المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'électronique

End-of-study project dissertation
for obtaining the State Engineer's degree in Electronics

Assessment of Deepfake Detection Techniques: A Study of Performance and Generalisation

Abderezak MECHENET
Under the direction of Mr. Sid-Ahmed BERRANI Prof.

Presented and defended publicly on 02/07/2024

Composition of the jury:

President	Dr. Mohamed Oussaid TAGHI	ENP
Promoter	Pr. Sid-Ahmed BERRANI	ENSIA
Examiner	Pr. Mourad ADNANE	ENP

ENP 2024
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
École Nationale Polytechnique

Département d'électronique

End-of-study project dissertation
for obtaining the State Engineer's degree in Electronics

Assessment of Deepfake Detection Techniques: A Study of Performance and Generalisation

Abderezak MECHENET
Under the direction of Mr. Sid-Ahmed BERRANI Prof.

Presented and defended publicly on 02/07/2024

Composition of the jury:

President	Dr. Mohamed Oussaid TAGHI	ENP
Promoter	Pr. Sid-Ahmed BERRANI	ENSIA
Examiner	Pr. Mourad ADNANE	ENP

ENP 2024
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي
École Nationale Polytechnique

Département d'électronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

Évaluation des techniques de détection de deepfake : une étude de la performance et de la généralisation

Abderezak MECHENET

Sous la direction de M. Sid-Ahmed BERRANI Prof. ENSIA, Sidi Abdellah

Présenté et soutenu publiquement le 02/07/2024

Composition du jury :

Président	Dr. Mohamed Oussaid TAGHI	ENP
Promoteur	Pr. Sid-Ahmed BERRANI	ENSIA
Examinateur	Pr. Mourad ADNANE	ENP

ENP 2024
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie.
www.enp.edu.dz

ملخص

التزييف العميق التطور هو ثمرة التطور في مجالات الذكاء الاصطناعي والتعلم العميق، وقد أدى ظهوره إلى إنشاء مجال جديد: اكتشاف التزييف العميق، وهو مجال متخصص في التحقق من صحة محتوى الوسائط. تهدف الكاشفات القائمة على الشبكات العصبية الالتفافية إلى اكتشاف العيوب التي تتركها عملية التوليد لاستخلاص استنتاجات حول الوسائط المتعددة. كانت دراستنا ممركزة حول تقييم كاشفات التزييف العميق الحالية، بهدف تقييم وتحسين المشكلة التي تم تشخيصها، وهي قدرة التعميم عبر مجموعات بيانات متعددة. تم اختبار منتجنا النهائي على مقاطع فيديو خارجية لاستنتاج الفرضية الموضوعة.

كلمات مفتاحية : اكتشاف التزييف العميق, التعميم, محتوى الوسائط.

Résumé

Les deepfakes sont le fruit du développement dans les domaines de l’intelligence artificielle et de l’apprentissage profond. Leur apparition a créé un nouveau domaine : la détection de deepfake, un domaine qui se spécialise dans la vérification de l’authenticité du contenu média. Les détecteurs basés sur les réseaux de neurones convolutionnels visent à détecter les artefacts laissés par le processus de génération pour tirer des conclusions sur les médias. Notre étude était centrée sur l’évaluation des détecteurs de deepfake existants, visant à évaluer et à améliorer notre problème diagnostiqué, qui est la capacité de généralisation à travers plusieurs ensembles de données. Notre produit final a été testé sur des vidéos externes pour conclure sur l’hypothèse établie.

Mots clés : Détection de deepfake, Généralisation, Contenu média.

Abstract

Deepfakes are the fruit of development in the fields of Artificial Intelligence and Deep Learning, their appearance created a new field: deepfake detection, a domain that specialises in the verification of the authenticity of media content. Convolutional Neural Networks based detectors aim at detecting artifacts left by the generation process to draw conclusions on the multimedia. Our study was centralised around the assessment of existing deepfake detectors, aiming to evaluate and improve our diagnosed problem which is the generalisation ability across multiple datasets, our final product was put to the test on external videos to conclude on the hypothesis established.

Keywords : Deepfake detection, Generalisation, Media content.

Acknowledgments

Most importantly, Al-Hamdulillah, all glory to god. What I have done would never be possible without him on my side. Every step, every blessing, every success, and every choice I made were all made possible thanks to god, he answered my prayers.

"Now, to all those close to me and those who helped me, the next words are straight from my heart. The ENP chapter might be over, but the story continues.

To my parents to whom I'm forever grateful, nothing would be possible if it weren't for them, their unwavering and unconditional support, from days where I was down on the ground to days where I was up high, we shared every emotion together. To my siblings, even if you are annoying sometimes, you have been a blessing to me in my life, I'm forever thankful to what we have been through together. To my grandmother, aunts and uncles, thank you all.

To Yousra, the one who stood by my side for 5 years, a time that felt like an eternity, but at the same time feels like it was only yesterday. You have been there for me from day one, and for that I'm forever in your debt, I am happy faith brought me together with such an amazing person.

To my friends, the Lycée and basketball crew: Amine "the ultimate hype-man", Anis "get it from the rebound", Aymen "La moussa", Farouk "I just throw it up", Hichem "18 years since I have known you", and Mehdi "the elbow", we have shared some great memories together, and I am glad I found people with whom I can be myself fully, which the best player of course, without forgetting Ghazli and Zinou.

To my prepa friends: Moh, Ramzi, Adel, Midou, Boualem, and Omar, separated by continents, united by our love to each-other and to l'annexe, and our hate on prepa.

To ELN, from my promotion "9rib ingénieurs", to Walid "la starte" and Samy, we have shared the best and worst moments of our lives so far together, from happy memories to trying to calm me, I could not have asked for more.

To my fellow IEEE committee members and IEEE crew, we might have had strokes trying to push forward, but we made the most out of it as we created some unforgettable moments along the way.

To the whole reason I was in this school and the reason I chose ELN in the first place, my math teacher Bachir and my aunts husband, you have set me into this path and gave me confidence during my choices.

"It is not our abilities that show what we truly are, it is our choices."

Abderezak-Funguy

Thanks

I would first like to thank my supervisor Mr. Sid-Ahmed Berrani who has helped me along the way. I sincerely appreciate his presence, his useful tips, unwavering support and insightful ideas, I could not have asked for more from a mentor.

I would also like to thank the jury members for taking their time and accepting to evaluate my long lasting work: Mr. Mohamed O. Taghi, the head of the electronics Department, and Mr. Mourad Adnane, our professor at the Ecole Nationale Polytechnique.

I would like to extend my gratitude for every professor who helped me during my five years at the school, what I learned from them is invaluable.

I would most certainly like to thank my supervisor and the head of the industrial engineering department for providing me with the necessary equipment so that I could carry this project properly.

I would also like to thank all my colleagues at the school, what we shared together can never be replaced.

Finally I would like to thank everyone that has helped me during my academic curriculum, even with a simple word.

Abderezak

Contents

List of Tables

List of Figures

Abreviation list

DL	*Deep Learning*
AI	*Artificial intelligence*
GAN	*Generative adversarial Networks*
AE	*Auto Encoders*
TTS	*Text-To-Speech*
VC	*Voice Cloning*
ML	*Machine Learning*
CNN	*Convolutional Neural Network*
FLOPS	*Floating-point operations*
LSTM	*Long Short-Term Mmeory*
DCT	*Discrete Cosine Transform*
DFT	*Discrete Fourier Transform*
DWT	*Discrete Wavelet Transform*
RNN	*Recurrent Neural Networks*
GPU	*Graphics Processing Units*
FF++	*FaceForensics++*
FS	*FaceSwap*
DF	*DeepFakes*

F2F	*Face2Face*
NT	*Neural Textures*
DFD	*DeepFakeDetection*
DFDCP	*Deepfake Detection Challenge-Preview*
DFDC	*Deepfake Detection Challenge*
CDF1	*Celeb-DF-v1*
CDF2	*Celeb-DF-v2*
FSh	*FaceShifter*
DF-1.0	*DeeperForensics-1.0*
WDF	*Wild-Deepfake*
FAVC	*FakeAVCeleb*
TP	*True Positive*
FP	*False Positive*
TN	*True Negative*
FN	*False Negative*
TPR	*True Positive Rate*
FPR	*False Positive Rate*
TNR	*True Negative Rate*
FNR	*False Negative Rate*
EER	*Equal Error Rate*
ACC	*Accuracy*
AUC	*Area Under the Curve*
ROC	*Receiver Operating Characteristic Curve*

F3Net	*Frequency in Face Forgery Network*
FAD	*Frequency Aware Decomposition*
LFS	*Local Frequency Statistics*
SPSL	*Spatial Phase Shallow Learning*
CORE	*Consistent Representation Learning for Face Forgery Detection*
RECCE	*End-to-end Reconstruction Classification Learning*
SBI	*Self Blended Images*
STG	*Source-Target Generator*
MG	*Mask Generator*
FFIW	*Fake Faces In the Wild*
AP	*Average Precision*
CUDA	*Compute Unified Device Architecture*
SSD	*Solid-State Drive*
HDD	*Hard Disk Drive*
RAM	*Random-Access Memory*
JSON	*JavaScript Object Notation*
CDF	*Celeb-DF*
Aug	*Data Augmentation*
RandER	*Random Erase*
BC	*Random Brightness Contrast*
Comp	*Image Compression*
Flip	*Horizontal Flip*
MLP	*Multi-Layer Perceptron*
LipFD	*Lip-syncing Forgery Detection*
FPS	*Frames Per Second*

Introduction

Introduction

Media content has always been important to human kind throughout history, from sculptures and paintings in ancient times, to photos, videos and audios nowadays, visual and audio-visual content has been always a way of telling and writing history, as visuals may speak louder than words. The manipulation of such content is no new deal, as it has been present for ages and for various reasons, from the attempt of glorification of great leaders in history to undermining rivals, picturing important events as greater than they were, eliminating enemies and political rivals, enhancing and restoring personal belongings or pieces of art, to more recent uses in entertainment, frauds, illicit content and attacking enemies or celebrities, resulting in fake news and enormous consequences.

In recent years, fake news has become an issue and a threat to public discourse, human society, and democracy. Fake news refers to fictitious news style content that is fabricated to deceive the public. False information spreads quickly through social media, where it can reach millions of users. In fact, 92% of YouTube users use the platform to gather information and knowledge according to [1]. This high popularity of videos highlights the need for tools to verify media content authenticity, as emerging technologies allow a convincing manipulation of videos.

Given the ease in obtaining and spreading misinformation through social media platforms, it is increasingly hard to know what to trust, which results in harmful consequences for informed decision making, among other things. Indeed, today we live in what some have called a “post-truth” era [2], that facilitates digital disinformation in order to manipulate public opinion.

Recent technological advancement and technologies in Artificial intelligence, computational power, and the availability of gigantic amounts of data have made it possible to create and spread the fake data. Advances in deep learning have seen the emergence of Auto-encoders [3] and Generative Adversarial Networks (GAN) [4], the two pillars and reason for the automatic creation of synthetic media, which has been attributed the label ”deepfakes”.

Deepfakes can vary nowadays, from swapping faces of two subjects [5], creating an audio of a person saying sentences they never said [6], to creating a video of the person saying fake audios and make it seem like they did by syncing the lip to the audio [7], resulting in hyper-realistic media that is hard to impossible to distinguish by the naked eye.

As a result, a new field surfaced in 2018 to battle these falsified videos: Deepfake detection. Early works exploited data-driven approaches [8], they next took an artifact based approach looking for spatial artifacts [9] or temporal inconsistencies [10]. To battle the common issues in this field, new works concentrated on developing more general detectors [11, 12] and shifting the focus towards overlooked forgeries such as LipSync [13].

In this study, our focus shifted towards deepfake detection for what it has to offer as an important and practical tool nowadays for the verification of media content. On that account, we have conducted a review on deepfake detection algorithms by evaluating these methods on existing datasets, while focusing on the issue they all suffer: The generalisation ability to unseen data that has not been considered in the training.

The main contributions of our work are as follows:

- Study and evaluation of deepfake detectors on available public datasets.
- A different training approach that leverages a diverse training data by dataset combinations and detector combination, for an improved generalisation.
- Implementation of the final result on wild deepfakes and outperforming a state of the art specialised detector.

This thesis is composed of six chapters structured as follows:

The first chapter serves as a general introduction to deepfakes, opening the eyes on media manipulation, backed with history facts, all the way into automatic manipulations we see today as deepfakes.

The second chapter exploits the deepfake technology, the most prominent generation methods alongside their bases and technical background, challenges they face and the consequences of such fabricated videos. Then, we will be looking at the deepfake detection principles, challenges, technical background and a thorough classification of existing methods to pave the way for the analysis.

The third chapter concentrates on what enables this hot research topic, that are the deepfake detection datasets and the evaluation metrics used to train and evaluate existing detection methods. We first go into detail about the identified datasets, giving thorough information and showcasing examples, before passing on to the definition of the evaluation metrics and the information they provide.

The fourth chapter concerns our methodology and approach on the topic. First, we study the identified deepfake detection algorithms from literature, giving details about each method and its features. Then, we breakdown the criteria for our choice of datasets and evaluation metrics included in the study.

the fifth and penultimate chapter is the experimentation part, where have detailed the experimental setup alongside setting a baseline for our models, and finishing off with the problems faced.

The final chapter is all about studying the generalisation ability of deepfake detectors, where a detector of choice is put into a thorough analysis to further improve this feature of deepfake detection.

Lastly, are presented a conclusion on the thesis and some perspectives for future works.

Chapter 1

Introduction to Deepfakes

1.1 Introduction

This chapter is a general introduction to deepfake technology, setting the tone for the following technical chapters.

We firstly define what deepfakes are, giving some insights on what defines them as an emerging technology and a topic of interest lately.

Then, we look at the different types of deepfakes, dividing them into two main categories: Audio deepfakes and visual deepfakes, giving the most prominent methods of each.

Then, we break down the evolution of deepfakes across time, giving historical facts and examples, from classic media manipulation to the automation of such process.

Finally, we move on to the various applications, from the positive to the negatives, before ending with a conclusion.

This chapter helps understand the impact of deepfakes on the modern world, underlying its importance, before we move on to its technical background for detection and generalisation.

1.2 Deepfake Definition

Deepfakes are synthetic media i.e. images, videos and audio recordings, that have been automatically manipulated intentionally using deep learning techniques by interested parties such as governments, film directors and editors or geeks for various purposes from entertainment and commercial use to deception, misinformation, disinformation, persuading the general public and attacking people of interest, resulting into hyper-realistic videos or audios that can not be distinguished by humans.

Deepfakes are the acronym and combination of two words, ”deep” which stands for deep learning since it relies on neural networks and artificial intelligence to get created and generated automatically, these models are trained on large datasets of people to learn to mimic a person’s facial expressions, mannerisms, voice and inflections, and ”fake” which stands for fake media that aims to mislead the receivers. Figure 1.1 shows the difference between a deepfake face and a real face, vouching for their realism.

These falsified media gained prominence due to the technological advancement seen in recent years in artificial intelligence and the powerful computational power leap made in the graphics processing units, they especially flourished due to the emergence of Generative Adversarial Networks (GAN) in 2014 that made their generation possible.

1.3 Types of Deepfakes

Deepfakes can be classified into two main categories: Visual deepfakes and audio deepfakes.

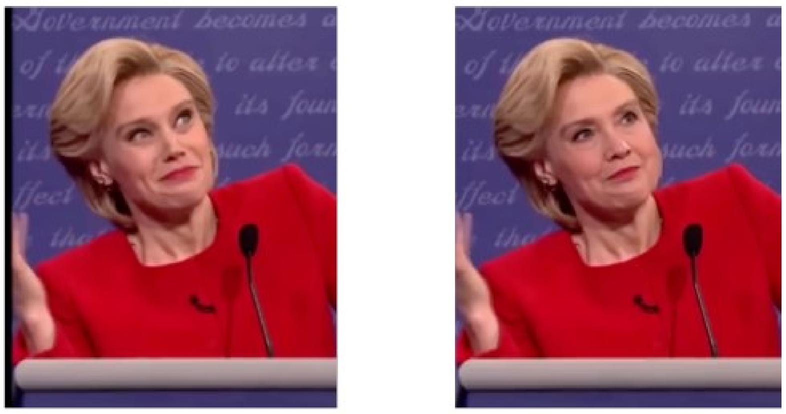


Figure 1.1: The difference between the real face (on the right) and the deepfake (on the left) [8].

1.3.1 Visual Deepfakes

These deepfakes include synthetic images and videos, and are the broader more invested in type of deepfake, this type includes a lot of deepfake type such as:

Entire Face Synthesis

This type of method generates non-existing faces usually using a powerful GAN such as style-GAN [14], style-GAN2 [15] and styleGAN2-Ada [16]. These models are trained on large public datasets of people to learn to mimic a person's facial expressions, mannerisms, voice and inflections. Moreover, these synthetic faces are becoming more and more realistic and advanced, mainly due to the investment to the development of GANs in recent years, what seemed impossible few years back is now achievable. These models utilize datasets such as Generated- Images [14] (100k StyleGAN), Faces [17] (100k-StyleGAN), DFFD [18] (100k-StyleGAN, 200k-ProGAN), and iFake-FaceDB [19] (250k-StyleGAN, 80k-ProGAN). This type of manipulation might be of good use to business such as video games and 3D modelling, but it also capable to get utilized for malicious use like opening fake account with realistic non-existing faces to spread fake information or to impersonation. Figure 1.2 shows an example of synthesised faces.

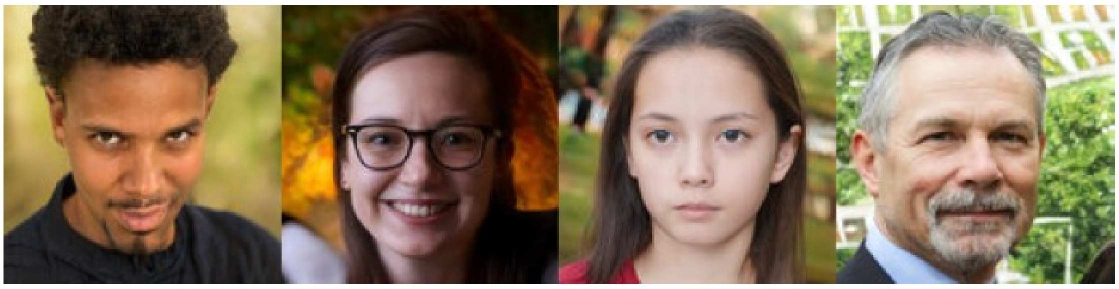


Figure 1.2: Examples of entire face synthesis [20].

Face Swapping

also known as identity swap, it is where the face of the source person is replaced with the face of the target person, creating a fake video or image of the target person doing actions the source person has done in reality. Such manipulations are divided into two main categories: 1) is the graphics-based such as FaceSwap [21] and 2) is the deep learning-based approaches such as DeepFakes [22]. These types pf deepfakes are designed for film industries and entertainment purposes where faces could be swapped, or to target the reputation and popularity of famous figures and celebrities by producing adult content. Figure 1.3 shows an example of swapping the faces of two people.

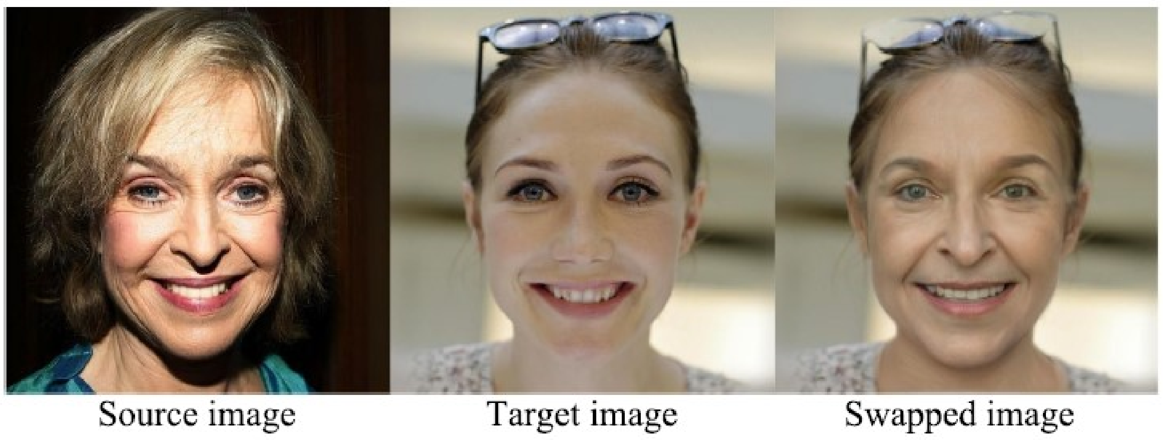


Figure 1.3: The face is swapped from left to right [20].

Attribute Manipulation

Also known as face editing or face retouching, it is all about changing aspects of the face, such as facial attributes like skin color, age, gender, eye colors, hair... or changing a person's sentiments to happy, sad, angry, afraid... Figure 1.4 depicts a perfect example of this manipulation with its two types. Such images are made through GANs such as the StarGAN [23], or applications like the AI face editor FaceApp [24].

Face Reenactment

Also known as expression swap, it is where the facial expressions of the target person are transferred into the face of the source person, where it could change the appearance of the last one, portraying him differently. Figure 1.5 shows an example of Swapping the facial expressions from the top to the results on the bottom. The manipulated media of such forgery could be images made by GAN architectures or videos made by the popular Face2Face [25] and Neural Textures [26].

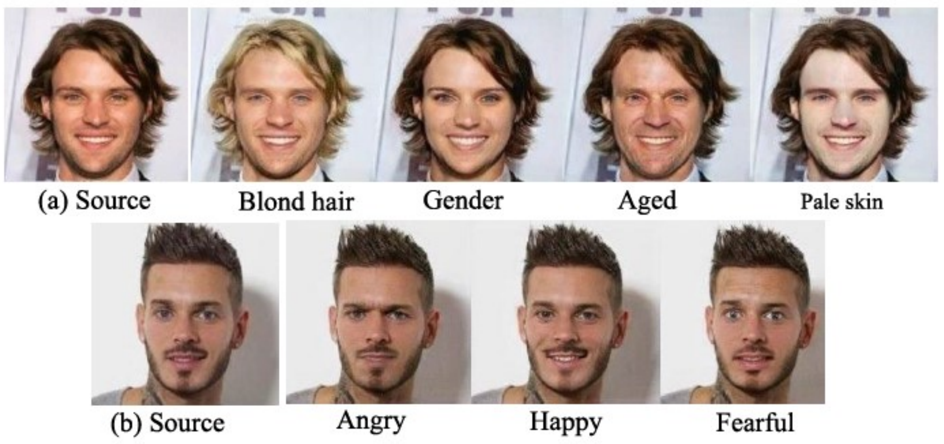


Figure 1.4: Examples of multiple attribute manipulations, from facial features on top to emotions on the bottom [20].

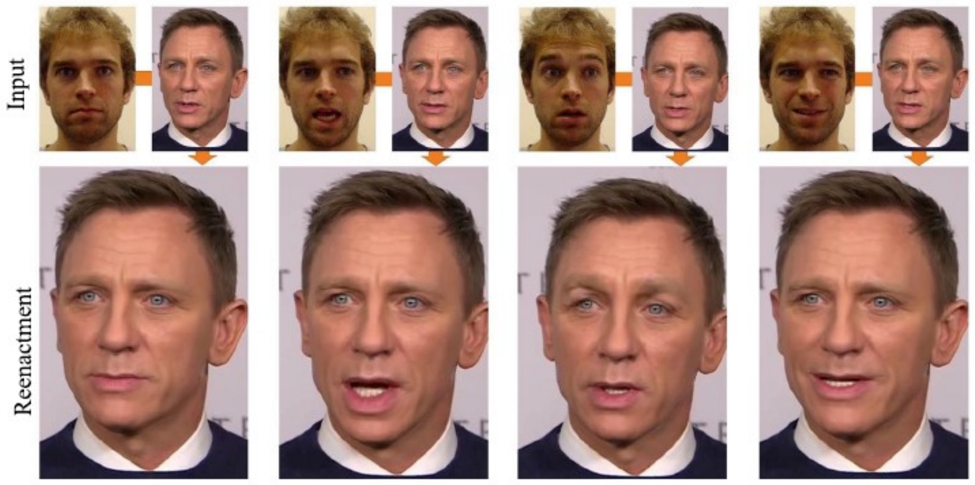


Figure 1.5: Face reenactment showing the target and source on top and the results on the bottom [20].

Face Morphing

It is a type of miscellaneous manipulation used for creating artificial biometric data samples that mimic the biometric data of multiple people, in other words, we can create a face that reassembles to two faces at the same time, such as in Figure 1.6, where the face in the middle does not exist, but is a morph created out of the two faces on the sides. This type of manipulation leads to correctly verifying the created morphed face images against a manipulated reference in a facial recognition system database if a morphed face image is stored as a reference. Hence, morphed face images constitute a significant threat to face recognition systems, as they contradict the core principle of biometrics, which is the unique link between the sample and its matching person. [20]

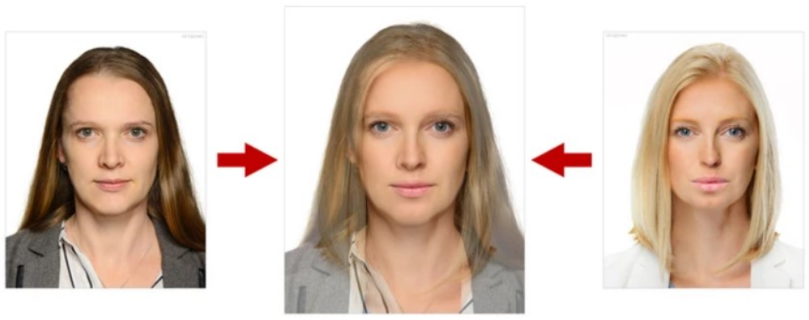


Figure 1.6: The morph at the middle is the result of the faces on the left and right [27].

Face De-identification

It is the process of altering facial images to protect the identity of the subject by obscuring or modifying its features while still preserving the overall characteristic for other purposes. Many techniques could be employed such as blurring or pixelation as demonstrated in Figure 1.7.

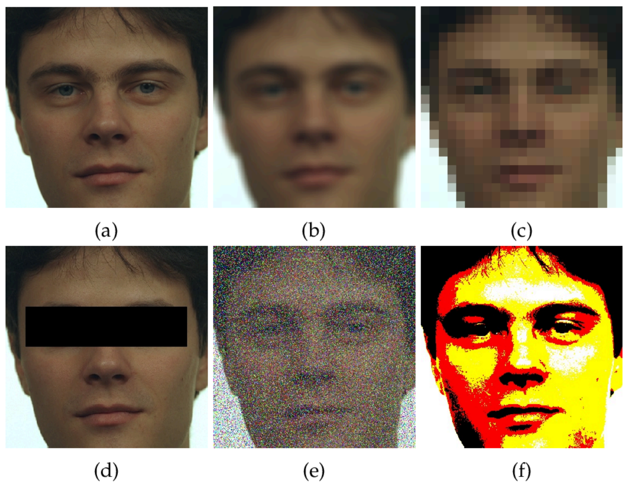


Figure 1.7: Five different techniques used for face de-identification to the source face in (a) [28].

Lip Syncing

As its name suggests, it synthesizes the expression of the target person in a video especially the lips, according to a given audio (audio-to-video) or a text (text-to-video), the source audio could also be synthesised like most cases, or real, making the target person saying sentences that he never did, in a well synchronized manner such as in Figure 1.8. The most famous example is [29] where a LipSync video of Obama broke the internet, marking the start of a new era of manipulation. This forgery is considered the most dangerous one, as audio-visual content is the most believable, and the possibilities are limitless.

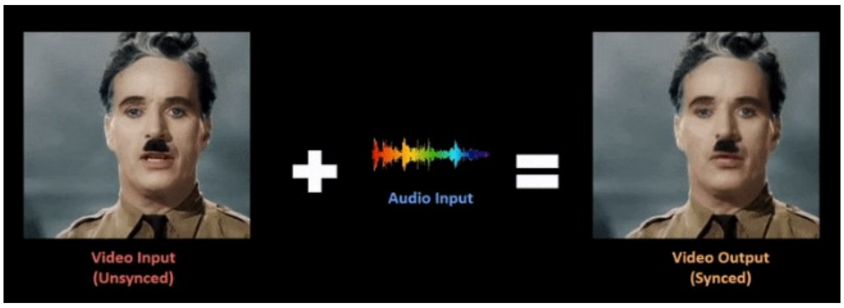


Figure 1.8: A simplified version of how LipSync works [7]

1.3.2 Audio Deepfakes

These types of deepfakes focus on generating the target's speaker voice using deep learning techniques to portray him saying what he never did. The fake audio can be generated using either text-to-speech synthesis (TTS) or voice cloning (VC). This type of deepfakes has become more sophisticated and realistic over the years, and the dangers only arise since it is an overlooked topic in deepfake detection, as much of the focus goes into the detection of videos and images (visual) deepfakes.

Synthetic voices are a big threat to automated speaker verification systems, voice-controlled systems deployed in IoT for example, and for misinformation, warfare and frauds since it enables cloning the voice of people of interest.

1.4 Evolution of Multimedia Manipulation

1.4.1 Manual Manipulations

The first ever recorded media manipulation dates back to 1860, after the 16th US president Abraham Lincoln got assassinated, one painter got creative to create a heroic-style portraits of the late president, where he superimposed his head into the body of southern politician John Calhoun, resulting into the infamous painting of Lincoln depicted in Figure 1.9. The painting also included the change of words below the hands from "strict constitution," "free trade," and "the sovereignty of the states" to "constitution," "union," and "proclamation of freedom". It was until 1969 that this painting was found to be a fake after an investigation was made.

In the 20th century, photography had a breakthrough, and more photographs were being taken for important figures, and well, also manipulated. The biggest example from the early 1900s was the soviet union, particularly Stalin's genius to use photo manipulation to his advantage, where many photos were altered such as in Figures 1.10 and 1.11. Stalin had the people opposed to him removed from historical photographs and events, which shows the power these manipulations yield even throughout history.

