Deepfake Detection in Videos Detecting Face Wrapping Artifacts with Convolutional Neural Network

by

Fahim Faisal
16304061
Shifat Sarwar
16301084
Fowzia Mohona
19241024

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2021

© 2021. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Fahim Faisal
16304061

Shifat Sarwar

Shifat Sarwar
16301084

Fowzia Almazi

Fowzia Mohona
19241024

Approval

The thesis/project titled "Deepfake Detection in Videos Detecting Face Wrapping Artifacts with Convolutional Neural Network" submitted by

1. Fahim Faisal (16304061)
2. Shifat Sarwar (16301084)
3. Fowzia Mohona (19241024)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 9, 2020.

Examining Committee:

Supervisor:
(Member)

[illegible]

A. M. Esfar-E-Alam
Senior Lecturer
Department
Institution

Co-Supervisor:
(Member)

Zavid Parvez
Digitally signed by Zavid Parvez
DN: cn=Zavid Parvez, o=Brac University, ou=CSE, email=zavid.parvez@bracu.ac.bd, c=US
Date: 2021.01.15 14:55:27 +06'00'

Dr. Mohammad Zavid Parvez
Assistant Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Mahbub Alam Majumdar, PhD
Professor
Department of Computer Science and Engineering
Brac University

Abstract

Alteration of video files by changing the face of a person on frame is Deepfake. In such manipulated contents a person's face is used on a video performing or saying something that they never actually said or did. Deepfake allows using a person's face without their consent in a video they never actually shot. The manipulation is done with the help of an artificial intelligent method called deep learning. The AI learns the facial features of an individual from their pictures and applies them to another face in the video. As popular people have more of their images on the internet their Deepfakes can be easily created and shared without their knowledge or consent. Previously fake videos were made with simple copy pasting and photo editing, which were easily detectable by simply examining them with our eyes. Now, with artificial intelligence it is a whole new game; Deepfake videos have become very difficult to detect and judge and a software mechanism has become a necessity to determine the authenticity of possibly manipulated videos. Thus, our team tries to build an artificial intelligent network that is capable of Deepfake detection and determine the authenticity of a video file. For our solution, we will be attempting to detect face-wrapping artifacts from a subject's face in a particular video frame using Convolutional Neural Networks (CNN). To detect the faces of a subject in frame we will be using Haar-cascade classifiers. For training the network we will use a custom model made using Xception algorithm which trains only by using the faces extracted from video frames. Here, we will use the available dataset for Deepfake detection on Kaggle.com. The video files from the dataset will be compressed to a lower quality to train and validate our models as most Deepfakes being shared online have lower qualities. There are many ways to find Deepfakes but most of them are not efficient enough in their detection rate. We plan to increase the rate by successfully analyzing a video in poor and good condition to determine whether it has been manipulated or not.

Keywords: Deepfake; Face Wrapping Artifacts; Xception; Machine Learning; Convolutional Neural Networks(CNN)

Dedication

We dedicate this to our parents who supported us all throughout our journey.

Acknowledgement

First and foremost, praises and because of Allah, the Almighty, for His showers of blessings throughout our research work to finish the thesis successfully.

We would like to express our deep and sincere gratitude to our thesis supervisor, A. M. Esfer e Alam, lecturer, BRAC University, Bangladesh, for giving us the opportunity to do research and providing invaluable guidance throughout this thesis. His dynamism, sincerity and motivation have deeply inspired us. He has taught us the methodology to carry out the thesis and to present the thesis works as clearly as possible. It was an excellent privilege and honor to figure and study under his supervision. We are extremely grateful for what he has offered me. We would also like to thank him for his friendship, empathy, and great sense of humor.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
CNN	Convolutional Neural Network
GAN	Generative Adverserial Network
LRCN	Long-term Recurrent Convolutional Networks
ML	Machine Learning
ReLU	Rectified Linear Unit
ResNet	Residual Network
RMS	Root Mean Square

Chapter 1

Introduction

Deepfake videos are synthetic videos made by teaching the computer to create one individual face on top of another in a video file. A Deepfake video works by using hundreds or thousands of pictures to train the computer into learning facial structures to recreate them. Davis in his article “How Deepfake technology actually Works” describes the process involved in the creation of Deepfakes. The process works by allowing the machine to learn details in a person’s face, changes in the shapes during different expressions to recreate similar expressions in the falsified face which corresponds to the expressions of the real person in the video. It uses Generative Adversarial Networks(GANs) to learn and recreate the details of a person’s face from thousands or hundreds of high quality images of a particular person. Here, two machine learning models work simultaneously to produce the output video. While one model creates the fake replacing the original the other detects discrepancies to give feedback. This process is continued until the detector can no longer detect the changes or the detection is the minimum.[22] More simple forms of Deepfake contents can be created using a single picture from which the AI learns the facial features to swap them on a different face or short length videos and gifs. Reface is an available application on Apple and Android application stores. It uses a single picture allowing its users to have their faces pasted on celebrity figures in short video clips. Deepfake creation is being developed in large companies like Reface these days so their level of improvement has significantly increased. While some frames in a Deepfake video feel like fakes, there are those where the body and face does not feel manipulated. In this day and age of rising computing power and advancements in artificial intelligence, it has become easier to create Deepfake videos, which are more and more harder to identify. Upon very strict observation people could find some flaws in the decoded videos with their eyes, but it would get past as an original to a large number of viewers. If it were to become the case, we would have to clearly see all videos as a detective trying to find a flaw. These measly flaws can also disappear upon better training and alteration of the videos.

Deepfake can especially harm celebrities or political figures as everyone has access to an abundance of pictures and videos associated with them. These can be used to blackmail political figures and cause embarrassment for celebrity figures. They can also depict false news in order to change public opinion. As mentioned by Greengard(2020) in his article “Will Deepfake do Deep Damage” such videos to be used for revenge porn where a person’s ex-partners face is placed in place of a porn star.[24] These acts committed by a certain person can truly hamper someone's life degrading them and harming their social status. In addition, it would be pretty hard to find the culprit, creating the videos as distribution of media files is so much easier. It would be really difficult to detect the source.

We learn more about the current state of Deepfake content in the world from an article in Tech Times. In his article Samson(2019) shares the research of an Amsterdam-based cybersecurity company Deeptrace. They determined an 84 percent rise of Deepfake content in the seven months. From the Deepfake contents found 96 percent consists of pornographic content featuring women and half of them are American and British celebrity figures. In addition they also shared details of porn sites dedicated specifically to Deepfake contents having about 13000 Deepfake videos in 9 different sites. They are also available in 8 of the top 10 porn sites. There are also new businesses which make Deepfake videos for a price. One such business asks for about 250 pictures of a person to create a Deepfake and some of them charge as low as 3 USD.[18]

Lets us look at some of the damages Deepfake video is doing in our technologically advanced society. Hollywood celebrities like Kristen Bell and Scarlett Johanson addressed their distress for not having a say on how their faces were being pasted in porn. Willen C. from Insider informs their readers about how this actress and many others from the entertainment industry has to go through such horror. These manipulated videos are very convincing and even if they are labeled as fake it degrades the subject of this manipulation. The article also sheds light on Scarlett Johansons failed attempt to copyright her pictures from being used in manipulation as the same laws are irrelevant overseas. [27] Other than popular actors and actresses Deepfake pornography can harm any public figure whose pictures are available in the internet. These lesser known figures will have a more difficult time in convincing the authenticity of the video in question. Scott D. form Elle shreds light on the story of Noelle Martin where she describes her harassment from Deepfake pornography. She explains she was convinced it was her but she has never done such acts. She knew it was just her face but she was helpless to stop such contents.[26]

Brown(2019) reports in her article "Half of Americans do not believe Deepfake could target them online" about 47% Americans not believing they could become victim of Deepfake contents.[12] This static shows how easily the public would believe in Deepfake content and help it spread fake news or harm others.

It has become so easy to make Deepfakes that any individual with a computer and an internet access can create them. Recently, more and more mobile applications are being made that allow Deepfake generation in the hands of any ordinary person with a mobile device. Some of these applications are Zao, Doublicat and FaceApp which offer different services using variations of the Deepfake technology. Currently the only use of creating Deepfakes is for entertainment purposes. Deepfake application users share contents of them being a superhero or performing an action scene from a famous movie with their own face pasted over the actors. Other than that Deepfake does is mostly associated to cause harm then to be used for something more relevant.

Hence, we can see the severity and dangers of Deepfake and how it is rapidly increasing. Detection technology can not help from people making the videos, but it can very well hamper the sharing of such content in the world wide web.

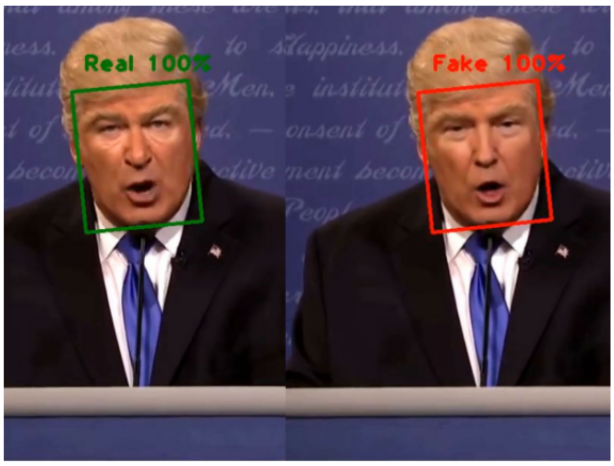


Figure 1.1: Deepfake featuring US president Donald Trump

We are using a method to detect face wrapping artifacts that are generated while creating Deepfakes. These artifacts can be found in the area surrounding the face that has replaced the original face. Our program will take the Deepfake video file as input and run our detection algorithm. This algorithm will use Convolutional Neural Networks(CNNs) to detect the facial artifacts and expose them. The differences our algorithm will detect are the condition of lighting and shape differences on the original and the fake part. We attempt to improve the effectiveness of the algorithm used through our research.

Li Lyu (2019), in their paper "Exposing DeepFake Videos By Detecting Face Warping Artifacts", has taken the path of using Artificial Intelligence (AI) to detect Deepfake contents. Their method is basically an AI algorithm that detects another Deepfake Algorithm. Their method focuses on training convolutional neural networks (CNN) with manipulated and real content. They tested their method using four different CNN models and had varying success rates from 84% to 99%. Although their results may look very promising, the authors have stated various issues that are not solved yet. They think that the presence of glitches are one of the main reasons that CNN can detect with such a success rate. So if the Deepfake technology is of better AI and of high resolution and quality, their model could fail to detect such videos. [19]

We will use the paper above for a guide and attempt to build a detection algorithm that could detect using more details than glitches alone. We will use CNN to detect more details like lighting condition differences on the face wrap and the surroundings and build a more advanced algorithm that can detect Deepfakes which could avoid simple glitches during their build up.

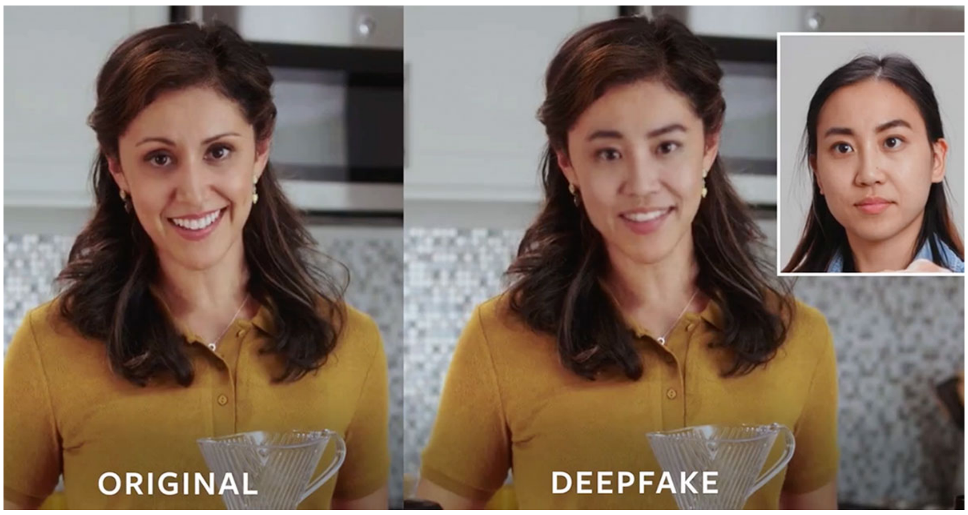


Figure 1.2: Comparison between Deepfake and Original

Chapter 2

Related Work

In their paper Amerini, Galteri, Caldelli, Bimbo (2019), "Deepfake Video Detection through Optical Flow Based CNN", attempts to differentiate between real and fake videos using a new forensic technique. Unlike other technologies, they are not using the single frame technique from taken videos, rather try to utilize optical flow fields to detect dissimilarities exploiting possible inter-frames. For that purpose they are using CNN Classifiers. Here, they obtain very promising results validation upon the FaceForensic++ dataset. To describe their method in detail, They are taking Optical Flow of the video, that is in this case a vector field, which is computed on two consecutive frames f(t) and f(t+1) to extract motion between the observer and the scene itself. They believe that it'll be able to discern between a Deepfake video and a regular video made by a video camera using the motion that is generated in these two types of videos. In their research the optical flow matrices take fake and unusual movements from the features of a person's face on a video frame to get an optical flow named PWC-Net from the data. Afterwards, Flow-CNN, a semi-trainable CNN, is created based on a pre-trained network. Following that, they perform test operations with VGG16 and ResNet50 networks. Finally one output unit followed by a sigmoid activation is placed at the end of the net for binary classification. From their obtained preliminary results it shows this kind of feature is able to point out some existing dis-homogeneous between the two analyzed cases and is very promising in detecting video manipulations.[17]

The authors Hasan and Salah (2019), from "Combating Deepfake Videos Using Blockchain and Smart Contracts," proposed a method to combat Deepfakes with the help of "Blockchain and Smart Contracts". Here, they try out an Ethereum Blockchain based solution which is capable of determining the legitimacy of a video file by providing credible and secure traceability to a trusted artist or publishing source where the content makers are individuals working as a filmmaker, photographer, paparazzi, reporters etc. all of whom working with video and image files. Their proposed system consists of entity relations, sequence diagrams, and algorithms used for Ethereum Smart Contracts to control and govern interactions and transactions among participants. They are storing the video EXIF data in a decentralized IPFS storage which is used to locate the correct owner of the content. It also has off-chain resources which is the Ethereum address of the owner linked to. It also contains a decentralized reputation system to maintain proper reputation which helps track down the video to its original owner. Their code is properly tested and can also be applied to other contents like audios, photos and manuscripts.[15]

Similar ideas like which Haya R. Hasan and Khaled Salah (2019) had been done before by a US Based start-up called Truepic. Hao (2018), "Deepfake-busting apps can spot even a single pixel out of place", writes about Truepic system which involves their user uploading pictures and videos to their own server to protect their integrity. So that any attempt at forgery can be easily discovered by comparing it with the content from the company's server. Here, the developers hope their proposed technology can be used to collaborate with social media companies in the future. The technology in Trupic also uses Blockchain to ensure credibility of stored content even though it means the success of their system depends on them gaining trust with all the images and video contents uploaded. [6]

We found more similar efforts are being made at websites like Gfycat to prevent Deepfake uploading. [8] They attempted this via Artificial Intelligence and Face Detection to spot inconsistency in the frames, especially in the facial area. When someone flags the video as a Deepfake, a second program is being run where the program checks if a video with similar facial features have been uploaded in the website before. If such video is found, then the program checks the facial feature between the new and the old video and if any dissimilarity has been found, then the video is flagged as a Deepfake, and removed. But for this method to work, regular companies can not possibly do it because they lack the data to check, wherein Gfycat has millions of videos. Also this method won't be very useful when a totally unique video such as CCTV footage is being tested.

