AI Course

Team Project Final Report

For students (instructor review required)

ⓒ2019 SAMSUNG. All rights reserved.

Samsung Electronics Corporate Citizenship Office holds the copyright of this document.

This document is a literary property protected by copyright law so reprint and reproduction without permission are prohibited.

To use this document other than the curriculum of Samsung innovation Campus, you must receive written consent from copyright holder.

|  |
| --- |
| Deep Fake Image Detection |

26/03/2023

Group 2: AI Space

**Nadeen Tarek**

**Rodha Al Marzooqi**

**Rukhiya Nafeesa Thayyil Kedaran**

**Aysha Asif**

**Pooja Meledath**

**Ahmad Saad**

Content

1. Introduction

1.1. Background Information

1.2. Motivation and Objective

1.3. Members and Role Assignments

1.4. Schedule and Milestones

2. Project Execution

2.1. Data Acquisition

2.2. Training Methodology

2.3. Workflow

2.4. System Diagram

3. Results

3.1. Data Preprocessing

3.2. Exploratory Data Analysis (EDA)

3.3. Modeling

3.4. User Interface (Interface)

3.5. Testing and Improvements

4. Projected Impact

4.1. Accomplishments and Benefits

4.2. Future Improvements

5. Team Member Review and Comment

6. Instructor Review and Comment

# 1. Introduction

## 1.1. Background Information

Deepfake technology is a form of artificial intelligence (AI) that has gained notoriety in recent years for its ability to create compelling videos and images often used to spread false information, damage reputations, or defraud people. The technology uses deep learning algorithms to analyze and manipulate visual data such as images and videos. Deepfake technology uses algorithms to analyze and manipulate existing images or videos to create new, fake content. For example, an algorithm may be used to swap the face of one person in a video with the face of another person, or to manipulate the speech of a person in a video to make them say something they did not say.The development of deepfake technology has been driven by advances in machine learning and computer vision, which have made it possible to create more sophisticated algorithms for analyzing and manipulating visual data. Despite its potential benefits, deepfake technology has raised concerns about its potential misuse. For example, deepfake technology could be used to create fake videos or images of political leaders or celebrities, which could be used to spread false information or to damage reputations. Deepfake technology could also be used to create fake evidence in criminal cases, or to violate the privacy of individuals by creating fake images or videos of them without their consent. As a result of these concerns, there has been growing interest in developing tools and technologies for detecting and mitigating the risks of deepfake technology. These include digital watermarking, which involves embedding invisible markers in images or videos to help identify their original source and ensure their authenticity, as well as machine learning algorithms that can be used to detect and identify deepfakes based on subtle visual cues.

## 1.2 Motivation and Objective

The rise of deep fake technology has raised concerns about the potential misuse of manipulated images and videos for various malicious purposes such as spreading fake news, propaganda, disinformation, or cybercrime. Deepfake images are computer-generated images that can be used to create fake identities, forge documents, or even blackmail individuals. Therefore, the motivation behind the detection of deep fake images is to preserve the authenticity and credibility of visual media and protect individuals' privacy and security.

The primary objective of an AI deep fake image detection project is to develop an accurate and reliable system that can automatically detect deep fake images with high precision and recall. This system should be able to distinguish between real and fake images by analyzing various features such as facial expressions, lighting, shadows, and pixel-level artifacts. The following are some specific objectives of an AI deep fake image detection project:

1. Developing deep learning models: Deep learning models, such as Convolutional Neural Networks (CNNs), can learn from large amounts of data and identify patterns and features that distinguish between real and fake images. Developing and fine-tuning these models for deep fake detection can significantly improve the accuracy of the system.
2. Collecting diverse and representative datasets: A deep fake image detection system should be trained on a diverse and representative dataset that includes different types of deep fakes, lighting conditions, and camera angles. This will ensure that the system can generalize well and detect deep fakes in real-world scenarios.
3. Evaluating and benchmarking the system: An AI deep fake image detection system should be evaluated and benchmarked against other state-of-the-art systems to assess its performance and identify areas for improvement. This can be achieved by using standardized metrics such as precision, recall, and F1 score.
4. Integrating the system into existing platforms: An AI deep fake image detection system can be integrated into various platforms such as social media, messaging apps, or online marketplaces to automatically detect and flag deep fake images. This can help prevent the spread of fake news, protect individuals' privacy, and enhance the overall trustworthiness of online media.

In conclusion, an AI deep fake image detection project can help address the growing concerns about the misuse of manipulated images and videos. By developing an accurate and reliable system that can detect deep fake images, we can safeguard the authenticity and credibility of visual media and promote a safer and more secure online environment.

## 1.3 Members and Role Assignments

**Action Plan**

Each member worked on researching and writing down one part of the report

* Goal - Rodha Al Marzooqi
* Abstract - Rukhiya Nafeesa Thayyil Kedaran
* Training method - Aysha Asif
* Data preprocessing - Nadeen Tarek
* Expected outcome - Pooja Meledath
* Role by member & Schedule summary- Ahmad Saad

**Coding**

Each member will work, train, test, and tune a model with the same dataset purpose:

* DensNet Architecture - Rukhiya Nafeesa Thayyil Kedaran
* VGGNet Architecture - Aysha Asif
* ResNet50 Architecture & Demo of the model - Nadeen Tarek
* Inception - Pooja Meledath & Ahmad Saad
* Customized CNN Architecture - Rodha Al Marzooqi

**Presentation**

Each of the team members added the results of their models in the slides and worked together on doing the rest of the presentation

* Explain the problem - Aysha Asif
* Solution Description - Rodha Al Marzooqi
* About the Dataset - Rukhiya Nafeesa Thayyil Kedaran
* Data preprocessing - Nadeen Tarek
* Each model used and results - Each will do a slide of their model
* Conclusion and future work - [Pooja Meledath](https://sicaiuae2023.slack.com/team/U04JLS1E724)

**Final Report**

Each of the team members added the details other models and the results obtained and the rest of the final report was taken from the action plan and the remaining part was divided among the team as follows:

* Background Information - [Pooja Meledath](https://sicaiuae2023.slack.com/team/U04JLS1E724)
* Motivation and objectives - Rodha Al Marzooqi
* Workflow & User Interface (Interface) - Nadeen Tarek
* System Diagram - Ahmad Saad
* Exploratory Data Analysis (EDA) - Aysha Asif
* Accomplishments and Benefit & Future Improvements - Rukhiya Nafeesa Thayyil Kedaran
* AI Project Work Breakdown Structure (WBS) - Ahmad Saad

