Final Year Project Report
on

A Deep Learning Framework for Generalized Deep-Fakes Detection

Submitted
in partial fulfillment of
the requirement of the degree for the Degree of Bachelor Of Technology

by

Sarvesh Jitendra Patil
(161090006)
Pranjal Ranjan
(161090046)
Ankit Jogen Parekh
(161090049)
Saksham Santosh Singh
(161090055)
Badhrinarayan Malolan
(161090058)

Under the Guidance of

Dr. Faruk S. Kazi

DEPARTMENT OF ELECTRICAL ENGINEERING
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
(An Autonomous Institute Affiliated to Mumbai University)
(Central Technological Institute, Maharashtra State)
Matunga, MUMBAI - 400019

A.Y. 2019-2020

DECLARATION OF STUDENTS

We declare that work embodied in this Final Year Project titled **"A Deep Learning Framework for Generalized Deep-Fakes Detection"** forms our own contribution of work under the guidance of Dr. Faruk S. Kazi at the Department of Electrical Engineering, Veermata Jijabai Technological Institute, Mumbai.

Sarvesh Jitendra Patil
Roll No.- 161090006

Pranjal Ranjan
Roll No.- 161090046

Ankit Jogen Parekh
Roll No.- 161090049

Saksham Santosh Singh
Roll No.- 161090055

Badhrinarayan Malolan
Roll No.- 161090058

Date:
Place: VJTI, Mumbai

APPROVAL SHEET

The credit report ***"A Deep Learning Framework for Generalized Deep-Fakes Detection"*** by Sarvesh Jitendra Patil, Pranjal Ranjan, Ankit Jogen Parekh, Saksham Santosh Singh and Badhrinarayan Malolan is found to be satisfactory and is approved for the Degree of Bachelor of Technology.

Dr. F.S. Kazi
Supervisor

Examiner

Examiner

Examiner

Place: *Veermata Jijabai Technological Institute, Mumbai.*
Date: / /

CERTIFICATE

This is to certify that Sarvesh Jitendra Patil (161090006), Pranjal Ranjan (161090046), Ankit Jogen Parekh (161090049), Saksham Santosh Singh (161090055) and Badhrinarayan Malolan (161090058) students of B. Tech (Electronics and Telecommunication), Veermata Jijabai Technological Institute, Mumbai have successfully completed the project titled ***"A Deep Learning Framework for Generalized Deep-Fakes Detection"*** under the guidance of Dr. Faruk Kazi.

Dr. F. S. Kazi	Dr. F. S. Kazi
Supervisor	***Head of Electrical Engineering Department***

Place: *Veermata Jijabai Technological Institute, Mumbai.*
Date: / /

Contents

Contents iv

List of Figures vi

List of Tables viii

SYNOPSIS 1
Abstract . . . 1
Technical Keywords . . . 1

PROJECT TIMELINE 2
Tasks . . . 2
Milestones . . . 3

1 INTRODUCTION 5

2 LITERATURE SURVEY 8
2.1 Generation of Deep-Fakes . . . 8
2.2 Detection of Deep-Fakes . . . 9
2.3 Interpretability Methods . . . 10

3 IMAGE DEEP-FAKES 11
3.1 Image Deep-Fakes Generation . . . 11
3.1.1 Synthetic Faces . . . 11
3.1.2 Latent Space Exploration . . . 13
3.2 Image Deep-Fakes Detection . . . 15
3.2.1 Dataset Compilation . . . 15
3.2.2 Deep Learning Architecture Families . . . 15
3.2.3 Results . . . 16

4 VIDEO DEEP-FAKES GENERATION 17
4.1 Collecting faces of real people . . . 17
4.2 Training the autoencoders . . . 18
4.2.1 Autoencoder Architecture . . . 18
4.2.2 Discriminator Architecture . . . 19
4.2.3 Training Process . . . 19
4.3 Creating Videos using trained autoencoders . . . 20
4.4 Deep-Fake Generation Results . . . 20

5 VIDEO DEEP-FAKES DATASETS 22
5.1 Deep-Fake Datasets . . . 22

6 VIDEO DEEP-FAKES DETECTION 26
6.1 Preliminaries . . . 26
6.1.1 Transfer Learning . . . 26
6.1.2 XceptionNet . . . 27
6.1.3 Domain Adaptation and Dataset Shift . . . 28

6.2 Single-Frame Deep-Fakes Detection . . . 28
6.2.1 Compilation of Image Datasets . . . 28
6.2.2 Experiment . . . 29
6.2.3 Results . . . 29
6.3 Multi-Frame Deep-Fakes Detection . . . 30
6.3.1 Compilation of Video Datasets . . . 30
6.3.2 Model Architecture . . . 30
6.3.3 Training . . . 31
6.3.4 Results . . . 31
6.4 Deepfake Detection Challenge . . . 32
6.4.1 About . . . 32
6.4.2 Dataset Curation . . . 32
6.4.3 Approach . . . 33
6.4.4 Leaderboard . . . 34

7 EXPLAINABLE AI FOR VISUAL DEEP-FAKES DETECTION 35
7.1 Motivation . . . 35
7.2 Stages of AI Explainability . . . 35
7.3 Preliminary investigation using Saliency and Gradient-weighted Class Activation Mapping . . . 36
7.4 Dataset and Training . . . 37
7.5 Results . . . 37
7.6 Intermediate Activations . . . 38
7.7 Local Interpretable Model-Agnostic Explanations (LIME) . . . 38
7.8 Layer-wise Relevance Propagation (LRP) . . . 39
7.9 Integrated Gradients . . . 41
7.10 Deconvolutional Network . . . 41
7.11 Guided Backpropagation . . . 42

8 AUDIO DEEP-FAKES DETECTION 43
8.1 Introduction . . . 43
8.2 Preliminaries . . . 43
8.2.1 Features . . . 43
8.2.2 Short-Time Fourier Transform (STFT) . . . 45
8.2.3 Spectrograms . . . 46
8.3 Dataset Curation . . . 46
8.3.1 Feature-Based Datasets . . . 46
8.3.2 Image-Based Datasets . . . 47
8.4 Training . . . 48
8.4.1 Feature-Based approach . . . 48
8.4.2 Image-Based approach . . . 48
8.5 Results . . . 49
8.6 Feature Importance Analysis for Averaged Features Dataset . . . 49
8.6.1 Introduction . . . 49
8.6.2 Pearson's Correlation Coefficient . . . 49
8.6.3 Forward Selection . . . 51
8.6.4 Backward Elimination . . . 51
8.6.5 Recursive Feature Elimination(RFE) . . . 52
8.6.6 LASSO Regularization . . . 53

9 Applications 54

10 Conclusion 56

Bibliography 58

List of Figures

1 Project Timeline . . . 2

1.1 Facially manipulated images of former President Barack Obama . . . 5
1.2 Examples of synthesized faces of StyleGANv2 from thispersondoesnotexist.com . . . 5
1.3 Overview of all the relevant subsystems of our Project . . . 7

3.1 History of GAN based Image Deep-Fakes . . . 11
3.2 Types of Image Deep-Fakes . . . 11
3.3 Basic GAN structure . . . 12
3.4 Generated Images for different Ψ . . . 13
3.5 Results of training StyleGAN2 on FFHQ Dataset . . . 13
3.6 Face Arithmetic . . . 13
3.7 Attribute Manipulation . . . 14
3.8 Face Interpolation . . . 14

4.1 Face Alignment . . . 17
4.2 FaceSwap-GAN encoder and decoder architecture . . . 18
4.3 Discriminator Architecture . . . 18
4.4 Training Phase . . . 19
4.5 Video Conversion . . . 19
4.6 Testing Phase . . . 20
4.7 Faceswapping A and B . . . 20
4.8 Deep-Fake Generation Results . . . 21

5.1 Samples from DFD: Green boxes are original, Red boxes are corresponding Deep-Faked images . . . 22
5.2 Samples from Celeb-DF: Green boxes are original, Red boxes are corresponding Deep-Faked images . 23
5.3 Samples from DFDC Preview: Green boxes are original, Red boxes are corresponding Deep-Faked images . . . 24
5.4 BCG Matrix for the visual and manipulation qualities of the three datasets . . . 24
5.5 Deep-Fake samples from the different datasets . . . 25

6.1 Intermediate Activations of Xception pretrained on ImageNet . . . 26
6.2 Validation accuracies (%) of models trained on individual datasets after one epoch . . . 26
6.3 Depthwise Separable Convolution . . . 27
6.4 XceptionNet Architecture . . . 27
6.5 Image dataset curation pipeline . . . 28
6.6 Video dataset curation pipeline . . . 30
6.7 (a) CNN+LSTM pipeline, (b) LSTM Network . . . 31
6.8 Ensemble Learning pipeline . . . 33
6.9 Our public leaderboard score over time . . . 34

7.1 Flowchart illustrating the three stages of AI explainability . . . 35
7.2 Saliency map representing areas where the trained CNN model (on DFD dataset) is looking at to detect facial manipulation . . . 36
7.3 (a) The face-swap of Jack Ma and Aamir Khan in the video, (b) Grad-CAM visualization of the fake faces of Jack Ma . . . 37

7.4 Input image and a slice of `block2_sepconv2_act` (Activation of the second sepconv layer of the second block) . . . 38
7.5 LIME . . . 38
7.6 1st Row shows three input faces and 2nd row shows their corresponding outputs on analysis with LIME 39
7.7 1st Row shows the input along with it's perturbations and 2nd row shows the LIME descriptions of these inputs. . . . 39
7.8 Layer-wise Relevance Propagation (Courtesy of heatmapping.org) . . . 40
7.9 Comparison of different heatmaps produced by the different LRP rules . . . 40
7.10 `lrp.sequential_preset_b_flat` compared on various perturbed inputs. . . . 41
7.11 `integrated gradients` compared on various perturbed inputs. . . . 41
7.12 `deconvnet` compared on various perturbed inputs. . . . 42
7.13 `guided backpropagation` compared on various perturbed inputs. . . . 42

8.1 Amplitude Plot and Root Mean Square Energy Plot of sample audio signal . . . 43
8.2 Chromagram of sample audio signal . . . 44
8.3 Spectral Centroid Plot of sample audio signal . . . 44
8.4 Spectral Bandwidth Plot of sample audio signal . . . 44
8.5 Spectral Rolloff Plot of sample audio signal . . . 45
8.6 Zero-Crossing Rate Plot of sample audio signal . . . 45
8.7 MFCC Plot of sample audio signal . . . 45
8.8 Feature Vectors extracted from audio data . . . 47
8.9 (a) Mel Spectrogram of sample audio file, (b) Cropped Mel Spectrogram . . . 47
8.10 (a) Amplitude plot of sample audio file, (b) Cropped Amplitude plot . . . 48
8.11 Architecture of LSTM used for the feature-based dataset . . . 48
8.12 Architecture of CNN used for the image-based datasets . . . 49
8.13 Pearson Correlation Heatmap . . . 50
8.14 Sequential Forward Selection(w. StdErr) . . . 51
8.15 Sequential Backward Elimination(w. StdErr) . . . 52
8.16 Score of n features from 1 to 37 . . . 52
8.17 Feature Importance using Lasso model . . . 53

9.1 Face Capture and Reenactment of RGB Video. Source: Face2Face . . . 54
9.2 (a) . De-ageing plays a prominent role in Netflix's popular crime drama 'The Irishman' with actors Al Pacino and Robert De Niro © Netflix/Alamy, (b) Deepfakery shaved several years off British actor Bill Nighy in 'Pokémon Detective Pikachu' © Warner Bros/Everett Collection/Alamy . . . 55

List of Tables

3.1 Distribution of train, val and test sets . . . 15
3.2 ResNet Family . . . 16
3.3 VGGNet Family . . . 16
3.4 DenseNet Family . . . 16

5.1 Distribution of (a) Gender and (b) Race in DFDC Preview dataset . . . 23

6.1 Distribution of all image datasets . . . 29
6.2 Classification Accuracy Metrics (%) For Self And Cross Testing, With And Without Transfer Learning 30
6.3 Distribution of all image datasets . . . 30
6.4 Classification Accuracy Metrics (%) For various Deep-Fake datasets. . . . 31

7.1 Comparison of Test Accuracies between differently scaled face datasets. . . . 38

8.1 Classification Accuracy Metrics (%) For Development and Evaluation sets. . . . 49

SYNOPSIS

Abstract

Deep-Fakes are emerging as a significant threat to society, with potential to become weapons of mass disinformation and chaos. Simple tools provide ways to produce such digital forgeries at a large scale which makes it crucial to develop counter-attacking approaches for detection of these Deep-Learning based manipulations. This project offers a holistic view of the Deep-Fake technology.

We start off by inspecting the latest method of facial synthesis, StyleGANv2 with which we generate synthetic faces and modify their attributes by exploring the latent space. We also test the capability of some deep learning architecture families to differentiate such generated faces from real faces. After that we analyze a Transfer Learning based Convolutional Neural Network framework for the task of Deep-Fake Video Detection on three of the latest released datasets – DeepFakeDetection (DFD), Celeb-DF, and DeepFakeDetectionChallenge (DFDC) Preview. Additionally, a custom dataset of high-quality Deep-Fakes is compiled and used for evaluation of models. The critical problem of dataset shift and its effect on domain adaptation is explored by comparing cross-dataset test accuracies, with and without the usage of Transfer Learning. The results indicate that even though Deep-Fake Detection is a highly domain specific task, there is a significant improvement in performance in terms of both single-domain classification accuracy and generalizability by utilizing Transfer Learning.

Deep-Learning based methods of Deep-Fake detection aim to accurately solve the issue at hand but may fail to convince a lay-person of its effectiveness, and thus lack the trust of the general public. Therefore the construction of interpretable and easily explainable models is imperative. In the last part of our work we propose a framework to explain our Deep-Fake Detection model by the usage of various Explainable AI techniques such as LRP and LIME. These are helpful in providing crisp visualizations of the salient regions of the image focused on by the model, which are crucial for the detection of manipulation or lack of it. We hope to use this approach to build trust between AI and human agents and to demonstrate the applicability of XAI in various real-life scenarios.

In addition to Deep-Fakes consisting solely of facial manipulation, we also delve into the subject of audio manipulation. We employ several approaches to tackle the problem of Deep-Fake Audio Detection using the ASVspoof2019 dataset.

Technical Keywords

Deep-Fakes, Digital Forgeries, Manipulation Detection, Convolutional Neural Networks, Transfer Learning, Generalizability, Dataset Shift, Domain Adaptation, Interpretability, Explainable AI (XAI), LRP, LIME

PROJECT TIMELINE

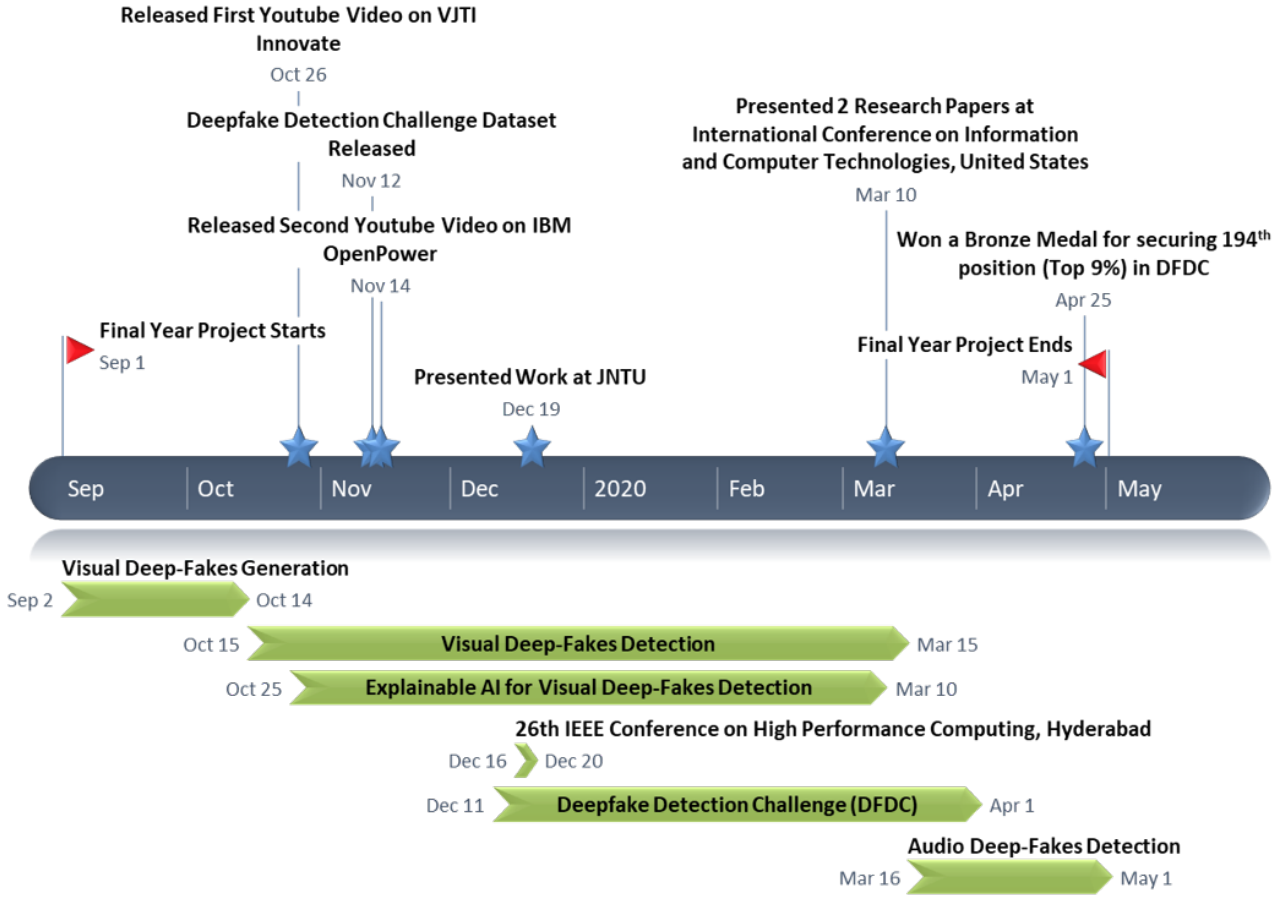


Figure 1: Project Timeline

Tasks

Deep-Fakes Video Generation

Our initial work was focused on generating Deep-Fake Videos. Utilising state of the art hardware, digging and exploring the existing work done on Deep-Fake generation, we developed a general idea of what deep-fakes are and how video clips can be processed and manipulated to generate them

Deep-Fakes Image & Video Detection

In the first discussion held with our respected mentor, Dr. Faruk S. Kazi, we were directed to make the understanding and experience obtained by working on deep-fakes generation as the launchpad to explore Deep-Fakes Detection. This field was yet to be explored in detail and was one of the most interesting cutting edge trends in technology. Thus, we directed our work on the detection of Deep-Fake Images and Videos

Explainable AI for Deep-Fakes Detection

In the subsequent discussions with our respected mentor Dr, Faruk S, Kazi, we showcased some of our preliminary results of Deep-Fake detection where we detected fake faces in popular Deep-Fake videos. In an effort to understand the working of our model, we began exploring the field of Explainable AI (XAI) and worked towards making our model more interpretable.

26th IEEE Conference on High Performance Computing, Hyderabad (HIPC)

We presented our progress on Deep-fake detection at the IEEE Conference on HiPC held at Hyderabad. The conference served as a platform for showcasing current work being done by researchers in the field of high power computing. Since Deep-fakes also make use of high power computational hardware, HiPC served as a great platform to both spread awareness as well as present our work to the right demographic.

We interacted with people from various different fields which included people from academia and working professionals. We received some very helpful inputs on the approaches that can be taken for our detection pipeline by people who specialized in the field of computer vision and AI

Deep-Fakes Detection Challenge (DFDC)

We participated in the Deep-Fake Detection Challenge which invites people around the world to build innovative new technologies that can help detect deepfakes and manipulated media. We tried various methods to improve detection performance on the given dataset which gave us an intuitive understanding of the affect of our various model architectures on the given task at hand. The dataset also gave an insight into deep-faked audios. This piqued our interest in the field of deep-fake audio detection.

Deep-Fakes Audio Detection

Moving forward from Deep-Fakes which were only visually manipulated, we decided to focus on the auditory component of Deep-Fakes. We experimented with several methods to identify spoofed audio from bonafide audio using a variety of approaches. Combined with the Deep-Fake Video Detection pipeline, an all-encompassing framework can be achieved which uses information from both the video as well as the audio to determine the authenticity of digital media.

Milestones

Papers published:

[1] Pranjal Ranjan, Sarvesh Patil, F.S. Kazi, **"Improved Generalizability of Deep-Fakes Detection using Transfer learning based CNN framework"**, under publication in the *3rd International Conference on Information and Computer Technologies 2020* in San Jose, United States (March 9-12, 2020).
Publication link: https://ieeexplore.ieee.org/document/9092019

[2] Badhrinarayan Malolan, Ankit Parekh, F.S. Kazi, **"Explainable Deep-Fake Detection using Visual Interpretability Methods"**, under publication in the *3rd International Conference on Information and Computer Technologies 2020* in San Jose, United States (March 9-12, 2020).
Publication link: https://ieeexplore.ieee.org/document/9092227

Conferences & Workshops:

[1] Demonstrated our DeepFake Detection Pipeline at the *Tech Data* and *IBM* stall in the **26th IEEE International Conference on High Performance Computing, Data, and Analytics(HIPC)** held at Hyderabad International Convention Centre, Hyderabad on 17-20 December, 2019.
Website link: https://openpowerfoundation.org/using-power-technology-to-detect-deepfakes/

[2] Presented our research work in the section *"Deepfake Experiences- IBM AI System"* of the **Academia-Research-Industry Collaborative Learning and Implementation of Industrial Applications using OpenPower AI by School of Information Technology, JNTU** at the Jawaharlal Nehru Technological University Campus, Hyderabad on 19 December, 2019.
Website link: https://jnturesource.com/jntuh-updates/artificial-intelligence-workshop-1.html

[3] Two research papers mentioned above were presented in the **3rd International Conference on Information and Computer Technologies(ICICT)** 2020 in San Jose, United States (March 9-12, 2020).
Website link: http://www.icict.org/

Video Demonstrations:

[1] **Detection of AI Deep-Fake**, in which we use our CNN detection model on the previously generated Deep-Fake video of Jack Ma starring as Aamir Khan of 3 Idiots on YouTube - *VJTI Innovate*.
Video Link: https://youtu.be/Zri0KIRc1LA

[2] **Deep-Fakes Detection for Facial Reenactment Manipulations**, in which we use our Xception model on a popular Deep-Fake video of Former US president Barack Obama made by famous comedian Jordan Peele on YouTube – *OpenPOWER AI Virtual University*.
Video Link: https://youtu.be/0IdKMYSylCk

Deepfake Detection Challenge:

[1] Our team **Refinex@CoE-VJTI** secured a final rank of **193** out of 2265 teams (top 9%) in *Facebook's Deepfake Detection Challenge* hosted on Kaggle, winning a **Bronze** band for the performance.
Leaderboard Link: https://www.kaggle.com/c/deepfake-detection-challenge/leaderboard

Chapter 1
INTRODUCTION

With the advent of the internet and social media platforms, the rate of information sharing is at its highest ever. While this has greatly helped in bringing the world closer, it also carries a severe risk of misinformation and propaganda wreaking havoc. In an era where recording and sharing of digital videos is extremely convenient, digitally modified media can pose serious implications based on the content being altered. Deep-Fakes are one such type of digital manipulations that are powered by recent advances in Deep-Learning to produce hyper-realistic forgeries.

Figure 1.1: Facially manipulated images of former President Barack Obama

The word “Deep-Fake” is a portmanteau of the words "Deep Learning" and "Fake". Hence as the name suggests, Deep-Fakes can be described as fake media (video, images, audio, text) generated by Deep Learning Algorithms such as GANs and Autoencoders. This term was originated after a Reddit user named “deepfakes” started creating face swapped videos of celebrities using Deep Learning. The impressive level of realism achieved by this technique, the large volume of freely available training data, and increasing access to advanced computing resources such as GPUs have resulted in the mass proliferation of these videos.

Video-based Deep-fakes are currently the most popular kind of Deep-fakes on the internet, with Face-swap being the most prevalent one. Faceswap, as the name suggests, involves swapping the faces of two different people in two different images or videos, thus creating two new deep-fakes. They are very popular on social media owing to applications such as FakeApp, FaceApp, DeepFaceLab, etc which make it very easy to create face swaps.

Facial Reenactment is another type of deep-fake where the expressions, pose and mouth movements of a target in a video can be manipulated by a controlling face. If we can generate target audio, we will be able to create a video of our target saying anything we want. These particular type of forgeries have raised a lot of concern because of their massive potential for misuse by nefarious individuals who may use this technology to impersonate prominent individuals such as politicians and actors to spread misinformation or to blackmail celebrities.

Figure 1.2: Examples of synthesized faces of StyleGANv2 from thispersondoesnotexist.com

Apart from these two video-based forgeries, possibly the most bewildering category of deep-faked content are synthesized faces. They are created using cutting edge Deep Learning networks such as StyleGAN which can generate a human face entirely from scratch simply by taking a noise vector as input. With the help of these advanced DL

approaches, it is now possible to create hyper-realistic images of people who have never existed. These could be used to create fake identities by associating a name of a person with the likeness of a face.

Audio Deep-Fakes started to gain momentum after a company called 'Dessa' uploaded its generated Joe Rogan audio Deep-Fake to YouTube. This brought attention towards the implications of audio spoofing being perfected and the criminal purposes it could be used for including spamming of consumers, impersonation of people such as politicians and celebrities either for spreading misinformation or bullying and harassment.