Another method of Deepfake detection is through detection of blinking on the generated face. This has not been done well on fake videos. Li, Chang, Lyu (2018), in their paper "Exposing AI Generated Fake Face Videos by Detecting Eye Blinking", describes their method which starts by detecting faces in each frame of a video clip. Secondly, the faces are aligned into the same coordinate system to disregard head movements and changes in orientations which are based on detection of facial landmark points that have been set up. Afterwards, regions corresponding to each eye are extracted out to form a stable sequence of frames. Upon completion of these pre-processing steps, eye blinking detection is done quantifying the degree of openness of an eye in each frame of a video using the Long-term Recurrent Convolutional Networks (LRCN) model where it is trained based on image datasets of eye open states of a person. Finally, the algorithm is tested to detect eye blinking on authentic and fake videos generated with the DeepFake algorithms from their dataset. From their results they determined the LRCN method to show the best results compared to CNN. LRCN takes advantage of long term dynamics to effectively predict eye state, such that it is more smooth and accurate, with temporal domain, LRCN can memorize the previous state and it knows if blinking has occurred before, the eye state in the next couple frames is very likely to be open . Thus, it knows if there is no trend of eye closing before, the eye state of the next frame is very likely to be open.[11]

The authors Hsu, Zhuang, Lee (2020), of "Deep Fake Image Detection Based on Pairwise Learning", proposed a deep learning based approach for detecting fake images by using the contrastive loss between pairs. Firstly, several state-of-the-art GANs are employed to generate the fake-real image pairs from the dataset. Next, the reduced DenseNet is developed to a two-streamed network structure to allow pairwise information as the input. After that procedure is done, the proposed feature network is trained using the pairwise learning to differentiate between real and fake images.Their fake face detector is based on the novel CFFN, with a backbone of DenseNet and Siamese network. Cross-layers features are investigated by the proposed CFFN and finally the pairwise learning is used to improve the property of proposed DeepFD. They used a dataset called CelebA which was trained on various GANs like DCGAN, WGAP, LSGAN, WGAP-GP and PGGAN. Their experimental results showed that the method that they proposed outperformed the other Deepfake image detector.[20]

Yang, Li, Lyu (2018), in their research ”Exposing Deep Fakes Using Inconsistent Head Poses”, proposed a new method to expose AI-generated fake face images or videos to a set of standard landmark locations, a process known as face alignment. They have followed the method based on observations that such deep fakes are created by splicing synthesized face regions into the original image and doing so, introducing errors that can be revealed when 3D head poses are estimated from the already extracted face images from videos. Here, they have used frames from 35 real and 35 fake videos in the UADFV dataset with a total number of 21,694 images to train their SVM classifier. Secondly, they further trained SVM classifiers based on the differences between head poses estimated using the full set of facial landmarks and those in the central face regions to differentiate Deepfakes from real images or videos as the central face region is from the synthesized face, the errors due to the mismatch of landmark locations from original and duplicate gives larger difference between two head poses.[10]

In this paper Koopman, Rodriguez, Geradts (2018), ”Detection of Deepfake Video Manipulation”, tried to use PRUN (Photo response non uniformity) analysis to detect the Deepfake video manipulation. Firstly, PRNU is a noise pattern found in digital images which are created by small factory defects in the light sensitive sensors of a digital camera [Lukas et al., 2006]. [2] For their dataset they have used 20-40 seconds video clips that are taken using a Canon camera and in .mov format. Then they have used open source Deepfake GUI OpenFaceSwap [Anonymous, 2018] to apply Deepfake to said videos. Then every frame of the videos were exracted and cropped in the face area in PNG format using ffmpeg. Then the PRUN analysis is done and the mean normalised cross correlation scores per video and the variance in normalised cross correlation scores per video are calculated. From the results they could not find any correlation between video authenticity and the variance in correlation scores. They also found a correlation between mean correlation scores and authenticity. The difference they determined to identify Deepfakes from originals is that the originals have higher mean normalized cross correlation scores. [7]

Nguyen, Fang, Yamagishi, Echizen (2019), in their research paper ”Multi-task Learning For Detecting and Segmenting Manipulated Facial Images and Videos”, have designed a convolutional neural network which detects Deepfake images and videos using a multi-task learning methodology where the manipulated regions from the facial regions are located for each query in their training model. Here, the information it gains by performing a single task is shared with the other task and thus it achieves better performance in multiple tasks. For their model they chose a semi-supervised learning approach for the betterment of the generality of the network. Their network has both an encoder and decoder which is Y-shaped. From their decoder output of a single branch segmentation of manipulated regions is performed while the output from another branch performs reconstruction of the input, improving the overall performance of the network. The results from their experiments using the datasets from FaceForensics and FaceForensics++ provides a good demonstration of the networks capability in detecting any face swaps or manipulation in videos and images.[16]

Chapter 3

Machine Learning Algorithms

3.1 Convolutional Neural Networks

Dickson (2020) shares that CNNs were first discovered in the 1980s and the system was capable of detecting handwritten digits. In recent times CNNs are used regularly to perform computer vision tasks which were previously impossible without this algorithm. Convolutional Neural Networks function by having a number of artificial neurons containing mathematical functions to calculate the weighted sum from multiple inputs. The algorithm works out a single activation value from all the inputs taken. Here, the weights of each neuron means a different feature found from the video or image file taken as input. When CNNs take in an image file as input the several layers of this algorithm generates multiple highlights containing relevant features extracted from the image known as activation maps. From its multiple layers the first layer detects all the edges in an image. This output is used as an input for the next layer to identify more detailed features and it keeps going on deeper with each layer leading to detect things like objects and faces. Thus in the final convolution layer the output value determines the possibility of detecting an object in question. If it was supposed to detect a car the result will contain a confidence score which is a value between 0 and 1 to determine the likeliness of the object detected to be a member of a "class" representing cars.[23]

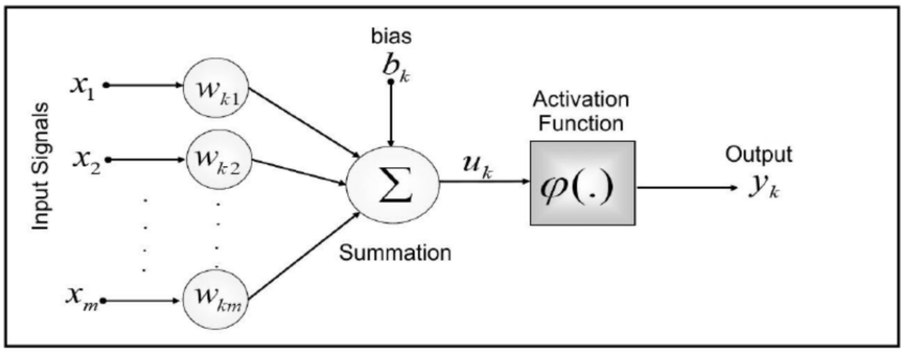


Figure 3.1: Structure of an artificial neuron

A typical CNN layers of an image file look like this

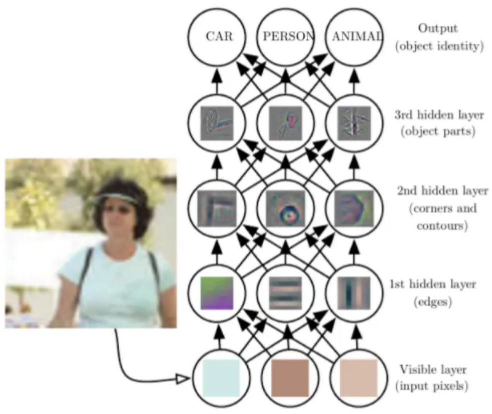


Figure 3.2: Layers of CNN of a single image file

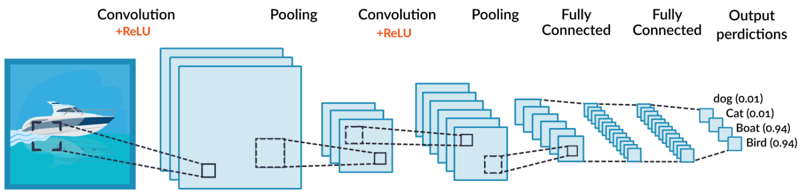


Figure 3.3: The CNN process

From the image above we can get an idea of how a CNN process is done. From the Input, a number of convolution layers are generated, which has an array of kernels that are learn-able. Then through sub-sampling, especially for images, it forces a sparse local connectivity pattern between neurons of adjacent layers, where each one is related to a small pool of the input. After the process Spatial arrangement is done. There are four properties for calculating the spatial size of the output volume:

W = The input volume size, which is the spatial size of output volume
K = The kernel field size of the layer
P = Amount of zero padding
S = Stride, which is important because it determines how the pixels are moved around. A stride size of 2 indicates we move two pixels at a time, because of this the value of stride must be a positive number.
The formula is given below:

$$\frac{W - K + 2P}{S} + 1 \tag{3.1}$$

The output is an integer, if it does not return an integer then the value of S is incorrect. One of the most important parts is pooling, which is a form of non-linear down-sampling. The most common form of pooling is with a size 2x2 filter with 2 strides.[21] The function is:

$$f_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b} \tag{3.2}$$

If we visualize this it looks like this:

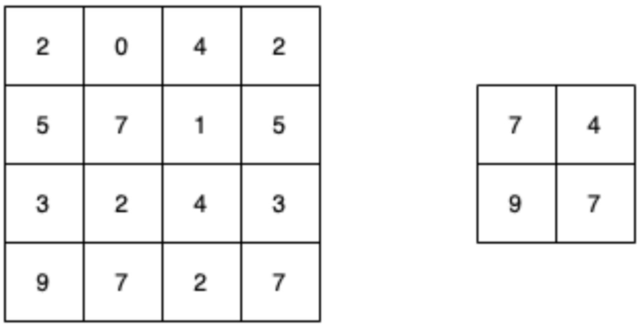


Figure 3.4: Example of max pooling on a 2x2 filter where S=2

Pooling is usually used for object detection. After pooling is done, we have ReLU layer, which is rectified linear unit layer that basically removes the negative values of an activation map, the function is

$$f(x) = max(0, x) \tag{3.3}$$

ReLU is preferred because it reduces training time but does not affect the accuracy to an extent which would compromise the model. Finally, after multiple convolutions and max pooling layers, the models are generated using fully connected layers. Fully connected layers have connections to the previous layers in the process.

