Detection of DeepFake using Computer Vision and Deep Learning

by

Anisur Rahman
19101631
Ehteshamul Islam Uschash
19101343
Faria Rahman
22241142
Shihaba Jamal Chowdhury Adiba
19101629
Tahmidul Labib
22241136

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
May 2023

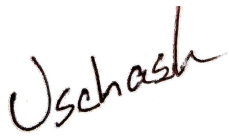
Declaration

It is hereby declared that

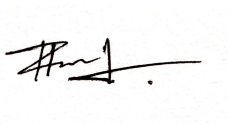
1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material 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:

Anisur Rahman
19101631



Ehteshamul Islam Uschash
19101343



Faria Rahman
22241142

Shihaba Jamal Chowdhury Adiba
19101629

Tahmidul Labib
22241136

Approval

The thesis titled "Detection of DeepFake using Computer Vision and Deep Learning submitted by

1. Anisur Rahman(19101631)
2. Ehteshamul Islam Uschash(19101343)
3. Faria Rahman(22241142)
4. Shihaba Jamal Chowdhury Adiba(19101629)
5. Tahmidul Labib(22241136)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May, 2023.

Examining Committee:

Supervisor:
(Member)

Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In this era of Metaverse and technological advancement, DeepFakes are one of the most alarming concepts. DeepFakes are mostly synthetically-generated manipulated photos or videos which is created swapping a face of a person using Deep learning, Generated Adversarial Network (GAN), autoencoder-decoder pairing structure. There are several other Deepfaking tools such as; FaceSwap, DeepFaceLab, DFaker, DeepFake-tensorflow etc. Using Generative Adversarial Network (GAN), DeepFakes have become smoother and more realistic in making the fake videos. DeepFakes can become concerning if it is used for political purpose, committing fraud, spreading misinformation, pornography, defamation on social media etc. This possesses a security threat on people's lives knowingly or unknowingly. As a result, it is visible that DeepFakes can be very distressing on the wrong hand if not detected properly. Our purpose is to detect the DeepFake videos as successfully as possible. We want to focus on detection using Deep learning approaches also using Image Recognition and Computer Vision. For the detection, we used a dataset of videos, which included both real and fake videos. We have successfully extracted it from Kaggle where we have found dataset of more than 2352 videos from DeepFake Detection Challenge(DFDC) and FaceForensics++. To detect the fake videos, we followed the method of employing temporal feature and exploring visual artifacts within frames. Employing temporal feature uses LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) whereas visual artifacts within frames mostly employs deep learning method to detect DeepFakes. We ensembled LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) to detect DeepFakes successfully. ResNeXt101_32x8d have been used to extract features and a custom CNN model is added with LSTM for better accuracy for detecting DeepFake. The accuracy of the model was 94.05%. After further improvement and with the introduction of learning rate schedulers, we were able to achieve better accuracy. We have used CosineAnnealingLR, CyclicLR, MultiStepLR and ReduceLROnPlateau as learning rate scheduler among which MultiStepLR gave the highest accuracy of 95.33%.

Keywords: DeepFake, CNN, LSTM, Learning Rate Scheduler

Acknowledgement

We would like to express our deepest gratitude to the Almighty for guiding and blessing us throughout the entire journey of our thesis, enabling its successful and fulfilling completion.

We would like to extend our heartfelt appreciation to our supervisor, Dewan Ziaul Karim Sir, for his unwavering guidance and support throughout research, which has been invaluable to the successful completion of this thesis.

Finally, we are grateful to our parents, whose encouragement and prayers has been crucial towards our work.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are living in an era of information and technological advancement. Day by day, we are getting introduced to more advanced Artificial Intelligence technologies which have made our lives easier and it has enabled endless possibilities to open new doors of advancement for us. The usage of CGI, VFX are most commonly used computer-generated graphics we have come across. The first usage of CGI was in 1973 in the Hollywood movie 'Westworld'. These computer-generated graphics have always been the talk of the town because, since 1973, it is getting improved day by day. It is always fascinating to see such rich usage of CGI, and VFX in the field of entertainment, sports, video games, etc. The recent headline is DeepFake videos which use Deep Learning supported by Artificial Intelligence. With the improvement of DeepFake generating, fake videos are becoming harder and harder to detect with the naked eye and quite impossible to detect without any help from any generated model. Deep neural networks (DNNs), in particular, are AI algorithms that can produce falsified films. This is a recent development in the problematic issue of internet misinformation. Despite the fact that digital image and video manipulation and fabrication are old concepts, the recent growth of DNN have made it much simpler and faster production of persuasive fake videos. In 2017, when an account on Reddit going by the name of "moniker" started posting fake pornographic videos created with the help of DNN-based face-swapping algorithm, DNN-generated fake videos first came to the notice of the general public. The term "DeepFake" has since been applied more generally to any AI-generated videos that impersonate other videos[1].

DeepFakes targets social media platform where rumors, conspiracies, and false information can spread quickly because users often follow the masses. In addition, a persistent "Infopocalypse" makes people believe they can only trust information that confirms what they already believe and originates from their social media, such as family, close friends, or relatives. In fact, even if they think it might be false, many individuals are open to everything that supports their preexisting beliefs. Cheap hardware, such as Effective Graphical Processing Units, are generally accessible, and as a result, low-quality films with minimally altered genuine content are already pervasive. More and more open source software is available for creating realistic, high-quality DeepFakes for misinformation. This makes it possible for users to almost flawlessly edit films, change expressions, swap faces, and synthesize speech with little to no artistic training.

DeepFake advancements has raised significant concerns regarding its potential malicious exploitation by child predators, enabling them to adopt alternative identities by utilizing live-stream videos to convincingly transform an adult's face into that of a child. Last but not least, DeepFakes can help spread malicious scripts. Researchers recently discovered a website dedicated to DeepFakes was using its customers' Computers to mine bitcoin. DeepFake enthusiasts might thus be the target of "Cryptojacking" as they are most likely to have powerful machines.[2]

Chapter 2

Research

2.1 Research Problem

Differentiating DeepFake with the naked eyes are one of the hardest tasks. Because DeepFake technology is getting smart day by day and making DeepFakes are getting easier. In the beginning, people had to have a deep knowledge about AI, Deep learning, etc. to make a DeepFake video. But recently, with the emerging technology of GAN and Deep Neural Network, anyone on the internet can make a DeepFake video without even having knowledge about it. With a simple video and face swapping technology, AI automatically generates the required DeepFake video. These DeepFake videos can cause serious problems in many aspects, such as political agenda, the social life of a particular people, political organization, etc. DeepFake videos can also be generated in a pornographic way to harm someone's reputation and generating this kind of videos can cause serious harm in someone's whole life which they are not even responsible for. These high-quality DeepFakes can be very deceiving to someone who does not have knowledge about DeepFake, AI, or other components needed to make DeepFake video. The problem generally upraised by DeepFakes are defamation, political conspiracy, etc. DeepFake could be a valid reason for weakening democracy. When a powerful national leader threatens another country or provides hate speech about others through a video, the opposition might have no clue if the video is real or fake. The other country can take necessary steps so fast without verifying. This type of activity can be the reason for damage to a whole nation. This can also mislead voters in various ways. At a point, people will lose interest in democratic politics, which is called Reality Apathy. Democracy do not work if the general people cannot believe it. At the same time, an economic crisis can also happen if the videos cannot be verified. The export-import between countries could shut down due to an unintentional fake threat. This can lead to a massive price hike due to shortage of commodities. Defaming celebrities or political leaders can also be a reason of DeepFake. It can also happen for normal people too. Even in this time, people are already paying to make a DeepFake video of someone they want to blackmail or threaten. Women are in a very threatening position. In future, it could be a very common issue. Gradually, this will reduce the trust issue in people. DeepFakes are likely to hinder digital literacy and citizens' trust in official information because fake videos of officials from the government claiming things that never happened cause people to question authorities. In fact, AI-generated spam, false news that is based on discriminatory content, bogus movies, and a slew

of conspiracy theories are all affecting people more and more these days. The "Information Apocalypse" or "Reality Apathy" phenomena is a result of people believing that much information, especially video, cannot be trusted. This is perhaps the most harmful element of DeepFakes, but it may not be misinformation per SE. Furthermore, due to their ingrained belief that everything they do not want to accept must be false, people may even dismiss real footage as fake. In other words, rather than people being tricked, the biggest worry is that they will start to see everything as fraud.
DeepFakes also represent a threat to Cybersecurity. The corporate sector has already demonstrated interest in defending themselves against viral scams since DeepFakes might be used to manipulate markets and stocks, for instance, portraying a CEO using racist or gender insults, or proclaiming a phony merger, falsely reporting financial losses or declaring bankruptcy, or representing them as having committed a crime. Additionally, bogus product or porn announcements could be utilized to harm brands, demand money, or humiliate management. DeepFake technology furthermore enables real-time digital impersonation of an executive, for example, while requesting a worker to make a swift money transfer or reveal critical information.[2]

2.2 Research Objective

The major goal of this study is to detect DeepFake videos as accurately as possible. The advancement of technology and AI is making DeepFakes very hard to detect with the naked eye. As a result, with the help of a combined model of CNN and LSTM, it is going to be able to detect if the video is DeepFaked or not. This will help to eradicate the confusion of people that the video they have seen of a personality that might contradict with their actual personality. This will be able to resolve any political issue that was aroused by a DeepFake video.

Chapter 3

Literature Review

3.1 Deepfake video creation

With the emergence of the pre-trained models and Artificial Intelligence, GAN, DeepFakes are becoming very easier through which swapping faces for various heinous crime are getting easier and easier every day. To emphasize the necessity of detection of DeepFake, in paper[3], Pavel Korshunov and Sébastien Marcel created DeepFakes based on GAN which was the open-source software, and preprocessed the whole dataset(VidTIMIT dataset). By using two face recognition models named VGC and FaceNet which is a deep neural network, they failed to achieve success to detect DeepFake. Both the VGC and FaceNet models wrongfully accepted the DeepFake videos in a major portion resulting in 85.62% for VGC and 95.0% for FaceNet.

To generate a DeepFake video, Generative Adversarial Network(GAN) plays a crucial part[4]. DeepFakes are mainly dependent on GAN as swapping faces is a necessity for creating a DeepFake video. GAN-based technologies are getting more and more attention day by day for creating realistic images and videos significantly hard for the human eye to catch.

Face swapping and Face reenactment are the two most popular techniques for generating DeepFake videos. There are several ways to generate by Face swapping. Convolutional Neural Network (CNN) was used by Iryna Korshunova, Wenzhe Shi et al[5], where their neural network uses multi layer architecture which has different branches embedded with convolutions. They also used linear rectification in the branches. Another technique for creating DeepFakes by face swap is proposed by Kyle Olszewski, Zemo li et al[6] in which they trained the Generative Network and made a 3D model of the face by using RGB image. The training method is required to make the face more realistic and dynamic. DeepFaceLab is another method which is a framework described in paper [7] by Ivan Petrov, Daiheng Gao et al, where they have announced DeepFaceLab as a dominant framework for Face Swap. Finally, Kevin Dale, Kaylan Sunkavalli et al, proposed Deep Generative model [8] which swaps faces by using a multilinear model that puts a trail on the facial expression.

Another process of DeepFakes is Face Reenactment. Some popular method includes Face2Face. Proposed by Justus Theis, Michael et al in [9], Face2Face is categorized as a novel approach which recognizes the target face and provides output based on a

source face by animating the target video and its facial expressions. In paper [10] by the same authors of Face2Face proposal, initiated another method FaceVR, same as Face2Face, a novel approach, altering the expression and facial emblem by Virtual Reality based Face Reenactment. Furthermore, HeadOn [11] method follows Face Reenactment compiling Real time DeepFake videos which tracks not only the facial expressions but also the movement of eyes, head motion etc.

DeepFake currently poses a greater threat than we can ever imagine[2][12]. In many industries, politics, journalism, etc. DeepFake can cause various trust issues in the mind of general people because DeepFake can be harder to spot and people who have a limited idea about the threats of DeepFake can easily be deceived on social media and other platforms will easily believe the video they see about a celebrity and political leader. So detecting DeepFake videos is a necessity more than ever.

3.2 DeepFake video detection using CNN

There are various ways of detecting DeepFake using the Convolutional Neural Network. In paper[13], Aditi Kohli et al, used Frequency CNN to detect the facial forgeries. Convolutional Neural Networks, or CNNs, were designed to map image or video data to an output variable. The spread of willing miscommunication through synthetically generated fake but realistic images and videos has become a threatening and significant problem in this modern world. Convolutional models are a class of deep learning which have been proven to be effective in all aspects. This tool (CNN) is pretty effective for analyzing pattern recognition and computer vision related problems. It has multiple layers containing convolutional layer, activation layer, an optional pooling layer followed by fully connected layer. So, basically it contains 8 learned layers where 5 of which are convolutional layers & rest 3 are linear or fully connected. Convolutional and fully linked layers are simply stacked on top of one another to create CNN architecture. CNN is stack based architecture so the deeper this kind of architecture becomes the more inaccurate it gets. This phenomenon mainly occurs due to the conventional backpropagation way CNNs are trained. During the CNN training phase, gradient information should be delivered reversed into the model to avoid it being somewhat attenuated as it passes through each layer of CNN. For CNNs with fewer layers, it is not a problem, but for CNNs with more layers, the gradient signal effectively degrades to noise by the time it reaches the network's first layer once more. Then, by adopting a fundamental strategy and broadly generalizing the notion of a direct link between layers, ResNet and DenseNet largely solve the issue of disappearing gradients. The usual input processing steps for the CNN architecture include kernel, stride, padding, and pooling. Where the second, fourth, and fifth convolutional layers' kernels are only connected to the kernel maps of the layers that came before them if they are on the same GPU. Additionally, the first convolutional layer applies 96 kernels and a 4-pixel stride to filter the input image, while the second layer applies 256 kernels to filter the output of the first layer. There are no pooling or normalizing levels between the following three layers. They are simply connected to one another. The third layer has 384 kernels and the fourth layer has 384 kernels. The third layer is 3x3x256 and the fourth layer is 3x3x192. The fifth layer has 256 kernels and is once more 3x3x192 in

size[14][15]. The accuracy score of CNN based models for detecting DeepFake videos is truly remarkable. Davide Cozzolino et al [16] published their research on various CNN-based models performed by various authors. Some of the models mentioned in the papers based on Convolutional Neural Network are MesoInception-4 which has an accuracy of 83.10% for High quality data and 70.47% for Low quality data. XceptionNet is another model mentioned in the paper which has better accuracy than MesoIncetion-4 and for High quality data the accuracy score is 95.73% and for low data the accuracy score is 81.00%.

Deep learning-based video modification techniques have become extensively available to the general public in recent years. With little to no work, anyone can learn how to make deepfake movies using a small number of victims or target images. This presents a significant societal problem for everyone who have photographs which are accessible to the public on the Internet, namely on social networking platforms. [17]

Because the spread of misleading information through convincingly realistic-looking but artificially produced images and videos has become a critical problem, effective manipulation detection systems are needed. Although discovering changed faces in still photos has received a lot of attention, finding altered faces in videos using the temporal information in the stream has received less attention. Recurrent convolutional models, a subtype of deep learning models, are particularly good at exploiting the temporal characteristics from picture streams in a number of applications. [18]

3.3 DeepFake video detection using Hybrid CNN and LSTM

Since LSTM networks are a subtype of RNNs, the knowledge learned from studying RNNs also applies to them because they are simpler systems. The standard RNN equations, which we obtain from differential equations, are significant because they serve as the initial model that specifies a clear logical path leading to the LSTM system design[19]. Though nowadays majority of the deepfake detection techniques heavily depends on the machine learning methods as it mainly generalises the process of identifying certain phenomena[20].