## 1.4 Schedule and Milestones

|  |  |
| --- | --- |
| **Timeline** | **Tasks/ Milestones** |
| Day 1-2 | 1. Work on project goals and objectives 2. Choose a suitable dataset for project 3. Decide on the evaluation metrics for the models 4. Research and select appropriate training methods for the selected models. 5. Begin writing the abstract for the project report |
| Day 3-4 | 1. Finalize the abstract 2. Begin data preprocessing for the selected dataset 3. Perform data augmentation on dataset |
| Day 5-7 | 1. Implement the VGG model and perform hyperparameter tuning 2. Implement the VGGFace model and perform hyperparameter tuning. 3. Evaluate the VGG and VGGFace models with train and validation accuracy/loss plots |
| Day 8-11 | 1. Implement the ResNet50 model and perform hyperparameter tuning 2. Begin implementation of the DenseNet model |
| Day 12-15 | 1. Evaluate the ResNet50 and DenseNet models with train and validation accuracy/loss plots 2. Choose the best-performing model for the final report 3. Finalize the model selection, and evaluate the final model using confusion metrics and ROC curves 4. Prepare the final presentation and report 5. Deliver the final presentation 6. Submit the final report and code |

# 2. Project Execution

## 2.1 Data Acquisition

This dataset consists of all 70k REAL faces from the Flickr dataset collected by Nvidia, as well as 70k fake faces sampled from the 1 Million FAKE faces (generated by StyleGAN).

The dataset was obtained from Kaggle. The data was not available in its desired tabular format but rather in the form of a directory structure. It contained a train, test, and valid directories, each of them having a fake and real subdirectory with a balanced sample of real and fake images. The train, valid and test sets consist of 100k,20k, and 20k images respectively, each of the images having a shape of (256,256,3).

## 2.2 Training Methodology

The traditional machine learning classifiers (Support Vector Machine Algorithm or naive algorithms), deep neural networks, convolutional neural networks (CNN), recurrent neural networks (RNN), long short-term memory (LSTM), and many other techniques can all be used to identify GAN-generated deep fake images.

The main objective is to distinguish the deep fake images and authentic images. We will test out various State-of-art models using transfer learning with required modifications and try a customized CNN model and choose the model that gives the most optimal results. The Convolutional Neural Network (CNN) is a highly potent tool for classifying and recognizing images. Comparatively, CNN needs less preprocessing than other classification techniques.

