

An Improved DeepFake Detection Using Deep Learning

by

Annay Paul

18301097

Binayak Kumar Dey

18301054

Md Mostafa

18301132

Maharin Mosfakin

18301119

Rukshana Amin Tonika

18301247

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
September 2022

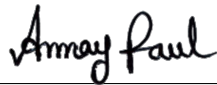
© 2022. Brac University
All rights reserved.

Declaration

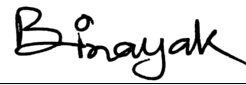
It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Annay Paul
18301097



Binayak Kumar Dey
18301054



Md Mostafa
18301132



Maharin Mosfakin
18301119



Rukshana Amin Tonika
18301247

Approval

The thesis titled “An Improved Deepfake Detection Using Deep learning” submitted by

1. Annay Paul (18301097)
2. Binayak Kumar Dey (18301054)
3. Md Mostafa (18301132)
4. Maharin Mosfakin (18301119)
5. Rukshana Amin Tonika (18301247)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 20, 2022.

Examining Committee:

Supervisor:



Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:

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

Head of Department:

Sadia Hamid Kazi
Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

With the emergence of Generative Models for creating Deepfake images, and videos of high quality, which are extremely realistic and hard to recognize, several social, and political issues have come to light. DeepFakes of celebrities and political personalities are used to exploit and spread misinformation that leads to several social and political unrest. Hence, the necessity to develop a methodology to detect such images and videos created with DeepFakes has arisen in recent times. Several state-of-the-art methodologies are in use for the purpose such as CNN, RNN, reverse engineering of GMs, neural networks, ensemble-based learning approaches, etc. As many machine learning-based approaches are already adopted, our aim is to improve the quality of detection of DeepFakes using models that utilize deep learning, in our study. The state-of-the-art methodologies have shown promising results when applied to popular datasets vastly used for training and research purposes. However, most methods are not robust enough to perform well in all kinds of general-purpose DeepFakes. Hence, in this paper, we have developed a new comprehensive and efficient framework that improves the DeepFake detection performance not only on general purpose but also on training purpose data.

Keywords: Deepfake, GAN, CNN, RNN, Reverse Engineering Of GM's

Acknowledgement

This research was done with the support of BRAC University. First and foremost, we give appreciation to the Great Allah for protecting us from harm during the COVID-19 epidemic, which allowed us to complete our study on time. We would want to take this occasion to express our gratitude to Dr. Md. Iqbal Hossain sir, our supervisor, for all of his efforts in guiding us and for allowing us to work for him. We would like to show our thanks to Dr. Golam Rabiul Alam sir for supporting us greatly from the beginning and for mentoring us and offering guidance while we conducted the research. And lastly, thanks to our parents for their kind prayers and support.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	ix
1 Introduction	1
1.1 Research Problem	1
1.2 Research Objectives	3
1.3 Background Studies	4
2 Related Work	7
3 Dataset And Preprocessing	9
3.1 Preprocessing	9
3.2 Dataset	10
4 Proposed Deepfake Detection Architecture	12
4.1 Motivation	12
4.2 Proposed deepfake detection framework	12
4.3 Short Description Of Proposed Model and Architectures Used For Comparison and Composition	13
4.4 Model Details	15
4.4.1 ConvNeXt:	15
4.4.2 LSTM	16
5 Experiments And Results	20
5.1 Parameter Settings	20
5.2 Evaluation Matrices	21
5.2.1 Adam	21

5.3	Performance Evaluation Matrices	22
5.3.1	Cross Entropy Loss	22
5.4	Result	23
5.4.1	Graphs On Loss function Calculation	24
6	Conclusion	28
6.1	Limitations	28
6.2	Future Work	29
	Bibliography	30

List of Figures

3.1	A Flowchart to showcase the steps for generating output through the model	10
4.1	A block diagram displaying the generic architecture of the proposed model	13
4.2	Standard, Depth-wise and Point-Wise Convolution	15
4.3	Block Modifications (a) Standard ResNeXt Block (b) Inverted Bottleneck Implementation	16
4.4	Block Design of ConvNeXt From	16
4.5	LSTM Dataflow	17
4.6	Cell State	17
4.7	Gate Layer	17
4.8	Forget Gate.	18
4.9	Input Layer, Sigmoid and tanh operation.	18
4.10	Updated Value.	19
4.11	Output Layer.	19
5.1	Loss Function: Swin Transformer architecture	24
5.2	Accuracy: Swin Transformer architecture	25
5.3	Loss Function: VGG16 architecture	25
5.4	Accuracy: VGG16 architecture	26
5.5	Loss Function: ConvNeXt architecture	27
5.6	Accuracy: ConvNeXt architecture	27

List of Tables

5.1	LSTM Blocks and Layers in each architecture	20
5.2	Number of parameters in each architecture	21
5.3	Number of Videos in each architectures	21
5.4	Performance comparison on each dataset.	23

Nomenclature

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

ANN Artificial Neural Network

CNN Convolutional Neural Network

DFDC Deepfake detection challenge

DNN Deep Neural Network

FF++ FaceForensics++

GANs Generative Adversarial Networks

LSTM Long Short Term Memory

ReLU Rectified Linear Unit

RESNET Deep Residual Networks

RNN Recurrent Neural Network

SWIN Shifted Window

VGG Visual Geometric Group

ViT Vision Transformers

Chapter 1

Introduction

In this era of the 21'st century in all of social media, we can see many forged contents, where the authenticity of the contents raises questions. It has come to our senses though we are not bothered by the authenticity of these contents always, many people are taking advantage of this fake content and the victims are experiencing horrible consequences. This problematic situation is caused by DeepFake.

DeepFake refers to fabricated multimedia such as photographs or videos of a person created using deep learning algorithms. Using deep neural architectures many open source software has been developed to create this manipulated multimedia. The availability of this software is so widespread that anyone can create these forged contents very easily. Many advanced technologies have been developed for several years to detect these deepfakes contents. These technologies work effectively in certain contexts. However, these do not guarantee satisfactory results in the identification of deepfake contents in a variety of settings, which is why new technologies are being developed day by day.

Deepfake detection is becoming increasingly relevant at the moment because it causes us to be misled by fake information, also creating chaos and economic loss. Recently, some criminals used deepfake and manipulated personal data and facial recognition systems which caused the Chinese government to lose \$76 million dollar. Then also we have seen US President Donald Trump mocking Belgium for staying in the Paris climate agreement. Following that, we saw Tom Cruise performing coin tricks on social media.[1] After some time, it became clear that the video was a hoax.

As aforementioned, this has encouraged us to focus on improving DeepFake detection. In DeepFake detection, we get to see the use of CNN, RNN, GANS, and many more. CNN gives high accuracy in image analysis. RNN is used with CNN to resolve temporal data problems.

1.1 Research Problem

Due to the exponential growth in the number of users for web applications and social media platforms for content creation and communication respectively the digital world is potentially becoming a resource for both information and misinformation alike. Leveraging this idea several malicious sites and technologies have emerged over the past few years that have attempted to exploit the users all around the globe by spreading misinformation. One such instance of a rising technology is the

creation of deepfake videos. According to [2], deepfakes can produce three various kinds of films, including those that use lip-syncing, face switching, and head puppetry. With the vast amount of technological establishments that are exposed to modern society, such videos may have several social and political repercussions. As such, many companies including the current tech giants such as Google, Facebook, etc. have come forward to participate in fighting the challenges imposed by deepfake videos [3][4]. Even while many deepfakes may be produced using conventional visual components or computer graphics, deep learning techniques like autoencoders and generative adversarial networks—which are already widely utilized in the computer vision field—are the most well-known and commonly employed underlying mechanism for deepfake synthesis. In order to train various models to produce convincing photos and videos, deepfake techniques may require a significant quantity of visual data. Celebrities and politicians are the primary targets of deepfakes since their images and videos are so readily available online [5].

In most concerted initiatives, certain similar flaws in deepfake detection have been observed, such as datasets-related issues, the ambiguity of not covering all accessible datasets by any particular method, and so on. Likewise, all existing databases include the contents of celebrities, politicians, and other well-known people. Furthermore, all current architectures are only applied to static visual contents.

We observed that a single source video was conducted to generate various sorts of alterations in the vast majority of instances. As an outcome, numerous fakes were created by oversampling a single face. Many DeepFake generators use minor adjustments to a source face in order to vary the facial expressions, making it hard to distinguish between altered and actual faces. Due to the sheer lack of variety and oversampling, the data is overfitted immediately even before discovering the important deepfake features. Despite learning manipulation aspects, the networks begin to encode the faces of the subject which causes poor performance [6].

In most datasets, celebrities are targeted in deepfake videos because thousands of pictures of celebrities are available on the internet, and the majority of these images are of the subject facing the camera. As a result, the quality of deepfakes has reached a stage where they are nearly indistinguishable from actual footage, and the focus has shifted towards generating these deepfake of ordinary people. A study conducted by Singh et al [7] showed that a limited number of images taken from a YouTube video can be used to generate photo-realistic artifacts via a GAN-based approach that are not detectable by the human eye. Thus the creation of deepfake videos has taken a great leap forward keeping the generation process no longer limited to the specific datasets already available.

Along with other issues, it has come to our attention that one particular method cannot provide effective accuracy for all datasets. According to [8], the accuracy rate from ResNet50 is 93.70% and VGG16 is 87.50% which might indicate that not all the deep neural network architectures will have satisfactory performances in several datasets. Other than that, all these architectures provide a high-performance rate only for images, not videos.