These 20th century manipulations revolved around the manual manipulation using brushes, color adjustments and other tools such as retouchers. More recent examples bring us to the 21st century, where the breakthrough of data and multimedia happened, adding to existing technologies and tools like Adobe Photoshop, made it easier to access

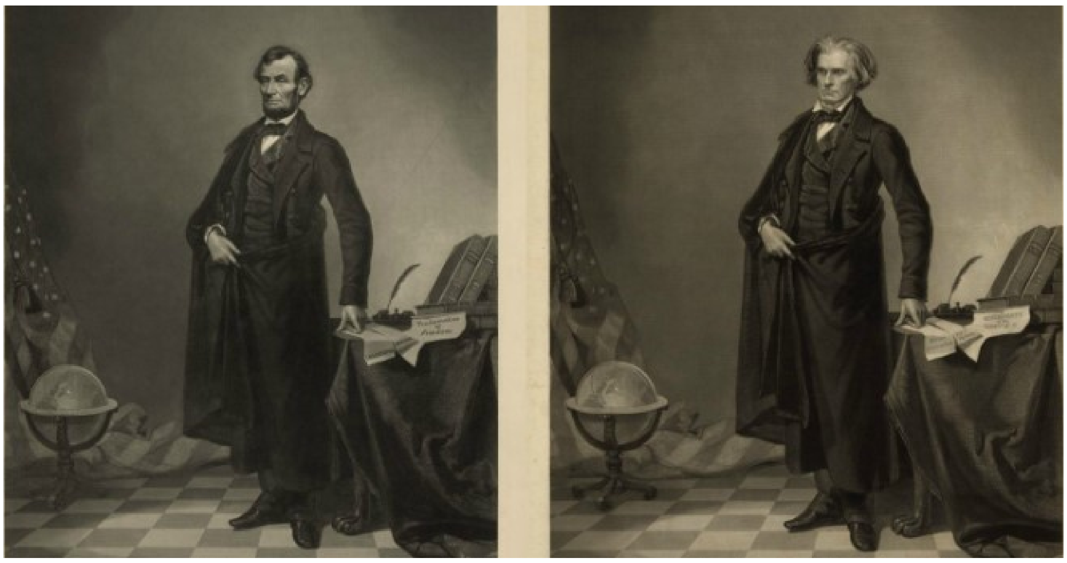


Figure 1.9: Abraham Lincoln manipulated image on the left, and the original image on the right [30]

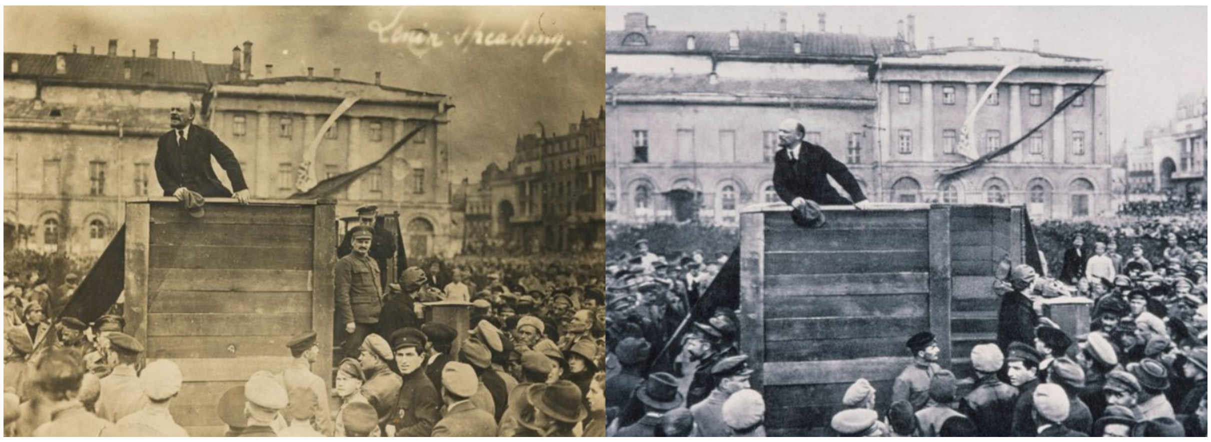


Figure 1.10: Original image on the left, and the manipulated image on the right showing the two men on the stairs have disappeared [31]

and alter multimedia content, making the end result believable for human perception as in Figure 1.12.

1.4.2 Automatic Manipulations

Aside from these traditional manipulation methods, advancements in Computer Graphics and Deep Learning (DL) techniques now offer a variety of different automated approaches for digital manipulation with better semantic consistency. The recent trend involves the synthesis of videos from scratch using auto-encoders, or Generative Adversarial Network (GAN) [33].

The first major leap made towards Deepfakes was the creation of GANs [4] back in 2014 which set the basis for what came after. An overview of deepfake history can be seen in Figure 1.13. An early notable academic project was video rewrite program [34] back

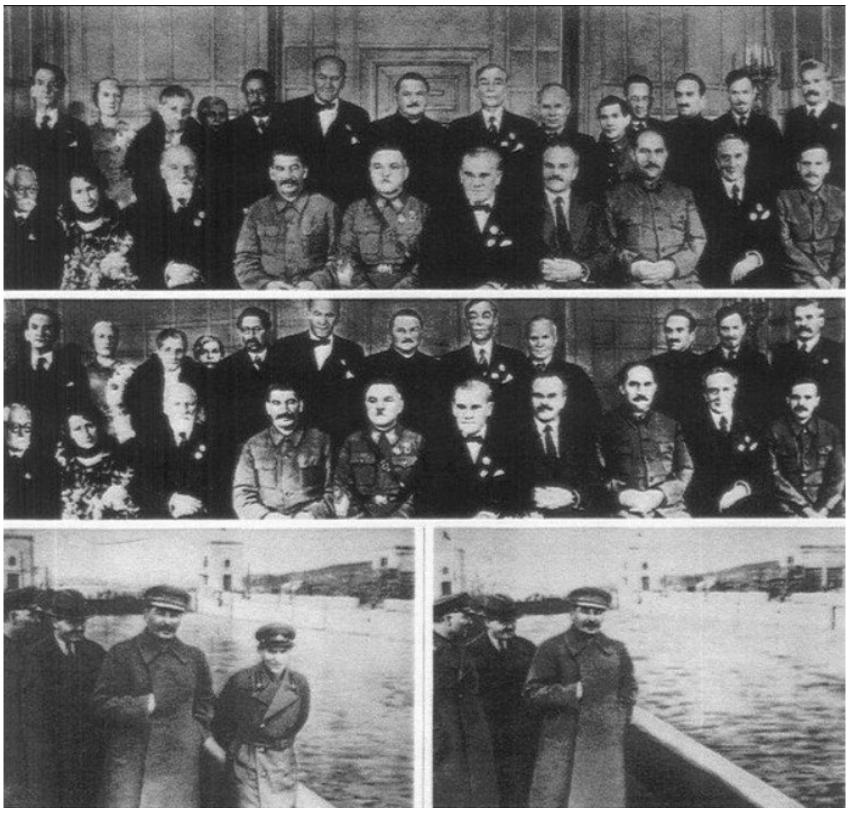


Figure 1.11: Two manipulation of mysteriously disappearing people [31]

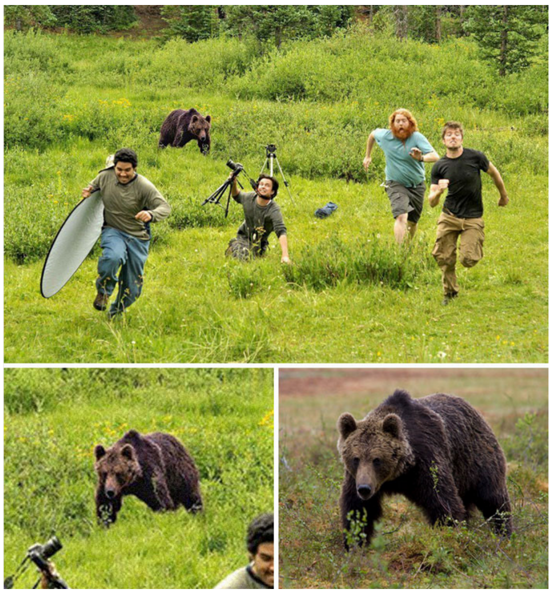


Figure 1.12: Photoshopped image taken from another frame (bottom right) to make the end result believable [32].

in 1997, it is the first software used to automatically reanimate facial movements in an existing video to a different audio track, and it achieved surprisingly convincing results. The first true deepfake as we know them today appeared back in 2017, when a Reddit user named ”deepfake” posted a series of computer generated videos of famous actresses swapped into adult content.

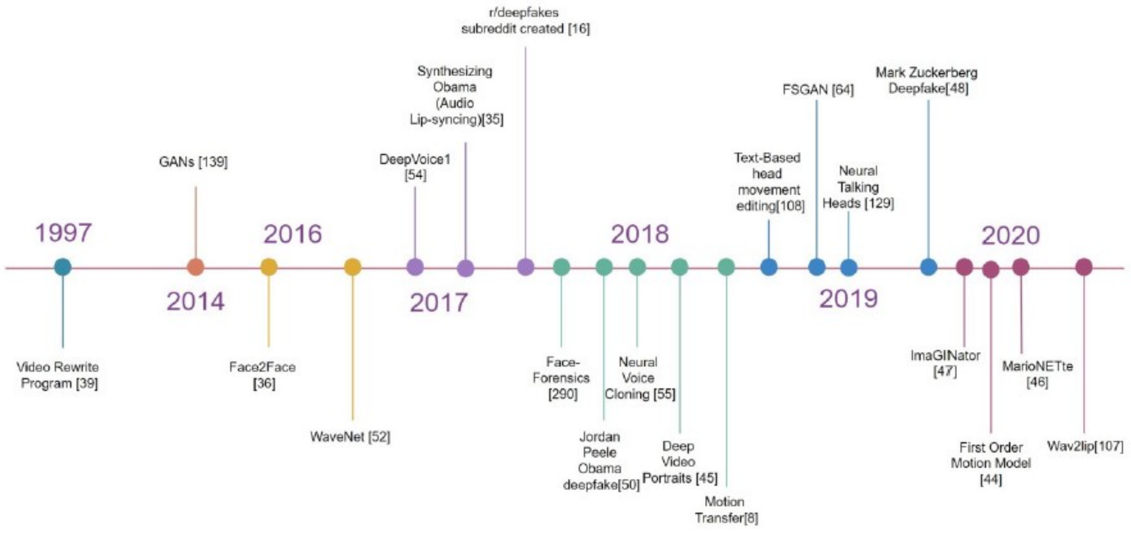

r/deepfakes subreddit created [16]
Synthesizing Obama (Audio Lip-syncing)[35]
GANs [139]
DeepVoice1 [54]
FSGAN [64]
Mark Zuckerberg Deepfake[48]
Text-Based head movement editing[108]
Neural Talking Heads [129]
1997
2016
2018
2020
2014
2017
2019
Video Rewrite Program [39]
Face2Face [36]
WaveNet [52]
Face-Forensics [290]
Neural Voice Cloning [55]
Jordan Peele Obama deepfake[50]
Deep Video Portraits [45]
Motion Transfer[8]
ImaGINator [47]
MarioNETte [46]
First Order Motion Model [44]
Wav2lip[107]

Figure 1.13: History of deepfakes [33]

Contemporary academic projects that led to the development of deepfake technology are Face2Face [25] and Synthesizing Obama [29], published in 2016 and 2017 respectively. Face2Face captures the real-time facial expressions of the source person as they talk into a commodity webcam. It modifies the target person's face in the original video to depict them, mimicking source facial expressions. Synthesizing Obama is a video rewrite 2.0 program, used to modify the mouth movement in the video footage of a person to depict the person saying the words contained in an arbitrary audio clip [33].

Today, deepfake applications and technology are easily accessible, such as FakeApp [24], FaceSwap [22], and ZAO, where ordinary users without an engineering background could be able to create fake videos, which led to the outbreak we see today. Moreover, Open source projects on Github such as DeepFaceLab [35] and Wav2Lip [7] make it even more possible to further develop this field.

Most deepfakes currently available on social platforms such as YouTube, Facebook, Instagram or Tiktok might seem harmless and fun. However, there are some examples of how deepfakes were used in a harmful manner, in 2018 a video went viral in which former U.S president Barack Obama appeared to be insulting his successor Donald Trump. In 2022, a TikTok account posting hyper-realistic videos of Tom Cruise went viral accumulating 1.4 million of views in just few days. Also in 2022, many deepfakes surfaced in the Russian-Ukrainian conflict, namely the surrender video of the Ukrainian president. Last year, also many deepfake videos and audios were produced by Israel to deceive the general public and to spread false information about the Palestinian Hamas.

1.5 Applications

Like any other technology developed through the course of history, deepfakes are no different in their impact according to their utilisation. Deepfakes could have a positive impact of businesses, industries and hobbyists. They also could have a negative impact on societies, industries, governments and famous public figures.

1.5.1 Beneficial Applications

Some of the most notable beneficial applications are:

Film Industry

They can for example help make digital voices of actors who lost theirs due to diseases, they can also update film footage instead of wasting time re-shooting it, they will also be able to recreate scenes from classical movies or star long-dead actors. Deepfake technology also allows for automatic and realistic voice dubbing for movies in any language. Face swapping in this case is the most beneficial as it enables the industry to include actors that were never present to shoot.

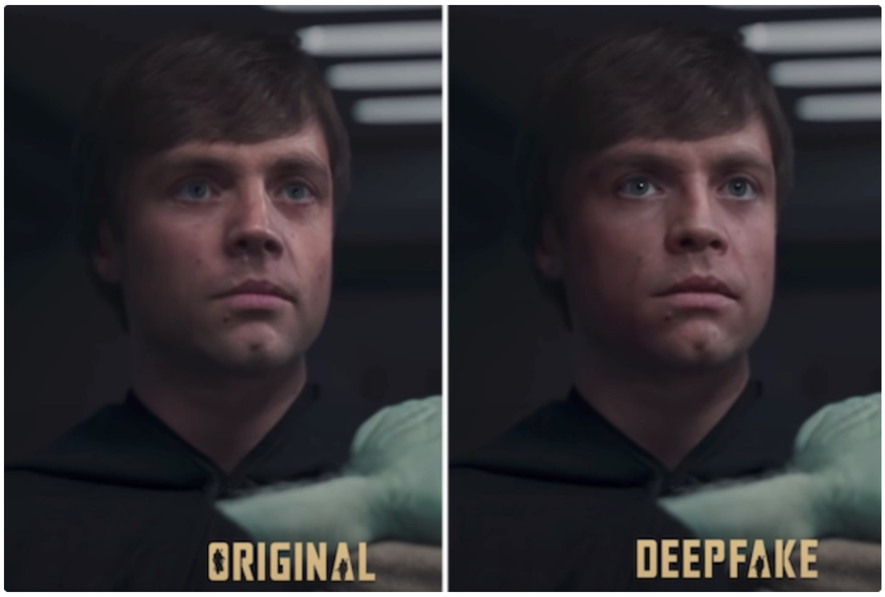


Figure 1.14: Example for a use in film-making [36]

Games and Entertainment

Enable multiplayer games and virtual chat worlds with increased telepresence, natural looking smart assistants, and digital doubles of people. Face synthesis is the most beneficial method augmenting the level of realism in video games with near-perfect non-existing faces.

Another example for entertainment is the generation of deepfakes by hobbyists and social media influencers to get views and to entertain the receivers on their platforms.

Business and Advertisement

Brands can contract supermodels who are not really supermodels, and show fashion outfits on a variety of models with different skin tones, heights, and weights. Further, deepfakes allow for super personal content that turns consumers themselves into models; the technology enables virtual fitting to preview how an outfit would look on them before purchasing and generate targeted fashion ads that vary depending on time, weather, and viewer [2]

1.5.2 Threatening Applications

One of the main, if not the main threat of deepfakes are misinformation and disinformation. Misinformation is defined as false or inaccurate information that is communicated, regardless of an intention to deceive, whereas disinformation is the set of strategies employed by influential society representatives to fabricate original information to achieve the planned political or financial objectives.

Deepfakes are expected to become the main source of intentionally spreading manipulated news and false information to intentionally affect the general public' opinion and to obscure reality. With the rapid spread of information in today's world due to the extensive use of social media, and the high consumption of content daily per user, deepfakes are destined to become harder to recognise, and more dangerous especially for ignorant people, and as a matter of fact, to everyone on the first encounter, the consequences are expected to be much more devastating.

Historically, deepfakes were made to make famous personalities and celebrities controversial among their fans and followers, such as the 2017 subreddit posts that made the target celebrities face controversy after being involved without their consent. This only confirms that deepfakes can be used to damage reputations and defame people, and the impact would be much greater for higher people of interest such as politicians, presidents or even billionaires.

Deepfakes could also be used for blackmailing people for monetary benefits, or even frauds and scams such as in 2019, where a cyber attack was orchestrated by the help of audio deepfakes to formulate a financial scam on a UK energy company, where they lost 243.000$ [37]. A voice-mimicking AI software was used to clone the voice patterns of the victim by training Machine Learning (ML) algorithms using audio recordings obtained from the internet. The damage of such technology would be more catastrophic if the concerned person was a military leader or a top government official.

Deepfakes are not limited to damage individuals, they could rather be used to manipulate elections, spread hate between groups, create political or religious unrest, and could even create warmongering situations by showing missiles being launched at enemies, which could all be resulting in conflicts and unprecedented consequences could follow.

Last but certainly not least, the recent events happening in the world, where this technology is used as a war crime against opponents. The major two examples are the:

- Russian-Ukrainian conflict, where many deepfakes were made by both parties to target high ranking officials, soldiers and many more, to obscure the reality and persuade the public to their cause, the most famous one is the LipSync video made picturing the Ukrainian president publicly ordering his troops to surrender [38], which has had a psychological impact on the soldiers and people before reality was revealed.

- In the Israel-Gaza war, where many deepfake images, videos and audios surfaced on the internet, making it harder to believe what's real from what is not.

1.6 Conclusion

In conclusion, we have defined deepfakes most thoroughly, while giving the most useful and famous types of deepfakes from audio deepfakes to visual deepfakes such as face swapping, entire face synthesis and face reenactment, with the latter being our interest throughout the study. Then, we successfully gave insights on the history and development of deepfakes, before detailing its applications in various fields and ways, especially in a negative way which was alarming enough to start the race between the two fields of deepfake generation and detection.

Chapter 2

Deepfakes, from Generation to Detection

2.1 Introduction

This chapter is dedicated solely to theoretical deepfake generation and detection, where we describe both fields neatly, as the understanding of the bases leads into a better understanding of the following chapters.

We first look at the deepfake generation methods and techniques namely face swapping, face reenactment and LipSync, the technical background in Generative Adversarial Networks and Auto-encoders, and finally the challenges and limitations it faces to further improve the technology.

Finally, we present deepfake detection, its technical background in Convolutional Neural Networks and Transformers, the most commonly used backbones such as: Xception and EfficientNet, then a thorough classification of existing detectors will be made to help understand the upcoming chapters, and finally the limitations and challenges it faces to keep up in the race with the generation field.

This chapter helps understand the theoretical background of deepfakes, setting the base for the upcoming chapters, it also enables us to understand the importance to detect this emerging technology, as it is the hot topic of today's world.

2.2 Deepfake Generation

Deepfake generation refers to the computer science field that has a goal of generating synthetic media, whether it is images, videos and audios. This field uses Generative Adversarial Networks (GANs) and Auto Encoder (AEs) to produce hyper-realistic and convincing forgeries.

2.2.1 Technical Background

The two pillars of deepfake generation are GANs [4] and AEs [3] with their many variants, in what comes next we will detail their basic architectures.

Generative Adversarial Networks

GANs are a revolutionary tool used to train generative models to generate realistic samples to mimic a data distribution. GANs are the combination of two neural networks: The Generator (G) that aims to create fake samples to replicate the data distribution, and the Discriminator (D) that evaluates the samples to distinguish between real data from the distribution and fake data generated by G, these two work in a minimax game, i.e. G attempts to fool D, while D tries to identify whether the data is real or fake. Over time, both networks improve their performance, the generator becomes so good at creating realistic samples that the discriminator fails to detect them. The basic structure of a GAN can be modeled as in Figure 2.1.

The mathematical minmax optimisation of G^* of G and D is as follows :

$$
\begin{aligned}
G^* &\in \arg\min_G \max_D V(G, D) && (2.1)\\
&= \arg\min\max \left(\mathbb{E}_{X\sim P_{data}(X)}[\log(D(X))] + \mathbb{E}_{Z\sim P_Z(Z)}[1 - \log(D(G(Z)))]\right) && (2.2)
\end{aligned}
$$

Where G^* is the optimal generator, Z is the input for the generator G(Z) with distribution probability P_Z, and returns X with a distribution probability P_G. The discriminator D(X) estimates the probability that X is from the distribution of training data P_{data}.

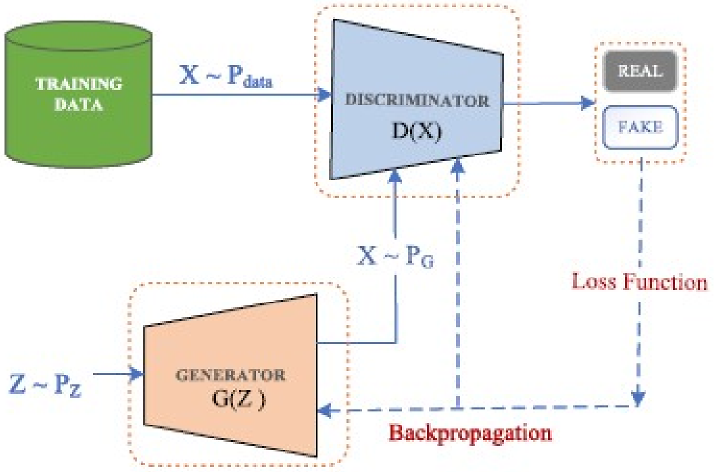


Figure 2.1: Basic GAN architecture [20].

Auto-Encoders

The core idea behind this method is the parallel training of two auto-encoders. Traditionally, the auto-encoder consists of two parts, an encoder network that performs a dimension reduction by encoding the data from the input layer into a reduced number of variables, and a decoder network that uses those variables to generate an approximation of the original input. The optimization phase is done by penalizing the difference between the input and the approximation.

The process of deepfake generation is to first train two auto-encoders E_A and E_B to reconstruct the faces of datasets A and B respectively. The idea is to share the weights of the encoding part, while also keeping the decoders separated. Secondly, an image containing the face of A can be encoded and then decoded using the decoder E_B as shown in Figure 2.2.

The intuition is to have an encoder that focuses on general information such as illumination, position and expression of the face, and a dedicated decoder for each person to reconstitute constant characteristic shapes and details of the person face.

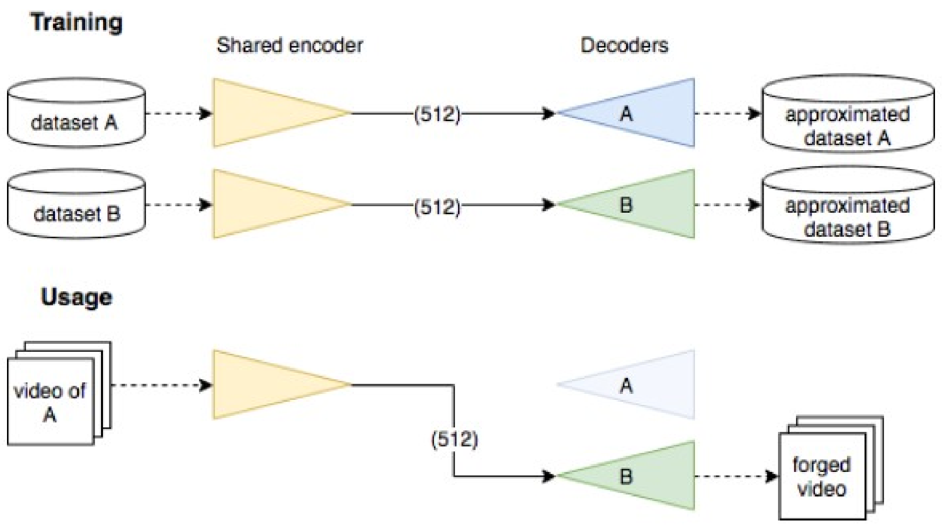


Figure 2.2: Basic Auto-Encoders architecture [8].

2.2.2 Generation Methods

[1]

Deepfakes since their appearance in 2017 have come in many ways, as many generation methods have been developed to make this synthetic data realistic. In our study, we will only be looking at visual deepfakes, to stay consistent with the deepfake detection task at our hand. Therefore, the generation methods included are:

Face Swapping

This technique embeds a source face into a target face, it has been produced by many methods over the years such as:

- Face Swap [21] and deepfakes [22]: Based on auto-encoders, they have multiple limitations such as the requirement for massive amounts of target images and blurry results.
- DeepFaceLab (2020) [39]:Based on auto-encoders, one of its main limitations is the inability to blend very different facial hues.
- Face Shifter (2020) [40]: Based on GANs, the main limitation is the poor ability to preserve face feature attributes.
- FALCO (2023) [41]: Based on GANs, its limitation is the poor ability to handle facial occlusion.

[1]For examples and more details about the following methods, check 1.3.1

Face Reenactment

This techniques blends the facial expression of a source face to a target face, some of the most prominent methods are:

- Face2Face (2016) [25]: uses 3D facial reconstruction, and affine transformation to transfer the facial expressions, it is sensitive to facial occlusions.
- HeadGAN (2021) [42]: It uses 3DMM for facial model to provide a 3D prior to the GAN.

Lip Syncing

This technique synthesizes the talker' lip to match a generated or given audio, some of the most prominent methods are: Wav2Lip (2020) [7] that has the limitation of poor resolution, TalkLip (2023) [43] which is unable to control pose and emotional variations, and GeneFace (2023) [44] that has to be applied on a cropped facial region.

2.2.3 Challenges of Deepfake Generation

Paired Training

Data pairing is important to generate high quality content, it is the process in which closely similar input samples are identified for training. This process is computationally costly.

Pose Variations and Distance from Cameras

Today' methods generate good results for frontal facial views, with the quality degrading when the person is not looking at the camera or is distant from it.

Temporal Coherence

A drawback to generation methods is the presence of temporal artefacts such as flickering and jitter between frames, which happens when a model works on frames without considering the temporal consistency.

Lack of Realism in Synthetic Audio and LipSync

Though the quality has improved, the main challenges are the lack of natural emotions, pauses, breathing, and the talking pace.

Occlusions

Which causes a problem to all computer vision tasks, it refers to regions of the faces being obscured by a hand, glasses or any other item.

2.3 Deepfake Detection

Deepfake images and videos detection dominate the research on monitoring multimedia manipulation, an emerging field that has the right intention of detecting such manipulations to improve confidentiality and preserve the integrity of content. However, it is not an easy task to detect such altered content, especially with the emergence of generative networks [4] and their improvement over the last years.

Forgery detection, or deepfake detection, refers to the field of computer science that has a goal of analysing the multimedia content, whether it is images, videos or audios, to determine whether or not the generated multimedia has been tampered with or is original [20]. Deepfake detection has gone past different eras, from the traditional data-driven CNNs, to the detection of specific patterns present in manipulated media, to the task of generalisation to unseen forgeries.

2.3.1 Technical Background

Most deepfake detectors rely on famous frames to construct their pipeline accordingly with their need and method, here we will be detailing the two most commonly used ones.

Convolutional Neural Networks

The CNN or ConvNet [45] is a special kind of deep-learning architecture that has a basic structure of 3 layers: Convolutional, pooling and fully connected as it can be seen in Figure 2.3.

- **Convolutional layer**: Its purpose is to perform feature extraction for the input tensor, in order to do that, a series of kernels (which is an array of numbers) are convoluted with the input tensor to obtain an arbitrary number of feature maps that is the same as the number of kernels. To construct a feature map, an element wise product between each element of the kernel and the elements of the input tensor, the results are then summed to obtain one element of the feature map according to the same element from the input. The convolution operation is done during the forward propagation, during the back propagation, the gradient descent optimisation is used to update learnable parameters (weights and kernels) according to the loss value. The feature value $Z^l_{i,j,k}$ at the position (i,j) at the K-th feature map of the l-th layer follows Equation 2.3:
$$Z^l_{i,j,k} = (W^l_k)^T x^l_{i,j} + b^l_k \tag{2.3}$$
Where W^l_k and b^l_k are the weight vector and bias term of the k-th filter of the l-th layer , respectively. $x^l_{i,j}$ is the input patch centered at location (i,j) of the l-th layer. Then, a nonlinear activation function is applied to detect nonlinear features such as sigmoid, tanh and ReLU. A nonlinear activation function A(.) can be expressed as:
$$a^l_{i,j,k} = A(Z^l_{i,j}) \tag{2.4}$$
where $a^l_{i,j,k}$ is the output after applying the non linear activation function.

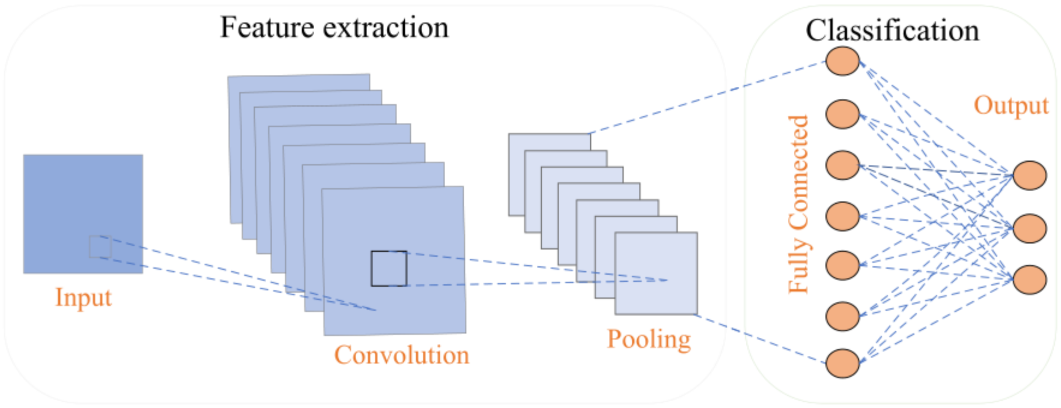


Figure 2.3: Basic CNN architecture [20].

- **pooling layer**: Provides a typical down-sampling operation to reduce the dimension of the feature maps to introduce translation invariance to small shifts and distortions and thereby decrease the number of subsequent learnable parameters. For each feature map we have:

  $$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall (m,n) \in R_{i,j} \tag{2.5}$$

  Where $R_{i,j}$ is the neighborhood around the location (i,j).

