Addis Ababa University

School of Graduate Studies

College of Natural Science

Department of Computer Science

Deepfake Video Detection Using Convolutional Vision Transformer

Deressa Wodajo Deressa

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

November, 2020

Addis Ababa University

School of Graduate Studies

College of Natural Science

Department of Computer Science

Deressa Wodajo Deressa

Advisor: Solomon Atnafu (PhD)

This is to certify that the thesis prepared by *Deressa Wodajo Deressa*, titled: *Deepfake Video Detection Using Convolutional Vision Transformer* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor:			
Examiner:			
Examiner:			

Abstract

The rapid advancement of deep learning models that can generate and synthesis hyper-realistic videos known as Deepfakes and their ease of access to the general public have raised concern from all concerned bodies to their possible malicious intent use. Deep learning techniques can now generate faces, swap faces between two subjects in a video, alter facial expressions, change gender, and alter facial features, to list a few. These powerful video manipulation methods have potential use in many fields. However, they also pose a looming threat to everyone if used for harmful purposes such as identity theft, phishing, and scam. Therefore, it is important to tell whether a specific video is real or manipulated to deter and mitigate the risks posed by Deepfakes. Thus, in this thesis work, we present a system that detects whether a specific video is real or Deepfake.

The proposed system has two components: the preprocessing and the detection component. The preprocessing prepares the video dataset for the detection stage. In the preprocessing, the face region is extracted in 224 × 224 RGB format. Data augmentation is applied to increase the dataset and also increase the accuracy of the model. For the detection component, we use a Convolutional Neural Network (CNN) and Vision Transformer (ViT). The CNN has only convolutional operations (without a fully connected layer), and its purpose is to extract learnable features. The ViT takes in the learned features as input and further encodes them for the final detection purposes.

The proposed system is implemented using PyTorch, an open-source machine learning library. The DeepFake Detection Challenge Dataset (DFDC) was used to train, validate, and test the model. The DFDC dataset contains 119,154 videos created using publicly available video generation deep learning models. Our model was trained on 162,174 face images extracted from the video dataset. Ninety percent of the face images are augmented during training and validation. We tested the model on 400 unseen videos, and have achieved 91.5 percent accuracy, an AUC value of 0.91, and a loss value of 0.32. Our contribution is that we have added a CNN module to the ViT architecture and have achieved a competitive result on the DFDC dataset.

Key Words: Deep Learning, Deepfakes, Deepfakes Video Detection, CNN, Transformer, Vision Transformer, Convolutional Vision Transformer, GAN

Dedication

To my mother Genet Tesfaye

Acknowledgments

I would like to acknowledge the following people and organizations who have helped me to undertake this thesis.

I would like to thank my supervisor Asso. Professor Solomon Atnafu for all the advice and mentorship. You have provided me the right guidance at the time I was your student, and when I was conducting this thesis work. Your supervision and expertise are much appreciated, and I'll pay it forward.

I would like to thank Dr. Yaregal Assabie, whose suggestion and valuable input early at the proposal stage have significantly shaped the course of this thesis. Your sound suggestions on my proposal research method, and the valuable discussions after that, has been a great input that really helped me well.

I would like to thank my friend Dr. Merhawi GebreEgziabher who has facilitated access to Google Cloud Platform, and for the many countless favors I'm indebted to through the years.

I would like to thank Kaggle.com, sci-hub.tw, and the Google Cloud Platform free-tier for allowing free access to their platforms. Without those free access platforms, this thesis would not have been realized on time.

I would like to give my deepest gratitude to my family for the support they have provided me during the preparation of this thesis work.

Last but not least, I would like to thank Jimma University for sponsoring my graduate studies and Addis Ababa University for providing financial assistance for this thesis work.

Table of Contents

List of Tables iii
List of Figures iv
List of Algorithms v
Acronyms and Abbreviation vi
Chapter 1 : Introduction 1
1.1 Background 1
1.2 Motivation 3
1.3 Statement of the problem 4
1.4 Objectives 5
1.5 Methods 6
1.6 Scope and Limitations 7
1.7 Application of Results 7
1.8 Organization of the Thesis 7
Chapter 2 : Literature Review 8
2.1 Deep Learning and Artificial Neural Networks 8
2.1.1 Activation Functions 12
2.1.2 Backpropagation 13
2.2 Convolutional Neural Networks 14
2.2.1 Convolution Layer 15
2.2.2 Pooling Layer 16
2.2.3 Fully Connected Layer 16
2.3 Recurrent Neural Networks 16
2.4 Transformer 18
2.4.1 Encoder and Decoder Stacks 19
2.4.2 Attention 19
2.4.3 Position-wise Feedforward Networks 20
2.4.4 Embeddings, Softmax, and Positional Encoding 20
2.4.5 Vision Transformer 20
2.5 Deep Generative Models 21
2.5.1 Autoencoders 22
2.5.2 Generative Adversarial Networks 23
2.6 Deepfakes 24

2.6.1 Face Synthesis 26
2.6.2 Face Swap 27
2.6.3 Facial Attributes Manipulation 28
2.6.4 Facial Expression Swap 28
2.7 Data Augmentation 29
Chapter 3 : Related Work 31
3.1 Deepfake Detection Using Neural Networks 31
3.2 Deepfake Detection Using Ensemble Method 33
3.3 Summary 33
Chapter 4 : Design of Model for Deepfake Video Detection 35
4.1 The Deepfake Video Detection Model 35
4.2 Preprocessing 35
4.2.1 Face Extraction 36
4.2.2 Similarity between Faces 40
4.2.3 Data Augmentation 43
4.3 The Detection Component 43
4.3.1 The Training component 43
4.3.1.1 Feature Learning 43
4.3.1.2 Convolutional Vision Transformer 46
4.3.2 The Validation Component 47
4.3.3 The Testing Component 47
Chapter 5 : Implementation and Evaluation 49
5.1 Dataset Preparation 49
5.2 Tools and Experiment Setup 50
5.3 Model Evaluation 50
5.4 Training and Experiment Results 51
5.5 Effects of Data Processing During Classification 53
Chapter 6 : Conclusion and Future Work 56
6.1 Conclusion 56
6.2 Contribution 57
6.3 Future Work 57
References 58

List of Tables

TABLE 4-1: FEATURE LEARNING COMPONENT PARAMETERS 45
TABLE 5-1: CVIT MODEL PREDICTION ACCURACY ON FACEFORENSICS++ DATASET 52
TABLE 5-2: ACCURACY OF OUR MODEL AND OTHER DEEPFAKE DETECTION MODELS ON THE DFDC DATASET 53
TABLE 5-3: AUC PERFORMANCE OF OUR MODEL AND OTHER DEEPFAKE DETECTION MODELS ON UADFV DATASET 53
TABLE 5-4: DL LIBRARIES COMPARISON ON DEEPFAKE DETECTION ACCURACY 55

List of Figures

Figure 2.1: A single artificial neuron with n inputs 10
Figure 2.2: A feedforward network 11
Figure 2.3: Sigmoid function 13
Figure 2.4: ReLU function 13
Figure 2.5: Step function 13
Figure 2.6: Leaky ReLU 13
Figure 2.7: A typical CNN architecture 14
Figure 2.8: A typical RNN architecture 17
Figure 2.9: Vision Transformer 21
Figure 2.10: Basic Autoencoder architecture 22
Figure 2.11: GAN architecture 24
Figure 4.1: Deepfake video detection model 36
Figure 4.2 Sample extracted fake face images 39
Figure 4.3: Sample extracted real face images 39
Figure 4.4: A Deepfake feature being copied from person A to person B 39
Figure 4.5: Face images used for training and testing 40
Figure 4.6: A stack of CONV blocks of the feature learning 44
Figure 4.7: Stack of convolutional operation of the feature learning 44
Figure 4.8: Feature learning 44
Figure 4.9: Patch and Position Embedding 46
Figure 5.1: Augmentations used in training and validation phase 49
Figure 5.2: train loss and valid loss progression 51
Figure 5.3: train accuracy and valid accuracy progression 51
Figure 5.4: ROC curve for 400 unseen DFDC videos 52
Figure 5.5: MTCNN non face region detection 54
Figure 5.6: BlazeFace non face region detection 54
Figure 5.7: face_recognition non face region detection 54
Figure 5.8: ROC Curve with no data preprocessing 54

List of Algorithms

ALGORITHM 4.1: FACE EXTRACTION 38
ALGORITHM 4.2: COMPUTATION OF THE DISTANCE BETWEEN TWO FRAMES IN A VIDEO 41
ALGORITHM 4.3: PREPARATION OF FACE IMAGES FOR TRAINING 42

Acronyms and Abbreviation

ANN	Artificial Neural Network
AD	Action Descriptors
AU	Action Units
Adam	Adaptive Moment Estimation
AUC	Area Under Curve
AI	Artificial Intelligence
AM	Attention Model
AR	Augmented Reality
AE	Autoencoder
AFW	Automatic Frame Weighting
BPTT	Backpropagation Through Time
CEO	Chief Executive Officer
CGI	Computer Generated Imagery
CV	Computer Vision
CT Scan	Computerized Tomography Scan
CAAE	Conditional Adversarial Autoencoder
CVAE-GAN	Conditional Variational Autoencoder-Generative Adversarial Network
CONV BLOCK	Convolutional block
CNN	Convolutional Neural Network
ConvNet	Convolutional Neural Network
CViT	Convolutional Vision Transformer
CycleGAN	Cycle-Consistent Adversarial Networks
DCGAN	Deep Convolutional Generative Adversarial Network
DL	Deep Learning
DFDC	Deepfake Detection Challenge Dataset
D	Discriminator
E	Encoder
XOR	Exclusive Or
EM	Expectation Maximization
FSGAN	Face Swapping Generative Adversarial Network
FACS	Facial Action Coding System
FAE	Facial Attribute Estimation
FAM	Facial Attribute Manipulation
FNN	Feedforward Networks
FFNN	Feedforward Neural Network
FC	Fully Connected Layer
GRU	Gated Recurrent Unit
GELU	Gaussian Error Linear Units
Gen	Generation

GAN	Generative Adversarial Network
GPT-3	Generative Pre-trained Transformer 3
G	Generator
GCP	Google Cloud Platform
i2i	Image-To-Image Translation
JPEG	Joint Photographic Experts Group
LAPGAN	Laplacian Pyramid Of Adversarial Networks
LSTM	Long Short Term Memory
MRI Scan	Magnetic Resonance Imaging Scan
MesoNet	Mesoscopic Network
MLP	Multilayer Perceptron
MTCNN	Multi-Task Cascaded Convolutional Networks
NN	Neural Networks
OC-FaceDect	One Class Face Detection
ROC	Receiver Operating Characteristic
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RSGAN	Region-Separative Generative Adversarial Network
ResNet	Residual Network
SliderGAN	Slider Generative Adversarial Network
SGD	Stochastic Gradient Descent
STYLEGAN	Style-Based Generator Architecture For Generative Adversarial
SVM	Support Vector Machine
TCL	Transpose Convolutional Layer
DFT	Unmasking Deepfakes With Simple Features
VAE	Variational Autoencoder
VR	Virtual Reality
ViT	Vision Transformer
VGG	Visual Geometry Group

Chapter 1 : Introduction

1.1 Background

Technologies for altering images, videos, and audios are developing rapidly [1, 2]. Techniques and technical expertise to create and manipulate digital content are also easily accessible. Currently, it is possible to seamlessly generate hyper-realistic digital images [3] with a little resource and an easy how-to-do instructions available online [4]. Deepfake is a technique which aims to replace the face of a targeted person by the face of someone else in a video [5]. It is created by splicing synthesized face region into the original image [2]. The term can also mean to represent the final output of a hype-realistic video created. Deepfakes can be used for creation of hyper-realistic Computer Generated Imagery (CGI), Virtual Reality (VR) [6], Augmented Reality (AR), Education, Animation, Arts, and Cinema [7]. However, since Deepfakes are deceptive in nature, they can also be used for malicious purposes.

Although the manipulation of videos is not new, the current wave of Deepfakes creation begun when a Reddit user posted an adult Deepfake videos [8]. This is just the beginning, but the implication is that it will get worse when an adversary tries to use it for malicious gains [4]. The prevalence of open source software and applications have helped with the creation of Deepfakes effortlessly. Due to this, the creation and dissemination of Deepfakes are increasing at an alarming rate. It's now easy to personify anyone with a single image making them say what they have never said [9] and do things that they have never done [10].

Deepfakes are difficult to distinguish from real digital videos [1] and are becoming a great public concern [4, 11]. Deepfakes can be used to cause harm to people and this can have a detrimental effect on people, political institutions and society at large [1]. Their dire consequences are already being felt as digital images of celebrities are being used in adult Deepfakes without their consent [4]. This has urged various concerned bodies to come up with a solution to detect and expose Deepfakes.

Since the Deepfake phenomenon, various authors have proposed different mechanisms to differentiate real videos from fake ones. As pointed by [12], even though each proposed mechanism has its strength, current detection methods lack generalizability. The authors noted that current existing models focus on the Deepfake creation tools to tackle by studying their

supposed behaviors. For instance, Yuezun et al. [13] and TackHyun et al. [14] used inconsistencies in eye blinking to detect Deepfakes. However, using the work of Konstantinos et al. [15] and Hai et al. [16], it is now possible to mimic eye blinking. The authors in [15] presented a system that generates videos of talking heads with natural facial expressions such as eye blinking. The authors in [16] proposed a model that can generate facial expression from a portrait. Their system can synthesis a still picture to express emotions, including a hallucination of eye-blinking motions.

We base our work on two weaknesses of Deepfake detection methods pointed out by [12, 17]: data preprocessing, and generality. Polychronis et al. [17] noted that current Deepfake detection systems focus mostly on presenting their proposed architecture, and give less emphasis on data preprocessing and its impact on the final detection model. The authors stressed the importance of data preprocessing for Deepfake detections. Joshual et al. [12] focused on the generality of facial forgery detection and found that most proposed systems lacked generality. The authors defined generality as reliably detecting multiple spoofing techniques and reliably spoofing unseen detection techniques.

Umur et al. [7] proposed a generalized Deepfake detector called FakeCatcher using biological signals (internal representations of image generators and synthesizers). They used a simple Convolutional Neural Network (CNN) classifier with only three layers. The authors used 3000 videos for training and testing. However, they didn't specify in detail how they preprocessed their data, and also they didn't use any data augmentation. From [18, 19, 20], it is evident that very deep CNNs have superior performance than shallow CNNs in image classification tasks. Hence, there is still room for another generalized Deepfake detector that has extensive data preprocessing pipeline and also is trained on a very deep Neural Network model to catch as many Deepfake artifacts as possible.

Lingzhi et al. [21] proposed a face X-ray method to capture any Deepfakes without knowing the Deepfakes process of generation and synthesis. Their approach is general, and they have achieved superior results than any current Deepfake detection, as documented in their paper. However, Joao [22], in their attempt to spoof Deepfake detectors that rely on biological signals and Deepfake generators fingerprints, have proposed GANprintR, which disrupts the generated facial images' internal representation, which the authors in [21] proposed system

uses to detect the forgery. In addition to this, their proposed system is not trained on any of the artifacts of Deepfakes. That means their approach will be highly disadvantaged when GAN methods like GANprintR are used to disrupt the internal representation of generated images. In response to this, we believe that there is a need to develop a general model that distinguishes current Deepfake artifacts and is trained on large datasets to learn as many Deepfake artifacts as possible.

Therefore, we propose a generalized Convolutional Vision Transformer (CViT) architecture to detect Deepfake videos using Convolutional Neural Networks and the Transformer architecture. We call our approach generalized for three main reasons. 1) Our proposed model can learn local and global image features using the CNN and the Transformer architecture by using the attention mechanism of the Transformer. 2) We give equal emphasis on our data preprocessing during training and classification. 3) We propose to train our model on a diverse set of face images using the largest dataset currently available to detect Deepfakes created in different settings, environments, and orientations.

1.2 Motivation

The current advancement of Deepfake is ongoing at a troubling rate. We believe that check and balance must be there to deter such developments before it reaches the point of no return. Deepfakes can be used to cause significant harm to politicians, individuals, celebrities, organizations, and society.

There are thousands of generated Deepfake images and videos online that can be accessed easily. Most of them are adult videos created by using celebrities' faces without their consent and permission. Deepfakes have the potential to be used to humiliate, blackmail, misinforms, and defame a particular subject. Scammers and criminals are also using Deepfakes. Recently, scammers have successfully transferred millions of dollars by impersonating the voice of CEOs through a phone call. They used Deepfake voice in each phone call. Deepfakes can alter medical evidences of CT and MRI scans for insurance fraud. Convincing propagandas can be created using Deepfake. Political parties might use it to target their opponents. Spies and government agents can use it to disseminate dangerous propaganda that might deteriorate relations or even cause a war. Deepfakes can be used to generate realistic fingerprints to unlock multiple users' devices. The damage that Deepfakes can cause is limitless.

In response to the Deepfake phenomenon, there is a growing interest in the research development of Deepfake detection. Current detection mechanism largely omits their data preprocessing pipelines, use very little dataset (usually in hundreds or thousands), and most of them doesn't follow a generalized approach. Our work aims to design and develop a general Deepfake detection technique trained on a diverse set of videos.

1.3 Statement of the problem

Deepfake poses an imminent threat. It is blurring the line between what is real and fake. The technology to create and diffuse Deepfakes is easily accessible, and we are ill-prepared to combat it if used for illegal purposes. A good example is STYLEGAN [23], a hyper-realistic image generator that has a corresponding website called "This Person Does Not Exist" which showcases generated faces [24] and Generative Pre-trained Transformer 3 (GPT-3) [25], a Deepfake text generator. The authors of GPT-3 refused to release their model for fear of its malicious use. A bot made of GPT-3 was posting comments and interacting with users for weeks on Reddit with no one noticing [26]

Despite the alarming Deepfake technology advancement, there exist inconsistent artifacts in Deepfakes, which can help to distinguish them from real videos. Such inconsistencies include mismatched facial landmarks [27], face warping artifacts [28], eye blinking [13], biological signals [7], Deepfake generators fingerprints, and a difference in eye colors [29].