Though there are several studies that have been and are being conducted on deepfake detection methodologies they suffer from some common issues as mentioned by Lyu et al in [9]. According to the author, the issues can be listed as the following challenges

- **Limitation:** One of the major concerns about deepfake detection algorithms is that though these methods have been trained on different existing datasets, it still faces different challenges regarding performance evaluation [10].
- **Performance evaluation:** Deepfake videos can be considered a binary classification problem. In the real world, these deepfake detection methods result in a blurrier picture. The binary classification approach must be expanded towards different classes of detection, in order to effectively address the complexity of mainstream media falsifications.
- **Explain-ability of Detection Results:** Several deepfake detection architectures which are data-directed, lack appropriate explainability due to the characteristic of DNN models, defined as the black-box[9]. If a numerical score related to a video that was made by using a synthesis algorithm is not validated by proper reasoning, it becomes of no use for practitioners.
- **Temporal Aggregation:** The majority of extant deepfake detection algorithms seem to be focused on frame-level binary classification. This system comprises two major flaws. Firstly, regardless of the fact that many deepfake images exhibit temporal irregularities, the temporal stability of frames is not specifically evaluated. Secondly, we must integrate scores over each frame while calculating such scores.

1.2 Research Objectives

The aim of the study is to combine elements of several existing works into one framework in order to develop a detection methodology that can generate an equivalent or more efficient outcome than the state-of-the-art methodologies. Going through the several existing experiments we have drawn the conclusion that all the methodologies have some limitations in terms of performance or robustness. Therefore, our aim is to evaluate the proposed framework for all the existing experimental dimensions in order to generate a robust model that will not lag in performance.

Dynamic face augmentation is a technique for preprocessing a dataset using the face's landmark location value. The positioning of the ears, eyes, nose, jawline, and mouth are considered landmark points of the face. Dlib is a cutting-edge architecture that can recognize the 68 distinct landmark places on the face ranging from 0-to 67 that are utilized to compute the polygon for face cutout [6].

An automated deep learning technique called CNN, uses a grid like system to analyse the data which is derived from the architecture of the animal visual system [11]. Additionally adaptable, it can learn spatial hierarchies of attributes ranging from basic to complex patterns. The fundamental types of layers in CNN include convolution, pooling, and completely connected layers. The first two layers- convolution and pooling, locate and retrieve the attributes. These qualities are converted into end outputs, like categorization, via the third layer, a completely connected one. Convolution layer, a form of linear process, is one of the crucial parts of CNN, which is made up of a sequence of arithmetic functions like convolution. Extracted characteristics can gain hierarchical order and grow more complex as one layer transmits its result into the following. Additionally, training is the method of altering kernels and other parameters to create a connection between the labels on the ground truth

and the outcome. Gradient descent and backpropagation optimization methods are used to establish the connection [12].

An effective neural network training approach for interpreting sequence data is recurrent neural networks. Feedforward neural networks from where RNNs have been developed, act in the same way as human brains. Other algorithms are unable to predict sequence information in the same manner as recurrent neural networks do. An RNN's input is a loop that repeats itself. Before drawing conclusions, it assesses the present input and what it has acquired from earlier entries [13].

The development of a hybrid deep neural network-based model that can enhance the performance of recognizing deepfake content is the main goal of this research. Dynamic face augmentation, as well as CNN and RNN-based models, are the architectures we're attempting to integrate to achieve our goal. We'll preprocess the available datasets using Face-Cutout, a dynamic face augmentation so that they don't overfit the models. The preprocessed datasets will next be trained using deep neural models based on CNN and RNN.

1.3 Background Studies

The practice of photo manipulation emerged in the 19th century and was immediately applied to film. With the advent of digital video, technology developed steadily and more fast during the 20th century. Beginning in the 1990s, university researchers created deepfake technology, and subsequently, amateurs on internet forums. There are still other online communities where people share deepfakes of politicians, celebrities, and other people, such as those on Reddit like r/SFW deepfakes, which do not distribute pornography. On websites that do not restrict deepfake pornography, several other online groups nevertheless exchange porn. An exclusive desktop program named FakeApp was introduced in January 2018. With the help of this software, users may quickly make videos in which their faces have been switched. 2019 saw the demise of FakeApp in favor of open-source rivals like Faceswap, command-line-based DeepFaceLab, and web-based programs like DeepfakesWeb.com. Since then, DeepFake is becoming more and more popular. Eventually, ordinary people are creating deepfake material using open source tools and websites like FaceApp, DEEPFAKES web, ReFace, FaceSwap, and others. Deep neural networks were used to create all of these applications. The most prevalent architectures for DeepFake content creation have mostly utilized Generative Adversarial Networks (GANs) and AutoEncoders. The most common DeepFake image models are GDWCT, STARGAN, ATGAN, STYLEGAN, STYLEGAN2. On the other hand, the state-of-the-art for image classifications is EfficientNet, ResNeXt, XceptionNet, ResNet, DenseNet, etc. Using neural networks, those models have been developed with a view to classifying images.

The developed model architectures are usually used in terms of generating deepfakes and detecting deepfakes which are discussed briefly:

- **DeepFake Creation:**

One of the core components in deepfakes is machine learning, which has facilitated deepfakes to be generated at a lower cost with quicker speed. Some frequently used model architectures in generating deepfakes are:

- **GANs:** Generative Adversarial Networks (GANs), a subset of the generative modeling process, use deep learning techniques like Convolutional Neural Networks. The generator model and the discriminator model are two sub-models of GANs. The generator model is instructed to manufacture the latest occurrences and the discriminator model is instructed to recognize the samples' authenticity. There are a lot of variations where GANs architecture is used to create deepfake images [14]. It generates new data samples based on the dataset it is trained on.
- **StarGAN:** It is an updated version of the GANs architecture that basically learns mappings between multiple domains. It uses a unified modeling architecture that allows numerous datasets and domains to be trained simultaneously in a single network [15].
- **Autoencoders:** A form of neural network that acquires information from a compact representation of raw data, is defined as an Autoencoder. It is made of two sub-models: (i) the encoder model- which compresses the given data, and (ii) the decoder model- which attempts to rebuild the given data from the encoders' part. After training, the encoder model is kept and the decoder one is erased. Then, a machine learning model would be used here. Subsequently, the encoder would be used to retrieve features from the original data which would be considered as a data preparation method.

Autoencoder is made up of three primary parts- encoder, decoder, and code. The encoder and decoder are fully integrated to form a feed forwarding mesh, with the code acting as a single layer with its own dimension. There are some variations in autoencoders such as - Denoising autoencoders, Sparse autoencoders, and Variational autoencoders.

In Denoising autoencoders, some noisy data are integrated with the input image where the raw image cannot be copied because of noises in the raw image. With proper fit function and other computations, it generates better image quality. A Sparse Autoencoder is a sort of autoencoder that uses sparsity to achieve a limitation in information flow. Regularization is a Sparse autoencoder modification in which sparsity restrictions are regularized and certain extra terms of the loss function are introduced with the goal of activating particular regions of nodes throughout the layer and therefore assisting in the discovery of distinct properties in the input data. Another type of autoencoder is Variational autoencoder which is used in difficult scenarios to determine the odds of creating a distribution from the input data. The functional output of this autoencoder is obtained by a sampling strategy. Its design is similar to that of Regularized autoencoders[16].

- **Deepfake Detection:** There are several faults in the way deepfake videos are produced. These abnormalities can be utilized to take advantage of detecting deepfakes. Several deep learning architecture-based models have been developed, which are particularly effective in detecting deepfake contents. Some of them are EfficientNet B0 to B7, Xception, and ResNet50. These models require large amounts of datasets.

Convolutional neural networks are for image classification as it gives high accuracy. Also can be scaled up for better result. An EfficientNet is based upon CNN and scaling methods. EfficientNet scales up all the network width, depth, and resolution uniformly. There are several versions of EfficientNet B0 to B7.

ResNet, also known as the Residual Network, is one kind of Deep Neural Network. The vanishing gradient problem is a typical challenge in deep neural networks and ResNet was the first to establish the notion of a skip connection. It allows us to build layers without making things too complicated. As a result, CNN may now be used much more effectively. ResNet is available in several flavors. ResNet50 is a version with 50 layers, the majority of which are convolutional layers. In deep feature extraction, ResNet50 is often used.

Traditional convolutional layers can be made more efficient with the help of depth-wise separable convolution. Google developed the Xception neural network. It stands for Extreme Inception. It is made up of a modified depth-wise separation convolution, and it outperforms Inception-v3 in terms of performance. Due to the efficient use of model parameters Xception neural architecture is used in many different types of image classification.

A recurrent neural network is a sophisticated deep learning model that can learn from data sequences. It is capable of storing information from earlier input in memory. As a result, it can manage temporal sequences and extract information. Long short-term memory is a different kind of recurrent neural network that is usually utilized for understanding temporal sequences from frames (LSTM). An efficient representation of spatial and temporal sequences is provided by this recurrent neural network.