- **Fully connected layer**: The final output of the CNN, where the decision about the input is made according to the features extracted earlier, the number of output nodes corresponds to the number of classes, for binary classification we have only one. A non linear activation function follows each fully connected layer, and finally a learning loss function is calculated to assess the results in comparison with the ground truth labels.

  Training a CNN aims at finding the global minima, i.e. the set of parameters to minimise the loss function. Many famous CNN architectures exist such as VGGNet [46], GoogleNet/Inception [47], and ResNet [48].

Transformers

Transformers are a type of deep learning networks introduced by Vaswani et al. [49] in 2017. It has since revolutionized natural language processing, and is now being applied to various new fields, in our context it is deepfake detection. Transformers follows an encoder decoder network using only self-attention without the need for convolutions or recurrences as illustrated in Figure 2.4. The main Modules of transformers are:

- **Input embedding**: Used to represent inputs such as words into continuous representations (eg vectors) of a dimension d_{model}, this is done by a simple linear layer. It does not only serve as a compatible representation, but could also model semantic similarity between words (or tokens).

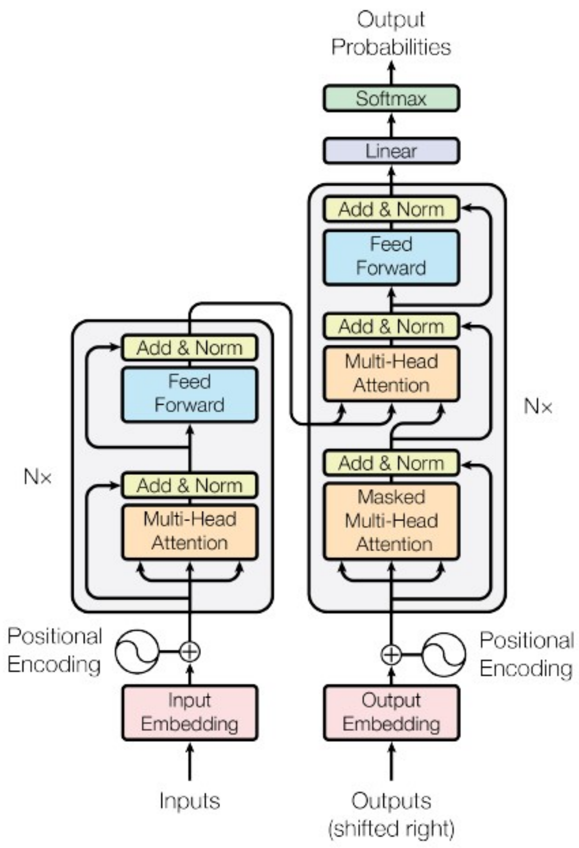


Figure 2.4: Transformer architecture [49].

- **Positional encoding**: Used to model the position of a token in a sequence, they have the same dimensions of the embeddings to facilitate the sum, The formula for positional encoding is :

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{2.6}$$

$$\text{PE}(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{2.7}$$

  Where pos is the position and i is the dimension.

- **Scaled-dot product attention**: Uses queries Q (determines what information to look for), keys K (identifies the relevance of parts of input sequence to the query) and values V (provides the information that is aggregated based on the relevance scores (attention weights)) following the pipeline of Figure 2.5 on the left. The Equation it follows is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.8}$$

  The scaling by $\sqrt{d_k}$ is done to avoid high magnitudes.

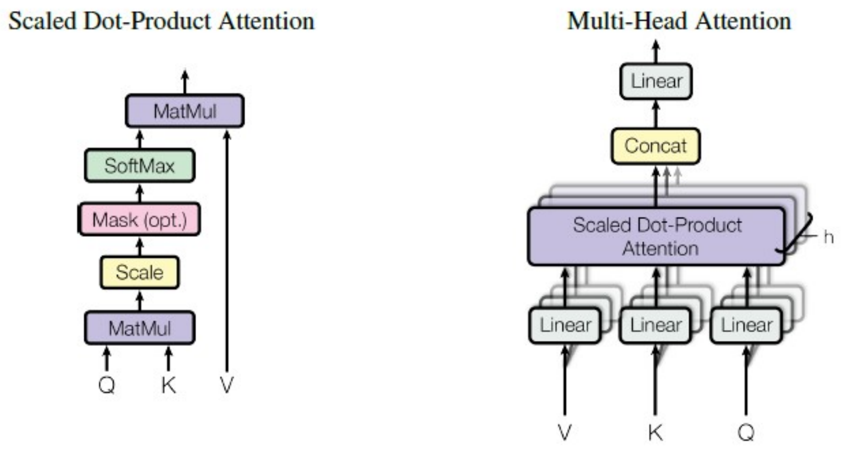


Figure 2.5: Scale-dot product attention and multi-head attention architectures [49].

- **Multi-head attention**: Instead of performing a single attention, queries, keys and values are linearly projected using weights W_i, on each of the projections the attention is performed in parallel, then all the results are concatenated together as explained in Equations 2.9 and 2.10 and illustrated in Figure 2.5 on the right.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{2.9}$$

  Where each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.10}$$

  Where W_i are projection matrices.

- **Self-attention**: Using the previous attention mechanisms, it helps weigh the importance (relationship) of tokens within an input or output sentence.

- **Encoder-decoder attention**: Helps keep track of the relationship between input and output sentences to make sure important tokens from the input are not lost in translation.

In the context of deepfake detection, Transformers could be very useful due to their feature extraction and sequence modeling abilities. In fact, they could be used in temporal detectors to detect temporal inconsistencies between frames, they could also serve in the multi-modal analysis by integrating information from audio and visual sources, on top of the attention mechanism of transformers that could be used to focus on the most relevant forgery patterns.

2.3.2 Backbones

Most of the CNN-based detectors are constructed around famous architectures, called backbones, in the following we will detail the three most used backbones in deepfake detection: Xception [50], EfficientNet [51], and ResNet [48].

Xception

After the classic CNN architecture such as VGGNet [46], a new style emerged with the creation of the Inception architecture in GoogleNet [47] that was the top performer on ImageNet [52] after its introduction for quite some time. What made this architecture different and special is the inception module that could be seen in Figure 2.6.

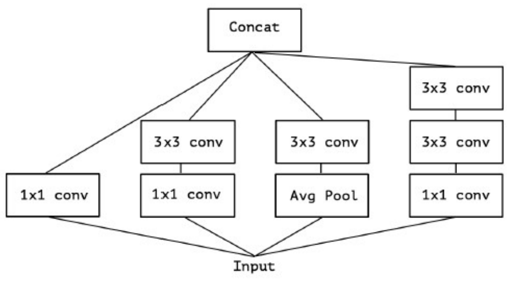


Figure 2.6: A simplified inception module architecture [50].

The inception module looks at cross-channel correlations via a set of 1x1 convolutions, mapping the input data into 3 or 4 separate spaces that are smaller than the original input space, and then maps these smaller spaces via regular 3x3 and 5x5 convolutions. In effect, the fundamental hypothesis behind Inception is that cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly.

Depthwise separable convolutions are an operation that appeared in 2014, they are spatial convolutions performed on each channel independently, followed by a pointwise convolution, i.e. 1x1 convolutions projecting the channels output by the depthwise convolution onto a new channel space, without including a non-linearity such as ReLu between the two, on the contrary of the extreme inception module.

Based on previous hypothesis and results, Chollet proposed the Xception architecture in [50] that is based entirely on depthwise separable convolutions as it may be seen in Figure 2.7, counting for 36 convolutional layers.

EfficientNet

This work introduced compound scaling to achieve better accuracy and results, it consists of up-scaling while still balancing the dimensions of the network i.e. height, width and resolution, which can be achieved by scaling each of them with a constant factor. An illustration of this method comparatively with other previous ones is in Figure 2.8. Intuitively, the compound scaling method makes sense because if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.[51].

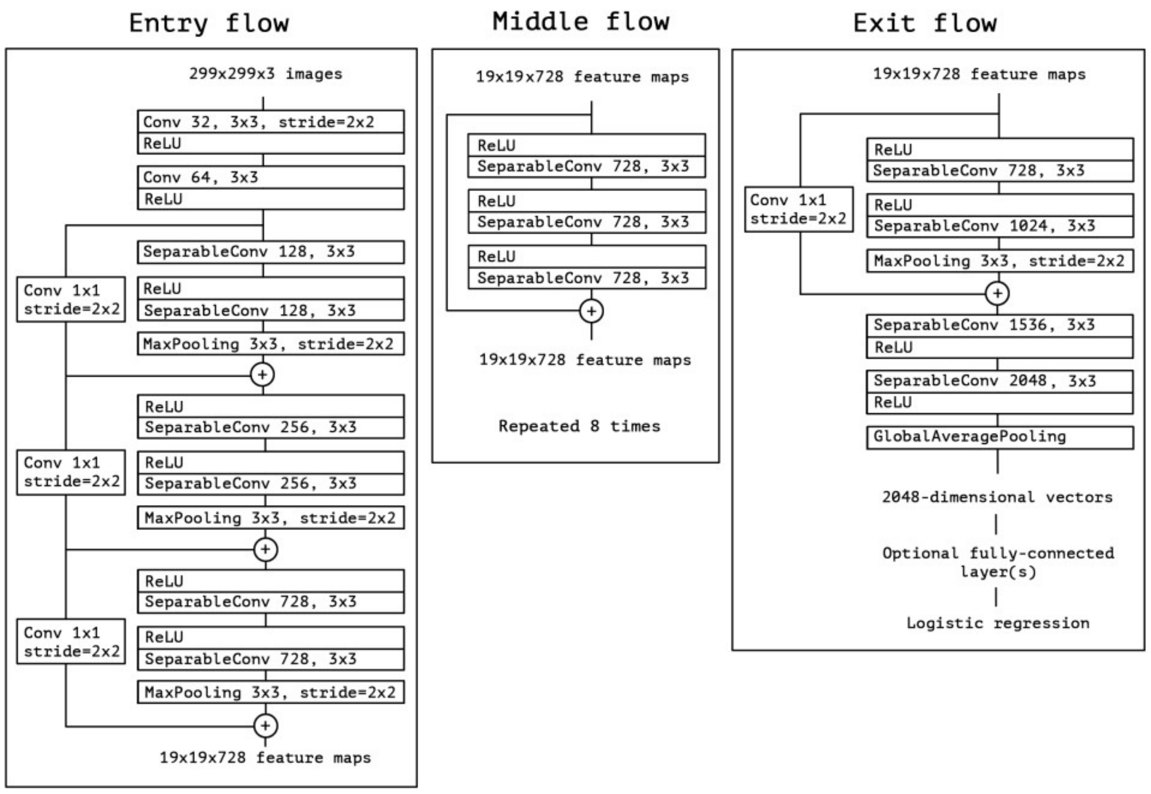


Figure 2.7: Xception detailed architecture [50].

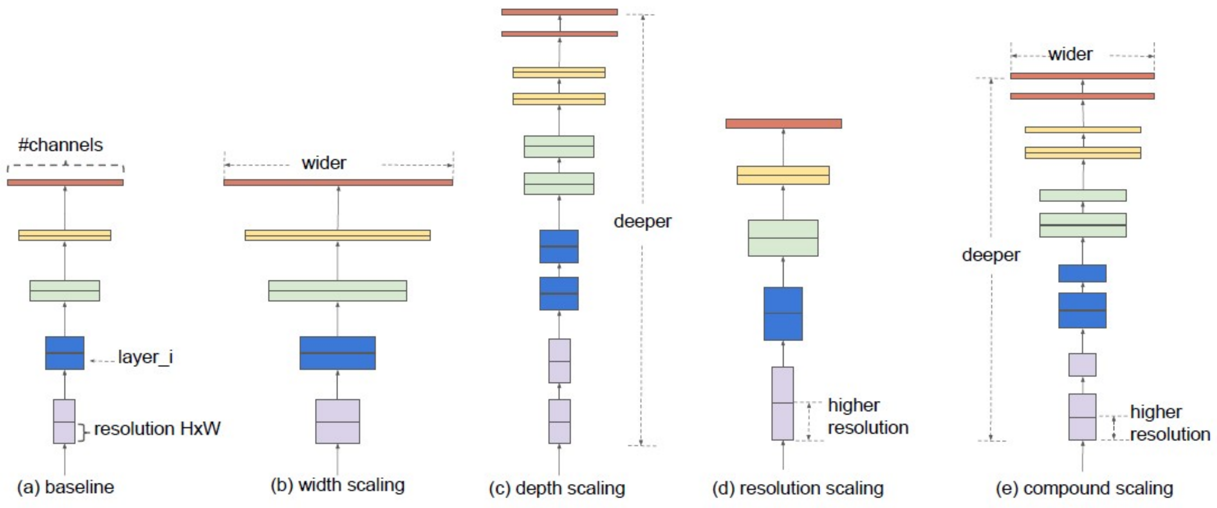


Figure 2.8: Comparison of different model scaling methods [51].

A CNN layer i is defined as $Y_i = F_i(X_i)$, where F_i is the operation, Y_i is the output tensor and X_i is the input tensor, with tensor shape $\langle H_i, W_i, C_i \rangle$, where H_i and $W_i t$ are spatial dimension, and C_i is the channel dimension. A CNN can be represented by a list of composed layer such as $N = F_k \odot ... \odot F_2 \odot F_1(X_1) = \odot_{j=1...k} F_j(X1)$. Usually, CNNs have multiple stages, each one has the same architecture for its layers, therefore a CNN is defined as :

$$\aleph = \odot_{i=1...s} F_i^{L_i}(X_{\langle H_i, W_i, C_i \rangle}) \tag{2.11}$$

Where $F_i^{L_i}$ denotes layer F_i is repeated L_i times at stage i. Figure 2.8(a) illustrates that the spatial dimension is gradually shrunk but the channel dimension is expanded over layers.

Unlike regular CNNs that try to find the best F_i, model scaling fixes the last one and focuses on expanding the dimensions (length L_i, width C_i, and resolution (H_i,W_i)), this approach simplifies the design problem for new resource constraints, to reduce design space as well, they scaled all layers uniformly with a constant ratio, the target was to maximize the accuracy for any given constraint, which can be formulated as follows:

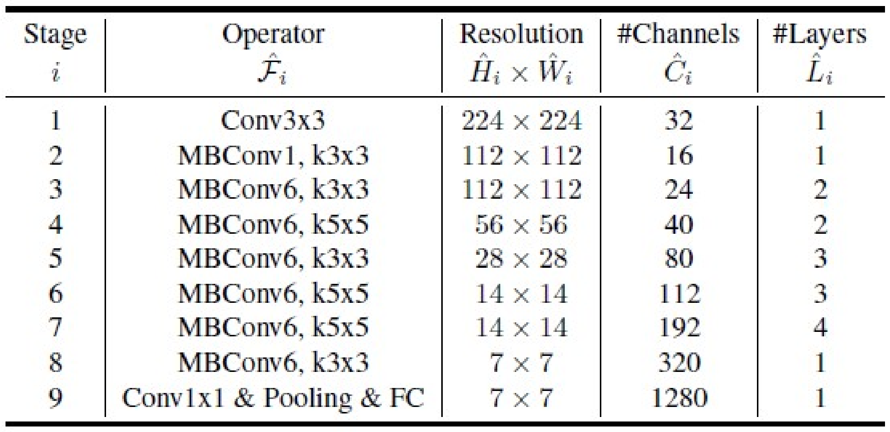
$$max_{d,w,r} Accuracy(\aleph(d,w,r)) \tag{2.12}$$

$$s.t.\aleph(d,w,r) = \odot_{i=1...s} \hat{F}_i^{d.\hat{L}_i}(X_{\langle r.\hat{H}_i, r.\hat{W}_i, w.\hat{C}_i \rangle}) \tag{2.13}$$

$$Memory(\aleph) \leq targetmemory \tag{2.14}$$

$$FLOPS(\aleph) \leq targetFLOPS \tag{2.15}$$

Where d,w,r are coefficient for scaling the network's depth, width and resolution. $\hat{F}_i$, $\hat{L}_i$, $\hat{H}_i$, $\hat{W}_i$, $\hat{C}_i$ are predefined parameters in Figure 2.9.



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 2.9: EfficientNet parameters and baseline architecture [51].

The study done by the authors of EfficientNet in [51] showed 2 key observations:

- **Observation 1**: Scaling up any dimension of network width (for smaller networks to capture more fine-grained features), depth (to capture more complex features and generalise better), or resolution (higher resolution images lead to capturing more fine-grained patterns) improves accuracy, but the accuracy gain diminishes for bigger models as it hits a plateau.

- **Observation 2**: In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during CNN scaling.

Based on the previous findings, [51] proposed a new compound scaling method formulated as follows:

$$depth : d = \alpha^{\phi}; width : w = \beta^{\phi}; resolution : r = \gamma^{\phi} \tag{2.16}$$

$$\alpha.\beta^2.\gamma^2 \approx 2; \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \tag{2.17}$$

Where ϕ is a user specific coefficient that controls how many more resources are available for model scaling, while α, β and γ control how to assign these extra resources. The Floating-point operations (FLOPS) are proportional to d, w^2 and r^2, hence the second equation keeps the FLOPS proportional to 2^{ϕ}.

On top of the model scaling, the baseline network was built to optimize both accuracy and FLOPs, using $ACC(m)x[FLOPS(m)/T]^w$ as the optimisation goal, where T is the target FLOPS and w=-0.07 a hyper-parameter. as shown in Figure 2.9, the EfficientNet-B0 is the baseline, with its main building block being the mobile inverted bottleneck MBConv to which they added squeeze and excitation optimisation.

Starting from EfficientNet-B0, the steps to up-scale are:

- STEP 1: $\phi = 1$, based on previous equations, the best values for constants are : α=1.2, β=1.1, γ=1.15.
- STEP 2: α, β, γ are fixed and the up-scaling is done with different ϕ values.

ResNet

Deep neural networks proved to be very effective in computer vision tasks such as image classification and object detection throughout the years, nevertheless, they also suffered greatly from the notorious vanishing gradient, which hamper convergence from the beginning, a problem that was addressed by a lot of means such as intermediate normalisation layers. Even when the models start converging, a new problem emerged which is the degradation, meaning that accuracy gets saturated then starts to degrade quickly, a problem that is not caused by overfitting, and could not be addressed by adding more layers due to the increase of training error, indicating that not all systems are easy to optimize.

To address the degradation problem, ResNet, short for deep residual learning network was introduced, where Instead of hoping each few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping [48], under the hypothesis that it is easier for networks to learn to adjust the input slightly than it is to learn a complex transformation that perfectly maps the input to the desired output.

Let H(x) be the underlying mapping that the stacked layers aim to fit, i.e.the ideal output to an input x, instead of letting the stacked layer approximate H(x), they let them approximate the residual function F(x) = H(x) - x, which is essentially the adjustment that needs to be made to the input x to approximate the desired output H(x). In other words, instead of learning the desired mapping H(x), the network tries to learn the residual mapping F(x), which is then added to the input to obtain the final output, which means H(x) is recast into F(x)+x.

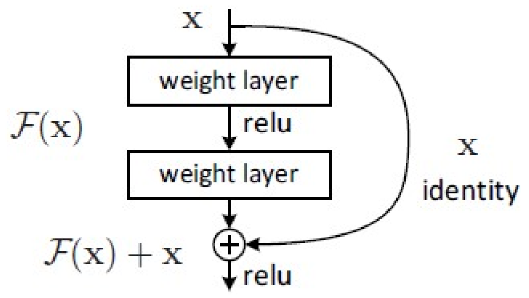

x
weight layer
$\mathcal{F}(\mathrm{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathrm{x}) + \mathrm{x}$
relu

Figure 2.10: Residual network building bloc [48].

Residual learning is adopted to every few stacked layer, Figure 2.10 show a building bloc, where identity mapping is made possible by shortcut connections, the problem can be defined as follows:

$$y = F(x, W_i) + x \tag{2.18}$$

Where Y and x denote the output and input vectors of the layers, $F(x, W_i)$ is the residual mapping to be learned, for the example of Figure 2.10, $F = W_2\sigma(W_1x)$, in which σ is ReLu. The operation F + x is performed by a shortcut connection and element-wise addition. The second nonlinearity is adopted after the addition. The shortcut connections add no additional parameters, not complexity in terms of computation, and they are an important factor for the comparison between plain and residual network to justify the effectiveness of the method.

The dimensions of x and F must be the same in the last equation, in case if it is not, a linear projection W_s is performed by the shortcut connection to match the dimensions, the formula becomes:

$$y = F(x, W_i) + W_s x \tag{2.19}$$

The plain network showed in Figure 2.11 in the middle is inspired by VGG nets, The convolutional layers mostly have 3×3 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. The network ends with a global average pooling layer and a 1000-way fully-connected layer with Softmax. The total number of weighted layers is 34 [48]. The network has lower complexity than VGGNet with 3.6 billion FLOPS, 18% of the 19.6 billion FLOPS of VGG-19.

Based on the previous plain network, the residual network is built by inserting shortcut connections, when having the same dimensions the identity shortcuts are used directly (solid lines), while a dimension adjustment is made when changing dimensions (dotted lines), the latter is done in two ways, either by zero padding the input x, or by using the Equation 2.19.

The architectures for the plain network and the residual network ResNet-34 are shown in Figure 2.11.

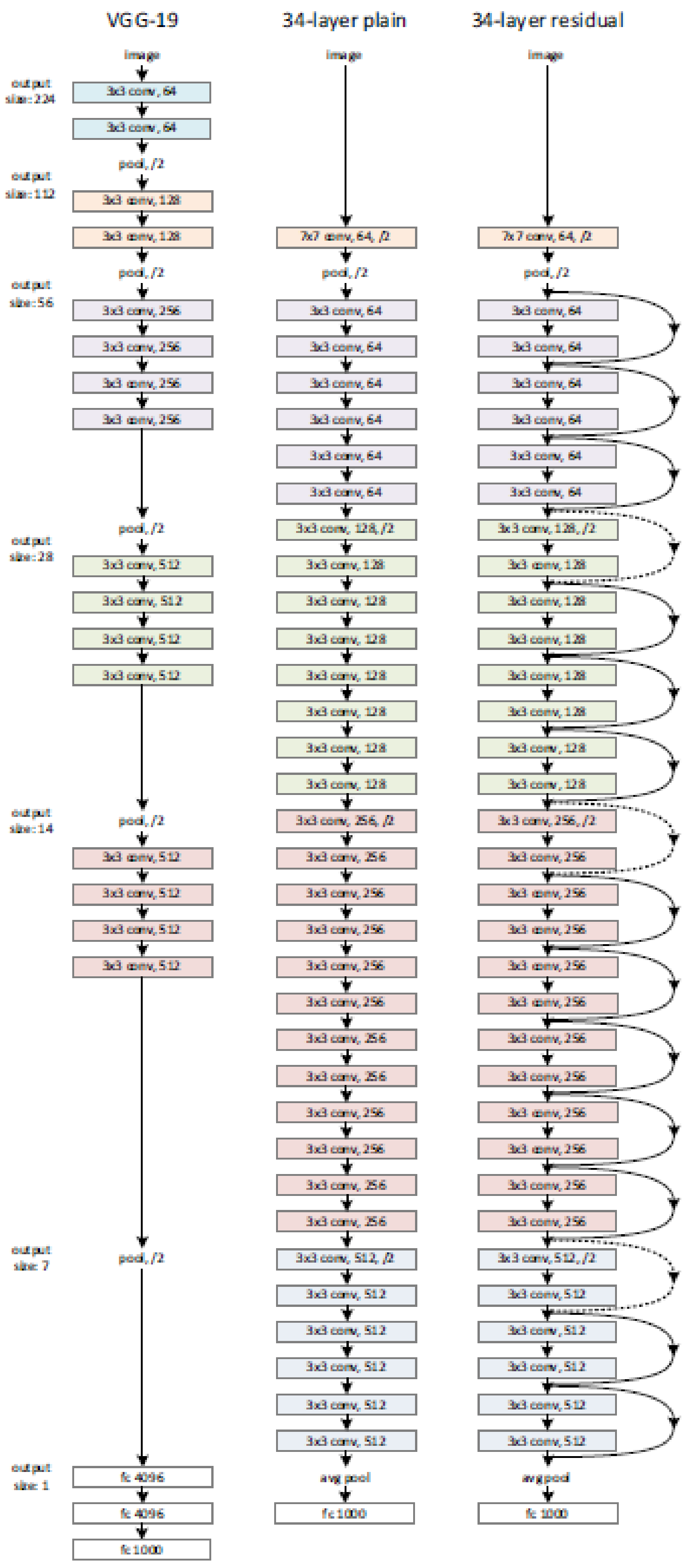


Figure 2.11: Architectures of VGG-19, plain network and residual network from left to right respectively [48].

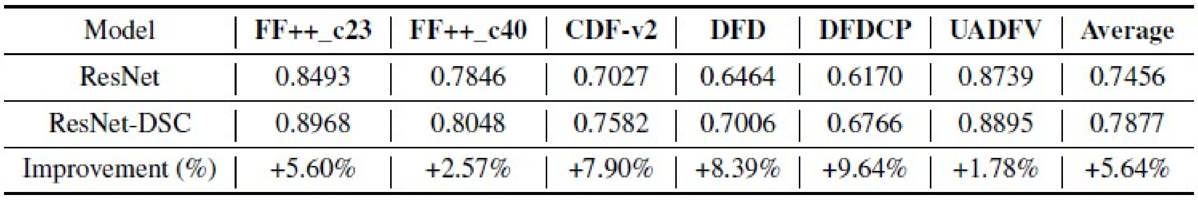
Choice of Backbone

The choice of backbone architecture is an important step when building a deepfake detector based on a CNN. Based on a study [53] done on 3 famous backbones to evaluate the performance of the same method, interesting results were shown.

The study included 4 deepfake detectors: CORE [54], SPSL [55], UCF [11] and Face-Xray [56], the three backbones Xception [50], EfficientNet-B4 [51] and ResNet34 [47] were all integrated into the frameworks of the detectors, the results are shown in Figures 2.13 and 2.14. When analysing the visualisation results we can notice that Xception and EfficientNet outperform ResNet34 despite having the similar number of parameters (between Xception and ResNet34), suggesting that the choice of backbones plays a crucial role in the final performance of the model, which is why most CNN based detectors use either Xception or EfficientNet-B4.

The question now is, why are Xception and EfficientNet-B4 more efficient than ResNet34 despite having nearly the same number of parameters? The answer to this question brings us to two main factors:

- **Depthwise convolution module**: This module is present in both backbones for the exception of ResNet, which makes it the point of difference, to argue this claim, [53] added the module to ResNet and got a substantial increase in the model' performance as shown in Figure 2.12



Model	FF++_c23	FF++_c40	CDF-v2	DFD	DFDCP	UADFV	Average
ResNet	0.8493	0.7846	0.7027	0.6464	0.6170	0.8739	0.7456
ResNet-DSC	0.8968	0.8048	0.7582	0.7006	0.6766	0.8895	0.7877
Improvement (%)	+5.60%	+2.57%	+7.90%	+8.39%	+9.64%	+1.78%	+5.64%

Figure 2.12: Effect of adding depthwise separable convolution to ResNet34 [53].

- **Model's scale**: This is also an important factor, as proven in the study, when adding more layers i.e. using ResNet50, the performance is higher, but this enhancement is limited as using too many layers does not give a better accuracy (when using ResNet152 the performance does not get any better).

2.3.3 Classification of Deepfake Detectors

Deepfake detectors differ from one another accordingly to their goals, some are designed for certain types of deepfakes such as Lip Syncing or Face Swapping, others are designed

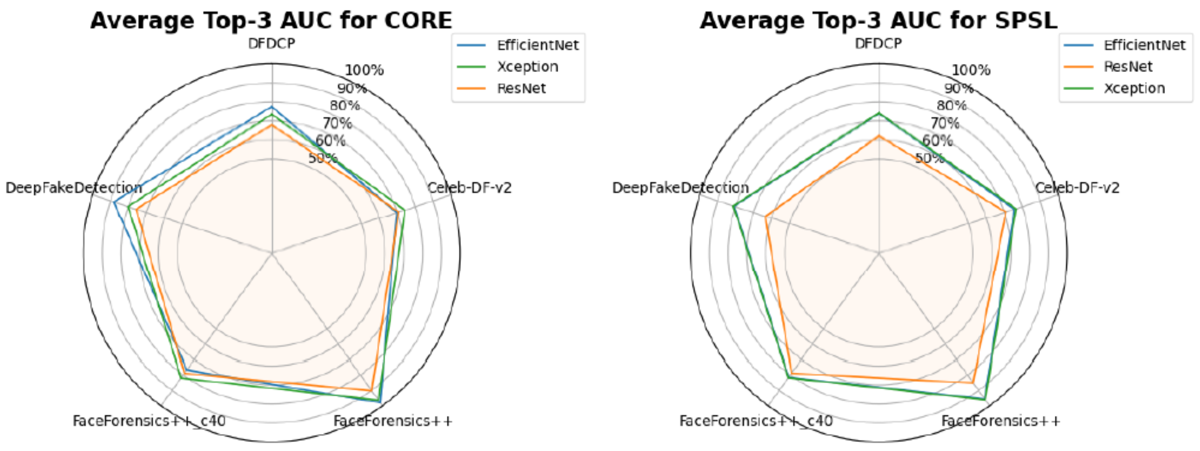


Figure 2.13: Backbone architecture effect on CORE and SPSL [53].

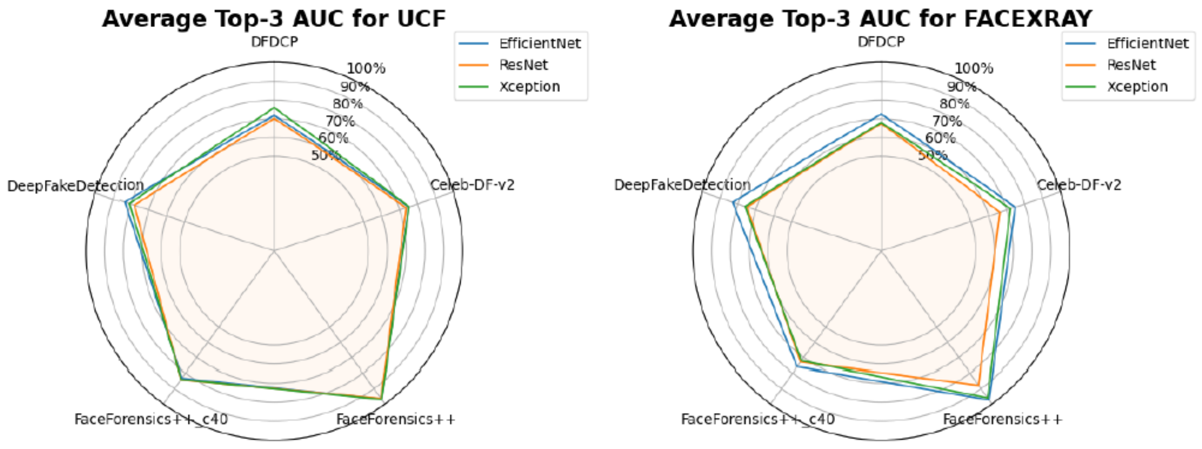


Figure 2.14: Backbone architecture effect on UCF and Face-Xray [53].

for general deepfake detection regardless of the method. These detectors can also vary due to their pipeline naturally, it might be about the base as some use CNNs and others would use Long Short-Term Memory (LSTM) based architectures, or simply the overall and general idea and reasoning behind the algorithm.

