. Activation Layers: These layers apply a non-linear activation function to the output of the previous layer, which introduces non-linearity into the network and allows it to learn complex patterns and relationships in the data.

4. Batch Normalization Layers: These layers normalize the output of the previous layer to reduce internal covariate shift, which can help to speed up training and improve the overall performance of the network.
5. Dropout Layers: These layers randomly drop out a portion of the neurons in the network during training, which can help to prevent overfitting.
6. Flatten Layers: These layers flatten the output of the previous layer into a one-dimensional vector, which is then fed to one or more fully connected layers.
7. Fully-Connected Layers: These layers perform the final classification or regression task by mapping the input features to the output classes using a set of weights and biases.

5. VGG16

VGG16 is a convolutional neural network architecture which was used to win ILSVR(Imagenet) competition in 2O14. The network contains 16 layers of convolutional and fully connected layers, hence the name "VGG16".

The architecture of VGG16 is characterized by its deepness and use of small 3x3 convolutional filters in each layer. The small filter size allows the network to capture more local features of the input image, and the deep architecture enables the network to learn more complex and abstract representations of the input. This network is a pretty large network and it has about 138 million parameters.

Since its introduction, VGG16 has become a popular pre-trained model for a variety of computer vision tasks. Its architecture has also served as a basis for the development of many other popular CNN models.

6. Transfer Learning

Transfer learning is a machine learning technique that involves taking a pre-trained model and using it as a starting point for a new, related task. The idea is that the pre-trained model has already learned to recognize a large number of features and patterns in a particular domain, and this knowledge can be leveraged to accelerate learning on a new task.

Transfer learning with VGG16 can be very effective for tasks that involve classifying images into similar categories to those in the original dataset. This is because the pre-trained model has already learned to recognize many low-level features and high-level concepts that are common to many types of images. By starting with these learned weights, the model can more quickly learn to recognize the new categories, as it does not need to learn all of the low-level features from scratch.

To use transfer learning with VGG16, the pre-trained model is first downloaded and its layers are frozen, meaning that their weights are kept fixed and not updated during training. A new fully connected layer is then added on top of the frozen layers, and this new layer is trained to classify the new dataset.

The new fully connected layer is typically initialized with random weights and trained using the new dataset, with the goal of learning to classify the new images into the desired categories. The frozen layers act as feature extractors, which can be used to extract high-level features from the images. The extracted features are then passed through the fully connected layer to make predictions.

CHAPTER 3

METHODOLOGY

1. System Design

1.1. System Architecture

1. Our proposed method performs binary classification of benign and malignant skin lesion images using transfer learning with pre-trained VGG16 model.

2. The dataset contains images of skin lesions that are either benign or malignant. The images are split into training and a testing set, and each set has a directory for benign images and a directory for malignant images.

3. Our method uses the ImageDataGenerator class from Keras to generate batches of augmented image data during training.

4. We define a base model with several convolutional and dense layers, and then define a pre-trained VGG16 model from the Keras library. The VGG16 model is then added on top of the base model with additional dense layers, and the weights of the VGG16 layers are frozen.

5. The model is then compiled using the Adam optimizer and binary cross-entropy loss function. An early stopping callback is defined to stoptraining if the validation loss does not improve for five epochs.

6. Finally, model is trained using the fit() method and the training and testing data generators, with a maximum of 100 epochs. The results are then plotted

Overall, this method builds a ml model for skin lesion classification using transfer learning with the VGG16 model. The dataset is preprocessed and augmented, and the model is trained with early stopping and other callbacks to optimize performance.