Many Deep-Fakes currently on social media platforms have been generated using Autoencoders. An autoencoder typically comprises of two parts - an encoder and a decoder. The job of the encoder is to reduce the image to a lower dimensional latent space. The decoder then takes this reduced representation and reconstructs the original image. Deep-Fakes utilize this architecture by having a universal encoder which encodes a person into the latent space. The latent representation contains key features about their facial features which then can be decoded with a model trained specifically for the target. Thus, the target's detailed information will be superimposed on the underlying facial features of the original video, represented in the latent space. A significant upgrade to this architecture attaches a generative adversarial network (GAN) to the decoder. A GAN trains a generator, in this case, the encoder, and a discriminator in an adversarial relationship. In the future, generation techniques will become increasingly sophisticated which will synthesize all forms of faces, poses and expressions; creating even more convincing deep-fakes that require lesser training data than ever- maybe just a single image.

Deep-Fakes can have various useful and productive purposes such as film dubbing, special effects, and educational purposes. The malicious purposes of deep-fakes, however, dominate the positive ones. Current Deep-fakes may be easily identifiable, but as time passes they keep improving in rendering quality and become more and more prevalent. This will ultimately result in Deep-Faked content being indistinguishable from the original and genuine counterpart. This will eventually lead to massive mistrust among the general public towards Journalism, the Government, and digital media in general. With the privacy and very identity of the common people compromised, this menace threatens to rip the very social fabric of our society. Governments, journalistic organizations, and technology companies have taken cognizance of this looming danger and have asked the scientific community to develop tools to automatically detect such doctored media. Hence it is crucial to develop effective and robust tools that automatically detect Deep-faked media. Companies such as Facebook, Google, and Microsoft have already begun their ventures to tackle Deep-fake detection by creating datasets to set a Benchmark of performance.

Our project aims to develop frameworks to detect Deep-Faked media such as images, videos, etc. We adopt a Deep-Learning approach where we train a Convolutional Neural Network architecture on the latest Deep-Fake datasets out there. We analyze and compare the results obtained on some of the latest Deep-Fake datasets providing an insight into the kind of data dealt with each of them.

Deep-Fakes are varied in quality of manipulation and often appear in different resolutions, encoding schemes, and efficiencies. We have explored the case of dataset shift where models are trained and tested on different distributions to simulate real-world usage scenarios. This highlights a need to improve the generalizability of deep learning based detection models, for which we study the positive effect of using Transfer Learning based architectures on same and cross-dataset test performances. The differences between the tasks of image and video detection are notable, due to the added temporal nature of video which encapsulates a lot of the discriminating features required. We have explored architectures that incorporate the temporal variations of video Deep-fakes and have trained a Hybrid convolutional LSTM model on video deep-fake datasets.

The black-box approach of Deep Learning has alienated applications where model interpretability is a primary concern, e.g.: Biomedical applications, Healthcare, Financial domains including High-Frequency Trading, etc. There is a clear need for explainability for AI-based solutions to justify their decisions and cultivate trust between human and AI agents. The benefits of investing in XAI are manifold, including assurances provided to business investors about the veracity of AI solutions and compliance with "Right to Explanation" according to GDPR legislation currently in effect across the EU. To provide greater transparency to the inner workings of our model, Explainable AI techniques have been experimented with that give us visualizations of the model's behaviour. We have experimented with both perturbation based and backward up based XAI techniques - Saliency, Grad-CAM, LIME, LRP, Integrated Gradients, Guided Backpropagation, and Deconvolutional Networks respectively.

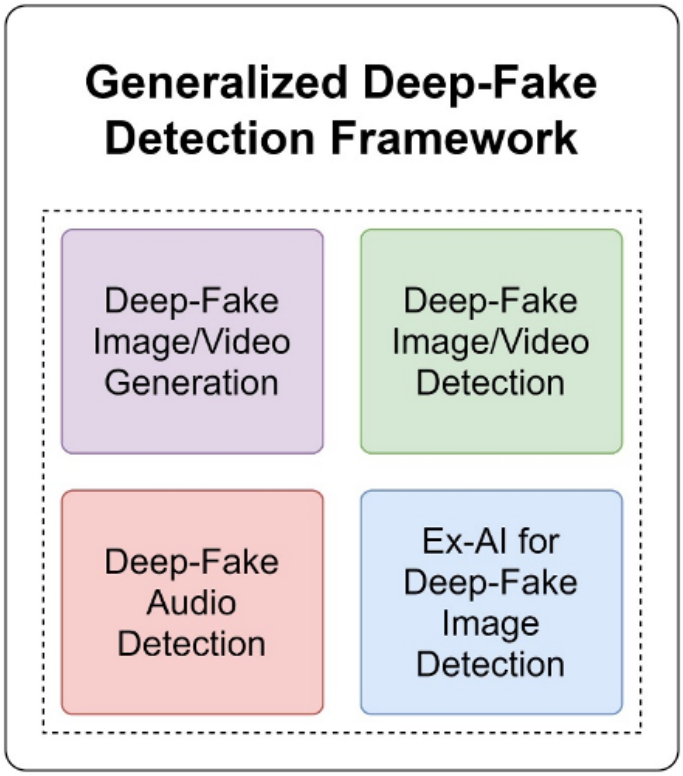


Figure 1.3: Overview of all the relevant subsystems of our Project

Chapter 2
LITERATURE SURVEY

Deep-Fakes are essentially digital forgeries, spanning over all three forms of media – images, videos, and audios. The study and generation of such deep-learning powered forgeries originated much before the coinage of the word - Deep-Fake, and their quality and ease of production has only improved with the onset of cheaper computing power and abundance of data. In order to generalize the detection of Deep-Fakes of different forms and at various granular levels, the study of different kinds of digital forgeries becomes imperative. We analyze the progress of such digital forgeries, till and beyond the emergence of the phenomena of Deep-Fakes. This provides us with a useful hindsight while developing detection systems for such digital forgeries, and in further attempts of explaining the predictions given by these systems.

We then study the literature on the detection of digital manipulations, by both learning and non-learning techniques and delve into the datasets created by several parties for this task. This creates a smooth passage into the background of this research study and offers a broad framework for our work on Deep-Fake detection. We look at different papers focussing on detecting manipulations in images, videos, and audio signals. The insights gained from this review allow us to frame our own ideas, which are implemented and described ahead in the research study.

Finally we end this survey by reviewing several papers on Explainable AI and its application in the field of Computer Vision. We study the mathematical techniques and their applicability to deep-learning based models for increasing interpretability and explainability. This constitutes the base of our motivation to use XAI for Deep-Fake detection models, in order to highlight and segment the manipulations created in images and videos.

2.1 Generation of Deep-Fakes

One of the most significant research papers that laid the foundation for facial manipulation was the Video Rewrite Program [1] in 1997. The objective of this work was to synthesize new facial animations controlled by different audio inputs. This was achieved by their framework which mapped each phoneme in the audio clips to their corresponding mouth movements - visemes, in the videos. It then reordered the different snippets of mouth to match the phoneme sequence of the new audio clip which is supposed to drive the video as per requirement. In the case of missing phonemes in the training set, the model selects the closest possible approximation and treats it similarly. The visemes are then merged with the background of the original footage, resulting in no change in the video except for the new lip synchronization. Video Rewrite combined several ideas from the domains of speech processing - segmenting and labeling phonemes, image processing - morphing techniques to blend the visemes with background, computer vision - tracking landmarks in the face and aligning them, and forged everything together for the automatic generation of driven visual speech. This type of manipulation is now termed as facial reenactment.

Some papers focussed on a complete computer graphics perspective on modeling the face and then introducing perturbations across it to generate new animation. One such work was Making Faces [2] in 1998, which captured the three-dimensional geometry and color information of human facial expressions to reconstruct photorealistic 3D face animations. The system involves sampling a set of points from the face to create a geometric model that accurately reconstructs the required facial animation. This opened up the area of three-dimensional graphics and realism for the generation of digital manipulations.

In 2001, one of the first pattern recognition based approaches was described in Active Appearance Models [3]. This work introduced a construction of statistical models with parameters that controlled shape and gray-level variation of features of faces. The model is trained by learning the relationship between perturbations in the model parameters and the errors thus induced in the generated image.

The next two decades brought together the different communities of computer graphics, deep learning, and signal processing to result in massive breakthroughs in areas of facial manipulation and animation. Face2Face [4] provided a markerless method for monocular facial reenactment in real-time by online transfer of facial expressions of a source actor captured by an RGB camera, to a target actor. Synthesizing Obama [5] focused on learning and manipulating lip sync of various clips of former US President Barack Obama. They achieved this by using a methodology similar to Video Rewrite Program, but used a Recurrent Neural Network to learn a mapping from raw audio features to mouth shapes. HeadOn [6] proposed a real-time source-to-target reenactment approach for complete human portrait videos that enabled the transfer of torso and head motion, face expression, and eye gaze. This involved the usage of RGB-D videos to create geometric proxies with embedded parameters to control the various moving parts.

The rise of Generative Adversarial Networks [7] led to many researchers architecting special GANs to generate

synthetic faces. One of the major breakthroughs came with ProGAN [8] which proposed a methodology of progressively growing both generator and discriminator of the setup. This results in faster training time and increased stability, allowing them to produce high resolution and high-quality faces. They added several features to this in their next paper on StyleGAN [9]. This proposed an alternative generator architecture that leads to an automatically learned, unsupervised separation of high-level attributes (e.g., pose and identity when trained on human faces) and stochastic variation in the generated images (e.g., freckles, hair), and it enables intuitive, scale-specific control of the synthesis. This resulted in photorealistic images, though sometimes being generated with artifacts like water droplets and phase boundaries. This problem was exposed, analyzed, and fixed in their proposal of StyleGAN2 [10]. Here a newer statistical approach was taken for style transfer, and a hierarchical generator with skip connections was used instead of progressive growing.

Deep Learning was leaping forward in the world of audio signals as well, with the paper on WaveNet [11]. This brought forward a deep neural network for generating raw audio waveforms, which when trained for the task of text-to-speech yielded the then state-of-the-art performance. GANSynth [12] outperformed WaveNet baselines on automated and human evaluation metrics, where a GAN was used to generate high-fidelity and locally-coherent audio waveforms. A different perspective of audio generation in the frequency domain was put forward with the paper on MelNet [13], which used two-dimensional time-frequency representations such as spectrograms in conjunction with a highly expressive probabilistic model and a multiscale generative network.

2.2 Detection of Deep-Fakes

Digital image forensics methods have been traditionally based on feature-based detection techniques. While digital signatures and other pre-embedded information can be used to verify the ingenuity of the image, this requires a more proactive approach to be taken by content producers and hence is not always viable. Hence, most detection techniques have to rely on the actual image itself. JPEG compression artifacts have been used in various papers. Luo et al. [14] introduces JPEG error analysis to identify whether bitmap images were previously compressed, estimate quantization steps, and detect the quantization table in a JPEG image. Their capability of detecting JPEG blocks as small as 8x8 helps to localize tampered areas in an image. Bianchi et al. [15] presents the hypothesis that tampered images may be doubly JPEG compressed, and proposes an algorithm that computes a likelihood map showing the probability of each 8x8 JPEG block being doubly compressed.

Apart from feature-based detection methods, deep learning has also been incorporated into the forgery detection task. Bharati et al. [16] propose a supervised deep Boltzmann machine learning algorithm that uses face patches to learn discriminative features aiding classification of original versus retouched images. J. H. Bappy et al. [17] introduce a framework that utilizes an LSTM network along with an encoder-decoder structure to create a mapping from low-resolution activations to pixel-wise predictions for manipulation localization. Recently, Convolutional Neural Networks (CNNs) have been chiefly used to detect manipulated images with great success. El-Latif et al. [18] propose applying Haar Wavelet Transform to features extracted using a CNN, the output of which is then fed to an SVM for classification. Kim et al. [19] proposes feeding hidden image features extracted via a high pass filter to a CNN and classifying it as original or manipulated. Median filtering image forensics is explored by J Chen. et al. [20] by employing consecutive convolutional and pooling layers for feature extraction and classification. Bayar and Stamm [21] observe that traditional CNNs try to capture the content of an image rather than features that are typical of manipulation techniques, and propose a new convolutional layer that is constrained to only learn a set of prediction error filters.

The emphasis on facial manipulation detection was made in the work of Rossler et al. [22]. They categorized facial forgeries into two categories – facial expression manipulation (facial re-enactment) and facial identity manipulation (face-swapping). They created a dataset – "FaceForensics++", consisting of over 5000 videos including forgeries created by four different facial manipulation methods. They also evaluated a number of forgery detection models on their corpus and detailed the results. They observed that the XceptionNet [23] outperformed all the other approaches on uncompressed as well as low-quality images. We build on this observation in this study on Deep-Fake detection and adopt the XceptionNet as our base model for all experimentation. The varying parameters in this study include the usage of Transfer Learning and different combinations of train and test sets chosen from the available database of real and Deep-Fake videos.

With the rise in generative structures for audio waveforms, one of the major challenges in audio forensics was distinguishing between real and synthesized speech. One of the early works in this domain was based on detecting synthetic speech using long term magnitude and phase information [24]. Log magnitude spectrum and phase-based features like frequency derivation and modified group delay were used in this study. Another paper [25] that focussed on vulnerabilities of automatic speaker verification (ASV) systems exploited fundamental frequency variation of

synthetic speech by exploiting spectral features for classification. The emphasis here was on three types of artifacts related to magnitude, phase, and pitch variation, which are introduced during the generation of synthetic speech. A recent research work on the application of bispectral analysis for detecting AI-synthesized speech [26] observed that generative deep neural networks used to synthesize speech can create specific and unusual spectral correlations that are absent from human speech. These are shown to be the result of the fundamental properties of the synthesis process, which would be difficult to eliminate without affecting other qualities of speech. Extraction of these features is then followed by a simple logistic regression classifier.

2.3 Interpretability Methods

We have been inspired by the rise of the field of "Explainable Artificial Intelligence" (XAI) which aims to demystify the various approaches of Machine Learning and Deep Learning, and allows the internal working of these models to be more transparent, providing easy-to-explain interpretations of their decisions to a human audience. The term XAI was popularized by DARPA's XAI program [27]. The roots of this term, however, can be traced to [28] and the pursuit of model interpretability dates even further back to the 90s with [29] using Saliency Maps for the interpretation of Neural Networks. There are significant benefits to investing in XAI, including business returns by means of satisfying investors and compliance with GDPR legislation, a part of which includes "Right to Explanation" currently in effect across the EU. More importantly, it caters to the fundamental social Responsibility of openness and ethical behaviour.

Our work would be incomplete without an extensive perusal of the landscape of XAI and the various methods of analyses of black box Deep Learning models to give us crisp visual representations superimposed on our input image. To achieve our goal of the Explainability of Deep-Fake detection, we continue with a survey of some XAI methods.

The introduction of saliency maps in K. Simonyan et al. [30] as an analysis tool piqued the interest of researchers to look more closely into their CNN. Selvaraju et al. [31] showcased Grad-CAM, a method to localize regions in the image that are important for its predicted class, with further applications in VQA (Visual Question Answering), which is recommended for interpretation of CNNs for image classification. Ribeiro et al. [32] introduced LIME, which preserves local fidelity to localize the interpretation of the model around the instance predicted. LIME has proved to be a versatile method to generate explanations from different kinds of Machine Learning models regardless of the model architecture since it doesn't need to glance into the model itself.

One of the most powerful and effective XAI methods out there is Layer-Wise Relevance Propagation (LRP) introduced by Bach et al. [33]. It evaluates a relevance score for every neuron by doing a backward pass in the deep neural network, thus elucidating why a specific decision was taken. LRP and its variants feature extensively in the results of this paper to plot visual heatmaps and highlight the salient features of the images.

Chapter 3
IMAGE DEEP-FAKES

The concept of manipulated and synthetic images is not new, although this task has required immense knowledge of graphics and availability of high-grade hardware to create forgeries of good quality. The era of deep-learning has completely changed the game, and there is no requirement of subject matter experts to create image deep-fakes anymore. Since the inaugural paper on GAN [7] in 2014, the quality of AI-generated images has improved exponentially. Collaborative research works are resulting into hyper-realistic synthetic images that are now extremely tough to distinguish from real images. The journey of GAN generated image deep-fakes can be observed in Figure 3.1. In the last two years we have reached to even greater heights in terms of the high quality of generated deep-fakes, which we will discuss in the rest of this chapter.

Figure 3.1: History of GAN based Image Deep-Fakes

In this chapter we investigate the generation of two kinds of image deep-fakes – Synthetic Faces, and Attribute Manipulation. We further explore the detection of synthetic deep-fakes by using a variety of deep learning architectures trained on a self-curated dataset. Throughout this chapter we focus on the concept of generalizability, as it is the main theme of our whole work, and towards this endeavour we generate data of multiple different distributions for our detection pipeline to work on.

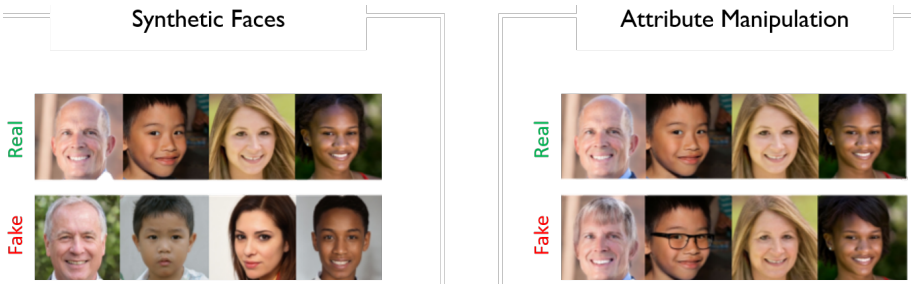


Figure 3.2: Types of Image Deep-Fakes

3.1 Image Deep-Fakes Generation

3.1.1 Synthetic Faces

This type of image deep-fakes refers to the generation of non-existent human faces using power GAN architectures like DCGAN, CoGAN, ProGAN, etc. The backbone of all these networks is the concept of Generative Adversarial Networks which was introduced by Ian Goodfellow and his colleagues in 2014. This was basically a game theoretic approach for generative modelling which utilized two structures of generator and discriminator. The training is incentivised in such a way that the generator tries to fool the discriminator, which the discriminator tries to correctly identify the samples generated by the generator. Once this architecture is trained, the generator can be taken out

and used for creating fake images of pristine quality. The GAN, therefore, captures the distribution of the data it is trained on and can subsequently be used to create new samples from the same distribution. An understanding of the basic GAN is essential to explore the more advanced architectures.

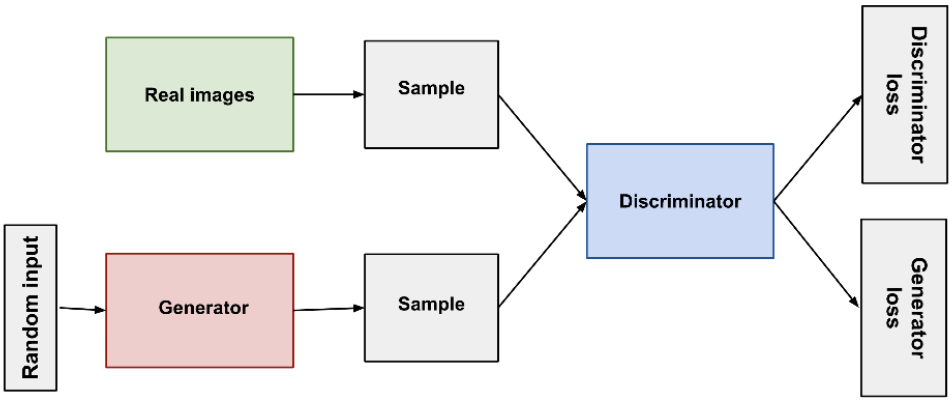


Figure 3.3: Basic GAN structure

Over the years, researchers have made significant improvements in the discriminator, the generator, the loss functions, and the overall training strategy so that the quality of generated synthetic faces is realistic. The latest among these is the StyleGAN [9] architecture created by NVIDIA in 2018, that introduces several ground-breaking changes in the basic structure of a GAN. This overcomes several limitations of the original GAN like the lack of control over the styles and their intensity at different levels of details in the images. We will discuss some of the fundamental modifications StyleGAN makes in the original GAN, and the effects of that in the generated images –

1. Progressive Growing of Images: StyleGAN is an extension of the progressive growing GAN (ProGAN) that is an approach for training generator models capable of synthesizing high-quality images using the incremental expansion of both the discriminator and generator models from a small resolution of 4x4 to a much higher resolution of 1024x1024. This is done by adding a new block to both the models to support the larger resolution after training the model on the smaller resolution.

2. Bilinear Sampling: The progressive growing GAN uses nearest neighbour upsampling in both the generator and the discriminator, instead of the transpose convolutional layers that are common in other generator models.

3. Mapping Network: A mapping network is used which takes a randomly sampled point from the latent space (z) as input and generates a style vector(ω). This style vector is special because its different elements control different visual features. This results into more disentanglement of features, which gives the ability of more fine-tuned generation of images by the generator. It is an eight layered MLP network. The style vector is transformed and incorporated into each block of the generator model after the convolutional layers via adaptive instance normalization or AdaIN.

4. Addition of Noise: Every convolutional layer in the generator is augmented by gaussian noise before the AdaIN operations. A different sample of noise is generator for every block and is controlled using the style scaling factors. The noise helps in introducing stochastic variation in features at different levels of details. This results in features like freckles, placement of hairs, wrinkles, etc to be more distinct among generated images.

5. Truncation Trick: In order to make sure the generated images follow a definite template, e.g. the human face, before the latent vector (ω) is fed into the generator, it is offset by an average latent vector (ϖ). The style scale (Ψ) determines the strength at which different styles are applied with respect to the "average face". Therefore, the new latent vector (ω') is defined in terms of ω, ϖ, and Ψ: $\omega' = \varpi + \Psi\ (\omega - \varpi)$. The effect of different style scales in the generated images can be seen in Figure 3.4. When Ψ is 0 we get the average face. As Ψ is increased, we see the faces gradually getting more distinct in terms of various styles. The style factor therefore is a key parameter that describes the distribution of generated faces.

StyleGAN2 [10] was introduced by NVIDIA in 2019, which improved the quality of generated images further with reduction in artifacts like blobs, water droplets and irregular background features. This was achieved by additions like weight demodulation, lazy regularization, path length regularization, and a different generator and discriminator structure that doesn't use progressive growing. This architecture achieves much better results than its predecessor.

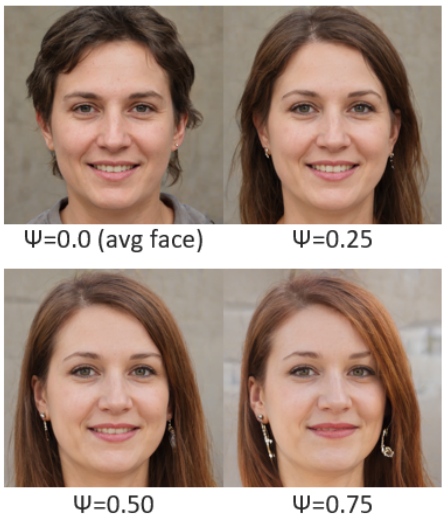


Figure 3.4: Generated Images for different Ψ

After training it on FFHQ dataset, which contains 70,000 real facial images, some of the results obtained are shown in Figure 3.5. We used this StyleGAN2 architecture as the reference model for all the results in the rest of the chapter as well.

Figure 3.5: Results of training StyleGAN2 on FFHQ Dataset

3.1.2 Latent Space Exploration

StyleGAN2 effectively learns the mapping from the high dimensional pixel space to its low dimensional latent space. After it has formed a mapping, we can sample the latent space to feed these latent vectors to the GAN and get faces in the pixel space. We can traverse the latent space from every point that represents a face, to get more faces. This can result in concepts like face arithmetic, attribute manipulation and interpolation between faces.

1. Arithmetic of Faces: Since faces are essentially vectors in the latent space of the GAN, we can apply vector arithmetic to their latent vectors and get some intuitive results. For example, we can start off with subtracting a neutral woman from a smiling woman to isolate the feature of smile. Now we can add this smile feature to a neutral man, and get a resultant smiling man. This is exhibited in Figure 3.6.

Figure 3.6: Face Arithmetic

2. Attribute Manipulation: Since StyleGAN2 provides relatively good feature disentanglement, we can find directions in the latent space of the GAN which affect isolated features in the generated image. This gives us a tool to modify only our desired parts of the faces, and this is in fact another type of image deep-fake. In order to find these directions, we can use classifiers like SVM which find a hyper plane in the latent space of the GAN which acts as a dichotomy for any two classes. If we move orthogonally to these hyperplanes, we will effectively morph the faces from one class to another gradually. An example is given in Figure 3.7.

