

UNITEC INSTITUTE OF TECHNOLOGY

A THESIS PRESENTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTING

**A Framework for Analysis and Comparison
of Deepfakes Detection Methods**

Student: Changjin WANG, 1494269

Principal Supervisor: Associate Professor Hamid SHARIFZADEH

Co-supervisors: Dr. Rachel FLEMING, Associate Professor Iman ARDEKANI

March 30, 2021

Abstract

With the rise of AI (Artificial Intelligence), people can already utilise Deepfakes technology to generate fake pictures and videos increasingly. Similar to all technologies, while bringing benefits, this technology also has its downsides, such as spreading false information and endangering public interests. In order to combat the harm of fake-face videos, researchers have proposed a variety of different deep forgery detection algorithms, and have achieved remarkable results. However, a common problem regarding these detection methods are that in-library detection can normally achieve high accuracy, but the performance severely degraded in cross-library detection. That is to say, there is a serious problem of insufficient generalisation ability.

The current mainstream detection method is to train a binary classification model on real videos and fake videos, and classify real videos and tampered videos through a classifier to distinguish true and false. In other words, Deepfakes detection is based on the evaluation criteria of the binary classification model to evaluate the performance of various detection methods. However, each evaluation criteria will have a different emphasis on different application scenarios and technical requirements. That is to say, using only some kind of single evaluation criteria cannot effectively compare the performance of different detection methods. In current literature, Deepfakes detection usually uses only the Area Under Curve (AUC) in the binary classification model as the evaluation standard. Nevertheless, AUC focuses on only the relative size of the probability value, and it does not consider the absolute size of the threshold and probability value. Moreover, when the data is very uneven, AUC may not properly assess the performance of the detection method.

To better compare the performance differences between various detection methods, this thesis provides analysis and comparison on the six Deepfakes detection methods of Two-stream, MesoNet, HeadPose, FWA, VA and Multi-task. To assess generalisation ability of these methods, I conduct intra-library and cross-library tests on the existing three fake face video datasets. For this purpose, accuracy (ACC) and error rate are used as evaluation criteria in this thesis and I will focus on analysing the impacts of three factors (dataset partitioning, data augmentation operations, and the detection threshold selection) on the generalisation ability of the Deepfakes detection methods.

Thus, the principal contributions of this thesis outline as follows: 1) an analytical cross-library platform for comparison of Deepfakes detection methods; 2) proposing new evaluation metrics based on data partitioning, augmentation, and threshold selection; 3) providing an overall performance indication of detection methods based on their generalisation performance on different data types.

Contents

Abstract	ii
1 Introduction	1
1.1 Overview of Deepfakes	1
1.2 Impacts of Deepfakes	3
1.3 Overview of Deepfakes Detection	4
1.4 Purpose of Deepfakes Detection	5
1.5 Thesis Structure	5
2 Literature Review	7
2.1 Deepfakes Creation	7
2.1.1 Software Applications Used for Deepfakes Creation	7
2.1.1.1 Fakeapp	7
2.1.1.2 Faceswap	8
2.1.1.3 Openfaceswap	9
2.1.1.4 DeepFaceLab	9
2.1.2 Face-swapping	10
2.1.2.1 Original Deepfakes	11
2.1.2.2 Faceswap-GAN	12
2.1.2.3 Fast Face-swap Using CNN [29]	13
2.1.2.4 On Face Segmentation, Face Swapping, and Face Perception [30]	14
2.1.2.5 FSGAN: Subject Agnostic Face Swapping and Reenactment [31]	15

2.1.2.6	FaceShifter: Towards High Fidelity and Occlusion Aware Face Swapping [32]	15
2.1.3	Face-reenactment	16
2.1.3.1	Face2face: Real-time face capture and reenactment of rgb videos [33]	16
2.1.3.2	Headon: Real-time reenactment of human portrait videos [34]	17
2.1.3.3	Synthesising obama: learning lip sync from audio [35]	18
2.1.3.4	Deep video portraits [36]	19
2.1.3.5	Few-Shot Adversarial Learning of Realistic Neural Talking Head Models [37]	19
2.1.3.6	Text-based Editing of Talking-head Video [38]	20
2.1.4	Summary	21
2.2	Deepfakes Detection	21
2.2.1	Fake Image Detection	22
2.2.1.1	Automated face swapping and its detection [40]	22
2.2.1.2	Two-Stream Neural Networks for Tampered Face Detection [46]	23
2.2.1.3	Multitask Learning for Detecting and Segmenting Manipulated Facial Images and Videos [49]	25
2.2.2	Fake Video Detection	25
2.2.2.1	Temporal Features Across Frames	26
2.2.2.1.1	Deepfake Video Detection Using Recurrent Neural Networks [54]	26
2.2.2.1.2	In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking [56]	27
2.2.2.1.3	Recurrent Convolutional Strategies for Face Manipulation Detection in Videos [59]	28
2.2.2.2	Visual Artifacts within Frames	29
2.2.2.2.1	Deep Classifiers	29
2.2.2.2.2	Shallow Classifiers	34

2.2.3	Summary and Research Gap	45
3	Deepfakes Datasets	48
3.1	UADFV	48
3.2	TIMIT	48
3.3	FaceForensics ++	49
3.4	DFD	50
3.5	DFDC	50
3.6	Celeb-DF	51
3.7	Comparison and Summary of Datasets	52
4	Methodology	54
4.1	Proposed Data Benchmark	54
4.2	Proposed Comparison Framework	55
4.2.1	Data Augmentation	56
4.2.2	Data Partitioning	58
4.2.3	Thresholding Selection	60
4.2.4	Accuracy Rate and Error Rate	62
4.2.4.1	Accuracy Rate (ACC)	62
4.2.4.2	Error Rate	64
5	Experimental Results and Analysis	67
5.1	Implementation Details	68
5.2	Limitation	70
5.3	Scenario Settings	71
5.3.1	Data Processing	71
5.3.2	Datasets Processing	72
5.3.3	Selection of Thresholds	74
5.4	Performance Analysis	74
5.4.1	Standard Condition	74
5.4.2	Data Augmentation	76
5.4.3	Dataset Partitioning	78

5.4.4	Thresholds Selection	81
5.5	Discussion and Summary	85
6	Conclusion	88
	References	92
A	Python Packages Used by the Proposed Framework	104
A.1	OS	104
A.2	Tensorflow	104
A.3	Keras	105
A.4	Numpy	105
A.5	CV2 (OpenCV-Python)	105
A.6	SYS	106
A.7	Dlib	106
A.8	Sklearn	106
A.9	Random	107
A.10	Matplotlib	107
A.11	Warnings	107
A.12	Attention_model	107
A.13	Time	108
A.14	Face_recognition	108
A.15	Imageio	108
A.16	Pickle	109
A.17	Image	109
A.18	Pandas	109
A.19	Torch.optim	110
B	Some Samples of the Proposed Framework Codes	111
B.1	Partial Codes of Adam Algorithm	111
B.2	Rotating Training Data for Data Augmentation	114
B.3	Partial Codes of Datasets Partitioning	118
B.4	Partial Codes of Thresholds Selection	120

List of Figures

1.1	Encoder-Decoder self-encoding and decoding process	2
2.1	Utilising encoder and decoder for creating a deepfake model [26]	11
2.2	FacesWap-GAN objective function (adding adversarial loss and perceptual loss, which GAN can be used to change faces) [28]	13
2.3	A schematic illustration of the fast face-swap using convolutional neural networks [29]	14
2.4	Overview of the method for face segmentation, face swapping, and face perception [30]	14
2.5	Overview of the FSGAN approach [31]	15
2.6	AEI-Net for the first stage of FaceShifter [32]	16
2.7	Face2Face real-time facial reconstruction [33]	17
2.8	HeadOn technique [34]	17
2.9	Implementation steps of learning lip sync from audio [35]	18
2.10	Deep video portraits capacitate the source actor to completely control a target video portrait [36]	19
2.11	Meta-learning architecture involves the embedder network [37]	20
2.12	Overview of text-based editing of talking-head video [38]	21
2.13	The classification of Deepfakes detection methods	22
2.14	An illustration outlining BoW extraction [40]	23
2.15	An illustration outlining two-stream network [46]	24
2.16	Using Face-level ROC comparing two-stream network and other methods [46]	24
2.17	Input and output being spoofed probability, as well as segmentation pictures of the manipulated areas in every frame of the input. [49]	25

2.18	Summary of end-to-end trainable recurrent Deepfakes video detection system [54]	26
2.19	Summary of LRCN method [56]	27
2.20	Comparison for real video and fake video blink detection [56]	28
2.21	Entire pipeline means a two-step process [59]	28
2.22	Network architecture for Meso-4 [62]	30
2.23	Network architecture for MesoInception-4 [62]	31
2.24	Summary of Deepfakes productive pipeline [65]	32
2.25	Rapid generation of negative samples [65]	33
2.26	Sixty-eight facial landmarks [69]	35
2.27	Summary of Deepfakes work-flow (Left) and method of exposing deep fakes using inconsistent head poses (Right) [69]	35
2.28	Distribution of cosine distance between two head three-dimensional unit vectors for real and fake face pictures [69]	36
2.29	ROC curves regarding the SVM classification outcomes [69]	36
2.30	Global consistency and illumination estimation errors [70]	38
2.31	Illumination estimation and geometric estimation errors [70]	38
2.32	Geometric estimation errors [70]	39
2.33	The results of OpenFace tracking regarding five equally spaced frames (top), and measure the intensity of an action unit AU01 (eyebrow lift) on this video clip (bottom) [75]	40
2.34	2D visualisation of 190-D functions [75]	41
2.35	Track video source origin utilising the suggested solution [81]	43
2.36	Summary of generating a training sample in the face X-ray method [83]	43
2.37	Each image has its own special noise mark [83]	44
3.1	Screenshot of initial videos from VidTIMIT dataset and low (LQ) and high quality (HQ) Deepfakes videos [85]	49
3.2	FaceForensics++ is a dataset of facial forgeries [87]	50
3.3	Some sample face exchange from the dataset [90]	51
3.4	Red box: corresponding Deepfakes photographs; Green box: real photographs [92]	52

4.1	Plot for diverse values of the gamma [95]	58
4.2	Proportion of dataset for proposed framework	59
4.3	FAR, FRR and EER	61
4.4	EER of the classifier [105]	65
5.1	ACC results on TIMIT dataset under different thresholds	82
5.2	HTER results on FaceForensics++ dataset under different thresholds	83
5.3	HTER results on Celeb-DF dataset under different thresholds	84

List of Tables

2.1	Accuracy for manipulation detection across all manipulation types [59] .	29
2.2	The average detection rate of Deepfakes video [62]	32
2.3	Face2Face video average detection rate [62]	32
2.4	AUC performance of the detecting face warping artifacts and other methods on UADFV and DeepfakeTIMIT datasets [65]	33
2.5	ROC curve AUC values for the classification for the test data [70]	39
2.6	ROC curve AUC values for the classification for Deepfakes test data [70]	39
2.7	ROC curve AUC values for the classification for the MesoNet test data [70]	40
2.8	ROC curve AUC values for the classification for FaceForensics test data [70]	40
2.9	Comparison of experimental results with two-class detector [83]	45
3.1	Comparison of Deepfakes datasets	53
4.1	Confusion Matrix	63
5.1	Data augmentation approaches used in this thesis	71
5.2	Random partitioning of TIMIT dataset	72
5.3	Random partitioning of FaceForensics++ dataset	72
5.4	People based partitioning of the TIMIT dataset	73
5.5	People based partitioning of FaceForensics++ dataset	73
5.6	Experimental results based on training on TIMIT dataset	75
5.7	Experimental results based on training on FaceForensics++ dataset . . .	75
5.8	Intra-library test results on TIMIT dataset with and without data augmentation	77

5.9	Intra-library test results on FaceForensics++ dataset with and without data augmentation	77
5.10	Cross-library test results (HTER) with and without data augmentation .	78
5.11	Intra-library test results on TIMIT dataset using different dataset partitioning methods	79
5.12	Intra-library test results on FaceForensics++ dataset using different dataset partitioning methods	79
5.13	Cross-library test results (HTER) using different data partitioning methods	80
6.1	The optimum method under dataset partitioning	89
6.2	The optimum method under data augmentation	90
6.3	The optimum method under thresholds selection	91

List of Abbreviations

AI	Artificial Intelligence
ACC	Accuracy
AUC	Area Under Curve
CNN	Convolutional Neural Networks
DFD	Deep Fake Detection
EER	Equal Error Rate
FAR	False Acceptance Rate
FRR	False Rejection Rate
FWA	Face Warping Artifacts
FSGAN	Face Swap Generative Adversarial Networks
GAN	Generative Adversarial Networks
HTER	Half Total Error Rate
HQ	High Quality
LQ	Low Quality
LRCN	Long-term Recurrent Convolutional Networks
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
VA	Visual Artifacts

Chapter 1

Introduction

1.1 Overview of Deepfakes

Deepfakes is a blend of the words, and its inspiration is derived from “deep learning” and “fake”. Deepfakes is a technology that can superimpose the target person’s facial image on the source person’s video, which realise the target person doing things the source person did in the video. Nowadays, deep learning models such as auto-encoders and generative adversarial networks have been widely used in the field of computer vision to solve different problems, such as image segmentation, face recognition, and multi-view facial image synthesis. The Deepfakes algorithms are also utilised to check a person’s facial expressions and movements and synthesise facial images of another person carrying out similar facial expressions and movements [1].

Deepfakes technology was firstly proposed at the end of 2017. At that time, the Encoder-Decoder self-encoding and decoding structure was used when constructing the model to restore those faces that were twisted arbitrarily during the test phase. The entire process involved five steps that is shown in Figure 1.1.

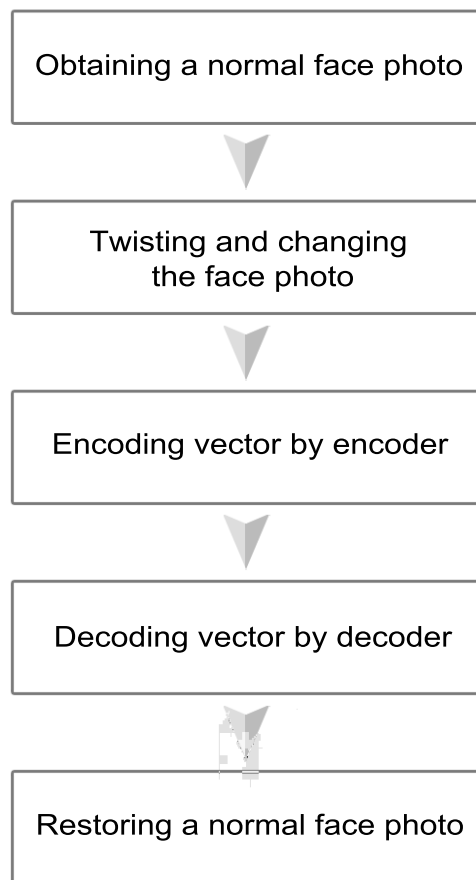


FIGURE 1.1: Encoder-Decoder self-encoding and decoding process

With the development of Deepfakes technology, GAN (Generative Adversarial Networks) technology was introduced based on the Encoder-Decoder framework. Making use of optimisation principle in game theory, the GAN algorithm not only decreases the number of model parameter and reduces model complexity under the same conditions but also makes the generated face extremely realistic, which can greatly reduce the dependence on the original photo and significantly improve the effect of changing faces, even mistaking the spurious one as the genuine one [2].

1.2 Impacts of Deepfakes

Deepfakes is arguably the fastest-growing AI technology [3] but it has become notorious because it was used to create fake videos. For instance, some people utilise this technique to fabricate pornographic contents or forge public speeches by politicians and so on. In fact, in addition to damaging the authenticity of the video, Deepfakes technology may also be used to create false evidence. For example, criminals can produce fake videos about enterprise leaders' misbehaviour in order to threaten and blackmail the company. What is worse, leading to the serious consequence that nobody believes the real videos as the continuous spread of videos generated by Deepfakes technology [4], [5]. Two law professors, Danielle Citron and Robert Chesney [6] call these effects as the scammer's bonus. Due to the fact that many people start to be aware of the harmfulness of Deepfakes, they would be sceptical of ordinary real videos, and it is easier to think them as fake videos.

However, Deepfakes also has its positive aspects, for instance, helping people who have lost their voice to create sounds, or updating movie clips without re-filming [1]. Yet, given current situation, we have to admit that the number of malicious uses of Deepfakes is still dominant to a large extent. With advanced deep neural networks (DNN) improved and a vast number of available data generated, leading humans, and even sophisticated computer algorithms have both difficulties in identifying forged images and videos [7].

Nowadays, producing forgery photographs and videos is becoming much simpler, and it only needs an identity picture or a brief video from the target person to achieve forgery. Thus, Deepfakes affects not only public celebrity but also common people [8]. For instance, in one fraud case occurred in March 2019, a group of criminals successfully imitated the voice of the CEO from the German parent firm of a British energy corporation, and cheated many colleagues and partners for 220,000 Euros scam just in one day [9]. Another case occurred in June of the same year, spies utilised AI to generate a non-existent portrait and profile, deceiving numerous contacts on LinkedIn, including government officials and political experts [10].

1.3 Overview of Deepfakes Detection

Given abused Deepfakes, some detection methods have been proposed by the researchers. Early methods were evolved around finding inconsistencies in the synthesis process. As research continued, the detection method gradually turned to extract automatically its distinction and discrimination by deep learning for detecting Deepfakes [11].

Deep forgery detection is generally considered to be a binary classification issue. Classifiers are utilised to distinguish real and tampered videos. This approach demands a massive database full of real and forgery videos for training model classification. Although the amount of forgery videos is increasing, it is limited in setting benchmarks for verifying varied detection approaches yet [12]. In order to resolve the problem, Marcel and Korshunov [13] created a Deepfakes database, including six hundred twenty videos based on a GAN model, adopting open-source Faceswap-GAN. The videos from a public VidTIMIT database is employed to produce high-quality and low-quality deep fake videos that effectively simulate blinks, mouth shapes and facial expressions [14]. In addition, UAFDV, FaceForensics, FaceForensics ++ and Celeb-DF databases are also used to test various deep fake detection.

CVPR 2020 received a paper by researchers from Microsoft Research Asia [15]. The paper proposed a new detection method, which neither needs the image data of changed faces nor masters its algorithm. The detection method could identify whether it is a composite picture just via means of doing X-Ray of images, and then it can also point out its synthetic boundary. This detection method has both recognisable and interpretive characteristics. At present, most face-changing algorithms can be separated into three components: detect, manipulate and blend. However, Face X-Ray by Microsoft Research Asia [15] is different, aiming to detect the errors in the third stage. In other words, previous studies have been detecting errors caused by changing faces, but the purpose of Face X-Ray is to detect the syncretic boundaries. The experimental outcomes show that Face X-Ray's accuracy is above ninety-five per cent, even without training around the corresponding dataset. Certainly, there is no perfect face-changing detection model. If the image is synthesised as a whole, then it is difficult to detect the

result by the Face X-Ray. In short, swapping face and face-swapping detection are like the relationships between spear and shield, and they promote and develop each other [16].

1.4 Purpose of Deepfakes Detection

Deepfakes has begun to hinder people's trust in information, because what people see is not entirely true, and the target group will be hurt by the fake content that brings various negative impacts. Deepfakes' abuse will increase the amount of false information, aggravate hatred comments, even cause political unrest, trigger public discontent, especially riots or wars [6].

Nowadays, the technology for creating Deepfakes is becoming more and more popular and can quickly spread through social media network. It is unnecessary for some Deepfakes to spread to considerable audiences, just passing them to the target audience would cause terrifying effects. For instance, intelligence agencies can harness false information created by Deepfakes to influence a politician or an official's decisions in the crucial department of a state agency, thereby endangering international and national security [6].

In the interest of addressing numerous problems caused by Deepfakes, many research efforts have been done by researchers to introduce effective detection algorithms.

1.5 Thesis Structure

Within this thesis, I look into the state of the art Deepfakes technologies with respect to creation and detection; I also put forward a framework for comparison and analysis of Deepfakes detection systems.

The rest of this thesis is arranged as follows: In Chapter 2, I will discuss several commonly used face-changing software and its merits and demerits, and will also outline two technical classifications for adopting Deepfakes to forgery face; Then, I will expound the classification based on Deepfakes detection methods and further discuss

forgery photograph and video detection algorithm, subsequently, I will study the gap existing in Deepfakes detection methods; In Chapter 3, I will demonstrate some current commonly used Deepfakes datasets and analyse them; In Chapter 4, I will describe relevant datasets used in the study and expound the methodology adopted in the research; In Chapter 5, I propose some criteria and methods and will verify them experimentally. In other words, I will analyse and compare according to these experimental results to compare to determine which method can detect Deepfakes relatively effectively under the different circumstances; In Final Chapter, I will do a conclusion regarding the research.

Chapter 2

Literature Review

2.1 Deepfakes Creation

Deepfakes is an associative AI technique, and its face and sound are processed by cloning. The video generated by using Deepfakes technique is very realistic, which even can make someone mix the spurious with the genuine. The technology itself does not have any malice, and the value of its development lies in meeting the specific needs of the public. In the first section of this chapter, I will review some software applications currently used for creating Deepfakes, whose key point is similar to other mobile/web apps, namely, these applications can be used by anyone without requiring much technical knowledge.

2.1.1 Software Applications Used for Deepfakes Creation

Currently, many deep face-changing programs are developed by Python programming language based on Tensorflow [17]. The advantages and disadvantages of several frequently-used face-changing programs are listed in the following.

2.1.1.1 Fakeapp

Fakeapp is a software of video face changer, those varied online face-changing small videos are completed by this software [18]. The features of this software are listed as below [19]:

- A powerful, diverse dataset that can easily create thousands of images from image sets and videos just in a few minutes.

- You can easily observe the progress of trained artificial intelligence by publishing frequent loss values and training previews.
- Through automatically segmenting, converting and stitching video frames, the task of converting faces in a video into a single button is reduced.

The software uses three steps to change the face, the process is listed as follows [19]:

- GET DATASET: In this step, the material video will be cut into photographs frame by frame. The program will automatically recognize and extract the facial data of the people in the picture.
- TRAIN: Training the model. According to a dataset generated during the first step, the machine will automatically help the user train the model to perform face replacement.
- CREATE: Generating video, completing the whole operation.

Advantages: it is easy to use, integrated GUI graphical interface, simple environmental installation. The users only need to download the main program and core files [20].

Disadvantages: error-prone, low efficiency, and slow updates [20].

2.1.1.2 Faceswap

FaceSwap can operate on multiple operating systems, such as Windows, Linux, and macOS. FaceSwap is an instrument that identifies and exchanges faces in photographs and videos via deep learning and the whole steps are implemented as follows [21]:

- Collect photos and/or videos
- Extract faces from the original photos
- Train a model with faces extracted from photos/videos
- Use a trained model to change own face/videos

Advantages: Github open-source software, fast update, high efficiency, seldom make mistakes, easy to debug after making errors, the latest version integrates GUI graphical interface [22].

Disadvantages: The user needs to be provided with a certain programming foundation. Firstly, install python. Secondly, use Python to compile and download various library files. Thus, the matching environment is more complicated [22].

2.1.1.3 Openfaceswap

Openfaceswap is a customized graphical interface version based on the open-source software Faceswap [18]. The software's basic face-changing steps are as follows[23]:

- Cutting the video into many pictures.
- Extracting faces from pictures.
- Training models with faces.
- Changing faces in the picture through the model.
- Integrating the pictures finishing face-changing into a video.

Advantages: Customized graphical image interface version based on Faceswap, integrating of required library files and environment, overlaying sub-folder to finish update by downloading Faceswap [23].

Disadvantages: it is difficult to solve problems after an error [23].

2.1.1.4 DeepFaceLab

DeepFaceLab as a tool which recognizes and exchanges face in photographs and videos by utilising deep learning [24]. The software has the following functions and features:

Functional characteristics [24]:

- Easy installation, almost zero environmental dependence, download and package the app and decompress it to run immediately (the biggest advantage).

- Added many new models.
- New structures for accessible to model experiments.
- Face pictures are reserved by the format of JPG, saving space and improving efficiency.
- CPU mode, the 8th generation Intel core can complete H64 model training in two days.
- Brand-new preview window for direct observation.
- Extract in parallel.
- Parallel conversion.
- DEBUG option is available in all stages.
- Supports multiple extractors such as MTCNN, DLIBCNN, S3FD.
- Support manual extraction, more accurate face area, better results.

Advantages: Customized Bat processing batch version based on Faceswap, low hardware requirements, 2G video memory can run, support manual capture of faces, integration of required materials and library files, is a powerful tool [25].

Disadvantages: complex and processing too many batches, facial data are not-general with other Deepfakes, and need to be re-screenshot [25].

2.1.2 Face-swapping

After introducing the software application that creates Deepfakes, next, I will focus on the main techniques for generating fake faces via Deepfakes, which generally divide into two categories, namely, face-swapping and face-reenactment. Face-swapping is to substitute automatically a face in the video or photograph for another face, and the identity also changes. Firstly, according to fake faces generated by Face-swapping, to discuss various Deepfakes methods.