Saikia et al.[21] has introduced a hybrid model of CNN and LSTM using VGG16 where they specifically targeted on the facial features of the deepfake videos to identify the distinguishing traits of the content. Additionally, their proposed method mainly focuses on analyzing facial characteristics as the process of warping introduces the noticeable distortion in the DeepFake videos. Furthermore, this approach involves utilizing this information by incorporating several preprocessing steps such as face extraction,frame extraction and optical flow field feature extraction followed by hybrid CNN-RNN modelling. As a result, they achieved 91.21% accuracy on FF++, 79.49% on Celeb-DF and 66.76% on DFDC. Cornia et al.[22] proposed Saliency Attentive model, a deep learning model which mainly focuses on the crucial parts of the input data consisting of extraction network and attention mechanism which assigns weights to point up salient regions. Moreover, it mainly improves

the performance and helps the model prioritize applicable information in sentiment analysis and object recognition.

Shivangi Aneja et al.[23] published their research on a new transfer learning approach, Deep Distribution Transfer(DDT) which mainly works on detecting facial forgery. Additionally, they set the goal of enabling a trained model on a single fake generation method to be adapted well to different datasets and prior unknown manipulation techniques. As a result, they introduced a mixture of model-based loss formulation which learns a multi-model distribution that encodes the initial forgery methods class categories. They also suggested a approach of spatial mix-up augmentation which helped them to enhance cross-domain generalization and finally enabled them to reach the accuracy of 92.23% on FF++, 81.21% on Google DFD, 60.79% on AIF, 74.28% on Dessa, 68.83% on Celeb-DF and 75.47% on combined. On another research done by Ranjan et al [24] proposed similar combined method of CNN and LSTM using XceptionNet reached the accuracy of 94.33% on DFD, 83.49% on Celeb-DF, 78.13% on DFDC and 79.62% on combined dataset. Moreover, as part of their research they mainly examined the effectiveness of CNN framework that utilizes Transfer Learning which helped visualizing intermediate activations for accessibility. Additionally, on their research the shabby performance of the separately trained models on various levels mainly highlights the problematic phenomenon of dataset shift in numerous testing environments. This finding suggests that more diverse generating procedure and pre-processing techniques are required while compiling datasets to capture real world circumstances. Also, by highlighting the disparities between the edited face and its surrounding areas are also a better data processing approach which was proposed by Chen et al [25]. They introduced the unified framework called FSSpotter which mainly explored temporal and spatial information in videos which has Spatial Feature Extractor (SFE) and Temporal Feature Aggregator (TFA) that mainly detects spatial evidence and temporal inconsistencies across frames. In terms of AUC (Area Under Curve) scores in detecting DeepFakes their suggested approach has shown greater performance.

Chapter 4

Workplan

First, we have made a dataset of 2352 videos which are 50% real and 50% deepfaked videos. We preferred two datasets. FaceForensic++ which has a total of 500 videos and DeepFake Detection Challenge Dataset of 1852 videos. All of the videos will be preprocessed by splitting the videos into frame and then detecting the face clearly on the videos. As we need to detect the face and track the expression and inconsistency of the face, we will crop the face on preprocessing stage. Furthermore, we augmented the data using Random Horizontal Flip, Color Jitter and then finally converted the frames into tensor. After normalizing the tensor, the features are extracted using ResNeXt101_32x8d. ResNeXt101_32x8d is helping to extract features/patterns which are considered meaningful. With the help of CNN layer, the model refines and transform the extracted features before sending the output to LSTM layer for further processing and classification. We have added LSTM to detect the temporal inconsistency between the frames which might indicate the abnormality thus detecting a DeepFake video. It will be done with Temporal sequence analysis. If consistency is found between the frames, it will be detected as Fake video.

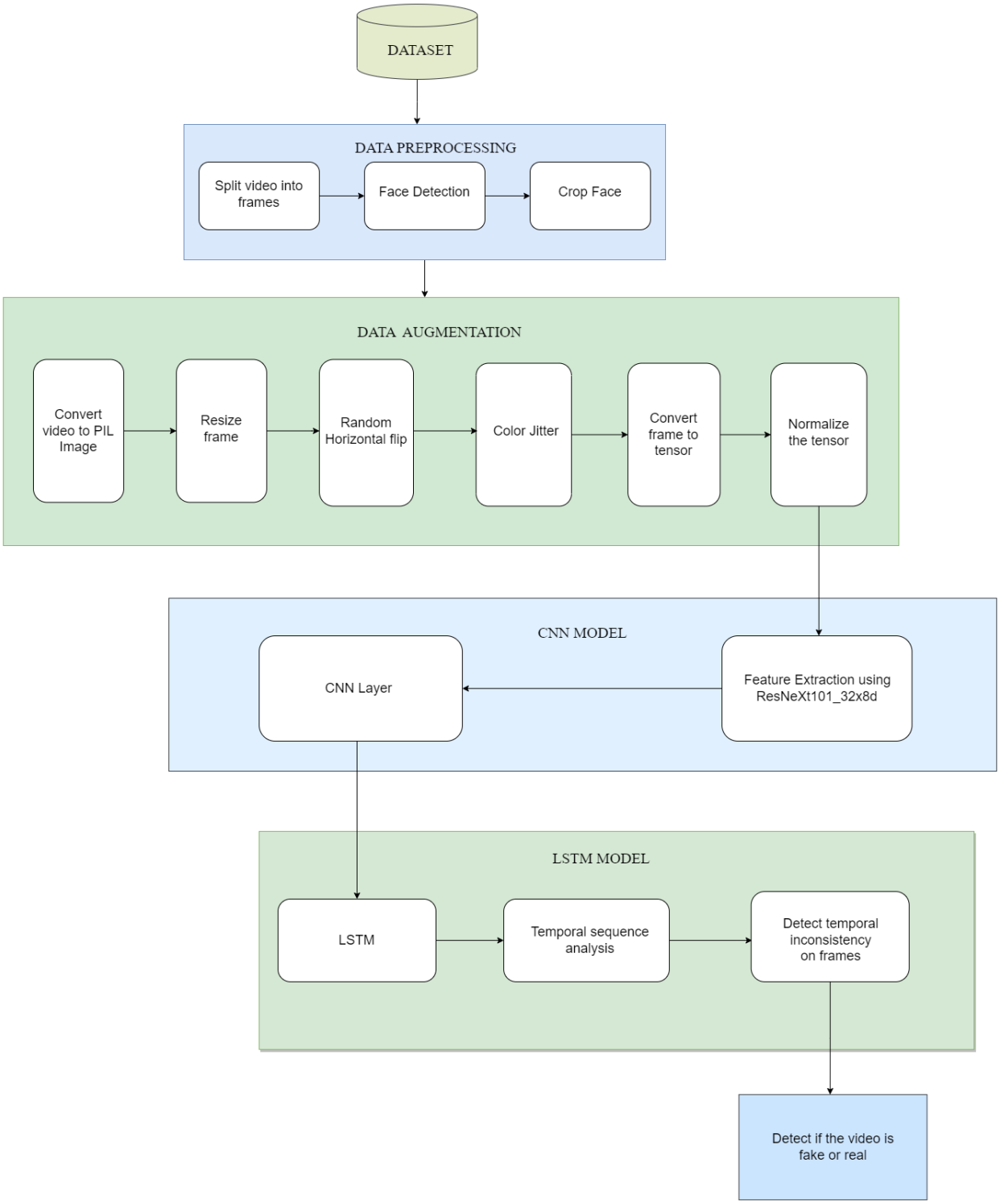


Figure 4.1: WorkFlow diagram

Chapter 5

Description of the data

5.1 Dataset and Data analysis:

5.1.1 Video Data:

For DeepFake videos, we have made a custom dataset consisting of 2352 videos from DFDC(DeepFake Detection Challenge) dataset and FaceForensics++. The quantity of videos taken from DFDC are 1852 and from FaceForensics++ we have taken 500 videos. To prevent an uneven dataset and skewed results, real videos and false videos are equal. From DFDC, 1000 Real videos and 852 Fake videos have been taken. From FaceForensics++ consisting of Face2Face, FaceSwap and NaturalTextures Deepfakes, 176 Real videos and 324 Fake videos have been enacted. The FaceForensics++ dataset was biased and consisted mostly fake videos. The real videos from the dataset are low quality. All the videos from DFDC dataset are of 10 seconds and contain on average 300 frames at 30FPS. So, the whole dataset from DFDC consists of 555,600 frames. FaceForensics++ videos have 300 frames at 30 FPS with 10 seconds. The total number of frames in FaceForensics++ are 150,000.

5.1.2 MetaData:

The MetaData.csv file contains the video information which is either the video is real or fake. The csv file helps to train the model and validate the data.

5.1.3 Data Preprocessing

The total number of videos in this dataset are 2352 and a huge number of frames(705,600) will be very difficult for the model. We have to split the videos into frames so that the model can detect inconsistency among the frames of faces.

Split into frames:

As there are different numbers of frames in the videos and an enormous amount of frames will overfit the model, the quantity of frames will be 200 for all the videos. The resolution of the frames will be 112x112. For detecting the inconsistency, we have used Long Short Term Memory(LSTM) as a result, for better accuracy of the LSTM model, the first 200 frames of the videos are split.

Face crop:

For the cropping the face, we have used built in $face\_recognition\_models{-}0.3.0.tar.gz$. The videos have a surrounding and might have multiple faces. The Face Recognition model crops the face as a result, there are 200 frames in the video of just facial images to detect inconsistencies by the LSTM layer.

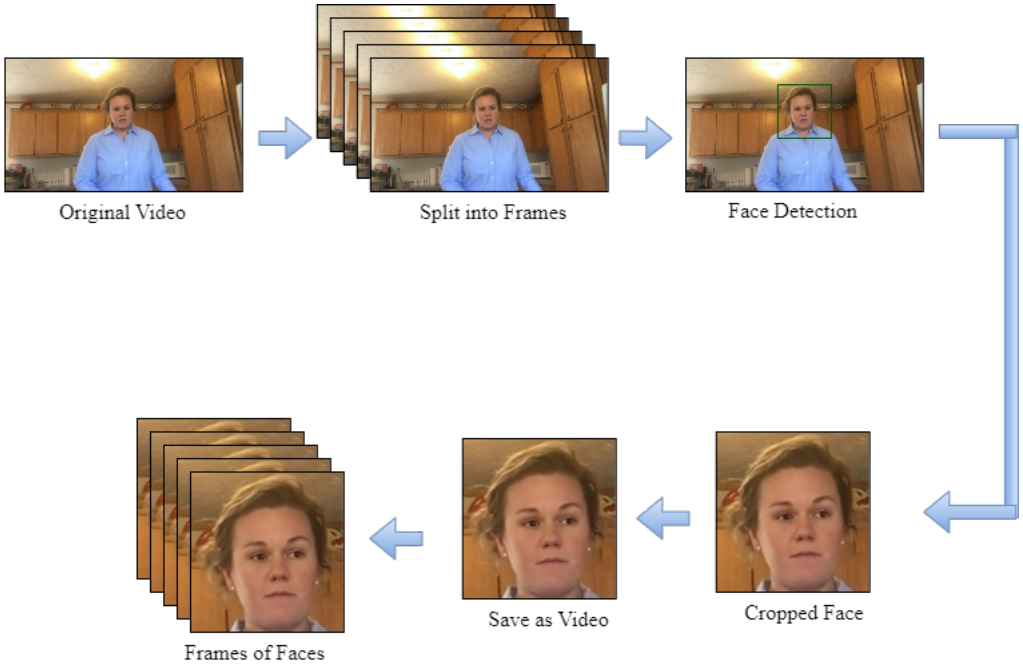


Figure 5.1: Real video frame split

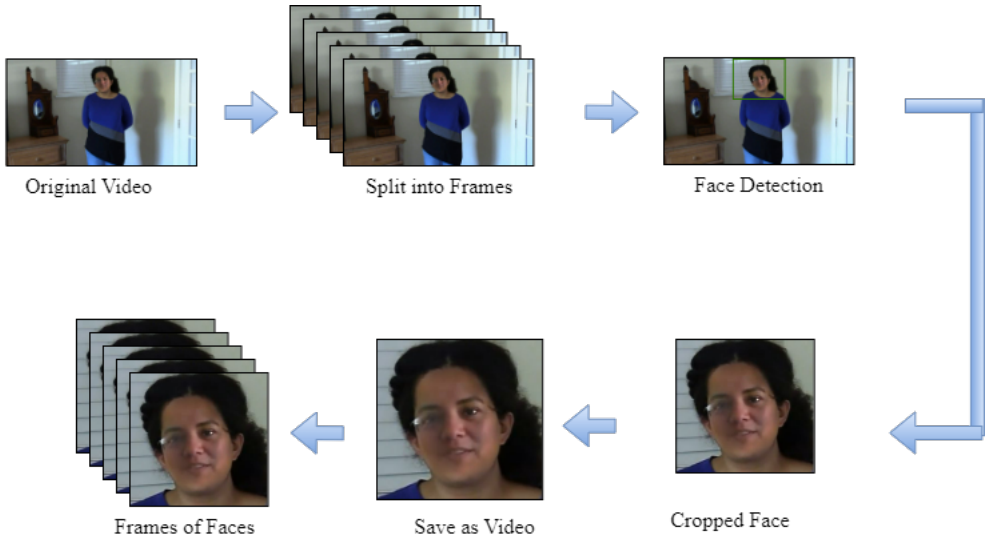


Figure 5.2: Fake video frame split

5.1.4 Data Augmentation

Convert videos to PIL Image:

PIL (Python Imaging Library) is a default python library that provides us the opportunity to do various image processing operation such as: manipulating and editing the extracted individual frames from the videos. PIL library gives us the chance to perform various operation, for example: adjusting the contrast, brightness which have been done in Color Jitter, applying filter which eventually helps our model to have better resolution frames.

Frame size:

All the frames have been resized to 112x112 for better accuracy. This action ensures the same and consistent dimension of each frames of the videos.

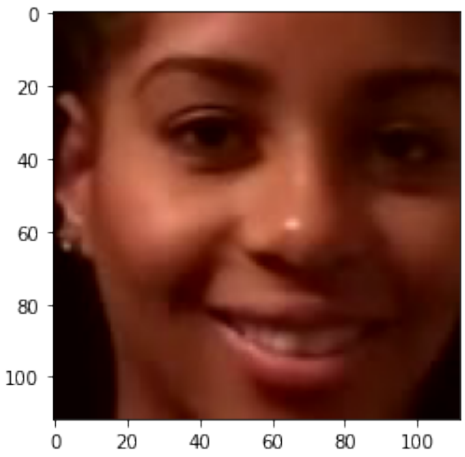


Figure 5.3: Frame size

Random Horizontal Flip:

For introducing diversity in the model training we have performed random horizontal flip. It will help the model to learn features that are not dependent on the specific orientation of faces, making it more robust to variations in the test data.

Color Jitter

For detecting the faces in frames more clearly, we conducted "Color Jitter" which is applying transformation to each frames. For adjustments, we have increased the values of Brightness, Contrast, Saturation by 40% and Hue by 10%. This ensures that each frames have spontaneous color adjustments and the model performs with variation of color distributions.

Converting frames to Tensor

As we are working on video data, it is very hard for the model to process a total number of 2352 videos with limited resources. By converting the frames of the videos into tensor, the model can analyze the data more efficiently and compute

more accurately. Tensors are known as multidimensional array which can represent the pixels of each frames as a numerical values. By converting the frames to tensors, we enable the model to perform mathematical operations on the pixel values, such as convolutions, matrix multiplications, and activations etc.