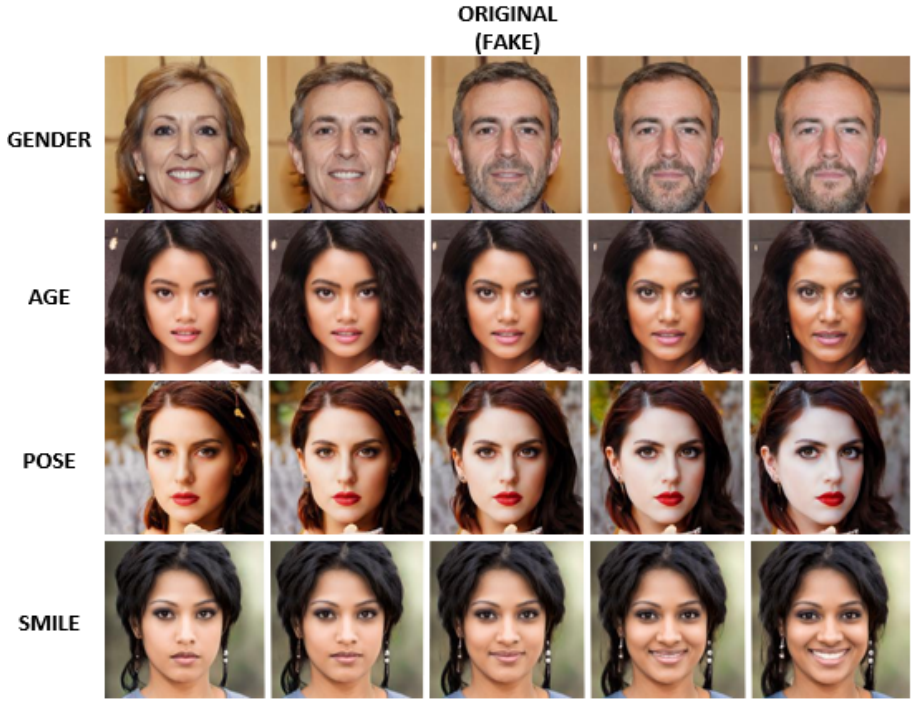


Figure 3.7: Attribute Manipulation

3. Face Interpolation: Since we have a continuous, disentangled space, we can explore the latent space between two faces, which are represented by points in the GAN domain. This is mathematically interpolation and we can use techniques like linear interpolation, quadratic interpolation, cubic interpolation etc to get the intermediate faces between two extreme faces. This is exhibited in Figure 3.8.

Figure 3.8: Face Interpolation

3.2 Image Deep-Fakes Detection

3.2.1 Dataset Compilation

We frame the task of image deep-fakes detection as a binary classification task, which can be learnt by training deep learning architectures on a dataset containing both fake and real faces. We choose the synthetic faces type of image deep-fakes as our training data. All fake images in the dataset are generated using the StyleGAN2 architecture, and the real images are sourced from the FFHQ dataset. Since our aim is generalized deep-fakes detection, we produce different distributions of fake images by varying the style factor (Ψ), which has been discussed before.

According to the standard practice in deep learning, we create train, val and test sets for our dataset. The train and val sets contain fake images generated with Ψ belonging to [0.5,0.6,0.7,0.8,0.9], while the test set contains fake images with Ψ belonging to [0.55,0.65,0.75,0.85,0.95]. This is done to ensure there is no dataset leakage and the test set is of a different distribution than the train and val set. A performance on such a dataset will help us judge the generalizability of our architectures for the task of image deep-fake detection. The detailed image distribution of train, val and test sets is given in Table 3.1. Each set contains equal number of real and fake images, and equal distribution of every Ψ in fake images. We also create two versions of this dataset, one each for sizes 64 and 128, to observe the effect of increasing size of input images on detection performance.

value of Ψ	0.5	0.6	0.7	0.8	0.9	total
train set	1500	1500	1500	1500	1500	7500
val set	500	500	500	500	500	2500
value of Ψ	0.55	0.65	0.75	0.85	0.95	total
test set	400	400	400	400	400	2000

Table 3.1: Distribution of train, val and test sets

3.2.2 Deep Learning Architecture Families

For the training phase, we need to choose deep learning architectures for image deep-fake detection. Here we conduct an exhaustive survey of three deep learning architecture families and their constituent models, in order of increasing parameters, and observe their performances. We chose these models so that corresponding architectures in the different families have roughly the same parameters so getting a better result would not just be a matter of having more parameters to fit the distributions. The three families are –

1. ResNet Family: The core idea of ResNets is introducing a so-called "identity shortcut connection" that skips one or more layers in the network. One of the problems ResNets solve is the famous known vanishing gradient. This is because when the network is too deep, the gradients from where the loss function is calculated easily shrink to zero after several applications of the chain rule. This result on the weights never updating its values and therefore, no learning is being performed. With ResNets, the gradients can flow directly through the skip connections backwards from later layers to initial filters. The variants we consider in our training are – resnet18, resnet34, resnet50, resnet101, and resnet152.

2. VGGNet Family: While previous derivatives of AlexNet focused on smaller window sizes and strides in the first convolutional layer, VGG addressed the depth of the networks. Instead of using large receptive fields like AlexNet (11x11 with a stride of 4), VGG uses very small receptive fields (3x3 with a stride of 1). Because there are now three ReLU units instead of just one, the decision function is more discriminative. The small-size convolution filters allows VGG to have a large number of weight layers; of course, more layers leads to improved performance. The variants we consider in VGGNet family are vgg11, vgg13, vgg16, and vgg19.

3. DenseNet Family: Dense Convolutional Network (DenseNet) connects each layer to every other layer in a feed-forward fashion. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. The variants we consider are densenet121, densenet169, densenet201, and densenet161.

3.2.3 Results

We get the results after training all thirteen models across the three architecture families on our compiled dataset for 20 epochs each with a learning rate of 1e-3 using ADAM optimizer. The results are tabulated in Tables 3.2, 3.3 and 3.4.

model	parameters	test accuracy(%) for size 64						test accuracy(%) for size 128					
		0.55	0.65	0.75	0.85	0.95	**total**	0.55	0.65	0.75	0.85	0.95	**total**
resnet18	11,705,410	87.5	82.3	71.8	64.0	56.4	**72.4**	89.9	87.1	84.5	71.8	63.9	**79.4**
resnet34	21,813,570	90.1	87.1	75.4	63.5	56.3	**74.5**	90.1	89.3	84.8	75.4	64.6	**80.8**
resnet50	25,615,938	90.1	86.9	78.3	68.3	59.9	**76.7**	92.8	91.9	87.3	75.4	67.0	**82.9**
resnet101	44,608,066	91.0	88.8	81.4	69.9	59.5	**78.1**	91.0	89.4	86.4	77.9	70.9	**83.1**
resnet152	60,251,714	92.1	90.1	82.3	71.0	61.4	**79.4**	93.5	92.6	89.5	79.9	69.5	**85.0**

Table 3.2: ResNet Family

model	parameters	test accuracy(%) for size 64						test accuracy(%) for size 128					
		0.55	0.65	0.75	0.85	0.95	**total**	0.55	0.65	0.75	0.85	0.95	**total**
vgg11	9,754,882	89.0	87.0	77.8	69.0	57.8	**76.1**	92.8	90.9	87.3	79.9	67.6	**83.7**
vgg13	9,939,778	90.3	89.1	79.5	66.6	58.0	**76.7**	95.1	93.1	87.4	78.5	66.9	**84.2**
vgg16	15,252,034	91.9	87.8	76.0	69.5	60.3	**77.1**	94.1	94.0	88.4	77.4	68.0	**84.4**
vgg19	20,564,290	90.8	87.6	79.8	69.4	58.8	**77.3**	96.3	93.8	88.0	78.1	67.0	**84.6**

Table 3.3: VGGNet Family

model	parameters	test accuracy(%) for size 64						test accuracy(%) for size 128					
		0.55	0.65	0.75	0.85	0.95	**total**	0.55	0.65	0.75	0.85	0.95	**total**
densenet121	8,009,090	89.4	89.0	77.5	67.9	60.1	**76.8**	91.0	89.8	87.0	81.1	70.0	**83.8**
densenet169	14,197,634	91.5	87.9	81.1	71.0	62.0	**78.7**	92.9	92.5	90.1	80.9	66.8	**84.6**
densenet201	20,069,250	92.4	91.4	83.0	73.0	60.4	**80.0**	94.0	92.5	88.4	83.9	71.5	**86.1**
densenet161	28,744,386	93.1	90.3	84.6	76.8	63.8	**81.7**	96.4	95.9	89.6	81.3	71.0	**86.8**

Table 3.4: DenseNet Family

The general trend that is followed is that DenseNet family is doing the best among all three families, followed by VGGNet family, and finally the ResNet family. As expected size 128 is doing much better than size 64 for all models. Densenet161 gives the best test performance of 86.8%, while the best performing models in ResNet family and VGGNet family are resnet152 with 85%, and vgg19 with 84.6%.

Chapter 4
VIDEO DEEP-FAKES GENERATION

For the ultimate aim of generalised Deep-Fake Detection, it is necessary to have Deep-Fakes from as many distributions as possible. In this section, an approach towards generating Deep-Fakes for the purpose of creation of the dataset is explored.

It is essential to have a relevant, cleaned dataset to train the model in the most optimum way possible and generate the desired results. Thus, Deep-Fake generation became a crucial aspect of the project contributing to the dataset and also giving an insight on the parameters concerned with the generation of Deep-Fakes which could later help in its detection.

In order to create high-quality Deep-Fakes, we made use of the publicly available repository FaceSwap-GAN [34]. It provides a denoising autoencoder and adversarial loss and attention mechanism for face swapping.
The generation of Deep-Fakes was performed using a 3-stage pipeline:

4.1 Collecting faces of real people

Firstly, videos of a few famous personalities were collected from:

- Cricket Post-Match Press Conferences
- Saturday Night Live Celebrity Clips
- Self-prepared clips of Students

The Dlib machine learning library was used to extract faces from frames in all the videos, and a dataset of multiple personalities was created, each having 500-1000 face images. All images were aligned in accordance with reference facial landmarks, and binary masks of eyes were created to aid generation.

Pre-processing involved detection, extraction and alignment of the desired faces. Following methods of detection and extraction were used:

- Using dlib as face detection and extraction framework
  - Results didn't align with the desired performance
- Face Extraction using Scene Detection
  - An automated tool to overcome the manual requirement of filtering the dataset for frames was developed
  - Using the frameworks of PyScenedetect and Face Tracking using OpenCV, filtering dataset was semi-automated

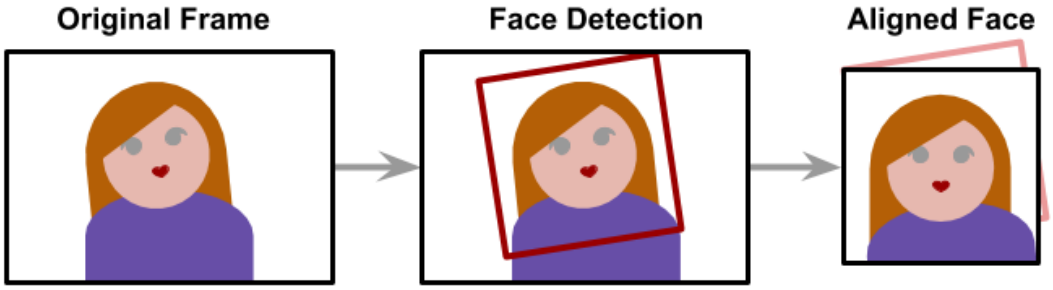


Figure 4.1: Face Alignment

4.2 Training the autoencoders

FaceSwap-GAN follows a generator-discriminator architecture to train and generate Deep-Fakes. The generator is an Convolutional Autoencoder, while the Discriminator is a PatchGAN based discriminator model.

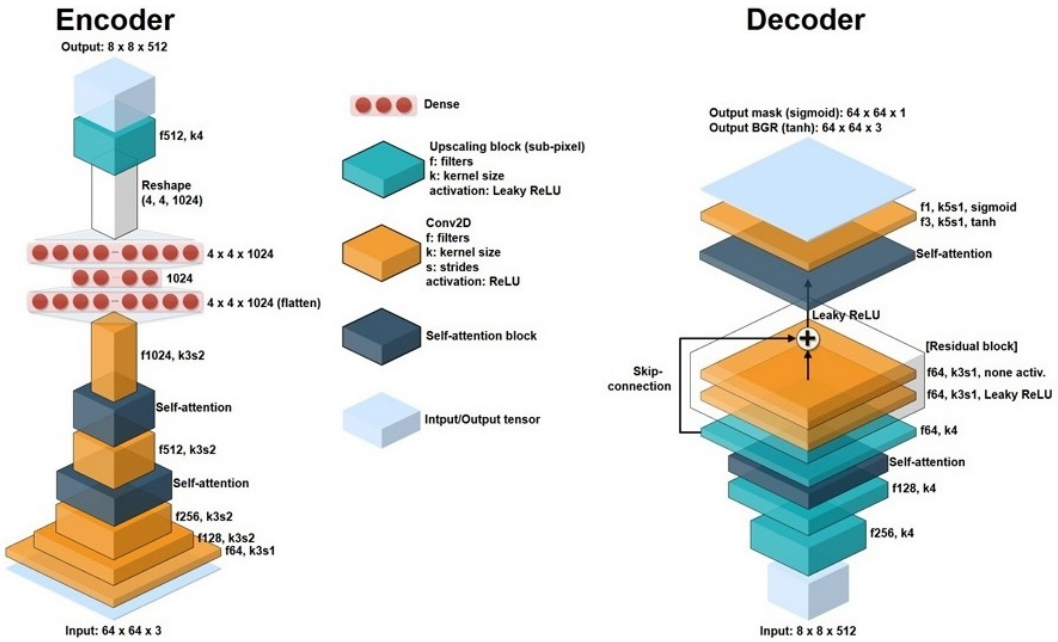


Figure 4.2: FaceSwap-GAN encoder and decoder architecture

4.2.1 Autoencoder Architecture

Encoder: The encoder uses 5 convolutional filters with two self-attention blocks. The self-attention blocks are crucial to model correlation and interaction with the neighbouring elements and their features extracted by multiple attention heads. The encoder reduces an image (in this context, the frame extracted from the frame of the video) to a lower dimensional latent space. It represents the key features of their facial features.
Decoder: The decoder reconstructs the image from the latent representation provided from the encoder. The model to be used for decoding is specifically trained for the target. This means the target's detailed information will be superimposed on the underlying facial and body features of the original video, represented in the latent space.

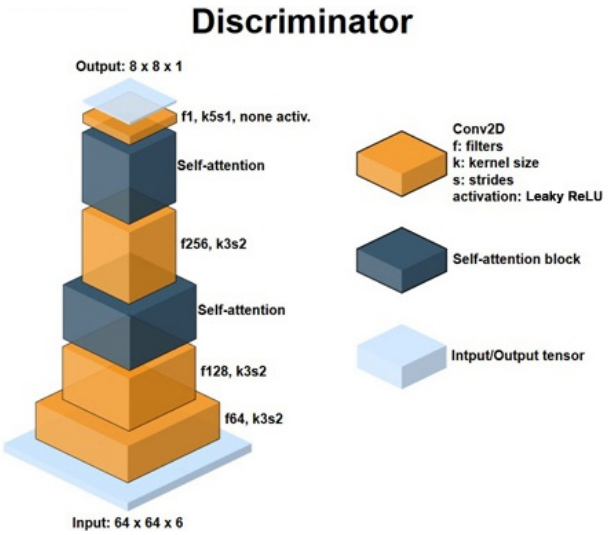


Figure 4.3: Discriminator Architecture

4.2.2 Discriminator Architecture

The discriminator consists of 4 convolutional layers. It accepts a 64*64 RGB image. The probability represents the discriminator's belief that the input image is a real face. During training, the discriminator is updated using real images from the training set, and images generated from the autoencoder. The discriminator is rewarded for assigning a low probability to generate faces, and a high probability to real faces.

In the training phase, traditional Convolutional Autoencoders are trained using individual person datasets to create realistic looking faces. They are then fed to a discriminator which is also trained to identify real faces against those generated from autoencoders.

4.2.3 Training Process

Binary Masks were prepared capturing the target face details of each subject. These masks model the image pixels as belonging to the region of interest. The training starts at the autoencoder, which swaps the masks to give swapped faces. The aim of the discriminator is to differentiate between the real (original face) and fake images (swapped face)

After multiple passes through the dataset up to an confident estimate that the generator is able to generate fake images that the discriminator fails to differentiate, the training is stopped. Using the masks, the swapped faces are now merged onto each frame during video conversion to obtain the final images

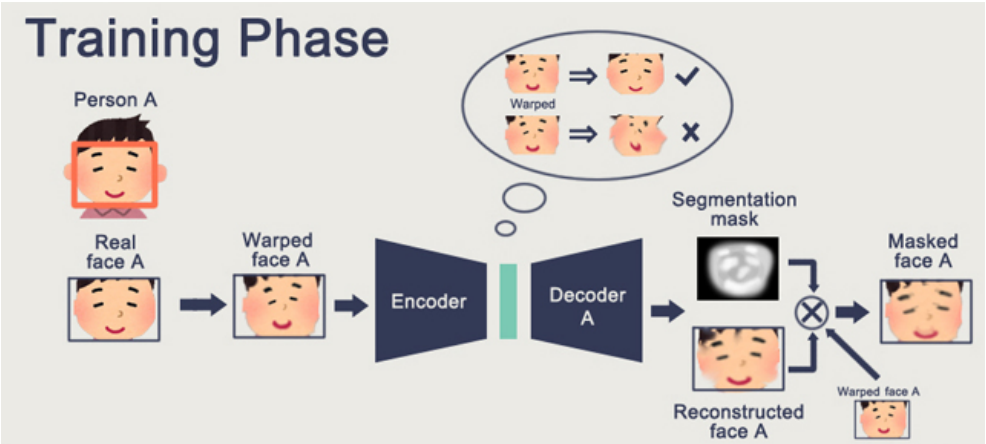


Figure 4.4: Training Phase

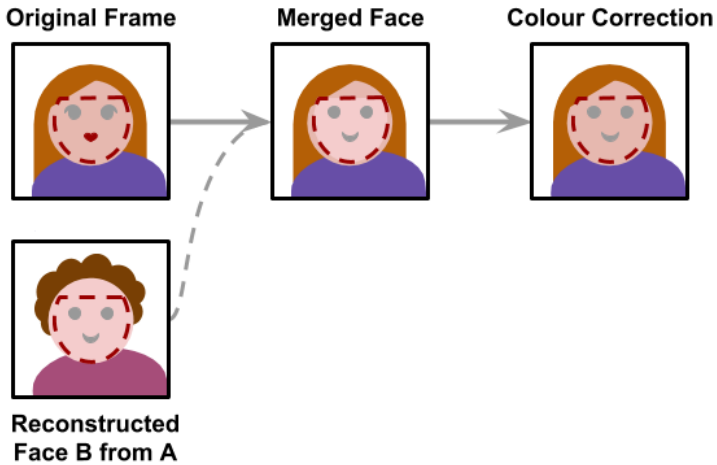


Figure 4.5: Video Conversion

4.3 Creating Videos using trained autoencoders

During Video Conversion, all frames are extracted from the input video. The extracted faces are aligned and then fed to the training model with each frame processed independently.

The features obtained after training are then superposed on the binary masks of the extracted faces of the target video.

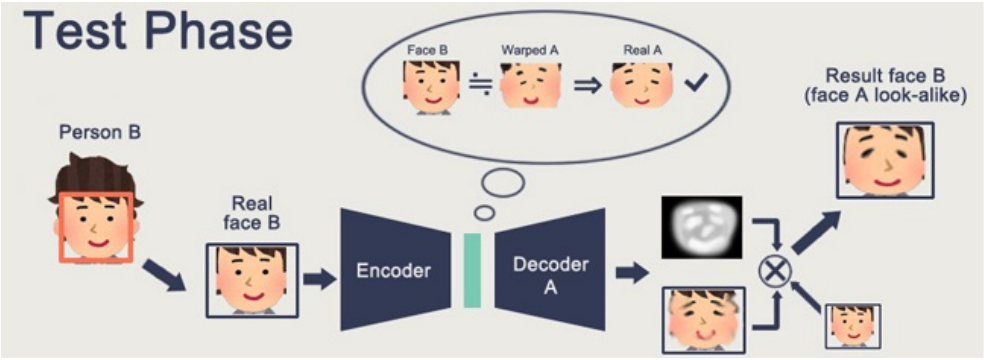


Figure 4.6: Testing Phase

4.4 Deep-Fake Generation Results

For generation of Deep-Fakes, pairs of personalities were created and their decoders were swapped, creating faceswaps of each other. We obtained 20 good quality Deep-Faked videos from this experiment, which we will be using in our custom test set explained in the next chapter.

Figure 4.7 showcases the general idea of faceswapping decoders of a pair. This template is used to show seven examples of Deep-Fakes generated by us in Figure 4.8. The captions in the figure follow the syntax: 'Person A and Person B'. The personalities are:

(a) Cricketers Dinesh Karthik and Ajinkya Rahane

(b) Singers Adam Levine and Justin Timberlake

(c) Classmates Ankit and Vaibhav

(d) Cricketers Yuvraj Singh and Rohit Sharma

(e) Classmates Adithya and Badhri

(f) Cricketers Rahul Dravid and Jaydev Unadkat

(g) Classmates Saksham and Sarvesh

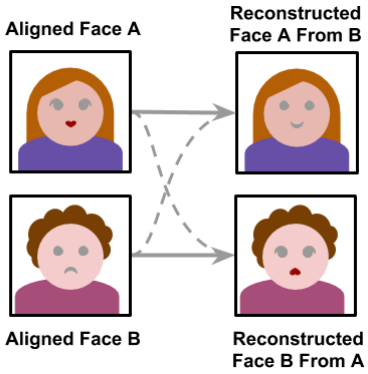
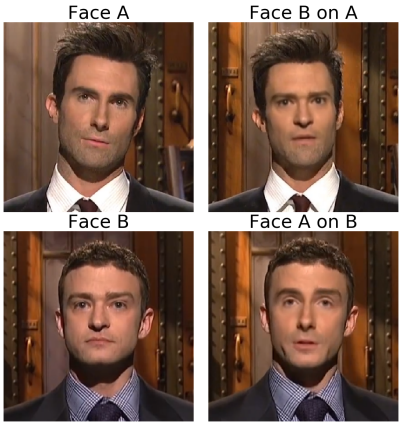
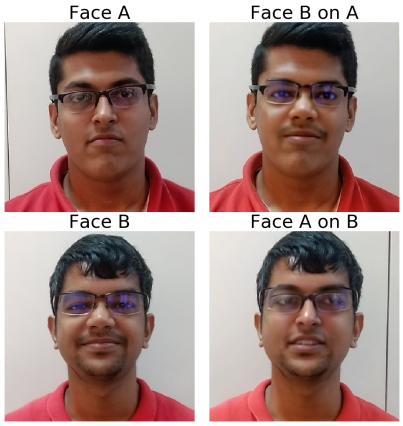


Figure 4.7: Faceswapping A and B

(a) Dinesh Karthik and Ajinkya Rahane

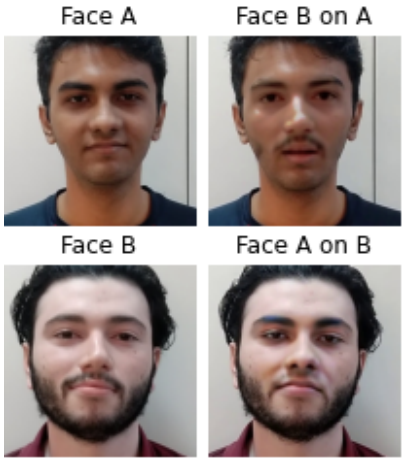


(b) Adam Levine and Justin Timberlake



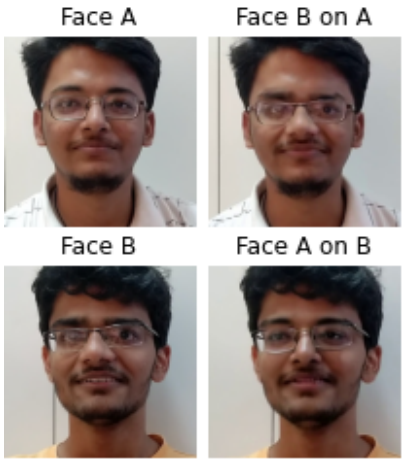
(c) Ankit and Vaibhav

(d) Yuvraj Singh and Rohit Sharma



(e) Adithya and Badhrinarayan

(f) Rahul Dravid and Jaydev Unadkat



(g) Saksham and Sarvesh

Figure 4.8: Deep-Fake Generation Results

Chapter 5
VIDEO DEEP-FAKES DATASETS

We frame the task of Deep-Fake detection as a binary classification task, the two classes being real and fake faces. Our approach is themed on supervised learning where we train a Convolutional Neural Network on sets of real and fake samples to estimate the probability of test images being affected by Deep-Fake manipulations. This section describes the various Deep-Fake video datasets used for this approach.

5.1 Deep-Fake Datasets

This work utilizes three of the latest Deep-Fake video datasets that were recently released publicly –

1. DeepFakeDetection (DFD) – Google in collaboration with Jigsaw released the DeepFakeDetection dataset, which has been incorporated into the Technical University of Munich and the University Federico II of Naples' new FaceForensics benchmark [22]. They hired 28 actors and recorded 363 original sequences of them speaking, making common expressions, and performing simple tasks. From these sequences, they used publicly available Deep-Fake algorithms to produce 3068 manipulated videos of high visual quality. The manipulations are not flawless, most of which are distinguishable by the naked eye from untouched videos.

