Melanoma Skin Cancer Diagnosis Using Machine Learning

Final Year Project Report

Presented

by

Muhammad Yahya Khan
CIIT/FA17-ECE-027/ISB

Qasim Muhammad Ashraf
CIIT/FA17-ECE-032/ISB

In Partial Fulfillment

of the Requirement for the Degree of

Bachelor of Science in Electrical (Computer) Engineering

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COMSATS UNIVERSITY ISLAMABAD

May 2021

Melanoma Skin Cancer Diagnosis Using Machine Learning

Final Year Project Report

Presented

by

Muhammad Yahya Khan
CIIT/FA17-ECE-027/ISB

Qasim Muhammad Ashraf
CIIT/FA17-ECE-032/ISB

In Partial Fulfillment

of the Requirement for the Degree of

Bachelor of Science in Electrical (Computer) Engineering

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COMSATS UNIVERSITY ISLAMABAD

May 2021

Declaration

We, hereby declare that this project neither as a whole nor as a part there of has been copied out from any source. It is further declared that we have developed this project and the accompanied report entirely on the basis of our personal efforts made under the sincere guidance of our supervisor. No portion of the work presented in this report has been submitted in the support of any other degree or qualification of this or any other University or Institute of learning, if found we shall stand responsible.

Signature:_____
Name

Signature:_____
Name

COMSATS UNIVERSITY ISLAMABAD
May 2021

Melanoma Skin Cancer Diagnosis Using Machine Learning

An Undergraduate Final Year Project Report submitted to the Department of
ELECTRICAL AND COMPUTER ENGINEERING

As a Partial Fulfillment for the award of Degree

Bachelor of Science in Electrical (Computer) Engineering

by

Name	Registration Number
Muhammad Yahya Khan	CIIT/FA17-ECE-027/ISB
Qasim Muhammad Ashraf	CIIT/FA17-ECE-032/ISB

Supervised by

Dr. Shahzad Ali Malik
Professor,
Department of Electrical and Computer Engineering
CU Islamabad

COMSATS UNIVERSITY ISLAMABAD
May 2021

Final Approval

This Project Titled
Melanoma Skin Cancer Diagnosis Using Machine Learning

Submitted for the Degree of
Bachelor of Science in Electrical (Computer) Engineering

by

Name	**Registration Number**
Muhammad Yahya Khan	CIIT/FA17-ECE-027/ISB
Qasim Muhammad Ashraf	CIIT/FA17-ECE-032/ISB

has been approved for

COMSATS UNIVERSITY ISLAMABAD

_____
Supervisor
Dr. Shahzad Ali Malik,
Professor

_____
Internal Examiner-1
Dr. Haroon Ahmed Khan,
Assistant Professor

_____
Internal Examiner-2
Mr. Dilshad Sabir,
Lecturer

_____
External Examiner
Name,
Designation

_____
Head
Department of Electrical and Computer Engineering

Dedication

Our efforts in this project are dedicated to our parents and teachers who have always been the source of our inspiration and motivate us throughout our lives. They support us through thick and thin, may Allah bless them all.

Acknowledgements

First of all, we are thankful and grateful to the creator of everything, Allah Almighty, under whose benevolence we exist and who has made us capable due to which we got an opportunity that we are studying at COMSATS University and also for completion of our project and BS ECE degree. There are many people who helped us throughout our journey of engineering and this project in particular. We are very thankful to those who helped us in completing our project.

Prof. Dr. Shahzad A. Malik, our respected supervisor is the one who has guided us on the right direction and also advised us throughout the project. In this difficult period of COVID-19 when we did not have adequate means of communication, he offered his precious time and his support without any hesitation, he understands students' limitations and problems, and provides a feasible solution to their problems and also motivates them to beat their limitations. It might not have been possible for us to finish our project if not for his kind and sincere help.

We are very grateful to our parents. They have always supported in our journey, financially and morally as well, without expecting anything back from us.

We are also grateful to our teachers, from school till university. We have learned a lot from them and without their support, we would not be here.

Lastly, we would like to thank our department (Electrical and Computer Engineering) and COMSATS University for the opportunity and preparatory education required to complete this project.

Muhammad Yahya Khan
Qasim Muhammad Ashraf

Table of Contents

List of Acronyms

AI: Artificial Intelligence
ML: Machine Learning
SVM: Support Vector Machine
ANN: Artificial Neural Network
CNN: Convolutional Neural Network
ReLU: Rectified Linear Unit
FLOPS: Floating point operations per second

List of Figures

Abstract

Early detection of Melanoma type of skin cancer is crucial for the person affected because it can lead towards the death of the person who is affected by this cancer type. If this cancer is diagnosed at early stages, then it can be taken care of easily.
Image classification using machine learning is an effective technique to detect Melanoma using the images of lesions, it can help medical diagnostic centers in early detection of Melanoma.
Machine learning techniques like Artificial neural network (ANN) and many more are used in classification of images.
The objective is to classify benign and malignant cancer images using multiple neural network architectures and then use an ensemble network to find the best result out of them all.

Chapter 1: Introduction

1.1: Overview

Advancement in technology and computer science has opened new doors for human beings. AI is a field of computer science which covers the broad idea of making intelligent machines using either both software and/or hardware which can act on their own without human intervention. ML is a field of AI using which we can create software that is able to predict an output based on the given input by training it.
One of the techniques that can be used in ML is image classification, which is best suited for detecting skin cancer. It is a technique that uses a database of predetermined patterns to help us compare objects and then group them into categories. Remote sensing, robot navigation, and biological imaging are just a few of the uses for classification algorithms.
Image classification allows for the categorization of pictures into benign and malignant categories.
supervised learning and unsupervised learning are two rule-based classification techniques used in image classification. We shall be able to anticipate the outcome or class of data in supervised learning, and a trained data set will be presented. Prior information is required before testing when using the supervised learning method, and this data must be obtained by the analyst.
The user can detect faults and correct them with supervised learning, but it takes longer and costs more money, and the training data is chosen by the analyst. It may not involve all the conditions to detect the skin cancer. Supervised classification also involves human intervention.
In unsupervised learning, no trained data will be provided but the classifier itself finds which category or class it belongs to. The user need not have any prior information, so no human intervention is required. Unsupervised classification is much faster when compared to supervised classification but the accuracy is lower. There are no human errors in this procedure, and no prior knowledge is required.

1.2: Background & Motivation

Background:

Skin is an important part of the human body; it protects us from harmful elements such as ultraviolet (UV) rays and help to maintain the body temperature.
Melanoma is one of the most common type of skin cancer, and it begins in skin cells called melanocytes and the disease can start growing from anywhere on the body.
Most moles do not develop into melanoma, but some do, and researchers have discovered genetic alterations in mole cells that may lead to the development of melanoma cells. UV light exposure is a primary risk factor for most melanomas. UV (ultraviolet) rays are mostly produced by sunlight. UV (ultraviolet) radiation can also be found in tanning beds and sunlamps. They come in a variety of colors, including brown, black, red, blue, and a blue-red combo. Melanoma is a skin cancer that only affects the top layer of the skin. The tumor is only a few millimeters to two millimeters

thick, and the surface may or may not be fractured. Cancer cells can spread to other organs and tissues, including the liver, brain, and bones.

Motivation:

The motivation here is to assist the medical diagnostic centers in detecting Melanoma at an early stage.
It is crucial to use and improve these given supporting imaging techniques in this situation since they have been found to improve and facilitate the diagnosing process.

1.3: Problem Statement

In the medical field, there is an increasing incidence of melanoma skin cancer cases, statistics show that there is about 27% survival rate (on the 5-year relative survival rate chart) for people who are diagnosed with melanoma at a distant stage, so it is critical to detect melanoma at an early stage. In this project we are going to merge two machine learning techniques, namely transfer learning and ensemble learning to classify between benign and malignant cancer images.

1.4: Aims & Scope

- To distinguish between benign and malignant melanoma skin cancer.
- To implement and test different neural network architectures for best results.
- To be able to detect within reasonable degree of accuracy.
- To try to improve on the accuracy of our model.

1.5: Organization of Report

We have organized our report in the following manner:
It has been divided into four chapters, namely:

- Introduction
- Methodology
- Design and Implementation
- Conclusion

Chapter 2: Melanoma Skin Cancer Diagnosis: Background and Literature

2.1: Machine Learning

2.1.1: Overview

AI is a field of computer science that focuses on the creation of intelligent computers that think and operate in the same way that people do. Some of the applications are as follows: speech recognition, problem solving, learning, and planning etc. ML is an AI application that allows a system to learn and grow from experience without having to be explicitly coded.
Deep learning is an ML technique that replicates the human brain's utilization of memories and formulation of patterns for use in real life decisions.

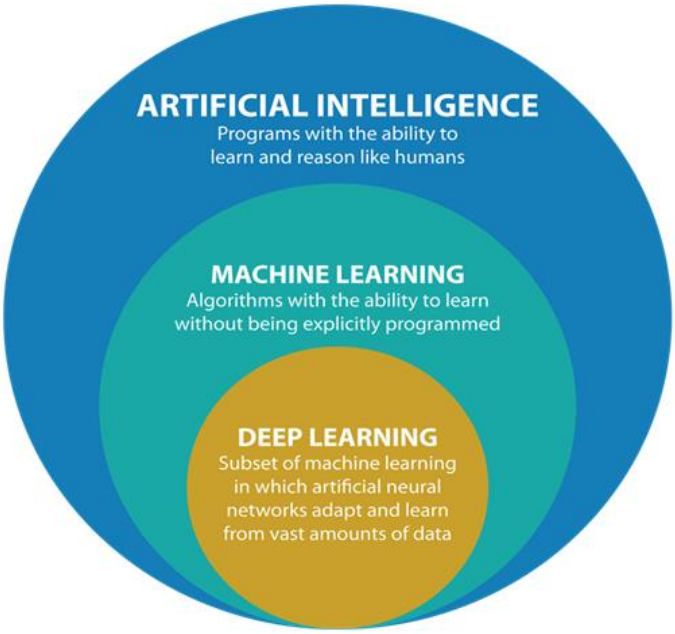


Figure 2.1: Relation between AI, ML and Deep Learning.

The basic definition of Machine Learning according to Arthur Samuel is:

"The field of study that gives computers the ability to learn without being explicitly programmed."

This is an older, informal definition; A more complete and modern definition according to Tom Mitchell is:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Generally, any ML problem can be assigned to one of two broad classifications: Supervised or Unsupervised learning

2.1.2: Types

Machine Learning can be further classified into three classes:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

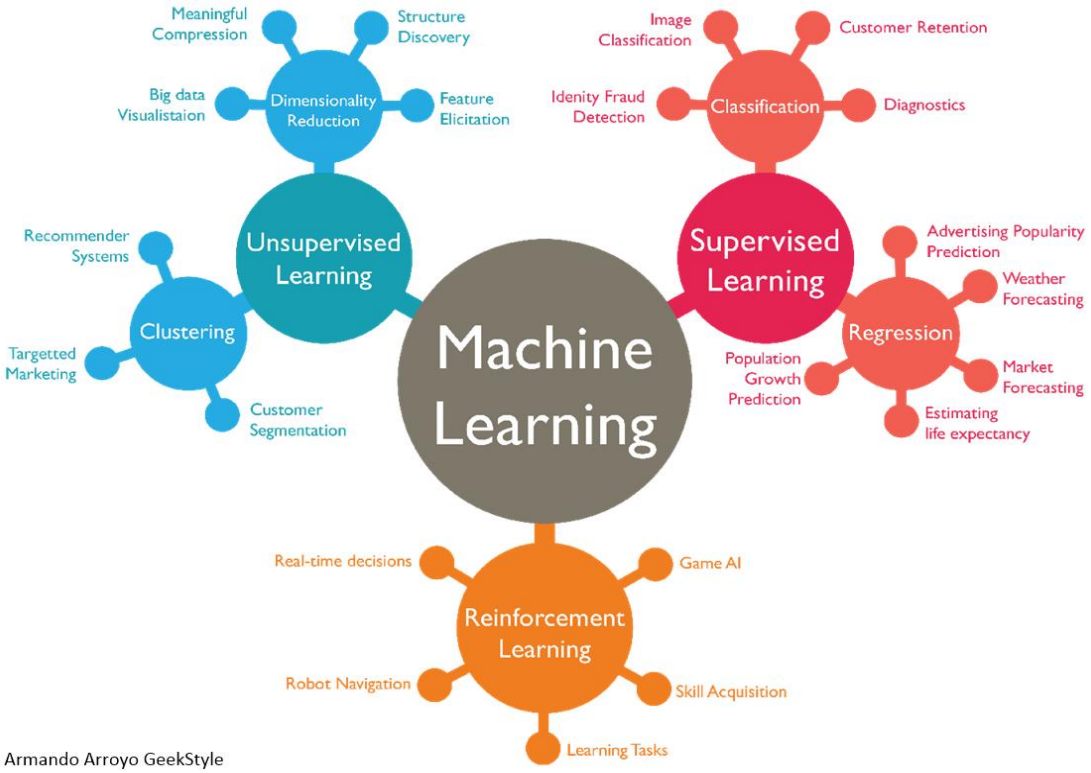


Figure 2.2: Classes of Machine Learning

The main focus for medical image diagnosis is on image classification from the branches of supervised learning.

2.1.3: Applications

There are numerous applications of ML, areas to which various existing ML systems have been applied are as follows:

- Computer programming
- Image recognition and speech recognition
- Medical diagnosis
- Agriculture
- Email management
- Robotics

- Mathematics

And many more.

The focus here is on the medical diagnosis applications of ML.

2.1.4: Techniques

Supervised learning can be further divided into two types: Classification and Regression
We aim to map input variables to some continuous function in a regression problem, which means we forecast results within a continuous output.
We aim to anticipate results in a discrete output in a classification problem. To put it another way, we are attempting to map input variables into distinct categories.

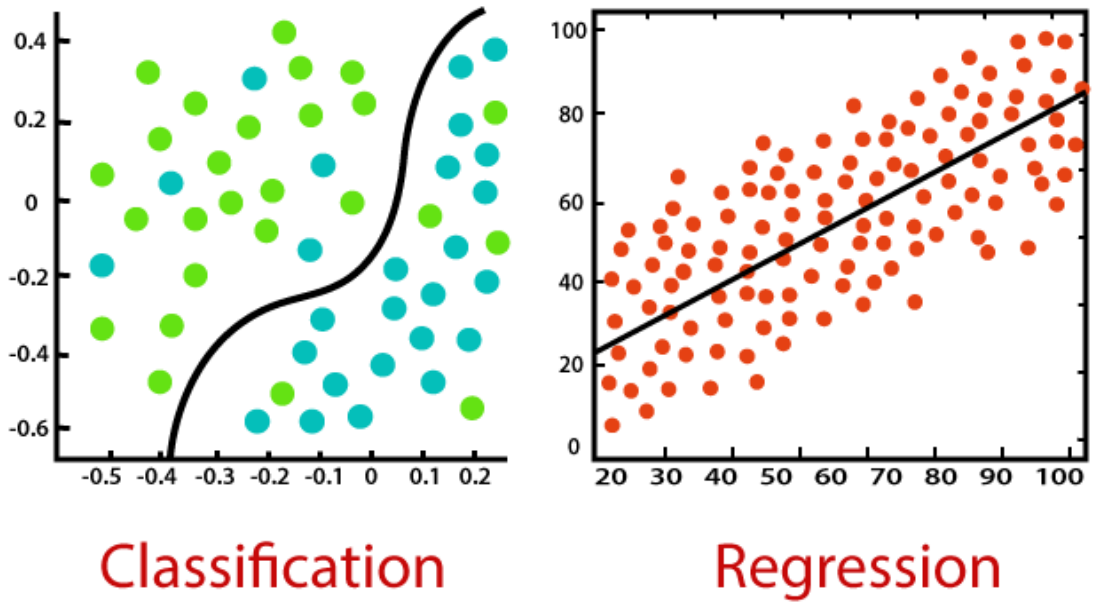


Figure 2.3: Comparison between classification and regression

In this project, we are dealing with a classification problem. As we are trying to differentiate between benign and malignant melanoma.