Normalizing the tensor

After getting the pixel values, it is necessary to standardize all the pixel value within range and distribution of pixel values for providing the input to the model. To maximize the efficiency in the training process, we have to ensure that the pixel values have a similar scale across different frames and videos.

Check for number of Real and Fake videos:

After uploading the processed videos to the memory, we have to check for the total number of Real and Fake videos uploaded in the memory. After getting the total videos of 2352, we move forward to checking the balanced distribution of Real and Fake videos.

5.1.5 Splitting the data:

The dataset will be split into an 8:2 ratio where 80% of the videos will be used for training the data and 20% of the videos will be used for test videos.

Training data

Among 80% of the training data, which is 1881 videos, the quantity of Real videos are 960 and the quantity of fake videos are 921.

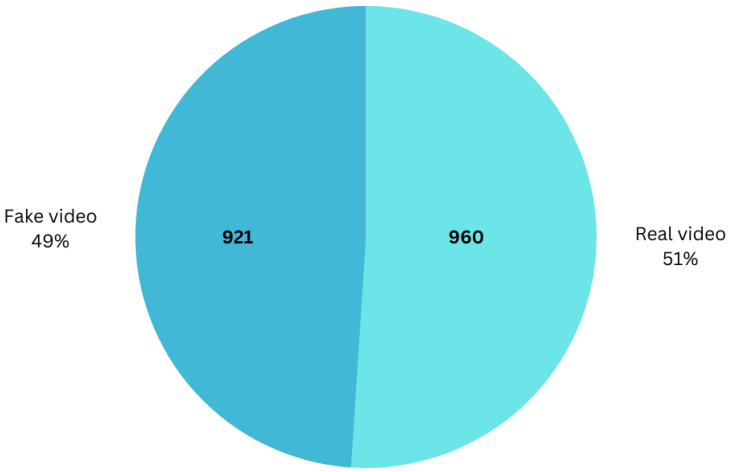


Figure 5.4: Training Dataset Distribution

Testing data

There are a total of 471 videos in testing data which consists of 216 Real videos and 255 Fake videos.

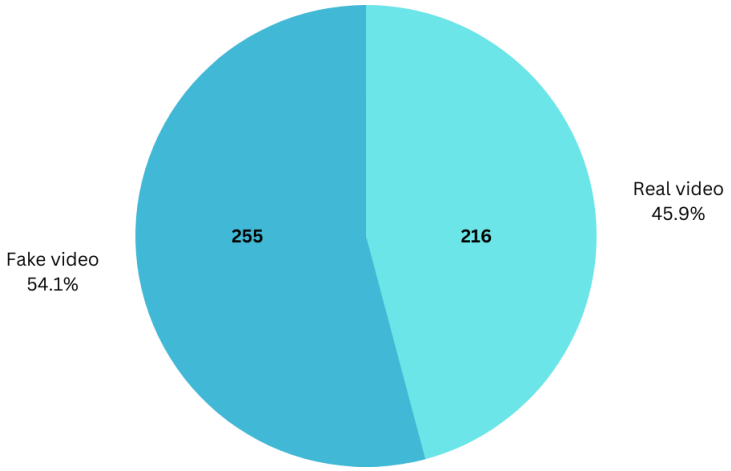


Figure 5.5: Testing Dataset Distribution

Chapter 6

Implementation of Proposed model

6.1 Model Architecture

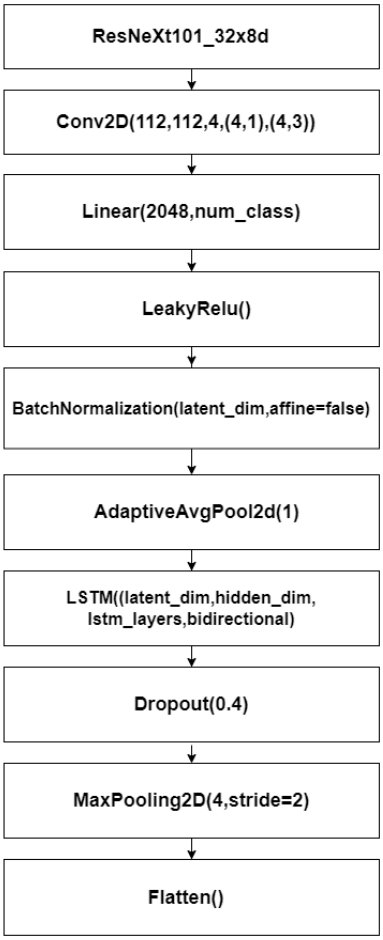


Figure 6.1: Model Architecture

The DeepFake detection model is an integrated CNN and LSTM model. CNN model is used to analyze the visual data and extract the feature. For extracting the feature, a ResNeXt architecture has been used and a custom CNN model has been used for analyzing the data. The CNN model's output will be used as the

LSTM model's input to analyze the temporal sequence and detect inconsistencies between the frames. The overall architecture combines convolutional operations, adaptive pooling, LSTM for temporal analysis, and linear layers to extract discriminative features from the input frames and learn representations that can be used for DeepFake detection.

6.2 $ResNeXt101\_32*8d$:

The ResNeXt101 model is based on the ResNet model. ResNeXt replicates a structural component that combines a number of transformations with the same topology. In addition to the depth and width dimensions, it also highlights cardinality (the size of the set of transformations) as a new dimension in comparison to ResNet. $ResNeXt101\_32*8d$ has 88791336 parameters. The ResNeXt architecture is known for its ability to capture high-level features and patterns from images. There are a total of 101 layers and the number of Cardinality is 32. This ResNeXt architecutre help the model to recognize general visual patterns present in both real and deepfake videos. The model can identify distortion from the patterns observed in fake videos

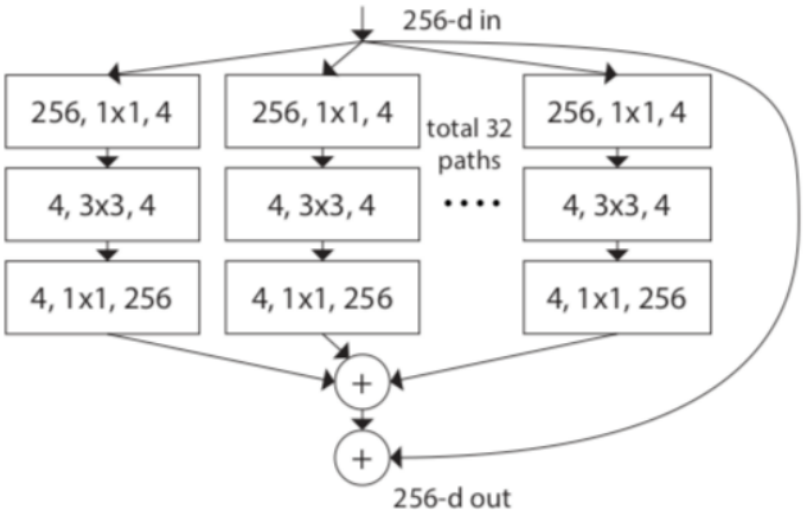


Figure 6.2: ResNeXt101_32x8d architecture

6.3 CNN model:

For the Deep Learning approach to detect Deep Fake, CNN models have proven to have the best accuracy. CNN is known to have multiple layers and each and every layer has different parameters to transform data using convolutional kernels. These transformations help to extract valuable characteristics. Basic CNN architectures consist of Convolution Layer, Pooling Layer, Fully Connected Layer. For our custom CNN model following layers have been used after we have extracted the features from $ResNeXt101\_32*8d$.

6.3.1 Convolution Layer:

For the custom CNN model, a layer of Conv2D has been added. As the frame size was 112x112, the input of Conv2D was 112x112. The kernel size is 3 and stride is (4,1). Padding for the Conv2D layer is (4,3). Afer applying Conv2D

$$\begin{aligned} w_{out} &= \frac{w - k + 2p}{s} + 1 \\ &= \frac{112 - 4 + 2 \times 4}{4} + 1 \\ &= 30 \end{aligned} \tag{6.1}$$

$$\begin{aligned} H_{out} &= \frac{H - k + 2p}{s} + 1 \\ &= \frac{112 - 4 + 2 \times 3}{1} + 1 \\ &= 115 \end{aligned} \tag{6.2}$$

The convolutional layer helps to further process and refine the extracted features by capturing more localized patterns and spatial relationships within the features .

6.4 Linear Layer(linear1):

The Linear Layer added to the model applies linear transformation of the output data from the Convolutional Layer. Linear Layer also known as Fully Connected Layer has input feature and output feature. The $in\_features$ of the layer is 2048 and $out\_feature$ is the same as the number of classes. The bias is set to True. The formula for the linear transformation is following:

$$y = Wx + b \tag{6.3}$$

Here, x is the input sample which has the shape($batch\_size$, 2048) and it is passed to network as a single tensor and using the formula it transform the input to the y which has the shape($batch\_size$, number of classes). b indicates bias whereas W= weight. The purpose of this linear layer is to adjust the dimensionality of the features to better align with the requirements of the subsequent LSTM layer. It helps in reducing the dimensionality if needed or expanding it to a higher-dimensional space to capture relevant information for sequence modeling. It also helps in summarizing the information learned from previous layers

6.5 Leaky Rectified Linear Unit Activation:

The Leaky ReLu activation operates by a fixed scaler which multiplies any input which is less than 0. Leaky ReLu helps the model capture a wider range of information. The formula for Leaky ReLU is given below:

$$f(x) = \begin{cases} 0.1x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \tag{6.4}$$

The Leaky ReLU activation can enhance and refine the learned features by introducing non-linearity and amplifying the activations.

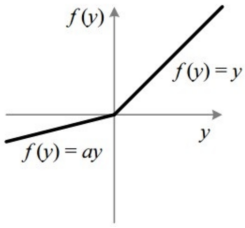


Figure 6.3: Leaky ReLu

6.6 Batch Normalization(BatchNorm1d):

To avoid overfitting of the data, Batch Normalization layer have been used. Also, Batch normalization is applied to normalize the input along the batch dimension. Because of overfitting of the data, the model will not be able to predict the test dataset properly. By adding more layers in the model, the Batch Normalization layer makes the model more stable and fast. It standardizes and normalizes the data which is output from LeakyReLu layer.

6.7 Adaptive Average Pooling 2D Layer (avgpool):

The final layer before providing the input to LSTM is Adaptive Average Pooling. The Avgpool layer dynamically computes the output size based on the input size and ensures a fixed output size regardless of the input size. Adaptive Average Pooling2D layer prepares the feature maps to be compatible with the LSTM layer. By reducing the spatial dimensionality to 1x1, the output of the Adaptive Average Pooling2D layer can be easily reshaped into a suitable input format for the LSTM layer.

6.8 LSTM model:

6.8.1 LSTM Layer

The Recurrent Neural Network (RNN) variant known as Long Short Term Memory (LSTM) features feedforward connections. They are unique RNN variant that addresses the issues with limited memory. The vanishing gradient issue in RNN was eliminated by LSTM, and they are made in a method that enables them to learn long-term dependencies of data and process it sequentially. In the context of deepfake detection, the LSTM layer processes each frame of the input video sequentially, capturing temporal information and comparing the frame properties at different time points. By analyzing the sequential nature of the frames, the LSTM layer can identify inconsistencies and patterns that may indicate the presence of deepfakes. The LSTM layer in our model has 2048 hidden units. The 2048 LSTM layer receives its input from the CNN network's output. After processing each frame in turn, LSTM compares its properties at various points in time. It is possible to tell whether a video is deep-faked or not by comparing the frames. Any video can be supplied to the model for prediction after training. The number of latent dimensions of the LSTM layer is 2048 which is the same as the parameter of hidden dimensions. The LSTM model has 1 layer. The LSTM model is not Bidirectional RNN as the

parameter was set to bidirectional =false. To summarize, LSTM enables the model to understand the context and temporal dynamics within the video, which can be helpful in detecting inconsistencies or patterns associated with deepfake manipulation. It also remembers important past frames and utilize that information to make predictions for the current frame.

6.9 Dropout layer:

The Dropout layer sets the input to 0 randomly at a dropout rate which we have set to 0.4 for our model. The dropout layer mainly deals with the overfitting of the model. In this case, 40% of the inputs will be zeroed out during training.

6.10 MaxPooling layer:

MaxPooling2D layer have been added after the dropout layer to downsample. The MaxPooling2D layer downsamples the input feature maps and reduce the spatial dimensions of the frames. As a result, it can help the model to capture more important features and reduce computational costs. After the MaxPooling layer, the dimensions of the frames have been reduced and helped the model to predict more efficiently. By using a pooling size of (4x4) and a stride of 2, the MaxPooling2D layer downsamples the feature maps by taking the maximum value within each 4x4 region and moving 2 units at a time and the input spatial dimensions are reduced by a factor of 2.

6.11 Flatten layer:

By using the Flatten layer, the model simplifies the data representation, reduce dimensionality, and ensure compatibility with classification algorithms, ultimately aiding in the accurate detection of DeepFake videos. It reshapes the input tensor and 2D or 3D feature maps into a 1D vector.

Chapter 7

Result and Analysis

After running the CNN and LSTM model, we have determined the accuracy with Training accuracy and Validation accuracy, Training loss and Validation loss, confusion matrix, precision, recall and f1 score. We have run 50 epochs with a learning rate of 5e-5.

7.1 Accuracy and Loss of Proposed model

7.1.1 Accuracy of Proposed Model

The training accuracy of the model is 99.52% after 50 epoch which started as 55.56% and after 25 epochs the training accuracy was 98.25%. There were a total of 188 batches with 10 videos in each batch. Analyzing the overall trend of training accuracy, it is observed that the model consistently improved from the beginning of training. This indicates that the model was able to capture important patterns and features in the training data, gradually refining its ability to make accurate predictions.
After various number of adjustments in the parameters of layers and changing the learning rate, we have found the highest accuracy with learning rate of 5e-5 which is 94.055%. With a learning rate of 1e-4, the accuracy was slightly above 90% and below 90% for learning rate of 1e-6. After tuning adjustments, using Adam optimizer provided better result. After 25 epochs, the accuracy was 94.479% and highest accuracy reached in the model was 95.966% on epoch number 30.

Figure 7.1: Training Accuracy and Testing Accuracy of proposed model

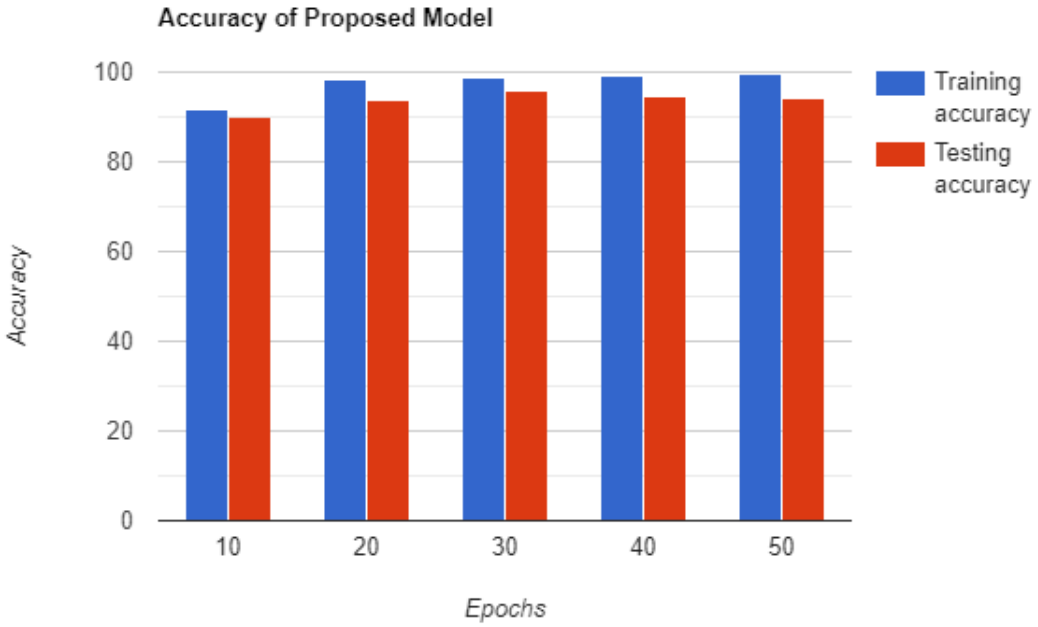


Figure 7.2: Accuracy of proposed model

7.1.2 Loss of Proposed Model

The loss function works as a determiner that differentiates between expected result and model generated result. For generating the loss, we used Cross Entropy loss which adjust the weight of the model during the training. The loss starts very high at the beginning of the model and after adjustments and tuning, the final loss of the training data is 0.127077 which started at 0.701769 and final loss of testing data is 0.253015 which was 0.501568 at the beginning. So, it is safe to say that the downfall of loss data predicts the success of the model.

