

**Department of Electrical and Computer Engineering North South  
University**

---



**Senior Design Project (499A and 499B)**

**On**

**DeepFake Video Detection**

**Course Instructor's name and initial:**

**Mohammad Ashrafuzzaman Khan (Azk)**

**Group Members:**

**Shayekh Mohiuddin Ahmed Navid-153124702**

**Shamima Haque Priya-1620069042**

**Umme Kulsum Ritu- 1511619042**

**Fatema Tuz Zohra- 1620191042**

**Department of Electrical & Computer Engineering**

**North South University**

**Submission date: 24/07/2020**

## **Letter of Transmittal**

July 2020,

To

DR. MOHAMMAD ASHRAFUZZAMAN KHAN

Associate Professor,

Department of Electrical and Computer Engineering,

North South University, Dhaka.

Subject: Submission of Senior Design Project “Deep Fake Video Detection” Report

Dear Sir,

We would like to submit our senior Design Project Report on “Title” as a part of our BSc program with due respect. The report deals with techniques and theories of Artificial intelligence and convolutional neural networks. The project is created to help individuals in our society to be able to detect any form of fake videos that can harm and jeopardize anybody’s reputation and the image they hold in an organization. We believe that the device establishment will help students and people overcome cyber bullying and make things better for cyber victims in our society.

Cyberbullying in Bangladesh is not only an act to be scorned but an offense and can be punishable by the ICT Act of 2006. The Act provides that a person who knowingly publishes, through a website or electronically, any false and indecent content has the intention of corrupting people. People who are likely to read, see or hear the content or cause damage to a person’s identity, or who may injure religious belief or instigate an offense against any person, shall be guilty of an attack against the individual publishing the material. A penalty with such a crime is prison and fine.

The Capstone project is very much valuable to every student of the engineering department. It helps a student understand the gap and mend the bridge between undergraduate and corporate life. It is considered to be a great learning experience for all engineering students. However, due to the situation

created by COVID 19 pandemic, we tried the highest competence to meet all the dimensions required from this report.

We are very much honored to send your useful appraisal and to obtain this paper. We are pleased that this paper is helpful and insightful, and we will take a straightforward view of the matter.

Sincerely Yours,

Shayekh Mohiuddin Ahmed Navid

153124702

Computer Science and Engineering, Department of ECE

North South University, Bangladesh.

.....

Shamima Haque Priya

1620069042

Computer Science and Engineering, Department of ECE

North South University, Bangladesh

.....

Umme Kulsum Ritu

1511619042

Computer Science and Engineering, Department of ECE

Department of ECE

North South University, Bangladesh

.....

Fatema Tuz Zohra

1620191042

Computer Science and Engineering, Department of ECE

North South University, Bangladesh

## **Approval**

The senior design project entitled “DeepFake Video Detection” by Shayekh Mohiuddin Ahmed Navid (153124702), Shamima Haque Priya (1620069042), Umme Kulsum Ritu (1511619042) and Fatema Tuz Zohra (1620191042) is approved in partial fulfillment of the requirement of the Degree of Bachelor of Science in Computer Science and Engineering on April, 2020 and has been accepted as a satisfactory project.

### **Supervisor:**

Dr. Mohammad Ashrafuzzaman Khan

Assistant Professor

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

### **Department Chairman:**

Dr. Mohammad Rezaul Bari

Associate Professor & Chairman

Department of Electrical and Computer Engineering

North South University

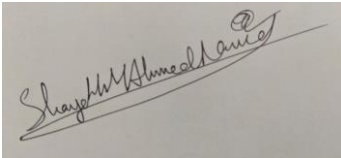
Dhaka, Bangladesh.

## **Declaration**

This is to certify that this project is our original work. No part of this work has been submitted elsewhere partially or fully to award any degree or diploma. Any material reproduced in this project has been appropriately acknowledged.

Students' Names and Signatures:

1. Shayekh Mohiuddin Ahmed Navid

A handwritten signature in black ink, reading "Shayekh Mohiuddin Ahmed Navid" with a circled '2' above the name.

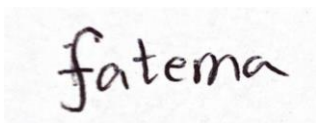
2. Shamima Haque Priya

A stylized handwritten signature in black ink, reading "Shamima Haque Priya".

3. Umme Kulsum Ritu

A handwritten signature in black ink, reading "Umme Kulsum Ritu".

4. Fatema Tuz Zohra

A handwritten signature in black ink, reading "fatema".

## **Acknowledgement**

By the Almighty's grace and mercy, we have completed our senior design capstone project entitled "Deep Fake Video Detection."

Foremost, we would like to express our sincerest gratitude to our advisor Dr. Mohammad Ashrafuzzaman Khan, for his continuous support in our capstone project progress throughout the whole CSE499A and CSE499B, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us all in our research, writing, and completion of this project.

Our sincere thanks also go to North South University, Dhaka, Bangladesh, to provide an opportunity in our curriculum while enabling us to have industrial level experience as part of our academics.

We are also grateful to all the people who took part in our research and shared their valuable opinions regarding this project.

Finally, we would like to thank our families and friends as their inspiration, support, encouragement, and guidance kept us focused and motivated.

## **Abstract**

The rapidly increasing number of photo editing tools like Photoshop and mobile apps such as Snapchat, Adobe Photoshop, PhotoScape, and many more makes it an effortless task to create fake content. We can also argue that the production of counterfeit videos has never been more unaffected. Therefore, our primary goal was to learn the techniques and steps for creating a false video in this case and detect whether an image is fabricated or real **(Sultani and Shah, 2019)**. This paper presents a fundamental learning approach in which computer graphics are separated from authentic photographic images. Our proposed method uses a convolutional neural network with a customized pool to optimize the current algorithms' best-performing feature extraction system. The issue was split into two stages by traditional approaches: visual design or instruction, and prediction. Nevertheless, the research community is now focused on deep learning, inspired by promising results in computer vision.

For several forensic applications, such as fake detections and locations, local descriptors based on image noise remains have been incredibly sufficient. The compression which degrades the data significantly does not usually make conventional imaging forensic techniques appropriate for images.

A forger can change an image with many different image editing operations when making a forged image. A forensic investigator's interest in developing the forensic algorithms that can detect many operations and manipulations of varying image editing has arisen since each one of them is tested by forensic examiners **(Rossler et al., 2019)**.

The rapid progress in the creation and exploitation of digital images has reached a point where significant concerns are raised about society's consequences. At best, this results in a loss of confidence in digital content, but may further cause damage by spreading false or fake information.

We suggested an automatic benchmark for identifying facial distortion to standardize estimating detection methods to centralize our project. The parameter is based mainly on the Deep-Fakes, Face2Face, Face Swap, and Neural textures in the random compression and facial manipulation scale **(Rossler et al., 2019)**. We have also created a falsified video detector and an internet browser plugin. The entire scheme fulfills the aim of creating a cyber-safety management system. We also used different neural network architectures to recognize and identify false and real images. We used a c40 compressed video dataset to plan, test, and evaluate the results during our initial stage.

Fake images and accurate photos taken from fake and actual videos were used in our data collection. Our model was trained in VGG-19 architecture during the initial stage, with a biased subdivision of data. Later, we equipped our model with various CNN architectures, which provided the best accuracy of 88 percent with Mesonet architecture. During our final training sessions, it was essential to have a vast number of data to manipulate the sequences and create our system. Therefore we decided to take part in a contest hosted by Facebook, DeepFake Detection. It was significant for the constructed model to be impartial and to have regular training in knowing which image is actual and which image is false. Therefore, after we developed the predictive working model, we imported a few other data sets from the competition. In simple terms, 45 of the 50 successful data sets were downloaded.

We carried out a detailed study of computer-driven forgery detectors based on the data. We demonstrate that, particularly in the presence of heavy compression, the use of additional domain-specific information increases forgery detection to unparalleled precision, and significantly outperforms human observers.



## **List of tables**

Table 1: List of abbreviations .....	10
Table 2: Results obtained during model training, testing and validation.....	42

## **List of figures**

Figure 1: Faceforensics++ (Learning to Detect Manipulated Facial Images) .....	19
Figure 2: VGG19 architecture .....	22
Figure 3: Resnet50 architecture.....	23
Figure 4: Inception Network .....	24
Figure 5: Inception Resnet v2 layer .....	25
Figure 6: VGGFace CNN Feature descriptors .....	26
Figure 7: Fourier Transformation Formula .....	27
Figure 8: Network Architecture of Meso-4.....	29
Figure 9: Inception Network .....	31
Figure 10: DenseNet Architecture .....	33
Figure 11: ResNet 50 and ResNext residual block .....	34
Figure 12: Backward pass through Batch Normalization Layers.....	36
Figure 13: Model loss result using Mesonet Architecture.....	43
Figure 14: Model Accuracy result using Mesonet Architecture .....	44
Figure 15: (a) & (b) Results of Epochs during model training .....	45

## **List of abbreviations**

Abbreviation	Full form
DL	Deep Learning
PCA	Principal Component Analysis
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
NN	Neural Network
VGG	Visual Geometry Group (University of Oxford, Developer of VGG nets)
IC	Independent Covariant (layer)
ZPA	Zero Phase Analysis

**Table 1: List of abbreviations**

## Table of Contents

<b>Chapter 1</b>	<b>1</b>
<b>1. Introduction</b>	<b>1</b>
1.1 Digital Society	1
1.2 Digital Forensics	2
1.3. DeepFakes	3
1.4. Face2Face and FaceSwap	4
1.5. Neural Textures	4
1.6 Aims and Objectives	5
1.7 Summary	6
<b>Chapter 2</b>	<b>7</b>
<b>2. Literature Review</b>	<b>7</b>
2.1 Introduction	7
2.2 Related works	7
2.2.1 Faceforensics++	8
2.2.2 Xception: Deep Learning with Depth wise Separable Convolutions	9
2.2.3 A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer	10
2.2.4 Distinguishing Computer Graphics from Natural Images Using Convolution Neural Networks	11
2.2.5 DeepFakes: how a pervert shook the world	11
2.2.6 Artificial Intelligence, Deepfakes and a Future of Ectypes	12
2.2.7. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift:	12
2.2.8 Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks:	13
2.2.9 Batch Normalization Biases Deep Residual Networks towards Shallow Paths:	13

2.3 Neural Networks	14
2.4 What is ANN?	15
2.5 Convolutional Neural Network	15
2.6 What is Transfer Learning?	16
2.7 Why is Transfer Learning Useful?	16
2.8 Summary	17
Chapter 3	18
3. Methodology	18
3.1 Introduction	18
3.2 Dataset	18
3.2.1 Faceforensics++	19
3.2.2 Training regimen	20
3.2.3 DeepFake detection challenge by Facebook	21
3.3 Various Neural Networks used for this research	21
3.3.1 VGG-19	21
3.3.2 Resnet50	23
3.3.3 InceptionNet	24
3.3.4 InceptionResnet	25
3.3.5 VGGFace	26
3.3.6 Simple Features	27
3.3.7 Mesonet	28
3.3.8 Dense121	32
3.3.9 ResNext	33
3.3.10 EfficientNet	34
3.3.11 Implementation of the Batch Normalization Layers	35
3.3.12 SphereFace	36

<b>3.3.13 ArcFace</b>	<b>36</b>
<b>3.4. Plugin</b>	<b>37</b>
<b>3.5 Overall Summary</b>	<b>41</b>
<b>Chapter 4: Results and Analysis</b>	<b>42</b>
<b>4.1 Introduction</b>	<b>42</b>
<b>4.2 Results</b>	<b>42</b>
<b>4.3 Overall Summary</b>	<b>45</b>
<b>Chapter 5: Conclusion &amp; Future Work</b>	<b>46</b>
<b>References</b>	<b>48</b>
<b>Appendix A</b>	<b>54</b>
<b>Appendix B</b>	<b>55</b>
<b>Team Photo</b>	<b>77</b>

# Chapter 1

## 1. Introduction

### 1.1 Digital Society

The handling of visual content in our digital society has become omnipresent and one of the most critical subjects. For a range of reasons, faces are of particular interest in modern engineering processes: firstly, the restoration and monitoring of human faces is a well-studied area of computer vision, the basis for this editing approach is to be learned. Furthermore, looks play a central role in social communication. A person's face may highlight a message or even spread a word by itself (**Section 3: Impact of Digital Media on Individuals, Organizations, and Society, 2019**). The present facial treatment techniques can be divided into two categories: manipulation of facial expressions and manipulation of face identities. The expanded use of new media affects people's everyday lives in a broader global context, work, and civil society, and how they interconnect and interact. The impact on individuals and organizations of this increased usage is hugely beneficial. However, not all effects are positive from the increased use of digital media. Hyper-connectivity, the growing digital connecting of people and things, can change social interaction patterns because online interaction can replace face-to-face time. Moreover, increasing automation (and the resultant fractured employment) is endangering the stability of jobs that are historically perceived to be professional in the developing world (**Ademu, Inikpi & Imafidon, Chris, 2012**). It is essential to understand the possibilities and risks of increasing digital media use to enable both the industry and consumers to better exploit the advantage and mitigate adverse effects.

As cyberbullying is increasing day by day, the importance of digital forensics is growing. Understanding digital forensic procedures will collect essential information for prosecuting suspects who jeopardize digital devices or networks **(marketing, media, and media, 2018)**.

Many businesses depend on digital technology and the Internet for their service and development, and many corporations rely on digital technologies for data collection, storage, and recovery. A great deal of data is generated, stored, and circulated by electronic means. Forensic experts need to collect evidence from digital devices more frequently **(Teens, Sexts & Cyberspace: The Constitutional Implications of Current Sexting & Cyberbullying Laws Note 20 William & Mary Bill of Rights Journal 2011-2012, 2020)**.

This project aims to explore various image counterfeiting fields and create a dynamic system for the prevention of cyberbullying. Digital forensics covers a vast area of digital forgeries, so our research mainly focuses on the prevention of image counterfeiting, using various digital forensics techniques.

## **1.2 Digital Forensics**

Digital forensics is an area of research primarily for the identification of image counterfeiting. Much of today's research focuses on picture splicing recognition, photocopying, or camera identification. Throughout recent years, scholars have developed several forensic methods for information to determine the authenticity and history of digital images. The invention of targeted editing detectors has resulted in many significant advances in information forensics **(Ademu, Inikpi & Imafidon, Chris, 2012)**. A significant drawback is an approach to image authentication. Because a forger has multiple editing activities, a forensic examiner has many forensic tests to see if a photo has been edited.

To overcome these issues, there has been a growing interest in designing universal forensic algorithms for many, if not all, editing operations. The danger of false news is widely known today. In a world where more than 100 million hours of video content are viewed daily on social networks, the dissemination of counterfeit videos is rapidly raising concerns **(Ademu, Inikpi & Imafidon, Chris, 2012)**. Although the

identification of image manipulation has improved significantly, the identification of digital video falsification is still a difficult task. Nevertheless, most camera recognition methods cannot be implemented directly to images, mainly due to extreme frame loss after the encoding of recordings.