Chapter 2

Related Work

A multitude of deep learning architecture based models have been created as a result of technological developments. Among all the existing architectures, some are highly effective in detecting deepfake contents which are mentioned below:

- **Reverse Engineering-Based Framework:**

Authors Asnani et al. in [17] proposed a reverse engineering-based approach that

Authors Asnani et al. in [17] proposed a reverse engineering-based approach that would infer the hyperparameters involved in the generative process. The framework consisted of two different networks namely, a finger-print estimation network (FEN) and a parsing network(PN). The networks would combine to predict the hyperparameters that are specific to a particular architecture model. They claimed that the FEN may be further developed to conduct DeepFake detection with functionality comparable to cutting-edge techniques.

Guarnera et al [18] have mentioned in their paper another reverse engineering-based approach where they perform the expectation-maximization (EM) algorithm which was suggested by Moon et al. [19] to extract the feature vector of a synthesized image which corresponds to the kernel size of the generative model employed in the image generation process.

- **Deep Learning Based DeepFake Detection:**

David and Edward, the authors, proposed a two-stage CNN analysis. CNN was used to extract frame characteristics in this scenario. A recurrent neural network is then developed using these characteristics. To record the transient difference between frames induced by the face-swapping procedure, they used LSTM for temporal sequence analysis to extract features from a temporal aware RNN network. More than six hundred videos, both real and fake from the HOHA website [20] are used.

Author Jatin and Sahil created a deep learning-based DeepFake to evaluate if a photo is authentic or otherwise, by using a detection system that uses CNN and ResNet50. The classification of images also made use of the Sigmoid activation function. Their methodology is based on two datasets that contain 140k genuine and synthetic faces in total. The genuine dataset features 70k actual faces from Nvidia's Flickr collection, whereas the false dataset has 70k artificial faces selected from a pool of 1 million fake faces created using GAN[8].

- **Hybrid Models For DeepFake Detection:**

Author Pan et al. presented a strategy that included four datasets developed using four distinct deepfake technologies and has yielded high accuracy results across all four datasets. They also manufactured a mechanism that included all the detection methods that were used to cast votes depending on the images if they were fake or real, which made the accuracy rate higher [21].

In a paper presented by author Ismail et al., another method has been used which was a hybrid of YOLO, CNN, and XGBoost. Here, to retrieve the face part from the video frames, the YOLO face detector has been used, and the InceptionResNetV2 is utilized to retrieve attributes of these faces which are provided to method XGBoost. From a dataset that was constructed by combining CelebDF and FaceForensics++, a high accuracy rate was accomplished [22].

- **Face Augmentation For CNN-based models:**

A dynamic Face augmentation process has been developed by authors Sowmen Das et al. in [6] which is mainly focused on the underlying information of the faces. With CNN-based models, this augmented data is trained in order to get a better feature extraction by which the overfitting problem in the existing datasets can be solved. In this research, it is shown that the existing models of CNN-based architectures can provide a better result with the Face-Cutout process for all of the existing datasets.

- **Multi-attentional Network:**

Author Hanqing Zhao et al. in [23] developed a multi-attentional DeepFake Detection network. In this research, detecting the originality of multimedia is categorized under fine-grained classification problems. They sub-sectioned their research into three main components. Firstly, to focus on different local parts of the image while extracting features, they used multiple spatial attention heads. Secondly, to zoom in on the almost indistinct shallow features, they enhanced the textural feature block. Finally, they used the aggregation of semantic characteristics of high-level regions which are extracted from attentional maps with the textural characteristics of low-level regions. In this research, they decomposed the attention into multiple regions for the collection of local features for the deepfake detection task and they used the BAP [Bilinear Attention Pooling] instead of using Global Average Pooling and shallow features focused and enhanced.

- **GAN Anomalies Based Detection:**

Focusing on the abnormal patterns visible on the most structured parts of a synthesized image Giudice et al. in **13** proposed an algorithm where they could generate 8 X 8 blocks via a JPEG pipeline. The blocks are further processed via a Discrete Cosine Transformation (DCT) which generates AC coefficients for each block. A beta value is calculated using 0-Centered Laplace Distribution on the coefficients which are used to acquire the GAN Specific Frequencies (GSF). Their work is promising as it requires less computational resources with performance that exceeds the state-of-the-art techniques.

Chapter 3

Dataset And Preprocessing

3.1 Preprocessing

Deepfakes in recent years have increased dramatically due to the invention of several video synthesis technologies and video generation architectures such as generative models. Hence, the amount of dataset publicly available is not only large, but also robust. Despite the fact that the data available in any of such datasets is in the same video format, surprisingly each dataset has certain aspects that add to its uniqueness. The primary requirement of processing such data arises in order to mitigate such significant varying criterions and establish a baseline standard on which several models can be trained and validated. As the term suggests pre-processing refers to preparing the data in order to process it before feeding it to any model.

In order to process our videos for detecting real and synthesized frames, we have to begin by cropping videos into frames that contain only the ‘face’ of a candidate. As different videos may contain different angles and sizes of faces, hence each ‘cropped’ video is likely to have a different dimension than others. Since resolution of each video frame is crucial in containing information at a pixel level, having different varieties of resolution for the same task may complicate the training and evaluation of our model. To solve this issue, we resize each video frame into a resolution of 112 – which acts as a standard value for face-cropped image size. To further reduce the impact that different color channels may have due to having different pixel intensities, we adapt a normalization using a base standard deviation and mean. To reiterate, the videos have been cropped around the face with a dimension of 112 x 112 and standard normalization values of mean = (0.485, 0.456, 0.406) and standard deviation = (0.229, 0.224, 0.225) have been applied as transformations to normalize the color channels.

After applying the face crops and transformations a desired number of videos are selected for shifting into the data loader. The data loader is responsible for stacking the model with frames. It further allows us to shuffle the data, select a particular length of frames, choose a particular batch size and so on. The data is divided into two groups: a training set (80%) and a validation set (20%) before being passed to the data loader. It is also ensured that the number of real:fake videos is 50:50.

As shown in Figure:3.1, the steps are mentioned below for the dataset proprocessing.

1. Faces are detected using python’s ”face_recognition” library

2. Detected faces are cropped and stored in a separate module using python's in-built functions from python's computer vision library named Open-CV
3. Videos are splitted into training (80%) and validation (20%) set where each set has equal number of real and fake videos.
4. Videos are loaded into the data-loader with a specified batch size, shuffle value and frame length.

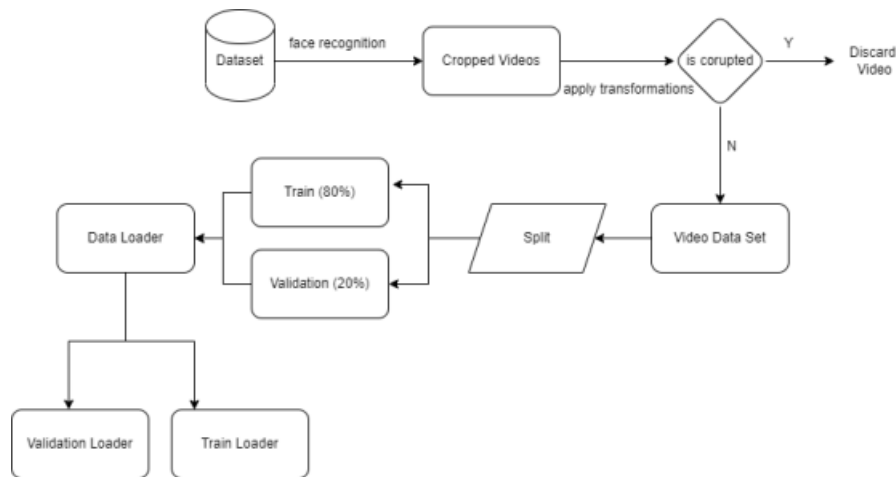


Figure 3.1: A Flowchart to showcase the steps for generating output through the model

3.2 Dataset

Forgery and video modification are not new anymore, but the usage of DNN has made the method of developing fake videos significantly more efficient, making them extremely difficult to detect. With the support of well-documented datasets, the deep fake identification method has improved. For different deep learning tasks, the utilization of a substantial, high-quality dataset is crucial so that it can minimize issues such as overfitting [24].

- **DFDC** One of the largest datasets is the DeepFake Detection Challenge (DFDC), which has more than 100000 clips produced by 3426 paid actors and utilizing various Deepfake, GAN-based, and other algorithms. This dataset includes footage of people in a range of inside and outside situations and different real lighting, without professional touch. It was created using two kinds of face-altering techniques that are unidentified and consists of 4113 fabricated videos that are of people from different genders, ethnicities, and generations. These videos are of individuals who consented to being recorded and shown alongside their modified photos in a dataset for machine learning. This dataset is also open to the public, which has made it easier to utilize [25].

- **Celeb-DF** In Celeb-DF, there are 590 videos that are original which were collected from Youtube. It includes video snippets of 59 celebrities varying from males to females, ethnicities, and generations, totaling 5639 DeepFake videos.
- **FF++** The collection consists of 70,000 PNG pictures at a dimension of 1024 by 1024. Regarding age, ethnicity, and picture background, it has a wide range. Since Flickr was used to crawl the photographs, all of that website's biases were carried across. A monument, a painting, or a photo of a photo was occasionally removed using Amazon Mechanical Turk.