2.1.2.1 Original Deepfakes

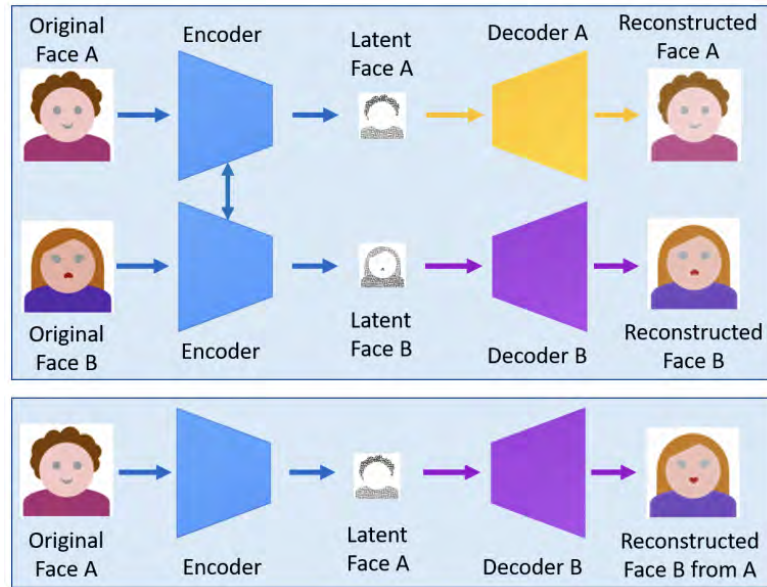


FIGURE 2.1: Utilising encoder and decoder for creating a deepfake model [26]

The face-changing video used this method had been uploaded by Reddit user "u" / Deepfakes in the post "r" / Deepfakes in 2017 [27] (the post has been blocked). This method as shown in Figure 2.1 adopts two pairs of encoder-decoders, and the two encoders share parameters during the period of training, but the decoders were trained separately. In use, input the original face A into the Encoder, and then connect to the Decoder of face B. The means of decoding can realise the replacement of B's face with A's face [26]. This method has the disadvantages of labor-intensive pattern and consuming a lot of computing resources. Thus, this method only can achieve relatively ideal results if there are a large number of target figure photographs and video materials (300 to 2000) as training data.

2.1.2.2 Faceswap-GAN

Adding perceptual and adversarial loss to a previous method based on Autoencoder can utilize GAN to achieve face replacement. The method has the following characteristics [28]:

1. The perceptual loss of VGGface: Perceptual loss improves the direction of the eyeball, making it more realistic and more consistent with the input face. It also smooths the artefact in the segmentation mask, which improves output quality.
2. Attention mask: The model predicts an Attention mask, which can help to handle shelter, eliminate human interference, and eventually create a skin tone that looks natural.
3. The resolution of input and output can be set: Supporting output range of resolution: 64×64 , 128×128 and 256×256 .
4. Face tracking/calibration using MTCNN and Kalman filter during video conversion: MTCNN is introduced for more stable detection and reliable face calibration. Kalman filter is utilized to obtain the location of the border box on each frame, eliminating facial jitter. (Kalman Filter is a classic method for motion model detection research)
5. Eye-aware training: Edge loss and reconstruction loss are introduced in an eye region to channel off the model to produce lifelike eyes.

Figure 2.2 indicates the Adversarial loss and Perceptual loss. The MAE loss above refers to the loss of the auto-encoding network of the Generator in the GAN network. In general, the lower the loss is, the better the encoder training of the self-encoding network is.

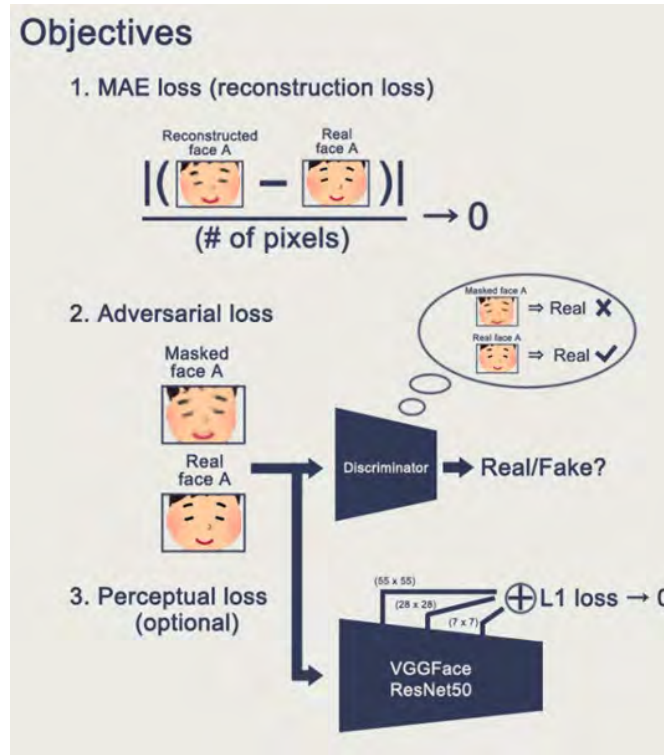


FIGURE 2.2: FacesWap-GAN objective function (adding adversarial loss and perceptual loss, which GAN can be used to change faces) [28]

2.1.2.3 Fast Face-swap Using CNN [29]

Affected by artistic style transfer, this method takes A’s face expression and posture as the content and B’s identity as the style, maintaining A’s content while generating B’s style. Input a single content image when it is training, at first, perform face alignment and face segmentation. The network transformation is an image of pyramid structure with different branch operations, and zero-padded convolution is performed in each branch. The loss function is defined in the feature space of the pre-trained VGG19 network, which principally comprises three parts - content loss, light loss, and style loss, and defines total variation regularization to ensure spatial smoothness. During the process of training, 200,000 content photographs and 120 style photographs are utilized to calculate Euclidean distance Loss of activation in the VGG-19 intermediate layer, which can achieve face replacement in almost actual time, as shown in Figure 2.3.

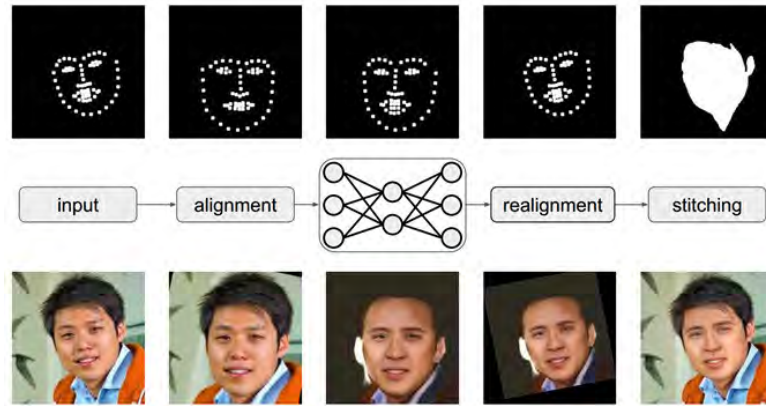


FIGURE 2.3: A schematic illustration of the fast face-swap using convolutional neural networks [29]

2.1.2.4 On Face Segmentation, Face Swapping, and Face Perception [30]

This approach based on the full convoluted neural network (FCN) face segmentation algorithm to implement face replacement. To this reason, the investigator uses 2D face landmarks to suit a 3D form. After that, split the face from the background and the shelter through pre-trained FCN and cover it to the corresponding 3D face. Finally, blend it to the target to complete the procedure of face-changing. The LFW dataset is utilized to test, and the impact of Inter-subject and Intra-subject face-changing on recognition is tested, as shown in Figure 2.4.

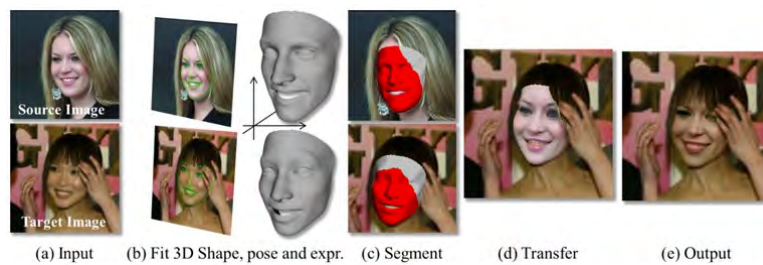


FIGURE 2.4: Overview of the method for face segmentation, face swapping, and face perception [30]

2.1.2.5 FSGAN: Subject Agnostic Face Swapping and Reenactment [31]

FSGAN is also from the previous laboratory. It is distinct from prior work, the FSGAN has unknowable characteristics, which can be implemented to pairs of faces, and do not need to train these faces. The network adopts a kind of new Poisson Blending Loss, which realises a combination of Poisson optimization and perceptual loss mutually that is shown in Figure 2.5. Finally, the investigator makes qualitative and quantitative comparisons. The outcomes indicate that the FSGAN method is better at the aspect of quality and quantity than their previous paper.

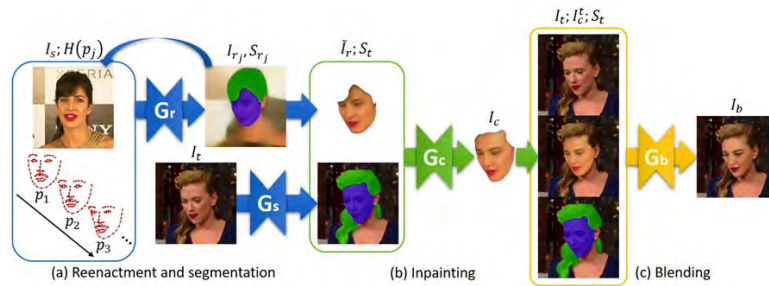


FIGURE 2.5: Overview of the FSGAN approach [31]

2.1.2.6 FaceShifter: Towards High Fidelity and Occlusion Aware Face Swapping [32]

The researchers proposed a new two-stage framework called FaceShifter [32]. This frame can complete the high-fidelity face-changing process and still work well when the face has some veils. Unlike those frameworks that utilize only limited information to complete the face change task, the first part of this framework integrates adaptively all the attributions of the target image, then generates a high-fidelity face change picture. In the interest of achieving vivid face exchange outcomes, in the first phase of the framework, the researchers design one GAN-based network for total and adaptive integration of target attributions and call it the Adaptive Embedding Integration Network (AEI-Net).

Besides that, researchers propose a new type of attribution encoder to extract multi-level attributions of face photographs. Meanwhile, they also propose a new type of

generator based on Adaptive Attentional Denormalization (AAD), which adaptively integrates characteristics and attributions of facial synthesis that is shown in Figure 2.6 [32].

To solve the problem of face shelter, the researchers added the second part - Heuristic Error Acknowledging Refinement Network (HEAR-Net) to the framework. This network repairs abnormal areas through self-supervised means without manual labeling.

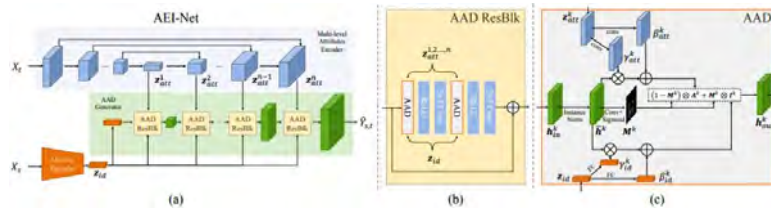


FIGURE 2.6: AEI-Net for the first stage of FaceShifter [32]

2.1.3 Face-reenactment

Face-reenactment is shifting the posture and expression from the source role to the target video, while the identity of the target subject retains unchanged. Various Deepfakes methods are discussed below based on the Face-reenactment.

2.1.3.1 Face2face: Real-time face capture and reenactment of rgb videos [33]

Face2Face implements instant face reconstruction, whose aim is to make the facial countenance of the objective video active by a root actor, and render again those processed output video in a vivid way. Firstly, the face detector detects the face in original picture and finds the key points on the face with the aid of Dlib and OpenCV, then fits the two 3D models. After that, adopting the face-targeted pix2pix conversion model to turn the key marking points over into the target face image that shows in Figure 2.7. Maybe it is because this method does not leave enough room for deep learning, so its effect is just alright, but many later paper authors use it as a comparative experiment.

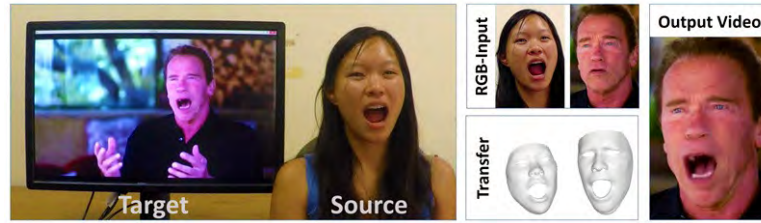


FIGURE 2.7: Face2Face real-time facial reconstruction [33]

2.1.3.2 Headon: Real-time reenactment of human portrait videos [34]

The method is from the same lab that launched Face2Face. The target actor' given short RGB-D video can construct a personalized geometric agent automatically which embeds parameterized eye, head, as well as kinematic torso models. Then, use intensive face trackers and model-to-frame to iterate the closest point (ICP), tracking source participants at the same time, finally render them into real-time videos. In other words, when tracking the primitive torso and face, investigators enabling the target person's mesh to deform. Utilising the deformed proxy of the concerned target individual's physique, investigators adopt novel view-based textures to create the lifelike output works that show in Figure 2.8.

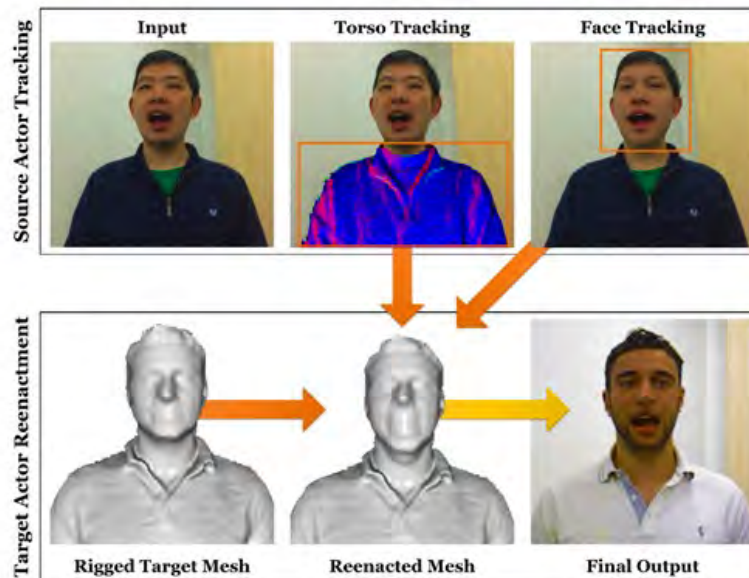


FIGURE 2.8: HeadOn technique [34]

2.1.3.3 Synthesising obama: learning lip sync from audio [35]

This paper principally studies a sequence mapping from the audio to the video. To simplify the problem, so the paper only focuses on the content of the area around the synthetic mouth, and other eyes, head, upper body, background etc. are fully retained. Given an audio sequence, the investigator first extracts feature as input to the RNN. Then, the RNN outputs a sparse mouth size corresponding to each frame of output videos. For each sparse mouth shape, the texture of the mouth and the lower part of the face is synthesized. And they are blended into the original video as output that shows in Figure 2.9.

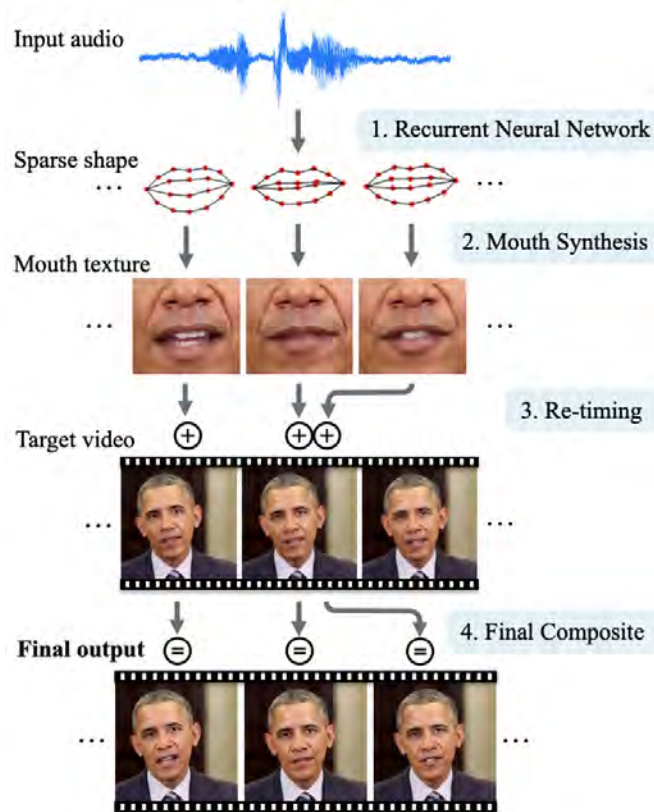


FIGURE 2.9: Implementation steps of learning lip sync from audio [35]

2.1.3.4 Deep video portraits [36]

Researchers have proposed a new method for synthesizing targets' full realistic photo video portraits in general static background. First, low-dimensional parameter representations of the two videos are obtained through monocular face rebuilding. The head poses, expressions or eye gazes can be shifted to the parameter space now (see the center in Figure 2.10). As the investigator is only interested in reproduction, so no attention is paid here to the modification of character and scene lighting (shadow background). Finally, the conditional input image is rendered on the input image and converted into a realistic video portrait of the target person (see right in Figure 2.10).

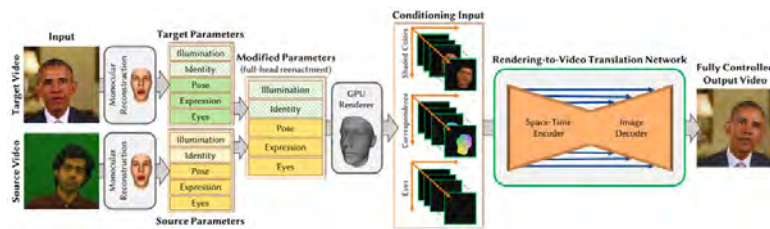


FIGURE 2.10: Deep video portraits capacitate the source actor to completely control a target video portrait [36]

2.1.3.5 Few-Shot Adversarial Learning of Realistic Neural Talking Head Models [37]

The characteristics of this method is that learning a new head model requires only a small amount of training material (8 frames of images or even a single frame of image) and a small amount of training time. The system did not demand a vast number of training cases, which needs to look at the picture only once to run.

Obtain an input source image and simulate a certain person's the motion in the target output video, thereby converting the initial image into a short video of the person talking. As can be seen from Figure 2.11, the researchers have adopted a "meta-learning" architecture. First, the embedded network maps the eye, nose, mouth size and other information in the input image and converts it into vectors. Second, the generative network draws portraits by face landmarks to copy human facial expressions in the video. Third, the discriminator network pastes the embedding vector from the input image

onto the landmark of the target video so that the input image can simulate motion in the video. Finally, the "authenticity score" is calculated. This score is used to check how well the source image matches the pose in the target video.

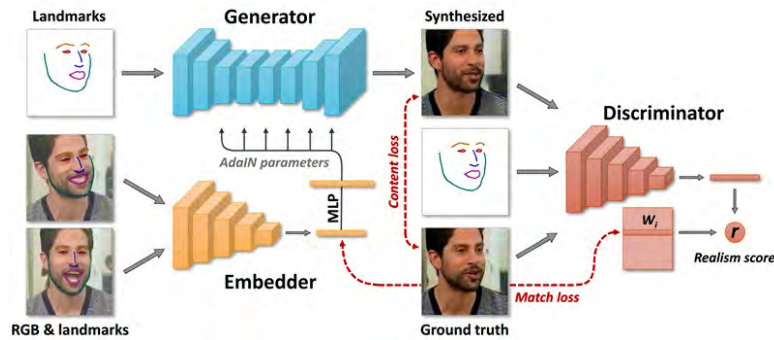


FIGURE 2.11: Meta-learning architecture involves the embedder network [37]

2.1.3.6 Text-based Editing of Talking-head Video [38]

Preset a text and an input video, the researchers implement text-based editing. Firstly, In Figure 2.12, bring the phonemes and the input audio into alignment, tracking per input frame for building the parameterized head model. After that, edit and operate the given text (such as correcting a spider into a fox), the investigator finds a snippet with a similar lip shape as the new word in the input video. Based on the above instance, researchers use viper and ox to build a fox. The researchers adopted mixed head parameters from homologous video frames and a re-timing background sequence to create a composite photograph, which creates a realistic frame utilising face-drawing methods. During the final video, the female seems to be talking about "fox", although she never said the word in the initial recording.

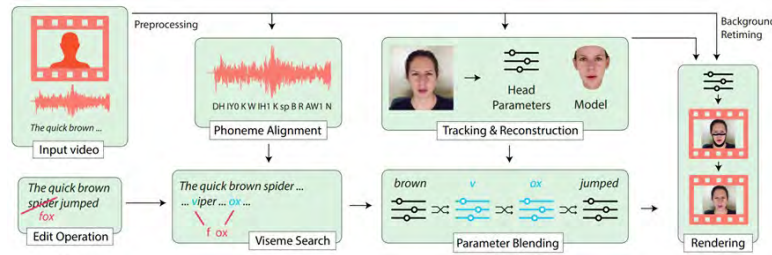


FIGURE 2.12: Overview of text-based editing of talking-head video [38]

2.1.4 Summary

This section first reviewed some current popular Deepfakes software applications and analysed their characteristics. Subsequently, the two types of fake faces generated by Face-swapping and Face-reenactment were discussed separately. In the next section, I discuss Deepfakes detection methods.

2.2 Deepfakes Detection

The methods of Deepfakes Detection are divided into two categories: false image detection and false video detection, as shown in Figure 2.13. The reason for this classification is that most picture detection algorithms cannot be utilized directly towards video detection because of the substantial degradation caused by video compression. Furthermore, videos have timing characteristics that alters with diverse frames, so they are difficult to detect by static picture detection algorithms [39]. The following discussion is in terms of fake image detection and fake video detection algorithms sequentially.

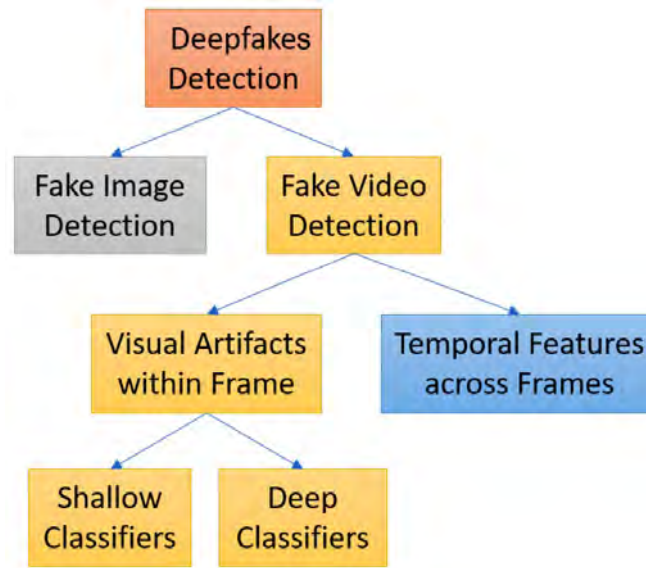


FIGURE 2.13: The classification of Deepfakes detection methods

2.2.1 Fake Image Detection

Existing image forgery fusion detection algorithms generally directly employ feature fusion or decision fusion technology. Among the images generated by deep learning, the images synthesized by the GAN model may be the most difficult to detect because they are synthesized based on GAN ability to learn about complicated input data distributions, as well as produce newly outputs with resembled input distributions. Thus, it has a good sense of reality and high quality. Due to this situation, researchers developed a variety of detection methods for fake image detection, and the section will review and discuss fake image detection in detail.

2.2.1.1 Automated face swapping and its detection [40]

The article in 2017 means the first essay for solving the problem of face-changing image detection via classical machine learning methods. The investigator used grid partitioning or SURF (speeded up robust features) [41] to extract key point descriptors, and used the K-means method to generate a bag of words features (see Figure 2.14) to obtain a code book histogram, and then use SVM [42], Random forest (RF) [43], MLP [44] and

other classifiers perform binary classification. The best experimental results reached 92 percent accuracy in their own fake face database created by the researchers based on LFW dataset [45].

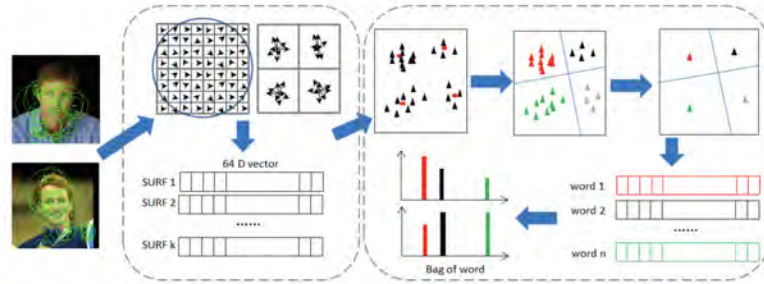


FIGURE 2.14: An illustration outlining BoW extraction [40]

2.2.1.2 Two-Stream Neural Networks for Tampered Face Detection [46]

For the sake of avoiding concentration on particular tampering evidence, as well as realise strong tamper detection, researchers have proposed the dual-stream network structure, which catches tampering evidence and local noise residual proof that shows in Figure 2.15. One of the streams is the CNN-based face classifications stream. The other stream is a triplet stream based on perdue characteristics. First face classifications stream based on the GoogleNet [47] train through real and tampered images, becoming a two-classifier. The second is a patch triplet stream based on a patch level with hidden characteristics, which can catch patterns like CFA [48] and Low-level camera feature such as local noise residuals. The investigator did not directly utilize the recessive analysis feature. But they trained a triplet network after extracting the recessive analysis feature so that this model could define the recessive analysis feature. Combining these two-streams could not only discover the high-level tampering proof but also obtain low-level noise residual features, which provides a good performance for face tampering detection. Researchers finally train and test by creating new datasets themselves.