For almost two hundred years, the controversial art of photography was discipline. Some call it an art form; others see it as glorified media manipulation. Photo manipulation does not just encompass the techniques used in a darkroom **(Belhassen & Stamm, 2016)**. It also refers to computer-based image editing, more commonly known as photoshopping or image editing. Manipulation of images can be done using various alternating techniques. You can apply different textures to uneven surfaces with digital publishing, remove color casts, transform regular pictures to cool 3D images, etc.

### **1.3. DeepFakes**

DeepFakes use a form of artificial intelligence called deep learning as a solution to photoshopping in the 21st century to create photographs of falsification events. Deep fake technologies can generate convincing yet fully fictional images **(Durall et al., 2019)**. DeepFakes has shown how computer graphics and visualization methods can be used to defame individuals by removing one another person's face. The word DeepFakes has commonly become a synonym for deep learning-based face substitution, but it is also the name of a specific method of deception shared across online forums. A face in a target series is replaced by a look found in a video source or image set **(Rossler et al., 2019)**. The system is based on two autoencoders with a standard encoder equipped to reconstruct source and target face training images, respectively. To crop and align the images, a face detector is used. The qualified source side encoder and decoder are added to the target face to create a fake picture. Then the auto-encoder output is combined with the rest of the file using Poisson image editing.



#### **1.4. Face2Face and FaceSwap**

Face2Face approach is one of the most common ways of manipulation of facial expressions. It enables the transition in real time of one person's facial expressions to another person with only material hardware. Face2Face is a facial reenactment program that transfers the manifestations of a source video to a target video while maintaining the target person's identity (**Cozzolino et.al, 2018**). The initial implementation is based on two video feed sources, with the manual keyframe selection. These frames are used to create a complex facial reconstruction that can be used to resynthesize the mask under various illuminations and gestures.

In another method, the task is to replace the face of a person with the face of another person rather than changing expressions. This group is referred to as FaceSwap. FaceSwap is a graphics-based approach for moving the face area from a video source to a video target. The face area is determined based on sparsely identified facial landmarks (**Faceswap, 2018**). Using those landmarks, the method uses blend shapes to match a 3D reference model. Using the input image's textures, this model is back-projected to the target image by reducing the discrepancy between the predicted structure and the identified landmarks. Eventually, the pattern made is combined with the image, and the correction of color is applied.

#### **1.5. Neural Textures**

The final applied benchmark method implemented in this project is the Neural Textures benchmark method. Neural textures are trained maps and are equipped as part of the purpose of scene capture. Neural textures have been processed in 3D mesh proxy charts, as in traditional compositions, but with considerably more details accessible from the high-dimensional attribute charts of recently deferred neural rendering pipelines. Neural textures are end-to-end, so the user can synthesize photo-realistic pictures even though the original 3D content is inaccurate. It uses the original video data to learn the

target person's neural texture, including a network rendering (Rossler et al., 2019). It is equipped with a failure in photometric reconstruction combined with an adversarial loss.

It has been popular for broad-based mobile apps such as Snapchat. DeepFakes can also swap faces by detailed analysis. Although Face Swapping can be performed in real-time with basic computer graphics techniques, DeepFakes requires training for each pair of images, provided that it is an important task. For the detection of fake photos, we have trained our model with various architectures. The description of various architectural networks is provided in the chapter (Nguyen, 2019).

### **1.6 Aims and Objectives**

In this research, we demonstrate that we can automatically and reliably detect these manipulations and thereby lead to significant margins for human observers. Our aim for the project is to build a system that will allow each individual to discover any notion of fake images that could destroy the reputation and cause harmful outcomes in their lives. The process is validated using various forms of available data for the detection of fake photos.

The objective include:

- The collection of datasets available on the Internet will be used for our operating framework and assuring that the dataset has the consent of the Institutional Review Board of North South University.
- Creation of the model and training the model using a considerable amount of data, with varieties of an architectural network.
- Examining the best neural network for processing the model and using it further in determining and producing the best results.

- Comparison of the obtained results to similar scholarly journals and carrying out a discussion based on that.
- Writing the thesis based on the achieved results and discussion carried out with further conclusions and recommendations related to the work.

## **1.7 Summary**

DeepFakeDetection has shown that false images can be produced by putting someone else's face on another individual's picture. This research wanted to demonstrate how easy it is to detect these fake images even more effortlessly than the human eye can observe. FaceForensics++, the paper that started digital forensics analysis, has completed its earlier work with XceptionNet architecture. We also used the benchmark methods to produce Fake photos using Face2Face and FaceSwap methods. We still have to evaluate our results against our intended objective for future references. Identifying different modifying picture editing and identification methods takes a significant deal of work and time. But due to time constraints, we have divided the dataset into sets of preparation, processing, and validation so far. In particular, we are building on recent advances in deep learning with convolutional neural networks to learn convincing picture characteristics.

## **Chapter 2**

### **2. Literature Review**

#### **2.1 Introduction**

Deep learning in a variety of applications has won tremendous success in recent years. This modern machine learning technology has developed exponentially and has been extended to several conventional operations and even to different areas with further possibilities (**Misal, 2019**). The high-performance computing system (HPC) (super-computing, cluster, sometimes), implements the solution to solve significant problems in most cases. Cloud infrastructure considered), which provides tremendous data-intensive business processing possibilities. As data explosion in size, variety, truthfulness, and quantity, measuring output utilizing business class servers and storage progressively are becoming increasingly tricky, generative models are becoming challenging to be studied via deep learning.

The paper intersects a variety of areas in computer vision and digital imaging forensics. In this section, we address the most relevant journals that we have looked through to understand the concept of fake video detection. Section 2.2 consists of the importance and use of artificial neural networks, followed up by what is transfer learning and the use of transfer learning in sections 2.3 and 2.4, followed up by related works of various journals in section 2.5. The chapter concludes with section 2.6, which gives an overall summary of the entire segment.

#### **2.2 Related works**

In this section, the papers are recognized in the context of artificial intelligence to classify, analyze, and summarize related works. It illuminates the advancement of expertise in this area by showcasing what has been done, what has usually been decided upon and developed.

### **2.2.1 Faceforensics++**

FaceForensics++ takes an approach to fake video detection. The inspiration behind this project was the rapid increase of artificial face-making techniques, which caused the loss of faith over digital content and the spread of fake news and misinformation. They have measured the degree to which such videos' detection is impossible even by humans **(Rossler et al., 2019)**.

As mentioned in the paper, modern editing tools have become widely popular for face editing. You are putting one person's face in another person's photo. Nowadays, lightweight tools help to render face swaps, such as snap-chat filters, using mobile phones. Identity modification is where one person's face replaces another person's face in the image or video **(Rossler et al., 2019)**.

The dataset consists of around 1.8 million images from 1000 above real and pristine videos. The dataset was later manipulated as it avoided as many face occlusions as possible, leaving about 509914 images facing the camera.

These four methods require the source and target actor video pairs as inputs. The video resolution varied from 480p, 720p and 1080p. After creating the dataset, they divided the videos into three sections for training, validation, and test set. The ratio was 720, 140, and 140 videos in that order for each of the categories. Forgery detection was treated as a binary classification problem resulting in each video to be fake or real.

A survey was conducted to see the accuracy of the benchmark of fake detection between computer science students. They received 50:50 of real and fake images and were given only a certain amount of time to determine whether the image was real or fake. From these results, it was concluded that humans failed to detect when Face2Face and Neural Textures were used **(Rossler et al., 2019)**.

For the forgery detection method, a conservative cropping method was used where the photo was enlarged by a scale factor of 1.3 around the center of the face tracked. It is better than other naïve approaches, including the entire picture, as opposed to this method, where only the face has been used.

So we can conclude that for the FaceForensics paper, XceptionNet brings out the best result for detecting fake images.

### **2.2.2 Xception: Deep Learning with Depth wise Separable Convolutions**

The inception hypothesis is a concept where a convolution layer attempts to tell filters in a very 3D region, with two spatial dimensions and a channel dimension; thus, one convolution kernel is tasked with cross-channel mapping correlations and spatial correlations at the same time (**Chollet, 2017**).

More specifically, the initial appearance of the standard Inception module occurs at cross channel correlations through a group of 1x1 convolutions. It maps the input file into three or four separate areas smaller than the first output region and thus, mapping all associations in these smaller 3D areas through normal 3x3 or 5x5 convolutions.

A variant of our source module "Extreme," consists of one spatial convolution per 1x1 convolution output stream. The degree "Extreme" version of a source node, the period between convolutions and extreme convolutions, supported this more reliable hypothesis, would initially use a 1x1 convolution to map cross-channel correlations, and then assign the spatial relationships of each output channel seriously (**Chollet, 2017**).

At one end of this continuum, a standard convolution corresponds to the case of a single segment; an extremely serious convolution corresponds to the opposite extreme wherever there is one section per channel; source modules are between splitting most channels into three or four parts (**Chollet, 2017**).

After observing the results, the authors suggest that it be possible to improve the Inception family of architectures employing substitution Inception modules with profoundly severe convolutions, i.e., by building models that could be stacks of profoundly critical convolutions. The authors tended to show that convolutions and intensely extreme convolutions lie at each end of a separate continuum, with source modules associating intermediate degree intent among them.

### **2.2.3 A Deep Learning Approach to Universal Image Manipulation Detection Using a**

#### **New Convolutional Layer**

After performing multiple experiments, the authors confirmed their proposed method. These experiments helped explain how to use CNN, and the algorithm could automatically learn how to identify various image manipulations without relying on pre-selected features or pre-processing (**Bayar, 2016**).

While the design of targeted editing detectors has led to many significant advances in data forensics, this approach to image authentication has a considerable loss. A forensic investigator will conduct a large number of forensic tests to determine if and how the image has been edited (**Bayar, 2016**).

These features are extracted from the image by a set of convolutional filters. The coefficients of which are learned using a technique known as back-propagation then aggregated using a pooling function.

Although CNN's can learn useful features for object recognition adaptively, they are not well suited to detect image manipulation in their current form.

Instead of learning how to identify the traces left by editing and manipulating, the convolutional layers will extract features that capture the image's content. To evaluate the performance of their proposed CNN model for image editing detection, they first built a database of unmodified and edited photos.

Their experimental image datasets consisted of 12 different camera models and devices with no prior tampering pre-processing. In this paper, they proposed an ideal CNN-based universal forgery detection technique that can automatically learn how to detect various image manipulations.

To prevent CNN from learning features that represent the content of the image, they proposed a new convolutional type specifically designed to suppress the content of the image and learn how to detect manipulation (**Bayar, 2016**).

## **2.2.4 Distinguishing Computer Graphics from Natural Images Using Convolution Neural Networks**

**Rahmouni, Nozick, Yamagishi, and Echizen ( 2017)** provide a deep-learning method for separating computer-generated graphics from real photographic images. The proposed method used a Convolutional Neural Network with a custom pooling layer to optimize the current best-performing extraction feature algorithms.

They tested their work on new photo-realistic computer graphics and proved that it outperforms state-of-the-art methods for local and total object classification. Recent advances in image processing show the importance of providing some tools to differentiate computer graphics from original photographic images. Only five different video games were deemed photo-realistic enough to reduce the collection to 1800 models.

Photographic images are high-resolution images taken from the RAISE dataset, which is converted from RAW to JPEG. Out of 3600, the images were split into learning, training, and assessment to create a full-size server. They defined a novel approach for classifying computer graphics and real photographic images that integrates the statistical role of extraction in CNN frameworks and seeks the best features for an active boundary. To challenge their algorithm with today's computer graphics, they've compiled a photo-realistic video-game collection of screenshots (**Rahmouni, Nozick, Yamagishi, and Echizen, 2017**).

## **2.2.5 DeepFakes: how a pervert shook the world**

In recent times, the rise of machine learning-based software used to create DeepFakes is terrifying the world. These DeepFakes are so realistic that they hardly leave traces of their fakeness (**Chawla, 2019**).

This paper aims to give the reader a clear understanding of the deep fakes, such as how deep fakes are produced and used. One of the significant ways to detect deepfakes is by calculating the rate of eye



blinking in a video. A person blinks every 2 seconds, which isn't typically seen in the DeepFakes. The outcome of all these experiments are outstanding, but deepfakes still have lots of area for improvement. To generate an excellent deepfake video, one must have millions of images, including images taken in different flashes of lightning and profile shots from different angles (**Chawla, 2019**). As most of the pictures found online are front faced, it restricts the quality of the Deepfakes detection method. Deepfakes detection method is fascinating and has significant potential for influencing society as it is a shining light that changed humanity's course.

#### **2.2.6 Artificial Intelligence, Deepfakes and a Future of Ectypes**

**Floridi (2018)** focuses on the meaning of authenticity and the use of AI to justify authenticity. The paper aims to discuss the various digital methods that could detect forgery and originality in the artifact region. The article also concentrated on novel methods used to detect fakes.

A computational approach for analysis strokes has been created using a data set of 300 drawings with over 80000 strokes. It turned out the way a person's stroke is as unique as his fingerprint or gait. Microsoft used an algorithm to paint just like Rembrandt! A 3D printer was used to make the painting more realistic, like Rembrandt's drawings. A software that gave voice to the script that was supposed to be read, John. F Kennedy, by analyzing 831 recordings of Kennedy (**Floridi, 2018**).

#### **2.2.7. Batch Normalization: Accelerating Deep Network Training by Reducing Internal**

##### **Covariate Shift:**

It was the first paper to come up with the concept of batch normalization. To figure out what batch normalization is- why it is useful and why it must be done, the authors targeted the problem of the internal covariate shift. Internal covariate shift is the change in the distribution of the nodes of a deep network, while an interface is training. If the internal covariate shift is eliminated, the training process is believed to speed up significantly. Therefore, batch normalization was proposed, which was determined to reduce

the internal covariate shift and considerably cut the training time. To fix the internal covariate shift, the solution proposed in the paper was to normalize each batch by both mean and variance. It was accomplished via normalization steps, in which the means and deviations of input layers were fixed (**Ioffe and Szegedy, 2015**). Batch normalization also had a positive effect on the gradient flow across the network, by reducing the dependency of gradients on parameter scale or the initial values. It enabled them to use higher learning rates without getting into the risk of divergence. It was also stated that batch normalization regularizes the model and reduces the need for dropout. Batch normalization also made it possible to use saturated nonlinearities by preventing the network from getting stuck in saturated models.

### **2.2.8 Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep**

#### **Neural Networks:**

The main focus of work here has been whitening neural networks' inputs, a novel Independent-Covariant (IC) layer. In this paper, the idea of batch normalization and dropout is combined with this IC layer. The method implemented in this paper achieved faster results than traditional whitening methods such as PCA and Zero-phase analysis (ZCA). They also performed experiments that reduced mutual information between the outputs of any two neurons by a factor of  $p^2$  and reduced the correlation coefficient factor by  $p$  (where  $p$  is the dropout probability). It can lead to a fast convergence speed. It has also been proposed that the IC layer be placed before the weight layer instead of the activation layer, as intended. This paper emphasizes the usage of BatchNorm and Dropout along with IC just before the weight layer for faster convergence and more stable training and better generalization performance (**Chen et al., 2020**).