3.2 Inception

Valigi, 2016 explains the concept of Inception in his article Short history of the Inception deep learning architecture from the paper Network in Network. In his article he talks about all versions of the Inception model from various research papers. He informs about conventional convolutional filters to only possess the ability to learn only linear functions of their inputs. To make CNNs learn non-linear functions the convolutional layers can be connected through multi-layered perceptrons which are mathematically equivalent to 1x1 convolutions. These increased abstraction ability disregards the need for fully connected layers at the top of the network. This way the number of parameters required becomes less reducing risks of overfitting and computational loads. However, they perform an operation called global average pooling that improves robustness and spatial translations. They do this by using spatially average feature maps at the final layer and immediately input the vectors to the software classifier.[4]

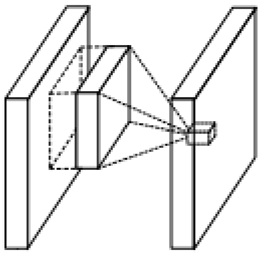


Figure 3.5: A linear convolution layer with a linear filter

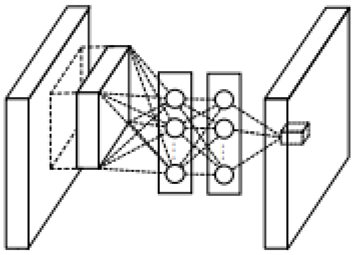


Figure 3.6: A linear convolution layer with a linear filter

Valigi then explains the next version through the paper "Going deeper with convolutions". The authors proposed a more improved version of the original Inception model in their paper This particular model improves the convergence of deep networks by introducing additional losses tied to classification error of intermediate layers. These layers are only used for training and their outputs are not taken into consideration during inference. [4]

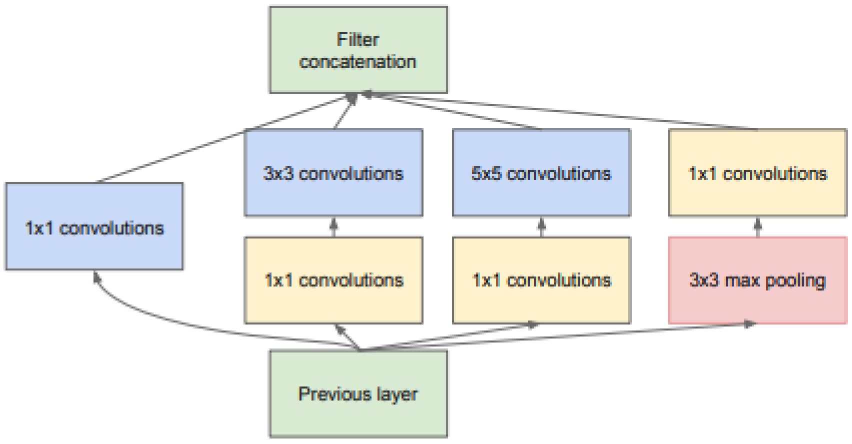


Figure 3.7: Inception Module with dimension reductions

Valigi, 2016 further shares about the authors Szegedy et al. (2015) behind the Inception v1 model from their work "Rethinking the Inception Architecture for Computer Vision". Here, they propose expressing any convolution with the kernel being larger than 3x3 by using a series of smaller convolutions like replacing 7x7 filters with a pair of 1x7 and 7x1 convolutional layers.

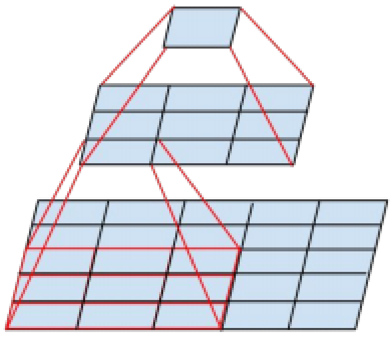


Figure 3.8: Mini-network replacing 5 x 5 convolution

The next model Inception v3 applies all the tricks of Inception v1 and Inception v2 on the same net. Finally, comes the Inception v4 model. Valigi shares a summary from the paper by Szegedy et al. (2016), "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning" which talks about the Inception v4 architecture. It is a more efficient version of Inception v3 containing more uniform architecture and has better recognition performance.[4]

3.3 ResNet

Before diving into details about Xception let's take a look at ResNet architecture. We will use a pretrained ResNet model to compare against the results of Xception in the later chapters. ResNet or Residual Net is also a CNN architecture that has hundreds or thousands of convolutional layers. It was introduced as the first neural network capable of training hundreds or thousands of layers without having the "vanishing gradient" problem. In neural networks the training process is done by back propagation that relies on gradient descent moving down the loss function to find the weights that minimize it. Thus, if it stumbles upon hundreds of layers the multiplication operation makes the gradient smaller and smaller until it vanishes. To overcome this problem ResNet stacks up on identity mappings, and skips over layers that have no initial purposes, while using the activation from previous layers again. As it skips the network is compressed into lesser layers allowing faster learning. On the next training, the layers are expanded and the feature space is explored thoroughly by the residual parts of the network.[28]

3.4 Xception

Chollet, 2017 in his research paper ”Xception: Deep Learning with Depthwise Separable Convolutions”, talks about Xception. It has emerged from conventional CNNs to Inception models whose architecture is based entirely on depth-separable convolution layers. This particular model derived from Google’s Inception models. The name Xception means “Extreme Inception”.[5] Here, the mapping of cross-channel correlations and spatial correlations within the feature maps of CNN will be completely decoupled. It consists of thirty-six convolution layers which forms the base of the network for feature extraction. To further simplify Xception architecture is a linear stack of depthwise separable convolution layers with residual connections.

Fabien, 2019 explains the Xception model and Depthwise Separable Convolutions. The Xception model has been developed by Google researchers that is entirely based on Depthwise Separable Convolutions. In an Xception model the data goes in through the entry flow, followed by the middle that is repeated eight times before it goes into the exit flow. The figure below shows how the Xception model looks like [14]

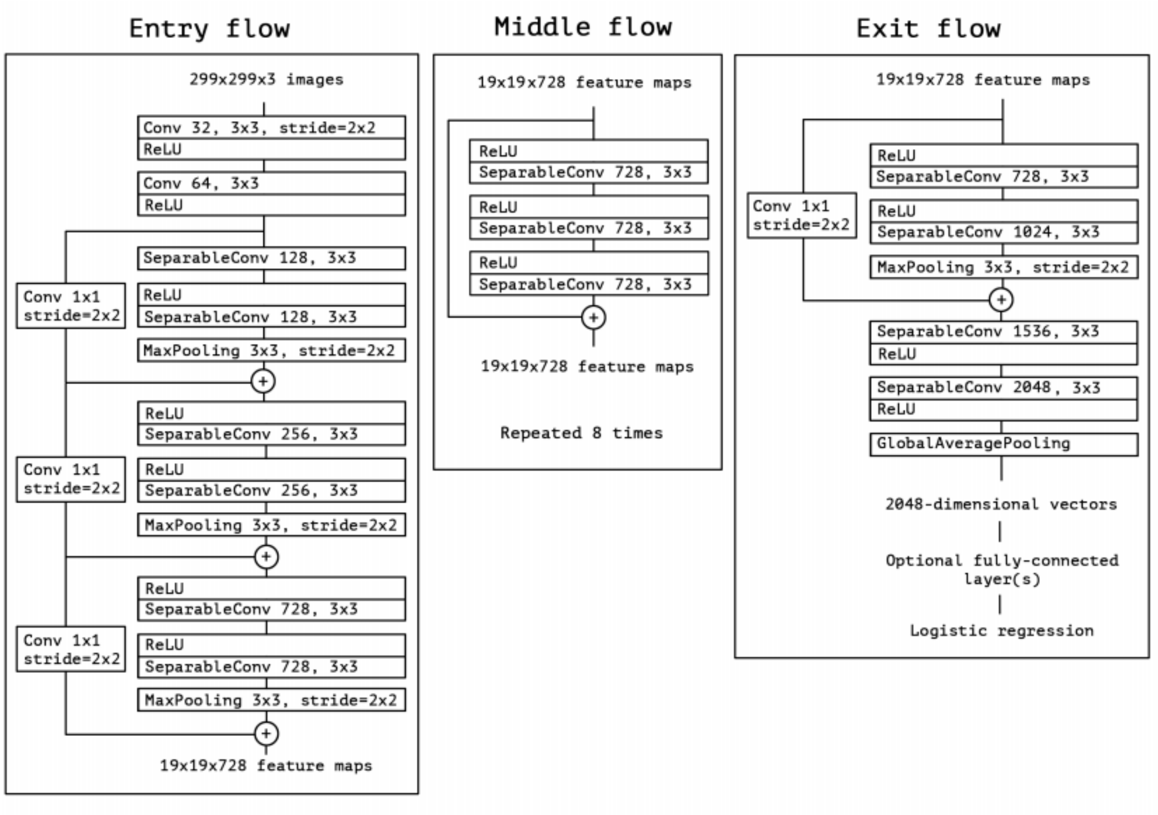


Figure 3.9: A look inside Xception model

3.5 Depthwise Separable Convolutions

To understand the first main point we take a deeper look into Depthwise Separable Convolutions. Wang, 2018 informs readers about Separable Convolutions. Here, he shares in details about the Depthwise Separable Convolution process. This particular convolution separates the kernel and performs two separate convolutions the depthwise and the pointwise convolutions.[9]

From Fabian, 2019 we also learn that Depthwise Separable Convolutions are supposedly more efficient in terms of computation time. Firstly, we venture into the details of the convolution process. We take an input image containing a certain number of input channels C and the certain number of dimensions A. Then we can apply a convolution of a certain filter size dxd. For a single kernel the operation cost is $K^2 * d^2 * C$ and for N number of kernels the cost becomes $K^2 * d^2 * C * N$. This is a very expensive operation that greatly increases the computation cost. Thus Depthwise Separable Convolution has been introduced.

Following the Depthwise Convolution we apply a convolution of size $d * d * 1$ rather than $d * d * C$. The computation is performed only over a single channel. Now the size of the first created volume is $K * K * C$ instead of $K * K * N$. Then we move on to the next step which is Pointwise Convolution.[14]

Wang, 2018 also explains pointwise convolution as being a type of convolution which only uses $1 * 1$ kernel . The kernel iterates through every single point and has a depth equal to the number of channels. In the next step the convolution is done with size $1 * 1 * N$ over the $K * K * C$ volume. This ends up being the size $K * K * N$. By following this method the computation has been decreased by a factor of $1/N$. However in Xception implementation the Pointwise convolution is performed before Depthwise Convolutions.

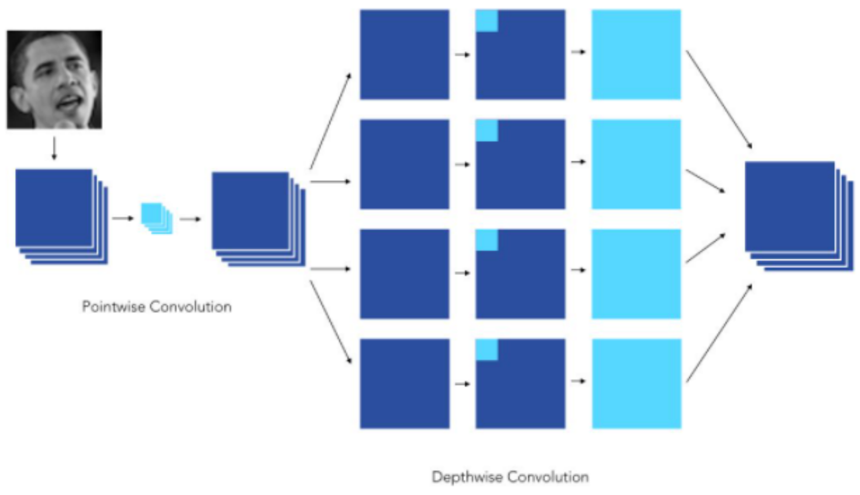
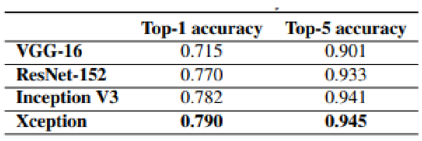


Figure 3.10: Implementation of Xception

Xception performs way better than Inception v3 on larger image classification datasets which contains 17,000 classes but we don't see any drastic improvements on the ImageNet dataset. Both Xception and Inception perform with the same number of parameters.



	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Figure 3.11: Implementation of Xception

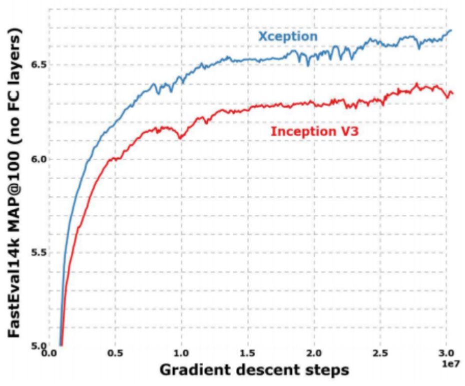


Figure 3.12: Xception versus Inception v3 on large dataset without fully connected layers

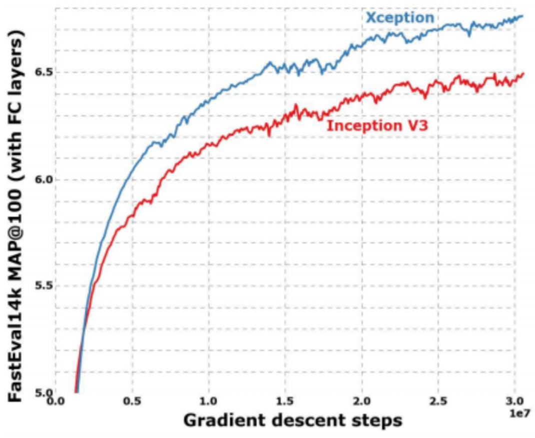
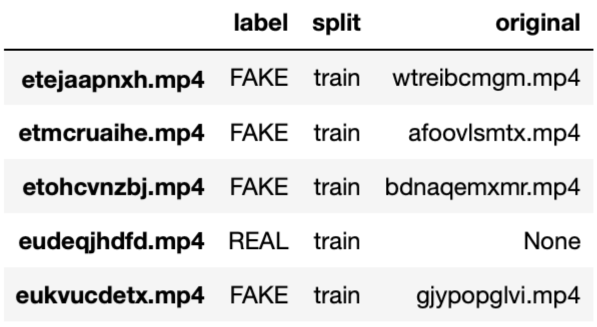


Figure 3.13: Xception versus Inception v3 on large dataset with fully connected layers

Chapter 4

Data Analysis

As the threat of misinformation grows with Deepfake many institutes and sources are made available online with required data and resources to help anyone willing to create reliable detection mechanisms. Kaggle.com allows their users to find and use data sets and hold machine learning competitions and one they have provided a dataset for a Deepfake detection challenge. This particular dataset contains 800 video files, where we have both a collection of fake and real videos. From this set of files 400 videos are used to train/validation and the other half id for testing purposes. Each video file in the training folder is identified by their labels which confirms if their contents are real or manipulated. If a video file is labeled as fake it has another label called original containing the name of the initial video file. Thus we choose to fully utilize this particular dataset and teach and test our model for detecting manipulated video files.
A short list of the data table from our dataset is provided showing how the data are placed in the table.