Chapter 4

Proposed Deepfake Detection Architecture

4.1 Motivation

Video and image synthesis, regularization, segmentation and localization problems and methods of solving them are gaining popularity gradually ever since the inception of Convolutional networks. However, due to the advent of a wide variety of vision transformers, computer vision tasks are now facing a drastic shift towards a new genre of architecture for similar problems that were previously solved using convolutional networks. The emergence of the ConvNeXt architecture was initiated to modernize the convolutional networks to the extent where they can be equivalent to their counterparts; transformers in terms of performance. Deepfake video detection problem is similarly a widely known area which incorporates several image synthesis based anomalies that integrated and detected at the core pixel level of an image. The core drive for executing this study was to analyze the performance of the ConvNeXt architecture and determine whether it is capable of adapting to solve a robust and complex problem such as determining whether a video is real or not.

4.2 Proposed deepfake detection framework

We create a straightforward and clear design for the overall model in order to provide a model that would primarily focus on the performance metrics of the backbone architecture, in our case the ConvNeXt convolutional network. Figure 4.1. depicts how our suggested model is represented. According to the figure, the input is passed to the core architecture; which extracts a feature map from the provided data. This feature map is passed through an adaptive average pooling layer and further processed by a Recurrent Neural network; LSTM. The final layer is a simple fully connected layer responsible to generate the required number of output classes.

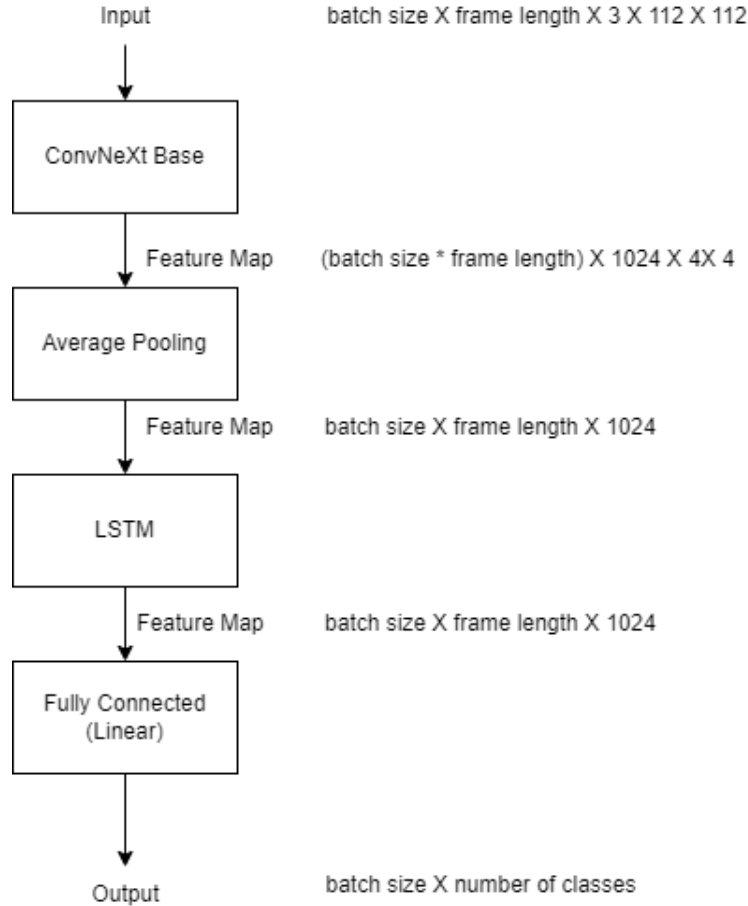


Figure 4.1: A block diagram displaying the generic architecture of the proposed model

4.3 Short Description Of Proposed Model and Architectures Used For Comparison and Composition

The baseline model’s core architectures and the core architectures utilized for comparative models are briefly introduced in this section. For a thorough analysis in understanding the performance comparison, we shift the backbone architecture used for extracting features from the frames while maintaining generic base composition for the entire model which consists of the following components: Backbone (VGG16, ConvNeXt, Swin Base), LSTM, Linear Layer, Average Adaptive Pool layer. While executing training and validation process an additional layer of dropout has been implemented.

- **VGG16:** VGG-Net is mainly categorized into 4 versions namely VGG11, VGG13, VGG16, VGG19 where the numbers 11, 13, 16, 19 represent the total number of convolutional layers and fully connected layers used in the model respectively. With an objective to improve AlexNet the implementation of VGG16 composed a significant change by reducing the size of the kernel. Due to having a dimension of 3 x 3 in the filters alongside a stride value 1 followed by a rectified linear unit to introduce non-linearity the complication

of computation is widely reduced. The size for the max pool layers is 2×2 with a stride value of 2.

- **ConvNeXt:** Similar to VGG-net and many modern architectures proposed, ConvNeXt also has its own set of variants namely ConvNeXt-T, ConvNeXt-S, ConvNeXt-B, ConvNeXt-L, ConvNeXt-XL where each variant differs in the number of channels which are respectively as follow: 96, 96, 128, 192, 256. The key idea that incurred the inception of this architecture was the design decisions that impacted the performance of ConvNets in transformers. By replacing the stem cell of the ResNet backbone with a patchify layer implemented using 4×4 ConvNeXt replaces ResNet-style stem cell with a patchify layer composed of a 4×4 convolution layer and a stride of 4[26]. An advancement in the network width from 64 to 96 while keeping the same channel number as that of swin-t, ConvNeXt manages to enhance its performance in detection-based tasks.
- **Swin Transformer:** ‘Swin’ is mainly elaborated into the terms “Shifted Windows” and is a vision transformer variant with a hierarchical way of processing images. Due to having a quadratic complexity with respect to the image size ViT tends to lack efficiency in scaling when working with high resolution images, hence, the inception of Swin transformers. Therefore, to mitigate the constraints of ViTs, Swin introduced to concepts, namely hierarchical feature maps which paved the requirement fine-grained prediction and shifted windows which allowed self-attention to be spanned in a crossed window fashion. Each Swin block consist of 2 sub units each having a normalization layer, attention module, normalization layer and multi-layer perceptron layer respectively. Window-MSA is used by the first sub-unit, in contrast Shifted Window-MSA module is integrated in the second sub-unit [27].
- **LSTM:** LSTM is one of the variations of Recurrent neural networks (RNNs), that works with long-term dependencies, particularly for sequence prediction [28]. LSTM contains feedback connections as well as the ability to process the single data points and complete sequence of input data. In LSTM, one of the key aspects is the cell state, which functions primarily as a conveyor belt. It travels the entire system, including some marginal linear operations. LSTMs employ a number of ‘gates’ that regulate the way of taking input in a data sequence, maintaining that information and leaving the system. A typical LSTM contains three gates: a forget gate, an input gate, and an output gate. These gates mainly function as filters, having their own network [29].
- **Average Adaptive Pool, Linear Layer and Dropout:** In order to synchronize the output dimensionality in correspondence with the provided input, the adaptive average pool layer is responsible in determining the kernel size. In order to do so, a mean of the pooling layer is computed in order to enhance the search space [30]. A linear layer simply consists of weights and biases that reinforces the dimensionality to our requirements. The inclusion of a dropout ensures that the model is forced to learn the relationships with respect to image and labels by randomly dropping weights with a desired probability.

4.4 Model Details

4.4.1 ConvNeXt:

The vision to introduce ConvNeXt architecture was to undertake the task of enhancing the existing ResNet models incrementally to establish a model which can equate to any hierarchical vision transformer. Hence the architecture incorporates several design features from both.

A generic stem cell of the ResNet model is composed of a 7×7 convolution layer having a stride value of 2, followed by a max pool, which results in a 4×4 down-sampling of the input images. As mentioned previously, Liu et al in [4] proposes a ‘patchify layer’ to replace the stem cell which consist of a 4×4 convolution layer having stride of 4. Figure 4.2 portrays that depth wise convolution can be utilized further as an improvement of the grouped convolutions, where the depth of refers to the numeric value of channel size.

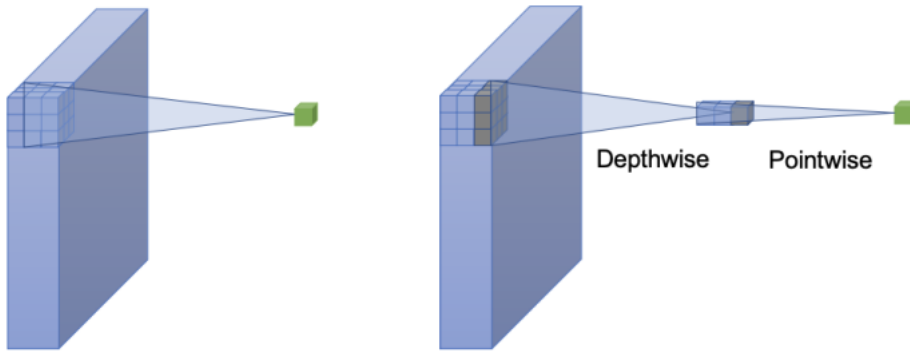


Figure 4.2: Standard, Depth-wise and Point-Wise Convolution

The channel number in this case is 96, which additionally brings the network performance to an increase along with an increase in FLOPs which is another metric for evaluating a networks performance. Further investigating the inverted bottleneck concept where the dimensions of the hidden layer of MLP is 4 times wider in comparison to the input dimension and by introducing this bottleneck to the base design, the network FLOP decreased significantly with a slight performance increase. Figure 4.3 (a) and (b) showcases this in block diagrams.