1.2. Flow Charts

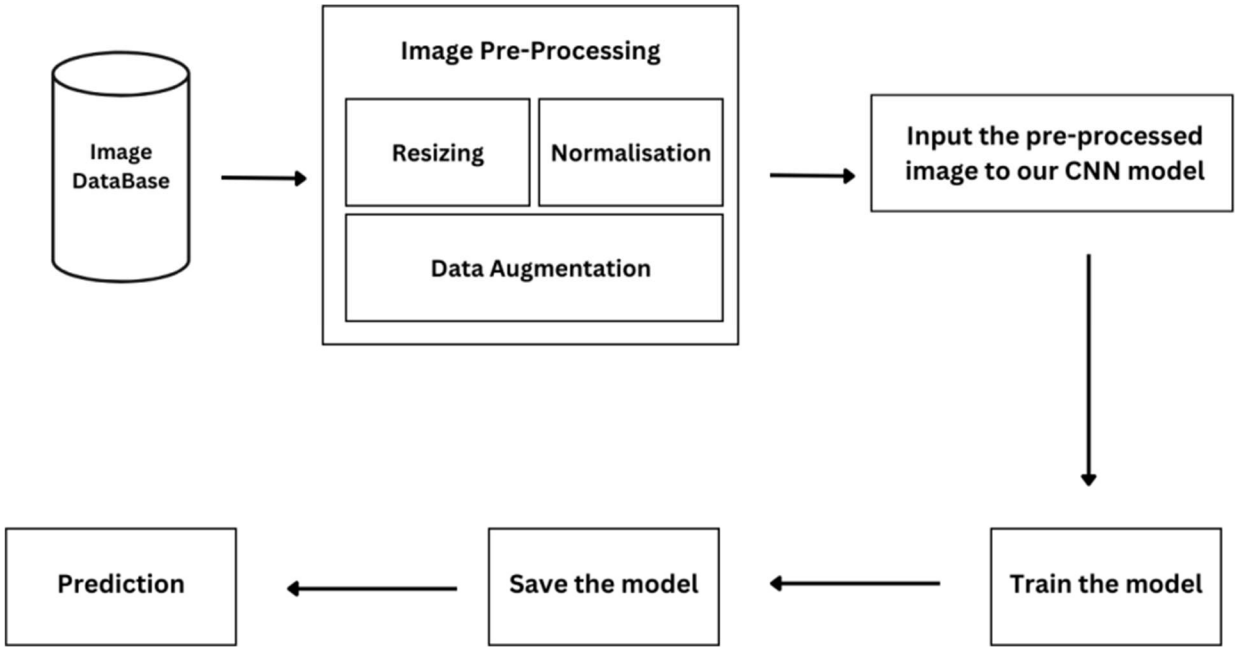

Fig 7. System Design of the proposed method

2. CNN Model & Proposed Method

```
Model: "model_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_9 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 global_average_pooling2d_2   (None, 512)              0
 (GlobalAveragePooling2D)

 dense_24 (Dense)            (None, 100)               51300

 dropout_12 (Dropout)        (None, 100)               0

 dense_25 (Dense)            (None, 1)                 101

=================================================================
Total params: 14,766,089
Trainable params: 51,401
Non-trainable params: 14,714,688
_________________________________________________________________
```

Fig 8. CNN Model Summary

The model summary provided shows the implementation of a CNN for image classification with transfer learning.

The base model of the CNN is defined as a sequence of layers using the Keras Sequential API. The base model includes three convolutional layers with 16, 64, and 64 filters respectively. The activation function used is ReLU, which is a commonly used non-linear activation function. The BatchNormalization layer is applied to normalize the output of the previous layer, which improves the learning stability and performance. The MaxPooling2D layer is used to down-sample the output of the previous layer, which reduces the dimensionality of the data.

Next, the pre-trained VGG16 model is defined using the Keras Functional API. The pre-trained model is initialized with the 'imagenet' weights, which are weights learned on a large dataset of images. The input tensor of the pre-trained model is defined with the same shape as the input tensor of the base model.

Then, the pre-trained model is added on top of the base model using the Functional API. The output of the pre-trained model is passed through a GlobalAveragePooling2D layer to reduce the dimensionality of the data. A Dense layer with 100 units and a ReLU activation function is added, followed by a Dropout layer with a rate of 0.2 to prevent overfitting. Finally, a Dense layer with a single unit and a sigmoid activation function is added to obtain the binary classification output.

The weights of the VGG16 layers are frozen to prevent them from being updated during training, so that the model can focus on learning the specific task at hand. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric.

Early stopping is implemented using the Keras EarlyStopping callback to monitor the validation loss and stop training when the validation loss stops improving after a certain number of epochs.