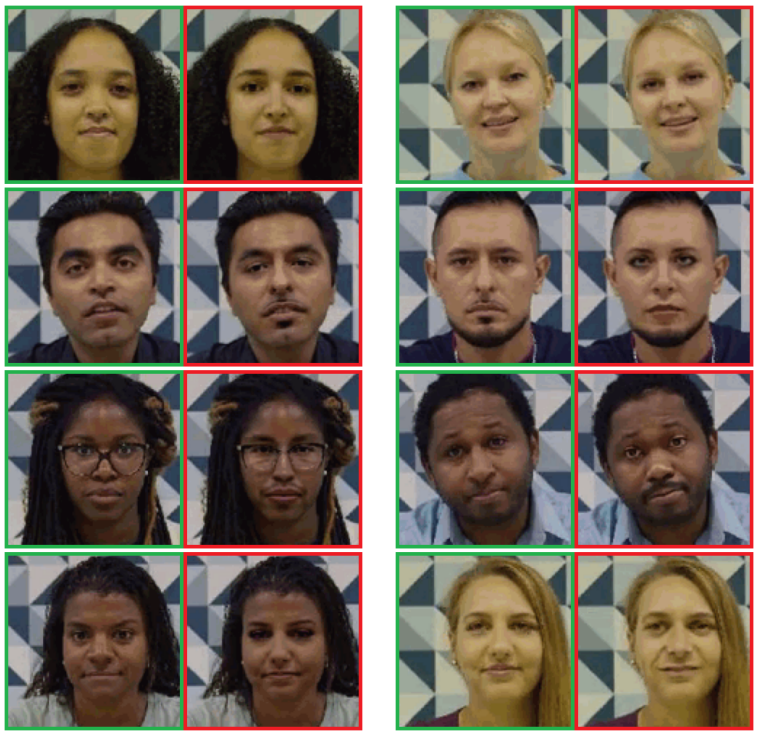


Figure 5.1: Samples from DFD: Green boxes are original, Red boxes are corresponding Deep-Faked images

2. Celeb-DF [35] – In an effort to improve quality of Deep-Fake datasets, this work used 408 real videos sourced from YouTube and synthesized 795 videos with improvements in the usual Deep-Fake generation model. The real videos are acquired from YouTube, and split in two sets. The first set consists of 158 videos of 13 celebrities with different gender and skin colour. The second set includes 250 videos, where each video contains a different subject. The synthesized videos are generated using the videos from the first set using a refined Deep-Fake generation algorithm. The visual quality is low but the quality of face-swaps is quite realistic. The creators of the dataset have also provided a train-validation split themselves, by reserving a subset of synthesized videos and their corresponding real videos for validation and allotting the rest for training. The various refinements in the Deep-Fake generation algorithm employed in this dataset are as follows:

   - Low resolution of synthesized faces: Older Deep-Fake algorithms generate faces of size 64x64 pixels, which required resizing to fit the various sizes of the faces of the target. This causes visual artifacts such as

smoothened skin and lack of skin details. For this dataset, the auto-encoder structure is improved by adding extra convolutional layers to the decoder, such that the new method is capable of generating faces of size 256x256 pixels.

- Color inconsistency: The disparity in colour and illumination between the donors' and targets' faces give rise to colour mismatch between original faces and synthesized faces. To alleviate this visual artifact, an augmentation procedure was performed on the training faces, randomly changing their brightness, colour distortion, contrast and sharpness. This increases the diversity of the training data and reduces the problem of colour mismatch.
- Visible facial parts of the original face: Previous algorithms tend to use a rigid mask as a bounding box for facial landmarks. This approach cannot adopt to targets of different facial structures. For this dataset, the initial mask is enlarged by a factor and then interpolated to get a non-rectangular smooth mask which fits better on the target faces.
- Temporal flickering: In a real video, the faces have consistent temporal differences. However, the synthesized videos often have inaccurate facial landmark locations, colour mismatching artifacts, etc. introducing inconsistent temporal changes in the form of flickering. In this dataset, as those components have been improved, the flickering has been greatly reduced.

Figure 5.2: Samples from Celeb-DF: Green boxes are original, Red boxes are corresponding Deep-Faked images

3. DeepFakeDetectionChallenge (DFDC) Preview [36] – The latest entrant in the category of Deep-Fake datasets has been compiled by Facebook AI. A set of 66 paid actors were chosen from a pool of crowdsourced individuals keeping in mind to capture the variability in gender, age and skin tone. Each actor recorded a set of videos in which they answered a set of pre-specified questions. The videos included various lighting conditions and head poses, and since the participants were allowed to have any background there was great visual diversity in the background. An approximate distribution of gender and race is shown in the following tables:

Gender	Distribution
Male	26%
Female	74%

(a)

Race	Distribution
Caucasian	68%
African-American	20%
East-Asian	9%
South-Asian	3%

(b)

Table 5.1: Distribution of (a) Gender and (b) Race in DFDC Preview dataset

The 66 actors were split into training and testing sets, which was done to avoid cross-set face swaps. Two methods were employed to generate face swaps with the intention of representing the real adversarial space of facial manipulation. The first method (named method A in the dataset) does not produce believable face swaps when the face of the individual is too close to the camera. The second method, method B generally produces lower-quality face swaps, and is representative of other face swap algorithms.

The first five seconds of the videos were cut as they were mostly subjects setting up their cameras. From the remaining period, multiple 15 second clips were extracted as data. The videos in the training set were un-augmented. However, in the test set, two clips out of three were subject to one and only one of the following augmentations:

- Reduce FPS of the video to 15
- Reduce resolution of the video by a factor of 1/4
- Reduce overall encoding quality

Overall, the dataset consists of 4,464 training videos and 780 testing videos. This dataset has a high visual quality as well as high manipulation quality.

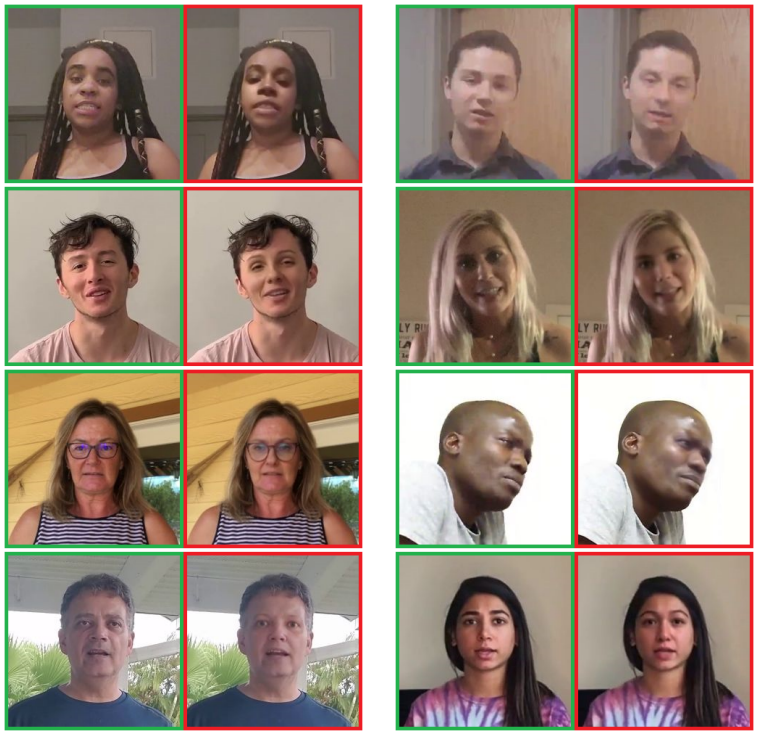


Figure 5.3: Samples from DFDC Preview: Green boxes are original, Red boxes are corresponding Deep-Faked images

The datasets can be broadly categorized based on their visual quality and their manipulation quality. The illustration 5.4 shows this categorization for the above three datasets.

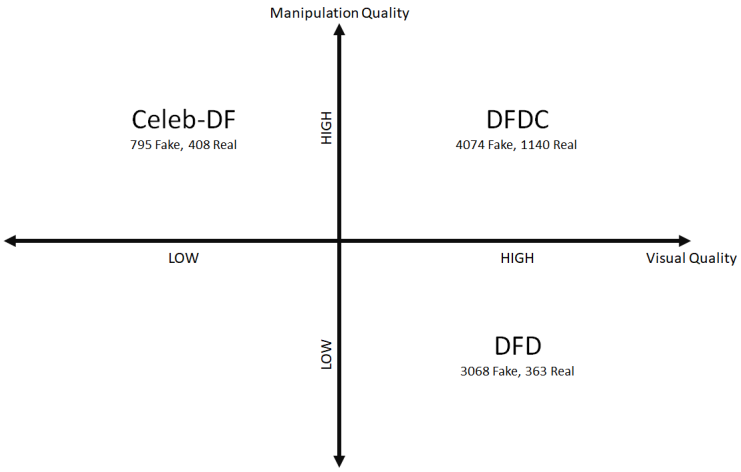


Figure 5.4: BCG Matrix for the visual and manipulation qualities of the three datasets

These three datasets present different distributions (in terms of visual and manipulation quality) for training, therefore to test generalizability of models trained separately on each dataset it is necessary to evaluate performance on a neutral dataset that is compiled with no particular bias towards any one distribution. For this purpose, we present our custom collection of Deep-Fakes only for the purpose of testing –

4. Custom Test Set – The first part of this dataset is comprised of Deep-Fakes generated by us, by utilizing carefully shot videos in similar conditions resulting in high quality manipulations. The second part includes some famous and indistinguishable Deep-Fakes uploaded on YouTube. These form a good representation of the type of manipulations likely to be released in the future for potentially malicious purposes. Therefore, testing on these videos allow us to analyse the ability of deep-learning models to generalize outside their train/test distribution for the purpose of Deep-Fake detection. There are 40 selfmade videos and 60 videos sourced from YouTube, both containing equal number of real and fake videos.

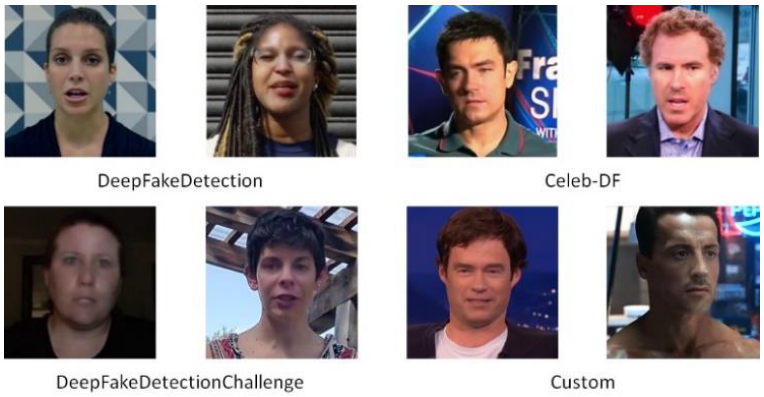


Figure 5.5: Deep-Fake samples from the different datasets

The first two video datasets are each split into train, validation and test sets in the ratio of 70:15:15 for both real and fake videos, while the DFDC Preview videos are segregated according to the suggested split. Combined train and test sets are also obtained by putting together splits of all three categories, additionally the custom set is also included as part of the combined test set. Each set is roughly comprising of equal proportion of real and fake videos.

Chapter 6
VIDEO DEEP-FAKES DETECTION

6.1 Preliminaries

6.1.1 Transfer Learning

Transfer Learning (TL) [37] is a technique which emphasizes on storing knowledge gained during learning a source task T_S, and utilizing that knowledge to better learn a different target task T_T. The extent to which it is useful depends on the similarity between the tasks and the domains on which they are performed. In Deep Learning, this is achieved by using the weights of a model trained for a source task to initialize another model set to be trained for the target task.

For this work we initialize our Xception models with ImageNet pretrained weights. ImageNet dataset contains a total of 14 million images and more than 20,000 sub-classes out which 2035 are in the category of "person", and over 4000 in "animal" and "bird" sections. It is expected that a model pretrained with ImageNet weights should be able to extract a lot of relevant features through convolutional layers that would also be beneficial for Deep-Fake detection. This intuition can be supported by Figure 6.1 where we visualize the activations [38] of the first convolutional layer of the pretrained XceptionNet when it is given a face as input. Since we are satisfied with the features being extracted, we freeze the convolutional layers and only tune the weights of the fully connected layers for the initial phase of training. This results in a significant boost in validation accuracy even after just one epoch as compared to an Xception model without Transfer Learning (Figure 6.2). After we reach the saturation point of the validation accuracy, we unfreeze the initial layers and train the whole model with a significantly lower learning rate. This whole process is known as "fine-tuning", and benefits performance by learning target task in a much more generalized manner. This results in a noticeable difference in the final test accuracies of all trained models as well, which can be seen in Table 6.2, and suggests that Transfer Learning helps in reducing overfitting.

Figure 6.1: Intermediate Activations of Xception pretrained on ImageNet

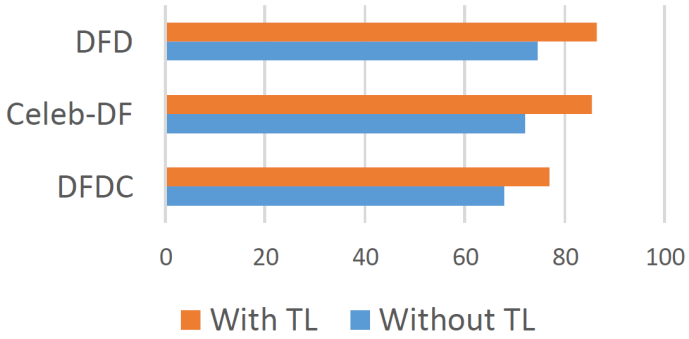
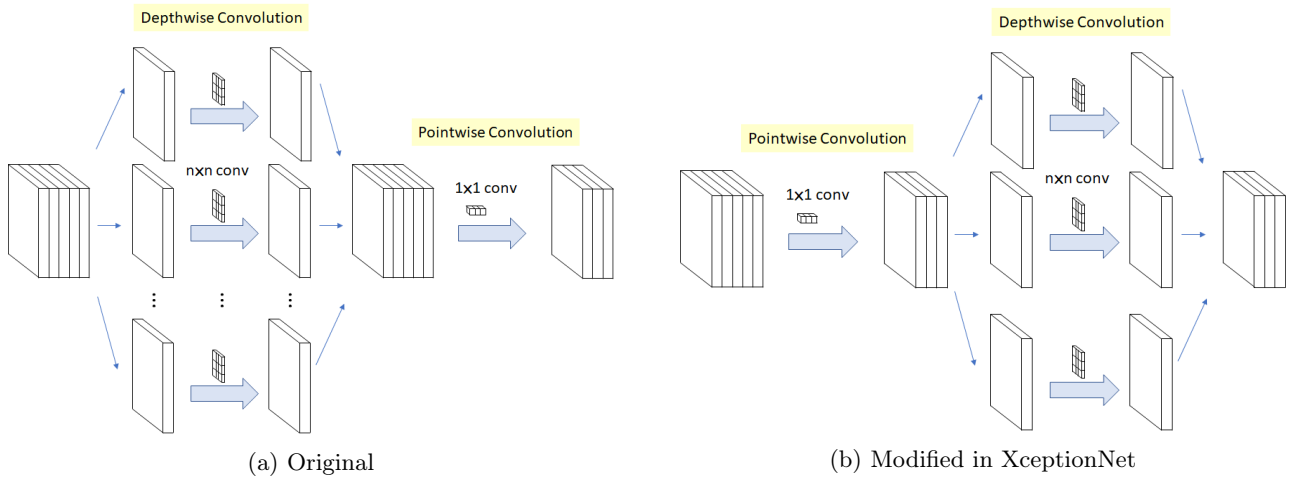


Figure 6.2: Validation accuracies (%) of models trained on individual datasets after one epoch

6.1.2 XceptionNet



(a) Original (b) Modified in XceptionNet

Figure 6.3: Depthwise Separable Convolution

In this report, the base model for all experimentation is the XceptionNet [23]. It features modified depthwise separable convolution layers, which help it achieve better accuracies than Inception-v3 on ImageNet ILSVRC and JFT datasets.

In Depthwise convolution, each channel is convolved with a seperate filter. In Pointwise convolution, a 1x1 convolution is applied to change the number of output channels. Depthwise Seperable Convolution does not require convolution across all channels like conventional convolution. This results in fewer connections and a smaller model.

In the original depthwise separable convolution, a depthwise convolution is followed by a pointwise convolution. Meanwhile, in the modified depthwise seperable convolution, the pointwise convolution is followed by the depthwise convolution.

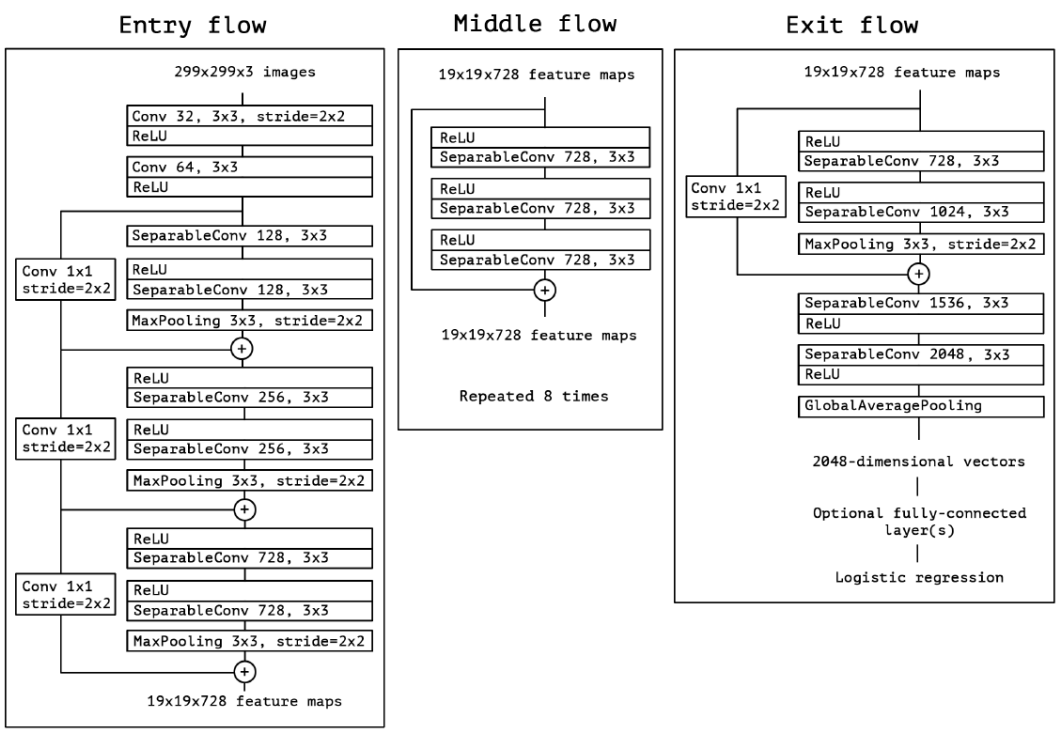


Figure 6.4: XceptionNet Architecture

6.1.3 Domain Adaptation and Dataset Shift

Most machine learning models assume that both training and test sets come from the same distribution D, which is the case of single-domain learning. The extension of this could be when the training set is drawn from a source distribution D_S while the test set is drawn from a different target distribution D_T. This is the more general case of multi-domain learning or Domain Adaptation [39], where most machine learning techniques find it difficult to give satisfactory performances as compared to single-domain learning. This problem of lack of generalizability is known as dataset shift [40] and is encountered often in tasks that have a possibility of varied and unseen distributions of test data. This can be caused by either a sample selection bias, or non-stationary environments.

The task of Deep-Fake Detection is one such example, where the generation process could introduce any number of variations in both the visual and manipulation quality. This is an example of a non-stationary environment for generation of data. Therefore, it is crucial to analyse the shortcomings that could result from using a single-domain modelling system. As we have already observed, all three datasets used in our work are distinct from each other and can be considered to be samples of different distributions. We hypothesize that a detection model trained on any one such dataset will get biased towards the experimental setup of the Deep-Fake generation process used by the compilers of that dataset, and give poor results on test data of any other distribution that is significantly different from the source distribution. To test this hypothesis, we use a cross-set evaluation process which gives us an indication of how different the three datasets are from each other and how generalizability can be improved. With every cross-set test result, we also vary the usage of Transfer Learning to show its effect on the classification accuracy.

The intuition behind this is that Transfer Learning, which utilizes its knowledge gained in a source task to perform a different target task, can be thought of as a complementary concept to Domain Adaptation, which employs knowledge of a task being performed in a source domain to perform the same task in a different target domain. The results will thus reflect the impact of these two techniques in the overall generalizability of Deep-Fake Detection.

6.2 Single-Frame Deep-Fakes Detection

6.2.1 Compilation of Image Datasets

We use the four datasets described in 5.1. For each video dataset, we segregate all videos into their respective splits, we extracted frames from these videos in order to create image corpuses for training the CNN. Since in all the three datasets the real videos are much lesser than fake videos, the rate of frame extraction is accordingly set to balance the number of images of both classes in the final dataset.

As all manipulation techniques considered in this work target only the facial region, we extract only faces present in each frame instead of full frames by using Dlib [41] face detection. An important parameter to be decided at this stage was the threshold of size of detected faces below which they should be discarded. This is because all the images fed into the CNN should be of the same dimensions and it was observed that enlarging faces produced many artifacts and distortions that could result into misclassification of real images. After experimenting with differently sized square images, we finalized the threshold to 128. All extracted faces were hence resized and stacked to form image datasets for all the categories.

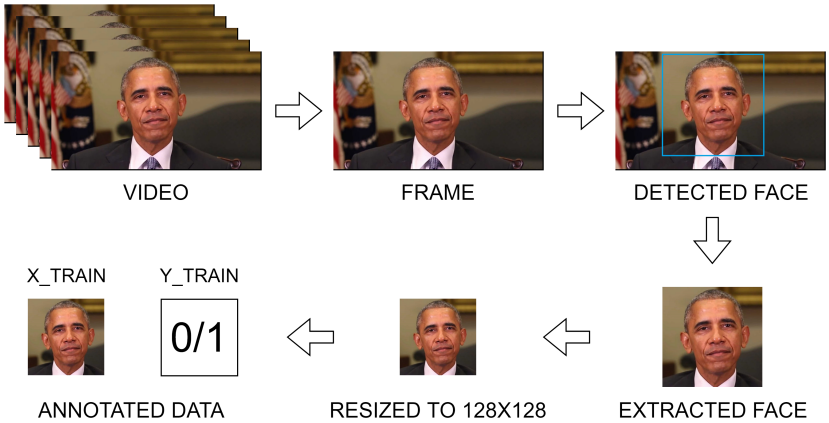


Figure 6.5: Image dataset curation pipeline

The final statistics for each set are displayed in Table 6.1.

Dataset	Train	Val	Test
DFDC	187080	36298	36302
Celeb-DF	173517	36629	37785
DFD	155019	32889	33187
Custom	-	-	32502
Combined	515616	105816	139776

Table 6.1: Distribution of all image datasets

6.2.2 Experiment

Training

We choose the XceptionNet [23] as our Convolutional Neural Network architecture, based on the observations of [22], [35], [36] and introduce no changes in the network. For training we use the Cyclical Learning Rate [42] to ensure uniformity in choosing learning rates for different datasets. The Adam optimization algorithm [43], cross-entropy loss function and mini-batch size of 512 images are employed.

For the rest of this discussion we refer to the various datasets as distributions, which are four in number – DFD, Celeb-DF, DFDC and combined (custom test set is also included in its test split).

We train a model for every distribution using their respective train and validation splits. For the case of without Transfer Learning, the models were initialized using He et al. initialization [44] and trained for 30 epochs. The transfer learning aided models were initialized with ImageNet weights before training with convolutional layers frozen for first 15 epochs. We unfreeze these layers for the next 15 epochs, and save the model with the best validation accuracy. At the end of this phase, we have four models (one for each distribution) each for with and without Transfer Learning ready for the testing phase.

Testing

The testing phase comprises of two parts – self-testing, and cross-testing. The self-testing is done for all models with their corresponding test split. For the cross-testing part, model of every distribution is evaluated on the test set of every other distribution and also separately on the custom test set. This is done for both with and without transfer learning models.

Note that the combined test set performance for models trained on individual distribution train splits is the same as the weighted (by number of samples of each set given in Table I on which evaluation is done) average of their self and cross-set test accuracies. Therefore, it provides a good measure of the cumulative performance of these models on all available test splits. The results are compiled in Table 6.2.

6.2.3 Results

There are a lot of inferences to be made from Table 6.2. The heavy impact of Transfer Learning can be seen invariably throughout the table, with the exception of only two blocks having higher accuracy for without TL than with TL. The model trained on the DFDC train set has the least self-test accuracy, which can be attributed towards its high quality Deep-Fake manipulations and precision in grouping people for face-swapping. The largest beneficiary of Transfer Learning is the model trained on Celeb-DF dataset, with a boost of 11.11%. The best performer on the custom test set from the single dataset train models is the DFD model with an accuracy of 70.57%, while the combined train split model achieves 73.20%. The worst performing cross-test pair is of Celeb-DF model tested on DFD. This makes sense since both datasets are opposite of each other in terms of manipulation and visual quality. The highest cross-test accuracy is 66.23% for the model trained on DFD and tested on DFDC. The combined test set performance is highest for the DFD model (74.03%) among the single distribution-based model, and the combined train model records the highest classification accuracy of 86.49%. This suggests that the XceptionNet has a vast learning capacity since it is able to learn the various manipulations present in all the three distributions, and is still not overfitting, conveyed by its impressive combined test accuracy.

Training sets for models	Test Sets									
	DFDC		Celeb-DF		DFD		Custom		Combined	
	w/o TL	with TL	w/o TL	with TL	w/o TL	with TL	w/o TL	with TL	w/o TL	with TL
DFDC	78.13	84.7	65.53	64.35	59.6	64.14	60.02	65.31	66.11	69.81
Celeb-DF	68.44	65.84	83.49	94.60	45.76	53.41	53.05	58.32	63.54	68.91
DFD	58.24	66.23	62.36	65.33	94.33	95.86	67.86	70.57	70.16	74.03
Combined	79.86	82.47	85.72	94.62	88.52	94.66	63.19	73.20	79.62	86.49

Table 6.2: Classification Accuracy Metrics (%) For Self And Cross Testing, With And Without Transfer Learning

6.3 Multi-Frame Deep-Fakes Detection

6.3.1 Compilation of Video Datasets

In the image datasets, faces from all videos were shuffled together and each face image was treated as an independent data point. On the other hand, in the video datasets faces from respective videos are arranged together as one data point. The individual data points were therefore sequences of faces from individual videos and a label indicating their ingenuity based on that of the video.

Similar to the compilation of image datasets, the videos are segregated into their respective splits and frames are extracted from individual videos. Faces from consecutive frames are taken so as to maintain the natural temporal changes while still being able to detect temporal anomalies caused by Deep-Fake algorithms. We extract 10 face images from each video. The thresholding rationale was carried from the compilation of image datasets, with the threshold also being kept 128. The final dataset consists of a number of subsets equal to the number of videos, with each subset containing faces from a corresponding video.

