LANCASTER UNIVERSITY

MASTERS THESIS

A Novel Approach To DeepFake Video Detection

Author:
Pritam MISHRA

Academic Supervisor:
Dr. Ignatius EZEANI
Co-Supervisor:
Dr. Rob YOUNG (Raytheon)

A thesis submitted in fulfillment of the requirements
for the degree of Master Of Science

in the

Department of Computer Science and Statistics
Lancaster University

School of Computing & Communications | Lancaster University

September 4, 2020

A Novel Approach To DeepFake Video Detection

Submitted by

Pritam MISHRA

for the degree of Master Of Science

at The University of Lancaster

in September, 2020

Abstract– *Manipulated images and videos have become increasingly realistic due to the tremendous progress of deep convolutional neural networks (CNNs). While technically intriguing, such progress raises a number of social concerns related to the advent and spread of fake information. Such concerns necessitate the introduction of robust and reliable methods for fake image and video detection [1]. In this project we investigated some of the state of the art detection methodologies including pre-trained CNNs (VGG16, ResNet50, Xception, InceptionResNetV2), Baseline CNN from scratch, CNN-LSTM architecture (Several variations - sequential, Integrated approach, altering pre-trained models) , Domain Adapted GAN (Discriminator), 3DCNN (A variation of C3D architecture) to evaluate and compare their performance on Kaggle deep-fake detection challenge data-set. The findings from this project indicates that performance of these state of the art methodologies rely on not only more computing resources, longer training time, accurate face detection, video compression rate, image noise, video resolution but also huge number of training examples to form a comprehensive training data distribution to discern between fake and authentic videos.*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing sophistication of mobile camera technology and the ever-growing reach of social media and media sharing portals have made the creation and propagation of digital videos more convenient than ever before[2]. With the rise of artificial intelligence (AI) and deep learning techniques, fake digital contents have proliferated in recent years. Fake footage, images, audios, and videos (known as deepfakes) can be a scary and dangerous phenomenon and can have the potential of altering the truth and eroding trust by giving false reality[3]. Deepfake (a portmanteau of "deep learning" and "fake") is a video manipulation technique that allows to generate synthetic media from a target video in which the target actor is being replaced by the likeliness of the actor in the source video/image. The number of fake videos and their degree of realism was earlier limited by lack of sophisticated editing tool, robust algorithms, high domain expertise, expensive computing power and high model training time. However, these challenges have been mitigated in recent years with the introduction of highly sophisticated deepfake algorithms and their availability as easy to use(without any programming knowledge or domain expertise) softwares on the internet, high-throughput computing resources through GPUs or cloud based solutions and availability of pre-trained models that do not require high computational power for training time. Thus, lately Deepfakes have garnered widespread attention for their uses in celebrity pornographic videos, revenge porn, fake news, hoaxes, and financial fraud which elicited responses from both industry and government to detect and limit their use[4].

Digital impersonation is increasingly realistic and convincing. Deepfake technology is the cutting-edge of that trend. It leverages machine-learning algorithms to insert faces and voices into video and audio recordings of actual people and enables the creation of realistic impersonations out of digital whole cloth. The end result is realistic-looking video or audio making it appear that someone said or did something [5].

Doctored imagery is neither new nor rare. Innocuous doctoring of images— such as tweaks to lighting or the application of a filter to improve image quality—is ubiquitous. Tools like Photoshop enable images to be tweaked in both superficial and substantive ways [6]. The field of digital forensics has been grappling with the challenge of detecting digital alterations for some time. Generally, forensic techniques are automated and thus less dependent on the human eye to spot discrepancies [7]. While the detection of doctored audio and video was once fairly straightforward, [8] the emergence of generative technology capitalizing on machine learning promises to shift this balance. It enables the production of altered (or even wholly invented) images, videos, and audios that are more realistic and more difficult to debunk than they have been in the past. This technology involves the use of “Generative Adversarial Network” (GAN)- a neural network based architecture in which two neural networks contest with each other in the form of a zero-sum game, to generate realistic looking doctored images or videos. The neural network begins as a kind of tabula rasa featuring a nodal network controlled by a set of numerical standards set at random [9]. If the network processes a broad array

of training examples, it should be able to create increasingly accurate models [5]. In comparison to private and academic efforts to develop deepfake technology, less is currently known about governmental research. Given the possible utility of deepfake techniques for various government purposes— including the need to defend against hostile uses—it is a safe bet that state actors are conducting classified research in this area. At the least, it can be said with confidence that industry, academia, and governments have the motive, means, and opportunity to push this technology forward at a rapid clip.

Figure 1.1: Example of Deedpfake Video | On the left Christial Bell(deepfake) and on the right Williem Dafoe | Hover the link on the image to watch the video [10]

While convincing manipulations of digital images and videos have been demonstrated for several decades through the use of visual effects, recent advances in deep learning have led to a dramatic increase in the realism of fake content and the accessibility in which it can be created [11, 12, 13, 14, 15]. These so called AI-synthesized media (popularly referred to as deep fakes) fall into one of three categories: (1) face-swap, in which the face in a video is automatically replaced with another person's face. This type of technique has been used to insert famous actors into a variety of movie clips in which they never appeared, and used to create non-consensual pornography in which one person's likeliness in an original video is replaced with another person's likeliness; (2) lip-sync, in which a source video is modified so that the mouth region is consistent with an arbitrary audio recording. For instance, the actor and director Jordan Peele produced a particularly compelling example of such media where a video of President Obama is altered to say things like "President Trump is a total and complete dip-****."; and (3) puppet-master, in which a target person is animated (head movements, eye movements, facial expressions) by a performer sitting in front of a camera and acting out what they want their puppet to say and do [16].

While there are certainly entertaining and non-nefarious applications of these methods, concerns have been raised about a possible weaponization of such technologies [5]. For example, the past few years have seen a troubling rise in serious consequences of misinformation from violence against citizens to election tampering [17, 18, 19]. The addition of sophisticated and compelling fake videos may make misinformation campaigns even more dangerous [16].

1.1 Diffusion of Deepfakes: A Looming Challenge for Privacy and National Security

Harmful lies are nothing new nor rare. However the ability to distort reality has taken an exponential leap forward with "deepfake" technology recently. This capability makes it possible to create audio and video of real people doing and saying things they never said or did in the first place. Machine learning techniques are escalating the technology's sophistication, making deep fakes ever more realistic and increasingly resistant to deep fake forensic techniques for detection. Deepfake technology has characteristics that enable rapid and widespread diffusion of fake videos online, putting it into the hands of both sophisticated and unsophisticated actors of the society [5]. Emma González, a student who survived the horrific shooting at Marjory Stoneman Douglas High School in Parkland, Florida, in February 2018, gained prominence during the "March for Our Lives" protest in Washington, D.C. A powerful still image of Emma ripping up the bullseye target began to circulate all over the Internet. But soon someone generated a fake version in which the torn sheet turned into a copy of the Constitution of the United States. Emma has likely suffered some degree of anguish over these episodes; and other Parkland victims likely felt maligned and discredited by this incident. Imagine the implications of a deep fake video, released just the day before an election, making it appear that a candidate for office has made an inflammatory statement. Hence, falsified imagery, in short, has already exacted significant costs for individuals in this society but the situation is about to get much worse with time [5].

Figure 1.2: CheapFake image of Emma González that went viral on the internet [5]

Early academic research related to deepfakes dates back to the Video Rewrite program, published in 1997, which modified existing video footage of a person to depict that person mouthing the words contained in a different audio track[20].It was the first system that fully automated facial reanimation by sounds produced by a video's subject and the shape of the subject's face. Contemporary academic projects have focused on creating more realistic videos and on improving such techniques. The "Synthesizing Obama" program, published in 2017, modifies video footage of former president Barack Obama to depict him mouthing the words contained in a separate audio track[4][12]. Therefore, it has the potential to damage reputation, relationships and our online reality. Events that corroborate such implications of deepfakes include,

- In January 2019, Fox affiliate KCPQ aired a deepfake of Trump during his Oval Office address, mocking his appearance and skin color[21].
- During the 2020 Delhi Legislative Assembly election campaign, the Delhi Bharatiya Janata Party used similar technology to distribute a version of an English-language campaign advertisement by its leader, Manoj Tiwari, translated into Haryanvi to target Haryana voters[4].
- In April 2020, the Belgian branch of Extinction Rebellion published a deepfake video of Belgian Prime Minister Sophie Wilmès on Facebook.[22] The video promoted a possible link between deforestation and COVID-19.
- In June 2019, a downloadable Windows and Linux application called DeepNude was released which used generative adversarial networks to remove clothing from images of women[23].
- In June 2019, the United States House Intelligence Committee held hearings on the potential malicious use of deepfakes to sway elections[24].

These examples are none but the tip of a disturbing iceberg. Like sexualized deep fakes, imagery depicting non-sexual abuse or violence might also threaten, intimidate, and inflict psychological harm on the depicted victim. In a marketplace of ideas flooded with deepfake videos, truthful facts will have difficulty emerging from the scrum. [5].

1.2 Current Solutions:

Early attempts for detecting deepfake videos were based on handcrafted features obtained from artifacts and inconsistencies of the fake video synthesis process. Recent methods, on the other hand, applied deep learning to automatically extract salient and discerning features to detect deepfakes [25, 26]. Recent forensic techniques exploit low-level pixel artifacts introduced during synthesis [27, 28, 29, 30, 31, 32]. Although, these techniques detect a variety of fakes with relatively high accuracy, like other pixel based techniques they suffer from simple laundering counter-measures which can easily destroy the measured artifact (e.g., additive noise, re-compression, resizing). The use of a physiological signal, eye blinking, to detect deepfakes was proposed in [33] based on the observation that a person in deepfakes has a lot less frequent blinking than that in untampered videos in general. The spatio-temporal inconsistency between image frames in deepfake videos have also been explored by Sabir et al.[34] and Guera et al.[35] Yang et al. [36] proposed a detection method by observing the differences between 3D head poses and Nguyen et al. [37] proposed the use of capsule networks for detecting manipulated images and videos More discussion on deepfake generation and current forensic techniques to detect deepfakes can be found in chapter 2.

1.3 General Project Aim

The aim of this project is to develop an efficient deepfake detection system with less computational resources in a limited time (within 12 weeks of time) to ensure high quality deliverables in action. The key research questions that needs to be addressed in order to achieve general aim of this project can be summarized as follows,

- How to extract keyframes from videos for developing efficient deepfake detection system?
- How can we detect faces of subjects accurately in a video?

- How to detect pixel level artifacts in fake videos?
- How can we detect temporal inconsistency within image frames of a video?
- Can we develop a detection model with low computational resources and high accuracy?
- How can we solve data imbalance between fake and real videos for training and evaluation?
- Can pre-processing techniques improve the performance of deepfake detection models?
- Can we avoid model overfitting during training?

1.4 Objectives

Following the research questions, the objectives were formed to address them in this project,

1.4.1 KeyFrame Extraction and Face Detection

A video contains sequences of several image frames and therefore to train a model with every possible image frame can be computationally expensive and challenging. Therefore approaches to identify video key-frames can be implemented in this case to reduce the volume of data and training time of the models. Face-manipulation in a video is one of the consequential aspects of detecting deepfakes and therefore obtaining the profile face could help our model learn inherent features from the images of the video. However, open source face detection algorithms sometimes fails to capture subject's face properly due to object in motion and change of background brightness in contrast to the moving subject.
Highlights:

- Extraction of key-frames from the video to obtain images-frames that could summarize a video rather using every image frame in the video.
- Comparison between several face extraction packages(OPENCV, DLIB , MTCNN [38], RetinaFace[39]) and their efficiency in accurate face extraction to implement the optimal face extraction methodology.

1.4.2 Deep CNN for Deepfake Detection

Building an end to end system with a deep CNN model architecture from scratch requires an enormous amount of time considering the training time of the model on convolution layers and full-connected layers, for training huge number of parameters. The evaluation of any model could only be possible after the training time which is challenging especially when so many experiments need to be conducted within 12 weeks of time. Therefore, the baseline model for this end to end system will not contain hundreds of CNN layers rather less than 20 layers to evaluate performance against other models. While with Transfer Learning, pre-trained models with fifty or even hundreds of layers can be operated to extract low level features from image frames followed by training on fully connected layers to evaluate deep CNN architecture in comparison to baseline model.
Highlights:

- Implementation of baseline model with Convolutional Neural Network from scratch.
- Implementation of several pre-trained CNN models (ResNet [40],VGG [41],Xception [42]) with fully-connected layers to evaluate performance on kaggle dataset.

- Choice of models that are computationally less expensive yet more efficient.
- comparing pre-processing techniques for improvement in model performance.
- Organised data selection and weighted average of performance metrics to avoid class imbalance.
- Use of regularization techniques to avoid model overfitting during training.

1.4.3 Capturing Spatio-Temporal Dependency with Video Classification Models

Video classification techniques require not only training and evaluating image frames extracted from a video rather the temporal dependency of the key frames can provide additional information regarding the sequence of the image frames. Therefore, sequence to sequence models along with a time-distributed CNN or other video classification approaches consider all sequential key-frames of each video as a single observation rather than considering each key-frame of videos as a single observation of the dataset.
Highlights:

- Implementation of video classification approaches(time-distributed CNN with RNN [35], 3D CNN [1] etc.) to capture temporal sequence of video key-frames.
- Evaluating performance of video classification models with pre-processing, regularization and weighted average of performance metrics in comparison to previous proposed approaches.

1.5 Report Overview

The main contributions of this project is summarized as follows. In Chapter 2, an overview of deepfake video generation and deepfake video detection methodologies are presented along with an in depth overview of the research that was conducted in understanding different architectures of neural network in deep learning. Chapter 3 describes the methodologies followed to implement an end-to-end deepfake video detection system on Kaggle deepfake challenge dataset. Chapter 4 covers the analysis and evaluation of proposed methodologies in comparison to popular approaches from research literature in detail. The interpretation of the results are discussed in chapter 5 along with limitations followed by Conclusions and future work in chapter 6.

Chapter 2

Related Work and Background

The creation of sophisticated fake videos has been largely relegated to Hollywood studios or state actors for a long time. Recent advances in deep learning, nevertheless, have made it significantly convenient to create sophisticated and compelling fake videos in no time. With relatively modest amounts of data and minimal computing power, an average person for example can create a fake video of a world leader confessing to illegal activities leading to a constitutional crisis or a military leader saying something racially insensitive leading to civil unrest in an area of military activity, or a corporate titan declaring that their profits are weak leading to global stock manipulation. Hence, the generation of deep fake videos pose a significant threat to our democracy, national security, and society. [16].

The first known attempt at trying to swap someone's face dates back to circa 1865.This was found in one of the iconic portraits of U.S. President Abraham Lincoln. The lithography, as presented in Figure 2.1, fuses President Abraham Lincoln's head on the body of Southern politician John Calhoun. After President Abraham Lincoln's assassination, demand for lithographies of him became so high that as a result engravings of his head on other bodies appeared almost overnight. [43][35].

Figure 2.1: Swapping of U.S. President Lincoln's head with politician John Calhoun's body in mid-19th century (left). Face Swapping of late-night TV hosts Jimmy Fallon and John Oliver (right) wth FakeApp. [35]

Recent advances [44, 45] in deepfake generation techniques (DcGAN, Autoencoders) have radically changed the playing field of image and video manipulation in the society. The democratization of modern deep learning tools such as Tensorflow [46] or a convenient API Keras [47] coupled with the open accessibility of the recent technical literature online and cheap access to compute infrastructure (cloud based free computing solution Google Colab, Kaggle) have propelled this paradigm shift. Convolutional Autoencoders [48, 49] and deep convolutional generative adversarial network (DcGAN) [50, 51] models have made tampering images and videos, which used to be reserved to highly-trained professionals with years of experience, a broadly accessible operation these days within reach of almost any average individual with a computer. Smartphone and desktop applications like FaceApp [52] and FakeApp [53] are built upon this progress of deep learning [35].

FaceApp automatically generates highly realistic transformations of faces in photographs that allows the user to change face hair style, gender, age and other attributes using a smartphone. While FakeApp is a desktop application that allows one to create what are now known as deepfake videos from real ones. Deepfake videos are doctored video clips which were first generated by a Reddit user, deepfake, who applied TensorFlow, image search engines, social media websites and public video footage to swap someone else's face into preexisting videos from online.[35].

Reenactment methods, like [54], are designed to transfer image facial expression of a subject from a source video to a target video. Face2Face [49], which was first introduced by Thies et al., is an advanced form of such face reenactment method. It performs a photorealistic facial reenactment of facial expression in real-time from a simple RGB-camera, as presented in Figure 2.2. The program requires only few minutes of prerecorded videos of the target person for a training sequence to reconstruct its facial model from the source. Then, at runtime, the program tracks both the expressions of the source and target actors video to render overlaying the target face with a morphed facial blend shape to fit the source facial expression [55].

Figure 2.2: Example of a Face2Face reenactment result from the demo video of the Face2Face paper[]

2.1 Digital Media Forensics

The field of digital media forensics aims to develop forensic techniques for the automated assessment of the integrity of an image or video for deepfake detection. Both feature-based [56, 57] and CNN-based [51, 58] integrity analysis methods have been explored in the literature to detect pixel level inconsistency or temporal inconsistency in facial features. For video-based digital forensics, the majority of the proposed solutions try to detect computationally cheap manipulations, such as dropped or duplicated frames [59] within a video or copy-move manipulations [60] within image frames. Techniques that detect face-based manipulations include methods that discerns the data distribution of computer generated faces from authentic ones such as Conotter

et al. [61] or Rahmouni et al. [62]. In biometry, Raghavendra et al. [63] recently proposed a deepfake detection methodology to detect morphed faces with two pre-trained deep Convolution Neural Networks and Zhou et al. [64] proposed detection of two different face swapping manipulations with the help of a two stream network. The use of a physiological signal such as eye blinking, to detect deepfakes was proposed in [33] based on the observation that a person in a deepfake video has a lot less frequent blinking compared to untampered videos. Recently, Nguyen et al. [37] proposed the use of capsule networks for detecting doctored images and videos with high accuracy. Deepfake detection methods mostly rely on the pixel level or spatio-temporal artifacts or inconsistency of intrinsic features between fake and real images or videos. Yang et al. [36] proposed an unique detection methodology by observing the differences between 3D head poses of subjects in real and fake videos comparing head orientation and position, which are then estimated based on 68 facial landmarks of the central face region of the subjects in the video. The 3D head poses were examined because they discovered a shortcoming in the deepfake face generation pipeline. The extracted features from these videos were then fed into an SVM classifier to obtain the detection results.

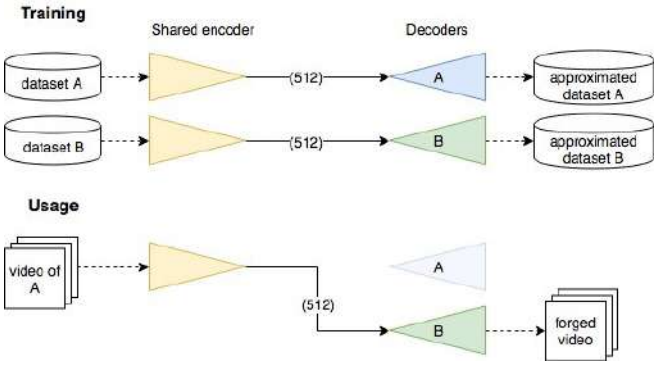
2.2 Face-based Video Manipulation Methods

Several methodologies that target face-based manipulations in video sequences have been proposed since the 1990s [20, 65]. Thies et al. demonstrated the first real-time expression transfer for faces from source to a target video and later proposed Face2Face [49]. Several face image synthesis techniques with deep learning have also been extensively proposed as explored and surveyed by Lu et al. [66]. Deep Convolutional Generative adversarial networks (DcGANs) were used for aging alterations to faces in images or videos [50], or to alter face attributes such as skin color [67]. Deep feature interpolation [68] shows remarkable results in altering face attributes such as age, facial hair or mouth expressions, which has been quite popular on social media lately. Similar results on attribute interpolations has also been achieved by Lample et al. [69].

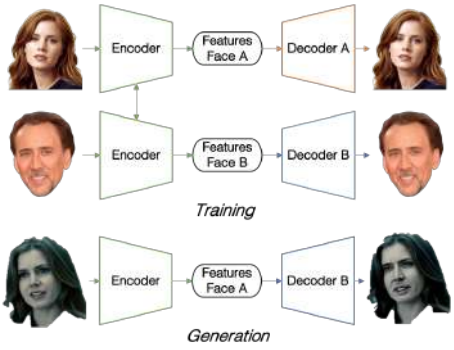
2.3 Exposing Methodology of DeepFake Video Generation

Deepfake is a deep learning methodology which aims to replace the face of a targeted person by the face of someone else in a video with the help of generation techniques such as Autoencoders or DcGAN. It first appeared in autumn 2017 as a script which was operated to generate face-swapped adult contents online. Afterwards, this methodology was improved by a small community to notably create a tempting user-friendly application accessible to anyone online called FakeApp [53] [55]. In this section, the methodology for generating DeepFake videos on FakeApp is explained in details for a deep understanding of the concepts and technology behind this.

The well known fact about deep learning techniques suggests these techniques were really successful to enhance the performance of image compression. In fact, the autoencoders have been applied especially towards dimensionality reduction, compact representations of images, and generative models learning [70]. Thus, autoencoders were able to extract more compressed representations of images with a minimized loss function and were expected to achieve better compression performance than existing image compression standards in case of digital image processing. The compressed image representations or latent vectors that current Convolutional Autoencoders learn through their encoder-decoder architecture are the first cornerstone behind the face-swapping capabilities of FakeApp [35]. The core idea lies in training of two convolutional autoencoders in parallel. The architecture of these autoencoders can vary according to the output size, desired training time, expected quality and available resources for training. Traditionally, an auto-encoder designates the



(a) Training and Generation of DeepFakes



(b) Training and Generation of DeepFakes with Example

Figure 2.3: Diving deep into the architecture of generating DeepFakes with FakeApp. (a) Top: the training parts with the shared encoder in yellow. Bottom: the usage part where images of A are decoded with the decoder of B [55]. (b). The faces of Actors Amy Adams and Nicolas Cage were trained with shared encoder (Top) and Finally face of Amy Adams has been swapped by Nicolas Cage at the time of Generation using Decoder B (Bottom) [35].

chaining of an encoder network followed by a decoder network. The purpose of the encoder is to perform a dimension reduction by encoding the data from the input layer into a reduced number of variables to produce latent variables. The goal of the decoder is then to utilize the latent variables to infer an approximation of the original input. The optimization of the entire autoencoder network is done by comparing the input and its generated approximation from decoder and penalizing the difference between the two, typically using a L^2 distance. In the case of the Deepfake technique, the original auto-encoder is fed with images of resolution 64 x 64 x 3 = 12, 288 variables, encodes those images on 1024 latent variables and then decoder uses inference or upsampling to generate images with the same size as the input [55].

2.3.1 Training and Video Generation

The process to generate Deepfake videos is to gather aligned faces of two different people A and B, then to train an auto-encoder E_A to reconstruct the faces of A from the dataset of facial images of A, and an auto-encoder E_B to reconstruct the faces of B from the dataset of facial images of B [55]. It is also important to note that if we train two autoencoders separately, they will be incompatible with each other. If two autoencoders are trained separately on different sets of faces, their latent spaces and representations will be different. This means that each decoder is only able to decode a single kind of latent representations which it has learnt during the training phase. This can be overcome by forcing the two set of autoencoders to share the weights for the encoder networks, yet using two different decoders. Nevertheless, all latent faces are produced by the same encoder which forces the encoder itself to identify common features in both faces. This can be easily accomplished due to the natural set of shared traits of all human faces (e.g. number and position of eyes, nose etc.) [35]. Once the optimization is done, any image containing a face of A can be encoded through this shared encoder but decoded with decoder of E_B [55]. This process is repeated for every image frames in the video to produce realistic looking deepfake videos. This principle is illustrated in Figure 2.3(a) and 2.3(b).

2.3.2 Vulnerabilities

In order to generate realistic looking deepfake videos and make training process of the autoencoders easier, the easiest face swap would have both the original face and target face under similar viewing and illumination

conditions [35]. However, this is usually not the case in practise.

- Multiple camera views, differences in lightning conditions or simply the use of different video codecs makes it difficult for autencoders to produce realistic face swaps under all conditions. This usually leads to swapped faces that are visually inconsistent with the rest of the scene [35]. This frame level inconsistency has been exploited in some of our proposed methodologies.
- Once the autoencoders are trained, in order to generate a fake video, face swapping operation needs to be performed for each of the frames of the target video. Nevertheless, to perform this frame-level operation , first a face detection model needs to accurately extract only the facial regions of the subjects that will be passed to the trained autoencoders. This is usually a second source of scene inconsistency between the swapped face and the reset of the scene. Since the encoder is not aware of the skin or other scene information it is very common to have boundary effects due to a seamed fusion between the new face and the rest of the frame [35].
- Since the autoencoder is operated frame-by-frame, it is completely unaware of any previous generated face that it may have created. This lack of temporal awareness is the source of multiple anomalies. The most prominent is an inconsistent choice of illuminants between scenes with frames, which leads to a flickering phenomenon in the face region common to the majority of fake videos. Although this phenomenon can be hard to appreciate to the naked eye in the best manually-tuned deepfake manipulations, it is easily captured by a pixel-level CNN feature extractor. The phenomenon of incorrect color constancy in CNN-generated videos is a well known and still open research problem in the computer vision field [71]. Hence, it is not surprising that an autoencoder trained with very constrained data fails to render illuminants correctly [35].

2.4 Exploiting Pixel level artifacts with CNN

There is a large body of literature on image and video forensic techniques to detect forgery [57]. However, because AI-synthesized content is a relatively new phenomena, there is a paucity of forensic techniques for specifically detecting fake videos generated from deep learning methodologies. One such example is based on the clever observation that deepfake videos generated with first generation of face-swap either didn't blink or didn't blink at the expected frequency [33]. This artifact has been detected due to the fact that training data used to synthesize faces typically did not depict the person with their eyes closed. Shortly after this forensic technique was made public, the next generation of deepafeke videos incorporated eye blinking into their methodology so now this technique is less effective [16]. This subsection investigates the approach that normally decomposes videos into image frames and explores visual artifacts on pixel intensities for each image frame to obtain discriminant features. These features could then be feed into a deep classifier to differentiate between fake and authentic videos[72].

Deepfake videos are normally created with limited resolutions, which require an affine face warping approach (i.e., scaling, rotation and shearing) to match the face proportions of the fake one with the original ones. Because of the resolution inconsistency between the warped face area and the surrounding context sometimes due to lighting conditions or subject movement, this process leaves artifacts that can be detected by CNN models such as VGG16 [41], ResNet50 [40] and ResNet152 [73]. A deep learning method to detect deepfakes based on these pixel level artifacts observed during the face warping step of the deepfake generation methodologies was proposed in [2]. From positive and negative examples, the regions of interest (RoI) were cropped by detecting the facial landmark points and fed to these networks. The aim is to expose the artifacts between

fake face area and surrounding area of the edges of a face. The regions of interest (ROI) are chosen as the rectangle areas that contains both the face and surrounding areas. Specifically, the regions of interest are determined by face detection algorithms looking for facial landmarks, as $[y_0 - \widehat{y}_0, x_0 - \widehat{x}_0, y_1 + \widehat{y}_1, x_1 + \widehat{x}_1]$, where y_0, x_0, y_1, x_1 denotes the minimum bounding box b which can cover all face landmarks excluding the outline of the cheek. The variables $\widehat{y}_0, \widehat{x}_0, \widehat{y}_1, \widehat{x}_1$ are all random values between $[0, \frac{h}{5}]$ and $[0, \frac{w}{8}]$, where h, w are height, width of b respectively. The RoIs are reshaped into 224 x 224 pixels to feed into the CNN models for training [2]. The authors of this proposed approach has evaluated their models on UADFV and DeepfakeTIMIT dataset and compared against the performance of other CNN models such as Two-stream CNN [64], MESO-4 [55], MesoInception-4 [55] and HeadPose [74]. The success of these pre-trained CNN models for deepfake video detection motivated this project to adopt transfer learning on pre-trained models(VGG16, ResNet50, Xception, InceptionResNetV2) into several proposed methodologies on kaggle deepfake dataset. In addition to these methodologies Meso-4 has also been explored in this project for deepfake detection.

2.5 Exploiting temporal inconsistency among image frames

The way deepfake videos are being generated, intra-frame inconsistencies and temporal inconsistencies between image frames of the video becomes quite evident due to the shortcoming in faceswap as discussed earlier. These spatio-temporal anomalies among image frames can be exploited to detect if a video under analysis is a doctored video or an authentic one. Based on the observation that temporal coherence is not enforced effectively in the synthesis process of deepfake videos as discussed earlier, Sabir et al. [34] leveraged the use of spatio-temporal features of video streams to detect deepfakes. Video manipulation is carried out on a frame-by-frame basis so that low level artifacts produced by face manipulation techniques are believed to further manifest themselves as temporal artifacts with inconsistencies across frames due to face movement or subject in motion. A recurrent convolutional model (RCN) was proposed based on the integration of the convolutional network DenseNet [75] and the gated recurrent unit cells [76] to exploit temporal discrepancies across frames [72] to investigate this hypothesis.

Likewise, Guera and Delp [35] highlighted that deepfake videos contain intra-frame inconsistencies and temporal inconsistencies between image frames present in the video. They then proposed the temporal aware pipeline method combining CNN with long short term memory (LSTM) for detecting deepfake videos. CNN is employed to extract frame-level features, which are then fed into the LSTM to capture temporal sequence between image features. A fully-connected network is finally used for classifying doctored videos from real ones based on the sequence of image features. The recent success (published in 2019) of these CNN-LSTM architectures for detecting deepfake videos as mentioned in this section, motivated this project to follow their footsteps and propose similar CNN-LSTM architecture for evaluating deepfakes. Very recently (June, 2020) Wang et al.[1] has exposed the performance of state of the art pre-trained 3DCNNs (3D ResNet, 3D ResNeXt, and I3D) for detecting temporal and visual inconsistency in deepfake videos. Hence, these methodologies were adopted in our proposed methodology to exploit these spatio-temporal inconsistency among image frames. The proposed methodology on CNN-LSTM and 3DCNN architectures have been explained with details in the next chapter.

Chapter 3

Methodology

This section will cover the operational design and implementation details of the proposed methodologies for each phase of the data pipeline to achieve the objectives described previously. This section will also provide background and reasoning behind the data representation, exploratory data analysis, proposed feature extraction approaches and model architectures implemented in this project. Furthermore, the section will delineate the development progression of the proposed deep neural network architectures with the help of the more advanced deep learning mechanics described in the background research section.

3.1 Data Pipeline

The major highlights of proposed DeppFake detection data pipeline can be summarized as follows,

- The data pipeline begins with analyzing metadata, which contains description of the video files. The exploratory data analysis on metadata and corresponding video files has been presented in detail on the next section. The selection of total 154 rows of data (77 real and 77 fake) from training dataset was performed at the early stages of the data pipeline as presented in Figure 3.1. Additionally, the corresponding video files were fetched from the training video repository by looking up the filenames of the data items or rows in metadata (154 of them in this case). Furthermore, all of the image frames from these video files were extracted with the help of OPENCV.
- The next stage of data pipeline consists of keyframe extraction by selecting image frames with maximum entropy from every image sequence as highlighted in Figure 3.1. More details about keyframe extraction can be found later in this chapter. Finally, these extracted keyframes were considered for face detection and extraction by several mainstream and state of the art face detection methodologies. Once the faces were extracted from the keyframes, the contrast of the images were boosted using OPENCV and the advantages of applying this technique has been evaluated in comparison to without contrast in chapter 4. The extracted faces from these keyframes for each video were saved into individual auto-generated repositories with name same as the filename of each video for further evaluation.
- The extracted faces which were saved into auto-generated repositories at second phase of the pipeline were manually evaluated to verify face extraction worked perfectly for all video files. Later, the face images from all 154 repositories corresponding to 154 video files were fetched and pre-processing(Normalization, Standardisation) of these images were performed in this stage. Finally, these images were trained and validated (70% data for training and 30% for validation) by our proposed deepfake detection models for final evaluation. The proposed methodology for each deepfake detection model can be found later in this chapter.