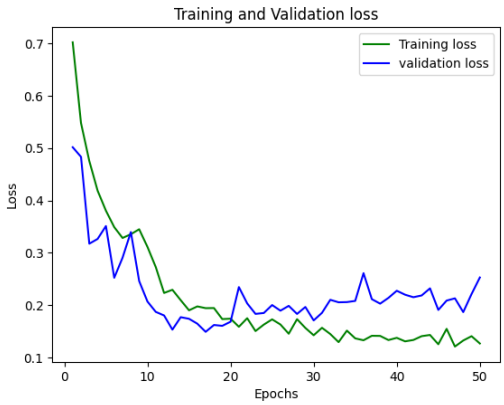


Figure 7.3: Training loss and Testing loss of proposed model

7.1.3 Confusion Matrix

The total number of testing data is 471 where 216 data is Real videos and 255 Videos are Fake videos. Our model has predicted at an accuracy of 94.055%. 242 Fake videos have been predicted correctly and 13 Fake videos were predicted as Real videos by the model. For Real videos, 201 videos have been predicted correctly whereas 15 videos were anticipated as Fake videos.

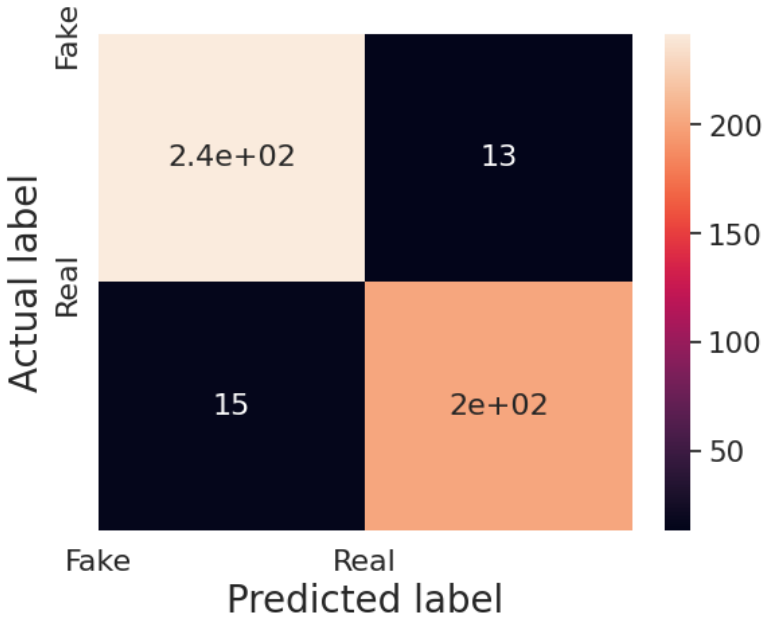


Figure 7.4: Confusion Matrix of proposed model

True positive = 242
False negative = 13
False positive = 15
True negative = 201

True positive indicates the Fake videos predicted correctly by the model. False negative is the value of fake videos which are labeled incorrectly by the model. For real videos, true negative values indicates correctly labeled videos by the model and false positive values which are predicted as Fake videos.

7.1.4 Precision, Recall, F1-score

For our model, precision indicates how accurately our model detects DeepFake videos out of all the videos it predicts as DeepFake whereas, recall indicates how accurately our model identifies DeepFake videos among all the actual DeepFake videos. The F1 score combines precision and recall to give an overall evaluation of the model's performance by taking account of the model's ability to detect DeepFake videos (recall) and the accuracy of those predictions (precision).

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.94	0.95	0.95	255
Real video	0.94	0.93	0.93	216
accuracy			0.94	471
macro avg	0.94	0.94	0.94	471
weighted avg	0.94	0.94	0.94	471

Table 7.1: Classification Report of proposed model

The F1-score is 0.95 for False videos and 0.93 for Real videos
F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.94 * 0.95}{0.94 + 0.95} \\ &= 0.95 \end{aligned} \tag{7.1}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.94 * 0.93}{0.94 + 0.93} \\ &= 0.93 \end{aligned} \tag{7.2}$$

7.2 Accuracy and Loss of other Models

7.2.1 ResNet152

ResNet-152 is a deep convolutional neural network architecture and it was introduced as part of the ResNet (Residual Network) family of models. It is an extension of the original ResNet model proposed by He et al.[26] in 2015, with 152 layers. The ResNet-152 architecture follows a basic building block called a residual block. It consists of a stack of convolutional layers. Each residual block has two main paths: the identity path and the shortcut path. The identity path performs a series of convolutional operations. The shortcut path directly connects the input to the output of the block. The ResNet-152 architecture has shown impressive performance in various computer vision tasks, particularly in image classification and object detection. It has been pre-trained on large-scale datasets, such as ImageNet, and can be fine-tuned or used as a feature extractor for transfer learning in various applications.

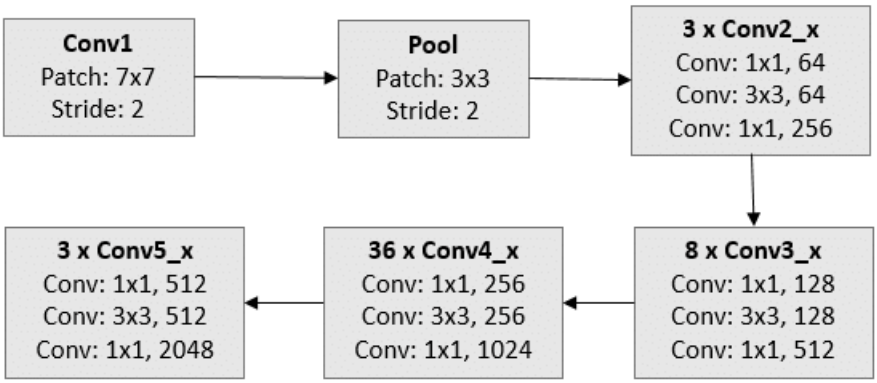


Figure 7.5: Basic ResNet152 Architecture [27]

7.2.2 Dataset Description for ResNet152

The dataset will be split into an 8:2 ratio where 80% of the videos will be used for training the data and 20% of the videos will be used for test videos.

Training data

Among 80% of the training data, which is 1881 videos, The quantity of Real videos are 910 and The quantity of fake videos are 971.

Testing data

There are a total of 471 videos in testing data which consists of 266 Real videos and 205 Fake videos.

7.2.3 Accuracy of ResNet152

The training accuracy of the model is 92.82% after 50 epoch which started as 50.93% and after 25 epochs the training accuracy was 88.94%. There were a total of 188

batches with 10 videos in each batch.
The final accuracy of the model is 91.51% after 50 epoch which is less than the proposed model.

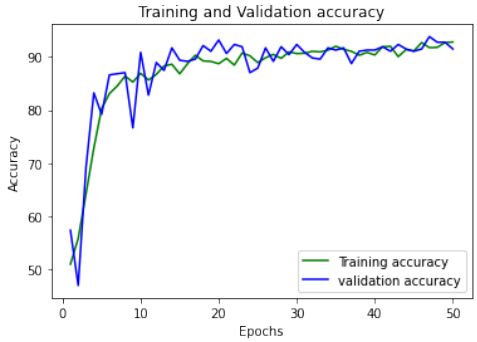


Figure 7.6: Training Accuracy and Testing Accuracy of ResNet152

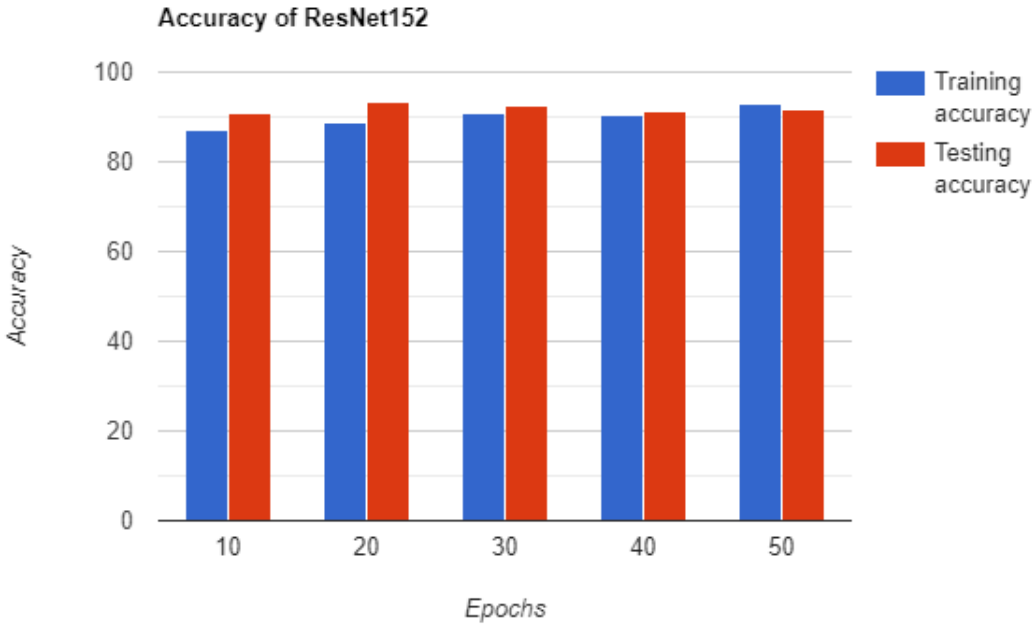


Figure 7.7: Accuracy of ResNet152

7.2.4 Loss of ResNet152

For generating the loss, we used Cross Entropy loss which adjust the weight of the model during the training. The loss starts very high at the beginning of the model and after adjustments and tuning, the final loss of the training data is 0.342545 which started at 0.706269 and final loss of testing data is 0.236721 which was 0.675880 at the beginning.

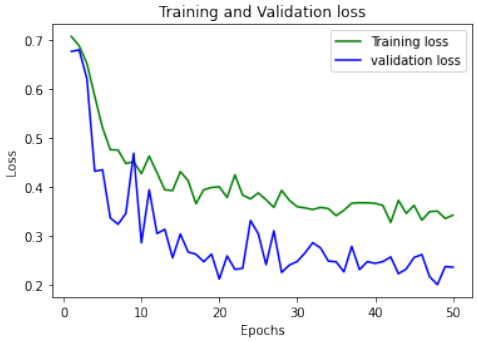


Figure 7.8: Training loss and Testing loss of ResNet152

7.2.5 Confusion Matrix of ResNet152

The total number of testing data is 471 where 266 data is Real videos and 205 Videos are Fake videos. ResNet152 has predicted at an accuracy of 91.51%. 196 Fake videos have been predicted correctly and 9 Fake videos were predicted as Real videos by the model. For Real videos, 235 videos have been predicted correctly whereas 31 videos were anticipated as Fake videos.

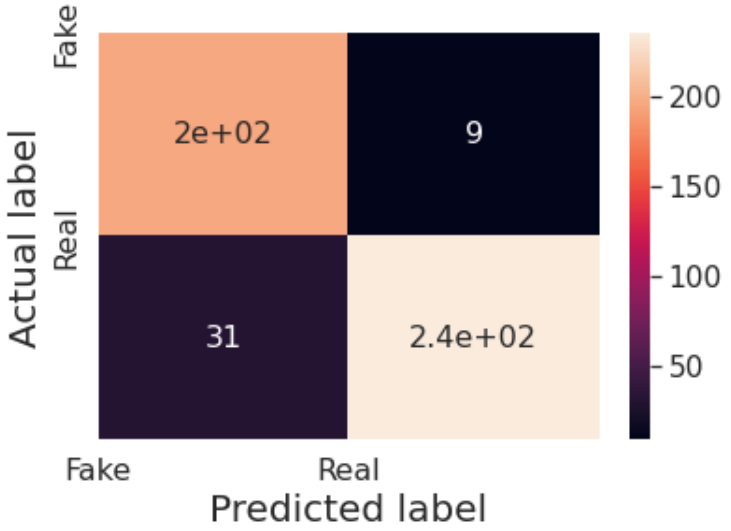


Figure 7.9: Confusion Matrix of ResNet152

True positive = 196
False negative = 9
False positive = 31
True negative = 235

7.2.6 Precision, Recall, F1-score of ResNet152

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.86	0.96	0.91	205
Real video	0.96	0.88	0.92	266
accuracy			0.92	471
macro avg	0.91	0.92	0.91	471
weighted avg	0.92	0.92	0.92	471

Table 7.2: Classification Report of ResNet152

The F1-score is 0.91 for False videos and 0.92 for Real videos
F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.86 * 0.96}{0.86 + 0.96} \\ &= 0.91 \end{aligned} \tag{7.3}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.96 * 0.88}{0.96 + 0.88} \\ &= 0.92 \end{aligned} \tag{7.4}$$

7.2.7 ResNeXt101_64x8d

ResNeXt101_64x8d is a variant of the ResNet architecture introduced by Xie et al[28]. It is designed to improve the representational power and efficiency of deep neural networks. ResNeXt101_64x8d has a cardinality of 64 and our proposed model ResNeXt101_64x8d has cardinality of 32. So, ResNeXt101_64x8d uses multiple parallel path instead of single convolutional path. Each path consists of a group of convolutional filters, and the cardinality determines the number of such groups. In this case, there are 64 groups of convolutional filters. The overall structure of ResNeXt101_64x8d is similar to the original ResNet architecture. It consists of a series of residual blocks stacked on top of each other. Each residual block includes multiple convolutional layers, batch normalization, and ReLU activation functions. The skip connections, or shortcut connections allows the network to learn residual mappings.

7.2.8 Dataset Description for ResNeXt101_64x8d

The dataset will be split into an 8:2 ratio where 80% of the videos will be used for training the data and 20% of the videos will be used for test videos.

Training data

Among 80% of the training data, which is 1881 videos, The quantity of Real videos are 931 and The quantity of fake videos are 950.

Testing data

There are a total of 471 videos in testing data which consists of 245 Real videos and 226 Fake videos.

7.2.9 Accuracy of ResNeXt101_64x8d

The training accuracy of the model is 96.07% after 50 epoch which started as 53.54% and after 25 epochs the training accuracy was 93.46%. There were a total of 188 batches with 10 videos in each batch.
The final accuracy of the model is 92.36% which is more than ResNet152 but less than propposed model.