As it is described by [12], current Deepfake detection approaches lack generality. Yuezun et al. [28] proposed an approach to detect Deepfakes by analyzing resolution inconsistencies. Since current Deepfakes can only generate images of a small resolution, generated images are warped and transformed before being swapped on another higher resolution image. Their method exploits these face warping artifacts that result from the Deepfake production pipeline. Similarly, Yuezun Li et al. [13] and Falko et al. [29] focused on inconsistencies found around the eye region. While Darius et al. [5] focused on Deepfake and Face2Face, the two earlier techniques that are used to create hyper-realistic forged videos. These two techniques generate seemingly real videos, but they leave traces such as face occlusions, and unrealistic facial reenactments. David and Edward [30] proposed a temporal-aware system that captures intra-frame inconsistencies and temporal inconsistencies between frames. Though these approaches can detect Deepfakes, they are bound to fail when new techniques are introduced [13].

Haya and Khaled [31] proposed Ethereum smart contracts to check the authenticity of certain digital media. They showed, instead of searching for traces of inconsistencies in Deepfakes, it is also possible to track the history of videos to check whether they come from an authentic source or not. The smart contracts are therefore used to trace and track the provenance of digital content to its source. However, Ethereum is not yet widely adopted. Thus, this approach still lacks generalizability.

The world of computer vision, as we know it, is dramatically changing. With it, the quality of Deepfakes also. As explained by Thanh et al. [8] and Yuezun Li et al. [13], Deepfake detection methods are still in their early-stage. Current detection methods mostly focus on the shortcomings of Deepfake generation techniques. The problem with such approaches is Deepfake creators commonly try to hide the creation method, which makes Deepfake detection difficult. As such, there is a need to focus on introducing general and scalable method. In relation to this, we propose a general way of detecting Deepfakes by using Convolutional Neural Networks (CNNs) and the Transformer. CNNs are popularly used in image recognition tasks and can help detect synthetically generated videos. Transformers are mainly used in Natural Language Processing (NLP) for sequence-to-sequence learning using a technique called attention mechanism. We leverage both the strength of CNNs and the Transformer to design a new general learning architecture for Deepfake detection. With our approach, we intend to design and develop a CViT model trained on very large image datasets to detect Deepfakes created in different settings, environments, and orientations.

1.4 Objectives

General Objective

The general objective of this research is to design and develop a model for Deepfake video detection using Convolutional Vision Transformer.

Specific Objectives

The specific objective of this research are

1. Review literatures and related work in the area of Deepfake detection.
2. Collect datasets released for training Deepfake detection models.

3. Design a model for generalized Deepfake detection that uses Convolutional Neural Networks and the Transformer.
4. Develop a face extraction algorithm that is used to realize the model
5. Develop a prototype to validate the proposed model.
6. Select the appropriate dataset to test the developed prototype.
7. Test and evaluate the performance of the prototype.

1.5 Methods

To conduct this research, current Deepfake video creation and detection mechanisms will be reviewed. Previous researches conducted around Deepfakes generation and detection will also be analyzed and evaluated to grasp the current state of Deepfakes. Special attention will be given to the GAN architecture, since GANs are predominantly used in Deepfake generations and synthesis. How GANs work and an efficient way to detect their generative behavior will be studied. The Transformer architecture will be reviewed and analyzed. We will analyze the gaps in the current research works and devise a mechanism to address the gaps.

After analyzing the current research gaps, we will propose a new DL model to address the implicated limitations. Consequently, various deep neural networks and the Transformer architecture will be reviewed and assessed to come up with a new model using a combination of a CNN and the Transformer architecture. DL models are data-driven. Hence, our DL model also requires a dataset. However, before feeding our acquired dataset into our proposed DL model, we will need to extract the face region first. Thus, we will develop a face extraction algorithm and other necessary algorithms that help preprocess our dataset. The face extraction and the preprocessing algorithms and the proposed DL model will be realized using a Python programming language and the PyTorch DL library.

Many datasets are available for Deepfakes research. The notable ones are The Deepfake Detection Challenge Dataset (DFDC) [32, 33], UADFV [27], DeepfakeTIMIT [4], and FaceForensics++ [34]. Other than this, there are also videos found on the internet. For this thesis, we will use the DFDC dataset for training, validating, and evaluating our prototype. Data augmentation is also used to increase our dataset and increase the accuracy of our proposed model.

To test our model, we will use three kinds of evaluation metrics: Accuracy, Receiver Operating Characteristic (ROC), Area Under the ROC Curve (AUC), and the loss function. The Accuracy tests how well our model performs on the dataset, the ROC tests our model's capacity, and the loss function tests how our model deviates from the correct class. All three will helps us determine our model's ability in detecting Deepfakes. Lastly, to know where our DL model stands compared to other previous research works on Deepfakes video detection, we will make a comparison between previous researches and our proposed model using the DFDC, UADFV, and the FaceForensics dataset.

1.6 Scope and Limitations

The model focuses on building Deepfake detection model. Its scope is Deepfake videos. In our work, we won't analyze lip-sync movements with the audio of a video and facial enactments. The proposed approach is not a replacement for the verification of real and fake videos. It deals with a detection approach that can help users to verify the authenticity of videos by analyzing deep embedded malicious artifacts.

1.7 Application of Results

Our generalized Deepfake detection model can be of great help for the verification and authenticity of real and fake videos. It will let users verify whether a video is Deepfake or not. The model could be used in conditions where verifiability is needed. This will have significant importance considering the current non-generalized approaches that cannot be used in multiple types of Deepfake videos.

1.8 Organization of the Thesis

The rest of this thesis report is organized as follows. Chapter two reviews literature works relevant to the proposed approach. Chapter 3 presents the related works. Chapter 4 presents the proposed model, and its components. Chapter 5 presents the implementation and experimental result of the proposed system. In chapter 6, conclusion is given and future work is suggested.

Chapter 2 : Literature Review

In this chapter, we will explore the methods of CNNs, Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM), Transformer, Deep Generative Models, the Deepfakes phenomenon, different face manipulation techniques, and the DL techniques used to create Deepfakes.

2.1 Deep Learning and Artificial Neural Networks

Deep learning (DL) is one of the essential breakthroughs made in artificial intelligence [35, 18], and machine learning [36, 37]. DL is a mathematical framework modeled after a human nervous system [38]. DL methods are considered state-of-the-art in Computer Vision, Image Processing, and Natural Language Processing problems [39, 40]. The power of DL to learn patterns from data has led to the progress of many advances in Computer Vision (CV) and Image Processing domains. One of this progress is in image recognition, image classification, and image generation. Deep Learning is also widely applied in various domains of applications such as self-driving cars, controlling drones, healthcare, finance, and security [41].

Traditional machine learning techniques rely on manual feature engineering, which required domain experts. They have limited capability in recognizing patterns and inferences in raw unprocessed data [42, 43]. The traditional machine learning techniques are labor-intensive, inaccurate, and time consuming compared to DL approaches. DL is a powerful learning technique that has surpassed traditional machine learning architectures in Computer Vision (CV) and Image Processing, both in efficiency and accuracy [44, 45]. DL architectures can discover the semantic relationship between data and its representation in a computer system. DL techniques are used as feature extractors and classifiers due to their ability to generate high-level data representations automatically [46]. DLs are capable and efficient at processing and analyzing visual media [47]. The learning capability of DL enables various applications to understand abstract information within audiovisual media such as sound, image, and video [8].

Deep learning techniques can be categorized into supervised, unsupervised, and semi-supervised learning types [48, 49]. In supervised learning, the learning algorithm is fed with

a pair of labeled input and corresponding target output is acquired [41, 50]. The training dataset is prepared independently of the learning algorithm. Supervised learning has two phases: the training phase and the testing phase [51]. In the training phase, the goal is to predict and classify the target data without the knowledge of its label. The accuracy of prediction and classification is calculated and adjusted by the learning algorithm until the performance threshold is reached. Such a performance threshold is measured using a function called objective function or loss function. In the testing phase, the learning algorithm prediction accuracy is tested using unlabeled data. The loss function helps to minimize the prediction error [52]. Most deep neural network architectures used in image classification are supervised learning [41, 52].

Unsupervised learning does not require labeled pairs of input and output data. Instead, the learning algorithm extracts similar patterns and features from unstructured training dataset [48, 51]. The learning algorithm is trained to discover hidden features of unlabeled data in order to make a decision for the test dataset [41, 50]. The test datasets are then categorized into their respective classes based on the communalities of the discovered features [40]. Deep generative models are types of unsupervised learning algorithm [51, 53].

Semi-supervised learning falls between supervised and unsupervised learning. It has the properties of both learning architectures [40]. The training dataset used in semi-supervised learning mostly contains a small amount of labeled data and a large amount of unlabeled data [54]. The model has to learn the hidden features of the labeled dataset to classify the unlabeled data. Semi-supervised learning methods are more accurate than unsupervised models and less laborious than supervised learning [40].

The goal of DL is to construct a function that classifies the training data correctly, so it can generalize to unseen test data [55]. The basis for such function is the concept modeled from the human brain system called Artificial Neural Network [40, 56]. Artificial Neural Networks (ANN), also known as Neural Networks (NN), is a mathematical model designed to mimic human cognition [45, 40]. NN consists of a basic computational element called neurons connected in a weighted directed graph [57, 58] that transmits input signals from an outside source to produce an output through a hierarchical nonlinear computation [38]. Neurons are represented as nodes that are interconnected in layers [59]. As shown in Figure 2.1 [59, 60],

they take one or more input signals from other neurons and produce only one output signal [61]. Each neuron is associated with a specific value called weights. Weights represent the connection strength between nodes, and they determine how signals travel through nodes [62]. It is with such information flow that we determine the learning capability of NN architecture. Commonly, NN has three layers: an input layer, a hidden layer, and an output layer [49]. The learning process in NN involves a layered computational mapping of an input $x \in \mathbb{R}^n$ and output $y \in \mathbb{R}^d$ by approximating a learning function $f(\cdot)$ [63, 55]. Such mapping is refereed as forward propagation. Depending on the domain of the problem, input x can be discrete or sequential while y is usually a probabilistic distribution of learned representation [63, 58]. The value of n and d represents the dimension of input and output features respectively. The output y represents the best learned state of function $f(\cdot)$ [63]. Mathematically, NN can be expressed as follows [59]:

$$f(x) = \sum_{i=1}^{n} w_i x_i + \beta = W^{\top} x_i \qquad (1)$$

where $f(x)$ is the learning function, w_i is the weight vector and $W^{\top}$ its transpose , x_i is the input neurons, β is the bias.

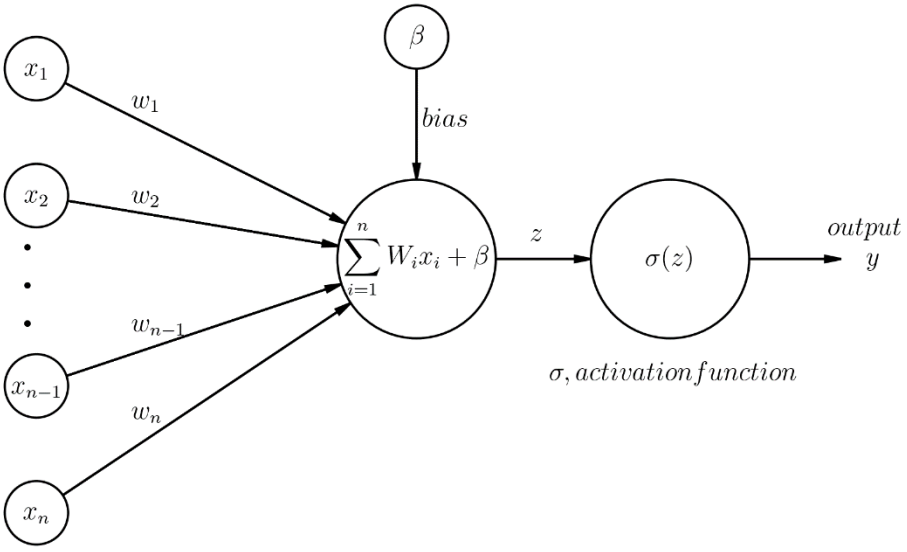

Figure 2.1: A single artificial neuron with n inputs

The earliest known NN model is the perceptron introduced by Rosenblatt for image classification tasks in 1943 [56, 64]. The perceptron has only two layers that map pixels of an image into their corresponding classes using an activation function called step function [38]. Step function outputs 1 for all positive values and 0 for all negative values. However, since perceptrons are linear models they could only solve a limited set of problems that require linear separability [65, 66], and it was shown, in 1969, that they are unable to classify simple XOR operations [49, 59]. The limitation of perceptrons highly hindered researches made on NNs until the new generation of NNs with the concept of backpropagation was introduced in the 1980s [38, 65, 67]. In 1986 Multilayer Perceptron with a learning algorithm called backpropagation was introduced and laid the foundation to the current wave of DL [45, 65]. Multilayer Perceptrons consists of 3 layers of NN. They are also called Feedforward Neural Network (FFNN) since the signal from the input layer pass through one direction [63]. An FFNN with many layers becomes a deep neural network [61, 63], hence the term Deep Learning.

FFNNs are a linear combination of multiple neurons that have at least two or more hidden layers [59]. Since the combination of linear neurons is linear, their combined effect is the same [68], and their relationship cannot be understood [63], and learning will not occur [55]. However, linear models can be extended to represent a nonlinear function by using a nonlinear transformation [63]. Activation functions are usually used to introduce non-linearity [68].

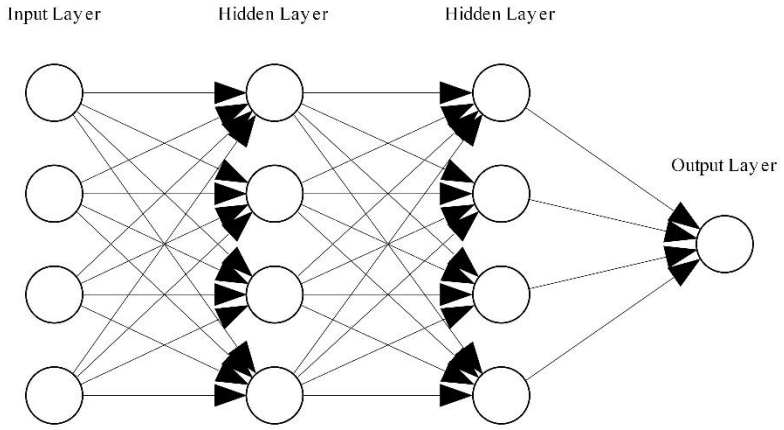

Figure 2.2: A feedforward network

2.1.1 Activation Functions

Activation functions are nonlinear functions that allow a large network of units to represent arbitrary functions [69] and learn complex mappings [56]. NNs with nonlinear activation functions can deal with more complex decision boundaries, and they can approximate nonlinear functions [70]. The universal approximation theorem states that an FFNN with a single hidden layer, a finite number of neurons, and a nonlinear activation function can approximate any continuous functions [69]. FFNNs are regarded as universal approximators [53] because they can approximate any continuous function [71]. Due to this, FFNNs are considered as general-purpose learning architecture [43]. This capability helps various DL architectures to learn and improve with experience [38, 71].

Three activation functions are commonly used in DL. They are the sigmoid function, the Rectified Linear unit (ReLU), and leaky ReLU [71]. They force all output values to be between specific thresholds. Sigmoid function clips all input values into an interval of 0 and 1 while ReLU clips negative values and passes the rest input value unchanged. Leaky ReLU has the same operation as ReLU. But, it changes large negative values into smaller ones [71, 56].

Sigmoid function is defined as [56]:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

Its gradient can be expressed in terms of the function itself (Eq. 3). It is continuous at all points, however it has a vanishing gradient problem which makes training deep NNs slower [56].

$$\frac{d(\sigma(x))}{dx} = (\sigma(x)\cdot(1 - \sigma(x))) \tag{3}$$

ReLU is defined as [56]:

$$ReLU = max(0, x) \tag{4}$$

$$\frac{d(ReLU(x))}{dx} = \begin{cases} 1 & if\ x > 0 \\ 0 & if\ x < 0 \\ undefined & if\ x = 0 \end{cases} \tag{5}$$

Leaky ReLU is defined as [56]:

$$LReLU(x) = \begin{cases} x & if\ x \geq 0 \\ 0.01x & if\ x > 0 \end{cases} \tag{6}$$

$$\frac{d(LReLU(x))}{dx} = \begin{cases} 1 & if\ x \geq 0 \\ 0.01 & if\ x > 0 \end{cases} \tag{7}$$

Figure 2.3 [72], 2.4 – 2.6 [56] shows activation functions.