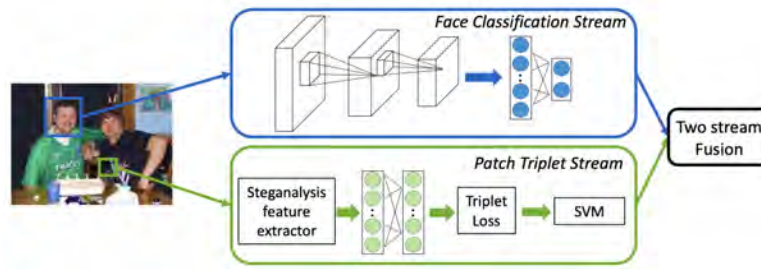


FIGURE 2.15: An illustration outlining two-stream network [46]

From the experimental results in Figure 2.16, by combining the detection score in terms of the two streams, the researchers' approach achieved more excellent performance than the single stream.

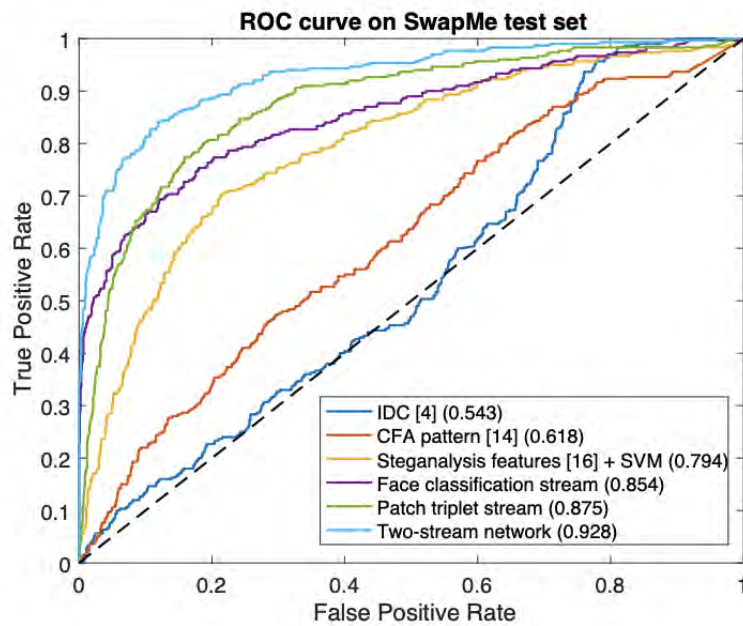


FIGURE 2.16: Using Face-level ROC comparing two-stream network and other methods [46]

2.2.1.3 Multitask Learning for Detecting and Segmenting Manipulated Facial Images and Videos [49]

Researchers have developed a multitask learning method, different from other single-target methods [50], [51], and this method is adopted to classify and segment the controlled facial picture simultaneously. In other words, the investigator offered a method outputs both the possibility of input being spoofed, as well as segmentation charts of a manipulated areas in each frame of the input, as showed Figure 2.17. Researcher’s auto-encoder contains an encoder and a Y-decoder and is trained in a semi-supervised way. The activation of coding features is for classification. That is to say, the output from a branch of decoder is utilized to segment, while the output of the other branch is utilized to reconstruct the input data. The information obtained from these tasks (classification, segmentation, and reconstruction) is shared among them, thereby improving the overall performance of the network. Experiments utilising FaceForensics [52] and FaceForensics ++ [53] dataset demonstrate the effectiveness of the network on face-swapping attacks and face exchange attacks, and its capability is to handle mismatching conditions for previously seen attacks. More significantly, adopting only a small quantity of data for a slight adjustment can enable the network to tackle invisible attacks. In addition, this method can also be utilized in the fake video detection.

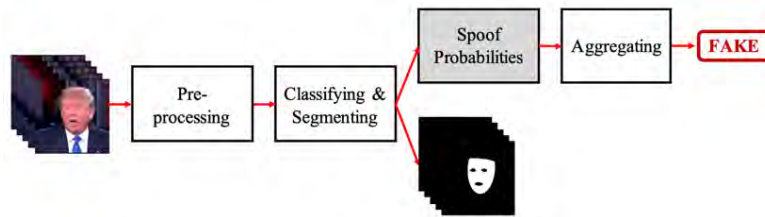


FIGURE 2.17: Input and output being spoofed probability, as well as segmentation pictures of the manipulated areas in every frame of the input. [49]

2.2.2 Fake Video Detection

At the present stage, the detection of fake video can be divided into temporal features across frames, which usually use recursive classification methods and visual artifacts

within the frame, commonly, after extracting specific features, use deep or shallow classifiers to complete the detection.

2.2.2.1 Temporal Features Across Frames

Temporal features across frames utilize time-related characteristics such as human blink frequency and mouth shape in a video to make judgments, usually through recursive classification methods. Next, I will discuss other related detection methods.

2.2.2.1.1 Deepfake Video Detection Using Recurrent Neural Networks [54]

Researchers proposed the end-to-end system. During the given video sequence, the previous CNN network uses ImageNet to train InceptionV3 model in advance [55] but it will remove the ultimate completely connected layer to create 2048-dimensional characteristic vector of per frame. The characteristic vector is input into the LSTM network. Followed by the 512-dimensional completely connected layer, the probability of authenticity and counterfeit is finally calculated using softmax, as shown in Figure 2.18.

The researchers collected 300 Deepfakes videos from the website and experimented with videos of different lengths of frame. From the outcomes, we can see that in less than two seconds of video (forty frames of video are sampled at twenty-four frames per second), this system can precisely forecast whether an analyzed segment is from a deeply faked video, which has an accuracy rate of up to 97 percent. However, it also has a disadvantage that it requires real and fake pictures as training data, thus it is inefficient.

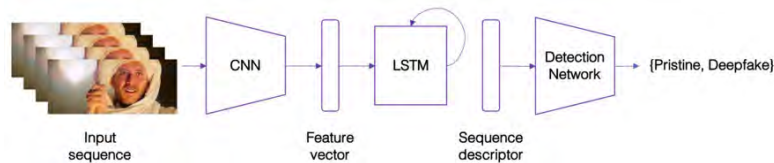


FIGURE 2.18: Summary of end-to-end trainable recurrent Deepfakes video detection system [54]

2.2.2.1.2 In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking [56]

The training samples based on the deep fakes synthetic video rarely have closed eyes, which is detected by the blink frequency of the video face detection. First, detect the face of each frame, locate the face area, and extract key points from a face, for instance, the tip of eyes, the mouth, the nose and the contours of cheeks. Next, resist interference caused by changes in head movement and face orientation in video frames, and use the landmarks-based face alignment algorithm to align a face area into the uniform coordinate space. Then, locate and extract the human eye to form a steady sequence, and send it for the LRCN network.

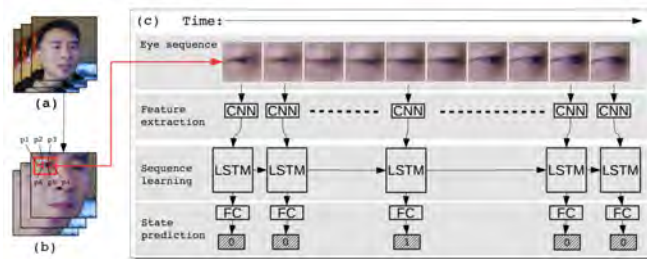


FIGURE 2.19: Summary of LRCN method [56]

LRCN network: determine the state of the eye, first, extract human eye features through VGG16 [57], input RNN in combination with LSTM unit, send the output to the completely connected layer, which figure out the state probability of open or closed eyes. Finally, use two-category classification cross-entropy loss function to train the CNN network, LSTM-RNN, and fully connected layers separately, as known in Figure 2.19.

The researchers also replace the LRCN method [58] with a two-category classification network of VGG16 and the eye aspect ratio (EAR) based technique in an experiment. As a consequence, compared with CNN 0.98 and EAR 0.79, LRCN presents the most excellent performance (0.99) based on the region under ROC (AUC). Finally, the blinking frequency of 34:1/min can be detected in the real video, but only 3.4 / min blinks in the false video, then I set a normal person’s blinking frequency threshold as 10 / min, whether this video is fake can be distinguished by us, as shown in Figure 2.20.

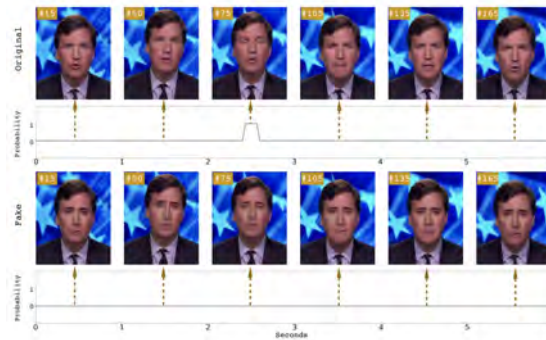


FIGURE 2.20: Comparison for real video and fake video blink detection [56]

2.2.2.1.3 Recurrent Convolutional Strategies for Face Manipulation Detection in Videos [59]

The essay draws on the fact that there are many methods processing videos through time information in the field of behavior recognition. First, perform face crop and alignment, and use ResNet [60] or DenseNet [61] as the backbone. Then, adopt one RNN network for end-to-end training. Researchers experimented with the combination of ResNet / DenseNet + one-way RNN / two-way RNN and tested on the FaceForensics ++ dataset, finding the structure of the two-way RNN + DenseNet reach state-of-the-art, as shown in Figure 2.21.

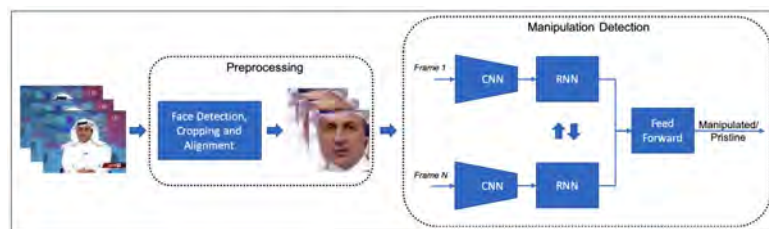


FIGURE 2.21: Entire pipeline means a two-step process [59]

Based on the FF ++ dataset, the researchers configured different CNN + RNN for experiments. The experimental results depict that DenseNet with alignment function and bidirectional recursive network performs best, as shown in Table 2.1.

Manipulation	Frames	FF++	ResNet50	DenseNet	ResNet50 +Alignment	DenseNet +Alignment	ResNet50 +Alignment +BiDir	DenseNet +Alignment +BiDir
Deepfakes	1	93.46	94.8	94.5	96.1	96.4	-	-
	5	-	94.6	94.7	96.0	96.7	94.9	96.9
Face2Face	1	89.8	90.25	90.65	89.31	87.18	-	-
	5	-	90.25	89.8	92.4	93.21	93.05	94.35
FaceSwap	1	92.72	91.34	91.04	93.85	96.1	-	-
	5	-	90.95	93.11	95.07	95.8	95.4	96.3

TABLE 2.1: Accuracy for manipulation detection across all manipulation types [59]

2.2.2.2 Visual Artifacts within Frames

In the previous section, I reviewed the methods of adopting temporal modes in cross-video frames, chiefly based on the deep recursive network models, which detect radical forgery videos. Next, I will study visual artifacts within the frame, which utilizes the defects on the edge of the image and the unnatural details such as facial features and facial shadows to judge, extract the specific features and complete the detection with deep or shallow classifiers. Then, these features are distributed into shallow or deep classifiers, which could distinguish videos between real and fake. Therefore, in this section, methods are grouped according to the type of classifier, namely, deep classifier or shallow classifier.

2.2.2.2.1 Deep Classifiers

Deepfakes videos usually have limited resolution, thus we must adopt the method of affine surface distortion (e.g. scaling, rotation and clipping). Since the distorted face areas are not consistent with the surrounding environment, which leaves fake shadow that can be detected by the CNN model. Next, let us first discuss the related detection methods based on the deep classifier.

- **Mesonet: a compact facial video forgery detection network [62]**

Indeed, micro-analysis based on image noise cannot apply to compressed video environments where picture noise is forced to reduce. Likewise, upon a higher level

of semantics, especially when a picture describes the human face, it is hard to recognize forgery images for the naked eye. Therefore, the researchers offer an intermediate method of adopting the deep neural network (DNN) with few layers.

Meso-4 network: there are four convoluted neural networks in succession, each layer is plus with Batch Normalisation [63] and Max pooling [64], and finally classify them by relying on two fully connected layers and sigmoid, as shown in Figure 2.22.

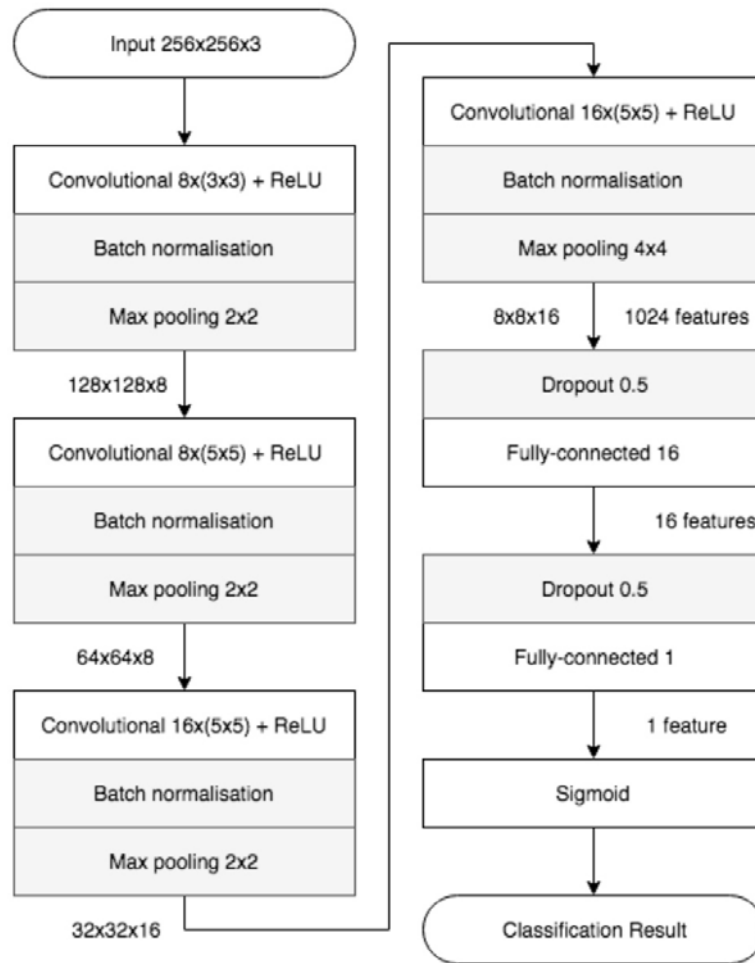


FIGURE 2.22: Network architecture for Meso-4 [62]

MesoInception-4: The two convoluted neural networks in front of Meso-4 were replaced with the variant inception-v1 module. The investigator notes that substituting more than a couple of layers with the Inception module will not bring more approving

consequences for classification. The design idea regarding this module is to superimpose outputs of two convoluted layers with diverse kernel forms, thereby enhancing the function space for model optimization, as shown in Figure 2.23.

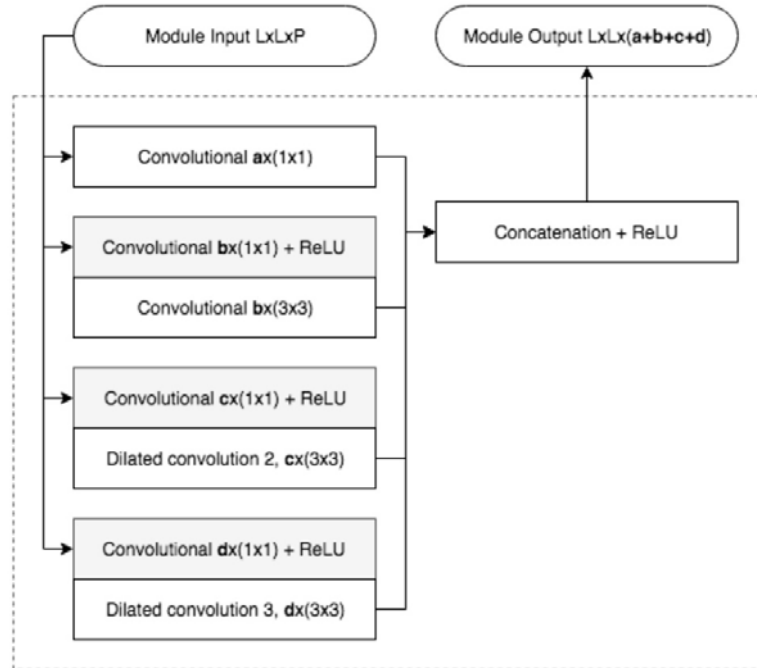


FIGURE 2.23: Network architecture for MesoInception-4 [62]

The researchers conducted experiments on the dataset about Deepfakes and Face-2Face created by themselves. In order to promote generalisation and robustness, input patches to operate several slight random transformations, consisting of scaling, rotation, horizontal flip, brightness and hue alterations.

The experimental outcomes exhibited in Table 2.2 and Table 2.3. Considering each frame respectively, the average detection rate of this method for Deepfakes video is 90 percent, and the average detection rate of Face2Face video under real diffusion conditions of the network is 95 percent.

Network	Deepfakes classification score		
Class	forged	real	total
Meso-4	0.882	0.901	0.891
MesoInception-4	0.934	0.900	0.917

TABLE 2.2: The average detection rate of Deepfakes video [62]

Network	Face2Face classification score		
Compression level	0	23 (light)	40 (strong)
Meso-4	0.946	0.924	0.832
MesoInception-4	0.968	0.934	0.813

TABLE 2.3: Face2Face video average detection rate [62]

- Exposing Deepfake Videos by Detecting Face Warping Artifacts [65]

This method is based on following properties in Deepfakes video that show in Figure 2.24 as a result of the restriction of production time and computing resources, Deepfakes algorithm can just synthesize face photographs with limited resolution, and it must be under affine transformation to match the source face configuration. Because the resolutions of the warped surface region and near surroundings are inconsistent, this distortion leaves unique fake shadows in the generated Deepfakes video, and these artifacts can be effectively captured by classical deep neural network models (such as VGG, ResNet, and so on).

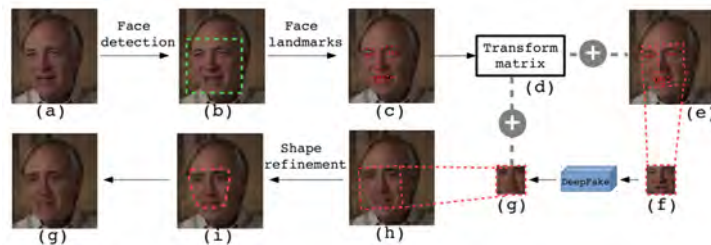


FIGURE 2.24: Summary of Deepfakes productive pipeline [65]

Figures: When obtaining training data about negative samples, considering the time and resource consumption in terms of the Deepfakes algorithm. Another aspect, since

the aim of this essay is to detect artifacts introduced by affine face warping step in the Deepfakes pipeline, the researcher directly simulates the affine surface warping step to simplify the negative sample generation process, as shown in Figure 2.25.

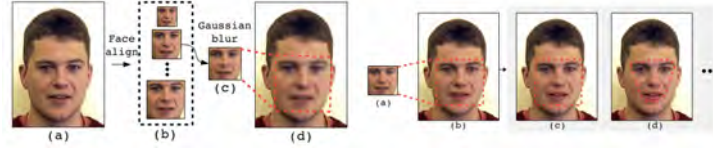


FIGURE 2.25: Rapid generation of negative samples [65]

When training, the investigator clipped the ROI regions of negative and positive samples to train the VGG16 [57], ResNet50, ResNet101 and ResNet152 [66] networks, and finally evaluated the performance of the algorithm on the UADDV [67] and Deepfakes-TIMIT datasets [68].

Methods	UADFV	DeepfakeTIMIT	
		LQ	HQ
Two-stream NN	85.1	83.5	73.5
Meso-4	84.3	87.8	68.4
MesoInception-4	82.1	80.4	62.7
HeadPose	89.0	-	-
Ours-VGG16	84.5	84.6	57.4
Ours-ResNet50	97.4	99.9	93.2
Ours-ResNet101	95.4	97.6	86.9
Ours-ResNet52	93.8	99.4	91.2

TABLE 2.4: AUC performance of the detecting face warping artifacts and other methods on UADFV and DeepfakeTIMIT datasets [65]

From these results, as shown in Table 2.4, it can be observed that classic deep learning models can effectively detect the existence of these artifacts, and these artifacts can be used to determine whether these videos are synthesized by Deepfakes algorithm.

Advantages: Creating negative samples as training data is just a straight-forward picture processing operation, hence, economizes a lot of computing resources and time.

Disadvantages: It may be easy to over-fitting the Deepfakes video with a specific distribution.

2.2.2.2.2 Shallow Classifiers

Shallow classifiers require a feature extractor that can excellently solve selectivity–invariance dilemma. One beneficial feature extractor can generate one feature, that is, can extract information that is meaningful for identifying the content of the picture, while ignoring irrelevant information, such as the pose of the animal. I discussed deep classifiers during the prior section, and the section is going to discuss related methods based on shallow classifiers.

- **Exposing Deep Fakes Using Inconsistent Head Poses [69]**

This method is based on the observation that stitching the synthesized face area into the original photograph to create Deepfakes. It will introduce errors when estimating the three-dimensional head pose (such as head direction and location) from the 2D face image. Researchers conduct experiments to prove the phenomenon, as well as classify this feature with the SVM classifier [42].

Specifically, researchers compare estimated head poses by coordinate points of the whole face that show in Figure 2.26 with head poses evaluated only by the central face area, finding they would be very similar in the real face. But for fake faces, because the middle face area comes from the synthetic face, the error will be relatively large between these two faces.

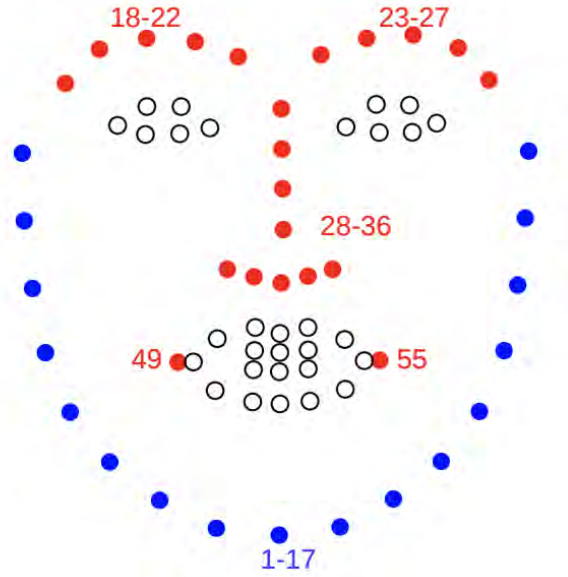


FIGURE 2.26: Sixty-eight facial landmarks [69]

In order to simplify the problem, the researchers only consider the head direction vector, and obtain two head three-dimensional unit vectors estimated from the full face and the center face respectively, and compare the cosine distances, as shown in Figure 2.27.

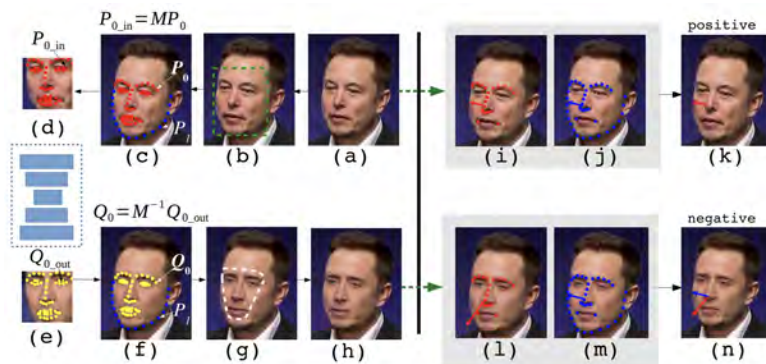


FIGURE 2.27: Summary of Deepfakes work-flow (Left) and method of exposing deep fakes using inconsistent head poses (Right) [69]

As noted in the experimental results from Figure 2.28, the cosine distance of a couple of head pose vectors estimated from real faces is centralized on a small range (up to 0.02). However, for Deepfakes's two vector cosine distance values is distributed in the range

of 0.02-0.08, so two different distributions indicate that they can be distinguished from each other in this way.

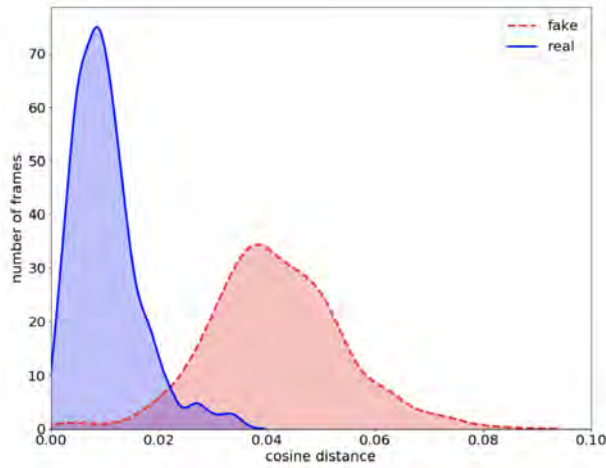


FIGURE 2.28: Distribution of cosine distance between two head three-dimensional unit vectors for real and fake face pictures [69]

The researchers further trained the SVM classifier in the UADDV dataset and the DARPA GAN Challenge dataset. The final AUROC (Area Under ROC) value is exhibited in Figure 2.29 below.