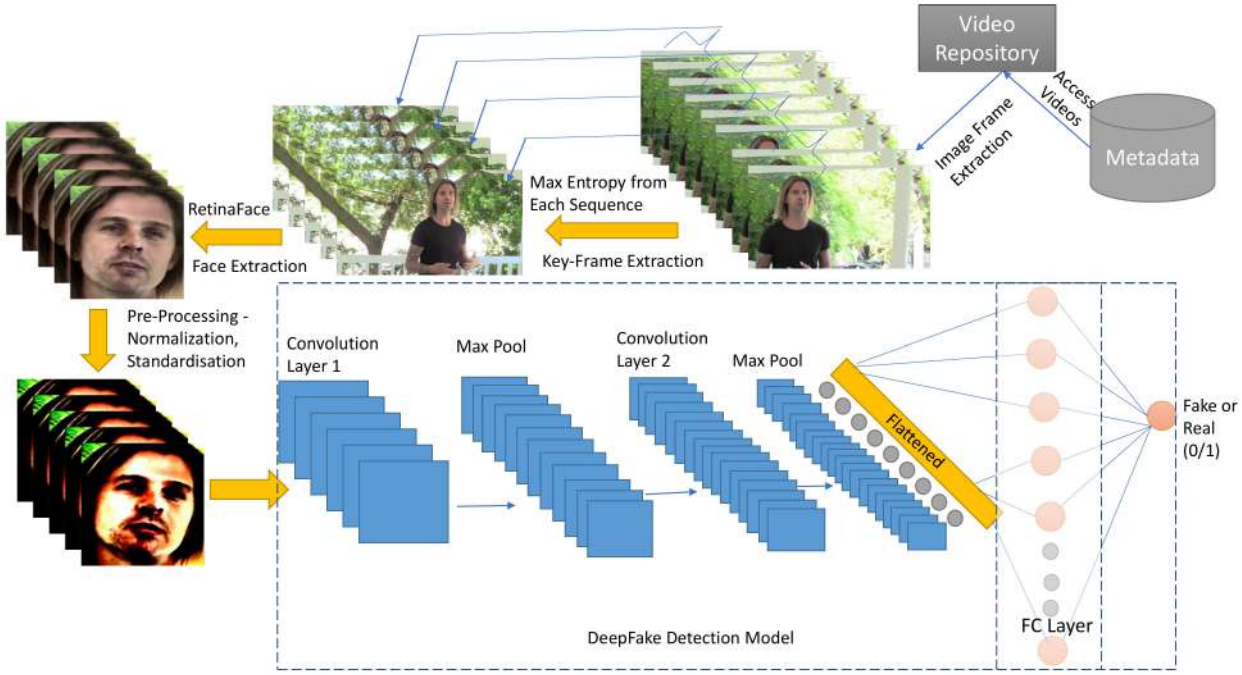


Figure 3.1: Data Pipeline of Proposed DeepFake Detection System

3.2 Data Description

In this project, dataset from Kaggle's deepfake detection challenge has been primarily trained and tested on our developed deepfake detection system. The dataset contains 800 samples of video files in total with fake and original videos for a cumulative size of more than 4 Gigabytes. Out of the 800 samples, 400 video files are kept for training and validation purposes while the other 400 video files are kept for test-set prediction. The training data also includes a JSON file such as metadata(metadata.JSON), which contains the description of each training samples or video files; whether it is fake or real, original or generated file, whether the file can be used for training-validation or test-set. If a video file (fake video) is generated from an original or real video and this original video is present in the dataset, "original" column contains the name of the source video file from which this video has been generated. Otherwise the value of "original" column is NULL for real videos which does not have any source file. The native data representation of the metadata is presented below with first row or observation of the dataset being highlighted for convenience.


```
{"aagfhgtpmv.mp4":{"label":"FAKE","split":"
train","original":"vudstovrck.mp4"},"aapnvo
gymq.mp4":{"label":"FAKE","split":"train","
original":"jdubbvfswz.mp4"},"abarnvbtwb.mp4
":{"label":"REAL","split":"train","original
":null},"abofeumbvv.mp4":{"label":"FAKE","s
plit":"train","original":"atvmxvwyns.mp4"},
"abqwwspghj.mp4":{"label":"FAKE","split":"t
rain","original":"qzimuostzz.mp4"},"acifjvz
```


Figure 3.2: Data Representation of Metadata with first row highlighted for reference

The metadata is read and transposed to transform the data representation in a tabular structure as presented below where the index of each observation represents the filename of the video. There are 400 observations or rows and 3 columns in metadata for training-validation.

Table 3.1: Transformed Data Representation of Metadata

	label	split	original
aagfhgtpmv.mp4	FAKE	train	vudstovrck.mp4
aapnvogymq.mp4	FAKE	train	jdubbvfswz.mp4
abarnvbtwb.mp4	REAL	train	None
abofeumbvv.mp4	FAKE	train	atvmxvwyns.mp4
abqwwspghj.mp4	FAKE	train	qzimuostzz.mp4

Missing Data

Let us check for any missing values in training-validation dataset of 400 samples.

Table 3.2: Missing Data within Metadata

	label	split	original
Total	0	0	77
Percent	0	0	19.25
Types	object	object	object

We obtained zero missing values for column "label" and column "split" while 77 missing values have been obtained for column "original" which corresponds to about 19.25% of the observations or rows. Nevertheless, it is also possible that all of the missing values in "original" column could be real videos for which the source file is NULL. Therefore, this hypothesis can be tested by passing only the observations for real videos and can be presented below,

Table 3.3: Percentage of Missing Data within real videos

	label	split	original
Total	0	0	77
Percent	0	0	100
Types	object	object	object

From Table 3.3, it can be observed all 77 missing values are real video files which does not have any source file and therefore value of the "original" column is NULL. Therefore, we can conclude there are no missing data in this dataset.

3.3 Exploratory Data Analysis

There are about 80.75% (323) videos in our training dataset to be fake while 19.25% (77) among 400 training samples are original videos. The distribution of real and fake videos can be presented below,-

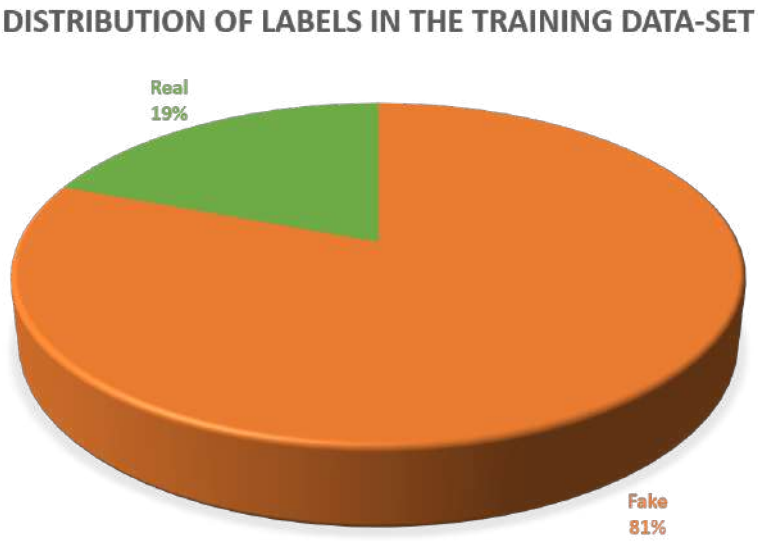


Figure 3.3: Distribution of labels within Metadata

It's possible that multiple fake videos can be obtained from same original or real video or all of the fake videos could have been doctored from unique original video files, therefore the unique values for each column can be checked to get additional information to confirm this theory.

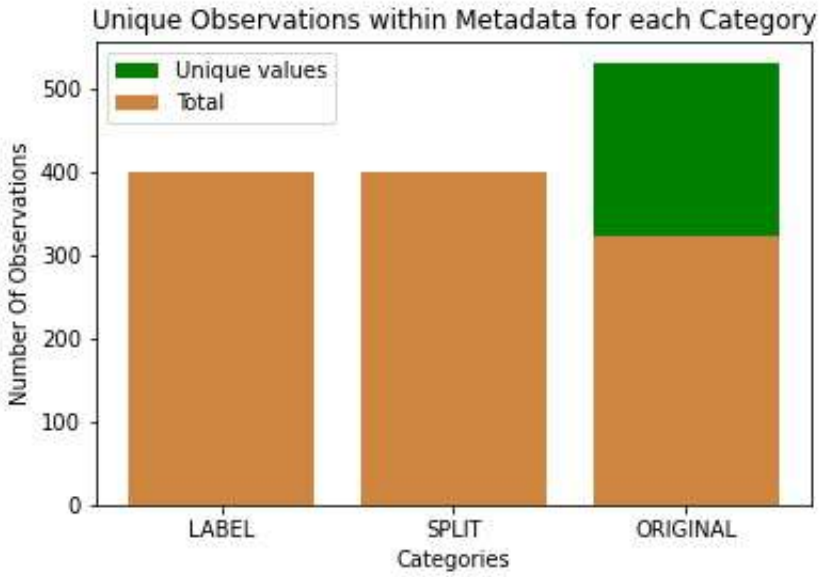


Figure 3.4: Unique Observations within Metadata for Each Category

From Figure 3.4, we can observe there are only two unique values for column LABEL which is 0 and 1, only one unique value for column SPLIT - "train" since we are using training dataset while there are 209 unique observations out 323 observations for column "Original", therefore we can conclude more than one fake videos have been generated from same source or original video in this dataset. Therefore, we can investigate which video source files from "Original" column have been considered most frequently to generate fake videos in this dataset.

Table 3.4: Most frequently referenced original video

	label	split	original
Total	400	400	323
Most frequent item	FAKE	train	atvmxvwyns.mp4
Frequence	323	400	6
Percent from total	80.75	100	1.858

From Table 3.4 it can be observed that the maximum frequency for an original file referenced to create a fake video in this dataset is 6 which corresponds to 1.85% of the total number of original video files (323). Hence, it can concluded not many fake videos have been doctored from the same original video file although we found 209 unique original files out of 323 fake videos created from these originals for this training dataset.

Now the Metadata has been explored in detail, let us randomly pick one row from metadata which contains a fake video and it's original or source video file to identify any visual differences between the two. The original video from which this fake video is generated, needs to be found in our training dataset since not all original videos could be found in our training dataset.

Table 3.5: Randomly chosen Fake video from Metadata

	label	split	original
eepezmygaq.mp4	FAKE	train	abarnvbtwb.mp4

Table 3.5 presents a randomly chosen row from metadata where both the fake video and it's corresponding original video are available in training dataset. The video files that corresponds to the file names (both fake and original) presented in table 3.5 can now be explored and compared for analysis. The resolution of all videos in the training dataset is 1920X1080 (full HD) and the time duration is 10 seconds. The total number of image frames for all videos are approximately 300. The videos presented in table 3.5 have 300 image frames with a resolution of 1920x1080 with all 3 color channels(RGB). Let us check one image frame out of 300 from both the fake video and it's corresponding original video to fathom any visual difference between the images.

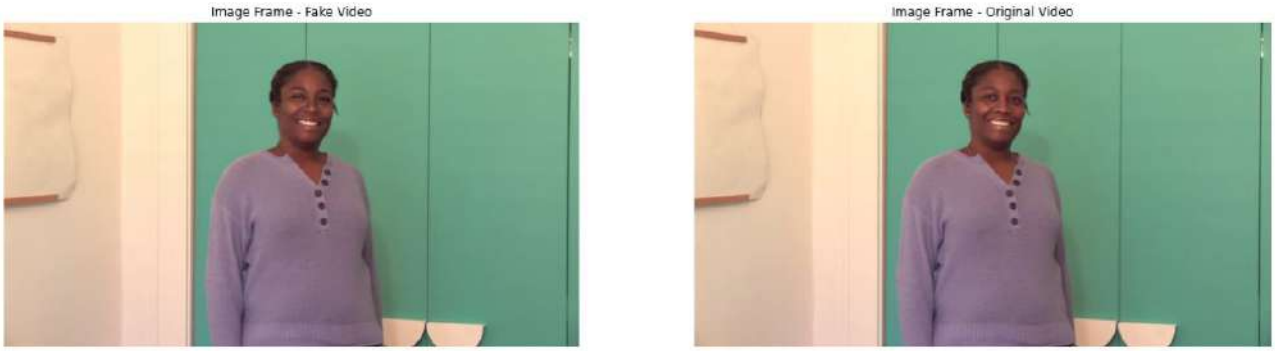


Figure 3.5: Frame number 65 presented from Real and Fake Videos

Looking at the these image frames from the videos it is really hard to distinguish the original image from it's doctored counterpart. If it is really hard for human eyes to distinguish the real image from the fake one, we could wonder how challenging task could this be for a deep learning model to discern these images. Let us take a look closely on the faces of these images and see if we could find any distinct pattern or visual characteristics which might help discerning the real image from the fake one.

Figure 3.6: Extracted Face from Frame number 65 presented from Real and Fake Videos

Figure 3.6 has been obtained by detecting face from images with state-of-the-art RetinaFace detection system, more details on face detection and proposed methodology for detecting face accurately for this dataset can be found in the next section. From the extracted faces we can observe some differences on the skin of our subject. For the fake image, the skin of the subject appears to be soft or little blurry which provides us the hint that softening of skin is an after effect of a doctored image. One possible explanation for softening the skin of the subject in doctored images could be the sharp differences among pixels between the real image and the fake face (overshadowing the real face) so that it does not become noticeable to the human eyes. Therefore, if we consider the entire image to develop our deep fake video detection system, the chances are really low for the model to discern such changes in those face pixels out of 1920x1080x3 number of pixels for each image. Nevertheless, if the model is developed based on the extracted faces from the image frames of the videos, the chances of detecting a fake video may increase substantially.

Digital images are subject to a wide variety of distortions during acquisition, processing, compression, storage, transmission and reproduction, any of which may result modification in visual quality. Objective image quality metrics can be classified according to the availability of an original (distortion-free) image, with which the distorted image is to be compared . structural similarity Index (SSIM) [77] compares local patterns of pixel intensities that have been normalized for luminance and contrast to discern between a distorted image and distortion free image. Let us take a look at the SSIM [77] scores among all the image frames of the original video mentioned in table 3.5.

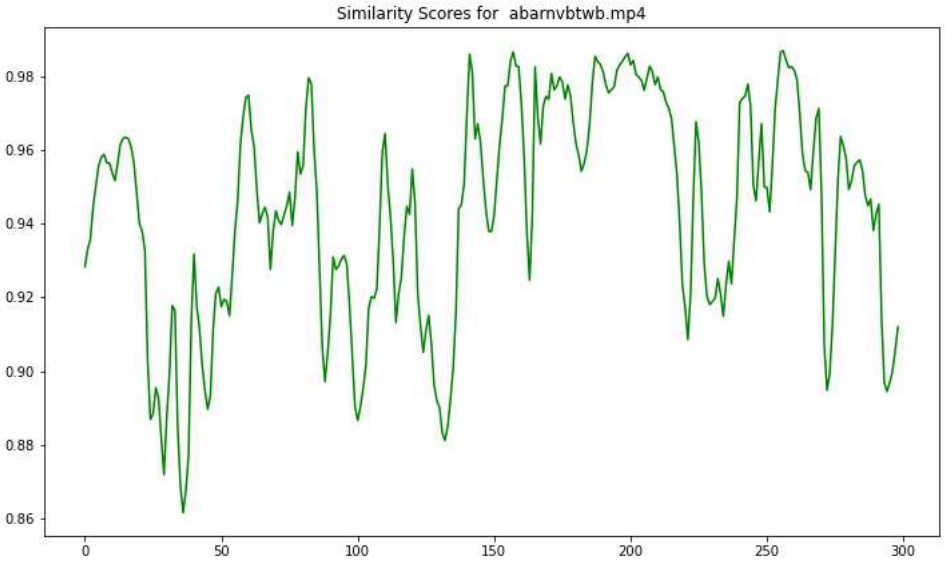


Figure 3.7: SSIM score for original video

From figure 3.7 we can observe there is high variation of SSIM for every 30 frames of the original video, this could be a result of the video being captured in 30 frames per second. As the scene in the video changes in every 30 frames for every second the structural similarity between the image frames keep changing. The variation within 30 frames could be a result of distortion in images due to it's compression algorithm. Let us observe the SSIM index for images frames in the corresponding fake video generated from this original video.

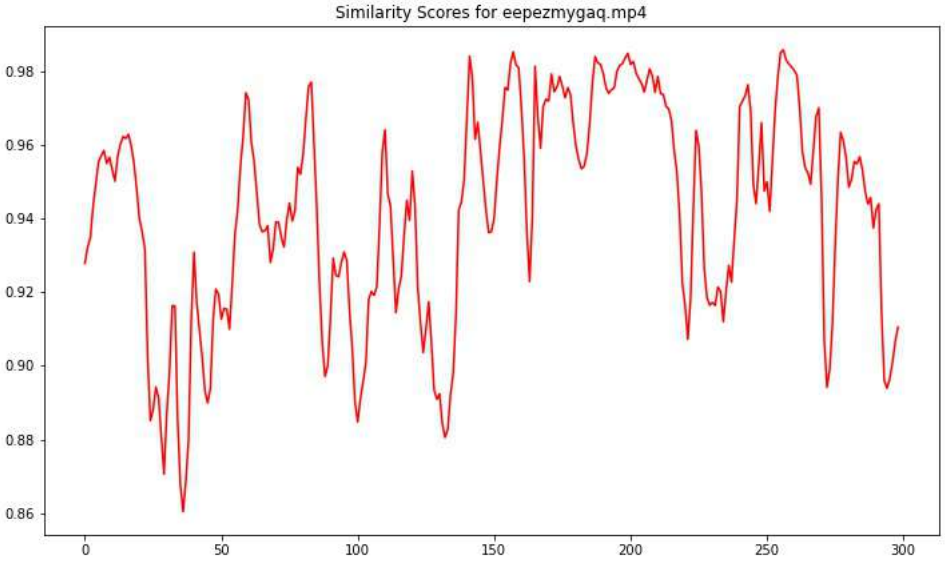


Figure 3.8: SSIM score for Fake video

From figure 3.8 we can observe a similar pattern of SSIM index for the fake video as the original video which has a high variation in every 30 frames. The variation of SSIM for every 30 frames might not always be

true if the subject and it's background is stationary while it's also possible for a fast moving subject in motion to have high variation of SSIM index within 30 frames on 1 sec interval of the video. Finally, let us compare SSIM index from both videos if we find any structural difference in image frames.

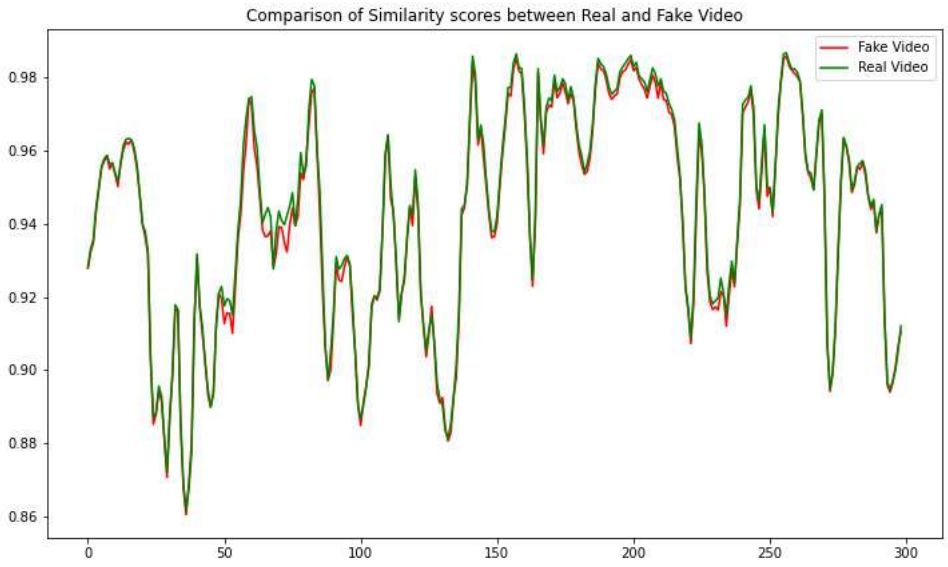


Figure 3.9: Comparison of SSIM score between original and fake video

Figure 3.9 presents a comparison of SSIM index between the images of original and fake video to inspect any distortion present in doctored images from real images for the nth image frame in both videos. We can observe some major distortions in image frames from fake video compared to it's original counterpart for image frames around 60, 75-80, 95, 240 and some minor distortions can also be found accross the other frames. We can conclude it's quite challenging to find structural difference between the doctored image and it's original counterpart due to noise or distortion with SSIM index. Nevertheless, this experiment could be performed on extracted faces to inspect if major distortion or perturbation can be found between the corresponding image frames of real and fake video.

3.4 Key-Frame Extraction

In Section 3.3, we discovered several salient characteristics of the video files in kaggle deepfake challenge dataset. One such characteristic that can have a huge impact on our deepfake detection system is the number of images frames present in each video. If we can recall from section 3.3, there were about 299 to 300 image frames on an average for each video with 1920x1080 resolution including 3 channels (R-G-B). If we take a closer look, the time interval of each video is only 10 seconds and therefore each video had almost 30 image frames per second. If we consider the full resolution of each images(including 3 channels) with all 299-300 images frames per video to develop our deepfake system, it would be not only computationally very expensive for this project but also the redundant image frames present in the videos will overfit our model as a result. In order to resolve this problem, we propose a key-frame extraction methodology to select only the keyframes with maximum information for each sequential interval of the videos.

3.4.1 Proposed Methodology for Key-Frame Extraction

In order to select key-frames from the video initially we proposed to randomly select indexes of key-frames from a Gaussian Distribution. The number of key-frames also required to be a constant number for each video to maintain consistency and avoid bias or overfitting our model. For example, if Video 3 produces 40 keyframes while video 1 produces 8 keyframes even if our model demonstrates high accuracy this would be biased towards image frames of video 3 in this example rather than generalizing the entire dataset. Based on the review of Kaggle notebooks, the number of key-frames for each video has been devised to be 17 [78]. The fellow kaggler had explained in his notebook that more number of keyframes does not improve the performance of the models for this dataset rather it becomes computationally expensive[78].

In several research Literature[79, 80] , Difference between histogram or Entropy among image frames has been implemented to extract key-frames from videos. However, one challenge of using difference in Entropy or histogram is the difference between two consecutive image frames is usually computed across all the image frames of the video. Then, based on a threshold value (for example, 0.3 if entropy difference is scaled between 0 to 1) these keyframes are selected from the video. The Entropy of an image frame can be defined as sum of product of the log of the inverse probability of appearance $p_f(k)$ with probability of appearance $p_f(k)$ [79]. This can be expressed as,

$$Entropy = -\sum_{k=0}^{max} \log \left[\frac{p_f(k)}{p_f(k)} \right] \tag{3.1}$$

Let us consider a typical frame from a video sequence, where the number of grey level is quantized to 256. Let $h_f(k)$ be the histogram of frame f and k the grey level, such that $0 < k < 2^{b-1}$, where b is the number of bits, in which the image quantization levels can be represented [79]. If the video frame is of the class M rows N columns, then the probability of appearance of this grey level $p_f(k)$ can be expressed as,

$$p_f(k) = \frac{h_f(k)}{M.N} \tag{3.2}$$

The big challenges for using this approach could be each video will result different number of key-frames based on threshold values but in this project we require 17 key-frames from each video. In addition to this challenge, it is also possible that all of the key-frames in a video can be selected from first 150 image frames of the video for example. This is possible if the subject and lighting conditions do not change much in first 150 frames, this has been the case in most of the videos in our dataset. Therefore, if the key-frames are only selected from one part of the video it might not generalize well sequentially the entire video.

Hence, the proposed approach introduces a methodology to resolve both of these challenges stated above. The proposed methodology suggests, all the images frames to be divided into 17 sequences of image frames. From each image sequence, the image-frame with maximum Entropy is selected as the key-frame of that shot or sequence and this process is repeated for all 17 sequences. This methodology is tested on our baseline reference model against the key-frame extraction methodology that selects 17 key-frames from random Gaussian distribution. The proposed methodology empirically shown significant improvement in performance when compared to random key-frame extraction both of which were tested on our baseline model with a batch-size of 64. The evaluation of the proposed methodology will be discussed with more details on the next chapter.

3.5 Face Detection and Image Extraction

In section 3.3, we discussed the key differences between a real image frame and a doctored image frame in videos. The highlight of our discussion was focused on face of our subjects in the videos where we could notice some minor to noticeable differences in subject's face through naked eyes while all other part of the image remains unaltered. Therefore, our proposed methodology takes into consideration the most consequential part of the image frames (i.e. face of subjects) in the video to develop an end to end automatic deepfake video detection system. In this section, we present the evaluation of most popular face extraction algorithms on kaggle deepfake challenge dataset which were implemented in this project to obtain the optimal face extraction methodology. Finally, we compare these face extraction algorithms with our proposed methodology (obtained 100% accuracy in face extraction) in terms of accuracy and time on Kaggle deepfake challenge dataset.

3.5.1 Evaluation Framework for Face Extraction

To evaluate the best possible face extraction algorithm for this project, a subset of the training videos has been taken under consideration for performance evaluation. A training dataset that contains 400 videos with each of them having approximately 299-300 image frames, is computationally expensive and exhaustive for this evaluation framework. Hence, a subset of 5 videos have been handpicked from training-validation dataset manually to ensure the versatility of face extraction algorithms for handling different lighting conditions portraying complex scenes and image dynamics. To provide a background of the videos selected for our evaluation framework, we present the details of these 5 videos with the help of an image frame from each of these videos as follows,

If we inspect the videos individually, we could discover that video 1 which has presented in Figure 3.10 (a) portrays a man walking from left to right in front of the camera probably in a video blog. Nevertheless, the background light changes drastically over time which has been depicted in figure 3.10 .(a), therefore edge detection is very challenging in some frames of this video which is why this video has been considered in this evaluation framework. Video 2 delineates two persons talking to each other in artificial light while their shadows are also visible. The subjects have different skin colors as well, this scenario can be challenging for the face extraction algorithm as only one side of the faces of these subjects are visible in this video. Video 3 does portray some complexity in terms of image dynamics , it has been shot in artificial lighting conditions and only one side of the face is shown in the video which has been shifted from left to right with different facial expressions. Video 4 is possibly the most challenging video among all as the video has been captured in extreme low light. It is really challenging even for human eyes to detect a face in video 4, let alone this needs to be detected by face extraction algorithm for all 299 frames with very high accuracy. The final video (video 5) selected for this evaluation is not very challenging compared to first four and therefore we can estimate if the extraction algorithm is versatile enough to produce high accuracy in any circumstances.

3.5.2 Haar-Cascade for Face Extraction

Haar Cascade classifier is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses "integral image" concepts to compute the features detected. Haar Cascades uses the Ada-boost learning algorithm which selects a small number of important features from a large set to provide an efficient result of classifiers then use cascading techniques to detect the face in an image. Haar cascade classifier is based on the Viola-Jones detection algorithm which is trained given some input faces and non-faces and training a classifier that identifies a face [81].

(a) Frame 30: Video 1: "aagfhgtpmv.mp4"

(b) Frame 30: Video 2: "aapnvogymq.mp4"

(c) Frame 30: Video 3: "abarnvbtwb.mp4"

(d) Frame 30: Video 4: "abarnvbtwb.mp4"

(e) Frame 30: Video 5: "abqwwspghj.mp4"

Figure 3.10: Videos Tested for Face Extraction Evaluation

Haar features:

A Haar-Feature is just like a kernel in CNN, except that in a CNN, the values of the kernel are determined by training, while a Haar-Feature is manually determined. Each feature results in a single value which is calculated by subtracting the sum of pixels under a white rectangle from the sum of pixels under the black rectangle [81].

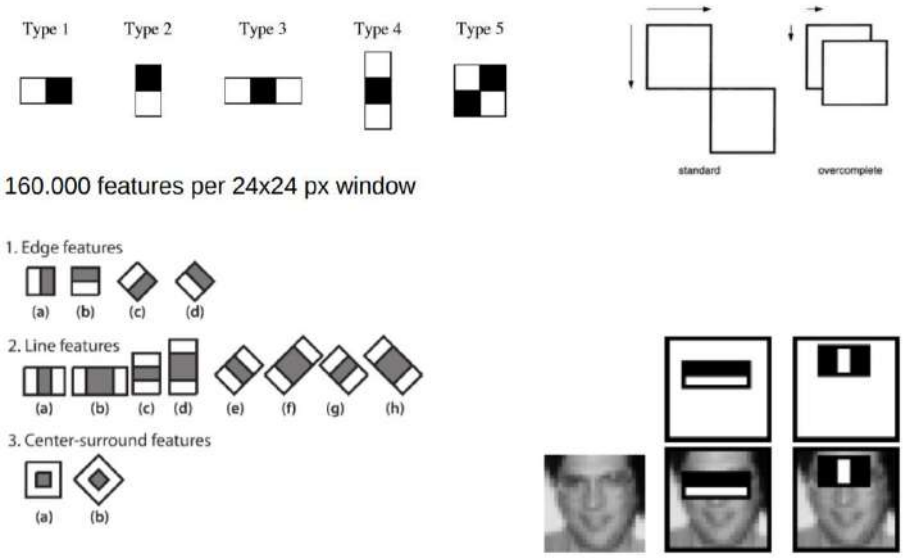


Figure 3.11: Haar Features [82]

Viola-Jones detection algorithm uses 24x24 as base window size and calculates the above features all over the image shifting by 1 PX. If all possible parameters of the haar features like position, scale, and type are considered this results in about 160,000+ features in total. So a huge set of features for every 24x24 PX needs to be evaluated in this case. To avoid this, redundant features are removed and only useful features are being selected with the help of AdaBoost classifier [81].

Rejection-Cascade with AdaBoost:

The Viola-Jones' detector uses AdaBoost, called a rejection cascade, which is a series of nodes, with each node being a definite multi-tree AdaBoosted classifier. Figure 3.12 shows the Viola-Jones' rejection cascade, composed of many boosted classifier groups of decision trees trained on the features from faces and non-faces or other training objects to be detected. In case of faces, almost all (99.9%) of the faces are found, but many non-faces (about 50%) are wrongly regarded as positive ones. This is feasible, because a sufficiently large number of such nodes will yield a false positive [82].

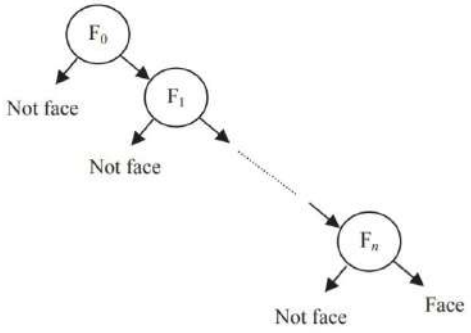


Figure 3.12: Rejection-Cascade with AdaBoost [82]

Evaluation:

As discussed in section 3.3.1, the evaluation of this face detection algorithm has been tested on the five videos selected, each of which contains 299 image frames and the duration of each video has been 10 seconds. The faces from the videos have been extracted with the face pixel values obtained by the algorithm with a padding of 25 pixels on each side of the rectangle to obtain the edges of the faces for our final prediction model to determine whether the video has been fake or not. The results for this evaluation can be presented as follows,

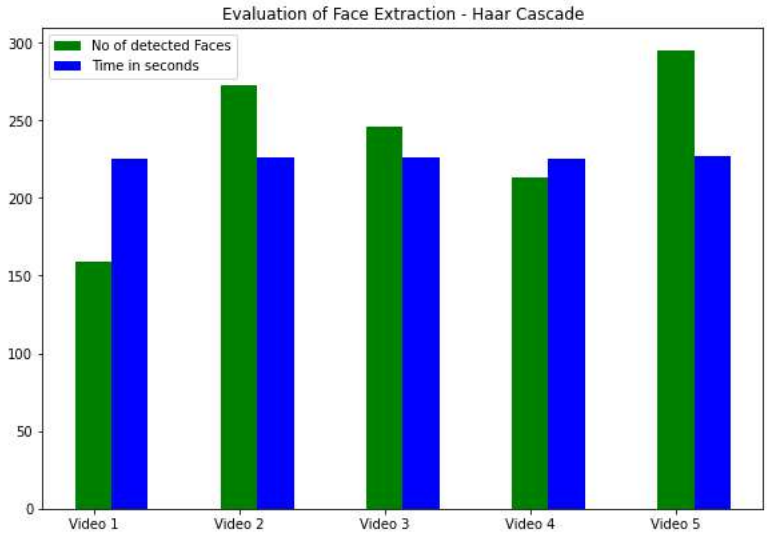
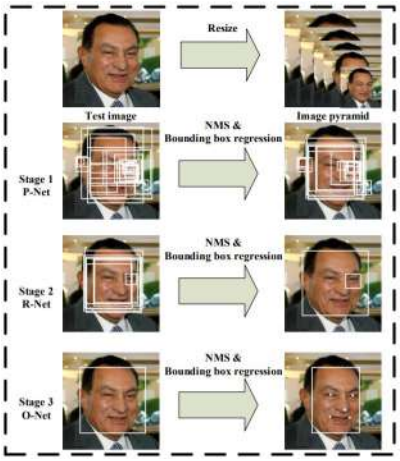
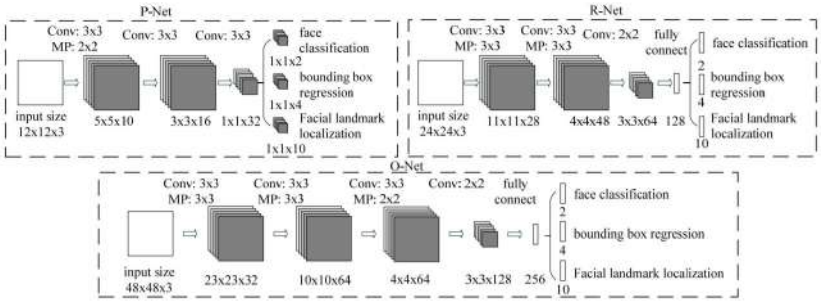


Figure 3.13: Evaluation of Haar Cascade for Face Extraction

It can be observed from Figure 3.12, Haar cascade performed poorly on video 1 where the background light changes a lot in contrast to our subject and the subject is also in motion. For video 5, which is not very complex scenario this classifier has manager to extract 295 faces out of 299 image frames correctly, while it has performed poorly for other videos in comparison. Therefore, we can not apply this methodology for our deepfake detection system since it would require a lot more accuracy and versatility from face extraction methodology to accurately classify faces in the first place which would help significantly in later stages to classify whether face images might be doctored or real.