The depth-wise convolution blocks are further shifted upwards in order to incorporate a larger kernel of size 7×7 . In the micro design level, a Gaussian Error Linear Unit or GELU is introduced along with a LayerNorm for normalization instead of BatchNorm. This activation function is considered an enhancement of its predecessor Leaky ReLU for being used in vision transformers. The standard ConvNeXt block is illustrated as shown in Figure: 4.4. Finally for down sampling, a separate 2×2 conv layer with stride 2 is added into the layer, which is a familiar strategy found in hierarchical transformers (eg: Swin).

ConvNeXt have different variants based on the number of channels C and the number of blocks B in each stage. Below is a list of the variants and the corresponding channel and block values.

- ConvNeXt-T: $C = (96, 192, 384, 768)$, $B = (3, 3, 9, 3)$

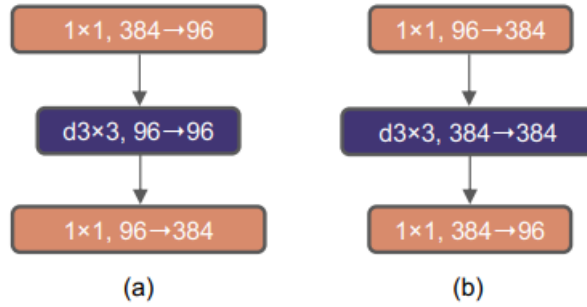


Figure 4.3: Block Modifications (a) Standard ResNeXt Block (b) Inverted Bottleneck Implementation

ConvNeXt Block

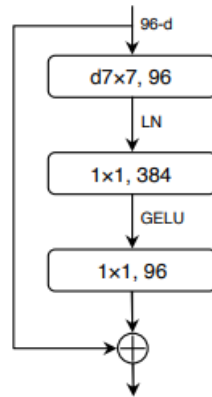


Figure 4.4: Block Design of ConvNeXt From

- ConvNeXt-S: $C = (96, 192, 384, 768)$, $B = (3, 3, 27, 3)$
- ConvNeXt-B: $C = (128, 256, 512, 1024)$, $B = (3, 3, 27, 3)$
- ConvNeXt-L: $C = (192, 384, 768, 1536)$, $B = (3, 3, 27, 3)$
- ConvNeXt-XL: $C = (256, 512, 1024, 2048)$, $B = (3, 3, 27, 3)$

4.4.2 LSTM

Long Short Term Memory networks, commonly known as "LSTMs," is a variation of RNN that usually overcome long-term dependencies [31]. Instead of one single neural network layer, LSTM has four layers that are combined in a unique way to prevent the problem of long-term dependency shown in Figure 4.5.

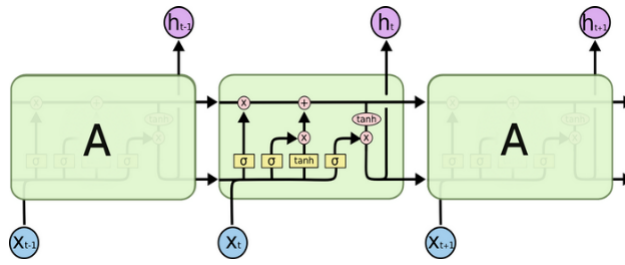


Figure 4.5: LSTM Dataflow

One of the fundamental elements, the cell state that is shown by the horizontal line all the way, basically operating like a conveyor belt. As illustrated in Figure 4.6, it traverses straight through the entire system, with some marginal linear computations.

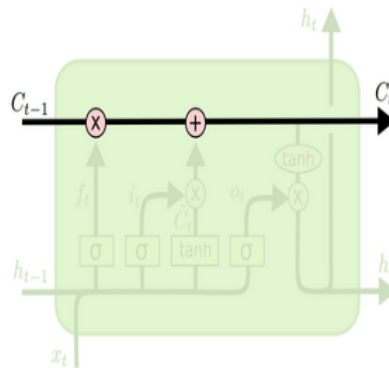


Figure 4.6: Cell State

The LSTM can erase as well as add new information to the cell state, which is controlled by gates, shown in Figure 4.7. Gates works like a method of selectively allowing information to pass through. The sigmoid layer produces numbers between 0(not pass datas) and 1(pass datas). LSTM generally has three gates including forget gate, input gate and output gate.

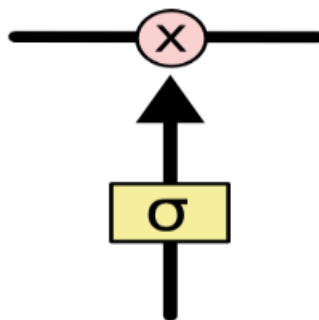
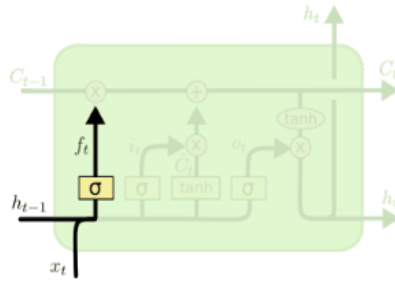


Figure 4.7: Gate Layer

- Forget Gate: The first stage of LSTM is determining which information will be discarded from the cell state which is done by the "forget gate key,". Forget layer is basically a sigmoid layer which examines h_{t-1} and x_t and produces a

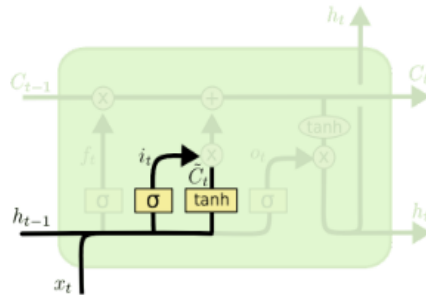
value between 0 and 1 for every value in the cell state . Figure 4.8 describes the forget layer.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (4.1)$$

Figure 4.8: Forget Gate.

- Input Gate: The stage is to determine which additional information will be recorded in the cell state which includes two parts. Initially, a sigmoid layer known as the "input gate layer" determines the information to be updated. A tanh layer is then used to generate a vector containing new values, \tilde{C}_t to add in the state. In the following step, these two will be combined for updating the state, demonstrated in Figure 4.9.



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4.2)$$

Figure 4.9: Input Layer, Sigmoid and tanh operation.

Candidate Memory Cell equation is as followed:

$$\tilde{C}_t = \mathbf{tanh}(W_C \cdot [h_{t-1}, x_t] + b_c) \quad (4.3)$$

Now, the values are upgraded from the previous cell state, C_{t-1} , to the new cell state, C_t by carrying out the preceding steps. The previous state is multiplied by, f_t forgetting the decided values. Then we make additions to $i_t \otimes \tilde{C}_t$. This is the updated set of candidate values, as shown in Figure 4.10, calculated from the amount of each state value was updated.

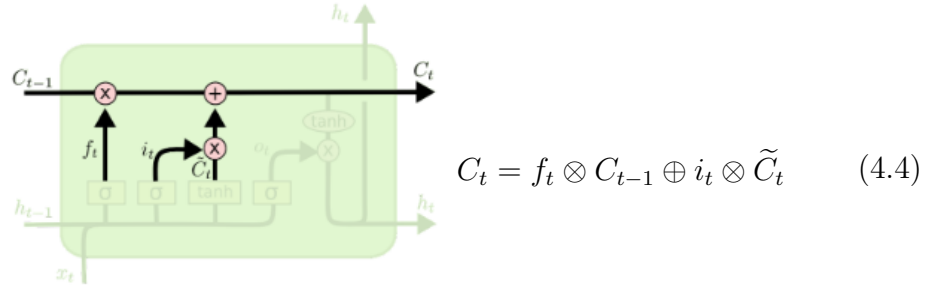


Figure 4.10: Updated Value.

- Output Gate: The final state is to determine the output which will be filtered, depending on the cell state. For this, a sigmoid layer is executed to determine which bits of cell state will be output. After that, tanh layer will be performing on the cell state for generating the values between 0 and 1. Finally, a multiplication will take place with this output and the sigmoid value and eventually generate the output shown in the Figure 4.11.

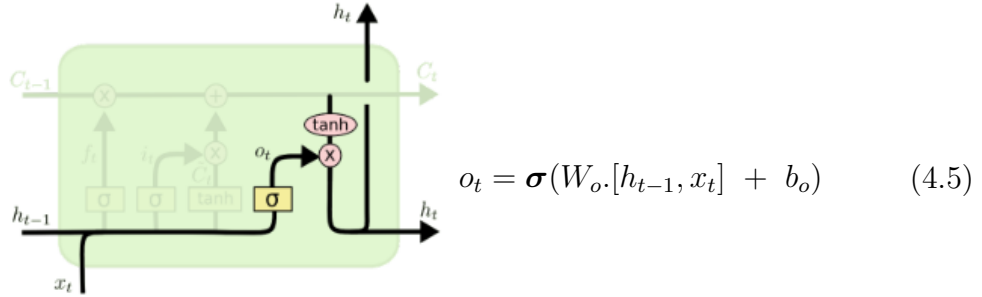


Figure 4.11: Output Layer.

Hidden State equation is as followed:

$$h_t = o_t \otimes \tanh(C_t) \quad (4.6)$$

The value of h_t from equation 4.6 is the output of the current timestamp. This value is passed to the next LSTM block at the timestamp $t + 1$.