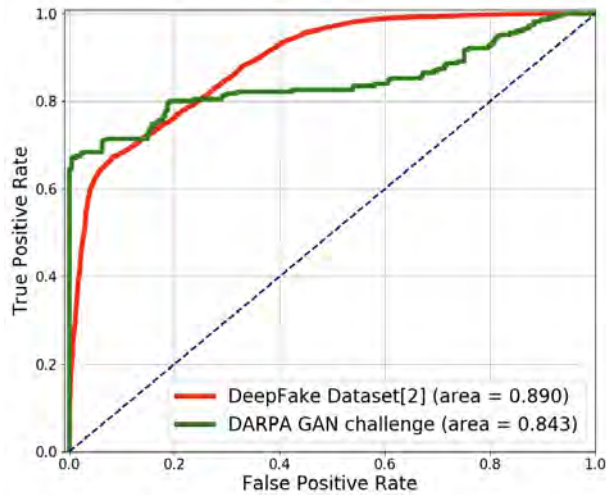


FIGURE 2.29: ROC curves regarding the SVM classification outcomes [69]

- **Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations [70]**

In view of the extant video generation methods, unique artificial visual artifacts [71] can be generated. These features can be easily detected by checking eyes, teeth, and facial contours. Researchers divide these artificial visual artifacts into the following divisions:

Global Consistency: it refers to inconsistent colour of left and right eye iris. Heterochromia is quite rare in reality, but the severity regarding this phenomenon in the generated face has great variance, as shown in Figure 2.30.

Illumination Estimation: During the method of Face2face [33], illumination estimation, geometric estimation and rendering pattern are explicitly used to model, but in the face created by deep learning, the error or inaccurate estimation of incident light is likely to cause related artifacts. This artefact regularly appears in the region around the nose, for instance, rendering extremely dark on one side. Likewise, the specular reflection in the eyes is either lost or simplified white spots in the face created by deep learning, as shown in Figure 2.30-2.31.

Geometry Estimation: the human face's inaccurate geometric estimation causes distinct boundaries, also high-contrast artificial artifacts to appear on boundaries of the human face and mask. In addition, partly blocked facial fraction, for instance, an incorrect hairstyle is going to lead to "hollowness". Moreover, regularly, teeth are not modeled at all, which is obvious in many videos. Teeth appear as small white dots rather than individual tooth in these videos, as shown in Figure 2.31-2.32.



FIGURE 2.30: Global consistency and illumination estimation errors [70]

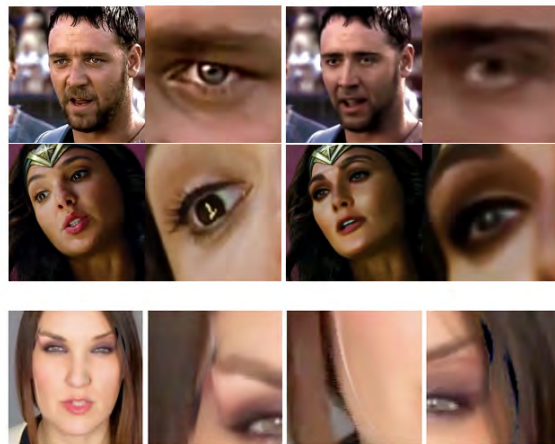


FIGURE 2.31: Illumination estimation and geometric estimation errors [70]



FIGURE 2.32: Geometric estimation errors [70]

By extracting these features to form the feature vector groups, the researchers aim at the full-face generated by GAN, Deepfakes as well as Face2face data, training respectively KNN [72], MLP [73], logistic regression model [74] and other classifiers. Because features describe specific artificial artifacts, from the results, these miniature classifiers also achieve classification task. It is also the great virtue of this technique compared to those using deep classifiers in terms of training data and time, although the effect is not rather ideal, as shown in Table 2.5-2.8.

Data	AUC (k-NN)	AUC (Colour)	P	N
ProGAN	0.852	0.814	424	580
Glow	0.843	0.752	716	580
ProGAN	0.802	0.764	1000	1000
Glow	0.796	0.704	1000	1000

TABLE 2.5: ROC curve AUC values for the classification for the test data [70]

Classifier	Eyes	Teeth	Eyes&Teeth	Crop
MLP-AUC	0.820	0.625	0.851	0.568
LogReg-AUC	0.775	0.625	0.784	0.402

TABLE 2.6: ROC curve AUC values for the classification for Deepfakes test data [70]

Classifier	Eyes	Teeth	Eyes&Teeth	Crop
MLP-AUC	0.838	0.727	0.830	0.815
LogReg-AUC	0.832	0.727	0.779	0.692

TABLE 2.7: ROC curve AUC values for the classification for the MesoNet test data [70]

Classifier	Nose	Face Border	Both	Crop
MLP-AUC	0.722	0.738	0.823	0.654
LogReg-AUC	0.710	0.843	0.866	0.770

TABLE 2.8: ROC curve AUC values for the classification for FaceForensics test data [70]

- **Protecting World Leaders Against Deep Fakes [75]**

In this method, the researchers customized this detection technology for national leaders and worldwide celebrities (POIs). Specifically speaking, the researchers believe that a person’s facial expressions and head movements show a unique pattern when they speak, it is called for soft biometric models. But the fake faces generated by Deepfakes and face-reenactment will not have this specific pattern, because their expressions are controlled.

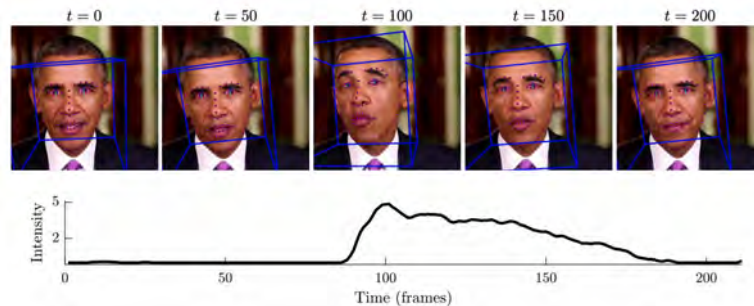


FIGURE 2.33: The results of OpenFace tracking regarding five equally spaced frames (top), and measure the intensity of an action unit AU01 (eyebrow lift) on this video clip (bottom) [75]

Method: Given a video clip, adopt the OpenFace2 [76], [77] open-source tool to track human face and head movements. The movements of facial muscles can encode into specific motion units (AU) [78]. OpenFace2 provides 17 AUs. The researcher removed

the blinking AU because the blinking itself has little effect. These AUs generate 20 feature vectors according to the rotation of the head about the x, z-axis, the 3D horizontal distance of mouth corner, as well as the 3D vertical distance of upper and lower lips, after that, utilize a Pearson correlation coefficient, which measures the similarity of the two vectors of these 20 features. Thus, it can generate $C = (20 \times 19) / (2 \times 1) = 190$ pairs of features, and each 10s video clip is compressed into a 190-dimensional feature vector, after that, adopt SVM to classify the true and false of the video, as shown in Figure 2.33.

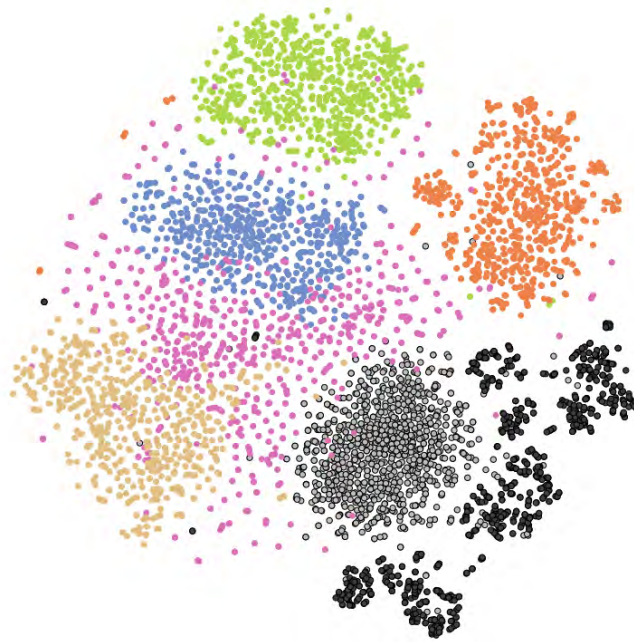


FIGURE 2.34: 2D visualisation of 190-D functions [75]

Figure 2.34 is a kind of t-SNE method [79] that converts a 190-dimensional feature vector into a 2-dimensional visualization. Different colours indicate some celebrities, or casually look for individuals, Obama who changed face, etc. In this low-dimensional representation, POIs can be distinguished from other ordinary people. Here, researchers use one-class SVM [80] to classify.

Researchers also tested on the video compression, video editing length, and speakers' context, and found that it is robust enough for video compression, but not accessible for different text contexts. Researchers suggest collecting more diverse data contexts for training. Except for this environmental effect, researchers have found that the reliability

regarding the action unit may be severely affected when POI always looks beyond the camera.

Similarly, researchers also considered that speech text content would make an effect on speech style and expression. Experiments with different texts also achieved an average AUC of 0.91, while other POI test results also reached AUCs of 0.93, 0.96, 0.92 and 0.98, which notes the method is robust enough to change text content.

- **Combating Deepfake Videos Using Blockchain and Smart Contracts [81]**

Researchers supply a block chain solution and universal framework, which helps users detect Deepfakes videos. Each video is associated with the smart contract, then it is linked upon its parent video, and every parent video has a link in its hierarchy to its children. This chain empowers the user to track and trace the source of digital content and from historical record to its original source, even if the digital content has been reproduced many times. The significant attribution concerning smart contracts is a unique hash value of Interstellar File System (IPFS) [82], whose use is to save videos as well as their meta data in the content-addressable and decentralized pattern. The pivotal functions and characteristics of smart contracts were tested to dispose of several common security challenges, for instance, the distributed refusal of service, which ensures the proposed solution to meet security requirements. The researcher's solution project concentrates on video content. Meanwhile, this solution framework is extremely general and universal, which can be applied to other forms of digital content, for instance, photographs, audios and manuscripts, as shown in Figure 2.35.

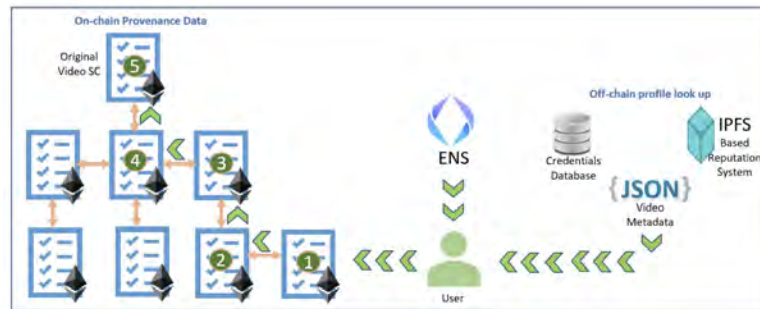


FIGURE 2.35: Track video source origin utilising the suggested solution [81]

- **Face X-ray for More General Face Forgery Detection [83]**

This technology is different from existing methods, because it can accurately detect "unknown" images, that is, no matter what kind of algorithm synthesizes those images, it can also detect without targeted training.

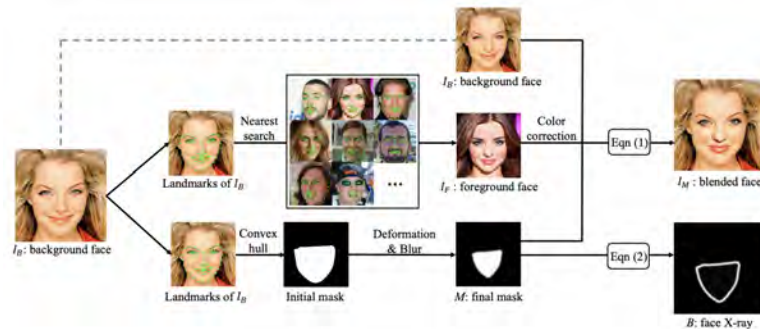


FIGURE 2.36: Summary of generating a training sample in the face X-ray method [83]

More specifically, it generates a grayscale image that shows whether a given input picture can be broken down into a mixture of two pictures from diverse sources (see Figure 2.36). After all, most methods for changing faces are to combine the generated images with existing images.

It means that Face X-Ray cannot only judge whether it is a composite picture but also point out where is composite, that is, it has both functions of recognition and interpretation.

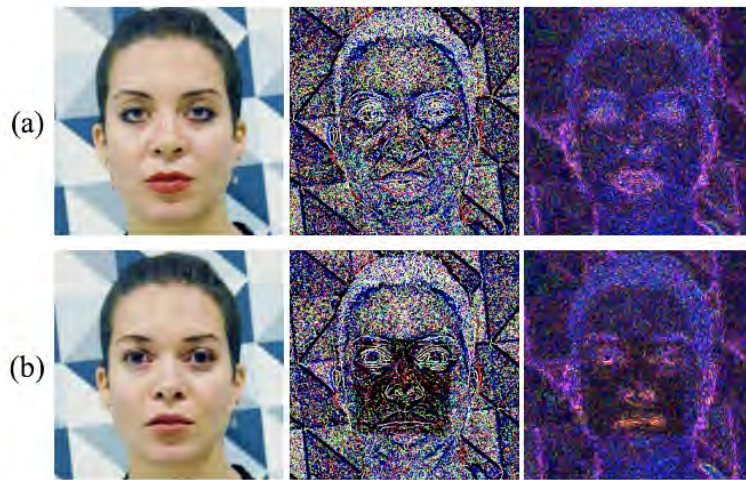


FIGURE 2.37: Each image has its own special noise mark [83]

The core ideology from the algorithm is that it can identify a unique mark of each image. There are many reasons for generating these marks, which may come from software factors like algorithms, or hardware factors like sensors, as shown in Figure 2.37.

Compared with the existing two-category face-changing detection, Face X-Ray can more effectively identify the face-changing images that have not been found, and can reliably predict the mixed regions, as shown in Table 2.9).

Model	Training set		Test set AUC				
	DF	BI	DF	F2F	FS	NT	FF++
Xception	v	-	99.38	75.05	49.13	80.39	76.34
HRNet	v	-	99.26	68.25	39.15	71.39	69.51
Face X-ray	v	-	99.17	94.14	75.34	93.85	90.62
	v	v	99.12	97.64	98.00	97.77	97.97
	F2F	BI	DF	F2F	FS	NT	FF++
Xception	v	-	87.56	99.53	65.23	65.90	79.55
HRNet	v	-	83.64	99.50	56.60	61.26	74.71
Face X-ray	v	-	98.52	99.06	72.69	91.49	93.41
	v	v	99.03	99.31	98.64	98.14	98.78
	FS	BI	DF	F2F	FS	NT	FF++
Xception	v	-	70.12	61.70	99.36	68.71	74.91
HRNet	v	-	63.59	64.12	99.24	68.89	73.96
Face X-ray	v	-	93.77	92.29	99.20	86.63	93.13
	v	v	99.10	98.16	99.09	96.66	98.25
	NT	BI	DF	F2F	FS	NT	FF++
Xception	v	-	93.09	84.82	47.98	99.50	83.42
HRNet	v	-	94.05	87.26	64.10	98.61	86.01
Face X-ray	v	-	99.14	98.43	70.56	98.93	91.76
	v	v	99.27	98.43	97.85	99.27	98.71
	FF++	BI	DF	F2F	FS	NT	FF++
Xception	-	v	98.95	97.86	89.29	97.29	95.85
HRNet	-	v	99.11	97.42	83.15	98.17	94.46
Face X-ray	-	v	99.17	98.57	98.21	98.13	98.52

TABLE 2.9: Comparison of experimental results with two-class detector [83]

Despite this, researchers also pointed out the deficiency of this method, which relies on a hybrid step, so it may not be suitable for fully synthesized images, and it may be tricked by adversarial samples.

2.2.3 Summary and Research Gap

In this chapter, I reviewed existing methods in Deepfakes creation and summarised some current available approaches for Deepfakes detection. It is worth noting that although the majority of detection methods can achieve great results, the capability from the detection technique still requires to be enhanced because test datasets as well

as test benchmarks are neither uniform.

To fight against the harm of fake-face videos, researchers have proposed a variety of different Deepfakes detection methods. However, a common problem of these detection methods is that they usually achieve higher accuracy in-library detection, but the performance seriously degraded in cross-library detections. In other words, there is a serious problem of insufficient generalisation ability.

Existing Deepfakes detections are usually dependent on the performance of the Area Under Curve (AUC) [84] as the evaluation standard. As a model evaluation index, AUC is used to evaluate the binary classification models, and can better reflect the pros and cons of the classifier when the sample ratio is imbalanced. However, AUC only exhibits the ranking ability and concentrates on the relevant magnitude of the probability value, which has nothing to do with thresholds and absolute size of the probability value. Moreover, it is not reflecting a forecast accuracy. This phenomenon depends on two characteristics of AUC. First, classification-threshold-invariant: AUC measures the quality of model predictions and has nothing to do with the classification threshold. The distribution range of the predicted value will not affect AUC. Even if the prediction value of positive examples is only a little higher than that of the negative examples, it is also a positive example to the AUC. Second, scale-invariant: AUC measures the predicted sort results, not absolute values. AUC is based on relative predictions, so any conversion that retains the relevant sort predictions will not affect AUC. Briefly, if the values regarding a model are multiplied by 2, the AUC will not change, but the offset between the model predicted value and the real value enlarged actually. Furthermore, when the data is pretty uneven, AUC may not be able to assess the performance conditions of the detection method.

In this thesis, while testing the Deepfakes detection method in the intra-library, I also conduct a cross-library test to discuss the impact of cross-library testing on detection performance. Meanwhile, I discuss the intra-library and cross-library tests and their generalisation performance. Then, I compare the influence of these factors on algorithms' performance in terms of data augmentation, dataset partitioning, and threshold selection. I further compare Deepfakes detection methods based on these criteria

as an objective framework for performance analysis of Deepfakes detection methods.

Chapter 3

Deepfakes Datasets

With the abuse of Deepfakes and the rapid development of its technology, the research of Deepfakes detection methods has become more challenging. The availability from Deepfakes large-scale dataset is an essential contributing factor to the improvement regarding Deepfakes detection methods. So far, there are UADFV dataset, Deepfakes-TIMIT (DF-TIMIT) dataset, FaceForensics++ dataset, Google Deepfakes detection (DFD) dataset, FaceBook Deepfakes detection challenge (DFDC) dataset and Celeb-DF dataset. In this chapter, I will briefly review these datasets.

3.1 UADFV

The UADFV dataset [69] is one of the earliest public databases used to detect Deepfakes. The database contains forty-nine real videos from Youtube. These videos are used to create forty-nine forgery videos via FakeApp mobile devices for target face. Then utilize Poisson image editing to mix the output of the auto-encoder with the rest of the images, and the total of dataset is 32752 frames.

3.2 TIMIT

The dataset is composed of real video and tampered video. One is the Vid-TIMIT audio-video dataset established by the University of Queensland (UQ) in Australia [85]. The other is from the Swiss Idiap Institute, it structured Deepfakes-TIMIT dataset based on the Vid-TIMIT dataset [86]. The Vid-TIMIT database contains 43 objects, each of which has screened 13 real videos. The Deepfakes-TIMIT dataset comprises 620

Deepfakes videos from the Vid-TIMIT dataset and contains 32 themes. Since the subjects in the original video mostly have front-side faces and the background is monochrome, the synthesis is relatively easy. In Deepfakes-TIMIT, there are twenty Deepfakes videos for each theme and trained two diverse models. Thereinto, one model generates ten videos with an output size of 64×64 (namely, Deepfakes-TIMIT-LQ), while the other ten videos generated by another model with an output size of 128 (namely, Deepfakes-TIMIT-HQ), as shown in Figure 3.1. These synthesized videos are created by adopting Faceswap-GAN. The dataset has a total of 10,537 original images, and 34,023 synthetic images extracted from 320 videos.

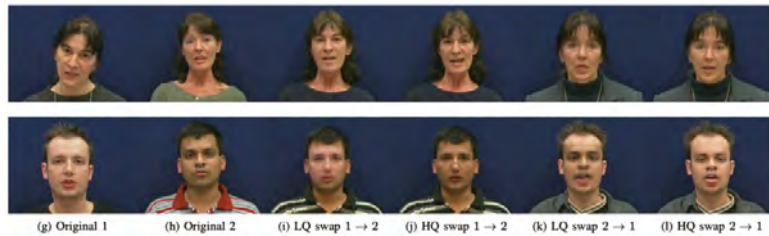


FIGURE 3.1: Screenshot of initial videos from VidTIMIT dataset and low (LQ) and high quality (HQ) Deepfakes videos [85]

3.3 FaceForensics ++

FaceForensics ++ [87] is a face-forged dataset that allows investigators to train deep learning-based methods in a supervised manner. It consists of 1,000 initial video sequences which have been manipulated with 4 automatic facial manipulation techniques: Deepfakes, Face2Face, NeuralTextures and FaceSwap. These data come from 977 YouTube videos, all of which contain traceable and mostly front-side faces without any cover, which enables the automatic tampering methods to generate realistic forgeries, as shown in Figure 3.2.



FIGURE 3.2: FaceForensics++ is a dataset of facial forgeries [87]

3.4 DFD

Google Research recorded hundreds of videos with the paid actors and the volunteer actors in 28 diverse scenarios, which are manufactured the DeepfakeDetection (DFD) dataset [88]. Then, over 3,000 of Deepfakes were generated from these videos by utilizing the openly available Deepfakes creation method. The resulting real and fake videos constitute the DFD dataset to support Deepfakes detection. Being part of the FaceForensics benchmark, the dataset is freely available to the research community and can be used to develop synthetic video detection methods.

3.5 DFDC

The Deepfake Detection Challenge (DFDC) dataset [89] was collected and established by FaceBook, which consists of 5K videos with two face modification algorithms. In order to capture various deep bionic technologies, FaceBook hires a team of paid actors. Each participant submitted a group of videos in which they completed a series of preassigned tasks. These videos involve various lighting conditions and head poses, and also take the diversity of axes (gender, skin colour, age) into account, in addition, participants can record their videos in any background that they need, which generates a visual diverse background. The population is composed of multiple races (such as Caucasians, African Americans, East Asians, South Asians, and so on) in this dataset that show Figure 3.3, whose gender ratio is about 54 percent for women and 46 percent

for men. In October 2019, the final dataset is tested on ICCV, and the complete dataset was released on NeuroIPS in December 2019.



FIGURE 3.3: Some sample face exchange from the dataset [90]

3.6 Celeb-DF

The Celeb-DF (v1) dataset [91] comprises real and Deepfakes composite videos, whose quality is similar to that of online videos. The Celeb-DF dataset involves 408 initial videos gathered from YouTube with distinct genders, ages and races, as well as 795 Deepfakes videos synthesized from these videos.

The Celeb-DF (v2) dataset [92] comprises real and Deepfakes composite videos, whose video quality also is similar to that of online dissemination videos. The Celeb-DF (v2) dataset greatly expanded than the previous Celeb-DF (v1), and the latter comprises only 795 Deepfakes videos. So far, Celeb-DF includes 590 initial videos gathered from YouTube with diverse races, genders and ages, as well as 5,639 corresponding Deepfakes videos, as shown in Figure 3.4.



FIGURE 3.4: Red box: corresponding Deepfakes photographs; Green box: real photographs [92]

3.7 Comparison and Summary of Datasets

It is observed that the Deepfakes fake face videos in the six datasets are mostly front face, complex situations of face cover are less. Thereinto, the DF-TIMIT, DFD, DFDC dataset synthetic quality is relatively high, and the FaceForensics++ dataset synthetic quality is the lowest. However, in the existing datasets, there are a small number of tempered videos whose synthetic effect is not satisfactory, for example, obvious fake facial contours and vague facial features, which mainly are caused by the inconsistency of the resolution between the falsified area and the surrounding area. Table 3.1 compares the above six Deepfakes fake face video datasets.

Chapter 3. Deepfakes Datasets

Datasets	Year	Ratio Fake : Real	Total Videos	Source	Participants Consent	Advantage	Disadvantage
UADFV [69]	2018	1:1	98	YouTube	No	Use Poisson image editing to mix the output of the auto-encoder with the rest of the image.	The number is a few, and the quality is low.
TIMIT [85], [86]	2019	1:1	1199	UQ	No	Face area accounts for a large proportion and contains two different sizes of synthetic videos generated by faceswap-GAN.	The background is relatively single, resulting in a lack of complex background features in the training samples.
FaceForensics ++ [87]	2019	1:0.25	5000	YouTube	No	Data scale is large, including the fake face videos of three different compression levels with compression rate 0, compression rate 23 and compression rate 40.	The synthesis quality is low, and there are more fake face videos with obvious tampering marks.
DeepfakeDetection [88]	2019	1:0.12	3363	Actors	Yes (28)	The synthesis quality is relatively high, considering the diversity of the scene.	The effect is too artificial, and the lighting does not match, the head pose is extreme.
DFDC [89]	2019	1:0.28	5214	Actors	Yes (28)	Quality is higher, diversity of role (gender, age, ethnicity, etc.) and visual diversity of background.	The effect is too artificial, and the lighting does not match.
Celeb-DF [91], [92]	2019	1:0.51	1203	YouTube	No	The quality of the synthesis is relatively high, which is similar to the quality of the current online video.	There are very similar videos in the training set and the test set.