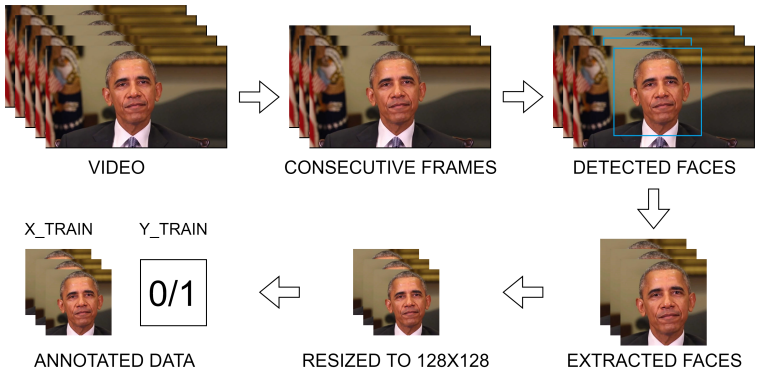


Figure 6.6: Video dataset curation pipeline

The final statistics for each set are displayed in Table 6.3.

Dataset	Train	Val	Test
Celeb-DF	275	60	58
DFD	255	54	54

Table 6.3: Distribution of all image datasets

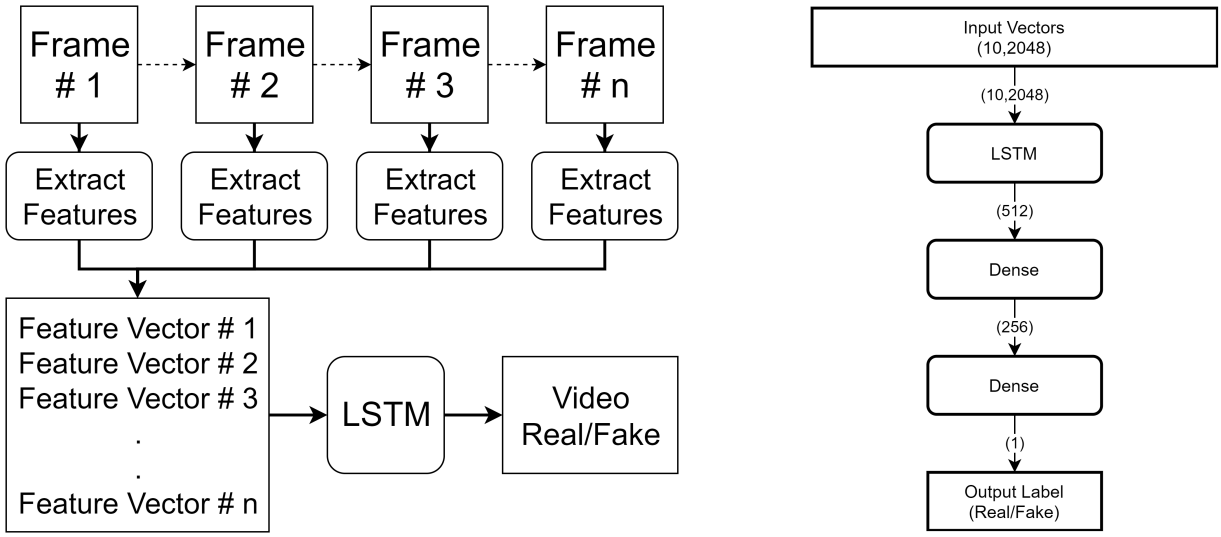
6.3.2 Model Architecture

The architecture of the model used for video classification can be broken down in two stages:

1. Feature Extraction
2. Long Short-Term Memory Network

An illustration of the architecture pipeline is shown in Figure 6.7a. The architecture is devised to be an end-to-end learning network, the inputs being a sequence of faces extracted from a video and the output being a prediction whether the video is real or fake. The process of feature extraction is carried out by a Convolutional Neural Network. We use XceptionNet as our CNN for this task. The XceptionNet extracts a 2048-dimensional feature vector from an input image. Thus, for each video we get 10 2048-dimensional feature vectors.

In order to process these vectors, we use a Recurrent Neural Network (RNN) to figure out the temporal changes in the vectors. Normal RNNs have the problem of short-term memory. It struggles to carry forward information from earlier time-steps to later time-steps as the length of the sequence grows. In order to counteract this issue, a new type of RNNs were developed, known as Long Short-Term Memory Networks (LSTMs) [45]. These networks have internal gates, which can allow passage of relevant information to later time-steps. We use an LSTM-based network to process the feature vectors extracted by the CNN. The output of the network is the probability of the video being fake. The architecture of the LSTM network used is shown in Figure 6.7b.



(a) (b)

Figure 6.7: (a) CNN+LSTM pipeline, (b) LSTM Network

6.3.3 Training

The models are trained using the datasets curated in 6.3.1. In order for the CNN to handle temporal input, we wrap it with a Time-Distributed layer. It applies the entire sequence of inputs to the same set of weights. Since each data point consists of 10 images, we keep the mini-batch size as 8. The CNN part of the network is initialized using ImageNet weights, while the LSTM is initialized using He initialization.

In order to create a baseline to compare the improvement of this pipeline, a CNN is also trained on the same dataset as the CNN+LSTM model. We use the XceptionNet for this task. Here, each data point is a single image, independent of other images that were extracted from the same video as the image.

6.3.4 Results

The results are summarized in Table 6.4. There is significant improvement in classification accuracy for both the datasets by using an LSTM in addition to the CNN. The Long Short-Term Memory network has captured the temporal variation in videos that the CNN model doesn't account for.

Dataset	CNN Accuracy	CNN+LSTM Accuracy
Celeb-DF	86.51 %	93.27 %
DFD	87.19%	92.96%

Table 6.4: Classification Accuracy Metrics (%) For various Deep-Fake datasets.

6.4 Deepfake Detection Challenge

6.4.1 About

AWS, Facebook, Microsoft, the Partnership on AI's Media Integrity Steering Committee, and academics collaborated together to build the Deepfake Detection Challenge (DFDC). The challenge was hosted by Kaggle. The challenge creators offered a total of $1,000,000 in prizes, making it one of the biggest challenges in terms of prize money. On October 21, an initial dataset was released named DFDC Preview [36]. On December 11, the full dataset was released, and the challenge began. Over 2000 teams participated in this challenge.

6.4.2 Dataset Curation

Datasets

There are 4 groups of datasets associated with the competition -

1. Training Set: This dataset is available for download to all participants. It contains over 100,000 videos and metadata files indicating which videos are real and which are fake. The participants are expected to use this dataset for training their models. For this work, we shall be focusing on this set as the labels are readily available.

2. Public Validation Set: A small set of 400 videos form the Public Validation Set which is hosted on Kaggle. When a kaggle notebook is committed, the notebook is made to generate predictions for this set. If the syntax of the submission file is correct, the notebook is allowed to be submitted.

3. Public Test Set: When the notebook is submitted after being committed, it runs on the Public Test Set, which is not accessible for participants. It is roughly ten times larger than the Public Validation Set. The scores on the public leaderboard are based on the performance of the participant on this set.

4. Private Test Set: This set is hosted outside of Kaggle, and is used to generate the private leaderboard. After the competition is over two final submissions from each participant is sent from Kaggle to the hosts who then run the code on this set. Once all participants have been assessed, the private leaderboard is released.

In the DFDC preview dataset, the creators had chosen a small set of actors for creating Deep-Fakes, and had split that set into a training set and a testing set. This ensured that there were no cross-set face swaps, preventing data leaking over to the training set from the testing set or vice versa. However, in the full version of the dataset released for the competition, the creators chose not to include metadata indicating any such split. This results in the splitting of the videos into training and testing sets in an arbitrary fashion, leading to potential cross-set leakage. In order to circumvent this issue, we needed to split the video dataset in such a way that the number of actors appearing in both sets was minimum. As doing so manually was impractical given the size of the dataset, an automated solution was found in the form of clustering.

Clustering

1. Face Embeddings: In order to perform clustering for any type of data, a vectorized representation of the data is required. We use the FaceNet framework to generate embeddings for the faces.

2. Principal Component Analysis (PCA): The FaceNet embedding is a 128-D dimensional vector. In order to reduce the time required for clustering, it is necessary to reduce the dimensionality of the embeddings to a much lower value. This is achieved by the means of PCA, which concentrates the variance present in the data in the first few dimensions. This helps us to only consider the axes containing the most variance and discard the rest, reducing the dimensionality of the data.

3. T-SNE: T-distributed Stochastic Neighbor Embedding, or T-SNE, is a machine learning algorithm which also performs dimensionality reduction. Its advantage over PCA is that it is able to capture complex polynomial relationships between features and represent them in lower dimensions unlike PCA, which is a linear algorithm. However, it is much more computationally expensive, which is why PCA is performed first on the raw data to reduce the dimensionality of the data.

4. DBSCAN: Density-based spatial clustering of applications with noise, or DBSCAN is a data clustering algorithm which groups points that are closely packed together. It does not require to specify the number of clusters, which is useful as we do not know the exact number of participants in the dataset. It is also quite robust to outliers.

Compilation of image dataset

Once the videos were clustered according to similarity in faces, we distribute clusters to training and validation sets. The process to compiling the dataset is similar to that in 6.2.1.

6.4.3 Approach

Model

After experiment with various model architectures, for our final submission we used ensemble learning. Due to the stochastic nature of learning algorithms used for training neural networks, each time a network is trained, it learns a different set of weights that map the inputs to the outputs. This ultimately results in varied performance on the training and the test set. Moreover, models with different number of parameters and different architectures may learn different features to classify the data, a combination of which may be required to perform well on the testing set. Hence, instead of using a single classifier, we use multiple classifiers trained individually and perform a weighted average on their predictions to get a final prediction. By this approach, the performance becomes less sensitive to the randomness of the training process. The weights of the weighted average can be adjusted to give more importance to the features extracted by particular models.

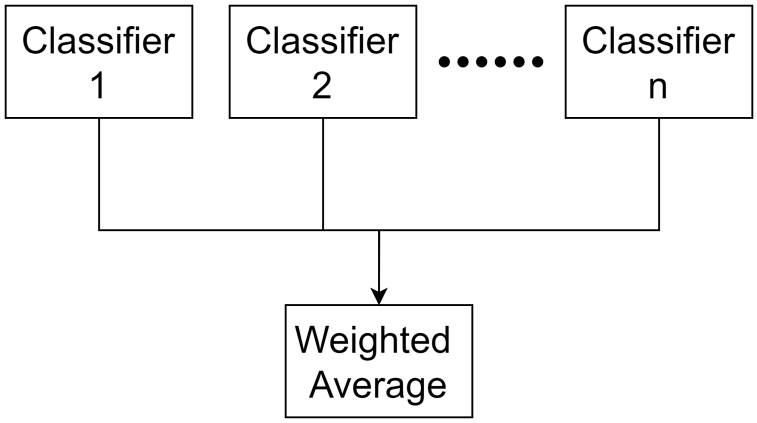


Figure 6.8: Ensemble Learning pipeline

Due to the higher visual quality of the videos, we take this opportunity to increase the dimensions of the images to 224x224, providing more details for the model to work with. We extract faces from 64 frames per video as input to the classifiers. Inferencing from the results obtained in 6.2.3, we choose to employ Transfer Learning in order to generalize to the test set distribution. We initialize the model with ImageNet weights.

Training

We train our models using the curated training and validation sets. We use the training set to optimize the models' weights. To avoid overfitting, we use Early Stopping by means of evaluating the models' performance on the validation set at the end of every epoch. When the performance does not improve for a certain number of epochs, we halt the training.

Testing

We use the public test set which is hosted on Kaggle as our testing set. Upto 2 submissions were allowed on Kaggle per day as part of the rules of this competition. The performance of the model is measured in terms of the log-loss score of the model. No other information regarding the testing is given, to prevent error-probing.

6.4.4 Leaderboard

The submission score is the log-loss score, calculated as shown:

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{6.1}$$

where
n is the number of videos being predicted
$\hat{y}_i$ is the predicted probability of the video being FAKE
y_i is 1 if the video is FAKE, 0 if REAL
$log()$ is the natural (base e) logarithm

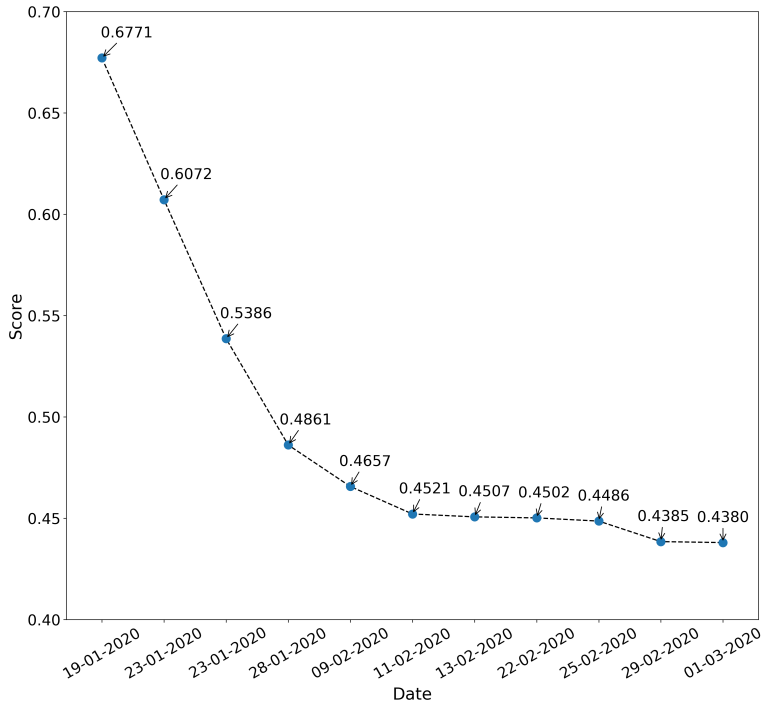


Figure 6.9: Our public leaderboard score over time

Chapter 7
EXPLAINABLE AI FOR VISUAL DEEP-FAKES DETECTION

7.1 Motivation

In many mission-critical scenarios, it is just not satisfactory to get the predictions for the concerned problem or task. Generally, task objectives or problem formalizations do not include this as an important component. Explaining the prediction or process leading to that prediction completes the task because getting a correct prediction only partially resolves the problem. Predictive performance on a test dataset and other similar metrics are given a lot of importance today and are kept as primary points of focus during modelling which can often lead to model training on features that are completely unrelated to the task. Many a time interpreting model decisions/outputs can help us learn more about the task, the data, and the cases in which the model will fail. Explanations may not be needed in low-risk environments (i.e. book recommender system) or in a task that has been already extensively studied (e.g. handwritten digit recognition).

Interpretability can be viewed as a learning experience towards the universal human desire to find reasons and logic over every scenario in the world. Humans usually conform to the anomalies to fit in their known knowledge structures and try to bridge the difference between contemplated and actual explanations. The biases picked by the deep learning models from training distributions or datasets can be scrutinized by interpretability. We can extract extra cognizance about the model and its decisions which paves the way for debugging and auditing. Interpretability also increases social acceptance among the general public which is usually not that technically equipped to understand all the accuracy metrics.

7.2 Stages of AI Explainability

Explainability is can be defined as making decisions or outputs from deep learning models transparent, which one can think of as making it interpretable (what result was that output supposed to have?), comprehensible (which data led to the decision?) and reproducible (given the data and specific circumstances, can we model a function that outputs the same decision?). We need to recognize and untangle explanatory factors in the underlying data to achieve this. Explainability can serve as a substitute to the decision-making system being scrutable by humans thus acting as an interface between the two.

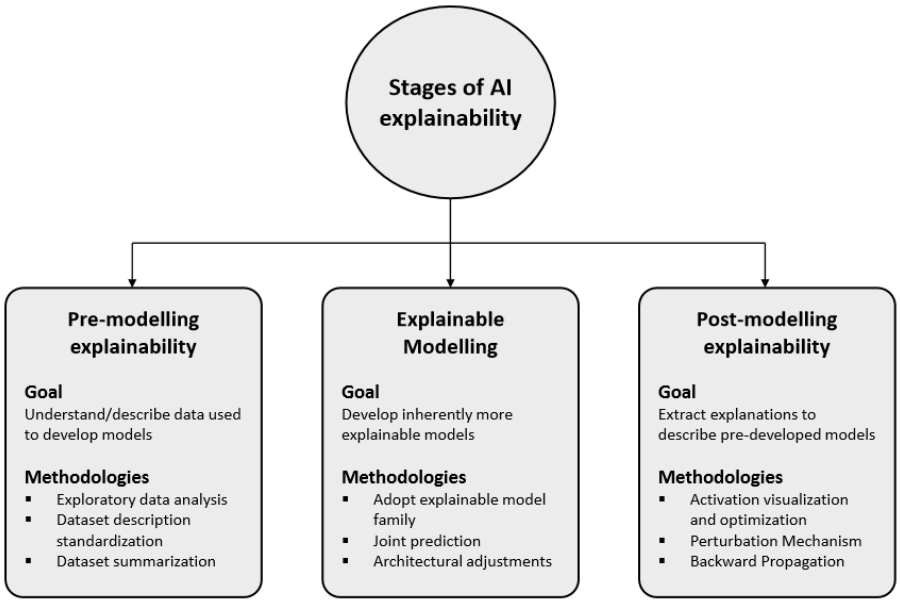


Figure 7.1: Flowchart illustrating the three stages of AI explainability

We have categorized the various stages of AI explainability that we implemented during our project timeline as the following:

1. **Pre-modelling explainability**, wherein we explored various datasets discussed in the previous chapters to incorporate a variety of distributions in terms of quality and resolution of Deep-Fakes. This was followed by dataset standardization and summarization to highlight the concepts of domain adaptation and dataset shift.

2. **Explainable modelling**, wherein we took XceptionNet as the base model for experimentation after a thorough inspection of the works of Rossler et al. [22] and Francois Chollet [23], which served as benchmarks for the choice of architecture for the model.

3. **Post-modelling explainability**, wherein we will analyze our model's behaviour in regards to features that the classifier is looking for and if it responds well to noise and affine transformations to evaluate the robustness of the classifier using various post-hoc explainability methods.

Currently, AI models are often developed with only predictive performance in mind. Thus, the majority of the literature pertaining to XAI is dedicated to explaining these pre-synthesized models. This bias of focus along with the recent popularity of XAI research has resulted in the development of numerous and diverse post-hoc explainability methods. We could then evaluate the robustness of the classifier in terms of the above points and get the explanations for the predictions. The methods we have tried include: Saliency maps and GradCAM, Intermediate Activations, Local Interpretable Model-Agnostic Explanations (LIME), Layer-wise Relevance Propagation (LRP), Integrated Gradients, Guided Backpropagation and DeconvNet.

The resultant explanations are often importance scores in terms of input features of the model. The scores are obtained by starting with the layer that is producing the given target, e.g. certain class logit, and estimating the contribution of neurons in the layer previous to that target. This process is repeated layer by layer until the contributions of input layer features are obtained. We have created heatmaps of the same to visualize the areas of the image that attract the attention of our model. Furthermore, we have attempted to test the robustness of our model to affine transformations and Gaussian Blur noise. The viewer will be able to intuitively visualize the regions of the image considered salient by our model for it's particular prediction.

7.3 Preliminary investigation using Saliency and Gradient-weighted Class Activation Mapping

The aim to use saliency maps is to represent the conspicuity or 'saliency'—at every location in the visual field by a scalar quantity and to guide the selection of attended locations, based on the spatial distribution of saliency. The most popular methods derive a gradient saliency map by back-propagating a gradient from the end of the network and project it onto an image plane. The gradient can typically be from a loss function, layer activation or class activation. Thus in order to get fine-grained details, it requires storage of the data necessary to compute a full backward pass on the input image. A preliminary investigation into the regions of interest for the decision of the model using saliency maps shows the primary areas of focus being the forehead and the face boundaries in Figure 7.2.

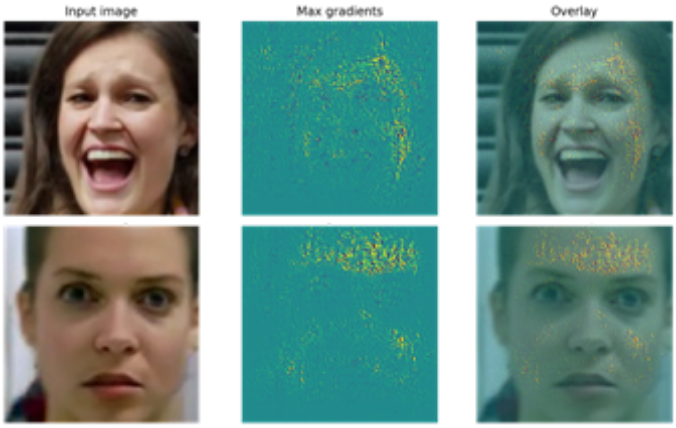
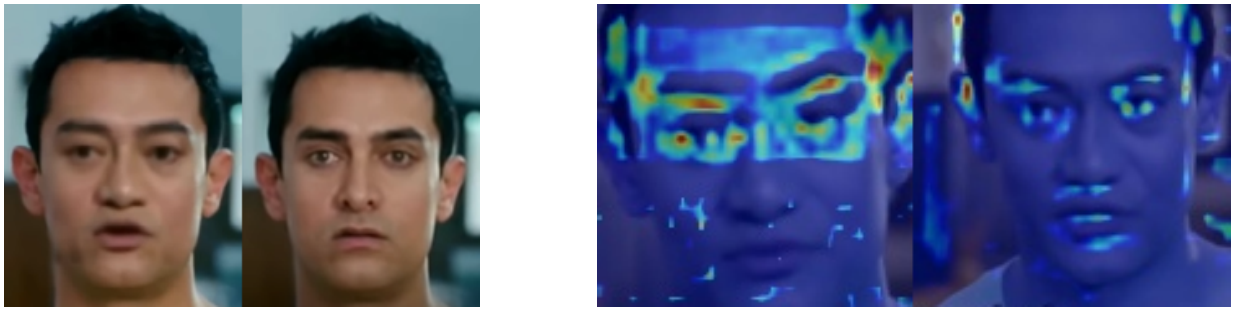


Figure 7.2: Saliency map representing areas where the trained CNN model (on DFD dataset) is looking at to detect facial manipulation

Grad-CAM is perhaps one of the most popular methods for interpreting CNN models. Here, we are trying to visualize how parts of the face affect the classifier's output by looking into the class activation maps. Class Activation Mapping (CAM) is one technique for producing heat maps to highlight class-specific regions of images. The output

of grad-CAM will be pixels that contribute to the maximization of a target function. Backpropagation can efficiently compute the gradients of the target function, with respect to the convolutional layer outputs. Given a face and whether it is fake or not as input, we forward propagate the image through CNN and then through task-specific computations we obtain a raw score for the category. The gradients are set to zero for the other class while for the desired class they are set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization (blue heatmap) which represents where the model has to look to make the particular decision.

We tried this method on the initially generated Deep-Fake video featuring Jack Ma as Aamir Khan in the famous scene of "What is a machine?" in the movie 3 idiots. We have showcased the face swap which was done in the video and the visualization result of the regions of interest for the convolutional neural network to detect facial manipulations in Figure 7.3. We can clearly observe the vast variation in the regions of interest of the model in-between frames. While in one frame we see the model focussing on the forehead and producing strong heatmaps near areas above the eyes and forehead, in another frame we see the prime regions localized are face boundaries, eyes, and lips. We can deduce that Grad-CAM can serve as an abecedarian approach towards localization of regions of interest for the task of detecting facial forgeries.



(a) (b)

Figure 7.3: (a) The face-swap of Jack Ma and Aamir Khan in the video, (b) Grad-CAM visualization of the fake faces of Jack Ma

7.4 Dataset and Training

We have solely focused on the DeepFakeDetection Dataset by Google and Jigsaw for the purposes of testing model interpretability. The fake component of the DFD dataset comprises Faceswap videos of consenting target actors, who in turn act as the real counterpart. Hence the manipulations are very localized and it can be easily verified if the model is looking at the right places. The traditional scale factor for cropping faces after detection using Dlib had been kept as 1.3x, however the face consisted of nearly 80% of the image in this case. It would be a lot more interesting to see how the model performs with a lot of background information to work with and if it is able to accurately localize the manipulations and artifacts. Hence, we trained the XceptionNet on the DFD dataset with a cropping scale of 2x where the face occupies not more than 60% of the entire image.

7.5 Results

We begin the description of our results by providing a summary of the test accuracies of our model displayed in 7.1. In case of the 1.3x model, we have not fed any significant background data to the model apart from the face itself. Hence, we can expect the model to accurately identify the features of the face, and unsurprisingly it fares better in the test set compared to the 2x model. We reiterate that our focus is not to achieve the highest accuracy on a benchmark. Considering this, we decided to showcase the results of only the 2x model. It would be interesting to see how well the model performs with a lot of background information. Based on the theme of our topic, we'll be showing only fake faces throughout the following sections. Also, we will test the robustness of our model to Gaussian blur noise and affine transforms. generate a better oversampled dataset that can be fed to models to learn the entire context of data.

Image Scale	Test Accuracy
1.3x	94.33%
2.0x	90.17%

Table 7.1: Comparison of Test Accuracies between differently scaled face datasets.