There are a lot of ways in which the deepfake detectors could be classified, it could be: Modality-based classification, architecture-based classification, artifact-based classification or detection-type classification. On our work, we implement and dive deeper into the artifact-based classification as it is the most global one. Based on this choice and based on [57], we can distinguish 4 types of detectors: spatial, temporal, frequency-based and Data driven detectors.

Spatial Based Detectors

These detectors analyse the spatial artefacts present in the frames of the videos, i.e. the visual artefacts and anomalies, they analyse single frames or sets of frames without looking at the temporal relation between them. They can be divided into two main categories:

- **Image-level Inconsistency**: The manipulation process often involves local and partial changes in faces rather than global generation (for the exception of entire-face synthesis), which results in local differences and inconsistencies that are the object of such detectors. Such artefacts may include inconsistencies like color difference between the target and source faces, unnatural head pose, inconsistency in facial features and landmarks such as the eyes or the mouth.
- **Local Noise Inconsistency**: Forgery methods usually add, remove or modify the content of images, resulting in abnormalities in noise distribution and textures, which are also known as GAN fingerprints. Many deepfake generation methods would focus on the face region and perfecting it, leaving the environment unattended, which causes some abnormalities such as blurry or distorted edges, mismatched lighting and unnatural patterns and distortions.

Temporal Based Detectors

These detectors analyse the time domain of videos, focusing on sets of frames as a continuous and harmonious set, analysing the relation between frames in 3 different manners. They use LSTM or Transformers to keep track of previous information.

- **Abnormal Physiological Information**: Deepfakes may overlook physiological features and biological signals, failing to imitate those of authentic individuals, resulting in inconsistent patterns, such as the eye blinking frequency or the unnatural head poses and movements across multiple frames.
- **Inter-frame Inconsistency**: These methods focus on the overall inconsistencies left behind by deepfakes within a set of frames in a specific time span, or between adjacent frames. Such temporal inconsistencies may include inter-frame image inconsistency or relationship between facial features and gestures, temporal jitter and flickering...
- **Multimodal Inconsistency**: Instead of focusing on a single modality when analysing the temporal characteristics of a video such as images or audios, these detectors focus on multiple modalities that could be text-visual or audio-visual, with the latter being the most interesting and hottest topic nowadays due to its relationship with Lip syncing forgeries, that we proved earlier are the most dangerous among existing deepfakes. The analysis made by multimodal detectors is done to sniff out inconsistencies between audio and images in frames, or disharmonious operations within each modality, the core idea is to fuse the features of both modalities and use them as single clue to draw conclusions on the authenticity of videos.

Frequency Based Detectors

Deepfakes as sophisticated as they look might overlook the frequency details, where a discrepancy between real and fake content can be found. These frequency differences and artefacts could be local or global frequency statistics, frequency components or frequency characteristic distributions. This type of detectors analyses the frequency information of frames to detect such anomalies and abnormalities to compare it with ordinary information, they could use CNNs fed with frequency features, famous transformations such as discrete wavelet transform (DWT) or discrete cosine transform (DCT), [58] uses both statistics and frequency components in a dual stream network. On top of frequency information, some combine spatial information to mine better for forgery clues.

Data Driven Detectors

Data driven detectors, otherwise known as naive detectors, are as their name suggests classical deep learning approaches aimed at learning specific patterns and features from both real and fake images, using an extensive amount of data, for the final binary classification on whether the content is real or fake. These detectors use vanilla CNNs for their tasks, and usually the infamous available architectures for image classification such as Xception, EfficientNet, HRNet or Resnet, where they adopt a transfer learning approach to train these pretrained CNNs on ImageNet usually for the deepfake detection task.

Table 2.1 summarizes the deepfake detection classes with details about the emphasis of each one.

Spatial	Image-level inconsistency	Color mismatches, feature inconsitencies and background noise
	Local noise inconsistency	Noise distribution and texture abnormalities
Temporal	Abnormal physiological information	Eye blinking frequency, unnatural head movement
	Inter-frame inconsistency	Temporal inconsistencties between frames
	Multimodal inconsistency	Inconsistency between multiple modalities within a set of frames or span of time
Frequency	Frequency	Frequency statistics and components
Data driven	Data driven	CNN data driven models aimed to learn specific patterns

Table 2.1: Classes of deepfake detectors.

2.3.4 Limitations of Deepfake Detection

Detecting deepfakes presents several challenges and limitations, stemming from the complexity and sophistication of deepfake technology as well as the evolving nature of detection methods. These limitations obstruct the advancement of this field in comparison to its counterpart, issues that need to be dealt with sooner or later for a better development. Some of these limitations are:

Generalisation Across Modalities

Deepfake detectors often specialize in analyzing specific modalities of media, such as images, videos, audio recordings, or multiple modalities at once in multimodal scenarios. Generalizing detection capabilities across different modalities can be difficult, leading to gaps in coverage and vulnerabilities to manipulation targeting specific modalities.

Forgery Method Dependence

Deepfake detectors are most of times designed for certain forgery methods and lack the ability to generalise to new and unseen methods, this phenomenon is known as the lack of generalisation (or overfitting) in the detection field. The experimental efforts has shown the alarming lack of generalisation of these detectors even for the same type of forgery with different generation algorithms (different datasets), making the current detectors vulnerable in real-life scenarios.

Contextual Variability

Deepfakes can vary widely in terms of content, context, and quality, making it challenging for detectors to reliably identify manipulation across diverse scenarios. Factors such as lighting conditions, camera angles, and background complexity can affect detection accuracy. This issue is slightly addressed in some datasets that include perturbations such as compression and noise, but they could do so little compared to real-life scenarios in terms of variability, as they include only a handful of image processing techniques.

Computational Complexity

Deepfake detection often requires significant computation resources as many other deep learning fields, adding on top of the operations computed in CNNs and Recurrent Neural Networks (RNNs) the analysis of high resolution videos and images and/or the processing of large datasets (up to 2TB of data), limited computational resources constrain greatly the task when training and especially in real life performances making it difficult and time consuming even when acquiring single powerful Graphics Processing Unit (GPU).

Lack of Open Source

Deepfake detection as important as it might seem lacks a very important aspect that enabled technological development in recent years which is the availability of open source materials. Although many works have published their codes and methodologies, the broad majority of works simply did not, making it a limiting factor in the development of this field.

2.3.5 Challenges of Deepfake Detection

As discussed earlier in what limits the current deepfake detection algorithms, challenges naturally would appear for this field to address and overcome, some of the most important and urgent ones are mentioned in [33] and [20] such as:

Deepfake Datasets

The availability of public deepfake datasets is an important aspect in deepfake detection as they help train the methods on one hand, and evaluate them on the other. Existing datasets [59], [60], [61] as much as they help in these tasks are limited, from one standpoint due to their appearance at least 3 to 4 years ago making their life span almost done as the performance became saturated in some of them, and from another standpoint due to their visible lower quality compared to existing deepfakes on the internet that have become more sophisticated, even when ensuring a high performance on these benchmarks we still risk a low performance in the wild.

Unknown Types of Attacks

The most challenging task currently is to build a robust detector against unknown types of attacks used to fool the detector into considering a deepfake as a real media.

Generalisation to Unknown Forgeries

As the technology continues to grow, new generation methods are surfacing, one more convincing than the other, for deepfake generation it is a breakthrough, for the detection it is a nightmare, as new methods need to be developed to generalise across all synthetic data, a task in which existing recent works [62, 11, 63] started to address but still is far from convincing results.

Adversarial Evolution

Deepfake creators continually adapt their techniques to evade detection, developing new algorithms and strategies to generate more convincing synthetic media. As a result, detection methods must constantly evolve to keep pace with these advancements.

2.4 Conclusion

In conclusion, we have successfully studied the theoretical basis of deepfake generation and detection, helping us understand the origin of such technologies. After a thorough analysis of the technical background for both fields, we analysed some of the most prominent deepfake generation methods of interest, while also listing their limitations. On top of that, we did a thorough classification of deepfake detectors to further understand the goal of each one, and we have finally ended with limitations and challenges facing off this field, some of which are important to us for our study.

Chapter 3

Datasets and Evaluation Metrics for Deepfake Detection

3.1 Introduction

After learning about deepfake detection's general motive and purpose, this chapter is designated to deepfake datasets and evaluation metrics used for deepfake detection.

First, we take a look at what enables us to train and evaluate the models, which are the datasets, going over the most famous public ones, their properties while giving examples and the forgery methods used.

Next, in the second part of the chapter, we break down the evaluation metrics used in this field, going over their expressions, meaning and the conclusions drawn from their given results. All of which to help us understand the detector's behaviour, before ending with a conclusion.

This chapter helps us understand the content of the datasets we use in our review and analysis and the utility and meaning of evaluation metrics in our classification task.

3.2 Deepfake Detection Datasets

After the development of deepfake detectors and algorithms, it is of utmost importance to evaluate these detectors across multiple datasets to study and assess their generalisation ability to new forgeries. It is to this end, that researchers have devoted to the development of such public datasets to set up benchmarks and a unified way to evaluate existing methods and compare between them.

In the following, we will firstly classify the datasets and then we will be looking at 11 state of the art deepfake detection datasets that are most commonly used.

3.2.1 Types of Datasets

Even for the same field, the deepfake detection datasets may vary greatly, and probably the most informative way to classify them is according to the specific task at hand, which may be detecting fake images, fake videos or fake audios. Therefore, we will be looking in the following at some of the datasets in each category.

Image Datasets

These datasets are used for the simple task of detecting whether a given image is real or fake. These datasets contain several forgery methods such as Face Swapping, Face Reenactment or entire face synthesis.

Even though this type of deepfakes is widely used, it is far from interesting in our study since the dangers it poses are minimal compared to their counterparts, on top of the utility of such deepfakes is mostly for fun with the exception of entire face synthesis that could cause security problems.

Some of the most renown datasets of this type are:

- **Fake Faces in the wild (FFIW) [64]**: It contains 131,500 images extracted from FaceForensics [59] dataset and from videos on YouTube.
- **100K-generated-images [14]**: It includes 100k images of synthesized face, bedroom, car and cat images made by a GAN generator trained on real images from FFHQ [65] and LSUN [66].
- **iFakeFaceDB [19]**: This dataset includes 87,000 224x224 face images, generated by processing some StyleGAN-generated synthetic images using the GAN-fingerprint Removal approach (GANprintR) proposed by Neves et al [19].
- **Diverse Fake Face Dataset (DFFD) [67]**: This dataset contains 299,039 images, including 58,703 real images sampled from three datasets (FFHQ [65], CelebA [68] and FaceForensics++ [69]) and 240,336 fake ones in four main facial manipulation types (identity swap, expression swap, attribute manipulation, and entire synthesis).

Video Datasets

Since the video deepfakes are the broader version and the origin of the technology, much more thought and research has been given to them in terms of detection, and development of datasets. Since these deepfakes present the highest risk and more attention, our focus was solely on them. More about this type is detailed in Section 3.2.2.

Audio Datasets

Audio manipulation has been around even before the word deepfake, but it has been associated to the technology as a general terminology. There are two major types of audio datasets, starting with the voice conversion challenge [70] that started in 2016 and include voice conversion audios, where a Target speaker is made saying what a source speaker has said before.

The second type of datasets is the ASVspoof [71] (Automatic Speaker Verification Spoofing and Countermeasures) challenge, from which the 2019 and 2021 versions contain audios that could be considered as deepfakes.

3.2.2 Video-based Datasets

This type of deepfake detection dataset has forged videos with different forgery methods, enabling the researchers to effectively evaluate the generalisation ability of their detectors, we will be looking at 11 state of the art datasets that are most commonly used, including the number of videos, the methods used to obtain them, the modalities and some additional useful information.

UADFV

The first dataset released for deepfake detection was UADFV [72] back in 2018. It consists of a total of 98 videos, where 49 are real videos collected from YouTube and manipulated by using the FakeApp [24] application to generate 49 fake videos. The average length of videos is 11.14 sec with an average resolution of 294×500 pixels. However, the visual quality of videos is very low, and the resultant alteration is obvious and thus easy to detect. It was originally developed to detect and expose the unnatural eye blinking in deepfakes, an example is in Figure 3.1.

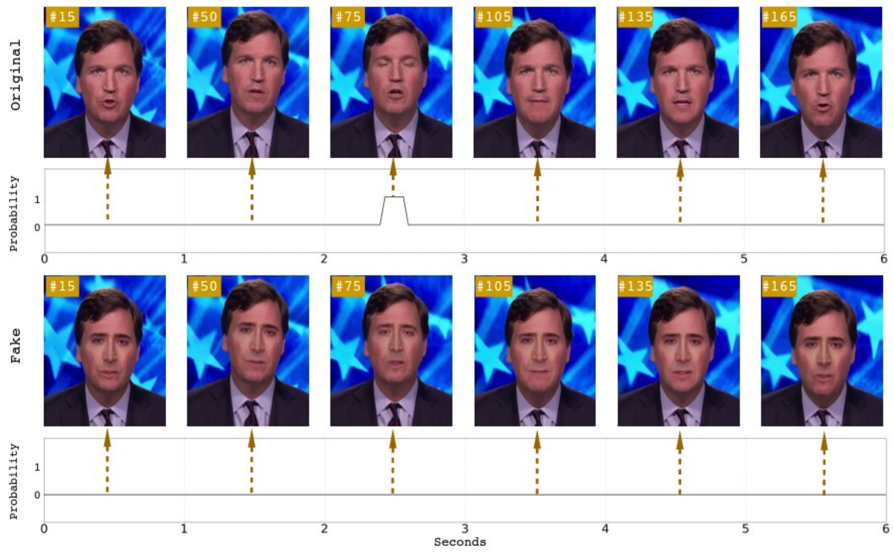


Figure 3.1: A temporal comparison between a real video (top) and a fake video (bottom) from the UADFV dataset in term of eye blinking pattern [72].

FaceForensics++ (FF++)

The most famous dataset for deepfake detection is FaceForensics++ [69], it appeared back in 2019 as an extension of FaceForensics [59]. It contains 1000 real videos collected from YouTube, and from which are generated 4000 fake videos, examples can be seen in Figure 3.2, the four forgery methods used are: FaceSwap [21] and Deepfakes [22] for face swapping, and Face2Face [25] and Neural Textures [26] for facial reenactment.

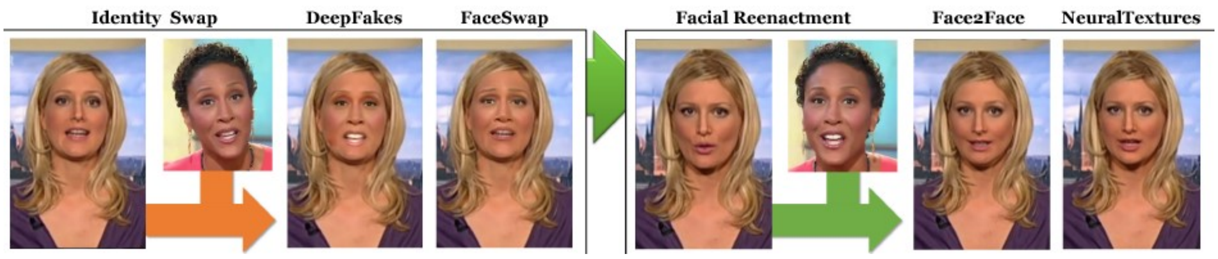


Figure 3.2: Examples of forgeries from FaceForensics++ dataset [69].

The modality of its videos is visual, and the average duration is about 18s per video. This dataset is available in three quality levels: raw version, c23 version which is the high quality (slightly compressed, constant rate quantization parameter equal to 23) and c40 which is the low quality (highly compressed, constant rate quantization parameter equal to 40), the compression format is H.264.

This dataset is the most important out of available dataset due to the consensus in the deepfake detection field to ***Train*** *the models on* ***FaceForensics++****, which makes it a stepping stone for almost all existing algorithms.*

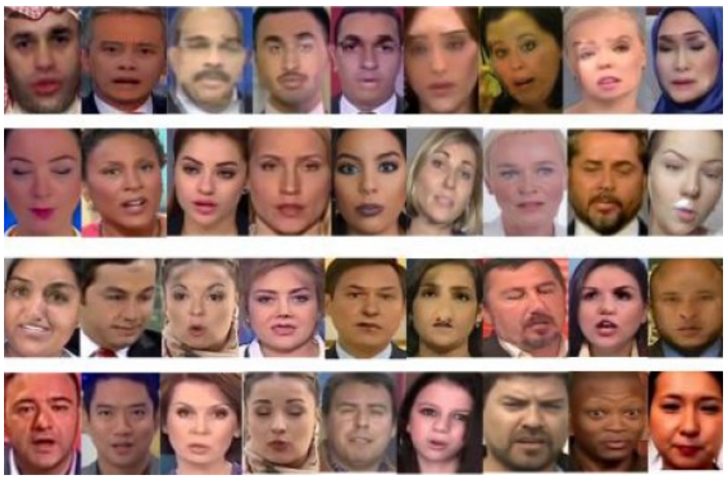


Figure 3.3: Examples of color mismatch and landmark inconsistencies in some frame of the FaceForensics++ dataset [69].

The issue with this dataset though is the inability to generalize to LipSync deepfakes, on top of the visual color and landmark inconsistencies present around the manipulated videos as shown in Figure 3.3, with deepfakes being row 1, Face2Face in row 2, FaceSwap in row 3 and Neural textures in row 4.

DeepFakeDetection (DFD)

DeepFakeDetection [73] is a dataset developed by Google, containing 363 real videos of 28 paid actors, depicting them in different scenarios such as walking from room to room or sitting down talking during a meeting, and from which 3068 fake videos were generated using 5 different face swapping methods, an example is shown in Figure 3.4, in which we can not even distinguish which is the real from the fake face, the quality leap compared to FaceForensics++ is apparent, however, it also presents many visual artefacts.

Deepfake Detection Challenge-Preview (DFDCP)

This dataset [74] contains 5,244 face videos of 66 subjects, 4113 from which are fake using two face swapping algorithms. It was released as a preview to the full dataset of

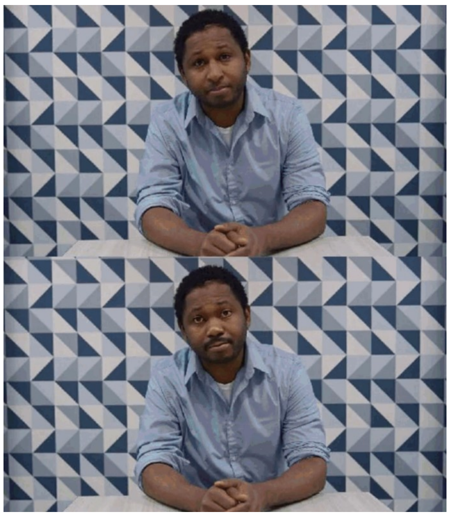


Figure 3.4: DeepFakeDetection dataset example [73].

the 2020 Deepfake Detection Challenge (DFDC), although the videos have their audio, it does not use any LipSync forgeries, making it classified as audio-visual used only for visual forgeries. An example is shown in Figure 3.6, where fake faces are on the right.

Figure 3.5: Deepfake detection challenge preview dataset example [74].

Deepfake Detection Challenge (DFDC)

Launched by Facebook in 2020, DFDC [61] is one of the largest public dataset in this field, counting for 23,654 real videos of 3426 paid actors, and with the help of **eight** forgery methods they generated 104,500 fake videos, the manipulation methods are: DFAE-128 [3], DFAE-256 [3], MM/NN face swap, NTH [75], FSGAN [5], StyleGAN [14], refinement (post processing step to improve perceptual quality), and audio swap. Examples are illustrated in Figure 3.6.

The average duration of a video is about 10s. Various augmentations (see Figure 3.7) have been applied on the videos from which we distinguish two types:

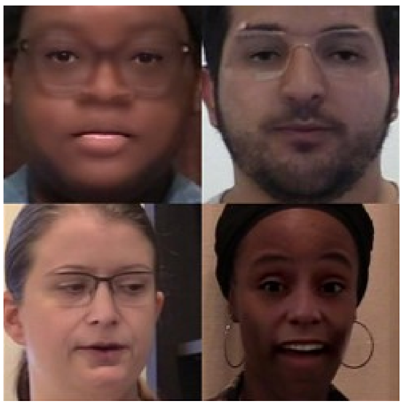


Figure 3.6: Deepfake detection challenge dataset fake samples [61].

- **Augmenters:** Applies geometric and color transforms, they have been applied on 70% of the videos, and some of the augmentations are: Gaussian blurring, brightening/darkening, adding contrast, altering the frame rate, converting to grayscale, horizontal flipping, audio removal, adding noise, altering the encoding quality, altering the resolution, and rotating.
- **Distractor:** Overlays various kinds of objects, it has been applied on 30% of the videos. The simplest distractors overlay random text, shapes, and dots onto each frame of a video and move around frame to frame. There can either be consistent movement (i.e. moving across horizontally or vertically) or random movement.

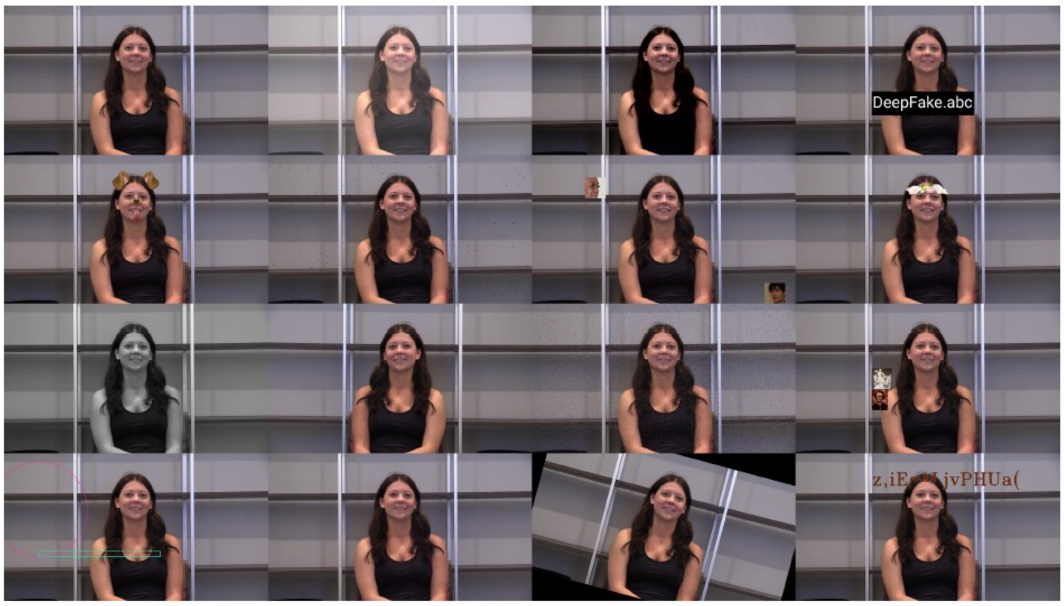


Figure 3.7: Examples of augmentations applied on a video from the DFDC dataset [61].

The main drawback of the dataset is that the quality level of data is different due to several deepfake generative abilities. Therefore, some samples have the problem of boundary mismatch, plus, source faces and target faces have different resolutions.

Celeb-DF-v1 (CDF1)

One of the most popular deepfake dataset is Celeb-DF [60] that is widely used for the evaluation of deepfake detectors. It came handy for having high quality deepfakes, trying to overcome the visible artefacts present in previous datasets.

Celeb-DF-v1 is the first version of the larger dataset presented in the following section, it contains 408 original videos collected from YouTube and containing several subjects (celebrities) of different ethnic groups, age and gender, and 795 fake videos obtained by a Face Swapping algorithm applied on the real videos. Some examples are displayed in Figure 3.8, where the first column represents the real faces, and the following ones are the manipulated faces derived from those original ones.

Celeb-DF-v2 (CDF2)

This dataset is the larger version and the usually referred to as Celeb-Df in most papers, it is derived from 590 real videos extracted from YouTube, resulting in 5639 manipulated samples generated through an improved DeepFake [22] algorithm. Samples from this dataset could be seen in Figures 3.8 and 3.9.

The high quality that the content of this dataset exhibits is no fluke, as much work and focus went into it. For instance, The dataset exhibits a large variation of face sizes, orientations, and backgrounds. In addition, some post-processing work is added by increasing the high resolution of facial regions, applying color transfer algorithms and inaccurate face masks.

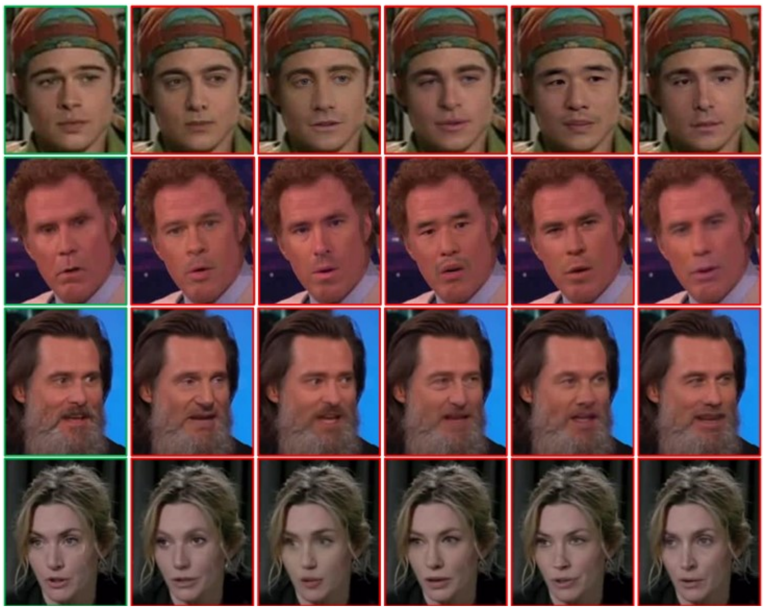


Figure 3.8: Examples from Celeb-DF dataset [60].

We can clearly confirm the previous claims through Figure 3.9, from which we can not event distinguish that these are fake samples, probably from the exception of the last row where the trans-gender and trans-racial swapping was made.

The downside of this dataset is the low data amount, as only few samples were used and were all downloaded from YouTube. The average duration of the videos is around 13s, and the only modality available is visual since it only focused on Face Swapping.

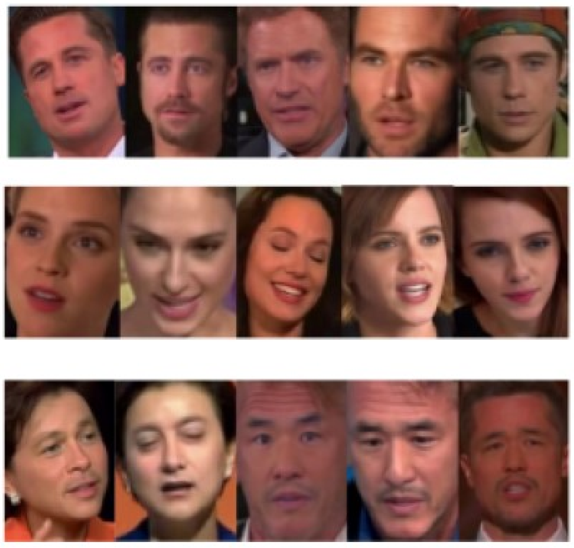


Figure 3.9: Fake samples from Celeb-DF dataset [60].

FaceShifter (FSh)

Unlike any other dataset that uses a public open source generator, this dataset uses a state of the art face swapping method named FaceShifter [40]. To generate this dataset, they applied This generation algorithm on the 1000 real videos of FaceForensics++ [69] to obtain the 1000 face swapped videos. Some examples are shown in Figure 3.10.

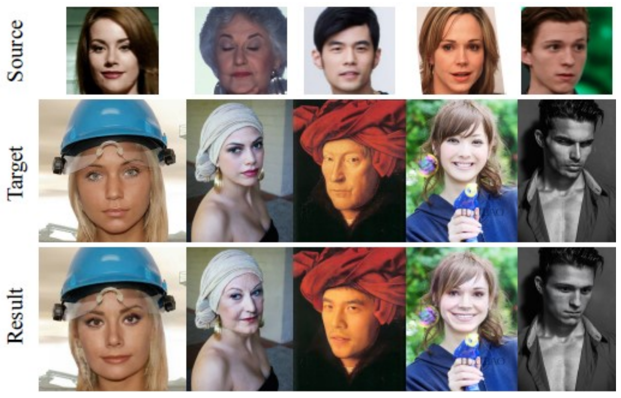


Figure 3.10: Examples of FaceShifter swapped faces [40].

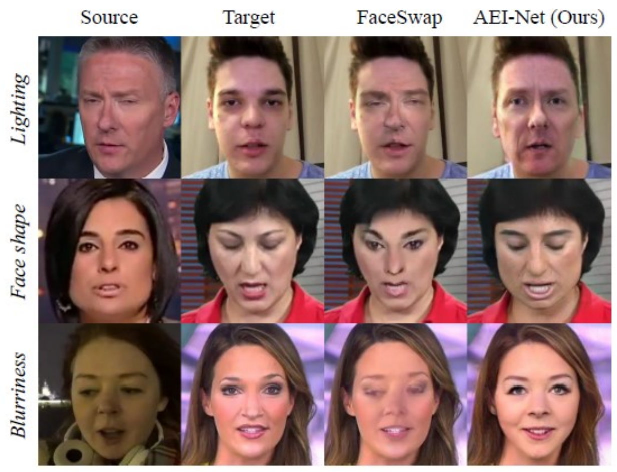


Figure 3.11: Difference between the quality of FaceSwap in FaceForensics++ and FaceShifter [40].