### **2.2.9 Batch Normalization Biases Deep Residual Networks towards Shallow Paths:**

This paper focuses on the essential benefit of batch normalization on residual networks. The authors explored the benefits and drawbacks of Residual Networks, also known as ResNet. Not only that, but they also proposed two additional methods SkipInit and Fixup. SkipInit is putting a scalar multiplier at the end

of every residual branch, and each multiplier is named (alpha). SkipInit is a one-line code change that can train profound residual networks without normalization. As long as alpha is initialized at a value of  $(1/\sqrt{d})$  or smaller where  $d$  is the total number of residual blocks (De and Smith, 2016). Fixup does the same thing but has a few other components. Initialization of remaining blocks to the identity is done by part 1 while multiplying the remaining branch by a factor-beta where beta is less than equal to  $1/\sqrt{d}$ . SkipInit only works well if the batch size is less than 1024 for ResNet50 on ImageNet. The authors provided the different benefits of using batch normalization in shallow and deep residual networks on CIFAR-10 and ImageNet datasets. This paper summarizes three things about batch norm:

1. It can train very deep ResNets, and therefore for shallow paths, a scalar multiplier would be enough. For batch sizes that are below 1024.
2. It improves conditioning, which helps to scale training to larger batch sizes.
3. It has a regularizing effect, which is highly beneficial for anyone who wants to achieve the highest test accuracy.

Batch normalization and SkipInit both at initialization bias deep residual networks towards shallow paths with ethical gradient values (De and Smith, 2016). Therefore this paper widely demonstrated how for small batches often batch normalization can be skipped. Instead of the batch norm, a series of initialization schemes, more regularizations can be used on mini-batches to achieve the same result.

## **2.3 Neural Networks**

Neural networks are a group of algorithms loosely based on the human brain to identify patterns. They interpret protocol or capture raw information as a form of censored data. The designs they know are numerical, stored in vectors, which must be converted into all real-life data, whether images, sounds, text, or time series (Thies et.al, 2019). Neural networks help us to form and classify clusters. You can see them on top of the data you save and maintain as a cluster and grouping layer (Activation Functions in Neural Networks, 2018). They help group unlabeled data according to differences with the example inputs and

distinguish data when they have to work on a classified data set. For our project, we trained our model using various neural networks. Each neural network provided different validation accuracy and loss results, which helped us determine which system provided the best environment for our trained model.

## **2.4 What is ANN?**

A computer model based on the structure and functions of biological neural networks is called an artificial neural network. Data flowing through the system will influence the ANN structure, as the neural network, based on that input and output, changes-or learns in a certain way.

ANN is considered as a nonlinear statistical data modeling method where complex correlations between inputs and outputs are modeled or patterns. An ANN has several benefits, but the fact that it can learn from analyzing data sets is one of the most known. In this way, ANN is used to approximate random functions (**What is an Artificial Neural Network (ANN)? - Definition from Techopedia, 2020**). ANN takes data samples to arrive at solutions, rather than entire data sets, saving time and money. ANNs are considered to be reasonably simple mathematical models for enhancing existing technologies for data analysis.

## **2.5 Convolutional Neural Network**

A Convolutional neural network (ConvNet / CNN) is a deep learning algorithm that can take in the image, give importance to different aspects/objects in the image (learnable weights and biases), and distinguish one from another. The requirement for preprocessing is much lower than other classification algorithms in a ConvNet. Although the filters are hand-engineered in rudimentary methods, ConvNets can learn these filters/characteristics by using backpropagation (**backpropagation bp algorithm: Topics by Science.gov, 2020**).

The master algorithms of computer vision have evolved from groundbreaking neural networks, which have gained significant attention by creating recipes to improve them. Standard attacks consist of

adding/deleting objects while using content from the same image or other sources. Most researchers, therefore, have concentrated on detecting close-ups in the image or through an image archive (**How convolutional neural networks see the world, 2020**). Recently, the multimedia legal community began to focus on the use of depth learning, particularly convolutional neural networks (CNN), based on impressive results in closely related areas of computer vision and pattern recognition.

## **2.6 What is Transfer Learning?**

Transfer learning is a concept in which you transfer the weights of an already trained (pre-trained) model to another model that is set on a different dataset. In practice, very few people train a whole Convolutional Network from the start (with random initialization), as having a data set of sufficient size is relatively rare. If we intend to train a model for some task and domain A is the classic supervised learning scenario of machine learning, we assume that we are provided with labeled data for the same task and domain (**On the importance of transfer learning, 2020**).

'Transfer learning' is usually used when we freeze all levels (on data set A), other than the penultimate layer, of pre-trained neural networks and train the neural network on dataset B, to learn the penultimate layer representation. In most cases, we replace the last layer with another layer of our choice (due to the number of outputs we need to address the new problem (**Kuehne, 2013**)). Fine Tuning works well. The network has already understood so much about lines, curves, and artifacts from the ImageNet dataset (which consists of millions of images) and can link them to the newer dataset. If the network has not improved through Fine Tuning, you may want to 'unfreeze' a few layers further.

## **2.7 Why is Transfer Learning Useful?**

Transfer learning can be critical in some systems and environments, where it is either expensive or impossible to obtain large data sets for training neural networks. Transfer learning has played a significant role in improving our predictive model's quality, particularly during the initial stage of our research, where data was scarce (**How Transfer Learning can be a blessing in deep learning models?, 2020**). Transfer

learning is currently receiving comparatively little attention compared to other fields of machine learning, such as unsupervised learning and validated learning, which have become increasingly popular. As the name suggests, Transfer learning is when an AI model is used to solve a single problem (e.g., image detection or classification). The same model is then used as a starting point to solve another similar problem. It is mainly used to speed up and develop the training cycle because a lot of time and money is spent on computer models (**Agarwal et al., 2019**). This method is intended to improve the model learning and shorten the time substantially and make the learning process fast for the current task. It can also be known as a means of addressing computer vision problems. We used our models for functions and areas, which are-while has impact-the low fruit in terms of the availability of data. We must also teach the transfer of our information into new tasks and areas to serve the long tail of the distribution.

## **2.8 Summary**

After analyzing all possible methods in the artificial sector, digital technologies appear to undermine everyone's confidence in our appraisal of realism or fidelity. We should fix the past of artifacts that are damaged. Therefore, to conclude, more work needs to be done to protect the history of artifacts in this field, whether it is a written document, photos, videos, or painting. Solutions are required to ensure the objects are original and genuine or create them.

## **Chapter 3**

### **3. Methodology**

#### **3.1 Introduction**

This segment serves to provide a thorough description of the process and techniques used to accomplish the goals and objectives of this research. After our extensive research, it was predicted that the best-predicted solution that we found was the implementation of convolutional neural networks for our predictive model for the detection of real and false images. The project includes creating a model that will determine whether a provided image is fake or real. The models generated using various neural networks, and the datasets used to train and test the model are also addressed in this section. In this section, we will be discussing the outcomes of the model, which was achieved while several networks trained it. The findings collected were tabulated, and using the results, the consequences of the findings were analyzed.

In the next section, 3.2 discusses the dataset, followed up by the training regimen and various conducted experiments. Section 3.3 gives a brief description of the neural network and the list of systems used for this project to train the model for achieving the best accuracy. Section 3.4 discusses the plugin created for the project, to build a whole system for the detection of fake images, concluding the chapter in section 3.5, providing an overall summary of the segment.

#### **3.2 Dataset**

The dataset consisted of original and modified fake videos generated by FaceSwap, Face2Face, DeepFakes, and many other benchmark methods. The ratio of counterfeit and original data was kept equal to train our model more efficiently. This set of the dataset contains various image manipulations, which are created using different benchmark methods. Hence, due to the diverse categories of images, the specified dataset is considered to be most precise for the building of image detection predictive models.

Now, to understand the research conducted by Faceforensics++, understanding the fully convolutional neural network theory and how CNN was used to exploit the sequences and identify fake videos using the four benchmark methods was crucial (**Dataset of FaceForensics++, 2020**).

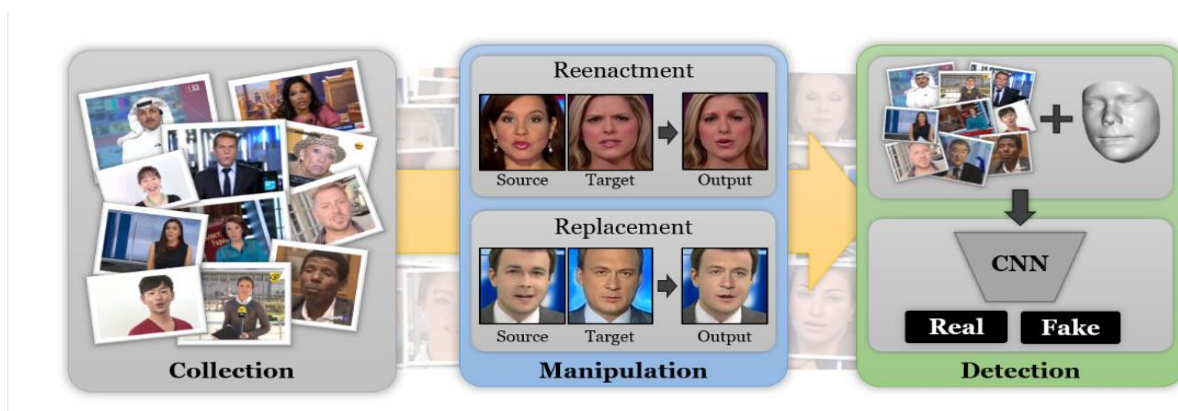
The CNN Image Classifications take an input image, process it, and classify it into specific categories. Computers interpret the image data as a pixel sequence, and this depends on the image quality.

For example, a 6x 6x three image is an RGB matrix, and a 4x 4x 1 model is a matrix of the grayscale color.

Technically, each input image will pass through a sequence of convolution layers with filters, pooling, fully connected layers, and use the Softmax function to identify an object with probabilistic values from 0 to 1.

### **3.2.1 Faceforensics++**

Dataset provided by Faceforensics++ were available on their website, and it was downloaded using a python script that was provided by the authors. The dataset's size was 10 TB, and it was impossible to download the dataset all at once. Hence, we downloaded the available dataset provided by the authors using the python script to train the model. The dataset consisted of 5 categories of manipulated sequences: DeepFakes, Face2Face, FaceSwap, DeepFakeDetection, and Neural Textures.



**Figure 1: Faceforensics++ (Learning to Detect Manipulated Facial Images)**



### **3.2.2 Training regimen**

We cast the forgery identification as a question of the distorted videos per frame binary classification. For original data, we gathered front-facing images from videos comprising news conferences, interviews, or reports about a particular product. Frames were carried out at a rate of 4 frames per second using python. We had initially intended to use the 6th frame. The sixth photos, though, included noise that didn't give us the exact results. The original data was kept in a separate folder after the final decision was made, and the removed images were stored in a different folder.

We split the dataset into a specified test, evaluation, and test collection for all tests, in the ratio of 8:1:1 video, respectively. All measurements are recorded using test set videos. We searched YouTube videos for fake data, such as news videos, film bloopers, etc. We removed the frames again and kept them in a separate folder called fakes. Even data were collected from the dataset given to us via the download script for the dataset. To train our models, we only used frames from the c40 video dataset during the initial stages. Eventually, we trained our model using c23 and raw images. All the photographs were resized to the scale of 254\* 254 before being used for training, respectively. We used VGG-19 architecture during our initial stage for our model testing, followed by various neural networks addressed in depth in section 3.3.

We configured a program named Fakeapp to work on the data, which was downloaded to produce more data, but the software was hard to access and wasn't time-effective when working at the gaming center.

Two folders were created to accumulate original data from original videos; one image from each folder has been taken and merged to generate one fake image with the help of FakeApp.

Upon gathering data from various sites, a data collection of size 180k for instruction, 22.5 k for research, and 22.5k for validation was eventually compiled. Our project needed a high-level GPU processor, as a prerequisite, and finally, we were able to succeed and finish our project with positive results.

### **3.2.3 DeepFake detection challenge by Facebook**

To manipulate the sequences and create our system, it was essential to have a massive set of data. Hence, we decided to participate in the DeepFake detection challenge contest organized by Facebook. We had to build our data environment to work to train our model for data collection applications. It is essential that the model developed is unbiased and has regular training of understanding which image is real and which image is fake. Hence, over time, we imported a few more datasets from the competition after setting up the framework for the model to work. In brief, 45 out of 50 data sets from the competition have been downloaded in total (**Deepfake Detection Challenge Dataset, 2020**).

### **3.3 Various Neural Networks used for this research**

To achieve the best accuracy, it is essential to understand the requirements of a predictive model to produce the best results. Regarding the number of neurons composing each layer, once you know the form of your training results, each parameter is entirely and uniquely defined. The number of neurons that form that layer is equal to the number of functions (columns) in your results. Many implementations of neural networks connect one additional node to a biased term. For the neural network, the pre-trained models include qualified weights. Thus if a pre-trained system is used for various classification processes, the number of steps to converge for the output decreases. That is because the features collected would usually be identical for the classification function. Instead of initializing the model with random weights, initializing it with specific architecture's pre-trained weights decreases the training time and is thus more effective.

#### **3.3.1 VGG-19**

We loaded a model with a VGG-19 architecture, without the top layer (which consists of fully connected layers). Each layer had a parameter called "trainable." To freeze the layers, we had to set the setting to 'False.' Thus, the weights of a single layer were considered to be frozen, meaning that this layer should

not be trained during the implementation of the model. In many cases, as we didn't want to change a layer's weights, we altogether avoided the backward pass by freezing the layers, which resulted in a significant speed increase (Garcia et al., 2017).

On the other hand, we still needed to train the algorithm to offer incorrect predictions if we did freeze it too early. Understand that the elementary shapes, figures, and objects have the most knowledge about the first few layers, really thinking about the dataset. To unfreeze the layers, we set the parameter 'Trainable' to 'True.' Thus, the layers were unfrozen, meaning that this layer should be trained. The accuracy achieved for this network was 97.2%.