Chapter 5

Experiments And Results

5.1 Parameter Settings

Experimental setups require an extensive tuning of parameters and hyperparameters that can dramatically influence performance. This can often lead to black-box testing where the output lacks the explain ability for a given input in case of a significant drop of rise in accuracy, but in many cases the parameter tuning is treated as the “cause” and the accuracy or likewise the performance achieved is considered as the “result”.

Environmental Set Up

In order to observe the true potential of the proposed models architecture we had to run a couple of adjustments.

1. Dropped the Last 2 layers from the ‘backbone’ architecture Essentially the last layer is a softmax function which has been implemented to predict classes equivalent to the size of the IMAGENet version-1 (which is 1000). The layer before this ensures that the proper dimension of output is processed into the softmax layer for the proper predictions.
2. Varying the number of Latent dimensions in LSTM Due to dropping off the last layers we lose the common interface for binding the output of our backbone architecture and passing the output forward to another layer. Hence, we need to adjust the latent dimensions of the LSTM layers in order to maintain synchronization in the dimensionality and downsampling of the input which is demonstrated in Table 5.1

Model	Number of LSTM Blocks	Number of layers per LSTM
VGG16 2 512	2	512
Swin base	1	1024
ConvNeXt	1	1024

Table 5.1: LSTM Blocks and Layers in each architecture

The number of batches we chose for our purpose is 4. We set the initial epoch size to 20 for both validation and training purposes. A standard learning rate of 0.00001 and a drop out of 0.6 has been used. The backbone architectures are all pretrained

on the IMAGENET Version 1 dataset, and we shall proceed with the pretrained values. Table 5.2 illustrates the number of parameters for each pre-trained model

Model	Pre-Trained on IMAGENET	Number of Parameters
VGG16	True	138.4 million
Swin base	True	87.8 million
ConvNeXt	True	88.6 million

Table 5.2: Number of parameters in each architecture

Due to time and space constraints we have chosen an optimal number of videos for each of the datasets, namely DFDC, FF++ and Celeb-DF, instead of using the entire dataset available in our disposal. The number of videos selected for each dataset is illustrated in Table 5.3.

Dataset	Number of Videos Present	Number of Videos used
DFDC	100,000	2,550
FF++	-	1,990
Celeb-DF	5,639	1,170

Table 5.3: Number of Videos in each architectures

5.2 Evaluation Matrices

5.2.1 Adam

Adam is one of the most popular algorithms that is highly used in deep learning because of the efficient and robust optimization. This optimization is enabled by incorporating the following features:

Starting with the computation of optimization problems using stochastic gradient descent. Additionally, the vectorization of minibatch stochastic gradient descent, which uses bigger sets of observations in a single minibatch, massively improves efficiency. In order to expedite convergence, Momentum included a system for compiling a collection of previous gradients. Adagrad made use of per-coordinate scaling to enable a preconditioner that was computationally effective. Per-coordinate scaling and a learning rate adjustment were separated by RMSProp.

Leaky averaging is one of the major features of Adam which is used for estimating both momentum as well as second moment of the gradient by employing exponentially weighted moving averages and state variables are:

$$\begin{aligned} v_t &\leftarrow \beta_1 v_{t-1} + (1 - \beta_1) g_t, \\ s_t &\leftarrow \beta_2 s_{t-1} + (1 - \beta_2) g_t^2. \end{aligned}$$

The weighted variables β_1 and β_2 are nonnegative. They frequently choose from $\beta_1 = 0.99$ and $\beta_2 = 0.999$. The variance estimation therefore grows even more slowly

than the momentum term. While initializing $v_0 = s_0 = 0$, there is a substantial bias towards lower values. The fact $\sum_{i=0}^t \beta^i = \frac{1 - \beta^{t+1}}{1 - \beta}$ that can be used to re-normalize terms in order to address this. The normalized state variables:

$$\hat{v}_t = \frac{v_t}{1 - \beta_1^t} \text{ and } \hat{s}_t = \frac{s_t}{1 - \beta_2^t}$$

At the beginning, the gradient must be rescaled in a way that is quite similar to RMSProp's, to get

$$g'_t = \frac{\eta \hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon}$$

In contrast to RMSProp, the update makes use of the momentum x instead of the gradient directly. Additionally, while scaling, the usage of $\frac{1}{\sqrt{\hat{s}_t} + \epsilon}$ instead of $\frac{1}{\sqrt{\hat{s}_t} + \epsilon}$ causes a slight change. We commonly choose $\epsilon = 10^{-6}$ because it offers a good balance among fidelity and numerical stability. The updated form:

$$x_t \leftarrow x_{t-1} - g'_t.$$

In state variables, the momentum and scale are very apparent. The terms are redefined because of their definition which can be updated by changing update condition and initialization. Furthermore, the combination of terms is quite simple and straightforward. Lastly, we may regulate the step length to resolve convergence-related problems by explicit learning rate, η .

5.3 Performance Evaluation Matrices

5.3.1 Cross Entropy Loss

The cross entropy loss among the input and the target is measured. It is essential in terms of training a classification model with C classes [32]. The alternative parameter weight needs to be a 1D tensor that provides each class a certain amount of weight. This is especially helpful if your training set is not balanced properly. The input is anticipated to include each class's unfiltered or raw and unnormalized score value. A Tensor of size $(C)(C)$ for unbatched input or $(minibatch, C)$ or $(minibatch, C, d_1, d_2, \dots, d_k)$ with $K \leq 1$ is required for the K -dimensional case. An additional benefit of it is for inputs with greater dimensions, including such determining cross entropy loss per-pixel for 2D images. The cross entropy loss includes the following:

Class generally falls within the $[0, C][0, C]$ range, where C presents the number of classes; in case of a fixed or constant index, this loss automatically recognizes this class index that may not fall within class range. When reduction set to "none," often addressed as unreduced loss is:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^T, \quad l_n = w_{y_n} \log \frac{\exp(x_n, y_n)}{\sum_{c=1}^C \exp(x_n, c)} \cdot 1 \{y_n \neq ignore_index\}$$

Here, x , y , C and w work as the input, the target, number of classes and weight respectively. N encompasses minibatch dimension and d_1, \dots, d_k for the K -dimensional

case. When reduction is not labeled as 'none'(default'mean').

For every single class, probabilities are necessary where more than one class of labels are needed for a minibatch item for smoothing the labels, blending the labels etc. When reduction is not labeled as 'none' (default 'mean'),

Here, x , y , C and w work as the input, the target, number of classes and weight respectively. N encompasses minibatch dimension and d_1, \dots, d_k for the K -dimensional case. When reduction is not labeled as 'none' (default 'mean'),

5.4 Result

After running the models for 20 epochs, we have recorded the mean train and validation accuracies along with the maximum train and validation accuracies. For comparing loss we are primarily focusing on the validation loss. This is illustrated in Table 5.4 . The first column represents the model, the second column showcases the Datasets used for each model, in our case for each model the second and all other columns will have three rows. To isolate the results for ConvNeXt we keep the model and corresponding records at the final row of the first column. As shown in Table 5.4 the lowest mean loss is 0.231 for FF++, highest mean training accuracy 99.507 and maximum training accuracy 99.920 is achieved for Celeb-DF while highest mean validation accuracy 95.226 and maximum validation accuracy 96.231 is achieved for FF++.

Model	Dataset	Mean Loss	Mean Train Accuracy	Maximum Train Accuracy	Mean Validation Accuracy	Maximum Validation Accuracy
Swin Base	DFDC	0.396	88.989	97.369	83.911	90.465
	FF++	0.406	87.716	97.056	82.998	92.720
	Celeb-DF	0.487	80.258	97.644	76.135	92.170
VGG16	DFDC	0.376	90.389	98.166	83.750	89.466
	FF++	0.424	89.327	99.436	84.573	92.211
	Celeb-DF	0.427	81.204	99.679	77.970	92.308
ConvNeXt	DFDC	0.281	90.321	98.912	89.771	97.613
	FF++	0.231	99.484	99.874	95.226	96.231
	Celeb-DF	0.438	99.507	99.920	90.811	93.162

Table 5.4: Performance comparison on each dataset.

5.4.1 Graphs On Loss function Calculation

Loss Function and Accuracy on Swin Transformer

Loss Function Value

While plotting the curves for showcasing the loss functions behavior; in Figure 5.1(a) and in Figure 5.1(c) it can be noticed that the loss function has a quicker convergence rate which is under 10 epochs for DFDC and FF++ respectively. The validation and training loss for DFDC have slightly larger fluctuations in comparison to FF++. The loss curve has a comparatively slower convergence rate as for Celeb-DF as shown in Figure 5.1(b) where the validation loss curve has a peak nearly at the tenth epoch.