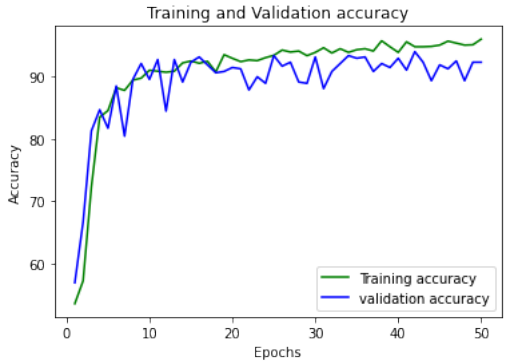


Figure 7.10: Training Accuracy and Testing Accuracy of ResNeXt101_64x8d

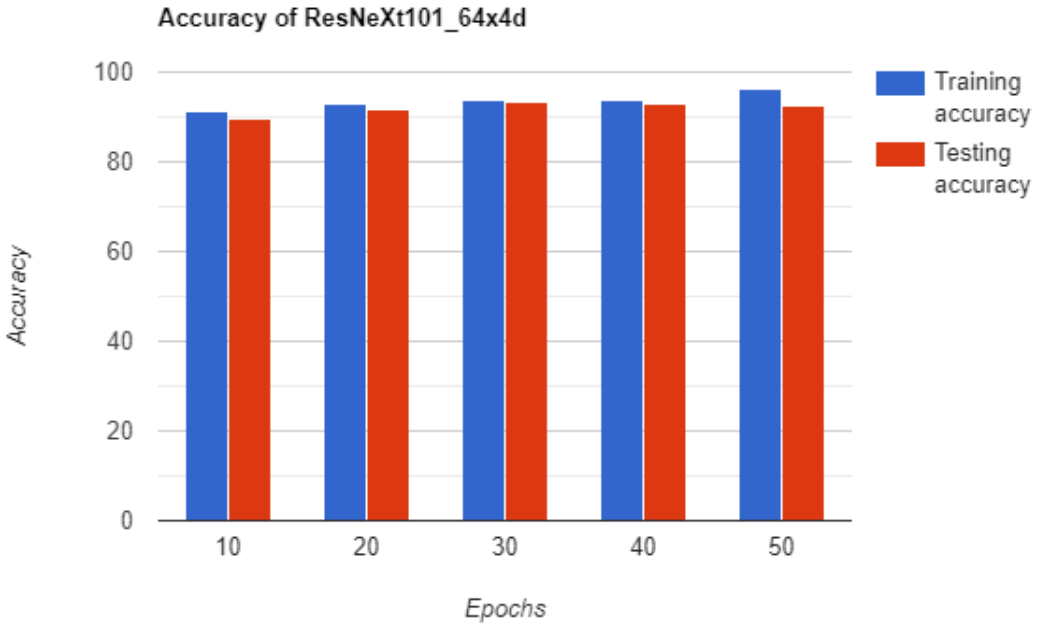


Figure 7.11: Accuracy of ResNeXt101_64x8d

7.2.10 Loss of ResNeXt101_64x8d

The loss starts very high at the beginning of the model and after adjustments and tuning, the final loss of the training data is 0.264879 which started at 0.699409 and final loss of testing data is 0.205366 which was 0.675796 at the beginning. So, it is safe to say that the downfall of loss data predicts the success of the model.

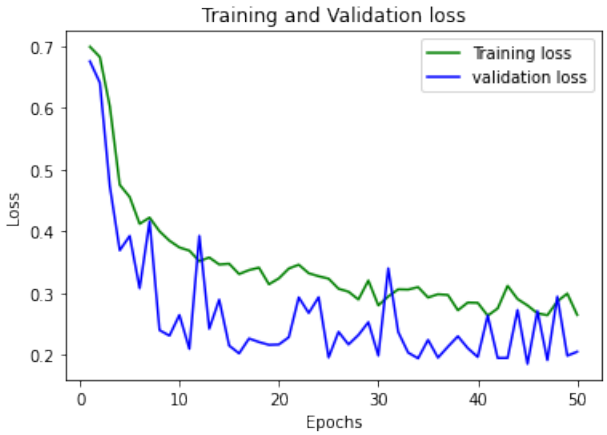


Figure 7.12: Training loss and Testing loss of ResNeXt101_64x8d

7.2.11 Confusion Matrix of ResNeXt101_64x8d

The total number of testing data is 471 where 245 data is Real videos and 226 Videos are Fake videos. $ResNeXt101\_64*4d$ has predicted at an accuracy of 92.36%. 209 Fake videos have been predicted correctly and 17 Fake videos were predicted as Real videos by the model. For Real videos, 226 videos have been predicted correctly whereas 19 videos were anticipated as Fake videos.

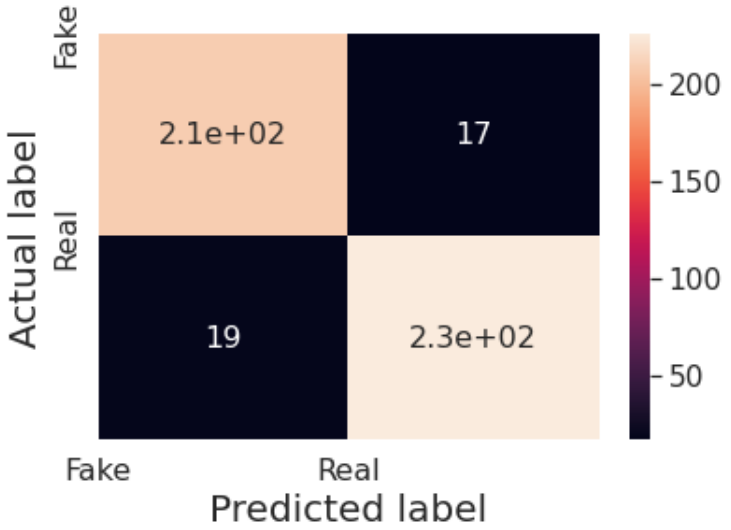


Figure 7.13: Confusion Matrix of ResNeXt101_64x8d

True positive = 209
False negative = 17
False positive = 19
True negative = 226

7.2.12 Precision, Recall, F1-score of ResNeXt101_64x8d

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.92	0.92	0.92	226
Real video	0.93	0.92	0.93	245
accuracy			0.92	471
macro avg	0.92	0.92	0.92	471
weighted avg	0.92	0.92	0.92	471

Table 7.3: Classification Report of ResNeXt101_64x8d

Finally, the F1-score is 0.92 for False videos and 0.93 for Real videos
F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.92 * 0.92}{0.92 + 0.92} \\ &= 0.92 \end{aligned} \tag{7.5}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.93 * 0.92}{0.93 + 0.92} \\ &= 0.93 \end{aligned} \tag{7.6}$$

7.3 Accuracy and Classification comparison of Models

Based on these results, the proposed model has the highest F1 score for fake videos (0.95) among all the models compared. ResNet152 and ResNeXt101_64x8d have slightly lower F1 scores for fake videos. However, for real videos, the proposed model and ResNeXt101_64x8d have the same F1 score of 0.93, while ResNet152 has a slightly lower F1 score (0.92).

Model	**Accuracy**	**Precision**	**Recall**	**f1-Score**
Proposed Model	0.9405	0.94	0.95	0.95
ResNet152	0.9151	0.86	0.96	0.91
ResNeXt101_64x8d	0.9236	0.92	0.92	0.92

Table 7.4: Classification Report of Models on Fake videos

Model	**Accuracy**	**Precision**	**Recall**	**f1-Score**
Proposed Model	0.9405	0.94	0.93	0.93
ResNet152	0.9151	0.96	0.88	0.92
ResNeXt101_64x8d	0.9236	0.93	0.92	0.93

Table 7.5: Classification Report of Models on Real videos

7.4 Model Training and Optimization Techniques

For achieving the best outcome of the model, we have tried an iterative approach of trial and error. Though this approach was becoming inefficient as we had to explore a huge range of learning rate. Analyzing the loss function and accuracy of the model, it was obvious that maintaining a fixed learning rate was not an effective strategy. To overcome this problem, we used Learning Rate Scheduler to adjust the learning rate as the training of the model progressed.

7.4.1 Learning Rate Scheduler

The learning rate scheduler operates by defining specific milestones or events during the training process. The learning rate is adjusted according to a predetermined schedule. These adjustments can be based on various factors such as the number of training epochs, the model's performance on the validation set, or other criteria. The choice of milestones and the decay factor was determined through a combination of experimentation and empirical observations. By reducing the learning rate at appropriate intervals, the learning rate scheduler aims enhance the model's stability, and potentially overcome issues such as overfitting or reaching suboptimal solutions. For our model, we have used

- CosineAnnealing Learning Rate
- Cyclic Learning Rate
- Reduce LR on Plateau
- MultiStep Learning Rate

7.4.2 CosineAnnealingLR

Cosine Annealing Learning Rate adjusts the learning rate in a cosine function. It provides a gradual and smooth decrease in learning rate during the training. Our model provides best accuracy with starting of higher learning rate of 5e-5. The initial plan was to check if the model provides better accuracy and stability with lower learning rate. After gradual decrement in learning rate, the model becomes more precise updating and reaching to optimal solution. As a result, the minimum learning rate was set to inspect the accuracy and loss of the model. The formula for learning rate in a certain epoch for CosineAnnealingLR is:

$$\text{learning\_rate} = eta_{\min} + (\text{base\_lr} - eta_{\min}) \cdot \left(1 + \cos\left(\frac{\pi \cdot \text{epoch}}{T_{\max}}\right)\right)/2 \quad (7.7)$$

The parameter for CossineAnnealingLR are given below:

base_lr:

The base_lr is the initial value of the learning rate. The value for base_lr is 5e-5.

T_max:

T_max represents the number of epoch for which CosineAnnealingLR will decrease the learning rate. After T_max epoch, the learning rate will reach to the minimum value which is eta_min. After reaching T_max, learning rate restarts from it's initial value provided in the optimizer. The T_max for our model is 40.

eta_min:

CosineAnnealingLR gradually decreases the learning rate and it reaches to a minimum value. The value for eta_min is 1.945e-5. After reaching the minimum value, the learning rate sets its value to 5e-5.

last_epoch:

This parameter indicates the index of the last epoch. It allows us to manually set the starting epoch index. By default, it is set to -1, which means that the scheduler will start from epoch 0.

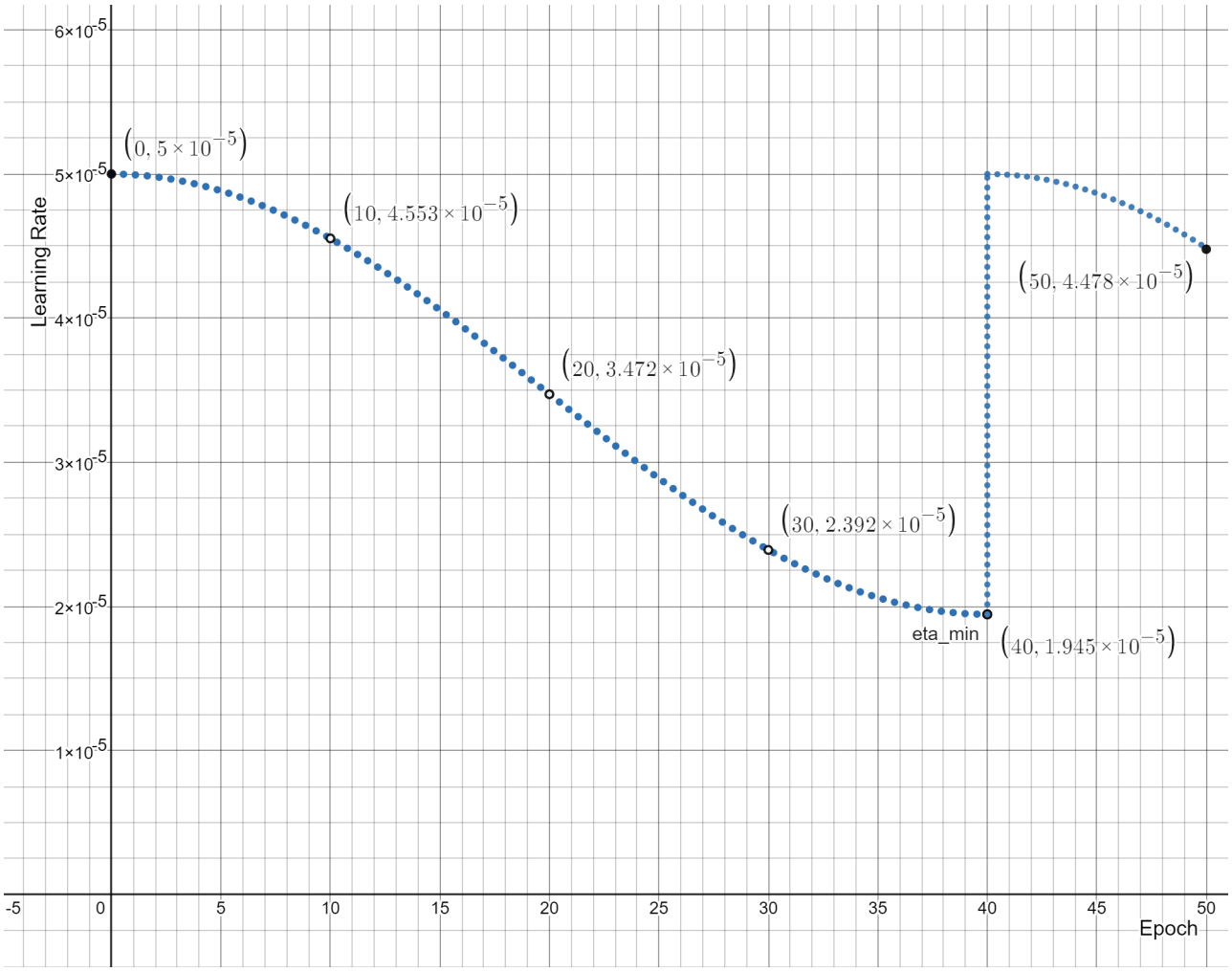
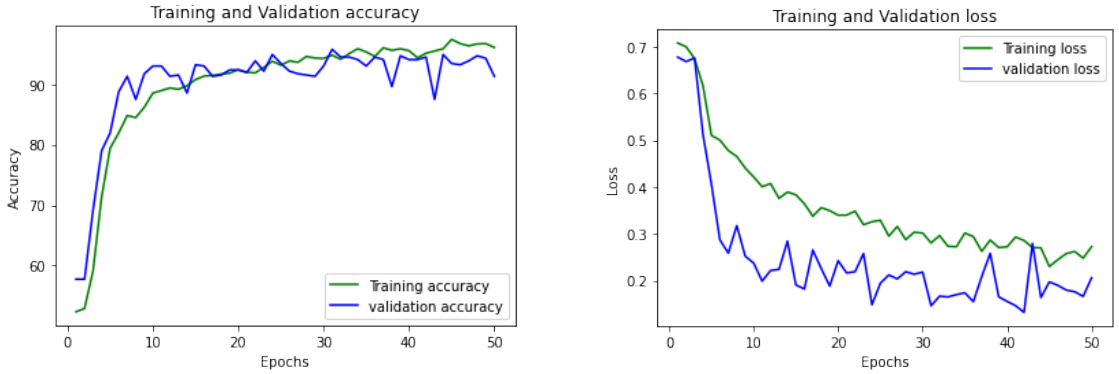


Figure 7.14: Cosine Annealing Graph

For initial epoch, learning rate was 5e-5. After performing cosine function, the learning rate gradually decreases and the learning rate after 10,20,30 epochs are 4.553e-5, 3.472e-5 and 2.392e-5 respectively. After reaching to (T_max,eta_min) the model reaches to final learning rate of the model and restarts the learning rate. So, from 40 epoch, the model starts training at 5e-5 learning rate and on final epoch(50th epoch) the model finally reaches to learning rate of 4.478e-5.