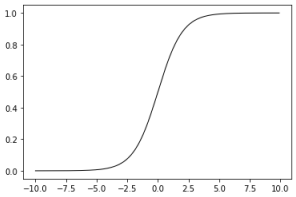

Figure 2.3: Sigmoid function

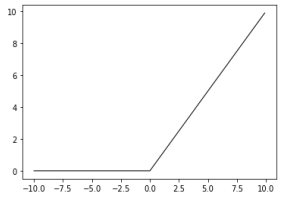

Figure 2.4: ReLU function

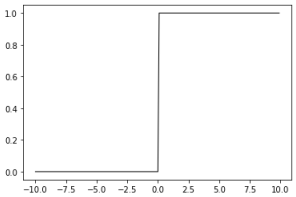

Figure 2.5: Step function

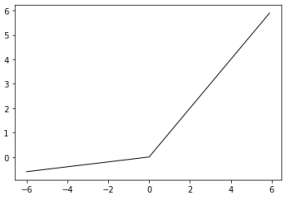

Figure 2.6: Leaky ReLU

2.1.2 Backpropagation

Optimization is one of the important component in DL. NN requires a lot of adjustments until its learning function outputs similar results as the target function. Trainable parameters weight w and bias β are iteratively optimized until the desired learning is achieved, and values that minimize loss function are obtained [72, 62]. This adjustment process is called parameter tuning, and the algorithm that updates the parameters is called backpropagation. The loss function measures the difference between the NN's output and the expected output. The derivative of the loss function is called gradient. Gradient descent refers to the calculation of a gradient on each weight in the NN for each training element. It is called gradient descent since we are descending the gradient to lower values [61] for finding the local minima of a learning function [37]. Backpropagation continually optimizes a NN model by adjusting

learning parameters in the steepest descent direction to minimizing an error [61]. It requires the derivative of the activation function for computing the steepest descent direction [73, 61]. Gradient descent based algorithms such as Stochastic Gradient Descent (SGD), and Adaptive Moment Estimation (Adam) are used for the optimization of NNs. SGD and Adam are first-order stochastic gradient-based optimization techniques [61]. Adam is a faster optimization algorithm, with SGD is slower but converges better than Adam. An algorithm is said to be convergent when it finds the local minimal of the function.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of FFNN supervised learning that learns from input data by extracting local features. They are inspired by the visual cortex of a human brain. CNNs are organized in multiple layers of convolution, pooling, activation (usually Sigmoid or ReLU), and fully connected layer (FC), as shown in Figure 2.7. Each layer produces an output called feature maps. Convolution, pooling, and activation layers are used for feature extraction, while FC is used for classification [43, 65]. In addition to the common layers, regularization layers such as dropout (a randomly selected subset of neurons are set to zero), batch normalization (input feature map is transformed to have zero mean and unit covariance), and learning rate with a step size (to speed up or slow down model convergence) are also used. CNNs produce very good results on input data that have a strong spatial correlation. CNNs are heavily used in computer vision tasks and are widely applied to various applications such as self-driving cars, robotics, speech synthesis, medical image analysis, and Natural Language Processing (NLP) [73].

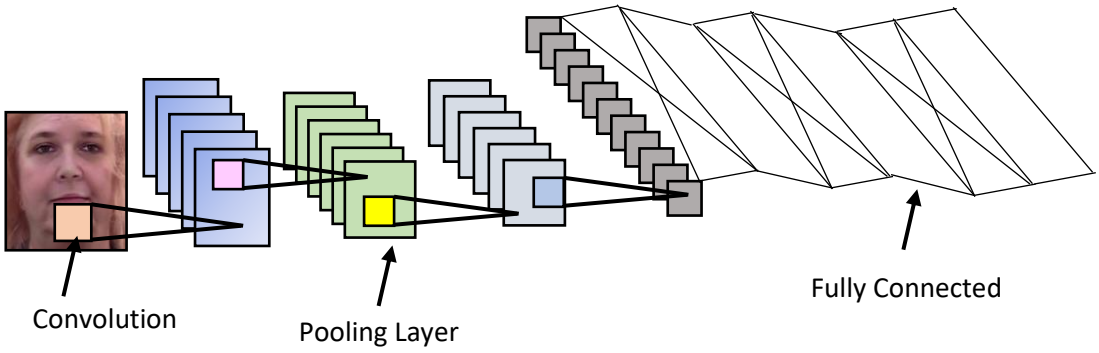

Figure 2.7: A typical CNN architecture

2.2.1 Convolution Layer

CNNs main operation is convolution. A convolution operation is the dot-product of a learnable kernel and an input tensor [74]. Kernels are a square matrix of the form $n_j \times n_j$ where n_j is a small (preferably odd) integer number, usually 3 or 5. Images are tensors of the form $r \times t \times d$, where $r \times t$ is the resolution, and d is the channel of the image. Each grid points $r \times t$ have a value called pixels. In convolutional operation of an image A of size $r \times r$, kernel K of size $j \times j$ is moved to the top-left corner of tensor A and region a_i is selected, where $i=j$. Each element of a_i is then multiplied with its corresponding element of n_j, and its result is summed and put in new tensor B_i, $B_i=\sum_{i=0}^{n} a_i n_i$. Kernel K is shifted across columns and rows of A until the last grid element is reached.

The number of columns and rows moved at one time is called the stride. For a given stride $s= \{1, 2 \ldots\}$, kernel K is shifted s columns to the right and s row down at each step [65, 75, 73]. Since images are highly correlated into regions, stride greater than two will lead to high information loss. Convolution will shrink the size of an input tensor which leads to the loss of information. To prevent this padding is used. Zero-padding is commonly used to preserve the output size [73]. The convolution operation is defined as [75]:

$$B_{lm} = (A*K)_{lm} = \sum_{f=0}^{n_j-1} \sum_{h=0}^{n_j-1} A_{l+f,\, m+h} K_{l+f,\ m+h} \tag{8}$$

Where B is tensor of $l \times m$ size, A is matrix representation of an image, and K is kernel

The dimension B_d of tensor B is computed as [75]:

$$B_d = \left\lfloor \frac{r-j}{s} + 1 \right\rfloor \tag{9}$$

Where B_d is the dimension of tensor B, r is size of matrix A, j is the size of kernel K

2.2.2 Pooling Layer

The pooling layer is used to reduce the size of a feature map, thus reducing the number of parameters in a CNN model [75, 74]. It also reduces the computation in the network [73, 74]. Pooling layer merges semantically similar features into one [36]. The size of the feature mask is reduced by using a downsampling mask. For example, a 2 × 2 downsampling mask will shrink the feature map dimension by half [37]. Different types of pooling exist. Average pooling computes the average value of a filter over a region in the image, while max-pooling computes the maximum values [46, 73].

2.2.3 Fully Connected Layer

The last layer of CNN is FC, and it is used for a classification task. Each unit of the FC layer receives input from the previous layer of feature maps [73], and then the scores of each class are computed [37]. Those scores are then used to update the network parameters using backpropagation.

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are FFNN with feedback loops that allow them to process sequential tasks [76, 77, 36]. The feedback loops are recurrent cycles over time [78]. They are types of supervised learning. RNNs exhibit temporal behavior that captures the dynamics of sequences via cycles in the computational graph and can adapt to dynamic changes in data [76, 69, 79]. They have internal states that act as memory, and which enables them to possess the ability to process time-dependent tasks [80, 69]. RNNs use backpropagation through time (BPTT), where error signals are propagated backward in time [78]. RNNs are applied in various tasks such as machine translation, image captioning, music generation, video analysis, Deepfake detection, and NLP [36].

RNNs have input, recurrent hidden, and output layers, as shown in Figure 2.8 [65]. RNNs intake a sequence of inputs and produce a sequence of outputs. The input layer receives data in a form a sequence of vectors with a time t such as $\{\ldots, X_{t-1}, X_t, X_{t+1}, \ldots\}$ where $X_t = (x_1, x_2, x_3, \ldots, x_N)$. The hidden units $h_t = (h_1, h_2, h_3, \ldots, h_M)$ in the hidden layer are connected with recurrent connections. The function used at time t is shared across

by weight matrixes W_{xh}, W_{hh}, and W_{hy} in the input layer, hidden layer, and output layer. RNN learning process is described in Eq. 10 [65] and Eq. 11 [65].

$$h_t = \sigma(W_{xh}X_t + W_{hh}h_{t-1}) \tag{10}$$

$$y_t = W_{hy}h_t \tag{11}$$

RNNs are situated for learning from sequential data. However, learning in RNN is difficult due to the vanishing and exploding gradient problem where the gradient of a function gets exponentially very small or very large as they are propagated back through time. These two problems hinder the ability of RNNs to learn long-term sequential dependencies in data [78].

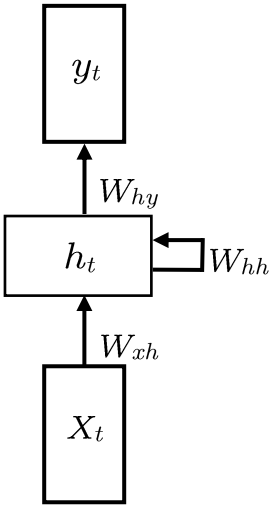

Figure 2.8: A typical RNN architecture

A principal concept in RNNs is the gating mechanism. The gating mechanism tries to solve the inherent gradient problem of RNNs by designing a sophisticated activation function consisting of affine transformation followed by a simple element-wise nonlinearity by using gating units [81]. Long-Short Term Memory (LTSM) is proposed by Hochreiter and Schmidhuber to address the vanishing gradient and exploding gradient problem of RNNs [63, 19]. LTSM introduced the notion of gates to remember the internal state of the network. There are three gates in LTSM: input gate, output gate, and forget gate. LTSM gates are vectors that control the flow of information [32]. The gates act as memory cells that pass the gradient without vanishing or exploding [61].

Gated Recurrent Units (GRU) was proposed by Cho et al. [82, 81]. They proposed an Encoder-Decoder unit that consists of two RNNs. The encoder and decoder of the model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The two RNNs adaptively capture dependencies of different time scales. The GRU has a gating unit that regulates the flow of information between the RNNs.

Both LTSM and GRU internal states can remember the existence of a specific feature in an input stream [81]. This ability of the networks creates a shortcut path that helps them to back-propagate without their gradient being vanished.

2.4 Transformer

The transformer is a NN architecture for language understanding [83]. The transformer is widely used for sequence modeling tasks [84]. It uses an attention mechanism to draw global dependencies between input and output. The concept of attention mechanism is first introduced by Dzmitry et al. [85] and is now widely applied to various domain tasks such as NLP, machine translation, and CV. The encoder-decoder architectures discussed in Section 2.3 have two well-known challenges [86, 87]. The first challenge is the encoder has to compress all the input information into a fixed-length vector, which then leads to a loss of information. The second challenge is that the architectures are unable to model alignment between input and output sequence since the decoder cannot selectively focus on relevant input while generating the output token. The Attention Model (AM) addresses those two challenges.

AM dynamically pays attention to only certain parts of input to help perform the task at hand effectively [86]. In AM, the decoder has access to the entire input sequence $X_t = (x_1, x_2, x_3 \ldots, x_N)$. AM uses attention weights α that are induced over the input sequence to prioritize a set of positions where the most relevant information is present. The AM holds information of correlation between words in NLP tasks and pixels in CV tasks. The weight α, thus, can be described as the vector of importance (since it holds relevant information) as it facilitates such correlation between input sequences in NLP and CV tasks.

Attention mechanisms can have different forms depending on the type of modeling approach. The transformer uses a self-attention mechanism that relates different positions of a single sequence to compute a representation of the sequence. The transformer has the following components: Encoder, decoder, attention mechanism, point-wise feedforward network, and embedding.

2.4.1 Encoder and Decoder Stacks

The transformer encoder and decoder each consists of six identical layers [83]. The encoder has two sub-layers, and the decoder has three sub-layers. The first two sub-layers in both the encoder and the decoder have the same structure. The first sub-layer is a multi-head self-attention mechanism. The second sub-layer is position-wise fully connected FFNN. A residual connection around each of the first two sub-layers is applied, followed by layer normalization. The residual connection is defined as $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layer itself. The output of the residual connection is dimension d_m=512. In the decoder stack, the self-attention sub-layer is modified in order to prevent positions from attending to subsequent positions.

2.4.2 Attention

The attention function is the mapping of a query and a set of key-value pairs to an output where the query, keys, values, and output are all vectors [83]. In Scale Dot-Product Attention, the input consists of queries, keys of dimension $d_{k,}$ and values of dimension d_v. The attention function is defined as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (12)$$

Where Q is the query, K is the key, and V is the value.

Multi-head attention (MSA) projects the queries, keys, and values h times with different, learned linear projection to d_k, d_k and d_v dimensions, respectively. The Multi-head attention is defined as:

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O \quad (13)$$

Where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, $W_i^Q \in R^{d_{model} * d_k}$, $W_i^K \in R^{d_{model} * d_k}$, and $W_i^V \in R^{d_{model} * d_v}$.

2.4.3 Position-wise Feedforward Networks

Each of the layers of the encoder and decoder stack contains Position-wise Feedforward Networks (FNN) [83]. The FNN is applied to each position separately and identically. The FNN is defined as:

$$FFN(x) = max(0,\ xW_1 + b_1)\ W_2 + b_2 \tag{14}$$

2.4.4 Embeddings, Softmax, and Positional Encoding

The transformer uses learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model} [83]. The Softmax is used to convert the decoder to predict the next-token probabilities. The transformer does not have a convolutional operation. As such, it doesn't have a mechanism to store local information. Thus, the transformer uses Positional Encoding to determine the relative and absolute position of the tokens in sequence.

2.4.5 Vision Transformer

Vision Transformer (ViT) [88] is a transformer model based on the work of [83]. The transformer and its variants (e.g., GPT-3 [25]) are predominantly used for NLP tasks. ViT extends the application of the transformer from the NLP problem domain to a CV problem domain. The ViT uses the same components as the original transformer model with slight modification of the input signal. In ViT, the input signals are an image with a 2D dimension. Hence, the images of the form $x \in R^{H \times W \times C}$ are reshaped in to a sequence of 2D patches of the form $x_p \in R^{N \times (p^2 \cdot C)}$ where $(H,\ W)$ is the resolution of the image, C is the channel of the image, (P, P) is the resolution of the image patch, and $N = HW/P^2$ is the number of patches. The image patch are flattened and mapped to D dimensions with trainable projection.

The architecture of the ViT is shown in Figure 2.9. The ViT has the following components: Patch Embeddings, Position Embeddings, and Encoder.

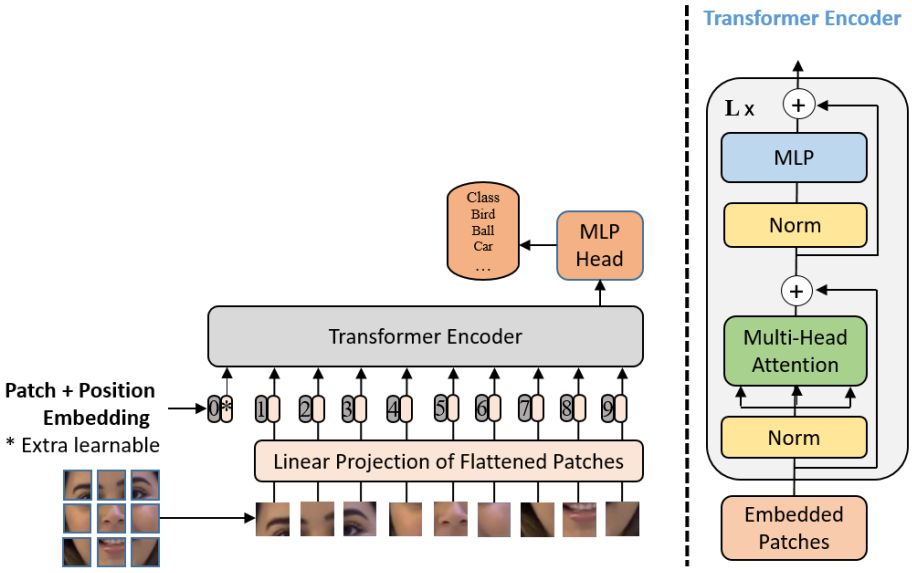

Figure 2.9: Vision Transformer

Patch Embedding and Position Embedding

The output of the linear projection N is called the Patch Embedding. The Position Embeddings are added to Patch Embedding to retain position information. The ViT uses learnable 1D Position Embeddings. The output of the Position Embeddings serves as input for the ViT Encoder.

Encoder

The ViT Encoder has a similar structure as the Encoder described in Section 2.4.1. LayerNorm is applied before every block and residual connection after every block. The Multilayer Perceptron (MLP) contains two layers with Gaussian Error Linear Units (GELU) nonlinearity.

2.5 Deep Generative Models

Deep Generative Models are types of unsupervised and semi-supervised learning that learn to generate new synthetic data [89]. They learn to capture the inner probabilistic distribution of individual classes to generate new data in any domain [90, 91]. Generative models, unlike FFNNs, and CNNs aim to predict feature maps given the label. They learn how a sample data is distributed (In case of an image, the distribution might be of the intensity of the pixel and color) and capture the probabilistic distribution of the sample to generate similar data [91]. For a given x and y as input and target variables, generative models estimate the distribution

of y given x, and the probabilistic distribution of y denoted as $P(y|x)$, and $P(y)$, respectively [92, 63]. Generative models are used in a wide range of data generation such as music, handwriting, faces, drawings, and object of different shapes [92].

2.5.1 Autoencoders

An Autoencoder (AE) is an FFNN with an encoder-decoder architecture that is trained to reconstruct its input data [43]. AEs, like FFNN, have three layers, as shown in Figure 2.10 [65]. The first part is an encoder that takes input data and transforms it into a smaller internal representation. The transformed representation is called code or bottleneck [92]. The middle (constricted) layer of AE is smaller than the input and the output layer, and it passes a reduced dimension of the original data. The second part is a decoder which converts the reduced representation into an output vector [65, 43]. Since the process of AE is lossy, the final output data can't be reconstructed as exactly as the input data. However, those reduced representations can be used to reconstruct the input data with high accuracy [43] by using loss function. The loss function measures the distance between the output of the decoder and the input [93]. AE goal is to learn the function A: $\mathbb{R}^n \rightarrow \mathbb{R}^p$ (encoder) and B: $\mathbb{R}^p \rightarrow \mathbb{R}^n$ (decoder) that satisfy [93]:

$$arg\min_{A,B} E[\Delta(X,\ B \circ A(X))] \tag{15}$$

where E is the expectation over the distribution of x, and Δ is loss function.