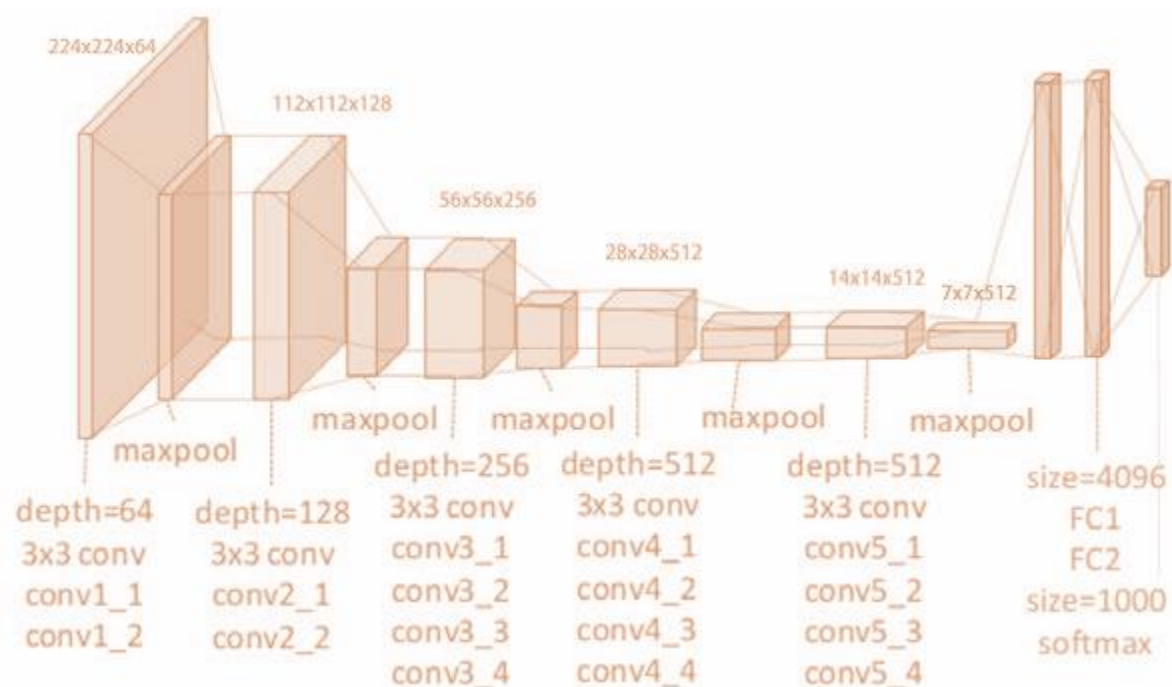


Figure 2: VGG19 architecture

### 3.3.2 Resnet50

The task for the profound fake detection challenge was first to understand whether the model would know whether the provided data is fake or real. ResNet50 is a ResNet prototype variation of 48 layers and 1 MaxPool layer and one average pool layer. It has operations of  $3.8 \times 10^9$  Floating points. It is a standard ResNet model, and we have studied the architecture of ResNet50. According to the structure provided by ResNets, ultra-deep Neural networks were educated, which means that the system can have hundreds or thousands of layers and yet perform highly (**Understanding ResNet50 architecture, 2020**). We implemented an algorithm in our code that can help us extract the labels from the metadata.json file included with each downloaded datasets. Fortunately, frames were obtained following those labels, and we put them in their designated folders: Real or fake. For every image received, we used OpenCV to extract only one specific part: the facial part of the provided pictures. The accuracy of the training session with the ResNet50 architecture was achieved to be 50%. However, for our testing session, our dataset lacked adequate amounts of data. During the training session, the metadata.json file was used to identify fake and real images. Nonetheless, the accuracy began to decrease slowly for this neural network.

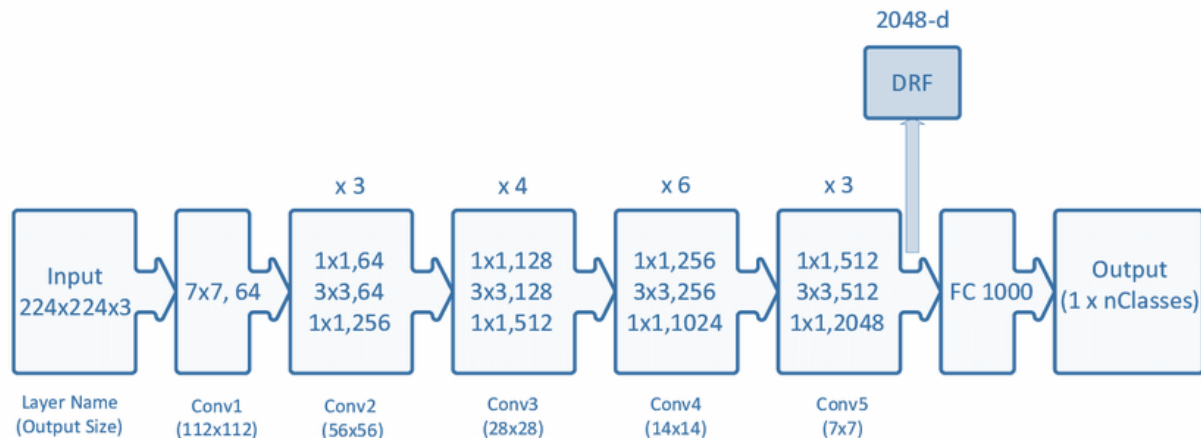


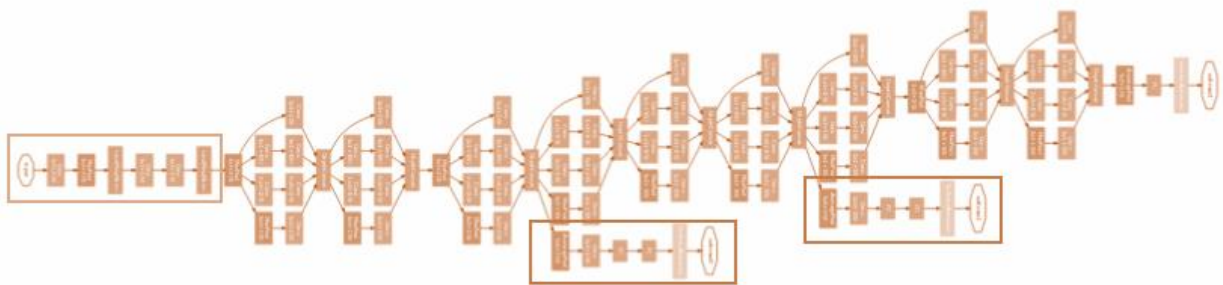
Figure 3: Resnet50 architecture

### **3.3.3 InceptionNet**

A significant step in the creation of CNN classifications was the Inception network. Before it started, most common CNNs had only stacked deeper and deeper convolution layers to achieve better results. On the other side, the Initiation network was dynamic (much engineered). It employed several techniques, both in speed and accuracy, to drive efficiency. The steady growth contributes to creating many network models (**How does the Inception module work in GoogLeNet deep architecture? - Quora, 2020**). The most often used models are:

- Inception v1.
- Inception v2 and Inception v3.
- Inception v4 and Inception-ResNet.

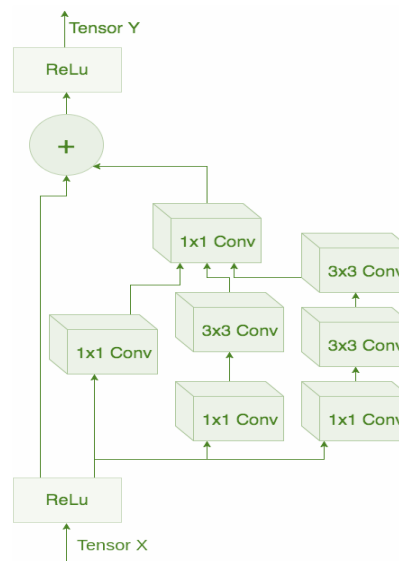
During the first training session, the achieved accuracy was satisfactory with this neural network. We used inceptionv3 for our first training session with the dataset from the DeepFake detection challenge. The accuracy achieved was 87%.



**Figure 4: Inception Network**

### 3.3.4 InceptionResnet

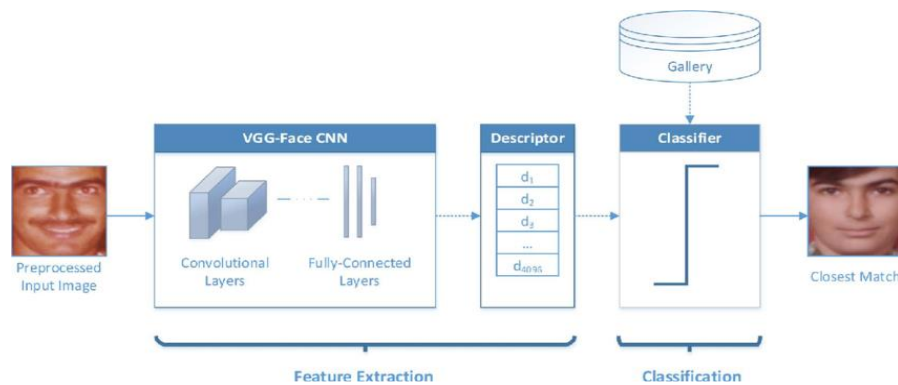
An intermediate starting module was suggested, motivated by the success of the ResNet. There are two Inception ResNet sub-versions: v1 and v2. The cost of Inception-ResNet v1 is similar to the cost of Inceptionv3. InceptionResnet v2 has a measurement expense comparable to Inceptionv4. For the specific modules and the reduction blocks, both subversions have the same structure. The hyper-parameter settings are only the difference **(Pretrained Inception-ResNet-v2 convolutional neural network - MATLAB inceptionresnetv2, 2020)**. After much changes with different architectures, we decided to train our model using the InceptionResnet structure for our image extraction process. We used the concept of fine-tuning and adjusted our parameters very precisely where we keenly observed the changes. We also achieved an accuracy of 89.7%. Followed up next week, we trained our model using InceptionResnet on our first trial. During our first trial, we trained our model with fully connected layers. For our second trial, we prepared our model with a GAP layer. During the training session with a GAP layer, our model provided better results with an accuracy of 87%.



**Figure 5: Inception Resnet v2 layer**

### 3.3.5 VGGFace

Agarwal et al., (2019) and Afchar et al., (2018), conducted experiments which shows an excellent rate of identification for Deepfake with more than 98% and Face2Face with 95%. The papers were collected from the DeepFake detection challenge competition and were reviewed, and we found a few solutions to improve the accuracy by changing our model's parameters. One of the most efficient ways to see an increase in efficiency is to train the model using different architectures. This time we decided to train our model using VGG Face. The implementation included class weight techniques from Keras. The concept of transfer learning is applied here, and pre-trained weights of the VGG Face model is used.



**Figure 6: VGGFace CNN Feature descriptors**

The model is consumed now as an auto-encoder, which will represent images as vectors. In the real class category, we had approximately 6000 data, and in the fake class category, we had 16000 data. Our model was biased towards the false class as the amount of data for the fake class category was more significant than the real class category. Thus class weights function in Keras was used, and the model was trained sequentially, first with genuine frames and later with fake structures. However, only this time, each real instance was used 2.5 times more to balance the phony class. The validation accuracy, however, still didn't increase, somewhat the validation loss decreased from approximately 0.032 to 4.

After the failure of the first attempt, we tried to implement a grid search. However, the Keras API acted as a barrier since the image generator only returns a batch of X and Y's and not the entire dataset on which the grid search should work, and the result would be achieved.

### **3.3.6 Simple Features**

In simple features, two different methods are being implemented. Fourier transformation and Radial profile. **Viola and Jones (2003)** address the issue, namely fake faces, for detecting particular artificial image material. They perform a modern machine-based learning approach to assess the essence of such diagrams. Their method is based on a classical frequency analysis of images that shows various high-frequency behaviors. The study of the frequency domain accompanied by a primary supervised or unmonitored classification identifies these artifacts in their system. Remember that it does not require vast volumes of data in this particular pipeline, which is a beneficial aspect of the data scrambling scenarios. However, they are launching the latest Faces-HQ sequence for the experimental assessment, used to complement the CelebA data collection and FaceForensics++. The Discrete Fourier Transform (DFT) is a statistical method used to decompose a binary signal into sinusoidal elements of varying frequencies from a spectrum of 0 (i.e., the constant rate, which coincides with the traditional picture value) to a maximum frequency, given its spatial resolution. For signals sampled on equidistant lines, it is relatively equivalent to the continuous Fourier transform. It can be measured as for two-dimensional data format M / N in the following way:

$$X_{k,\ell} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x_{n,m} \cdot e^{-\frac{i2\pi}{N} kn} \cdot e^{-\frac{i2\pi}{M} \ell m}.$$

**Figure 7: Fourier Transformation Formula**



The Radial Profile Plot displays the luminance at the position of all pixels. It corresponds to a calculation of the entire target's entire width in the middle of the plot at maximum half (FWHM). It also displays the Gaussian + Constant suit, the context value well away from the Gaussian profile, and the profile peak value. This approach analyzes a gray image and generates a profile plot with a normalized blended width around concentric circles, depending on the distance between the image points (**An Interactive Guide To The Fourier Transform – BetterExplained, 2012**). The center of the rectangle that borders on the current ROI is identified automatically as the stage. In a dialog box, the location of this point can be changed. The size of the pixel values around a circle is at a given distance from the end. This circle has a middle position and the length as a radius from the location. The built-in amplitude is determined by the number of pixels in the image frame, which gives normalized relative values.

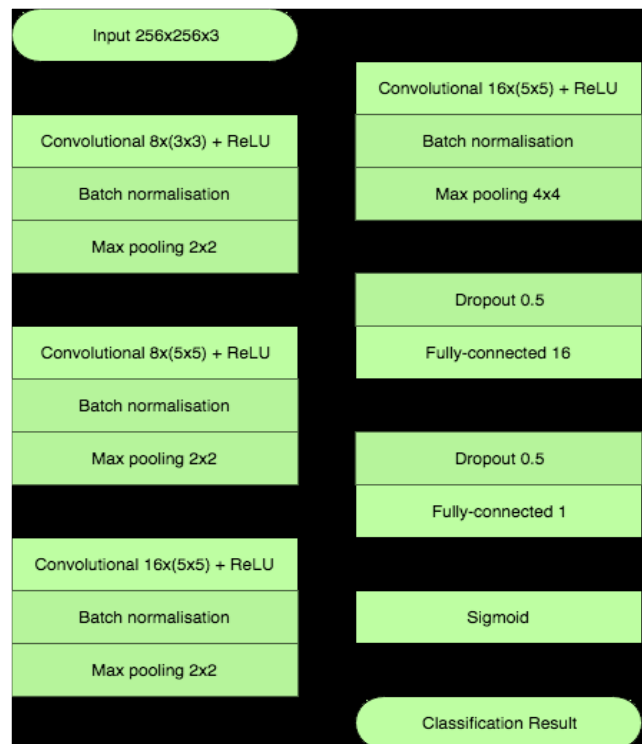
After the implementation of these methods, the classification layers are adjusted. The classifiers can be SVM, random forests, or a neural network. For all of the classification layers, a small amount of data was used. Considering there were some problems with the code during preprocessing or in the neural network architecture, the model gave a validation accuracy of 100% for all the different classifiers. However, when the model was rerun using the Kaggle dataset, it couldn't identify or define any functions. Hence, it couldn't predict whether the provided inputs were real or fake.

Conclusively, 20,000 data were used, and the accuracy achieved after model training was 17%. The accuracy was considered to be reliable as, during the preprocessing stage, it was stated that the model was not able to differentiate whether the provided input was fake or real.