Some CNN architecture that we intend to employ is as follows:-

1. VGG

3. ResNet50

4. DenseNet121

5. Inception

6. Customized CNN (MesoNet)

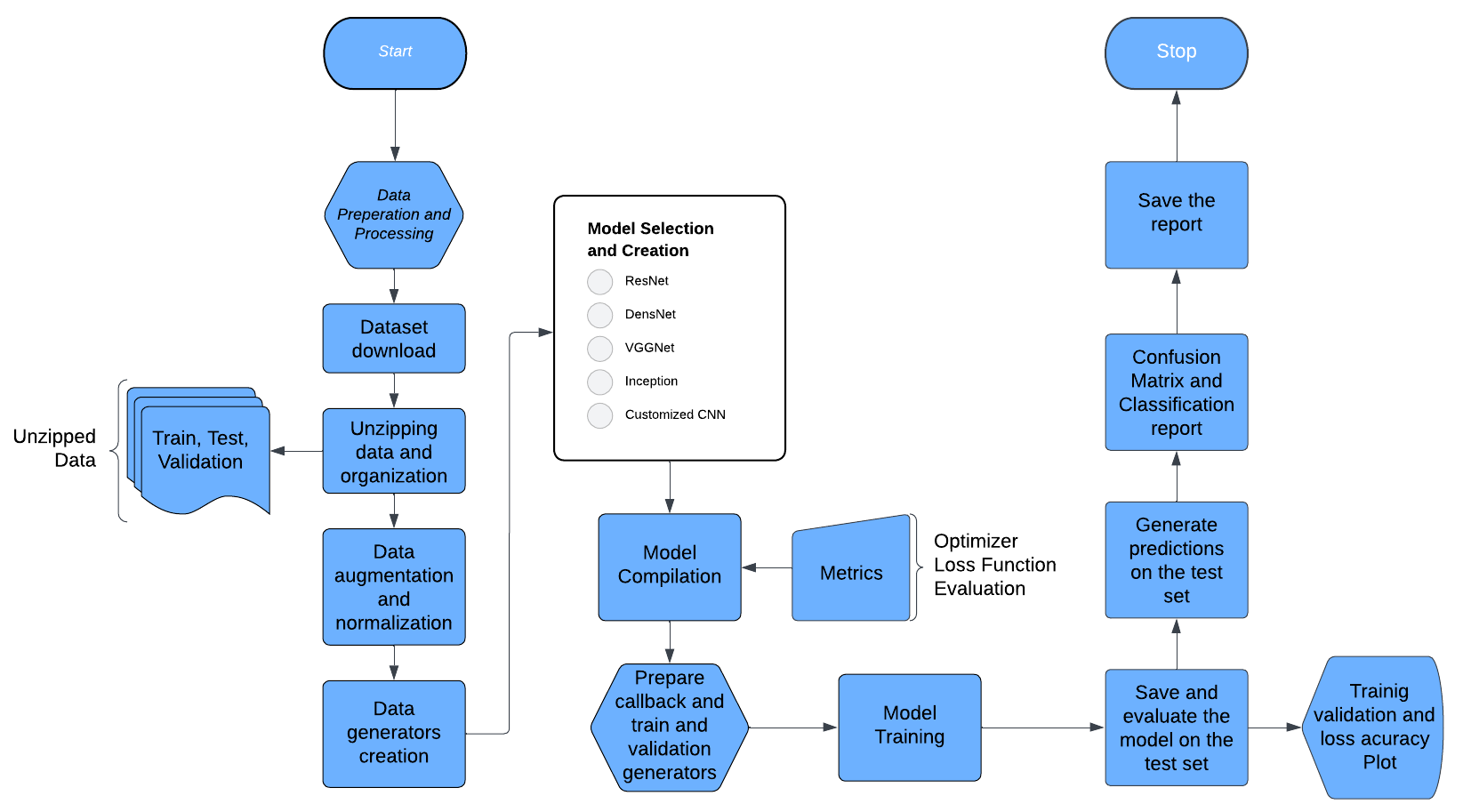
## 2.3 Workflow

1. Data Acquisition
   * Obtain the deep fake image dataset from Kaggle or other sources
   * Split the dataset into the train, validation, and test sets with a balanced sample of real and fake images
2. Data Preprocessing
   * Resize images to the required input size for each model (64\*64)
   * Normalize pixel values and perform data augmentation if necessary
3. Model Selection and Implementation
   * Choose and implement various state-of-the-art models for deep fake detection, such as VGG, ResNet50, DenseNet121, Inception, and a customized CNN
   * Choose the most suitable optimizer and apply binary\_crossentropy loss during the model training
   * Apply transfer learning and fine-tune the models with the deep fake dataset
4. Model Training
   * Train each model using the train set. The following are applied during training:
     1. Create model Checkpoints to save the best models’ weights in a checkpoint file when maximum accuracy is achieved, so the model can be loaded later to continue the training from the state saved.
     2. Reduce the learning rate when the model accuracy stops improving.
     3. Stop training the model when the validation set accuracy has stopped improving.
   * Validate the models using the validation set to confirm that it's not overfitting
   * Perform hyperparameter tuning to improve the performance of each model
5. Model Evaluation
   * Evaluate each model using the test set and compute performance metrics such as precision, recall, F1 score, and ROC curves
   * Compare the performance of each model and choose the best-performing model for deep fake image detection
6. Integration and Deployment
   * Integrate the selected model into existing platforms or develop a standalone application for deep fake image detection
   * Deploy the system to production for real-world use
7. Documentation and Reporting
   * Document the entire process, including data acquisition, preprocessing, model selection, training, evaluation, and deployment
   * Prepare a final report and presentation, highlighting the project's motivation, objectives, methodology, results, and conclusions
8. Future Work and Improvements
   * Analyze the results and identify areas for improvement in the deep fake detection system
   * Propose and implement new techniques or models to enhance the performance of the system
   * Continuously update the system with new datasets and emerging deep fake techniques to maintain its effectiveness in real-world scenarios

## 

## 2.4 System Diagram

The diagram for the entire project’s workflow for building models is shown below. For a detailed diagram about each of the models used, you can refer to section 3.3 which has each model’s architecture in details.



3. Results

## 3.1. Data Preprocessing

Some data preprocessing is required to use this dataset without overfitting the model. First, we Imported the dataset from Kaggle to the coding environment using Kaggle API. Then we normalized all the train, test, and validation images and resized them to a smaller size (64\*64) to ensure faster training. Then, we performed some data augmentation on the training set to generate modified copies of the dataset by rotating, zooming, and flipping the existing images.

## 3.2 Exploratory Data Analysis (EDA)

### Visualizing the data

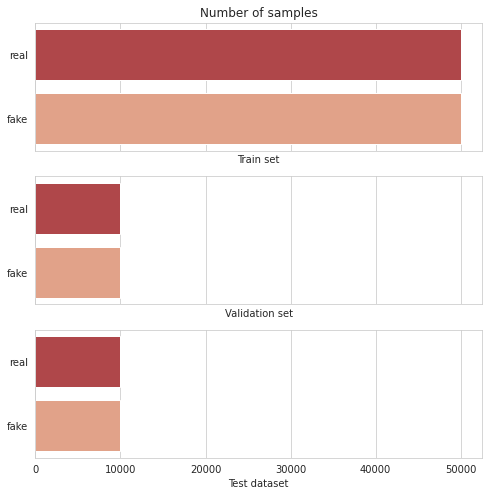
|  |  |
| --- | --- |
|  |  |

### The shape of the Images

****

The images have the shape : (256,256,3) 256px in width and height, 3 corresponds to the number of channels. Since the images are colored, these images are RGB images indicating 3 channels: Red, Green, and Blue.

### The number of samples per class

****

The dataset consists of an equal number of images representing each class. Hence, the dataset is balanced. This prevents the model from being biased towards one class.

## 3.3 Modeling

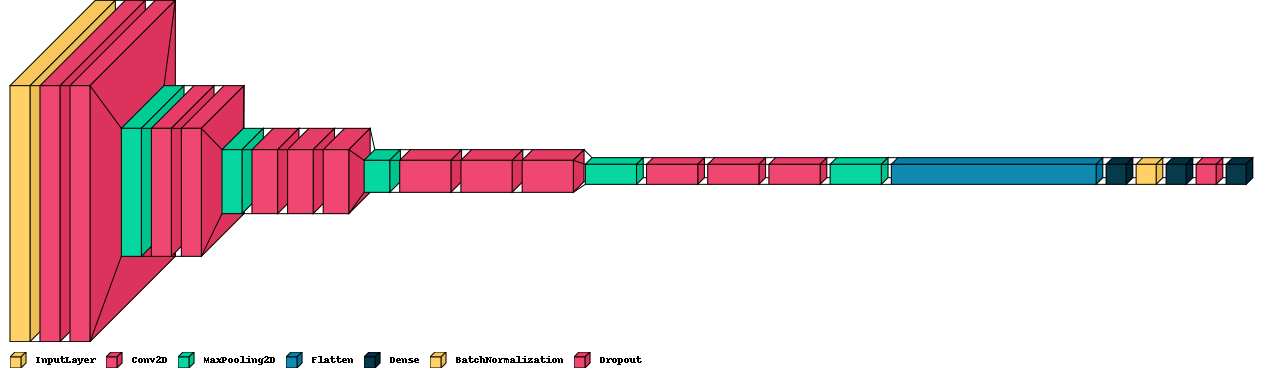
To be able to find the best model for detecting deep fake images, we tried 5 different models. Some of them are pre-trained state-of-the-art models and some models were customized CNN models that were trained from scratch.

### VGGNet

VGG, short for **Visual Geometry Group**, is a standard Image classification pre-trained on a subset of the Imagenet dataset classifying 1000 different classes. VGGNet16 and VGGNet19 were used for DeepFake detection.VGG16 and VGG19 differ in their architecture on the basis of the number of Convolutional Networks. The top layer or fully connected layer of the VGGNet was removed to facilitate the Binary classification required for the required use case: Deep fake or real images.

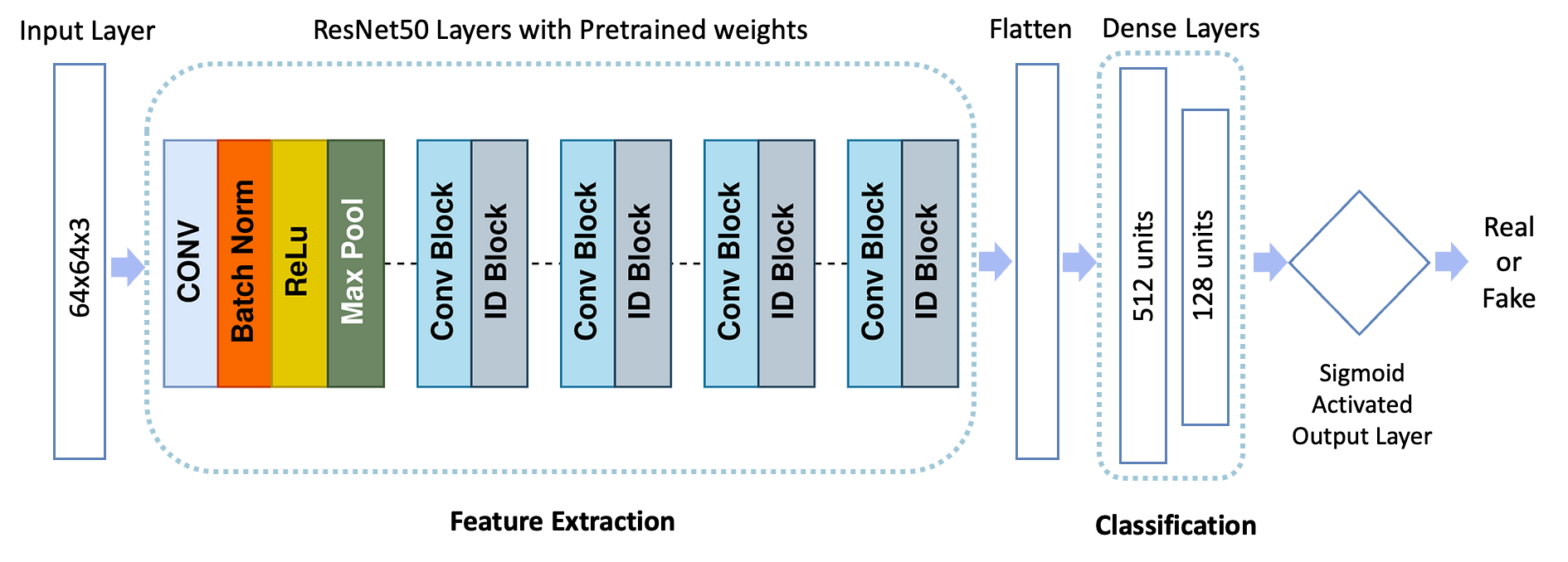
The Image dataset was preprocessed and augmentation was applied to the training set to enrich the dataset. For the classification, a fully connected layer consisting of 2 Dense layers with Batch Normalization between them, followed by a Dropout layer. Lastly,a Dense layer with a Sigmoid Activation function, as the output layer was used to classify real and fake images.