TABLE 3.1: Comparison of Deepfakes datasets

This field regarding Deepfakes dataset is developing extremely promptly. With Deepfakes technology move onwards, researchers will persist in enriching the data for these existing datasets, which will continue to cooperate with their partners in this area. We firmly believe that supporting a thriving research community can assist to reduce the potential harm caused by the abuse of synthetic video technology. The release of Deepfakes dataset based on the FaceForensics benchmark symbolizes a significant step towards this goal.

Chapter 4

Methodology

In the previous chapters, the related technologies and applications of Deepfakes were reviewed and paid attention to some prevailing detection methods currently used by researchers for identification of Deepfakes videos. Meanwhile, I also outlined several mainstreamed Deepfakes datasets utilised by researchers to facilitate training and testing of detection methods. In this chapter, I will discuss the data benchmarking and the proposed method used in this research for comparison analysis.

4.1 Proposed Data Benchmark

Nowadays, six kinds of datasets are used widely. The experiment will use three of them (namely, TIMIT, FaceForensics++, and Celeb-DF) as a data benchmark. As one of the earliest public datasets, the UADFV dataset has little data and low quality. The effect of the DeepfakeDetection dataset is too artificial, the lighting does not match, and the head posture is extreme. The DFDC dataset has large data, but there are videos with high similarity in the training set and the test set. Relatively speaking, the face area of the TIMIT dataset is big relatively, and it contains two different sizes of face swap-GAN composite videos. The FaceForensics ++ dataset has a large data scale, including three different compression levels of fake face videos with a compression rate of 0, a compression rate of 23, and a compression rate of 40. The quality of the Celeb-DF dataset is high relatively, similar to the quality of current online videos. Therefore, this experiment uses TIMIT, FaceForensics++, and CelebDF datasets as the proposed data

benchmark.

To effectively train and test detection methods, this thesis carries out several experiments on diverse scenarios in the form of data and database. The reason why I choose this style is that if using repeated data or only one dataset to train, verify and test the model, it will cause the model to suffer from over fitting in the source domain data. Therefore, in this thesis, I use the TIMIT dataset and the FaceForensics++ dataset for training, verification, and testing at a rate of approximately 7:2:1. I also use the Celeb-DF dataset for cross-library testing.

FaceForensics++ and TIMIT datasets were chosen because FaceForensics++ dataset is a deep video dataset for benchmark testing of deep forgery detection algorithms. Moreover, FaceForensics++ adopted four advanced methods to create fake videos, namely, Face2Face (facial reproduction), FaceSwap (facial recognition manipulation), Deepfakes (facial identity manipulation) and NeuralTextures (using GAN for facial reproduction). The videos in the FaceForensics++ dataset not only come from different tampered methods but also the video quality is divided into three levels that are raw video, high-quality video (using h.264, parameter 23 for compression) and low-quality video (parameter 40). The TIMIT dataset has similarities to the FaceForensics ++ dataset. They both belong to a deep video dataset, and its synthesised video is also generated using Faceswap-GAN, whose video is also divided into low quality (LQ) and high quality (HQ). Based on these characteristics of the FaceForensics++ and TIMIT datasets, so they are more fit as benchmark data.

The Celeb-DF dataset has higher resolution, fixed colour consistency and background diversity. More importantly, its video quality is similar to that of current online videos, so this thesis uses the dataset for cross-library testing.

4.2 Proposed Comparison Framework

Our literature review found that the existing research literature rarely compares performance among multiple detection methods. Even when comparing different detection

methods, they are all based on the performance of AUC. This thesis will select the six Deepfake detection methods of Two-stream, MesoNet, HeadPose, FWA, VA, Multi-task, and propose a comparison framework that has not been used before to compare and analyse the detection methods. This framework designs various scenarios in the form of data and datasets for the experiments. The thesis will analyse and compare the methods based on the experimental results obtained. The analysis will be based on data augmentation, data partitioning, and thresholding selection. Data augmentation intends to expand the training samples, improving the diversity of data so that the model can adapt to a wide range of application environments. Data partitioning adopts random partitioning and person partitioning, which ensure the universality and preciseness of the experimental results. Threshold selection is an important factor for Deepfakes detection because different thresholds can directly affect the test results. In the next subsections, I introduce data augmentation, data partitioning, and thresholding selection. I will also use the accuracy rate (ACC) and error rate as the evaluation standard.

4.2.1 Data Augmentation

Data augmentation is an effective means to redouble a dataset and make the dataset as diverse as possible. Its primary meaning is to rotate data, cut data, change the colour difference, distort the characteristics, change the size, and increase the noise regarding original data. For the data augmentation of Deepfakes detection, most face recognition will require face alignment. Thus, random cropping might be meaningless here. Face alignment automatically locates key feature points of the face based on the input face image, such as eyes, nose tip, mouth corner points, eyebrows, and contour points of various parts of the face. Through face alignment, I can locate each part of the face and extract the corresponding part features. Even if part of the face is missing, it can be restored through the face alignment feature. In image classification, although random cropping of pictures during deep learning training has become a pretty usual data augmentation method, the feature of face alignment makes this data augmentation method lost its practicality. In other words, the data expanded by tailoring will not affect detection performance. But the face transformed by methods such as stretching

may be useful for Deepfakes detection. Therefore, the proposed framework adopts five methods: stretch, rotation, brightness change, channel offset, and flip horizontally to augment the data.

Our proposed framework uses a faster data augmentation method than Torchvision which is based on Pillow [93]. Our method is implemented based on OpenCV [94], which is faster than Torchvision because *opencv_transforms* in the OpenCV is intended as a faster drop-in replacement for Pytorch's Torchvision. It can make fast image augmentation for PyTorch computer vision pipelines. In OpenCV, most conversion speeds are between 1.5 times to 4 times, and the speed of large image resizing is up to 10 times. In the framework, 300×300 image blocks are used to train the cascaded network. The framework adopts two stages to data augmentation for all data in the training set and testing set. In the first stage, given an image, intercept 300×300 image blocks (namely, a center image block and four corner image blocks) from five fixed positions in the image, and cut out twenty random image blocks from other positions in the image. In other words, one image can get twenty-five 300×300 image blocks. In the second stage, for each image block obtained in the first stage, with thirty as the step size, the rotation operation is carried out. And all the image blocks are mirrored symmetrically in the horizontal direction. For each image block, using rotation and mirroring operations, another seven different image blocks can be obtained. Meanwhile, our framework uses OpenCV to perform a non-linear transformation of luminance contrast, namely, gamma correction [95]. It is different from linear transformation. The changing intensity is different and non-linear for different original brightness values, as shown in Figure 4.1. Furthermore, our framework also uses Mat [96] in OpenCV to offset the channel accordingly to make the dataset more suitable for the experimental environment.

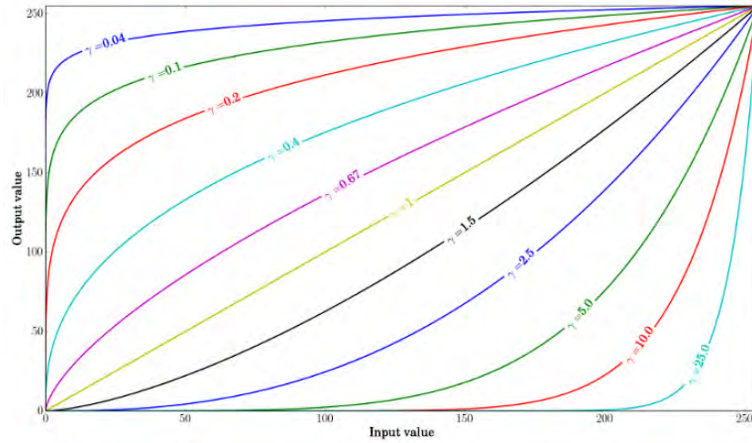


FIGURE 4.1: Plot for diverse values of the gamma [95]

The experimental results show that data augmentation does not improve the generalisation performance from the detection method. A reasonable explanation is data augmentation will change the pixel value and it is easy to destroy the shallow height frequency information. Therefore, during Deepfakes detection, it will interfere with the extraction of basic tampering features. I will further analyse and explain it in the next chapter.

4.2.2 Data Partitioning

This framework first divides the dataset into a training set and a testing set. Since the model construction process also needs to check the model, check the configuration of the model, and the degree of training for any overfitting or underfitting issues [97]. Thus, the training data will be divided into two parts. One is the training set for training, and the other is the validation set for checking the model. The validation set can be reused, mainly to help us build the model. Based on this, our framework divides the FaceForensics++ dataset and TIMIT dataset into a training set, validation set, and testing set at a ratio of 7:2:1, as shown in Figure 4.2.

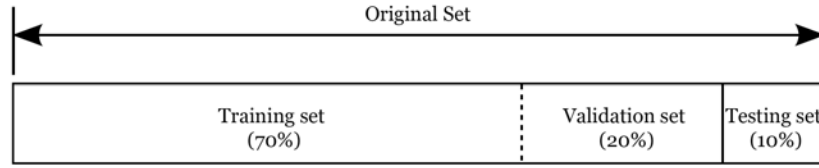


FIGURE 4.2: Proportion of dataset for proposed framework

The training set is used to train the model, and then the validation set is used to verify the effectiveness of the model and select the model that obtains the best results. After it verifies by the validation set, I will use the testing set to evaluate the accuracy and error of the model. Moreover, the testing set is only used for model checkout, and the network parameter configuration cannot be adjusted according to the results of the testing set. Otherwise, it will cause the model to overfit in the testing set. The pseudo code has been outlined Algorithm 1.

Algorithm 1: The training set and validation sets are used during training

```

for each epoch do
  for each training data instance do
    propagate error through the network;
    adjust the weights;
    calculate the accuracy over training data;
  end
  for each validation data instance do
    calculate the accuracy over the validation data;
  end
end
if the threshold validation accuracy is met then
  exit training;
else
  continue training;
end

```

To eliminate the correlation between data and avoid batch processing the same combination repeatedly, which lead to the generalisation ability declines. There are two primary forms of dataset partitioning in Deepfakes datasets: one is random partitioning, that is, directly using the video as a unit and dividing it into three datasets according to a prescribed ratio. The second is to divide by person, that is to say, taking into account

the person in the video based on controlling the proportion, to ensure that the video of the same person only appears in one dataset. The reason for dividing by person is although the person in the dataset has different actions and expressions, there are still many actions and expressions performed by the same person. Dividing by a person can prevent the data from being arranged according to a degree of rules, make it more in line with the real distribution of the data, thereby preventing overfitting. Therefore, dividing the dataset by person helps to verify further the generalisation performance of the detection method. The Celeb-DF dataset has higher resolution, and its video quality is similar to the video quality of current online streaming services today, so our framework uses Celeb-DF dataset for cross-library testing.

The experiments show different partitioning methods will have some influences on both Intra-library and cross-library testing and the size of the dataset also has a degree of impact on the test results. In the next chapter, I will analyse and compare them according to the experimental results.

4.2.3 Thresholding Selection

The threshold is also called critical value [98], which refers to the lowest or highest value that an effect can produce. In the binary classification, different thresholds directly affect the classification results of the test, resulting in diverse degrees of false alarms and missed detections. Therefore, in intra-library and cross-library tests, the framework first uses the validation set of the same dataset as the training set, that is, the data with small domain offset determines the threshold under the EER criterion. The reason for choosing the threshold under the EER criterion is that EER (Equal Error Rate) [99] unifies the two parameters of FAR (False Acceptance Rate) [100] and FRR (False Rejection Rate) [101] into one parameter to measure the overall performance of the algorithm. FAR and FRR are two parameters of the same algorithm. Putting them in the same coordinate, as shown in Figure 4.3, FAR decreases with the rise of the threshold, and FRR increases with the increase of the threshold. Another reason is that it is difficult to know the scene of the test video in practical applications, and the threshold can only determine with the help of existing videos.

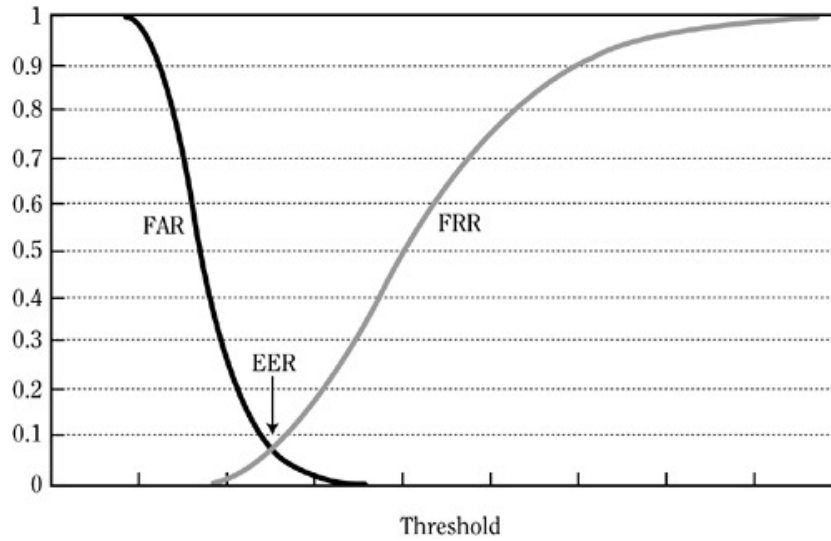


FIGURE 4.3: FAR, FRR and EER

In addition to this threshold selection method, our framework also selects other threshold selection methods for comparison. In the intra-library test, the Softmax [102] classification criterion added, threshold sets 0.5. In the cross-library test, one method is to increase the threshold of the Softmax classification criterion. The other is to use the validation set of the same dataset as the test data. That is to say, the threshold determined under the EER criterion for data of larger domain offset.

Our framework uses the training model of the TIMIT dataset as an example for testing. In this process, data augmentation is not used, and the dataset is divided by the person. The experimental results found the accuracy rate obtained by the threshold under the Softmax criterion is lower than that obtained by the threshold under the EER criterion in the intra-library test. In the cross-library test of the FaceForensics++ dataset, the HTER obtained by the threshold under the Softmax criterion is higher than the HTER gained by the threshold under the EER criterion. However, the threshold determined by the data with larger domain offset is better than the threshold determined by the data with smaller domain offset. In the cross-library test of the Celeb-DF dataset, two thresholds determined by different domains have different effects on each detection method. In the next chapter, I will evaluate the generalisation ability of the detection method by comparing the thresholds obtained from different data offsets in

the domain.

4.2.4 Accuracy Rate and Error Rate

4.2.4.1 Accuracy Rate (ACC)

Accuracy means an index for evaluating classification patterns. In other words, the accuracy rate indicates the proportion of the correct results that the pattern foretells. The definition of accuracy (ACC), namely equation 4.1 is as follows:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (4.1)$$

Take the popular binary classification matter that models are nothing more than attempting to distinguish a prediction with positive and negative. The number of correct predictions in the test set (the positive sample recognised as a positive sample, or the negative sample recognised as a negative sample) divided by the total mass of data in a test set means accuracy rate. When using scikit-learn [103] to call the classifier for classification, the score value returned by the model is the accuracy rate.

Next, I am going to introduce a confusion matrix for calculating the accuracy of a machine learning algorithm for classifying data toward corresponding tags.

In machine learning, confusion matrix [104], alias error matrix is the regular format for representing accuracy assessment and it is determined by matrix's modality with n rows and n columns. It is used for assessing performance regarding supervised learning algorithms intuitively. Confusion matrix holds the square matrix with size $(n_classes, n_classes)$, where $n_classes$ represents the number of classes. Per row of this matrix denotes an instance in an authentic class, as well as per column signifies an example in a predicted class (implementation approach adopted by Tensorflow and scikit-learn). It also means per row expresses an example in a predicted class, and per column denotes an example in an authentic class. Through the confusion matrix, it is easy to observe whether the system will confound the two classes, which is also the origin of the confusion matrix name. Per row denotes an example in the predicted class, and per column expresses an example in an authentic class. It is simple to detect

whether a system will confuse two classes via a confusion matrix, which is also the origin of the confusion matrix name.

A confusion matrix is an especial type of contingency table or cross-tabulation. It has two dimensions (true value "actual" and predicted value "predicted"), both of which have the same a collection of "classes". In the contingency table, per a combination of class and dimension means a variable. Visually, the contingency table represents the frequency distribution of multiple variables in the form of a table. It can see from Table 4.1 that the correct positive or negative is the observation result of the correct prediction that is True Positive and True Negative. False Positive and False Negative indicate false alarm.

	Predicted Class		
	Class = Yes	Class = No	
Actual Class	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

TABLE 4.1: Confusion Matrix

True Positive (TP) - These are positive values by the correct forecast that means an actual category value is yes, and the predicted category value is also yes. For instance, if the face in the picture is real, the prediction class lets you know the same thing.

True Negative (TN) - These are negative values by the correct forecast that means the value of the actual category is no, and the value of the forecasted category is also no. For example, if the face in the picture is fake, and the prediction class lets you know the same thing.

False negative and false positives, when predicted class conflicts with the actual class, these values will have appeared.

False Positive (FP) - when the predicted category is yes, and the actual category is no. For instance, if the face in the picture is fake, but the prediction class tells you that the

face in the picture is real.

False Negative (FN) - If the actual category is yes, but the predicted category is no. For example, if the face in the picture is real, and the prediction class tells you that the face in the picture is fake.

Therefore, for binary classification, the accuracy can also calculate according to the positive and negative categories, as shown below equation 4.2:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.2)$$

Thereinto:

TP = True Positive means the positive samples of model prediction are the positive;

TN = True Negative means the negative samples of model predicted are the negative;

FP = False Positive means the negative samples of model predicted are the positive;

FN = False Negative means the positive samples of model predicted are the negative;

4.2.4.2 Error Rate

Error rate and accuracy are working in two opposite directions. Error rate indicates a proportion of misclassifications by the classifier. Error rate and accuracy are indicators evaluated from both negative and positive sides, and their sum is 1 exactly. The correct rate is the higher, Error rate is the lower; Correct rate is the lower, Error rate is the higher. The computational equation 4.3 as follows:

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \quad (4.3)$$

The error rate includes two parts: one is equal error rate (EER) calculated by threshold when missed detection rate and the false alarm rate are equivalent on the validation set. The second is the half-total error rate (HTER) obtained by employing the previous threshold on a test set.

Equal Error Rate (EER):

Equal Error Rate (EER) means the biometric security system algorithm, which pre-determines thresholds regarding its false acceptance rate and false rejection rate. The common value called an equal error rate when the ratio is equal. This value expresses that a proportion of false rejections is equal to a proportion of false acceptances. The equal error rate value is lower, and the accuracy of the biometric system is higher.

Equal Error Rate (EER) is the point on the ROC curve [105] (receiver operating characteristic curve), which matches an equal probability of mis-sort with a positive sample or negative sample. The point obtained for crossing ROC curve by a diagonal line ([0,1]-[1,0] line) of the unit square. The EER of the classifier shown in Figure 4.4 should be an abscissa value corresponding to the crossing of a blue dashed line and red ROC curve, which equals about 0.17.

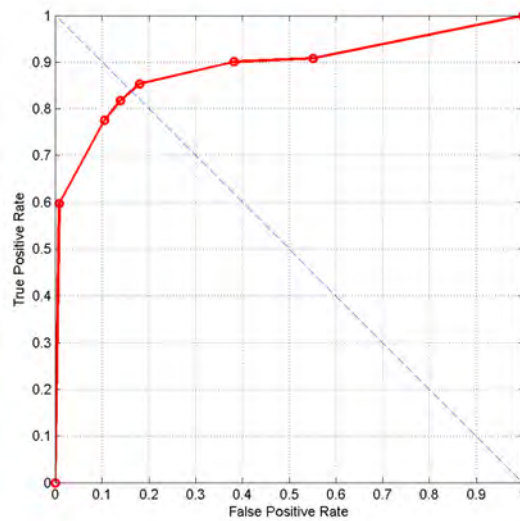


FIGURE 4.4: EER of the classifier [105]

The equation 4.4 for equal error rate is as follows:

$$EER = FPR_{val} = FNR_{val} \quad (4.4)$$

In the equation, FPR is the false alarm rate, FNR is the missed detection rate, and val represents the result on the verification set.

FPR (False Positive Rate) is the false alarm rate, which can be understood as how many of all inverse classes are predicted as positive (positive class prediction errors), equation 4.5 as follows:

$$FPR = \frac{FP}{FP + TN} \quad (4.5)$$

FNR (False Negative Rate) is the missed detection rate, which can understand as to how many of all positive classes predicted to be negative (anti-class prediction errors), equation 4.6 as follows:

$$FNR = \frac{FN}{TP + FN} \quad (4.6)$$

TP means True Positive, FP means False Positive, FN means False Negative, TN means True Negative

Half Total Error Rate (HTER)

In the detection method, a criterion that often used is HTER, the full name is half total error rate, and the calculation equation 4.7 is:

$$HTER = \frac{FPR_{test} + FNR_{test}}{2} \quad (4.7)$$

In the equation, FPR is the false alarm rate, FNR is the missed detection rate, and test represents the result on the test set.

Chapter 5

Experimental Results and Analysis

In view of comparing and analysing the performance regarding six Deepfakes detection methods, this thesis would emphasise the comparative experiments at the following three aspects of the division pattern, data augmentation and threshold selection. The experimental environment is in a Windows computer equipped with Intel (R) Core i7 9700T 2.0Ghz 8 Core 12MB Cache, GeForce GTX 1650 SUPER OC 4G and 16GB DDR4 2666 MT/s (PC4-21300). The program codes have been developed in Python. The Python packages used in the proposed framework have been listed in Appendix A and some samples of our codes have been included in Appendix B.

In machine learning algorithms, the domain where the training set is located is called the source domain, the domain where the test set is located is called the target domain, and the difference between the two domains is called the domain shift. For evaluating the performance of models under different domain shifts, the experiments are mainly carried out intra-library testing and cross-library testing. Because video scenes and recording mode are similar in the same dataset's video data, the target domain and the source domain have a smaller shift in the intra-library test, which is used to assess the learning capability of the pattern. However, video data vary notably in diverse datasets. That is to say, the target domain and source domain have a larger shift in cross-library testing, which is used to assess the generalisation performance of the model. Because the real videos in the FaceForensics++ dataset and the Celeb-DF dataset are partly the same (both from YouTube), to ensure the experimental preciseness, the model trained in the FaceForensics++ dataset is not operated cross-library testing in

the Celeb-DF dataset in the thesis.

In this thesis, as mentioned previously, accuracy (ACC) and error rate are used to evaluate the performance of the algorithms. Intra-library testing is evaluated with accuracy and error rates, and cross-library tests are only evaluated with error rates. The error rate includes two sections: One is the equal error rate (EER) calculated by threshold when missed detection rate and false alarm rate are equal on the verification set.

Given the different technologies adopted in each detection method, the detection time also would be diverse. In this thesis, while the ACC and error rate are utilised to assess the performance of the detection method, and time for detection also is used as one of the evaluation standards.

5.1 Implementation Details

For comparing the performance of the six detection methods objectively, the training is in accordance with the test environments during our experiments. For the input of the network, firstly, the convolutional neural network detector in the dlib library [106] and pre-training weights are used to frame the face area. Then, according to the nose tip point coordinates detected by the FAN-2D network [107], I move the face frame horizontally to ensure the nose on the vertical line of the face frame. To make the model learn about the information of the face and the surrounding, the face frame-centred area expanded by 1.3 times is used as the input image. In the experiment, if the gradient value of network layers is greater than 1 and multiplied repeatedly, the weight value will become pretty large, resulting in a gradient explosion. Gradient explosion will cause network instability, leading to failure to learn from training data, and even appear NaN weight values that can not be updated. To avoid this situation, I use weights regularisation [108]. I limit the size of weights through regularisation to prevent gradient explosions. In other words, regularisation is mainly to restrict overfitting by regularising network weights.

To keep consistency between the training of each model, the loss functions of the six

detection methods all adopt the cross-entropy, which is usually employed in classification problems. Adam algorithm with the first-order optimisation is used for training in the experiment, it substitutes the conventional stochastic gradient descent procedure, in which the algorithm can renew neural network weights iteratively based on training data. Unlike traditional stochastic gradient descent, the Adam algorithm designs individualistic adaptive learning rates for diverse parameters via calculating first-order moment evaluation and second-order moment evaluation of the gradient. Adam algorithm [109] gains the benefits from AdaGrad [110] and RMSProp [111] simultaneously. Adam like the RMSProp algorithm not only computes the adaptive parameter learning rate based on the mean value of first-order moments, but it also offers fully use of the mean value of second-order moments of the gradient (namely partial variance). Specifically, the algorithm calculates the exponential moving average of the gradient, and the hyperparameters β_1 and β_2 control the decline rate of these moving averages. Adam has an outstanding performance in practice, achieving excellent results quickly, so it has notable advantages over other kinds of random optimisation algorithms. Based on this, the adaptive Adam algorithm is adopted in the optimisation process of our experiments and an early termination strategy used to prevent overfitting. The sample code has been included in Appendix B. The entire training process is divided into two stages. The learning rate of the first stage is $1e-4$. If the loss of the verification set of 6 iterations does not decrease, and then it enters the second stage; the learning rate of the second stage is $1e-6$, if the loss of the verification set of 15 iterations does not reduce, the training will end. Other parameters of the optimiser are used with their default values. The pseudo code in Algorithm 2 shows how the algorithm works.