During training, the model is fit on the training data generator and validated on the test data generator for 100 epochs

CHAPTER 4

IMPLEMENTATION AND RESULTS

1. Implementation Details

Import Required Libraries

```
[ ] # Download from the internet if not already installed
    import sys
    !{sys.executable} -m pip install numpy > /dev/null
    !{sys.executable} -m pip install matplotlib > /dev/null
    # download tensorflow-gpu instead if you have a graphics card
    # go to: https://www.tensorflow.org/install/gpu
    !{sys.executable} -m pip install tensorflow > /dev/null
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ] import os
    import numpy as np
    import matplotlib.pyplot as plt
    from tensorflow.keras.models import Sequential
    from tensorflow.keras import layers, losses
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.applications.vgg16 import VGG16
    from tensorflow.keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
    from tensorflow.keras.models import Model
    from tensorflow.keras.callbacks import EarlyStopping
    from tensorflow.keras.applications.vgg16 import VGG16
    from tensorflow.keras.layers import Input, Flatten, Dense, Dropout, BatchNormalization, GlobalAveragePooling2D
    from tensorflow.keras.models import Sequential, Model
    from tensorflow.keras.callbacks import EarlyStopping
    from tensorflow.keras.optimizers import Adam
    from keras.applications import ResNet50,MobileNet, DenseNet201, InceptionV3, NASNetLarge, InceptionResNetV2, NASNetMobile
    from keras.callbacks import Callback, ModelCheckpoint, ReduceLROnPlateau, TensorBoard
    from keras.optimizers import Adam
    from keras import backend as K
    import gc
```

Fig 8. Importing necessary libraries

Importing and processing our data

```
[ ] BATCH_SIZE = 64
    IMG_SIZE = 224
    sigmaX = 10  # standard deviation for Gaussian filter

    train_dir = '/content/drive/MyDrive/data/train/'
    test_dir = '/content/drive/MyDrive/data/test/'


    train_benign_dir = os.path.join(train_dir, 'benign')
    train_malignant_dir = os.path.join(train_dir, 'malignant')
    test_benign_dir = os.path.join(test_dir, 'benign')
    test_malignant_dir = os.path.join(test_dir, 'malignant')
```

```
train_malignant_images = len([entry for entry in os.listdir(train_malignant_dir) if os.path.isfile(os.path.join(train_malignant_dir, entry))])
train_benign_images = len([entry for entry in os.listdir(train_benign_dir) if os.path.isfile(os.path.join(train_benign_dir, entry))])
test_malignant_images = len([entry for entry in os.listdir(test_malignant_dir) if os.path.isfile(os.path.join(test_malignant_dir, entry))])
test_benign_images = len([entry for entry in os.listdir(test_benign_dir) if os.path.isfile(os.path.join(test_benign_dir, entry))])

print("Training images: ", train_malignant_images+train_benign_images)
print("Benign training images:", train_benign_images)
print("Malignant training images:", train_malignant_images)
print("Testing images:", test_benign_images+test_malignant_images)
print("Benign testing images:",test_benign_images)
print("Malignant testing images",test_malignant_images)
```

Fig 9. Dataset Loading

```
categories = ['Train Benign', 'Train Malignant', 'Test Benign', 'Test Malignant']
counts = [train_benign_images, train_malignant_images, test_benign_images, test_malignant_images]

plt.bar(categories, counts)
plt.title('Dataset Distribution')
plt.xlabel('Categories')
plt.ylabel('Number of Samples')
plt.show()
```

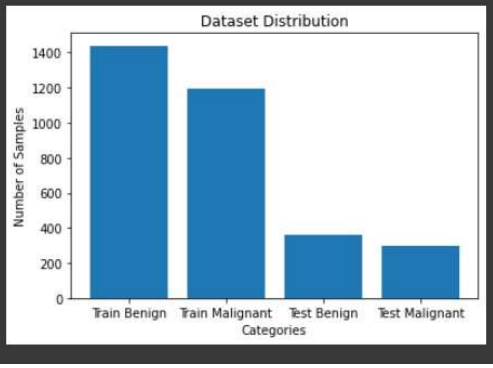

Fig 10. Dataset distribution

```
train_image_generator = ImageDataGenerator(
                                            rescale = 1./255,
      rotation_range=40,
      horizontal_flip=True,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      fill_mode='nearest',
      validation_split=0.4,
      brightness_range=[0.4,1.5])
test_image_generator = ImageDataGenerator(rescale = 1./255,
      rotation_range=40,
      horizontal_flip=True,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      fill_mode='nearest',
      validation_split=0.4,
      brightness_range=[0.4,1.5])

train_data_gen = train_image_generator.flow_from_directory(batch_size=32,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(224, 224),
                                                           class_mode='binary')

test_data_gen = test_image_generator.flow_from_directory(batch_size=32,
                                                         directory=test_dir,
                                                         target_size=(224, 224),
                                                         class_mode='binary')
```

```
Found 2637 images belonging to 2 classes.
Found 660 images belonging to 2 classes.
```