7.6 Intermediate Activations

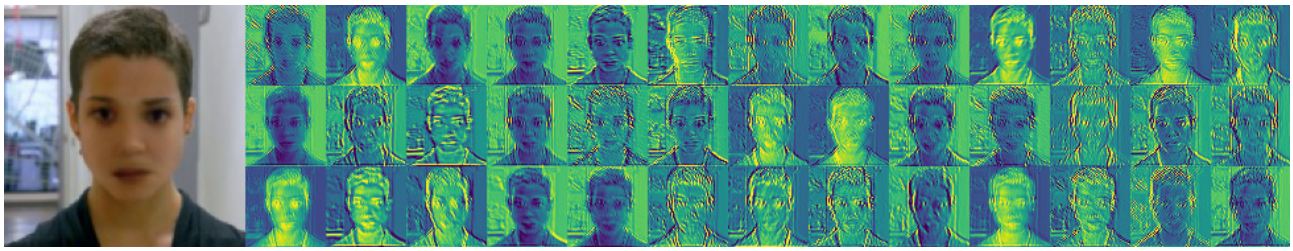


Figure 7.4: Input image and a slice of `block2_sepconv2_act` (Activation of the second sepconv layer of the second block)

We have showcased the activations of the `block2_sepconv2_act` layer for an input image in Figure 7.4. This particular layer detects various forms of edges present throughout the human face like the forehead, eyes, mouth, and jawline, while the activation retains most of the information present. The activations of the deeper layers become increasingly abstruse and less visually interpretable, as they start extracting complex features like structures and shapes of eyes, nose, and ears which play a prominent role in deciding whether the face is fake. The activation outputs of the deeper layers carry less information on the whole face and more information about localized face features.

In the deepest layers of the CNN model, we see that the activations have reduced by a very significant amount considering the fact that the patterns encoded in the filters are very intricate and those patterns are absent in the input image. The network effectively decomposes large features into smaller and more complex attributes repeatedly, with raw data going in (in our case, faces with background), and getting repeatedly transformed so that background information gets filtered out while useful facial information gets magnified and refined.

7.7 Local Interpretable Model-Agnostic Explanations (LIME)

LIME is a popular method of creating easy to explain interpretations of ML models. Local refers to local fidelity which is needed to really reflect the behaviour of the classifier "around" the instance being predicted. LIME is able to explain any model without needing to 'peek' into it, so it is model-agnostic. Intuitively, an explanation is a local linear approximation of the model's behaviour. While the model may be very complex globally, it is easier to approximate it around the vicinity of a particular instance. Treating the model as a black box, we perturb the instance we want to explain and learn a sparse linear model around it as an explanation.

We denote the classification model under consideration as follows: f: $\mathbb{R}_d \rightarrow \mathbb{R}$ and $g \in G$ as the explanation of the said model. Hence, the classification probability of a particular input x would be $f(x)$. This probability acts as a binary signal for the association of x with a certain class. To define a locality measure around x we use $\pi_x(z)$ to represent proximity between an instance z and x. To measure the unfaithfulness of g we define a loss measure $\mathcal{L}(f, g, \pi_x)$ for the locality of π_x. Finally, we minimize this loss $\mathcal{L}$ while keeping the complexity of the explanation $\Omega(g)$ low enough to preserve local fidelity and ensure interpretability by humans.

Following is the explanation provided by LIME:

$$\xi(x) = \underset{g \in G}{argmin}\ \mathcal{L}(f, g, \pi_x)\ +\ \Omega(g) \tag{7.1}$$

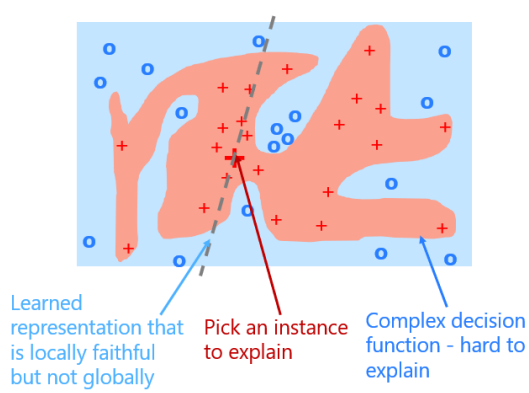


Figure 7.5: LIME

The loss function is defined as:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z)(f(z) - g(z'))^2 \tag{7.2}$$

Here, z is a perturbed data point in the original data space, z' is the corresponding interpretable representation and $\pi_x(z)$ weighs the samples based on its similarity to data point x.

Figure 7.6: 1st Row shows three input faces and 2nd row shows their corresponding outputs on analysis with LIME

Using the LIME [46], we present intepretable visualizations of our input image superimposed on the prediction based attention slice of our model as seen in Figure 7.6. We see that LIME is able to capture the relevant areas surrounding the face that resulted in it's classification. Little background data occupies the relevant slice, hence we can visually confirm that the model is looking at the right place.

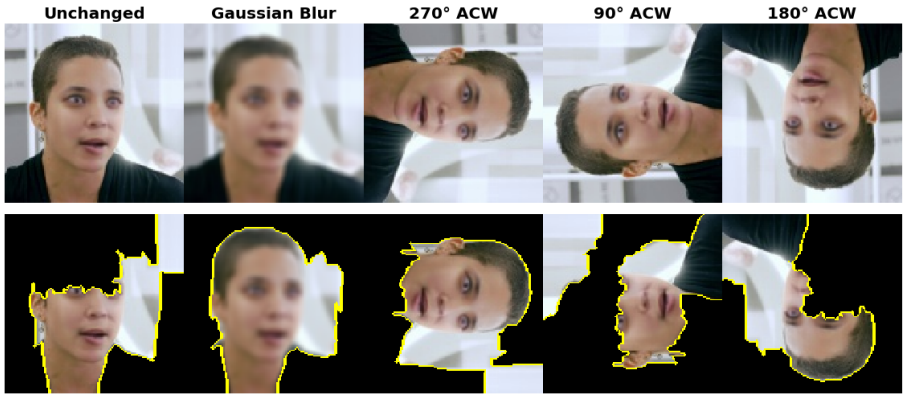


Figure 7.7: 1st Row shows the input along with it's perturbations and 2nd row shows the LIME descriptions of these inputs.

Despite not being trained on rotated images, we see that the model has produced similar results on blurred and rotated images as seen on Figure 7.7. More importantly, all these images have been detected as fake by our model, which points to it's robustness.

7.8 Layer-wise Relevance Propagation (LRP)

LRP is a method that identifies important pixels by running a backward pass in the neural network. The backward pass is a relevance redistribution procedure, where neurons that contribute the most to the higher-layer receive most relevance from it. LRP produces a heatmap in the input space indicating the relevance of each pixel contributing to the final classification outcome. In contrast to the susceptibility maps produced by guided backpropagation, LRP is able to directly highlight the positive contributions to the network classification in the input space.

Consider a deep neural network consisting of layers of neurons, the output of an upper-layer single neuron z_j would be,

$$z_j = \sum_i z_i \times w_{ij} + b_j \tag{7.3}$$

where z_i are the outputs of lower-layer neurons in the forward pass and w_{ij}, b_j are weights and biases of corresponding connections between neurons.

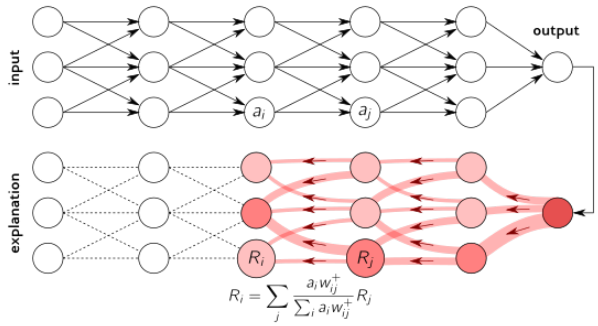


Figure 7.8: Layer-wise Relevance Propagation (Courtesy of heatmapping.org)

At first, the relevance score of the output layer neurons is initialized to the prediction score of the target class of interest c, $f_c(x)$, followed by the computation of a layer-by-layer relevance score for lower level neurons. This is done by computing a relevance message $R_{i\leftarrow j}$ from a lower level neuron to all it's upper level neurons.

$$R_{i\leftarrow j} = \frac{z_i \times w_{ij} + \frac{\epsilon \times sign(z_j) + \delta \times b_j}{N}}{z_j + \epsilon \times sign(z_j)} \times R_j \tag{7.4}$$

$$R_i = \sum_j R_{i\leftarrow j} \tag{7.5}$$

Then all these relevance messages are summed up to compute the relevance score of that lower level neuron. This process is continued till the input layer neurons. In Equation 7.4, ϵ is a stabilizing factor which has a small value and δ is a multiplicative factor which is set to either 0 or 1. We will use this formulation as the epsilon rule and the alpha-beta rules and the LRP sequential preset rules with and without Bias are shown for comparison.

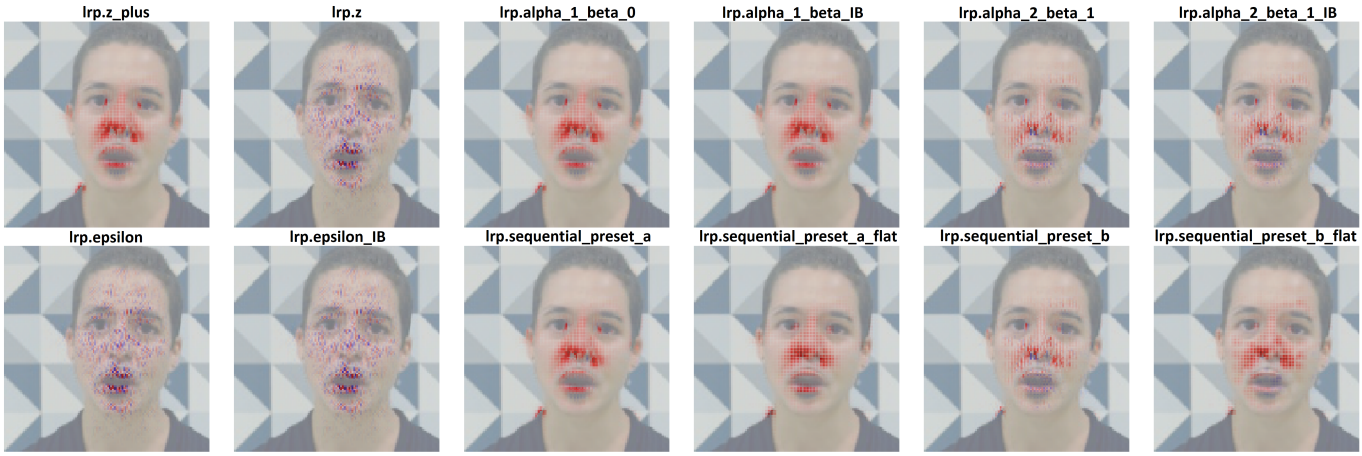


Figure 7.9: Comparison of different heatmaps produced by the different LRP rules

With the help of iNNvestigate [47], we have managed to leverage their suite of LRP methods to sufficiently localize the regions of manipulation to the nose and mouth region in our input image as shown in Figure 7.9 on the next page. All the given methods are reformulations of the fundamental principle of LRP with each relevance score of each neuron being backpropagated all the way to the input layer, LRP Sequential being designed specifically for CNNs. The heatmap suggests that the artifacts found in the nose and mouth region of the face are essential for the classification of this particular image.

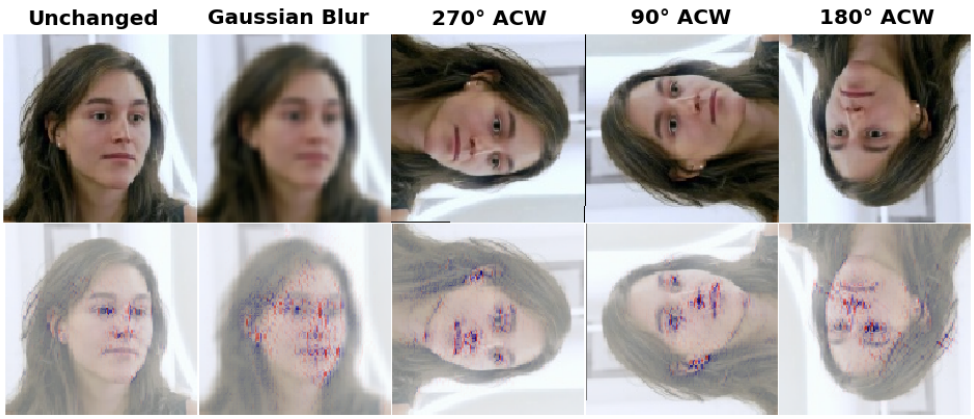


Figure 7.10: `lrp.sequential_preset_b_flat` compared on various perturbed inputs.

We've displayed the results of LRP by perturbing the input in Figure 7.10 by way of Gaussian Blur and Affine Transformations. We have fixed the method used as `lrp.sequential_preset_b_flat` and applied the same variations to the input and contrasted it with the input heatmap. Apart from the noise, we see that LRP has preserved the structural qualities of the original heatmap. These experiments aim to firmly establish the validity of these methods and contrast it's performance on a variety of inputs.

7.9 Integrated Gradients

Most of the known backpropagation interpretability methods have a fundamental weakness of not satisfying two elemental axioms—Sensitivity and Implementation Invariance. An attribution method satisfies Sensitivity if, for every input and baseline that differ in one feature but have different predictions, the differing feature is given a non-zero attribution. Implementation Invariance is the property that the attributions are always identical for two functionally equivalent networks (networks with their outputs are equal for all inputs, despite having very different implementations). Integrated Gradients is designed to be guided by these axioms and its implementation requires no change to the concerned network and simply a few calls to the standard gradient operator can be done to implement it. We have presented results of applying integrated gradients on the very same input used in LRP for comparison in Figure 7.11

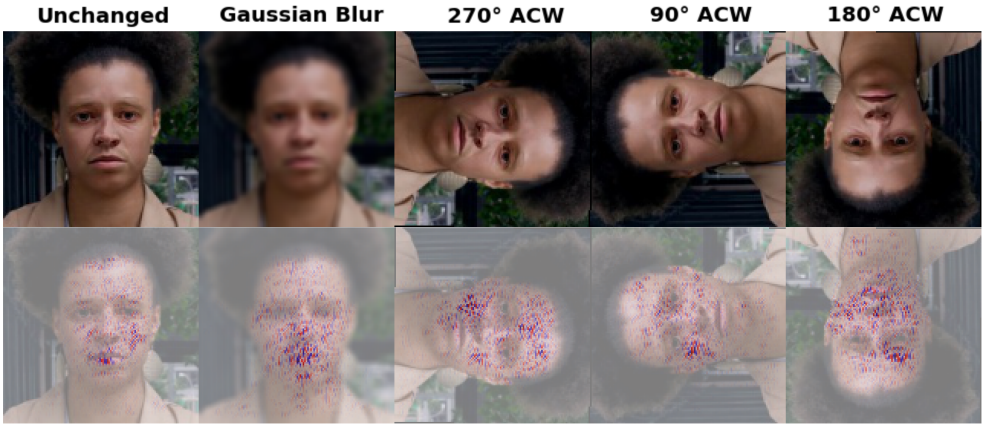


Figure 7.11: `integrated gradients` compared on various perturbed inputs.

7.10 Deconvolutional Network

Initially proposed as a method to perform unsupervised learning, Deconvnet implemented here is not used to train the model or to perform any learning tasks. It is just as a tool to delve into an already trained convolutional network. A deconvnet maps the features to pixels which is quite the opposite of convolutional network which performs the

mapping from pixels to features. A deconvnet uses the same components(filtering, pooling) of a convnet but in reverse. To inspect a convnet, a deconvnet is connected to each of its layers providing a continuous path back to image pixels. To begin, convnet is presented with an input image and features computed throughout the layers. For examination of a given convnet activation, we set all other activations in the layer to zero and pass the feature maps as input to the attached deconvnet layer. Now until the complete input pixel space is traversed, we sequentially unpool, rectify, and filter to reconstruct the activity in the layer beneath that gave rise to the chosen activation.

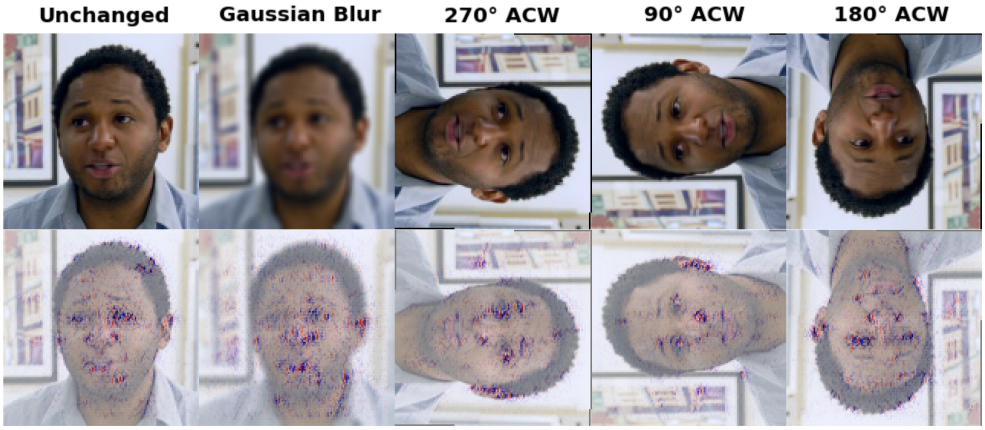


Figure 7.12: `deconvnet` compared on various perturbed inputs.

7.11 Guided Backpropagation

Gradients look hazy in a weird interference pattern when there is a cancellation of positive and negative influences of the paths during backpropagation. By keeping only the paths that lead to positive influence on the class score and suppressing the negative influence ones we can generate much cleaner looking heatmaps. This approach is used in Guided Backpropagation. Negative gradients at a particular ReLU neuron, state that this neuron has a negative influence on the class that we are trying to visualize. Guided Backpropagation combines vanilla backpropagation and DeconvNets when dealing with the ReLU nonlinearity. In Guided Backpropagation, the absolute values of the gradient of the output with respect to the input nodes are shown as a heatmap, with the additional twist that negative gradients are set to zero at the rectification layers of the network. Guided Backpropagation gives better image resolution than saliency mapping techniques however the visualization resulting from their method usually stays the same despite different classes. Hence there might be a trade-off between resolution quality and pointing out wherein the image is considered important for the network.

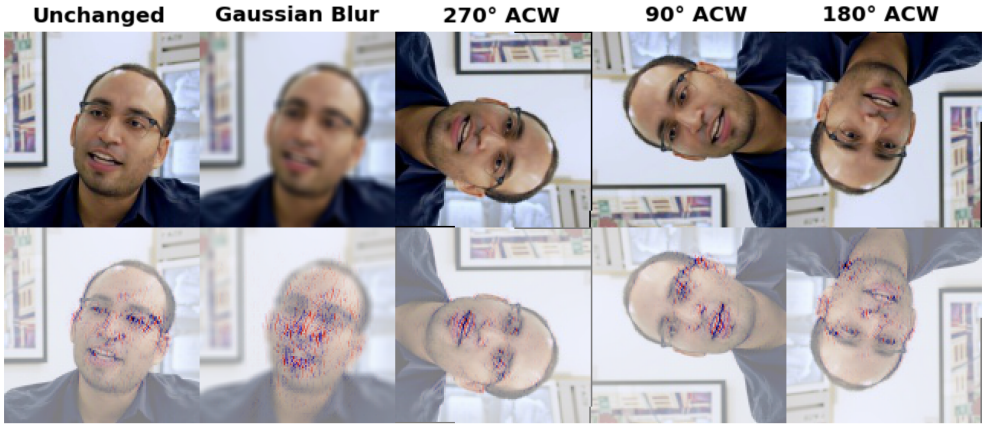


Figure 7.13: `guided backpropagation` compared on various perturbed inputs.

Chapter 8
AUDIO DEEP-FAKES DETECTION

8.1 Introduction

In addition to manipulation of faces, audio manipulation is another manner in which digital media can be altered to potentially pervert the original intention of the speaker in the media. While less prevalent than facial forgeries, audio manipulation can lend credibility to Deep-Faked videos by impersonating the voice of the target in the video. This makes the detection of Deep-Fake audio just as crucial as detection of image-based manipulations.

In this work, we explore two approaches to detect such audio based manipulations:

1. Feature-based approach: In this approach, we extract relevant features from the audio signal, and use them to classify the audio as bonafide or spoofed. The features extracted are as follows:
   (a) Root Mean Square Energy
   (b) Chroma Features
   (c) Spectral Centroid
   (d) Spectral Bandwidth
   (e) Spectral Rolloff
   (f) Zero-Crossing Rate
   (g) Mel Frequency Cepstral Coefficients
2. Image-based approach: In this approach, we represent audio signals with their Mel Spectrograms and and use them to classify audio files.

8.2 Preliminaries

8.2.1 Features

1. *Root Mean Square Energy:* Given a signal $x(n)$, the Root Mean Square Energy (RMSE) is given by:

$$\sqrt{\frac{1}{N}\sum_{n}|x(n)|^2} \tag{8.1}$$

For a general signal, the RMSE is calculated as a single value from all the values present in the signal. However, due to the non-stationary nature of audio signals, it is preferable to compute the energy over short intervals to capture the variation over time. Different types of speech may have different energy content, making it an important feature to consider for audio classification.

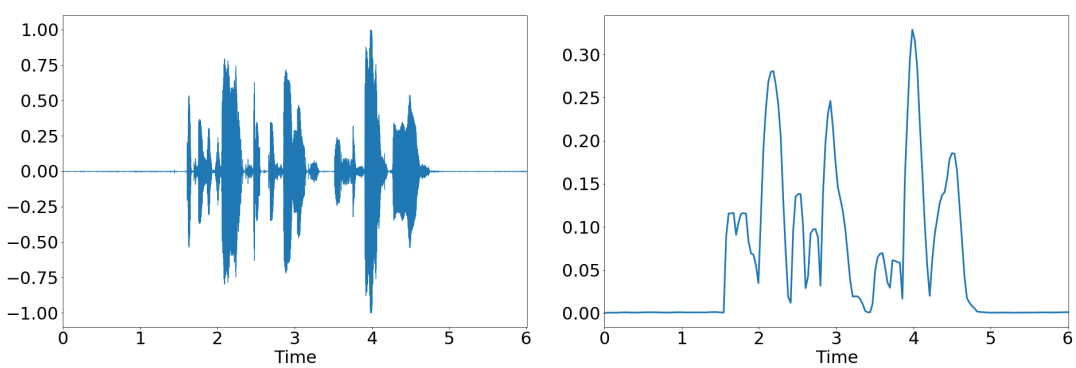


Figure 8.1: Amplitude Plot and Root Mean Square Energy Plot of sample audio signal

2. *Chroma Features:* The chroma features represent the frequency spectrum as 12 pitch classes. The entire frequency is divided into 12 bins denoting the 12 chroma present in the musical octave.

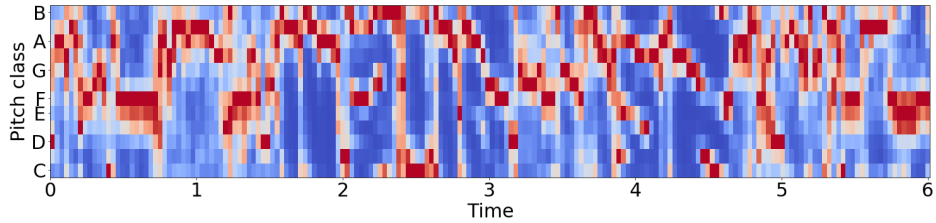


Figure 8.2: Chromagram of sample audio signal

3. *Spectral Centroid:* The spectral centroid is indicative of the nature of the frequency prevalent in the signal. For instance, a higher spectral centroid value indicates a concentration in the high frequency region in the spectrum. It is computed for the ith audio frame as follows:

$$\mu = \frac{\sum_{k=1}^{N} f(k) \cdot f(k)}{\sum_{k=1}^{N} m(k)} \tag{8.2}$$

where $m(k)$ is the magnitude at the kth frequency bin, $f(k)$ is the centre frequency at the kth frequency bin.

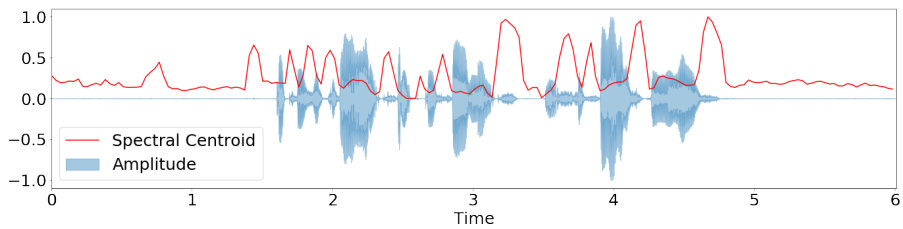


Figure 8.3: Spectral Centroid Plot of sample audio signal

4. *Spectral Bandwidth:* The range of frequencies in the audio signal spectrum which correspond to one-half of the peak magnitude of the spectrum is known as the spectral bandwidth of the audio signal. It is a measure of the extent of the frequency spectrum. The pth order spectral bandwidth is given by:

$$\left(\sum_{k} m(k) \left(f(k) - \mu \right)^{p} \right)^{\frac{1}{p}} \tag{8.3}$$

where $m(k)$ is the magnitude at the kth frequency bin, $f(k)$ is the centre frequency at the kth frequency bin and μ is the spectral centroid.