Accuracy and Loss of Cosine Annealing Learning Rate:



(a) Cosine Annealing accuracy

(b) Cosine Annealing loss

Figure 7.15: Accuracy and Loss of Cosine Annealing Learning Rate

The training accuracy increased from 52.37% to 96.07%, which suggests that the model was becoming more accurate in its predictions on the training data. The training loss steadily decreased from 0.7079 to 0.2301, indicating that the model was learning and improving over time. The testing accuracy started at 57.75% and improved to a peak of 95.75% at epoch 31. The testing loss decreased from 0.6777 to 0.1319. The testing accuracy fluctuated slightly after epoch 31, but generally remained above 90%. The final accuracy of the model is 91.30%.

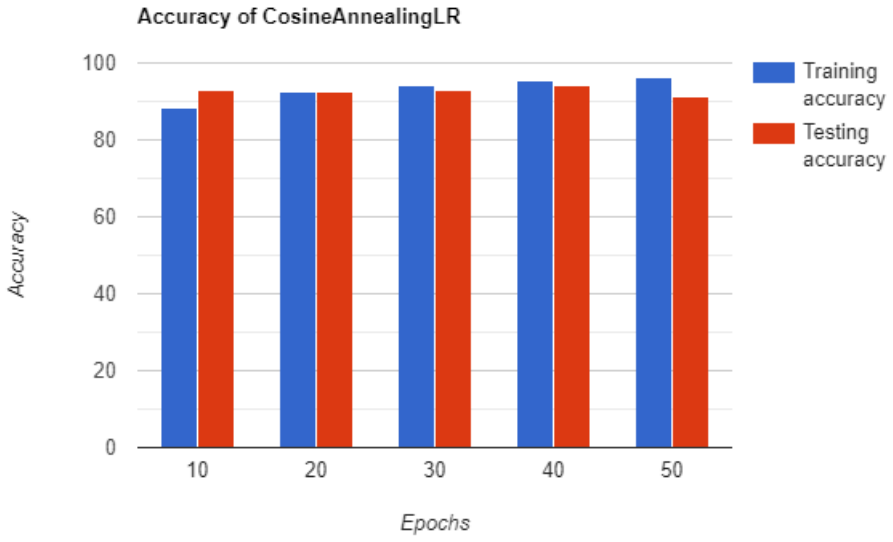


Figure 7.16: Accuracy of Cosine Annealing LR

Confusion Matrix of Cosine Annealing Learning Rate:

The total number of testing data is 471 where 246 data is Real videos and 225 Videos are Fake videos. Using CosineAnnealingLR, our model has predicted at an accuracy of 91.30%. 217 Fake videos have been predicted correctly and 8 Fake videos were predicted as Real videos by the model. For Real videos, 213 videos have been predicted correctly whereas 33 videos were anticipated as Fake videos.

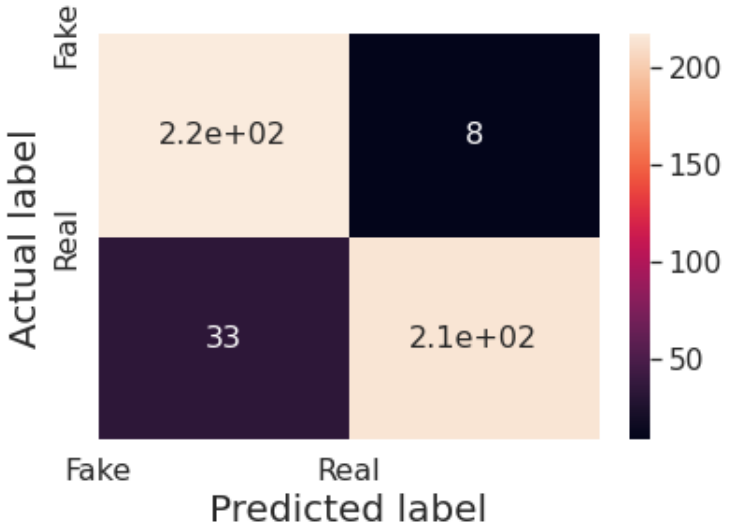


Figure 7.17: Confusion Matrix of Cosine Annealing LR

True positive = 217
False negative = 8
False positive = 33
True negative = 213

Precision, Recall, F1-score of Cosine Annealing Learning Rate:

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.87	0.96	0.91	225
Real video	0.96	0.87	0.91	246
accuracy			0.91	471
macro avg	0.92	0.92	0.91	471
weighted avg	0.92	0.91	0.91	471

Table 7.6: Classification Report of Cosine Annealing Learning Rate

The F1-score is 0.91 for False videos and 0.91 for Real videos
F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.87 * 0.96}{0.87 + 0.96} \\ &= 0.91 \end{aligned} \tag{7.8}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.96 * 0.87}{0.96 + 0.87} \\ &= 0.91 \end{aligned} \tag{7.9}$$

7.4.3 CyclicLR

Cyclic Learning Rate involves cyclically varying the learning rate between a minimum and maximum value during training. This cyclic variation intends to help the model escape from poor local minima, explore different regions of the loss landscape, and potentially find better solutions. The CyclicLR scheduler operates in cycles, with each cycle consisting of two phases: the increasing phase and the decreasing phase. During the increasing phase, the learning rate is gradually increased from the minimum value (base_lr) to the maximum value (max_lr) over a specified number of iterations (step_size_up). This linear increase allows the model to explore different areas of the loss landscape. The CyclicLR scheduler can be configured to follow different patterns within each cycle. We have chosen "Triangular" mode. The parameters of CyclicLR are given below:

base_lr:

This parameter specifies the minimum learning rate in the cyclic schedule. The learning rate will not go below this value during training. For our model, the minimum learning rate is set to 0.

max_lr:

This parameter specifies the maximum learning rate in the cyclic schedule. The learning rate will not exceed this value during training. The maximum learning rate is set to 0.01.

step_size_up:

This parameter sets the number of steps or iterations it takes to increase the learning rate from the base_lr to the max_lr. During this phase, the learning rate gradually increases in a linear manner. In this case, the learning rate will increase for 2000 steps before entering the decreasing phase.

step_size_down:

This parameter determines the number of steps or iterations for the decreasing phase where the learning rate decreases back to the base_lr. Setting it to None means that the scheduler will use the same number of steps as step_size_up for the decreasing phase as well.

mode:

This parameter specifies the shape of the cyclic learning rate schedule within each cycle. It can be set to 'triangular' or 'triangular2'. For our model, the 'triangular' mode is used. The "triangular" mode provides a smooth and continuous transition between the increasing and decreasing phases of the learning rate schedule. It also allows the model to explore a larger portion of the loss landscape, which can lead to improved generalization.

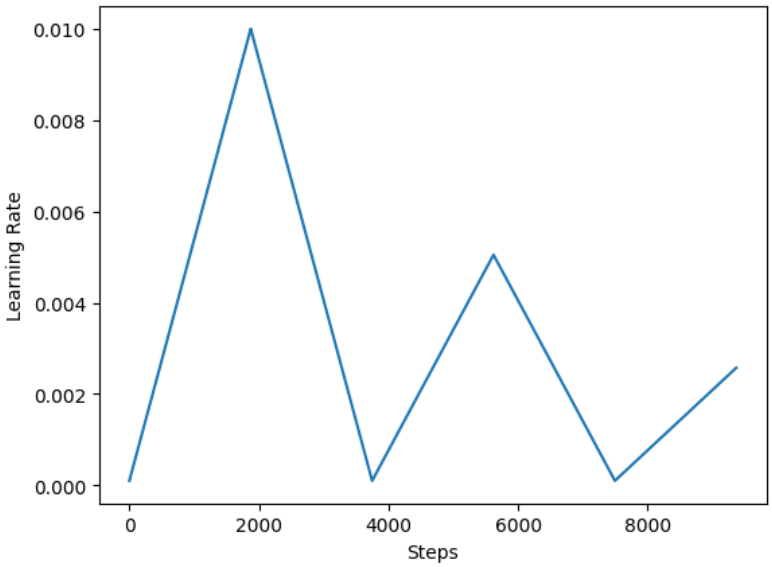
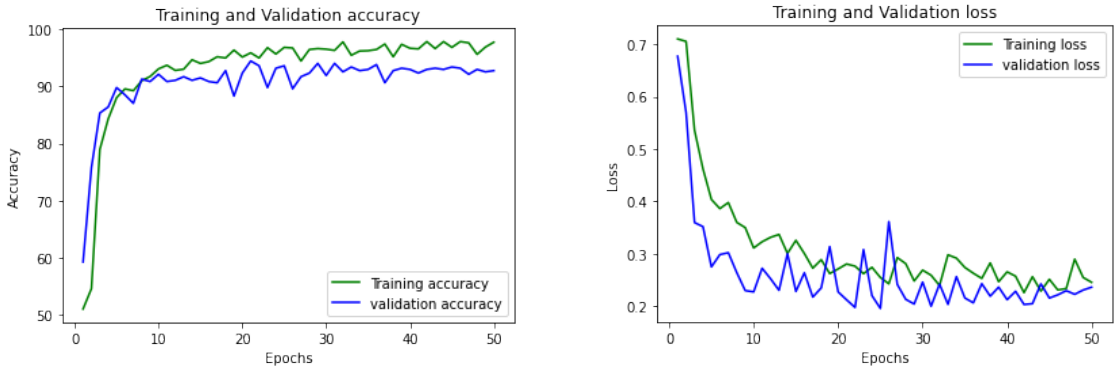


Figure 7.18: Cyclic LR graph

Accuracy and Loss of Cyclic Learning Rate:



(a) CyclicLR accuracy

(b) CyclicLR loss

Figure 7.19: Accuracy and Loss of CyclicLR

During the training process, the model's accuracy steadily improved, reaching a peak of 97.87% at epoch 46. The initial accuracy of training data was 50.98% and the final accuracy was 97.77%.
The initial accuracy on the test dataset was around 59.24%. As the training progressed, the accuracy steadily improved, reaching a peak of 94.48% at epoch 21. However, after epoch 21, the accuracy fluctuated slightly but remained relatively high, ranging from 88.32% to 94.48%. The final accuracy of the model using CycliLR was 92.78%.
The loss value also decreased throughout the training process, indicating that the model was learning and becoming more accurate. The loss started at 0.710275 and reached its lowest point at 0.195157 at epoch 25. After epoch 25, the loss remained relatively stable, ranging from 0.195157 to 0.361157. The loss was 0.0235826 after 50 epochs.

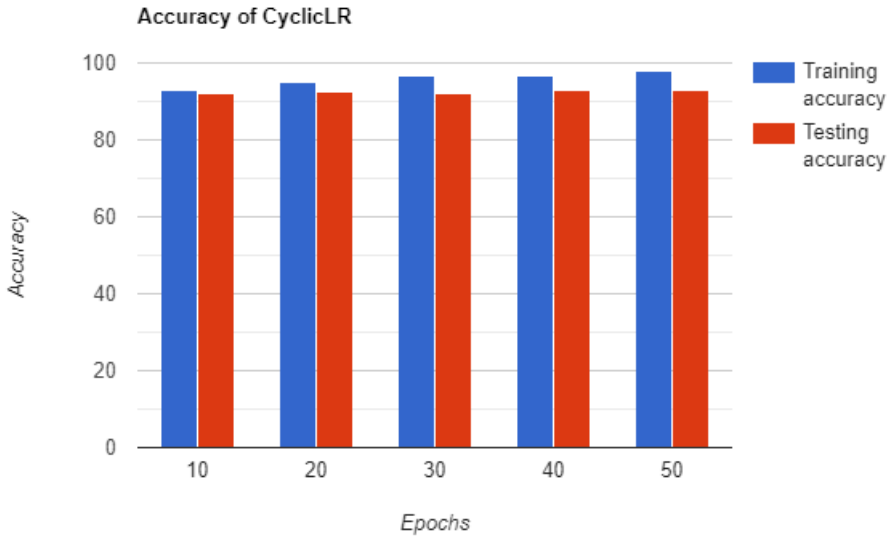


Figure 7.20: Accuracy of CyclicLR

Confusion Matrix of Cyclic Learning Rate:

The total number of testing data is 471 where 224 data is Real videos and 247 Videos are Fake videos. Using CyclicLR, our model has predicted at an accuracy of 92.78%. 227 Fake videos have been predicted correctly and 20 Fake videos were predicted as Real videos by the model. For Real videos, 210 videos have been predicted correctly whereas 14 videos were anticipated as Fake videos.

True positive = 227
False negative = 20
False positive = 14
True negative = 210

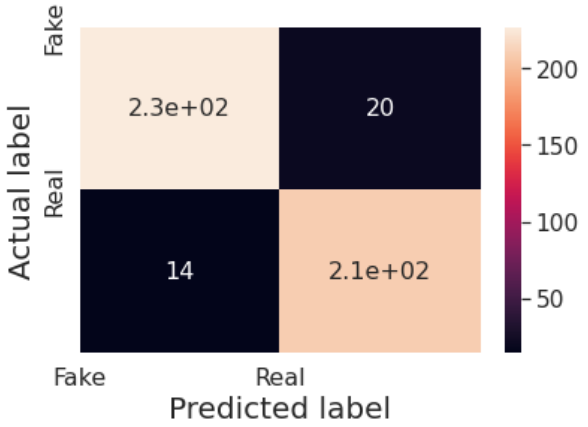


Figure 7.21: Confusion Matrix of CyclicLR

Precision, Recall, F1-score of Cyclic Learning Rate:

The F1-score is 0.93 for False videos and 0.93 for Real videos

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.94	0.92	0.93	247
Real video	0.91	0.94	0.93	224
accuracy			0.93	471
macro avg	0.93	0.93	0.93	471
weighted avg	0.93	0.93	0.93	471

Table 7.7: Classification Report of Cyclic Learning Rate

F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.94 * 0.92}{0.94 + 0.92} \\ &= 0.93 \end{aligned} \tag{7.10}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.91 * 0.94}{0.91 + 0.94} \\ &= 0.93 \end{aligned} \tag{7.11}$$

7.4.4 ReduceLROnPlateau

The "ReduceLROnPlateau" scheduler adjusts the learning rate during training based on the behavior of the model's validation loss. "ReduceLROnPlateau" scheduler monitors the validation loss or metric and reduce the learning rate when the model's performance on the validation set stops improving. As we struggled to get better accuracy from epoch 40 on our proposed model, ReduceLROnPlateau ensures to get better accuracy by adjusting the learning rate by observing the progress so far. It also prevents large learning rate reductions that might hamper the progress.
The parameter for ReduceLROnPlateauLR are given below:

mode:

This parameter specifies whether the monitored quantity should be minimized or maximized. It can be set to 'min', 'max', or 'auto'. For our model, the scheduler is set to minimize the monitored quantity (e.g., validation loss) as 'min'. By setting the mode to 'min', the scheduler focuses on reducing the loss and helps the model converge to a better solution.

factor:

This parameter determines the factor by which the learning rate will be reduced. Our value for factor is 0.1. So, the learning rate will be multiplied by 0.1 after monitoring the validation loss.

patience:

This parameter sets the number of epochs to wait before reducing the learning rate if no improvement is observed. If the monitored quantity does not improve for patience consecutive epochs, the learning rate will be reduced. For our model, patience was set to 5.

threshold:

This parameter specifies the threshold for measuring the improvement in the monitored quantity. If the improvement is less than the threshold=0.01, it is considered insignificant.