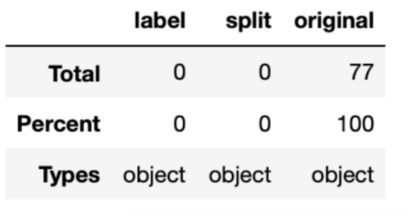


	label	split	original
etejaapnxh.mp4	FAKE	train	wtreibcmgm.mp4
etmcruaihe.mp4	FAKE	train	afoovlsmtx.mp4
etohcvnzbj.mp4	FAKE	train	bdnaqemxmr.mp4
eudeqjhdfd.mp4	REAL	train	None
eukvucdetx.mp4	FAKE	train	gjypopglvi.mp4

Figure 4.1: Portion of the dataset json file

We can see a label is given to every video file and if they are fake and the link of the original video is available.

The next table shows the total number of missing data in the original column which is the list of all unaltered videos.



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

Figure 4.2: Number of REAL videos in dataset

A bar chart showing the number of REAL and FAKE videos in the dataset In

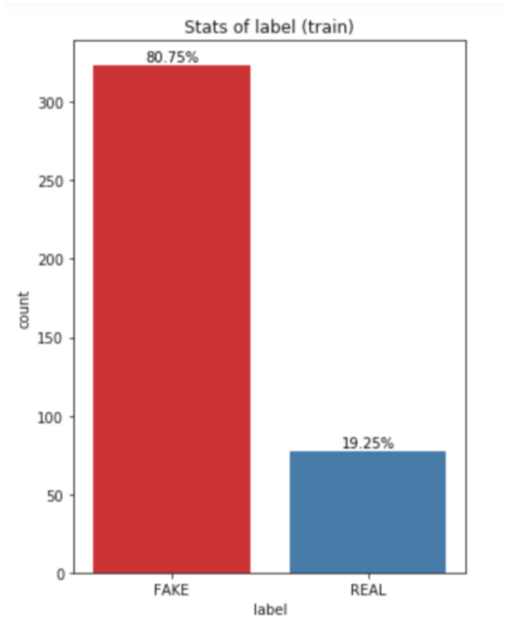


Figure 4.3: Percentage of REAL and FAKE video files in dataset

the next page we'll show some random frames from REAL and FAKE video files in datasets, showing different positions and movements of the subject are shown below. These video files will be used to train our program and learn the differences between REAL and FAKE.

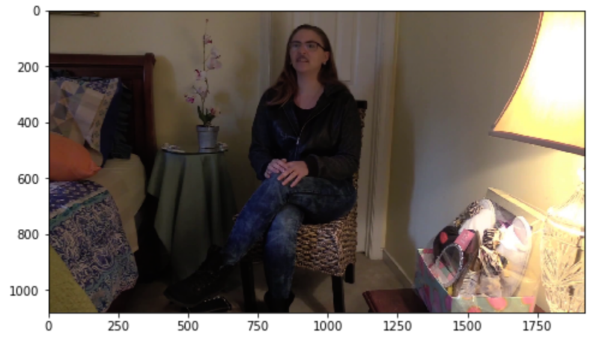


Figure 4.4: Sample frame A from Fake videos

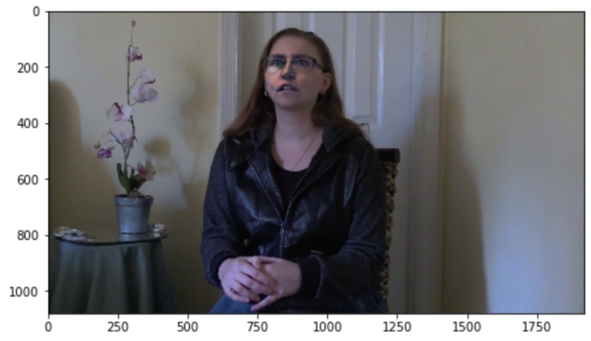


Figure 4.5: Sample frame B from Fake videos

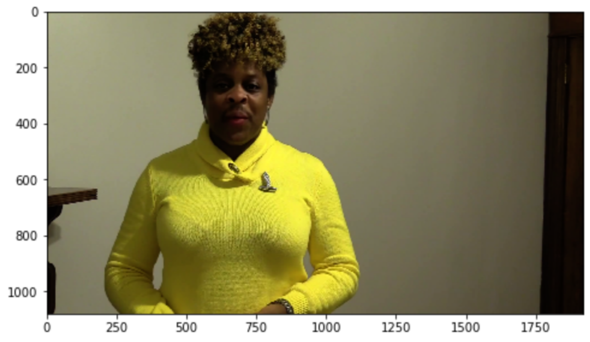


Figure 4.6: Sample frame C from Fake videos

Figure 4.7: Sample frame A from Real videos

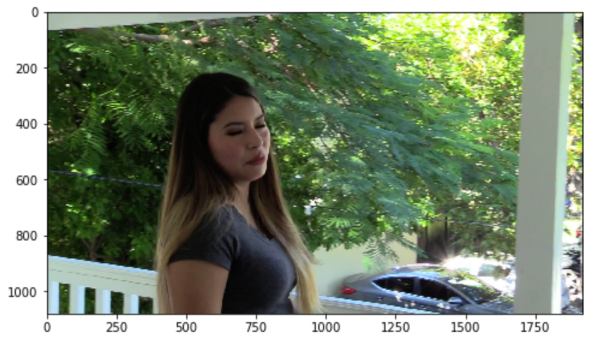


Figure 4.8: Sample frame B from Real videos

Figure 4.9: Sample frame C from Real videos

This particular dataset contains various types of video files in different lighting conditions and videos containing multiple subjects which is required to detect possible deepfakes in such cases.

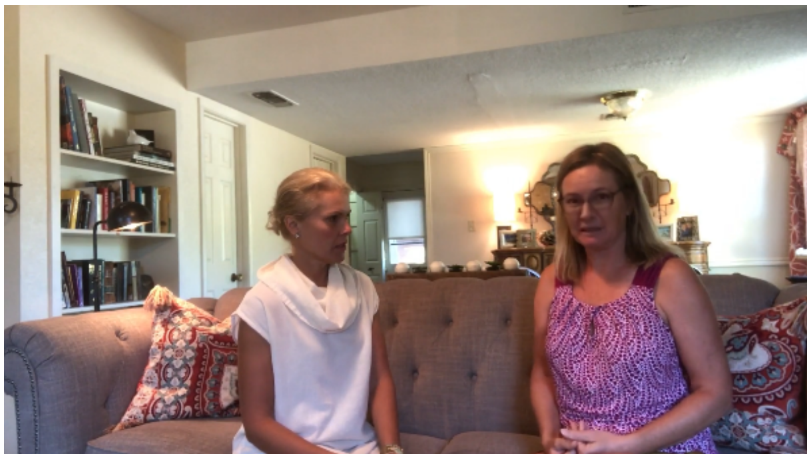


Figure 4.10: Video frame containing two subjects, only one of their faces is manipulated

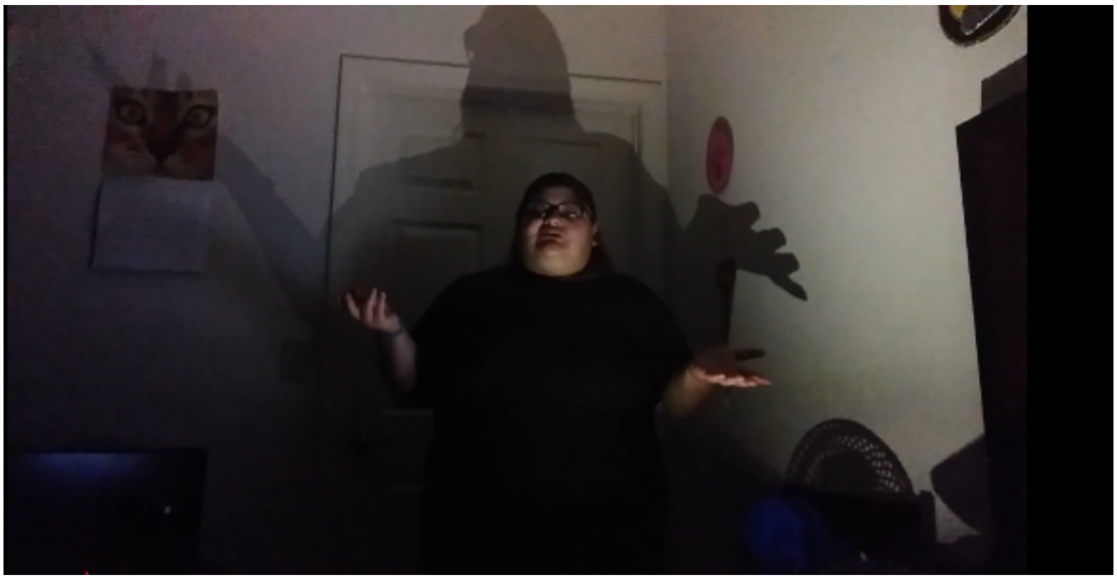


Figure 4.11: Video taken in low lighting condition

Now let's look at some individual video files which might not be helpful to train at all which we will consider as garbage data. One of the frame from such video looks like Figure 4.12 For these types of videos, we will discard them from our dataset. We

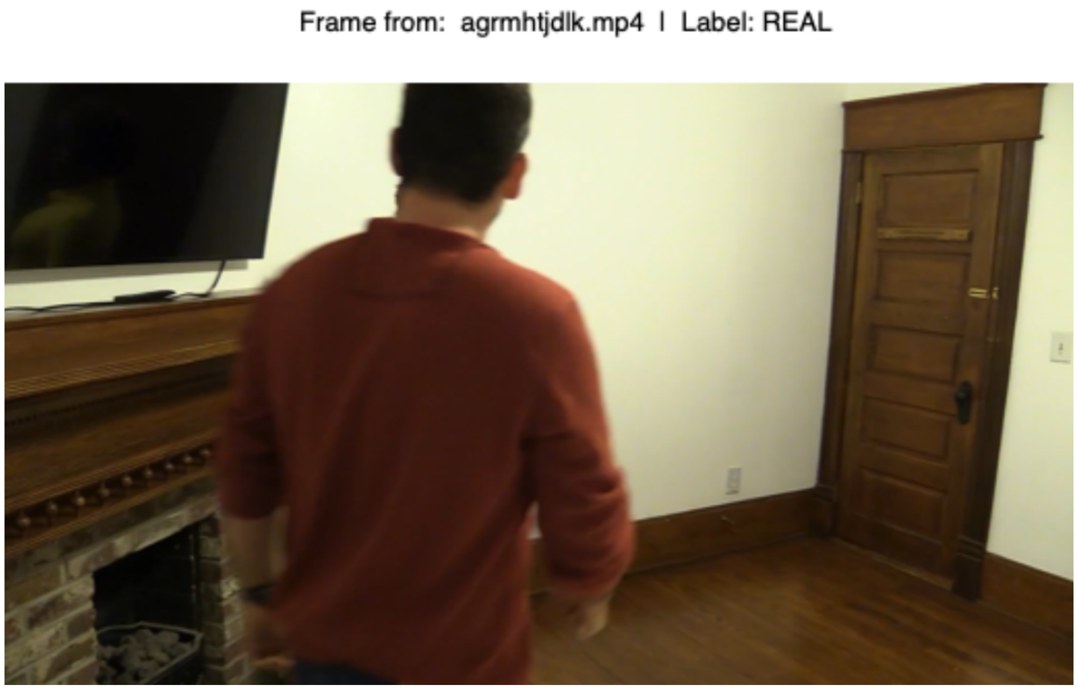


Figure 4.12: Video frame without any faces in it

didn't find the best way to remove them automatically, so we used python to show frames of videos and manually decided whether or not we should keep the videos for the dataset. This way we ended up with 387 videos for our dataset. We chose

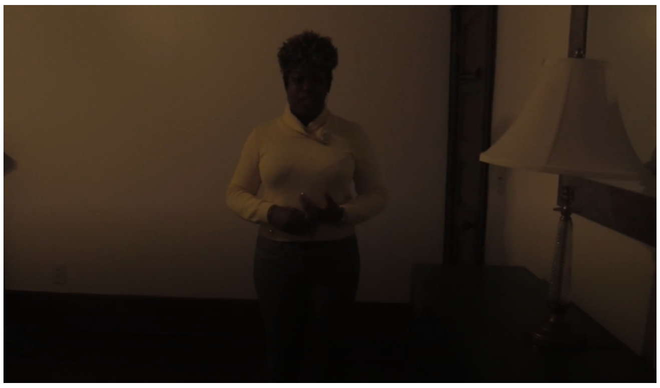


Figure 4.13: Very low light video

to disregard and skim through videos with very bad lighting conditions like the one in figure 4.13 as well because the face detection mechanism cannot detect any face from such video files. The face is not even clear for the naked eye to detect. Because the number of videos we will use for training is not high enough we will disregard videos such as the one in figure 4.13.

Chapter 5

Data Processing

The first step of processing all the data that we have is understanding the limitations we have at this moment. Since access to a good GPU equipped computer was not possible, the biggest bottleneck to train our data would come down to the GPU Video encoding when using ffmpeg. Tensorflow would also gain some advantages if used with CUDA cores of NVIDIA GPUs. As such we used CPU to process our data.