### **3.3.7 Mesonet**

In the DeepFake detection challenge, one of the scholarly journals worked on their project using Mesonet. **Afchar et al. (2018)** demonstrate a steady detection rate for DeepFake of more than 98% and Face2Face of 95%. During the last decades, interactive pictures and videos have been popularized as smartphones

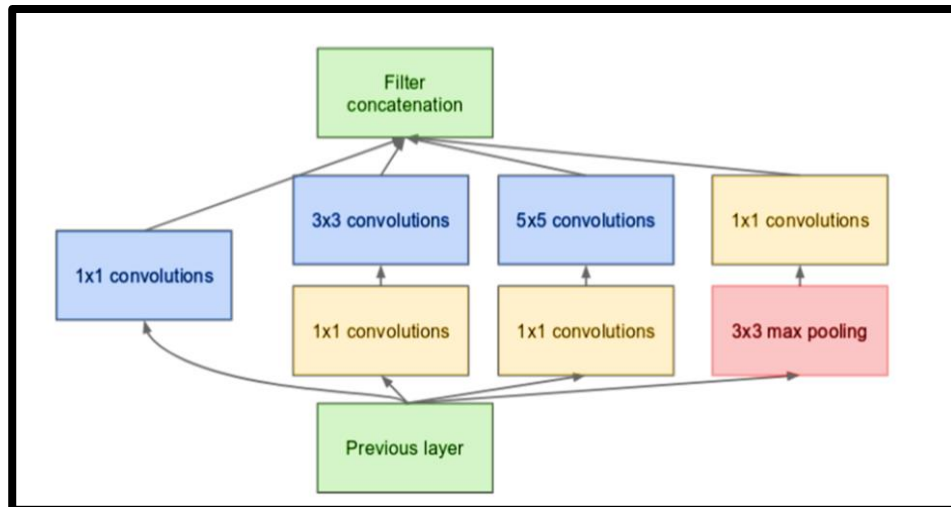
and social networks (Afchar, Nozick, Yamagishi, and Echizen, 2018). Following the enormous use of digital photography, for example, the use of modeling tools such as Photoshop has expanded image modification techniques. Digital forensic photographs are devoted to the identification of counterfeit pictures to monitor their transmission. Compressed image objects are also studied to provide valuable insights into image processing. The experiment's length ranges between two and three minutes with a fixed resolution of 854 X 480 pixels. An H.264 codec format, but with different degrees of compression, allows them to be evaluated under actual circumstances. They investigated how additional facial fragmentations can be identified using the new architecture and the DeepFake dataset. The FaceForensics++ data collection contains over one thousand faked Face2Face-approached videos and their initial dataset as the correct candidate. The data collection is now split into a preparation, evaluation, and research package.



**Figure 8: Network Architecture of Meso-4**

A further benefit of the FaceForensics++ System is the extension of the proposed architecture's use into another classifications function, as it allows fewer missing compressed videos to test the robustness of their models at various compression rates (**Afchar, Nozick, Yamagishi and Echizen, 2020**). A further benefit of the FaceForensics++ System is the extension of the proposed architecture's use into another classifications function, as it allows fewer missing compressed videos to test the robustness of their models at various compression rates.

During the initial stages, only 20% of the dataset was used. However, as different methods from different papers were implemented, it was necessary to use the entire dataset to see the peak difference observed during different architectures and different classifiers being implemented. After the whole dataset was extracted, methods of 'Mesonet' paper were applied. However, the top of the Keras layer didn't work, as stated in the article, and it was becoming difficult to call the object to train the model. The implementations didn't work as predicted. The model was able to predict whether the input was real or fake. However, it wasn't possible to train the model overall. Later on, a visit to Kaggle was committed to checking the different Mesonet methods that were implemented. One of the implementations was Mesonet using inception layers. The inception network is a dynamic (highly designed) network. It has used various tricks in terms of consistency and precision to improve efficiency. Its ongoing growth leads to several network models. Understanding updates will help us create personalized classifications that are both quickly and reliably configured. As stated before, the computational cost of deep neural networks is high. The developers reduce the number of input channels to make it cheaper by adding a 1x 1 convolution between 3x 3 and 5x 5 convolution. While adding a new operation can sound counterintuitive, 1x1 convolutions are much less expensive than 5x5 convolutions, while still helping minimize the number of input channels.



**Figure 9: Inception Network**

Using this layer, we created a model. The code for the model was taken from Kaggle, and then it was used in the model used for the deep fake detection challenge. The weight provided by the mesonet inception layer was made, and later the classification layer was taken for the built-in model. After this, the entire Mesonet layer was trained using a neural network architecture. But the training didn't take place. It is because it was taking a tremendous amount of execution time. Nine hours to be exact. It was imperative to check the execution time to see the running time complexity. After this, steps from the Keras layers were reduced. Eventually, 20% of the datasets were used after this observation. And for each epoch, it took 1 hour. There were 50 epochs. And the accuracy reached 88%. The most exciting part was that, as soon as the epochs started, the accuracy began to increase. Therefore, the assumption is, even though the execution time will increase at the end of the experiment, by expanding the dataset, the accuracy may increase further than the current stage.

During the implementation of Mesonet, the tensor flow was accidentally downgraded. For which, the old TensorFlow CPU was installed, and the tensor flow GPU was not. For which, the model took more time for each epoch. After this, the tensor flow CPU was uninstalled, and the TensorFlow GPU was installed. The version of the library was 1.13 for TensorFlow GPU. GPU is known to be the core of artificial

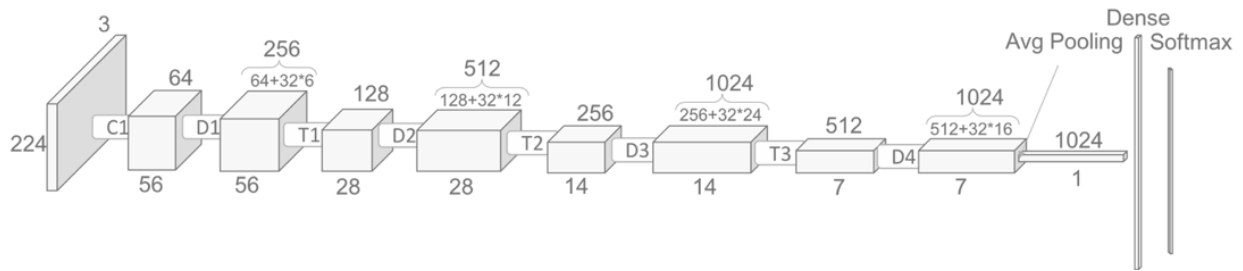
intelligence for Deep Learning. This chip processor works for comprehensive graphical and mathematical calculations and is a single-chip processor, which frees CPU cycles for other work. In fundamental education, the host system runs on the CPU and operates on GPU as a CUDA system.

CPU performs specific functions such as processing of 3D images, vector equations, etc. GPU can have a set of bandwidth limitations, while processors may perform streamlined and complicated tasks for long periods. I.e., it may be challenging to transfer vast volumes of data to the GPU. GPUs have been designed for bandwidth. CPUs have been designed for latency (memory access time).

### **3.3.8 Dense121**

Another experiment that was conducted was the implementation of the Dense121 network. This network is also known as the Dense121 network. Dense Convolutionary Network (DenseNet) that links each layer in a feed-forward manner. In the case of standard L-layer convolutional networks, the network has direct  $L(L+1)/2$  links-one between each layer and the layer that preceded it. The feature maps of all previous layers are used as inputs for each layer, and the feature maps themselves are used as inputs for all layers **(Huang, Liu, van der Maaten and Weinberger, 2017)**.

The persuasive benefits of DenseNet are to ease the issue of fades, improve the replication of features, and promote reuse of features and the number of parameters considerably. The opposite-intuitive result of this dense pattern of networking is that less parameters are needed than conventional convolutional networks, as redundant maps must not be re-learned. Traditional feed architecture can be interpreted as algorithms with a layer-by-layer configuration. Each layer reads and writes to the following layer from its preceding layer **(Nikouei et.al, 2018)**.



**Figure 10: DenseNet Architecture**

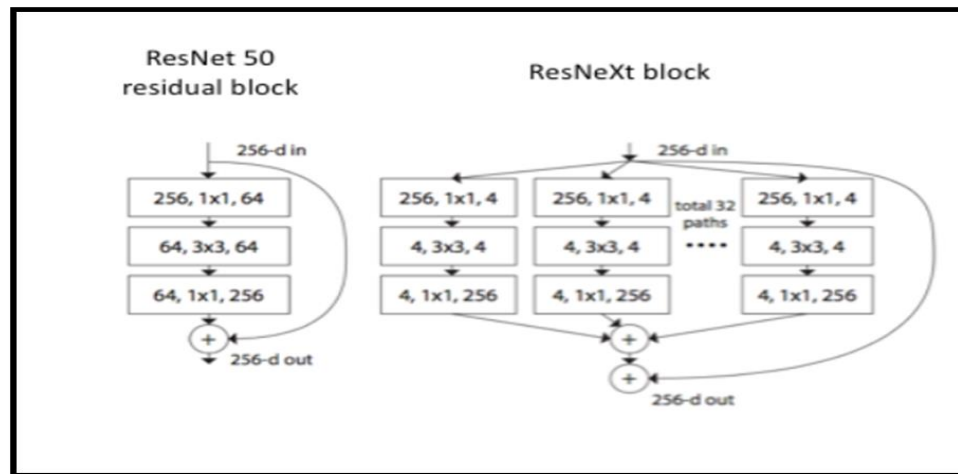
DenseNet's suggested architecture specifically distinguishes between network added information and retained information. DenseNet layers are microscopic, for example, 12 function maps per layer. The "collective information" of the system is only supplied with a few character-maps, and the other feature-maps are left unchanged.

20% of the full dataset was extracted and used for training the model, with the Dense121 network. The accuracy that was conceived was not satisfactory. It was below 80%. Also, DenseNet architecture consists of many parameters. As a result, it took some time to converge. Also, it was observed that the DenseNet architecture was biased towards the minority class, i.e., the real quality. However, there were no conclusive results about how the biasness occurred.

### **3.3.9 ResNext**

The data from face forensics, c40 level videos for the real class, was then extracted to images and was later merged with the existing Facebook DeepFakes detection challenge dataset (only the real quality). It is because there was a shortage of an adequate amount of data for the actual class. There is approximately 2, 50,000 data for the real quality from 1, 50,000, which is a substantial improvement. After the frame extraction, ResNeXt architecture was used for model training (**Zhang & Peng, 2018**). The ResNeXt architecture is a deep network enhancement that replaces the uniform residual block by a technique used

in the original models for a "split-transformation-merge," that is, linking paths within a node. Fundamentally, the input block is projected into a set of low-sizes (channel). They add a few convolutionary filters separately before the convergence of the effects, rather than executing convolutions over the entire input region. The ResNeXt architecture recreates the ResNet models to replace the ResNeXt block.



**Figure 11: ResNet 50 and ResNext residual block**

The architecture was not compatible with Keras for which the architecture was later replaced with methods of EfficientNet paper.

### **3.3.10 EfficientNet**

Scaling does not affect the function of the node. Therefore a strong baseline network is best and then scale it with the proposed compound range in various sizes. The M-NASNet architecture is similar to EfficientNet as it was found in the same search area. The method of the ConvNets scaling will be reconsidered and discussed. They investigate the critical problem in particular: is there a theory method of expanding ConvNets, which can increase precision and efficiency?

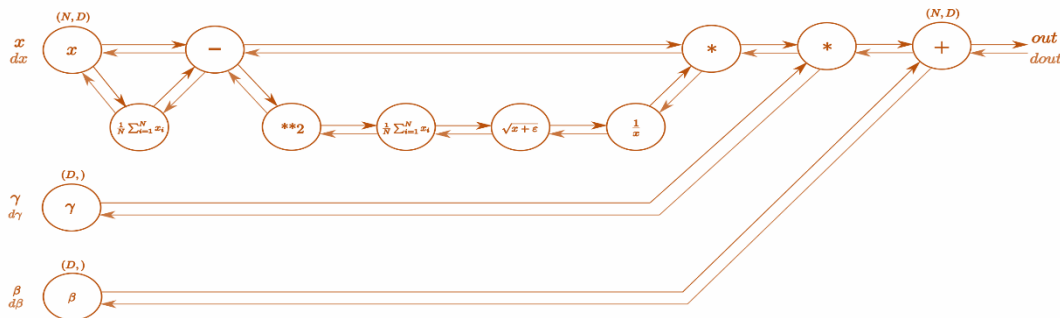
The observational analysis reveals that all network width/depth/resolution measurements are essential to match. Interestingly, such balancing can be accomplished by merely increasing every parameter by a

constant ratio (Tan and Le, 2019). They suggest a basic but powerful scaling approach based on this observation. They note empirically that various aspects of scaling are not distinct. Intuitively, they can expand network size for higher-resolution images, so that more extensive fields will help to collect identical functionalities that have more pixels in more large photos. To capture more finely-grained images and more pixels in high quality, we can also expand network width when the condition is low. These intuitions suggest that various scaling dimensions can be organized and balanced instead of traditional scaling.

The accuracy achieved from this architecture is disappointing, which is 67%, to be exact, for the first ten epochs. However, there is a chance of improving accuracy as the epochs were taking only 35 minutes. In conclusion, EfficientNet didn't provide the expected precision.

### 3.3.11 Implementation of the Batch Normalization Layers

During the training session, it is uncovered that the more fundamental impact of batch normalization makes the optimization landscape smoother, which induces more stable behavior of the gradients, eventually allowing faster training. Even though batch normalization is believed to reduce internal covariate shift (ICS), this paper states that we have little evidence supporting it, and they proved that in some cases, BatchNorm doesn't reduce ICS. What BatchNorm does is it smoothens the landscape of optimization problems, but it isn't the only thing that does it; other normalization techniques also achieve this, and they all perform similarly during training.





### **Figure 12: Backward pass through Batch Normalization Layers**

BatchNorm ensures the gradients are more predictive, allowing a larger range of learning rates and the convergence to get faster. It is mentioned that the Lipschitz-Ness of loss and gradients are improved in models with BatchNorm. The authors emphasized how the internal covariant is not a good predictor of training performance, and on top of that, in many cases, the batch norm does not even reduce the shift. However, it parameterized the underlying optimization problem and made it more stable. It was also observed that batch normalization encourages the training process to converge to more low minima, and those are often believed to facilitate better generalization.

#### **3.3.12 SphereFace**

With improved performance, a different approach to learn biased facial features with angular margin was recommended by utilizing an angular softmax loss with CNNs. This method is known as SphereFace to distinguish facial characteristics. A-Softmax loss provides a pleasant geometric interpretation of a hypersphere multitude by restricting learned features, intrinsically matching the former that faces also lie on a non-linear multiple. This link makes A-Softmax very useful for face representation learning. Competitive results in various popular benchmarks demonstrate today's superiority and high potential (Liu et al., 2020).

Ideal facial characteristics are expected to be less than a minimum intra-class distance below a suitable metric space. However, this criterion can be adequately met with a few existing algorithms. A-Softmax loss can be considered geometrically to impose discriminative limitations on a multitude of the hypersphere, which matches that before where faces are also of multiform. Besides, a parameter  $m$  can adjust the angular margin size quantitatively.