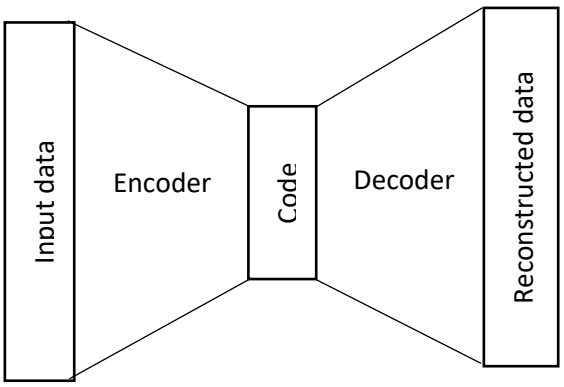

Figure 2.10: Basic Autoencoder architecture

There are different types of AE. Variational AEs (VAEs) are the most popular AE due to their improved representational capability [93] and the fact that their function approximation

produces small error [91]. VAE uses a probabilistic Gaussian distribution to model data using latent variables [63]. The aim is to maximize the similarity between the data and the latent variables using latent variables distributions with respect to the parameters of the encoder and the decoder [91, 63].

2.5.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are generative models that are designed to create generative models of data from samples [65, 94]. GANs are invented in 2014 by Ian Goodfellow [94]. GANs contain two adversarial networks, a generative model G, and discriminative model D. The generator and the discriminator act as adversaries with respect to each other to produce real-like samples [95]. The generator's goal is to capture the data distribution. The goal of the discriminator is to determine whether a sample is from the model distribution or the data distribution [94]. The generator G maximizes the probability of D making a mistake. While the discriminator D maximizes the probability of assigning the correct label to both examples and samples from G. The two adversarial model networks can be seen as a minimax two-player game. The generator creates new set of images by mapping a noise vector z in latent space to an image $G(z) \longrightarrow x$, and the discriminator D classifies an image as real or fake in a form of $D(x) \longrightarrow [0,1]$ [96]. GANs architecture is shown in Figure 2.11 [35]. Both the generator and discriminator compete to minimize the objective function [94]:

$$\min_{g}\max_{D} V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_{z}(z)}[log(1\text{-}D(G(z)))] \qquad (16)$$

where $\min_{g}\max_{D} V(D,G)$ is the objective function, x is a sample drown from the distribution p_{data} and z is a sample drown from noise distribution p_z generated by G.

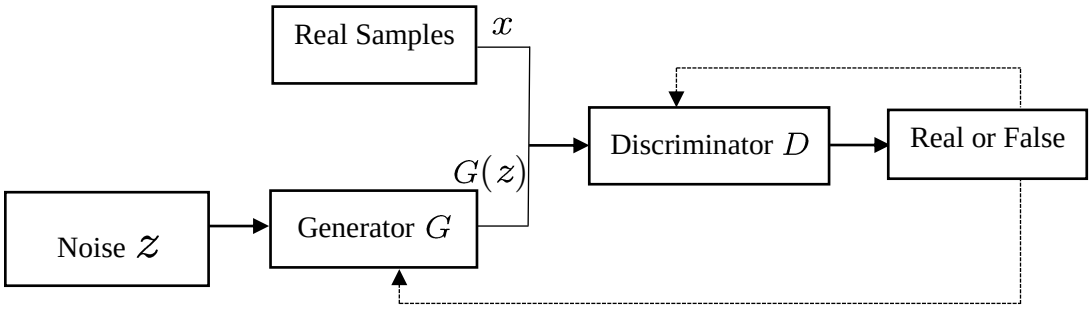

Figure 2.11: GAN architecture

The two adversarial networks are both differentiable functions that take input samples and random variable z. The objective function optimizes the performance of both networks until they cannot discriminate their sample sources correctly. That is, the generator G has captured the internal representation and distribution of the real sample data [35, 97].

Since the inception of GANs, there has been extensive study on their behavior, and many variants of GAN have been proposed in the process. DCGAN, WGAN, STYLEGAN, CycleGAN, and LAPGAN are some of the proposed GAN architectures [35]. GANs can be used to generate synthetic but realistic data such as texts, sounds, images, and videos [97]. They are also used in generating missing parts in data, image editing, NLP, Medical Imaging, image manipulation, image-to-image translation, image super-resolution, video animation, and rendering of virtual environments [95].

2.6 Deepfakes

Deepfakes are a relatively new phenomenon that is developing rapidly. Deepfakes are inevitable consequences of the quest to build a robust and general computer vision. They are the culmination of an ambitious drive to push the boundary of what is possible in Artificial Intelligence (AI) and DL. The term Deepfake can be used to describe both the technology to create Deepfakes and also the generated Deepfake content itself. In this thesis, we use the term Deepfake to refer to the content created by deep generative DL models. The Deepfake phenomena emerged in late 2017 by a user called Deepfakes on a microblogging site Reddit [98]. Reddit is a microblogging site with thousands of subreddits. The Reddit user Deepfakes created the "/r/Deepfakes" subreddit, in which he uploaded Pornographic videos he has generated using DL techniques. The user also released the source code he has used to create

the Deepfakes. Soon enough, videos attracted millions of viewership. Other users synthesized new Deepfakes by using the source code. Websites dedicated to Deepfakes started to emerge. Clones of the source code with some improved versions (e.g., DeepFaceLab [99]) begun to circulate. Alarmed by the implication of the Deepfakes, Reddit removed both the account and the Deepfakes subreddit in February 2018. However, the public interest in creating and using Deepfakes for malicious intent, profit, and entertainment is so strong that the ban had little effect on the momentum of Deepfakes generation and synthesis [8, 100].

Deepfake is a portmanteau of DL and fake [100]. Deepfake is generated and synthesized by deep generative models such as GANs and AEs [94, 100]. Deepfake is created by swapping between two identities of subjects in an image or video [101]. Deepfake can also be created by using different techniques such as face swap [102], puppet-master [9], lip-sync [103, 104], face-reenactment [105], synthetic image or video generation, and speech synthesis [106]. Supervised [89, 107, 108], and unsupervised image-to-image translation [109] and video-to-video translation [110, 111] can be used to create highly realistic Deepfakes.

The first Deepfake technique is the FakeAPP [8] which used two AE network. FakeApp's encoder extracts the latent face features, and its decoder reconstructs the face images. The two AE networks share the same encoder to swap between the source and target faces, and different decoders for training.

Most of the Deepfake creation mechanisms focus on the face region in which face swapping and pixel-wise editing are commonly used [3]. In the face swap, the face of a source image is swapped on the face of a target image. In puppet-master, the person creating the video controls the person in the video. In lip-sync, the source person controls the mouse movement in the target video, and in face reenactment, facial features are manipulated [101]. The Deepfake creation mechanisms commonly use feature map representations of a source image and target image. Some of the feature map representations are the Facial Action Coding System (FACS), image segmentation, facial landmarks, and facial boundaries [100]. FACS is a taxonomy of human facial expression that defines 32 atomic facial muscle actions named Action Units (AU) and 14 Action Descriptors (AD) for miscellaneous actions. AUs describes muscle-based atomic facial expressions. Every facial expression can be described using FACS and AUs. AD represents head poses, gaze direction, and jaw movement. Image segmentation is a

partitioning of a small segment of an image. Facial land marks are a set of defined positions on the face, such as eye, nose, and mouth positions [112].

The Deepfake creation mechanisms usually follow variations of the following six approaches [100]:

1. The architecture is responsible to the whole processes of Deepfake creation. That is, the architectures generates, synthesizes, and maps features vectors of source image to target image.
2. Train an AE to extract facial expression from source facial image, and then swap the extracted expressions onto target facial image.
3. Add additional encoding (e.g., FACS, AU or AD) from target source before passing to decoder.
4. Convert the intermediate face or body representation to the desired identity.
5. Use optical flow.
6. Create composite of source image and pass it to another network to refine the realism.

2.6.1 Face Synthesis

Image synthesis deals with generating unseen images from sample training examples [113]. It refers to the processor of synthesizing image features from some form of image description. Image synthesis techniques are applicable in digital media to create realistic special effects, removing unwanted objects from photos, increasing images resolution from low-resolution images, creating images from textual descriptions, altering and modifying facial poses, and changing photography to artwork [96, 114]. They are also being used to generate and synthesize hyper-realistic Deepfakes [32].

GANs are used mainly in face synthesis. GAN architecture, such as StyleGAN [23] and FSGAN [102], synthesize highly realistic-looking images. Based on the GANs generator and discriminator network arrangement, there is three class of image synthesis method: the direct method class, hierarchical method class, and iterative method class. In the direct method class of image synthesis, the generator and discriminator have the network structure shown as in Figure 2.11. In the hierarchical method, the GAN model has two generator and two discriminators that perform independent tasks using parallel or sequential interactions. In the

iterative method class, multiple generators perform similar processes while also sharing weights between each other.

Image-to-image translation (i2i) GANs learn how to transform an input image pattern into an output image. StyleGAN, SliderGAN [115], and other GAN-based image translation models can transfer patterns and expressions from the source facial attributes to the target image. The image patterns and facial expressions the i2i GAN models transform include head pose, hair color, age, and gender. The translation methods have achieved photo-realistic results.

Face image synthesis techniques are used in face aging, face frontalization, and pose guided generation. GANs are also used in generating face images to sketch, vice versa [116]. Face aging GANs transfer facial image to another while preserving identity [96]. Conditional Adversarial AE (CAAE) [117] combines AE E and GAN's Generator G to directly produce an image with the desired age attribute without requiring a paired sample. In CAAE, Convolutional AE maps the face attributes to the latent vector. Generator G performs a deconvolutional (reverse convolution) operation and projects the face manifold to the target age group. Face frontalization GANs change the face orientation in an image. Pose guided face image generation maps the pose of an input image to another image. Pose guided techniques can be used in creating Deepfake videos by mapping a source subject onto an image in a sample video.

2.6.2 Face Swap

Face swap or identity swap is a GAN based method that creates realistic Deepfake videos. The face swap process inserts the face of a source image in a target image of which the subject has never appeared [101]. It is most popularly used to insert famous actors in a variety of movie clips [118]. Face swaps can be synthesized using GANs and traditional CV techniques such as FaceSwap (an application for swapping faces) and ZAO (a Chines mobile application that swaps anyone's face onto any video clips) [101].

Face Swapping GAN (FSGAN) [102], and Region-Separative GAN (RSGAN) [119] are used for face swapping, face reenactment, attribute editing, and face part synthesis. RSGAN is a VAE-GAN that has two VAEs and a decoder. One VAE encodes the hair region and the other encodes the face region [100]. Face swap techniques use face alignment, Gauss Network

optimization, and image blending. Deepfake FaceSwap uses two AEs with a shared encoder that reconstructs training images of the source and target faces [101]. The processes involve a face detector that crops and aligns the face using facial landmark information [120]. A trained encoder and decoder of the source face swap the features of the source image to the target face. The autoencoder output is then blended with the rest of the image using Poisson editing [120].

2.6.3 Facial Attributes Manipulation

Facial attributes are properties that describe human understandable visual features [121]. Facial attributes manipulation, also known as, face editing or face retouching, modifies the facial attributes of a target image. The attributes include hair, skin color, gender, age, and adding glasses [101]. Facial attributes are used in face verification, face recognition, face retrieval, and face image synthesis [121].

Facial attribute creation consists of two sub processes: Facial Attribute Estimation (FAE) and Facial Attribute Manipulation (FAM). FAE classifier looks for the presence of an attribute in a given image, and FAM applies GANs to synthesize or remove the desired attributesGANs are usually used to modify facial attributes. FaceApp is a GAN based application that can be used for facial manipulations. The application is used for age progression.

2.6.4 Facial Expression Swap

Facial expression (face reenactment) swap alters one's facial expression or transforms facial expressions among persons. Expression reenactment turns an identity into a puppet [100]. Using facial expression swap, one can transfer the expression of a person to another one [122].

Various facial reenactments have proposed through the years. CycleGAN is proposed by Jun-Yan et al. [123] for facial reenactment between two video sources without any pair of training examples. CycleGAN is an i2i that learns until the distribution of source image and the target images are undistinguishable using cycle consistency loss. Conditional VAE-GAN (CVAE-GAN) is also a face reenactment GAN architecture where the generator G is conditioned on an attribute vector or class label [100].

Face2Face manipulates the facial expression of a source image and projects onto another target face in real-time [75]. Face2Face creates a dense reconstruction between the source

image and the target image that is used for the synthesis of the face images under different light settings [120]. Deferred Neural Rendering and NeuralTextures are rendering techniques proposed by [124]. The authors presented a new way of synthesizing imperfect 3D into a perfect photo-realistic rendering. Their method has two components, Deferred Neural Rendering and NeuralTextures. Deferred Neural Rendering is an image synthesis that combines traditional graphics methods with learnable components. NeuralTextures store learned feature maps that enables learned synthesis and rendering. NeuralTextures can be used to transfer person specific neural texture, facial animation, and facial expressions transfer. Others networks such as Synthesizing Obama [9], MocoGAN, Vid2Vid, GATH, GANimation, GANotation, FaceID-GAN, FSGAN, and FaceFeat-GAN are some of the proposed architecture for face reenactment [100].

Facial expression swap gives a malicious attacker the power to impersonate an identity of someone to say things they have never said or things they never did. The attacker can use expression swap to steal identity, spread misinformation, defamation, and tamper an evidence [100].

2.7 Data Augmentation

Data augmentation is a technique of creating new data from existing datasets using various transformation functions. Data augmentation is applied to increase the dataset for robust learning [130]. In such a case, we would want to create images similar to our dataset but with slightly different features. Data augmentation helps the DL model to learn from different data orientations and helps to better generalize to unseen data. DL models work best when they are trained on large datasets. Collecting large datasets is not always possible, and sometimes almost impossible, such as in image recognition of endangered species. Additionally, images we collect might not have rich features such as different textures, orientations, and luminosity, to list a few.

In DL, data augmentation can have two approaches. In the first approach, the DL model is fed with an already augmented data before the training phase. This process is called offline data augmentation. Offline data augmentation reduces the time of training. However, it is less flexible as it affects the probability of generating and testing a different set of images on the fly. Offline data augmentation increases the amount of data on disk. In the second approach,

the DL model augments data while training. This process is called online data augmentation. In online data augmentation, the transformation occurs using a predefined probability. This type of data augmentation helps to try and test various image orientations that increase the learning accuracy.

Chapter 3 : Related Work

With the rapid advancement of the Convolutional Neural Network (CNNs) [37, 125], generative adversarial networks (GANs) [94], and its variants [95] , it is now possible to create hyper-realistic images [126], videos [127] and audio signals [9, 128] that are much harder to detect and distinguish from real untampered audiovisuals. The ability to create a seemingly real sound, images, and videos have caused a steer from various concerned stakeholders to deter such developments not to be used by adversaries for malicious purposes [1]. To this effect, there is currently an urge in the research community to come with Deepfake detection mechanisms.

Deepfake detection methods fall into three categories [28, 100]. Methods in the first category focus on the physical or psychological behavior of the videos, such as tracking eye blinking or head pose movement. The second category focus on GAN fingerprint and biological signals found in images, such as blood flow that can be detected in an image. The third category focus on visual artifacts. Methods that focus on visual artifacts are data-driven, and require a large amount of data for training. Our proposed model falls into the third category. In this chapter, we will discuss various architectures designed and developed to detect visual artifacts of Deepfakes.

3.1 Deepfake Detection Using Neural Networks

Darius et al. [5] proposed a CNN model called MesoNet network to automatically detect hyper-realistic forged videos created using Deepfake [129] and Face2Face [130]. The authors used two network architectures (Meso-4 and MesoInception-4) that focus on the mesoscopic properties of an image. The authors collected various videos from different online platforms and FaceForensics. All collected videos are compressed by the H.264 codec. The authors visualized the network layers to evaluate their qualities and limitations and found out the eyes and mouth play an important role in Deepfake detection. Their method yielded 98% accuracy in Deepfake detection and 95% accuracy for Face2Face reenactments. The proposed MesoNet model uses a shallow CNN architecture, which has less capacity compared to Deep CNNs. The authors also used small training dataset.

Yuezun and Siwei [28] proposed a CNN architecture that takes advantage of the image transform (i.e., scaling, rotation and shearing) inconsistencies created during the creation of

Deepfakes. Their approach targets the artifacts in affine face warping as the distinctive feature to distinguish real and fake images. Due to this, their method doesn't require a training dataset for Deepfake detection, which saves time and computing resources. Their method compares the Deepfake face region with that of the neighboring pixels to spot resolution inconsistencies that occur during face warping. They used Visual Geometry Group (VGG)16 [19], ResNet50, ResNet101 and ResNet152 [20] CNN models on UADFV [27] and DeepfakeTIMIT [4] datasets and videos from YouTube. Their result accuracy is better than the accuracy found by MesoNet and HeadPose [27]. However, current Deepfake generation techniques such as NeuralTexture [124] can directly transfer textures of a source image to a target image without having resolution inconsistency problems.

Huy et al. [131] proposed a novel deep learning approach to detect forged images and videos. The authors focused on replay attacks, face swapping, facial reenactments and fully computer generated image spoofing. They extended the capability of capsule network, originally designed for computer vision problems, to solve digital forensic problems. The authors used a pre-trained VGG-19 network to extract latent features and a new CNN method. They tested the proposed method on Idiap REPLAY-ATTACK [132], Nguyen et al. [129], Rahmouni et al. [133] and MesoNet [5] datasets and their result outperformed detection methods proposed by [5, 129, 133].

Daniel Mas Montserrat et al. [120] proposed a system that extracts visual and temporal features from faces present in a video. Their method combines a CNN and RNN architecture to detect Deepfake videos. The proposed system uses an Automatic Frame Weighting (AFW) mechanism and a GRU. Deepfake video detection systems predict their class probability by giving a prediction to each frame and then averaging the result. Such an approach has its drawback. The first drawback is that the face detection DL model might incorrectly detect nonface regions as a face region. The second drawback is some videos might include more than one face where only one of them is Deepfaked, and the DL model might detect only one of the real frames. To address these limitations, the authors proposed AFW to weight each frame in a video using logit metrics computed by a pre-trained EfficientNet-b5 model. The facial features, the logits, and the AFW are calculated on all of the frames in the video and are passed to the GRU component. Then, the GRU unit combines the result to predict all of the faces in a video. They used the DFDC dataset, and their model achieved 92.61 percent

accuracy. The proposed AFW mechanism doesn't address the inherent defect of the DL libraries face detection anomaly.