Beforing diving into processing, let's learn some information about the tools we're about to use

5.1 Tools Used

5.1.1 Tensorflow

Goldsborough (2016), shares the information about a symbolic math library mainly based on dataflow and differentiable programming known as Tensorflow. This library can perform operation with multiple CPUs and GPUs and is not limited to running on single systems. The Tensorflow system is designed to permit clean deployment of compassion amongst various platforms, and transforms single computer gadgets to clusters of servers to mobile and other devices. All computations in TensorFlow are expressed as stateful dataflow graphs. From the paper we learned Neural Networks can carry out operations on multidimensional facts arrays, that are known as tensors. Thus, TensorFlow gets its name from the arrays. This tool isa gadget mastering machine that operates on a big scale and in heterogeneous environments. As for the flow part of its name it uses dataflow graphs to symbolize computation. The nodes in the dataflow graph are mapped across multiple machines in a cluster, and inside a gadget throughout more than one computational device. Its structure is designed to offer flexibility to any utility developer. TensorFlow supports many applications by providing a focal point on education and inference on deep neural networks. The TensorFlow Distributions library implements an imaginative and prescient of opportunity concept tailored to the cutting-edge deep-mastering paradigm of quit-to-quit differentiable computation. Building on basic abstractions, it gives bendy constructing blocks for probabilistic computation. Distributions offerspeedy, numerically strong strategies for producing samples and computing statistics, e.g., logdensity. Bijectors offer composable volume-monitoring modifications with automated caching.Together those permit modular creation of excessive dimensional distributions and modifications now no longer viable with preceding libraries. Computational graphs are used by TensorFlow to get the system to learn about the algorithms. [3]

5.1.2 Torch

Collobert, Bengio and Mariethoz (2002), in their paper about the machine learning library known as Torch informs us about the creation and usage of Torch. Here, this library is built based on an item situated worldview. Moreover, Torch is executed using the language C++. This library is accompanied by some expansive classes for the disentanglement of alteration due to existing calculations. The first of these classes is the DataSet class. This particular Torch class is tasked with handling all information. The next class is called Machine which functions by representing a black-box which takes an input and some parameters to provide with an output. The third class is called Trainer and it selects an optimal set of parameters for the Machine class following a given criteria and DataSet. Furthermore it also performs test operations using a different or same DataSet. Then comes the fourth class known as Measurer and it performs printing of files on different proportions of interest which could be something like a classification or a mean-squared error. From all the classes a basic idea of Torch can be established. This library first uses the DataSet to produce a given number of "preparing models". Secondly, the Trainer offers these models to the Machine which provides an output that is fed back to the Trainer to fix the parameters for the Machine. As this process continues the Measures runs to find the efficiency of the system. However, some Machines can only be trained by some specific Trainers. Gradient Machines can only train with the help of gradient descent. Similarly, Support Vector Machines can only train by a trainer specialized on constrained quadratic problems. Finally, distribution including Gaussian mixture models needs either gradient descent or Exception-Maximization Trainer. [1]

5.1.3 OpenCV

OpenCV is a computer vision library. Computer Vision is defined for understanding meaningful descriptions of physical objects from the image. OpenCV was built to provide an infrastructure for computer vision. This library has a huge range of optimized machine learning and computer vision algorithms. These algorithms include object identification, detecting and recognizing faces, object movement tracking, etc. OpenCV provides support for C++, Python, Java and MATLAB programming languages and works on Windows, Linux, Android and Mac Operating Systems.

The common features in OpenCV are read and write images, save and capture images/videos, filter or transform the image, detecting faces,eyes,cars in images or videos, perform feature detection, background subtraction, and tracking objects.

Major functionalities of OpenCV are image and video processing, object and feature detection, computational photography. [25]

From an education material on stackshare Unknown n.d. Informs us OpenCv was designed to speed up computation and provide more attention to real-time applications. This library faster performance is because it is optimized with C/C++ language and takes advantage of multi-core processing. **28**

5.2 Face Detection

Right now we are using the Haar-Cascade classifiers to detect objects. It's an open-source classifier which can be found in the openCV github repository. We have an ObjectDetector class in which we're feeding some of the classifiers from haar-cascade to detect Faces and Eyes for now. We might add some more detection (eg. Smile, profile, neck/chin) later on to have better accuracy. Then for now we're taking some real and fake videos and passing them through detect_objects functions which use the ObjectDetector class to detect and show objects all with the help of OpenCV. This is the initial step towards finding, as this step makes sure we have a proper pixel location of the faces so we can distinguish between the faces and the background before we dive deeper into detecting the face wrapping artifacts to reach a conclusion using our proposed method.

Fabien in his article "A guide to face detection using python" describes the working of the Haar-Cascade classifiers in detail. In Haar features are some common features in most human faces which includes, a dark eye region or a bright nose region. These features are detected using intensity difference within a face. The calculation is done by the addition of pixels in black areas and subtraction in white areas.

$$RectangleFeature = \sum(Pixels_{blackarea}) - \sum(Pixels_{whitearea}) \tag{5.1}$$

The above equation is applied as a convolutional kernel over the entire image. When the features get selected, they are trained with Adaboost classification to create an ensemble model. This helps to point towards a single feature that has the best negative and positive examples. Cascade classifiers are used to increase detection rate and reduce computation time. It rejects sub-windows that do not contain a face and accepts the ones that do. Multiple classifiers are applied to every sub-windows. The classifiers here are trained using Adaboost. [13]

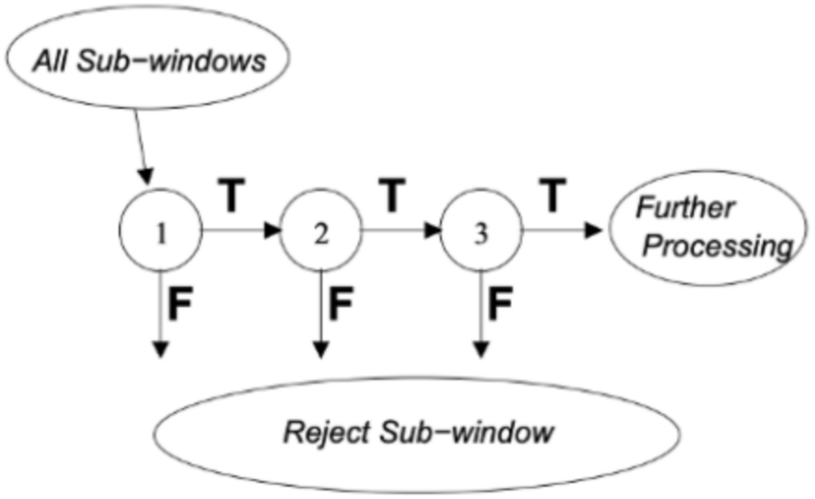
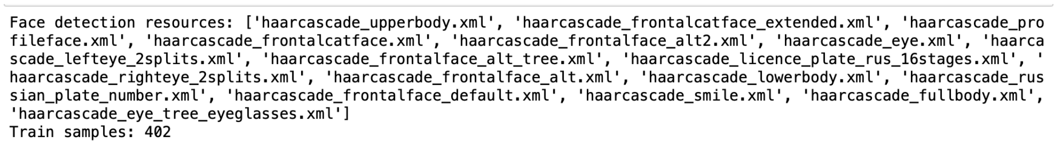


Figure 5.1: Cascade Classifiers Working Mechanism

So to start the face detection process, we load the haar-cascade classifiers and see if can detect faces properly



```
Face detection resources: ['haarcascade_upperbody.xml', 'haarcascade_frontalcatface_extended.xml', 'haarcascade_pro
fileface.xml', 'haarcascade_frontalcatface.xml', 'haarcascade_frontalface_alt2.xml', 'haarcascade_eye.xml', 'haarca
scade_lefteye_2splits.xml', 'haarcascade_frontalface_alt_tree.xml', 'haarcascade_licence_plate_rus_16stages.xml', '
haarcascade_righteye_2splits.xml', 'haarcascade_frontalface_alt.xml', 'haarcascade_lowerbody.xml', 'haarcascade_rus
sian_plate_number.xml', 'haarcascade_frontalface_default.xml', 'haarcascade_smile.xml', 'haarcascade_fullbody.xml',
'haarcascade_eye_tree_eyeglasses.xml']
Train samples: 402
```

Figure 5.2: Haar-cascade classifiers

The results came out like this:

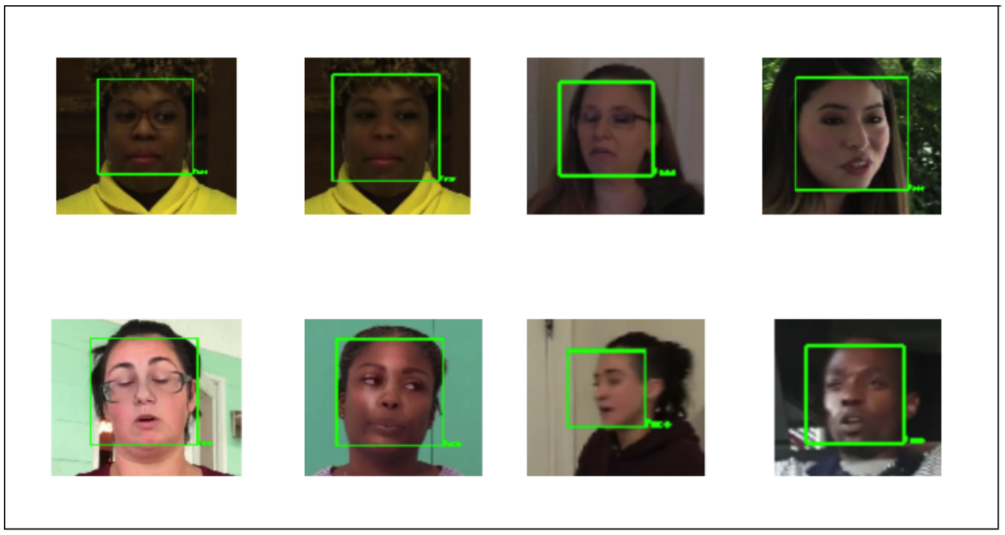


Figure 5.3: Face Detection Results

5.3 Seperation and Training

Since we are using 387 videos to train our model, we separate those video files for training and validation. We have used a 80/20 split for training, validation and already have 400 videos for testing.

For the training and validation dataset, we took 15 frames from each video, detected the faces in there, saved a separate image file of just the face and labeled the data in relation to their parent video file, We classified two classes, train_dataset and validation_dataset. Here are a couple examples of the final images we used for training.

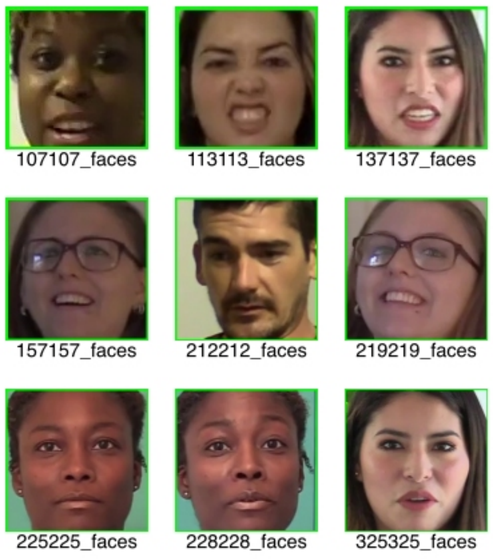


Figure 5.4: Final Images for the model (REAL)

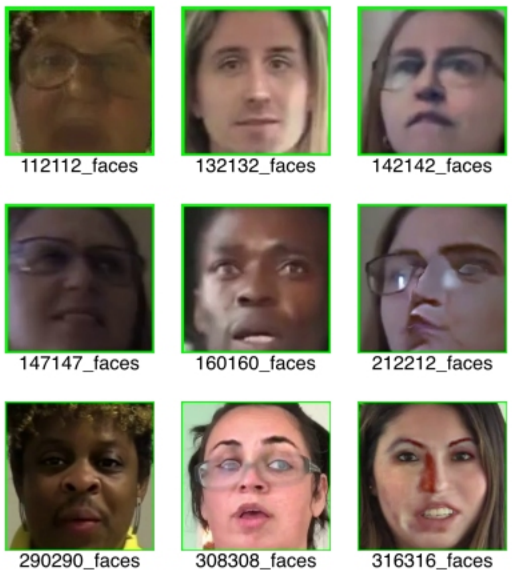


Figure 5.5: Final Images for the model (FAKE)

Then the model was defined as such: Since we are building our custom Xception model, we are using their pre-trained model's weight ‘imagenet‘ as a base. Then we are applying our custom classifier GlobalAveragePooling2D, Dense at 1024 with relu activation.

The kernel size is the size of the filters used in feature detection. Next, stride represents the number of steps we should move in each convolution step and lastly padding represents the number of pixels added to an image during processing. After convolution in one layer batch normalization is performed. In this layer the outputs from the previous layers are optimized and it allows each layer to learn more independently.
ReLU represents rectified linear activation function which is in charge of transforming the weighted sum input from the node into the activation of the node or an output. This is a piecewise linear function which outputs the inputs directly if it's positive or else it outputs a zero.

Lastly, Max Pooling or maximum pooling is an operation that calculates the largest value in each feature map. This highlights the most present features in the map.

The loss function for the model is Binary crossentropy, activation function is relu and for optimizer we chose Adam. Binary crossentropy answers with only two choices yes or 1 and no or 0. Cross entropy is a difference between two probability distributions and entropy alone is the number of bits required to transmit a randomly selected event from a probability distribution.

$$Loss = \frac{1}{outputsize} \sum_{i=1}^{outputsize} y_i \times \log \hat{y}_i + (1 - y_i) \times \log(1 - \hat{y}_i) \tag{5.2}$$

This formula allows us to calculate the loss in terms of 1 or 0.
This formula allows us to calculate the loss in terms of 1 or 0. Here, the variable represents the i-th scalar value in model output. Binary crossentropy is very useful for model training and can solve many classification problems at the same time as each classification choice has only one of two possibilities yes or no.