#### **3.3.13 ArcFace**

The design of appropriate loss functions to increase discrimination is one of the main challenges in feature learning using Deep Convolutional Neural Nets (DCNNs) for broad face recognition. To achieve compactness in the intra-class, center loss penalizes the distance between the deep elements and their corresponding class centers. Due to the exact match of the geodesic distance of the hyper ground, ArcFace has a clear geometrical interpretation (**Deng, Guo, Xue and Zafeiriou, 2020**). The loss of Softmax has several inconveniences, one being that it does not explicitly optimize feature embedding to ensure higher similarities for interclass samples and interclass diversity. It results in a deep face performance gap under significant variations of intra-class appearances, such as poses and age differences. A technique has been suggested to boost the differential ability of object embedding obtained from DCNNs to collect facial recognition. The additive angular loss function is also recommended. ArcFace is simple to introduce, needs little more overhead computation, and can converge quickly. ArcFace has a better geometric attribute, despite the numerical similarity between ArcFace, CosFace, and SphereFace, as the angled margin corresponds precisely with the geodesic distance. Using ArcFace data structure loss function, it would have helped us classify and enhance the model's overall performance.

### **3.4. Plugin**

The section of the plugin is divided into two parts. The first part is the plugin itself, and the second part is the client-side/ server-side. The plugin allows the user to crop an image from a specific website with the help of a snipping tool. As soon as the image is cropped with the snipping tool, it is first saved in our computer and then sent to the server-side Editor.html. However, the pictures that will be cropped should contain facial images. Once the image is sent to the server, the image is considered to be an input. On the client-side, there are two options. The first one is to save the file on our desktop, and the second option is to send the data to the model where the image gets processed. The plugin's client part is coded mainly

using Javascript (**Dogaru, 2020**). Unfortunately, JavaScript cannot be used to connect with the model built with Python programming language. So, we used Ajax to send our input image from the client to the server-side using the REST API.

After making the prediction, the model has to return the predicted message whether the image is real or fake to the client-side of the system (plugin) again to display the expected output.

For the google plugin's client-side, the languages we used to implement are HTML, CSS, JavaScript, and a library of Javascript, which is JQuery. Chrome extensions support the JQuery library. JQuery is a JavaScript library that simplifies the HTML DOM tree's delivery and control and the handling of incidents, CSS animation, and Ajax. We used Jcrop( a jQuery image cropping tool), making it easier for us to build the plugin (**Medium, 2020**).

The manifest file contains all the basic information of the plugin, which we created using the instructions provided by google chrome website. We made sure to inject a content script so that the content script could interact with the web pages we would visit. We allowed content script and background script to interact by adding "background.js" in the manifest file. In the background.js, we have added a listener which is fired every time a browser action takes place. So, after the image is cropped, it is sent to the "editor.html page" where the image is saved into the computer. The image is converted into a string before sending it to the model using the "JSON.stringify()" function. After the image is cropped, the user will be able to send it to a specific model (in this case, the model we have developed). After the user clicks the option, 'upload to model,' the browser actions send the JSON string to the model for processing. Once the user clicks the 'upload to model' button, the system uses an Ajax function to connect the model to the plugin. Ajax function initiates a post request to the server-side. On the other hand, the client-side waits for the response of the server-side.

Due to a few complications, this part of the plugin was coded in Google Collab. As google collab can run a virtual server, few libraries needed to be imported. E.g., Socket, Flask, Run\_with\_ngrok. Using the code from the Flask library, a test function was created just to check whether it is possible to get a return function on the server. Once the test didn't run, the Socket library was imported. A new hostname was retrieved. Once the google collab was running as a server, it returned "running flux on Google Collab," on the working PC.

The next step was to check whether the working PC was receiving an image. Several attempts were made to get an outcome. However, the response received was not satisfactory. After several attempts, the following libraries were imported:

Import flask

Import request

Import make response

Import response

Import jsonify from the flask

And Import pil.

However, even though several libraries were imported, the designed library files could not return an image to the working PC. The try/catch method was applied. No response was received. It was understandable that the working PC was collecting data. However, it failed to identify the file as an image. After importing the Pil library, request. getdata action was performed again. Still, no response was received. Later an option called 'Secure. Filename' was obtained, and the .filename was inserted. The .filename was then renamed to App.conflict.imagedirectory. However, the action still failed to get an output.

After several failed attempts, the file was later converted to a NumPy. Array object. Using the `imdecode` function from the OpenCV library, the data was converted to a NumPy. Array object. Unfortunately, this attempt failed as well.

Finally, `'request.files'` (which is an Ajax function) was used, and the request was sent to the server, which was later accessed using the `.read` function. The entire process will give the user a string of images that will later be converted to a NumPy.array object using `NumPy.array.uint8`. The converted image is stored as an NT image variable and resized using `OpenCV.2resize` function. After the image is inserted into the model, it is converted to a colored image so that the model can identify for prediction. As the model considers four dimensions, the image variable will have an increase in one dimension, using `expand.dimension`. The image was then inserted into the `model.predict`. However, it still didn't provide the expected output. It is because the TensorFlow graph (a TensorFlow function) needs to be used as a default while the backend needs to be used. The function was coded according to the requirement. Therefore the backend of TensorFlow can be used for the session. After the requirement is defined, the image is inserted into the model. Predict which is later converted to a string of images that is then returned to the client-side.

Before the importing `model.predict`, it is important to load the mesonet architecture, and the weights for the mesonet architecture followed up by the rest of the mesonet architecture procedures to work on the predicted model. An extra function would be defining the session, where it is important to mention that the tensor flow needs to use the library's backend. After which, the user needs to declare the global variables using `tensorflow.global_variables_initializer` (init) followed up by `session. Run (init)`. Disable the eager execution using TensorFlow to avoid errors. To load the server, use threads. The server is loaded in port 80. To perform this action, the import threading library is imported. Using `threading.thread`, `target=app.run`, write down the hosting Id (0.0.0.0), and the port number (80 for this research). Each time

a unique is generated for this server, it is performed in Google Collab, and it is developing the server. This unique is generated using ngrok. Install ngrok using pip. Ngrok. Once the key is generated, store the key in the client-side. The rest of the step of snipping the images to sending a request to the server to upload the model to prediction is performed. And that is how the output is achieved.

### **3.5 Overall Summary**

From the above discussion, it can be seen that several architectures were used for the implementation of the model, and the objective to achieve better accuracy was assured. The entire work procedure of the plugin system is broadly discussed. All the details of the work methods and predicted outputs are provided. There will be a broad discussion on the analysis of the obtained results with the accuracy values presented in the following chapter, followed by a conclusive decision to predict which architecture provided the best outcome for the project work.

## **Chapter 4: Results and Analysis**

### **4.1 Introduction**

This segment of the analysis is considered one of the most significant sections of this report since it shows the quality of the measurements obtained concerning the various approaches used in the previous discussion. This segment makes the data gathered more efficiently. The following unit, section 4.2, offers an overall overview of the results obtained for each architecture, to be addressed in this segment of this chapter, accompanied by an overall review of the entire category, in section 4.3, thereby concluding the episode.

### **4.2 Results**

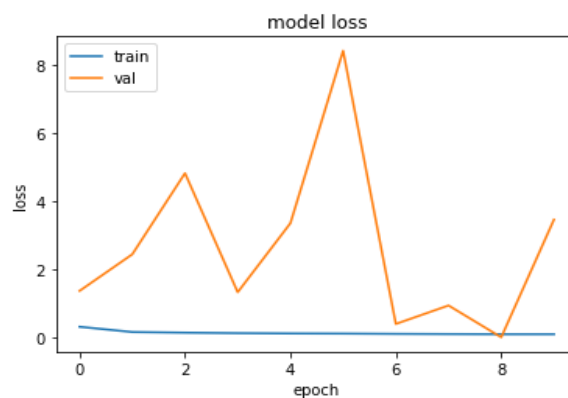
The model was trained with 10 different architectures. The predicted results obtained from each model is summarized in a table below:

<b>Architecture</b>	<b>Predicted Output</b>
VGG-19	97.02% (Training)
InceptionNet	87%
Resnet50	50%
InceptionResnet	87%
VGGFace	Validation Loss: 0.032 (initial) 4 (final)
Simple Features	17%
<b>Mesonet</b>	<b>88%</b>
Dense121	Below 80%
EfficientNet	67%

**Table 2: Results obtained during model training, testing and validation**

From the above-summarized table, it can be seen that VGG-19 provided the best result for the predicted output. However, it is not justified. During the initial stage of model training, most of the images belonged from one class, and the model was trained in a biased ratio. Our model was skewed against the fake class as the volume of data was higher for the fake class category than the real class category. Hence, it is not considered to be the ideal result.

For DeepFake Image identification, the mesonet activation approach has performed exceptionally well. We checked this concept on an in-depth Facebook dataset in our experiment. The mesonet lives up to its standards and achieves maximum accuracy of 88 percent on test sets compared to state-of-the-art accuracy, with a respectable loss of 0.08. The following two graphs are a representation of the results achieved using Mesonet architecture.



**Figure 13: Model loss result using Mesonet Architecture**



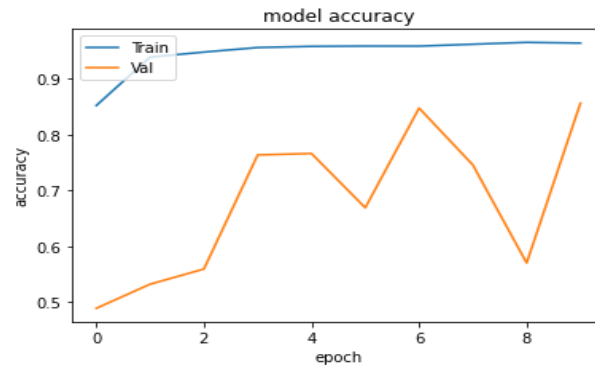


Figure 14: Model Accuracy result using Mesonet Architecture

```

WARNING:tensorflow:From C:\Users\Lab-User\Anaconda3\envs\499Azk\lib\site-packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From C:\Users\Lab-User\Anaconda3\envs\499Azk\lib\site-packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

Epoch 1/5
2287/2287 [=====] - 719s 314ms/step - loss: 3.3732 - acc: 0.7893 - val_loss: 3.0939 - val_acc: 0.8070

Epoch 00001: val_loss improved from inf to 3.09385, saving model to Checkpoints\weights.hdf5
Epoch 2/5
2287/2287 [=====] - 453s 198ms/step - loss: 3.3704 - acc: 0.7897 - val_loss: 3.0923 - val_acc: 0.8071

Epoch 00002: val_loss improved from 3.09385 to 3.09234, saving model to Checkpoints\weights.hdf5
Epoch 3/5
2287/2287 [=====] - 440s 192ms/step - loss: 3.3692 - acc: 0.7898 - val_loss: 3.0923 - val_acc: 0.8071

Epoch 00003: val_loss did not improve from 3.09234
Epoch 4/5
2287/2287 [=====] - 432s 189ms/step - loss: 3.3723 - acc: 0.7896 - val_loss: 3.0923 - val_acc: 0.8071

Epoch 00004: val_loss did not improve from 3.09234
Epoch 5/5
2181/2287 [=====>...] - ETA: 17s - loss: 3.3663 - acc: 0.7900

```

(a)

```

Epoch 1/10
2287/2287 [=====] - 1130s 494ms/step - loss: 0.2519 - acc: 0.8887 - val_loss: 1.7460 - val_acc: 0.7348

Epoch 00001: val_loss improved from inf to 1.74598, saving model to Checkpoints\weights.hdf5
Epoch 2/10
2287/2287 [=====] - 1097s 479ms/step - loss: 0.1579 - acc: 0.9434 - val_loss: 1.7671 - val_acc: 0.7469

Epoch 00002: val_loss did not improve from 1.74598
Epoch 3/10
2287/2287 [=====] - 1095s 479ms/step - loss: 0.1254 - acc: 0.9617 - val_loss: 2.1156 - val_acc: 0.7697

Epoch 00003: val_loss did not improve from 1.74598
Epoch 4/10
2287/2287 [=====] - 1094s 478ms/step - loss: 0.1043 - acc: 0.9722 - val_loss: 1.5504 - val_acc: 0.7460

Epoch 00004: val_loss improved from 1.74598 to 1.55037, saving model to Checkpoints\weights.hdf5
Epoch 5/10
2287/2287 [=====] - 1092s 478ms/step - loss: 0.0909 - acc: 0.9792 - val_loss: 2.1050 - val_acc: 0.7878

Epoch 00005: val_loss did not improve from 1.55037
Epoch 6/10
1480/2287 [=====>.....] - ETA: 5:54 - loss: 0.0813 - acc: 0.9840

```

(b)

Figure 15: (a) & (b) Results of Epochs during model training

### 4.3 Overall Summary

The discussion started in the previous section shows that Mesonet architecture produces the best result for DeepFake detection. There were several limitations during this research, as we lacked enough resources. Due to the pandemic, the last training sessions performed were not up to the expectation set for the available set. It is believed that, with more resources, a better outcome with better accuracy could have been achieved.

## **Chapter 5: Conclusion & Future Work**

Deep learning has been applied extensively for addressing numerous complicated problems like Big Data Processing, Perception, and control at the human level. However, deep learning has also been used to develop applications that can place anonymity, democracy, and national security in danger. Most of the dark devices that have appeared lately are "deeply inaccurate." Deep learning algorithms can generate false videos and photographs that people are unable to differentiate from real images.

Consequently, it is essential to develop technology that automatically identifies and analyzes digital visual media's credibility. The risks of optical facial manipulation are already generally known. We have two potential network architectures to identify these forgeries effectively and with the low computational expense. One primary feature of deep learning is the capacity to solve a specific problem without a

previous theoretical analysis being required. Therefore, this is important that we grasp the roots of this approach to determine its strengths and shortcomings. We have also invested a significant deal of time visualizing our networks' layers and filters.

The consistency of Deepfakes is growing, and identification methods must also be strengthened. The reason is that AI should even repair what has failed. Detective methods are still in the early stage and various techniques, although fragmented data sets have been proposed and evaluated. A way to boost detection efficiency is to build a deep-fault series of revised tests to verify detection methods' continued improvement. It facilitates the detection models' training method, especially those centered on deep learning that requires extensive training. On the other side, the new identification techniques rely primarily on the drawbacks of deep-ground pipelines and thereby stop the competitors' vulnerability in targeting them. In adverse conditions in which the attackers typically seek not to reveal these profound innovative technology, this sort of information and expertise is not accessible. It is a challenge to develop detection methods, and future research needs to focus on introducing robust, scalable, and widespread methods.

For criminal and court trials, recordings and photos were commonly used as testimony. We may include their expertise with computing or law enforcement and their knowledge of capturing, evaluating, and examining the content provided by new technology technical professionals as testimony in the courts. The advancement of computer education and AI technologies can be used to alter this digital material. The views of experts will not be adequate to authenticate this data, because experts themselves cannot distinguish manipulated information. This dimension will also be taken into account in courtrooms, where pictures and images are used to support guilt for the presence of multiple visual deception procedures.

Therefore, explainable AI is a work path in computer vision required to encourage and utilize digital media forensics and machine learning developments.

We comprehended in particular that the face is an essential factor in the detection of DeepFake fabricated aspects. We expect that more technologies will arise to facilitate a more accurate and secure knowledge of deep networks.

## **References**

Ademu, Inikpi & Imafidon, Chris. (2012). THE NEED FOR DIGITAL FORENSIC INVESTIGATIVE FRAMEWORK. International Journal of Engineering Science. 2. 388-392.