(a) Sequential Stages in MTCNN Pipeline



(b) Architecture of MTCNN

Figure 3.14: Pipeline of MTCNN cascaded framework that includes three-stage multi-task deep convolutional networks [38]

3.5.3 MTCNN for Face Extraction

Multi-task Cascaded Convolutional Networks or in short MTCNN is a deep cascaded multi-task framework which exploits the inherent correlation between face detection and alignment to boost up their performance. This framework leverages a cascaded architecture with three stages of carefully designed deep convolutional networks to predict face and landmark location in a coarse-to-fine manner. This face detection methodology achieves superior accuracy over the state-of-the-art techniques on the challenging FDDB and WIDER FACE benchmarks for face detection, and AFLW benchmark for face alignment, while keeps real time performance. This methodology consists of three stages. In the first stage, it produces candidate windows quickly through a shallow CNN. Then, it refines the windows to reject a large number of non-faces windows through a more complex CNN. Finally, it uses a more powerful CNN to refine the result and output facial landmarks positions. The model is called a multi-task network because each of the three models in the cascade (P-Net, R-Net and O-Net) are trained on 3 different tasks such as face classification, bounding box regression and facial landmark localization [38]. The stages of the methodology has been presented in figure 3.14.

Stage 1:

Stage 1 includes a fully convolutional network called Proposal Network (P-Net), to obtain the candidate facial windows and their bounding box regression vectors. Then candidates are calibrated based on the estimated bounding box regression vectors. After this non-maximum suppression (NMS) is applied to merge highly overlapped candidates.

Stage 2:

All candidates are fed to another CNN, called Refine Network (R-Net), which further rejects a large number of false candidates, performs calibration with bounding box regression, and conducts NMS.

Stage 3:

This stage is similar to the second stage, but in this stage, the aim is to identify face regions with more supervision. In particular, the network will output five facial landmarks' positions [38].

Evaluation:

The same evaluation framework has been considered to evaluate the performance of this face detection methodology. The results have been presented in terms of number of image frames with faces accurately extracted from these videos. This has been observed that overall the accuracy of MTCNN is considerable high compared to Haar Cascades, however the time elapsed for the detection is also considerably high. The evaluation is presented as follows,

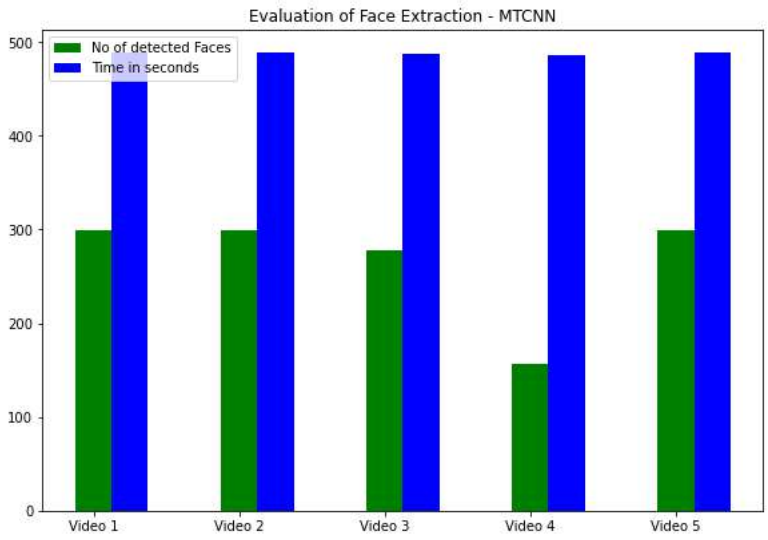


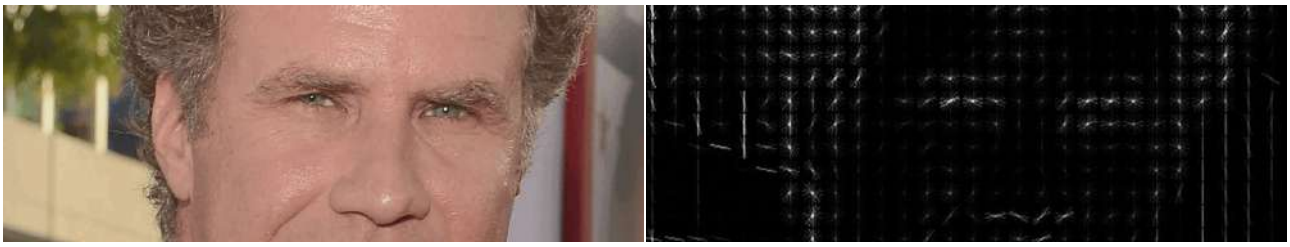
Figure 3.15: Evaluation of MTCNN for Face Extraction

From figure 3.14, we can observe that MTCNN has obtained 100% accuracy in terms of face extraction for video1, video 2 and video 5. While the performance on video 3 is still manageable , it performed very poorly to extract faces from video 4. MTCNN has managed to extract only 157 image frames (50% accuracy) with faces from video 4. As we could recall that video 4 has been captured in very low light and therefore MTCNN has failed to detect the faces in this video. Therefore, we can conclude this project requires a more versatile yet accurate face extraction algorithm to efficiently capture the faces from these videos.

3.5.4 DLIB for Face Extraction

The face detection methodology is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. This is one of the mainstream face detection methodology that could be found in research literature [83, 84]. According to this face detection methodology, for every single pixel, the goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. An arrow is drawn to show in which direction the pixel is getting darker. If this process is repeated for every pixel, pixels are replaced by these arrows which is considered as gradients that show the flow from light to dark across the entire image. Saving the gradient for every single pixel gives us way too much detail and therefore the image is fragmented into 16x16 pixels each. In each square, how many gradients point in each major direction (how many point up, point up-right, point right, etc...) is counted and that square in the image is replaced with the arrow directions that were the strongest. The end result is presented in figure 3.16,

To find faces in the HOG image depicted in figure 3.15 (b), the detection algorithm finds the part of the image that looks the most similar to a known HOG pattern that was extracted from several other training faces.



(a) Original Image **(b)** HOG Representation of the Image

Figure 3.16: Finding Face representation with HOG [85]

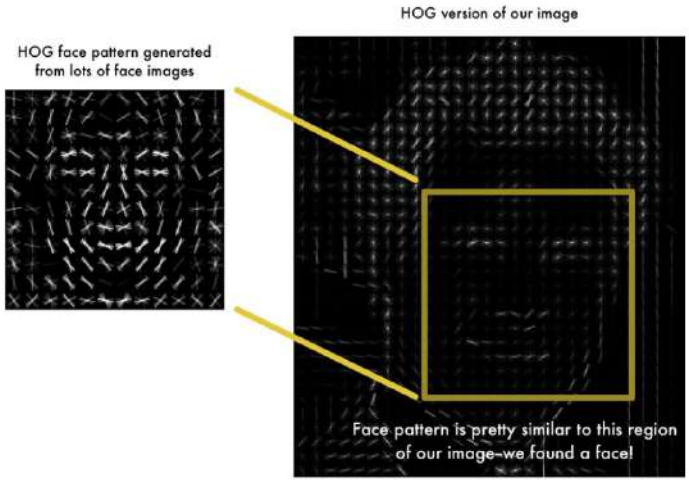


Figure 3.17: Face Detection in DLIB with Histogram Of Gradients [85]

Evaluation:

From figure 3.17, we can observe that DLIB as part of face_recognition package has obtained 100% accuracy in terms of face extraction for video 3 and video 5. While the performance on video 1 and video 2 is still manageable, the worst performance has been found when tested on video 4 (0% accuracy), although the video is very challenging for any face detection algorithm to detect faces due to lack of sufficient light in the images. Therefore, this face extraction algorithms turns out to be less efficient in terms of accuracy and versatility. Hence, this face extraction algorithm can not be considered in this project.

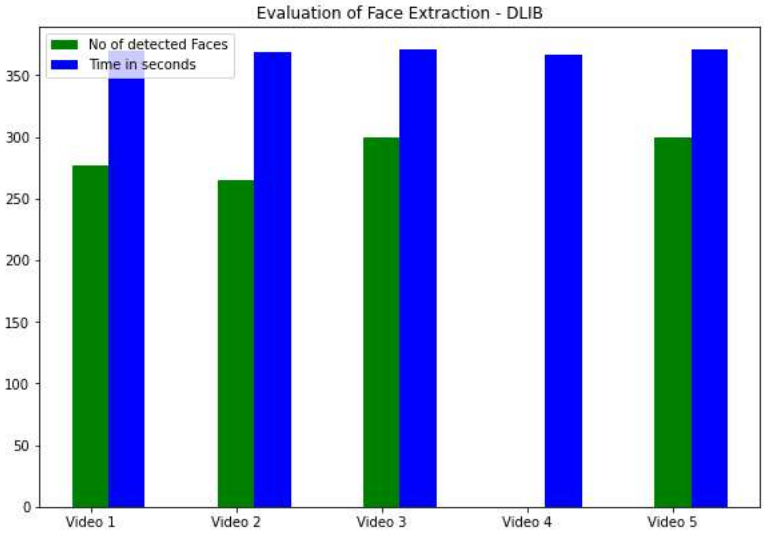


Figure 3.18: Evaluation of DLIB for Face Extraction

3.5.5 Retina-Face for Face Extraction

Though tremendous strides have been made in uncontrolled face detection, accurate and efficient face localisation in the wild remains an open challenge. This face detection system presents a robust single-stage face detector, named Retina-Face, which performs pixel-wise face localisation on various scales of faces by taking advantages of joint extra-supervised and self-supervised multi-task learning. The major contributions of this face extraction methodology are in these following aspect - a) Five facial landmarks on the WIDER FACE dataset have been annotated and observed significant improvement in hard face detection with the assistance of this extra supervision signal, b) A self-supervised mesh decoder branch was added for predicting a pixel-wise 3D shape face information in parallel with the existing supervised branches, c) On the WIDER FACE hard test set, Retina-Face outperforms the state of the art average precision (AP), d) By employing lightweight backbone networks, Retina-Face can run real-time on a single CPU core for a VGA-resolution image [39].

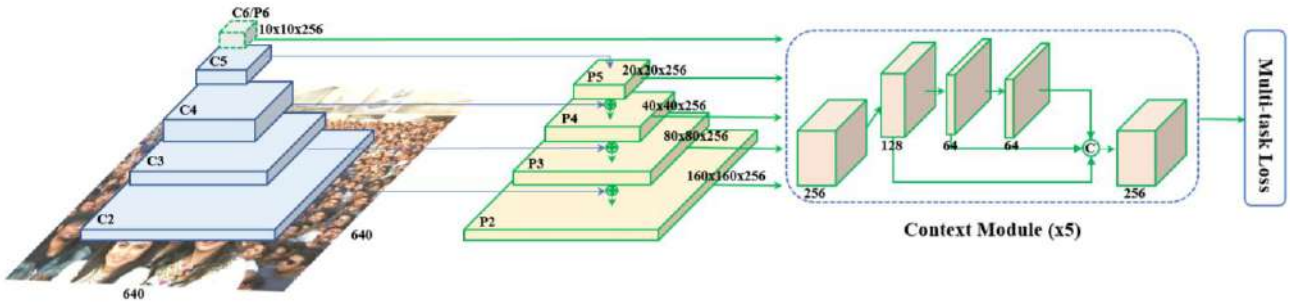


Figure 3.19: Overview of Single-Stage Dense localisation approach in Retina-Face [39]

Mesh Decoder:

The mesh decoder (mesh convolution and mesh up-sampling), which is a graph convolution method based on fast localised spectral filtering has been directly employed in this methodology. In order to achieve further acceleration, a joint shape and texture decoder similarly was implemented to the method in [86], contrary to [87] which only decoded shape. Graph convolutions have been implemented in this methodology for fast decoding. A 2D convolutional operation is a "kernel-weighted neighbour sum" within the Euclidean grid receptive field, similarly, graph convolution also employs the same concept however the neighbour distance is calculated on the graph by counting the minimum number of edges connecting two vertices. After the shape and texture parameters are predicted , an efficient differentiable 3D mesh renderer [88] was employed to project a coloured mesh onto a 2D image plane. Once the rendered 2D face is obtained ,the pixel-wise difference of the rendered and the original 2D face were computed for self-supervision. [39].

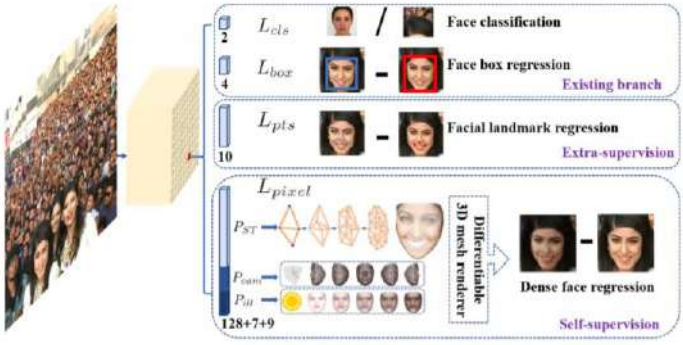


Figure 3.20: Retina-Face: extra-supervised and self-supervised multi-task learning in parallel with the existing box classification and regression branches [39]

Feature Pyramid:

RetinaFace employs feature pyramid levels from P2 to P6, where P2 to P5 are computed from the output of the corresponding ResNet residual stage (C2 through C5) using top-down and lateral connections as in [89, 90]. P6 is calculated through a 33 convolution with stride=2 on C5. C1 to C5 is a pre-trained ResNet-152 [73] classification network on the ImageNet-11k dataset while P6 are randomly initialised with the "Xavier" method [91].

Context Module:

Inspired by SSH [92] and PyramidBox [93], we also apply independent context modules on five feature pyramid levels to increase the receptive field and enhance the rigid context modelling power. Drawing lessons from the champion of the WIDER Face Challenge 2018 [94], we also replace all 3 3 convolution layers within the lateral connections and context modules by the deformable convolution network (DCN) [95, 96], which further strengthens the non-rigid context modelling capacity.

Loss Head:

For negative anchors, only classification loss is applied. For positive anchors, the proposed multi-task loss is calculated. A shared loss head (1 1 conv) across different feature maps Hn Wn 256 has been employed. For the mesh decoder, the pre-trained model [86] was applied, which is a small computational overhead that allows for efficient inference.

Evaluation:

From the results presented in figure 3.20, we can observe that Retina-Face has successfully extracted 299 faces accurately from each videos which corresponds to 100% accuracy on these five videos. This test has been further extended to all 154 videos and 100% accuracy has been achieved through Retina-Face.

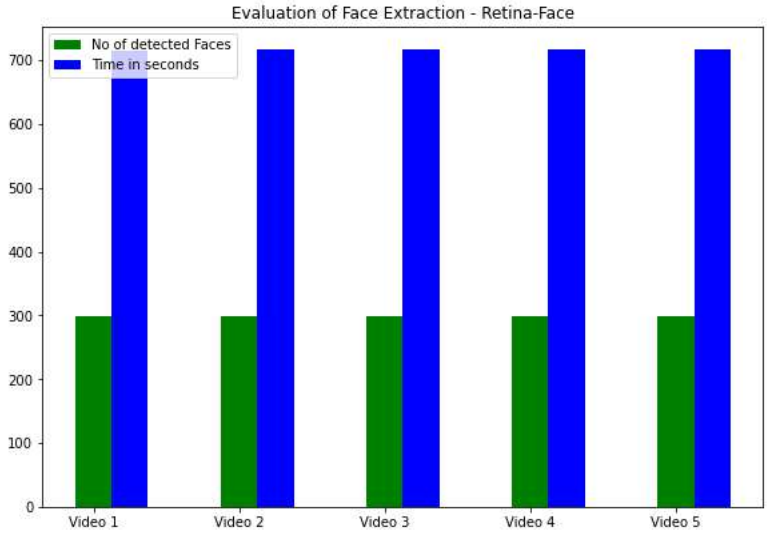


Figure 3.21: Evaluation of RetinaFace for Face Extraction

Proposed Methodology for Face Extraction:

Our proposed methodology for face extraction in this project employs Retina-Face face extraction methodology on top of boosted contrast on every image frame of these videos. The possible advantages of image frames with boosted contrast for our final deepfake detection model will be discussed in chapter 4 with results. The average time elapsed for Retina-Face to extract faces from 299 images frames with boosted contrast from the videos take 27 seconds more compared to extracting faces from images with no additional contrast.

3.6 Pre-processing

Early research on back propagation (BP) algorithms found improvement in performance of training neural networks on: (i) selection of better Loss functions [97, 98]; (ii) different choices for activation functions [98, 99] and, (iii) selection of dynamic learning rate and momentum [100, 101]. Later, it was found out that the techniques for Pre-processing the data also plays an important role in effecting the performance of Neural Networks, where pre-processing the data encourage high accuracy and less computational cost associated to the learning phase (better convergence) [102]. As discussed previously in data pipeline for deepfake detection system, two pre-processing technique have been employed in this project to ensure better convergence of gradient of Loss function.

3.6.1 Normalization

In order to normalize the pixel values of the images after face extraction phase, all the image pixels were first converted to NumPy Floating data type and further divided by 255.0 across all channels (R-G-B).

3.6.2 Standardisation

The standardisation on image pixels is usually done to centralise the data for better convergence. In this project, each individual image with all three channels (R-G-B) has been standardised with the help of per image standardisation method present in Tensorflow[46].

3.6.3 Proposed Pre-processing Methodology

The proposed methodology for pre-processing introduces Normalization of images first followed by per image standardisation for each image. The advantage of using proposed methodology compared to only per image standardisation or only Normalization or Standardisation followed by Normalization has been evaluated in terms of model performance in chapter 4. This is to be noted, for all our proposed deepfake detection models Normalization-Standardisation combination is used in this order. However, in all our proposed transfer learning approaches Normalization-Standardisation was followed by another pair of Normalization-Standardisation applied after the prediction(low level image features) of pre-trained models(VGG16 [41], Xception[42] etc.). This additional pre-processing methodology has been evaluated as well in chapter 4.

3.7 Pritam's Baseline Model for DeepFake Detection with CNN

As discussed in section 2.4, the pixel level inconsistencies in deepfake videos can be detected with the help of a Convolutional Nueral Network. In this Section, we explore a simple baseline convolution neural network with few layers to detect pixel artifacts within the face of the subjects in deepfake videos. Compared to some of the deep Convolutional Neural Network, this baseline model not only trains input data very quickly with less memory and computing power due to the less number of layers but also performs fairly well compared to other deep networks(Evaluation in chapter 4). Inspired by the latest success of pre-trained models such as VGG16 [41],ResNet50 [40],Xception [42] etc. on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for Image classification, this project has adopted some of the elements from these architectures to develop the baseline model for deepfake detection system. The input size of the images for the baseline model has been considered to be 224x224 pixels for all 3 channels (i.e. 224x224x3 for each input image), which is conventional input size for most pre-trained models for image classification at ILSVRC.

3.7.1 Model Architecture

The baseline model follows a Conv-Pool-Conv-Pool architecture adopted from mainstream Imagenet models that starts with a convolutional layer followed by BatchNormalization [103] layer and finally a max pooling layer; this block of 3 layers are repeated several times to obtain low level image features. The low level image features are then flattened (by a Flatten layer) to transform the data into one dimensional array. The one dimensional image data is further connected to fully-connected (FC) layers and finally through sigmoid activation function at the final FC layer binary classification is achieved at the end of the network. For the baseline model, only three FC layers have been implemented out of which the first one contains 1024 nodes followed by 128 nodes and finally 1 node for binary classification.

3.7.2 Model Variations

The efficacy of convolutional neural network can sometimes vary significantly if some of the parameters of the model can be refined to obtain the optimal performance. The convolution layers of the baseline model

contains a kernel size of 3x3 across all the layers. The output channel sizes for four convolution layers for this network have been altered in 3 variations for training the model such as 16-16-32-32, 32-32-64-64 and 64-64-128-128 respectively [104]. 64-64-128-128 variation for output channels of the convolution layers has been adapted from VGG16 and ResNet architecture. The output channel size for the convolution layers are usually set as power of 2[24]. therefore this has been tested for variation 16-16-32-32 to choose the optimal parameters for the baseline model. Figure 3.21 presents the architecture for the baseline model with 16-16-32-32 variation.

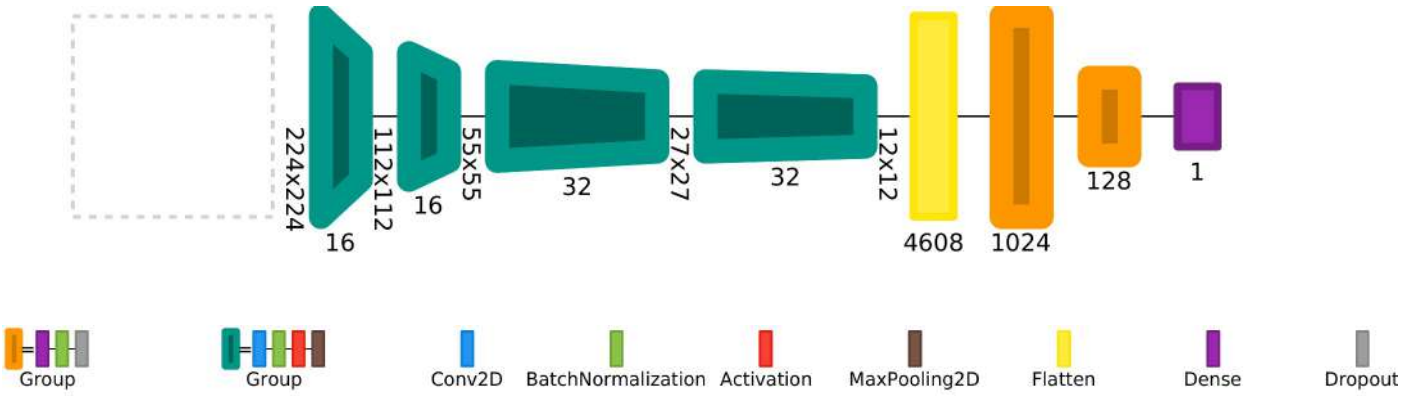


Figure 3.22: Overview of baseline model (16-16-32-32) architecture

3.7.3 Training

Figure 3.22 groups the number of layers into blocks which contains stack of layers in order to simplify and summerize the architecture of the baseline model. Training the baseline model is one of most consequential part of deepfake detection system. Therefore, several techniques and optimizations have been considered to converge the model efficiently without overfitting data during training. In order to avoid overfitting of data, BatchNormalization [103] has been used after every convolution layer, a dropout [105] of 0.2 has been applied after every FC layer and Early-Stopping is applied with a patience of 5 epochs. Binary Crossentropy has been the loss function for training the model and Adaptive Moment Estimation (ADAM) has been devised as the optimization algorithm with default parameters. The batch-size devised for the baseline model has been 64 and the model was trained for 50 epochs.

3.8 Transfer Learning for DeepFake Detection

Transfer Learning is a machine learning methodology where the knowledge gained in one domain can be transferred to a related domain so that isolated learning paradigm can be solved with re-usability of common knowledge. There were several advantages of using the baseline model with few convolution and max pool layers in this project as discussed in the previous section. Nevertheless, few convolution layers and limited number of training samples may not adjust the weights of the convolution layers to produce low-level image features efficiently from image data. Deep Convolution Neural Networks on the other hand has so many layers to train and requires huge computation power and enormous amount of training data. In order to solve both of these challenges, a pre-trained Network which was trained on millions of images from a similar domain (ImageNET in our case) can be applied to our image data to produce low level image features which could further be trained with multi-layer perceptron (MLP) or FC layers in the end. The advantages of this approach are no training required (unless fine-tuned), deep convolution layers may perform more efficiently, pre-trained network weights can produce efficient low level features. Hence, transfer learning is really useful if there is

insufficient data for a new domain that can be taken care of by adapting the representation of neural network to this new domain, if there is a big pre-existing data pool that can be transferred to this domain or task. In this project, the pre-trained models from ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which were trained on millions on images from IMAGENET dataset, have been adapted to solve our domain specific task of detecting deepfake images.

3.8.1 VGG16

VGG16 was introduced in 2014 by Karen Simonyan and Andrew Zisserman in the paper titled Very Deep Convolutional Networks for Large-Scale Image Recognition [41].The model achieved a 92.7% top-five test accuracy in ImageNet in ILSVRC-2014 [106]. The architecture of VGG16 can be summerised as,

- The maximum filter size is 3 x 3 and the minimum size is 1 x 1. This means that a smaller filter size with a larger quantity is used, compared to a larger filter size and smaller quantity for AlexNet; this results in fewer parameters compared to AlexNet.
- Convolution stride is 1 and padding is 1 for a 3 x 3 convolutional layer. Max pooling is performed over a 2 x 2 window with a stride of 2.
- Three nonlinear ReLU functions are used instead of a single one in each layer, which makes the decision function more discriminative by reducing the vanishing gradient problem and enabling the network to learn deeply. Learning deeply here means learning complex shapes, such as edges, features, boundaries, and so on [106].

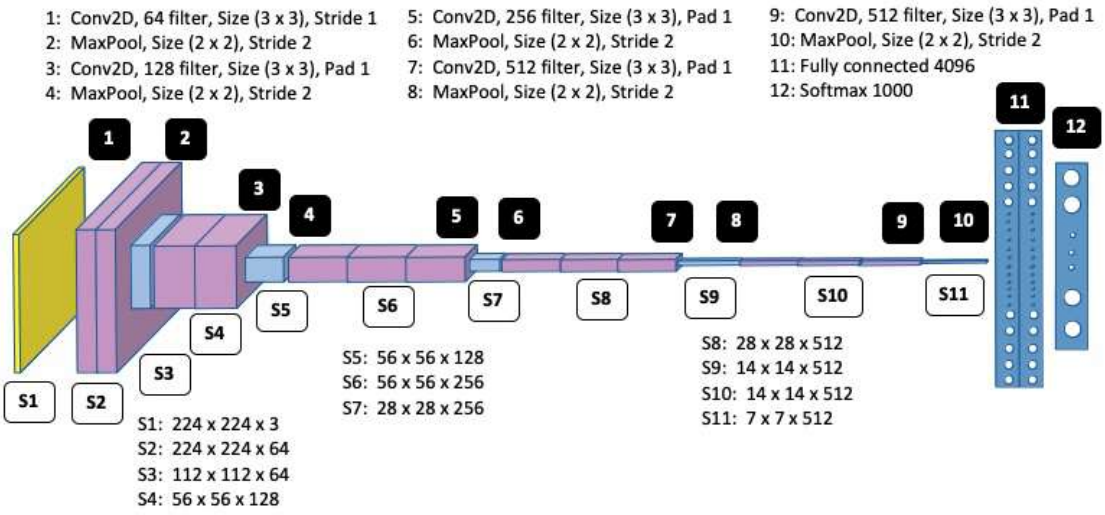


Figure 3.23: Model Architecture of VGG16 [106]

Proposed Methodology:

The proposed methodology suggests to deploy pre-trained VGG16 model as an Image feature extractor to transform the image data into low level image features with the help of convolutional operation. The idea has been adapted from mainstream image classification approaches that suggests to utilize the pre-trained weights/convolution filters of VGG16 by removing the top layers (Flatten and FC layers) in this pre-trained model and replacing it with custom FC layers and regularization. Hence, the top layers of VGG16 have been removed in this project and these were replaced by FC layers of 1024, 64, 1 nodes respectively. The pre-trained

convolution layers were not trained in this approach however the low level image features extracted from VGG16 were further pre-processed (Normalized and per Image Standardised) and connected to BatchNormalization [103] and FC layers for deepfake classification.

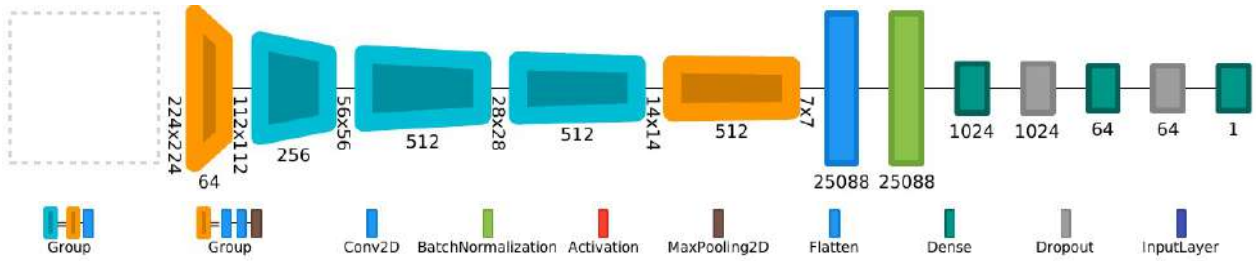


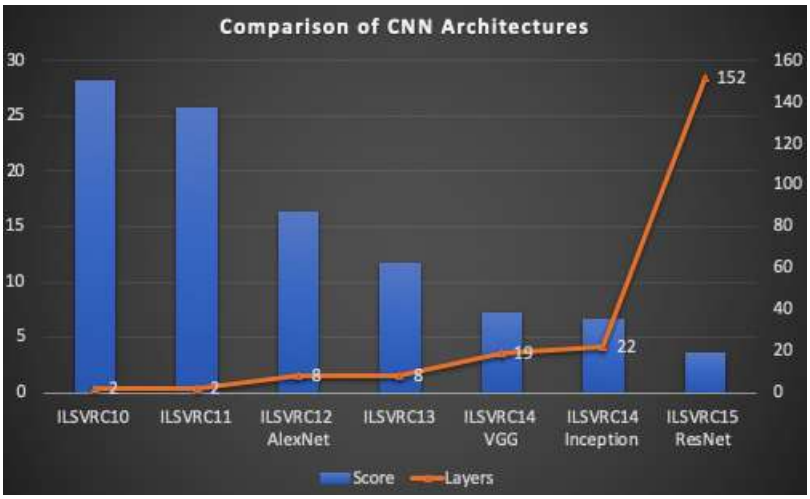
Figure 3.24: Overview of Proposed model with VGG16

Training:

Several techniques and optimizations have been considered to converge the model efficiently without overfitting data during training. In order to avoid overfitting of data, BatchNormalization has been used after VGG16 features extraction , a dropout of 0.5 has been applied after every FC layer since millions of parameters were required to train at FC layers in this case and Early-Stopping is applied with a patience of 5 epochs. Binary Crossentropy has been the loss function for training the model and Adaptive Moment Estimation (ADAM) has been devised as the optimization algorithm in this methodology with default learning rates. The batch-size devised for this proposed methodology has been 64 and the model was trained for 50 epochs. The default batch-size is usually 32 in keras for training sequential models in mini-batches. A larger batch-size may improve the consistency and performance of the model to converge quickly so it was adopted however it requires huge amount of resources or processing power for parallel execution of data simultaneously. Same training methodology and parameters have been applied to all our transfer learning models.

3.8.2 ResNet50

ResNet, introduced by Kaiminh He, Xiangyu Zhang, Shaoquing Ren, and Jian Sun in the paper titled Deep Residual Learning for Image Recognition [40], was developed to address the accuracy degradation problem of deep neural networks with an increase in depth. This degradation is not caused by overfitting, but results from the fact that after some critical depth, the output looses the information of the input, so the correlation between the input and output starts diverging resulting in an increase in inaccuracy [106]. In order to utilize the principles of residual network for deep convolutional network, this model has been selected to evaluate against it's predecessors. The performance of ResNet in comparison to other popular models at ILSVRC can be presented as follows,



The key features of ResNet model can be summerised as follows:

- The degradation problem is addressed by introducing a deep residual-learning framework.
- The framework introduces the concept of a shortcut or skips connection, which skips one or more layers.
- The underlying mapping between the input and the next layer is H(x).
- The nonlinear layer is F(x) = H(x) – x, which can be reframed as H(x) = F(x)+x, where x is the identity mapping.
- The shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. It has been presented in the following figure [106].

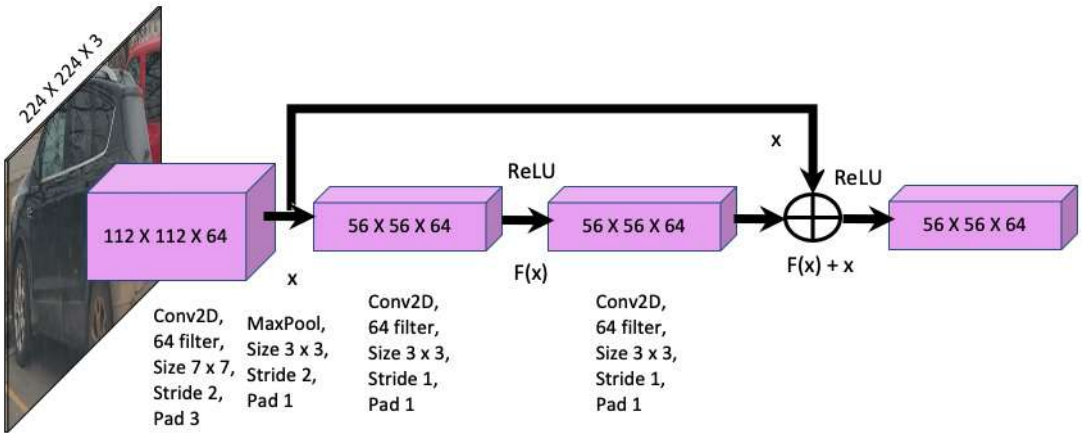


Figure 3.25: Overview of residual block in ResNet [106]

- Figure 3.24 also suggests that the operation F(x) + x is performed by a shortcut connection and the addition of elements.
- Identity shortcut connections add neither an extra parameter nor computational complexity.