Fig 11. Image Data Augmentation

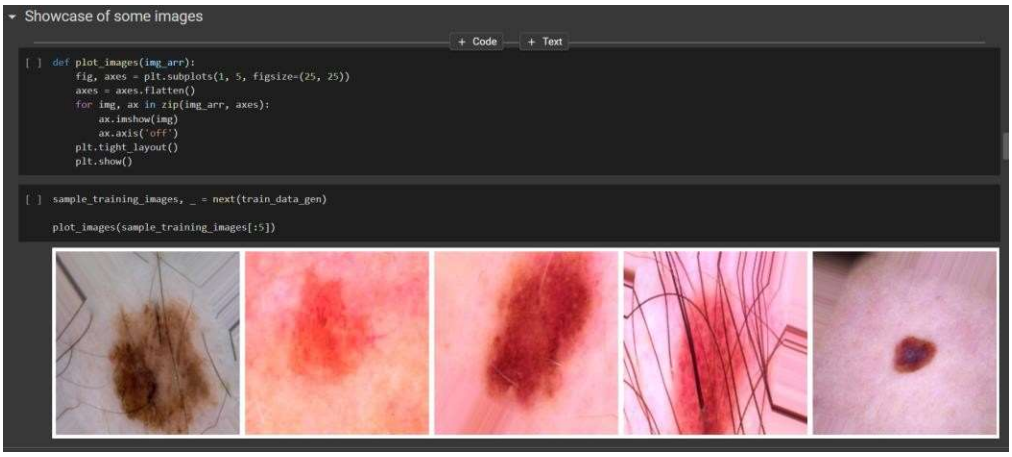

Fig 12. Dataset image showcase

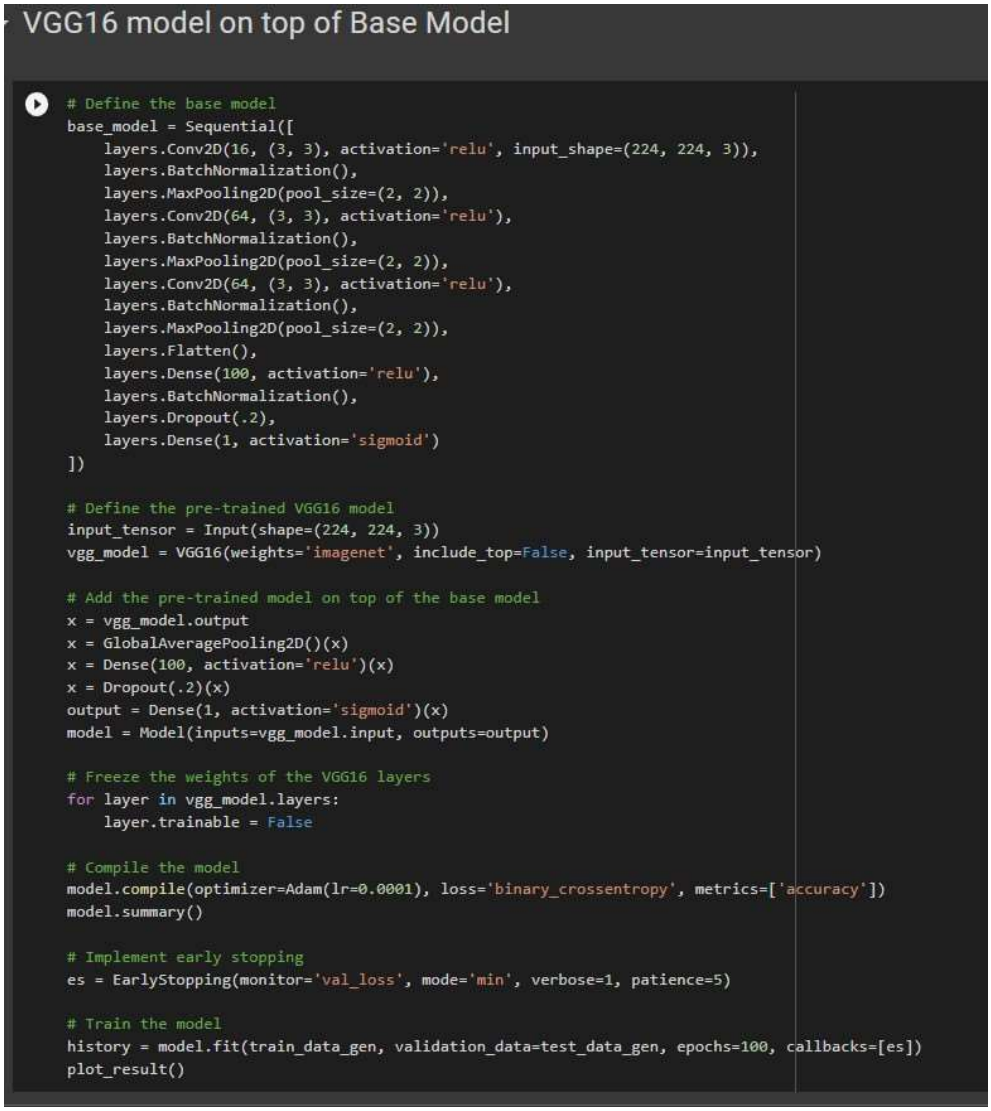

Fig 12. Proposed CNN Model

2. Results

- This is the final result of the VGG16 model on top of base layer model.

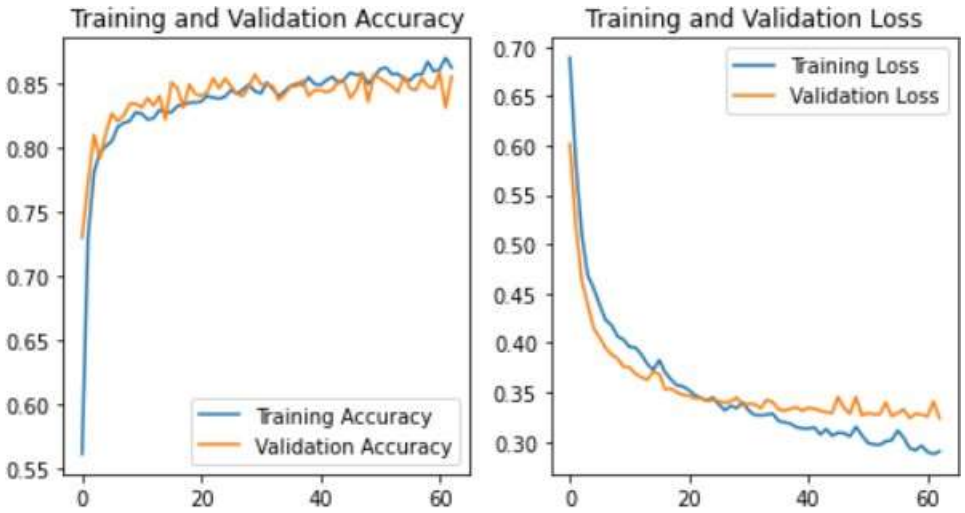