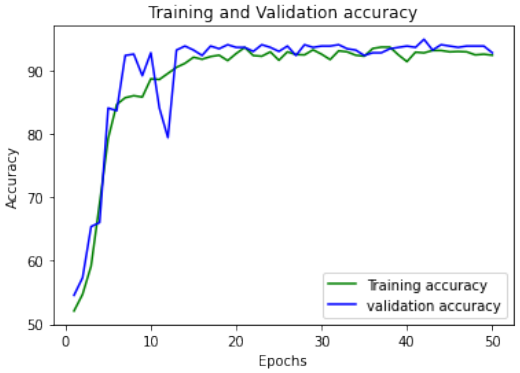
threshold_mode:

This parameter determines how the threshold is interpreted. It can be set to 'rel' or 'abs'. In 'rel' mode, the threshold is relative to the initial value of the monitored quantity. In 'abs' mode, the threshold is an absolute value.

min_lr:

This parameter specifies the lower bound for the learning rate. The scheduler will not reduce the learning rate below this value. The value for min_lr is 0.

Accuracy and Loss of ReduceLROnPlateau:



(a) ReduceLROnPlateau accuracy

(b) ReduceLROnPlateau loss

Figure 7.22: Accuracy and Loss of ReduceLROnPlateau

During the training process, the model started with an initial training accuracy of 52.10%. As the training progressed, the model showed improvements in accuracy, steadily increasing its performance. The highest training accuracy achieved was 93.67% in epoch 37. The final training accuracy was 92.40%. Training loss measures the dissimilarity between the predicted and actual values during training, the initial value was 0.702869. The final loss of training was 0.335139.
Regarding test accuracy, the highest accuracy achieved was 94.90% in epoch 42 whereas, the initial accuracy was 54.56%. However, after epoch 42, there was a slight decrease in test accuracy. It indicates the model's generalization performance started to decline or become less stable. The final accuracy of the model was 92.78%. The test accuracy showed improvement up to epoch 42, followed by a slight decline, while the test loss initially decreased but slightly increased after epoch 30. The final loss of the model was 0.197027. These observations suggest that the model achieved a reasonably good level of performance but started to exhibit signs of overfitting or diminishing returns as the training progressed further.

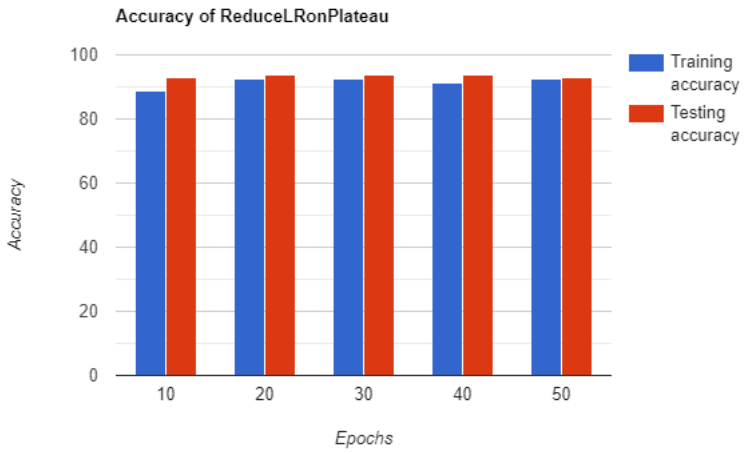


Figure 7.23: Accuracy of ReduceLROnPlateau:

Confusion Matrix of ReduceLROnPlateau:

The total number of testing data is 471 where 246 data is Real videos and 225 Videos are Fake videos. Using ReduceLROnPlateau, our model has predicted at an accuracy of 92.78%. 210 Fake videos have been predicted correctly and 15 Fake videos were predicted as Real videos by the model. For Real videos, 227 videos have been predicted correctly whereas 19 videos were anticipated as Fake videos.

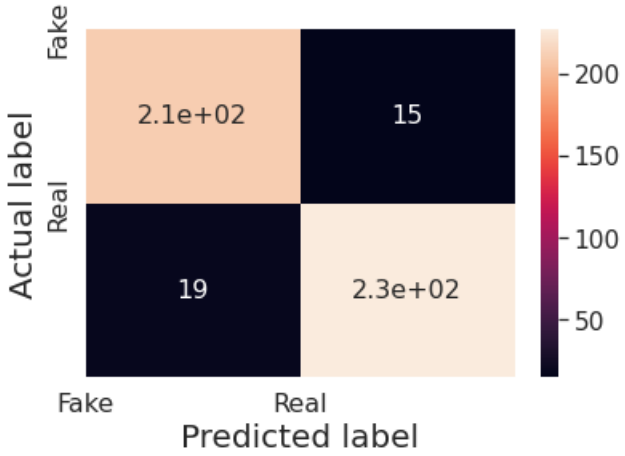


Figure 7.24: Confusion Matrix of ReduceLROnPlateau

True positive = 210
False negative = 15
False positive = 19
True negative = 227

Precision, Recall, F1-score of ReduceLROnPlateau:

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.92	0.93	0.93	225
Real video	0.94	0.92	0.93	246
accuracy			0.93	471
macro avg	0.93	0.93	0.93	471
weighted avg	0.93	0.93	0.93	471

Table 7.8: Classification Report of ReduceLRonPlateau

The F1-score is 0.93 for False videos and 0.93 for Real videos
F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.92 * 0.93}{0.92 + 0.93} \\ &= 0.93 \end{aligned} \tag{7.12}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.94 * 0.92}{0.94 + 0.92} \\ &= 0.93 \end{aligned} \tag{7.13}$$

7.4.5 MultiStepLR

The MultiStepLR scheduler helps us adjust the learning rate during the training. It divides the training process into different phases, and at specific points called "milestones," it changes the learning rate to guide the optimization process. After using 3 different LR schedulers and observing the accuracy and loss curve, we needed a LR scheduler which we can control manually by providing the milestone in which learning rate will change. Unlike Cosine Annealing, where the learning rate decreases gradually following a cosine curve, the MultiStepLR scheduler allows us to define the decay strategy at each milestone. Also, MultiStepLR scheduler provides a stable learning rate schedule with clearly defined adjustments at specified epochs. Unlike CyclicLR or ReduceLROnPlateau, which may introduce fluctuations or rapid changes in the learning rate, the MultiStepLR scheduler allows for smoother transitions. Overall, MultiStepLR scheduler provides us with control and flexibility over the learning rate schedule, potentially achieve more stable and consistent training behavior. The parameters for MultiStepLR are:

milestones:

It is a list of epochs at which we want to adjust the learning rate. For our model, the milestones are set to [5, 15], which means that the learning rate will be adjusted at epochs 5 and 15.

gamma:

The gamma parameter determines how much the learning rate should be adjusted at each milestone. It is a multiplicative factor that controls the magnitude of the adjustment. Gamma is set to 0.1, indicating that the learning rate will be multiplied by 0.1 when each milestone is reached.

last_epoch:

This parameter specifies the index of the last epoch. By default, it is set to -1, which means that the scheduler will start from the beginning. It can be set to a different value if we want to resume training from a specific epoch.
In summary, MultiStepLR scheduler is initialized with an optimizer and a list of milestones. At each milestone epoch, the learning rate is adjusted by multiplying it with the specified gamma factor. This allows us to manually control the learning rate schedule, making adjustments at specific points during training. The last_epoch parameter determines the starting point for the scheduler

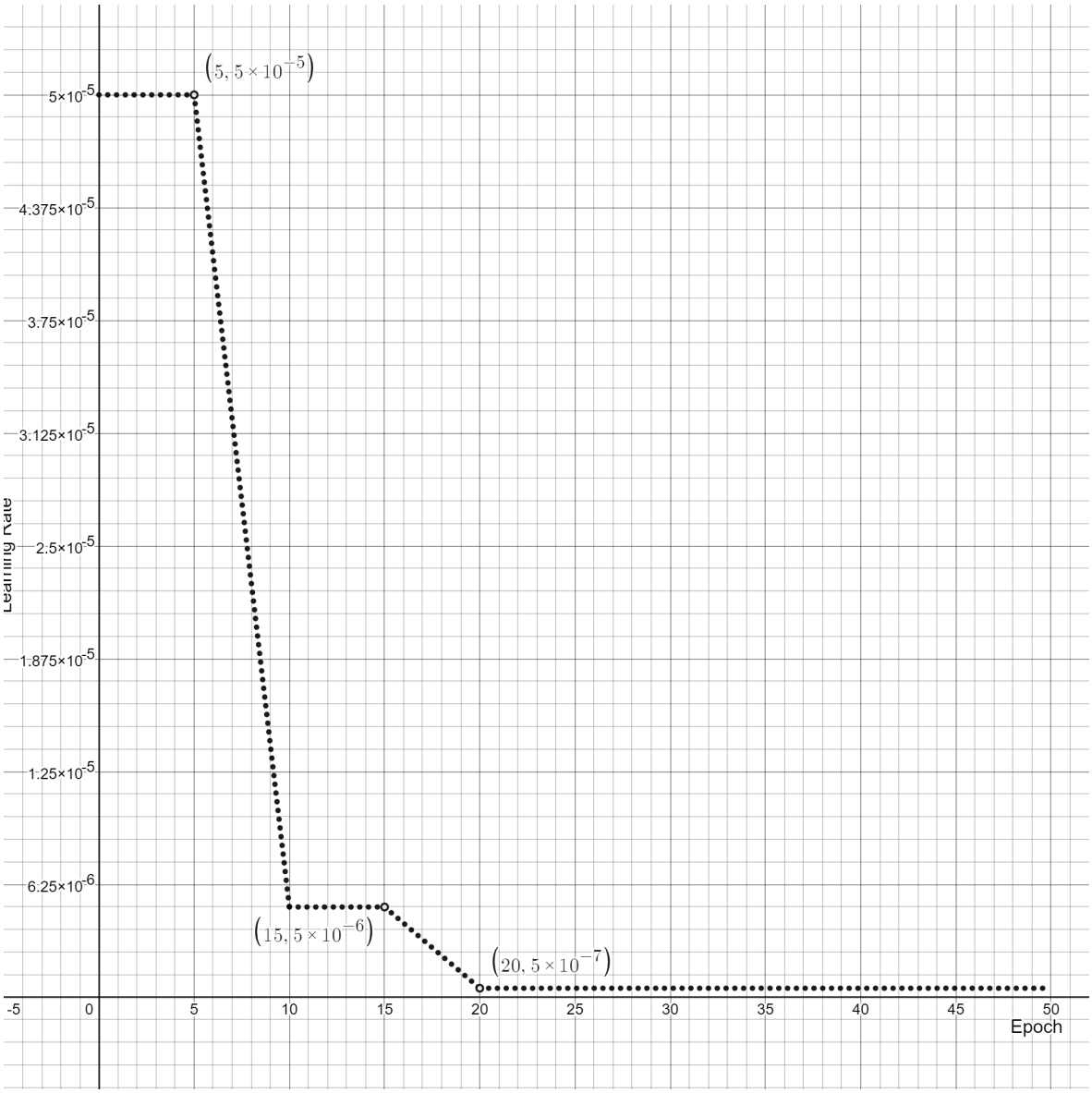
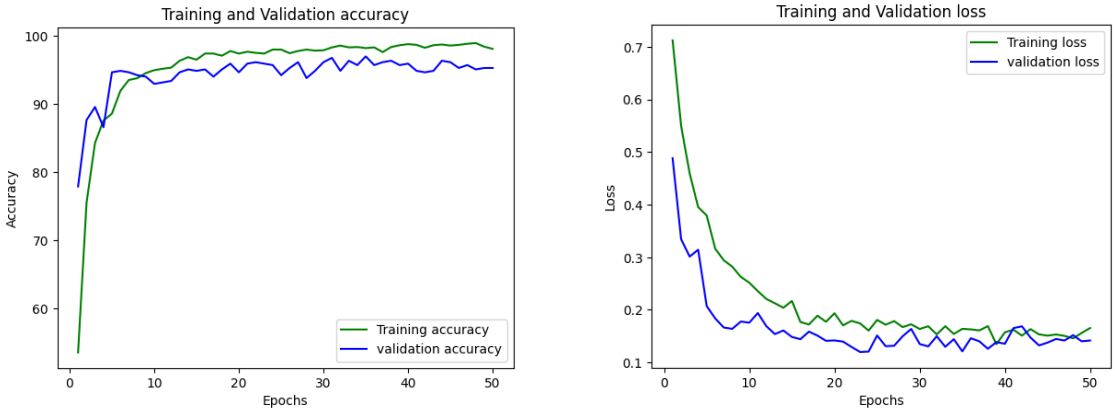


Figure 7.25: MultiStepLR graph

Accuracy and Loss of MultiStepLR:

In the training process, the model started with an initial training accuracy of 53.54%. As the training progressed, the model exhibited a consistent improvement in accuracy, achieving a peak accuracy of 98.99% in epoch 48. The model's improvement in training accuracy began to show from the early epochs, with a noticeable increase in accuracy starting from epoch 2. From epoch 2 onwards, the model exhibited a consistent upward trend, indicating a steady learning process. The last recorded training accuracy was 98.14% in epoch 50, which indicates that the model maintained a high level of performance at the end of training. Regarding training loss, the initial value was 0.712566. As the training progressed, the loss consistently decreased, indicating that the model's predictions gradually aligned more closely with the true labels. The lowest training loss achieved was 0.134682 in epoch 39, which signifies that the model effectively minimized the dissimilarity between its predictions and the actual values.



(a) MultiStepLR accuracy

(b) ReduceLROnPlateau loss

Figure 7.26: Accuracy and Loss of MultiStepLR

The initial test accuracy was 77.92% and the highest accuracy obtained was 97.03% in epoch 35. This demonstrates that the model achieved a high level of generalization, accurately predicting unseen data. However, after epoch 35, there was a slight decrease in test accuracy. The final recorded test accuracy was 95.33% in epoch 50, indicating a reasonably good performance on unseen data at the end of training. The lowest test loss recorded was 0.119841 in epoch 23 and the final test loss was 0.141677.

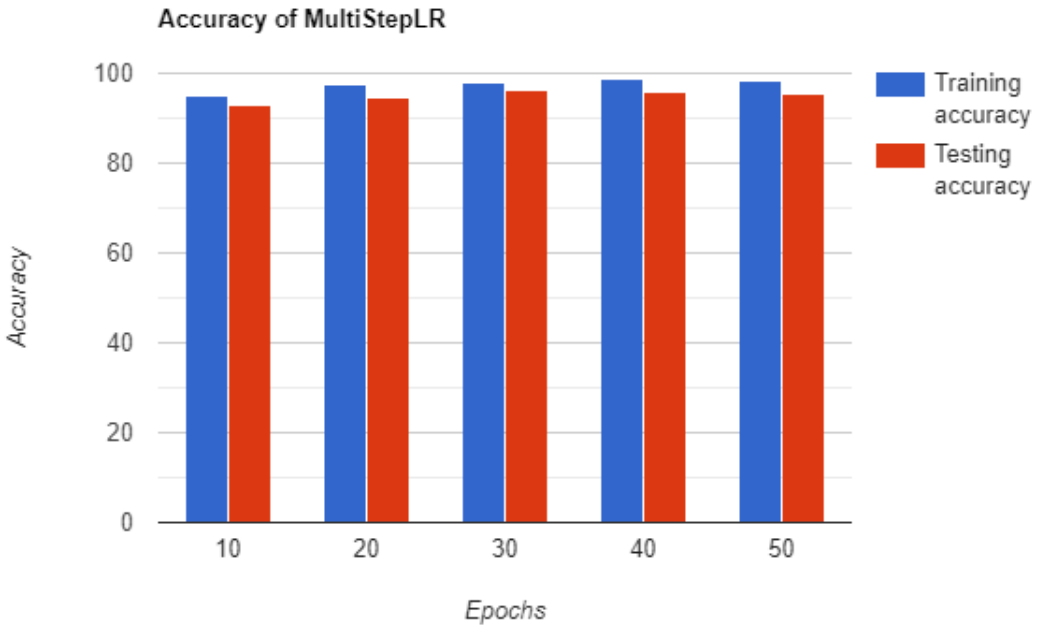


Figure 7.27: Accuracy of MultiStepLR

Confusion Matrix of MultiStepLR:

The total number of testing data is 471 where 234 data is Real videos and 237 Videos are Fake videos. Using MultiStepLR, our model has predicted at an accuracy of 95.33%. 228 Fake videos have been predicted correctly and 9 Fake videos were predicted as Real videos by the model. For Real videos, 221 videos have been predicted correctly whereas 13 videos were anticipated as Fake videos.