3.2 Deepfake Detection Using Ensemble Method

Md. Shohel Rana and Andrew H. Sung [134] proposed a DeepfakeStack, an ensemble method (A stack of different DL models) for Deepfake detection. The ensemble is composed of XceptionNet, InceptionV3, InceptionResNetV2, MobileNet, ResNet101, DenseNet121, and DenseNet169 open source DL models. They used 49 real and 51 fake videos of the FaceForensics++ dataset and extracted 101 frames from each video in both classes. DeepfakeStack classifier takes the prediction of the seven DL ensembles and makes a decision based on their accumulated result. The authors used a small dataset, which affects the proposed model detection capacity on a diverse set of Deepfakes found on the internet.

Junyaup Kim et al. [135] proposed a classifier that distinguishes target individuals from a set of similar people using ShallowNet, VGG-16, and Xception pre-trained DL models. The main objective of their system is to evaluate the classification performance of the three DL models. For evaluation, they used the Disguised Faces in the Wild (DFW) 2018 dataset. They used a total of 11,657 face images that are classified into three categories: 2,403 real faces, 4,814 disguised faces, and 4,440 similar imposters. From the experiment, they found that ShallowNet had a poor at classifying fake and untampered class images while Xception performed relatively well with 62 percent accuracy.

3.3 Summary

The authors in [5] proposed two models to detect Deepfake that are generated by the methods of Face2Face and Deepfake. However, there are currently more than two Deepfake generators that are used in Deepfake creation and synthesis, such as StyleGAN [23] and FSGAN [102], that their proposed model doesn't recognize. The authors used a shallow CNN model with small learning parameters. The detection method of the authors in [28] focuses on Deepfake generations that creates face image in a fixed size, which leaves resolution inconsistencies on the images synthesized. However, new Deepfake generators, such as NeuralTextures, don't leave resolution inconsistencies in the generated Deepfake videos. The authors also tested their model on 96 videos, which is very small compared to currently available datasets for Deepfake detection, and have achieved Area Under the ROC Curve (AUC) of 97.4 on the

UADFV dataset. The authors in [135] tested three different DL pre-trained models, and all of the models performed poorly with 62% accuracy. The authors have shown that current image classifiers are unable to classify and detect Deepfakes.

The continued advancements in the development of Deepfakes creation tools will make the detection of Deepfake videos more challenging. Hence, it is in the best interest of our societal wellbeing to deter such developments by thoroughly examining each Deepfake generation technique. Even though we have to study the characteristics of each Deepfakes, detection techniques developed only for some Deepfake generators will not suffice. CNNs are commonly used in computer vision applications, especially in image recognition and face region detections. Transformers are currently being used in CV tasks. We will take advantage of both CNN and Transformer image detection properties to design a model that detects Deepfake videos. In light of this, we propose a generalized CViT model that is trained on the largest dataset currently available for Deepfake detections to detect as many Deepfake artifacts as possible.

Chapter 4 : Design of Model for Deepfake Video Detection

In this chapter, we present a methodology to design and develop a Deepfake video detection model. We will discuss each component of the model, algorithms used to implement the model, the data preprocessing pipeline, and the CViT classifier. We will also present the method used to train, validate, and test the CViT model.

4.1 The Deepfake Video Detection Model

The Deepfake video detection model consists of two components: the preprocessing component and the detection component. The preprocessing component consists of the face extraction, the similarity between faces, and data augmentation. The detection components consist of the training component, the validation component, and the testing component. The training and validation components contain a Convolutional Vision Transformer (CViT). The CViT has a feature learning component that learns the features of input images and a ViT architecture that determines whether a specific video is fake or real. The testing component applies the CViT learning model on input images to detect Deepfakes. Our proposed model is shown in Figure 4.1.

4.2 Preprocessing

Processing plays a paramount role in DL since DL learning models learn from data. The preprocessing component's function is to prepare the raw dataset for training, validating, and testing our CViT model. The preprocessing component has three sub-components: the face extraction, the similarity between faces, and the data augmentation component. The face extraction component is responsible for extracting face images from a video in a 224×224 RGB format. The similarity between faces computes the correlation between a set of face images to determine their supposed class (real or fake). This process of checking whether a set of extracted face images are real or fake is required due to the nature of our dataset, where Deepfake videos have pristine frames. Hence, the similarity between faces helps to mitigate the cumbersome process of checking each extracted image class manually. Data augmentation increases the dataset and also increases the prediction accuracy of our model. The three sub-components are used in the training and validation stage, while only the face extraction is used in the testing stage, as implicated in Figure 4.1.

Our proposed model preprocessing component accepts a video as input. Current Deepfake generation and synthesis mechanisms predominantly focus on the face region [98]. Most of the datasets released for the detection of Deepfakes are video clips with face manipulations. Therefore, the identification and extraction of the face region is the first step in our preprocessing pipeline.

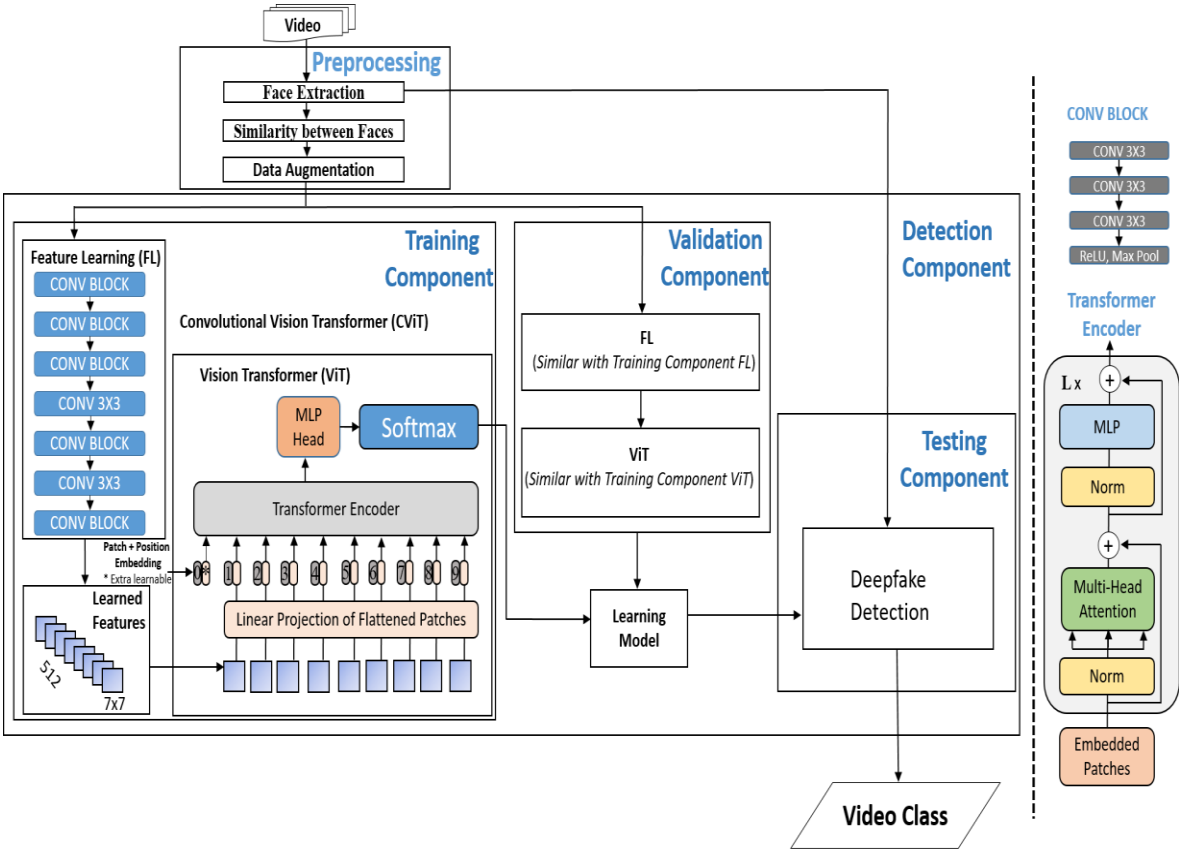

Figure 4.1: Deepfake video detection model

4.2.1 Face Extraction

The face extraction component prepares the video data into an image format for training, validating, and testing our proposed CViT model. The face extraction component detects and extracts the face regions using a face extractor. Using this component face regions are extracted, resized, changed to an image format, and are stored on disk.

The dataset we used for this work is the DeepFake Detection Challenge Dataset (DFDC) dataset [32]. The DFDC dataset is created for the Kaggle's Deepfake Detection Challenge. We chose the DFDC dataset for five main reasons. (1) The DFDC dataset is currently the largest publicly available dataset purposely developed for Deepfake detections. (2) The

dataset is created using several Deepfake generation mechanisms (see Section 2.5) and non-learned methods (traditional CV techniques, such as FaceSwap). (3) The dataset is produced using 3,426 different individuals that have distinct facial features and color tones, making it the most diverse dataset. (4) The video clips are recorded in various natural settings, such as indoor and outdoor, which is ideal for robust DL classifiers. (5) Creating a robust detection architecture based on the DFDC dataset will ultimately help us detect most Deepfakes found on the internet.

The DFDC dataset contains more than 100,000 videos created using different Deepfake generation methods. The DFDC Dataset is organized into 50 folders. In each folder, there are video clips and a metadata file containing the definitions of video name, label, split, and original (describes the original video name used for the creation of the fake video otherwise NaN). Each video clip runs for 10 seconds. The training folder contains 16,974 real video clips and 89,629 fake video clips. Each Deepfake has a corresponding original video clip, and each video clip has attributes in the metadata.

Thirty random frames were selected from each video to capture the face region, as shown in Algorithm 4.1. Each real video in the DFDC is used to synthesize one or more Deepfake videos at the ratio of 1:6. To mitigate this imbalance, we increased the number of frames saved from the real videos and lowered the number of frames saved from fake videos. Besides, some images of Deepfake videos are untampered images. We removed those possible untampered images from fake videos manually by inspecting their features, and also by computing the distance between each extracted frames to look for different faces in a video. Algorithms 4.2 shows the computation of the difference between face images in a video.

The face images are saved on disk by dividing the dataset folders into three categories, folder 0-33 for a train set, folder 34-45 for the validation set, and folder 46-49 for the test set. In each of the three categories, there are fake and real subfolders for the fake and the real face images, respectively. At the training stage, those subfolders represent the classes of our data. Algorithm 4.3 shows how the extracted faces are categorized and stored. Figure 4.2 and Figure 4.3 shows a sample of the extracted faces.

At preprocessing time, some videos found in folders from 40-49 failed to load, or the face extraction DL libraries would return an exception. This scenarios happens because the videos

were deliberately prepared using different compression ratios, have a wide range of qualities, and image resolution [32]. Most of the videos we analyzed in those folders have lower resolution and poor quality. We reasoned that, if we use those folders to create our validation set, we could test the robustness of our model. Hence, we chose those folders to be our validation and test sets.

Algorithm 4.1: Face extraction

```
Input: Video input of size N
Output: Face image of size m
Initialize f to 30 // number of frames to extract
Initialize faceweight to BlazeFace neural face detector weights
Initialize facerecognition to face_recognition weights
Initialize width to 224
Initialize height to 224
Initialize resized_face ← {}
for each video in N do:
    faces ← faces extracted from video using faceweight
    face_locations ← face locations detected
                    using facerecognition
    for each face_location in face_locations do:
        top, right, bottom, left ← face location coordinates
        face_image ← face[top:bottom, left:right]
        resized_face ← append resize(face_image, (width,height))
    end for
end for
Return resized_face
```

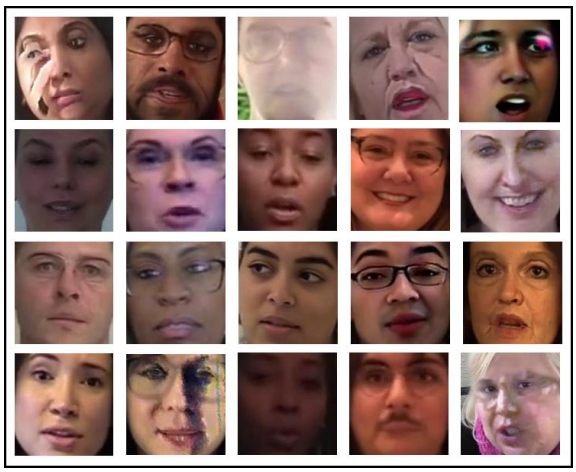

Figure 4.2 Sample extracted fake face images

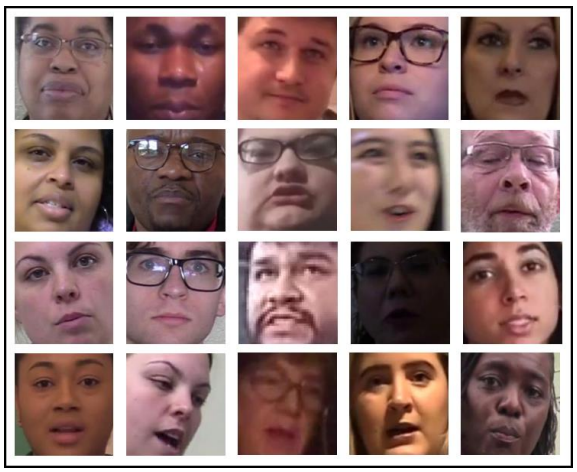

Figure 4.3: Sample extracted real face images

The face extraction process used at the testing component is slightly different than that of the training, and validation component. The face extraction process of the testing component extracts 50 frames of face images from the test videos using face recognition DL libraries. All face images have 224×224 RGB format. Deep Generative Models used for Deepfake creation sometimes will miss the face region during a Deepfake generation, leaving some of the artifacts in other areas of the video frame. Those artifacts are mostly found near the face region, as shown in Figure 4.4. Such artifacts are seen frequently for videos recorded when the subject is facing to the side, and the likelihood of those artifacts increases when the video involves more than one person.

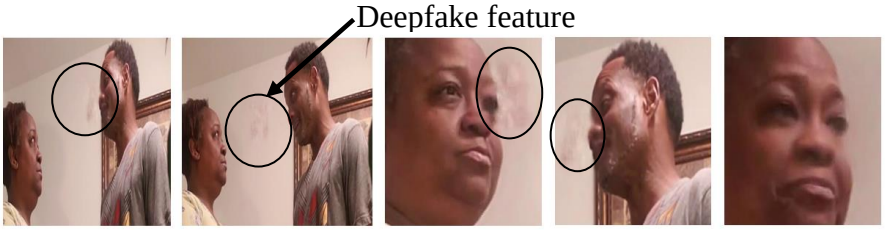

Figure 4.4: A Deepfake feature being copied from person A to person B

To include those artifacts (Deepfake features), we skipped the face location extraction we made in the DL library at the training stage, and we saved the faces the DL library extracts (Algorithm 4.1). We haven't included those padded faces on the training preprocess since we might add small background images, as in Figure 4.5 2) B, that the learning algorithm might

pick as a Deepfake feature, and might potentially hinder the detection pipeline. Figure 4.5 shows the face images we used for training and prediction.

Figure 4.5: Face images used for training and testing

1) A and 2) A are used for training, and 1) B and 2) B are used for prediction

4.2.2 Similarity between Faces

The authors of the DFDC dataset chose not to manipulate all of the fake frames of the DFDC dataset. Instead, they randomly selected the frames of the videos and applied the Deepfake generators. Hence, there are unmanipulated frames within fake videos in the DFDC dataset. The unmanipulated frames found in the fake videos need to be removed from the training dataset because they will affect the learning processes of our proposed model.

The DFDC dataset contains more than 100,000 videos, and it is very time-consuming to manually compare the fake and the unmanipulated frames, and remove them. Thus, the similarity between faces component compares extracted frames in videos to compute their similarity, as shown in Algorithm 4.2. The similarity between faces component's logic is, if the face images within a video are similar, then they likely belong to the same class (real, or fake). Since the Deepfake generators alter the face region significantly, a difference within frames tells us there are some modifications around the face region.

The similarity between faces component is applied to videos that we found ambiguous to distinguish between real or fake. The videos are mainly prepared in a low resolution, dark rooms, and have undistinguishable Deepfake artifacts. The DFDC dataset folders from 40 to 49 mostly contain videos with such characteristics.

Algorithm 4.2: Computation of the distance between two frames in a video

```
Input: image_list of size N
Output: similar images
Initialize similariy ← {}, Initialize encode ← {}
for each image in N do:
    encode ← encode image using face_recognition
end for
for each i in encode do:
    similarity[name] ← append ith name index
    similarity[value] ← initialize ith value index to 0
    store ← append ith face encode
    for each  j in encode do:
        if i!=j and i is not in store then:
            store ← append jth face encode
            face_distance ← distance between ith and jth encode
            if face_distance < 0.5 then:
                similarity[value] ← update ith and jth value by 1
            end if
    end for
end for
Return similarity
```

Algorithm 4.3: preparation of face images for training, validation, and testing

Input: metadata *k* of videos, image of size N

Output: DFDC{train[real, fake], validation[real, fake], test[real, fake]}

```
file_num ← 0 // the dataset have 50 folders
destination ← 'train'
image = {}
for loc in location do:
    metadata ← Read k
    for image in N do:
        if file_num > 34 and < 45 then:
            destination ← 'validation'
        else if file_num > 45 then:
            destination ← 'test'
        end if
        if metadata[image] [label] is in [real, fake] then:
            image ← append DFDC destination, label
    end for
end for
Return image, destination, label
```

4.2.3 Data Augmentation

Data augmentation processing techniques are readily available in many DL models and libraries. We chose Albumentations (see Section 5.1) and used online data augmentation (see Section 2.7). To determine the amount of data we should augment during training, we tested the data augmentation by altering the percentage of augmentation from 10 to 90 percent. Data augmentation of 60 to 90 percent works well for our dataset. We didn't save much time by choosing a lower percent. Therefore, we augmented 90 percent of our dataset.