Fig 13. Result of final CNN model

- This is the result of a base VGG16 model after training on the dataset for 100 epoch with early stopping.

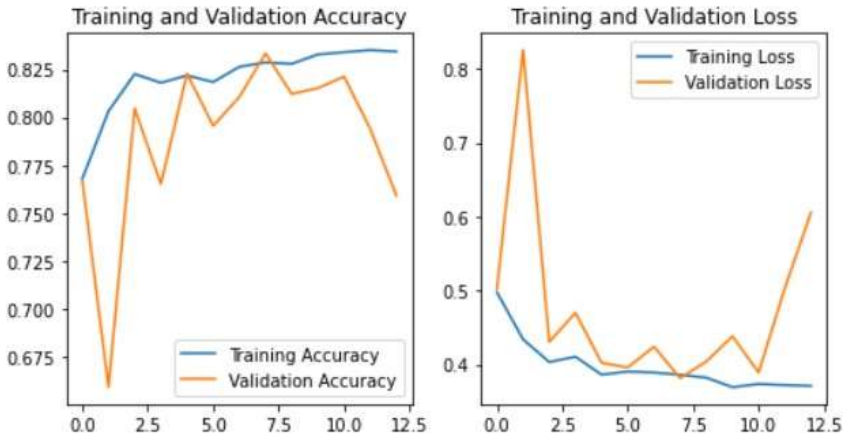

Fig 13. Result of the pretrained VGG16 model

- This is the result for a base Sequential model with added layers and early stopping