Afchar, D., Nozick, V., Yamagishi, J. and Echizen, I., 2018. Mesonet: A Compact Facial Video Forgery Detection Network. [Online] arXiv.org. Available at: <https://arxiv.org/abs/1809.00888>

Agarwal, S., Farid, H., Gu, Y., He, M., Nagano, K. and Li, H., 2019. Protecting World Leaders against Deep Fakes. [online] Openaccess.thecvf.com. Available at: [https://openaccess.thecvf.com/content\\_CVPRW\\_2019/papers/Media%20Forensics/Agarwal\\_Protecting\\_World\\_Leaders\\_Against\\_Deep\\_Fakes\\_CVPRW\\_2019\\_paper.pdf](https://openaccess.thecvf.com/content_CVPRW_2019/papers/Media%20Forensics/Agarwal_Protecting_World_Leaders_Against_Deep_Fakes_CVPRW_2019_paper.pdf)

Ai.facebook.com. 2020. Deepfake Detection Challenge Dataset. [online] Available at: <https://ai.facebook.com/datasets/dfdc/>

Bayar, Belhassen & Stamm, Matthew. (2016). A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. 5-10. 10.1145/2909827.2930786.

Belhassen Bayar and Matthew C. Stamm, 2016. A deep learning approach to universal image manipulation detection using a new convolutional layer. In ACM Workshop on Information Hiding and Multimedia Security, pages 5–10. 3, 6, 7,13, 14

Betterexplained.com. 2012. An Interactive Guide To The Fourier Transform – Betterexplained. [online] Available at: <https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/>

Blog.keras.io. 2020. How Convolutional Neural Networks See The World. [online] Available at: [https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html#:~:text=VGG16%20\(also%20called%20OxfordNet\)%20is,\(ImageNet\)%20competition%20in%202014.](https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html#:~:text=VGG16%20(also%20called%20OxfordNet)%20is,(ImageNet)%20competition%20in%202014.)

Chen, G., Chen, P., Shi, Y., Hsieh, C., Liao, B. and Zhang, S., 2020. Rethinking The Usage Of Batch Normalization And Dropout In The Training Of Deep Neural Networks. [online] arXiv.org. Available at: <https://arxiv.org/abs/1905.05928v1>

Chollet, F., 2017. Xception: Deep Learning With Depthwise Separable Convolutions. [online] arXiv.org. Available at: <https://arxiv.org/abs/1610.02357v3>

Chawla, R., 2019. Deepfakes : How A Pervert Shook The World. [online] Semantic scholar.org. Available at: <https://www.semanticscholar.org/paper/Deepfakes-%3A-How-a-pervert-shook-the-world-Chawla/c3b3a6d27dbbfed4df630b39fc0a8a6692b1828a>

Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva, 2018 . Recasting residual-based local descriptors as convolutional neural networks: an application to image Forgery detection. In

ACM Workshop on Information Hiding and Multimedia Security, pages 1–6, 2017. 3, 6, 7, 13, 14

14 forgery detection network. arXiv preprint arXiv:1809.00888. 3, 6, 7, 13, 14

De, S. and Smith, S., 2016. Batch Normalization Biases Residual Blocks Towards The Identity Function In Deep Networks. [online] arXiv.org. Available at: <https://arxiv.org/abs/2002.10444v2>

Deng, J., Guo, J., Xue, N. and Zafeiriou, S., 2020. ArcFace: Additive Angular Margin Loss For Deep Face Recognition. [Online] arXiv.org. Available at: <https://arxiv.org/abs/1801.07698v3>

Digital Media and Society. 2019. Section 3: Impact Of Digital Media On Individuals, Organizations And Society. [online] Available at: <https://reports.weforum.org/human-implications-of-digital-media-2016/section-3-impact-of-digital-media-on-individuals-organizations-and-society/>

Dogaru, C., 2020. How To Create A JQuery Image Cropping Plugin From Scratch - Part I. [online] Code Envato Tuts+. Available at: <https://code.tutsplus.com/tutorials/how-to-create-a-jquery-image-cropping-plugin-from-scratch-part-i--net-20994>

Durall, R., Keuper,M., Pfrendt, F. and KeuperarXiv, J., 2019. Unmasking DeepFakes with simple Features. [online] 3(1). Available at: < <https://arxiv.org/abs/1911.00686>>

FaceSwap. <https://github.com/MarekKowalski/FaceSwap/>. Accessed: 2018-10-29.1, 2

Fakeapp. <https://www.fakeapp.com/>. Accessed: 2018-09-01. 4

Floridi, L., 2018. Artificial Intelligence, Deepfakes and a Future of Ectypes. Philosophy & Technology, 31(3), pp.317-321.

Garcia-Gasulla, Dario & Parés, Ferran & Vilalta, Armand & Moreno, Jonatan & Ayguadé, Eduard & Labarta, Jesús & Cortés, Ulises & Suzumura, Toyotaro. (2017). On the Behavior of Convolutional Nets for Feature Extraction. Journal of Artificial Intelligence Research. 61. 10.1613/jair.5756.

GitHub. 2020. Dataset of Faceforensics++. [online] Available at:  
<https://github.com/ondyari/FaceForensics>

H. Kuehne, H. Jhuang, R. Stiefelhagen, and T. Serre, 2013. Hmdb51: A large video database for human motion recognition. In HPCSE.

Heinonline.org. 2020. Teens, Sexs & Cyberspace: The Constitutional Implications Of Current Sexting & Cyberbullying Laws Note 20 William & Mary Bill Of Rights Journal 2011-2012. [online] Available at:  
<https://heinonline.org/HOL/LandingPage?handle=hein.journals/wmbrts20&div=27&id=&page=>

Huang, G., Liu, Z., van der Maaten, L. and Weinberger, K., 2017. Densely Connected Convolutional Networks. [Online] arXiv.org. Available at: <https://arxiv.org/abs/1608.06993>

Ioffe, S. and Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training By Reducing Internal Covariate Shift. [online] arXiv.org. Available at:  
<https://arxiv.org/abs/1502.03167v3>

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B. and Song, L., 2020. Sphereface: Deep Hypersphere Embedding For Face Recognition. [online] arXiv.org. Available at:  
<https://arxiv.org/abs/1704.08063v4>

Marketing, O., media, S. and media, S., 2018. Social Networks: Dangers Of Social Media. [online] IONOS Digitalguide. Available at: <https://www.ionos.com/digitalguide/online-marketing/social-media/social-networks-dangers-of-social-media/>

Mathworks.com. 2020. Pretrained Inception-Resnet-V2 Convolutional Neural Network - MATLAB Inceptionresnetv2. [online] Available at:  
<https://www.mathworks.com/help/deeplearning/ref/inceptionresnetv2.html#:~:text=Incep>



[tion%2DResNet%2Dv2%20is%20a,%2C%20pencil%2C%20and%20many%20animals.&text=The%20network%20has%20an%20image,of%20299%2Dby%2D299.](#)

Medium. 2018. Activation Functions In Neural Networks. [online] Available at: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

Medium. 2020. Creating a Screenshot Taking Chrome Extension from Scratch. [Online] Available at: <https://medium.com/iamdeepinder/creating-a-screenshot-taking-chrome-extension-from-scratch-cdb0e33a0013>

Medium. 2020. on the Importance Of Transfer Learning. [Online] Available at: <https://medium.com/ultimate-ai/on-the-importance-of-transfer-learning-c372e904f524#:~:text=To%20deal%20with%20the%20lack,similar%20problems%20quickly%20and%20effectively.&text=This%20reduces%20the%20need%20for,task%20we%20are%20dealing%20with.>

Medium. 2020. How Transfer Learning Can Be A Blessing In Deep Learning Models? [Online] Available at: <https://towardsdatascience.com/how-transfer-learning-can-be-a-blessing-in-deep-learning-models-fbc576dc42>

Misal, D., 2019. Significance of Transfer Learning In The World Of Deep Learning. [online] Analytics India Magazine. Available at: <https://analyticsindiamag.com/transfer-learning-deep-learning-significance/>

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zeroshot learning by convex combination of semantic embeddings. In ICLR, 2014.

NeuralTextures: Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. ACM Transactions on Graphics 2019 (TOG), 2019. 1, 2, 3, 4, 14

Nguyen, H., Yamagishi, J. and Echizen, I., 2019. Use of a Capsule Network to Detect Fake Images and Videos. arXiv, [online] 2(1). Available at: <https://arxiv.org/abs/1910.12467>

Nikouei, S., Chen, Y., Song, S., Xu, R. and Choi, B. and Faughnan, T., (2018). Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN. arXiv, [online] 1(1). Available at: <https://arxiv.org/abs/1805.00330>

OpenGenus IQ: Learn Computer Science. 2020. Understanding Resnet50 Architecture. [online] Available at: <https://iq.opengenus.org/resnet50-architecture/#:~:text=ResNet50%20is%20a%20variant%20of,explored%20ResNet50%20architecture%20in%20depth.>

Rahmouni, N., Nozick, V., Yamagishi, J. and Echizen, I., 2017. Distinguishing computer graphics from natural images using convolution neural networks. 2017 IEEE Workshop on Information Forensics and Security (WIFS).

Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J. and Niessner, M., 2019. Faceforensics++: Learning To Detect Manipulated Facial Images. [online] Openaccess.thecvf.com. Available at: [https://openaccess.thecvf.com/content\\_ICCV\\_2019/html/Rossler\\_FaceForensics\\_Learning\\_to\\_Detect\\_Manipulated\\_Facial\\_Images\\_ICCV\\_2019\\_paper.html](https://openaccess.thecvf.com/content_ICCV_2019/html/Rossler_FaceForensics_Learning_to_Detect_Manipulated_Facial_Images_ICCV_2019_paper.html)

Science.gov. 2020. Backpropagation Bp Algorithm: Topics By Science.Gov. [online] Available at: <https://www.science.gov/topicpages/b/backpropagation+bp+algorithm>

Sultani, W. and Shah, M., 2019. Real-World Anomaly Detection in Surveillance Videos. [Online] arXiv.org. Available at: <https://arxiv.org/abs/1801.04264v3>

Tan, M. and Le, Q., 2019. Efficientnet: Rethinking Model Scaling For Convolutional Neural Networks. [online] arXiv.org. Available at: <https://arxiv.org/abs/1905.11946>

Techopedia.com. 2020. What Is An Artificial Neural Network (ANN)? - Definition From Techopedia. [online] Available at: <https://www.techopedia.com/definition/5967/artificial-neural-network-ann>

Viola, P. and Jones, M., 2003. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001,.

Visual Data Synthesis via GAN for Zero-Shot Video Classification by Chenrui Zhang and Yuxin Peng, Institute of Computer Science and Technology, Peking University

Yu-Gang Jiang, Guangnan Ye, Shih-Fu Chang, Daniel Ellis, and Alexander C Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In ICMR, 2011.

Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In CVPR, 2015.

Zhang, C. and Peng, Y., 2018. Visual Data Synthesis via GAN for Zero-Shot Video Classification. arXiv, [online] 1(1). Available at: < <https://arxiv.org/abs/1804.10073> >

## **Appendix A**

Table 1: List of abbreviations      8

Table 2: Results obtained during model training, testing and validation      49

Figure 1: Faceforensics++ (Learning to Detect Manipulated Facial Images)      31

Figure 2: VGG19 architecture      34

Figure 3: Resnet50 architecture      35

Figure 4: Inception Network      36

Figure 5: Inception ResNet v2 layer      37

Figure 6: VGGFace CNN Feature descriptors	37
Figure 7: Fourier Transformation Formula	39
Figure 8: Network Architecture of Meso-4	41
Figure 9: Inception Network	42
Figure 10: DenseNet Architecture	44
Figure 11: ResNet 50 and ResNeXt residual block	46
Figure 12: Backward pass through Batch Normalization Layers	47
Figure 14: Model Accuracy result using Mesonet Architecture	56
Figure 15: (a) & (b) Results of Epochs during model training	57

## **Appendix B**

### **Github Repository:**

<https://github.com/shayekh00/CSE499-04-FakeVideoDetection>

### **Browser Plugin Code:**

#### **Background.html**

<html>

```
<head>

<script src="skin/js/jquery.min.js"></script>

<script src="background.js"></script>

</head>

<body>

<embed style="width:1px;height:1px;" type="application/x-openinie-zhucai"></embed>

</body>

</html>
```

### **Background.js**

```
var simpleScreenshot = {

    currentImage: "",

    imageCrop: "",

    imageObj: "",

    imageBlob: "",

    virtualImage: "",

    virtualCanvas: "",

    devicePixelRatio: "",

    uploadUrl: 'http://s1.taisp.net/upload.php',

    screenshotLocalName: "",

    linkBlobFile: "",

    getScreenshot: function () {

        simpleScreenshot.getDevicePixelRatio();

        chrome.tabs.captureVisibleTab(null, {}, function (image) {
```

```

        simpleScreenshot.currentImage = image;

    });

},

openEditorPage: function () {

    this.preCropImage();

    chrome.tabs.create({'url': chrome.extension.getURL('editor.html')}, function (tab) {

        simpleScreenshot.croplImage();

    });

},

preCropImage: function () {

    this.virtualImage = new Image;

    this.virtualCanvas = document.createElement('canvas');

    this.virtualCanvas.width = this.imageObj.w * this.devicePixelRatio;

    this.virtualCanvas.height = this.imageObj.h * this.devicePixelRatio;

    this.virtualImage.src = this.currentImage;

},

croplImage: function () {

    var x = 0 - this.imageObj.x;

    var y = 0 - this.imageObj.y;

    this.virtualCanvas.getContext('2d').drawImage(this.virtualImage, x *
this.devicePixelRatio, y * this.devicePixelRatio, this.virtualImage.width, this.virtualImage.height)

    this.virtualCanvas.toBlob(function (blob) {

        simpleScreenshot.imageBlob = blob;

```

```

        simpleScreenshot.linkBlobFile = window.window.URL.createObjectURL(blob);

        }, {type: 'image/png', encoding: 'utf-8'});

        this.imageCrop = this.virtualCanvas.toDataURL("image/png");

    },

    localizeStrings: function () {

        jQuery('.localized_string').each(function () {

            if (jQuery(this).data('l') != undefined)

                jQuery(this).text(chrome.i18n.getMessage(jQuery(this).data('l')));

        });

    },

    getDevicePixelRatio: function () {

        this.devicePixelRatio = window.devicePixelRatio;

    },

    getScreenshotLocalName: function () {

        var date = new Date();

        return date.getDate() + '___' + Math.random() + '.png';

    }

}

chrome.browserAction.onClicked.addListener(function (tab) {

    simpleScreenshot.getScreenshot();

    setTimeout(function () {

        chrome.tabs.executeScript(null,

            {code:
"window.simple_screenshot_overlay_toggle();"});

```

```

        }, 100);

});

chrome.extension.onMessage.addListener(

    function (request, sender, sendResponse) {

        if (request.screenshotCoords != undefined) {

            simpleScreenshot.imageObg = request.screenshotCoords;

            simpleScreenshot.localName = simpleScreenshot.getScreenshotLocalName();

            simpleScreenshot.openEditorPage();

        }

    }

);