The complete ResNet model architecture can be presented in the following diagram,

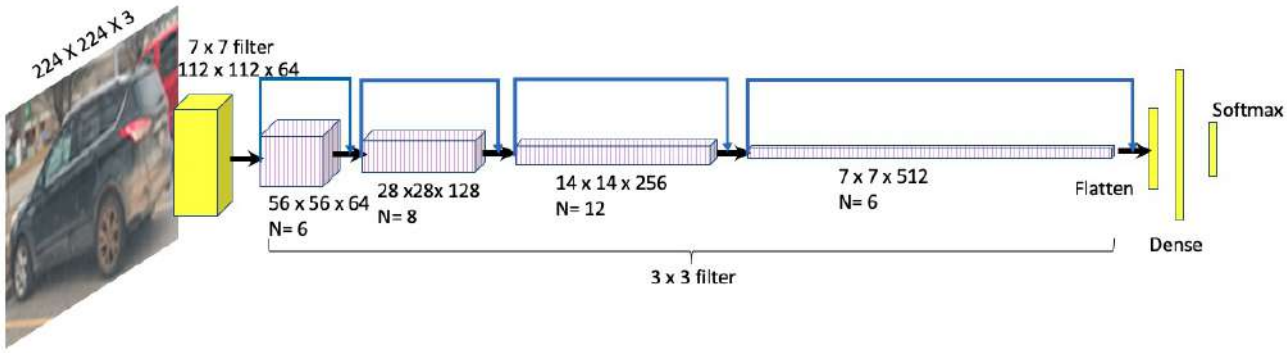


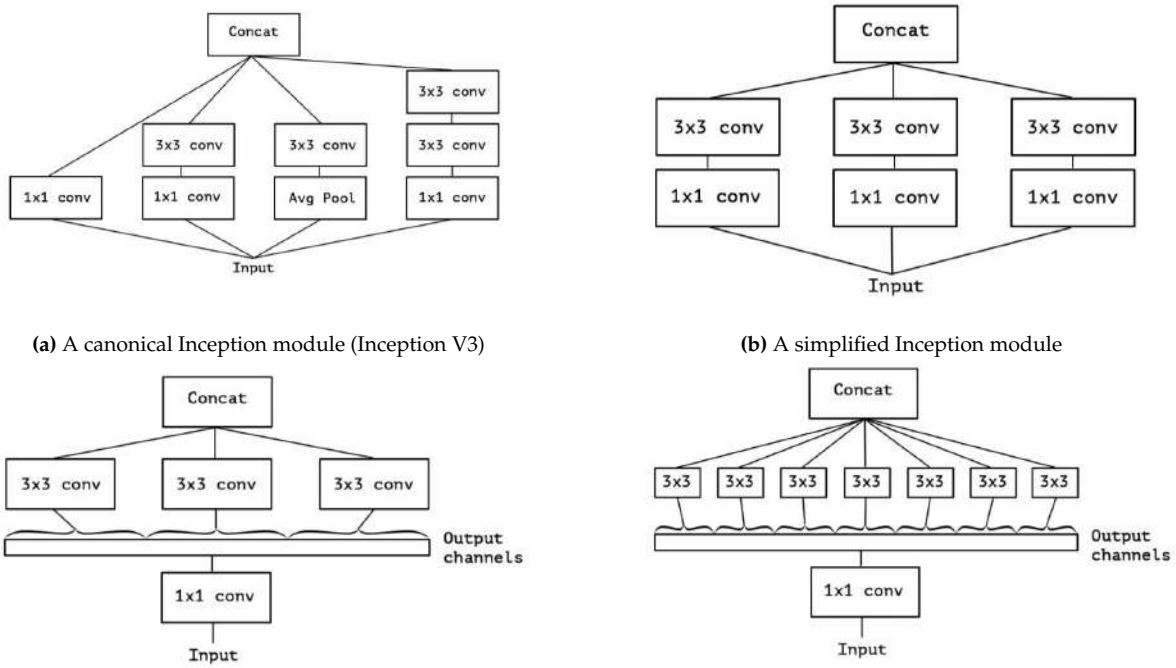
Figure 3.26: Overview of ResNet Architecture [106]

Proposed Methodology:

The proposed methodology suggests to deploy pre-trained ResNet50 model as an Image feature extractor to transform the image data into low level image features with the help of convolutional operation. The idea has been adapted from mainstream image classification approaches that suggests to utilize the pre-trained weights/convolution filters of ResNet50 by removing the top layers (Flatten and FC layers) in this pre-trained model and replacing it with custom FC layers and regularization. Hence, the top layers of ResNet50 have been removed in this project and these were replaced by FC layers of 1024, 64, 1 nodes respectively. The same Flatten and FC layer architecture has been considered for all of our pre-trained models for transfer learning to maintain consistency so that these models could be compared with each other for performance evaluation. The pre-trained convolution layers were not trained in this approach however the low level image features extracted from ResNet50 were further pre-processed (Normalized and per Image Standardised) and connected to BatchNormalization and FC layers for deepfake classification.

3.8.3 Xception

Even though, Residual module solves the accuracy degradation challenge in deep convolution layers as discussed before, the kernel size for convolution layers were fixed throughout the network. The inception module on the other hand applies multiple convolution layers with different kernel sizes and same pooling layer to input data and the result is concatenated. This allows the model to take advantage of multi-level feature extraction. There this model has been proposed as it takes advantage of both inception module and residual connections together in one architecture. A convolution layer attempts to learn filters in a 3D space, with 2 spatial dimensions (width and height) and a channel dimension; thus a single convolution kernel is tasked with simultaneously mapping cross-channel correlations and spatial correlations. This idea behind the Inception module is to make this process easier and more efficient by explicitly factoring it into a series of operations that would independently look at cross-channel correlations and at spatial correlations. More precisely, the typical Inception module first looks at cross-channel correlations via a set of 1x1 convolutions, mapping the input data into 3 or 4 separate spaces that are smaller than the original input space, and then maps all correlations in these smaller 3D spaces, via regular 3x3 or 5x5 convolutions [42]. This has been presented in Figure 3.26(a). The fundamental hypothesis behind Inception is that cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly. Consider a simplified version of an Inception module that only uses one size of convolution (e.g. 3x3) and does not include an average pooling tower (figure 3.26.(b)). This Inception module can be reformulated as a large 1x1 convolution followed by



(a) A canonical Inception module (Inception V3)

(b) A simplified Inception module

(c) A strictly equivalent reformulation of the simplified Inception module.

(d) An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

Figure 3.27: Evolution of Xception Module from Inception Module [42]

spatial convolutions that would operate on non-overlapping segments of the output channels (figure 3.26. (c)). This observation naturally raises the question: what is the effect of the number of segments in the partition (and their size)? Would it be reasonable to make a much stronger hypothesis than the Inception hypothesis, and assume that cross-channel correlations and spatial correlations can be mapped completely separately?[42]

An "extreme" version of an Inception module, based on this stronger hypothesis, would first use a 1x1 convolution to map cross-channel correlations, and would then separately map the spatial correlations of every output channel. This is shown in figure 3.26(d). This extreme form of an Inception module is almost identical to a depthwise separable convolution. A depthwise separable convolution, commonly called "separable convolution" in deep learning frameworks such as TensorFlow and Keras, consists in a depthwise convolution, i.e. a spatial convolution performed independently over each channel of an input, followed by a pointwise convolution, i.e. a 1x1 convolution, projecting the channels output by the depthwise convolution onto a new channel space. Two minor differences between and "extreme" version of an Inception module and a depthwise separable convolution are [42] :

- The order of the operations: depthwise separable convolutions as usually implemented (e.g. in TensorFlow) perform first channel-wise spatial convolution and then perform 1x1 convolution, whereas Inception performs the 1x1 convolution first.
- The presence or absence of a non-linearity after the first operation. In Inception, both operations are followed by a ReLU non-linearity, however depthwise separable convolutions are usually implemented without non-linearities.

It can be noted that, there is a discrete spectrum between regular convolutions and depthwise separable

convolutions, parametrized by the number of independent channel-space segments used for performing spatial convolutions. A regular convolution (preceded by a 1x1 convolution), at one extreme of this spectrum, corresponds to the single-segment case; a depthwise separable convolution corresponds to the other extreme where there is one segment per channel; Inception modules lie in between, dividing a few hundreds of channels into 3 or 4 segments. These observations suggest it may be possible to improve upon the Inception family of architectures by replacing Inception modules with depthwise separable convolutions, i.e. by building models that would be stacks of depthwise separable convolutions [42].

Xception Architecture:

This deep neural network architecture is based entirely on depthwise separable convolution layers. This architecture suggests the mapping of cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled. Because this hypothesis is a stronger version of the hypothesis underlying the Inception architecture, the proposed architecture was named Xception, which stands for "Extreme Inception". The architecture of the model is presented in the figure below. The Xception architecture has 36 convolutional layers forming the feature extraction base of the network. The 36 convolutional layers are structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules. In short, the Xception architecture is a linear stack of depthwise separable convolution layers with residual connections [42].

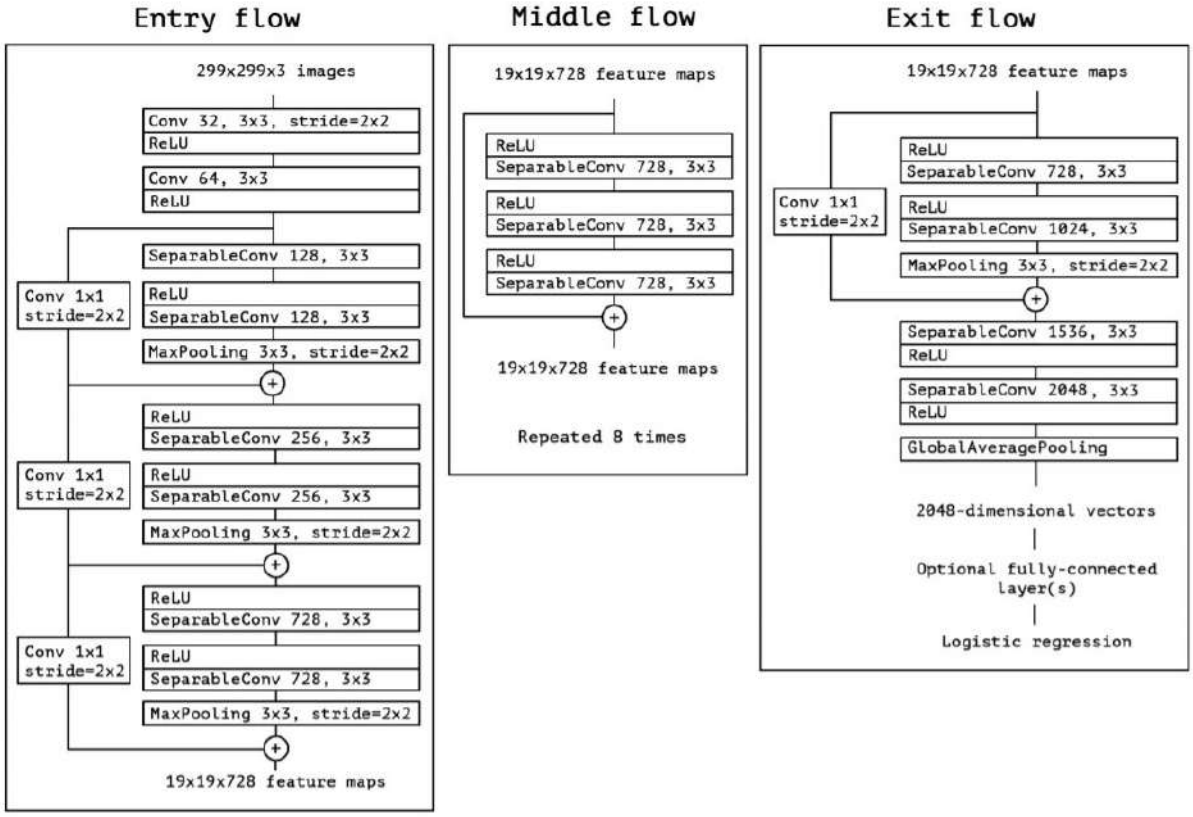


Figure 3.28: Overview of Xception Architecture [42]

Proposed Methodology:

The proposed methodology suggests to deploy pre-trained Xception model as an Image feature extractor to transform the image data into low level image features with the help of convolutional operation. The idea has been adapted from mainstream image classification approaches that suggests to utilize the pre-trained weights/convolution filters of Xception by removing the top layers (Flatten and FC layers) in this pre-trained model and replacing it with custom FC layers and regularization. Hence, the top layers of Xception have been removed in this project and these were replaced by FC layers of 1024, 64, 1 nodes respectively. The same Flatten and FC layer architecture has been considered for all of our pre-trained models for transfer learning to maintain consistency so that these models could be compared with each other for performance evaluation. The pre-trained convolution layers were not trained in this approach however the low level image features extracted from Xception were further pre-processed (Normalized and per Image Standardised) and connected to BatchNormalization and FC layers for deepfake classification.

3.8.4 InceptionResNetV2

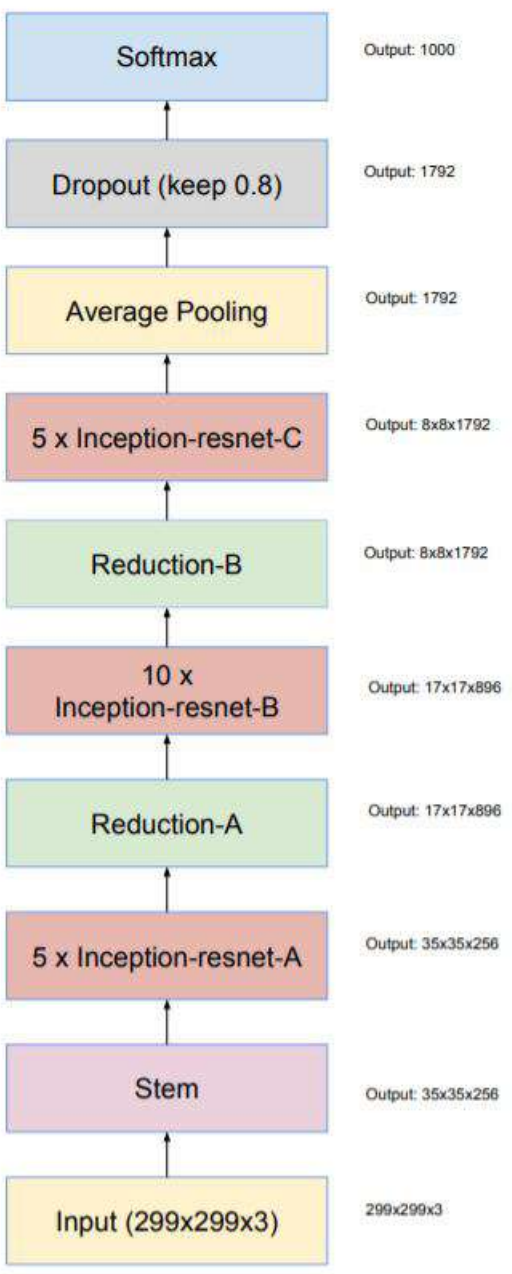
Very deep convolutional networks have been central to the largest advances in image classification performance in recent years. One example is the Inception architecture that has been shown to achieve very good performance at relatively low computational cost. Recently, the introduction of residual connections in conjunction with a more traditional architecture has yielded state-of-the-art performance in the 2015 ILSVRC challenge; its performance was similar to the latest generation Inception-v3 network. This raises the question of whether there are any benefit in combining the Inception architecture with residual connections [107]. InceptionResNetV2 introduced specialized "Reduction Blocks" which has been applied to change the width and height of the grid. The earlier versions (Inception- v1,v2 and v3) didn't explicitly have reduction blocks and introduction of BatchNormalization in this model produces higher accuracy at a lower epoch. The combination of residual connections along with inception modules although similar to Xception model, this model has performed better compared to Xception on ILSVRC benchmark(see Figure 5.1).

Residual Inception Blocks:

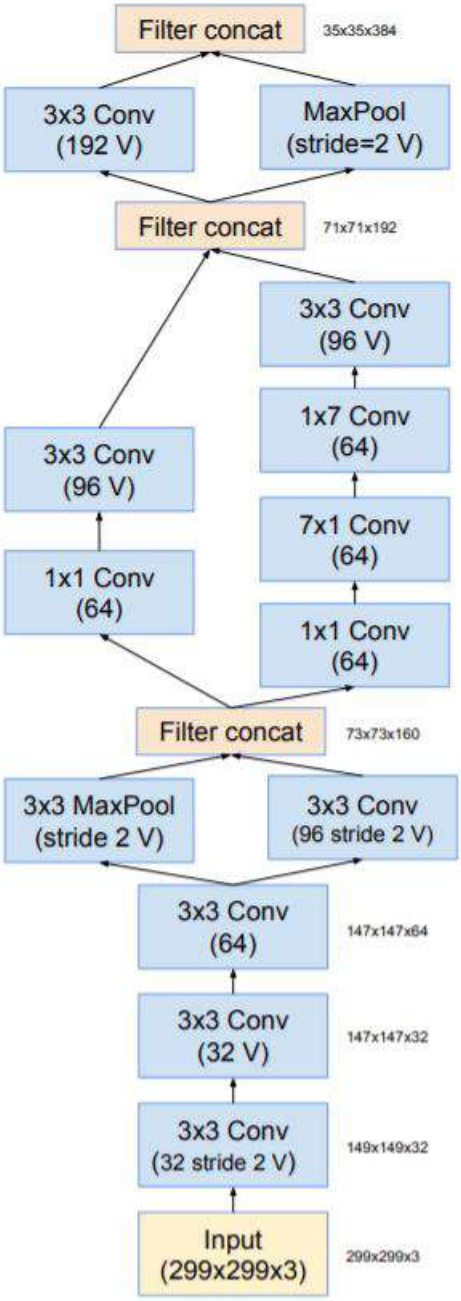
For the residual versions of the Inception networks, cheaper Inception blocks are used than the original Inception. Each Inception block is followed by filter-expansion layer (1 x 1 convolution without activation) which is used for scaling up the dimensionality of the filter bank before the addition to match the depth of the input. This is needed to compensate for the dimensionality reduction induced by the Inception block.Another small technical difference between the residual and non-residual Inception variants is that in the case of Inception-ResNet, batch-normalization is only on top of the traditional layers, but not on top of the summations. It is reasonable to expect that a thorough use of batchnormalization should be advantageous however it also requires a substantial amount of computing resources [107]. The overall architecture of InceptionResNetV2 is presented in Figure 3.28(a) and the structure of each module has been presented in 3.28 (b) and 3.29 (a),(b),(c),(d) and (e). Same proposed methodology has been applied to this model compared to Xception model with transfer learning only the pre-trained model here is InceptionResNetV2.

3.9 MesoNet: A Compact Facial Video Forgery Detection Network

MesoNet presents a Convolutional Neural Network to automatically and efficiently detect face tampering in videos, and particularly focuses on two recent techniques used to generate hyper-realistic forged videos:

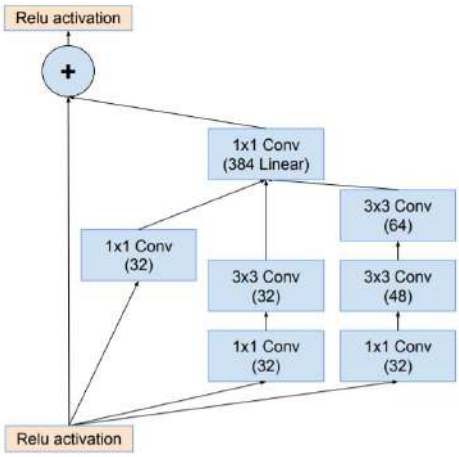


(a) The Summarised Model Architecture of Inception-ResNetV2

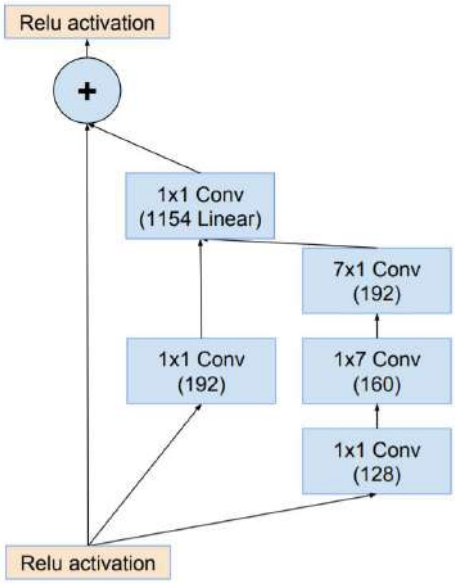


(b) The schema for stem of the Inception-ResNet-v2 networks

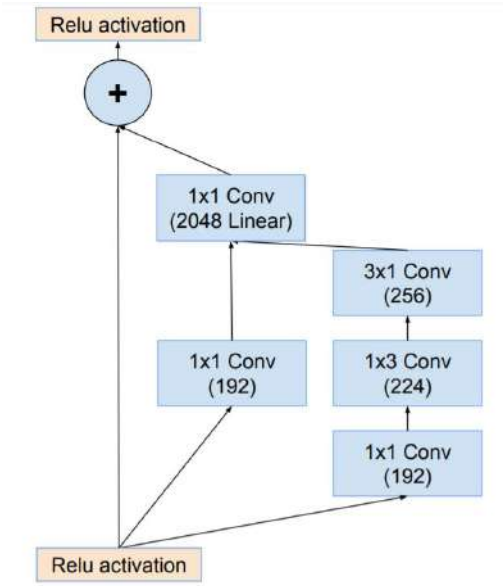
Figure 3.29: Model Overview of Inception-ResNet-V2



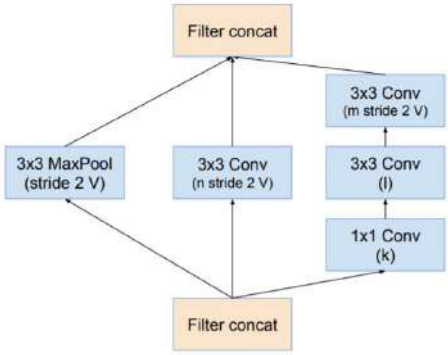
(a) The schema for 35 x 35 grid (Inception-ResNet-A) module of the Inception-ResNet-v2 network.



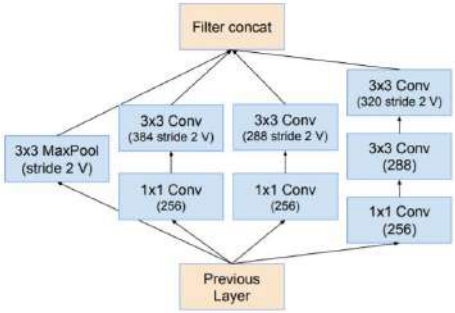
(b) The schema for 17 x 17 grid (Inception-ResNet-B) module of the Inception-ResNet-v2 network



(c) The schema for 8x8 grid (Inception-ResNet-C) module of the Inception-ResNet-v2 network



(d) The schema for 35 x 35 to 17 x 17 grid-reduction module. Reduction-A on 3.28(a)



(e) The schema for 17 x 17 to 8 x 8 grid-reduction module. Reduction-B on 3.28(a)

Figure 3.30: Overview of individual modules in Inception-ResNet-V2 [107]

Deepfake and Face2Face. Therefore, this CNN architecture is adopted in this project due to less convolution layers, faster training time, less computation power and memory requirements and this network has been designed especially to detect mesoscopic artifacts of image pixels (on face images) in deepfake videos which might not be visible to our naked eyes. Traditional image forensics techniques are usually not well suited to videos due to the compression that strongly degrades the data. Thus, this methodology follows a deep learning approach and presents a low number of layers to focus on the mesoscopic properties of images [55].

3.9.1 Meso-4 Architecture

This methodology proposes to detect forged videos of faces by placing this method at a mesoscopic level of analysis. Indeed, microscopic analyses based on image noise cannot be applied in a compressed video context where the image noise is strongly degraded. Similarly, at a higher semantic level, human eye struggles to distinguish forged images [108], especially when the image depicts a human face [109, 110]. That is why this methodology adopts an intermediate approach using a deep neural network with a small number of layers [55].

This network begins with a sequence of four layers of successive convolutions and pooling, and is followed by a dense network with one hidden layer. To improve generalization, the convolutional layers use ReLU activation functions that introduce non-linearities and Batch Normalization [103] to regularize their output and prevent the vanishing gradient effect, and the fully-connected layers use Dropout [105] to regularize and improve their robustness[55].

Proposed Methodology:

According to the Meso-4 [55] architecture, the input shape of the images for the network implemented has been 256x256x3, this has been modified to 224x224x3 in our proposed methodology to maintain identical input shape for all our proposed neural networks and avoid exhausting computational resources for reshaping images for every proposed methodology. The batch-size for training the data has been maintained as 64 like all other proposed CNN architectures that have been implemented for this project. The batch-size of training data selected by the authors of Meso-4 is 75, clearly there is no significant difference is batch-size considering the limited resources and timeline for this project. The weight optimization devised for this project has been ADAM with default parameters which is consistent across all other proposed CNN models discussed before and the authors of MESO-4 have also used the same weight optimizer with default parameters. The proposed Meso-4 architecture is presented as follows in Figure 3.30.

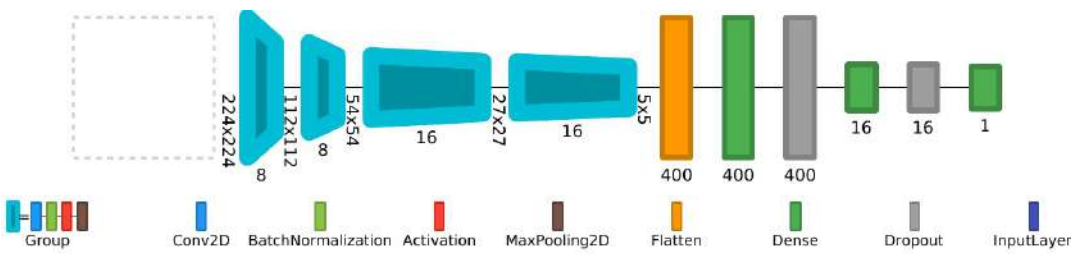


Figure 3.31: Overview of proposed Meso-4 Architecture

3.10 CNN with LSTM for DeepFake Video Classification

Previously, we have introduced many proposed methodologies to classify image frames from the video to be doctored or not. Nevertheless, the spatio-temporal relationship between the image frames of the videos were not considered in those methodologies rather images frames of the videos were trained as each individual image for image classification. Therefore, those proposed methodologies were effective at detecting pixel level artifacts or inconsistencies in deepfake videos as discussed in section 2.4. However, there can be frame level inconsistencies between the image frames of a video (for example frame 1 might look real to naked eyes whereas frame 5 can have severe artifact) as discussed in section 2.5 and that can only be detected by analyzing spatio-temporal features of frame sequences of deepfake videos. In this methodology, the spatio-temporal features allow not only analyzing the pixels of these image frames but also the entire sequence of frames are considered to detect anomalies in a video. The idea to integrate Time-distributed CNN with Long Short Term Memory (LSTM) in this project evolved from mainstream video classification approaches in recent times such as action recognition in video sequences [111]. Furthermore, this approach has been very effective for capturing temporal features and shown improved performance in face anti-spoofing[112] and also deepfake detection [35]. This project aims to achieve exactly the same objective viz. anti-spoofing or anti-forgery detection from extracted faces of deepfake videos. Therefore, this advanced methodology has been implemented to evaluate if noticeable improvement can be found in comparison to our previous methodologies.

3.10.1 Model Architecture

In previous methodologies, the convolutional neural networks proposed usually contains convolutional layer, max pooling layer and other layers viz. BatchNormalization, Flatten, FC etc. Nevertheless, simple convolution layer can not capture spatio-temporal relationship between sequence of image frames in the videos. Therefore, an advanced type of Recurrent Neural Network (Long Short Term Memory) has been devised for this methodology to capture contextual dependency between the sequence of image frames in the videos. However, we also require convolutional neural network to use convolution operation on image frames and extract low level image features. Therefore, the combination of CNN with LSTM is proposed to predict deepfake detection model. A simple convolution operation however requires only 3 dimensions width, height and input channels while recurrent neural network requires input data/image features along with number of sequences or timestamps for which temporal relationship between the input sequence can be captured and utilized for classification. For example, if we have 30 frames in sequence for each video, RNN requires 30 as a parameter to capture the dependency between these 30 image frames. However, if all the image frames in every video is trained by the same CNN, the weights of the network will remain same for each sequence in a video. This idea can generalize the extracted features based on same weights of the network, therefore it is required to implement a CNN for every sequence of the video. This is where Time-Distributed wrapper for CNN becomes very useful as it produces distributed CNN for every timestamp or sequence. This means several number of CNN will be trained in parallel and produce image features for every input sequence.

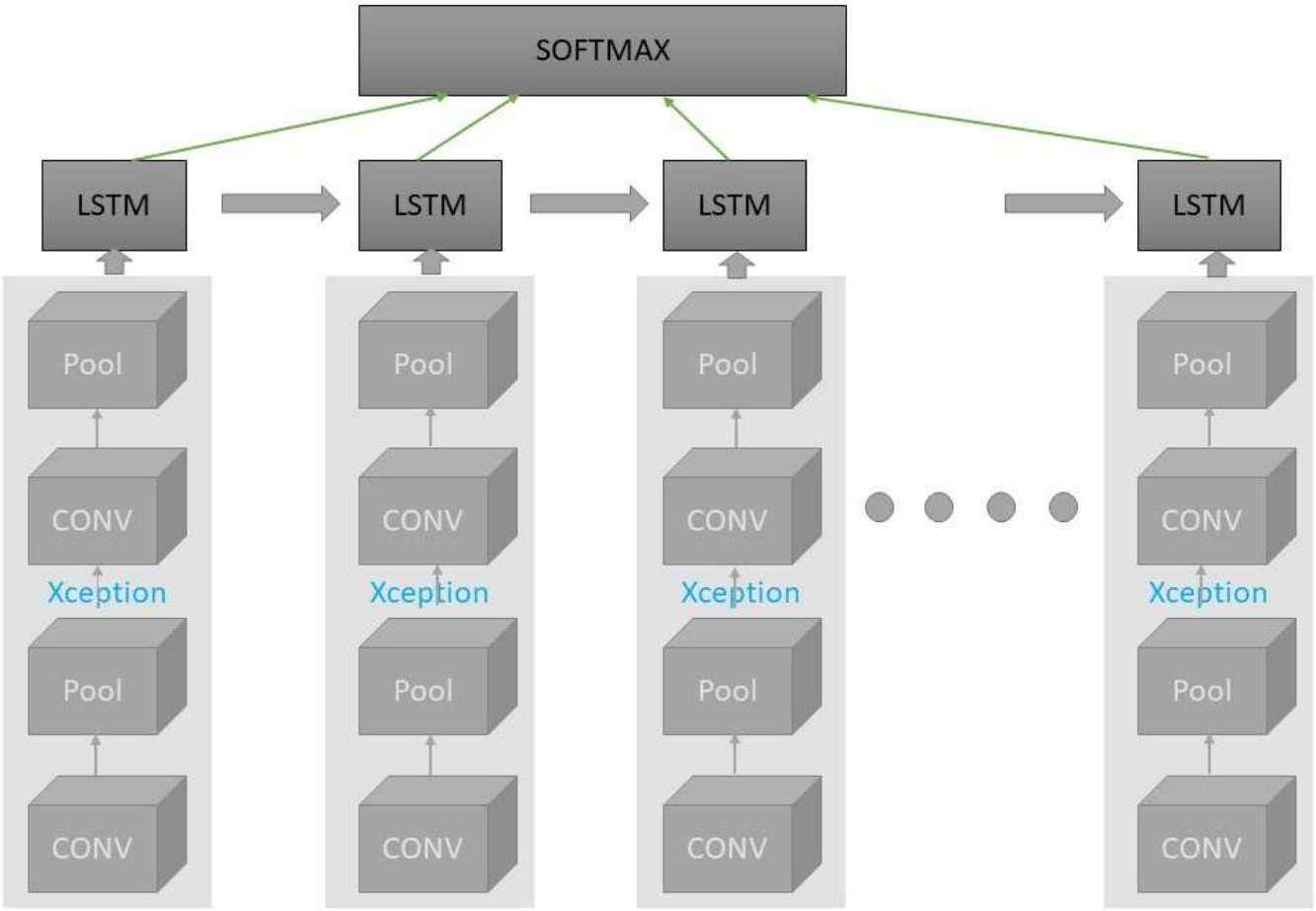


Figure 3.32: Overview of proposed CNN-LSTM: an integrated approach Architecture

3.10.2 Proposed Methodology

The proposed methodology introduces Time-Distributed CNN architecture along with 2 LSTM layers, the first layer contains LSTM units equal to number of input sequences in the video and the second contains 1 unit and acts as decision layer based on the sequence data from it's previous layer. The pre-trained model Xception as discussed before has been implemented with time-distributed wrapper (as CNN) in this project to transform the image sequences into low level features in same sequence. Xception and ResNet50 models were selected in this case as it is an advanced pre-trained model with less number of layers compared to InceptionResNetV2, which saves a lot of computational resources and time. Two model variations of this architecture have been implemented in this project. The variations of the model can be discussed as follows,

- **Time-Distributed CNN with LSTM: An Integrated Approach:** As the name suggests in this approach, the time-distributed CNN is coupled with LSTM layers as a single model and therefore the parallel CNNs will be trained along with LSTM layers in this case for each epoch and batch of image data.
- **Time-Distributed CNN with LSTM: Two Sequential Networks:** In this approach the image features from the time-distributed CNN is first obtained. Then, the image data obtained from CNN is normalized and standardised for each image intensities and finally trained in a separate LSTM network for deepfake classification.