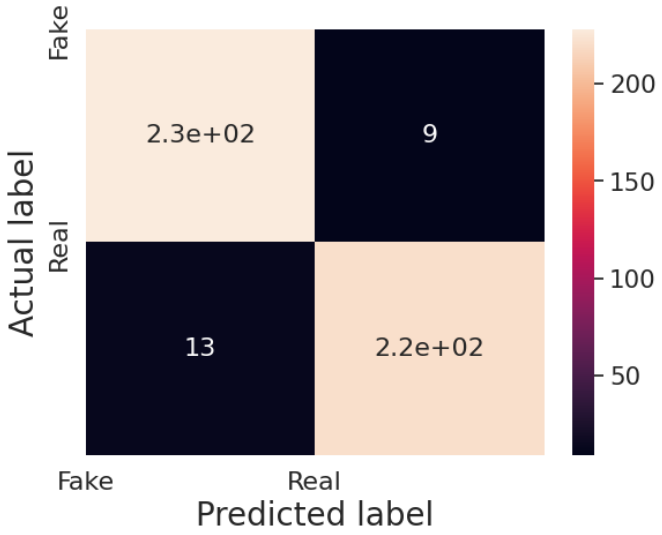


Figure 7.28: Confusion Matrix of MultiStepLR:

True positive = 228
False negative = 9
False positive = 13
True negative = 221

Precision, Recall, F1-score of MultiStepLR:

	Precision	**Recall**	**f1-Score**	**Support**
Fake video	0.95	0.96	0.95	237
Real video	0.96	0.94	0.95	234
accuracy			0.95	471
macro avg	0.95	0.95	0.95	471
weighted avg	0.95	0.95	0.95	471

Table 7.9: Classification Report of MultiStep Learning Rate

The F1-score is 0.95 for False videos and 0.95 for Real videos
F1-score for Fake videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.95 * 0.96}{0.95 + 0.96} \\ &= 0.95 \end{aligned} \tag{7.14}$$

F1-score for Real videos

$$\begin{aligned} F1 - score &= 2 * \frac{precision * recall}{precision + recall} \\ &= 2 * \frac{0.96 * 0.94}{0.96 + 0.94} \\ &= 0.95 \end{aligned} \tag{7.15}$$

7.4.6 Accuracy and Classification Report comparison of Proposed model and LR schedulers

Regarding the learning rate schedulers,we can observe that the MultiStepLR Learning Rate Scheduler outperformed the other schedulers in terms of F1 score for both fake and real videos. It indicates that this scheduler was able to find a good balance between precision and recall, resulting in accurate identification of deepfake videos while minimizing false positives and false negatives. The MultiStepLR scheduler achieves the highest F1 score of 0.95 for both fake and real videos. The Cyclic and ReduceLROnPlateau schedulers have similar F1 scores for both categories, while the Cosine Annealing scheduler has the lowest F1 scores of 0.91 for both fake and real videos.
In summary, the proposed model showed strong performance in detecting fake videos, as indicated by its high F1 score. The learning rate scheduler played a supporting role in optimizing the model's performance, and the specific choice of scheduler could further influence the F1 scores achieved.

Model	**Accuracy**	**Precision**	**Recall**	**f1-Score**
Proposed Model	0.9405	0.94	0.95	0.95
CosineAnnealingLR	0.913	0.87	0.96	0.91
CyclicLR	0.9278	0.94	0.92	0.93
ReduceLROnPlateau	0.9278	0.92	0.93	0.93
MultiStepLR	0.954	0.95	0.96	0.95

Table 7.10: Classification Report of Model and LR scheduler on Fake videos

Model	**Accuracy**	**Precision**	**Recall**	**f1-Score**
Proposed Model	0.9405	0.94	0.95	0.95
CosineAnnealingLR	0.913	0.96	0.87	0.91
CyclicLR	0.9278	0.91	0.94	0.93
ReduceLROnPlateau	0.9278	0.94	0.92	0.93
MultiStepLR	0.954	0.96	0.94	0.95

Table 7.11: Classification Report of Model and LR scheduler on Real videos

Chapter 8

Grad-CAM Implementation

8.1 Grad-CAM

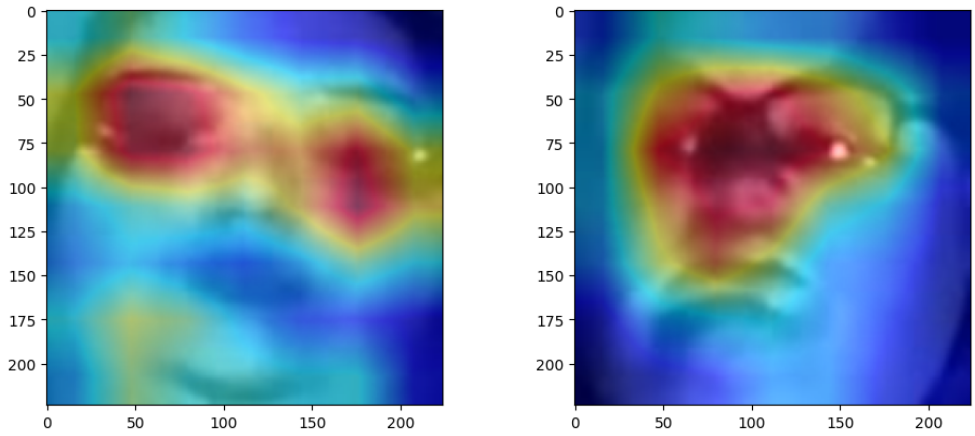
Grad-CAM, short for Gradient-weighted Class Activation Mapping, is a technique used in deep learning and computer vision to visualize the areas of an image that contribute the most to the prediction made by a convolutional neural network (CNN). It provides insights into the regions that the model focuses on when making a decision, helping to explain the model's behavior. Deepfake images or videos are created by manipulating or replacing certain regions within the original content. Grad-CAM can help identify these manipulated regions by highlighting the areas where the model concentrates its attention. By visualizing the activated regions, it becomes easier to pinpoint the parts of the image that have been tampered with.
Deepfake videos are created by manipulating or replacing specific parts of a person's face, and these manipulations can result in discrepancies or inconsistencies that can be detected by deepfake detection models.
Eye inconsistencies: Eyes are highly expressive and contain various subtle details that are unique to individuals. Deepfake videos often struggle to accurately replicate these details, leading to inconsistencies.

- **Blinking artifacts:** Deepfake models may have difficulty accurately replicating the natural blinking patterns of a person, leading to unusual or unnatural blinking in the generated video.
- **Eye shape and alignment:** Deepfake manipulation can result in slight misalignment or distortion of eye shapes. This can manifest as irregular eye proportions or misplacement relative to other facial features.
- **Eye movements:** In some cases, deepfake videos may fail to replicate natural eye movements and gaze shifts, resulting in unrealistic or unnatural eye motion.

Nose inconsistencies: The nose is another facial feature that deepfake models often struggle to manipulate convincingly.

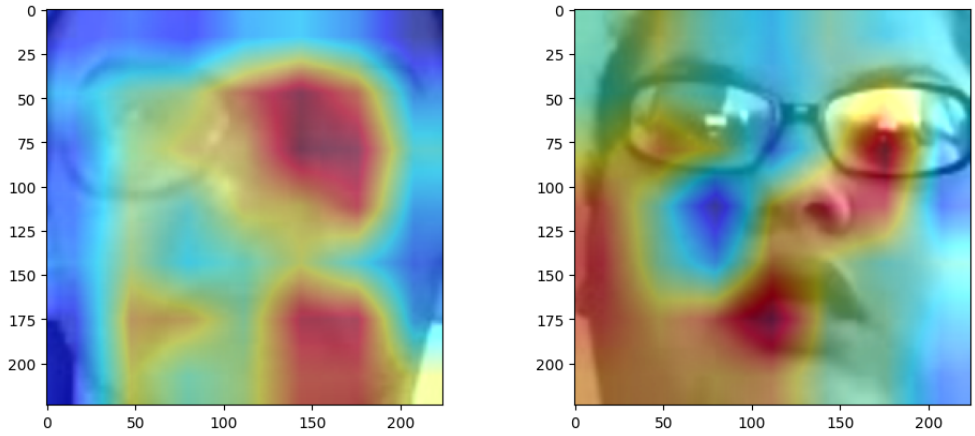
- **Shape distortion:** Deepfake manipulations may cause the nose to appear unnaturally stretched, squashed, or warped.
- **Shadows and lighting:** Lighting conditions in the original video can cast shadows on the nose, which might not be accurately reproduced in the deepfake video, leading to inconsistencies in shadow patterns.

- **Nasal tip alignment:** Misalignment or irregularities in the position or shape of the nasal tip compared to the rest of the face can indicate manipulation.



(a) Inconsistency in both eyes (b) Inconsistency in both eyes and nose

Figure 8.1: Grad-CAM has identifying strong activation in the specific region



(a) Inconsistency in an eye (b) Inconsistency in nose and mouth

Figure 8.2: Grad-CAM showing minimal activation in the particular region

Chapter 9

Conclusion

In this study, we tried to find a more accurate solution to the issue of detecting DeepFake videos. Further advances in face swapping technology, however, will result in more complex Deepfake videos that are harder for current algorithms to detect. As a result, new databases and broader strategies will need to be created in the future. The overall running effectiveness, detection precision, and—most crucially—false positive rate must all be increased for wider practical application. Additionally, detection techniques need to be more resistant against counter-forensic technology, social media washing, and real-world post-processing techniques. In terms of technology, knowledge, and expertise, digital media forensic analysts and forgery producers are always battling. Future consideration will be given to the projection we examine in this study.

Bibliography

[1] S. Lyu, "Deepfake detection: Current challenges and next steps," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2020, pp. 1–6. DOI: 10.1109/ICMEW46912.2020.9105991.

[2] M. Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, pp. 39–52, Nov. 2019. DOI: 10.22215/timreview/1282.

[3] P. Korshunov and S. Marcel, "Deepfakes: A new threat to face recognition? assessment and detection," *CoRR*, vol. abs/1812.08685, 2018. arXiv: 1812.08685. [Online]. Available: http://arxiv.org/abs/1812.08685.

[4] X. Xuan, B. Peng, J. Dong, and W. Wang, "On the generalization of GAN image forensics," *CoRR*, vol. abs/1902.11153, 2019. arXiv: 1902.11153. [Online]. Available: http://arxiv.org/abs/1902.11153.

[5] I. Korshunova, W. Shi, J. Dambre, and L. Theis, "Fast face-swap using convolutional neural networks," *CoRR*, vol. abs/1611.09577, 2016. arXiv: 1611.09577. [Online]. Available: http://arxiv.org/abs/1611.09577.

[6] K. Olszewski, Z. Li, C. Yang, *et al.*, "Realistic dynamic facial textures from a single image using gans," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5439–5448. DOI: 10.1109/ICCV.2017.580.

[7] I. Perov, D. Gao, N. Chervoniy, *et al.*, "Deepfacelab: A simple, flexible and extensible face swapping framework," *CoRR*, vol. abs/2005.05535, 2020. arXiv: 2005.05535. [Online]. Available: https://arxiv.org/abs/2005.05535.

[8] K. Dale, K. Sunkavalli, M. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," *ACM Transactions on Graphics*, vol. 30, pp. 1–10, Dec. 2011. DOI: 10.1145/2024156.2024164.

[9] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of RGB videos," *CoRR*, vol. abs/2007.14808, 2020. arXiv: 2007.14808. [Online]. Available: https://arxiv.org/abs/2007.14808.

[10] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Facevr: Real-time facial reenactment and eye gaze control in virtual reality," *CoRR*, vol. abs/1610.03151, 2016. arXiv: 1610.03151. [Online]. Available: http://arxiv.org/abs/1610.03151.

[11] J. Thies, M. Zollhöfer, C. Theobalt, M. Stamminger, and M. Nießner, "Headon: Real-time reenactment of human portrait videos," *CoRR*, vol. abs/1805.11729, 2018. arXiv: 1805.11729. [Online]. Available: http://arxiv.org/abs/1805.11729.

[12] A. Deshmukh and S. B. Wankhade, "Deepfake detection approaches using deep learning: A systematic review," in *Intelligent Computing and Networking*, V. E. Balas, V. B. Semwal, A. Khandare, and M. Patil, Eds., Singapore: Springer Singapore, 2021, pp. 293–302, ISBN: 978-981-15-7421-4.

[13] A. Kohli and A. Gupta, "Detecting deepfake, faceswap and face2face facial forgeries using frequency cnn," *Multimedia Tools and Applications*, pp. 1–20, May 2021. DOI: 10.1007/s11042-020-10420-8.

[14] D. Güera, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, and E. J. Delp, "A counter-forensic method for cnn-based camera model identification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1840–1847. DOI: 10.1109/CVPRW.2017.230.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[16] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "Faceforensics++: Learning to detect manipulated facial images," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1–11. DOI: 10.1109/ICCV.2019.00009.

[17] S. Tariq, S. Lee, and S. S. Woo, "A convolutional LSTM based residual network for deepfake video detection," *CoRR*, vol. abs/2009.07480, 2020. arXiv: 2009.07480. [Online]. Available: https://arxiv.org/abs/2009.07480.

[18] E. Sabir, J. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, "Recurrent convolutional strategies for face manipulation detection in videos," *CoRR*, vol. abs/1905.00582, 2019. arXiv: 1905.00582. [Online]. Available: http://arxiv.org/abs/1905.00582.

[19] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. arXiv: 1808.03314. [Online]. Available: http://arxiv.org/abs/1808.03314.

[20] A. A. Maksutov, V. O. Morozov, A. A. Lavrenov, and A. S. Smirnov, "Methods of deepfake detection based on machine learning," *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 408–411, 2020.

[21] P. Saikia, D. Dholaria, P. Yadav, V. Patel, and M. Roy, *A hybrid cnn-lstm model for video deepfake detection by leveraging optical flow features*, 2022. arXiv: 2208.00788 [cs.CV].

[22] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "Predicting human eye fixations via an lstm-based saliency attentive model," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 86–90, 2018. DOI: 10.1109/tip.2018.2851672.

[23] S. Aneja and M. Nießner, *Generalized zero and few-shot transfer for facial forgery detection*, 2020. arXiv: 2006.11863 [cs.CV].

[24] P. Ranjan, S. Patil, and F. Kazi, "Improved generalizability of deep-fakes detection using transfer learning based cnn framework," *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, pp. 86–90, 2020.

[25] P. Chen, J. Liu, T. Liang, *et al.*, "Fsspotter: Spotting face-swapped video by spatial and temporal clues," *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2020.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[27] L. Nguyen, D. Lin, Z. Lin, and J. Cao, "Deep cnns for microscopic image classification by exploiting transfer learning and feature concatenation," May 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351550.

[28] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR*, vol. abs/1611.05431, 2016. arXiv: 1611.05431. [Online]. Available: http://arxiv.org/abs/1611.05431.