A list of classification algorithms is as follows:

- Logistic Regression
- Naïve Bayes
- Stochastic Gradient Descent
- K-Nearest Neighbours
- Decision Tree
- Random Forest
- Support Vector Machine
- Artificial Neural Network

In this project, we are going to use artificial neural networks to solve our problem. We preferred this method over others due to its versatility, while using neural networks we have a lot of freedom. We can add and remove layers according to our needs and make the network optimized for our application.

2.2: Neural Network

2.2.1: Overview

The basic definition of neural network is:

"A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus, a neural network is either a biological neural network, made up of biological neurons, or an artificial neural network, for solving AI problems."

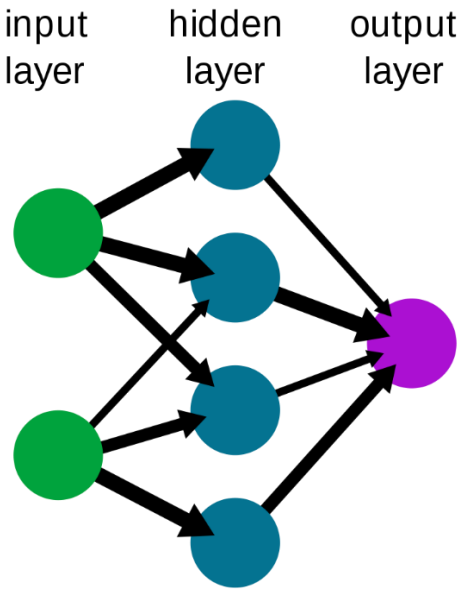


Figure 2.4: A simple neural network

The Neural Network is one of the most powerful learning algorithms. Neural networks are algorithms that are designed to recognize patterns and are loosely modelled after the human brain. They can interpret data by clustering or labelling raw input, which is a form of machine perception. They recognize patterns in numerical vectors into which all real-world data, such as text, sound, time series, and images, should be translated.

2.2.2: Types

There are different types of neural networks. They each have their own set of principles and set of rules, and each has a distinct and special strength. A list of types is given below:

1. Feedforward Neural Network

2. Radial Basis Neural Network
3. Multilayer Perceptron Neural Network
4. Convolutional Neural Network
5. RNN
6. MNN
7. Sequence Driven Network

2.2.3: Advantages

Neural Networks have various advantages over other techniques. Some of which are as follows:

1. Store information on the entire network
2. Ability to work with inadequate data
3. Good fault tolerance
4. Distributed memory:
5. Gradual corruption:
6. Ability to train machine:
7. Ability of parallel processing

2.2.4: Convolutional Neural Networks (CNN)

CNN is a form of feed-forward neural network used in artificial intelligence. It is a popular tool for image recognition. The receptive field is what CNN uses to extract each and every component of the input picture. It gives weights to each neuron depending on the receptive field's importance. So that it can distinguish between the significance of neurons. There are three sorts of layers in CNN's architecture: (1) convolution, (2) pooling, and (3) completely connected.[2]

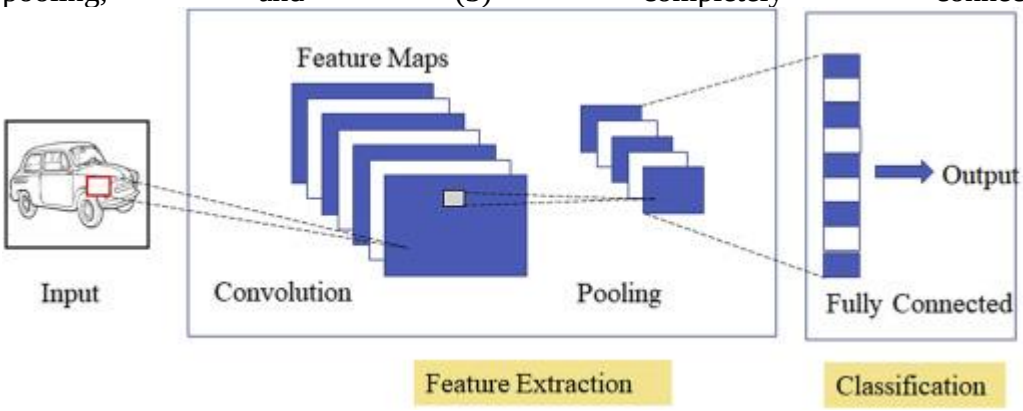


Figure 2.5: Example of CNN [2]

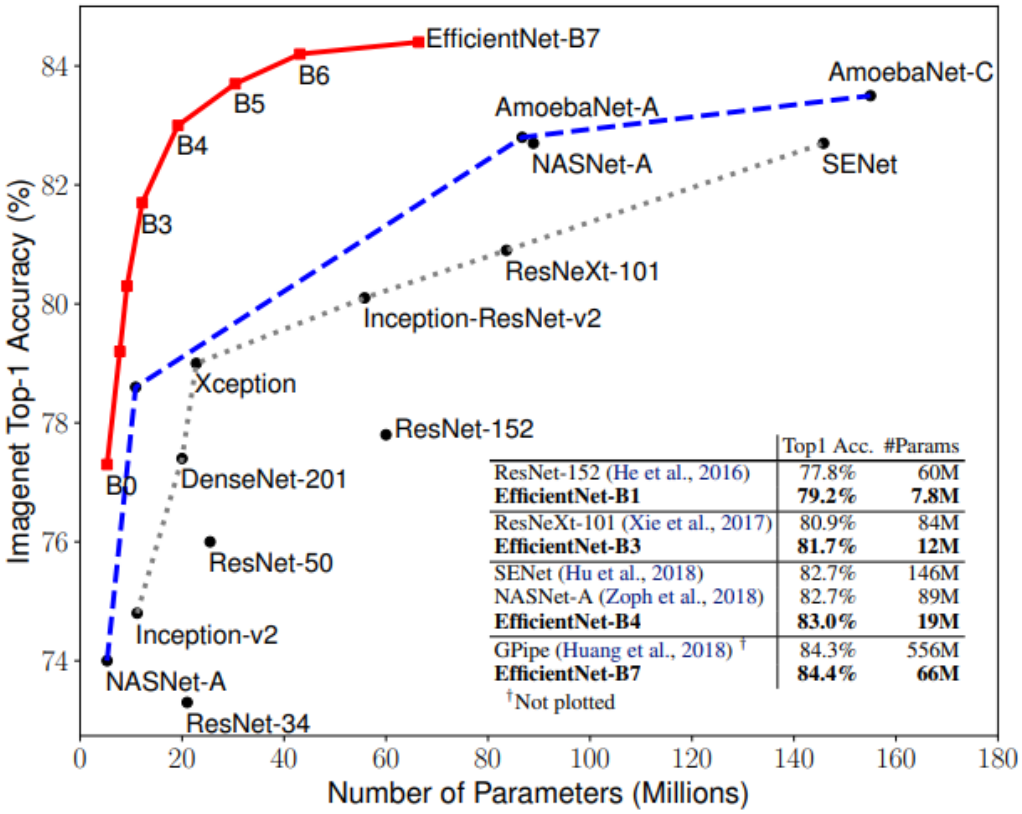
For the past decade, neural networks using convolutions have primarily been used as a classifier for image processing.
There are several hidden levels, as well as an input and output layer, make up a typical CNN network.
A CNN's hidden layers are usually made up of a series of convolutional layers.

The most common activation function is ReLU, which is usually followed by other operations.

2.2.5: Architectures

There are many CNN architectures. In this project, the following architectures will be used:

- ***EfficientNet***
  It is a convolutional neural network design and scaling approach that uses a compound coefficient to scale all depth/width/resolution dimensions evenly. [3]
  The following graph shows a comparison of EfficientNet architectures and other architectures:



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Figure 2.6 Comparison of different architectures [3]

1. EfficientNet-B0
   Number of parameters: 5.3M
   Number of FLOPS: 0.39B
   [3]

2. EfficientNet-B1
   Number of parameters: 7.8M

Number of FLOPS: 0.70B
[3]

3. EfficientNet-B2
   Number of parameters: 9.2M
   Number of FLOPS: 1.0B
   [3]

4. EfficientNet-B3
   Number of parameters: 12M
   Number of FLOPS: 1.8B
   [3]

- ***MobileNet & MobileNetV2***

1. MobileNet:
   It's a type of convolutional neural network designed specifically for mobile and embedded vision.[4]
2. MobileNetV2:
   It's a mobile-friendly version of a convolutional neural network. It's built on an inverted residual structure, with residual connections connecting bottleneck levels.[5]

- ***ResNet & ResNetV2***

1. ResNet50:
   It is a ResNet version with 48 Convolution layers, one MaxPool layer, and one Average Pool layer. There are 3.8 x 109 floating point operations in it.[6]

2. ResNet50V2:
   ResNet50V2 is a modified version of ResNet50 that performs better on the ImageNet dataset than ResNet50 and ResNet101.[6]

- ***VGG16 & VGG19***

VGG is a convolutional neural network design that has been around for a long time. It was based on a study on how to make such networks denser.[7]

1. VGG16:
   The padding and MaxPool layer of the 2x2 stride 2 filter were always the same in this design, which favoured 3x3 convolution layers with stride 1.[7]

2. VGG19:

It is a variation of the VGG model, which has 19 layers, VGG19 has a FLOP count of 19.6 billion.[7]

- ***Xception***

The network's feature extraction foundation is made up of 36 convolutional layers.[6]

2.3: Transfer Learning

2.3.1: Overview

Transfer learning is the process of using features learned on one problem to solve a new, similar problem.
Transfer learning is used when the dataset contains insufficient data to train a full-scale model from scratch.
This is a workflow of transfer learning:

- Take layers from a model that has already been trained.
- Freeze them to prevent any of the information they contain from being lost during future training rounds.
- On top of the frozen layers, add some new, trainable layers.
- They'll figure out how to apply the old features to a new dataset and make predictions.
- On your dataset, train the new layers.

In our case we have used the following models for transfer learning:

- EfficientNet-B0
- EfficientNet-B1
- EfficientNet-B2
- EfficientNet-B3
- MobileNet
- MobileNetV2
- ResNet50
- ResNet50V2
- VGG16
- VGG19
- Xception

All of the above-mentioned models have been trained on ImageNet.

ImageNet:
ImageNet is a big visual database that was designed to help in the development of visual object identification software.

Over 14 million pictures have been hand-annotated to show what items are seen, with bounding boxes in at least one million of them.
There are over 20,000 categories in ImageNet, with a typical category like "balloon" or "strawberry" containing several hundred images.

ImageNet refers to the ImageNet Large Scale Visual Recognition Challenge in terms of deep learning and Convolutional Neural Networks (ILSVRC).
The aim of this image classification challenge is to develop a model capable of properly classifying 1,000 different item categories from a single input image.
Models are trained using 1.2 million training pictures, 50,000 validation images, and 100,000 testing images.
These 1,000 image categories represent object classes that we come across in our daily lives, including dog and cat species, various household objects, vehicle types, and much more.

These models have been trained for a general dataset, when we use transfer learning to apply them on a specific dataset, we get good results.

2.3.2: Pre-Trained Models

Why do we use pre-trained models?

This is desirable for a variety of reasons, including the following:

- Useful Learned Features
- State-of-the-Art Performance
- Easily Accessible

How to use pre-trained models?

Some of these usage patterns can be summarized as follows:

- Classifier
- Standalone Feature Extractor
- Integrated Feature Extractor
- Weight Initialization

Each method can be effective in terms of developing and training a deep convolutional neural network model while also saving time. In our case, we are going to use pre-trained models as classifiers.

Keras gives us access to a number of high-performing pre-trained models designed for image recognition.
They're accessible through Keras API and include functions for loading a model with or without pre-trained weights, as well as preparing data in a way that a particular model might expect (e.g., scaling of size and pixel values).
Keras will download the required model weights the first time a pre-trained model is loaded.

Weights are saved in the (keras/models/) directory locally and loaded from there the next time they're needed.

2.4: Literature Survey

Detection of melanoma skin cancer has been the focus of many researchers. Melanoma skin cancer is extremely severe; thus, it must be discovered as soon as possible. Many researchers have applied preprocessing techniques such as filtering, edge detection, and the watershed approach, as well as the wavelet method, thresholding technique, and the ABCD rule for feature extraction.

[11]. In this paper, approaches like gamma-correction are used to improve the input. Segmentation is done via edge detection and automated thresholding, while feature extraction is done with the ABCD rule.

[12]. In this paper, first the input image is converted to gray scale and filtered. The second stage is the filtered is segmented using OTSU'S algorithm. Then for feature extraction STOLZ algorithm is used. Then TDS value is calculated using which classification is done.

[13]. In this paper, the k-nearest neighbours' method, logistic regression, decision tree, and SVM were employed as classifiers. With SVM, the created system obtained 92 percent accuracy.

[14]. In this paper the authors have discussed the Advancements in digital image-based AI solutions for skin cancer detection, as well as significant obstacles and future potential to develop these AI systems to help dermatologists (skin cancer specialists) identify skin cancer.
Some of the challenges that they are discussed are.

- Performance of deep learning and unbalanced datasets.
- Race, ethnicity, and population
- Patients' medical history and clinical data
- In some cases, Biopsy is a must.

While some of the opportunities are such as

- Diverse datasets
- Data augmentation
- Multiple models for diagnosis of skin cancer

[15]. The authors of this study look at hyperparameter search techniques. Performances in terms of accuracy are checked and documented during the study using various optimization approaches.

[16]. In this paper, the authors propose using fully convolutional networks (FCNs) to segment skin lesions automatically, developed a novel parallel integration approach for combining complimentary data from various segmentation stages to provide a final segmentation result with precise localization and well-defined lesion borders.

[17]. In this paper, the authors found that EfficientNets perform well for skin lesion classification. In the final ensemble strategy, various EfficientNets were present, although the largest ones performed best. This suggests that a combination of input resolutions is a suitable choice for skin lesion categorization in a multi-scale setting.

[18], In this paper, the authors introduced an ensemble-based deep learning approach that helps dermatologists distinguish between seven different types of skin lesion categories. When compared to previous lesion classification approaches, these networks in an ensemble method yielded substantially better outcomes.