Optimizers such as Adam are algorithms that function to change attributes of neural networks to reduce losses during processing. They reduce the losses and provide us with a satisfactory result. We will use Adam (Adaptive Moment Estimation)for optimization. It is fast and convergence is rapid and rectifies vanishing learning rate with high variance. Using Adam is simple and it takes less memory and is invariant to diagonal rescale of the gradients. Moreover, it is effective for problems containing a large number of data or parameters or both and problems with noisy or sparse gradients. Although the computation cost is higher than other optimizers Adam is best to train neural networks in the least time with most efficiency.

Some important terms when training the model are:

- **Epoch:** One epoch means the algorithm will iterate through the whole dataset one time. We use more than a single epoch to get better accuracy as the weight values are changed multiple times in the neural network to provide a more optimal result. The right number of epochs needed is different for each dataset and for our purposes we try the training in different stages with different numbers of epochs.
- **Batch size:** These refers to the number of training examples present in a batch. We cannot push the entire dataset into the neural network to iterate at once so they are divided in batches.
- **Learning rate:** It is simply the rate in terms of probability that a model learns from training neural networks. Its value is in the range of 0 to 1. A learning rate shows how fast a model is trained to solve the problem. With a lesser learning rate more epochs are necessary as small changes appear in the weights in each iteration, but a high learning rate provides faster changes and requires lesser epochs. If a learning rate is very high the model converges very fast leading to a suboptimal result then again if it is too small the process may get stuck.

The training was done on a Laptop with Intel i5 5200U Mobile Processor with no dedicated GPU. If we had access to a newer machine with a better process and GPU we could have done a lot more training and fine tuning the algorithm.

The training of the model is done in two stages. For the first stage we chose 5 epochs and a batch size of 16. First state learning rate we chose $1e^{-3}$. Then for the next stage we chose 20 epochs with batch size of 8. The learning rate was $1e^{-4}$. A trained model is compiled after each iteration and the last generated model is kept as the final. We had 16 batch sizes in the first stage and 32 in the second stage.

We can see the results of that training in the next page.

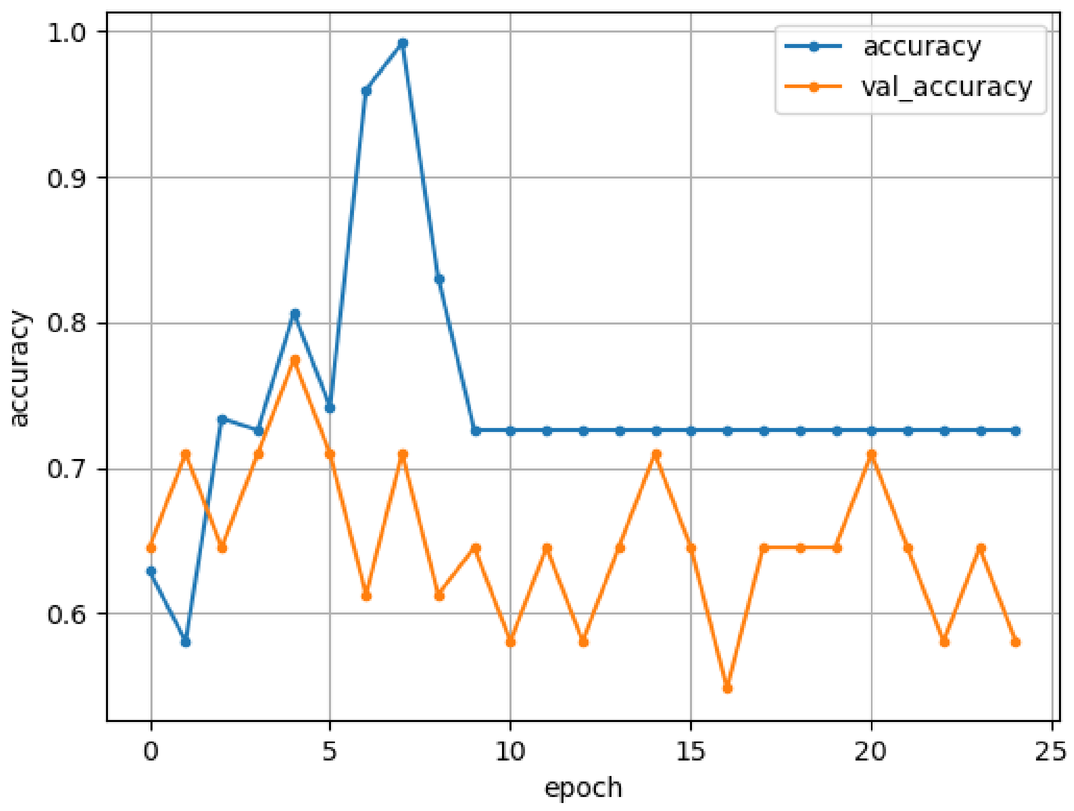


Figure 5.6: Accuracy Graph for first model

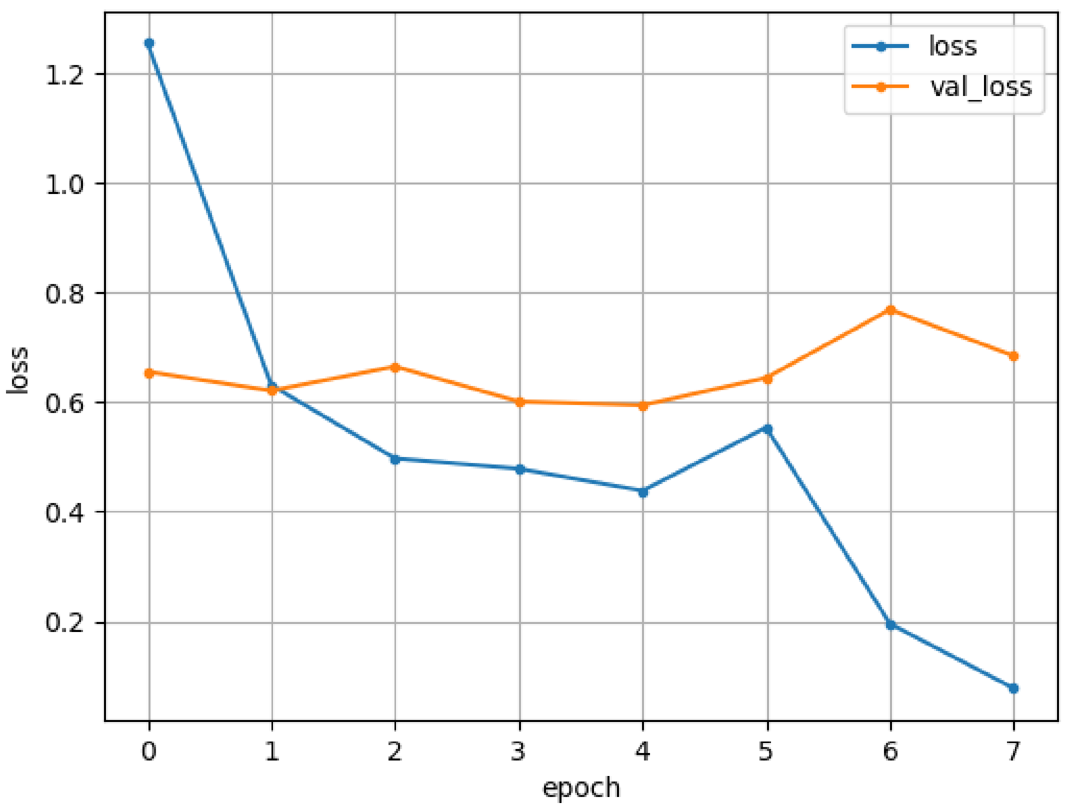


Figure 5.7: Loss Graph for first model

The first final model generated validation accuracy of 0.7141 (Mean) and 0.6871 (Median) after a couple of iterations.

This could be made even better. So we started training a second model, after a few trial and error, we came up with the final version with the same epoch of 5 in first stage and 20 in second stage. But in this model we changed the learning rate of the first stage to $5e^{-}3$ and the second stage to $5e^{-}4$. We increased the batch size to 64 and 32 respectively for the first and second stage.

The accuracy and the loss graph for the final model:

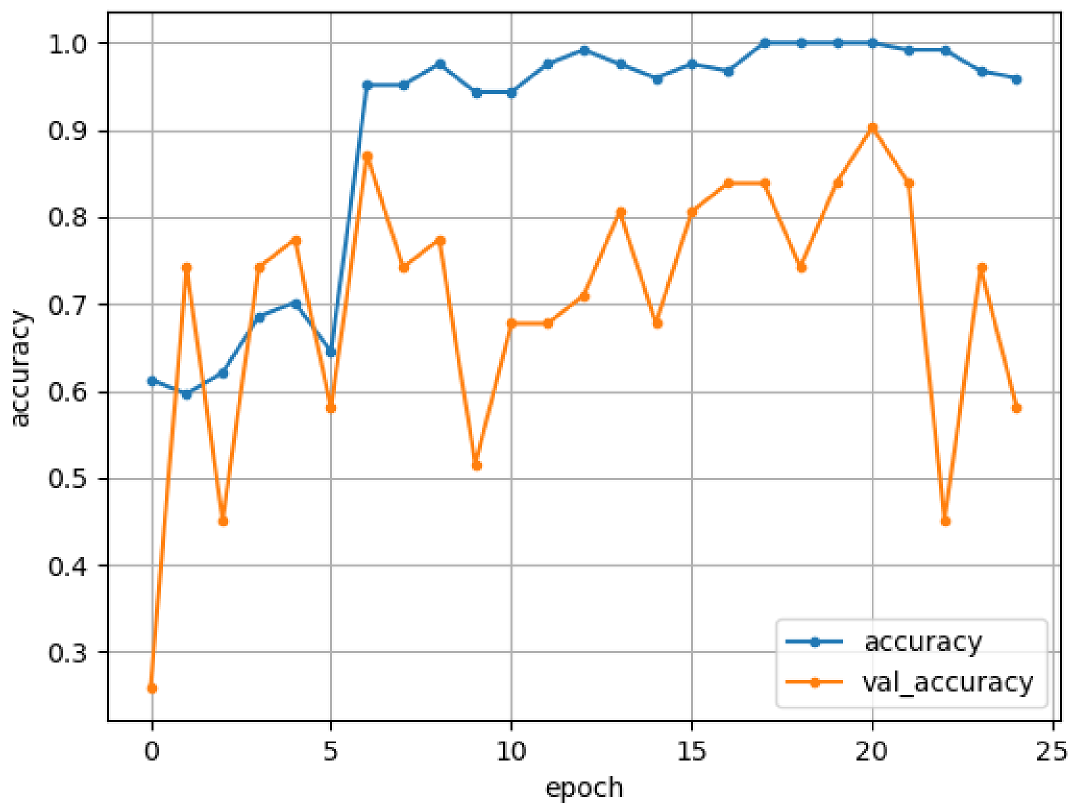


Figure 5.8: Accuracy Graph for second model

We tried a variant of different prediction method, out of all of them, first image prediction was the worst and mean and median of multiple images seemed to have the best validation accuracy.

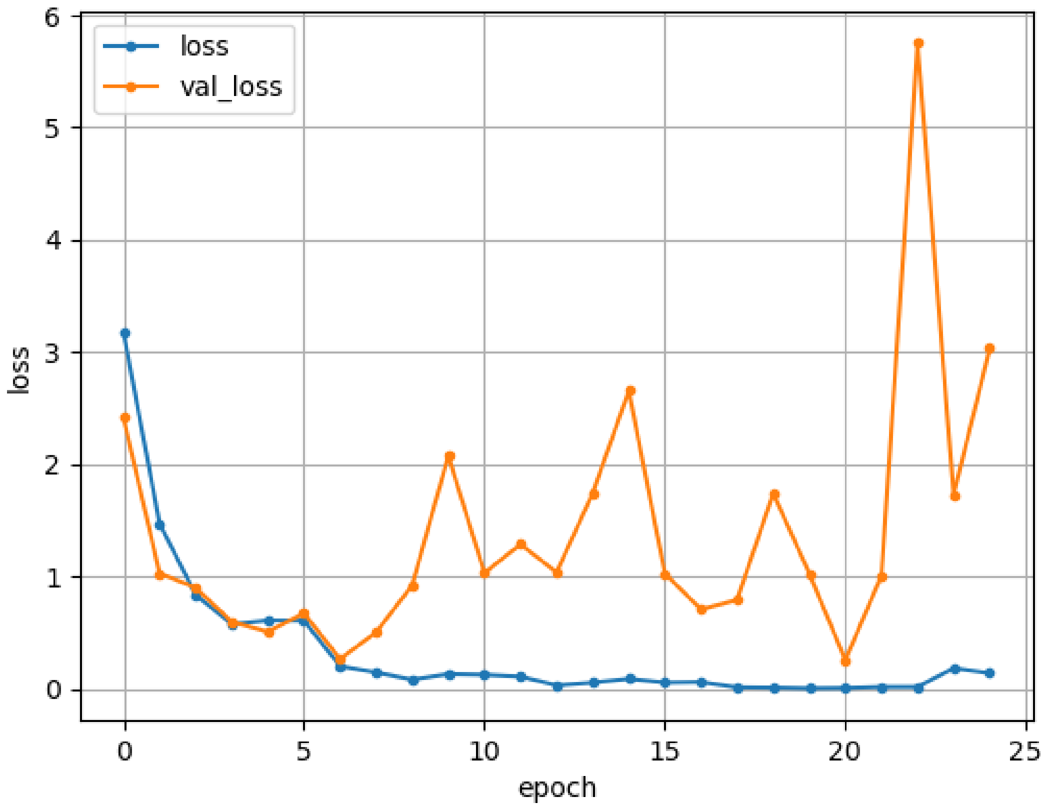


Figure 5.9: Loss Graph for second model

As we can see from Figure 5.8, at epoch 12 the validation accuracy rate of 0.8387 (Mean), and for median it was 0.8065, this was the best result we could achieve so far. After epoch 12 there was likely noise and the the model was overfitting.