3.11 Generative Adversarial Network for DeepFake Detection

In previous methodologies, we tried to identify forgery in images or sequence of images through combination of CNNs or pre-trained CNNs and RNN. But what if we could generate the deepfake images ourselves from a different dataset of real images and try to discern the difference in image representation between a real image and a doctored image. This idea could then be applied to our Kaggle deepfake detection challenge dataset to verify if this neural network has learned to discern distortion or artifacts in faces of the images compared to authentic ones. Therefore, this proposed methodology could discern between a real face and a distorted or noisy face in general as this has been trained on 10,000 images in celebrity FaceA [113] dataset compared to 154 videos in kaggle dataset. Furthermore, the discriminator model is domain adapted to kaggle dataset so that weights of discriminator network is fine-tuned during training on kaggle dataset. The difference in image representation may be captured by the discriminator network at the time of training DcGAN (Deep Convolutional Generative Adversarial Network) to generate realistic looking fake celebrity faces for 280 epochs and finally the data representation in our kaggle dataset can be learned during further training. Figure 3.32 presents an overview of DcGAN architecture.

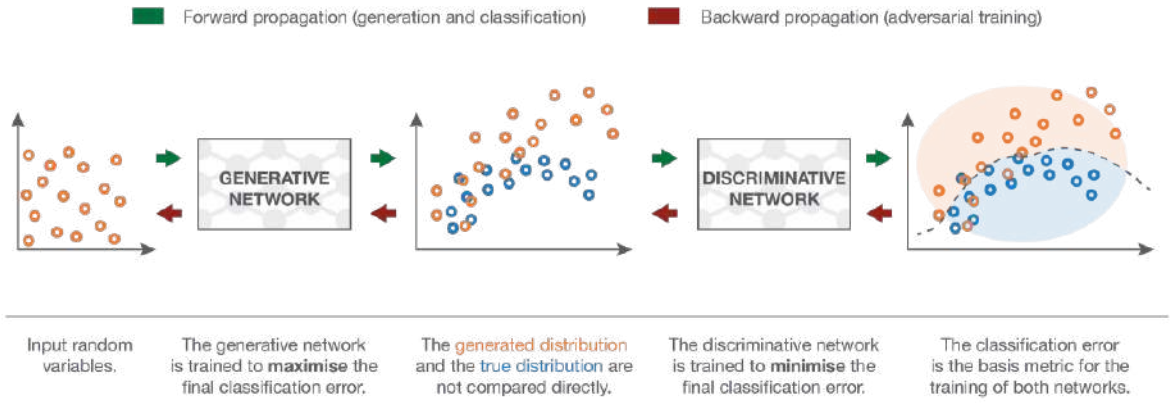


Figure 3.33: Overview of proposed DcGAN Architecture [114]

3.11.1 Proposed Methodology

The proposed methodology introduces two neural networks Discriminator and Generator contest with each other in the form of a zero sum game. The idea has been adopted from the machine learning framework designed by Ian Goodfellow in 2014. In this project, DcGAN has been introduced to generate fake images from Large-scale CelebFaces Attributes (CelebA) Dataset [113] and after the training process the discriminator is further trained on kaggle deep fake challenge dataset. In order to save computing resources and time, the input shape of the images were devised as 64x64x3. Similar to all our previous models ADAM has been applied as our optimization algorithm in this case.

Model Architecture:

The architecture of Generator and Discriminator network can be discussed as follows. The initial input of Generator Network is simply a (1, 100) noise vector, which passes through 4 Convolutional layers with up-sampling and a stride of 2 to produce a result of an RGB image of size (64, 64, 3). To achieve this, the input

vector is projected onto a 1024-dimensional output to match the input of the first Convolution layer [114]. The architecture of the Generator network has been presented with the help of the following figure.

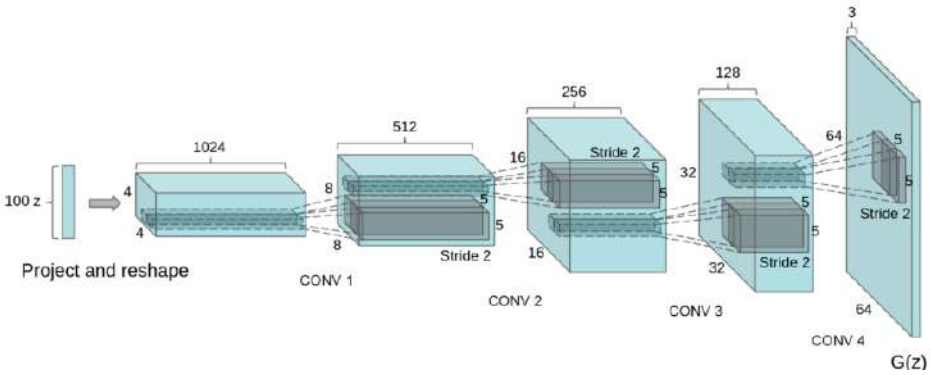


Figure 3.34: Model Architecture of Generator Network [114]

The input image of the discriminator network has been (64, 64, 3), same as Generator's output. The input images were passed it to 4 standard convolution layers, again with a stride of 2. In the final output layer, the image data was transformed by a Flatten Layer to a single dimensional vector. The output of Flatten layer was then fed to a sigmoid function, which discerns the image to be real or fake (a single value representing the probability in the range [0,1] - Real = 1 or Fake = 0). The Discriminator architecture can be presented by the help of the following figure.

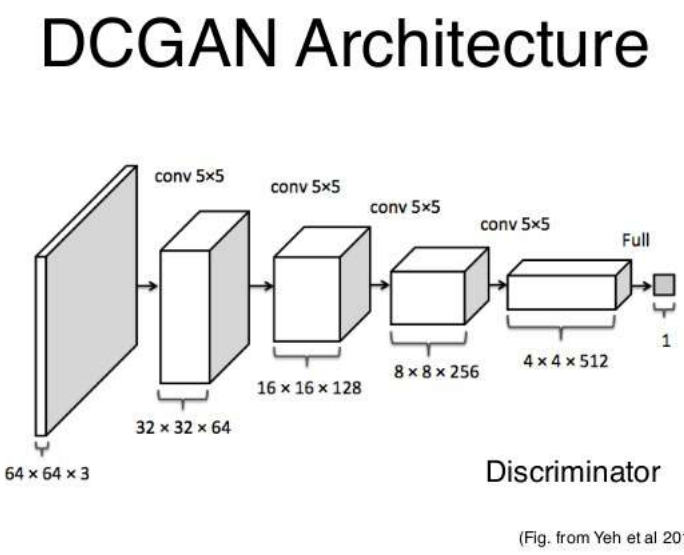


Figure 3.35: Model Architecture of Discriminator Network [114]

Loss Function:

The loss function is designed such that the players Discriminator (D) and Generator (G) are pitted against each other. At a particular iteration, D tries to get better at classifying x and . Its parameters denoted by θ are trained to maximize the loss to distinguish between the real and generated samples. In the same iteration, G is also trained. The parameters of G denoted by ϕ are optimized such that the discriminator is not able to distinguish

between x and . G is essentially trained to produce images which are more realistic such that the discriminator is confused. Ideally, ϕ is trained to minimize the same loss that θ is maximizing. Hence it is similar to a zero-sum game where the players have total competition (Loss of D is gain for G and vice versa)[115]. The loss function of DcGAN can be expressed as,

$$\min_{\phi}\max_{\theta} V(D_\theta, G_\phi) = E_{x\sim p_d(x)}\left[\log D_\theta(x)\right] + E_{z\sim p_z(z)}\left[\log(1 - D_\theta(G_\phi(z)))\right] \tag{3.3}$$

This is a typical binary cross entropy loss, where the real data samples x are given label 1 and generated samples $\hat{x} := G(z)$ are given label 0 [115].

Weight Initialization:

Weight Initialization can be consequential towards training stable DcGANs. The model weights need to be zero centered with slight increase of standard deviation(0.02). This technique may stabilize Generator and Discriminator network and possibly prevent the vanishing gradient problem and exploding gradient problems. In order to achieve this random variables in our model (the random noise vector) has been introduced. How weight initialization can affect the learning process in a neural network is presented below [114].

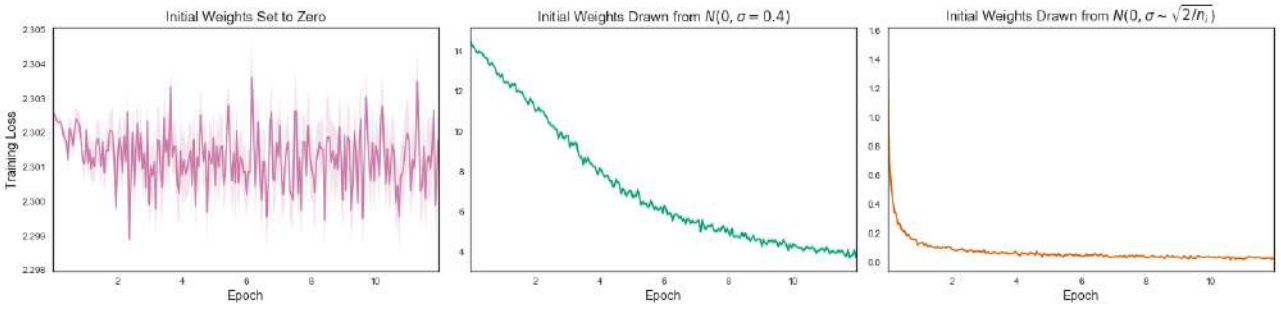


Figure 3.36: Effect of weight initialization on training DcGAN [114]

Further Optimization through Label Smoothing:

One regularization method that can be applied during training Generative Adversarial Networks is called Label smoothing. This optimization technique essentially prevents Discriminator network from being over-confident or under-confident in its predictions. If Discriminator network becomes too certain that the given image is real or doctored image that looks real, Generator can exploit that fact and continuously start to generate only images of that pattern and in turn, cease to improve. This problem can be resolved by setting the class labels to be in the range [0, 0.3] for the negative classes and [0.7, 1] for the positive ones. This may prevent the overall probabilities from getting very close to the two thresholds [114].

Label Noise:

This technique is also called Instance Noise. By adding a small amount of error to the labels (let's say 5%), this tends to make the true and predicted distributions more spread out and thus start to overlap with each other. This in turn makes fitting a custom distribution of generated images easier in the learning process [114]. The advantage of label smoothing and instance noise during training DcGAN can be presented below.

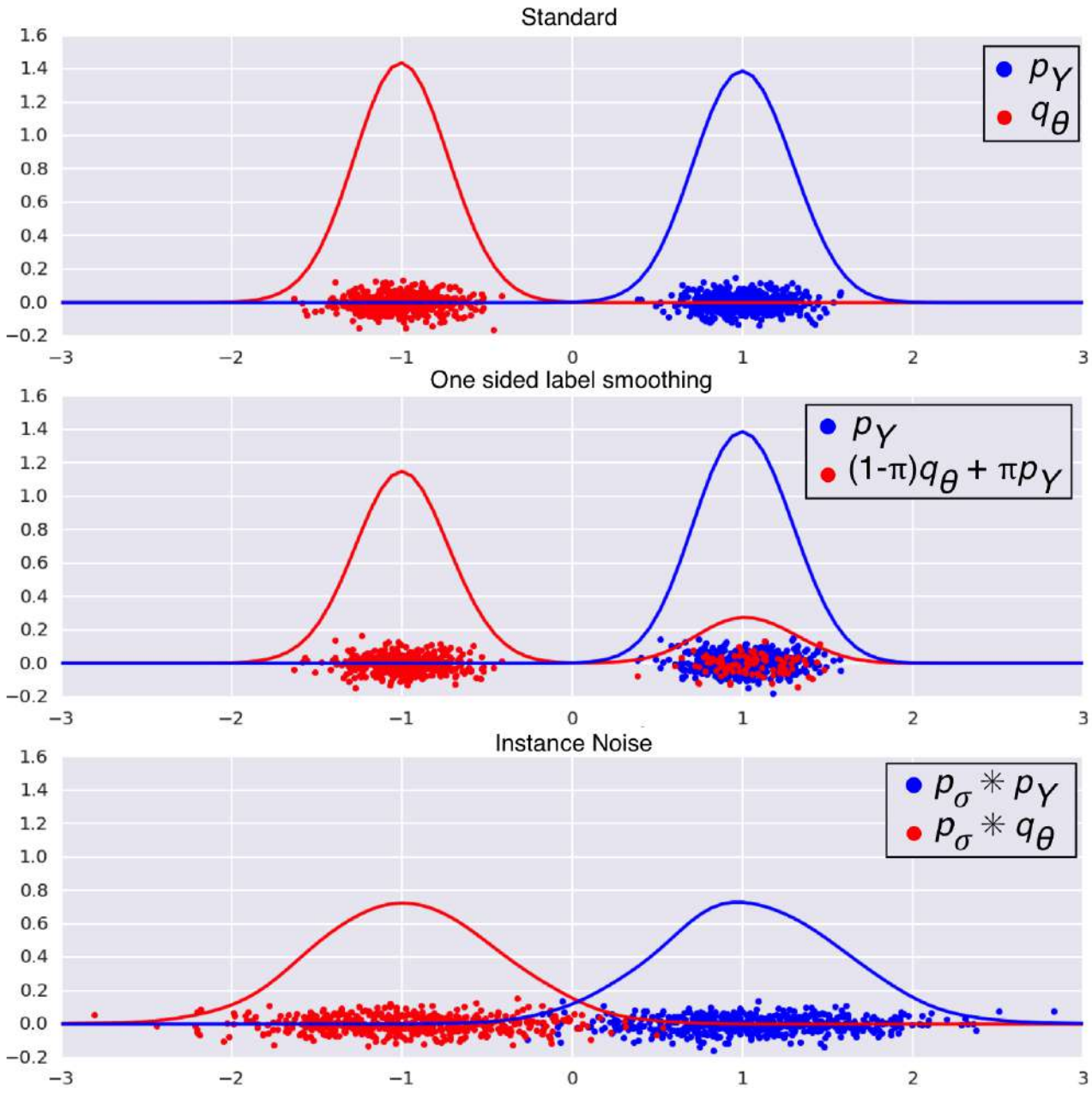


Figure 3.37: Effect of Label smoothing and label noise on training DcGAN [114]

3.12 Three-Dimensional Convolutional Neural Network for DeepFake Video Classification

Inspired by the recent success of 3DCNN in video classification for capturing spatio-temporal features, this convolution Neural Network has been implemented in this proposed methodology. As discussed in section 2.5, the spatio-temporal inconsistency in image frames of the video could discern a fake video from an authentic one. Rather than combining time-distributed CNN followed by RNN this methodology performs convolution operation on a sequence of images where the number of images in each sequence for a video is referred as

the time steps which is the third dimension for this network. Therefore, compared to parallel execution of CNN with distributed CNN and followed by RNN on those features, this network is able achieve all of it in one network sequentially for each 3D-convolution layer, which may be more consistent in practise. The architecture of the proposed model has been adopted from C3D [1] network, a simple, yet effective approach for spatio-temporal feature learning using deep 3-dimensional convolutional networks which had a recent success on a large scale supervised video dataset. A homogeneous architecture with small 3 x 3 x 3 convolution kernels in all layers is among the best performing architectures for 3D convolutional networks and therefore the authors of C3D demonstrated the learned features from C3D with a simple linear classifier outperforms state-of-the-art methods on 4 different benchmarks and are comparable with current best methods on the other 2 benchmarks. This motivated this project to adopt the architecture of C3D for deepfake video classification. Nevertheless, the full architecture of C3D could not be implemented due to limited computing resources. Therefore, a short and concise version of C3D model has been implemented as the proposed methodology for this project.

3.12.1 Proposed Model Architecture

The proposed methodology introduces 5 3d convolutional layer instead of 8 layers proposed in C3D. All of the convolution kernels have kernel size 3x3x3 like the authors of C3D have suggested and for this reason they have obtained very high accuracy. This concept has been adopted by the authors of c3D from the behaviour of 2D convolution models. The output channel sizes for each of the 3DCNN layers of proposed methodology are 64, 128, 256, 256, 256 respectively. Each of these 3D convolution layers are followed by a BatchNormalization Layer and a MaxPooling3D layer with pool size and strides as (2, 2, 2) except the first MaxPooling3d layer that has pool size and strides as (1, 2, 2). These parameters were adopted from c3D architecture. After these 10 layers (not including BatchNormalisation) a Flatten layer is introduced followed by two FC layer of 2048 nodes and finally 1 node for decision. Each FC layers were followed by a Dropout of 0.5 to avoid overfitting between FC layers. The optimization algorithm used for training the model has been ADAM with default parameters and binary cross-entropy has been implemented as our loss function. The model was trained for 50 epochs with a batch-size of 1 as this approach was computational expensive and exhaustive for limited computational resources available for this project. To avoid model overfitting Early stopping was also implemented in this project with a patience of epoch 15 due to low batch size.

Figure 3.38: Proposed Model Architecture of 3DCNN

Chapter 4

Evaluation And Analysis

The kaggle deepfake public data set as discused in chapter 3, had one repository for training and validation that includes 400 videos and metadata file while the other repository had only 400 videos for test set. Before any evaluation could commence, the labels (fake or real) for the videos of the kaggle deepfake test set was required as it was not included anywhere in the dataset. Later, it was discovered that kaggle deepfake public dataset has been developed from DFDC [116] (deep fake detection challenge) dataset by Facebook AI. DFDC [116] dataset contains 100,000 total clips sourced from 3426 paid actors and currently the largest dataset for deepfake detection. The metadata that has information regarding all the video clips in DFDC has been filtered (on column "split") to obtain the labels for the test set of kaggle dataset. The test set has a total 400 videos, however due to time and computational resource constraints only top 100 videos were considered for evaluation in this project.

4.1 Evaluation And Analysis of Proposed Pre-Processing Methodologies

4.1.1 Evaluation of Proposed Scaling Techniques on Loss Convergence

The performance of Multi-layer Perceptrons (MLP) trained with Back Propagation Artificial Neural Network (BP-ANN) method is highly influenced by the size of the datasets and the data pre-processing techniques used[102]. The learning algorithm such as mini-batch gradient descent based algorithm that iteratively minimizes the loss function for any neural network suffers from some drawbacks such as getting trapped in local minima, slow convergence and network stagnancy [117, 118, 119]. Early research on back propagation (BP) algorithms found improvement in performance of training neural networks on: (i) selection of better Loss functions [97, 98]; (ii) different choices for activation functions [98, 99] and, (iii) selection of dynamic learning rate and momentum [100, 101]. Later, it was found out that the techniques for Pre-processing the data also plays an important role in effecting the performance of Neural Networks, where pre-processing the data encourage high accuracy and less computational cost associated to the learning phase (better convergence) [102]. In this section, we will provide empirical evidence for our choice of proposed pre-processing technique compared to other mainstream pre-processing techniques. In order to compare the proposed approach with other mainstream pre-processing approaches, we applied a random key-frame selection (initial key-frame selection methodology) from Gaussian Distribution as discussed in section 3.10 on each videos for a total number of 154 videos (77 real and 77 fake). From each videos 17 key-frames were selected and further from each key-frame our proposed face extraction methodology was employed to extract faces from these key-frames. The extracted face images were reshaped into 224x224x3 resolution, this has been explained in chapter 3. Finally, several pre-processing techniques have been applied to these face images and for each technique the data was fed to the

same proposed deepfake detection model. Transfer Learning with pre-trained Xception model which was discussed in section 3.8.3 has been deployed for this experiment as our reference model. The hyper-parameters and parameters of the model were exactly the same as discussed in chapter 3, during training for each individual pre-processing technique. The evaluation of this experiment might provide us interesting insights about a faster way of converging to local/global minima with the help of a pre-processing technique. The results of these experiment can be presented as follows,

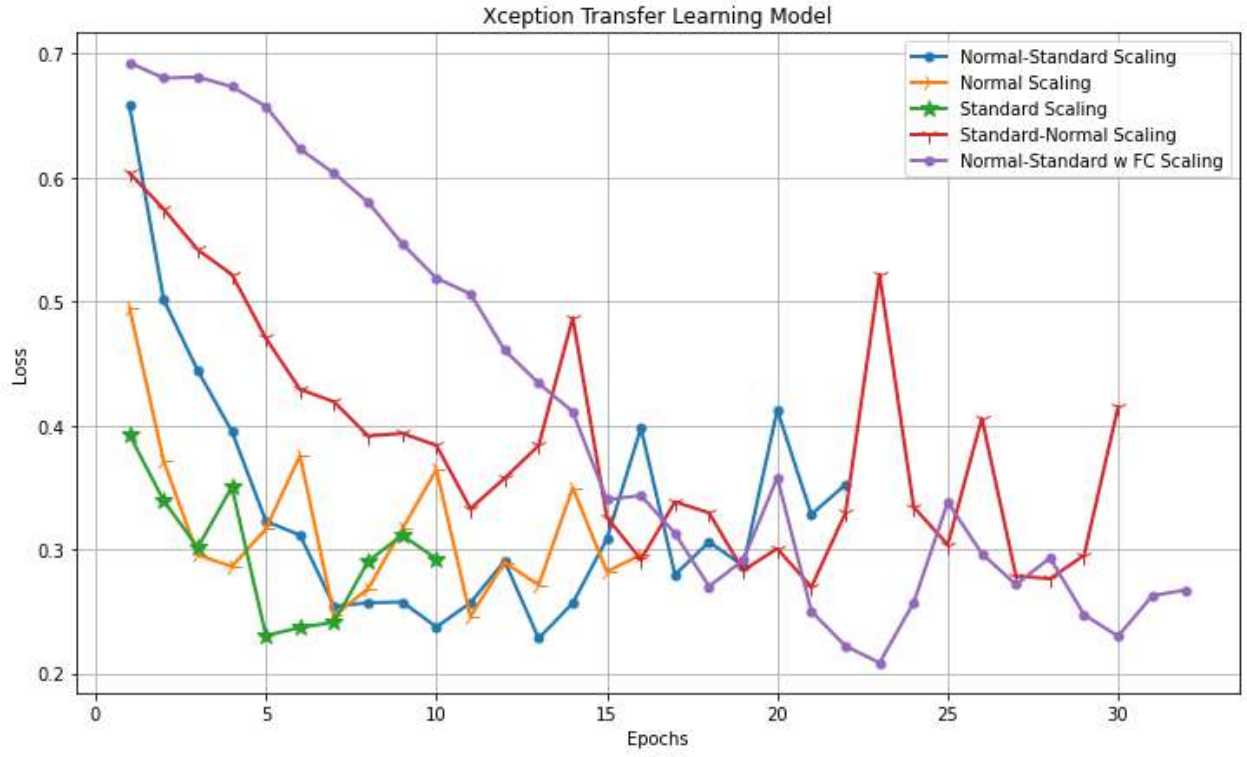
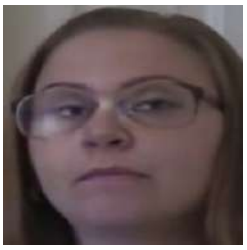
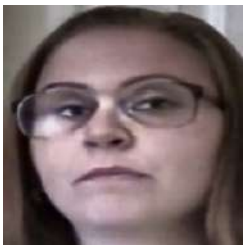


Figure 4.1: Effect of Pre-processing techniques (Scaling) in Loss Convergence

From Figure 4.1 we can observe 5 different pre-processing techniques have been compared including our proposed pre-processing techniques. The first pre-processing technique is a combination of Normalization followed by a per image Standardisation which is shown in figure by the blue line. The second pre-processing technique was only applying Image normalisation and nothing else which is presented in the figure by the orange line. The third technique was only using per image Standardisation which is represented by the green line. The fourth technique was a combination of applying per image Standardisation first followed by image normalisation on data, which is represented by the red line. Finally the fifth pre-processing technique is a special case which only applies to proposed models with transfer leaning approach. The fifth pre-processing technique applies Normal scaling followed by per image standardisation which was our first case but additionally the data is also scaled (scaling before FC layers) after the prediction from the pre-trained models and further fed to FC layers in a new model. Normalization followed by per image standardisation is applied to data before feeding this to the model with FC layers. Therefore the fifth approach can be represented as Normal-standard-Normal-Standard. From the figure 4.1, we can observe only per image standardisation has very fast convergence on 5th epoch, however the loss value of normal-standard combination is even lower than only standardisation on 13th epoch. This difference although seems minimal when this experiment was repeated few times, based on the randomness of the optimization process sometimes the difference could be noticeable. Nevertheless, the loss value is minimum compared to only standardisation for each case, which



(a) Fake Image without Boosted Contrast



(b) Fake Image with Boosted Contrast

Figure 4.2: Effect of Boosted Contrast on Fake Images

is also clear from this figure. Finally, if we consider the fifth pre-processing approach, this may have reached to it's global minima at the cost of 23 epochs. Therefore, our proposed pre-processing approach which was Normal-Standard scaling empirically shown improvement in loss convergence which would project more accuracy for our model. This is to be noted, the Normal-standard-Normal-Standard scaling is also our proposed approach but is limited to only proposed transfer learning models.

4.1.2 Evaluation of Boosted Contrast on Images for Baseline Model

Contrast is the distinction between lighter and darker pixels of an image, and therefore it makes objects or details of an image more obvious due to the pixel differences. Increasing contrast on an image will increase the difference between light and dark pixel values so bright pixel will become brighter while dark pixels will become darker. Therefore, boosted contrast on images could maximize accuracy on object or face detection as the edges of an object and background can be separated easily due to the differences in pixel values. Nevertheless, this also means every details of the image can be well analyzed and therefore to analyze the details of the faces in our project this methodology could be really useful. Previously in chapter 3, we discovered that in fake or doctored images the facial skin appears blurry which suggests that pixel intensities or values in that region have very low difference in pixel value. Therefore, boosting the intensity of the images might unveil the facial artifacts or noise distinctly.

If we take a closer look on Figure 4.2 we can see after boosting contrast on a fake image the skin tone of the subject in the forehead becomes darker while the cheeks are bright however the picture without contrast is very hard to discern if there's any artifact present in the face. Therefore to test this hypothesis, we extracted face images both with contrast and without contrast then evaluated the accuracy on each of these data on our baseline model. This is to be noted, in this experiment the key-frame selection from the videos were random based on Gaussian distribution. The result of this experiment can be presented below,

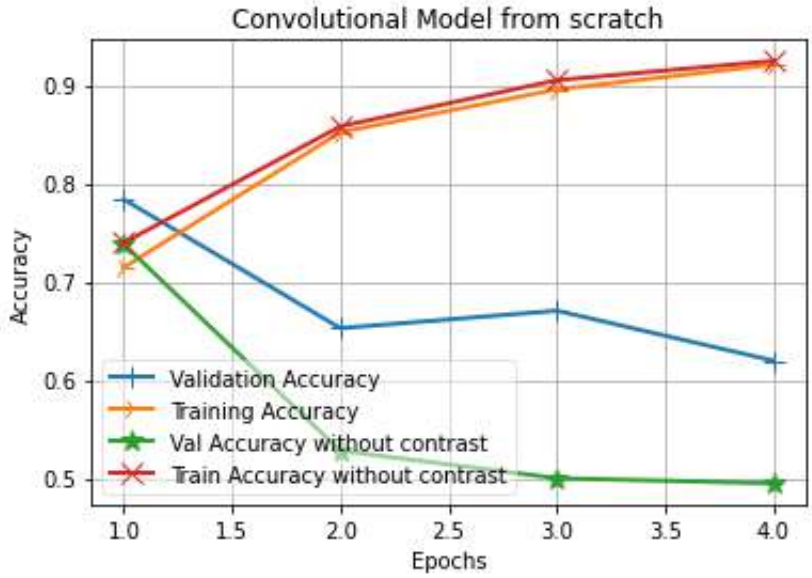


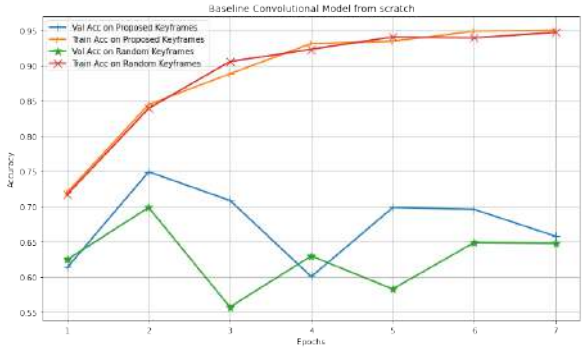
Figure 4.3: Effect of contrast on Baseline Model

From Figure 4.3, we can clearly suggest that boosting contrast on images improves performance on our validation data. Therefore, from the next section we'll work with images with boosted contrast as our default choice.

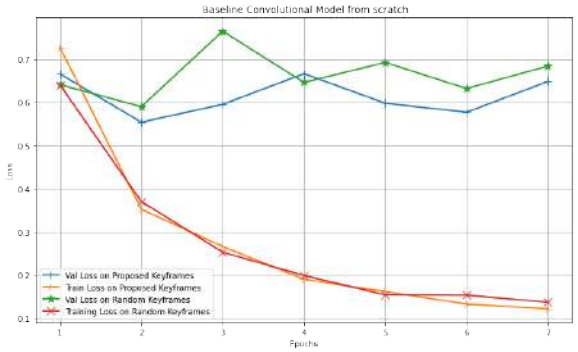
4.2 Evaluation And Analysis of Key-Frame Extraction Techniques on Baseline Model

In chapter 3, we discussed the proposed methodology for extracting key-frames from the video files on our dataset. In that section additionally it was explained that we initially applied a random key-frame extraction methodology that selects key-frame indexes based on 17 random numbers between 1 to 299 from a Gaussian distribution. The random key-frame extraction methodology although extracts key-frames very fast this might not be an efficient way to eliminate redundant image frames from the video files. We also covered the reasoning behind choosing our proposed methodology in that section. In this section, we will compare the performance of these methodologies in terms of improvement of performance on our baseline model. The methodology for baseline model for this experiment has already been explained in chapter 3 section 3.4. We could have selected any other deepfake detection proposed model for this experiment, nevertheless baseline model is selected as it has fewer convolution layers and therefore it saves a lot of training time and computational power compared to our other proposed deepfake models. In order to be absolutely certain that our proposed methodology improves baseline model performance compared to random key-frame methodology we have conducted two experiments on baseline convolution model from scratch. In our first experiment we have implemented the baseline model with all the hyper-parameters same as explained in section 3.7 in chapter 3, except the batch-size. In our first experiment we have devised the batch-size of our data to be 128. The evaluation of these methodologies can be presented as follows,

From Figure 4.4 (a) we can observe that training of baseline model for both the methodologies were finished by the end of epoch 7. If we can recall the implementation details of baseline model from section 3.4 this happened as a result of Early stopping with a patience of 5 epochs. We can observer from both the loss chart on 4.4 (b) or 4.4 (a) the loss or accuracy of the model did not improve from epoch 2 to epoch 7 hence



(a) Effect of Proposed Key-frame Extraction on Baseline Model Accuracy



(b) Effect of Proposed Key-frame Extraction on Baseline Model Loss

Figure 4.4: Evaluation of Proposed Key-Frame Extraction vs Random Key-Frame Extraction on Baseline Model(Batch-size 128)

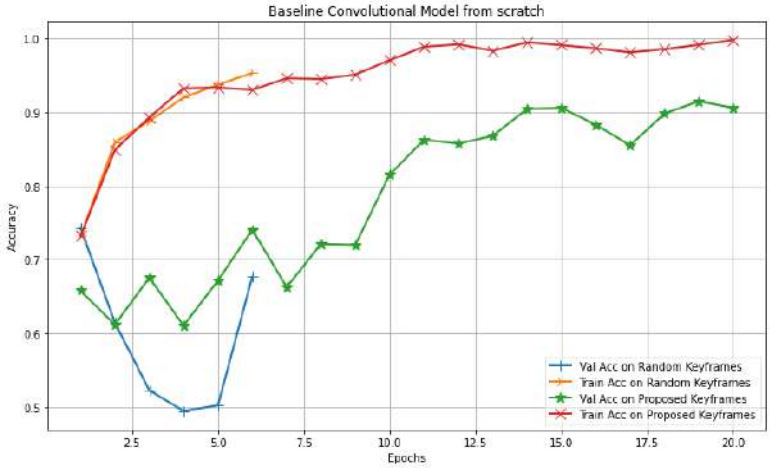
the training was ceased to avoid data over-fitting. From both accuracy and loss plots, we can observe that our proposed methodology on baseline model performed better on validation dataset for all epochs except epoch. For training deep neural networks, usually the epoch with highest accuracy or minimum loss is saved as a checkpoint for the model. Hence, we can confirm from this experiment and observing the performance of epoch 2 in figure 4.4 (a) and (b) that our proposed methodology performs better than random key-frame selection.