Chapter 3: Method, Implementation and Results

3.1: Flow Diagram

The flow of our project is visually explained in the following diagram:

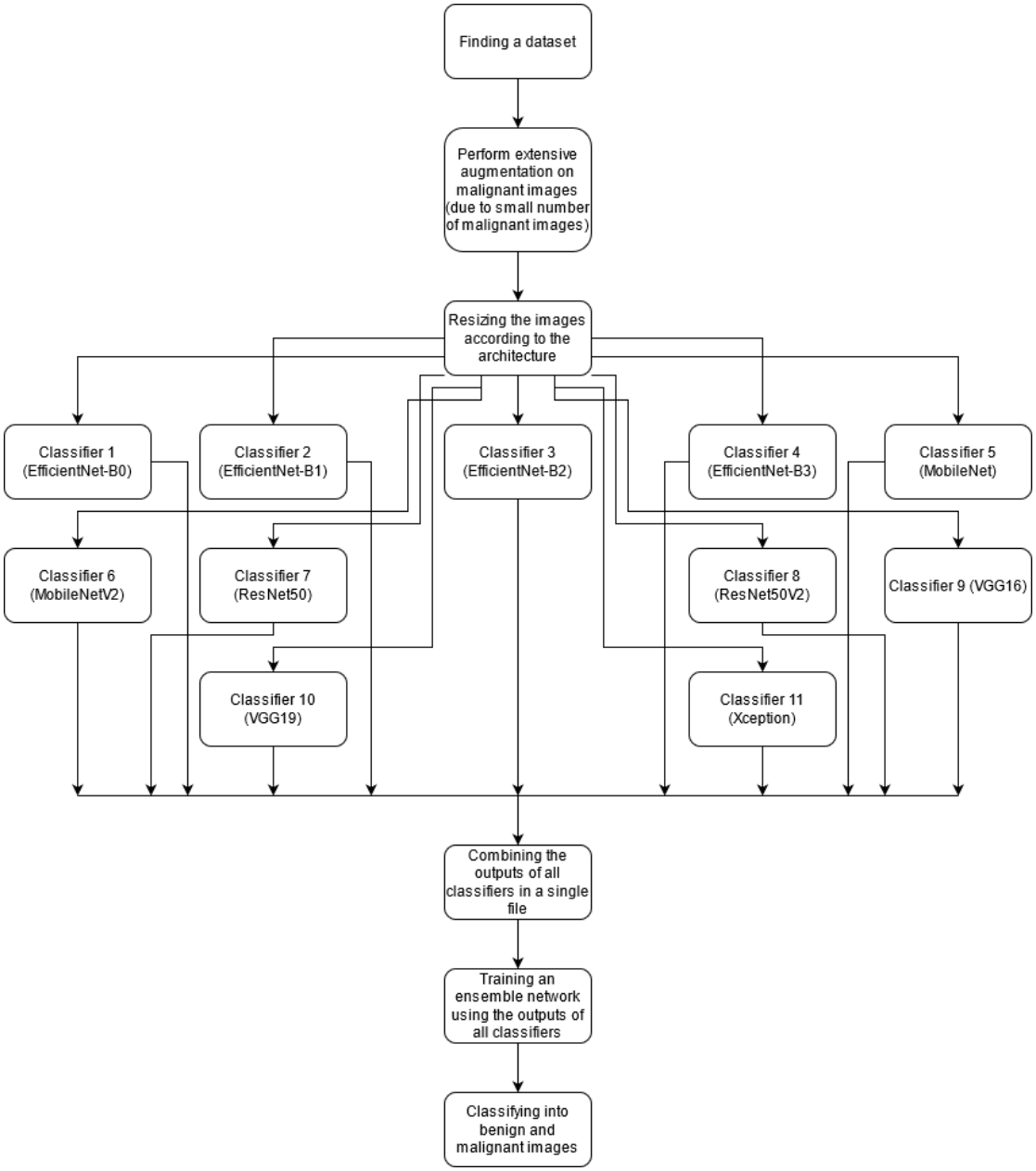


Figure 3.1: Process Flow Diagram

3.2: Dataset

ISIC 2020 Challenge Dataset:

A total of 33,126 dermoscopic training photos of unique benign and malignant skin lesions from over 2,000 patients are included in the dataset. Using a unique patient identity, each image is linked to one of these people. Histopathology was used to confirm all malignant diagnoses, while expert agreement, longitudinal follow-up, and histopathology were used to confirm benign diagnoses. A comprehensive article that describes all aspects of this dataset is available as a pre-print that has not yet been peer reviewed. The dataset was generated by the International Skin Imaging Collaboration (ISIC).[9]

After analysing the dataset, we found out that the images do not require segmentation, the dataset is already optimized for training models. The images in this dataset have different resolutions, so we needed to resize them for training different models.
When we used the ground truth given in the csv file, we found out that 32,542 images out of the 33,126 are of benign tumours and the remaining 584 images are of malignant tumours.
This ratio between malignant and benign images is not suitable for training neural networks efficiently.
If there is a big difference in ratio between classes, it creates class imbalance, and the neural network becomes biased.
In classification problems, it is better to have a 50-50 ratio in between classes or at least close to 50% to avoid class imbalance.
In our case, we have a ratio of 55-45 between malignant and benign images. We performed extensive augmentation on the 584 malignant images and increased their number to 40,880.

3.3: Python

It is used for:

- Software development,
- Mathematics,
- System scripting.

What can Python do?

- Python may be used to create server-side web applications.
- Python may be used in combination with other applications to create processes.
- Python has the ability to connect to database systems. It's also capable of reading and writing files.
- Python can process enormous quantities of data and perform complicated computations.
- Python may be used for both quick prototyping and production-ready software development.

Why Python?

- Python can run on a number of platforms (Windows, Mac, Linux, Raspberry Pi, etc.).

- Python's syntax enables programmers to build programs in fewer lines than other programming languages.
- Python is an interpreter language, which means that code may be run right away once it is written. As a result, prototyping may be completed in a short period of time.
- There are three approaches to Python: procedural, object-oriented, and functional.
- Python has a mathematical influence and shares some similarity to the English language.
- In contrast to other programming languages, Python uses new lines rather than semicolons or parentheses to finish tasks.
- Python uses indentation and white space to indicate scope, such as the scope of loops, functions, and classes. For this reason, curly brackets are widely employed in various programming languages.

3.4: Preprocessing

Augmentation:
Image augmentation artificially creates training images using a variety of processing techniques or a combination of techniques, such as random rotation, shifts, shear, and flips, among others.

The dataset did not have an adequate number of images for malignant tumors so we had to perform augmentation on the existing 584 images.
For this purpose, we used a community developed python library from GitHub: imgaug.[10]

We performed the following operations on our images:

- Fliplr
- Flipud
- GaussianBlur
- MedianBlur
- LinearContrast
- GammaContrast
- LogContrast
- CLAHE
- MultiplyAndAddToBrightness
- MultiplyHue
- MultiplySaturation
- ChangeColorTemperature
- ShearX
- ShearY

- pillike.Affine

We performed these operations on the malignant images' multiple times using different variables, sometimes with combining two operations.
On a single image, augmentation was performed a total number of 69 times. After performing these operations on 584 images, we got 40,880 images of malignant tumors. In the following figure we can see a single image and its augmented variations:

Figure 3.2: Augmentation

We used a python script to run this process. This augmented dataset was then used to train models.

The pre-built Keras image data generator also allows us to perform augmentation, but it would have cost us a lot of time as the image data generator would have to perform augmentation for every model separately. So, we instead used another library to perform augmentation on images first and then feed them into the neural networks.

3.5: TensorFlow, Keras & CUDA

TensorFlow:
TensorFlow is an open-source library for a variety of machine learning applications. Both low-level and high-level APIs are available.

Keras:
Keras is a Python-based deep learning API that runs on top of TensorFlow, a machine learning platform.
Keras is an industry-strength framework built on top of TensorFlow 2.0 that can scale to huge clusters of GPUs or a whole TPU pod. It only provides APIs at a high level.
Both of these are machine learning frameworks, that provide high-level APIs for building and training models with ease.

CUDA:
Nvidia CUDA, a parallel computing platform and programming paradigm for general computing, was developed on its own GPUs (graphics processing units).

By utilising the parallelizable component of the calculation, CUDA assists developers in speeding up compute-intensive applications. CUDA is used by these frameworks.

The following image explains relation between the three:

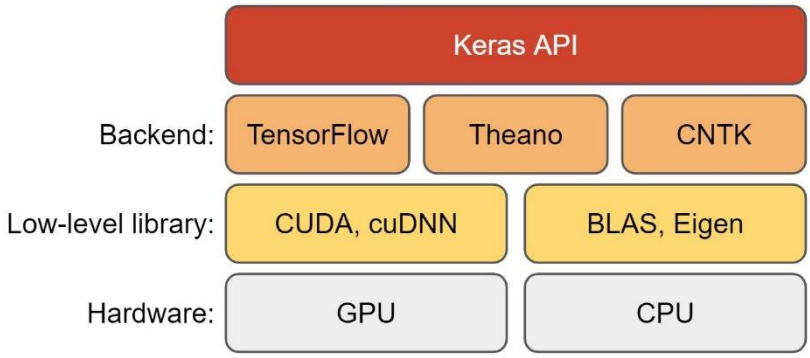


Figure 3.3: Relation between Keras, TensorFlow and CUDA

The Keras core library's state-of-the-art pre-trained networks are among the best-performing Convolutional Neural Networks on the ImageNet challenge over the last few years. These networks also show a strong ability to transfer learning, such as feature extraction and fine-tuning, to images outside the ImageNet dataset.
We have used Keras to import prebuilt neural network architectures and train them on our dataset.

3.6: Training

There is a lot of stuff to consider before we start training these models.
First of all, the most important thing is the hardware we are using for training.
In our case we have performed our training on a computer with the following specifications:

CPU: Xeon E5-1620 v2 (4 cores, 8 threads)
GPU: GTX 1080 (8 GB VRAM)

We have used CUDA to utilize the maximum potential of this GPU. GTX 1080 is a CUDA-enabled GPU from Nvidia, it has 2560 CUDA cores.

We could not train models like EfficientNet-B7 etc. due to the video memory limit of 8GB.
Some of these models require very high image sizes on input, for example: EfficientNet-B7 requires an image size of 600x600.

In order to run EfficientNet-B7 we would have to reduce the batch size to a small amount, that would make the model inefficient, reducing its accuracy.

Explaining the technical terms that are going to be used in this section:

Image Size:
The term "image size" refers to the height and width of an image in pixels.

Activation:
An activation function in a neural network describes how the weighted sum of the input is converted into an output from a layer's node or nodes.

Optimizers:
Optimizers are methods or strategies for decreasing losses in a neural network by altering parameters such as weights and learning rate.
Optimizers are used to solve optimization problems by reducing the function's size.

Epochs:
Each sample in the training dataset has the ability to alter the internal model parameters once each epoch.

Batch Size:
The batch size is the number of samples processed before the model is changed.
A batch size must be more than one and less than or equal to the training dataset's sample count.

Metrics:
A metric is a function that is used to evaluate the performance of a model.
Metric functions are similar to loss functions, however the outcomes of assessing a metric aren't utilized in the training of the model.

Losses:
The loss value indicates how well or poorly a model performs after each optimization iteration.

3.6.1: CNN Architectures

As all of these models have been trained on ImageNet, they have been trained for 1000 classes. We use (include_top = False) to remove the top layer of the model, which allows to add a new output layer that has only 2 classes.
Our objective is to classify between two classes (benign and malignant).

The following values for some of the above-mentioned parameters have been used throughout the training of all models:

Activation: Sigmoid function has been used on the final layer of all models. The sigmoid function always returns a value between 0 and 1 so it is best suited for our classification problem.

Optimizer:

It's a combination of the 'gradient descent with momentum' and the 'root mean square propagation' algorithms on the surface.

Metric: Two functions have been used for all models, one of them is AUC. The other one is accuracy; it just calculates how often predictions equal labels.

Loss: Binary cross-entropy has been used for all models. It calculates the loss in cross-entropy between true and anticipated labels. Cross-entropy loss, commonly known as log loss, is a metric for evaluating the performance of a classification model whose output is a probability value between 0 and 1. As the predicted probability differs from the actual label, cross-entropy loss increases.

The following values for remaining of the before-mentioned parameters have been used differently for some models:

Model Name	Image Size	Epochs	Batch Size
EfficientNetB0	**224 x 224**	**6**	**32**
EfficientNetB1	**240 x 240**	**6**	**32**
EfficientNetB2	**260 x 260**	**6**	**16**
EfficientNetB3	**300 x 300**	**6**	**16**
MobileNet	**256 x 256**	**3**	**32**
MobileNetV2	**256 x 256**	**3**	**32**
ResNet50	**224 x 224**	**6**	**32**
ResNet50V2	**224 x 224**	**6**	**32**
VGG16	**224 x 224**	**6**	**32**
VGG19	**224 x 224**	**6**	**32**
Xception	**299 x 299**	**6**	**16**

As we know that from the above-mentioned table, three models take a higher resolution image as input compared to others, so we had to decrease the batch size to 16 in order to avoid memory errors.

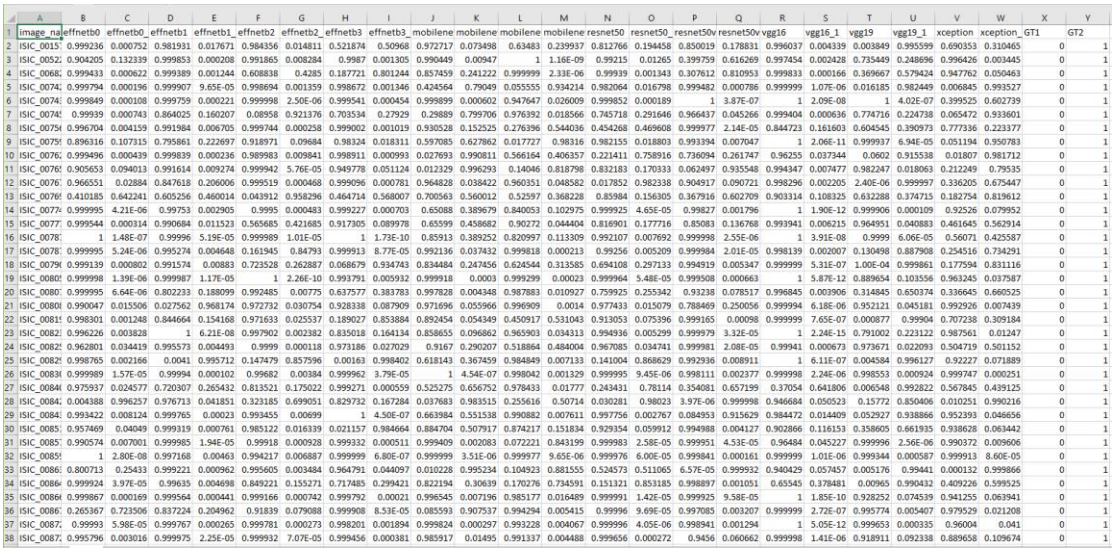
We also ran these lightweight models, MobileNet and MobileNetV2 with a batch size of 64, just for the purpose of testing.

3.6.2: Ensemble

Ensemble Methods:
Ensemble techniques are learning algorithms that create a group of classifiers and then categories incoming data points based on a (weighted) vote of their predictions. Ensemble techniques integrate several learning algorithms to produce greater prediction performance than any single learning algorithm.
All of the CNN architectures that we have used to train models give output in the form of two nodes, one represents benign and the other represents malignant. The output is in the form [0.99, 0.01], one of these values is the probability of an image being benign and the other is the probability of it being malignant.
After we have trained all of the 11 models, we then create notebooks on excel and store the predictions of all of these 11 models on our dataset.

Then we merge all these notebooks into a single excel (.csv) file, after that we add the ground truth given in the dataset to this (.csv) file.