Algorithm 2: Adam

Require: Step size ϵ (by default: 0.001);
 Require: Exponential decay rate of moment estimation, ρ_1 and $\rho_2 \in [0, 1)$ (by default: 0.9 0.999)
 Require: Small constants for numerical stability δ (by default: 10^{-8})
 Require: Initial parameters θ
 Initialise first-order and second moment variables $s = 0, r = 0$
 Initialise time step $t = 0$
while not reached stopping criterion **do**
 Extract a mini-batch containing m samples $x^{(1)}, \dots, x^{(m)}$ from the training set, corresponding to the target $y^{(1)}$
 Compute gradient: $g \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
 $t \leftarrow t + 1$
 Update the biased first-order moment variable: $s \leftarrow \rho_1 s + (1 - \rho_1) g$
 Update the biased second moment variable: $r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$
 Correct the deviation of the first-order moment: $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$
 Correct the deviation of the second moment: $\hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$
 Calculation update: $\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r} + \delta}}$ (Apply operations element by element)
 Application update: $\theta \leftarrow \theta + \Delta\theta$
end

In the structure of the model, Multi-task, MesoNet, FWA, HeadPose and VA directly refer to the model code provided in the literature [49], [62], [65], [69], [70] respectively, and Two-stream is a structure reproduction, according to [46].

5.2 Limitation

Every experiment will have some limiting factors, which will have some impacts on the experimental results. One of the factors is the impact of the experimental equipment on the experiment. The level of equipment configuration determines the running speed and efficiency of the experiment. Therefore, the equipment used in this experiment will meet the configuration requirements of the experiment as far as possible. Although it is not the best configuration, it may ensure to the greatest extent that the experiment can obtain a good result. Another factor is the error caused by the system. There are differences in the system operating environment or running methods that will cause differences in experimental results. The most distinct difference is if the data is divided by random partitioning in the experiment, then the obtained results will be different

each time. An additional factor is the choice of parameters. Every experimenter has a diverse understanding of the experimental parameters. Because of this, they select different parameters within a reasonable scope in most cases. Although the differences are few in these selected parameters, they can also impact the experimental results. For example, in this experiment, the parameters selected for data augmentation. Furthermore, this experiment adopts data with a small domain offset to determine the threshold under the EER criterion. It is because it is hard to know the scenario of the test video in practical applications. Thus, the threshold can only be determined with the help of existing videos. This experiment will circumvent these limiting factors to the greatest extent through carrying out several experiments, in which relatively stable experimental results are obtained.

5.3 Scenario Settings

In order to effectively train and test the detection methods, I work on diverse scenarios in the form of data and database for comparative experiments. Data processing includes the contents whether to perform data augmentation, and database processing includes different division approaches. Furthermore, the thesis also discusses the impact of different thresholds on the results.

5.3.1 Data Processing

Data augmentation aims to expand the training samples by computer means, enhancing the diversity of data, so that the model can adapt to a wide range of application environments, and it has a wide range of applications in target recognition and target detection. Common data augmentation methods include stretching, rotating, flipping, etc. I use five data augmentation methods for testing, as shown in Table 5.1.

Type	Stretching	Rotation	Brightness Change	Channel Offset	Flip Horizontal
Parameter	[0.8, 1.2]	[-15, 15]	[0.8, 1.2]	[-30, 30]	Yes

TABLE 5.1: Data augmentation approaches used in this thesis

5.3.2 Datasets Processing

If using repeated data to train, verify and test the model, it would cause the model to produce serious overfitting on the source domain data. Therefore, this thesis divides the TIMIT dataset and the FaceForensics++ dataset into the training set, validation set and test set at a ratio of approximately 7:2:1. There are two main forms of dataset partitioning, one is random partitioning, that is, the video is directly used as the unit and then it is divided into three sets according to the prescribed proportion (the partitioned information is shown in Tables 5.2 and 5.3). The second is to divide by individuals, that is, considering the people in the video on the basis of the control ratio, which ensures that the video of the same person only appears in one dataset (the partitioned information is shown in Tables 5.4 and 5.5). The reason for adopting these two data partitioning approaches is to ensure the universality and preciseness of the experimental results. The Celeb-DF dataset has higher resolution, and its video quality is similar to the video quality of current online streaming services today, so the thesis uses Celeb-DF dataset for cross-database testing.

	Dataset	Number of real faces	Number of fake faces		
			Low quality (C40)	High quality (C23)	Total
Video	Training set	396	223	223	446
	Verification set	104	75	75	150
	Testing set	64	28	28	56
Frame	Training set	75,216	23,949	23,949	47,898
	Verification set	18,119	7,768	7,768	15,536
	Testing set	12,226	2,936	2,936	5,872

TABLE 5.2: Random partitioning of TIMIT dataset

	Dataset	Number of real faces	Number of fake faces			
			Original (C0)	Low quality (C40)	High quality (C23)	Total
Video	Training set	750	710	704	690	2104
	Verification set	300	186	208	220	614
	Testing set	150	110	94	106	310
Frame	Training set	348,414	340,678	339,862	332,083	1,012,623
	Verification set	152,859	88,088	101,638	104,075	293,801
	Testing set	12,226	50,996	44,443	50,646	146,085

TABLE 5.3: Random partitioning of FaceForensics++ dataset

	Dataset	Number of real faces	Number of fake faces		
			Low quality (C40)	High quality (C23)	Total
Video	Training set	316	190	190	380
	Verification set	146	90	90	180
	Testing set	106	70	70	140
Frame	Training set	59,073	20,267	20,267	40,534
	Verification set	26,064	9,469	9,469	18,938
	Testing set	20,008	7,517	7,517	15,034

TABLE 5.4: People based partitioning of the TIMIT dataset

	Dataset	Number of real faces	Number of fake faces			
			Original (C0)	Low quality (C40)	High quality (C23)	Total
Video	Training set	720	712	706	693	2111
	Verification set	220	210	212	208	630
	Testing set	120	108	110	104	322
Frame	Training set	369,751	365,634	362,553	355,877	1,084,064
	Verification set	111,766	105,630	106,636	104,622	316,888
	Testing set	58,603	57,751	53,719	50,789	162,259

TABLE 5.5: People based partitioning of FaceForensics++ dataset

From Tables 5.2-5.5, I can see the type, quantity, and quality of the videos used in our experiments. I will explain some of the parameters from these tables: Generally speaking, high quality means that the video image is clear and stable, and the image transmission is smooth. Low quality means that the video is vague and even the video appears trailing phenomenon. The HQ and LQ videos used in this experiment are obtained through H.264 [112] video compression technology. H.264 means a new generation of digital video compression format after MPEG4 jointly proposed by the International Organization for Standardization (ISO) and the International Telecommunication Union (ITU). The most noticeable advantage of H.264 is its high data compression ratio. Under the same video quality, the compression ratio of H.264 is more than 2 times that of MPEG-2 and 1.5 to 2 times that of MPEG-4. Besides that, video quality also depends on a parameter, which is Constant Rate Factor (CRF) [113]. The range of the CRF scale is 0–51, where 0 represents lossless, the default is 23, and 51 might be the worst quality possible. Lower values usually result in higher quality, so the reasonable range is generally between 17 to 28. The parameter CRF used for high-quality video compression in our experiments is 23, while the CRF used for low-quality video compression is 40.

5.3.3 Selection of Thresholds

For binary classification problems, different thresholds directly affect the classification results of the test, resulting in different false alarms detection. Therefore, in the intra-library and cross-library testing, the threshold is determined under the EER criterion using the validation set of the same library as the training set, that is, the data with small domain offset. This is because it is difficult to know about the scene of the test video in practical applications, and the threshold can only be determined with the help of existing videos. In addition to this threshold selection method, I also tried to compare other selection methods: for intra-library test, the Softmax classification criterion is added, that is, the threshold is 0.5; for cross-library test, on one hand, our experiments add the threshold of the Softmax classification criterion, on the other hand, I use validation sets where is in the same library with test data to determine thresholds under EER criteria.

5.4 Performance Analysis

5.4.1 Standard Condition

This thesis determines the following benchmark conditions: data augmentation is used in data processing, random partition is performed in database processing, data with smaller domain offset is used in threshold selection, and the threshold determined under the EER criterion. The experimental results obtained are shown in the Tables 5.6 and 5.7.

Test Dataset	TIMIT				FaceForensics++	Celeb-DF
	EER	HTER	ACC	TIME (h)	HTER	HTER
Two-stream	0.6	2.1	93.3	15.4	50.1	52.3
MesoNet	0.4	0.5	97.8	14.6	40.6	31.7
HeadPose	0.5	1.8	96.6	15.2	44.9	52.7
FWA	0.2	0.2	98.7	13.3	36.7	34.9
VA	0.4	0.6	93.1	12.8	45.2	51.5
Multi-task	0.1	0.1	99.7	14.5	39.6	34.6

TABLE 5.6: Experimental results based on training on TIMIT dataset

Test Dataset	FaceForensics++				TIMIT	Celeb-DF
	EER	HTER	ACC	TIME (h)	HTER	HTER
Two-stream	10.1	8.5	91.5	17.9	35.0	38.5
MesoNet	3.8	2.7	94.6	16.8	33.8	31.3
HeadPose	3.1	1.9	95.6	17.6	41.4	50.3
FWA	1.9	5.4	98.2	15.3	31.8	31.1
VA	9.2	9.6	90.8	14.6	40.9	46.9
Multi-task	2.2	1.5	98.9	17.2	39.7	33.9

TABLE 5.7: Experimental results based on training on FaceForensics++ dataset

According to the experimental results, observations and conclusions can be derived as follows:

1) There are good results in the test sets of the intra-library evaluations. The accuracy rate of the TIMIT library exceeds 93 per cent, and the FaceForensics++ library also exceeds 90 per cent. Among them, Multi-task performs best, with an accuracy rate of more than 98 per cent. The performance of Two-stream and VA is relatively inferior, indicating that the features they extract are not as good as those extracted by the Multi-task module.

2) Comparing with the test results in the intra-library evaluations, the cross-library results are not ideal, and the minimum of HTER is more than 30 per cent, indicating that it is difficult for these models to identify videos with large domain shifts. The performance of FWA among the six detection methods is relatively steady, which might be due to the fact that FWA directly simulates the affine surface warping step to simplify

the negative sample generation process and extract some of the tampering features at high frequencies.

3) Intra-library and cross-library testing, FWA shows relatively good performance. In the intra-library test, FWA has an accuracy rate of 98.7 per cent in the TIMIT dataset, and FWA accuracy rate also achieves 98.2 per cent in the FaceForensics++ dataset. In the cross-library test, FWA's HTER exceeds 30 per cent, but compared with other detection methods, it still shows a better performance, especially HeadPose that its HTER exceeds 50 per cent, which is nearly 20 per cent higher than FWA's HTER. In summary, in the intra-library test, due to the small domain offset, the six detection methods all have good learning ability and strong ability to detect correctly. In the cross-library test, due to the large domain offset, that it, target size, background complexity and resolution size of different datasets, and varied qualities of synthetic fake faces, all these factors lead enormous disparities in data distribution, causing that the model cannot make correct judgment, and the test result is not good either.

4) The time required for each detection method to run is varied. One of the reasons is that the detection method itself uses diverse technologies. However, in the process of the experiments, I found that different configurations of a device will also affect the time required for computing. Even with the same configuration, the time required for each test is affected by some certain factors. For instance, the utilisation rate of CPU and GPU, heat radiation situation, as well as whether the CPU overlocks. Therefore, the detection time cannot use as a precise evaluation criterion.

5.4.2 Data Augmentation

From the benchmark conditions, this section aims to expound the influence of different data processing methods on fake face detection. Specifically, I discuss the impact of data augmentation on the test results.

Data Augmentation Evaluating Indicator	Yes			No		
	EER	HTER	ACC	EER	HTER	ACC
Two-stream	0.6	2.1	93.3	0.6	2.0	93.6
MesoNet	0.4	0.5	97.8	0.4	0.7	97.6
HeadPose	0.5	1.8	96.6	0.5	1.9	95.9
FWA	0.2	0.2	98.7	0.2	0.4	98.5
VA	0.4	0.6	93.1	0.9	0.7	93.2
Multi-task	0.1	0.1	99.7	0.2	0.2	99.5

TABLE 5.8: Intra-library test results on TIMIT dataset with and without data augmentation

Data Augmentation Evaluating Indicator	Yes			No		
	EER	HTER	ACC	EER	HTER	ACC
Two-stream	10.1	8.5	91.5	6.4	5.6	91.9
MesoNet	3.8	2.7	94.6	4.1	3.1	93.8
HeadPose	3.1	1.9	95.6	3.3	2.3	96.2
FWA	1.9	5.4	98.2	3.6	6.9	97.7
VA	9.2	9.6	90.3	10.2	10.3	89.5
Multi-task	2.2	1.5	98.9	3.2	3.1	98.1

TABLE 5.9: Intra-library test results on FaceForensics++ dataset with and without data augmentation

The HTER results provided in Tables 5.8 to 5.10 show that data augmentation does not improve the generalisation performance of the detection in a general sense, but it leads to a lack of consistency in the average error. In the TIMIT dataset, the HTER of Multi-task is 0.1, which is the smallest value in the six detection methods. The maximum value is Two-stream, and its HTER is 2.1. In the FaceForensics++ dataset, HeadPose’s HTER value is minimum, which is 1.9, and the VA’s HTER value is maximum, which is 9.6. Specifically, for the intra-library test, the accuracy fluctuation rate of the six detection methods is within roughly 1 per cent. Taking the test result of TIMIT dataset as an example, the accuracy rates of data augmentation are respectively: Two-stream 93.3, MesoNet 97.8, HeadPose 96.6, FWA 98.7, VA 93.1, Multi-task 99.7. Accuracy rates without data augmentation are respectively, Two-stream 93.6, MesoNet 97.6, HeadPose 95.9, FWA 98.5, VA 93.2, Multi-task 99.5; for cross-library test, the range of HTER changes is mostly between -5 per cent to 5 per cent. After using data augmentation,

Train Dataset Test Dataset Data Augmentation	TIMIT				FaceForensics++	
	FaceForensics++		Celeb-DF		TIMIT	
	Yes	No	Yes	No	Yes	No
Two-stream	50.1	48.4	52.8	45.6	39.8	45.1
MesoNet	40.6	40.1	30.8	46.1	34.1	28.6
HeadPose	41.8	43.1	31.5	48.3	35.3	29.1
FWA	37.8	32.6	35.9	36.4	30.8	34.4
VA	44.6	43.1	52.7	40.9	40.3	43.4
Multi-task	39.1	38.8	35.3	43.5	40.5	45.6

TABLE 5.10: Cross-library test results (HTER) with and without data augmentation

HTER values are respectively: Two-stream 52.8, MesoNet 30.8, HeadPose 31.5, FWA 35.9, VA 52.7, Multi-task 35.3. HTER values without data augmentation are respectively: Two-stream 45.6, MesoNet 46.1, HeadPose 48.3, FWA 36.4, VA 40.9, Multi-task 43.5. Taking FWA as an example, when the model trained with the TIMIT dataset is tested with the FaceForensics++ dataset, compared to the data without augmentation, HTER is reduced by about 5 per cent, but it is increased by 0.5 per cent when testing the Celeb-DF dataset. A reasonable explanation is that data augmentation would change the pixel value and easily destroy the shallow high-frequency information. Therefore, in the fake face detection, data augmentation would interfere with the extraction of basic tampering features. In other words, the use of data augmentation in fake face detection cannot improve the generalisation performance of the model and can be considered as unnecessary.

5.4.3 Dataset Partitioning

This section discusses the impact of different dataset partitions on the fake face video detection. According to the analysis of the experimental results in the previous section, during the training and testing of this section, only results without data augmentation have been considered and provided in Tables 5.11-5.13.

Dataset Partitioning Type Evaluating Indicator	Random Partitioning			Person Partitioning		
	EER	HTER	ACC	EER	HTER	ACC
Two-stream	0.5	0.4	98.5	10.4	13.9	88.6
MesoNet	0.3	1.5	98.8	5.5	5.3	91.8
HeadPose	0.4	1.3	98.9	1.8	4.1	96.2
FWA	0.1	0.2	99.7	5.3	5.1	95.8
VA	0.6	0.3	99.3	4.3	6.1	93.6
Multi-task	0.1	0.1	99.8	1.5	2.3	97.7

TABLE 5.11: Intra-library test results on TIMIT dataset using different dataset partitioning methods

Dataset Partitioning Type Evaluating Indicator	Random Partitioning			Person Partitioning		
	EER	HTER	ACC	EER	HTER	ACC
Two-stream	10.6	10.4	89.4	8.4	9.3	90.6
MesoNet	3.6	6.7	93.5	0.9	2.3	96.4
HeadPose	3.5	7.1	93.2	1.3	3.1	97.1
FWA	3.2	2.3	97.3	5.3	5.1	95.8
VA	6.4	5.6	94.1	4.5	5.7	93.8
Multi-task	3.3	3.1	97.8	2.7	3.6	96.5

TABLE 5.12: Intra-library test results on FaceForensics++ dataset using different dataset partitioning methods

The following conclusions can be observed from these tables:

1) For the intra-library test of the TIMIT dataset, the accuracy of all detection methods has decreased to a certain extent under the database classification criteria of dividing by people. Their accuracy is respectively: Two-stream 88.6, MesoNet 91.8, HeadPose 96.2, FWA 95.8, VA 93.6, Multi-task 97.7. At the same time, the equal error rate and the average error rate have increased. Their HTER values are respectively: Two-stream 13.9, MesoNet 5.3, HeadPose 4.1, FWA 5.1, VA 6.1, Multi-task 2.3. Relatively speaking, the accuracy and HTER value of Multi-task change the least, and its performance is more advantageous than other detection methods.

2) For the intra-library test of the FaceForensics++ dataset, it is difficult to ensure the impact of the two-division methods on the performance of the detection. There is no consistency in the direction of change in accuracy, equal error rate, and average error rate. In the random partitioning, accuracy is respectively: Two-stream 89.4, MesoNet

Train Dataset Test Dataset Dataset Partitioning Type	TIMIT				FaceForensics++	
	FaceForensics++		Celeb-DF		TIMIT	
	Random	Person	Random	Person	Random	Person
Two-stream	43.8	40.6	44.8	43.8	45.3	31.5
MesoNet	40.3	37.8	45.5	40.2	28.6	28.3
HeadPose	41.1	38.6	44.6	41.3	27.8	27.2
FWA	32.8	30.4	36.3	41.1	34.4	19.4
VA	42.4	38.3	40.6	42.2	43.6	35.2
Multi-task	39.1	35.3	43.3	35.6	45.8	38.9

TABLE 5.13: Cross-library test results (HTER) using different data partitioning methods

93.5, HeadPose 93.2, FWA 97.3, VA 94.1, Multi-task 97.8. HTER values are respectively: Two-stream 10.4, MesoNet 6.7, HeadPose 7.1, FWA 2.3, VA 5.6, Multi-task 3.1. In the person partitioning, accuracy is respectively: Two-stream 90.6, MesoNet 96.4, HeadPose 97.1, FWA 95.8, VA 93.8, Multi-task 96.5. HTER values are respectively: Two-stream 9.3, MesoNet 2.3, HeadPose 3.1, FWA 5.1, VA 5.7, Multi-task 3.6. It can be seen from Table 5.12 that the change is irregular with accuracy and HTER value in the six detection methods, so effectively analysis cannot be carried out.

3) For cross-library testing, except for the Celeb-DF dataset, the average error rate of the database by the person partitioning is significantly lower than that of the random partitioning. For the Celeb-DF dataset, the average error rates are high or low in the case of two partitions.

The above results show that different partitioning methods would have some influences on intra-library and cross-library testing. Careful observation and analysis can also find that the size of the dataset has some impacts on the test results as well. For a small-scale dataset like TIMIT, random partitioning is easy for the model to learn about the characteristics of the face object itself rather than the tampering characteristics of changing face. Thus, the HTER in the intra-library test is low, but the cross-library test HTER is high. Take FWA as an example, the HTER value of the test in the intra-library is 0.2, and in the cross-library test, its HTER value is 36.3. For a large-scale dataset such as FaceForensics++, random partitioning occurs rarely, HTER in the intra-library tests is significantly lower than person partitioning. In the six detection methods, only

the FWA's HTER value (2.3) in the random partitioning was significantly lower than the FWA's HTER value (5.1) in the person partitioning. And even HTER based on the person partitioning would also be decreased in the MesoNet. For cross-library testing, which is consistent with the situation of TIMIT dataset, that is, the randomly divided HTER is higher, this is because FaceForensics++ dataset is larger in scale and diverse in data, which enhances the generalisation ability of the model.

In brief, for Deepfakes detection, due to the lack of large-scale database, the number of data and diversity is not sufficient, so random partitioning may cause the trained model to be more consistent with the features of the face object in the source domain, so it cannot be generalised well to the target domain. In other words, the dataset should be divided by person.

5.4.4 Thresholds Selection

This section discusses the impact of different threshold selections on the test results. As shown in the previous experiments, I know data augmentation may not promote the generalisation ability of the model effectively. Also, I saw in data partitioning, the generalisation advantages of random partitioning are not as good as dividing by person. Therefore, during our experiments with different threshold selection, I will not use data augmentation and I divide the dataset by person, and select the threshold under Softmax criterion and EER criterion to test. The pseudo code for this has been provided in Algorithm 3. The experimental results are shown in Figures 5.1-5.3.

Figures 5.1-5.3 indicate test results in the TIMIT dataset training model under different threshold values. Y-axis indicates evaluation measurement, X-axis indicates the threshold values and models.

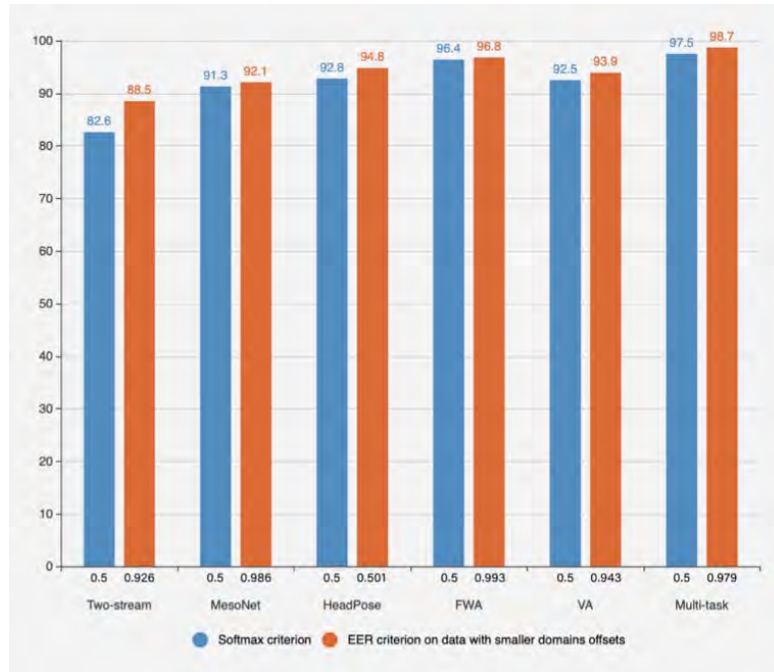


FIGURE 5.1: ACC results on TIMIT dataset under different thresholds

1) I still follow the benchmark conditions (data augmentation is used in data processing; random partitioning is performed in dataset processing; data with smaller domain offset is used in threshold selection; the threshold is determined under the EER criterion) of the experiment and use ACC as the evaluation standard in the intra-library test. In previous experiments, I have known the detection performance of data with smaller domain offsets is better than the that of data with larger domain offsets. Therefore, I used the threshold under the EER criterion with smaller domain offsets and the threshold under the Softmax criterion, which compare the performance of detection methods. In Figure 5.1, in the intra-library test, the accuracy rate obtained by the threshold value (0.5) under the Softmax criterion is lower than the accuracy rate obtained by the threshold value under the EER criterion, and Two-stream has the most decrease. Two-stream's ACC is 82.6 under Softmax criterion, while under EER criterion, Two-stream's ACC is 88.5, which declined by about 6 per cent. But for the FWA with better performance in the previous analysis, the difference between the two accuracy rates is small, ACC is 96.4 and 96.8 respectively.

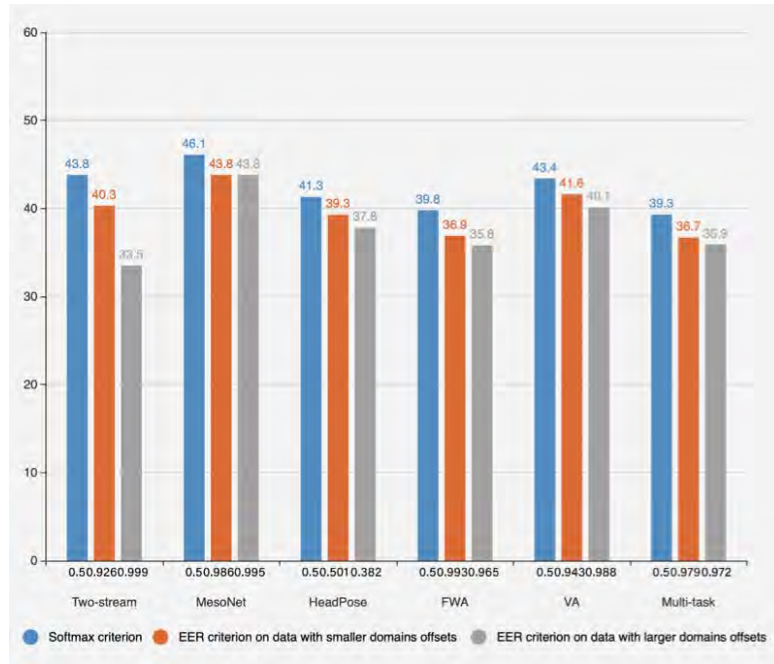


FIGURE 5.2: HTER results on FaceForensics++ dataset under different thresholds

2) In Figure 5.2, in the cross-library test of the FaceForensics++ library, the HTER obtained with the threshold (0.5) under the Softmax criterion is higher than the HTER obtained with the threshold under the EER criterion. Taking Two-stream as an example, the HTER under the Softmax criterion is 43.8, and the HTER under the EER criterion is 33.5. The difference exceeds 10 per cent between them, which is the maximum difference. The threshold determined by the data with a larger domain offset is better than the threshold determined by the data with a smaller domain offset. Except for MesoNet (both two HTER values are 43.8), and HTER of other detection methods have a certain degree of decline. HeadPose are 39.3 and 37.8, FWA are 36.9 and 35.8, VA are 41.6 and 40.1, Multi-task are 36.7 and 35.9.