In this method, they focused on improving the fidelity of face swapping. Meaning, in order to have a perceptually appealing results, the target and sources faces must share the same properties in terms of lighting, resolution, pose and expression.

In order to achieve this high fidelity face swapping, they proposed a two stream network consisting of a GAN-based network named Adaptive Embedding Integration Network (AEI-Net) for a thorough integration of target attributes, and a Heuristic Error Acknowledging Refinement Network (HEAR-Net) to refine the results. The results of this method compared for FaceSwap is shown in Figure 3.11, where the difference of quality is clear as daylight between the results obtained using FaceSwap for FaceForensics++ and FaceShifter.

DeeperForensics-1.0 (DF-1.0)

DF-1.0 [76] is another large scale dataset for deepfake detection, counting for 50k original videos and 10k fake videos obtained using a face swapping algorithm that used a novel conditional autoencoder named DF-VAE (the same used for DFDC dataset), which assured the high quality of presented fake samples as seen in Figure 3.12.

This dataset also exhibits great diversity in actor's identity (ages between 20 to 45, gender, skin tone with 4 different skin tones and ethnicity with 26 different nationalities), head poses, 7 different angles and 8 different face expressions.

Furthermore, to imitate real life scenarios, perturbations and distortions were added such as color saturation, local block-wise, color contrast change, Gaussian blur, Gaussian noise, JPEG compression, video compression and sometimes a mix of multiple perturbations could be applied.

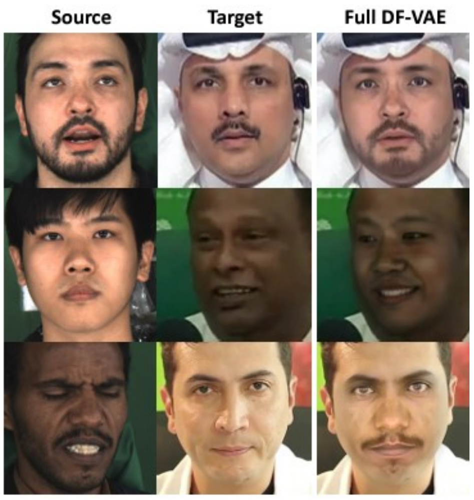


Figure 3.12: Few diverse examples from the DeeperForensics dataset [76].

Wild-Deepfake (WDF)

On the contrary of previous datasets, Wild-deepfakes [77] contains 3805 real samples and 3509 fake samples all collected from the internet, making it the most realistic one. It covers diverse scenes, multiple people in each scene and rich facial expressions.

FakeAVCeleb (FAVC)

Since multi-modal deepfake detection is greatly overlooked, and most of the audio-visual datasets used lately are private, this public dataset [78] that appeared in 2021 is an audio-visual multimodal dataset, that has 500 real videos, and from which 19,500 fake videos were generated using 4 forgery methods that are FaceSwap [21] and FSGAN [5] for face swapping, SV2TTS [79] for audio cloning, and finally Wav2Lip [7] for LipSync deepfakes.

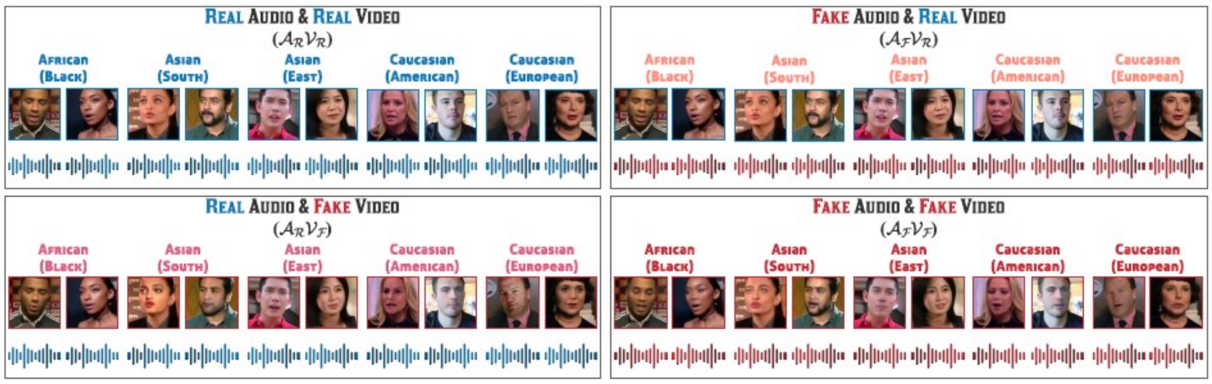

Real Audio & Real Video
($\mathcal{A}_\mathcal{R}\mathcal{V}_\mathcal{R}$)
Fake Audio & Real Video
($\mathcal{A}_\mathcal{F}\mathcal{V}_\mathcal{R}$)
Real Audio & Fake Video
($\mathcal{A}_\mathcal{R}\mathcal{V}_\mathcal{F}$)
Fake Audio & Fake Video
($\mathcal{A}_\mathcal{F}\mathcal{V}_\mathcal{F}$)
African (Black)
Asian (South)
Asian (East)
Caucasian (American)
Caucasian (European)

Figure 3.13: The four categories of FakeAVCeleb dataset [78].

We can divide this dataset into the next 4 main combinations that are represented visually in Figure 3.13:

- **Real video real audio**: These are the real samples of the dataset, appearing on the top left in Figure 3.13.
- **Real video fake audio**: This subset contains the 500 real videos with a cloned audio obtained using a transfer-learning based approach on a real-time voice cloning tool (SV2TTS). This subset is not Lip Synced!
- **Fake video real audio**: This type consists of fake videos with real audios, the fake video is obtained using face swapping through FaceSwap [21] or FSGAN [5], or face reenactment using wav2lip [7] to lip sync the face to the real audio. They generated 9000 such videos, and the pipeline could be seen in Figure 3.14 on the middle.
- **Fake video fake audio**: This subset is the largest one with 10,000 videos, obtained by the combination of the previous two subsets' methods, face swapping is conducted by faceswap [21] and FSGAN [5], the fake audios are generated by SV2TTS, and finally to combine the two wav2lip [7] is used to LipSync the videos with the audios, as shown on the right of Figure 3.14.

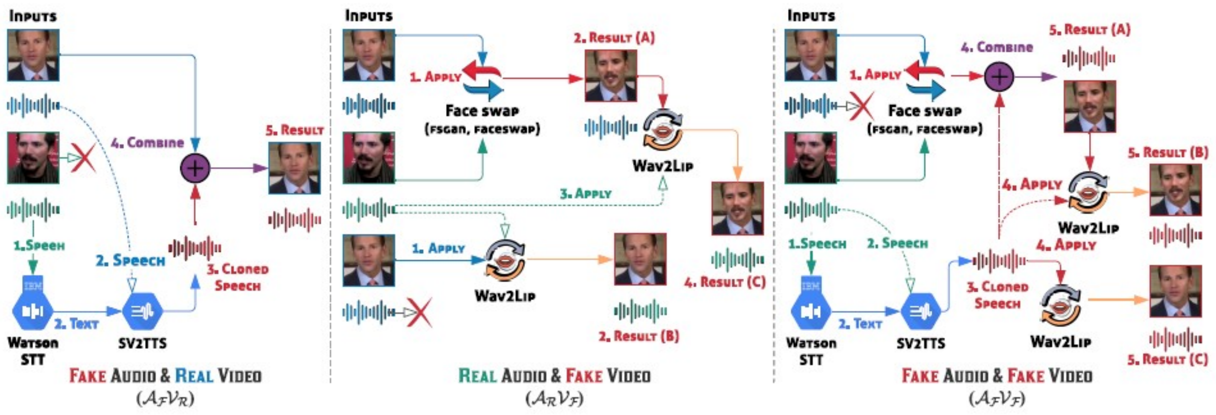


Figure 3.14: Pipeline of FakeAVCeleb dataset [78].

3.3 Evaluation Metrics

Evaluation metrics, also known as performance metrics, are necessary for every classification/detection task, which the deepfake detection is classified as one. Here, we detail more about metrics and their origin and expressions, on top of giving the meaning of each one of them, the study was based on [80].

3.3.1 The Confusion Matrix

Deepfake detection is after all a binary classification in the perspective of deep learning. It takes as inputs actual positives (1) or actual negatives (0), and associates a binary value to each input that we note predicted positive (1) or predicted negative (0). For instance, in the terminology of deepfake detection, positives refer to deepfakes and negatives refer to real videos/images.

	Predicted positive	Predicted negative
Positive input	TP	FN
Negative input	FP	TN

Table 3.1: Confusion matrix of a binary classifier.

A fundamental tool used in evaluating a binary classifier is the confusion matrix that summarises the success and failure of the classification model [80], on the Y axis are the actual values and on the X axis are the predicted values, the classification is correct/ true when the two values match, and it is incorrect/false when the two values do not match. Table 3.1 shows the correct predictions in green (TP stand for **True Positive** and TN stand for **True Negative**) whereas the false predictions are shown in red (FP stands for **False Positive** and FN stands for **False Negative**).

3.3.2 Precision and Recall

Based on the four fundamental values presented in 3.3.1, two new and equally important metrics for binary classifiers have been proposed : Precision and recall.

Precision of a binary classifier is the fraction of correctly predicted positives (correctly predicting a deepfake) among all the predicted positives (all the videos predicted as deepfakes whether it is correct or false), Equation 3.1 shows a formal definition of the metric. In the confusion matrix, it is the fraction of true samples in the first column.

$$precision = \frac{TP}{TP + FP} \tag{3.1}$$

this metric shows *how many times a predicted deepfake is correct among all media predicted as deepfake.*

Recall of a binary classifier is the fraction of correctly predicted positives (deepfakes) among all positive samples (deepfake inputs), it is the fraction of the true samples in the first row of the confusion matrix, it is formulated as follows :

$$recall = \frac{TP}{TP + FN} \tag{3.2}$$

This metric shows *how many times a predicted deepfake is correct among all input deepfakes.*

3.3.3 True and False Positive Rates

Focusing on positive predictions, we have two metrics: **True Positive Rate (TPR)** or Correct Detection Rate (CDR) as the fraction of predicted positive samples among all positive input samples which is the fraction of TP in the first row of the confusion matrix, and **False Positive Rate (FPR)** or False Alarm Rate (FAR) as the fraction of predicted positive samples among all actually negative input samples, which is the fraction of FP in the second row of the confusion matrix. We note that TPR is another name for recall. Equations 3.3 and 3.4 show these two metrics.

$$TPR = \frac{TP}{TP + FN} \tag{3.3}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.4}$$

3.3.4 True and False Negative Rates

Similarly to true and false positive rates, there are: **True Negative Rate (TNR)** which is the fraction of predicted negative among all negative input samples, and **False Negative Rate (FNR)** which is the fraction of predicted negatives among all actually positive input samples as shown in Equations 3.5 and 3.6.

$$TNR = \frac{TN}{TN + FP} \tag{3.5}$$

$$FNR = \frac{FN}{FN + TP} \tag{3.6}$$

Equation 3.5 shows the amount of times a real media is predicted correctly among all real media, where Equation 3.6 shows the amount of times the detector predicted a deepfake as real from all deepfake inputs.

3.3.5 Equal Error Rate

When focusing on error rates we consider the two most important and conflicting pair: **FPR** and **FNR**, as increasing one leads to reducing the other. From a practical standpoint, we need to keep both rates within an acceptable threshold and find the right balance.

Based on this hypothesis, a new metric has been proposed which is **Equal Error Rate (EER)** which is the point where FPR and FNR are equal. Equal error rate is sometimes no necessarily a good metric where these two types of errors have different levels of importance. For example, for detecting critical deepfakes (fake news that can influence how people cast their votes) we can often tolerate more false positives (false alarms) than false negatives (missed alarms).

3.3.6 Accuracy

Accuracy gives the most balanced indication on the overall performance of a binary classifier and is defined based on all four fundamental values as shown in Equation 3.7

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.7}$$

Accuracy of a binary classifier is the fraction of correctly predicted samples among all existing samples, in other words, how many times the detector got the prediction right.

3.3.7 Receiver Operating Characteristic Curve and Area Under the Curve

Receiver Operating Characteristic curves are usually used to measure the performance of a binary classifier that outputs a score for the prediction.

Let S be the test set, f(s) is the output score for a sample s ∈ S, this value lies within the interval [a,b], let t ∈ [a,b] be the prediction threshold for our classifier, we assume the classifier works as follows (which is mostly the case):

$$class(s) = \begin{cases} positive, iff(s) \geq t \\ negative, otherwise \end{cases}$$

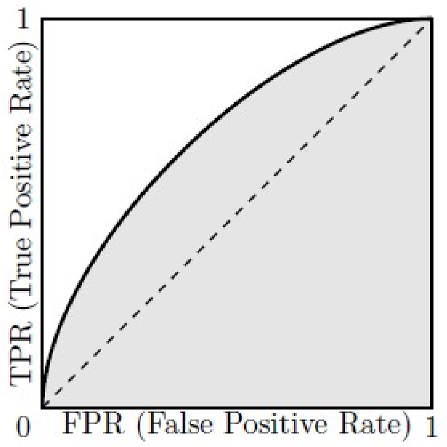


Figure 3.15: Representative ROC curve, the AUC is the area in gray [80].

By changing t between a and b continuously, we can get a continuous curve that describes how TPR and FPR change values from (0,0) to (1,1) on the 2D plane, resulting in the ROC curve of a binary classifier.

Tthe Area under the curve (AUC) is defined as the surface below the ROC curve, shown in gray in Figure 3.15. We note that the lowest value it could take is 0.5 which is the case of a random model, and the highest is 1 for a perfect model.

In terms of deepfake detection, AUC is very informative on the performance and behaviour of the model, as it links True Positive Rate with False Positive Rate, meaning that it shows how many times the model is correctly predicting deepfakes, while at the same time minimising wrongfully classifying real videos for deepfakes.

3.4 Conclusion

In conclusion, we have explored through this chapter two pillars of deep learning models, which are datasets and evaluation metrics. For the datasets, we have seen the forgeries they included on top of some examples, while also mentioning some of their disadvantages. As for the evaluation metrics, we have defined the relevant ones on top of the information they provide us about the performance of algorithms. Therefore, we have effectively set the tone for the next chapters, as we will chose from these datasets and metrics the most useful and informative ones to help us get through the study efficaciously.

Chapter 4

Deepfake Detection: Evaluation Methodology

4.1 Introduction

In this chapter, we set the basis for the experiments coming up next, where after a thorough research, we have identified detection algorithms of interest, the most prominent and diverse datasets, and finally the most insightful evaluation metrics.

First and foremost, the four deepfake detectors selected are detailed with great emphasis on the methodology, the pipeline and the loss functions included, all of which to help us understand the interest of the methods. Next, a new detector which is SBI [12] was added due to its interesting results and methodology. Then, a LipSync detector has been added to the study to further improve the study's depth to multi-modal detectors.

Next, we set the evaluation methodology for our experiments, first by establishing a criteria for the choice of datasets we will use later on top of an analysis to understand the differences, then we move on to the evaluation metrics selection alongside the provided information by each one.

Finally, a conclusion on the approach is made to conclude the theoretical part of the project, while giving some perspectives on the deepfake detection field.

4.2 Deepfake Detectors

In our study, since almost all datasets are made using face swapping and face reenactment techniques, a forced choice is to investigate detectors that are compatible with the available data. Therefore, we have focused more on Spatial and Frequency detectors from Section 2.3.3 of the second chapter, these detectors are the first four from Table 4.1, they were chosen for the evaluation in Chapter 5.

The fifth detector which is SBI was introduced later in the study because of its high performance and interesting pre-processing step, a solution for a better generalisation in Chapter 6. The last detector is LipFD, which was only included in Chapter 6 for testing and not in the evaluation study since we did not have the necessary material (datasets and source code). All detectors are included in Table 4.1.

Detector	Modality	Purpose of use
Frequency in Face Forgery Network (F3Net)	Unimodal	Evaluation
Spatial Phase Shallow Learning (SPSL)		
Consistent Representation Learning (CORE)		
End-to-end Reconstruction Classification Learning (RECCE)		
Self Blended Images (SBI)		Generalisation
Lip-syncing Forgery Detection (LipFD)	Multimodal	Test on lip-syncing videos

Table 4.1: Deepfake detectors selected, alongside thei modality and the purpose of their use

4.3 Frequency in Face Forgery Network (F3Net)

This detector was introduced in 2020 by Qian et al. [58] to effectively use frequency forgery clues in a CNN based architecture for deepfake detection.

4.3.1 Overview

One of the pioneering methods in face forgery detection, this methods studies frequency discrepancy between real and fake images since visual artefacts could be hidden using some post processing (eg compression as shown in Figure 4.1-(a)). Research found subtle forgery patterns in high frequency bands on one hand (see Figure 4.1-(b) FAD), and a difference in local statistics between manipulated and real samples on the other (see Figure 4.1-(b) LFS).

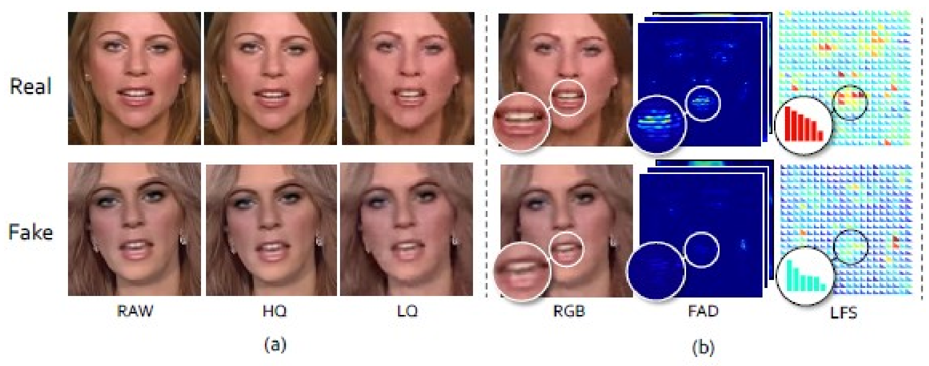


Figure 4.1: (a) shows the visual quality difference between real and fake faces under different compression levels. (b) shows the subtle artifacts and different statistics of the forged faces [58].

Based on the aforementioned information, Frequency in Face Forgery Network (F3Net) was built to effectively use frequency domain information in CNNs through its three key parts, *Frequency-aware image decomposition (FAD)* is aimed to learn subtle forgery patterns through image decomposition, *Local Frequency statistics (LFS)* module to describe the frequency-aware statistical discrepancy between real and forged faces. These two branches, are further gradually fused through a cross-attention module, named MixBlock, which encourages information interaction between the FAD and LFS branches.

4.3.2 Pipeline

The pipeline of the method during training can be seen in Figure 4.2, where there is a two stream network in FAD and LFS, and then a Mixblock for the fusion between the two information, this architecture can be detailed as follows:

FAD: Frequency-Aware Decomposition

This module does not rely on fixed filtering configurations but rather on learnable filters. The resulting frequency components can be inversely transformed to the spatial domain and used as an input to a CNN.

They used N base filters to decompose to low, middle and high frequency bands, and then added learnable filters f_w, the frequency response is obtained by $f_{base} + \sigma(f_w)$ where $\sigma(x) = \frac{1-exp(-x)}{1+exp(-x)}$, therefore the decomposed image to an input in one frequency domain is :

$$y_i = D^{-1}\{D(x) * [f^i_{base} + \sigma(f^i_w)]\} \tag{4.1}$$

D is the discrete cosine transform, it was used because of its wide utilisation, nice layout and its application in compression algorithm making it compatible towards the description of compression artifacts out of the forgery patterns.

LFS: Local Frequency Statistics

Since it is infeasible to to mine forgery artifacts directly from the spectral representation, LFS is important. To an input image, a sliding window DCT is applied to extract localized frequency responses, and then counting the mean frequency responses at a series of learnable frequency bands. These frequency statistics re-assemble back to a multi-channel spatial map that shares the same layout as the input image.

So as shown in Figure 4.2, for each window $p \in x$, the statistics are gathered in each frequency band (colors), the statistics in each band are :

$$q_i = log_{10}||D(p) * [f^i_{base} + \sigma(f^i_w)]||_1 \tag{4.2}$$

Log is applied to balance magnitude in each band.

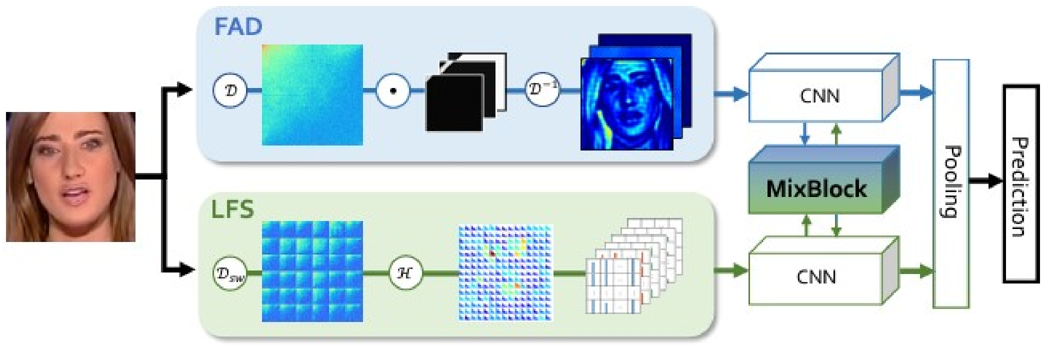


Figure 4.2: F3Net pipeline [58].

Two-Stream Collaborative Learning Framework

The two previous modules mine clues that are different but complementary. To fuse the two-stream FAD and LFS features, a cross-attention fusion is used every several Xception blocs as shown in the Figure 4.3, the difference with the simple concatenation is that they calculated the cross-attention weight using feature maps. (The cross-attention matrix is adopted to augment the attentive features from one stream to another).

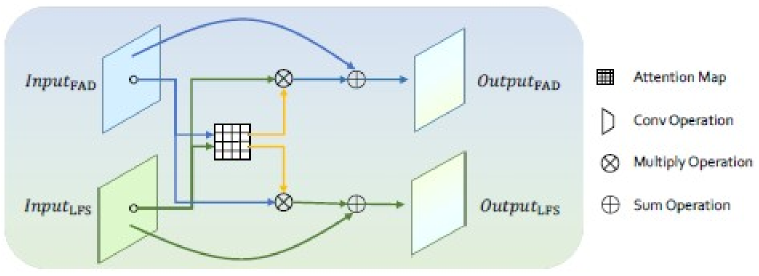


Figure 4.3: F3Net MixBlock architecture [58].

4.3.3 Loss Functions

Cross Entropy

For this task, since it is considered as a binary classification as the network gives a probability of an input being real (ground truth set at 0) or fake (ground truth set at 1), they used the most widely used and compatible loss function which is the **Cross entropy loss**, otherwise known as Log loss.

The formula for this loss is :

$$L(y, \hat{y}) = -(yLog(\hat{y}) + (1 - y)Log(1 - \hat{y})) \tag{4.3}$$

Where :

- y is the ground truth (0,1)
- $\hat{y}$ is the prediction score of the positive class (i.e. P(y=1))

For a dataset containing N samples, the loss becomes:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} yLog(\hat{y}) + (1 - y)Log(1 - \hat{y}) \tag{4.4}$$

4.3.4 Reported Results of the Original Paper

The detector being trained on FaceForensics++ [69] c23 version, the learning rate was set to 0.002, SGD as an optimizer and a batch size of 128, the results are in Table 4.2, where they only tested on FF++ with its three compression levels.

method	ACC (raw)	AUC (raw)	ACC (c23)	AUC (c23)	ACC (c40)	AUC (c40)
F3Net	99.95%	99.8%	97.52%	98.1%	90.43%	93.3%

Table 4.2: F3Net paper results.

4.4 Spatial Phase Shallow Learning (SPSL)

This detector was introduced in 2021 by Liu et al. [55] to capture the up-sampling artefacts in face forgery, combining spatial information with phase spectrum information.

4.4.1 Overview

This method uses both frequency and spatial information to detect face forgeries. On the frequency side, it uses phase spectrum to look for artefacts left by up-sampling operations which are a pivotal step in deepfake generation, the choice of phase spectrum was due to the fact that it is more sensitive to up-sampling than amplitude spectrum as illustrated in Figure 4.4, as the pixel-wise difference increases when up-sampling operations' number increases.

They then blend this information (after applying inverse DFT on phase spectrum to obtain the spatial form) with the spatial information of the image to feed to the CNN.

Additionally, they shallowed the network by dropping many convolutional layers to force the model to concentrate on local regions which are rich in texture information and lack high-level semantic, claiming they are more important.

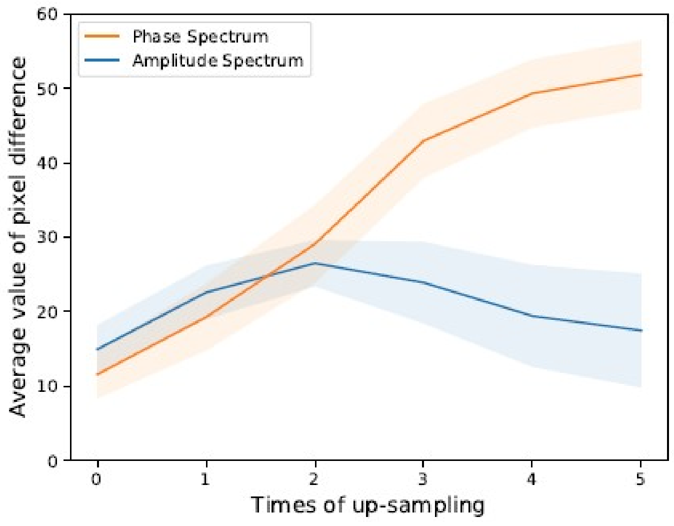


Figure 4.4: Average pixel wise difference in reference to up-sampling for phase spectrum and amplitude spectrum [55].

4.4.2 Pipeline

Since upsampling is a must for deepfake generation, first phase information is studied, then to apply it to the CNN backbone, a spatial domain representation of this phase spectrum is constructed and then concatenated with the RGB image to obtain an RGBP-4channel image that serves as the input into the CNN backbone network. Therefore, the Pipeline of the method can be divided into two main parts :

Capturing Upsampling Artefacts Via Phase Spectrum In Face Forgery

Claim 1: *Phase spectrum is more sensitive to up-sampling artifacts and therefore helps face forgery detection.*

In the following, to simplify calculations, all mathematical derivations are based on 1D signals. The basic notation is x(n) and X(u) represents the 1D discrete signal and its Discrete Fourier Transform (DFT) respectively, n is the spatial location of the signal and u is the frequency. A(u) is the amplitude spectrum and P(u) is the phase spectrum. c(n) and C(u) represent the kernel and its DFT representation. * is the convolution operation.

- **Proving that phase spectrum keeps more frequency components**:

  First, we have that :

  $$X(u) = R(u) + j.I(u) \tag{4.5}$$

  Where R(u) and I(u) denote the real and imaginary part of X(u) respectively. The amplitude spectrum is :

  $$A(u) = \sqrt{R^2(u) + I^2(u)} \tag{4.6}$$

  Axiom 1: *Low-frequency components dominate the frequency domain for a natural image, and many high frequency components are very small even tend to zero.*

  Based on the previous axiom, $\exists u_k \in U = [u_0, u_1, ...u_m]$, from which for every value > k we have $A(u_k) \approx 0$

  According to Equation 4.6 we have :

  $$A(u_k) \approx 0 <=> R(u_k) \approx \pm 0, I(u_k) \approx \pm 0 \tag{4.7}$$

  For phase spectrum, the definition is

  $$P(u) = arctan\frac{I(u)}{R(u)} \tag{4.8}$$

  For every $u_k \in U$ we have

  $$P(u_k) = arctan\frac{I(u_k)}{R(u_k)} \approx arctan \pm 1 = \pm\frac{\pi}{4} \tag{4.9}$$

- **Capturing the up-sampling artefacts**:

Proof. The increase in spatial resolution in 2D corresponds to the extension of time domain in 1D. Let's assume that the input was up-sampled by two:

$$\hat{x}(n) = \begin{cases} x(\frac{1}{2}n), n = 2k \\ 0, n = 2k+1 \end{cases} \tag{4.10}$$

Where k=0,1,...,N-1, we then have:

$$\begin{gathered} \hat{X}(u) = \frac{1}{2N} \sum_{n=0}^{2N-1} \hat{x}(n) e^{-j\frac{2\pi un}{2N}} \\ \hat{X}(u) = \frac{1}{2N} \sum_{k=0}^{N-1} \hat{x}(2k) e^{-j\frac{2\pi u2k}{2N}} \\ \hat{X}(u) = \frac{1}{2N} \sum_{n=0}^{2N-1} x(n) e^{-j\frac{2\pi 2un}{2N}} \\ \hat{X}(u) = a_0 + a_1 e^{j\theta_1} + ... + a_{2N-1} e^{j\theta_{2N-1}} \end{gathered} \tag{4.11}$$

We have then $\hat{x}(n) = x(\frac{1}{2}n) <=> \hat{X}(u) = X(2u)$ with Equation 4.11, which leads to the conclusion that the increase of spatial resolution will result in the compression in the frequency domain which is consistent with the property of Fourier Transform (FT).

Based on the findings in Equations 4.7 and 4.9, we assume the phase spectrum $X_p(u)$ and the amplitude spectrum $X_a(u)$ of the original images x(n):

$$\begin{gathered} X_a(u) = a_0 + a_1 e^{j\theta_1} + ... + a_k e^{j\theta_k} \\ X_p(u) = p_0 + p_1 e^{j\theta_1} + ... + p_{N-1} e^{j\theta_{N-1}} \end{gathered} \tag{4.12}$$

The corresponding up-sampling is:

$$\begin{gathered} X_a^{up}(u) = a_0 + a_1 e^{j\theta_1} + ... + a_k e^{j\theta_k} + a_N e^{j\theta_N} + a_{N+1} e^{j\theta_{N+1}} + ... + a_{N+k} e^{j\theta_{N+k}} \\ X_p^{up}(u) = p_0 + p_1 e^{j\theta_1} + ... + p_{2N-1} e^{j\theta_{2N-1}} \end{gathered} \tag{4.13}$$

Let $y_A(n)$ the output of a convolution layer and $Y_A(u)$ its DFT, we have :

$$y_A(n) = x(n) * c(n) <=> Y_A(u) = X_a(u) C(u) \tag{4.14}$$

We get then :

$$Y_A(u) = f_0 + f_1 e^{j\theta_1} + ... + f_{k+N-1} e^{j\theta_{k+N-1}} \tag{4.15}$$

The corresponding up sampling is :

$$Y_A^{up}(u) = f_0 + f_1 e^{j\theta_1} + ... + f_{k+N-1} e^{j\theta_{k+N-1}} + f_{2N-1} e^{j\theta_{2N-1}} + ... + f_{2N+k-1} e^{j\theta_{2N+k-1}} \tag{4.16}$$

First, inverse discrete Fourier transform (IDFT) is applied on the phase spectrum to obtain its spatial representation p(n), then it is concatenated with x(n) in the channel dimension to obtain the RGBP, the output of this result passed through a convolution is:

$$Y_{A+P}(u) = f_0 + f_1 e^{j\theta_1} + ... + f_{2N-2} e^{j\theta_{2N-2}} \tag{4.17}$$

The up-sampling result is then:

$$Y_{A+P}^{up}(u) = f_0 + f_1 e^{j\theta_1} + ... + f_{3N-2} e^{j\theta_{3N-2}} \tag{4.18}$$

Intuitively, the learned frequency components are N according to Equation 4.18 when leveraging the phase and the image together, but it is N-k when only using amplitude spectrum according to Equation 4.16.