The file looks like this:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	image_na	effnetb0	effnetb0_	effnetb1	effnetb1_	effnetb2	effnetb2_	effnetb3	effnetb3_	mobilene	mobilene	mobilene	mobilene	resnet50	resnet50_	resnet50v	resnet50v	vgg16	vgg16_1	vgg19	vgg19_1	xception	xception_	GT1	GT2
2	ISIC_0015	0.999236	0.000752	0.981931	0.017671	0.984356	0.014811	0.521874	0.50968	0.972717	0.073498	0.63483	0.239937	0.812766	0.194458	0.850019	0.178831	0.996037	0.004339	0.003849	0.995599	0.690353	0.310465	0	1
3	ISIC_0052	0.904205	0.132339	0.999853	0.000208	0.991865	0.008284	0.9987	0.001305	0.990449	0.00947	1	1.16E-09	0.99215	0.01265	0.399759	0.616269	0.997454	0.002428	0.735449	0.248696	0.996426	0.003445	0	1
4	ISIC_0068	0.999433	0.000622	0.999389	0.001244	0.608838	0.4285	0.187721	0.801244	0.857459	0.241222	0.999999	2.33E-06	0.99939	0.001343	0.307612	0.810953	0.999833	0.000166	0.369667	0.579424	0.947762	0.050463	0	1
5	ISIC_0074	0.999794	0.000196	0.999907	9.65E-05	0.998694	0.001359	0.998672	0.001346	0.424564	0.79049	0.055555	0.934214	0.982064	0.016798	0.999482	0.000786	0.999999	1.07E-06	0.016185	0.982449	0.006845	0.993527	0	1
6	ISIC_0074	0.999849	0.000108	0.999759	0.000221	0.999998	2.50E-06	0.999541	0.000454	0.999899	0.000602	0.947647	0.026009	0.999852	0.000189	1	3.87E-07	1	2.09E-08	1	4.02E-07	0.399525	0.602739	0	1
7	ISIC_0074	0.99939	0.000743	0.864025	0.160207	0.08958	0.921376	0.703534	0.27929	0.29889	0.799706	0.976392	0.018566	0.745718	0.291646	0.966437	0.045266	0.999404	0.000636	0.774716	0.224738	0.065472	0.933601	0	1
8	ISIC_0075	0.996704	0.004159	0.991984	0.006705	0.999744	0.000258	0.999002	0.001019	0.930528	0.152525	0.276396	0.544036	0.454268	0.469608	0.999977	2.14E-05	0.844723	0.161603	0.604545	0.390973	0.777336	0.223377	0	1
9	ISIC_0075	0.896316	0.107315	0.795861	0.222697	0.918971	0.09684	0.98324	0.018311	0.597085	0.627862	0.017727	0.98316	0.982155	0.018803	0.993394	0.007047	1	2.06E-11	0.999937	6.94E-05	0.051194	0.950783	0	1
10	ISIC_0076	0.999496	0.000439	0.999839	0.000236	0.989983	0.009841	0.998911	0.000993	0.027693	0.990811	0.566164	0.406357	0.221411	0.758916	0.736094	0.261747	0.96255	0.037344	0.0602	0.915538	0.01807	0.981712	0	1
11	ISIC_0076	0.905653	0.094013	0.991614	0.009274	0.999942	5.76E-05	0.949778	0.051124	0.012329	0.996293	0.14046	0.818798	0.832183	0.170333	0.062497	0.935548	0.994347	0.007477	0.982247	0.018063	0.212249	0.79535	0	1
12	ISIC_0076	0.966551	0.02884	0.847618	0.206006	0.999519	0.000468	0.999096	0.000781	0.964828	0.038422	0.960351	0.048582	0.017852	0.982338	0.904917	0.090721	0.998296	0.002205	2.40E-06	0.999997	0.336205	0.675447	0	1
13	ISIC_0076	0.410185	0.642241	0.605256	0.460014	0.043912	0.958296	0.464714	0.568007	0.700563	0.560012	0.52597	0.368228	0.85984	0.156305	0.367916	0.602709	0.903314	0.108325	0.632288	0.374715	0.182754	0.819612	0	1
14	ISIC_0077	0.999995	4.21E-06	0.99753	0.002905	0.9995	0.000483	0.999227	0.000703	0.65088	0.389679	0.840053	0.102975	0.999925	4.65E-05	0.99827	0.001796	1	1.90E-12	0.999906	0.000109	0.92526	0.079952	0	1
15	ISIC_0077	0.999544	0.000314	0.990684	0.011523	0.565685	0.421685	0.917305	0.089978	0.65599	0.458682	0.90272	0.044404	0.816901	0.177716	0.85083	0.136768	0.993941	0.006215	0.964951	0.040883	0.461645	0.562914	0	1
16	ISIC_0078	1	1.48E-07	0.99996	5.19E-05	0.999989	1.01E-05	1	1.73E-10	0.85913	0.389252	0.820997	0.113309	0.992107	0.007692	0.999998	2.55E-06	1	3.91E-08	0.9999	6.06E-05	0.56071	0.425587	0	1
17	ISIC_0078	0.999995	5.24E-06	0.995274	0.004648	0.161945	0.84793	0.999913	8.77E-05	0.992136	0.037432	0.999818	0.000213	0.99256	0.005209	0.999984	2.01E-05	0.998139	0.002007	0.130498	0.887908	0.254516	0.734291	0	1
18	ISIC_0079	0.999139	0.000802	0.991574	0.00883	0.723528	0.262887	0.068679	0.934743	0.834484	0.247456	0.624544	0.313585	0.694108	0.297133	0.994919	0.005347	0.999999	5.31E-07	1.00E-04	0.999861	0.177594	0.831116	0	1
19	ISIC_0080	0.999998	1.39E-06	0.999987	1.17E-05	1	2.26E-10	0.993791	0.005932	0.999918	0.0003	0.999299	0.00023	0.999964	5.48E-05	0.999508	0.000663	1	5.87E-12	0.889654	0.103556	0.963245	0.037587	0	1
20	ISIC_0080	0.999995	6.64E-06	0.802233	0.188099	0.992485	0.00775	0.637577	0.383783	0.997828	0.004348	0.987883	0.010927	0.759925	0.255342	0.93238	0.078517	0.996845	0.003906	0.314845	0.650374	0.336645	0.660525	0	1
21	ISIC_0080	0.990047	0.015506	0.027562	0.968174	0.972732	0.030754	0.928338	0.087909	0.971696	0.055966	0.996909	0.0014	0.977433	0.015079	0.788469	0.250056	0.999994	6.18E-06	0.952121	0.045181	0.992926	0.007439	0	1
22	ISIC_0081	0.998301	0.001248	0.844664	0.154168	0.971633	0.025537	0.189027	0.853884	0.892454	0.054349	0.450917	0.531043	0.913053	0.075396	0.999165	0.00098	0.999999	7.65E-07	0.000877	0.99904	0.707238	0.309184	0	1
23	ISIC_0082	0.996226	0.003828	1	6.21E-08	0.997902	0.002382	0.835018	0.164134	0.858655	0.096862	0.965903	0.034313	0.994936	0.005299	0.999979	3.32E-05	1	2.24E-15	0.791002	0.223122	0.987561	0.01247	0	1
24	ISIC_0082	0.962801	0.034419	0.995573	0.004493	0.9999	0.000118	0.973186	0.027029	0.9167	0.290207	0.518864	0.484004	0.967085	0.034741	0.999981	2.08E-05	0.99941	0.000673	0.973671	0.022093	0.504719	0.501152	0	1
25	ISIC_0082	0.998765	0.002166	0.0041	0.995712	0.147479	0.857596	0.00163	0.998402	0.618143	0.367459	0.984849	0.007133	0.141004	0.868629	0.992936	0.008911	1	6.11E-07	0.004584	0.996127	0.92227	0.071889	0	1
26	ISIC_0083	0.999989	1.57E-05	0.99994	0.000102	0.99682	0.00384	0.999962	3.79E-05	1	4.54E-07	0.998042	0.001329	0.999995	9.45E-06	0.998111	0.002377	0.999998	2.24E-06	0.998553	0.000924	0.999747	0.000251	0	1
27	ISIC_0084	0.975937	0.024577	0.720307	0.265432	0.813521	0.175022	0.999271	0.000559	0.525275	0.656752	0.978433	0.01777	0.243431	0.78114	0.354081	0.657199	0.37054	0.641806	0.006548	0.992822	0.567845	0.439125	0	1
28	ISIC_0084	0.004388	0.996257	0.976713	0.041851	0.323185	0.699051	0.829732	0.167284	0.037683	0.983515	0.255616	0.50714	0.030281	0.98023	3.97E-06	0.999998	0.946684	0.050523	0.15772	0.850406	0.010251	0.990216	0	1
29	ISIC_0084	0.993422	0.008124	0.999765	0.00023	0.993455	0.00699	1	4.50E-07	0.663984	0.551538	0.990882	0.007611	0.997756	0.002767	0.084953	0.915629	0.984472	0.014409	0.052927	0.938866	0.952393	0.046656	0	1
30	ISIC_0085	0.957469	0.04049	0.999319	0.000761	0.985122	0.016339	0.021157	0.984664	0.884704	0.507917	0.874217	0.151834	0.929354	0.059912	0.994988	0.004127	0.902866	0.116153	0.358605	0.661935	0.938628	0.063442	0	1
31	ISIC_0085	0.990574	0.007001	0.999985	1.94E-05	0.99918	0.000928	0.999332	0.000511	0.999409	0.002083	0.072221	0.843199	0.999983	2.58E-05	0.999951	4.53E-05	0.96484	0.045227	0.999996	2.56E-06	0.990372	0.009606	0	1
32	ISIC_0085	1	2.80E-08	0.997168	0.00463	0.994217	0.006887	0.999999	6.80E-07	0.999999	3.51E-06	0.999977	9.65E-06	0.999976	6.00E-05	0.999841	0.000161	0.999999	1.01E-06	0.999344	0.000587	0.999913	8.60E-05	0	1
33	ISIC_0086	0.800713	0.25433	0.999221	0.000962	0.995605	0.003484	0.964791	0.044097	0.010228	0.995234	0.104923	0.881555	0.524573	0.511065	6.57E-05	0.999932	0.940429	0.057457	0.005176	0.99441	0.000132	0.999866	0	1
34	ISIC_0086	0.999924	3.97E-05	0.99635	0.004698	0.849221	0.155271	0.717485	0.299421	0.822194	0.30639	0.170276	0.734591	0.151321	0.853185	0.998897	0.001051	0.65545	0.378481	0.00965	0.990432	0.409226	0.599525	0	1
35	ISIC_0086	0.999867	0.000169	0.999564	0.000441	0.999166	0.000742	0.999792	0.00021	0.996545	0.007196	0.985177	0.016489	0.999991	1.42E-05	0.999925	9.58E-05	1	1.85E-10	0.928252	0.074539	0.941255	0.063941	0	1
36	ISIC_0086	0.265367	0.723506	0.837224	0.204962	0.91839	0.079088	0.999908	8.53E-05	0.085593	0.907537	0.994294	0.005415	0.99996	9.69E-05	0.997085	0.003207	0.999999	2.72E-07	0.995774	0.005407	0.979529	0.021208	0	1
37	ISIC_0087	0.99993	5.98E-05	0.999767	0.000265	0.999781	0.000273	0.998201	0.001894	0.999824	0.000297	0.993228	0.004067	0.999996	4.05E-06	0.998941	0.001294	1	5.05E-12	0.999653	0.000335	0.96004	0.041	0	1
38	ISIC_0087	0.995796	0.003016	0.999975	2.25E-05	0.999932	7.07E-05	0.999456	0.000381	0.985917	0.01495	0.991337	0.004488	0.999656	0.000272	0.9456	0.060662	0.999998	1.41E-06	0.918911	0.092338	0.889658	0.109674	0	1

Figure 3.4: Final notebook form

As we can see from the figure above, we have 22 columns for 11 models, 1st column (A) contains image names, then 2 columns for each model and last 2 columns (X and Y) for ground truth.

Now that we have the outputs of all the 11 models that we have run, we are going to run a custom CNN model using data from this notebook with the outputs of those 11 models as its inputs and malignant or benign classification as output.

The new neural network model has 1 input layer and 6 dense layers. The following are the details:

- Input layer: It has 22 nodes
- 1st layer (Dense): It has 32 nodes
- 2nd layer (Dense): It has 64 nodes
- 3rd layer (Dense): It has 128 nodes
- 4th layer (Dense): It has 256 nodes
- 5th layer (Dense): It has 512 nodes
- Output layer (Dense): It has 2 nodes

We can see the details in the following image:

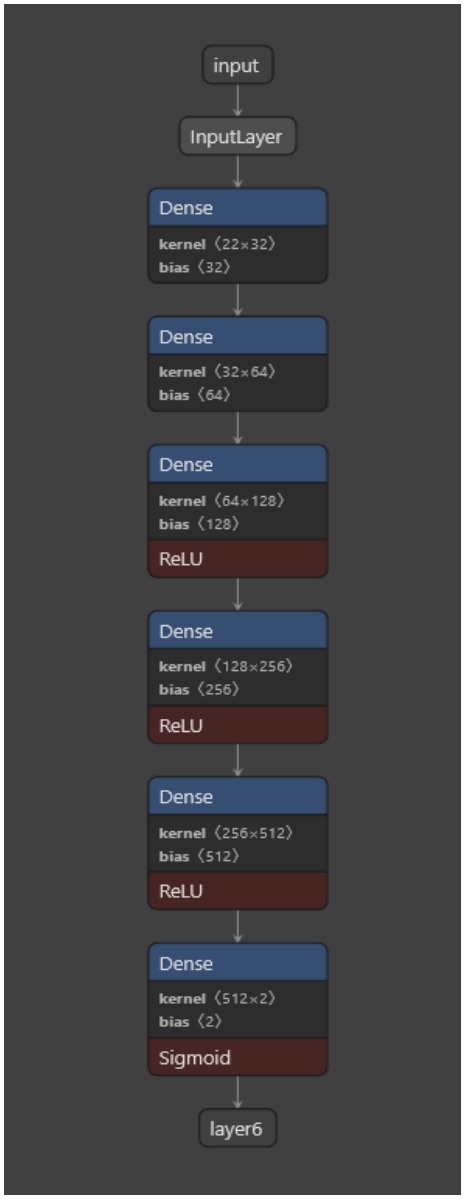


Figure 3.5: Ensemble Model Details

3.7: Results

3.7.1: CNN Architectures

- EfficientNet-B0
  We can see the accuracy and loss in the image below:

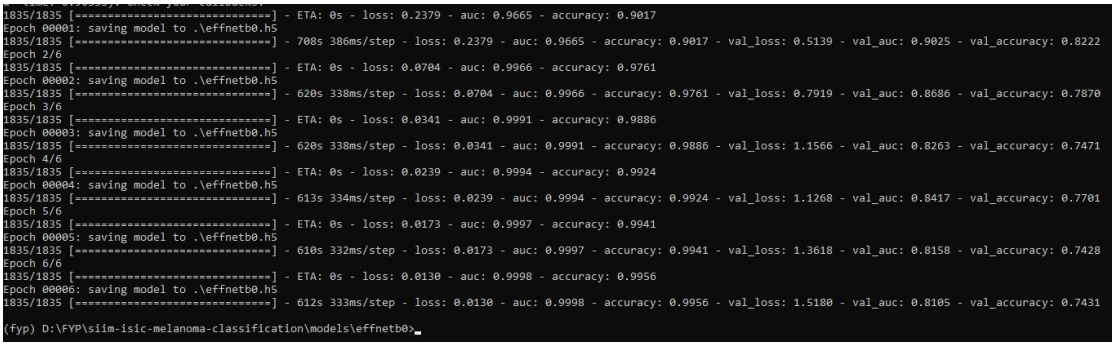


Figure 3.6: EfficientNet-B0

At the end of training:

- Train loss = 0.0130
- Train accuracy = 0.9956
- Validation loss = 1.5180
- Validation accuracy = 0.7431

A graph to compare train and validation accuracy with respect to epochs is given below:

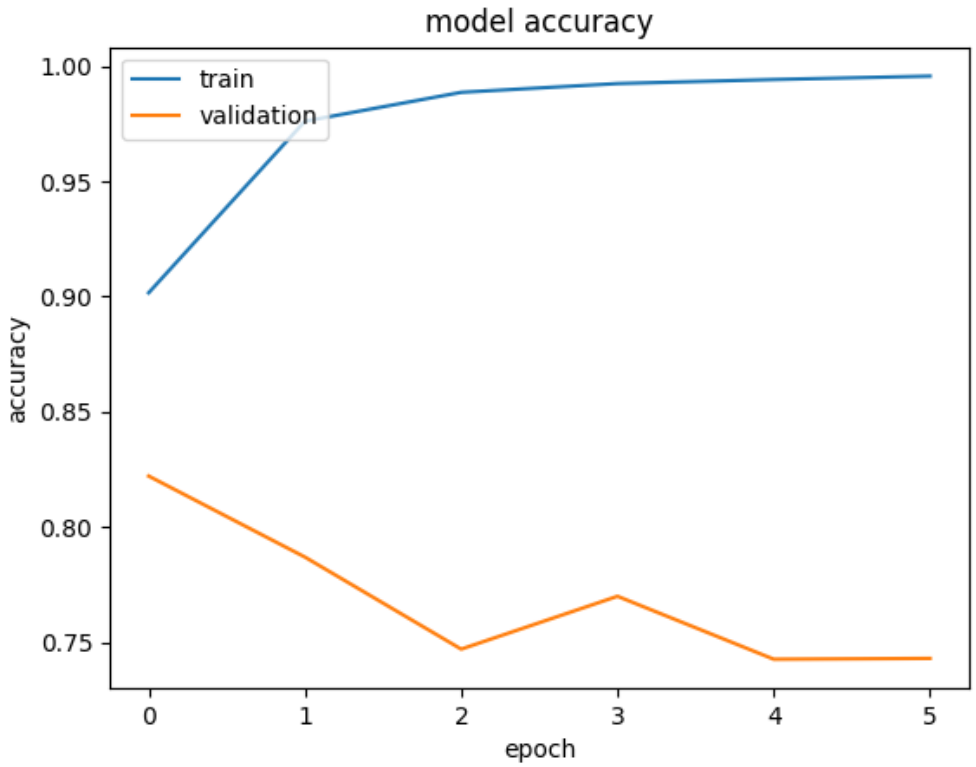


Figure 3.7: EfficientNet-B0 Graph

- EfficientNet-B1
  We can see the accuracy and loss in the image below:

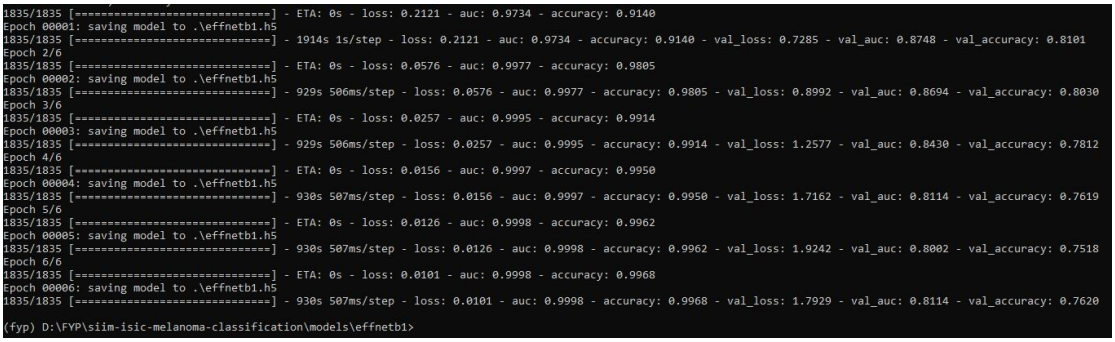


Figure 3.8: EfficientNet-B1

At the end of training:

- Train loss = 0.0101
- Train accuracy = 0.9968
- Validation loss = 1.7929
- Validation accuracy = 0.7620

A graph to compare train and validation accuracy with respect to epochs is given below:

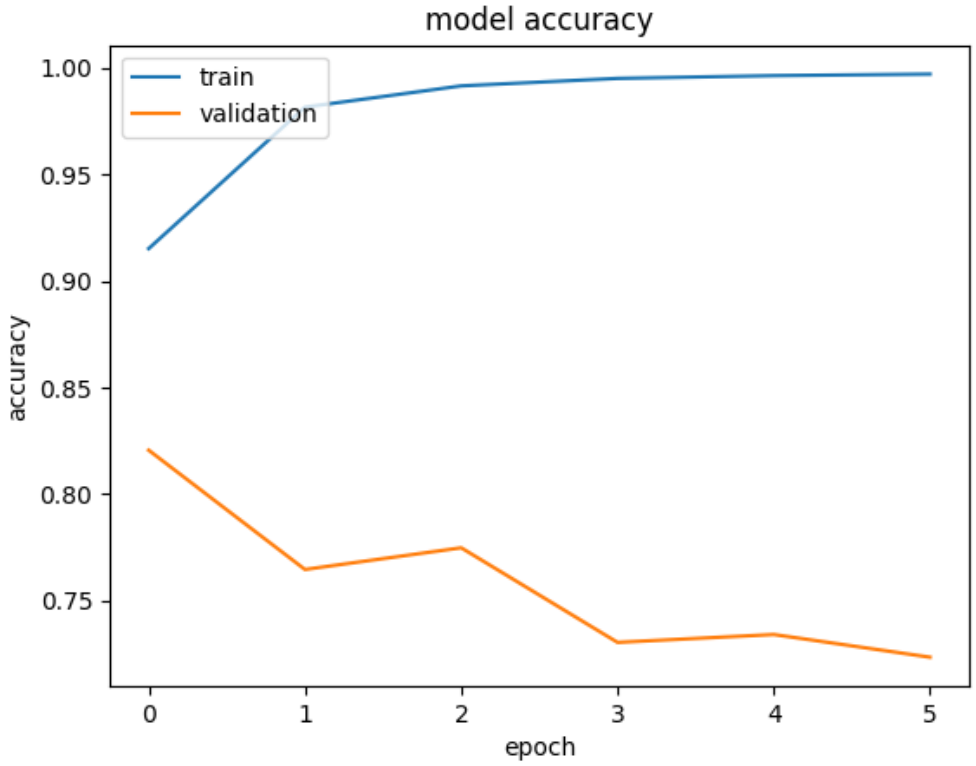


Figure 3.9: EfficientNet-B1 Graph

- EfficientNet-B2
  We can see the accuracy and loss in the image below:

```
3671/3671 [==============================] - ETA: 0s - loss: 0.1877 - auc: 0.9788 - accuracy: 0.9259
Epoch 00001: saving model to .\effnetb2.h5
3671/3671 [==============================] - 7098s 2s/step - loss: 0.1877 - auc: 0.9788 - accuracy: 0.9259 - val_loss: 0.8538 - val_auc: 0.8579 - val_accuracy: 0.7793
Epoch 2/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0500 - auc: 0.9982 - accuracy: 0.9834
Epoch 00002: saving model to .\effnetb2.h5
3671/3671 [==============================] - 6842s 2s/step - loss: 0.0500 - auc: 0.9982 - accuracy: 0.9834 - val_loss: 1.2111 - val_auc: 0.8217 - val_accuracy: 0.7382
Epoch 3/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0242 - auc: 0.9994 - accuracy: 0.9919
Epoch 00003: saving model to .\effnetb2.h5
3671/3671 [==============================] - 6859s 2s/step - loss: 0.0242 - auc: 0.9994 - accuracy: 0.9919 - val_loss: 1.4759 - val_auc: 0.8181 - val_accuracy: 0.7565
Epoch 4/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0180 - auc: 0.9996 - accuracy: 0.9940
Epoch 00004: saving model to .\effnetb2.h5
3671/3671 [==============================] - 6868s 2s/step - loss: 0.0180 - auc: 0.9996 - accuracy: 0.9940 - val_loss: 1.4570 - val_auc: 0.8295 - val_accuracy: 0.7572
Epoch 5/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0126 - auc: 0.9998 - accuracy: 0.9956
Epoch 00005: saving model to .\effnetb2.h5
3671/3671 [==============================] - 6836s 2s/step - loss: 0.0126 - auc: 0.9998 - accuracy: 0.9956 - val_loss: 1.9644 - val_auc: 0.8036 - val_accuracy: 0.7462
Epoch 6/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0103 - auc: 0.9998 - accuracy: 0.9968
Epoch 00006: saving model to .\effnetb2.h5
3671/3671 [==============================] - 6300s 2s/step - loss: 0.0103 - auc: 0.9998 - accuracy: 0.9968 - val_loss: 2.8518 - val_auc: 0.7607 - val_accuracy: 0.7190

(fyp) D:\FYP\siim-isic-melanoma-classification\models\effnetb2>
```

Figure 3.10: EfficientNet-B2

At the end of training:

- Train loss = 0.0103
- Train accuracy = 0.9998
- Validation loss = 2.8518
- Validation accuracy = 0.7190

A graph to compare train and validation accuracy with respect to epochs is given below:

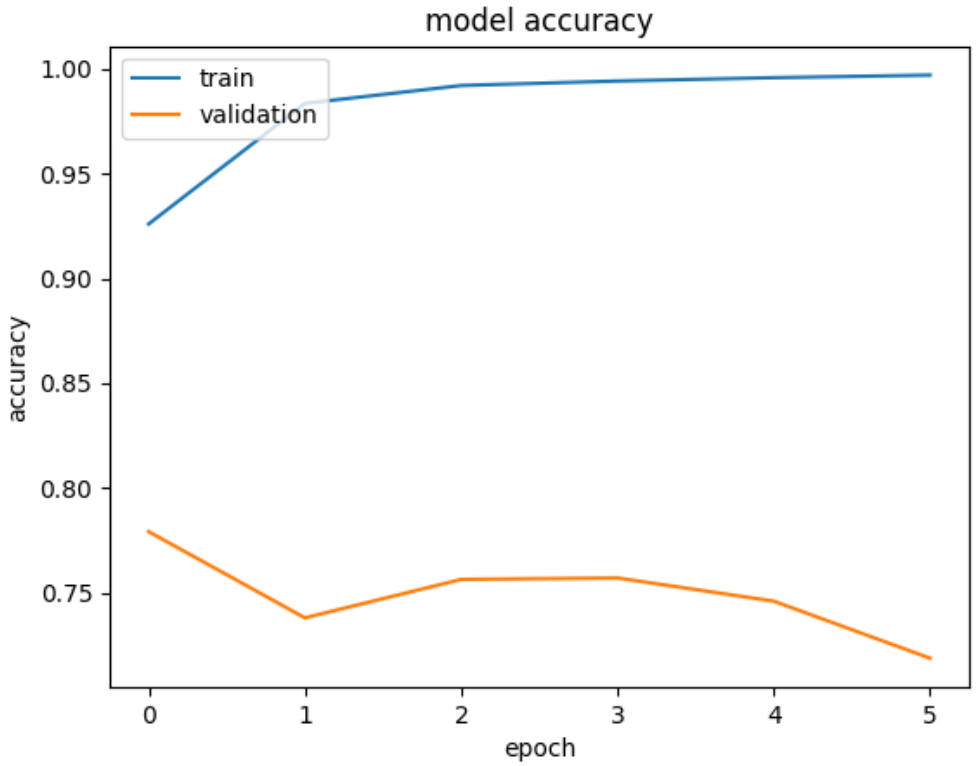


Figure 3.11: EfficientNet-B2 Graph

- EfficientNet-B3
  We can see the accuracy and loss in the image below:

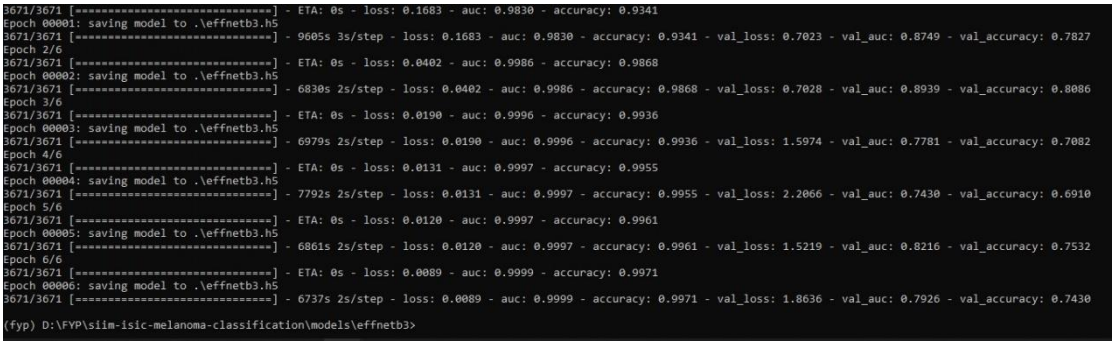


Figure 3.12: EfficientNet-B3

At the end of training:

- Train loss = 0.0089
- Train accuracy = 0.9971
- Validation loss = 1.8636
- Validation accuracy = 0.7430

A graph to compare train and validation accuracy with respect to epochs is given below:

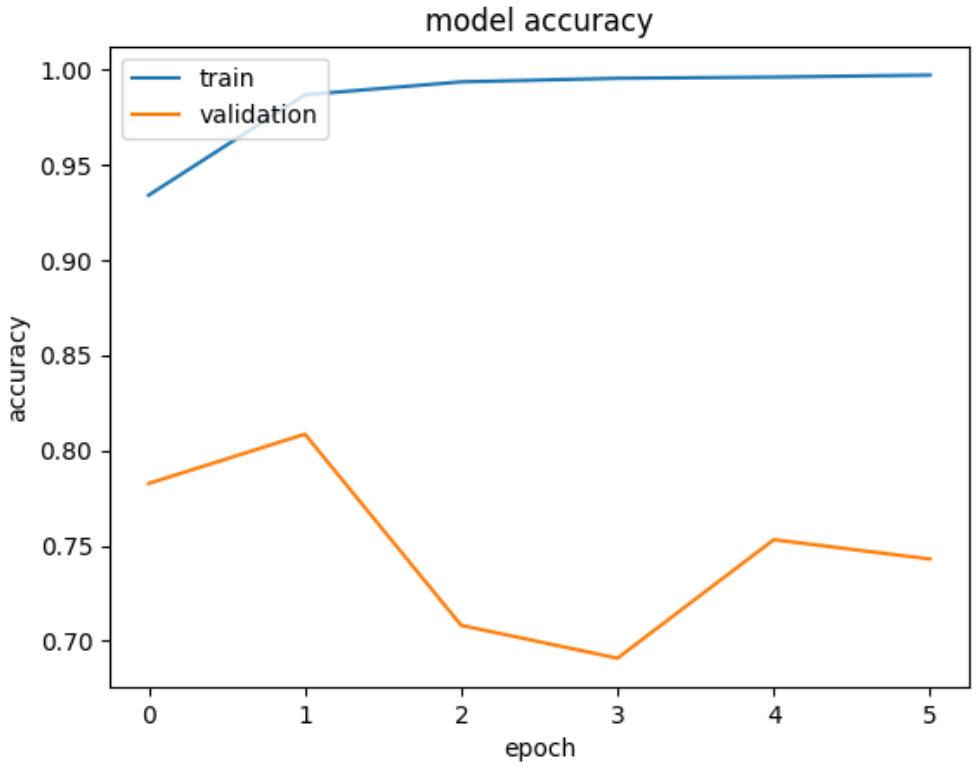


Figure 3.13: EfficientNet-B3 Graph

- MobileNet
  We can see the accuracy and loss in the image below:

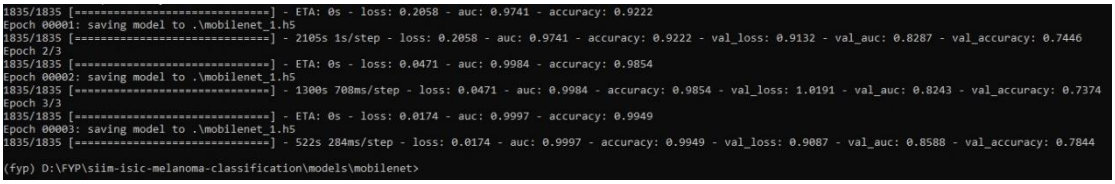


Figure 3.14: MobileNet

At the end of training:

- Train loss = 0.0174
- Train accuracy = 0.9949
- Validation loss = 0.8588
- Validation accuracy = 0.7844

A graph to compare train and validation accuracy with respect to epochs is given below:

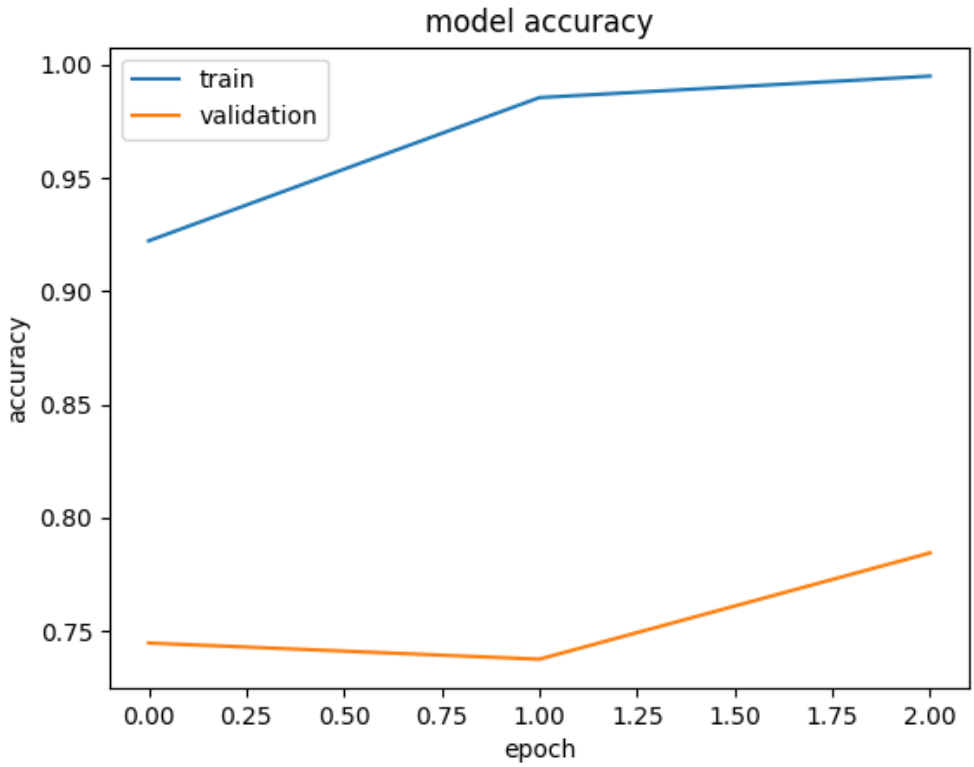


Figure 3.15: MobileNet Graph

- MobileNetV2
  We can see the accuracy and loss in the image below:

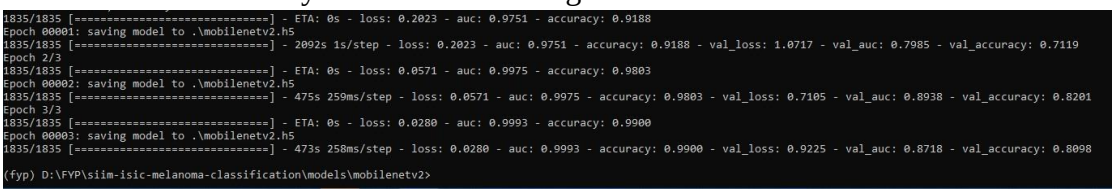


Figure 3.16: MobileNetV2

At the end of training:

- Train loss = 0.0280
- Train accuracy = 0.9900
- Validation loss = 0.9225
- Validation accuracy = 0.8098

A graph to compare train and validation accuracy with respect to epochs is given below:

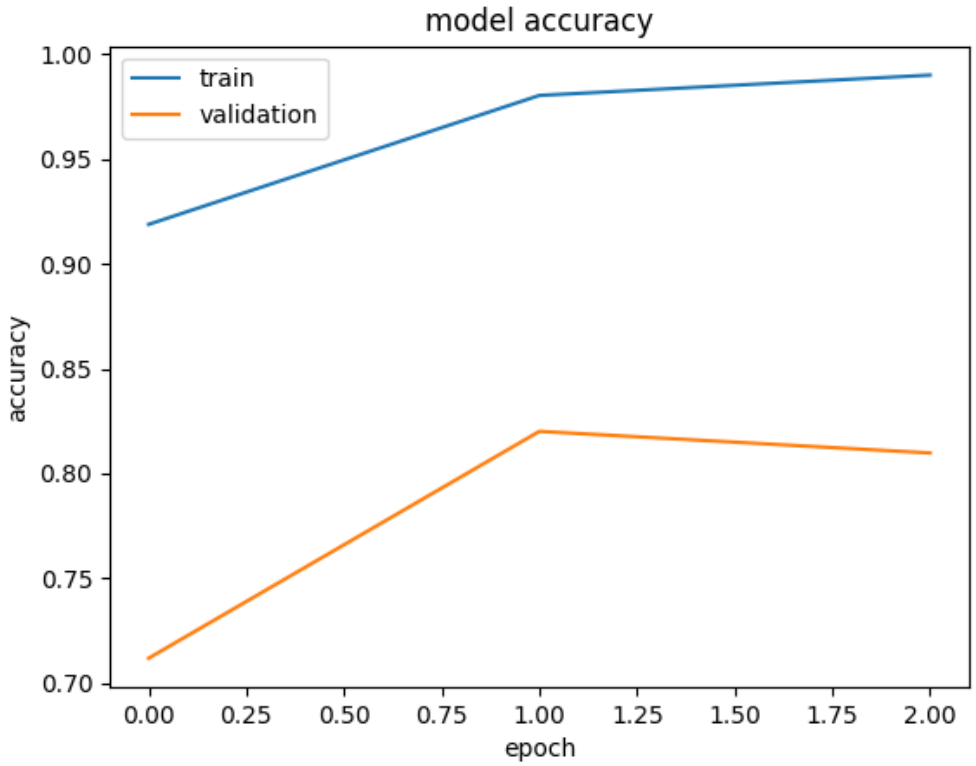


Figure 3.17: MobileNetV2 Graph

- ResNet50
  We can see the accuracy and loss in the image below:

```
1835/1835 [==============================] - ETA: 0s - loss: 0.1374 - auc: 0.9882 - accuracy: 0.9476
Epoch 00001: saving model to .\resnet50.h5
1835/1835 [==============================] - 2039s 1s/step - loss: 0.1374 - auc: 0.9882 - accuracy: 0.9476 - val_loss: 1.2930 - val_auc: 0.8012 - val_accuracy: 0.7466
Epoch 2/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0346 - auc: 0.9988 - accuracy: 0.9885
Epoch 00002: saving model to .\resnet50.h5
1835/1835 [==============================] - 807s 440ms/step - loss: 0.0346 - auc: 0.9988 - accuracy: 0.9885 - val_loss: 0.8344 - val_auc: 0.8735 - val_accuracy: 0.8039
Epoch 3/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0247 - auc: 0.9994 - accuracy: 0.9916
Epoch 00003: saving model to .\resnet50.h5
1835/1835 [==============================] - 870s 474ms/step - loss: 0.0247 - auc: 0.9994 - accuracy: 0.9916 - val_loss: 1.3351 - val_auc: 0.8308 - val_accuracy: 0.7619
Epoch 4/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0170 - auc: 0.9996 - accuracy: 0.9943
Epoch 00004: saving model to .\resnet50.h5
1835/1835 [==============================] - 601s 327ms/step - loss: 0.0170 - auc: 0.9996 - accuracy: 0.9943 - val_loss: 1.2451 - val_auc: 0.8264 - val_accuracy: 0.7527
Epoch 5/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0152 - auc: 0.9996 - accuracy: 0.9948
Epoch 00005: saving model to .\resnet50.h5
1835/1835 [==============================] - 600s 327ms/step - loss: 0.0152 - auc: 0.9996 - accuracy: 0.9948 - val_loss: 1.6843 - val_auc: 0.8009 - val_accuracy: 0.7434
Epoch 6/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0124 - auc: 0.9997 - accuracy: 0.9964
Epoch 00006: saving model to .\resnet50.h5
1835/1835 [==============================] - 604s 329ms/step - loss: 0.0124 - auc: 0.9997 - accuracy: 0.9964 - val_loss: 1.6049 - val_auc: 0.8007 - val_accuracy: 0.7414

(fyp) D:\FYP\siim-isic-melanoma-classification\models\resnet50>
```

Figure 3.18: ResNet50

At the end of training:

- Train loss = 0.0124

- Train accuracy = 0.9964
- Validation loss = 1.6049
- Validation accuracy = 0.7414

A graph to compare train and validation accuracy with respect to epochs is given below:

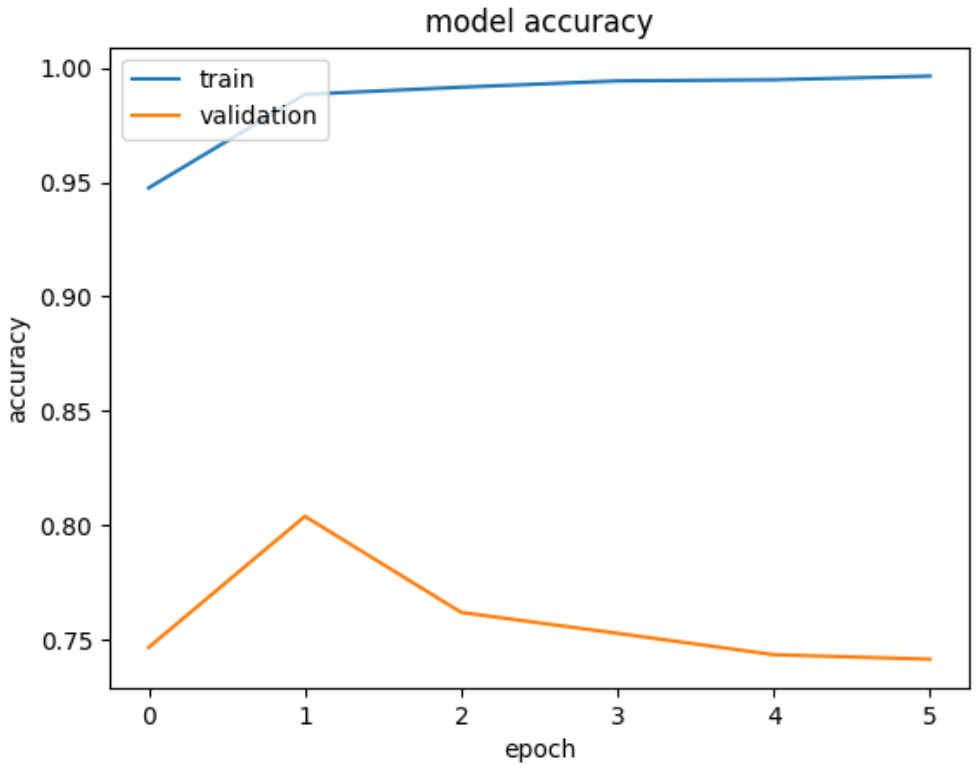


Figure 3.19: ResNet50 Graph

- ResNet50V2
  We can see the accuracy and loss in the image below:

```
1835/1835 [==============================] - ETA: 0s - loss: 0.1396 - auc: 0.9880 - accuracy: 0.9467
Epoch 00001: saving model to .\resnet50v2.h5
1835/1835 [==============================] - 538s 293ms/step - loss: 0.1396 - auc: 0.9880 - accuracy: 0.9467 - val_loss: 1.0202 - val_auc: 0.8277 - val_accuracy: 0.7576
Epoch 2/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0353 - auc: 0.9990 - accuracy: 0.9879
Epoch 00002: saving model to .\resnet50v2.h5
1835/1835 [==============================] - 514s 280ms/step - loss: 0.0353 - auc: 0.9990 - accuracy: 0.9879 - val_loss: 1.3583 - val_auc: 0.7839 - val_accuracy: 0.7031
Epoch 3/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0205 - auc: 0.9995 - accuracy: 0.9932
Epoch 00003: saving model to .\resnet50v2.h5
1835/1835 [==============================] - 515s 281ms/step - loss: 0.0205 - auc: 0.9995 - accuracy: 0.9932 - val_loss: 1.3802 - val_auc: 0.7838 - val_accuracy: 0.7026
Epoch 4/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0178 - auc: 0.9995 - accuracy: 0.9943
Epoch 00004: saving model to .\resnet50v2.h5
1835/1835 [==============================] - 515s 280ms/step - loss: 0.0178 - auc: 0.9995 - accuracy: 0.9943 - val_loss: 1.6315 - val_auc: 0.7705 - val_accuracy: 0.7017
Epoch 5/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0155 - auc: 0.9996 - accuracy: 0.9949
Epoch 00005: saving model to .\resnet50v2.h5
1835/1835 [==============================] - 515s 281ms/step - loss: 0.0155 - auc: 0.9996 - accuracy: 0.9949 - val_loss: 1.5262 - val_auc: 0.8055 - val_accuracy: 0.7381
Epoch 6/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0127 - auc: 0.9997 - accuracy: 0.9957
Epoch 00006: saving model to .\resnet50v2.h5
1835/1835 [==============================] - 515s 280ms/step - loss: 0.0127 - auc: 0.9997 - accuracy: 0.9957 - val_loss: 1.8766 - val_auc: 0.7650 - val_accuracy: 0.7035

(fyp) D:\FYP\siim-isic-melanoma-classification\models\resnet50v2>
```

Figure 3.20: ResNet50V2

At the end of training:

- Train loss = 0.0127
- Train accuracy = 0.9957
- Validation loss = 1.8766

- Validation accuracy = 0.7035

A graph to compare train and validation accuracy with respect to epochs is given below:

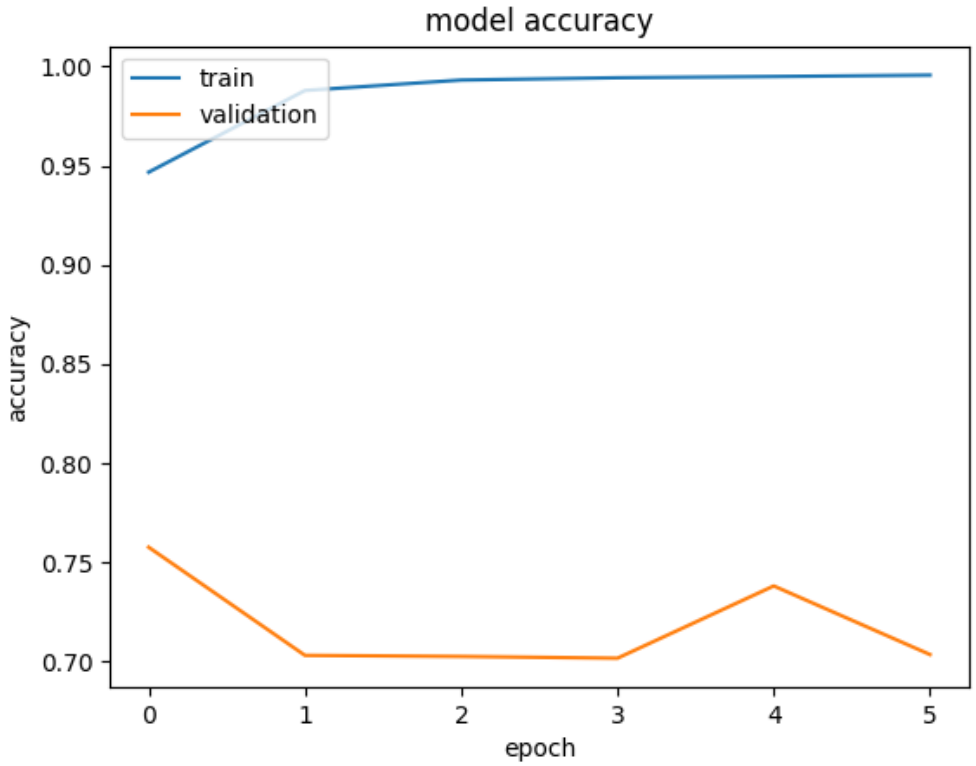


Figure 3.21: ResNet50V2 Graph

- VGG16
  We can see the accuracy and loss in the image below:

```
1835/1835 [==============================] - ETA: 0s - loss: 0.3361 - auc: 0.9309 - accuracy: 0.8514
Epoch 00001: saving model to .\vgg16.h5
1835/1835 [==============================] - 1937s 1s/step - loss: 0.3361 - auc: 0.9309 - accuracy: 0.8514 - val_loss: 0.3629 - val_auc: 0.9312 - val_accuracy: 0.8487
Epoch 2/6
1835/1835 [==============================] - ETA: 0s - loss: 0.1437 - auc: 0.9870 - accuracy: 0.9460
Epoch 00002: saving model to .\vgg16.h5
1835/1835 [==============================] - 986s 537ms/step - loss: 0.1437 - auc: 0.9870 - accuracy: 0.9460 - val_loss: 0.8117 - val_auc: 0.8534 - val_accuracy: 0.7665
Epoch 3/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0683 - auc: 0.9965 - accuracy: 0.9763
Epoch 00003: saving model to .\vgg16.h5
1835/1835 [==============================] - 624s 340ms/step - loss: 0.0683 - auc: 0.9965 - accuracy: 0.9763 - val_loss: 0.6993 - val_auc: 0.8900 - val_accuracy: 0.8136
Epoch 4/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0449 - auc: 0.9982 - accuracy: 0.9847
Epoch 00004: saving model to .\vgg16.h5
1835/1835 [==============================] - 627s 342ms/step - loss: 0.0449 - auc: 0.9982 - accuracy: 0.9847 - val_loss: 0.5426 - val_auc: 0.9087 - val_accuracy: 0.8230
Epoch 5/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0283 - auc: 0.9991 - accuracy: 0.9901
Epoch 00005: saving model to .\vgg16.h5
1835/1835 [==============================] - 625s 341ms/step - loss: 0.0283 - auc: 0.9991 - accuracy: 0.9901 - val_loss: 1.4861 - val_auc: 0.8063 - val_accuracy: 0.7237
Epoch 6/6
1835/1835 [==============================] - ETA: 0s - loss: 0.0233 - auc: 0.9992 - accuracy: 0.9926
Epoch 00006: saving model to .\vgg16.h5
1835/1835 [==============================] - 630s 343ms/step - loss: 0.0233 - auc: 0.9992 - accuracy: 0.9926 - val_loss: 1.4696 - val_auc: 0.7899 - val_accuracy: 0.6960

(fyp) D:\FYP\siim-isic-melanoma-classification\models\vgg16>
```

Figure 3.22: VGG16

At the end of training:

- Train loss = 0.0233
- Train accuracy = 0.9926
- Validation loss = 1.4696
- Validation accuracy = 0.6960

A graph to compare train and validation accuracy with respect to epochs is given below:

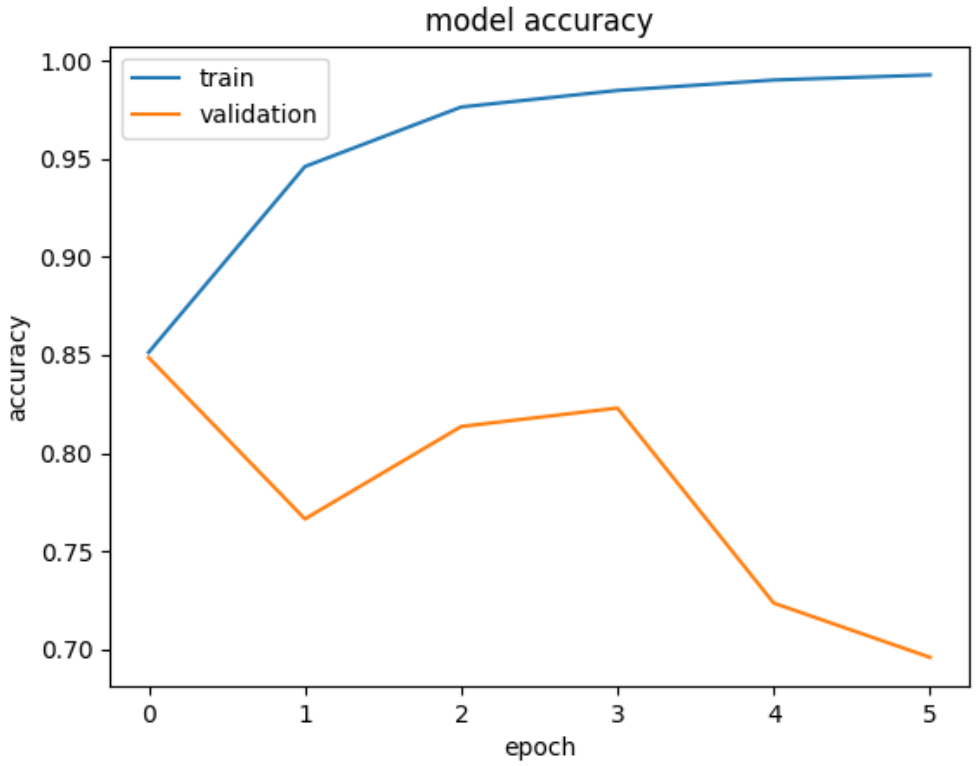


Figure 3.23: VGG16 Graph

- VGG19
  We can see the accuracy and loss in the image below:

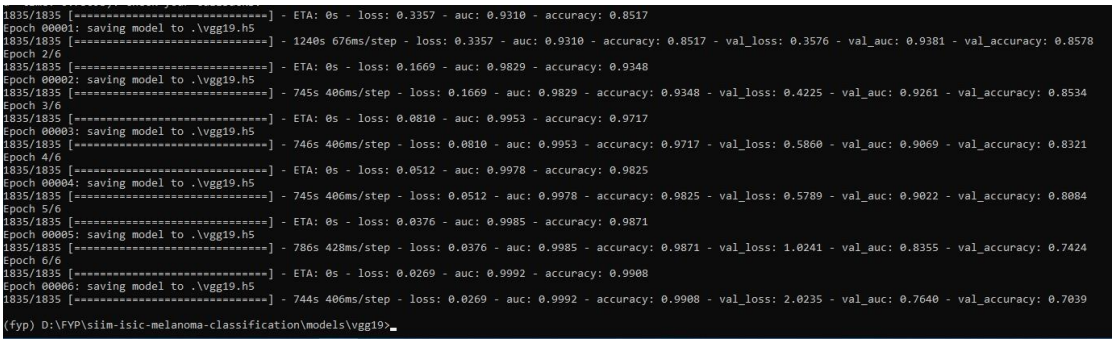


Figure 3.24: VGG19

At the end of training:

- Train loss = 0.0269
- Train accuracy = 0.9908
- Validation loss = 2.0235
- Validation accuracy = 0.7039

A graph to compare train and validation accuracy with respect to epochs is given below:

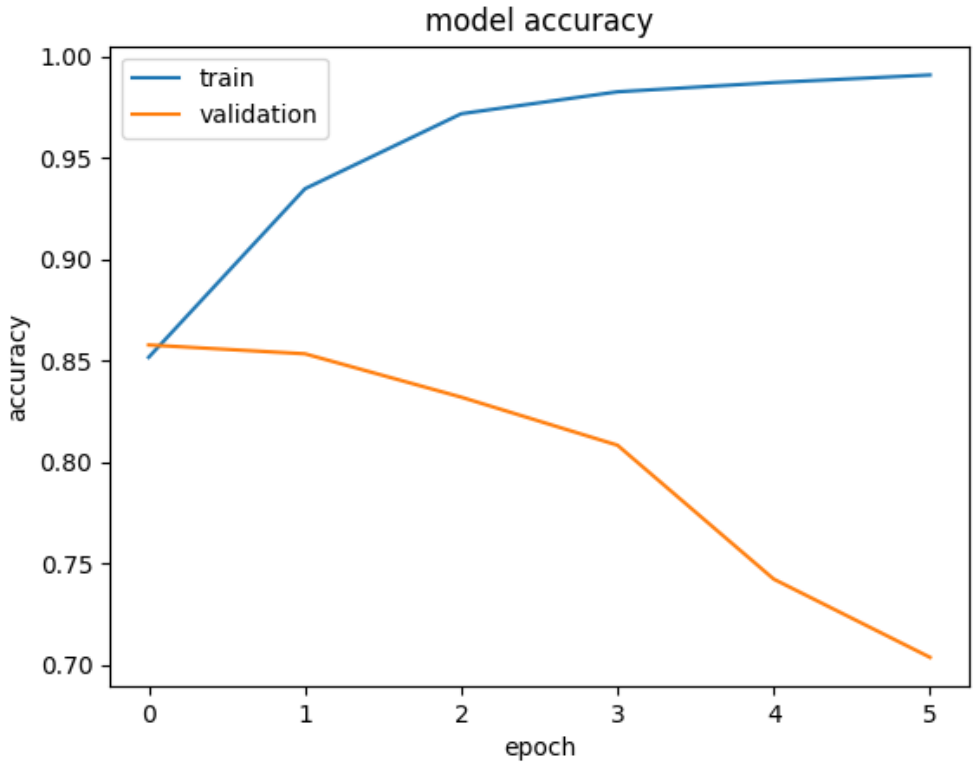


Figure 3.25: VGG19 Graph

- Xception
  We can see the accuracy and loss in the image below:

```
3671/3671 [==============================] - ETA: 0s - loss: 0.1244 - auc: 0.9906 - accuracy: 0.9526
Epoch 00001: saving model to .\xception.h5
3671/3671 [==============================] - 7331s 2s/step - loss: 0.1244 - auc: 0.9906 - accuracy: 0.9526 - val_loss: 1.3413 - val_auc: 0.7925 - val_accuracy: 0.7106
Epoch 2/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0219 - auc: 0.9995 - accuracy: 0.9934
Epoch 00002: saving model to .\xception.h5
3671/3671 [==============================] - 7090s 2s/step - loss: 0.0219 - auc: 0.9995 - accuracy: 0.9934 - val_loss: 1.3264 - val_auc: 0.8067 - val_accuracy: 0.7218
Epoch 3/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0135 - auc: 0.9996 - accuracy: 0.9959
Epoch 00003: saving model to .\xception.h5
3671/3671 [==============================] - 7117s 2s/step - loss: 0.0135 - auc: 0.9996 - accuracy: 0.9959 - val_loss: 1.4565 - val_auc: 0.7954 - val_accuracy: 0.7103
Epoch 4/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0122 - auc: 0.9997 - accuracy: 0.9964
Epoch 00004: saving model to .\xception.h5
3671/3671 [==============================] - 7053s 2s/step - loss: 0.0122 - auc: 0.9997 - accuracy: 0.9964 - val_loss: 2.0605 - val_auc: 0.7453 - val_accuracy: 0.6773
Epoch 5/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0091 - auc: 0.9998 - accuracy: 0.9973
Epoch 00005: saving model to .\xception.h5
3671/3671 [==============================] - 7020s 2s/step - loss: 0.0091 - auc: 0.9998 - accuracy: 0.9973 - val_loss: 2.0742 - val_auc: 0.7708 - val_accuracy: 0.7120
Epoch 6/6
3671/3671 [==============================] - ETA: 0s - loss: 0.0073 - auc: 0.9999 - accuracy: 0.9977
Epoch 00006: saving model to .\xception.h5
3671/3671 [==============================] - 7023s 2s/step - loss: 0.0073 - auc: 0.9999 - accuracy: 0.9977 - val_loss: 1.3922 - val_auc: 0.8132 - val_accuracy: 0.7296

(fyp) D:\FYP\siim-isic-melanoma-classification\models\xception>
```

Figure 3.26: Xception

At the end of training:

- Train loss = 0.0073
- Train accuracy = 0.9977
- Validation loss = 1.3922
- Validation accuracy = 0.7296

A graph to compare train and validation accuracy with respect to epochs is given below:

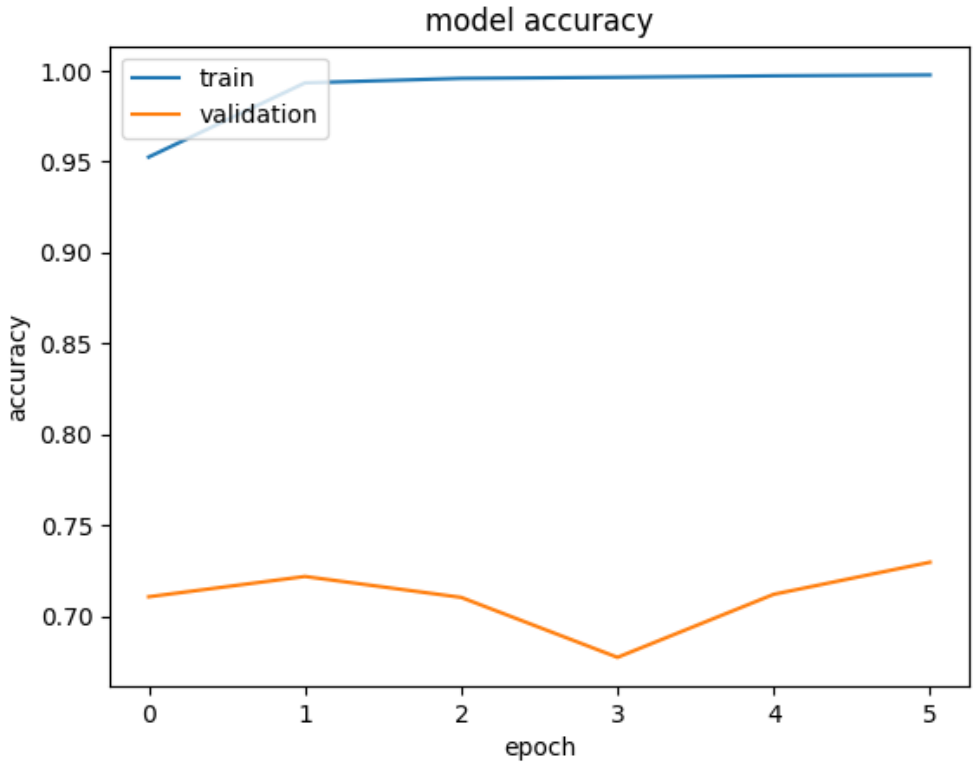


Figure 3.27: Xception Graph

3.7.2: Ensemble

After we have collected the predictions of all CNN models, we use them as an input (X) for a new neural network and use the ground truth as output (Y).