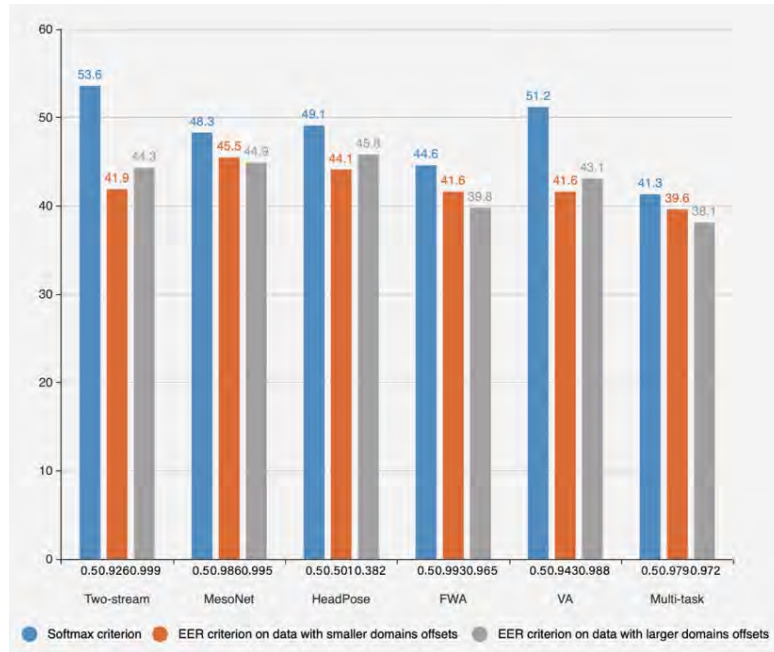


FIGURE 5.3: HTER results on Celeb-DF dataset under different thresholds

3) In Figure 5.3, in the cross-library test of the Celeb-DF library, the two thresholds determined by different domains have different effects on each detection method. For MesoNet, FWA and Multi-task, the threshold value determined by utilising larger domain offset data is better than that determined by smaller domain offset data. However, for Two-stream, HeadPose and VA, the situation is just the opposite. Besides that, the threshold calculated by HeadPose under the EER criterion is biased towards 0.5. And thresholds calculated by other detection methods under the EER criterion are all close to 1, which indicates that diverse detection methods have different detection sensitivities for detection objects with distinct synthesis qualities.

Algorithm 3: Threshold Selection

```

Choose an initial configuration
Choose an initial Threshold  $T > 0$ 
Opt: select a new configuration that is a stochastic
      perturbation of old configuration
      calculate  $\Delta E = \text{quality}(\text{new configuration}) - \text{quality}(\text{old}$ 
              configuration)
if  $\Delta E > -T$  then
  | old configuration=new configuration;
end
if a long time no improvement in quality or too numerous iterations then
  | lower Threshold  $T$ ;
end
if period of time no change in quality anymore then
  | end;
end
GOTO Opt

```

It should be noted that in practical applications, it is difficult to obtain data similar to the target domain in order to calculate the threshold under the EER. This section evaluates the generalisation ability of a model by comparing the thresholds obtained from different data offsets in the domain.

5.5 Discussion and Summary

In order to effectively test the six detection methods, our set of experiments used different data processing methods to compare them, including whether to augment the data, use diverse data division methods, and use different thresholds to find out how these affect the results. The experiment first determines a benchmark condition, then subsequent experiments are adjusted based on this condition. The experimental results obtained under the benchmark conditions show that the tests in the intra-library all have obtained good results relatively, and the accuracy of Multi-task has even reached more than 98 per cent. However, the cross-library experiment results are not pretty satisfactory. The HTER is high relatively, even the lowest HTER is more than 30 per cent, indicating that it is difficult for these six detection methods to identify videos with large domain offsets. In brief, in the intra-library and cross-library tests, FWA's ACC is 98.7

and HTER is 31.1. The performance of the FWA detection method is relatively better than the other five detection methods.

Next, starting from the benchmark conditions, the experiment uses data augmentation to test its impact on these six detection methods. In the TIMIT dataset, Multi-task's HTER is 0.1, which is the smallest value in the six detection methods. Two-stream's HTER is 2.1, which is the maximum value. However, in the FaceForensics++ dataset, HeadPose's HTER is minimum, which is 1.9, and the VA's HTER is maximum, which is 9.6. Experimental results show that data augmentation does not effectively improve the generalisation performance of the detection method, instead leads to a lack of consistency in the average error. Based on the results obtained, I consider that data augmentation would alter the pixel value, and easily destroy the high-frequency information in the shallow layer, it would interfere with the extraction of tampered features. Therefore, data augmentation is unnecessary in the Deepfakes detection method.

In our experiments, I also divided the dataset differently, namely, random partitioning and partitioning by the person, in order to compare and analyse the influence of different data partitioning on Deepfakes detection. The results I acquired is that different data partitions would have some impact on the Deepfakes detection method, no matter it is tested in the intra-library or cross-library. In the smaller dataset like TIMIT, the HTER obtained by random partitioning is low in the intra-library, while the HTER obtained by random partitioning in the cross-library test is high. Among them, FWA is particularly evident, its HTER of the test is 0.2 in the intra-library, and in the cross-library test, its HTER is 36.3. In the large-scale dataset like FaceForensics++, there is rarely a situation that the HTER obtained by random partitioning is apparently lower than dividing by person significantly in the intra-library test. In the six detection methods, only the FWA, its HTER is 2.3 in the random partitioning, and FWA's HTER is 5.1 in the person partitioning. In the cross-library test, it is the same as the case of the TIMIT library. The reason for this result is that the FaceForensics++ library is large in scale and diverse in data, which improves the generalisation ability of the model. However, considering the existing dataset, especially large-scale datasets are still relatively unavailable. The random division may result in model learning more inclined to the

characteristics of the source domain face objects, so the dataset should be divided by person.

Different thresholds were also used to test the impact of different thresholds on the six Deepfakes detection methods. In the intra-library test, the accuracy rate obtained with the threshold value (0.5) under the Softmax criterion is lower than the accuracy rate obtained under the threshold value under the EER criterion. Among them, Two-stream has the most decline. Two-stream's ACC is 82.6 under Softmax criterion, while under EER criterion, Two-stream's ACC is 88.5, which declined by about 6 per cent. FWA is better, which is 96.4 and 96.8, and the difference is minimum. In the cross-library test of the FaceForensics++ library, the HTER obtained by the threshold value (0.5) under the Softmax criterion is higher than the HTER gained by the threshold value under the EER criterion, and the maximum difference exceeds 10 per cent. Taking Two-stream as an example, the HTER is 43.8 under the Softmax criterion, and the HTER is 33.5 under the EER criterion. The difference is 10.3 between them. In contrast, the threshold value determined by the data with larger domain offset is better than the threshold value determined by the data with smaller domain offset. In the cross-library test of the Celeb-DF library, the two thresholds determined by different domains have different effects on each detection method. It is worth mentioning that the threshold calculated by HeadPose under the EER criterion is 0.382, which bias towards 0.5. But the thresholds calculated by other detection methods under the EER criterion are 0.999 (Two-stream), 0.995 (MesoNet), 0.965 (FWA), 0.988 (VA) and 0.972(Multi-task), which all close to 1, which shows different Deepfakes detection methods have diverse detection sensitivities for the detection objects with different synthetic qualities.

Chapter 6

Conclusion

With the rapid development of deep learning technology, the synthesised Deepfakes face is becoming more and more vivid. To identify and fight against the fake face generation technologies, researchers proposed some CNN-based classifiers for detection. This thesis proposed a new framework to reproduce the detection effects of the six models of Two-stream, MesoNet, HeadPose, FWA, VA and Multi-task. Intra-library testing and cross-library testing have conducted on the FaceForensics++, TIMIT, and Celeb-DF databases. Experiments showed that these detection methods can better detect videos with a small domain offset (in the intra-library), but they perform poorly on videos with a large domain offset (in the cross-library). As for it, this thesis discussed some factors that may affect generalisation ability. The experimental conclusions are summarised as follows:

- 1) The dataset partitioning has a direct impact on the performance of the detection method. In the intra-library test of the TIMIT dataset, the accuracy rate obtained by person partitioning has comparing with the dataset by random partitioning. The declining scope of Two-stream is the most noticeable. In the random partitioning, the accuracy rate of Two-stream is 98.5 per cent, while in the person partitioning, it drops to 88.6 per cent. Meanwhile, the equal error rate and the average error rate of each detection method all increased to some extent. In the intra-library test of the FaceForensics++ dataset, although the two partitioning methods have an impact on the detection method, there is no any regularity, and the change directions of accuracy rate, equal error rate, and average error rate are varying. In the cross-dataset test, the average error rate of the Celeb-DF dataset is high or low in the two partitioning forms. But in

the other two datasets, the average error rate of the dataset divided by the person is significantly lower than random division. For instance, MesoNet’s HTER obtained by random partitioning is 45.5 per cent, and the HTER obtained by person partitioning is 40.2 per cent, and there is a drop 5 per cent. Furthermore, for a small-scale dataset like TIMIT, random partitioning makes it easy for the model to learn the characteristics of the face object itself rather than the tampering characteristics of the changing face. Thus, HTER is low in the intra-library test, but HTER is high in the cross-library test. For a large-scale dataset such as FaceForensics++, random partitioning rarely occurs the HTER of intra-library testing is significantly lower than person partitioning. Even HTER of some detection methods would decline via person partitioning. For instance, MesoNet, the HTER obtained by random partitioning is 6.7, while the HTER obtained by dividing by person is 2.3. Given that there are not numerous large datasets available, to improve generalisation performance, the database should be divided by person. The optimum method under dataset partitioning is shown in Table 6.1.

Dataset Partitioning	Intra- library		Cross-library	
	Random Partitioning	Person Partitioning	Random Partitioning	Person Partitioning
Two-stream	-	-	-	-
MesoNet	-	-	-	-
HeadPose	-	Best (in FaceForensics++)	Best	-
FWA	-	-	-	Best
VA	-	-	-	-
Multi-task	Best (in TIMIT and FaceForensics++)	Best (in TIMIT)	-	-

TABLE 6.1: The optimum method under dataset partitioning

2) In a general sense, the data augmentation technology that can improve the generalisation performance of the detection method is not effective in the detection of deep fake face video. In the intra-library test, the accuracy of the six detection methods fluctuates within 1per cent. In the cross-library test, the range of HTER change is mostly between 0.5 per cent to 5 per cent. Taking FWA as an example, when the model trained with TIMIT dataset in testing FaceForensics++ dataset, HTER reduced by about 5 per cent without data augmentation. But it increased by 0.5 per cent when testing the Celeb-DF dataset. These comparisons show that data augmentation can interfere with them when Deepfakes detection method extracts tampering features. Therefore, data augmentation cannot effectively improve the generalisation performance of the detection method. The optimum method under data augmentation is shown in Table 6.2.

Data Augmentation	Intra- library		Cross-library	
	Yes	No	Yes	No
Two-stream	-	-	-	-
MesoNet	-	-	-	Best
HeadPose	Best (in FaceForensics++)	Best (in FaceForensics++)-	-	-
FWA	-	-	Best	-
VA	-	-	-	-
Multi-task	Best (in TIMIT)	Best (in TIMIT)-	-	-

TABLE 6.2: The optimum method under data augmentation

3) Cross-library testing must consider the distribution of source domain data and target domain data. Directly using the threshold determined by the source domain data usually results in a higher error rate in the target domain. In the intra-library test, the accuracy rate obtained with the threshold value (0.5) under the Softmax criterion is lower than the accuracy rate gained under the threshold value under the EER criterion. Among them, the decrease of Two-stream is the most, at about 6 per cent. In the cross-library test of the FaceForensics++ dataset, the HTER obtained by the threshold value (0.5) under the Softmax criterion is higher than the HTER obtained by the threshold value under the EER criterion, and the maximum difference exceeds 10 per cent. In the cross-library test of the Celeb-DF dataset, for MesoNet, FWA and Multi-task, the threshold determined by the data with a larger domain offset is better than the threshold determined by the data with a smaller domain offset. For Two-stream, HeadPose and VA, the results are just the opposite. Moreover, the threshold calculated by HeadPose under the EER criterion is biased towards 0.5, while the threshold calculated by other detection methods under the EER criterion is close to 1. These comparisons note that the detection objects of different synthesis qualities have different effects on different detection methods. The optimum method under thresholds selection is shown in Table 6.3.

Thresholds Selection	Intra- library		Cross-library	
	Softmax criterion	EER criterion	Softmax criterion	EER criterion
Two-stream	-	-	-	-
MesoNet	-	-	-	-
HeadPose	-	-	-	-
FWA	-	-	-	-
VA	-	-	-	-
Multi-task	Best	Best	Best	Best

TABLE 6.3: The optimum method under thresholds selection

Since deep networks are data-driven, the source, target size, synthetic method, and synthesis quality of fake face videos have brought tremendous challenges to the design of forgery face detection methods based on deep networks. Research on the generalisation ability of detection methods is a broad and complex issues, which has enormous theoretical and practical significance for the practical application of Deepfakes detection methods.

References

- [1] M. Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9(11), pp. 39–52, 2019.
- [2] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, 2018.
- [3] K. Saxena and N. John, "Deepfake is a fast-growing monster. here's why man and machine must join hands to tame it," *The EconomicTimes*, 2019. [Online]. Available: <https://prime.economictimes.indiatimes.com/news/69050623/technology-and-startups/deepfake-is-a-fast-growing-monster-heres-why-man-and-machine-must-join-hands-to-tame-it->.
- [4] D. Harris, "Deepfakes: False pornography is here and the law cannot protect you," *Duke L. Tech. Rev.*, vol. 17, pp. 99–127, 2018.
- [5] J. Fletcher, "Artificial intelligence, and some kind of dystopia: The new faces of online post-fact performance," *Theatre Journal*, vol. 70(4), pp. 455–471, 2018.
- [6] B. Chesney and D. Citron, "Deep fakes: A looming challenge for privacy, democracy, and national security," *Calif. L. Rev.*, vol. 107, p. 1753, 2019.
- [7] C. Vaccari and A. Chadwick, "Deepfakes and disinformation: Exploring the impact of synthetic political video on deception, uncertainty, and trust in news," *Social Media+ Society*, vol. 6(1), pp. 1–13, 2020.
- [8] M. Albahar and J. Almalki, "Deepfakes: Threats and countermeasures systematic review," *Journal of Theoretical and Applied Information Technology*, vol. 97(22), pp. 3242–3250, 2019.

References

- [9] S. Catherine, “Fraudsters used ai to mimic ceo’s voice in unusual cybercrime case,” *The Wall Street Journal*, 2019. [Online]. Available: <https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402?mod=searchresults&page=1&pos=1>.
- [10] S. Raphael, “Experts: Spy used ai-generated face to connect with targets,” *AP NEWS*, 2019. [Online]. Available: <https://apnews.com/bc2f19097a4c4fffaa00de6770b8a60d>.
- [11] S. J. Sohrawardi, A. Chintha, B. Thai, S. Seng, A. Hickerson, R. Ptucha, and M. Wright, “Poster: Towards robust open-world detection of deepfakes,” In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2613–2615, 2019.
- [12] P. Korshunov and S. Marcel, “Deepfakes: A new threat to face recognition? assessment and detection,” *arXiv preprint arXiv:1812.08685*, 2018.
- [13] S. Marcel and P. Korshunov, “Vulnerability assessment and detection of deepfake videos,” In *The 12th IAPR International Conference on Biometrics (ICB)*, pp. 1–6, 2019.
- [14] C. Sanderson, “The vidtimit database (no. rep_{work}),” *IDIAP*, 2002.
- [15] “Microsoft research asia (msra),” *Microsoft Research Lab*, [Online]. Available: <https://www.microsoft.com/en-us/research/lab/microsoft-research-asia/>.
- [16] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, and B. Guo, “Face x-ray for more general face forgery detection,” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5001–5010, 2020.
- [17] *TensorFlow*, [Online]. Available: <https://www.tensorflow.org/>.
- [18] C. Tang, S. Chen, L. Fan, L. Xu, Y. Liu, Z. Tang, and L. Dou, “A large-scale empirical study on industrial fake apps,” In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 183–192, 2019.

References

- [19] T. Sathish, T. S. Abinaya, B. Anupriya, and L. Uma, "Manual fakeapp detection using sentimental analysis through webpage," *Semantic Scholar*, pp. 208–221, 2018.
- [20] I. Beridze and J. Butcher, "When seeing is no longer believing," *Nature Machine Intelligence*, vol. 1(8), pp. 332–334, 2019.
- [21] "Faceswap," *Microsoft*, [Online]. Available: <https://faceswap.dev>.
- [22] P. Ketan, "Microsoft's new face swap app includes bing image search and face swapping technology," *Gadgets 360*, 2017. [Online]. Available: <https://gadgets.ndtv.com/apps/news/microsoft-face-swap-app-launch-garage-bing-image-search-background-swapping-1708170>.
- [23] A. Amy, "All that's not fit to print: Fake news and the call to action for librarians and information professionals," *Emerald Group Publishing*, 2019.
- [24] Z. Zhang, "Detect forgery video by performing transfer learning on deep neural network, (doctoral dissertation)," *Sam Houston State University*, 2019.
- [25] B. Paris and J. Donovan, "Deepfakes and cheap fakes," *United States of America: Data Society*, 2019.
- [26] A. Zucconi, "Deepfakes and cheap fakes," *Understanding the Technology Behind DeepFakes*, 2018. [Online]. Available: <https://www.alanzucconi.com/2018/03/14/understanding-the-technology-behind-deepfake>.
- [27] r/deepfakes, "Deepfakes and cheap fakes," *Reddit*, 2017. [Online]. Available: <https://www.reddit.com/r/deepfakes>.
- [28] faceswap GAN, "Deepfakes and cheap fakes," *Github*, 2019. [Online]. Available: <https://github.com/shaoanlu/faceswap-GAN>.
- [29] I. Korshunova, W. Shi, J. Dambre, and L. Theis, "Fast face-swap using convolutional neural networks," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3677–3685, 2017.
- [30] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, "On face segmentation, face swapping, and face perception," *2018 13th IEEE International Conference on Automatic Face Gesture Recognition*, pp. 98–105, 2018.

- [31] Y. Nirkin, Y. Keller, and T. Hassner, "Fsgan: Subject agnostic face swapping and reenactment," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7184–7193, 2019.
- [32] L. G. Li, J. M. Bao, H. Yang, D. Chen, and F. Wen, "Faceshifter: Towards high fidelity and occlusion aware face swapping," *arXiv preprint arXiv:1912.13457*, 2019.
- [33] J. Thies, M. Zollhofe, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2387–2395, 2016.
- [34] J. Thies, M. Zollhöfer, C. Theobalt, M. Stamminger, and M. Nießner, "Headon: Real-time reenactment of human portrait videos," *ACM Transactions on Graphics (TOG)*, vol. 37(4), pp. 1–13, 2018.
- [35] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: Learning lip sync from audio," *ACM Transactions on Graphics (TOG)*, vol. 36(4), pp. 1–13, 2017.
- [36] H. Kim, P. Garrido, A. Tewari, W. P. Xu, J. Thies, M. Niessner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt, "Deep video portraits," *ACM Transactions on Graphics (TOG)*, vol. 37(4), pp. 1–14, 2018.
- [37] E. Zakharov, A. Shysheya, E. Burkov, and V. Lempitsky, "Few-shot adversarial learning of realistic neural talking head models," *In Proceedings of the IEEE International Conference on Computer Vision*, pp. 9459–9468, 2019.
- [38] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: Image synthesis using neural textures," *arXiv preprint arXiv:1904.12356*, 2019.
- [39] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi, "Deep learning for deepfakes creation and detection," *arXiv preprint arXiv:1909.11573*, 2019.
- [40] Y. Zhang, L. Thing, and V. L. L. Zheng, "Automated face swapping and its detection," *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, pp. 15–19, 2017.

References

- [41] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *European conference on computer vision*, pp. 404–417, 2006.
- [42] C. C. Chang and C. J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2(3), pp. 1–27, 2011.
- [43] J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information Forensics and Security*, vol. 7(2), pp. 432–444, 2012.
- [44] L. Zheng, S. Duffner, K. Idrissi, C. Garcia, and A. Baskurt, "Siamese multi-layer perceptrons for dimensionality reduction and face identification," *Multimedia Tools and Applications*, vol. 75(9), pp. 5055–5073, 2016.
- [45] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," *Technical Report 07-49, University of Massachusetts, Amherst*, 2008.
- [46] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Two-stream neural networks for tampered face detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1831–1839, 2017.
- [47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.
- [48] P. Ferrara, T. Bianchi, A. D. Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of cfa artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7(5), pp. 1566–1577, 2012.
- [49] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, "Multi-task learning for detecting and segmenting manipulated facial images and videos," *arXiv preprint arXiv:1906.06876*, 2019.
- [50] J. H. Bappy, C. Simons, L. Nataraj, B. Manjunath, and A. K. Roy-Chowdhury, "Hybrid lstm and encoder-decoder architecture for detection of image forgeries," *IEEE Transactions on Image Processing*, vol. 28(7), pp. 3286–3300, 2019.

- [51] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: Using capsule networks to detect forged images and videos," *In ICASSP 2019-2019 IEEE International Conference on Acoustics*, pp. 2307–2311, 2019.
- [52] A. R. ssler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics: A large-scale video dataset for forgery detection in human faces," *arXiv preprint arXiv:1803.09179*, 2018.
- [53] A. R. ssler, L. V. D. Cozzolino, J. Thies, C. Riess, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," *arXiv preprint arXiv:1901.08971*, 2019.
- [54] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, 2018.
- [55] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.
- [56] Y. Li, M. C. Chang, and S. Lyu, "In ictu oculi: Exposing ai created fake videos by detecting eye blinking," *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, 2018.
- [57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [58] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *In CVPR*, pp. 2625–2634, 2015.
- [59] E. Sabir, J. X. Cheng, A. Jaiswal, W. AbdAlmageed, I. Masi, and P. Natarajan, "Recurrent convolutional strategies for face manipulation detection in videos," *Interfaces (GUI)*, vol. 3(1), pp. 80–87, 2019.
- [60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *In CVPR*, pp. 770–778, 2016.

References

- [61] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *In CVPR*, pp. 4700–4708, 2017.
- [62] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: A compact facial video forgery detection network," *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, 2018.
- [63] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, pp. 1–7, 2015.
- [64] N. Murray and F. Perronnin, "Generalized max pooling," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2473–2480, 2014.
- [65] Y. Li and S. Lyu, "Exposing deepfake videos by detecting face warping artifacts," *arXiv preprint arXiv:1811.00656*, 2018.
- [66] K. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [67] J. Wu, K. Feng, X. Chang, X., and T. Yang, "A forensic method for deepfake image based on face recognition," *In Proceedings of the 2020 4th High Performance Computing and Cluster Technologies Conference 2020 3rd International Conference on Big Data and Artificial Intelligence*, pp. 104–108, 2020.
- [68] P. Korshunov and S. Marcel, "Deepfakes: A new threat to face recognition? assessment and detection," *arXiv preprint arXiv:1812.08685*, 2018.
- [69] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8261–8265, 2019.
- [70] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 83–92, 2019.

References

- [71] C. Chen, M. N. Do, and J. Wang, "Robust image and video dehazing with visual artifact suppression via gradient residual minimization," *In European Conference on Computer Vision*, pp. 576–591, 2016.
- [72] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers security*, vol. 21(5), pp. 439–448, 2002.
- [73] T. Windeatt, "Ensemble mlp classifier design," *In Computational Intelligence Paradigms*, pp. 133–147, 2008.
- [74] M. A. Mansournia, A. Geroldinger, S. Greenland, and G. Heinze, "Separation in logistic regression: Causes, consequences, and control," *American journal of epidemiology*, vol. 187(4), pp. 864–870, 2018.
- [75] Y. M. Gu, M. M. He, K. Nagano, and H. Li, "Protecting world leaders against deep fakes," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 38–45, 2019.
- [76] T. Baltrusaitis, P. Robinson, and L. P. Morency, "Openface: An open source facial behavior analysis toolkit," *In IEEE Winter Conference on Applications of Computer Vision*, pp. 1–10, 2016.
- [77] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L. P. Morency, "Openface 2.0: Facial behavior analysis toolkit," *In 13th IEEE International Conference on Automatic Face Gesture Recognition*, pp. 59–66, 2018.
- [78] P. Ekman and W. V Friesen, "Measuring facial movement," *Environmental psychology and nonverbal behavior*, pp. 56–75, 1976.
- [79] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.
- [80] M. Yousef, "K-means based one-class svm classifier," *In Database and Expert Systems Applications: DEXA 2019 International Workshops BLOKDD*, vol. 1062, pp. 45–53, 2019.
- [81] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, vol. 7, pp. 41 596–41 606, 2019.
- [82] "Lpfs is the distributed web," *LPFS*, [Online]. Available: <https://ipfs.io/>.