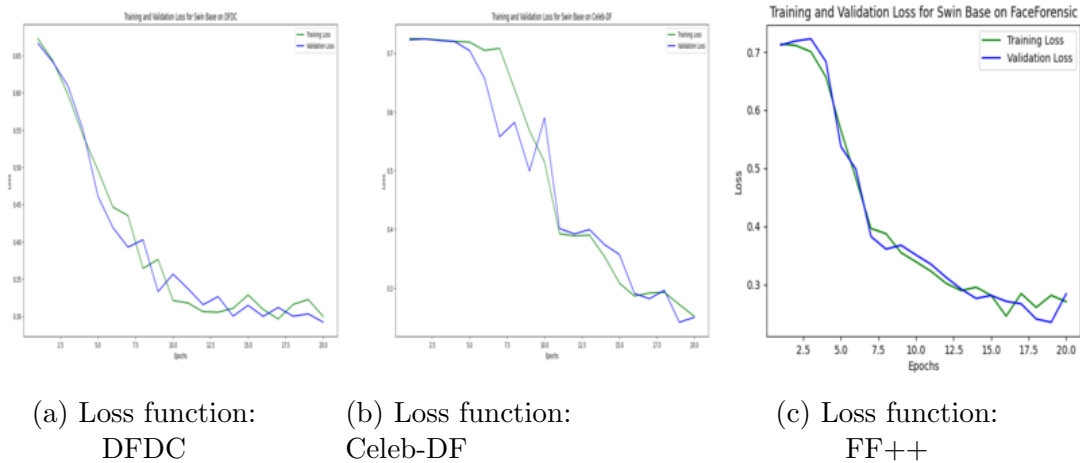
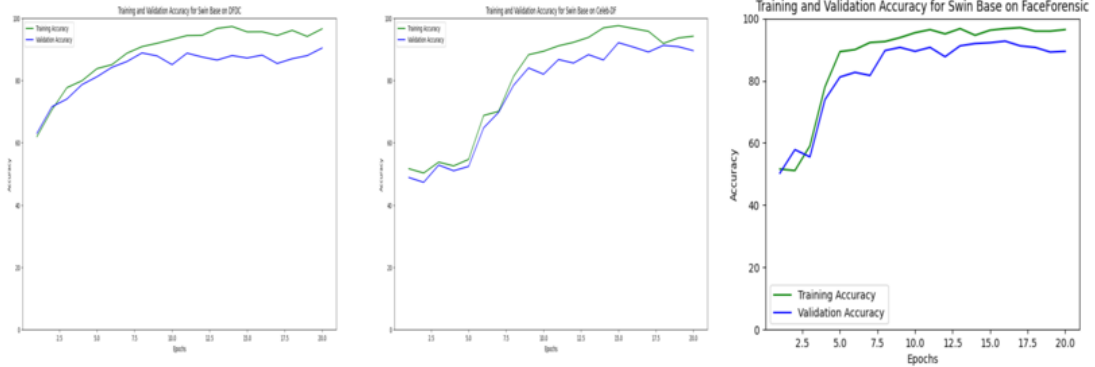


Figure 5.1: Loss Function: Swin Transformer architecture

Accuracy

As illustrated in Figure 5.2(a) and (c) the both validation and training curves are increasing till the fifth epoch and have an almost parallel linear shape up to the twentieth epoch for the Swin on the datasets DFDC and FF++ respectively. The curve for Celeb-DF shows a similar trend of increase for both validation and training curves till the tenth epoch and is almost slightly fluctuating thereonwards, which is shown in Figure 5.2(b).



(a) Accuracy:
DFDC

(b) Accuracy:
Celeb-DF

(c) Accuracy:
FF++

Figure 5.2: Accuracy: Swin Transformer architecture

Loss Function and Accuracy on VGG16: Loss Function Value

A similar trend of convergence rate is visible for VGG16 as illustrated in Figure 5.3(a) and 5.3(b) for DFCC and FF++ respectively. In both cases, the curves are at a steady decline from the tenth epoch. The decline for the Celeb-DF loss curve is slightly imbalanced and the variation between the validation and train curve is slightly greater in comparison to the previous ones as shown in Figure 5.3(b)



(a) Loss function:
DFDC

(b) Loss function:
Celeb-DF

(c) Loss function:
FF++

Figure 5.3: Loss Function: VGG16 architecture

Accuracy

The accuracy and validation curves for VGG16 on the datasets DFDC and FF++ are increasing till the seventh epoch and are almost linear and parallel in shape with slight fluctuations, which is shown in Figure 5.4(a) and (c) respectively. The validation and accuracy of DFDC for VGG16 are increasing till almost the tenth epoch as demonstrated in Figure 5.4(b), however post epoch ten, they still have increasing tendencies with a small slope.

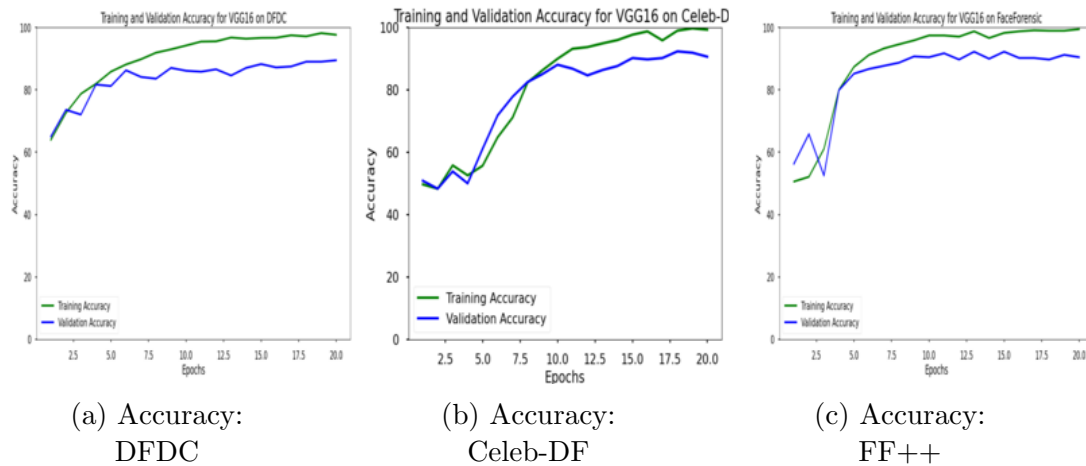


Figure 5.4: Accuracy: VGG16 architecture

Loss Function and Accuracy on ConvNeXt:

Loss Function Value

For ConvNeXt the loss function shown in Figure 5.5(b) training loss is an almost linear curve with a slight peak at the and the validation curve is another linear curve which is almost parallel to the train curve with a peak around the second epoch. As illustrated in Figure 5.5(c) the training and validation curves decline quickly under five epochs and the remaining part of the curve for training is stable whereas the curves for validation loss is slightly increasing. Finally, the loss curve for DFDC as shown in Figure 5.5(a) is decreasing; however the validation curve has several minima and maxima and can be interpreted as an increasing curve.

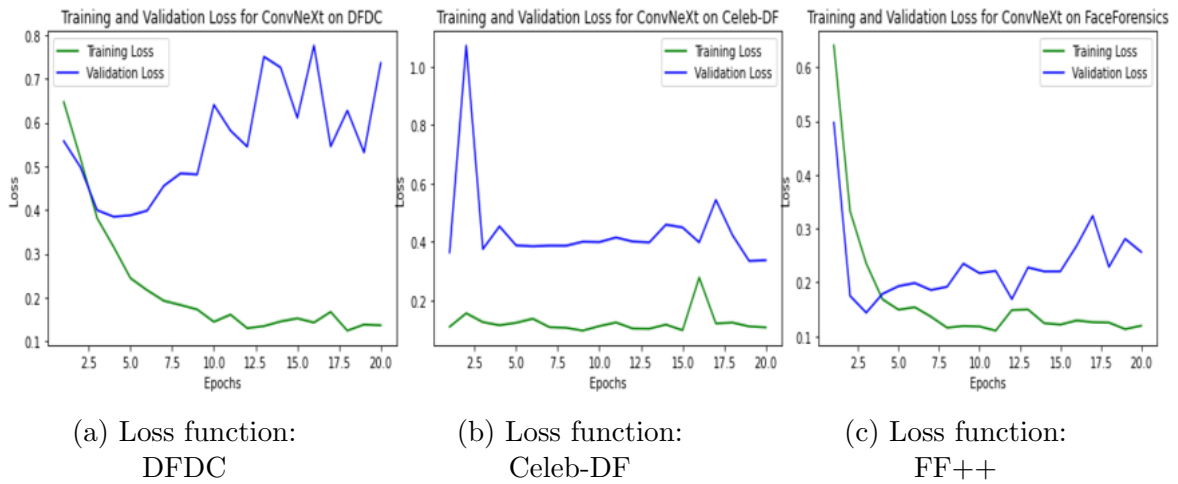


Figure 5.5: Loss Function: ConvNeXt architecture

Accuracy

In the case of ConvNeXt, both training and validation curves are increasing till the fourth epoch and are almost parallel straight lines with high accuracies till the final epoch as shown in Figure 5.6(a) and (c) respectively for DFDC and FF++. With slight declines in the beginning around the third to fourth epoch and in the fifteenth to seventeenth, the remaining curves are almost parallel straight lines for Celeb-DF as illustrated in Figure 5.6(b)