To be absolutely certain about these observations, we conducted another experiment with Baseline model but this time with a batch-size of 64 which is constant for all our proposed deepfake detection models. The results can be presented in Figure 4.5.

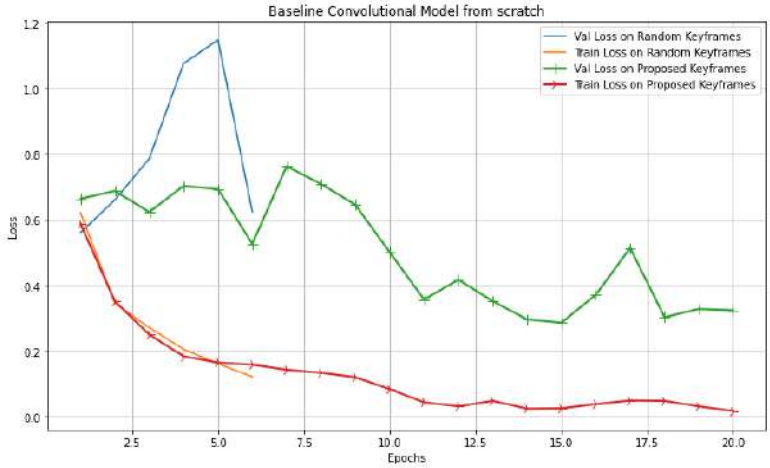
We can observe, From Figure 4.5 (a), a significant improvement in accuracy with proposed key-frame extraction methodology compared to random key-frame extraction from Gaussian distribution. Additionally, the training process lasted for 20 epochs as the validation loss keep improving during the training process unlike random key-frame extraction. In the case of random key-frame extraction the loss did not improve after epoch 1 and therefore as mentioned above early-stopping ceased training to avoid data over-fitting. The patience of 5 epochs for early-stopping has been applied in both of our methodologies. Therefore, we can conclusively claim our proposed methodology is superior compared to random key-frame extraction with empirical evidence.

4.3 Evaluation of Pritam's Baseline Model for DeepFake Detection

Previously in chapter 3, we discussed the methodology to employ our baseline convolution model from scratch that includes details of convolution layers, regularization methodologies and choice of parameters and hyper-parameters during the training process. We also mentioned 3 variations of this network 16-16-32-32(variation 1), 32-32-64-64(variation 2) and 64-64-128-128 (variation 3) implemented in this project to analyze the versatility of this network. In this section, the evaluation of these model will be explained by identifying key insights from training-validation accuracy and loss charts. The accuracy and loss charts for training and validation process for each epoch of training has been presented in figure 4.6, 4.7 and 4.8 respectively. From the corresponding figure we can observe our baseline model(var 1) has reached a maximum validation accuracy of 86% at epoch 6 with a corresponding loss value of .36 on our validation set during the training while var 2 and var 3 has



(a) Effect of Proposed Key-frame Extraction on Baseline Model Accuracy



(b) Effect of Proposed Key-frame Extraction on Baseline Model Accuracy

Figure 4.5: Evaluation of Proposed Key-Frame Extraction vs Random Key-Frame Extraction on Baseline Model(Batch-size 64)

reached maximum accuracy of close to 90% on 15th and 12th epoch respectively. The training process ceased by the end of 11, 19, 17 epochs respectively for all 3 variations due to the early stopping of patience 5 epochs which suggests that the validation loss did not improve from epoch 6, 14, 12 respectively. The training accuracy reaches close to 98% with a minimal loss below 0.1 for all our baseline convolution models from scratch. We can conclude our baseline models have performed fairly well on deepfake challenge dataset during training, however this hypothesis could further be tested on a test dataset which will be discussed and compared with other proposed models later in this chapter.

4.4 Evaluation of Proposed Models with Transfer Learning for DeepFake Detection

The methodology for proposed deepfake detection models which augments transfer learning (on VGG16 [41], ResNet50 [40], Xception [42], InceptionResNetV2 [107]) along with FC layers were discussed along with implementation details in section 3.8. In this section, we will analyse the performance of these models on our deepfake challenge training validation dataset. Like we discussed before, all of these models were trained with exactly same training parameters and hyper-parameters which was discussed in chapter 3 so that all these models can be compared with each other.

The accuracy and loss charts for our proposed VGG16 model is presented in figure 4.9. We can observe from figure 4.9, that our proposed VGG16 model with transfer learning has reached a maximum accuracy of almost 94% on validation dataset on epoch 19 with a corresponding loss value of just below 0.2. We can clearly see that training validation curves for both accuracy and loss charts do not have very high variations within two consecutive epochs. Therefore we can infer that the model optimizes the loss function very consistently as the gradient of the loss function converges smoothly without too much variations for consecutive epochs. The training accuracy and loss charts appear to be very consistent for proposed VGG16 model.

Figure 4.10 presents the accuracy and loss charts for proposed ResNet50 model on training and validation dataset. We can observe some intriguing results for this model, the maximum validation accuracy achieved for this model is around 93% on 10th epoch while the minimum loss found is .25 on 9th epoch. This might appear to be an anomaly or discrepancy however, the accuracy might not always signify the performance of a model which is why other performance metric such as precision, recall, F1 score is considered to evaluate the performance of a model. Nevertheless, the loss value in the training process is an accurate measure to confirm if the model is still learning or the model getting over-fitted. Therefore, in this case our proposed ResNet50 performs best on 9th epoch and the corresponding accuracy for 9th epoch is about 91-92% which has been saved as the model checkpoint for this project. It can also be noted that proposed ResNet50 stopped training any further after 14th epoch which is exactly 5 epochs after 9th epoch due to early-stopping implemented in this project. The training accuracy and loss values as presented in the figure 4.10 reaches very close to 98% and 0.03 respectively, it would be interesting to see if this is a result of model over-fitting or not while evaluating the performance of the model on our test-set.

The performance of proposed Xception model has been delineated in terms of accuracy and loss values for each epoch of training in figure 4.11. The results demonstrated in this model is intriguing due to it's fast convergence of gradient on just 8th epoch. We believe this is due to Batch-Normalisation Layer within Xception model architecture in between convolution layers. There is no surprise that Batch-Normalisation plays an important role in CNN in terms of not only faster convergence of gradients but also it provides a regularization effect that avoids model overfitting. We can observe from the corresponding figure, this proposed model

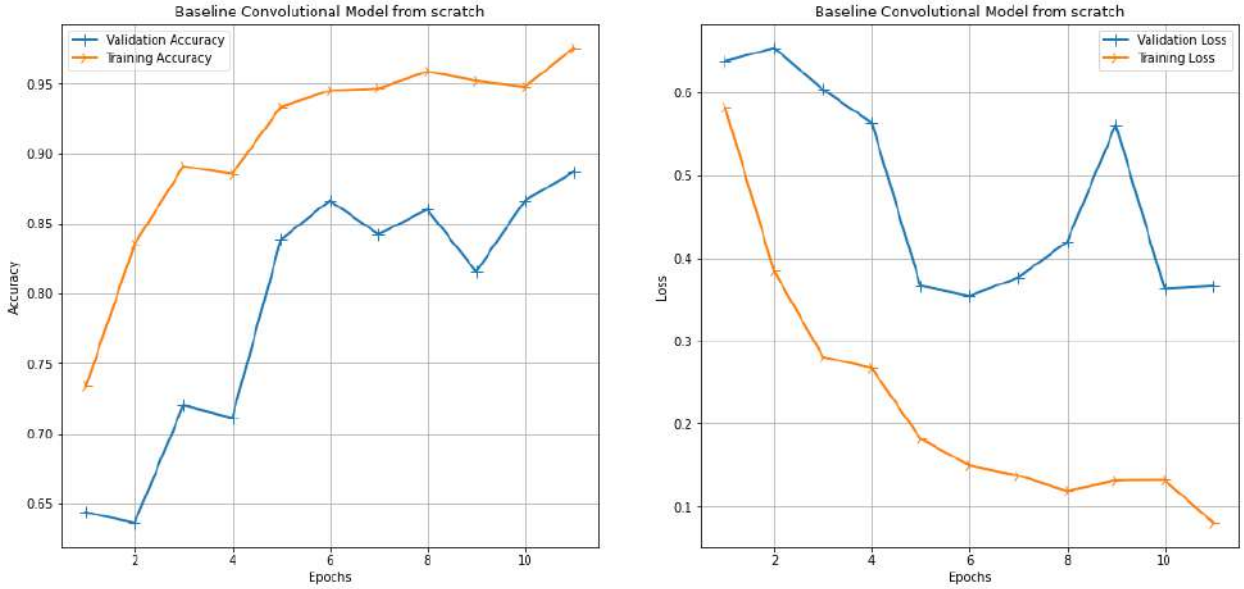


Figure 4.6: Evaluation of Proposed Baseline Convolutional Model from Scratch(var 1)

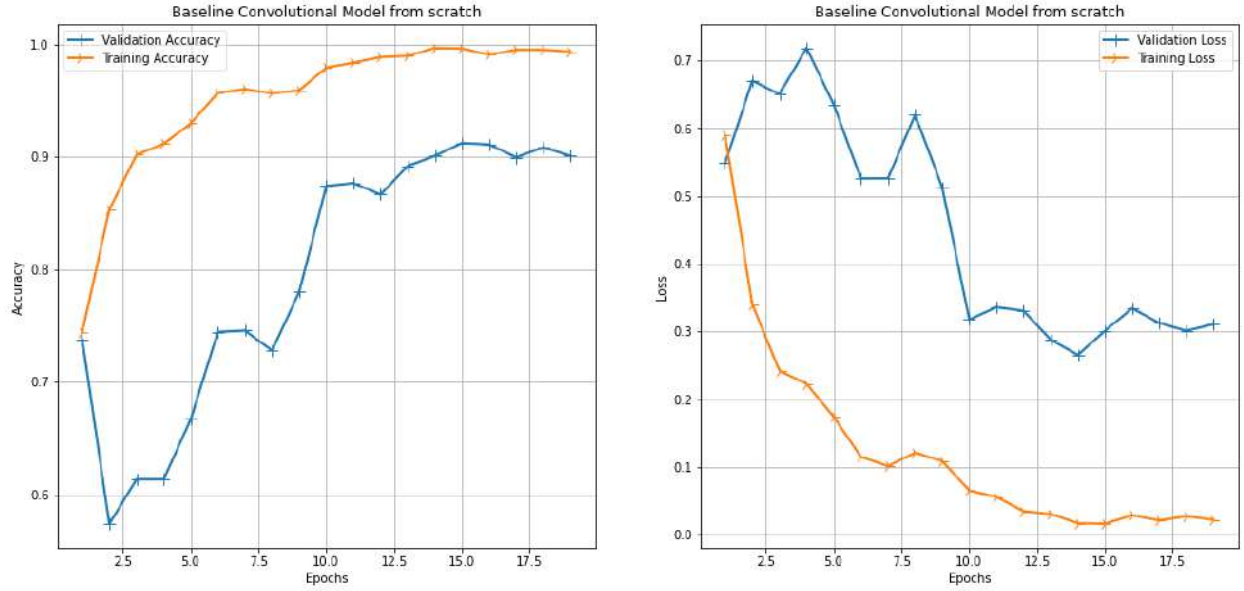


Figure 4.7: Evaluation of Proposed Baseline Convolutional Model from Scratch(var 2)

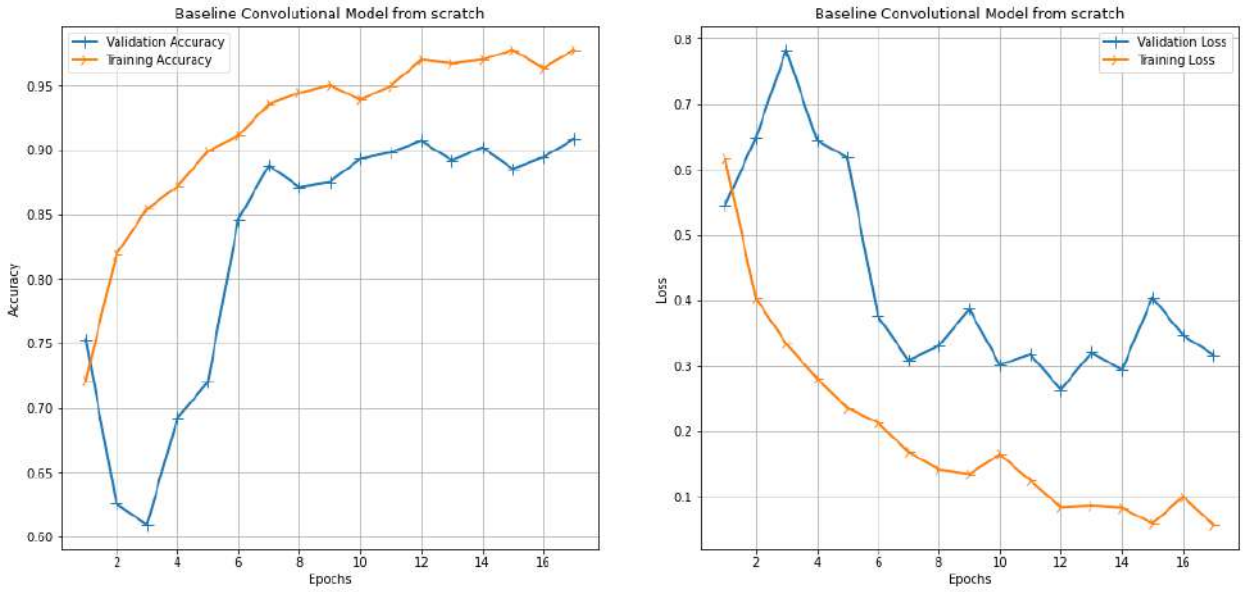


Figure 4.8: Evaluation of Proposed Baseline Convolutional Model from Scratch(var 3)

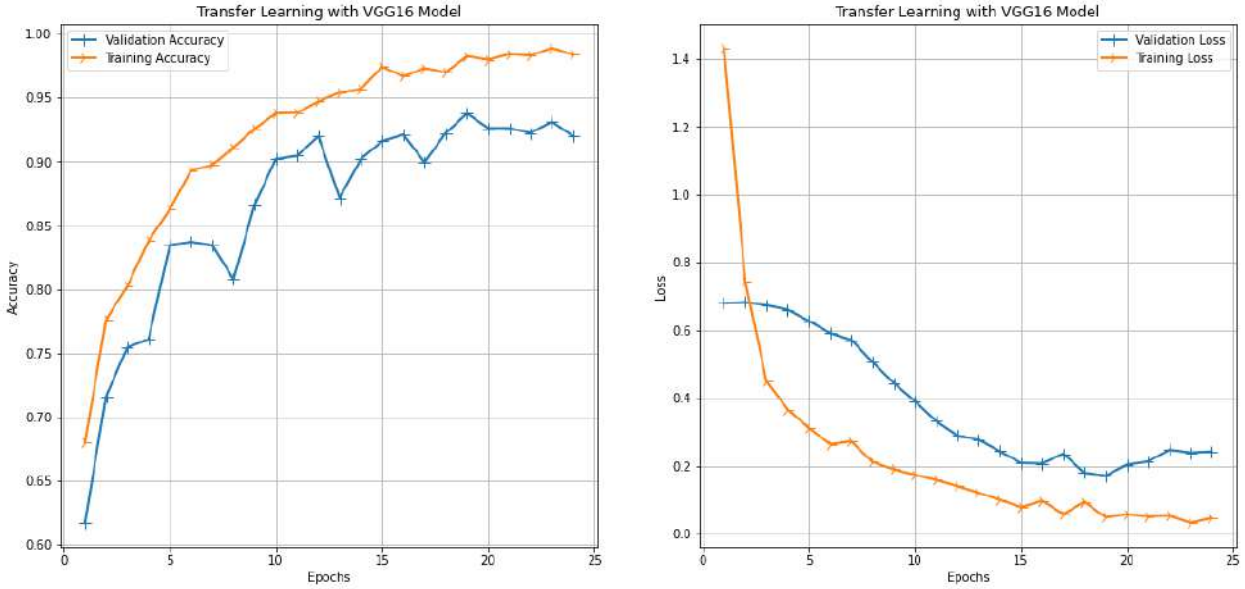


Figure 4.9: Evaluation of Proposed Transfer Learning with VGG16 Model

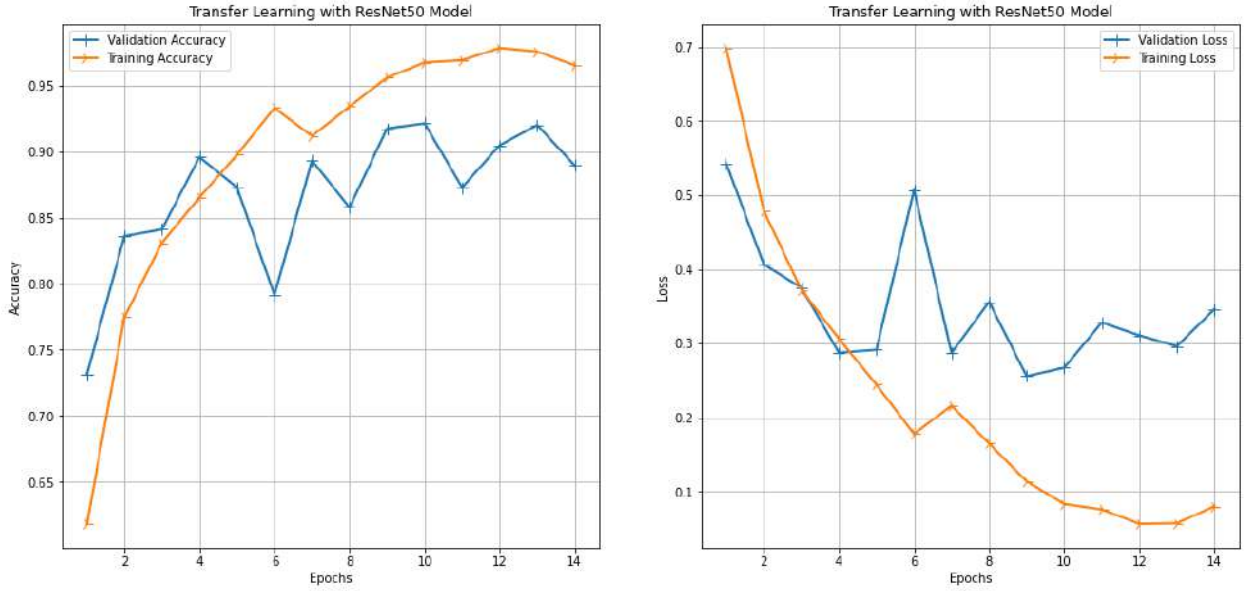


Figure 4.10: Evaluation of Proposed Transfer Learning with ResNet50 Model

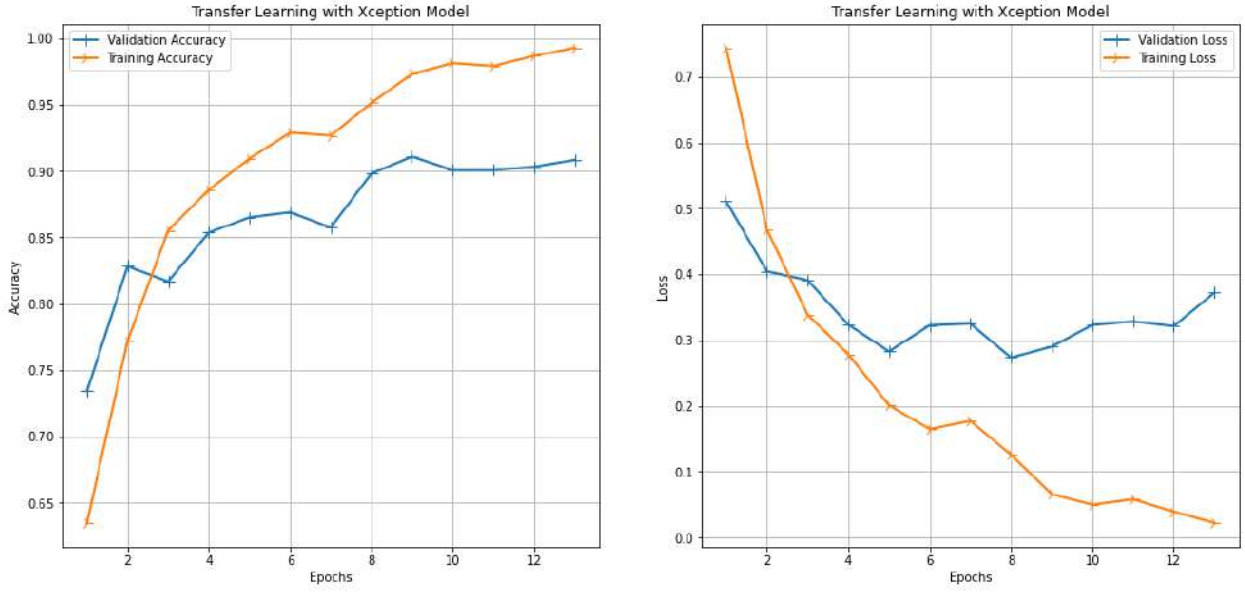


Figure 4.11: Evaluation of Proposed Transfer Learning with Xception Model

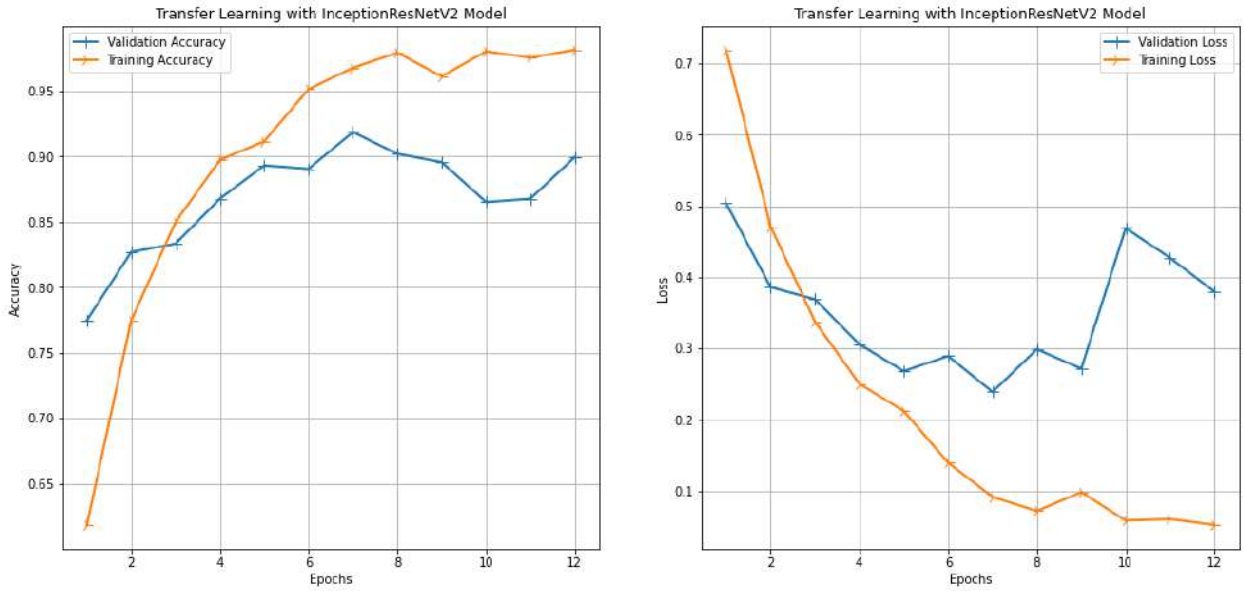


Figure 4.12: Evaluation of Proposed Transfer Learning with InceptionResNetV2 Model

has achieved a validation accuracy of 91% on epoch 9 with a loss value of just under 0.3. We can find a similar contradiction which was explained earlier that model accuracy increases while the model ceases to learn further. Therefore epoch 7 for porposed Xception model can be saved as model check-point in this project. The training accuracy and loss values as presented in the figure 4.11 reaches very close to 98% and 0.04 respectively, it would be interesting to see if this is a result of model over-fitting or not while evaluating the performance of the model on our test-set.

Figure 4.12 demonstrates the training validation performance of proposed InceptionResNetV2 model for each epoch with a batch-size of 64 which has been our default parameter for all the experiments explained from section 4.3. The performance of the proposed model also suggests faster convergence of gradient of the loss function due to application of BatchNormalisation as discussed previously. In this case, the proposed model has converged on epoch 7 and ceased training at the end of epoch 12 due to early-stopping as explained previously. This model is especially very deep in nature in terms of number of convolution layers however BatchNormalisation and residual layers helps this model to avoid vanishing gradient problems or model over-fitting. We can observe from figure 4.12 that the our proposed model has obtained a validation accuracy of 92% on 7th epoch and the corresponding loss (0.25 approximately) for this epoch is also minimum in this case. The training accuracy and loss values as presented in the figure 4.12 reaches very close to 98% and 0.02 respectively, it would be interesting to see if this is a result of model over-fitting or not while evaluating the performance of the model on our test-set.

4.5 Evaluation of Proposed MESONET Model for DeepFake Detection

In section 3.9, we discussed the architecture and the proposed methodology for MESONET- a compact facial video forgery detection network implemented in this project from scratch. This network has been designed especially to detect forged images and videos efficiently by it's authors as explained in chapter 3. Therefore it would be really interesting to see how the performance of this proposed network compares with our previous proposed methodologies.

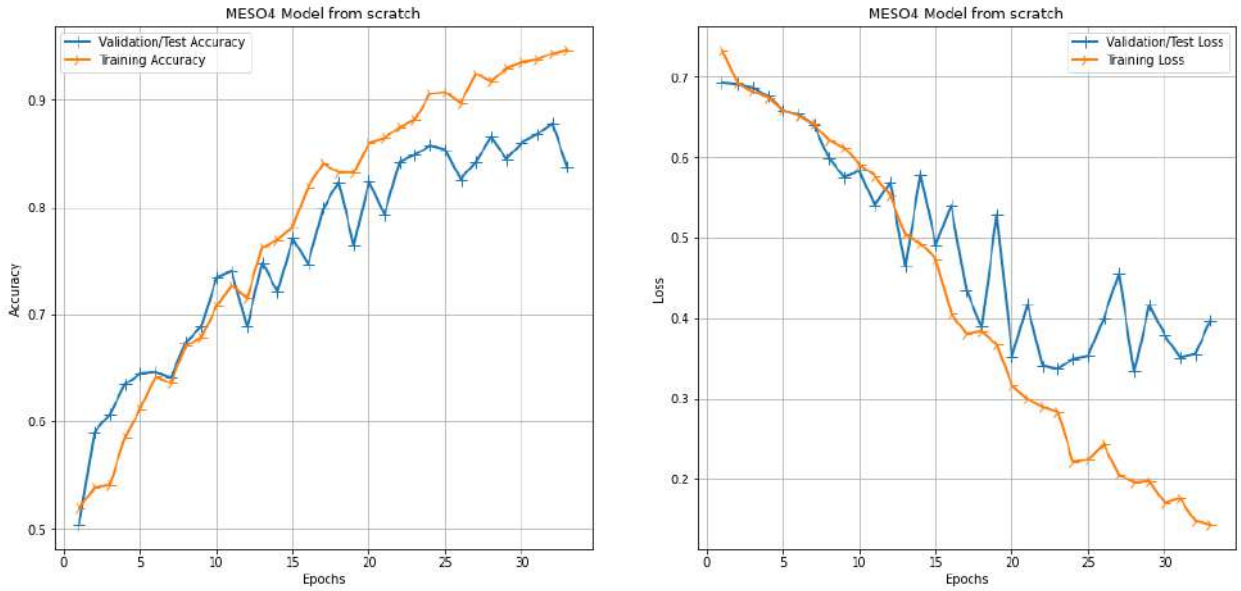


Figure 4.13: Evaluation of Proposed MESONET (MESO4) Model

The accuracy and Loss charts of MESO4 Network has been presented in figure 4.13, the training and validation performance have been demonstrated by the orange and blue line respectively. We can observe that training and validation accuracy and loss curves have been very close to each other compared to all other proposed methodologies explained above. This may explain the chances of model overfitting is minimal in this case. We can also observe some sharp downfall in validation accuracy and increased loss in between epochs 11-12, 13-14, 18-19 and 33-34. The possible explanation for this could be the gradient of the loss function has many troughs and crests and therefore the optimization algorithm is adapting it's learning rate and momentum so that it does not stuck at a local minima. Another probable explanation for this behaviour could be the model could not accurately predict the outcome on that particular batch of images in those epochs. Nevertheless, the model has performed very well considering it has achieved a maximum validation accuracy of 87% and a loss value of .32 on epoch 28. Although the performance of our baseline models during validation is pretty close to the performance of this proposed network, the training accuracy of the baseline models were higher compared to this which might suggest model overfitting. The maximum training accuracy for this proposed model has reached about 94% with a loss value of close to 0.1 on 33rd epoch. The hypothesis regarding more training accuracy on our previous methodologies could be a result of model overfitting can only be proved if the performance of these models could be compared with each other on our test dataset.

4.6 Evaluation of Proposed Time-Distributed CNN with LSTM Model for deepfake Detection

Previously, most of our proposed methodologies were limited on predicting each key-frames (images) rather spatio-temporal relationship between these key-frames in a video were not considered in those approaches. In this section, our proposed time-distributed CNN with LSTM model, discussed in section 3.10 evaluates sequence of images rather individual key-frames and this approach is compared with the performance of our previous methodologies. In this project, two variations of this proposed model have been trained and validated for comparison. 1> sequential approach - which suggests pre-trained exception model predicts low

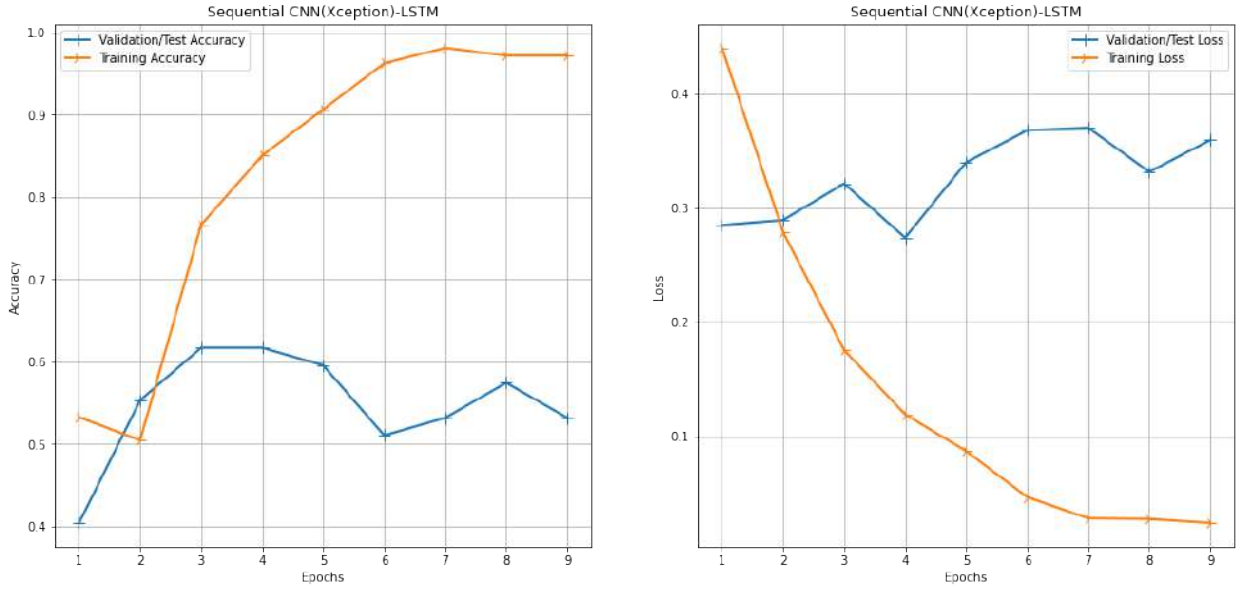


Figure 4.14: Evaluation of Proposed Time-Distributed Xception with LSTM Model(Two Sequential Networks)

level image features which is further trained into a separate LSTM based network, so there are two networks in this case that's why we named it sequential approach. 2> Integrated approach- This approach introduces an integrated model that combines Xception model with LSTM layers and trains this consolidated model as one single neural network. The evaluation of both variations of this proposed model has been presented in figure 4.14 and 4.15.

It can be observed from figure 4.14 the proposed sequential CNN with LSTM model has reached a maximum accuracy of 62% on epoch 4 with a corresponding loss value of 0.27 approximately on validation data. The curves of training and validation loss represented by orange and blue line respectively follows quite a different pattern from epoch 3 to epoch 9. Looking at the accuracy and loss charts from figure 4.14 it is clear that training accuracy is going very high while the validation accuracy does not improve too much. This suggests either the model might be overfitting or the minibatches from each epoch has different data distribution. This hypothesis can be confirmed based on the model's performance on our testset which will be discussed later in this chapter.