The model was trained using Adam Optimizer with a learning rate or step size of 0.001 with Binary cross entropy as the loss function to be optimized.



### ResNet50

Pre-trained ResNet50 v1 model was fine-tuned on the deep fake dataset to classify between fake and real images. The ResNet model stands for Residual Network which is considered a type of artificial neural network (ANN) that forms networks by stacking residual blocks. ResNet50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). Multiple variations of the ResNet hyper-parameters and optimizers were tried, and the architecture of the model with the best performance is shown in the diagram below. First, the model had an input layer of size 64x64x3 which represents the images' width, height, and color channel (RGB). For the feature extraction, the ResNet50 convolutional layers with the pre-trained weights of ImageNet were used. For the classification, a fully connected layer that consists of 2 Dense layers of 512,128 units respectively was used. Finally, a sigmoid-activated output layer was used to classify real and fake images. The model was trained using an Adam optimizer with a learning rate of 0.01 that optimizes for binary cross-entropy loss.

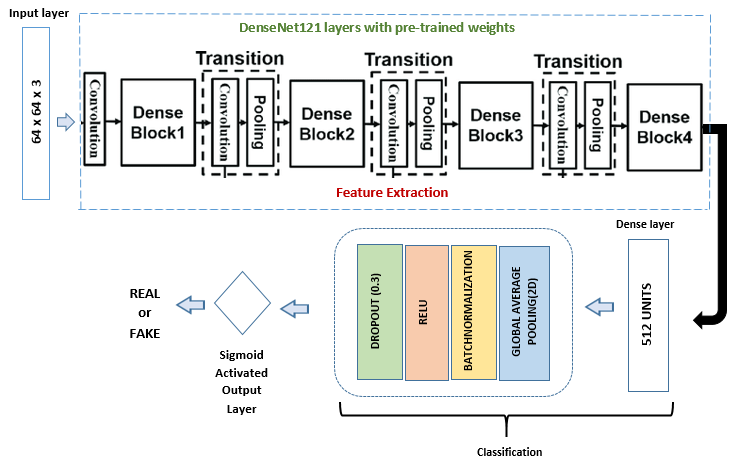


### DenseNet121

To classify the deepfake and real images DenseNet121 was fine-tuned on the given data set. The DenseNet121 is a pre-trained neural network technique primarily used for image recognition tasks. In a DenseNet architecture, each layer is connected directly with every other layer, hence the name Densely Connected Convolutional Network. The DenseNet is based on the convolution neural net (CNN) architecture.

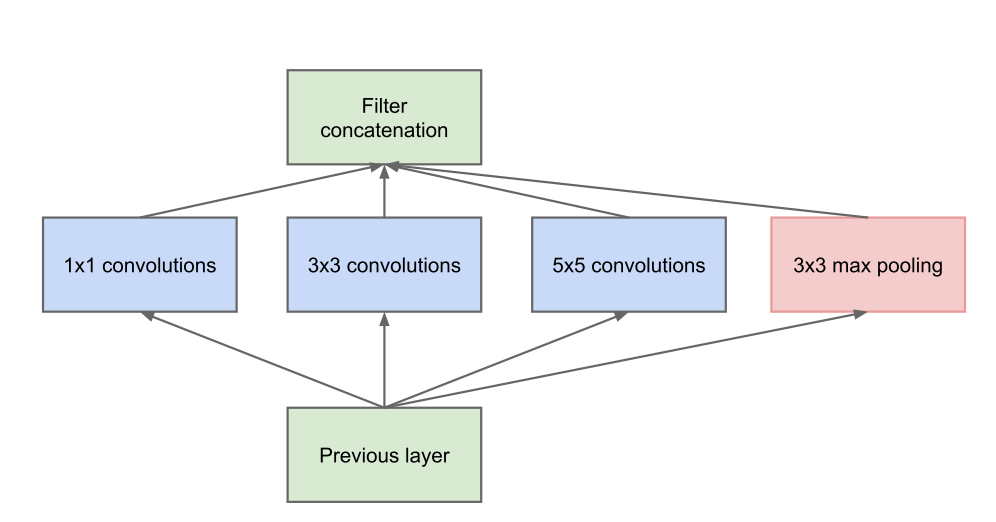
We used the weights of pre-trained DenseNet, but we redefined the dimension of the last fully connected layer of DenseNet121 which consists of a dense layer of

512 units to fit the classification. The initial learning rate for the optimizer choices is set at 0.001. The ﬁnal output layer with sigmoid activation is used for deepfake classiﬁcation. Multiple parameters were deployed and the best-performing model architecture given below.