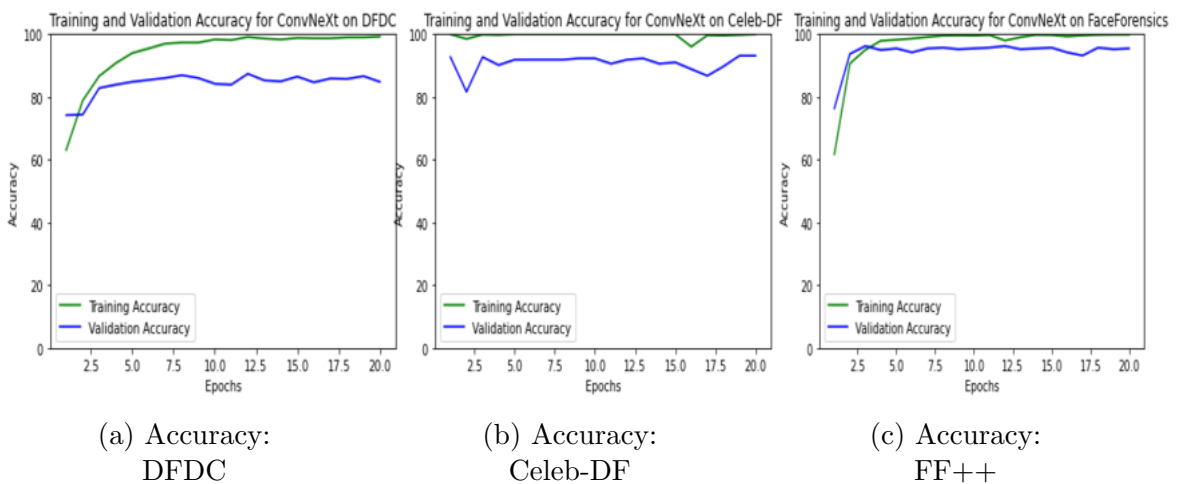


Figure 5.6: Accuracy: ConvNeXt architecture

Chapter 6

Conclusion

DeepFakes is creating an alarming situation for the trustworthiness of online information. Using different components of the most recent and effective deepfake detection or image classifier algorithms, we are approaching the creation of a system that will be able to find out whether multimedia content is real or forged. Deepfake detection architectures that have been trained on a variety of datasets have difficulty adapting their performance to multiple datasets. Therefore, this research aims to prove by experimental analysis that our proposed model is better than recent computer vision architecture in terms of detecting forged multimedia content combining CNN and RNN which are based on neural networks architecture.

6.1 Limitations

Deep learning experiments often require several layers of set up that include modulating data, sanitization of data, scheduling of several parameters that can affect performance of the architecture, customization of the architectural layers and so on. Device performance, environment configurations, speed of resource access are some regular constraints that may affect results acquired and may have subtle variations; which may range from minor to major differences depending on the use cases. As we have strictly focused on introducing the ‘evaluation of performance for our proposed’ architecture and showcase how it can be equivalent or superior to the recently developed models that are being widely used for similar tasks, we have inadvertently omitted several aspects related to experimental results. Below we showcase some essential factors that can be potential causes of refutation for our work.

- **Parameter Tuning** In order to derive and showcase the best results, our findings on parameter data have led to less exciting values that could not be represented that caused us to proceed with preset standard values. Additionally, in case of experimental derivation of values, the performance on each individual model on each individual dataset has been discarded; instead the desired value was chosen based on the performance on a significant number of videos and was later used for all other datasets. Many deep learning tasks also include scheduling of learning rate, in order to fine tune convergence, however, skipping such mechanisms was opted as these may lead to further computational complexity; hence preferring less complexity over efficiency of result values.

- **Hardware Constraints** The experiments were executed on google colab-
ratory notebooks, popularly known as “Google Colab”. The notebooks are
hosted on google cloud servers and can only be run with a specified allocated
disk space of 357.27 GB and RAM of 12 GB. Though as a freely available re-
source, the amount of memory and storage provided is dramatically large, yet
for our purpose the size of the dataset and the number of data being loaded
at times exceeds the provided disk space. In addition to this, the GPU access
is limited to 12 hour per day for a single account and for training and vali-
dating models it requires almost one-third of that duration which means that
computation wise lesser time for evaluating and comparing models.

6.2 Future Work

Considering this work as an initial set up for providing a guideline on the perfor-
mance of the proposed ‘backbone convolutional network’, experimenting with the
core architecture model can be proceeded further. A primary follow-up to our cur-
rent work will include an expansion of the standard base architecture and modifica-
tion of its internal layers and experimentation with several hyperparameter tuning
and inclusion of a variety of sequence processing segments. Encompassing several
pretrained models that can fuse with the existing architecture is also a lucrative
thought to work upon. As showcased in our experimental results, hardware con-
straints have been one of the primary obstacles to make the best use of the total
number of data available. Expanding the currently used dataset and finding opti-
mum results for a large amount of data will provide insights on not only performance
in terms of accuracy but also in terms of scalability.

Bibliography

- [1] D. Biswas, “Most shocking deepfake videos of 2021,” *Analytics*, 2021.
- [2] S. Lyu, “Deepfake detection: Current challenges and next steps,” in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, London, UK: IEEE, Jul. 2020.
- [3] E. Strickland, *Facebook AI launches its deepfake detection challenge*, en, <https://spectrum.ieee.org/facebook-ai-launches-its-deepfake-detection-challenge>, Accessed: 2022-9-20, Dec. 2019.
- [4] *Blog*, en, <https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html>, Accessed: 2022-9-20.
- [5] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, *et al.*, “Deep learning for deepfakes creation and detection: A survey,” en, *Comput. Vis. Image Underst.*, vol. 223, no. 103525, p. 103525, Oct. 2022.
- [6] S. Das, S. Seferbekov, A. Datta, M. S. Islam, and M. R. Amin, “Towards solving the DeepFake problem : An analysis on improving DeepFake detection using dynamic face augmentation,” Feb. 2021. arXiv: 2102.09603 [cs.CV].
- [7] S. Singh, R. Sharma, and A. F. Smeaton, “Using GANs to synthesise minimum training data for deepfake generation,” Nov. 2020. arXiv: 2011.05421 [cs.CV].
- [8] J. Sharma and S. Sharma, “Challenges and solutions in DeepFakes,” Sep. 2021. arXiv: 2109.05397 [cs.CV].
- [9] S. Lyu, “Deepfake detection: Current challenges and next steps,” in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, London, UK: IEEE, Jul. 2020.
- [10] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, “Celeb-DF: A large-scale challenging dataset for DeepFake forensics,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020.
- [11] D. H. Hubel and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex,” en, *J. Physiol.*, vol. 195, no. 1, pp. 215–243, Mar. 1968.
- [12] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” en, *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018.
- [13] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” en, *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018.

- [14] I. Papastratis, *Deepfakes: Face synthesis with GANs and autoencoders*, en, <https://theaisummer.com/deepfakes/>, Accessed: 2022-9-20, Jun. 2020.
- [15] J. Brownlee, *A Gentle Introduction to Generative Adversarial Networks*. 2019.
- [16] P. Pedamkar, *Autoencoders*, en, <https://www.educba.com/autoencoders/>, Accessed: 2022-9-20, Oct. 2019.
- [17] V. Asnani, X. Yin, T. Hassner, and X. Liu, “Reverse engineering of generative models: Inferring model hyperparameters from generated images,” Jun. 2021. arXiv: 2106.07873 [cs.CV].
- [18] L. Guarnera, O. Giudice, and S. Battiato, “DeepFake detection by analyzing convolutional traces,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA: IEEE, Jun. 2020.
- [19] Y.-Y. Kim, H. Kim, W. Lee, H.-L. Choi, and I.-C. Moon, “Black-box expectation-maximization algorithm for estimating latent states of high-speed vehicles,” en, *J. Aerosp. Comput. Inf. Commun.*, vol. 18, no. 4, pp. 175–192, Apr. 2021.
- [20] D. Guera and E. J. Delp, “Deepfake video detection using recurrent neural networks,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Auckland, New Zealand: IEEE, Nov. 2018.
- [21] D. Suo, S. Zhang, Z. Song, S. Wang, Y. Li, and X. Fan, “Simultaneous determination of 21 sulfonamides in poultry eggs using ionic liquid-modified molecularly imprinted polymer SPE and UPLC-MS/MS,” en, *Molecules*, vol. 27, no. 15, p. 4953, Aug. 2022.
- [22] D. Pan, L. Sun, R. Wang, X. Zhang, and R. O. Sinnott, “Deepfake detection through deep learning,” in *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, Leicester, UK: IEEE, Dec. 2020.
- [23] H. Zhao, T. Wei, W. Zhou, W. Zhang, D. Chen, and N. Yu, “Multi-attentional deepfake detection,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021.
- [24] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, “Celeb-DF: A large-scale challenging dataset for DeepFake forensics,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA: IEEE, Jun. 2020.
- [25] O. de Lima, S. Franklin, S. Basu, B. Karwoski, and A. George, “Deepfake detection using spatiotemporal convolutional networks,” Jun. 2020. arXiv: 2006.14749 [cs.CV].
- [26] A. Singh, *ConvNext: The return of convolution networks - augmented startups - medium*, en, <https://medium.com/augmented-startups/convnext-the-return-of-convolution-networks-e70cbe8dabcc>, Accessed: 2022-9-20, Feb. 2022.
- [27] J. Loy, *A comprehensive guide to microsoft’s swin transformer*, en, <https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c>, Accessed: 2022-9-20, May 2022.
- [28] *What is LSTM - introduction to long short term memory*, en, <https://intellipaat.com/blog/what-is-lstm/>, Accessed: 2022-9-20, Feb. 2022.

- [29] R. Dolphin, *LSTM networks*, en, <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9>, Accessed: 2022-9-20, Oct. 2020.
- [30] G. J. van Wyk and A. S. Bosman, “Evolutionary neural architecture search for image restoration,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary: IEEE, Jul. 2019.
- [31] *Understanding LSTM networks*, en, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>, Accessed: 2022-9-20.
- [32] *CrossEntropyLoss — PyTorch 1.12 documentation*, en, <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>, Accessed: 2022-9-20.