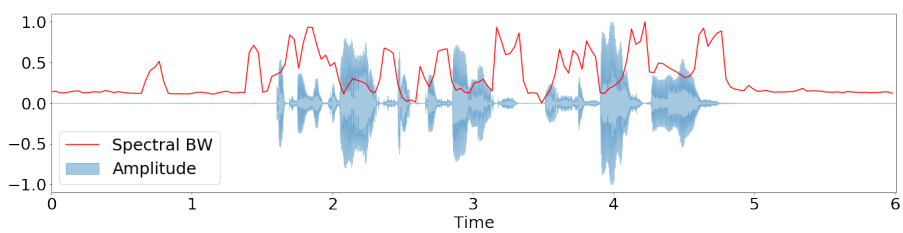


Figure 8.4: Spectral Bandwidth Plot of sample audio signal

5. *Spectral Rolloff:* The spectral rolloff is the point in the frequency below which 85% of the energy present in the spectrum resides. This provides the general shape of the spectra by ignoring the outlier higher frequencies and focusing on the portions of the spectra where most of the energy is present. It is given by:

$$\underset{f_r \in \{1,\ldots,N\}}{\arg\min} \sum_{k=1}^{f_r} m(k) \geq 0.85 \cdot \sum_{k=1}^{N} m(k) \tag{8.4}$$

where f_r is the rolloff frequency, and the $m(k)$ is the magnitude at the kth frequency bin.

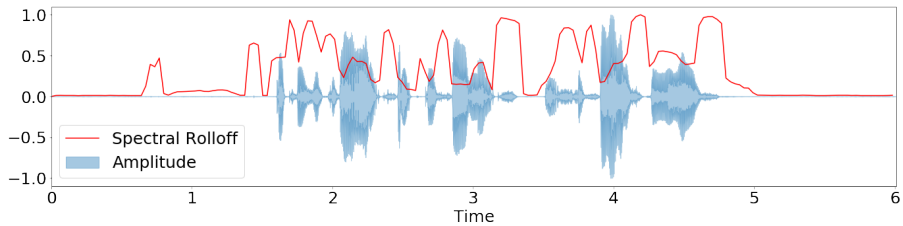


Figure 8.5: Spectral Rolloff Plot of sample audio signal

6. *Zero-Crossing Rate:* The rate of sign-changes in a signal is referred to as the zero-crossing rate. Lower frequency signals have a lower zero-crossing rate due to lesser oscillations per second as compared to higher frequency signals. It is given by the formula:

$$\frac{1}{2W_L}\sum_{n=1}^{W_L}|sgn[x(n)] - sgn[x(n-1)]| \tag{8.5}$$

where $x(n)$ is the audio signal, W_L is the length of the window, and sgn is the signum function.

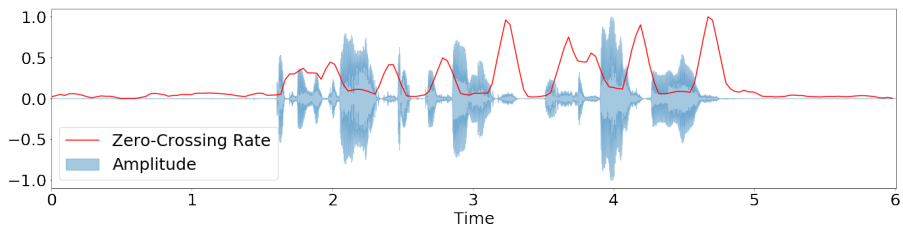


Figure 8.6: Zero-Crossing Rate Plot of sample audio signal

7. *Mel Frequency Cepstral Coefficients (MFCC):* MFCCs are a small (10-20) set of features which help to characterise the nature of the frequency spectrum. They are computed by taking the Cosine Transform of the Mel-scaled spectrogram of the signal.

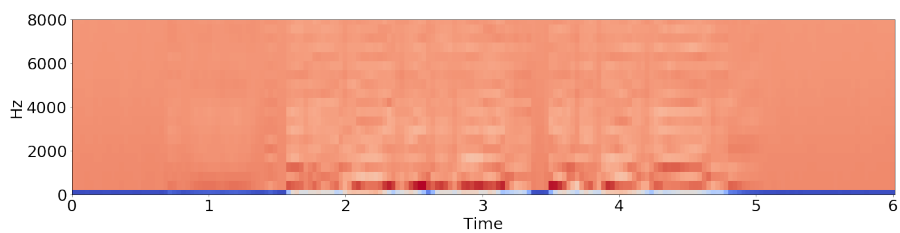


Figure 8.7: MFCC Plot of sample audio signal

8.2.2 Short-Time Fourier Transform (STFT)

For analysis of audio data, the frequency domain is generally preferred. This is because the frequency composition of the audio signal plays a vital part in the characteristics of the signal, which the frequency domain expresses both comprehensively and intuitively. The Discrete Fourier Transform is generally used to observe which frequencies are present in a signal and in what quantities are they present. However, audio signals tend to be non-stationary, i.e. their characteristics change over time. Computing the Fourier Transform on the entire audio signal will provide a spectrum averaged over time. Instead, if the Fourier Transform is computed over sections of the audio signal and then combined together, it provides a focus on the signal properties at a particular point in time. It also helps to observe the change in the frequency spectrum over time. In order to smoothen the combined Fourier transforms, a sliding window approach could be used in which the audio segments on which the transforms are computed overlap to some extent. This approach is known as the Short Time Fourier Transform (STFT) of the signal. For a given signal $x(n)$ and a window $w(n)$, the STFT is given as:

$$STFT\{x(n)\}(h,k) = X(h,k) = \sum_{n=0}^{N-1} x(n+h)w(n)e^{-i2\pi\frac{kn}{N}}. \tag{8.6}$$

8.2.3 Spectrograms

The STFT is a complex-valued matrix, and hence cannot be viewed directly. Instead, it is visualised using its log-spectra, $20\log_{10}(X(h,k))$. A reason to use logarithm instead of just the absolute value is that the logarithm roughly approximates the hearing sensitivity of the human ear, making the visualization more pronounced in terms of features of auditory importance. This visualisation when viewed as a heatmap is referred to as a spectrogram.

In order to improve the representation of the frequnecy spectra, we may introduce the information about human auditory perception into the visualization. This is achieved on the magnitude axis previously by using the log-spectra, which helps roughly mimic the human ear's sensitivity. The other dimension is the frequency axis, which can also be subjected to such transformation. In this work, we use the Mel-Scale to transform the frequency axis into a logarithmic quantity.

A well-known approximation to convert the frequency scale to the mel scale is as follows:

$$m = 2595\ \log_{10}\left(1 + \frac{f}{700}\right). \tag{8.7}$$

The resulting spectrogram is known as the Mel Spectrogram. It emphasizes on the nature of humans to perceive both loudness and frequencies on a non-linear scale. We make use of the Mel Spectrogram as representations of the audio data.

8.3 Dataset Curation

The Automatic Speaker Verification Spoofing And Countermeasures Challenge 2019 (ASVspoof 2019) [48] dataset developed by Google AI and Google News among others is used in this work. The dataset encompasses two partitions considering two scenarios: Physical Access (PA), and Logical Access (LA). Both the partitions are derived from the VCTK base corpus which includes speech data captured from 107 speakers (46 males, 61 females). For the PA scenario, it is assumed that speech data is captured using a microphone. The spoofed speech data are recordings of bonafide speech which assumed to be covertly captured and then re-introduced to the audio system using a replay system. Meanwhile for the LA scenario, the spoofed audio is generated using Text-To-Speech (TTS) synthesis and voice conversion (VC) technologies.

In this work, the Logical Access dataset will be explored. The dataset consists of three splits, namely training, development and evaluation which comprise the speech from 20 (8 male, 12 female), 10 (4 male, 6 female) and 48 (21 male, 27 female) speakers respectively. The three datasets are disjoint in terms of speakers, which avoids cross-set leakage. The recording conditions of the source data are identical. The training and development sets contain spoofed audio generated using the same spoofing algorithms (referred to as known attacks). The evaluation set however also contains spoofed audio generated using different algorithms (referred to as unknown attacks).

8.3.1 Feature-Based Datasets

The seven features described in 8.2.1 are computed for audio data using a sliding frame approach, hence they are each time-varying sequences. The order of the features in the feature vector extracted from the audio is shown in Figure 8.8.

Averaged Features Dataset

For this dataset, the time-average of each feature sequence was calculated and used as a feature. The temporal variations of the features are discarded, and each feature sequence is represented by its mean. In this dataset, every audio file is represented by a vector of 37 features.

Feature Sequences Dataset

While extracting the features from audio data, if the window length and the hop length of the sliding frame is kept the same for all the feature computations, all the feature sequences are of the same length. Combining all the features together, we get a vector consisting of 37 features that is varying in time. Thus, in this dataset each audio file is represented by a time-varying feature vector. We keep the window length as 2048 samples and the hop length of the window as 512 samples.

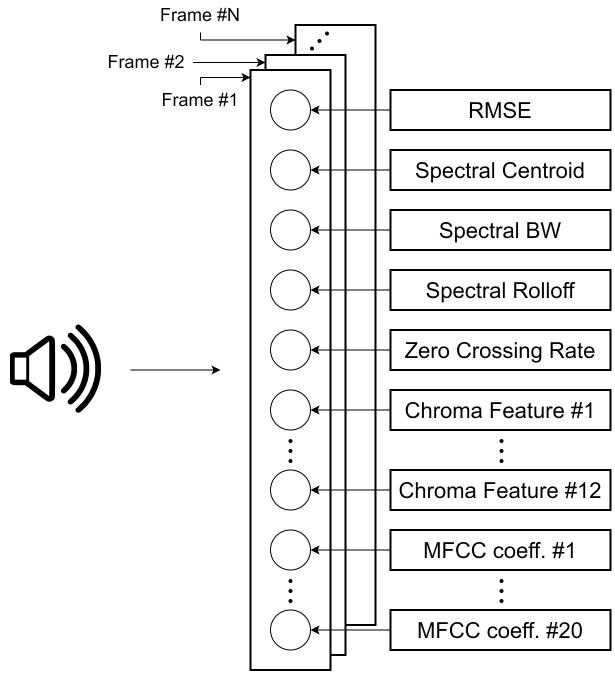


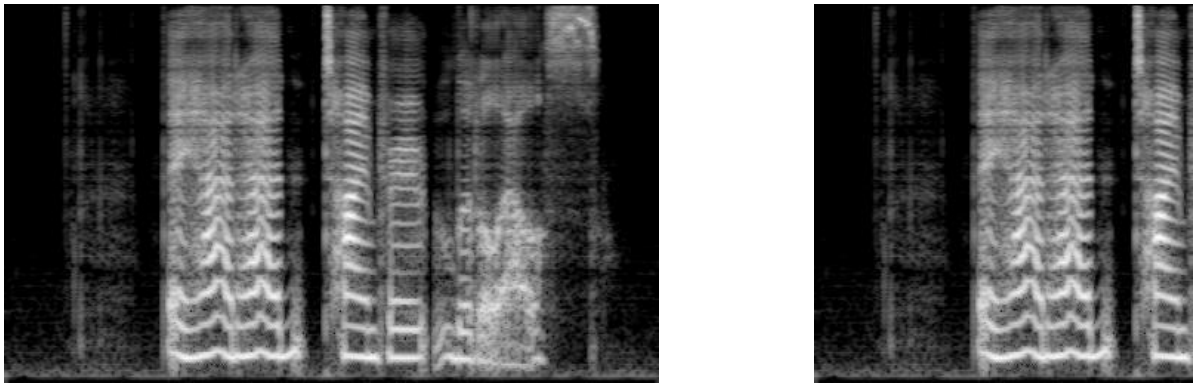
Figure 8.8: Feature Vectors extracted from audio data

8.3.2 Image-Based Datasets

Mel Spectrogram Dataset

The librosa library has been used to generate the Mel Spectrograms of the audio data. The number of mel levels used to represent the frequency axis is chosen to be 256. Since the lower threshold of frequencies audible to the human ear is approximately 20 Hz, frequencies below 20 Hz are not represented in the spectrogram. Due to the sampling rate of the audio files being 16 KHz, by the Nyquist sampling theorem the audio data contains frequencies only up to 8 KHz. The frequency range of the spectrogram is thus between 20 Hz and 8 KHz. The length of the window is set as 1024, and the Hann window function is chosen. The hop length of the window is 256.

The Mel Spectrogram is generated as an image, of height 256 due to the number of mel levels and of width depending on the length of the audio. Since a Convolutional Neural Network is being used to perform the classification, the images in the dataset need to be of the same size. In order to avoid discarding too many images for being too short in terms of width, we set a width threshold of 256. Images narrower than 256 pixels are discarded, while for images wider than 256 pixels, the excess width is trimmed off from the right end. This results in a dataset containing images of size 256x256. Each spectrogram corresponds to rougly 2 seconds of its original audio file.



(a) (b)

Figure 8.9: (a) Mel Spectrogram of sample audio file, (b) Cropped Mel Spectrogram

Amplitude Plot Dataset

In order to observe the improvement in detection accuracy due to usage of mel spectrograms, a baseline data set is created which consists of time-based amplitude plots of the audio signals. The matplotlib library is used to plot the audio files.

(a) (b)

Figure 8.10: (a) Amplitude plot of sample audio file, (b) Cropped Amplitude plot

8.4 Training

In order to perform the task of binary classification of the audio signals as either bonafide or spoofed, various models are trained on the datasets curated in 8.3. The details of the models and the training parameters are covered in this section.

8.4.1 Feature-Based approach

For the Averaged Features Dataset, we use a Support Vector Machine (SVM) [49] as the model for classification. The Radial Basis Function Kernel is used as the kernel for the SVM.

For the Feature Sequences Dataset, we use a Long Short-Term Network. The architecture of the network is shown in Figure 8.11. The model is trained using the binary cross-entropy loss function and optimized using the Adam optimizer. The mini-batch size is set as 32.

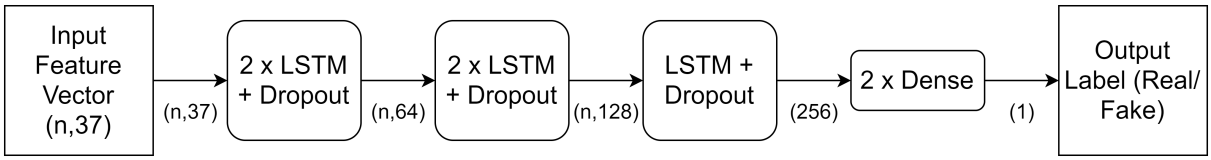


Figure 8.11: Architecture of LSTM used for the feature-based dataset

8.4.2 Image-Based approach

The architecture of the Convolutional Neural Network used for this task is shown in Figure 8.12. The choice of optimizer is the Adam optimizer. The binary cross-entropy loss function is used as the loss function for training. A mini-batch size of 64 is used.

We train a model on the training partitions of both the datasets generated, and validate on the development partition. Finally, we note the evaluation partition accuracy for each model.

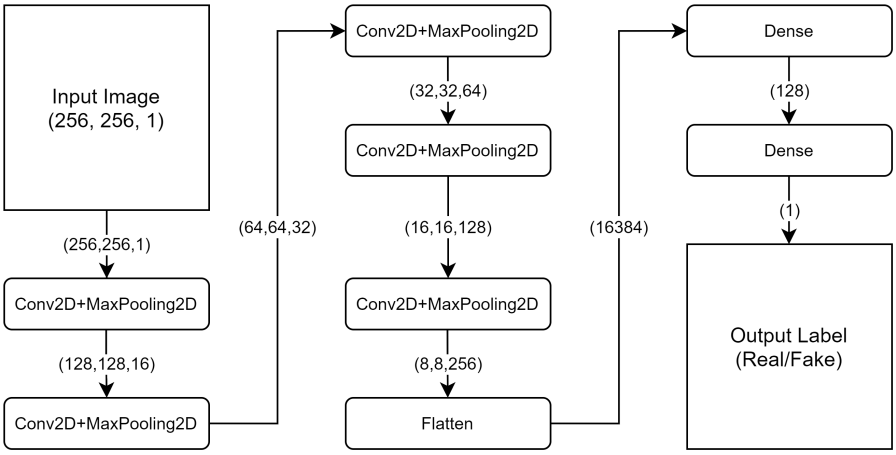


Figure 8.12: Architecture of CNN used for the image-based datasets

8.5 Results

The results of the experiment are displayed in Table 8.1. The classification metrics are for the known attacks scenario (development set) and the unknown attacks scenario (evaluation set). Using feature sequences instead of their averages prove to provide a small benefit in classification for both the scenarios. Both the CNN approaches, however fare better than the feature-based approaches. There is significant improvement in classification accuracy by using Mel-Spectrograms over amplitude plots as representation of the audio data.

Approach	Dataset	Development	Evaluation
SVM	Averaged Features	85.36%	81.97%
LSTM	Feature Sequences	87.58%	83.96%
CNN	Amplitude Plot	80.88	84.34%
CNN	Mel Spectrogram	98.17%	89.14%

Table 8.1: Classification Accuracy Metrics (%) For Development and Evaluation sets.

8.6 Feature Importance Analysis for Averaged Features Dataset

8.6.1 Introduction

Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. Popular examples for types and methods of feature importance scores include statistical correlation scores, coefficients calculated as part of linear models, decision trees, and permutation importance scores. In this section, we are going to explore some of these techniques for analysis of the averaged features dataset. These techniques are broadly classified into 3 categories: filter methods, wrapper methods, and embedded methods. Filter methods filter out relevant features from a large set of features using a parameter value which acts as the filter. The filtering in this section is done using the Pearson correlation matrix. Wrapper methods evaluate on a specific machine learning algorithm to find optimal features. The wrapper methods used in this section include Forward Elimination, Backward Elimination, and Recursive Feature Elimination(RFE). Embedded methods fix features during the model designing process wherein feature selection is done by observing each iteration of the model training process. The most commonly used embedded methods are Regularization methods which penalize a feature given a coefficient threshold. In this section, the Lasso Regularization is implemented.

8.6.2 Pearson's Correlation Coefficient

Pearson's correlation coefficient is the test statistics that measure the statistical relationship, or association, between two continuous variables. It is known as the best method of measuring the association between variables of

interest because it is based on the method of covariance. It gives information about the magnitude of the association, or correlation, as well as the direction of the relationship. Pearson's correlation coefficient can take a range of values from +1 to -1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable. A value less than 0 indicates a negative association; that is, as the value of one variable increases, the value of the other variable decreases. The formula for Pearson's correlation coefficient (r) is given below in equation:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n(\sum x^2) - n(\sum x)^2][n(\sum y^2) - n(\sum y)^2]}} \tag{8.8}$$

In Figure 8.13, we have plotted the Pearson correlation heatmap to see the correlation of independent variables with the output variable 'label'. The label is 1 if the audio is a fake and 0 if it is real. We observe that due to a large number of features there is no strong correlation between the individual features and the label. The highest Pearson's correlation coefficient observed is 0.37 which is for the features spectral bandwidth, 3rd chroma feature and the 9th Mel Frequency Cepstral coefficient.

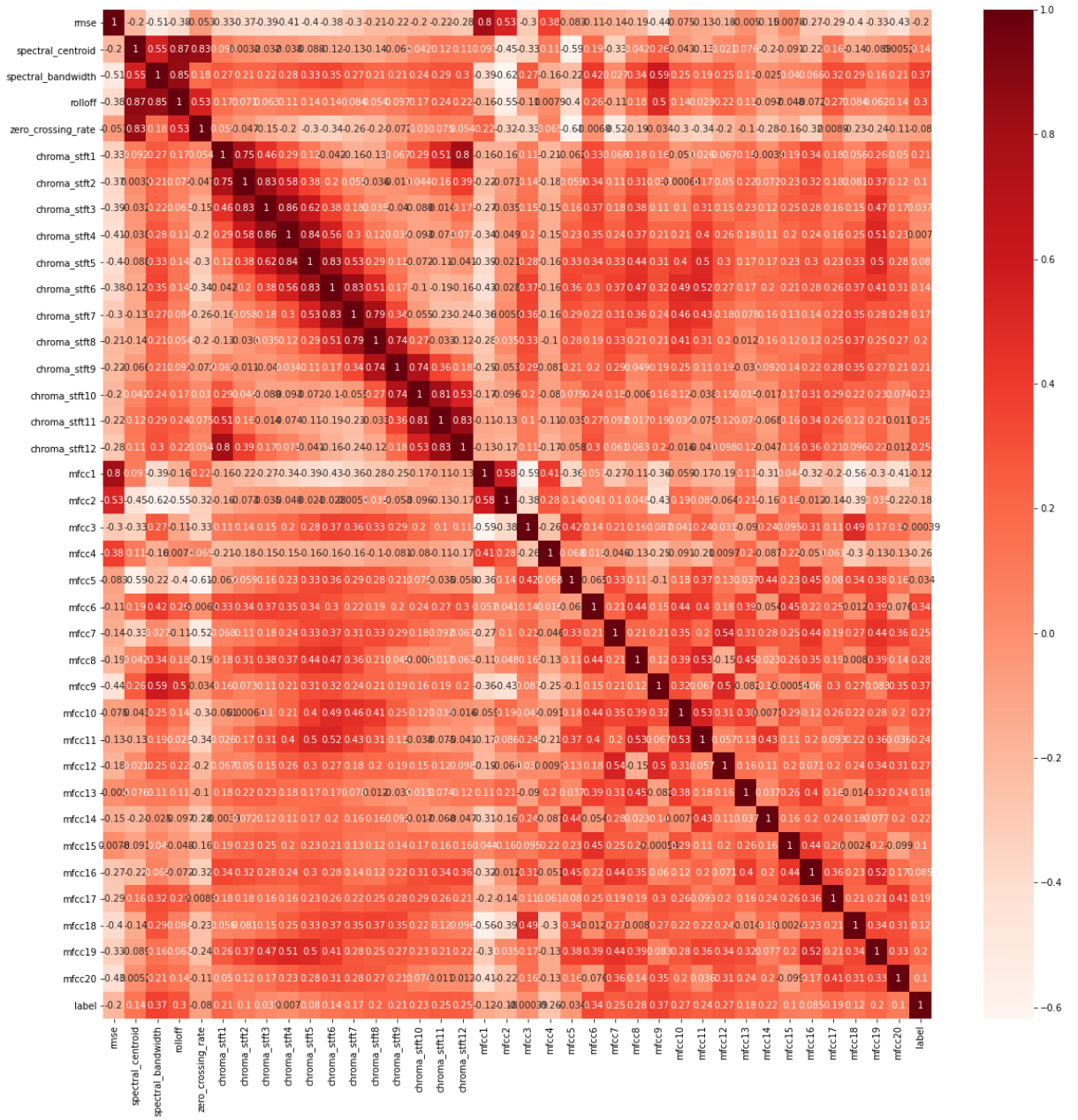


Figure 8.13: Pearson Correlation Heatmap

8.6.3 Forward Selection

Forward selection is an iterative method in which we start with having no feature in the model. We keep adding a feature each iteration which best improves our model performance till an addition of a new variable does not improve the performance. The metric value for each feature is calculated on the cross-validation dataset. The feature offering the best metric value is selected and appended to the list of features. The metric value is computed for each set of features and feature offering the best metric value is appended to the list of relevant features. This process is repeated until we have the desired number of features. Thus we can look at the feature set as a hyperparameter. Forward feature selection allows us to tune this hyperparameter for optimal performance.

We have displayed the performance metrics for sequential forward elimination with respect to standard error in Figure 8.14. We used the plot_sequential_feature_selection function of the Mlextend library and setting the forward parameter as 'True' to observe the variation of performance with an increase in the number of features.

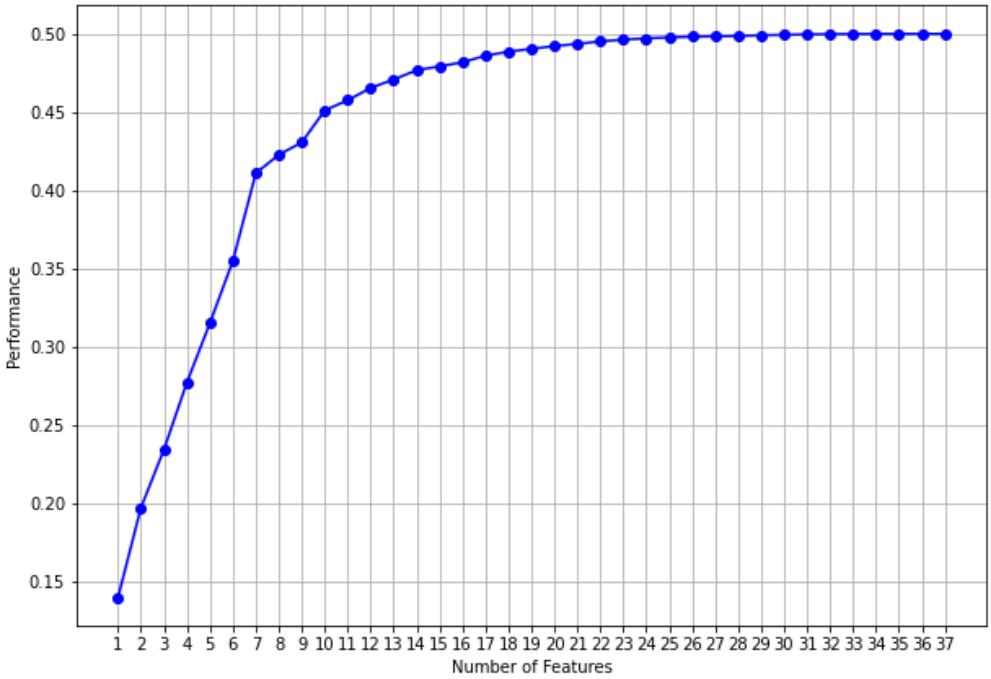


Figure 8.14: Sequential Forward Selection(w. StdErr)

The final list of important features given by forward selection are as follows: ['spectral_bandwidth', 'mfcc7', 'mfcc4', 'mfcc3', 'mfcc14', 'mfcc6', 'chroma_stft4', 'zero_crossing_rate', 'mfcc12', 'mfcc8', 'chroma_stft9', 'mfcc2', 'chroma_stft1', 'mfcc16', 'mfcc11', 'mfcc15', 'mfcc20', 'mfcc18', 'chroma_stft7', 'chroma_stft11', 'mfcc1', 'mfcc5', 'rmse', 'chroma_stft6', 'mfcc9', 'chroma_stft5']

8.6.4 Backward Elimination

This method is exactly reverse the forward elimination method. In backward elimination, we start with the full model (including all the independent variables) and then remove the insignificant feature with the highest p-value. This process repeats again and again until no improvement is observed on the removal of features resulting in getting the final set of significant features. Usually, in most cases, a 5% significance level is selected. This means the p-value will be 0.05. The significance value for the inclusion of the feature is determined by the user. We are using the OLS model for this method which stands for "Ordinary Least Squares".