References

- [83] L. Li, J. Bao, T. Zhang, H. Yang, D. Chen, F. Wen, and B. Guo, "Face x-ray for more general face forgery detection," *arXiv preprint arXiv:1912.13458*, 2019.
- [84] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30(7), pp. 1145–1159, 1997.
- [85] P. Korshunov and S. Marcel, "Deepfakes: A new threat to face recognition? assessment and detection," *arXiv preprint arXiv:1812.08685*, 2018.
- [86] "Vidtimit," [Online]. Available: <http://conradsanderson.id.au/vidtimit/>.
- [87] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Face-forensics++: Learning to detect manipulated facial images," *In Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–11, 2019.
- [88] "Deepfakedetection," [Online]. Available: <https://ai.googleblog.com/2019/09/contributing-data-to-Deepfakes-detection.html>.
- [89] "Deepfakes detection challenge," [Online]. Available: <https://Deepfakedetectionchallenge.ai>.
- [90] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. Ferrer, "The deepfakes detection challenge (dfdc) preview dataset," *arXiv preprint arXiv:1910.08854*, 2019.
- [91] Y. Z. Li, X. Yang, P. Sun, H. G. Qi, and S. W. Lyu, "Celeb-df: A large-scale challenging dataset for deepfakes forensics," *arXiv preprint arXiv:1909.12962*, 2019.
- [92] "Celeb-df(v2)," [Online]. Available: <http://www.cs.albany.edu/~lsw/celeb-deepfakeforensics.html>.
- [93] A. Clark, "Pillow," 2010. [Online]. Available: <https://python-pillow.org/>.
- [94] G. Bradski and A. Kaehler, "Learning opencv: Computer vision with the opencv library," *O'Reilly Media, Inc.*, 2008.
- [95] D. Sengupta, A. Biswas, and P. Gupta, "Non-linear weight adjustment in adaptive gamma correction for image contrast enhancement," *Multimedia Tools and Applications*, pp. 1–28, 2020.
- [96] A. A. Tazehkandi, "Computer vision with opencv 3 and qt5: Build visually appealing, multithreaded, cross-platform computer vision applications," *Packt Publishing Ltd*, 2018.

References

- [97] A. D. Gavrilov, A. Jordache, M. Vasdani, and J. Deng, "Preventing model overfitting and underfitting in convolutional neural networks," *International Journal of Software Science and Computational Intelligence (IJSSCI)*, vol. 10(4), pp. 19–28, 2018.
- [98] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," *In 2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015.
- [99] H. Hofbauer and A. Uhl, "Calculating a boundary for the significance from the equal-error rate," *In 2016 International Conference on Biometrics (ICB)*, pp. 1–4, 2016.
- [100] R. M. Diez, C. Conde, and E. Cabello, "Automatic detection of the optimal acceptance threshold in a face verification system," *In International Workshop on Biometric Authentication*, pp. 70–79, 2004.
- [101] E. C. Lee, K. R. Park, and J. Kim, "Fake iris detection by using purkinje image," *In International Conference on Biometrics*, pp. 397–403, 2006.
- [102] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," *In ICML*, vol. 2(3), pp. 7–16, 2016.
- [103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, J. Vanderplas, and et al., "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [104] S. Visa, B. Ramsay, A. L. Ralescu, and E. V. D. Knaap, "Confusion matrix-based feature selection," *MAICS*, vol. 710, pp. 120–127, 2011.
- [105] D. J. Hand, "Measuring classifier performance: A coherent alternative to the area under the roc curve," *Machine Learning*, vol. 77, pp. 103–123, 2009.
- [106] D. King, "Dlib c++ library," 2012. [Online]. Available: <http://dlib.net>.

References

- [107] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2d and 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks)," *In Proceedings of the IEEE International Conference on Computer Vision*, pp. 1021–1030, 2017.
- [108] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.
- [109] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [110] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci.*, vol. 6(5), pp. 566–568, 2019.
- [111] F. Zou, L. Shen, z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of adam and rmsprop," *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 11 127–11 135, 2019.
- [112] I. E. Richardson, "The h. 264 advanced video compression standard," *John Wiley and Sons*, 2011.
- [113] J. Bienik, M. Uhrina, and P. Kortis, "Impact of constant rate factor on objective video quality assessment," *Telecommunication Technologies and Optics 2017*, vol. 15(4), pp. 673–682, 2017.
- [114] M. Summerfield, "Programming in python 3: A complete introduction to the python language," *Addison-Wesley Professional*, 2010.
- [115] S. Raschka and V. Mirjalili, "Python machine learning: Machine learning and deep learning with python, scikit-learn, and tensorflow 2," *Packt Publishing Ltd*, 2019.
- [116] "Keras," [Online]. Available: <https://keras.io>.
- [117] S. V. D. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in science and engineering*, vol. 13(2), pp. 22–30, 2011.
- [118] A. Kaehler and G. Bradski, "Learning opencv 3: Computer vision in c++ with the opencv library," " *O'Reilly Media, Inc.*", 2016.

References

- [119] M. Lutz, "Programming python," " O'Reilly Media, Inc.", 2001.
- [120] "Sklearn," [Online]. Available: <http://scikit-learn.org>.
- [121] J. D. Hunter, "Matplotlib: A 2d graphics environmen," *Computing in science and engineering*, vol. 9(3), pp. 90–95, 2007.
- [122] Y. Peng, X. He, and J. Zhao, "Object-part attention model for fine-grained image classification," *IEEE Transactions on Image Processing*, vol. 27(3), pp. 1487–1500, 2017.
- [123] O. Deniz, G. Bueno, J. Salido, and F. D. la Torre, "Face recognition using histograms of oriented gradients," vol. 32, pp. 1598–1603, 2011.
- [124] N. Ketkar, "Introduction to pytorch," *In Deep learning with Python*, pp. 195–208, 2017.

Appendix A

Python Packages Used by the Proposed Framework

The framework was developed utilising Python; In the following, we will present the main Python packages used in the framework development.

A.1 OS

The os module [114] is a pattern for accessing and operating relevant functions of the system in Python official library. The os pattern offers the portable method of employing the operating system functions. Use the interface provided in the os module to achieve cross-platform access. The main functions of the os module: system-related, directory and file operations, command execution and process management.

A.2 Tensorflow

TensorFlow [115] is a Python library for numerical calculations, which can describe a data flow graph of data calculations. Initially, TensorFlow developed by engineers and researchers from Google Brain Group (belonging to Google Machine Intelligence Research Institute) for research on deep neural networks and machine learning. However, the system's versatility allows it to extensively used in additional calculations field.

A.3 Keras

Keras [116] is an open-source artificial neural network library, which was compiled by Python. Microsoft-CNTK, Tensorflow and Theano used it as the high-level application program interface for designing deep layer, debugging, applying, evaluating, and visualizing learning pattern. Keras wrote by the object-oriented pattern via its code construction. Its structure is completely modular and extensible. Keras considers difficulty of use and experience of user from documents and operational mechanisms, and tries to predigest the difficulty for achieving complicated algorithms. Moreover, Keras sustains mainstream algorithms in the domain of artificial intelligence, comprising the neural networks with recursive and feedforward constructions.

A.4 Numpy

NumPy (Numerical Python) [117] is the open-source mathematical computing extension of Python. It can employ to store and process vast matrices, which is effective than nested list construction of Python (the construction can also use to describe matrix) and supports matrix operations and vast dimensional arrays. Moreover, it also gives numerous mathematical function libraries for computing arrays.

NumPy gives numerous high-level mathematical programming tools, for instance, vector processing, matrix data types, and complicated arithmetic libraries. And it can say that it designed for strict digital processing.

A.5 CV2 (OpenCV-Python)

OpenCV (Open Source Computer Vision Library) [118] is the cross-platform computer vision library published according to the BSD license (open source). It gives many functions that implement computer vision algorithms very efficiently. OpenCV has a wide range of applications, including image stitching, image noise reduction, product quality inspection, human-computer interaction, face recognition, motion recognition, motion tracking, UAV (Unmanned Aerial Vehicle) driving, etc. OpenCV also provides

a machine learning module, which can use machine learning algorithms, for instance, K nearest neighbors, normal Bayes, support vector machines, artificial neural networks, decision trees, and random forests.

A.6 SYS

Python's sys pattern gives an accessing for an interpreter that apply to use and maintain variables, and it can have interactive functions with the interpreter. In layman's terms, the sys module [119] is accountable for intercommunication between Python interpreter and program, and it gives a series of variables and functions, which manipulates a running environment of Python.

A.7 Dlib

Dlib [106] means the open-source library to machine learning, which contains many algorithms for machine learning. It is very convenient to use, including the header files directly, and does not depend on other libraries. Dlib can assist the user generates various complicated software of machine learning, which resolve real problems. At present, Dlib has employed in academic domains and industry widely, comprising mobile phones, embedded devices, robotics and large-scale high-performance computing surroundings.

A.8 Sklearn

Scikit-learn, referred to as sklearn [120], is a commonly used third-party module in machine learning and sustains four classes machine learning algorithms, that is regression, classification, clustering and dimensionality reduction. And sklearn also has three modules: data processing, feature extraction, and model evaluator.

A.9 Random

Random implements [119] distributed pseudo-random quantity generators. As integers, taking a uniform choice from the scope. As a sequence, taking a consolidated choice from the random elements. A function for generating the random arrangement of the list, and a function for the random sampling without replacement.

A.10 Matplotlib

The matplotlib [121] is a 2D plotting library for Python, which is inspired by MATLAB. It gives a suite of command API alike to Matlab, which is pretty fitting for interactive drawing, which can adapt as the drawing control, and let into GUI applications. With Matplotlib, developer generates plots, bar graphs, power spectra, histograms, error graphs, scatter plots, and so on with only several rows of code.

A.11 Warnings

Python sends out a warning via invoking warn() function from the warnings module [114]. Warning messages usually used to remind users of some errors or outdated usage. When these situations occur, we don't want to throw exceptions or exit the program directly. Warning messages usually are written to sys.stderr, and the handling of warnings can be flexibly changed, such as being ignored or turned into an exception. Warnings' processing can be diverse according to warning division, the version of a warning message, and the source position of a warning message. The repetition of specific warnings for the same source location usually suppressed.

A.12 Attention_model

The attention model [122] implemented to picture identification, which simulates the action of the focus of eyes on diverse objects. When the neural network identifies images or word, it concentrates on several characteristics each time, and the identification

is further accurate. The most intuitive way to estimate the influence of characteristics is the weight. Therefore, the outcome of the attention model is first to gauge the weight of per characteristic during each recognition. Then, performing a weighted summation of the characteristics. The weight is greater, the contribution of the feature is the more excellent to identification currently.

A.13 Time

Python's time module [114] allows using all functions with time is relevant. Most of them only invoke holding the same names' Platform C library functions. However, there may be some differences between different platforms. The time module is available in all versions of Python. It is a built-in library for Python specifically to handle time.

A.14 Face_recognition

Face_recognition [123] is an open-source face recognition library for Python, which supports Python 3.3+ and Python 2.7. The library is pretty simple to implement a face recognition program, and the environment configuration is also pretty easy. The library can directly use the already trained model without retraining locally. Generally, ordinary computers can directly run the recognition program, and the hardware environment requirements are not high.

A.15 Imageio

Imageio[119] is a Python library that gives an accessible interface to view and record various picture data, comprising scientific formats, animated images, videos, and volume data. It is cross-platform, can run on Python 2.7 and 3.4+, and is easy to install.

Imageio has a relatively simple core and can provide a common interface for different

file formats. The kernel is responsible for reading data from diverse sources (such as HTTP) and provides a simple API for the plug-in to access the original data. All file formats are implemented in the plugin.

A.16 Pickle

The pickle library [115] is used to convert between Python-specific types and Python data types. Pickle gives a simplistic vitality function, which stores objects on the hard disk drive in the modality of records. And it only uses in Python. Nearly every data type (dictionaries, lists, classes, sets, and so on) in Python serializes with the pickle. The data after the pickle serialization is unwell readability, and generally, humans do not be recognizable.

A.17 Image

The Python image library (PIL) [115] is a third-party image processing library for Python, due to its powerful functions and a large number of users, it has almost regarded as the official Python image processing library. PIL has a long history, and it only supports python2.x originally. Later, it is ported to pillow library of Python3. Pillow is called a friendly fork for PIL. Its function is similar to PIL, but it supports python3.

A.18 Pandas

The pandas [115] means an open-source Python library relies on NumPy, which employed widely for analysing data rapidly, cleaning data and preparing data. The source of its name made up of the two words "Panel data" (an econometric noun). In short, can think of Pandas as the Python version of Excel.

A.19 Torch.optim

The torch.optim [124] is a library that implements various optimization algorithms, and it supports most commonly used methods, as well as the interface is sufficiently versatile to enable the integration of more complex methods in the future.

In the interest of using torch.optim, the user needs to construct an optimizer object, which can sustain the current parameter status, and renew parameters according to the calculated gradient. For building an optimizer, the user needs to give it an iterable containing the parameters to be optimized (must be Variable objects). Then, the user can set the parameter options of the optimizer, such as learning rate, weight decay.

Appendix B

Some Samples of the Proposed Framework Codes

B.1 Partial Codes of Adam Algorithm

```
1  def step(self, closure=None):
2
3      loss = None
4      if closure is not None:
5          loss = closure()
6
7      for group in self.param_groups:
8          for p in group['params']:
9              if p.grad is None:
10                 continue
11                 grad = p.grad.data
12                 if grad.is_sparse:
13                     raise RuntimeError('Adam does not support
14                                     sparse gradients, please consider
15                                     SparseAdam instead')
16                 amsgrad = group['amsgrad']
17
18                 state = self.state[p]
```

```

17
18     # State initialization
19     if len(state) == 0:
20         state['step'] = 0
21         # Exponential moving average of gradient
           values
22         state['exp_avg'] = torch.zeros_like(p.data)
           # [batch, seq]
23         # Exponential moving average of squared
           gradient values
24         state['exp_avg_sq'] = torch.zeros_like(p.
           data)
25         if amsgrad:
26             # Maintains max of all exp. moving avg.
               of sq. grad. values
27             state['max_exp_avg_sq'] = torch.
               zeros_like(p.data)
28
29     exp_avg, exp_avg_sq = state['exp_avg'], state['
           exp_avg_sq']
30     if amsgrad:
31         max_exp_avg_sq = state['max_exp_avg_sq']
32     beta1, beta2 = group['betas']
33
34     state['step'] += 1
35     bias_correction1 = 1 - beta1 ** state['step']
36     bias_correction2 = 1 - beta2 ** state['step']
37
38     if group['weight_decay'] != 0: # Perform
           weight decay (L2 regularization)
39         # 6. grad(t)=grad(t-1)+ weight*p(t-1)

```

```

40         grad.add_(group[ 'weight_decay' ], p.data)
41
42         # Decay the first and second moment running
           average coefficient
43         # Calculate  $m(t)$ :  $m(t)=\beta_1*m(t-1)+(1-\beta_1)$ 
           *grad
44         exp_avg.mul_(beta1).add_(1 - beta1, grad)
45         # Calculate  $v(t)$ :  $v(t)=\beta_2*v(t-1)+(1-\beta_2)$ 
           )*grad^2
46         exp_avg_sq.mul_(beta2).addcmul_(1 - beta2, grad
           , grad)
47         if amsgrad:
48             # Maintains the maximum of all 2nd moment
           running avg. till now
49             # Iteratively change the value of
           max_exp_avg_sq (take the maximum value),
           and pass it to the next time, retaining
           the previous gradient information
50             torch.max(max_exp_avg_sq, exp_avg_sq, out=
           max_exp_avg_sq)
51             # Use the max. for normalizing running avg.
           of gradient
52             denom = (max_exp_avg_sq.sqrt() / math.sqrt(
           bias_correction2)).add_(group[ 'eps' ])
53         else:
54             # Calculate  $\sqrt{v(t)}+\epsilon$ 
55             #  $\sqrt{v(t)}+\epsilon = \text{denom} = \sqrt{v(t)}/$ 
            $\sqrt{1-\beta_2^t}+\epsilon$ 
56             denom = (exp_avg_sq.sqrt() / math.sqrt(
           bias_correction2)).add_(group[ 'eps' ])

```

```

57         # step_size=lr/bias_correction1
           =lr/(1-beta_1^t)
58         step_size = group[ 'lr ' ] / bias_correction1
59         #p(t)=p(t-1)-step_size*m(t)/
           denom
60         p.data.addcdiv_(-step_size , exp_avg , denom)
61
62     return loss

```

B.2 Rotating Training Data for Data Augmentation

```

1  import cv2
2  import math
3  import numpy as np
4  import os
5  import pdb
6  import xml.etree.ElementTree as ET
7
8  class ImgAugemention():
9      def __init__(self):
10         self.angle = 90
11
12         # rotate_img
13         def rotate_image(self , src , angle , scale=1.):
14             w = src.shape[1]
15             h = src.shape[0]
16             # convet angle into rad
17             rangle = np.deg2rad(angle) # angle in radians
18             # calculate new image width and height
19             nw = (abs(np.sin(rangle)*h) + abs(np.cos(rangle)*w))*
                scale

```

```

20     nh = (abs(np.cos(rangle)*h) + abs(np.sin(rangle)*w))*
        scale
21     # ask OpenCV for the rotation matrix
22     rot_mat = cv2.getRotationMatrix2D((nw*0.5, nh*0.5),
        angle, scale)
23     # calculate the move from the old center to the new
        center combined
24     # with the rotation
25     rot_move = np.dot(rot_mat, np.array([(nw-w)*0.5, (nh-h)
        *0.5, 0]))
26     # the move only affects the translation, so update the
        translation
27     # part of the transform
28     rot_mat[0, 2] += rot_move[0]
29     rot_mat[1, 2] += rot_move[1]
30     # map
31     return cv2.warpAffine(
32         src, rot_mat, (int(math.ceil(nw)), int(math.ceil(nh)
33             )),
34         flags=cv2.INTER_LANCZOS4)
35
36 def rotate_xml(self, src, xmin, ymin, xmax, ymax, angle,
37     scale=1.):
38     w = src.shape[1]
39     h = src.shape[0]
40     rangle = np.deg2rad(angle) # angle in radians
41     # now calculate new image width and height
42     # get width and heigh of changed image
43     nw = (abs(np.sin(rangle)*h) + abs(np.cos(rangle)*w))*
        scale

```

```

42     nh = (abs(np.cos(rangle)*h) + abs(np.sin(rangle)*w))*
        scale
43     # ask OpenCV for the rotation matrix
44     rot_mat = cv2.getRotationMatrix2D((nw*0.5, nh*0.5),
        angle, scale)
45     # calculate the move from the old center to the new
        center combined
46     # with the rotation
47     rot_move = np.dot(rot_mat, np.array([(nw-w)*0.5, (nh-h)
        *0.5, 0]))
48     rot_mat[0, 2] += rot_move[0]
49     rot_mat[1, 2] += rot_move[1]
50     # rot_mat: the final rot matrix
51     # get the four center of edges in the initial
        matrix and convert the coord
52     point1 = np.dot(rot_mat, np.array([(xmin+xmax)/2, ymin,
        1]))
53     point2 = np.dot(rot_mat, np.array([xmax, (ymin+ymax)/2,
        1]))
54     point3 = np.dot(rot_mat, np.array([(xmin+xmax)/2, ymax,
        1]))
55     point4 = np.dot(rot_mat, np.array([xmin, (ymin+ymax)/2,
        1]))
56     # concat np.array
57     concat = np.vstack((point1, point2, point3, point4))
58     # change type
59     concat = concat.astype(np.int32)
60     print(concat)
61     rx, ry, rw, rh = cv2.boundingRect(concat)
62     return rx, ry, rw, rh
63

```

```

64 def process_img(self, imgs_path, xmls_path, img_save_path,
65 xml_save_path, angle_list):
66     # assign the rot angles
67     for angle in angle_list:
68         for img_name in os.listdir(imgs_path):
69             # split filename and suffix
70             n, s = os.path.splitext(img_name)
71             # for the sake of use yolo model, only process
72             # '.jpg'
73             if s == ".jpg":
74                 img_path = os.path.join(imgs_path, img_name
75 )
76                 img = cv2.imread(img_path)
77                 rotated_img = self.rotate_image(img, angle)
78                 save_name = n + "_" + str(angle) + ".jpg"
79                 cv2.imwrite(img_save_path + save_name,
80 rotated_img)
81                 print("log: [%sd] %s is processed." % (
82 angle, img))
83                 xml_url = img_name.split('.')[0] + '.xml'
84                 xml_path = os.path.join(xmls_path, xml_url)
85                 tree = ET.parse(xml_path)
86                 file_name = tree.find('filename').text #
87                 # it is origin name
88                 path = tree.find('path').text # it is
89                 # origin path
90                 # change name and path
91                 tree.find('filename').text = save_name #
92                 # change file name to rot degree name
93                 tree.find('path').text = save_name #
94                 # change file path to rot degree name

```

```

86     root = tree.getroot()
87     # if angle in [90, 270], need to swap
      width and height
88     if angle in [90, 270]:
89         d = tree.find('size')
90         width = int(d.find('width').text)
91         height = int(d.find('height').text)
92         # swap width and height
93         d.find('width').text = str(height)
94         d.find('height').text = str(width)
95
96     for box in root.iter('bndbox'):
97         xmin = float(box.find('xmin').text)
98         ymin = float(box.find('ymin').text)
99         xmax = float(box.find('xmax').text)
100        ymax = float(box.find('ymax').text)
101        x, y, w, h = self.rotate_xml(img, xmin,
      ymin, xmax, ymax, angle)
102        # change the coord
103        box.find('xmin').text = str(x)
104        box.find('ymin').text = str(y)
105        box.find('xmax').text = str(x+w)
106        box.find('ymax').text = str(y+h)
107        box.set('updated', 'yes')
108        # write into new xml
109        tree.write(xml_save_path + n + "_" + str(
      angle) + ".xml")

```

B.3 Partial Codes of Datasets Partitioning

```

1 import os

```

```

2 import random
3 import math
4 import shutil
5
6 def data_split(old_path):
7     new_path = 'data'
8     if os.path.exists('data') == 0:
9         os.makedirs(new_path)
10    # Traverse each triple returned by os.walk() and put the
11    # contents in three variables respectively
12
13    for root_dir, sub_dirs, file in os.walk(old_path):
14
15        for sub_dir in sub_dirs:
16            file_names = os.listdir(os.path.join(root_dir,
17            sub_dir)) # Traverse each sub-directory
18
19            random.shuffle(file_names)
20            for i in range(len(file_names)):
21                if i < math.floor(0.7*len(file_names)):
22                    sub_path = os.path.join(new_path, '
23                    train_set', sub_dir)
24                elif i < math.floor(0.8*len(file_names)):
25                    sub_path = os.path.join(new_path, 'val_set'
26                    , sub_dir)
27                elif i < len(file_names):
28                    sub_path = os.path.join(new_path, 'test_set
29                    ', sub_dir)
30                if os.path.exists(sub_path) == 0:
31                    os.makedirs(sub_path)

```

```
27         shutil.copy(os.path.join(root_dir, sub_dir,
28                               file_names[i]), os.path.join(sub_path,
29                               file_names[i]))
30
31 if __name__ == '__main__':
32     data_path = 'old_data'
33     data_split(data_path)
```

B.4 Partial Codes of Thresholds Selection

```
1 import cv2
2 import os
3 import numpy as np
4 import pandas as pd
5 from tqdm import tqdm
6
7 image = cv2.imread("")
8 def rgb2gray(image):
9     h = image.shape[0]
10    w = image.shape[1]
11    grayimage = np.zeros((h,w),np.uint8)
12    for i in tqdm(range(h)):
13        for j in range(w):
14            grayimage [i,j] = 0.144*image[i,j,0]+0.587*image[i,
15                j,1]+0.299*image[i,j,1]
16
17    return grayimage
18
19 def otsu(image):
20     # width and height
21     h = image.shape[0]
22     w = image.shape[1]
```

```
21  # Get the total pixels
22  m = h*w
23
24  otsuimg = np.zeros((h, w), np.uint8)
25  initial_threshold = 0
26  final_threshold = 0
27  # Initialize the statistical parameters of the number of
   gray levels
28  histogram = np.zeros(256, np.int32)
29  # Initialize the statistical parameters of the distribution
   of each gray level in the image
30  probability = np.zeros(256, np.float32)
31
32  # Statistics of each gray level
33  for i in tqdm(range(h)):
34      for j in range(w):
35          s = image[i, j]
36          histogram[s] = histogram[s] + 1
37  # Statistical parameters of the distribution of each gray
   level in the image
38  for i in tqdm(range(256)):
39      probability[i] = histogram[i]/m
40
41  for i in tqdm(range(255)):
42      w0 = w1 = 0 # The gray number of foreground and
   background
43      fgs = bgs = 0
44      for j in range(256):
45          if j <= i:
46              # The proportion of foreground pixels in the
   entire image is accumulated
```

```

47         w0 += probability[j]
48         fgs += j * probability[j]
49     else:
50         # The proportion of background pixels in the
51         entire image is accumulated
52         w1 += probability[j]
53         bgs += j * probability[j]
54     u0 = fgs / w0
55     u1 = bgs / w1
56     G = w0*w1*(u0-u1)**2
57     if G >= initial_threshold:
58         initial_threshold = G
59         final_threshold = i
60     print(final_threshold)
61
62 for i in range(h):
63     for j in range(w):
64         if image[i, j] > final_threshold:
65             otsuimg[i, j] = 255
66         else:
67             otsuimg[i, j] = 0
68
69 return otsuimg
70
71 grayimage = rgb2gray(image)
72 otsuimage = otsu(grayimage)
73
74 cv2.imshow("grayimage", grayimage)
75 cv2.imshow("otsuimage", otsuimage)
76
77 cv2.waitKey()

```