4.3 The Detection Component

The detection of Deepfakes is done after the face image features are learned. The learning of the image features occurs after a careful design of appropriate components. In any DL model, training, validation, and testing is a standard workflow. Hence, our detection component comprises the training component, the validation components, and the testing component. In this section, we will briefly describe each element of our proposed model.

4.3.1 The Training component

The training component is the principal part of the proposed model. It is where the learning occurs. DL models require a significant time to design and fine-tune to fit a particular problem domain into its model. In our case, the foremost consideration is to search for an optimal CViT model that learns the features of Deepfake videos. For this, we need to search for the right parameters appropriate for training our dataset. The proposed CViT model consists of two components: Feature Learning (FL) and the ViT. The FL extracts learnable features from the face images. The ViT takes in the FL as input and turns them into a sequence of image pixels for the final detection process.

4.3.1.1 Feature Learning

The Feature Learning (FL) component is a stack of convolutional operations. The FL component follows the structure of VGG architecture [19]. The FL component differs from the VGG model in that it doesn’t have the fully connected layer as in the VGG architecture, and its purpose is not for classification but to extract face image features for the ViT component. Hence, the FL component is a CNN without the fully connected layer.

The FL component has 17 convolutional layers, with a kernel of 3 x 3. The convolutional layers extract the low level feature of the face images. All convolutional layers have a stride and padding of 1. Batch normalization to normalize the output features and the ReLU activation function for non-linearity are applied in all of the layers. The Batch normalization function normalizes change in the distribution of the previous layers [131], as the change in between the layers will affect the learning process of the CNN architecture. A five max-pooling of a 2 x 2-pixel window with stride equal to 2 is also used. The max-pooling operation reduces dimension of image size by half. After each max-pooling operation, the width of the convolutional layer (channel) is doubled by a factor of 2, with the first layer having 32 channels and the last layer 512. The FL component is shown in Figures 4.6-4.8.

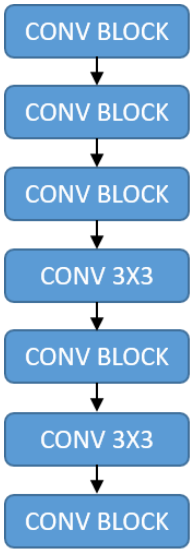

Figure 4.6: A stack of CONV blocks of the feature learning

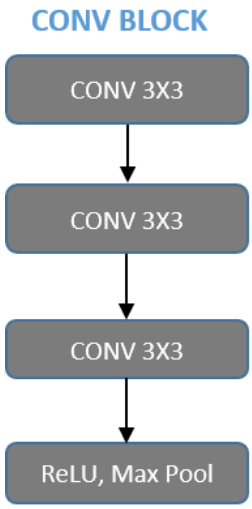

Figure 4.7: Stack of convolutional operation of the feature learning

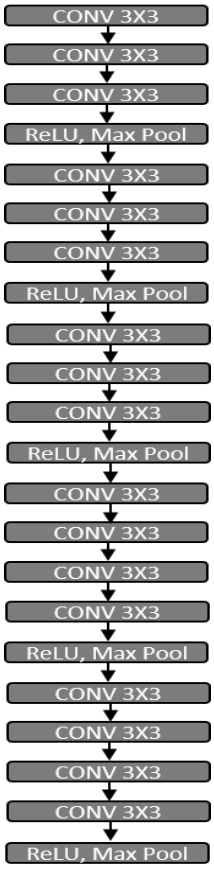

Figure 4.8: Feature learning

The FL component has three consecutive convolutional operations at each layer, except for the last two layers, which have four convolutional operations. We call those three

convolutional layers as CONV Block for simplicity. Each convolutional computation is followed by batch normalization and the ReLU nonlinearity. The FL component has 10.8 million learnable parameters, as shown in Table 4-1.

Table 4-1: Feature learning component parameters

Layer	FL Computation	Kernel	Stride	Input Size	Output Size	Number of Parameters
1	Convolution 1	3 × 3	1	224 × 224 × 3	224 × 224 × 32	896
	Batch Norm	3 × 3	(2,2)	224 × 224 × 32	224 × 224 × 32	64
	Convolution 2	3 × 3	1	224 × 224 × 32	224 × 224 × 32	9,248
	Batch Norm	3 × 3	(2,2)	224 × 224 × 32	224 × 224 × 32	64
	Convolution 3	3 × 3	1	224 × 224 × 32	224 × 224 × 32	9,248
	Batch Norm	3 × 3	(2,2)	224 × 224 × 32	224 × 224 × 32	64
2	Convolution 4	3 × 3	1	112 × 112 × 32	112 × 112 × 64	18,496
	Batch Norm	3 × 3	(2,2)	112 × 112 × 64	112 × 112 × 64	128
	Convolution 5	3 × 3	1	112 × 112 × 64	112 × 112 × 64	36,928
	Batch Norm	3 × 3	(2,2)	112 × 112 × 64	112 × 112 × 64	128
	Convolution 6	3 × 3	1	112 × 112 × 64	112 × 112 × 64	36,928
	Batch Norm	3 × 3	(2,2)	112 × 112 × 64	112 × 112 × 64	128
3	Convolution 7	3 × 3	1	56 × 56 × 64	56 × 56 × 128	73,856
	Batch Norm	3 × 3	(2,2)	56 × 56 × 128	56 × 56 × 128	256
	Convolution 8	3 × 3	1	56 × 56 × 128	56 × 56 × 128	147,584
	Batch Norm	3 × 3	(2,2)	56 × 56 × 128	56 × 56 × 128	256
	Convolution 9	3 × 3	1	56 × 56 × 128	56 × 56 × 128	147,584
	Batch Norm	3 × 3	(2,2)	56 × 56 × 128	56 × 56 × 128	256
4	Convolution 10	3 × 3	1	28 × 28 × 128	28 × 28 × 256	295,168
	Batch Norm	3 × 3	(2,2)	28 × 28 × 256	28 × 28 × 256	256
	Convolution 11	3 × 3	1	28 × 28 × 256	28 × 28 × 256	590,080
	Batch Norm	3 × 3	(2,2)	28 × 28 × 256	28 × 28 × 256	256
	Convolution 12	3 × 3	1	28 × 28 × 256	28 × 28 × 256	590,080
	Batch Norm	3 × 3	(2,2)	28 × 28 × 256	28 × 28 × 256	256
	Convolution 13	3 × 3	1	28 × 28 × 256	28 × 28 × 256	590,080
	Batch Norm	3 × 3	(2,2)	28 × 28 × 256	28 × 28 × 256	256
5	Convolution 14	3 × 3	1	14 × 14 × 256	14 × 14 × 512	1,180,160
	Batch Norm	3 × 3	(2,2)	14 × 14 × 512	14 × 14 × 512	512
	Convolution 15	3 × 3	1	14 × 14 × 512	14 × 14 × 512	2,359,808
	Batch Norm	3 × 3	(2,2)	14 × 14 × 512	14 × 14 × 512	512
	Convolution 16	3 × 3	1	14 × 14 × 512	14 × 14 × 512	2,359,808
	Batch Norm	3 × 3	(2,2)	14 × 14 × 512	14 × 14 × 512	512
	Convolution 17	3 × 3	1	14 × 14 × 512	14 × 14 × 512	2,359,808
	Batch Norm	3 × 3	(2,2)	14 × 14 × 512	14 × 14 × 512	512
	Max Pool	2 × 2	2	14 × 14 × 512	7 × 7 × 512	----
Total number of parameters						10,808,128

The FL takes in an image of size 224 x 224 x 3, which is then convolved at each convolutional operation. The FL internal state can be represented as $(C,\ H,\ W)$ tensor, where C is the channel, H is the height, and W is the width. The final output of the FL is a 512 x 7 x 7 spatially correlated low level feature of the input images, which are then fed to the ViT architecture.

4.3.1.2 Convolutional Vision Transformer

Our Vision Transformer (ViT) component is identical to the ViT architecture described in Section 2.4.5. The FL component and the ViT component makes up our Convolutional Vision Transformer (CViT) model. We named our model CViT since the model is based on both a stack of convolutional operation and the ViT architecture.

The input to the ViT component is a feature map of the face images. The feature maps are split into seven patches and are then embedded into a 1 x 1024 linear sequence. The embedded patches are then added to the position embedding to retain the positional information of the image feature maps. The position embedding has a 2 x 1024 dimension. The patch embedding and the position embedding are shown in Figure 4.9.

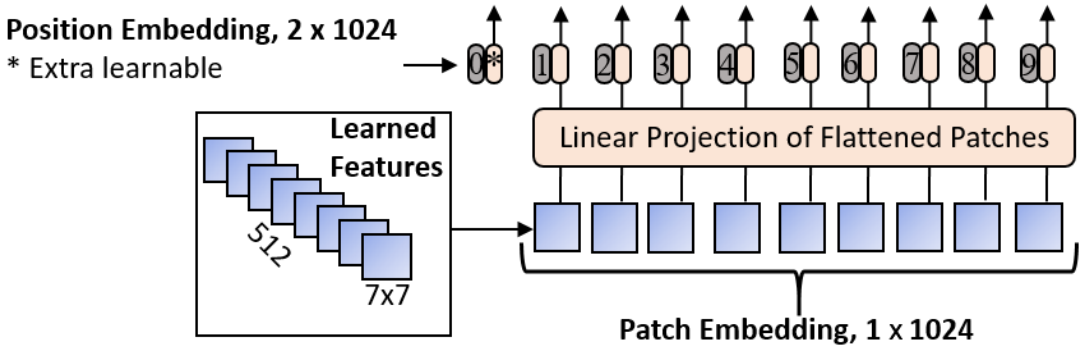

Figure 4.9: Patch and Position Embedding

The ViT component takes in the position embedding and the patch embedding and passes them to the Transformer. The ViT Transformer uses only an encoder, unlike the original Transformer. The ViT encoder consists of MSA and MLP blocks. The MLP block is an FFN. The Norm normalizes the internal layer of the transformer. The Transformer has 8 attention

heads. The MLP head has two linear layers and the ReLU nonlinearity. The MLP head task is equivalent to the fully connected layer of a typical CNN architecture. The first layer has 2048 channels, and the last layer has two channels that represent the class of Fake or Real face image. The CViT model has a total of 20 weighted layers and 38.6 million learnable parameters. Softmax is applied on the MLP head output to squash the weight values between 0 and 1 for the final detection purpose.

4.3.2 The Validation Component

The validation component is similar to that of the training component. The validation component contains the FL and the ViT, which are also the building blocks of the training component. The validation component can be regarded as a part of the training part since they both complement each other.

The validation component is a process that fine-tunes our model. It is used to evaluate our CViT model and helps the CViT model to update its internal state. It helps us to track our CViT model's training progress and its Deepfake detection accuracy. We use the validation component to search for the right CViT model that is optimal for our DFDC dataset. Searching for the optimal model involves checking for different hyperparameters until the best ones are found. Hence, we use the validation component to validate our CViT learning progress and its ability to detecting Deepfakes.

4.3.3 The Testing Component

This section is where we classify and determine the class of the faces extracted in a specific video. Thus, this section addresses our thesis objectives. In the testing component, we added additional padding (see Section 4.2.1) to the extracted face to detect Deepfake artifacts found in the near face region. The face extraction libraries we used in the testing component sometimes wrongly identify nonface regions as a face. To remove those anomalies, we used a filtering mechanism that hasn't been discussed in other Deepfake detection papers before us. In our filtering method, the faces images detected by one face extraction library is re-processed by another detection library to check if it is indeed a face image

The CViT model detects the videos into a fake class if it is a Deepfake video or a real class if the video is not modified. The convolutional layers of our FL component outputs feature maps that are locally correlated. The output of the FL layer is then used by the ViT to extracts distant

feature maps. The ViT output is then squashed by a sigmoid function discussed in Chapter 2 Section 2.1.1 and passes the value to the binary cross-entropy function (a loss function). The binary cross-entropy function then classifies the CViT feature maps.

To determine a specific video is real or fake, the extracted faces are predicted using our trained model. The model predicts each face image class probability. Then, the sigmoid function maps the class probability of the value of the image between 0 and 1. For final classification, an average of each class value is computed, and the class with the highest probability score is considered as the video class.

Chapter 5 : Implementation and Evaluation

In this chapter, we present the tools and experimental setup we used to design and develop the prototype to implement the model. We will present the results acquired from the implementation of the model and give an interpretation of the experimental results.

5.1 Dataset Preparation

DL models learn from data. As such, careful dataset preparation is crucial for their learning quality and prediction accuracy. BlazeFace neural face detector [136], MTCNN [137] and face_recognition [138] DL libraries are used to extract the faces. Both BlazeFace and face_recognition are fast at processing a large number of images. The three DL libraries are used together for added accuracy of face detection. The face images are stored in a JPEG file format with 224 × 224 image resolution. A 90 percent compression ratio is also applied. We prepared our datasets in a train, validation, and test sets. We used 162,174 images classified into 112,378 for training, 24,898 for validation and 24,898 for testing with 75:15:15 ratios, respectively. Each real and fake class has the same number of images in all sets.

We used Albumentations for data augmentation. Albumentations is a python data augmentation library which has a large class of image transformations. Ninety percent of the face images were augmented, making our total dataset to be 308,130 facial images. Figure 5.1 shows the 12 augmentation techniques we used and their corresponding transformed images.

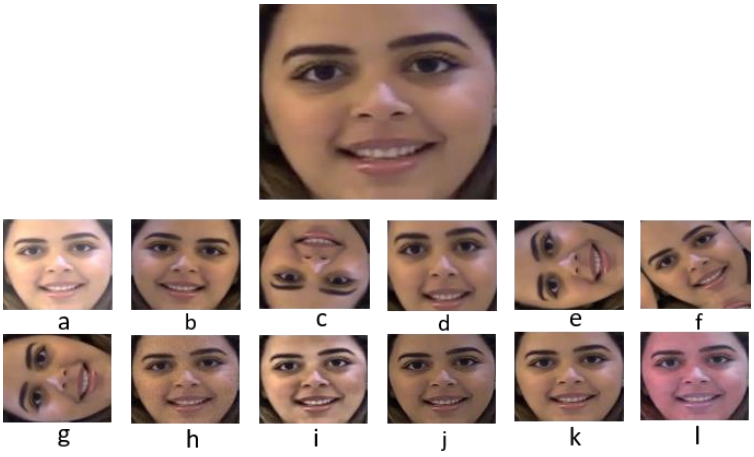

Figure 5.1: Augmentations used in training and validation phase

a) *RandomBrightnessContrast* ***b)*** *HorizontalFlip* ***c)*** *VerticalFlip* ***d)*** *IAAPerspective* ***e)*** *RandomRotate90* ***f)*** *ShiftScaleRotate*

g) *Transpose* ***h)*** *IAAAdditiveGaussianNoise* ***i)*** *CLAHE* ***j)*** *IAASharpen* ***k)*** *IAAEmboss* ***l)*** *HueSaturationValue*

5.2 Tools and Experiment Setup

We used the Python programming language. We chose Python because there are python libraries readily available for all of our DL workflow pipeline. We trained our model using a PyTorch machine learning model. For preprocessing, and training we have used Albumentation, BlazeFace neural face detector, face_recognition, OpenCV-Python (Python bindings designed to solve computer vision problems), and NumPy Python libraries.

We prototyped our CViT model on Kaggle.com and Google Cloud Platform (GCP). Kaggle provides free access to NVIDIA TESLA P100 GPUs for 30 hours per week and 9 hours per session, and TPU v3-8 for 30 hours per week and 3 hours per session. On the GCP, we used NVIDIA Tesla T4 GPU that runs on the Google Deep Learning Image, PyTorch 1.4.0 and fastai, m56, PyTorch 1.4.0 (and fastai) with CUDA 10.0 and Intel® MKL-DNN, and Intel® MKLversion. We used Lenovo Windows 10, Core i7 7 Gen for data acquisition, and face extraction.

We used Kaggles' GPU and TPU to train, validate, and test different CViT models that have optimal learning capacity for our dataset. One epoch takes approximately 13 minutes on TPU, and 33 minutes on GPU to train our dataset. Our training on Kaggle runs for at most 25 iterations, which gives us a clue as to how our model will behave in subsequent epochs. It's possible to save the state of the CViT model for further training. However, we found it inconvenient considering the speed of our network connection. When we think we have found a better performing CViT classifier model, we train it on GCP for a longer epoch, usually 50-100 epochs.

5.3 Model Evaluation

The classification process takes in 30 facial images and passes it to our trained model. To determine the classification accuracy of our model, we used a log loss function. A log loss described in Equation 17 classifies the network into a probability distribution from 0 to 1, where $0>y<0.5$ represents the real class, and $0.5>=y<1$ represents the fake class. We chose a log loss classification metric because it highly penalizes random guesses and confident false predictions. Log loss metrics is used by Facebook's Deepfake Detection Challenge to rank the accuracies of the proposed Deepfake detection models [139, 140].

$$LogLoss = -\frac{1}{n}\sum_{1}^{n}[y_i log\ (\widehat{y}_i) + log(1 - y_i)\ log(1 - \widehat{y}_i)] \quad (17)$$

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

Another metric we used to measure our model capacity is the ROC and AUC metrics [141]. The ROC is used to visualize a classifier to select the classification threshold. AUC is an area covered by the ROC curve. AUC measures the accuracy of a classifier.

5.4 Training and Experiment Results

The overall training processes enables our CViT model to learn from data. The CViT model is trained using the binary cross-entropy loss function. A mini-batch of 32 images are normalized using mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225]. The normalized face images are then augmented before being fed into the CViT model at each training iterations. Adam optimizer with a learning rate of 0.1e-3 and weight decay of 0.1e-6 is used for optimization. The model is trained for a total of 50 epochs. The learning rate decreases by a factor of 0.1 at each step size of 15. Figure 5.2 and 5.3 shows the training and validation progress of our model.