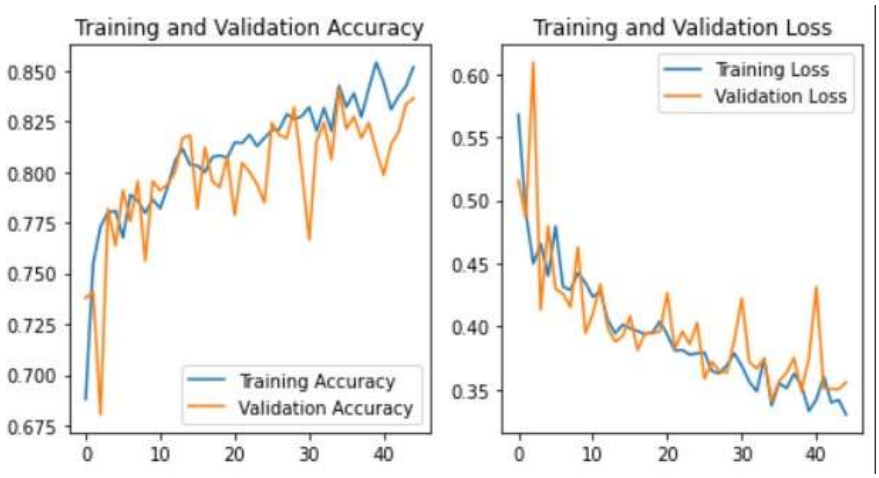

Fig 13. Result of base model alone

CHAPTER 5

CONCLUSION

The report proposes a Convolutional Neural Networks (CNN) based approach for classifying melanoma, with the goal of helping doctors and patients detect and identify skin cancer as benign or malignant. While the traditional approach to identifyingskin cancer can be time-consuming, the CNN model presented in the paper offers a more efficient and accurate way for healthcare professionals to detect and diagnose skin cancer using a set of random images.

Additionally, in the future, the aim is to develop an application that can be easily used by individuals at home to test if a random mole is skin cancer or not. This application would leverage the proposed CNN model, which has demonstrated high accuracy in identifying skin cancer classes, to provide a convenient and accessible way for people to assess their skin health. With this application, people can perform a self-check and seek medical advice promptly if they detect any abnormalities, potentially contributing to the early detection and treatment of skin cancer.

References

1. R. Maurya, Surya Kant Singh, A. K. Maurya and A. Kumar, "GLCM and Multi Class Support vector machine based automated skin cancer classification," 2014 International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2014, pp. 444-447, doi: 10.1109/IndiaCom.2014.6828177.
2. E. Nasr-Esfahani et al., "Melanoma detection by analysis of clinical images using convolutional neural network," 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Orlando, FL, USA, 2016, pp. 1373-1376, doi: 10.1109/EMBC.2016.7590963.
3. Y. Jusman, I. M. Firdiantika, D. A. Dharmawan and K. Purwanto, "Performance of Multi Layer Perceptron and Deep Neural Networks in Skin Cancer Classification," 2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech), Nara, Japan, 2021, pp. 534-538, doi: 10.1109/LifeTech52111.2021.9391876.
4. Mariam A. Sheha Cairo University Mai S .Mabrouk MUST University Amr Sharawy Cairo University ,"Automatic Detection of Melanoma Skin Cancer using Texture Analysis", International Journal of Computer Applications,2012.
5. N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, H. Kittler, and A. Halpern, ‘‘Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC),’’ 2019, arXiv:1902.03368. [Online]. Available: https://arxiv.org/abs/1902.03368
6. M. Afzal Ismail, Nazia Hameed & Jeremie Clos. Deep Learning-BasedAlgorithm for Skin Cancer Classification (pp 709-719). PICTCCE.
7. Choudhari, Sarika and Seema Biday. “Artificial Neural Network for SkinCancer Detection.” (2014).
8. https://www.kaggle.com/datasets/fanconic/skin-cancer-malignant-vs-benign