We have displayed the performance metrics for sequential backward elimination with respect to standard error in Figure 8.15. We used the plot_sequential_feature_selection function of the Mlextend library and setting the forward parameter as 'False' to observe the variation of performance with an increase in the number of features.

The final list of important features given by backward elimination are as follows: ['rmse', 'spectral_bandwidth', 'zero_crossing_rate', 'chroma_stft1', 'chroma_stft4', 'chroma_stft5', 'chroma_stft6', 'chroma_stft7', 'chroma_stft9', 'chroma_stft11', 'mfcc1', 'mfcc2', 'mfcc3', 'mfcc4', 'mfcc6', 'mfcc7', 'mfcc8', 'mfcc9', 'mfcc10', 'mfcc11', 'mfcc12', 'mfcc13', 'mfcc14', 'mfcc15', 'mfcc17', 'mfcc18', 'mfcc20']

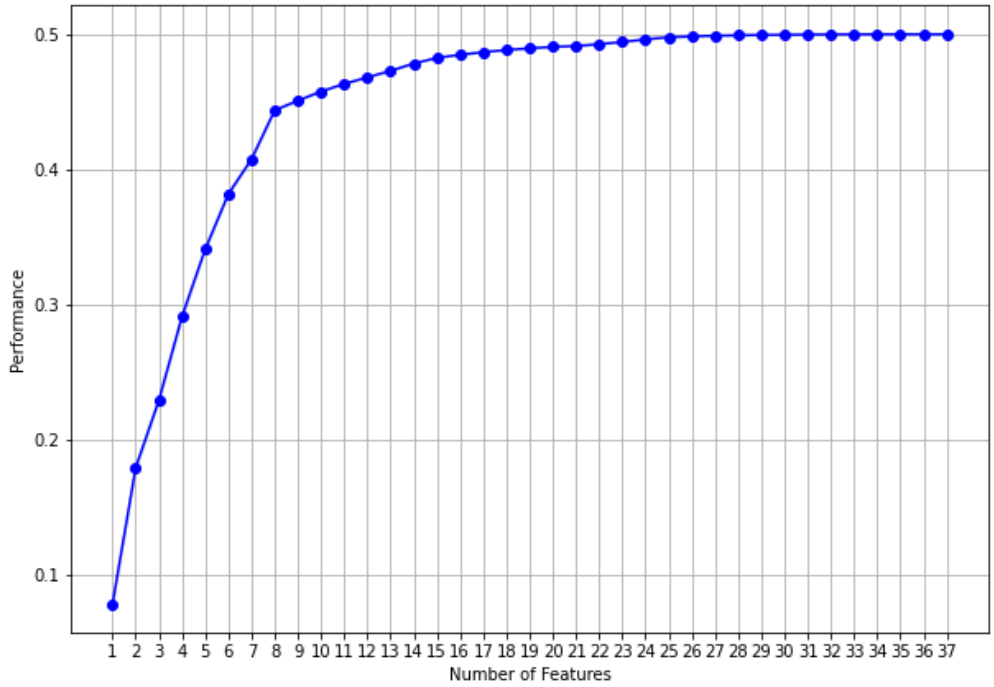


Figure 8.15: Sequential Backward Elimination(w. StdErr)

8.6.5 Recursive Feature Elimination(RFE)

The Recursive Feature Elimination (RFE) method works by recursively removing attributes and building a model on those attributes that remain. It uses the accuracy metric to rank the feature according to their importance. The RFE method takes the model to be used and the number of required features as input. It then gives the ranking of all the variables, 1 being most important. RFE attempts to eliminate dependencies and collinearity that may exist in the model. RFE requires a specified number of features to keep, however it is often not known in advance how many features are valid. To find the optimal number of features cross-validation is used with RFE to score different feature subsets and select the best scoring collection of features.

In Figure 8.16, we have displayed the scores for n number of features ranging from 1 to 37 computed by RFE method and the method returned 13 as the optimum number of features with a score of 0.340960.

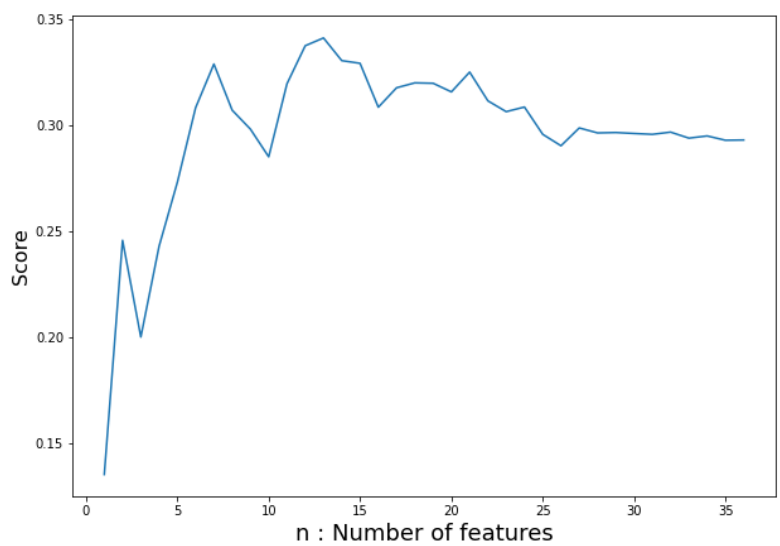


Figure 8.16: Score of n features from 1 to 37

We fed 13 as number of features to RFE and got the final set of features given by RFE method, as follows: ['spectral_bandwidth', 'zero_crossing_rate', 'chroma_stft4', 'chroma_stft6', 'chroma_stft7', 'chroma_stft9', 'mfcc2', 'mfcc3', 'mfcc4', 'mfcc6', 'mfcc8', 'mfcc12', 'mfcc14']

8.6.6 LASSO Regularization

When predictor features are highly correlated with each other and there are many predictors, the general solution is to reduce variance at the cost of introducing some bias. This approach is called regularization and is almost always beneficial for the predictive performance of the model. LASSO regression cost function is a type of linear regression that uses shrinkage. The acronym "LASSO" stands for Least Absolute Shrinkage and Selection Operator. Shrinkage is where data values are shrunk towards a central point, like the mean. This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients.

If the feature is irrelevant, LASSO penalizes its coefficient and make it 0. Hence the features with coefficient 0 are removed and the rest is taken. In the Figure 8.17, we have displayed feature importance computed using Lasso regularization and we observe that the method selects all the features except the spectral roll-off feature which has a coefficient of zero.

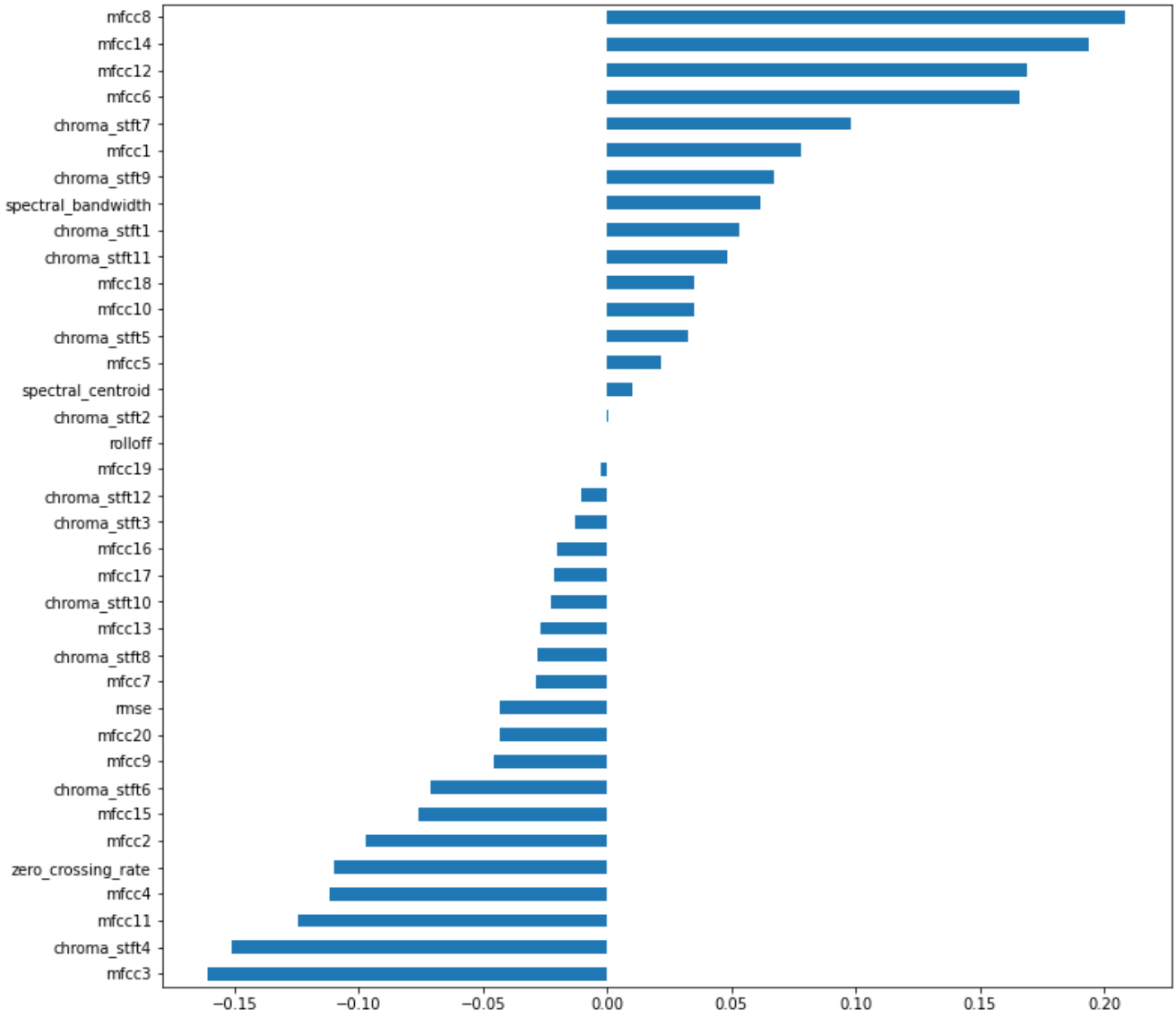


Figure 8.17: Feature Importance using Lasso model

Chapter 9
Applications

Deep-Fake Detection has various practical benefits to different entities. It can be used to detect forged videos of celebrities deliberately saying something controversial and also fake pornographic videos that have become a menace. This helps to provide a sense of security, especially for women. Governmental agencies can use these algorithms and software to check for forged digital content which can cause political or religious tensions between countries, to fool the public and influence results in election campaigns, or create chaos in financial markets by creating fake news.

These are some levels our project is applicable at:

- **Individual:**
  It can be used as a personal forgery detection tool wherein users want to check the authenticity of the digital media they are receiving or sharing with someone else.
- **Commercial:**
  Social media platforms can integrate our services to detect and automatically flag manipulated/suspicious content. The project has commercial viability and can be integrated with the software of a subscribing enterprise or company thus strengthening our objective at a wider level.
- **Judiciary:**
  Our network can be used in courts to disprove fabricated/tampered image/video evidence.

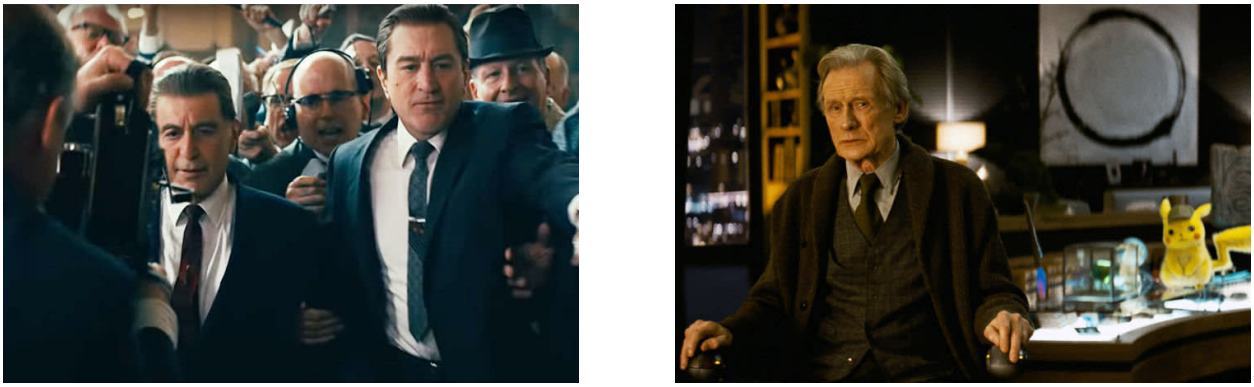
Deep-Fakes have the potential to harm individuals as well as a democracy which has serious consequences. But there is a beneficent side to Deep-Fake technology which is not noxious. One of the many uses of Deep-Fake technology is to improve video quality, turning raw videos into more finished ones as if they were made by a cognoscente of the field which includes coloring of black and white films.

Though Deep=Fake technology is primarily regarded as a visual tool, it can also serve as an audio manipulation technique that opens up a range of applications in the film industry. We can train a machine to study the voice of an actor and generate the voice in case of an actor's absence due to some emergency. Synchronization of lip movement, facial expressions with the actor's script can also be done as shown in Figure 9.1. This application sometimes helps in dealing with censorship issues or sound recording complications in the production stage. The cost of creating to more relatable experience for international audiences by paying regional language actors can totally be eliminated and hence the technology is used in multiple language dubbing.

Figure 9.1: Face Capture and Reenactment of RGB Video. Source: Face2Face

The future can be an all-out replacement of actors with Deep-Fake impersonator leading to a significant reduction in-studio and casting costs. The possibility of creating new actors is not a far-fetch, wherein producers and directors choose features they need from different actors, such as facial expression of one, the body language of another, and the voice from another actor, creating altogether a new artificially generated actor. This could lead to the fulfillment of a director's vision and also cutting the exorbitant cost of a renowned actor.

Another use-case of Deep-Fake technology is refined face editing and special effects in the post-production stage. Age reversal to create a younger version of an actor like some examples mentioned in Figure 9.2 is the most popular application currently. Editing scenes where an actor is absent or not able to participate due to emergencies, miscommunication, or conflicts is also used widely as many films released today have been using computer-generated scenes where the characters are portrayed by actors that have passed away.



(a) (b)

Figure 9.2: (a) . De-ageing plays a prominent role in Netflix's popular crime drama 'The Irishman' with actors Al Pacino and Robert De Niro © Netflix/Alamy, (b) Deepfakery shaved several years off British actor Bill Nighy in 'Pokémon Detective Pikachu' © Warner Bros/Everett Collection/Alamy

Chapter 10
Conclusion

A custom test set to test the problem of dataset shift was prepared. The custom dataset constitutes of Deep-Fakes generated using FaceSwap-GAN and Deep-Fakes uploaded on YouTube. In FaceSwap-GAN an autoencoder architecture was employed to generate Deep-Fakes. We have showcased some results from the generation process to address the high-quality of manipulations that can be achieved using open source tools and computational resources.

The results displayed in Table 6.2 reflect that the intuition behind the effectiveness of Transfer Learning for Deep-Fake Detection is accurate since it holds true for all the different distributions considered. The problem of dataset shift due to varying test environments is also visible through the poor cross-set performances of models trained separately on the three different distributions. This conveys the idea that a task as sensitive as facial manipulation detection requires more variance in pre-processing and generation techniques for dataset compilation in order to capture the real-world conditions of differently distributed test data. This is supported by the result of the model trained on combined train set giving the highest accuracies for the custom and combined test sets since it is composed of samples from all the three different distributions and effectively contains more variety for learning the given task. The implications of false negatives are too costly for this task, so a deep-learning approach requires extensive and exhaustive datasets for training. Interestingly none of the models could score a higher accuracy than 73.20% on the custom test set, which goes to show there is still a wide gap between the quality of Deep-Fakes released in large public datasets and the manipulations generated by a few motivated creators. Even though these manipulations are on YouTube for entertainment and awareness, one can wonder how much damage is possible by similar people who are waiting to strike at the right time.

We extracted features of the face from multiple consecutive frames using the CNN model and fed them to the LSTM network as an alternative training approach. We trained the combination of CNN and LSTM pipeline to account for the temporal discontinuities between successive frames in a Deep-Fake video and observed that the accuracy increased significantly when compared to the previous approach of training on independent faces.

On the interpretability front, we have obtained a collection of results that propose explanations to the predictions given by our classifier model in terms of heatmaps or image concept slices, also the noisy input and affine transform results point to our model achieving robustness and rotational invariance to a great extent. Hence, we have substantiated the performance of our model towards the task of detecting Deep-Fake images from a video in a way that even a lay-person can be convinced. There is a consistency observed between these interpretability methods in terms of the regions of interest highlighted by them, with many of them focusing their attention on the same regions in the image. Thus, the utilization of XAI techniques furthers our understanding of complex models and provides an arena to present some essential context to the seemingly obtuse decisions arrived at by AI.

To detect audio manipulations, we trained Support Vector Machine classifier, Long Short Term Network, and Convolutional models using feature-based and image-based approaches. Averaged features and feature sequences datasets were created using multiple speech signal characteristics for the feature-based approach. Amplitude plots and Mel spectrograms datasets were created for the image-based approach. We observed that the feature sequences dataset in the feature-based approach and Mel spectrogram dataset in the image-based approach lead to significant improvement in development and evaluation sets.

During the project timeline, we realized that some areas limited the scope of the project due to the time constraint and the limited research done in those areas. Like for example, the final step in the generation of Deep-Fakes is to merge the manipulated fake face back into the original frame. While this sounds like an easy task, it is where face-swap goes wrong. The algorithm to stitch a face back onto an image is hard-coded and lacks the flexibility to detect mistakes. Also, each frame is processed independently; there is no temporal correlation between them, meaning that the final video might have some flickering. This is the part where more research is needed.

Researchers focused on Deep-Fake detection are largely using smart applications of artificial intelligence to expose the deceptive images. In fact, the detection methods are similar enough to the methods used to make Deep-Fakes that the research on detection inadvertently provides a roadmap for improving the fakes. Worse, the Deep-Fake discriminators themselves can be used directly in conjunction with generators to form a GAN in the Deep-Fake generation process to improve their output.

A range of papers has discovered telltale signs of Deep-Fakes, including unnatural blinking patterns, distortion in facial features, inconsistencies across images within a video (especially concerning lighting), incongruities between the speech and mouth movements of the subject, and even learning to note the absence of biometric patterns specific to world leaders. However, these detection methods are likely to be short-lived. The blinking example is especially informative. When researchers at the University of Albany noted that Deep-Fakes could be detected based on

irregular blinking patterns, it took only a few months for new Deep-Fakes to emerge that had corrected for this particular imperfection.

There has been limited research on localizing forgeries in images and videos featuring multiple faces due to resolution and focus issues. Further, even highly successful Deep-Fake detection methods are difficult to scale considering the spectrum of distributions spawned by various types of Deep-Fakes datasets in regards to the quality of Deep-Fakes, varying resolutions and a multitude of forgery methods.

Our future work concerns deeper analysis and experimentation with different model architectures, training the model on more types of manipulated media created using Deep Learning techniques to cover a wider spectrum of Deep-Fakes and then applying the interpretability methods on those models. The scale factor which is a key parameter in the interpretation of the model can be tweaked for general experimentation thus amounting to the different resolutions for which the predictions can be interpreted. A new approach towards the training process can be taken wherein instead of keeping it supervised, we feed a feature vector of the manipulation on the face along with the face to make sure that the model learns in a semi-supervised fashion and check for predictions.

There is increasing public disquiet about the consequences of Deep-Fakes and the possibility that stricter rules slowing down research in many fields related to AI. As the technology to forge digital media improves rapidly, we are heading for a future in which digital content cannot be easily trusted. Refining our critical thinking skills is the best way to protect ourselves from fake media content. Being skeptical about the media content that you receive or encounter online, especially if it is designed to be infuriating, abhorrent and sensationalistic is a good practise. To lessen the spread of fake content, double-checking for its genuineness before reposting it on social media is very important.

Bibliography

[1] C. Bregler, M. Covell, and M. Slaney, "Video Rewrite: Driving Visual Speech with Audio," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, p. 353–360, 1997.

[2] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making Faces," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, p. 55–66, 1998.

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[4] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2Face: Real-Time Face Capture and Reenactment of RGB Videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2387–2395, 2016.

[5] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing Obama: Learning Lip Sync from Audio," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[6] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Headon: Real-Time Reenactment of Human Portrait Videos," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS) - Volume 2*, p. 2672–2680, 2014.

[8] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[9] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, 2019.

[10] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," *arXiv:1912.04958v2*, 2019.

[11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499*, 2016.

[12] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," *arXiv:1902.08710v2*, 2019.

[13] S. Vasquez and M. Lewis, "MelNet: A Generative Model for Audio in the Frequency Domain," *arXiv:1906.01083v1*, 2019.

[14] W. Luo, J. Huang, and G. Qiu, "JPEG Error Analysis and Its Applications to Digital Image Forensics," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 480–491, 2010.

[15] T. Bianchi and A. Piva, "Image Forgery Localization via Block-Grained Analysis of JPEG Artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, 2012.

[16] A. Bharati, R. Singh, M. Vatsa, and K. W. Bowyer, "Detecting Facial Retouching Using Supervised Deep Learning," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 1903–1913, 2016.

[17] J. H. Bappy, C. Simons, L. Nataraj, B. S. Manjunath, and A. K. Roy-Chowdhury, "Hybrid LSTM and Encoder–Decoder Architecture for Detection of Image Forgeries," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3286–3300, 2019.

[18] E. I. A. El-Latif, A. Taha, and H. H. Zayed, "A Passive Approach for Detecting Image Splicing using Deep Learning and Haar Wavelet Transform," *International Journal of Computer Network and Information Security*, vol. 11, no. 5, pp. 28–35, 2019.

[19] D.-H. Kim and H.-Y. Lee, "Image Manipulation Detection using Convolutional Neural Network," *International Journal of Applied Engineering Research*, vol. 12, no. 21, p. 11640–11646, 2017.

[20] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median Filtering Forensics Based on Convolutional Neural Networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, 2015.

[21] B. Bayar and M. C. Stamm, "A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, p. 5–10, 2016.

[22] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–11, 2019.

[23] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[24] X. Tian, S. Du, X. Xiao, H. Xu, E. S. Chng, and H. Li, "Detecting synthetic speech using long term magnitude and phase information," in *IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)*, pp. 611–615, 2015.

[25] M. Pal, D. Paul, and G. Saha, "Synthetic Speech Detection Using Fundamental Frequency Variation and Spectral Features," *Computer Speech & Language*, vol. 48, no. C, p. 31–50, 2018.

[26] E. A. AlBadawy, S. Lyu, and H. Farid, "Detecting AI-Synthesized Speech Using Bispectral Analysis," in *Workshop on Media Forensics at CVPR*, 2019.

[27] D. Gunning and D. Aha, "DARPA's Explainable Artificial Intelligence (XAI) Program," *AI Magazine*, vol. 40, pp. 44–58, 2019.

[28] M. Lent, W. Fisher, and M. Mancuso, "An Explainable Artificial Intelligence System for Small-Unit Tactical Behavior," in *Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence*, p. 900–907, 2004.

[29] N. J. S. Morch, U. Kjems, L. K. Hansen, C. Svarer, I. Law, B. Lautrup, S. Strother, and K. Rehm, "Visualization of neural networks using saliency maps," in *Proceedings of International Conference on Neural Networks (ICNN)*, vol. 4, pp. 2085–2090 vol.4, 1995.

[30] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," in *2nd International Conference on Learning Representations, (ICLR), Banff, AB, Canada, April 14-16, Workshop Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2014.

[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

[32] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 1135–1144, 2016.

[33] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLOS ONE*, vol. 10, no. 7, pp. 1–46, 2015.

[34] "Faceswap-GAN." https://github.com/shaoanlu/faceswap-GAN/.

[35] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-DF: A New Dataset for DeepFake Forensics," *arXiv:1909.12962*, 2019.

[36] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. Ferrer, "The Deepfake Detection Challenge (DFDC) Preview Dataset," *arXiv:1910:08854*, 2019.

[37] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[38] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Neural Networks," *ECCV 2014, Part I, LNCS 8689*, 2014.

[39] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan, "A theory of learning from different domains," *Machine Learning Journal*, vol. 79(1-2), pp. 151–175, 2010.

[40] M. Torres, T. Raeder, R. Alaiz-RodríGuez, N. Chawla, and F. Herrera, "A Unifying View on Dataset Shift in Classification," *Pattern Recognition*, vol. 45, no. 1, p. 521–530, 2012.

[41] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research*, vol. 10, no. 60, pp. 1755–1758, 2009.

[42] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, 2017.

[43] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, 12 2014.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, p. 1026–1034, 2015.

[45] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM," in *9th International Conference on Artificial Neural Networks (ICANN)*, vol. 2, pp. 850–855, 1999.

[46] "LIME." https://github.com/marcotcr/lime/.

[47] "iNNvestigate." https://github.com/albermax/innvestigate/.

[48] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. W. D. Evans, T. H. Kinnunen, and K. A. Lee, "ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection," in *20th Annual Conference of the International Speech Communication Association*, pp. 1008–1012, 2019.

[49] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning Journal*, vol. 20, no. 3, p. 273–297, 1995.