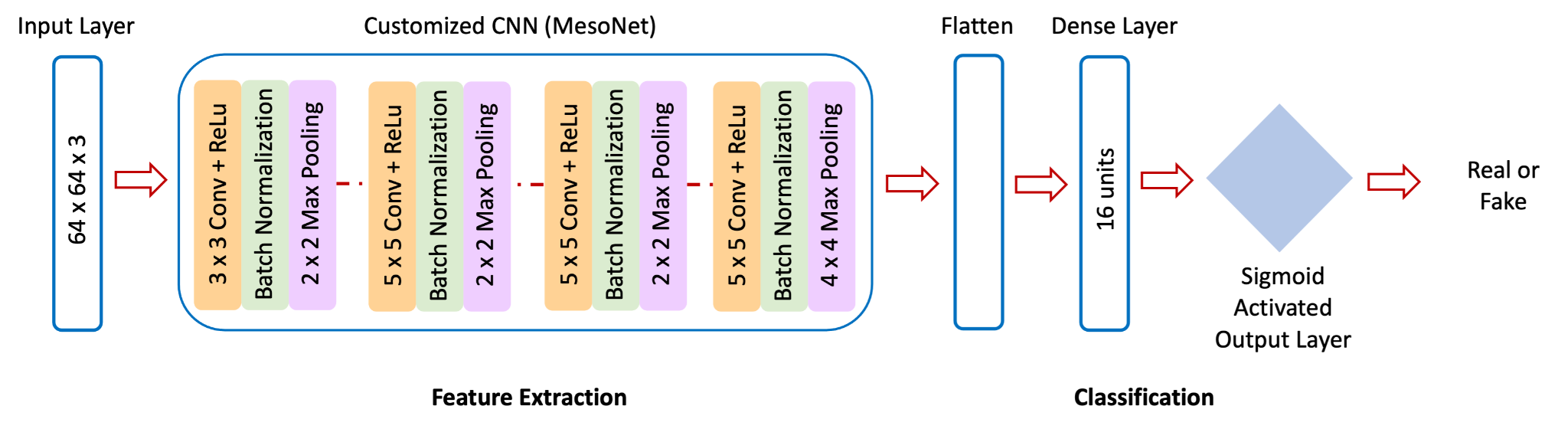
### Inception

The InceptionV3 model is a deep CNN architecture designed for image classification tasks. It is based on the “inception modules” concept which are parallel layers with different kernel sizes concatenated together to allow the model to learn different spatial patterns in input images. When used in the code, it is imported without the top layers and the Global Average Pooling Layer is included. The base model layers are frozen to preserve their weights. The output is flattened and connected to two dense layers (512 and 128) with ReLU activation and Batch Normalization. The final output layer employs a sigmoid activation for binary classification.

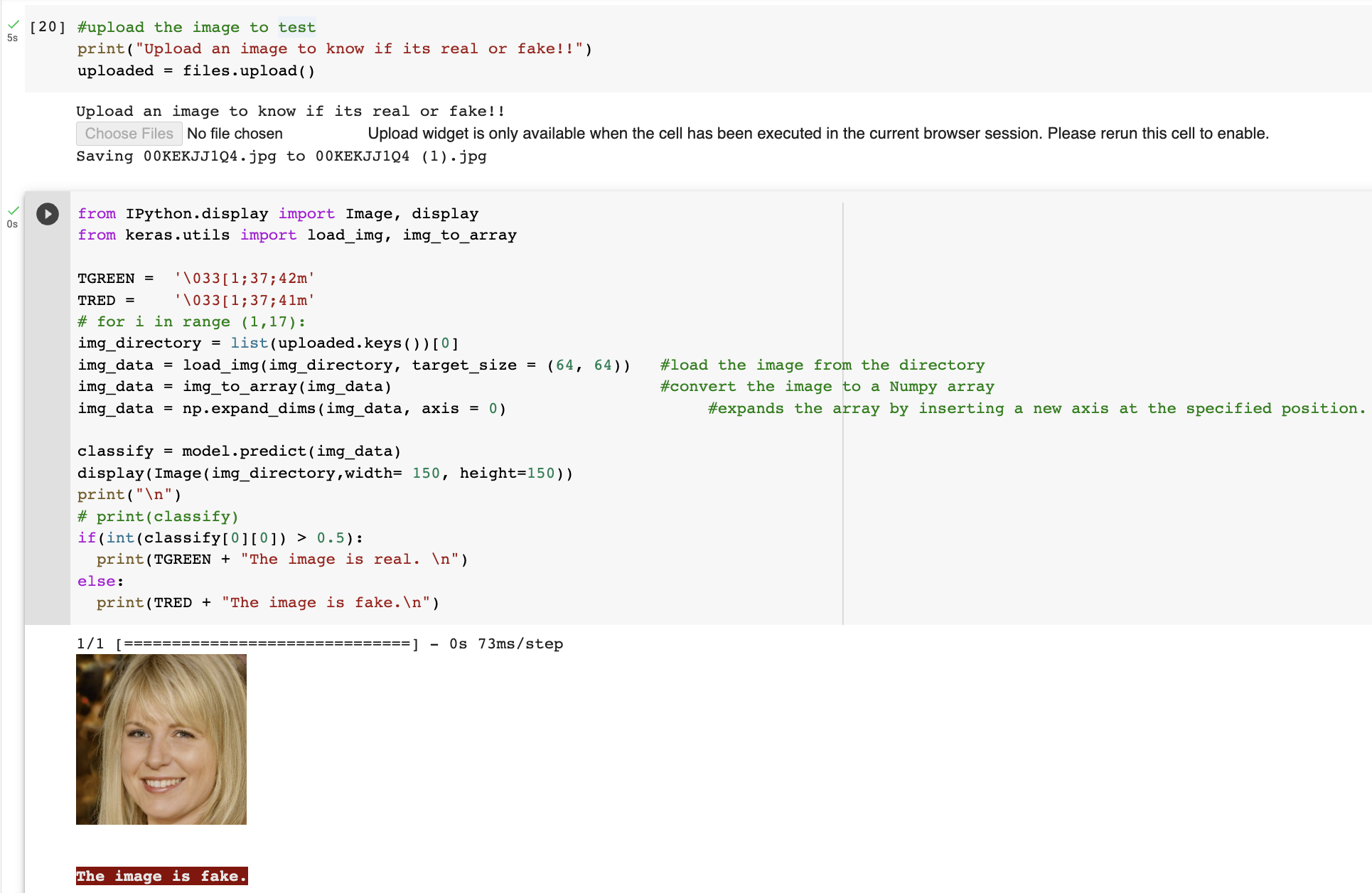


### Customized CNN (MesoNet)

The MesoNet adopts a mesoscopic method for detecting image manipulation. The model consists of four convolutional layers, with the initial layer utilizing 3x3 kernels and the following layers using 5x5 kernels. Each convolutional layer is succeeded by a batch normalization layer and a pooling layer. The final convolutional layer connects to a fully-connected layer that features 16 neurons, followed by dropout and the output layer. The MesoNet follows a traditional CNN architecture, with a final sigmoid classification layer. The figure below shows the model implemented in detail.



## 3.4 User Interface (Interface)

We mainly worked on creating a demo for the 5 models, where you can upload any image for them through the .ipynb file and the model can predict if its fake or real as shown in the diagram below. However, as part of future improvements for this project, a website or a mobile app can be developed to allow users to upload images in an easier manner to verify if this image is real or fake. 