The training-validation of proposed integrated CNN with LSTM approach terminated by the end of 15th epoch due to implementation of early-stopping with a patience of 5 epochs. Therefore, the gradient of the loss function for this proposed approach did not improve after 10th epoch. It can also be observed from figure 4.15, the maximum accuracy (about 56%) of the model on validation data has been reached on 10th epoch and the minimum loss (close to 0.24) found on the same epoch. Therefore, the minimum loss value on 10th epoch should be saved as our model checkpoint in this case. It can also be observed that validation accuracy drops sharply from seventh to eight epoch and the corresponding training accuracy increases steadily. One possible explanation for this could be the gradient of the loss function for a specific batch of videos on those epochs could not be minimized. The difference between the sequential approach and the integrated approach was not only the architecture of these models but also the pre-processing (normalisation and per image standardisation) step which was applied only to the sequential approach after the base model (Xception in this case) predicted low level features from images. We believe pre-processing step in the sequential approach might help with faster convergence of gradient of the loss function of the mode3.

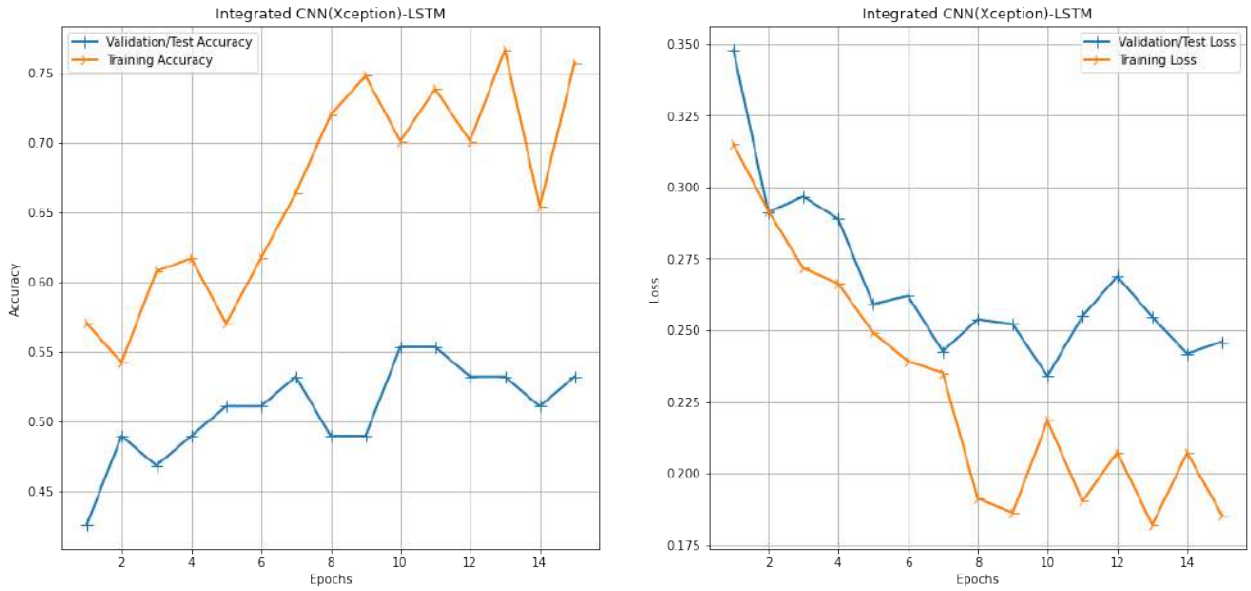


Figure 4.15: Evaluation of Proposed Time-Distributed Xception with LSTM Model(An Integrated Approach)

4.7 Evaluation of Proposed Three-Dimensional CNN Model for DeepFake Detection

The alternative approach that could also capture spatio-temporal relationship between an image sequence is three-dimensional CNN or 3DCNN, where the third dimension being time steps or in our case the number of key-frames in each video (17 for this project). The key difference between CNN with LSTM and this proposed methodology is that the spatio-temporal relationship of images are being considered at convolution layers in this case. The evaluation of our proposed 3D-CNN model has been presented in figure 4.16.

The methodology to implement 3D-CNN is little different compared to others due to the amount of computing resource it requires at the time of training. Therefore, the batch-size for this model needed to be very low to increase the stability of this model. Increased batch-size requires more Dynamic memory (RAM) and more GPU moemory as well. All of our experiments were done using GTX 1660 ti with 8 GB of RAM, however this experiment was beyond scope of the limited resources available for this project. Nonetheless, limiting the batch-size to 1 made the model stable but the variation in accuracy and loss is high, since increased batch size not only helps with faster convergence of gradient but also generalizes the loss much better. Therefore, the patience for early stopping in this case were set to 15 epochs than 5. Figure 4.16 presents the accuracy and loss charts of proposed 3DCNN from 5th epoch as the loss on first epoch was very high and as a result the plot was not able to distinguish the training and validation curve after 20th epoch, both the curves appeared to be overlapped. In order to resolve this challenge the plot starts from fifth epoch. It can be observed from the figure that, this proposed model has reached it's minimum loss (close to 0.73) at 39th epoch and 15 epochs later training was ceased due to early stopping. The validation accuracy of the model for the corresponding epoch is about 58% while the maximum accuracy has been 66% on 29th epoch. The high variations in training and loss curve is evident since the batch-size is only 1 for this model. Looking at the training and validation Loss curves it can be inferred that chances of overfitting for this model is quite low. The training accuracy of this model does not go beyond 78% which also suggests the low chances of model over-fitting. This hypothesis will be tested later in this chapter with the help of a test-set.

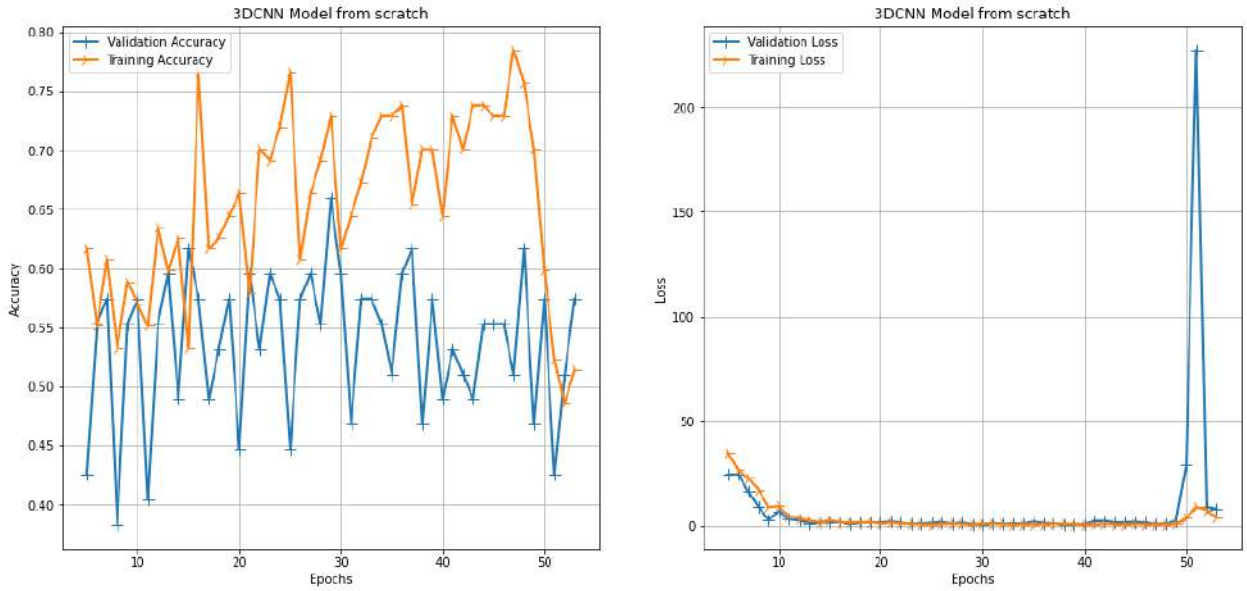


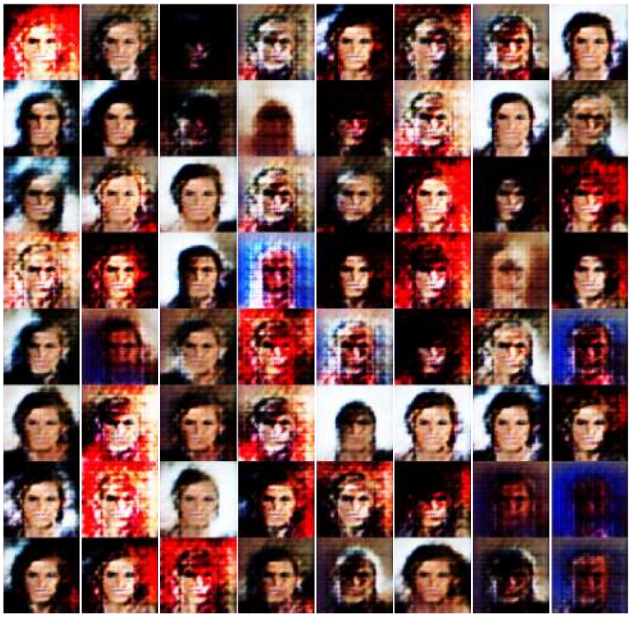
Figure 4.16: Evaluation of Proposed Three-Dimensional CNN Model

4.8 Evaluation of Proposed Generative Adversarial Network

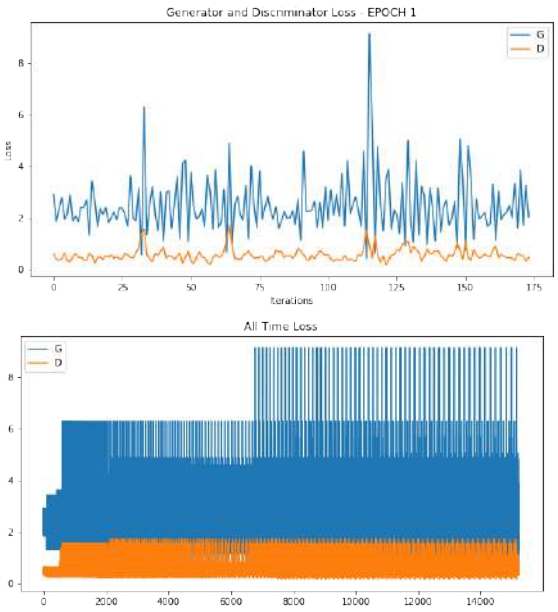
Previously, in section 3.8 we discussed the implementation details and methodology to develop our generative adversarial network that initially has been trained on celebrity face dataset. The idea behind this is, as Generator network learns to improve in generating more realistic looking fake images, the discriminator network also learns to distinguish between noisy fake images with facial artifacts and real images. This way Discriminator network learns better in general to distinguish between a real image and a doctored image irrespective of specific distribution of a particular image dataset. After training on celebrity face images the Discriminator network is further trained on our deepfake challenge dataset to learn the specific distribution of our image dataset and we believe this strategy could improve the performance of Discriminator network to differentiate between fake and real images. Therefore we trained the entire GAN for generating deepfake images with celebrity faces and later only pre-trained discriminator network was trained on our deepfake dataset.

4.8.1 Training GAN on celebrity-face dataset

Training Generative Adversarial network has been very challenging for this project as it requires huge amount of computational resources and time. Nevertheless, training GAN for 280 epochs on a laptop with 8 GB RAM and GTX 1660 ti GPU, it took more than 5 hours to train the adversarial neural networks that could produce realistic looking fake images of celebrities. The images generated by the Generator network at epoch 1 and epoch 280 has been presented in figure 4.15 (a) and 4.16 (a) respectively. It can be observed from these figures that starting at epoch 1 all of the images appeared to be very noisy and one could hardly distinguish if any face exist in those images. However we see a huge difference looking at the face images on epoch 280, not only the images look very realistic to human eyes but also the diversity of all 64 images generated is very impressive. It is to be noted that if the Generator has produced an image from a distribution that looks real and if Discriminator network becomes fooled by the image, sometimes Generator tends to exploit that vulnerability and produce images from the same distribution every time. This reduces the diversity of the Generator and simultaneously both Generator and Discriminator fails to learn something new in future.



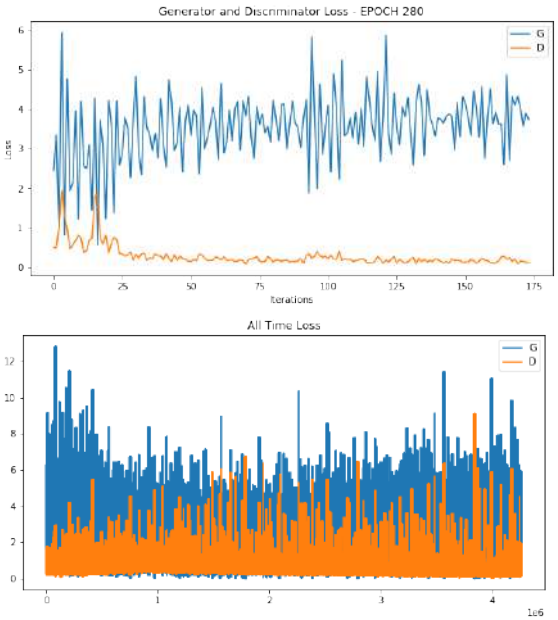
(a) Face Images Generated at Epoch 1



(b) Discriminator and Generator loss at Epoch 1

Figure 4.17: Training Proposed GAN on celebrity faces (Epoch 1)

(a) Face Images Generated at Epoch 280



(b) Discriminator and Generator loss at Epoch 280

Figure 4.18: Training Proposed GAN on celebrity faces (Epoch 280)

To avoid such circumstance, label smoothing and label noise has been implemented in this project, this has been discussed on chapter 3 in details. It can be clearly observed the 64 images generated on epoch 280 has very high diversity among the faces. The training loss for epoch 1 and epoch 280 has been presented in figure 4.15 (b) and 4.16 (b) respectively along with all time loss of discriminator and Generator on corresponding epochs respectively. The mean loss of Discriminator on epoch 1 has been found to be 0.37 with a standard deviation of 0.36 while the mean loss of Generator network on the corresponding epoch has been 2.1 with a standard deviation of 1.4. The all time loss chart presented on 4.15 (b) demonstrates similar insights here. The orange curve shows the loss of Discriminator while the blue curve shows the loss of Generator in figure 4.15 (b) and 4.16 (b). The mean loss of Discriminator on epoch 280 has been found to be 0.15 with a standard deviation of 0.05 while the mean loss of Generator network on the corresponding epoch has been 2.7 with a standard deviation of 0.7 from figure 4.16 (b). These insights clearly indicates that Both Discriminator and Generator have improved their loss value during training for 280 epochs. Nevertheless, if the loss of discriminator goes minimal like we could observe on 280th epoch, the generator loss definitely will increase as a result. The reason behind this is both the network is trying to minimize their loss in a zero-sum game so if one wins the other will lose. This also depends on a specific image batch which is being trained as well. For example if the Generator produces a fake image that might appear indistinguishable among the batch of real images trained during convolution layers in Discriminator but for different set of images, Generator might not learn well from that distribution. If we observe the all time loss plot from figure 4.16 (b) there has been many times in the middle of the training process where Discriminator loss went higher than Generator Loss and the vice-versa is also true during many iterations of the training. Figure 4.15 (b) and 4.16 (b) demonstrates the losses of Generator and Discriminator for each training step or epoch (epoch 0 or epoch 280 here) followed by the all time loss of Generator (G) and Discriminator (D) till that corresponding epoch. The top figure of 4.15 (b) and 4.16 (b) displays the number of iterations to minimize the gradient of individual G and D networks on x axis and the corresponding loss curve on y axis.

4.8.2 Training Discriminator on DeepFake challenge dataset

Once GAN is trained on celebrity faces dataset for 280 epochs, the discriminator model is saved and further loaded to train on deepfake challenge dataset. The performance of proposed discriminator network has been presented in figure 4.17 in the form of of loss and accuracy chart. From figure 4.19, we can observe that our proposed model has reached a minimum loss of 0.34 on epoch 17 and the corresponding accuracy for this epoch is 87% on validation dataset. It can also be noted that the loss value did not improve after 17th epoch on validation set even though accuracy did improve by about 3 %, therefore this model ceased training at the end of 22nd epoch due to early stopping implemented with patience of 5 epochs. This can also be observed that training accuracy of this model has been exponentially going high and after 17 epoch it seems to be over-fitting and reached close to 99%. It can be verified whether this model is accurate enough or being over-fitted once the model is tested on our test dataset later in this chapter.

4.9 Evaluation of Proposed Methodologies on Test dataset

Kaggle deepfake challenge dataset has a separate repository which contains 400 videos kept for test-set evaluation of deepfake detection models. Nevertheless, due to limited computing resources and time 100 videos were selected out of these 400 videos to extract key-frames and further faces of the subjects and saved into separate repositories for each video of the test-set. The extracted key-frames containing faces of the subjects from these videos were then tested on all our proposed models to evaluate performance of our models. Models that

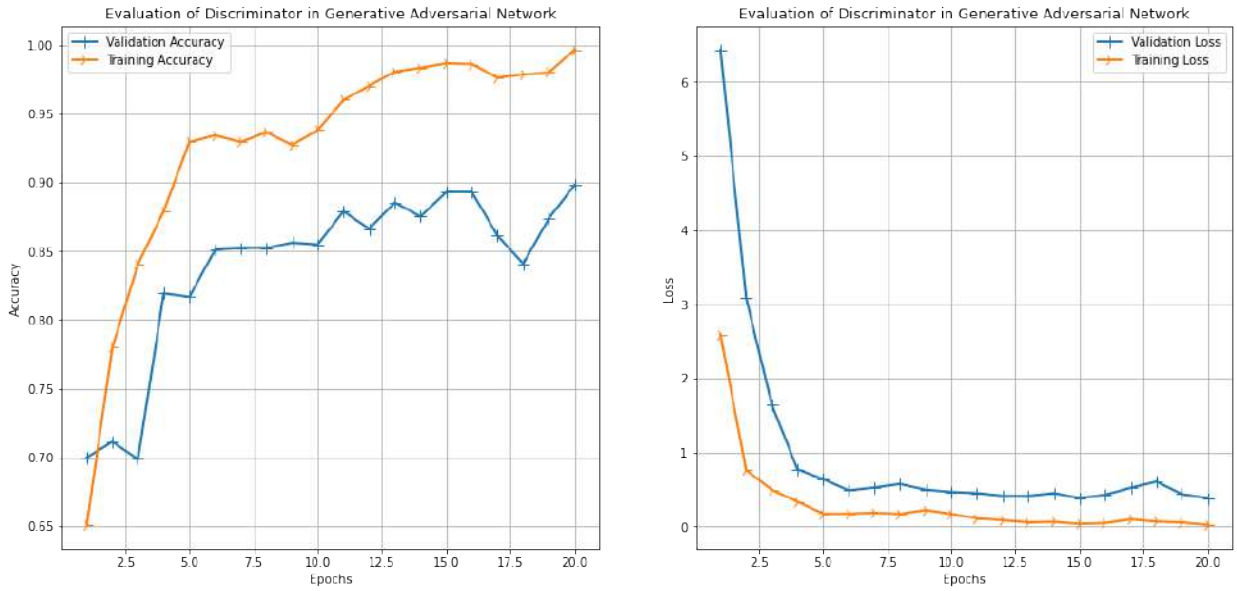


Figure 4.19: Evaluation of Proposed Discriminator on DeepFake Challenge Dataset

only predicts one key-frame of the video to be fake or real, have been evaluated by taking mean prediction value of 17 key-frames per video and if the mean exceeds 0.5 the predicted outcome reported as 1 otherwise 0. The performance of these models for predicting each video to be fake or not have been finally measured in terms of accuracy, weighted precision, weighted recall, weighted f1 score and log loss. The performance of these models have been presented in the table below.

Table 4.1: Evaluation of Proposed DeepFake Detection Models

Proposed Model Name	Accuracy	Precision	Recall	F1	Log Loss
CNN from scratch (Pritam's Baseline var 1)	65	62	65	58	0.67
CNN from scratch (Pritam's Baseline var 2)	59	54	59	54	0.79
CNN from scratch (Pritam's Baseline var 3)	52	54	52	53	0.84
MESONET	49	58	49	49	0.93
Integrated CNN(Xception)-LSTM	64	41	64	50	0.65
Integrated CNN(Xception)-LSTM(Fine Tuned)	58	57	58	58	14.5
Integrated CNN(ResNet50)-LSTM(Fine Tuned)	63	55	63	53	12.77
Sequential CNN(Xception)-LSTM	56	52	56	53	1.46
Transfer Learning with VGG16	53	53	53	53	0.92
Transfer Learning with ResNet50	54	59	54	55	0.87
Transfer Learning with ResNet50(Fine Tuned)	64	59	64	53	12.43
Transfer Learning with Xception	47	50	47	48	18.3
Transfer Learning with InceptionResNetV2	53	57	53	54	0.8
Discriminator (GAN)	52	48	52	49	16.57
Three Dimensional CNN	48	57	48	48	17.96

From Table 4.1, we can observe accuracy, average weighted precision, weighted recall, weighted f1 score and log loss scores of proposed 15 models on kaggle deepfake test set. The top accuracy (65%) has been achieved by Pritam's Baseline variation 1 (16-32-16-32) followed by Integrated CNN(Xception)-LSTM and Transfer

Learning on ResNet50(Fine Tuned) with 64% accuracy, a close third has been Integrated CNN(ResNet50)-LSTM(Fine Tuned) with 63% weighted accuracy. The (Fine Tuned) version of the model implies all of the layers of the pre-trained model has been trained on our kaggle training dataset unlike simple transfer learning approaches for which frozen weights of the pre-trained models have been used (no training required). As a result, the fine tuned models require a lot of training time and it is computationally expensive. From that perspective, baseline model variation 1 not only produced highest accuracy on the testset but also has very few layers and lowest number of output channels among all other baseline variations. simple transfer learning approaches also performs much faster and therefore proposed Integrated CNN(Xception)-LSTM model could be regarded as one of best models due to lowest log loss (0.65). To our surprise, the log loss of baseline model variation 1 comes very close to integrated CNN-LSTM methodology while the model complexity and training time is very low in comparison.

The performance of all these models were affected by the class imbalance of the 100 videos considered for this test set. Out of 100 videos 64 were real where 34 of them were fake. Since weighted average of these performance metrics were considered in table 4.1, even though models that have performed really well on the major class (real videos in this case), the performance was penalised while calculating average weighted precision ,recall and f1. It can also be noted that, data distribution in test set on first 100 videos could be different from the training data distribution of 154 videos considered in this project (the training dataset had 400 videos in total). This could be one of the other reasons for proposed models to not produce very high performance on the test set. In the next chapter, we would discuss how these proposed methodologies compare against the top 5 deepfake detection models from kaggle revealed recently[116].

Chapter 5

Discussion

The aim of this project was to implement an optimal deepfake detection system following the data pipeline(keyframe extraction, face extraction, data pre-processing and finally deepfake detection model) on kaggle deepfake challenge dataset. The implementation of several proposed methodologies have been demonstrated to achieve general aim of this project. In this chapter, we will briefly summarise how the objectives of this project have been achieved to address the key research questions discussed in chapter 1. Following that, this chapter will also cover a comparison between proposed methodologies implemented in this project in view of kaggle deepfake competition. Finally, the challenges throughout this project along with resolution will be discussed to delineate the big picture.

5.1 Reflection on Project Objectives

5.1.1 KeyFrame Extraction

KeyFrame Extraction was a consequential part of the deepfake detection data pipeline. This has been achieved by obtaining the image frame with maximum entropy within each sequence of image frames in a video. The methodology has been compared against random keyframe selection from Gaussian Distribution and proved to be more efficient compared to random keyframes(see chapter 4 for evaluation).

5.1.2 Face Extraction

Several face detection methodologies have been compared in this project and finally RetinaFace was determined as the best detection model for proposed deepfake detection pipeline.

5.1.3 DCNN to detect pixel level artifacts

Pritam's baseline model along with Mesonet and transfer learning on ResNet50, VGG16, Xception, Inception-ResNetV2 models were implemented to detect pixel level artifacts on faces of the subjects in videos.

5.1.4 Effect of Pre-processing on model performance

The proposed Normalization-Standardisation pre-processing methodology has been compared against other pre-processing techniques and the evaluation shows our proposed approach not only achieves minimum loss but also faster convergence compared to other tested methodologies(see chapter 4 for evaluation).

5.1.5 Avoiding Data Imbalance

To avoid data imbalance, the number of training samples were selected equally (77 real and 77 fake) for both fake and real videos in training. The performance metrics considered to evaluate testset also includes weighted average of all performance metrics(accuracy, precision, recall, f1 score) to inaccurate results due to data imbalance.

5.1.6 Avoiding Model Overfitting

This has been achieved in this project with the help of BatchNormalization layer, Dropout, EarlyStopping into all our deepfake detection models.

5.1.7 Capturing Spatio-Temporal inconsistency

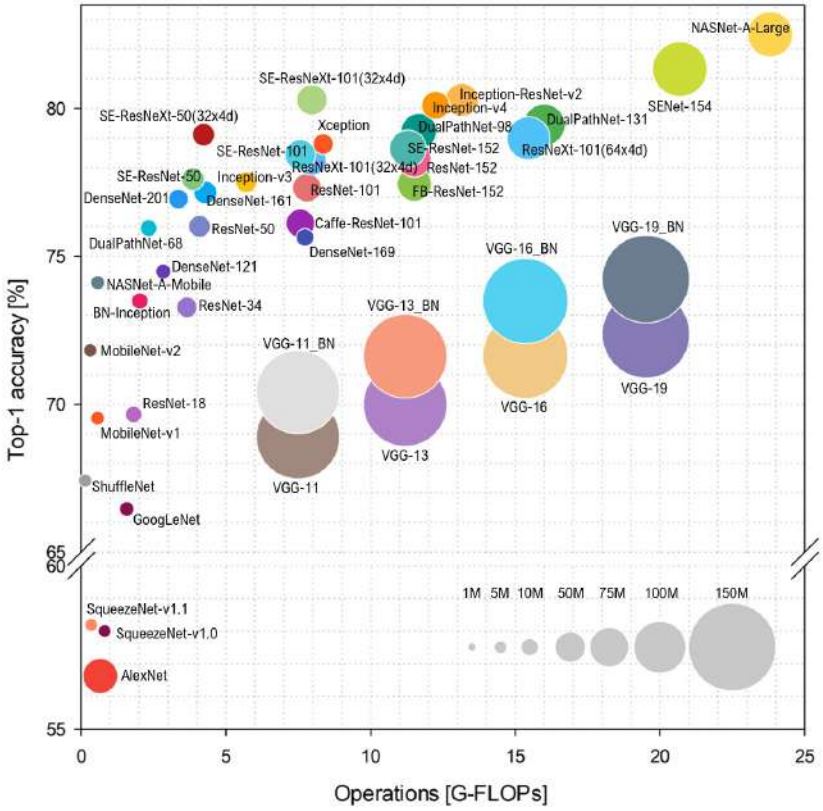
Models that capture spatio-temporal dependency between image frames in a video has been implemented in this project such as 3DCNN, Integrated CNN-LSTM, Sequntial CNN-LSTM etc. to discern deepfakes from real videos.

5.1.8 Choice of models(less computationally expensive yet more efficient)

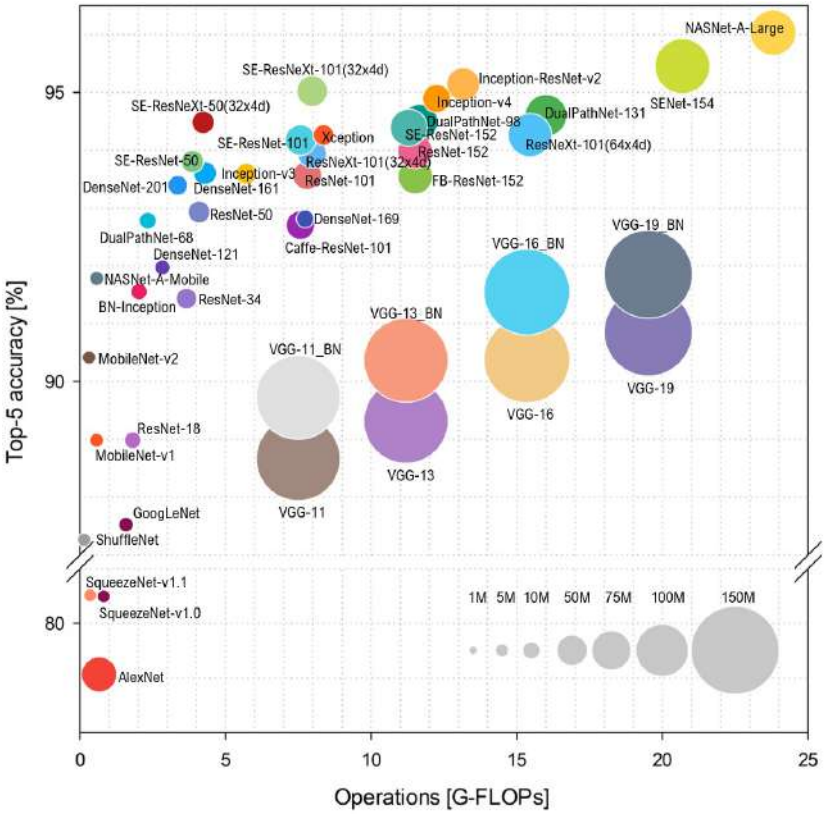
Pritam's baseline model has very few convolution layers with BatchNormalization (this helps in faster convergence of loss gradient) while simple transfer learning approaches did not require any training on pre-trained models except the FC layers or MLP followed by it (this requires training but significantly less training on DCNN). Our choice of models for transfer learning can be justified by the performance benchmark of these models on ILSVRC as presented in the figure below. The memory complexity in terms of GPU memory required against Top-1 and Top-5 accuracy acquired on the ImageNet-1k validation set as presented in Figure 5.1 justifies our choice of all of the pre-trained models for transfer learning. The choice of CNN-LSTM for both variations have used pre-trained models with frozen weights(no training at CNN level, only the RNN is trained) except the fine tuned models with LSTM(this requires more training). Finally, proposed 3DCNN architecture has very few layers as discussed in chapter 3, yet this model performs really well on our dataset.

5.2 Limitations

Exploring all of the proposed methodologies (including model variations) explained in chapter 3 posed a plethora of challenges on this project within limited time and with very limited computing resources. The primary computing resource for this project was a laptop with I5 processor, 8 GB of RAM and GTX 1660 Ti as graphics processor Unit. The secondary computing resource has been Google Colab (free version) which has been utilized intermittently in this project.
The use of secondary computing resource had been especially helpful in times when deep learning environment posed surmountable challenges due to version incompatibility between several deep learning packages (for example openCV, keras, face_detection, DLIB, MTCNN, Pytorch, Tensorflow) on primary computing resource. Nevertheless, the secondary computing resource was about 3-4 times less efficient for evaluating models compared to primary computing resource. Since all of the deep learning methodologies developed in this project was on keras with tensorflow as the default tensor backend engine, one of the major challenge during this project was tensorflow version incompatibility. The data pipeline, model building, execution - all of it has been migrated from tensorflowV1 to tensorflowV2 lately, while some packages were not compatible with



(a) Ball chart of the Top-1 accuracy vs. computational complexity for ImageNet Models



(b) Ball chart of the Top-5 accuracy vs. computational complexity for ImageNet Models

Figure 5.1: Ball chart reporting the Top-1 and Top-5 accuracy vs. computational complexity. The size of each ball corresponds to the model complexity. (a) Top-1; (b) Top-5 [120]

latest version of tensorflowV2. To obtain a stable build of tensorflow-GPU version with CUDNN and Cuda tookit (for accelerated processing of data with GPU) compatible with other packages for example OPENCV, Face_Detection, DLIB, MTCNN was a huge challenge for this project within a limited time. The Anaconda environment had been reinstalled several times with these deep learning packages to avoid version incompatibility and conflicts on deep learning environment (Example- multiple Numpy, tensorflow packages in the environment). The challenge is even more complicated when it was investigated that a specific version of Tensorflow (Tensorflow v2.1) is only compatible with CUDA Toolkit version 10.0 above and keras 2.3 for example, therefore as more packages were installed more challenges needed to be resolved on top of developing complex deep learning architectures, long training time and very limited available computing resources.

5.2.1 Limitation Of Available Resources And Resolution

The second biggest challenge during this project was very limited computing resources available and the amount of RAM available on primary resources was not enough to train such complicated proposed deep learning models. This challenge was resolved with these following approaches-

- The virtual memory of the primary resource was increased to 100 Gigabytes in addition with 8 Gigabytes of RAM. There was a possibility that processing huge amount of data through secondary hard disk (much slower compared to RAM) as virtual memory could corrupt the disk. Nevertheless, there was no other alternatives available than to take such risks.
- The development of the project was divided into three phases to avoid exhausting the memory of the primary resource, the phased can be discussed as follows,
  - **Phase 1:** Phase 1 includes accessing the Metadata JSON file, that contains descriptions of our deepfake video files. From this dataset 77 rows with label real and 77 with fake videos were selected as our training dataset. The corresponding video filenames present in this dataset (154 rows) were then analyzed with OPENCV to extract key-frames using our proposed key-frame extraction methodology. Once these 17 key-frames were extracted from a video file, the key-frame images were analyzed with our proposed face-detection methodology. Furthermore, these face images were extracted with a padding of 25 pixels and saved to a repository which has the name of this corresponding video file. The process has been repeated for all 154 video files selected for our training dataset.
  - **Phase 2:** Once the phase 1 is finished, python programs holding large share of CPU memory during phase 1 were terminated and as a result memory were released after this phase. In phase 2, another python program reads from the image key-frames within the repositories that were saved during phase 1, consolidates all of the image frames into one big Numpy array of size 2618x224x224x3. This numpy array is further normalized and standardized for each image and the corresponding labels of these images were fetched from Metadata JSON file. Finally image data and labels were split into training(70%) and validation data(30%) and finally the entire train validation tuple was saved into a Numpy dump file.
  - **Phase 3:** Since train-test split requires huge amount of RAM after phase 2 memory was released by terminating all python processes. In this phase, training and validation data were restored from this Numpy dump file and finally applied on our proposed deepfake detection models for evaluation.