CNN Architecture

For this one, they considered that the difference between pristine faces and fake ones is the local low-level features (eg. textures, colors) instead of global high-level semantic features such as the face or objects within an image since they share most of the last one. To this end, they dropped many convolutional layers from the Xception [50] backbone to reduce the receptive field of the network.

4.4.3 Loss Functions

Cross Entropy Loss

The only used loss function is the cross entropy loss which is logical considering the fact that we have a binary classifier on our hand. (See 4.3.3 for more details).

4.4.4 Reported Results of the Original Paper

The detector was trained on FaceForensics++, ADAM was used as an optimizer with a learning rate of 0.002, the Xception blocks they retained were block1-3 and block 12. The results are in Table 4.3.

method	ACC (c23)	AUC (c23)	ACC (c40)	AUC (c40)	AUC (CDF2)
SPSL	91.5%	95.33%	81.57%	82.82%	76.88%

Table 4.3: Paper results of SPSL.

4.5 Consistent Representation Learning (CORE)

This spatial detector was introduced in 2022 by Ni et al. [54] to enhance the consistency in deepfake detection by constraining the distance between different augmentations of the same image.

4.5.1 Overview

This detector uses erasing-based augmentations to capture more general forgery features to avoid overfitting. To capture more intrinsic forgery representations, they enforced the consistency between different augmentations of the same image using a regularisation method.

First, a pair of views are obtained using random data augmentation. Then, an encoder network extracts the feature representations, a consistency loss is computed between the two to ensure the consistency. Finally, a classifier Network assigns the supervised label for each representation.

4.5.2 Pipeline

As discussed earlier, this framework includes data augmentation, a most used deep learning technique when faced with underfitting and overfitting issues, this is considered as a preprocessing step, it generates 2N images for an input of N images.

Since the method also relies on feature extraction to differentiate between pristine and fake faces, an encoder is used, which is basically a CNN backbone, for the same image there are two different views sharing the same encoder network.

A consistency loss is adopted to ensure the model learning intrinsic features, and finally a classifier network which is simply a feed forward network is used for the binary classification.

As illustrated in Figure 4.5, the pipeline consists of:

Data Augmentation

Given a set of transformations T, and an input image x, two transformations t_1 and t_2 are randomly chosen from T, we then get two view of the input image x as in 4.19 to serve as inputs for the network.

$$x_1 = t_1(x), x_2 = t_2(x) \tag{4.19}$$

The augmentations implemented are:

- RaAug: Which consists of:
  - Random Erasing : With the following parameters:
    - Scale fact (0.05,0.2)
    - Aspect ratio (0.5,0.2)
    - Probability 1/3
  - Random Resized Crop : With the parameters:
    - Scale fact (1/1.3,1)
    - Aspect ratio (0.9,1.1)
    - Probability 1/3
  - No augmentation : With a probability of 1/3
- DFDC-Selim: Which includes: Quality compression, Gaussian noise, Gaussian blur, Random shift, Random scale.

Encoder Network

Xception [50] was used as the encoder network f, it maps the two views x1 and x2 into two d-dimensional representation vectors as follows:

$$f_1 = f(x_1); f_2 = f(x_2) \tag{4.20}$$

These two representation vectors are fed into the classification network and the consistency loss computation (Section 4.5.3) as shown in Figure 4.5.

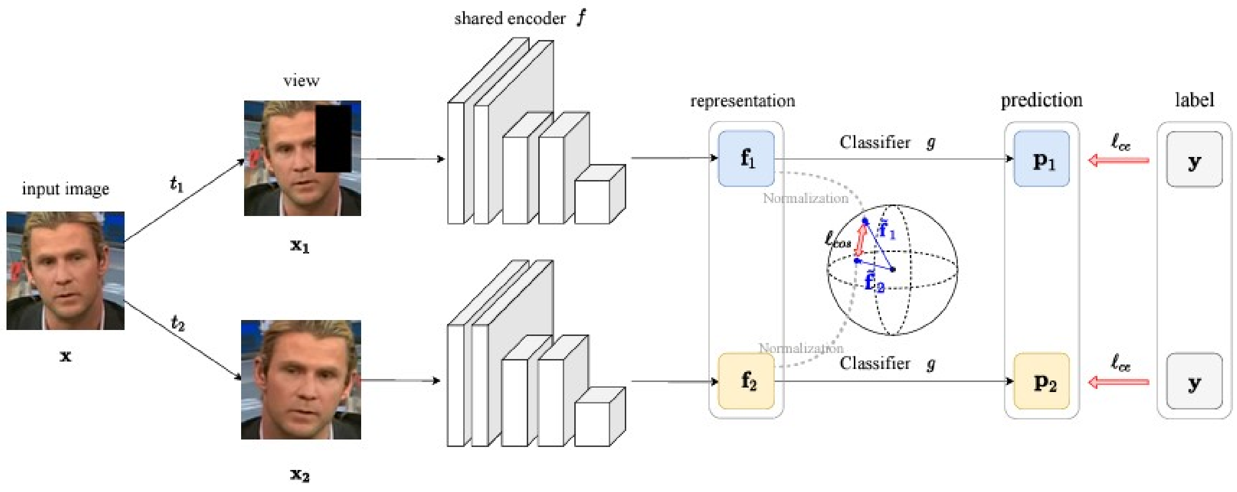


Figure 4.5: CORE pipeline and architecture [54].

Classification Network

The classifier denoted as g is a feed forward neural network containing a linear layer and a Softmax normalization layer to map the representation vector into a probability p = g(f) to draw a conclusion on the fakeness of the input.

4.5.3 Loss Functions

Consistency Loss

The consistency loss is used to penalize the distances of the representation vectors that are extracted from different views of the same original image [54]. For that, they adopted cosine similarity loss as follows :

$$l_{cos}(f_1^{(n)}, f_2^{(n)}) = (1 - \tilde{f}_1^{(n)} \tilde{f}_2^{(n)})^2 \tag{4.21}$$

With $\tilde{f} = \frac{f}{||f||^2}$ is the normalized vector of the representation vector f.

As illustrated in Figure 4.5, the representation vector are first normalized by an L2 normalisation layer before passed to the consistency regularisation. For N pairs of input images, the loss becomes :

$$L_c = \sum_{n=1}^{N} l_{cos}(f_1^{(n)}, f_2^{(n)}) = \sum_{n=1}^{N} (1 - \tilde{f}_1^{(n)} \tilde{f}_2^{(n)})^2 \tag{4.22}$$

The reason they chose cosine similarity loss is because it only pulls the angles of the vectors to be similar, ignoring the norms. The reason behind it is that they did not want the representations to be identically similar, since some views have parts of the image cut out, forcing a similarity might have a negative impact on the learning.

Classification Loss

For this, they used the standard cross-entropy loss since it is a binary classification. Since having two pairs of views for an image, the classification loss could be written as follows:

$$L_{ce} = \sum_{n=1}^{N} l_{ce}(p_1^{(n)}) + l_{ce}(p_2^{(n)}) \tag{4.23}$$

Overall Loss

The overall loss of the framework is the combination of the previous two, which can be formulated as follows:

$$L = L_{ce} + \alpha L_c \tag{4.24}$$

Where α is the balance weight for the two losses. Following some experiments, $\alpha = 2$ was the value that gave the best performance.

4.5.4 Reported Results of the Original Paper

The detector was trained of FaceForensics++ c23, ADAM was used as an optimizer with a learning rate of 0.0002, a mini-batch of 32 and an image size of 299x299, they adopted a weight of (4,1) for the cross-entropy loss, the training was on 30 epochs, with an early stopping if no gains are observed within 5 epochs.

Dataset	FF++ raw		FF++ c23		FF++ c40		CDF2	DFDCP	DFD
Metric	ACC	AUC	ACC	AUC	ACC	AUC	AUC	AUC	AUC
CORE	99.97%	100%	97.61%	99.66%	87.99%	90.61%	79.45%	75.74%	93.74%

Table 4.4: CORE paper results.

4.6 End-to-end Reconstruction Classification Learning (RECCE)

This detector was introduced in 2022 by Cao et al. [9] to learn the discrepancy between real and fake faces through a reconstruction classification network.

4.6.1 Overview

To face the two main issues of deepfake detection that are generalising to unknown forgeries and learning the essential information about forgeries and their clues, the proposed method has reconstruction-classification learning for both issues respectively.

For reconstruction learning an encoder-decoder network is used to learn representations of real faces instead of fake ones, since they they are more likely to be compact, the reconstruction has a loss that keeps real images close, and real from fake as far as possible. Therefore the network aims at a sound reconstruction of real images, and a poor one for fake images, because the discrepancy information which reveals forgery cues is progressively strengthened at the decoder side.

Then in the representation learning, bipartite graphs are used to model the feature relationship in order to reason about forgery clues present in the decoder features, in a multi-scale manner since it has multiple blocks each serving for a different characteristic.

Finally, the reconstruction difference indicates probable forged regions, on top of the feature map resulted from the representation module, the Reconstruction-guided attention uses these information to facilitate classification.

4.6.2 Pipeline

As discussed earlier and as it is illustrated in Figure 4.6, the architecture of the model contains 3 different modules that are:

Reconstruction Learning

This module aims at reconstructing real faces accurately, in order to avoid overfitting to specific forgery patterns. Given an input image $X \in \mathbb{R}^{h \times w \times 3}$ and the reconstruction network f which is based on an encoder decoder structure using Xception as its backbone.

In order to learn robust representation of the real faces, white noise was added to obtain $\tilde{X}$, thus the image reconstruction is:

$$\hat{X} = f(\tilde{X}) \tag{4.25}$$

During the training, the loss computed between input real images and their reconstructed counterparts is:

$$L_r = \frac{1}{|R|} \sum_{i \in R} ||\hat{X}_i - X_i||_1 \tag{4.26}$$

Where R is the set of real samples in a mini-batch and |R| is the cardinality of R.

They also used a metric-learning loss to make real images close and real-fake images faraway. **F** denotes the output features of an encoder or decoder bloc, they applied global average pooling to F to obtain the feature vector $\bar{F}$ for each input sample, the loss is then:

$$L_m = \frac{1}{N_{RR}} \sum_{i \in R, j \in R} d(\bar{F}_i, \bar{F}_j) - \frac{1}{N_{RF}} \sum_{i \in R, j \in F} d(\bar{F}_i, \bar{F}_j) \tag{4.27}$$

Where R and F denote the set of real and fake samples respectively. N_{RR} and N_{RF} are the total number of (real, real) pairs and (real,fake) pairs, respectively. d(· ; ·) is a pair-wise distant function based on the cosine distance:

$$d(a; b) = \frac{1 - \frac{a}{||a||_2} \cdot \frac{b}{||b||_2}}{2} \tag{4.28}$$

The first part of L_m encourages the representation of real faces to be compact, while the second part maximizes the difference between real and fake faces. The compactness of representation of fake data is not constrained since the differ greatly according to the forgery method. The metric-learning loss is applied to the output of the encoder block and to each one of the decoder blocks.

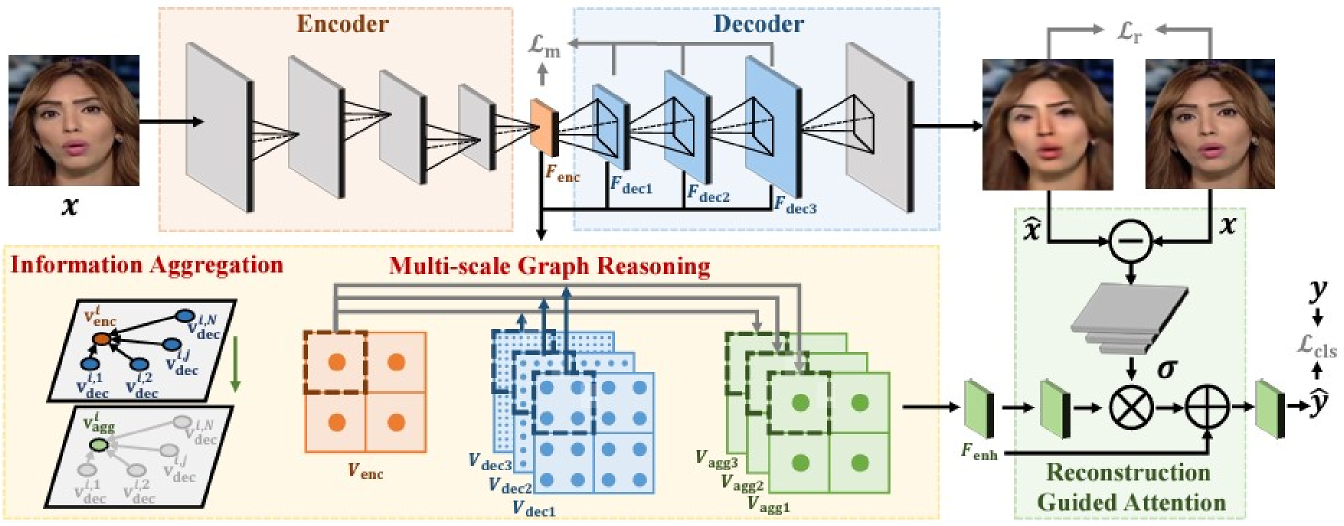


Figure 4.6: RECCE pipeline and architecture [9].

Multi Scale Graph Reasoning

Its a module that combines latent features of the decoder blocs and the encoder output into a bipartite graph to effectively exploit forgery clues captured by the decoder for the final classification.

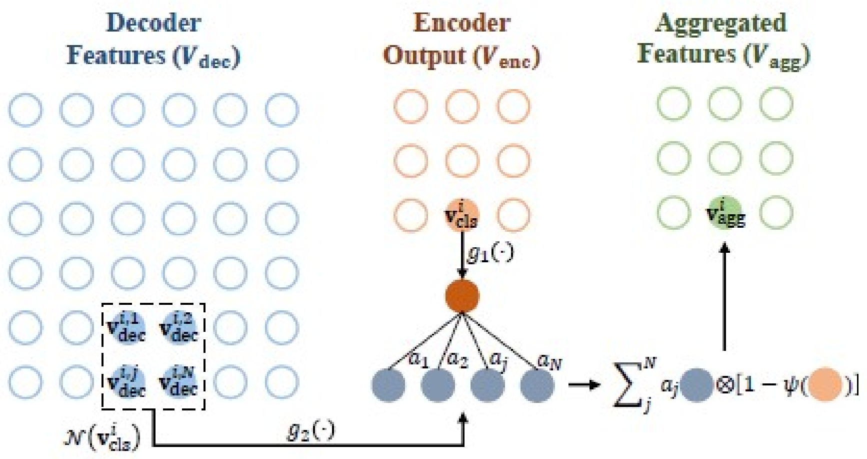

Decoder Features (V_{dec})
Encoder Output (V_{enc})
Aggregated Features (V_{agg})
v^i_{cls}
v^i_{agg}
$g_1(\cdot)$
$v^{i,1}_{dec}$ $v^{i,2}_{dec}$ $v^{i,j}_{dec}$ $v^{i,N}_{dec}$
a_1 a_2 a_j a_N
$\sum_j^N a_j \otimes [1-\psi(\cdot)]$
$\mathcal{N}(v^i_{cls})$
$g_2(\cdot)$

Figure 4.7: Illustration of the proposed multi-scale graph reasoning, the figure is best viewed in color. [9]

Here, we take the feature maps of a decoder block for a given scale for description. As shown in Figure 4.7, we model the encoder output and the decoder features, i.e., $F_{\text{enc}}, F_{\text{dec}}$, as two vertex sets $V_{\text{enc}} = \{v_i^{\text{enc}}\}_{i=1}^{h_1\times w_1}$ and $V_{\text{dec}} = \{v_i^{\text{dec}}\}_{i=1}^{h_2\times w_2}$, where each vertex represents an embedding vector from the original feature maps. $N(v_i^{\text{enc}}) = \{v_i^{j,\text{dec}}\}_{j=1}^{N}$ denotes the set of vertices in V_{dec} which is linked to v_i^{enc}, and which would be aggregated together. N is the number of vertices in the set [9].

As shown in Figure 4.7, the neighborhood of the orange solid vertex is the blue solid vertices in the dotted box. Given $v_i^{\text{enc}}, v_i^{j,\text{dec}}$, they are first projected to a shared embedding space with two neural nets, $g_1(\cdot)$ and $g_2(\cdot)$, to get $\tilde{v}_i^{\text{enc}}, \tilde{v}_i^{j,\text{dec}}$, respectively.

Next, weight coefficient a_j is computed to indicate the importance of $v_i^{j,\text{dec}}$ to v_i^{enc} as follows:

$$a_j = \frac{\exp(\phi(\tilde{v}_i^{\text{enc}} \| \tilde{v}_i^{j,\text{dec}}))}{\sum_{v_i^{l,\text{dec}} \in N(v_i^{\text{enc}})} \exp(\phi(\tilde{v}_i^{\text{enc}} \| \tilde{v}_i^{l,\text{dec}}))} \tag{4.29}$$

Where $\|$ denotes the concatenation operation and ϕ is a single-layer network. Then, they compute a $[0, 1]$-valued vector based on v_i^{enc} using a non-linear transformation $\sigma(\cdot)$ to generate a feature richness measurement for $\tilde{v}_i^{\text{enc}}$ in the channel level. During information aggregation, they particularly enhanced the channels of $\tilde{v}_i^{j,\text{dec}}$ when the weight of the corresponding channels of $\tilde{v}_i^{\text{enc}}$ is small. The aggregated feature vector v_i^{agg} is computed by:

$$v_i^{\text{agg}} = \sum_{j=1}^{N} a_j \tilde{v}_i^{j,\text{dec}} \odot (1 - \sigma(v_i^{\text{enc}})) \tag{4.30}$$

Where $\odot$ is the element-wise multiplication.

The multi-scale aggregation is due to the fact that forgery traces could be found in different scales. The aggregated features $\{v_i^{\text{agg}}\}$ in different scales are concatenated with

v_i^{enc} and then pass through a sigmoid function followed by two fully-connected layers to produce the enhanced feature vector v_i^{enh} with the same channel dimension as v_i^{enc}. Finally, v_i^{enh} are assembled spatially to obtain the enhanced feature maps F_{enh} for the following reconstruction guided attention.

The previous passage was strongly based on [9].

Reconstruction Guided Attention

This module shifts its focus to the difference between reconstructed forged faces and input forged faces which they hypothesized it indicates the forged regions, all of which to facilitate classification.

As illustrated in Figure 4.6, given the reconstructed image $\hat{X}$ and the original image X, the pixel-wise difference is calculated to obtain the difference mask m:

$$m = |\hat{X} - X| \tag{4.31}$$

Where $|\cdot|$ denotes the absolute value function. Given F_{enh}, the enhanced feature map mentioned in the last Section, an attention map is computed based on the difference mask and applied spatially to F_{enh} to obtain F'_{enh}. Subsequently, F'_{enh} is added to F_{enh} to produce the attended output features:

$$F'_{\text{enh}} = \sigma(f_1(m)) \odot f_2(F_{\text{enh}}) \tag{4.32}$$

$$F_{\text{att}} = F'_{\text{enh}} + F_{\text{enh}} \tag{4.33}$$

Where f_1 and f_2 denote convolutional operations, σ is the sigmoid function, and $\odot$ represents element-wise multiplication. F_{att} serves as an input to the classifier network.

4.6.3 Loss Functions

The overall loss of the method includes the reconstruction loss from Equation 4.26 and the metric-learning loss from Equation 4.27 on top of the cross-entropy loss L_{cls} (Equation 4.4) since it is a binary classification, we obtain:

$$L = L_{cls} + \lambda_1 L_r + \lambda_2 L_m \tag{4.34}$$

Where λ_1 and λ_2 are weight parameters for balancing different losses.

4.6.4 Reported Results of the Original Paper

The detectors was trained on FaceForensics++ with a batch size of 32, ADAM was used as an optimizer with a learning rate of 0.0002 and a weight decay of $1e^{-5}$. λ_1 and λ_2 from 4.34 are empirically set to 0.1. Results are reported in table 4.5.

Dataset	FF++ c23		FF++ c40		CDF2	WDF	DFDC
Metric	ACC	AUC	ACC	AUC	AUC	AUC	AUC
RECC	97.06%	99.32%	91.03%	95.02%	68.71%	64.31%	69.06%

Table 4.5: RECCE paper results.

4.7 Self Blended Images (SBI)

This detector was introduced in 2022 by Shiohara et al. [12] where a state of the art preprocessing method was introduced to try and imitate the common forgeries presented in deepfakes to produce a more challenging data.

4.7.1 Overview

Since most detectors suffer from overfitting and lack generalisation ability, this method comes in play to mitigate that problem.

The method aims at creating new synthesized data called SBIs to try and imitate the common forgery patterns and traces in deepfakes, this data being harder for detectors, its goal is to make more robust and general detectors.

To do so, a pipeline of three main modules is set which contains the Source-Target Generator who's role is to generate the pseudo target and source images from a single pristine image, then there is the mask generator that is going to generate the blending mask of the previous two images, and finally the blending module to create the SBI. This framework is flexible and could be integrated with any given detector.

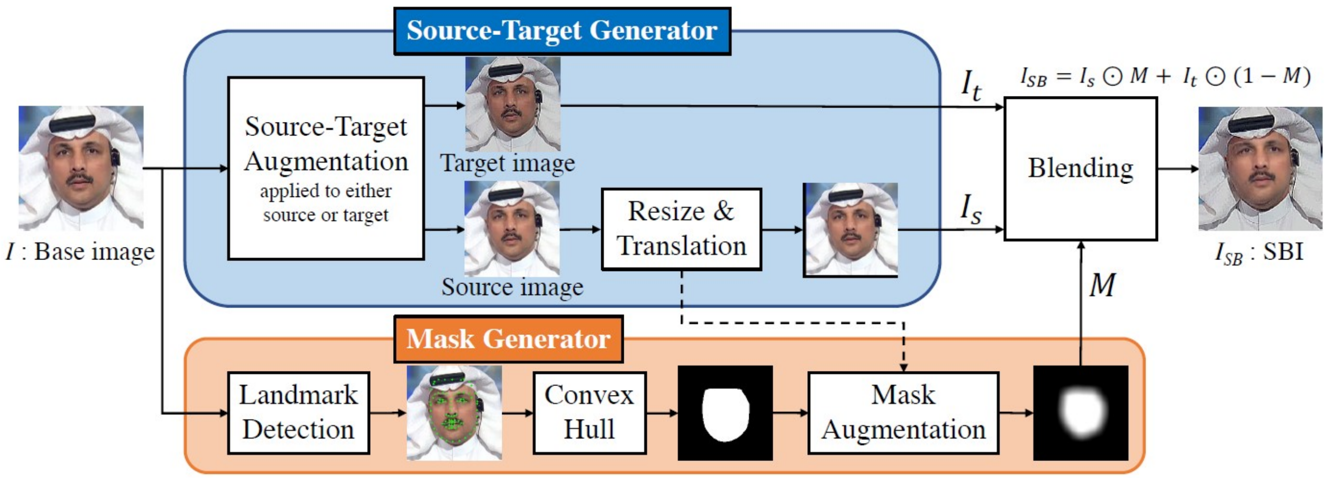


Figure 4.8: SBI generation architecture [12].

4.7.2 Pipeline

To generate hardly recognisable data to push the detectors to learn more general patterns, SBIs were proposed as illustrated in Figure 4.8, the pipeline consists of Source-target generator, mask generator, and a blending module, before feeding the image to the CNN network which was EfficientNet [51] in the original paper. The main parts of the architecture are:

Source-Target Generator (STG)

It first creates two copies of the input image, then some transformations are randomly applied to one of the images to make the target image such as: Shifting the values of RGB channels, hue, saturation, value, brightness, and contrast as color transformations, as for the frequency transformations they either downsample or sharpen the input. This step is crucial to generate statistical inconsistencies.

Then to reproduce blending boundaries and landmark mismatches, the source image (the one that has not been modified) is resized following two parameter u_h and u_w that are sampled from a uniform distribution $[u_{min}, u_{max}]$, this image is then zero padded or center-cropped to have the same size as the original and then translated following a vector t=$[t_h, t_w]$.

Mask Generator (MG)

This module provides a gray scale mask to blend the source and target images. In order to obtain it, facial landmarks are calculated for the input image and the mask is initialized by calculating convex hull from the predicted landmarks.

Then, the mask is deformed by elastic deformation before being smoothed by two Gaussian filters. After the first smoothing the pixel values less than 1 are changed to 0, this means that the mask is eroded if the kernel size of the first Gaussian filter is larger than that of the second one and is dilated in the opposite case.

Finally, MG varies the blending ratio of the source image. This can be achieved by multiplying the mask image by a constant r that is uniformly sampled from $\{0.25,0.5,0.75,1,1,1\}$.

Blending

The source image I_s and the target image I_t are blended to obtain I_{SB} as follows :

$$I_{SB} = I_s \odot M + I_t \odot (1 - M) \tag{4.35}$$

Some examples of SBIs and their pristine images are illustrated in Figure 4.9.

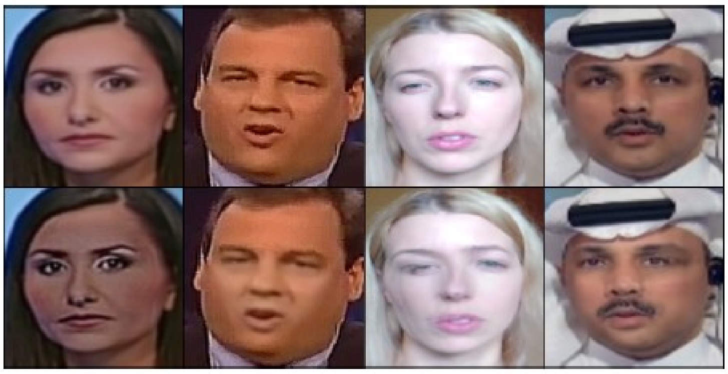


Figure 4.9: Pristine images on top and their SBI on the bottom [12].

4.7.3 Loss Functions

Cross Entropy Loss

Since we have a binary classification after the CNN and the SBI generation, the suitable loss function is the cross entropy (See Section 4.3.3 for details).

4.7.4 Reported Results of the Original Paper

The detector is trained on real images from FaceForensics++ and their SBIs for 100 epochs with a batch size of 32 and a learning rate of 0.001, SAM was used as an optimizer, and only 8 frames were used per video. They also used some data augmentations, i.e., Image Compression, RGB Shift, Hue Saturation Value, and Random Brightness Contrast. The results are reported in table 4.6.

Dataset	FF++	CDF2	DFD	DFDC	DFDCP	FFIW
SBI	99.64%	93.18%	97.56%	72.42%	86.15%	84.83%

Table 4.6: SBI paper results.

4.8 Lip Forgery Detection (LipFD)

A multi-modal LipSync detector that was proposed by Liu et al. [13] in 2024 to tackle the shortcoming of LipSync based deepfake detection.

4.8.1 Overview

This novel deepfake detector concentrates on LipSync videos, a forgery type that is highly overlooked in this field, making existing works unreliable when it comes to these forgeries. The core idea behind the method is to capture the discrepancy between lip movement and audio signal since these two have a strong temporal correlation, additionally, since the head region is correlated to the speech and lips, they also shifted the attention to its movement to serve as additional information.

4.8.2 Pipeline

The pipeline is illustrated in Figure 4.10, it consists of:

- **Global feature encoder**: Used to capture and encode the correlation and temporal features between lips and speech using a Transformer model ViT:L/14 [81], they integrate visual forgery traces from multiple scales of the head to the previous features.

  The convolution is conv, the image I is cropped into 3 series as $\{c_N^h, c_N^f, c_N^l\}_i$, $i \in \{0, ..., T-1\}$, where N is the batch size and T is the window size. Image I is embedded into F_G as global feature:

  $$F_G = V_T^i(\text{Conv}(I, \theta_{\text{Conv}})) \tag{4.36}$$

  $$\{c_N^h, c_N^f, c_N^l\}_i = \text{Crop}(I, \{1.0, 0.65, 0.45\}), \quad i \in \{0, 1, 2\} \tag{4.37}$$

  Where θ_{Conv} are the parameters of Conv. The encoder is constrained by L_{RA}.

- **Region awareness**: It is where the attention of the model is adjusted by using some weights for each component. For each crop $c \in \{c_N^h, c_N^f, c_N^l\}_i$, region feature is defined as $F_R = E_{GR}(c, F_G, \theta_{GR})$ and the weight is formulated as follows:

  $$_{c_i^j}\!= \text{RA}([F_G|\{F_R\}_i^j], \theta_{\text{RA}}), \quad c_j \in \{c_h, c_f, c_l\} \tag{4.38}$$

  Where c_i^j denotes the i-th feature in c_j and θ_{RA} is the parameters of the region awareness module. The final feature F is:

  $$F = \frac{1}{T} \cdot \frac{\sum_{i,j}(_{c_i^j} \cdot [F_G|\{F_R\}_i^j])}{\sum_{i,j} {}_{c_i^j}} \tag{4.39}$$

- **Forgery Detection**: The final feature is fed to a Multi-Layer Perceptron (MLP) [82] for binary classification.