```

### **Editor.html**

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title class="located_string">Title</title>

    <link rel="stylesheet" href="skin/css/style_editor.css">

    <script type="text/javascript" src="skin/js/jquery.min.js"></script>

```



```

<script type="text/javascript" src="skin/js/editor.js"></script>

<script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>

</head>

<body>

    <div class="editor_box">

        <div class="control_box">

            <a download="screen.png" href="#" class="save_button"><span class="localed_string" data-
            l="Save">Save First</span></a>

            <!--      <a href="#" class="upload_to_modal"><span class="localed_string" data-
            l="UploadtoServer">Send this file to ML</span></a> -->

            <!-- NAVID CODES!!! -->

            <form id="upload-file" method="post" enctype="multipart/form-data">

                <fieldset>

                    <label for="file">Select a file</label>

                    <input name="file" type="file">

                </fieldset>

                <fieldset>

                    <button id="upload-file-btn" type="button">Upload</button>

                </fieldset>

            </form>

```

```
<!-- NAVID CODES!!! -->

</div>



</div>

<script type="text/javascript" src="../skin/js/editor.js"></script>

</body>

</html>
```

### **Manifest.json**

```
{

  "background": {

    "page": "background.html"

  },

  "browser_action": {

    "default_icon": "icon.png",

    "default_title": ""

  },

  "default_locale": "en",

  "description": "",

  "icons": {

    "128": "icon.png"

  },

  "manifest_version": 2,
```

```
"name": "Simple screenshot",

"permissions": [

    "tabs",

    "activeTab"

],

"version": "1.0",

"content_scripts": [

    {

        "matches": [

            "<all_urls>"

        ],

        "js": [

            "/skin/js/jquery.min.js",

            "/skin/js/Jcrop.js",

            "/skin/js/page.js"

        ],

        "css": [

            "/skin/css/Jcrop.css",

            "/skin/css/style.css"

        ]

    },

    {

        "matches": [

            "https://ajax.googleapis.com/*"

        ],

        "js": [

            "/skin/js/jquery.min.js",

            "/skin/js/Jcrop.js",

            "/skin/js/page.js"

        ],

        "css": [

            "/skin/css/Jcrop.css",

            "/skin/css/style.css"

        ]

    }

],

"content_security_policy": "script-src 'self' https://ajax.googleapis.com; object-src 'self'"

}
```

```
]
}
```

## **Flask Server with Rest API**

!pip install flask-ngrok

Collecting flask-ngrok

Downloading

[https://files.pythonhosted.org/packages/af/6c/f54cb686ad1129e27d125d182f90f52b32f284e6c8df58c1bae54fa1adbc/flask\\_ngrok-0.0.25-py3-none-any.whl](https://files.pythonhosted.org/packages/af/6c/f54cb686ad1129e27d125d182f90f52b32f284e6c8df58c1bae54fa1adbc/flask_ngrok-0.0.25-py3-none-any.whl)

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from flask-ngrok) (2.23.0)

Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.6/dist-packages (from flask-ngrok) (1.1.2)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->flask-ngrok) (2020.6.20)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->flask-ngrok) (3.0.4)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->flask-ngrok) (2.10)

Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests->flask-ngrok) (1.24.3)

Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python3.6/dist-packages (from Flask>=0.8->flask-ngrok) (1.1.0)

Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python3.6/dist-packages (from Flask>=0.8->flask-ngrok) (1.0.1)

Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python3.6/dist-packages (from Flask>=0.8->flask-ngrok) (2.11.2)

Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.6/dist-packages (from Flask>=0.8->flask-ngrok) (7.1.2)

Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from Jinja2>=2.10.1->Flask>=0.8->flask-ngrok) (1.1.1)

Installing collected packages: flask-ngrok

Successfully installed flask-ngrok-0.0.25

```
# !pip install tensorflow-gpu==1.13.1
```

```
!pip install tensorflow-gpu==1.14.0
```

Collecting tensorflow-gpu==1.14.0

Downloading

[https://files.pythonhosted.org/packages/76/04/43153bdfcf6c9a4c38ecdb971ca9a75b9a791bb69a764d652c359aca504/tensorflow\\_gpu-1.14.0-cp36-cp36m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/76/04/43153bdfcf6c9a4c38ecdb971ca9a75b9a791bb69a764d652c359aca504/tensorflow_gpu-1.14.0-cp36-cp36m-manylinux1_x86_64.whl) (377.0MB)

Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (0.10.0)

Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.1.2)

Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (0.35.1)

Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (0.8.1)

Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.12.1)

Requirement already satisfied: keras-applications>=1.0.6 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.0.8)

Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (3.12.4)

Requirement already satisfied: gast>=0.2.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (0.3.3)

Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (0.2.0)

Collecting tensorboard<1.15.0,>=1.14.0

Downloading

<https://files.pythonhosted.org/packages/91/2d/2ed263449a078cd9c8a9ba50ebd50123adf1f8cfbea1492f9084169b89d9/tensorboard-1.14.0-py3-none-any.whl> (3.1MB)

Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.1.0)

Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.15.0)

Collecting tensorflow-estimator<1.15.0rc0,>=1.14.0rc0

Downloading

[https://files.pythonhosted.org/packages/3c/d5/21860a5b11caf0678fbc8319341b0ae21a07156911132e0e71bffd0510d/tensorflow\\_estimator-1.14.0-py2.py3-none-any.whl](https://files.pythonhosted.org/packages/3c/d5/21860a5b11caf0678fbc8319341b0ae21a07156911132e0e71bffd0510d/tensorflow_estimator-1.14.0-py2.py3-none-any.whl) (488kB)

Requirement already satisfied: numpy<2.0,>=1.14.5 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.18.5)

Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-packages (from tensorflow-gpu==1.14.0) (1.32.0)

Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras-applications>=1.0.6->tensorflow-gpu==1.14.0) (2.10.0)

Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from protobuf>=3.6.1->tensorflow-gpu==1.14.0) (50.3.0)

Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow-gpu==1.14.0) (1.0.1)

Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow-gpu==1.14.0) (3.2.2)

Requirement already satisfied: importlib-metadata; python\_version < "3.8" in /usr/local/lib/python3.6/dist-packages (from markdown>=2.6.8->tensorboard<1.15.0,>=1.14.0->tensorflow-gpu==1.14.0) (2.0.0)

Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from importlib-metadata; python\_version < "3.8"->markdown>=2.6.8->tensorboard<1.15.0,>=1.14.0->tensorflow-gpu==1.14.0) (3.2.0)

ERROR: tensorflow 2.3.0 has requirement tensorboard<3,>=2.3.0, but you'll have tensorboard 1.14.0 which is incompatible.

ERROR: tensorflow 2.3.0 has requirement tensorflow-estimator<2.4.0,>=2.3.0, but you'll have tensorflow-estimator 1.14.0 which is incompatible.

Installing collected packages: tensorboard, tensorflow-estimator, tensorflow-gpu

Found existing installation: tensorboard 1.13.1

Uninstalling tensorboard-1.13.1:

Successfully uninstalled tensorboard-1.13.1

Found existing installation: tensorflow-estimator 1.13.0

Uninstalling tensorflow-estimator-1.13.0:

Successfully uninstalled tensorflow-estimator-1.13.0

Found existing installation: tensorflow-gpu 1.13.1

Uninstalling tensorflow-gpu-1.13.1:

Successfully uninstalled tensorflow-gpu-1.13.1

Successfully installed tensorboard-1.14.0 tensorflow-estimator-1.14.0 tensorflow-gpu-1.14.0

pip install tensorflow

import tensorflow as tf

```
print(tf.__version__)

from google.colab import drive

drive.mount("/content/drive/")

Mounted at /content/drive/

%cd "/content/drive/My Drive/Colab Notebooks/"

/content/drive/My Drive/Colab Notebooks


import pandas as pd

import os

import numpy as np

from sklearn.metrics import log_loss

from keras import Model, Sequential

from keras.models import Sequential


from keras.layers import *

from keras.optimizers import *

from sklearn.model_selection import train_test_split

import cv2

from tqdm.notebook import tqdm

import glob

from keras import backend

# import tensorflow as tf

from keras.models import load_model, model_from_json
```



```

from keras.preprocessing.image import load_img

from keras.preprocessing.image import img_to_array

from keras.preprocessing.image import ImageDataGenerator

from keras.callbacks import ModelCheckpoint

import keras

from tensorflow.python.keras.backend import set_session

from tensorflow.python.keras.models import load_model


tf.compat.v1.disable_eager_execution()

print(tf.compat.v1.get_default_graph())


session = keras.backend.get_session()

init = tf.global_variables_initializer()

session.run(init)


# tf_config = some_custom_config

# sess = tf.Session(config=tf_config)

# graph = tf.get_default_graph()


def InceptionLayer(a, b, c, d):

    def func(x):

        x1 = Conv2D(a, (1, 1), padding='same', activation='elu')(x)

```

```
x2 = Conv2D(b, (1, 1), padding='same', activation='elu')(x)
```

```
x2 = Conv2D(b, (3, 3), padding='same', activation='elu')(x2)
```

```
x3 = Conv2D(c, (1, 1), padding='same', activation='elu')(x)
```

```
x3 = Conv2D(c, (3, 3), dilation_rate = 2, strides = 1, padding='same', activation='elu')(x3)
```

```
x4 = Conv2D(d, (1, 1), padding='same', activation='elu')(x)
```

```
x4 = Conv2D(d, (3, 3), dilation_rate = 3, strides = 1, padding='same', activation='elu')(x4)
```

```
y = Concatenate(axis = -1)([x1, x2, x3, x4])
```

```
return y
```

```
return func
```

```
def define_model(shape=(256,256,3)):
```

```
    x = Input(shape = shape)
```

```
    x1 = InceptionLayer(1, 4, 4, 2)(x)
```

```
    x1 = BatchNormalization()(x1)
```

```
    x1 = MaxPooling2D(pool_size=(2, 2), padding='same')(x1)
```

```
    x2 = InceptionLayer(2, 4, 4, 2)(x1)
```

```
    x2 = BatchNormalization()(x2)
```

```

x2 = MaxPooling2D(pool_size=(2, 2), padding='same')(x2)

x3 = Conv2D(16, (5, 5), padding='same', activation = 'elu')(x2)
x3 = BatchNormalization()(x3)
x3 = MaxPooling2D(pool_size=(2, 2), padding='same')(x3)

x4 = Conv2D(16, (5, 5), padding='same', activation = 'elu')(x3)
x4 = BatchNormalization()(x4)

if shape==(256,256,3):
    x4 = MaxPooling2D(pool_size=(4, 4), padding='same')(x4)
else:
    x4 = MaxPooling2D(pool_size=(2, 2), padding='same')(x4)

y = Flatten()(x4)
y = Dropout(0.5)(y)
y = Dense(16)(y)
y = LeakyReLU(alpha=0.1)(y)
y = Dropout(0.5)(y)
y = Dense(1, activation = 'sigmoid')(y)

model=Model(inputs = x, outputs = y)

model.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-4))

#model.summary()

global graph

graph = tf.get_default_graph()

```

```

    return model

model=define_model()

model.load_weights('499\MesoNet Implementation\Dataset\Checkpoint\weights.hdf5')

from flask import Flask, request, make_response, Response

import json

from PIL import Image

import io

import cv2

import numpy as np

from flask import jsonify


#from flask_restful import Resource, Api

import socket

print(socket.gethostbyname(socket.getfqdn(socket.gethostname()))

from flask_ngrok import run_with_ngrok

from flask import Flask

app = Flask(__name__)

run_with_ngrok(app) #starts ngrok when the app is run

@app.route("/")

def home():

    return "<h1>Running Flask on Google Colab!</h1>"

```

```

@app.route('/',methods=['GET','POST'])

def get_and_return_from_server():

    # isthisFile=None

    # #fetched_data=request.get_data()

    # try:

    #     #isthisFile = request.files.get('file')

    #     isthisFile = request.files['file'].read()

    # except Exception as err:

    #     print(err)


    # data =request.files['file']

    # img = Image.open(data)


    # data =request.files['file']

    # filename = secure_filename(file.filename) # save file

    # filepath = os.path.join(app.config['imgdir'], filename);

    # file.save(filepath)

    # cv2.imread(filepath)

```

```

# file = request.files['file']

# npimg = np.fromfile(file, np.uint8)

# file = cv2.imdecode(npimg, cv2.IMREAD_COLOR)


filestr = request.files['file'].read()

npimg = np.fromstring(filestr, np.uint8)


# img = np.fromfile(data, np.uint8)

# print(type(img))

#img = Image.open(request.files['file'])


#UNCOMMENT THESE PARTS FOR MODEL PREDICTION!!!!

# img = np.array(img)

#print(npimg.shape)

img = cv2.resize(npimg,(256,256))

img = cv2.cvtColor(np.array(img), cv2.COLOR_BGR2RGB)

img = np.expand_dims(img, axis=0)


with graph.as_default():

    # set_session(sess)

    backend.set_session(session)

    preds = model.predict(img)

# pred= model.predict(img)

```

```

# print(preds[0][0])

# return jsonify(preds)


# UNCOMMENT THE ABOVE LINE FOR PRINT THE SHAPE OR SHOWING THE IMAGE OF CAPTURED!!

# print(img.shape)

# img.show()

# del img


return str(preds[0][0])


# app.run()


import threading

try:
    threading.Thread(target=app.run(), kwargs={'host':'0.0.0.0','port':80}).start()
except:
    print("ERRRRRRR")

# img = cv2.resize(npimg,(256,256))

# img = cv2.cvtColor(np.array(img), cv2.COLOR_BGR2RGB)

# img = np.expand_dims(img, axis=0)


# with graph.as_default():

```

```

# # set_session(sess)

# backend.set_session(session)

# preds = model.predict(img)


# str(preds[0][0])

from keras.preprocessing import image

import numpy


img_width =256

img_height =256

real_picturePath='499/Others/Real.png'

fake_picturePath='499/Others/Fake.png'


# def make_prediction(picturePath,img_width,img_height):

#     test_image= image.load_img(picturePath, target_size = (img_width, img_height))

#     test_image = image.img_to_array(test_image)

#     test_image = numpy.expand_dims(test_image, axis = 0)

#     test_image = test_image.reshape(img_width, img_height)

#     result = model.predict(test_image)

#     return result


def make_prediction(picturePath,img_width,img_height):

```



```
test_image= image.load_img(picturePath, target_size = (img_width, img_height))
```

```
test_image = image.img_to_array(test_image)
```

```
img = cv2.resize(test_image,(256,256))
```

```
img = cv2.cvtColor(np.array(img), cv2.COLOR_BGR2RGB)
```

```
img = np.expand_dims(img, axis=0)
```

```
result = model.predict(img)
```

```
return result
```

```
print('Actually Real','Predicted:', make_prediction(real_picturePath,img_width,img_height) )
```

```
print('Actually Fake','Predicted:', make_prediction(fake_picturePath,img_width,img_height) )
```

## Team Photo



**Team Members (From the left): Umme Kulsum Ritu, Shayekh Mohiuddin Ahmed Navid, Shamima Haque Priya and Fatema Tuz Zohra**