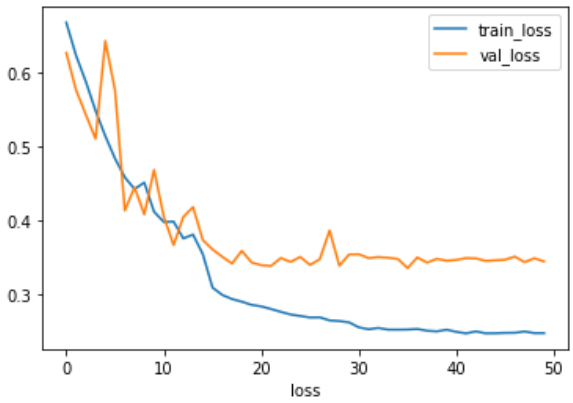

Figure 5.2: train loss and valid loss progression

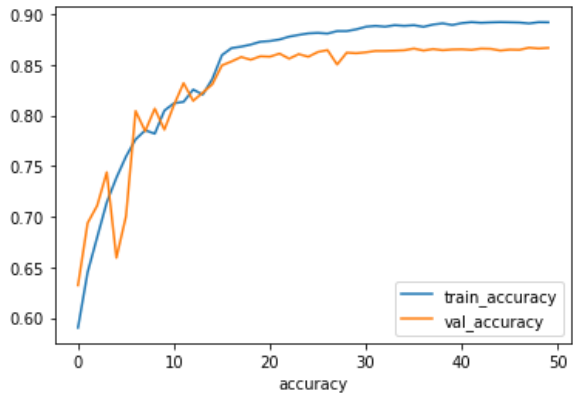

Figure 5.3: train accuracy and valid accuracy progression

We present our result using accuracy, AUC score, and loss value. We tested the model on 400 unseen DFDC videos and achieved 91.5 percent accuracy, an AUC value of 0.91, and a loss value of 0.32. The loss value indicates how far our models' prediction is from the actual target

value. From the result, we can deduce that our model is very good at predicting Deepfakes. As can be seen from Figure 5.4, the model performs well at Deepfake detection.

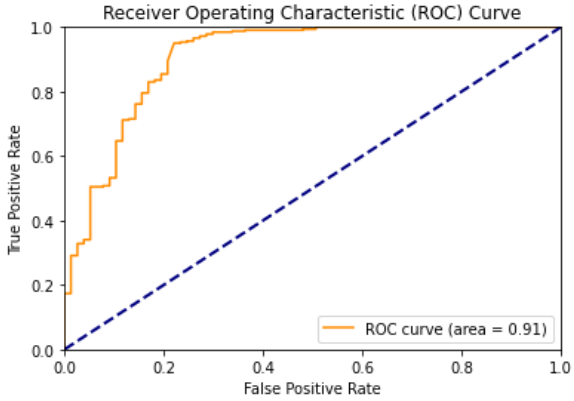

Figure 5.4: ROC curve for 400 unseen DFDC videos

For Deepfake detection, we used 30 face images from each video. The amount of frame number we use affects the chance of Deepfake detection. However, accuracy might not always be the right measure to detect Deepfakes as we might encounter all real facial images from a fake video (fake videos might contain real frames). This anomaly will dupe the model to predict potentially zero accuracies. We have tried to avoid such scenarios by using three face extraction DL libraries in our workflow and picked the best one for our final detection.

We compared our result with other Deepfake detection models, as shown in Table 5.2 and Table 5.3. From Table 5.1, 5.2, and 5.3, we can see that our model performed well on the DFDC, UADFV, and FaceForensics++ dataset. However, our model performed poorly on the FaceForensics++ FaceShifter dataset. The reason for this is because visual artifacts are hard to learn, and our proposed model likely didn't learn those artifacts well.

Table 5-1: CViT model prediction accuracy on FaceForensics++ dataset

Dataset	Accuracy
FaceForensics++ FaceSwap	69%
FaceForensics++ DeepFakeDetection	91%
FaceForensics++ Deepfake	93%
FaceForensics++ FaceShifter	46%

Dataset	Accuracy
FaceForensics++ NeuralTextures	60%

Table 5-2: Accuracy of our model and other Deepfake detection models on the DFDC dataset

Method	Validation	Test
CNN and RNN-GRU GRU [120]	92.61%	91.88%
Our Deepfake Classifier	87.25%	91.5%

Table 5-3: AUC performance of our model and other Deepfake detection models on UADFV dataset

Method	UADFV	FaceForensics++ FaceSwap	FaceForensics++ Face2Face
MesoNet	84.3%	96%	92%
MesoInception	82.4%	98%	93.33
Our CViT Classifier	93.75%	69%	69.39%

5.5 Effects of Data Processing During Classification

A major potential problem that affects our model accuracy is the inherent problems that are in the face detection DL libraries (MTCNN, BlazeFace, and face_recognition). Figure 5.5, Figure 5.6, and Figure 5.7 show images that were misclassified by the DL libraries. The figures summarize our preliminary data preprocessing test on 200 videos selected randomly from 10 folders. We chose our test set video in all settings we can found in the DFDC dataset: indoor, outdoor, dark room, bright room, subject sited, subject standing, speaking to side, speaking in front, a subject moving while speaking, gender, skin color, one person video, two people video, a subject close to the camera, and subject away from the camera. For the preliminary test, we extracted every frame of the videos and found the 637 nonface region.

Figure 5.5: MTCNN non face region detection

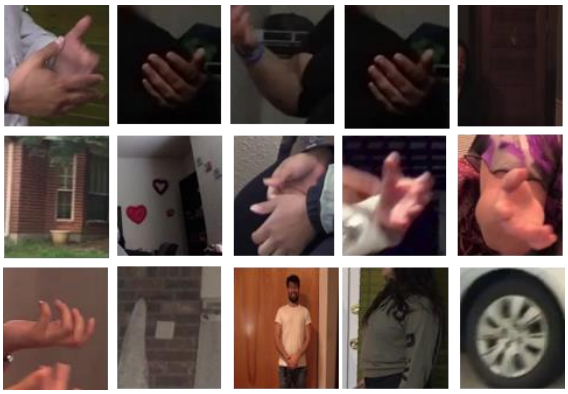

Figure 5.6: BlazeFace non face region detection

Figure 5.7: face_recognition non face region detection

We tested our model to check how its accuracy is affected without any attempt to remove these images, and our models' accuracy dropped to 69.5 percent, and the loss value increased to 0.4, as shown in Figure 5.8.

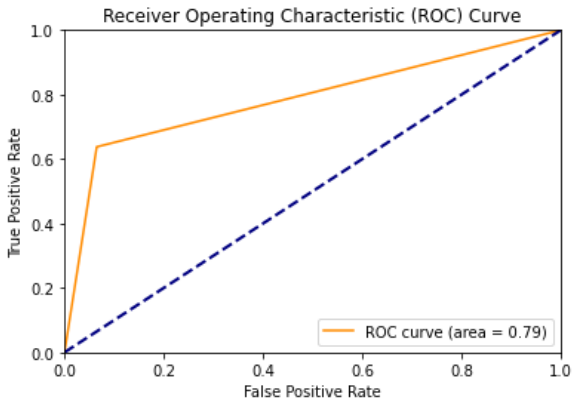

Figure 5.8: ROC Curve with no data preprocessing

To minimize nonface regions and prevent wrong predictions, we used the three DL libraries and picked the best performing library for our model, as shown in Table 5-4. As a solution, we used face_recognition as a "filter" for the face images detected by BlazeFace. We chose face_recognition because, in our investigation, it rejects more false-positive than the other two models. In our test, face_recognition performs better than Blazeface and MTCNN library. Hence, we used face_recognition for final Deepfake detection.

Table 5-4: DL libraries comparison on Deepfake detection accuracy

Dataset	Blazeface	Face_recognition	MTCNN
DFDC	83.40%	**91.50%**	90.25%
FaceForensics++ FaceSwap	56%	**69%**	63%
FaceForensics++ FaceShifter	40%	**46%**	44%
FaceForensics++ NeuralTextures	57%	**60%**	60%
FaceForensics++ DeepFakeDetection	82%	**91%**	79.59
FaceForensics++ Deepfake	87%	**93%**	81.63%
FaceForensics++ Face2Face	54%	61%	**69.39%**
UADF	74.50%	**93.75%**	88.16%

Chapter 6 : Conclusion and Future Work

6.1 Conclusion

In this thesis work, we have discussed the general concepts of DL models that laid the foundation for advanced image and video manipulations. CNNs, GANs, AEs, and their variants have made it possible to generate and synthesize hyper-realistic images and videos, which collectively are known as Deepfakes. Transformers have also proven to be a powerful DL learning model both in NLP tasks and CV. Deepfakes open new possibilities in digital media, VR, robotics, education, and many other fields. On another spectrum, they are technologies that can cause havoc and distrust to the general public. Their potential use as a weapon to cause harm to others in the form of identity theft, political propaganda, and appearance in adult content without consent calls for ways to detect such ill formed contents.

In light of this, we have designed and developed a generalized model for Deepfake video detection using CNNs and Transformer, which we named Convolutional Vison Transformer. We called our model a generalized model for three reasons. 1) Our first reason arises from the combined learning capacity of CNNs and Transformer. CNNs are strong at learning local features, while Transformers can learn from local and global feature maps. This combined capacity enables our model to correlate every pixel of an image and understand the relationship between nonlocal features. 2) We gave equal emphasis on our data preprocessing during training and classification. 3) We used the largest and most diverse dataset for Deepfake detection. The model was implemented using Pytorch and trained on Tesla T4 GPU. The CViT model was trained on a diverse collection of facial images that were extracted from the DFDC dataset. The model was tested on 400 DFDC videos and has achieved an accuracy of 91.5 percent. Our new Deepfake detection model has the potential to be used in the detection of Deepfakes intended for malicious intents. Our proposed CViT model is efficient because it's trained on a diverse set of videos recorded in various environment settings, light conditions, face orientations, and skin color.

Our proposed CViT model sheds some hope on our ability to protect one another from unwanted harm and keep us safe. Still, our model has a lot of room for improvement. In the future, we intend to expand on our current work by adding other datasets released for Deepfake research to make it more diverse, accurate, and robust.

6.2 Contribution

The contribution of this thesis work are:

- ✓ In this work, we presented a Convolutional Vision Transformer, a new model that adds a CNN layer to Vision Transformer.
- ✓ We tested our new model on Deepfake detection and have achieved a competitive result.
- ✓ We have shown that the choice of face detection algorithms can directly affect a Deepfake detection model's accuracy. We have compared three DL libraries on the DFDC, UADFV, and FaceForensics++ dataset and found that the face_recognition DL library performs better on those datasets for Deepfake detection.

6.3 Future Work

Current Deepfake videos contain untampered images within their frames, as in the DFDC dataset. Filtering those real images manually or using some other form of image comparison technique is not powerful enough compared to DL techniques. It would be highly convenient if the current Deepfake detection models are included in the existing face extraction DL libraries to facilitate Deepfake detection preprocessing workflow.

In the future, we intend to expand our training dataset to include other datasets.

References

[1] Bobby Chesney and Danielle Citron, "Deep Fakes A Looming Challenge for Privacy, Democracy, and National Security," 2019. [Online]. Available: https://ssrn.com/abstract=3213954. [Accessed 28 October 2019].

[2] Lilei Zheng, Ying Zhang and Vrizlynn L.L. Thing, "A survey on image tampering and its detection in real-world photos," *Elsevier,* vol. 58, pp. 380-399, 2018.

[3] Ali Khodabakhsh , Raghavendra Ramachandra, Kiran Raja and Pankaj Wasnik, "Fake Face Detection Methods: Can They Be Generalized?," in *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*, Darmstadt, 2018.

[4] Pavel Korshunov and Sebastien Marcel, "DeepFakes: a New Threat to Face Recognition? Assessment and Detection," arXiv preprint arXiv:1812.08685v1, December 2018.

[5] Darius Afchar, Vincent Nozick, Junichi Yamagishi and Isao Echizen, "MesoNet: a Compact Facial Video Forgery Detection Network," arXiv preprint rXiv:1809.00888v1, September 2018.

[6] Avishek Joey Bose and Parham Aarabi , "Virtual Fakes: DeepFakes for Virtual Reality," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2019, pp. 1-1.

[7] Umur Aybars Ciftci, Ilke Demir and Lijun Yin, "FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals," arXiv preprint rXiv:1901.02212v2, August 2019.

[8] Thanh Thi Nguyen, Cuong M. Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen and Saeid Nahavandi, "Deep Learning for Deepfakes Creation and Detection," arXiv preprint arXiv:1909.11573v1, September 2019.

[9] Supasorn Suwajanakorn, Steven M. Seitz and Ira Kemelmacher-Shlizerman, "Synthesizing Obama: Learning Lip Sync from Audio," *ACM Transactions on Graphics,* vol. 36, no. 4, July 2017.

[10] John Brandon, "Terrifying high-tech porn: Creepy ‘deepfake’ videos are on the rise," Fox News, 16 February 2018. [Online]. Available: https://www.foxnews.com/tech/terrifying-high-tech-porn-creepy-deepfake-videos-are-on-the-rise. [Accessed 25 October 2019].

[11] Sakshi Agarwal and Lav R. Varshney, "Limits of Deepfake Detection: A Robust Estimation Viewpoint," arXiv:1905.03493v1, May 2020.

[12] Joshua Brockschmidt, Jiacheng Shang and Jie Wu, "On the Generality of Facial Forgery Detection," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, Monterey, CA, USA, IEEE, 2019, pp. 43-47.

[13] Yuezun Li, Ming-Ching Chang and Siwei Lyu, "In Ictu Oculi: Exposing A Generated Fake Face Videos by Detecting Eye Blinking," arXiv preprint arXiv:1806.02877v2, 2018.

[14] TackHyun Jung, SangWon Kim and KeeCheon Kim, "DeepVision: Deepfakes detection using human eye blinking pattern," *IEEE Access,* vol. 8, pp. 83144-83154, April 2020.

[15] Konstantinos Vougioukas, Stavros Petridis and Maja Pantic, "Realistic Speech-Driven Facial Animation with GANs," *International Journal of Computer Vision,* vol. 128, p. 1398–1413, May 2020.

[16] Hai X. Pham, Yuting Wang and Vladimir Pavlovic, "Generative Adversarial Talking Head: Bringing Portraits to Life with a Weakly Supervised Neural Network," arXiv:1803.07716, March 2018.

[17] Polychronis Charitidis, Giorgos Kordopatis-Zilos, Symeon Papadopoulos and Ioannis Kompatsiaris, "A Face Preprocessing Approach For Improved Deepfake Detection," arXiv:2006.07084v1 , June 2020.

[18] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, Nevada, USA, Curran Associates, Inc., 2012, pp. 1097-1105.

[19] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," in *the 2nd International Conference on Learning Representations (ICLR 2015)*, San Diego, April 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, June 2016.

[21] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen and Baining Guo, "Face X-ray for More General Face Forgery Detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, IEEE, August 2020, pp. 5000-5009.

[22] Joao C. Neves, Ruben Tolosana, Ruben Vera-Rodriguez, Vasco Lopes, Hugo Proenca and Julian Fierrez, "GANprintR: Improved Fakes and Evaluation of the State of the Art in Face Manipulation Detection," arXiv:1911.05351v4, July 2020.

[23] Tero Karras, Samuli Laine and Timo Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," arXiv:1812.04948, December 2018.

[24] Tero Karras, Samuli Laine and Timo Aila, "This Person Does Not Exist," NVIDIA, December 2019. [Online]. Available: https://thispersondoesnotexist.com/. [Accessed 09 10 2020].

[25] OpenAI, "OpenAI API," OpenAI, 11 06 2020. [Online]. Available: https://openai.com/blog/openai-api/. [Accessed 09 10 2020].

[26] Will Douglas Heaven, "A GPT-3 bot posted comments on Reddit for a week and no one noticed," MIT Technology Review, 08 10 2020. [Online]. Available: https://www.technologyreview.com/2020/10/08/1009845/a-gpt-3-bot-posted-comments-on-reddit-for-a-week-and-no-one-noticed/. [Accessed 09 10 2020].

[27] Xin Yang, Yuezun Li and Siwei Lyu, "Exposing Deep Fakes Using Inconsistent Head Poses," arXiv preprint arXiv:1811.00661v2, November 2018.

[28] Yuezun Li and Siwei Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts," arXiv preprint arXiv:1811.00656v3, May 2019.

[29] Falko Matern, Christian Riess and Marc Stamminger, "2019 Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations," in *IEEE Winter Applications of Computer Vision Workshops (WACVW)*, Hawaii, January 2019.

[30] David Guera and Edward J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, New Zealand, November 2018.

[31] Haya R. Hasan and Khaled Salah, "Combating Deepfake Videos Using Blockchain and Smart Contracts," *IEEE Access,* vol. 7, pp. 41596 - 41606, March 2019.

[32] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang and Cristian Canton Ferrer, "The DeepFake Detection Challenge Dataset," arXiv:2006.07397v3, June 2020.

[33] Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram and Cristian Canton Ferrer, "The Deepfake Detection Challenge (DFDC) Preview Dataset," arXiv:1910.08854v2, October 2019.

[34] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies and Matthias Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images," arXiv preprint arXiv:1901.08971v3, August 2019.

[35] Licheng Jiao and Jin Zhao, "A Survey on the New Generation of DeepLearning in Image Processing," *IEEE Access,* vol. 7, Dec 12, 2019.

[36] Yann LeCun, Yoshua Bengio and Geoffrey Hinton, "Deep learning," *Nature,* vol. 521, 28 May 2015.

[37] Md Zahangir Alom, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal and Vijayan K. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics,* vol. 8, no. 3, p. 292, March 2019 .