4.8.3 Loss Functions

Region Awareness Loss

They noticed that the lip information is the most crucial for final classification, and all other information should be supplementary. Hence, this loss encourages the model to focus more on c_l, it is formulated as follows:

$$L_{RA}(\theta_{GR}, \theta_{RA}) = \sum_{j=1}^{N} \sum_{i=1}^{T} k \cdot \exp({}_i^{\text{max}} - {}_i^h) \tag{4.40}$$

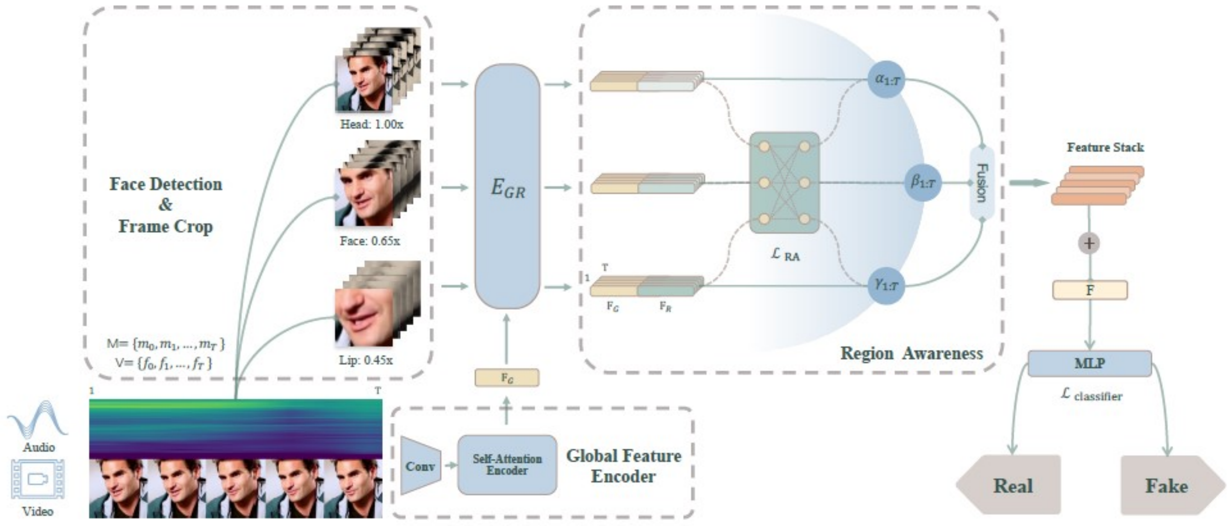


Figure 4.10: Pipeline of LipFD deepfake detector [13].

Where $^{\max}_{i}$ is the max weight in feature stacks, $^{h}_{i}$ is the non-cropped region. k is a hyper-parameter used to adjust the steepness of the loss.

Classification Loss

They used the binary cross entropy loss, the most suitable for the binary classification. (See Section 4.3.3).

4.8.4 Reported Results of the Original Paper

The detector was trained on a private dataset named LRS2 that only its preprocessed validation set has been released for result reproduction, the dataset included deepfakes generated by: Wav2Lip [7], MakeItTalk [83], and TalkLip [43]. The results are reported in Table 4.7.

Dataset	LRS2		FF++		DFDC	
Metric	ACC	AP	ACC	AP	ACC	AP
LipFD	95.27%	93.08%	95.10%	76.98%	94.53%	78.61%

Table 4.7: LipFD paper results.

4.9 Choice of Datasets

Table 4.8 gives a general overview on the datasets studied during the project. The colored cells represent the datasets used in the experiments.

dataset	Real videos	Fake videos	Year	Forgery methods	Average duration	Modality	Advantages	Limitations
UADFV	49	49	2018	1	11.4s	visual	Early release	Low data amount
FF++	1000	4000	2019	4	18s	visual	Multiple methods	Visible artifacts
DFD	363	3068	2019	5	12s	visual	High quality, Multiple methods	Low data amount, Only face swapping
DFDCP	1131	4119	2019	2	30s	visual	Challenging videos	Only face swapping
CDF1	408	795	2019	1	13s	visual	realistic manipulation	Low data amount
DFDC	23654	104500	2020	8	30s	visual	Various techniques	Quality discrepancy
CDF2	590	5639	2020	1	13s	visual	realistic manipulation	Only face swap, Low data amount
FSh	1000	1000	2020	1	20s	visual	Highest faceswap quality	Low data amount
DF-1.0	50000	10000	2020	1	-	visual	Large scale dataset	Only one forgery method
WDF	3805	3509	2021	-	-	visual	Realistic videos	Unknown manipulations
FAVC	500	19500	2021	4	7.8s	Audio-visual	Includes Lip-Sync	Only one Lip-Sync method

Table 4.8: Overview of studied datasets.

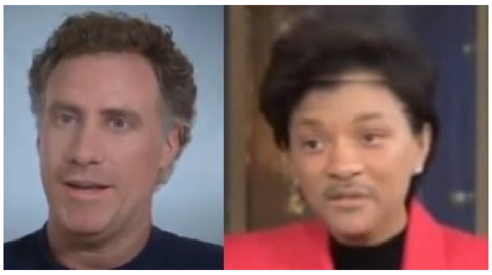
The criteria for the choice is:

- First and foremost, the goal was to obtain the maximum amount of data, to assess the deepfake detection method thoroughly, high amount of data means higher chance to assess the generalisation ability.
- The computational cost was a factor, as very large datasets like DFDC or DeeperForensics were almost impossible to get for two reasons: 1)- Limited Download ability of the Wifi 2)- Limit of the machine used would have been exceeded with either one. For DFDC we settled with DFDCP the smaller version, and for DeeperForensics-1.0 we settled with DFD and CDF1 and CDF2 for having nearly the same forgery method and quality.
- Having the most amount of forgery methods was also a criterion, on top of having the highest quality videos, which excluded UADFV for not having both.

- FaceForensics++ although having visible artifacts in many of its samples is a staple in deepfake detection, as per the general consensus in this field, it is the dataset **almost all detectors (for the exception of those training on private dataset) train with.**
- For wild-Deepfake, we tried to acquire the dataset, but the authors never sent the download link, even after several attempts at filling the agreement form.
- Other datasets exist, but we settled with the most used, and the high quality ones.

4.10 Dataset Analysis

In order to understand the performance of detectors in the following chapters, we ought to study these datasets and the artefacts they possess. For that, we have sampled videos from each dataset to debug the artefacts within, results found are reported in Table 4.9 supported by Figures 4.11, 4.12 and 4.13.


(a) CDF1

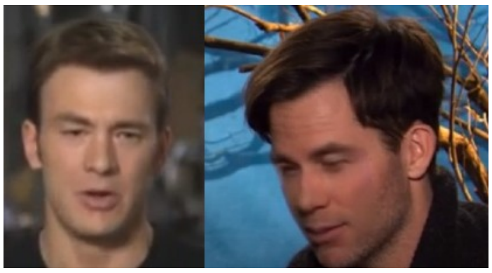
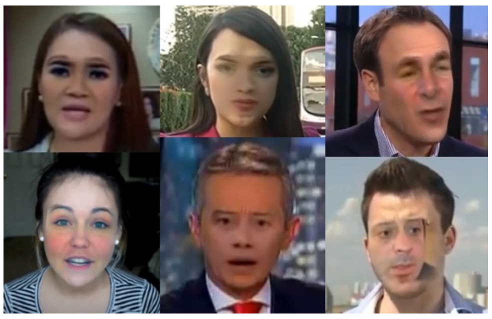

(b) CDF2

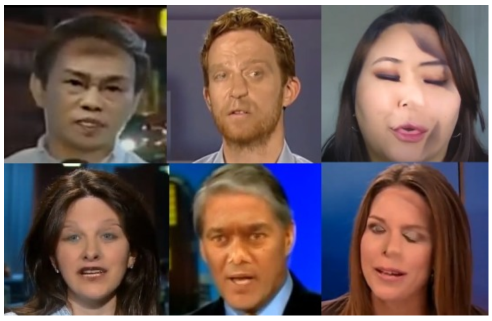
Figure 4.11: Artefacts of CDF datasets.

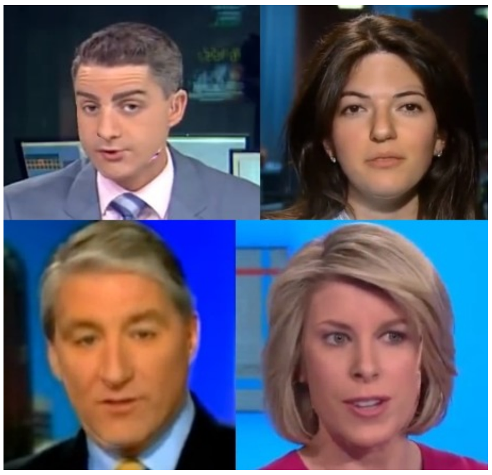
Dataset		Artefacts
FF++	FF-DF (4.12a)	Visible splicing boundaries (row 1 col 2), blending artefacts, low quality synthesized faces (row 1 col 3), synthesized face orientation (row 2 col 3), temporal jitter and flickering (row 2 col 1), visible parts of original face (row 1 col 1).
	FF-FS (4.12b)	Temporal jitter and flickering, distortions (row 1 col 2 & 3), color mismatch (row 1 col 1), unnatural physical signals (row 2 col 3), visible parts of original face (row 2 col 1).
	FF-F2F (4.12c)	Movement of facial edges (row 2 col 2), inconsistent physical signals (row 2 col 1), visible parts of original face (row 1 col 2).
	FF-NT (4.12d)	Artefacts on the mouth region (distortions and unnatural movements)
CDF1 & CDF2 (4.11)		Mild facial edges movement and blur to hide blending artefacts
DFD (4.13a)		Blending artefacts (row 3 col 2), visible splicing boundaries (row 2 col 2), color mismatch (row 2 col 2), landmark mismatch (row 2 col 1), movement of facial edges (row 3 col 1), temporal flickering (row 1 col 2)
FSh (4.13b)		Visible parts of original face (row 1 col 3 and row 2 col 1 & 3), facial edges movement (row 1 & 2 col 2), unnatural physical signals (row 1 col 1)
DFDCP (4.13c)		Blur to hide artefacts, face distortions, facial edges movement

Table 4.9: Artefacts present on datasets.

The number in front of the dataset in the Table represents the Figure number illustrating the mentioned artefacts.


(a) FF-DF


(b) FF-FS


(c) FF-F2F

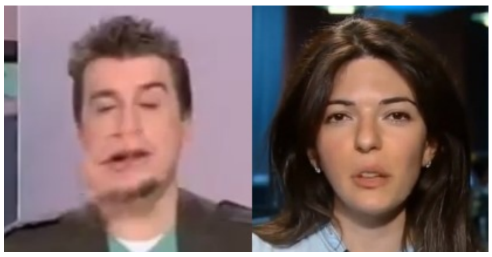
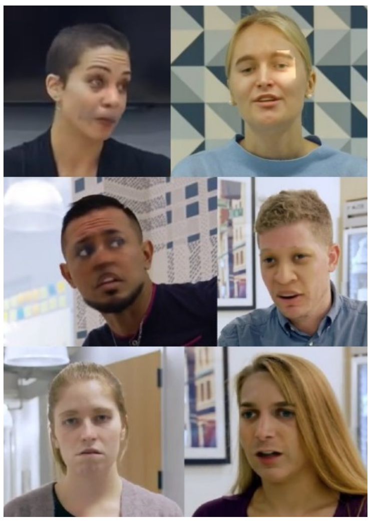
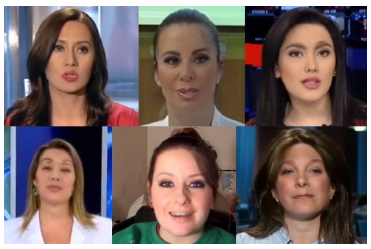

(d) FF-NT

Figure 4.12: Artefacts of FF++ subsets.


(a) DFD


(b) FSh

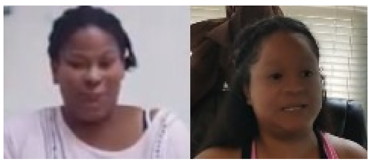

(c) DFDCP

Figure 4.13: Artefacts of remaining datasets.

4.11 Choice of Metrics

Table 4.10 summarizes the most important and used metrics in deepfake detection by order of importance:

Metrics	Interpretation
Area under the curve (AUC)	Percentage of correctly identifying deepfakes while minimising identifying real videos as deepfakes
Accuracy (ACC)	Percentage of correct predicitions
Average precision (AP)	Percentage of correct positive predictions (identifying deepfakes)
Equal error rate (EER)	Threshold at which FPR and FNR are the same (identifying real videos as deepfakes and deepfakes as real videos)

Table 4.10: Deepfake detection metrics.

4.12 Conclusion

To conclude this part of our project, we have ended with the deepfake detectors studied, which has included two frequency detectors in F3Net and SPSL, two spatial detectors in CORE and RECCE, the groundbreaking SBI method, and a LipSync detector in LipFD, their understanding should help us in the upcoming chapters, where we will be setting a baseline and assessing these methods using the datasets analysed and evaluation metrics chosen.

Chapter 5

Deepfake Detection Evaluation: Experimental Results

5.1 Introduction

In this chapter, we finally get to the practical part of our project, where we assess the selected unimodal deepfake detectors that are presented in Chapter 4 following the evaluation methodology presented on the same Chapter.

First, we present the experimental setup used for all our upcoming experiments, from the hardware to the software that helped us get through our work. Next, the preprocessing step has been described, which helps prepare the data to be fed to the detection algorithms chosen for either training or testing.

Then, we have evaluated the first four unimodal deepfake detectors from Table 4.1 in Chapter 4: F3Net, SPSL, CORE, and RECCE on the datasets available, which has enabled us to successfully evaluate their overall performances on unseen data during their training, on top of a fair comparison between the detectors and between our obtained results with those reported in the original papers, from which we noticed a generalisation problem we have focused on in Chapter 6.

Finally, as for the fifth detector from Table 4.1: SBI, it was discovered later. Therefore, it was not included in the evaluation, but only in the generalisation study in Chapter 6 because of its high performance. The sixth detector from Table 4.1 was not included also because of the lack of material.

This chapter serves as a stepping stone for the next one, helping us understand the behaviour of deepfake detectors in order to effectively address some of their shortcomings, on top of setting a baseline for us to compare the results to.

5.2 Experimental Setup

5.2.1 Hardware

For our experiments we used a workstation with 256GB of SSD, 1.8TB of HDD, 32GB of RAM and a xeon processor.

The GPU we used is a Geforce RTX 3060 with a VRAM of 12GB, paired with a 12 GB GDDR6 memory, the operating frequency is 1320 MHz, which could be boosted up to 1770 MHz.

5.2.2 Software

For the implementations, source codes are provided later, the framework used was **PyTorch 2.4** with **Python 3.10** and **CUDA 12.1** to run the GPU, the codes were ran on an Ubuntu 22.04.4 LTS Jammy.

5.3 Preprocessing of Input Data

The overall preprocessing of deepfake detection follows a standard procedure, with few adjustments made for each method, and some slight changes of models used, but the general and complete preprocessing pipeline is the following:

5.3.1 Frame Sampling

The first step in the preprocessing pipeline is the frame sampling of videos, this is a crucial step as studies proved it to be an important factor for the performance of models.

There are two modes of sampling: **Fixed number of samples** which would apply on all videos no matter its Frames Per Second (FPS). The second mode is **Fixed stride** where the sampling rate or frequency would be the same. The first mode is the used one in this field since it is considered less expensive computationally. **We fixed the number of frames sampled per video to 32.**

5.3.2 Face Detection

The next step is to detect the faces on each video/frame, this is either done using a Dlib library that provides a face detection algorithm, or using Retina face, both identify a bounding box enclosing the face.

5.3.3 Landmark Extraction

Extracting facial landmarks provides valuable information about its geometry and structure, it also provides the information for the face alignment process. These landmarks are special points on the face such as the nose, mouth, jawline, eyes... The model used is a pre-trained shape predictor.

5.3.4 Face Alignment

Once the faces are detected, the alignment is the process to transform the images to a standardized pose. To align the faces, an affine transformation is used, which is a linear mapping that preserves the shape of the face. The transformation is estimated based on the detected landmarks and a set of target landmarks.

5.3.5 Face Cropping

The cropping is necessary to only keep the area of interest which is the face, therefore the model will not concentrate on the background or uninteresting details, but on only relevant features to extract the inconsistencies of the deepfake generation. **Margin** is a parameter that determines the area around the aligned face to be included in the cropped image, which was fixed to 1.3 to capture the most relevant information.

5.3.6 Mask Extraction

This step is optional and only includes the datasets that provide masks, which is obtained using convex hull applied on the face, there are also some algorithms that rely on these masks such as Face X-ray [56] and SBI [12] for blending faces. The mask could also refer to the forged regions of the face, an information provided by the dataset authors.

5.3.7 Saving Data

After completing the preprocessing steps, the landmarks, frames and masks (if available) are all saved into a structured directory format.

5.3.8 Arrangement

This step is optional, it is motivated by the need for a unified and convenient way to load different datasets. Each dataset typically has its own distinct structure and organization, making it hard to handle them uniformly.

To that end, they used a unified approach by managing the dataset information into a **JSON file**. The rearranged structure organizes the data in a hierarchical manner, grouping videos based on their labels and data splits (i.e.,train, test, validation). Each video is represented as a dictionary entry containing relevant metadata, including file paths, labels, compression levels (if applicable).

A visual representation of the preprocessing step is presented in Figure 5.1.

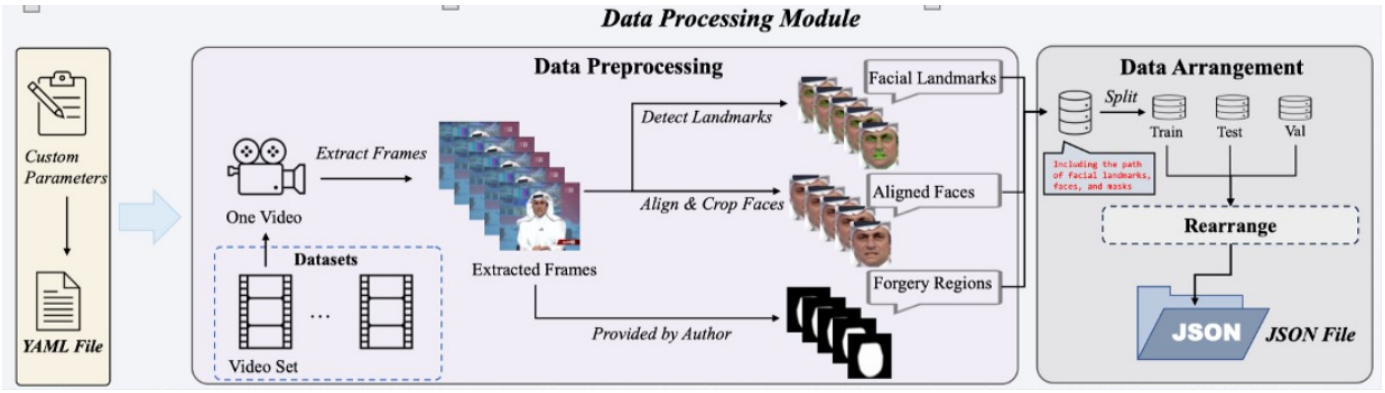


Figure 5.1: Preprocessing and data arrangement pipeline [53].

5.4 Implementation of Selected Deepfake Detectors

5.4.1 Source Codes

Our codes and work were based on Table 5.1:

Detector	Code	Published
F3Net [58]	https://github.com/yyk-wew/F3Net	ICCV 2020
SPSL [55]	https://github.com/SCLBD/DeepfakeBench	CVPR 2021
CORE [54]	https://github.com/niyunsheng/CORE	CVPRW 2021
RECCE [9]	https://github.com/VISION-SJTU/RECCE	CVPR 2022
SBI [12]	https://github.com/mapooon/SelfBlendedImages	CVPR 2022
LipFD [13]	https://github.com/AaronComo/LipFD	2024

Table 5.1: Source codes.

5.4.2 Baseline

For this section, we will be re-implementing the detectors and reproducing the paper' results, on top of testing on new datasets for some of them. Tables 5.2 through 5.5 highlight the implementation results on all 4 metrics.

DFDCP [74] was not included due to the later acquirement of the dataset.

- **ACC**

Detector	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
CORE [54]	90.42%	89.19%	83.47%	88.95%	84.7%	66.08%	70.5%	80.11%	54.23%
RECCE [9]	90.61%	**91.48%**	84.27%	**90.79%**	**88.1%**	67.57%	66.58%	82.41%	53.59%
SPSL [55]	85.58%	67.77%	65.73%	67.9%	66.59%	**68.03%**	**71.24%**	**88.98%**	**57.53%**
F3Net [58]	**90.76%**	88.57%	**84.46%**	88.56%	84.18%	67.15%	59.08%	68.47%	48.67%

Table 5.2: Accuracy of implemented methods.

- **AUC**

Detector	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
CORE [54]	95.82%	98.4%	94.53%	98.03%	92.31%	71.83%	74.11%	84.8%	56.48%
RECCE [9]	96.23%	98.32%	94.35%	97.46%	**94.77%**	73.16%	**74.18%**	**84.79%**	56.18%
SPSL [55]	93.1%	97.72%	89.02%	99.23%	86.45%	**80.35%**	73.15%	82.69%	**63.04%**
F3Net [58]	**96.53%**	**98.56%**	**94.82%**	**98.59%**	94.14%	75.35%	72.98%	79.42%	51.94%

Table 5.3: Frame-level Area Under the Curve for implemented methods.

- **EER**

Detector	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
CORE [54]	0.1058	0.05	0.1607	0.058	0.1337	0.3423	0.3217	**0.2352**	0.4631
RECCE [9]	0.0988	0.0555	0.1578	0.0696	**0.1073**	0.3271	**0.3204**	0.25	0.4698
SPSL [55]	0.1377	0.0698	0.1904	0.0379	0.2	**0.261**	0.335	0.2544	**0.4051**
F3Net [58]	**0.0982**	**0.0437**	**0.1524**	**0.036**	0.1283	0.3178	0.3323	0.2831	0.4897

Table 5.4: Equal Error Rate of implemented methods.

- **AP**

Detector	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
CORE [54]	98.93%	98.73%	95.01%	98.24%	93.23%	81.42%	**82.94**%	**97.86**%	55.5%
RECCE [9]	99.06%	98.62%	95.01%	97.79%	**95.85**%	81.64%	82.92%	97.52%	55.03%
SPSL [55]	98.22%	98.01%	91.47%	99.33%	82.19%	**88.38**%	82.53%	97.49%	**61.08**%
F3Net [58]	**99.14**%	**98.88**%	**95.33**%	**98.95**%	95.31%	83.5%	81.94%	97.01%	51.62%

Table 5.5: Average Precision of implemented methods.

Discussion about the Performance of Detectors

For what comes next, we divide our data into two main categories: Within-domain which includes the dataset our model have been trained on, which in this case is FaceForensics++ [69], and the cross-domain which includes all the other datasets that we test our models on.

- We can clearly see that all detectors perform well on the within-domain dataset, and on all four generation algorithms as well, with Neural Textures [26] being the most challenging for most detectors.
- In the cross-domain evaluation, we can notice a significant drop of performance in comparison to the within-domain one, and that is noticeable for all detectors.
- FaceShifter causes the lowest performances, which can be traced back to it having not only a different forgery method than all other datasets, but also different artefacts than all of them. (This claim is supported later on in the study in 6.2.1).
- The phenomenon we are witnessing in the performance drop suggest an overfitting of these methods to a dataset type, furthermore it can be considered as a lack of generalisation to unseen manipulations (as the performance on FaceShifter [40] suggests), and to even a higher quality deepfakes (as the performances on CDF1 and CDF2 [60] suggests).
- To support the claim of CDF datasets having a higher quality than FF++, we present some examples from both datasets knowing that FF++ contains a subset that has deepfakes method [22], and CDF datasets are based on an advanced version of deepfakes [22], which means both have the same forgery technique, Figures 5.2 and 5.3 illustrate some examples from FF-DF and CDF respectively.

When analysing the samples, we can clearly notice the quality discrepancy between the FF-DF samples and CDF ones, this difference forces our detectors to learn some forgery patterns that are not necessarily present in other datasets (splicing boundaries, low quality deepfakes and inconsistent face orientation respectively in Figure 5.2), which justifies the performance drop we saw in Tables 5.2 to 5.5.

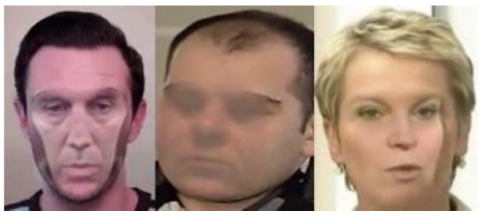


Figure 5.2: Visible artifacts in the FF-DF subset. [60]

Figure 5.3: Samples from Celeb-DF datasets. [84]

Comparison between Detectors

When analysing the results we got in Tables 5.2 through 5.5, we can compare between the detectors studied in all datasets as follows:

- **FF++ [69]**: We got F3Net [58] as the best detectors according to all 4 metrics, which is considered a logical result since this detector was only tested on FF++ on the original paper, making it more suitable since the authors probably optimized the network for it. However, RECCE [9] has the better classification ability on most manipulations according to the accuracy reported in Table 5.2.
- **CDF1 and CDF2 [60]**: The best classifier is SPSL according to the accuracy in Table 5.2, while SPSL [55] and RECCE are the best overall detector and the most reliable on CDF1 and CDF2 according to the AUC and the EER in Table 5.3 and Table 5.4 respectively. Additionally, The best deepfake detectors out of only deepfakes on CDF1 and CDF2 are SPSL and CORE [54] respectively according to Table 5.5.
- **DFD [73]**: The best classifier is SPSL according to Table 5.2, while RECCE is the best overall detector according to Table 5.3. However, The most reliable and the best deepfake detector is CORE according to Tables 5.4 and 5.5, with RECCE being a close second.
- **Face Shifter [40]**: The best detector on all four metrics is SPSL, while CORE and RECCE have similar results.

The question now is, why did our detectors perform the way they did? The answer lies in both the method and the data. For instance, F3Net was only tested on FF++ in the original paper as seen in Table 5.6, which makes it the most suitable and the top contender. Similarly, SPSL and CORE have both high performances on CDF and CDF&DFD respectively, since they were included in the papers, making them optimized for this type of dataset. Although RECCE has not had a strong performance in generalisation in its original paper, it exceeded it in our implementation, showing signs of improvement.

On the other hand, all datasets have up-sampling while generating the fake samples, which make the results from SPSL justified, because no matter the quality of videos, they still could capture the artefacts left by up-sampling, which is the case for the Face Shifter results, where this dataset has many samples that lack visual artifacts present in FF++, which made detectors like CORE and even RECCE fail even though they are considered generalisation algorithms.

5.4.3 Comparison with the Results Reported in Original Papers

Previously, we have set the baseline for each method, now we compare the results obtained with those reported in the original papers, Tables 5.6 through 5.9 highlight the results for all 4 detectors based on the highly used frame-level AUC metric.

The underlined values means they are lower than those reported in the benchmark paper by Yan et al. [53].

- **F3Net** [58]

F3Net	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
Paper	98.1%	-	-	-	-	-	-	-	-
Benchmark	96.35%	97.93%	97.96%	98.44%	93.54%	77.69%	73.52%	79.75%	59.14%
Implementation	96.53%	98.56%	94.82%	98.59%	94.14%	75.35%	72.98%	79.42%	51.94%

Table 5.6: Comparison of implementation and paper results for F3Net.

- **SPSL** [55]

SPSL	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
Ppaer	95.33%	-	-	-	-	-	76.88%	-	-
Benchmark	96.10%	97.81%	97.54%	98.29%	92.99%	81.50%	76.50%	81.22%	64.37%
Implementation	93.10%	97.92%	89.02%	99.23%	86.45%	80.35%	73.15%	82.68%	63.05%

Table 5.7: Comparison of implementation and paper results for SPSL.

- **CORE** [54]

CORE	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
Paper	99.66%	-	-	-	-	-	79.45%	93.74%	-
Benchmark	96.38%	97.87%	98.03%	98.23%	93.39%	77.98	74.28%	80.18%	60.32%
Implementation	95.82%	98.4%	94.53%	98.03%	92.31%	71.83%	74.11%	83.08%	59.18%

Table 5.8: Comparison of implementation and paper results for CORE.

- **RECCE** [9]

RECCE	FaceForensics++					CDF1	CDF2	DFD	FSh
	Overall	FF-F2F	FF-DF	FF-FS	FF-NT				
Paper	99.32%	-	-	-	-	-	68.71%	-	-
Benchmark	96.21%	97.97%	97.79%	97.85%	93.57%	76.77%	73.19%	81.19%	60.95%
Implementation	96.23%	98.32%	94.35%	97.46%	94.77%	73.16%	74.18%	84.79%	56.48%

Table 5.9: Comparison of implementation and paper results for RECCE.

Discussion about the Results Compared to the Papers

Based on the previous experiments reported in Tables 5.6, 5.7, 5.8 and 5.9, the following points are worth noting:

- First and foremost, we used AUC as our primary metric from this point forward due to its informative nature, as it gives us a general overview on the detector at our hand. The higher the AUC, the better our model is at detecting deepfakes while minimising detecting real samples as fake.
- In terms of results reproduction, we have been successful, although many of the results obtained in the implementation were slightly off from those reported in the paper, especially for CORE [54] in Table 5.8 where from the exception of DFD [73], the results were off by 0.17% to even 6.15% for CDF2 and CDF1 [60] respectively.
- Our implementation differs from the original papers in dataset used, since we used far more for a broader picture about the performance of detectors.
- The difference in results for the same dataset can be traced back to the preprocessing of samples used in our training, which was chosen to be unified across all detectors.

5.5 Final Discussion

By analysing the results we have obtained, we can clearly notice a problem that all detectors suffer from which is the drastic performance drop on testing datasets that possess either different forgery methods, more advanced versions of the same forgery methods, and even higher quality deepfakes that have different to no artefacts compared to the training dataset.

Based on these findings, we identify our area of interest which is the clear shortcoming of deepfake detectors: **The lack of generalisation**. Additionally, we also identify what might have caused this flaw which is the training process and the data used in it, which led us to rethink it in Chapter 6.

Based on the results reported in Tables 5.2, 5.3, 5.4, and 5.5, we chose the deepfake detector we continue our study with, which is **RECCE**. This choice was based on the strong performances it has exhibited.

For the second detector we have included in the generalisation study in Chapter 6, we have selected SBI which we re-implemented prior to that. The choice of this detector is based on its high performance and distinctive pre-processing step which creates more challenging data for a detector, we will then use it to combine with RECCE for a better generalisation.