The implementation of deepfake detection system in three phases not only reduced the enormous overhead of system RAM (since python processes have the tendency to hold RAM even after terminating python programs and processes) but also the modular approach saved enormous amount of time. After phase 2, this Numpy dump file of about 2 GB could be uploaded to google drive to utilize our secondary computing resource if the primary computing resource is exhausted. This has been the case several times. The primary computing

resource (Laptop) was also restarted in many cases and resumed execution after phase 2 Numpy dump for faster evaluation compared to Google Colab.

5.2.2 Limitation of High Training Time And Resolution

The limited amount of available computing resources along with 12 weeks of time to finish this project have been the major reason behind restricting training and validation dataset on a total of 154 videos out of 400 videos from kaggle deepfake challenge training dataset. Another big reason for choosing 154 videos as our training dataset is to avoid any data imbalance between major and minor class labels. For example if the model is trained on 323 fake videos and 77 real videos, the model might obtain very high accuracy based on it's performance on major class (fake videos), but in reality it would be hard to justify the performance of the models other than evaluating F1 score or testing it on test-set. It took more than 5 hours for phase 1 to extract image key-frames with faces in respective repositories for 154 videos as proposed in our methodology. Therefore, key-frame and face extraction from 400 videos on our primary resource would have been a daunting task. For the same reason, the test-set for this project contains 100 videos out of 400 videos from kaggle deepfake test-set.

5.2.3 Limitation Of Training Data Distribution

Neural Networks require huge amount of data to optimize the weights of it's network and learn from the data distribution. However, if enough data were not trained or one specific data distribution is trained on a neural network, this might not generalize the data distribution well and the network might perform poorly on data outside it's training distribution. This has been the case with our proposed models when tested on a test-set which is intuitively a different data distribution compared to our training dataset, performed poorly. This hypothesis can be justified as all of the proposed models have achieved very high accuracy on validation dataset which was selected randomly as 30% of our training set before any training. The number of advanced neural network architectures implemented in this project has been truly versatile in nature. Hence, if the proposed methodologies could have been trained on 400 videos and tested on all 400 videos there could have been possibility for a common data distribution in both cases for a far better performance on a test-set.

5.3 Comparison of Proposed Models

Having several setbacks (discussed in the following section), the proposed deepfake detection models performed exceptionally well considering the training data distribution was not dispersed enough to perform well on the test-set. From table 4.1 we can observe that Integrated CNN(Xception)-LSTM model along with Fine-Tuned ResNet model along with baseline model variation 1 performed the best on kaggle deepfake challenge test-set. As mentioned earlier, most models performed poorly due to training data distribution did not generalize well to perform well on this test-set. Since, the test-set was developed by selecting first 100 videos out of 400, there's a big possibility that if these proposed models were tested on all 400 videos, they could have performed better considering the training data distribution is close to the test-set distribution.
This can also be noted that all Pritam's baseline models have performed surprisingly well on the testset considering low model complexity compared to some of the complicated model architectures implemented in this project. Due to very few number of layers in these models obtaining very high accuracy could be an ideal choice for projects that has limitation on computational resources and time. This is more prominent since the accuracy of baseline model variation 1 has been better than state of the art CNN-LSTM approach found in research literature [35] for deepfake detection.

5.3.1 Discussion on Proposed Models in view of Kaggle Competition

After reviewing limited number of open literatures on DeepFake Detection and video classification research, some of the publicly available Kaggle competition notebooks were reviewed to validate relevance of some of the methodologies implemented in this project. The fellow Kaggle Expert "Human Analog" [78] demonstrated in one of his public notebooks that with pre-trained model ResNext50(a variation of ResNet50), he was able to achieve a LogLoss score of 0.46 which was at that time top 50 scores in the public leaderboard. He had trained only 3-4 layers of this network while in our proposed methodologies we followed two variations of ResNet50 - 1) frozen weights with no training and 2) training all the layers in ResNet50. The Log Loss for these two models in our testset turned out to be 0.87 and 12.43(fine tuned) respectively. Hence we could argue, that if we could have trained ResNet50 across all 400 videos and tested on all 400 videos we could have achieved a LogLoss similar to .46 compared to 12.43 on training all layers of ResNet50. One of the top notebooks publicly available was from kaggle Grandmaster "Ian Pan" who ranked 17th by the end of the competition with a LogLoss score of 0.25 on public leaderboard and ranked 5th on Kaggle Private Leaderboard with a LogLoss score of 0.46. The methodology he followed was custom 3D CNN with Inception module, ResNet Block implemented and finally used an ensemble methodology to vote between all different variations of his 3D CNN model. In this project, we managed to implement a variation of C3D, a simple 3D CNN architecture for video classification. Nevertheless, our proposed C3D model was exhausting GPU resources with out of memory error and could not be executed without reducing the batch size to 1. Hence, it can be empirically inferred that implementing variation of pre-trained 3D CNN (example ResNet-3D) was way beyond the scope of this project let alone ensembling those approaches on a very limited computing resources. The performance of top 5 winning solutions from kaggle competition (discovered late in this project) in comparison to our top performing proposed methodology can be presented as follows,

Table 5.1: Performance of top 5 wining solution vs top proposed methodology

Team/Model Name	Overall Log Loss
Selim Seferbekov	0.4279
WM	0.4284
NTechLab	0.4345
Eighteen Years Old	0.4347
The Medics	0.4371
Proposed Integrated CNN(Xception)-LSTM	0.65
CNN from Scratch (Pritam's Baseline Model var 1)	0.67

The top submission, Selim Seferbekov, has applied MTCNN [38] for face detection and an EfficientNetB-7 for feature encoding. Nevertheless, it has been evaluated in this project that MTCNN [38] can not efficiently detect or extract faces very accurately in complex imaging scenarios. The second ranked solution, WM, applied the Xception [42] architecture for frame-by-frame feature extraction, and a WSDAN model for augmentation, our proposed top model used same Xception network as image encoding. The third solution, NTechLab, has applied an ensemble of EfficientNets in addition to using the mixup augmentation during training. The fourth solution, Eighteen Years Old , has applied an ensemble of frame and video models, including EfficientNet, Xception, ResNet, and a SlowFast video-based network; tailoring a score fusion strategy. This methodology motivates us to ensemble all our proposed models and apply fusion strategy to combine the predictions in a voting scheme. This has been explained in our future work as well. Finally, the Medics, also applied MTCNN [38] for face detection, as well as an ensemble of 7 models, 3 of which were 3D CNNs; this methodology could be very very computationally expensive with limited computational resources [116].

After reviewing the kaggle leaderboard, it can be asserted that all of our proposed methodologies were on right directions and if enough computing resources were available during this project, the Log Loss scores on entire kaggle test-set would have been very comparable with Log Loss scores of other Kagglers. Nevertheless, even with all these challenges our top models Integrated CNN-LSTM Baseline model var 1 achieved a Log Loss score of 0.65 and 0.67 respectively even on just first 100 test videos trained on only 154 videos. The model architecture of baseline model variation 1 as explained in section 3.7 suggests the model complexity, number of parameters and training time of this model compared to any of the top 5 kaggle solutions is significantly low. If we ensemble all 15 of our proposed deepfake models then it might reach close to the complexity of any top 5 solutions from kaggle.

Chapter 6

Conclusion

Taking everything into account it can be inferred that the general aim of the project has been achieved. This project investigated and developed several deepfake detection architectures that could be applied to identify deepfake videos on social media and other online platforms. Furthermore through prediction metrics it has been proved that the proposed methodologies are effective at selecting key-frames and extracting subject's faces from the video while being able capture a pattern representations of the underlying features of these face images to learn and identify fake images from real. Based on the performance metrics obtained on our test-set in table 4.1, we could eagerly test these methodologies on other deepfake datasets and with more computing resources to test the real strength of these proposed methodologies.
This project has implemented several state of the art deepfake detection methodologies some of which were way beyond the scope of this project with limited resources and time. The project also investigated mainstream and state of the art face detection models along with entropy based key-frame extraction from videos (a mainstream approach for key-frame extraction). The project also explored out of the box methodologies like training GAN on celebrity faces dataset which is a different data distribution compared to training data distribution however Discriminator learns to discern between distorted images from real one. Finally, this was trained on training data distribution to fine-tune the weights of the neural network (Discriminator). This methodology has achieved 90% accuracy on our validation set with just few convolution layers compared to our proposed baseline model and MESONET both of which have achieved around 85% accuracy.

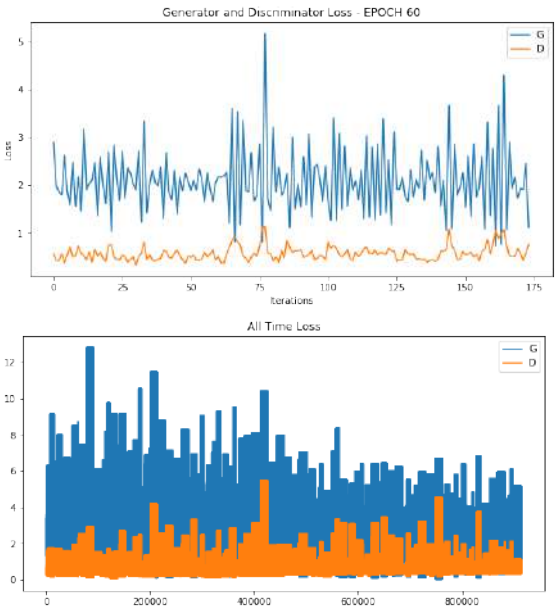
6.1 Future Work

There is always a lot of room for improvement in any deep learning methodology and likewise this project is no exception. Although, the number of proposed methodologies evaluated in a short period of time is astounding, there is still some room for improvement in these approaches. One future approach that could significantly improve the performance of these models are parameter tuning, especially working with manual learning rates. This takes a longer time to evaluate manually the learning rate with momentum and decay which might optimize the gradient of the loss function during back propagation more efficiently. Another future work that could significantly improve the performance of deepfake detection system is to ensemble all of our proposed deepfake detection models together. It means the prediction of all of our proposed models will be consolidated together in a voting scheme, where the final prediction will be decided based on majority of votes by the members or proposed models in this case. It would be also very interesting if we could train and evaluate a pre-trained 3D CNN model(like ResNet-3D) in future on kaggle dataset with adequate computing resources.

Appendix A

Appendix

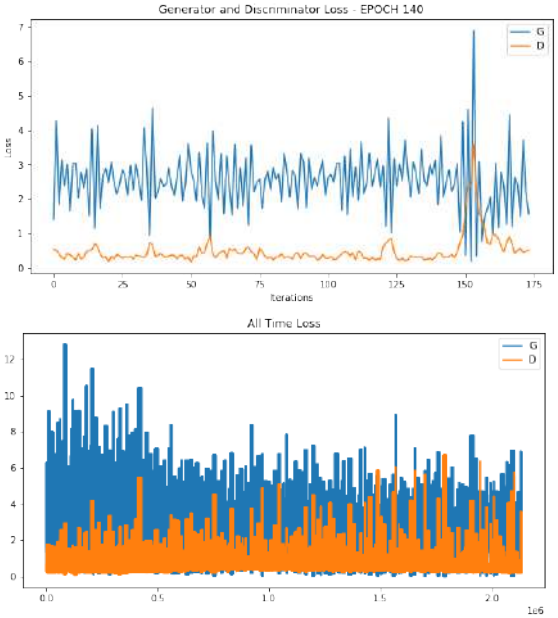
(a) Face Images Generated at Epoch 60



(b) Discriminator and Generator loss at Epoch 60

Figure A.1: Training Proposed GAN on celebrity faces (Epoch 60)

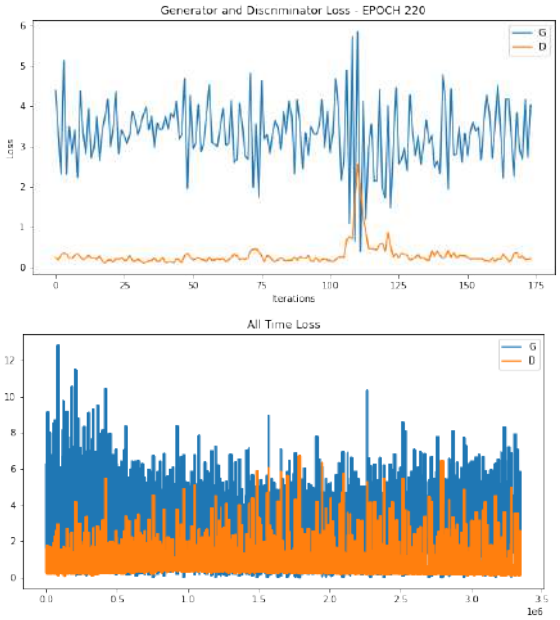
(a) Face Images Generated at Epoch 140



(b) Discriminator and Generator loss at Epoch 140

Figure A.2: Training Proposed GAN on celebrity faces (Epoch 140)

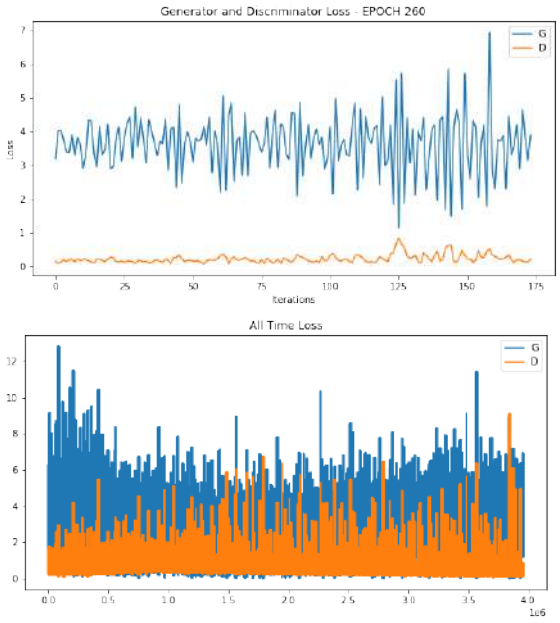
(a) Face Images Generated at Epoch 220



(b) Discriminator and Generator loss at Epoch 220

Figure A.3: Training Proposed GAN on celebrity faces (Epoch 220)

(a) Face Images Generated at Epoch 260



(b) Discriminator and Generator loss at Epoch 260

Figure A.4: Training Proposed GAN on celebrity faces (Epoch 260)

Bibliography

[1] Du Tran et al. “C3D: Generic Features for Video Analysis”. In: *CoRR* abs/1412.0767 (2014). arXiv: 1412.0767. URL: http://arxiv.org/abs/1412.0767.

[2] Yuezun Li and Siwei Lyu. “Exposing DeepFake Videos By Detecting Face Warping Artifacts”. In: *arXiv preprint arXiv:1811.00656* (2018).

[3] H. R. Hasan and K. Salah. “Combating Deepfake Videos Using Blockchain and Smart Contracts”. In: *IEEE Access, vol. 7* (2019), pp. 41596–41606.

[4] *Deepfake*. Wikipedia. Aug. 2019. URL: https://https://en.wikipedia.org/wiki/Deepfake.

[5] Chesney Robert and Citron Danielle Keats. “Deep Fakes: A Looming Challenge for Privacy, Democracy, and National Security”. In: *California Law Review 1753 (2019), U of Texas Law, Public Law Research Paper No. 692, U of Maryland Legal Studies Research Paper No. 2018-21* (2018).

[6] Stan Horaczek. “Spot Faked Photos Using Digital Forensic Techniques”. In: *POPULAR SCIENCE* (July 2017).

[7] Tiffanie Wen. “The Hidden Signs That Can Reveal a Fake Photo”. In: *BBC FUTURE* (June 2017).

[8] Tiffanie Wen. “How DARPA’s Fighting Deepfakes”. In: *FAST COMPANY* (Apr. 2018).

[9] Larry Hardesty. “Explained: Neural Networks, MIT NEWS”. In: *MIT NEWS* (Apr. 2017).

[10] *Deepfake: Willem Dafoe - coming out video*. Ten Deepfake. July 2020. URL: https://www.youtube.com/watch?v=P9gDoo8xZdc&feature=youtu.be&t=99.

[11] H. Kim et al. “Deep video portraits”. In: *ACM Transactions on Graphics* (2018).

[12] Kemelmacher-Shlizerman Ira Suwajanakorn Supasorn Seitz Steven M. “Synthesizing Obama: Learning Lip Sync from Audio”. In: *ACM Trans. Graph. 36 (4):* (2017), 95:1–95:13.

[13] Koki Nagano et al. “paGAN: real-time avatars using dynamic textures”. In: *SIGGRAPH Asia Technical Papers, page 258. ACM* (2018).

[14] Albert Pumarola et al. “GANimation: Anatomically-aware facial animation from a single image”. In: *CoRR, abs/1807.09251* (2018).

[15] J. Thies et al. “Headon: Real-time reenactment of human portrait videos”. In: *ACM Transactions on Graphics* (2018).

[16] Shruti Agarwal and Hany Farid. “Protecting World Leaders Against Deep Fakes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2019).

[17] Kevin Roose and Paul Mozur. “Zuckerberg was called out over Myanmar violence. here’s his apology”. In: *The NewYork Times* (2018).

[18] Scott Shane and Mark Mazzetti. “Inside a 3-year Russian campaign to influence U.S. voters”. In: *The NewYork Times* (2018).

[19] Amanda Taub and Max Fisher. “Where countries are tinderboxes and Facebook is a match”. In: *The NewYork Times* (2018).

[20] Slaney Malcolm Bregler Christoph Covell Michele. “Video Rewrite: Driving Visual Speech with Audio”. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques 24:* (1997), pp. 353–360.

[21] Swenson Kyle. "A Seattle TV station aired doctored footage of Trump's Oval Office speech. The employee has been fired". In: *The Washington Post* (Jan. 2019).

[22] "#TellTheTruthBelgium". In: *Extinction Rebellion Belgium* (Apr. 2020).

[23] Koebler Jason Cole Samantha Maiberg Emanuel. "This Horrifying App Undresses a Photo of Any Woman with a Single Click". In: *Vice* (June 2019).

[24] O'Sullivan Donie. "Congress to investigate deepfakes as doctored Pelosi video causes stir". In: *CNN* (Nov. 2019).

[25] O. de Lima and S. et al. Franklin. "Deepfake detection using spatiotemporal convolutional networks". In: *arXiv preprint arXiv:2006.14749* (2020).

[26] I. Amerini and R. Caldelli. "Exploiting prediction error inconsistencies through LSTM-based classifiers to detect deepfake videos". In: *In Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security (pp. 97-102)* (2020).

[27] V. Badrinarayanan, A. Kendall, and R Cipolla. "SegNet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), 2481-2495* (2017).

[28] *A voice deepfake was used to scam a CEO out of $243,000*. Damiani, J. Sept. 2019. URL: https://www.forbes.com/sites/jessedamiani/2019/09/03/a-voice-deepfake-was-used-to-scam-a-ceo-out-of-243000/.

[29] *Chinese deepfake app Zao sparks privacy row after going viral*. The Guardian. Sept. 2019. URL: https://www.theguardian.com/technology/2019/sep/02/chinese-face-swap-app-zao-triggers-privacy-fears-viral.

[30] L et al. Guarnera. "Preliminary forensics analysis of deepfake images". In: *arXiv preprint arXiv:2004.12626* (2020).

[31] M. A. Younus and T. M. Hasan. "Effective and fast deepfake detection method based on Haar wavelet transform". In: *International Conference on Computer Science and Software Engineering (CSASE) (pp. 186-190). IEEE* (2020).

[32] *Media Forensics (MediFor)*. Turek, M. 2019. URL: https://www.darpa.mil/program/media-forensics.

[33] Chang Li Y., M. C., and S Lyu. "In ictu oculi: Exposing AI created fake videos by detecting eye blinking". In: *IEEE International Workshop on Information Forensics and Security (WIFS) (pp. 1-7). IEEE* (2018).

[34] Cheng et al. Sabir E. "Recurrent convolutional strategies for face manipulation detection in videos." In: *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 80-87)* (2019).

[35] D. Guera and E. J Delp. "Deepfake video detection using recurrent neural networks". In: *15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) (pp. 1-6). IEEE* (2018).

[36] X. Yang, Y. Li, and S Lyu. "Exposing deep fakes using inconsistent head poses." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8261-8265). IEEE* (2019).

[37] H. H. Nguyen and J. et al. Yamagishi. "Capsule forensics: Using capsule networks to detect forged images and videos". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2307-2311). IEEE.* (2019).

[38] K. Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE Signal Processing Letters, 23(10):1499–1503* (2016).

[39] J. Deng et al. "Retinaface: Single-stage dense face localisation in the wild". In: *arXiv:1905.00641* (2019).

[40] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[41] K. Simonyan and A Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[42] F. Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *arXiv preprint arXiv:1610.02357v2*, (2016).

[43] S. Lorant. "Lincoln; a picture story of his life". In: *Norton* (1969).

[44] P. Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5967–5976, Honolulu, HI* (July 2017).

[45] J. Y. Zhu et al. "Unpaired imageto-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE International Conference on Computer Vision, pages 2242–2251,Venice, Italy* (Oct. 2017).

[46] M. Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *Proceedings of the USENIX Conference on Operating Systems Design and Implementation, 16:265–283,Savannah, GA* (Nov. 2016).

[47] *Keras*. F. Chollet et al. 2015. URL: https://keras.io/.

[48] A. Tewari et a. "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 1274–1283,Venice, Italy* (Oct. 2017).

[49] J. Thies et al. "Face2Face: Real-time face capture and reenactment of rgb videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2387– 2395, Las Vegas, NV* (June 2016).

[50] G. Antipov, M. Baccouche, and J. L. Dugelay. "Face aging with conditional generative adversarial networks". In: *arXiv:1702.01983* (Feb. 2017).

[51] D. Guera et al. "A counter-forensic method for CNN-based camera model identification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1840–1847,Honolulu, HI* (July 2017).

[52] *FaceApp*. Wireless Lab. Dec. 2016. URL: https://www.faceapp.com/.

[53] *FakeApp*. FakeApp. Jan. 2018. URL: https://www.fakeapp.org/.

[54] S. Fan and R. Wang et al. "Human perception of visual realism for photo and computer-generated face images". In: *ACM Transactions on Applied Perception (TAP), 11(2):7* (2014).

[55] Nozick et al. Afchar D. "MesoNet: a compact facial video forgery detection network". In: *IEEE International Workshop on Information Forensics and Security (WIFS) (pp. 1-7). IEEE* (2018).

[56] H. T. Sencar and editors N. Memon. "Digital Image Forensics". In: *Springer New York* (2013).

[57] H. T. Sencar and editors N. Memon. "Photo Forensics". In: *MIT Press Ltd* (2016).

[58] D. Guera et al. "Reliability map estimation for cnn-based camera model attribution". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Lake Tahoe, NV* (Mar. 2018).

[59] W. Wang and H. Farid. "Exposing digital forgeries in interlaced and deinterlaced video". In: *IEEE Transactions on Information Forensics and Security, 2(3)* (2007).

[60] P. Bestagini et al. "Local tampering detection in video sequences". In: *Proceedings of the IEEE International Workshop on Multimedia Signal Processing, pages 488–493, Pula, Italy* (2013).

[61] V. Conotter et al. "Physiologically-based detection of computer generated faces in video". In: *Proceedings of the IEEE International Conference on Image Processing, pages 248–252, Paris, France* (2014).

[62] N. Rahmouni et al. "Distinguishing computer graphics from natural images using convolution neural networks". In: *Proceedings of the IEEE Workshop on Information Forensics and Security, pages 1–6, Rennes, France* (2017).

[63] R. Raghavendra et al. "Transferable deep-cnn features for detecting digital and print-scanned morphed face images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1822–1830, Honolulu, HI.* (2017).

[64] P. Zhou et al. "Two-stream neural networks for tampered face detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1831–1839, Honolulu, HI.* (2017).

[65] K. Dale and K. Sunkavalli et al. "Video face replacement". In: *ACM Transactions on Graphics, 30(6):1–130* (2011).

[66] Z. Lu and Z. Li et al. "Recent progress of face image synthesis". In: *arXiv:1706.04717* (2017).

[67] K. Dale and K. Sunkavalli et al. "Conditional cyclegan for attribute guided face image generation". In: *arXiv:1706.04717* (2017).

[68] K. Dale and K. Sunkavalli et al. "Deep feature interpolation for image content changes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6090–6099, Honolulu, HI* (2017).

[69] G. Lample et al. "Fader networks: Manipulating images by sliding attributes". In: *Advances in Neural Information Processing Systems, pages 5967–5976, Long Beach, CA* (2017).

[70] Y. Liao, Y. Wang, and Y. Liu. "Graph regularized autoencoders for image representation". In: *IEEE Transactions on Image Processing, 26(6):2839–2852* (2017).

[71] Y. Qian et al. "Recurrent color constancy". In: *Proceedings of the IEEE International Conference on Computer Vision, pages 5459–5467, Venice, Italy* (2017).

[72] Thanh Thi Nguyen et al. "Deep Learning for Deepfakes Creation and Detection: A Survey". In: *arXiv e-prints, arXiv:1909.11573* (2019).

[73] K. He et al. "Deep residual learning for image recognition". In: *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778)* (2016).

[74] X. Yang, Y. Li, and S Lyu. "Exposing deep fakes using inconsistent head poses". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8261-8265). IEEE* (2019).

[75] G et al. Huang. "Densely connected convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4700- 4708)* (2017).

[76] K. Cho and Y et al. Bengio. "Learning phrase representations using RNN encoderdecoder for statistical machine translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1724-1734)* (2014).

[77] Z. Wang, A. C. Bovik, and H. R. Shei. "Image quality assessment: From error visibility". In: *structural similarity," IEEE Trans. Image Process., vol* (2004).

[78] *Inference Demo*. Human Analog- Kaggle. Feb. 2020. URL: https://www.kaggle.com/humananalog/inference-demo.

[79] Siddu P. Algur and Vivek R. "Video Key Frame Extraction using Entropy value as Global and Local Feature". In: *CoRR* abs/1605.08857 (2016). arXiv: 1605.08857. URL: http://arxiv.org/abs/1605.08857.

[80] Markos Mentzelopoulos and Alexandra Psarrou. "Key-frame extraction algorithm using entropy difference". In: Jan. 2004, pp. 39–45. DOI: 10.1145/1026711.1026719.

[81] *Face Identification using Haar cascade classifier*. Analytics Vidhya. Mar. 2019. URL: https://medium.com/analytics-vidhya/haar-cascade-face-identification-aa4b8bc79478.

[82] L. Cuimei et al. "Human face detection algorithm via Haar cascade classifier combined with three additional classifiers". In: *Proc. 13th IEEE Int. Conf. Electron. Meas. Instrum. (ICEMI), pp. 483–487* (2017).

[83] C. Shu, X. Ding, and C. Fang. "Histogram of the oriented gradient for face recognition". In: *Tsinghua Science and Technology* 16.2 (2011), pp. 216–224.

[84] Lourdes Ramirez Cerna. "Face Detection: Histogram of Oriented Gradients and Bag of Feature Method". In: *In Proceedings of the International Conference on Image Processing, Computer Vision and Pattern Recognition, Las Vegas, NV, USA* (July 2013).

[85] *Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning*. Adam Geitgey, Medium. July 2016. URL: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

[86] Y. Zhou et al. "Dense 3d face decoding over 2500fps: Joint texture and shape convolutional mesh decoders". In: *arXiv pp. (2,3,4,6,8)* (2019).

[87] A. Ranjan et al. "Generating 3d faces using convolutional mesh autoencoders". In: *ECCV* (2018).

[88] K. Genova et al. "Unsupervised training for 3d morphable model regression". In: *CVPR* (2018).

[89] Y. Lin et al. "Feature pyramid networks for object detection". In: *CVPR, pp(1,2,4)* (2017).

[90] Y. Lin et al. "Focal loss for dense object detection". In: *ICCV, pp(1,2,4)* (2017).

[91] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS* (2010).

[92] M. Najibi et al. "Ssh: Single stage headless face detector". In: *ICCV* (2017).

[93] X. Tang et al. "Pyramidbox: A context-assisted single shot face detector". In: *ECCV* (2018).

[94] C. C. Loy et al. "Wider face and pedestrian challenge 2018: Methods and results". In: *arXiv:1902.06854* (2018).

[95] J. Dai et al. "Deformable convolutional networks". In: *ICCV* (2017).

[96] X. Zhu et al. "Deformable convnets v2: More deformable, better results". In: *arXiv:1811.11168* (2018).

[97] M. Ahmad and F.M.A. Salam. "Supervised learning using the cauchy energy function". In: *International Conference on Fuzzy Logic and Neural Networks* (1992).

[98] Pravin Chandra and Yogesh Singh. "An activation function adapting training algorithm for sigmoidal feedforward networks". In: *Neurocomputing, 61: p. 429-437* (2004).

[99] et al S. C. Ng. "Fast convergence for back propagation network with magnified gradient function". In: *Proceedings of the International Joint Conference on Neural Networks 2003, 3: p. 1903-1908* (2003).

[100] R.A. Jacobs. "Increased rates of convergence through learning rate adaptation". In: *Neural Networks, 1988. 1: p. 295–307* (1988).

[101] M.K. Weir. "A method for self-determination of adaptive learning rates in back propagation". In: *Neural Networks, 1991. 4: p. 371-379* (1991).

[102] Nazri Mohd Nawi, Walid Hasen Atomi, and M.Z. Rehman. "The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks". In: *Procedia Technology, Volume 11, Pages 32-39, ISSN 2212-0173* (2013).

[103] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[104] *A guide to an efficient way to build neural network architectures- Part II: Hyper-parameter selection and tuning for Convolutional Neural Networks using Hyperas on Fashion-MNIST*. TowardsDataScience. May 2018. URL: https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7.

[105] N. Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research, 15(1):1929–1958* (2014).

[106] Krishnendu Kar. *Mastering Computer Vision with TensorFlow 2.x*. Packt, 2018. ISBN: 9781838827069.

[107] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[108] V. Schetinger et al. "Humans are easily fooled by digital images". In: *arXiv preprint arXiv:1509.05301* (2015).

[109] B. Balas and C. Tonsager. "Face animacy is not all in the eyes: Evidence from contrast chimeras". In: *Perception, 43(5):355– 367* (2014).

[110] S. Fan et al. "Human perception of visual realism for photo and computer-generated face images". In: *ACM Transactions on Applied Perception (TAP), 11(2):7* (2014).

[111] A. Ullah et al. "Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features". In: *IEEE Access* 6 (2018), pp. 1155–1166.

[112] Z. Xu, S. Li, and W. Deng. "Learning temporal features using LSTM-CNN architecture for face anti-spoofing". In: *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. 2015, pp. 141–145.

[113] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[114] *DCGANs — Generating Dog Images with Tensorflow and Keras*. TowardsDataScience. Aug. 2019. URL: https://towardsdatascience.com/dcgans-generating-dog-images-with-tensorflow-and-keras-fb51a1071432.

[115] P. Manisha and Sujit Gujar. "Generative Adversarial Networks (GANs): What it can generate and What it cannot?" In: *CoRR* abs/1804.00140 (2018). arXiv: 1804.00140. URL: http://arxiv.org/abs/1804.00140.

[116] Brian Dolhansky et al. "The DeepFake Detection Challenge Dataset". In: (2020). arXiv: 2006.07397 [cs.CV].

[117] MUHAMMAD ZUBAIR REHMAN and NAZRI MOHD. NAWI. "STUDYING THE EFFECT OF ADAPTIVE MOMENTUM IN IMPROVING THE ACCURACY OF GRADIENT DESCENT BACK PROPAGATION ALGORITHM ON CLASSIFICATION PROBLEMS". In: *International Journal of Modern Physics: Conference Series* 09 (2012). ISSN: 2010-1945. DOI: 10.1142/s201019451200551x.

[118] "Predicting Noise-Induced Hearing Loss (NIHL) and Hearing Deterioration Index (HDI) in Malaysian Industrial Workers using GDAM Algorithm". In: 3 (2012), pp. 179–197.

[119] N. M. Nawi, M. Z. Rehman, and M. I. Ghazali. "Noise-induced hearing loss prediction in malaysian industrial workers using gradient descent with adaptive momentum Algorithm". In: *International Review on Computers and Software* 6.5 (Sept. 2011), pp. 740–748. ISSN: 18286003.

[120] S. Bianco et al. "Benchmark Analysis of Representative Deep Neural Network Architectures". In: *IEEE Access* 6 (2018), pp. 64270–64277.