## 3.5. Testing and Improvements.

### VGGNet

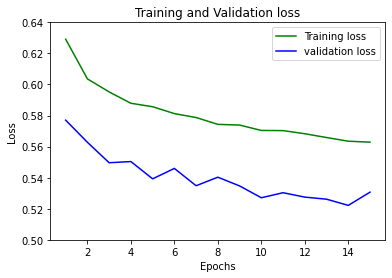
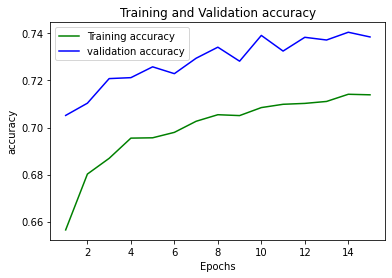
Many different variations of the VGGNet were implemented.

The variations tried include:

* adding more dense layers
* dense layers with many units
* adding and removing Batch Normalization layer
* adding and removing dropout layer
* changing the rate of the dropout layer
* using different optimizers

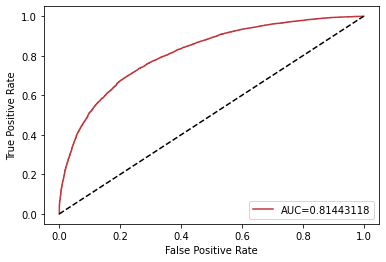
However, the best variation of the VGGNet gives an accuracy of 76%.

Following is the validation and training accuracy and loss curves are presented below.

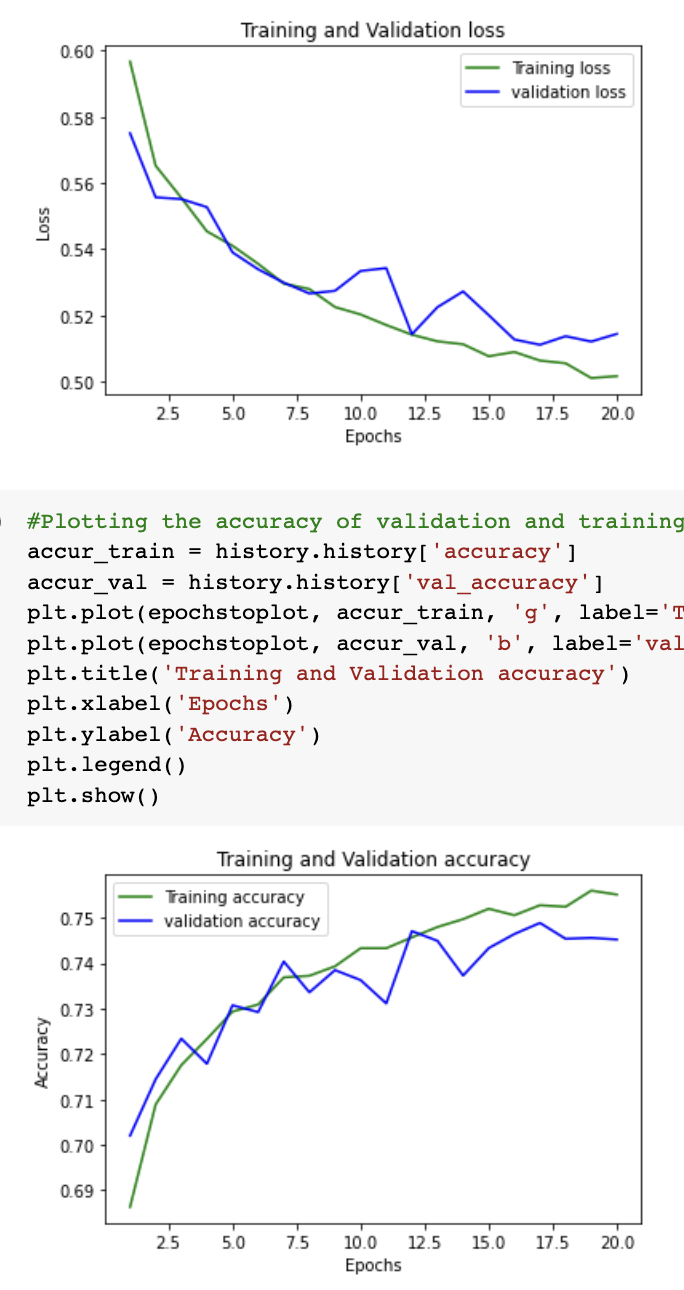
Following is the classification metrics:

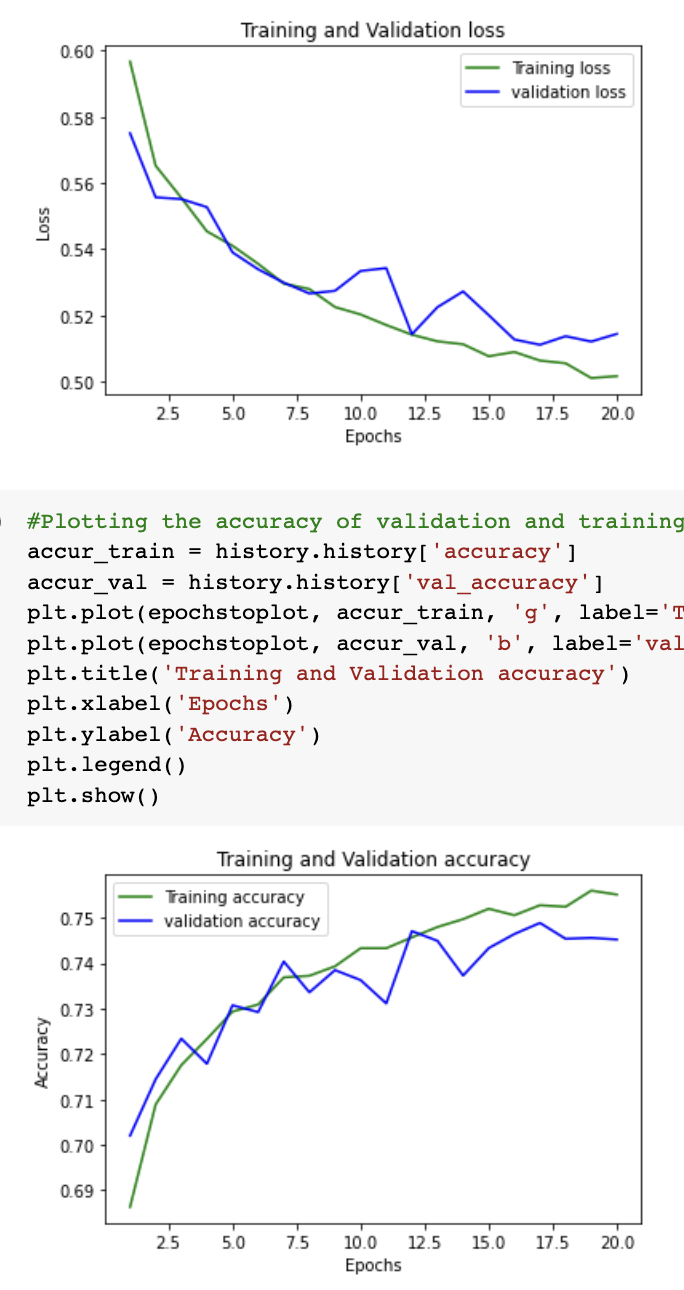
| **Precision** | | **Recall** | | **F1-score** | | **Accuracy** |
| --- | --- | --- | --- | --- | --- | --- |
| **Real** | **Fake** | **Real** | **Fake** | **Real** | **Fake** |
| 0.7389 | 0.734 | 0.7319 | 0.7417 | 0.7354 | 0.7378 | 0.74 |