[38] Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni and Silas Franco dos Reis Alves, Artificial Neural Networks A Practical Course, Switzerland: Springer Nature, August 24, 2016.

[39] Moacir A. Ponti, Leonardo S. F. Ribeiro, Tiago S. Nazare, Tu Bui and John Collomosse, "Everything you wanted to know about Deep Learning for Computer Vision but were afraid to ask," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, Niterói, 2017.

[40] Mohammad-Parsa Hosseini, Senbao Lu, Kavin Kamaraj, Alexander Slowikowski and Haygreev C. Venkatesh, "Deep Learning Architectures," in *Deep Learning: Concepts and Architectures*, vol. 866, Switzerland, Springer Nature , 2020, pp. 1-24.

[41] Darpan Pandey, Kamal Niwaria and Bharti Chourasia, "Machine Learning Algorithms: A Review," *International Research Journal of Engineering and Technology,* vol. 06, no. 02, pp. 916-922, February 2019.

[42] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco-Hernandez, Lenka Krpalkova, Daniel Riordan and Joseph Walsh, "Deep Learning vs. Traditional Computer Vision," in *Proceedings of the 2019 Computer Vision Conference (CVC)*, Las Vegas, 2019.

[43] M. Arif Wani, Farooq Ahmad Bhat, Saduf Afzal and Asif Iqbal Khan, Advances in Deep Learning, Singapore: Springer Nature, 2020.

[44] Abhijit Ghatak, Deep Learning with R, Singapore: Springer Nature, 2019.

[45] Zhaoqiang Xia, "An Overview of Deep Learning," in *Deep Learning in Object Detection and Recognition*, Singapore, Springer Nature, 2019, pp. 1-18.

[46] M. Arif Wani, Mehmed Kantardzic and Moamar Sayed-Mouchaweh, "Trends in Deep Learning Applications," *Deep Learning Applications,* vol. 1098, pp. 1-7, 2020.

[47] Muhammad Shahroz Nadeem, Virginia N. L. Franqueira, Xiaojun Zhai and Fatih Kurugollu, "A Survey of Deep Learning Solutions forMultimedia Visual Content Analysis," *IEEE Access,* vol. 7, May 7, 2019.

[48] Mohamed Alloghani, Dhiya Al-Jumeily, Jamila Mustafina, Abir Hussain and Ahmed J. Aljaaf, "A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science," in *Supervised and Unsupervised Learning for Data Science*, Switzerland, Springer Nature, 2020, pp. 3-21.

[49] Ajay Shrestha and Ausif Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access,* vol. 7, pp. 53040-53065, 2019.

[50] Seyedali Mirjalili, Hossam Faris and Ibrahim Aljarah, "Introduction to Evolutionary Machine Learning Techniques," in *Evolutionary Machine Learning Techniques Algorithms and Applications*, Singapore, Springer Nature , 2020, pp. 1-7.

[51] K. Sindhu Meena and S. Suriya, "A Survey on Supervised and Unsupervised Learning Techniques," in *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*, Switzerland, Springer Nature, 2019, pp. 627-642.

[52] Lars Schmarje, Monty Santarossa, Simon-Martin Schroder and Reinhard Koch, "A Survey on Semi-, Self- and Unsupervised Learning for Image Classification," July 2020.

[53] Yulong Lu and Jianfeng Lu, "A Universal Approximation Theorem of Deep Neural Networks for Expressing Distributions," arXiv:2004.08867v2, April 2020.

[54] Jesper E. van Engelen and Holger H. Hoos , "A survey on semi-supervised learning," *Machine Learning,* vol. 109, pp. 373-440, 2020.

[55] Gilbert Strang, Linear Algebra and Learning from Data, Wellesley: Wellesley- Cambridge Press, 2019 .

[56] Mohit Goyal, Rajan Goyal, P. Venkatappa Reddy and Brejesh Lall, "Activation Functions," in *Deep Learning: Algorithms and Applications*, Switzerland, Springer Nature, 2020, pp. 1-28.

[57] Eugene Charniak, Introduction to Deep Learning, London: MIT Press, 2018.

[58] Piyush Kaul and Brejesh Lall, "Theoretical Characterization of Deep Neural Networks," in *Deep Learning: Concepts and Architectures*, Switzerland, Springer Nature, 2020, pp. 25-60.

[59] Neha Yadav, Anupam Yadav and Manoj Kumar, An Introduction to Neural Network Methods for Differential Equations, Springer, 2015.

[60] Paul Wilmott, Machine Learning an Applied Mathematics Introduction, Panda Ghana Publishing, 2019.

[61] Jeff Heaton, Artificial Intelligence for Humans, Volume 3: Neural Networks and Deep Learning, Heaton Research,, December 31 2015.

[62] Fábio M. Soares and Alan M.F. Souza, Neural Network Programming with Java, Birmingham: Packt Publishing, 2016 .

[63] Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016.

[64] Sandro Skansi, Introduction to Deep Learning From Logical Calculus to Artificial Intelligence, Springer International Publishing, 2018.

[65] Charu C. Aggarwal, Neural Networks and Deep Learning: A Textbook, Switzerland: Springer International Publishing, September 2018.

[66] Denis Rothman, Artificial Intelligence By Example, Birmingham: Packt Publishing, 2020.

[67] Terrence J. Sejnowski, The Deep Learning Revolution, London,: MIT Press, 2018.

[68] Leonardo De Marchi and Laura Mitchell, Hands-On Neural Networks, Birmingham: Packt Publishing, May 2019.

[69] Stuart J. Russell and Peter Norvig, Artificial Intelligence A Modern Approach Fourth Edition, Pearson, 2020.

[70] Ovidiu Calin, Deep Learning Architectures A Mathematical Approach, Switzerland: Springer Nature, 2020.

[71] C.-C. Jay Kuo, "Understanding Convolutional Neural Networks with A Mathematical Model," *Journal of Visual Communication and Image Representation,* vol. 41, pp. 406-413, November 2016.

[72] Leonardo De Marchi and Laura Mitchell, Hands-On Neural Networks, Birmingham: Packt Publishing, 2019.

[73] Andriy Burkov, The Hundred-page Machine Learning Book, Andriy Burkov, 2019.

[74] Neena Aloysius and Geetha M, "A Review on Deep Convolutional Neural Networks," in *International Conference on Communication and Signal Processing*, India, 2017.

[75] Umberto Michelucci, Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection, Apress, 2019.

[76] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale and Orlando De Jesús, Neural Network Design 2nd Edtion, Martin Hagan, 2014.

[77] El-Ghazali Talbi, "Optimization of deep neural networks: a survey andunified taxonomy," hal-02570804v2 , June 2020.

[78] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak and Shahrokh Valaee, "Recent Advances in Recurrent Neural Networks," arXiv:1801.01078v3, February 2018.

[79] Zachary C. Lipton and John Berkowitz, "A Critical Review of Recurrent Neural Networks for Sequence Learning," arXiv:1506.00019v4, October 2015.

[80] Yong Yu , Xiaosheng Si, Changhua Hu and Jianxun Zh, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation,* vol. 31, no. 7, pp. 1235 - 1270, July 2019.

[81] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," arXiv:1412.3555v1, Daecember 2014.

[82] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," arXiv:1406.1078, June 2014.

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin, "Attention Is All You Need," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017.

[84] Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer and Hannaneh Hajishirzi, "DeLighT: Very Deep and Light-weight Transformer," arXiv:2008.00623v1, August 2020.

[85] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv:1409.0473, May 2014.

[86] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath and Varun Mithal, "An Attentive Survey of Attention Models," arXiv:1904.02874v1, April 2019.

[87] Albert Zeyer, André Merboldt, Ralf Schlüter and Hermann Ney, "A comprehensive analysis on attention models," in *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, Canada.

[88] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit and Neil Houlsby, "AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE," arXiv:2010.11929v1, October 2020.

[89] Taesung Park, Ming-Yu Liu, Ting-Chun Wang and Jun-Yan Zhu, "Semantic Image Synthesis with Spatially-Adaptive Normalization," arXiv:1903.07291v2, 2019.

[90] Achraf Oussidi and Azeddine Elhassouny , "Deep Generative Models: Survey," in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, IEEE, 2018, pp. 1-8.

[91] Carl Doersch, "Tutorial on Variational Autoencoders," arXiv:1606.05908v2, 2016.

[92] Harshvardhan GM, Mahendra Kumar Gourisaria, Manjusha Pandey and Siddharth Swarup Rautaray, "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review,* vol. 38, Nov 2020.

[93] Dor Bank, Noam Koenigstein and Raja Giryes, "Autoencoders," arXiv:2003.05991v1, March 2020.

[94] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozairy, Aaron Courville and Yoshua Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Montreal, 2014.

[95] Yongjun Hong, Uiwon Hwang, Jaeyoon Yoo and Sungroh Yoon, "How Generative Adversarial Networks and Their Variants Work: An Overview," *ACM Computing Surveys (CSUR),* vol. 52, no. 1, February 2019.

[96] Xian Wu, Kun Xu and Peter Hall, "A Survey of Image Synthesis and Editing with Generative Adversarial Networks," *TSINGHUA SCIENCE AND TECHNOLOGY,* vol. 22, no. 6, pp. 660-674, 2017.

[97] Ceren Güzel Turhan and Hasan Sakir Bilge , "Recent Trends in Deep Generative Models: a Review," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, IEEE, 2018, pp. 574-579.

[98] Siwei Lyu, "Deepfake Detection: Current Challenges and Next Steps," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, vol. 1, London, IEEE Computer Society, 2020, pp. 1-6.

[99] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou and Weiming Zhang, "DeepFaceLab: A simple, flexible and extensible face swapping framework," arXiv:2005.05535, May 2020.

[100] Yisroel Mirsky and Wenke Lee, "The Creation and Detection of Deepfakes: A Survey," arXiv:2004.11138v3, September 2020.

[101] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales and Javier Ortega-Garcia, "DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection," arXiv:2001.00179v3, June 2020.

[102] Yuval Nirkin, Yosi Keller and Tal Hassner, "FSGAN: Subject Agnostic Face Swapping and Reenactment," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 7184-7193.

[103] Prajwal K R, Rudrabha Mukhopadhyay, Jerin Philip, Abhishek Jha, Vinay Namboodiri and C V Jawahar, "Towards Automatic Face-to-Face Translation," in *the 27th ACM International Conference on Multimedia (MM '19)*, New York, 2019.

[104] K R Prajwal, Rudrabha Mukhopadhyay, Vinay P. Namboodiri and C V Jawahar, "A Lip Sync Expert Is All You Need for Speech to Lip Generation," arXiv:2008.10010v1, Aug 2020.

[105] Sourabh Dhere, Suresh B. Rathod , Sanket Aarankalle, Yash Lad and Megh Gandhi, "A Review on Face Reenactment Techniques," in *2020 International Conference on Industry 4.0 Technology (I4Tech)*, Pune, India, February 2020 .

[106] Mike Price and Matt Price, "Playing Offense and Defense with Deepfakes," Zerofox, Las Vegas, 2019.

[107] Xun Huang, Ming-Yu Liu, Serge Belongie and Ming-Yu Liu, "Multimodal Unsupervised Image-to-Image Translation," arXiv:1804.04732, Aug 2018.

[108] Kuniaki Saito, Kate Saenko and Ming-Yu Liu, "Few-Shot Unsupervised Image Translation witha Content Conditioned Style Encoder," arXiv:2007.07431v3, July 2020.

[109] Arushi Handa, Prerna Garg and Vijay Khare, "Masked Neural Style Transfer using Convolutional Neural Networks," in *International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering - (ICRIEECE)*, 2018.

[110] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz and Bryan Catanzaro, "Video-to-Video Synthesis," arXiv:1808.06601, December 2018.

[111] Arun Mallya, Ting-Chun Wang, Karan Sapra and Ming-Yu Liu, "World-Consistent Video-to-Video Synthesis," arXiv:2007.08509, July 2020.

[112] Brais Martinez, Michel F. Valstar, Bihan Jiang and Maja Pantic, "Automatic Analysis of Facial Actions: A Survey," *IEEE Transactions on Affective Computing,* vol. 10, no. 3, pp. 325-347, 2019.

[113] He Huang, Phillip S. Yu and Changhu Wang, "An Introduction to Image Synthesis with Generative Adversarial Nets," arXiv:1803.04469v1, March 2018.

[114] Fathima Shirin M and Meharban M.S, "Review on Image Synthesis Techniques," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, India, March 2019.

[115] Evangelos Ververas and Stefanos Zafeiriou , "SliderGAN: Synthesizing Expressive Face Images by Sliding 3D Blendshape Parameters," *International Journal of Computer Vision volume,* vol. 128, p. 2629–2650, June.

[116] Jun Yu,, Xingxin Xu, Fei Gao, Shengjie Shi, Meng Wang, Dacheng Tao and Qingming Huang, "Towards Realistic Face Photo-Sketch Synthesis via Composition-Aided GANs," arXiv:1712.00899v4, January 2020.

[117] Zhifei Zhang, Yang Song and Hairong Qi, "Age Progression/Regression by Conditional Adversarial Autoencoder," arXiv:1702.08423, March 217.

[118] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano and Hao Li, "Protecting World Leaders against deep fakes.," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, California, 2019.

[119] Ryota Natsume, Tatsuya Yatagawa and Shigeo Morishima, "RSGAN: Face Swapping and Editing using Face and Hair Representation in Latent Spaces," arXiv:1804.03447, April 2018.

[120] Daniel Mas Montserrat, Hanxiang Hao, S. K. Yarlagadda, Sriram Baireddy, Ruiting Shao, Janos Horvath, Emily Bartusiak, Justin Yang, David Guera, Fengqing Zhu and Edward J. Delp,

"Deepfakes Detection with Automatic Face Weighting," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA, IEEE, 2020, pp. 2851-2859.

[121] Xin Zheng, Yanqing Guo, Huaibo Huang, Yi Li and Ran He, "A Survey of Deep Facial Attribute Analysis," *International Journal of Computer Vision,* vol. 128, p. 2002–2034, March 2020.

[122] Hasam Khalid and Simon S. Woo, "OC-FakeDect: Classifying Deepfakes Using One-class Variational Autoencoder," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA, IEEE, 2020, pp. 2794-2803.

[123] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," arXiv:1703.10593, August 2017.

[124] J. Thies, M. Zollhofer and M. Nießner, "Deferred Neural Rendering: Image Synthesis using Neural Textures," arXiv:1904.12356v1, April 2019.

[125] Rahul Haridas and Jyothi R L, "Convolutional Neural Networks: A Comprehensive Survey," *International Journal of Applied Engineering Research (IJAER),* vol. 14, no. 03, pp. 780-789, 2019.

[126] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang and Wenzhe Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," arXiv preprint arXiv:1609.04802v5, May 2017.

[127] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov and Victor Lempitsky, "Few-Shot Adversarial Learning of Realistic Neural Talking Head Models," arXiv preprint arXiv:1905.08233v2, September 2019.

[128] Chris Donahue , Julian McAuley and Miller Puckette , "Adversarial Audio Synthesis," in *Seventh International Conference on Learning Representations (ICLR 2019)*, New Orleans, 2019.

[129] Huy H. Nguyen, Ngoc-Dung T. Tieu, Hoang-Quoc Nguyen-Son, Vincent Nozick, Junichi Yamagishi and Isao Echizen, "Modular Convolutional Neural Network for Discriminating between Computer-Generated Images and Photographic Images," in *the 13th International Conference on Availability, Reliability and Security (ARES 2018)*, Hamburg, August 2018.

[130] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt and Matthias Nießner, "Face2Face: Real-time Face Capture and Reenactment of RGB Videos," *Communications of the ACM,* vol. 62, no. 1, pp. 96-104, January 2019.

[131] Huy H. Nguyen, Junichi Yamagishi and Isao Echizen, "Capsule-Forensics: Using Capsule Networks To Detect Forged Images And Videos," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, Brighton, May 2019.

[132] Ivana Chingovska, Andre Anjos and Sebastien Marcel, "On the Effectiveness of Local Binary Patterns in Face Anti-spoofing," in *the International Conference of Biometrics Special Interest Group (BIOSIG)*, Darmstadt, September 2012.

[133] Nicolas Rahmouni, Vincent Nozick, Junichi Yamagishi and Isao Echizen, "Distinguishing Computer Graphics from Natural Images Using Convolution Neural Networks," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, Rennes, December 2017.

[134] Md. Shohel Rana and Andrew H. Sung, "DeepfakeStack: A Deep Ensemble-based Learning Technique for Deepfake Detection," in *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, New York City, May, 2020.

[135] Junyaup Kim, Siho Han and Simon S. Woo, "Classifying Genuine Face images from Disguised Face Images," in *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, 2019.

[136] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran and Matthias Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," arXiv:1907.05047v2, Jul 2019.

[137] Timesler, "Pretrained Pytorch face detection (MTCNN) and recognition (InceptionResnet) models," [Online]. Available: https://github.com/timesler/facenet-pytorch. [Accessed 6 10 2020].

[138] Adam Geitgey, "The world's simplest facial recognition api for Python and the command line," Github, [Online]. Available: https://github.com/ageitgey/face_recognition. [Accessed 24 September 2020].

[139] Cristian Canton, "Deepfake Detection Challenge," Facebook, 11 December 2019. [Online]. Available: https://www.kaggle.com/c/deepfake-detection-challenge/overview/evaluation. [Accessed 1 October 2020].

[140] Katarzyna Janocha and Wojciech Marian Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," arXiv:1702.05659v1, February 2017.

[141] Andrew P.Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition,* vol. 30, no. 7, pp. 1145-1159, July 1997.

[142] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167v3, 2015.

[143] Connor Shorten and Taghi M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big data,* vol. 6, 2019.

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: DERESSA WODAJO DERESSA

Signature: _____

Date: November 28, 2020_____

Confirmed by advisor:

Name:

Signature: _____

Date: _____