Chapter 6

Result

We have attempted a number of times to optimize the results. We first started with a downscaled image of 150x150, and only the dense layer was modified. After training that model we did not achieve the best result, we had around 0.5 of accuracy. Later on we worked with the full 300x300 images with more modifications to the dense layer, SeperableConv and MaxPooling, which then got us the results of 0.8387 accuracy.

Testing our model also proved difficult, if we had access to a workstation CPU and GPU, we would evaluate every frame of the video we're trying to test. Since our testing machine was very slow, we had to resort to testing 5 frames out of 30 frames in a second of video. We also tried to see if a lower bit-rate video which has more artifacts would prove difficult to evaluate, which was true. We used ffmpeg to convert the video with a crf 40 flag, which is a very low bitrate video and tested them as well. The results are as follows:

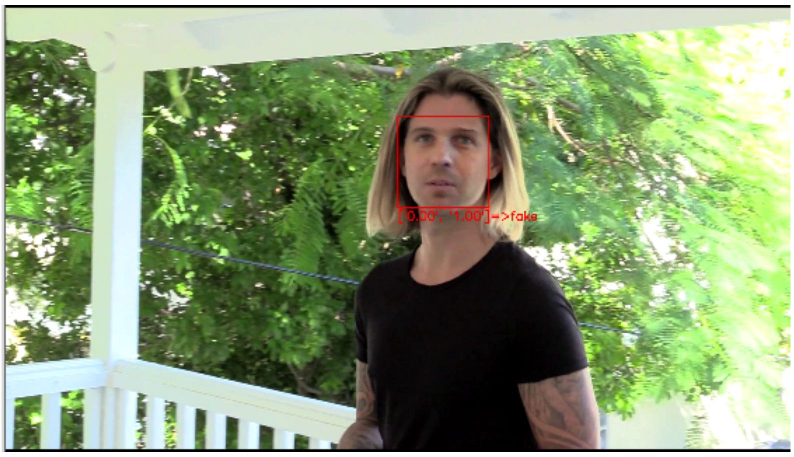


Figure 6.1: Detection in high light environment

The above figure shows a successful artifact detection in this particular video frame from a high quality video where the subject is outside with high light intensity.

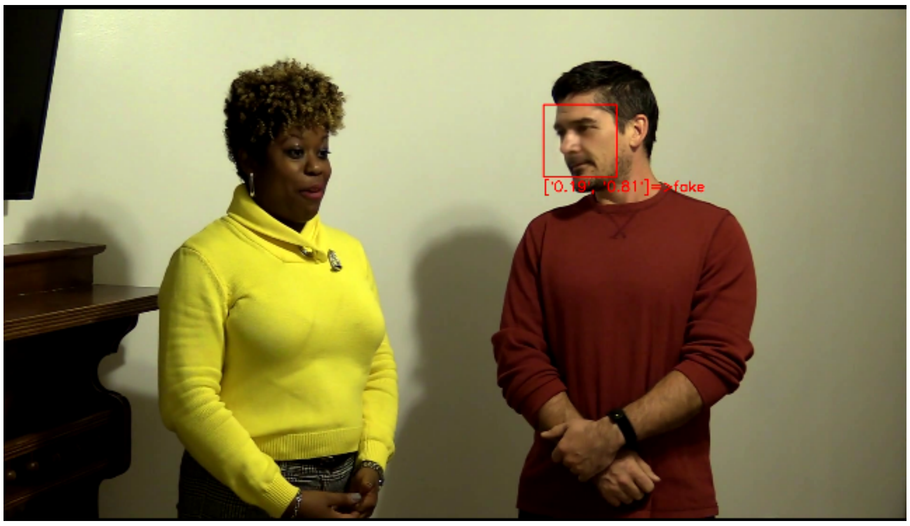


Figure 6.2: Detection with multiple subjects in frame

Here, we are unable to detect two faces but in this case it is successful in detecting the one face that has been swapped.

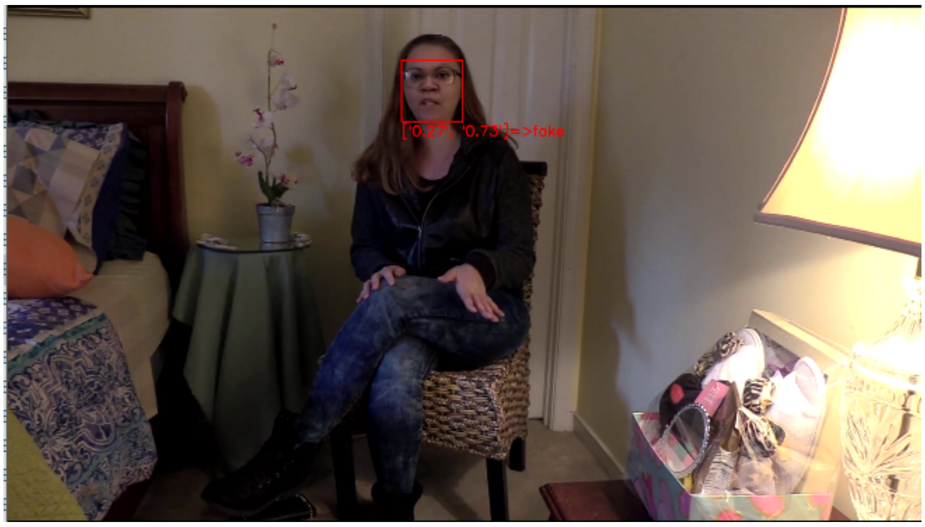


Figure 6.3: Detection with subject in low light setting

This particular video frame shows successful artifact detection from a high quality video where the subject is sitting back in a low light setting.

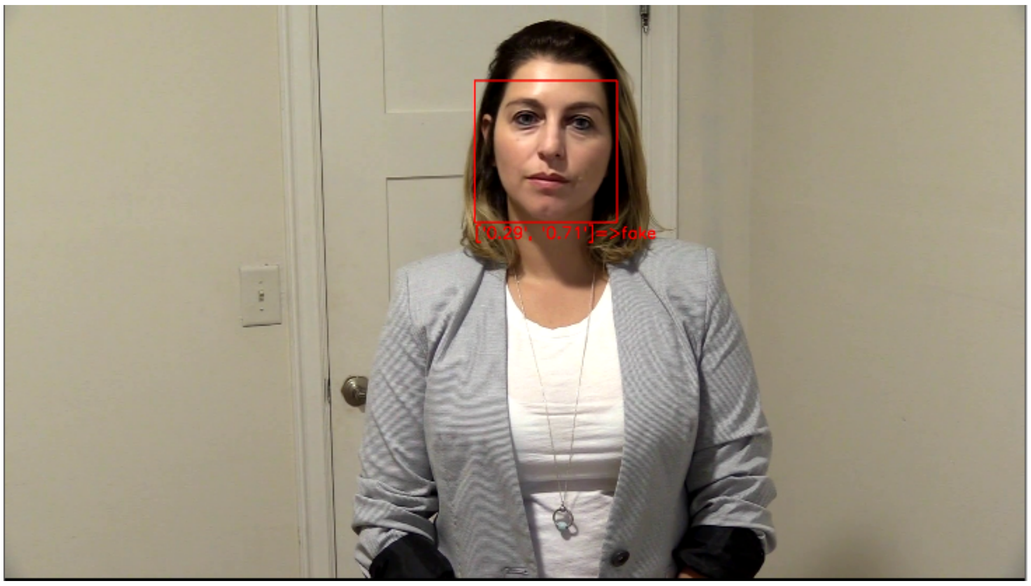


Figure 6.4: Detection in normal conditions

Figure 6.4 shows detection in a standard environment. Although detection rate is slightly low in low lighting but it detects well in standard lighting conditions where the face is clearly visible and subjects do not make rapid movements.

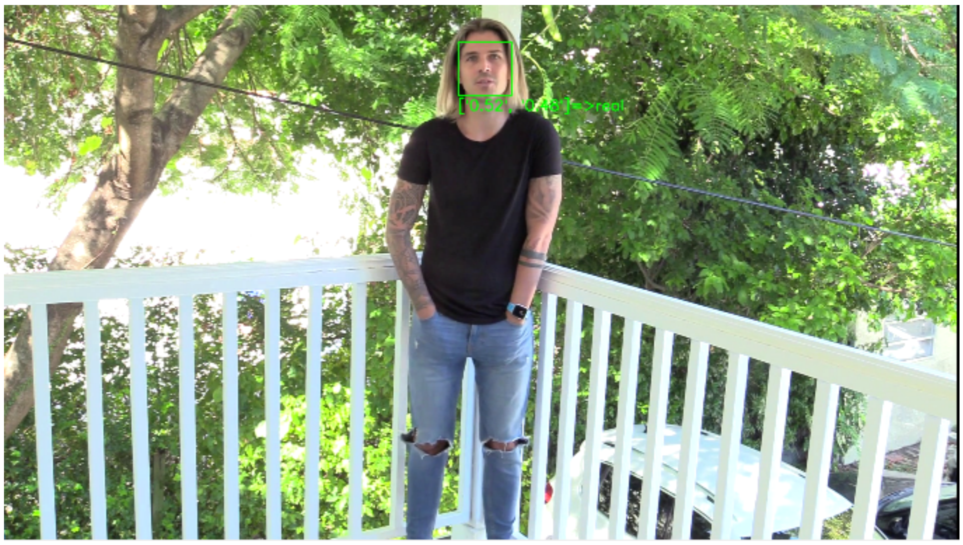


Figure 6.5: No artifact detection in strong lighting

The first detection is in an environment with very good lighting. Although it detects no artifacts in this frame but it detects artifacts in other frames where light directly falls on subjects face.

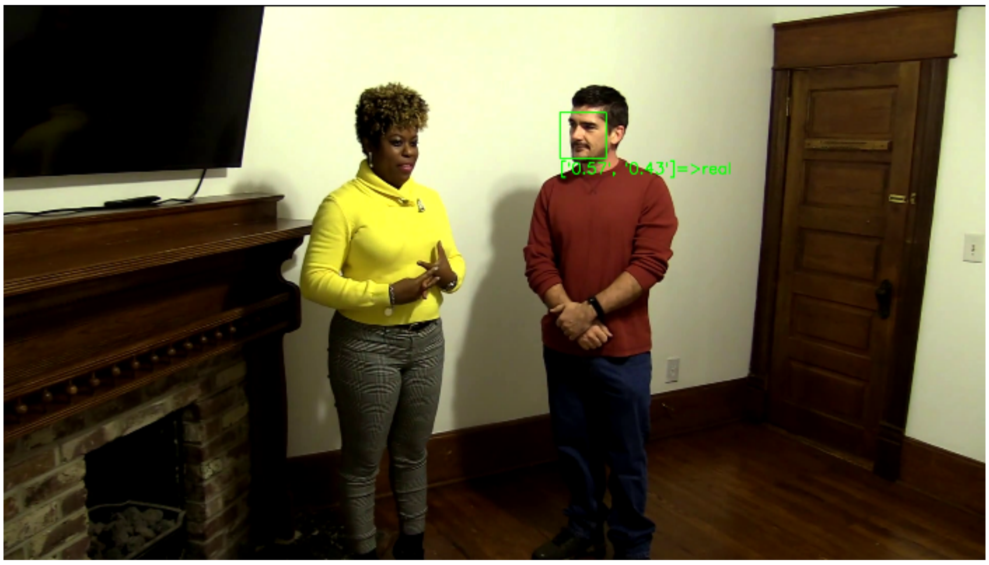


Figure 6.6: No detection with multiple subjects in frame

The second shows the detection of real faces when multiple subjects are present. Here, only one face is detected and it shows our inefficiency in detecting Deepfakes that will have multiple face swaps in a single frame.

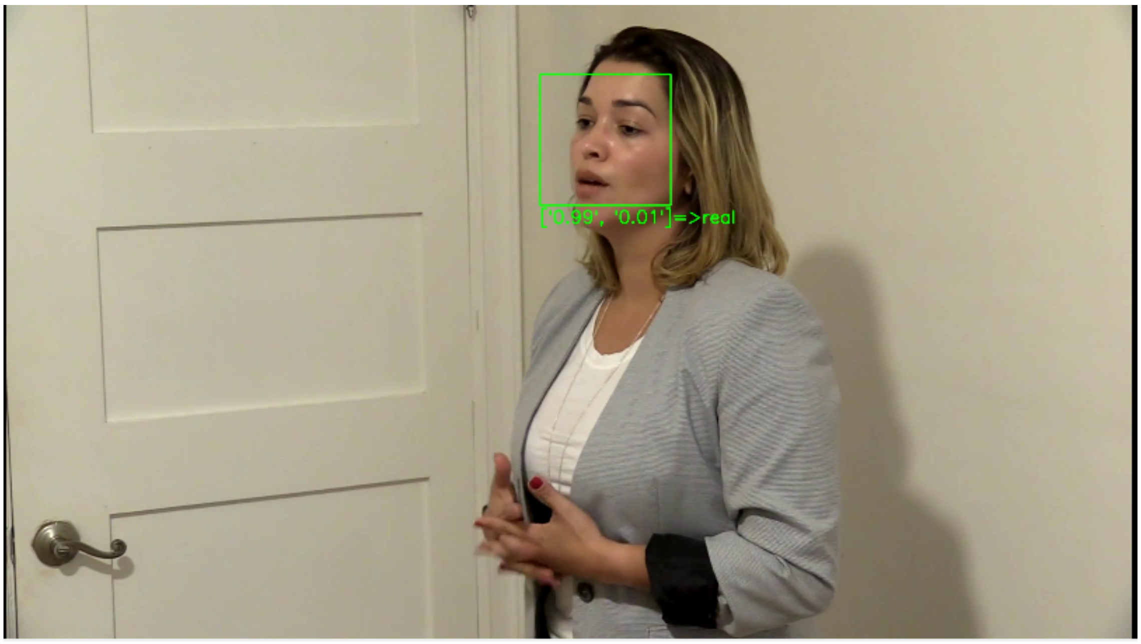


Figure 6.7: No detection in normal condition

The bottom figures show no artifact detection in standard lighting conditions. This type of videos with less movements and good lighting have the best detection rates.