### ResNet50

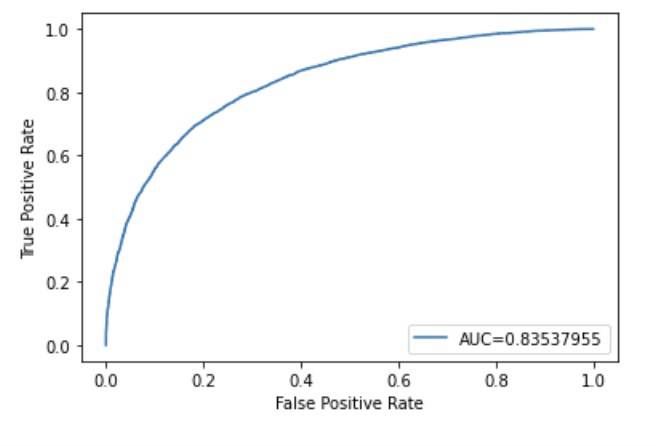
7 different variations of the ResNet50 model were tried, these variations include increasing and decreasing the number of Dense layers, adding and removing the dropout layer, and changing the optimizer type and the pooling type. However, the best-performing model was the ResNet50 model architecture presented in section 3.3.2. The models’ validation and training accuracy and loss curves are presented below.





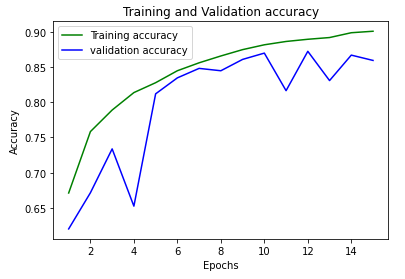
The model was then tested using the 20k images in the test set and the classification report for it is presented in the table below. The ROC curve as well for the same model is shown below and the AUC was calculated to be 0.8353.

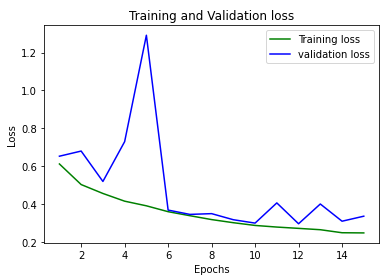
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **f1-Score** | **Support** |
| **Fake** | 0.7315 | 0.8065 | 0.7672 | 10000 |
| **Real** | 0.7844 | 0.704 | 0.7420 | 10000 |
| **Accuracy** |  |  | 0.75525 |  |
| **Macro avg** | 0.7580 | 0.7553 | 0.7546 | 20000 |
| **Weighted avg** | 0.7580 | 0.7553 | 0.7546 | 20000 |



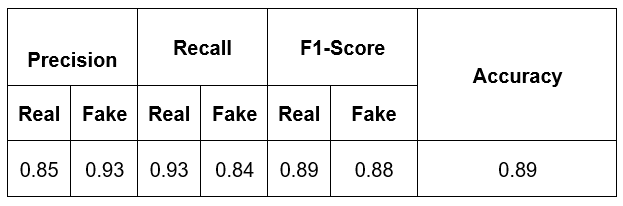
### DenseNet121

5 different variations of the DenseNet121 model were tried, these variations include adding and removing the dropout layer, changing the optimizer type and changing/reducing the learning rate and increasing the number of epochs. However, the best-performing model was the DenseNet121 model architecture presented in section 3.3.3. It gives an accuracy of 89%. Figure below shows the models’ validation and training accuracy and validation and training loss curves.



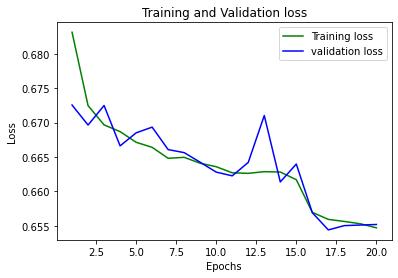


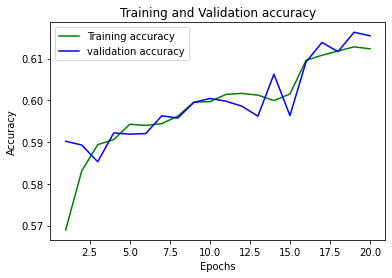
Further classification report is shown in Table below.

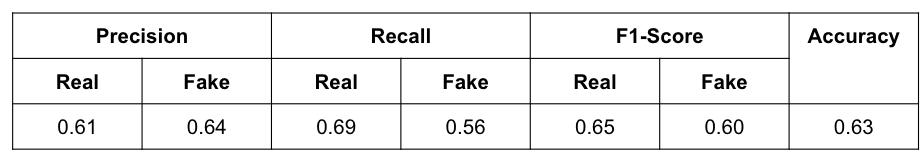


### Inception

Different variations of the Inception model were tested with adding and removing dense layers, adding and removing dropout layers and using different optimizers. The best inception model gives an accuracy of 63%.