We can see the accuracy and loss in the image below:

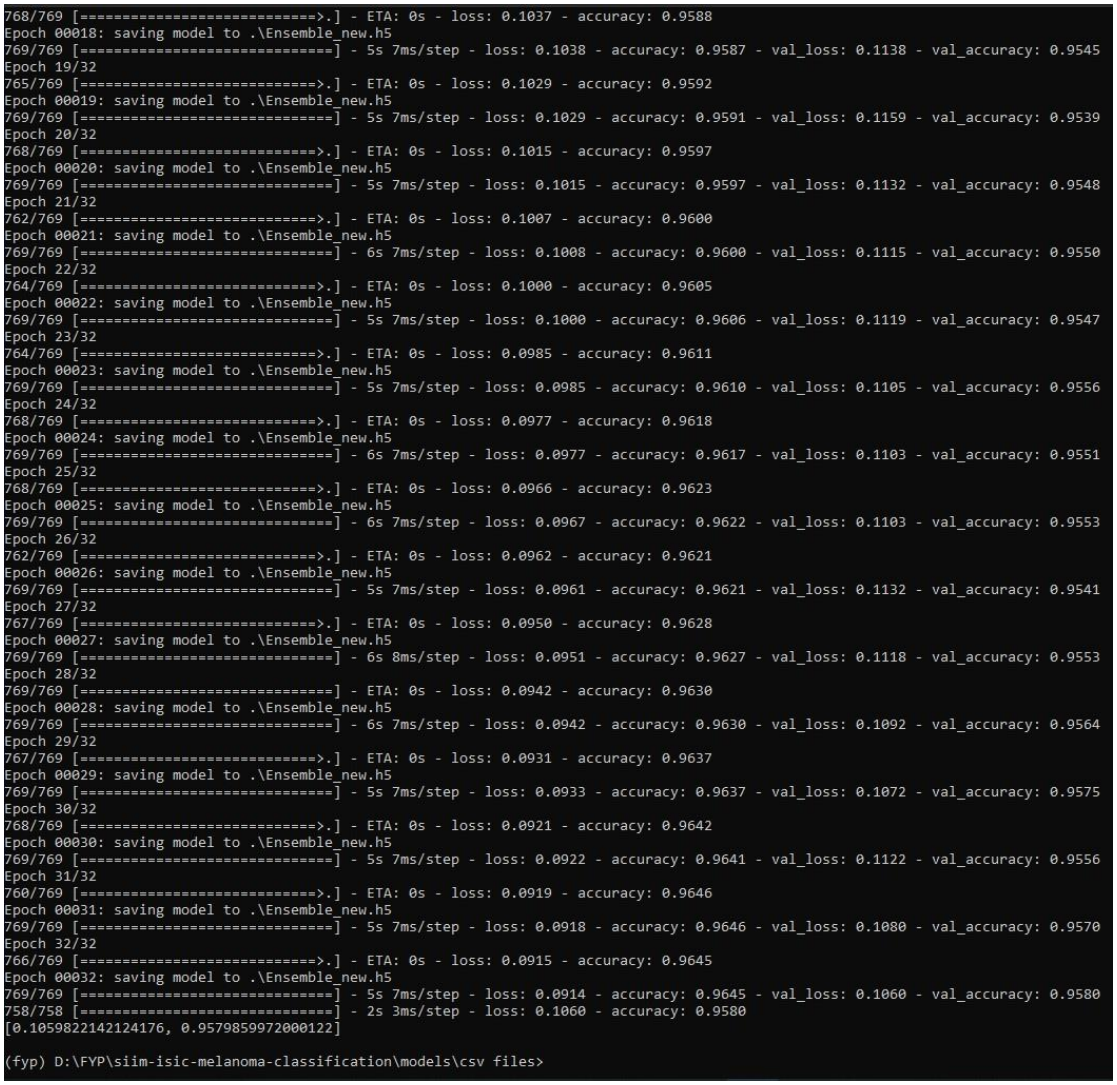


Figure 3.28: Ensemble

At the end of training:

- Train loss = 0.1291
- Train accuracy = 0.9524
- Validation loss = 0.1361
- Validation accuracy = 0.9502

A graph to compare train and validation accuracy with respect to epochs is given below:

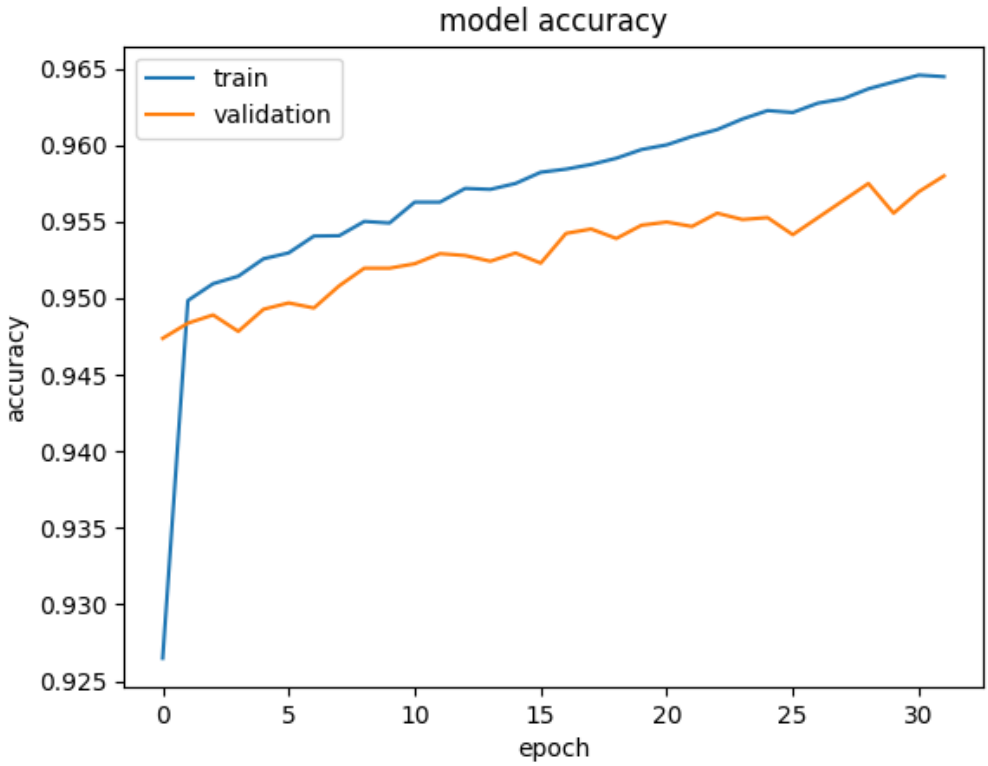


Figure 3.29: Ensemble Accuracy Graph

A graph to compare train and validation loss with respect to epochs is given below:

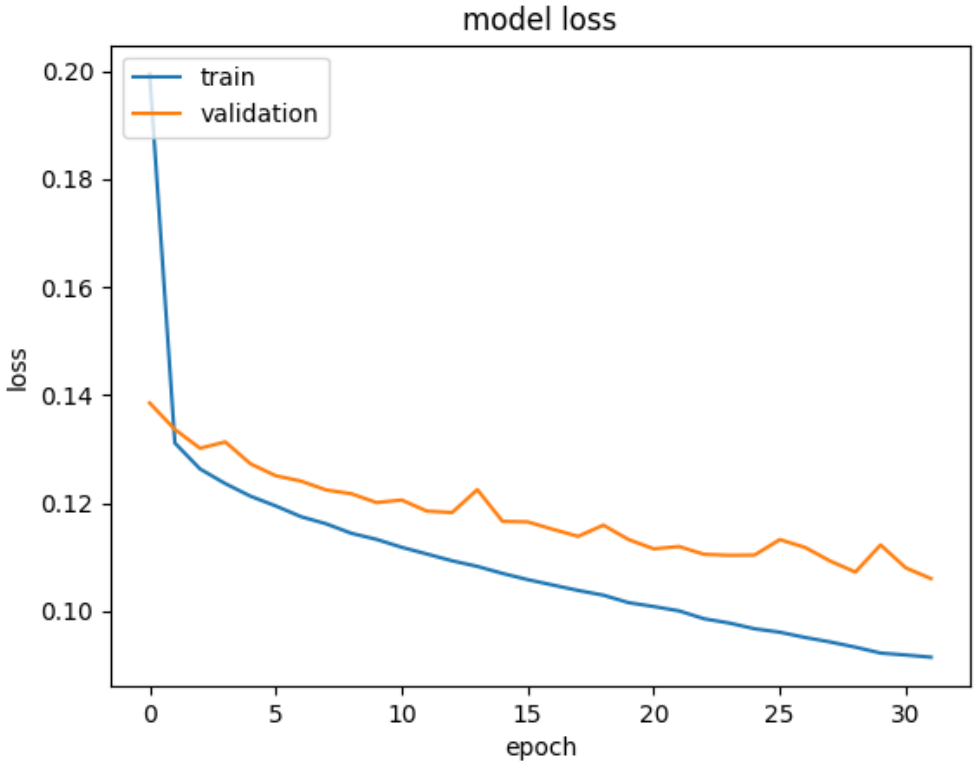


Figure 3.30: Ensemble Loss Graph

We also ran our model on the test images in the **ISIC 2020 Challenge Dataset** by Kaggle and submitted the results as well. Although it was a late submission, but still out of curiosity we wanted to test its accuracy on an unlabeled dataset. We got an accuracy of 70% on the test dataset.

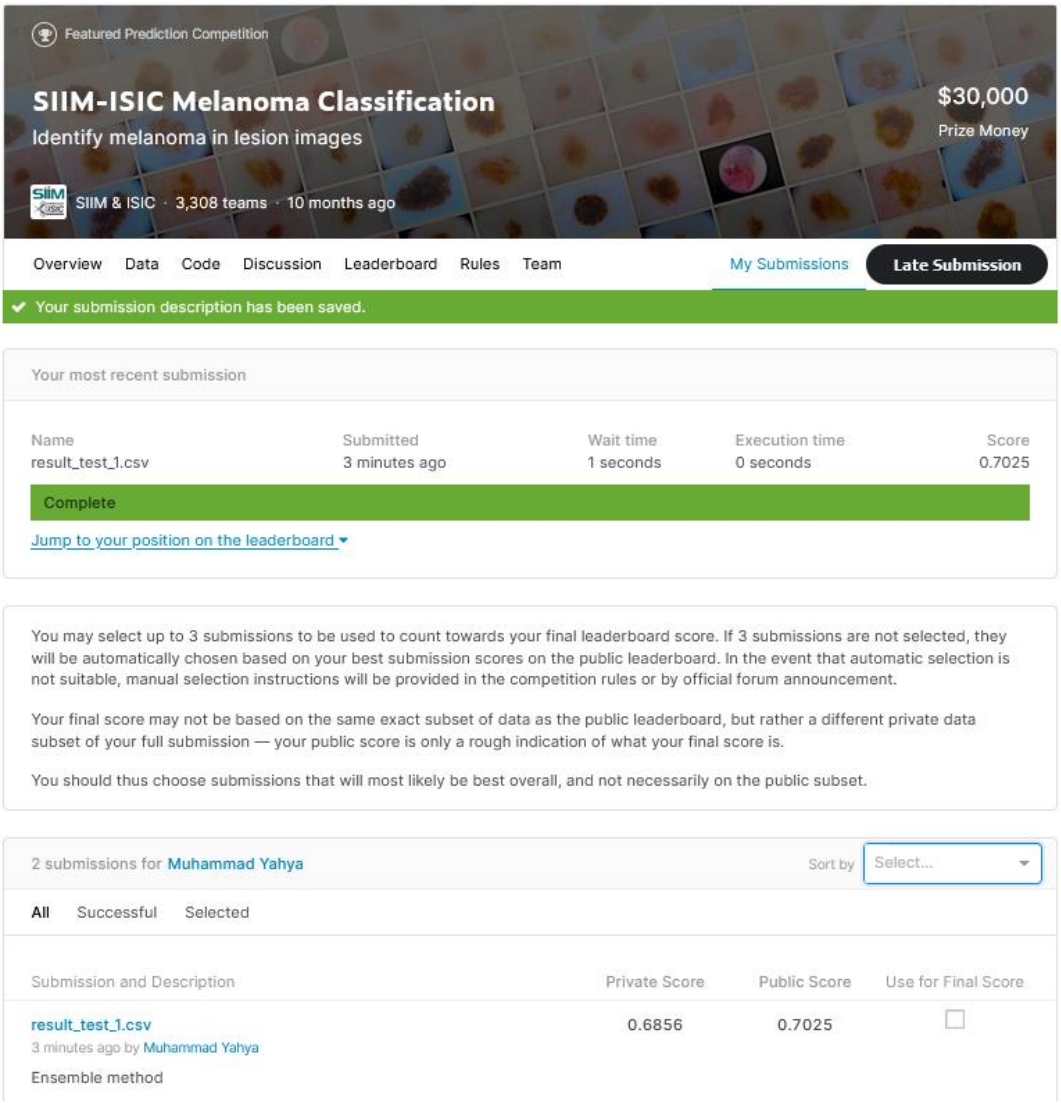


Figure 3.31: Kaggle Submission

Chapter 4: Conclusion

We have learned a lot throughout the course of this project, ML is a field with huge potential and vast number of applications. Transfer learning is a very important concept that makes ML applications a lot easier for us. We have used multiple CNN architectures to classify images and then used an Ensemble network to classify these images. The best accuracy we could get out of all CNN architectures was around 80% but for the Ensemble network, we got an accuracy of around 95%. To conclude, we found out that using Ensemble learning technique we can get better results as compared to using just a single model.

Bibliography

[1] Dirk Schadendorf, Alexander CJ van Akkooi, Carola Berking, Klaus G Griewank, Ralf Gutzmer, Axel Hauschild, Andreas Stang, Alexander Roesch, and Selma Ugurel. Melanoma. The Lancet, 392(10151):971–984, 2018.

[2] Siddharth Verma, Rashmi Agrawal, 10 - Deep neural network in medical image processing, Editor(s): Valentina Emilia Balas, Brojo Kishore Mishra, Raghvendra Kumar, Handbook of Deep Learning in Biomedical Engineering, Academic Press, 2021, Pages 271-292, ISBN 9780128230145, https://doi.org/10.1016/B978-0-12-823014-5.00002-8.

[3] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International Conference on Machine Learning. PMLR, 2019.

[4] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[5] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[6] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[8] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[9] Rotemberg, V., Kurtansky, N., Betz-Stablein, B., Caffery, L., Chousakos, E., Codella, N., Combalia, M., Dusza, S., Guitera, P., Gutman, D., Halpern, A., Helba, B., Kittler, H., Kose, K., Langer, S., Lioprys, K., Malvehy, J., Musthaq, S., Nanda, J., Reiter, O., Shih, G., Stratigos, A., Tschandl, P., Weber, J. & Soyer, P. A patient-centric dataset of images and metadata for identifying melanomas using clinical context. Sci Data 8, 34 (2021). https://doi.org/10.1038/s41597-021-00815-z

[10] Alexander Jung, imgaug, (2020), GitHub Repository, https://github.com/aleju/imgaug

[11] Shivangi Jain, Vandana jagtap, Nitin Pise, Computer Aided Melanoma Skin Cancer Detection Using Image Processing, Procedia Computer Science, Volume 48, Pages 735-740, ISSN 1877-0509,2015.

[12] Nayana Banjan, Prajkta Dalvi, Neha Prakash Athavale, Melanoma Skin Cancer Detection by Segmentation and Feature Extraction using combination of OTSU and STOLZ Algorithm Technique,2017.

[13] Joanna Jaworek-Korjakowska, Paweł Kłeczek, "Automatic Classification of Specific Melanocytic Lesions Using Artificial Intelligence", BioMed Research International, vol. 2016, Article ID 8934242, 17 pages, 2016. https://doi.org/10.1155/2016/8934242

[14] Goyal, M., Knackstedt, T., Yan, S., & Hassanpour, S. (2020). Artificial intelligence-based image classification for diagnosis of skin cancer: Challenges and opportunities. Computers in Biology and Medicine, 104065

[15] Nurshazlyn Mohd Aszemi and P.D.D Dominic, "Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms" International Journal of Advanced Computer Science and Applications (IJACSA), 10(6), 2019. http://dx.doi.org/10.14569/IJACSA.2019.0100638

[16] L. Bi, J. Kim, E. Ahn, A. Kumar, M. Fulham and D. Feng, "Dermoscopic Image Segmentation via Multistage Fully Convolutional Networks," in IEEE Transactions on Biomedical Engineering, vol. 64, no. 9, pp. 2065-2074, Sept. 2017, doi: 10.1109/TBME.2017.2712771.

[17] Gessert, Nils, et al. "Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data." MethodsX 7 (2020): 100864.

[18] A. H. Shahin, A. Kamal and M. A. Elattar, "Deep Ensemble Learning for Skin Lesion Classification from Dermoscopic Images," 2018 9th Cairo International Biomedical Engineering Conference (CIBEC), Cairo, Egypt, 2018, pp. 150-153, doi: 10.1109/CIBEC.2018.8641815