Chapter 7

Comparison

7.1 Raw video vs. Encoded Video

As we tested our model against the raw video provided in the testing dataset, the results were pretty good. But one of the limitation could be when the tested videos are of very bad quality, to simulate this we encoded the video in ffmpeg with crf 40 flag, which results in a very blocky and video full of artifacts. The results are shown below:

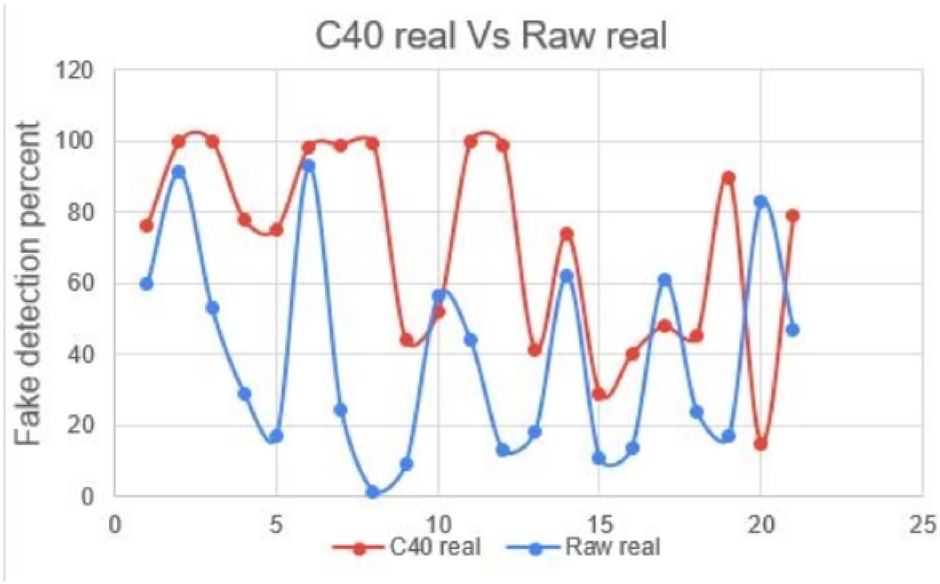


Figure 7.1: Raw video vs C40 video Score (Real Videos)

As we can clearly see, the compressed video clearly performs much worse than raw videos provided in the test set. But we still are expecting good results as it can be seen from the raw results, and the compressed videos are compressed to an extreme which usually is not very common at all.

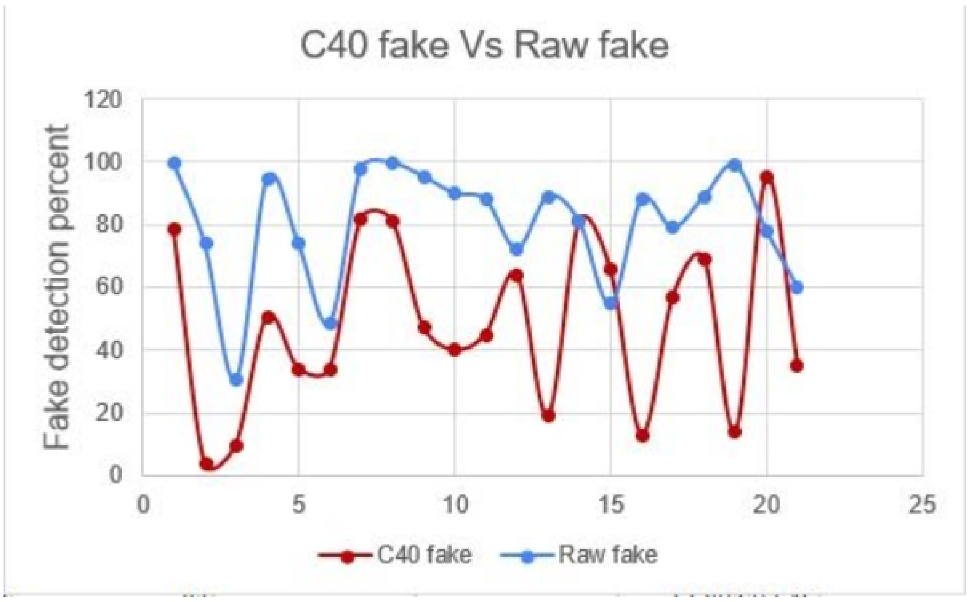


Figure 7.2: Raw video vs C40 video Score (Fake Videos)

7.2 Compare with Deepfake Detection from Inconsistent Head Poses

The SVM classifier achieves an AUROC of 0.89 in the proposed method for detecting deepfakes from head poses.[10] Our method in comparison is better as it uses more detailed features which are the face wrapping artifacts. Thus, the accuracy value of 0.8387 proves artifact detection using Xception architecture to have a similar result.

7.3 Compare with Detection by Optical Flow based CNN

They ran their proposed method on the FaceForensics dataset. From their available dataset they use 720 of the videos for training, 120 for validation and another 120 for testing. The result on VGG16 network yields 81.61 percent and ResNet50 yields 75.46 percent. [17] Again our method of artifact detection yields a higher percentage and thus better detection.

7.4 Compare with another Detection by Face Wrapping Artifacts

The paper that inspired our research still has a better result or similar result in comparison to our result. They have trained and validated the network using different datasets whereas we only focused on a single dataset. A better comparison is thus not possible as we performed the test on different datasets. In any case the ResNet50 network used in detecting face wrapping artifacts scores 97.4 on UADF dataset and scores an astounding 99.9 percent on DeepfakeTMIT dataset for low quality video files and 93.2 percent on high quality files. [19]

7.5 Limitations

Our model was trained on a single dataset with a low amount of batch sizes because of the lack of workstation GPU. Training on CPU was very time consuming even with lower batch size, as such we couldn't properly fine tune the model. We believe with even more data and access to faster hardware we can bring the results to 0.90 or more. And lastly for testing the videos, again for hardware constrains we used only a few frames from a typical 30fps video, resulting in a worse score for testing videos.

Chapter 8

Conclusion

As we live in an era of constant contact with social media sites, it is fairly easy for us to share and view Deepfake contents believing them to be true. As mentioned before, we are no longer able to judge the authenticity of media files with the help of our eyes alone. For these reasons' steps must be taken to ensure Deepfake content is filtered away and becomes easily identifiable. As technology advances even further, we are yet unsure of the capabilities a computing system may process by the next decade allowing further advances in creating and distributing fake contents like Deepfake will become even more constant. Thus, steps need to be taken to detect Deepfake and they have to be constantly updated as deepfake content gets more advanced. Any method chosen has to be made in a manner that it would be able to test videos for Deepfakes with utmost accuracy during content uploading and it has to be fast to ensure profitability in detection mechanism. In our case, Testing our model proved to be very difficult, if we had access to a workstation CPU and GPU, we would have evaluated every frame of the video we're trying to test. Since our testing machine was very slow, we had to resort to testing 5 frames out of 30 frames in a second of video. We also tried to see if a lower bit-rate video which has more artifacts would prove difficult to evaluate, which was true.

8.1 Future Plan

Our algorithm may provide a satisfactory result for some video files in evaluation but it is not efficient enough. Sometimes, it finds artifacts in videos without any manipulation done on them and we have yet not been able to test and train the model in a larger dataset with both low and high quality videos. We would like to change that in our future works where we would like to conduct the training of the network with a larger number of frames taken from a single video file and significantly increasing the batch size. We would also like to try and figure out a different approach to reduce the computation speed while detecting Deepfakes. The latest technology in Reface allows a 10 to 15 seconds Deepfake video in just about thirty second to a minute from a single photograph. But it takes about five minutes to determine legitimacy of a five second clip on a normal machine. As the speed and quality of Deepfakes increases, it should be the same for detection mechanisms to hold a value in fighting this unjust power of the AI systems. This process would require us to have a better object detection mechanism than our current algorithm to quickly detect faces and learn its features.

Bibliography

[1] J. M. Ronan Collobert Samy Bengio, “Torch: A modular machine learning software library,” Tech. Rep., Oct. 2002.

[2] M. G. Jan Lukáš Jessica Fridrich, “Digital camera identification from sensor pattern noise,” Tech. Rep., 2006.

[3] P. Goldsborough, “A tour of tensorflow,” Fakultät für Informatik Technische Universität München, Tech. Rep., Oct. 2016.

[4] N. Valigi, “Short history of the inception deep learning architecture,” Oct. 2016. [Online]. Available: https://nicolovaligi.com/history-inception-deep-learning-architecture.html.

[5] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” Google, Inc., Tech. Rep., 2017.

[6] K. Hao, “Deepfake-busting apps can spot even a single pixel out of place,” Nov. 2018. [Online]. Available: https://www.technologyreview.com/2018/11/01/139227/deepfake-busting-apps-can-spot-even-a-single-pixel-out-of-place/.

[7] Z. G. Marissa Koopman Andrea Macarulla Rodriguez, “Detection of deepfake video manipulation,” University of Amsterdam Netherlands Forensic Institute, Tech. Rep., 2018.

[8] L. Matsakis, “Artificial intelligence is now fighting fake porn,” Feb. 2018. [Online]. Available: https://www.wired.com/story/gfycat-artificial-intelligence-deepfakes/.

[9] C.-F. Wang, “A basic introduction to separable convolutions,” Aug. 2018. [Online]. Available: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728.

[10] S. L. Xin Yang Yuezun Li, “Exposing deep fakes using inconsistent head poses,” University at Albany, State University of New York, University at Albany, State University of New York, USA, Tech. Rep., Nov. 2018.

[11] S. L. Yuezun Li Ming-Ching Chang, “Exposing ai generated fake face videos by detecting eye blinking,” University at Albany, Tech. Rep., 2018.

[12] E. Brown, “Half of americans do not believe deepfake news could target them online,” Nov. 2019. [Online]. Available: https://www.zdnet.com/article/half-of-americans-do-not-believe-deepfake-news-could-target-them-online/.

[13] M. Fabien, “A guide to face detection in python,” Apr. 2019. [Online]. Available: https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1.

[14] ——, “Xception model and depthwise separable convolutions,” Mar. 2019. [Online]. Available: https://maelfabien.github.io/deeplearning/xception/.

[15] K. S. Haya R. Hasan, “Combating deepfake videos using blockchain and smart contracts,” Tech. Rep., 2019.

[16] J. Y. Huy H. Nguyen Fuming Fang and I. Echizen, “Multi-task learning for detecting and segmenting manipulated facial images and videos,” SOKENDAI (The Graduate University for Advanced Studies), National Institute of Informatics, The University of Edinburgh, SOKENDAI (The Graduate University for Advanced Studies), Kanagawa, Japan, National Institute of Informatics, Tokyo, Japan, The University of Edinburgh, Edinburgh, UK, Tech. Rep., Jun. 2019.

[17] A. D. B. Irene Amerini Leonardo Galteri and R. Caldelli, “Deepfake video detection through optical flow based cnn,” Tech. Rep., 2019.

[18] D. Samson, “Number of deepfake videos online rises 84 percent in less than a year,” Oct. 2019. [Online]. Available: https://www.techtimes.com/articles/245628/20191009/number-of-deepfake-videos-online-rises-84-percent-in-less-than-a-year.htm.

[19] S. L. Yuezun Li, “Exposing deepfake videos by detecting face warping artifacts,” University at Albany, State University of New York, University at Albany, State University of New York, USA, Tech. Rep., May 2019.

[20] C.-Y. L. Chih-Chung Hsu Yi-Xiu Zhuang, “Deep fake image detection based on pairwise learning,” National Pingtung University of Science and Technology, Department of Management Information System, National Pingtung University of Science and Technology,1, Shuefu Road, Neipu, Pingtung 91201, Taiwan, Tech. Rep., Jan. 2020.

[21] “Convolutional neural networks,” Jun. 2020. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[22] H. Davis, “How deepfake technology actually works,” 2020. [Online]. Available: https://screenrant.com/deepfake-videos-explained-how/.

[23] B. Dickson, “What are convolutional neural networks (cnn)?,” Jan. 2020. [Online]. Available: https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/#:%7E:text=Convolutional%20neural%20networks%2C%20also%20called,a%20postdoctoral%20computer%20science%20researcher.&text=The%20early%20version%20of%20CNNs,)%2C%20could%20recognize%20handwritten%20digits..

[24] S. Greengard, “Will deepfakes do deep damage?,” Jan. 2020. [Online]. Available: https://cacm.acm.org/magazines/2020/1/241708-will-deepfakes-do-deep-damage/fulltext.

[25] “Keras vs opencv – differences between opencv and keras,” Jul. 2020. [Online]. Available: https://data-flair.training/blogs/keras-vs-opencv/.

[26] D. Scott, “Deepfake porn nearly ruined my life,” Feb. 2020. [Online]. Available: https://www.elle.com/uk/life-and-culture/a30748079/deepfake-porn/.

[27] C. Willen, “Kristen bell says she was 'shocked' to learn that her face was used in a pornographic deepfake video,” Jun. 2020. [Online]. Available: https://www.insider.com/kristen-bell-face-pornographic-deepfake-video-response-2020-6.

[28] “Keras resnet: Building, training scaling residual nets on keras,” [Online]. Available: https://missinglink.ai/guides/keras/keras-resnet-building-training-scaling-residual-nets-keras/#:%7E:text=Residual%20Network%20(ResNet)%20is%20a,or%20thousands%20of%20convolutional%20layers.&text=ResNet%20stacks%20up%20identity%20mappings,the%20activations%20from%20previous%20layers..