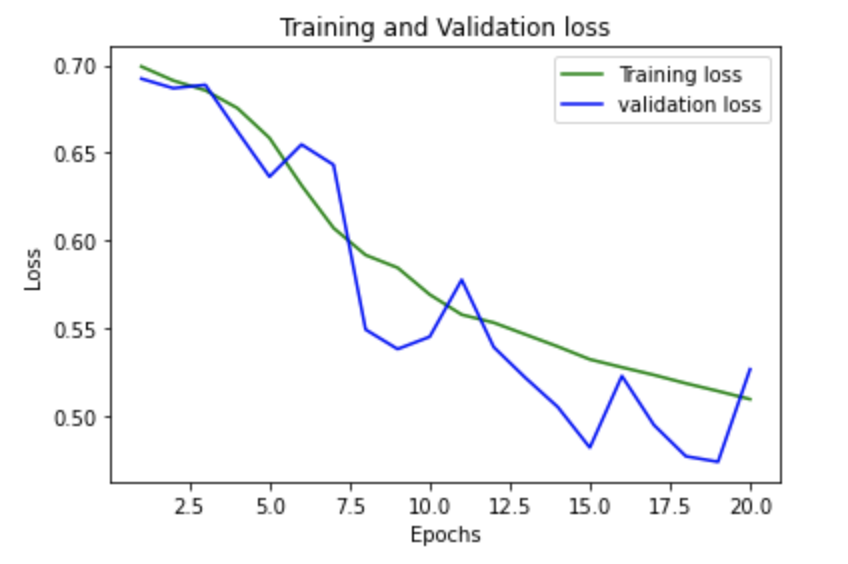


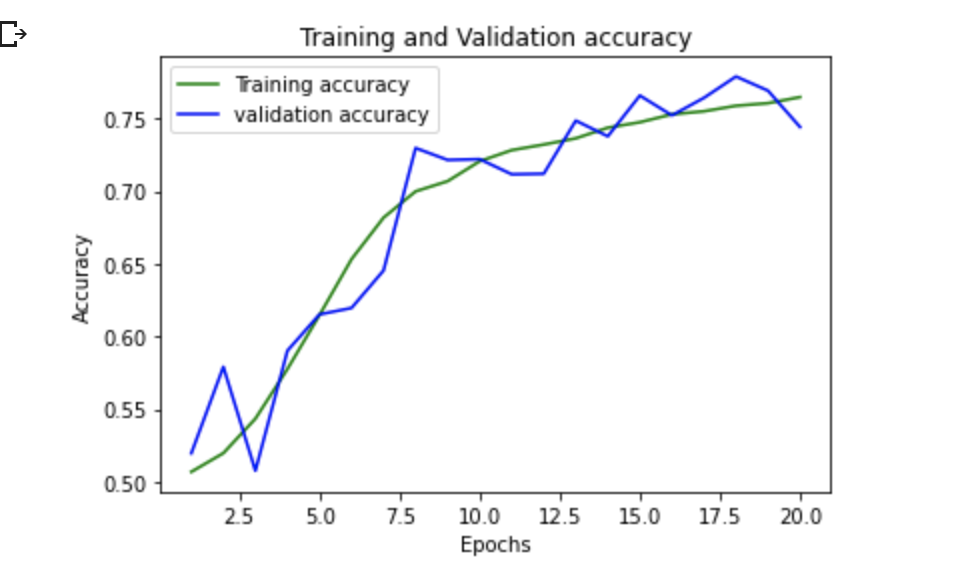


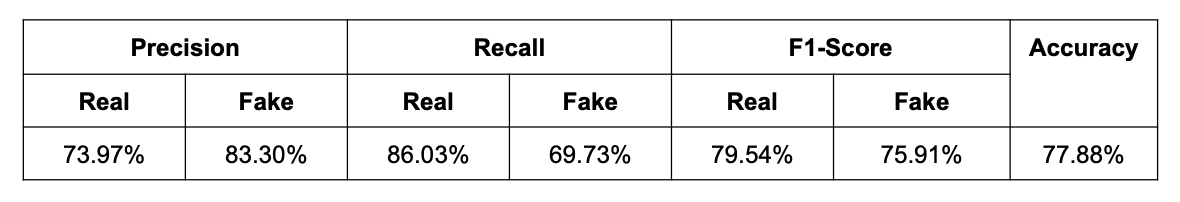


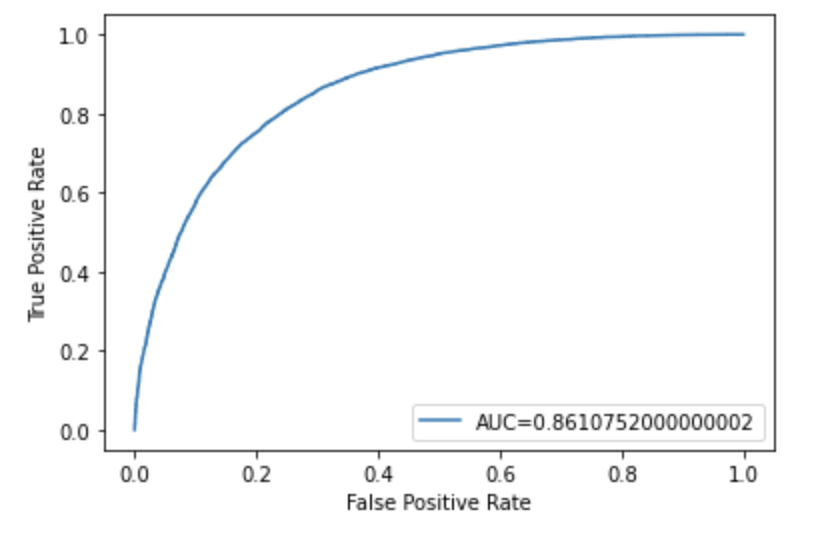
### Customized CNN (MesoNet)

Testing out the Customized CNN (MesoNet) model according to the diagram above gave the following results with almost 78% accuracy as shown in the figures and table below.



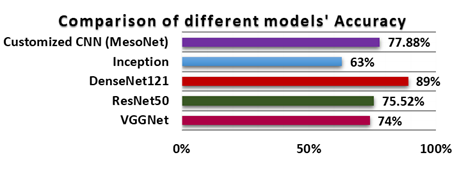






### Comparison of different models’ Accuracy

In summary, we tried 5 models some were pre-trained models and some were trained from scratch. Among the 5 different models we trained, DenseNet121 had the highest accuracy of 89% and an F1-Score for 0.7546 detecting real and fake images. The DenseNet validation and training accuracy reflect that the model is not overfitting.

****

# 4. Project Impact

## 4.1. Accomplishments and Benefits

This research paper has proposed various transfer learning techniques for feature extraction from the input deepfake/real image. Initially, the input image was preprocessed by rescaling, normalizing and performing various data augmentation methods on the training dataset to ensure faster training. Further, the proposed transfer learning methods have obtained accuracy in detecting the deepfake image with 89%, 74%, 77.88%, 75.46 and 63% using the models such as DenseNet121, VGGNet, Customized CNN, ResNet50, and Inception, respectively. Hence, compared to the other models and previous related research works , the proposed pretrained model approach called DenseNet121 has the highest accuracy in detecting the deep fake images and the real images. However, we came to the conclusion that our approach towards the detection of Real and DeepFake images has shown promising results with higher accuracy rates. However, we aim to make the model more efficient and better in the future.

Our research work will lend a hand to cybersecurity specialists to overcome deep fake-based cybercrimes by accurately exposing deepfake images.

## 4.2 Future Improvements

Future research could focus on enhancing our model's accuracy so that it can also detect Deepfake images and videos as well. Based on the model, a web application may be developed to make it easier for cybersecurity professionals to detect DeepFake image/video fraud and prevent deep fake-based cybercrimes.

# 5. Team Member Review and Comment

|  |
| --- |
|  |

|  |  |
| --- | --- |
| NAME | REVIEW and COMMENT |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 6. Instructor Review and Comment

|  |  |  |
| --- | --- | --- |
| CATEGORY | SCORE | REVIEW and COMMENT |
| IDEA | \_\_/10 |  |
| APPLICATION | \_\_/30 |  |
| RESULT | \_\_/30 |  |
| PROJECT MANAGEMENT | \_\_/10 |  |
| PRESENTATION & REPORT | \_\_/20 |  |
| TOTAL | \_\_/100 |  |