Finally, the lip-syncing detector LipFD was not evaluated due to the unavailability of datasets and even source codes. We have only used it after its late implementation to test out lip syncing deepfakes in Chapter 6.

5.6 Conclusion

To conclude, we have seen through this chapter the basis for our experiments in the experimental setup used and the preprocessing that facilitate the evaluation. Next, we have successfully reproduced the results of 4 state of the art methods in F3Net [58], SPSL [55], CORE [54] and RECCE [9] and analysed the obtained results in order to make a fair comparison between them, to finally choose RECCE to use in the generalisation study in the following chapter. Additionally, we re-implemented the best deepfake detector in SBI [12] to also use in the generalisation study, and LipFD [13] to test on lip syncing videos and compare the result to our method.

Chapter 6

Rethinking Training for a Better Generalisation

6.1 Introduction

In this chapter, we assess the the generalisation ability of deepfake detectors by taking a different training approach. We use AUC as our metric because of its informative nature as detailed in Table 4.10.

First, we use the available datasets by training the first selected detector RECCE on multiple combinations, not only to add more training data which proved to be efficient for other deep learning tasks, but also to introduce multiple forgeries and to study their effects on each other. The combinations include at first only 2 datasets, where we analyse the results obtained before moving on to the second part which has included 3 datasets in the combination to further improve the results, we also included cross-testing and data augmentation to improve the performances.

Then, motivated by the idea behind SBI in Chapter 4 which introduces more general and challenging samples for training, we have combined the pre-processing step of this method with RECCE as a classifier model to enhance the generalisation ability.

Finally, we have tested the most promising result which was the combination of detectors on external LipSync videos, and compared the results to those achieved by the state of the art LipSync detector LipFD [13].

This chapter serves as a review on the generalisation of deepfake detectors, where we have addressed the problem differently using the available resources, all of which has helped us draw a final conclusion on the feasibility of our methods in future works.

6.2 Re-training of RECCE

After the analysis carried out in Chapter 5 and the dataset analysis in Table 4.9, we have noticed a great diversity between datasets, a quality difference especially. To remedy, we re-train RECCE using multiple dataset combination in order to study:

- The effect of introducing new samples in training on the reminder of the datasets.
- The generalisation ability of deepfake detectors when trained differently to the consensus of this field which is training on the train set of FaceForensics++ alone.

6.2.1 Combination of Two Datasets

Training Setup

We have trained the selected model with images of a 256×256 resolution. We have not used any data augmentation or learning rate decay. We have used Adam as an optimizer with a learning rate of $2\times e^{-4}$ and a weight decay of $5\times e^{-4}$. We have sampled 32 frames per video, and 8 workers to load the data. The only difference between combinations is in the batch size, the number of epochs and the training time. Details about these information are given in Table 6.1. Information about the data distribution of all combinations is reported in Table 6.2.

Training datasets	Batch size	Number of epochs	Time/epoch	Best checkpoint
FF++ [69] & CDF1 [60]	32	5	3h15	5th epoch
FF++ [69] & CDF2 [60]	16	6	7h15	1st epoch
FF++ [69] & DFD [73]	32	4	7h	4th epoch
FF++ [69] & FSh [40]	32	5	3h45	3rd epoch
FF++ [69] & DFDCP [74]	16	6	6h35	5th epoch

Table 6.1: Training information about dataset combinations.

	FF++ train set		Added dataset train set		
	Real	Fake	Real	Fake	Forgery method
FF+&CDF1	720	2880	372	731	Advanced deepfakes
FF++&CDF2	720	2880	412	5309	Advanced deepfakes
FF++&DFD	720	2880	363	3068	Unmentioned, 5 face swapping methods
FF++&FSh	720	2880	-	720	Face Shifter
FF++&DFDCP	720	2880	1000	3776	2 face swapping methods

Table 6.2: Dataset combination stats.

The real videos of Face Shifter are the same as those of FaceForensics++, therefore we only load them once.

Characteristics of deepfake datasets are detailed in Chapter 3, and information about their artefacts is reported in Table 4.9.

Results of the Two Dataset Combination

Tables 6.3 and 6.4 summarize the results we have got with the combinations used on all datasets and on the subsets of FaceForensics++ respectively.

The gray cells indicate that the dataset was used in training. Bold numbers indicate an increase of performance, while italic number indicate a drop. Underlined values indicate a drop of over 10% from the baseline.

Combination	FF++	CDF1	CDF2	DFD	FSh	DFDCP
FF++	96.23%	73.16%	74.18%	84.79%	56.48%	71.04%
FF++ & CDF1	**97.91%**	**99.85%**	**94.41%**	**86.58%**	**66.97%**	**72.83%**
FF++ & CDF2	*95.50%*	**99.27%**	**98.93%**	*83.21%*	**58.80%**	**71.64%**
FF++ & DFD	**96.78%**	*72.71%*	*69.49%*	**99.79%**	**60.11%**	**71.24%**
FF++ & FSh	**97.72%**	*57.04%*	*63.98%*	*83.09%*	**99.00%**	**73.83%**
FF++ & DFDCP	**97.19%**	*68.24%*	**74.48%**	*84.09%*	**60.84%**	**93.41%**

Table 6.3: Results of two dataset combination on testing datasets.

Combination	Overall	FF-F2F	FF-DF	FF-FS	FF-NT
FF++	*96.23%*	*98.32%*	*94.35%*	*97.46%*	*94.77%*
FF++ & CDF1	**97.91%**	**98.48%**	**98.10%**	**98.22%**	**96.82%**
FF++ & CDF2	*95.50%*	*96.97%*	**94.60%**	**98.69%**	*91.72%*
FF++ & DFD	**96.78%**	*97.75%*	**96.31%**	**98.05%**	**94.99%**
FF++ & FSh	**97.72%**	*98.27%*	**97.92%**	**98.63%**	**96.05%**
FF++ & DFDCP	**97.19%**	*97.36%*	**97.45%**	**98.76%**	**95.21%**

Table 6.4: Results of dataset combination on FF++ subsets.

Discussion about the Results of the Two Dataset Combination

When analysing the reported results in Tables 6.3 and 6.4, we can clearly distinguish the different effects of combinations on testing datasets, that could be summed up in the following.

- **First Combination: FF++ & CDF1**: This combination generalised the best especially on FSh where a substantial 10.29% increase is noticed, on top of being the only combination to improve of DFD and all FF++ subsets. The reasons behind this improvement are many. For instance, this dataset presents the highest quality deepfakes as discussed in 3.2.2. Additionally, the dataset represents 1/4 of the total combination as illustrated in Table 6.2, 2/3 coming from its advanced face swapping method, which is considered a good distribution. The probable setback of the dataset is its little artefacts as illustrated in Table 4.9, although it is complemented with the artefact-heavy FF++.

- **Second Combination: FF++&CDF2**: This combination improved more on CDF1 because of the higher amount of samples as illustrated in Table 6.2, which has subsequently hurt the performance on FF++ and DFD, since 57% of the total data comes from only one forgery method, which also caused the model to not improve much on FSh and DFDCP, since this dataset has also few artefacts as shown in Table 4.9.

- **Third Combination: FF++&DFD**: This combination improved slightly on FF++ and DFDCP, improved on FSh and Face swapping subsets of FF++, which we all trace back to this dataset having 5 face swapping algorithms that took 43.61% of the entire training data, it had a performance drop of CDF1 and CDF2 though since its samples present many artefacts that are not in the CDF datasets as shown in Table 4.9. The probable setback of this combination is the low training time, where the model was still improving when it stopped according to Table 6.1.

- **Fourth Combination: FF++&FSh**: This combination improved on FF++ and DFDCP while having a performance drop in DFD, which could be traced back to the artefacts FSh presents that are fairly different to those of DFD, same could be said about the difference with CDF1 and CDF2, where we had a drastic drop of performance (16.12% and 10.2% respectively). This drop is traced back to: The different data quality and artefacts as detailed in Chapter 3 and Table 4.9. There is also the data distribution, as FSh makes 1/5 of the overall set, while only bringing fake samples which resulted in an unfair fake/real ratio. The training time was not the issue in this case as we retrained the model for longer (10 epochs) and got even worse performances on CDF1 and DFD as illustrated in Table 6.5.

Dataset	FF++	CDF1	CDF2	DFD	FSh	DFDCP
Baseline	96.23%	73.16%	74.18%	84.79%	56.48%	71.04%
FF++&FSh	97.71%	<u>53.12%</u>	<u>64.01%</u>	*82.15%*	99.21%	**74.39%**

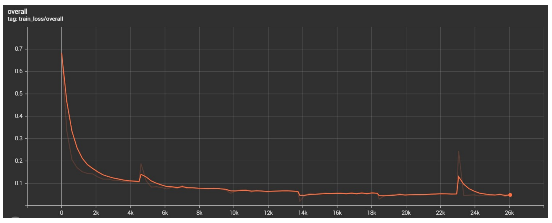
Table 6.5: Results when training on the FSh combination for 10 epochs.

- **Fifth Combination FF++&DFDCP**: The second best combination, it improved on FSh and on FF++ and CDF2 slightly, which we could trace back to it having 2 face swapping methods that took 45% of the total train set, while also providing real samples, which did not bias its artefacts heavily. The drop on DFD is the difference in artefacts as shown in Table 4.9. The drop on CDF1 while improving on CDF2 could be traced back to CDF1 having fewer samples to test on, as their artefacts and deepfake quality are fairly close.
- **Batch size**: The choice of this hyper-parameter proved to be important in our study, which means it could have impacted the DFDCP and CDF2 combinations. This issue was unresolved for the reminder of the study, as we only used a batch size of 16 due to hardware problems.
- **Overfitting**: This problem is recurrent for all combinations even the best one as illustrated in Figure 6.1, meaning our detector is converging more to the train set representation.
- **Data Distribution**: One additional problem with the available datasets is the ratio of fake to real videos within the dataset as detailed in Table 6.2 which is large for most. As a result, one single real face may appear in many manipulated videos, thus driving the detector to over-fit not only to the artefacts mentioned in Table 4.9, but also to the faces within the dataset. This finding solidifies even more the overfitting case of the FSh combination which had a ratio of 5 fake samples to 1 real sample. This claim is supported already by Das et al. [85].

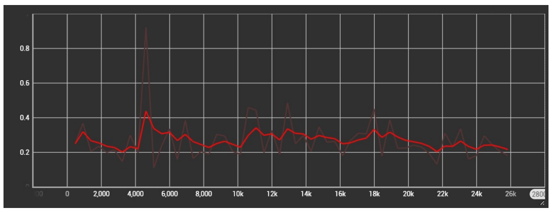
Results using Cross-testing

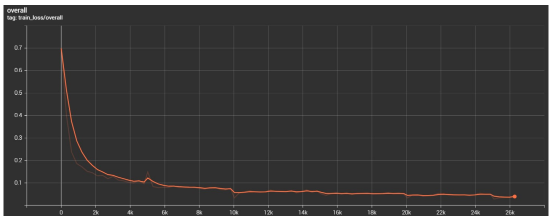
Since it is infeasible to change datasets or to create a new one, a solution we propose to get around the shortcoming of the training is the cross-testing method.

Since deep learning models test on unseen data to save the best checkpoint for the model, we propose to not only test with test-set of the training dataset, but to also use the test set of an external dataset, that way even if the model starts converging to the data representation of the train set, we could capture the trade-off between good within-domain



(a) CDF1 combination train loss.


(b) CDF1 combination test loss.



(c) FSh combination train loss.

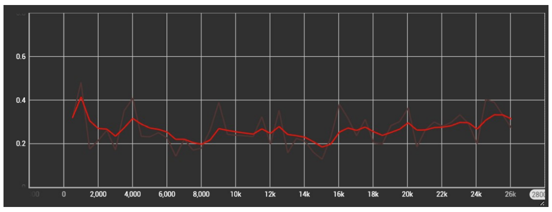

(d) FSh combination test loss.

Figure 6.1: Train and test loss for the CDF1 and FSh combinations.

performance and acceptable cross-domain performance (generalisation). This method is then proposed as a cure to overfitting to enhance generalisation ability.

We have used the DFDCP combination since it had a potential for improvement. For that, we have used CDF1 as our cross-test data, since it had the overall best impact when incorporated within a training, we hypothesize that the model that generalises better to it would improve overall.

Tables 6.6 and 6.7 represent a comparison between the baseline, the normal training approach and the Cross-testing approach for FF++&DFDCP combination.

Dataset	FF++	CDF1	CDF2	DFD	FSh	DFDCP
Baseline	96.23%	73.16%	74.18%	84.79%	56.48%	71.04%
Normal testing	**97.19%**	*68.24%*	**74.48%**	*84.09%*	**60.84%**	**93.41%**
Cross-testing	*96.21%*	**74.82%**	**78.42%**	**86.87%**	**59.14%**	**92.03%**

Table 6.6: Cross-testing results with DFDCP combination on all datasets.

Dataset	FF++	FF-F2F	FF-DF	FF-FS	FF-NT
Baseline	96.23%	98.32%	94.35%	97.46%	94.77%
Normal testing	**97.19%**	*97.36%*	***97.45%***	**98.76%**	**95.21%**
Cross-testing	*96.21%*	*97.75%*	**94.67%**	**98.37%**	*94.07%*

Table 6.7: Cross-testing results with DFDCP combination on FF++ subsets.

When we analyse the results obtained in Table 6.6, the following points stand out:

- The performance on CDF1 improves as expected since we chose the closer model to it, which inevitably increased the performance on CDF2 even more since these two datasets go together.

- We can clearly see an increase on DFD as well, a result that could be explained by those obtained in Table 6.3, where the CDF1 combination was the only one to improve on DFD.

- Performance on the training datasets decreased from the normal testing results (by 0.98% for FF++ and 1.38% for DFDCP) which was inevitable since the checkpoints taken for both experiments serve a different purpose, as the normal testing seeks the best performance on the training data.

- Another observation is that the cross-testing improved the face swapping algorithms and had a negative impact on face reenactment algorithms based on Table 6.7, which could simply be traced back to the fact that CDF1 is based on an advanced face swapping method.

- Performance on FSh dropped by 1.7%, a result explained by the difference of this dataset with CDF1, and even though CDF1 served FSh well in Table 6.3, this time around it is more about similarities, which are minimal between these two as showcased in the analysis on datasets.

We can then conclude that this method is very effective when used right, and based on all our experiments we have successfully implemented it to maximise the generalisation ability of a deepfake detector even with the shortcoming of datasets. This technique is also used in Section 6.2.2.

6.2.2 Combination of Three Datasets

Following our previous experiments, we thought about improving the generalisation even more by using a combination of 3 datasets that would be either the best or complementary to one another. For that, we have chosen first the best 2 combinations from Table 6.3 which are FF++&CDF1 and FF++&DFDCP for the first experiment, and then for the complementary representation we have chosen FF++&CDF1 with FF++&FSh, a natural choice that needs no further explanation.

We modify some of the training setup parameters as follows:

- We sample 8 frames per video to investigate what we have found in [12] and from information shared by experts.

- We use a smaller learning rate of $2e^{-5}$ to avoid early convergence.

- We use pre-trained weights provided by the author and train for 5-7 epochs to fine tune the model on the combination.

- We cross-test using FSh for the first combination, and using DFDCP for the second combination.

Results of the Three Dataset Combination

Results of the training are reported in Tables 6.8 and 6.9.

Dataset	FF++	CDF1	CDF2	DFD	FSh	DFDCP
Baseline	96.23%	73.16%	74.18%	84.79%	56.48%	71.04%
FF++&CDF1&DFDCP	96.29%	**98.23%**	**94.15%**	**89.99%**	65.75%	91.62%
FF++&CDF1&FSh	**97.02%**	98.12%	93.98%	84.83%	97.62%	76.05%

Table 6.8: The results of the 3 dataset combination on all datasets.

Dataset	FF++	FF-F2F	FF-DF	FF-FS	FF-NT
Baseline	96.23%	**98.32%**	94.35%	97.46%	94.77%
FF++&CDF1&DFDCP	96.29%	97.43%	94.84%	**97.92%**	94.99%
FF++&CDF1&FSh	**97.02%**	98%	**96.75%**	97.79%	**95.53%**

Table 6.9: The results of the 3 dataset combination on FF++ subsets.

Discussion and Analysis on the Three Dataset Combination Results

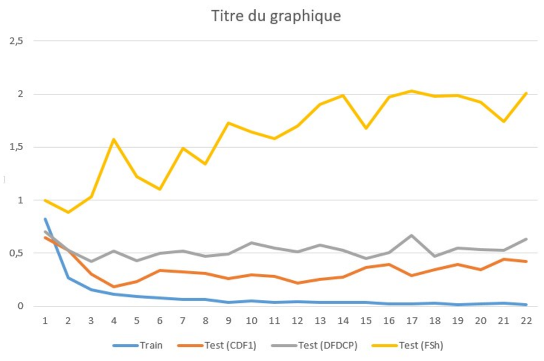
When we analyse the results of Tables 6.8 and 6.9 and Figure 6.2 the following points jump out:

- By comparing the results of Table 6.8 and Table 6.3, we can clearly see that the within-domain datasets have a slight performance drop compared to the two dataset combination, which can be explained by the fact that we added another dataset, decreasing the focus on a single dataset to focus more on the overall set.

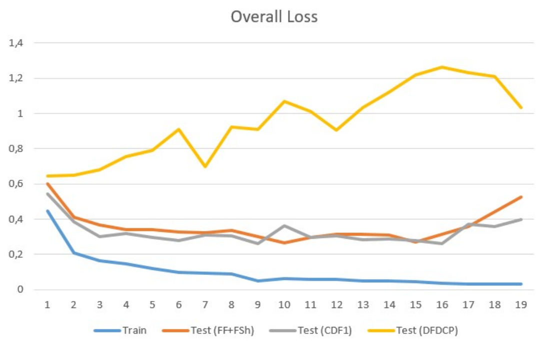
- For the generalisation, the first combination with DFDCP had the better performance, which was expected since the datasets used had both a positive impact on all of the others. As for the second combination, we had the best improvement on DFDCP thus far, but nearly no impact on DFD, which could traced back to CDF1 and FSh cancelling each-other's effects on this dataset.

- The same goes for Table 6.9 results, where the effect of both added datasets was basically added with one another.

- For the graphs presented in Sub-Figures 6.2a and 6.2b, we can clearly see a slight overfitting from the within-domain datasets, which could be traced back to the use of only 8 frames per video and the lack of regularisation, but it also reinforces the choice of just few epochs to minimise the damage. As for the cross-testing dataset, we can clearly see that the model is diverging from its representation, which insinuates that there is still a generalisation and overfitting issue even after we added more training data and optimised some hyper parameters.



(a) FF++&CDF1&DFDCP



(b) FF++&CDF1&FSh

Figure 6.2: Loss graphs for both three-datasets-combination.

Results Using Data Augmentation

To limit the overfitting we have noticed in all previous experiments, we propose a regularisation technique which is data augmentation, the augmentations used alongside the criteria for their choice and their implementation details are illustrated in Table 6.10, where we analysed previous deepfake detection works [11, 12, 86, 63, 9, 54, 53, 85] to choose the augmentations.

Augmentation	Criterion	Implementation detail
Image Compression	Most used	The quality interval is [40, 100], with a probability of 0.5.
Horizontal Flip	Most promising	The probability is 0.5.
Random Brightness Contrast	Real world scenarios	The limit for both is [-0.2, 0.2], with a probability of 0.5.
Gaussian Blur		With a limit of [3, 7] and a probability of 0.5.
Random Erasing	Compatibility[1]	We generate between 1 to 5 rectangles of sizes varying between (10, 10) to (25, 25), the fill is (R,G,B)=(0,0,0).

Table 6.10: Data augmentation used

Experimental results are reported in Table 6.11, where we trained on all 5 augmentations, then trained on sub-parts and compared with the baseline and with the model trained without augmentations.

We note that Wo/ refers to without, and W/ refers to with. Also, RandER is Random Erasing, BC is Brightness Contrast, Comp is image compression and Flip is Horizontal Flip.

Dataset	FF++	CDF1	CDF2	DFD	FSh	DFDCP
Baseline	96.23%	73.16%	74.18%	84.79%	56.48%	71.04%
Wo/ Aug	96.29%	98.23%	94.15%	**89.99%**	**65.75%**	91.62%
W/ Aug	96.36%	97.78%	94.50%	88.86%	63.94%	92.78%
Wo/ RandER	96.73%	98.86%	**95.69%**	88.65%	64.31%	**93.96%**
Wo/ BC&Comp	**96.82%**	**98.99%**	95.22%	88.72%	61.58%	93.24%
Wo/ Flip	96.36%	98.73%	95.36%	87.31%	64.23%	93.55%

Table 6.11: Data augmentation analysis results.

The gray cells indicate the used datasets for training, bold values are the best results, the underlined are the worst.

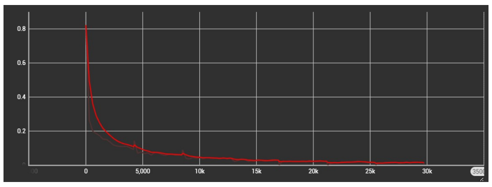
By analysing the results illustrated in Table 6.11 and Figure 6.3, we can break down the following:

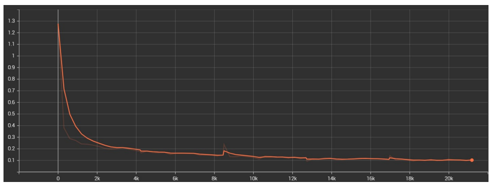
- First and foremost, we compare between the results we obtained when using data augmentation with those we got without augmentation, we can definitely notice from Table 6.11 an increase of performance in two training datasets that are FF++ and DFDCP, but had a drop of performance in CDF1 which could be traced back to the drop of performance on FSh that was used for the cross-testing which has influenced directly the model used for testing. Additionally, when looking at the Losses from Figure 6.3 we can see the test losses for all three testing datasets (CDF1 and DFDCP for the intra-testing) and FSh for the cross-testing, when using augmentation the loss has a more natural curve for CDF1 (i.e. continuously decreasing), a better loss for DFDCP and FSh, where it increased less this time for the latter, showing us some signs of improvement this time in regard of the overfitting we suffered from in earlier experiments.

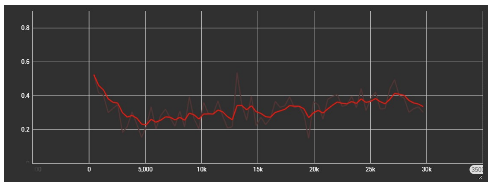
- To understand why the better loss curve does not translate to a higher AUC brings us back to the loss function used in RECCE as mentioned in Section 4.6.3, where we can see that it does not include the classification loss solely, but it also includes a reconstruction loss and a differentiation loss, this is also found on the other metrics where they are better for the augmented training.

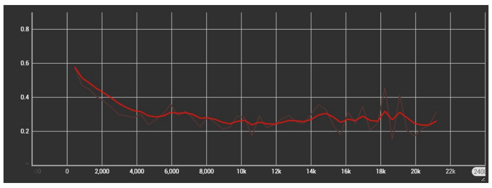
- To study the effect of different augmentations on our model, we retrained it by excluding some of them as shown in Table 6.11, we can clearly see the best overall generalisation is obtained when excluding Random Erasing, which was previously affecting our training the exact different way we wished it to be, as it was erasing regions from the face and preventing the model to learn valuable information instead of pushing it to concentrate on multiple regions, which makes sense since our exact model needs the entire face for reconstruction.

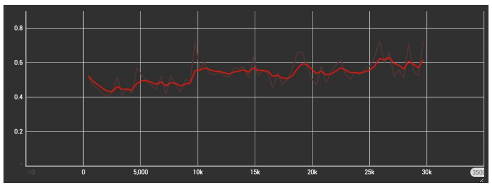
- Random brightness contrast and image compression also hurt the learning process especially for FF++ and CDF1 since they probably obscured forgery traces for both these datasets that are known for having compression and low resolution videos respectively.

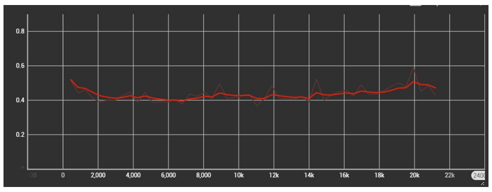

(a) Train Wo/ aug

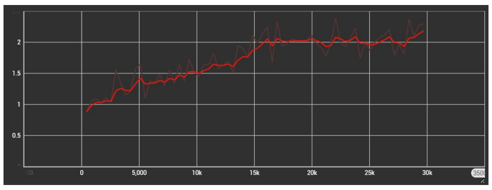

(b) Train W/ aug


(c) Test on CDF1 Wo/ aug


(d) Test on CDF1 w/ aug


(e) Test on DFDCP Wo/ aug


(f) Test on DFDCP W/ aug


(g) Test on FSh Wo/ aug

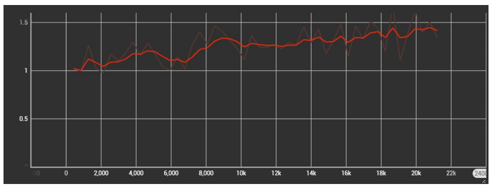

(h) Test on FSh W/ aug

Figure 6.3: Train and test loss graphs with (right) and without (left) augmentations.

- Horizontal Flip had the least negative impact, highlighting its importance in our study since it only provides a different view for our samples, which has always proved to be useful for computer vision tasks.

6.3 Combination of SBI and RECCE

On our quest on finding the best generalisation possible by our detector, we will through the following experiment study the effectiveness and the potential of Self Blended Images (SBIs) [12] that we have studied through Chapter 4. That being said, we have added the pre-processing step of SBI to our selected detector RECCE.

6.3.1 Training Setup for the RECCE-SBI Combination

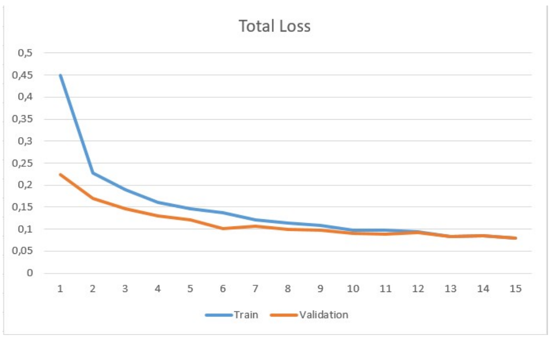
In the preprocessing step, we sample 16 frames per video given that they only used 8 in the original paper of SBI [12], additionally we set the inputs resolution to 256×256 to stay consistent with our previous experiments.

For the training, we have used SAM optimizer with a learning rate of 0.001, we have trained the model with a batch size of 16 for 15 epochs. Data augmentation was applied including Image Compression, RGB Shift, Hue Saturation Value, and Random Brightness Contrast to stay consistent with the original paper.

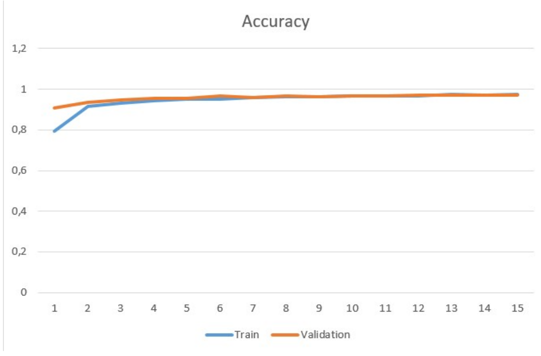
The biggest take away from this training is that we only use the 720 real videos from FF++, and then generate to each video its SBI, resulting in only 1440 total videos for our train set, which is in fact insignificant when compared to previous experiments, helping greatly in training time and needing less computational power.

6.3.2 Results of the RECCE-SBI Combination

Results are reported in Table 6.12, supported by the loss curves from Figure 6.4.



(a) Loss (Batch size = 16).



(b) Accuracy (Batch size = 16).

Figure 6.4: Loss and Accuracy graphs for RECCE and SBI combination with a batch size of 16.

Dataset	FF++	CDF1	CDF2	DFD	FSh	DFDCP
Baseline	96.23%	73.16%	74.18%	84.79%	56.48%	71.04%
RECCE+SBI	83.77%	**82.17%**	**79.09%**	**86.62%**	**72.06%**	**72.38%**

Table 6.12: Performances of RECCE&SBI combination on all datasets.

The metric used on the last row is the video-level AUC.

6.3.3 Analysis and Discussion about the RECCE-SBI Combination

We can clearly see from Figure 6.4a that we have stopped the training prematurely, we could not preview that the loss would still be decreasing by epoch 15 from our previous experiments, which could surely be traced back to the difference in the optimiser used which is SAM as opposed to ADAM in this training, on top of the difference in data used, which could very much be challenging as mentioned in [12].

Even though we did not explore the full potential of our model in training, knowing that we only trained it for 15 epochs whereas the original paper trained it for 100 epochs, we still had some strong results as reported in Table 6.12, where we can clearly see we improved on all datasets, for some by fine margins such as Face Shifter (+14.58%) that posed the greatest challenge in our experiments thus far, all of which was done without using a single sample from these datasets.

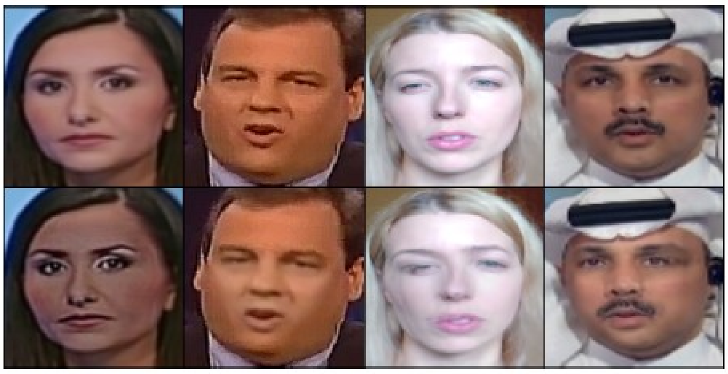


Figure 6.5: Examples or real images on top and their SBIs on the bottom. [12].

Another result that jumps out is the performance on FF++, which has dropped significantly in comparison to our baseline. We explain this phenomenon first by the fact that our baseline already had FF++ as a whole in its train set which is not the case for the second training using SBI. Additionally, FF++ has face reenactment algorithms as opposed to the other datasets that only include face swapping methods, which we hypothesize has hurt the performance on this dataset.

These results take our assumptions a step further, confirming that the biggest issue the deepfake detection field is facing is the data it uses for its training, as these detectors are quite advanced and held back by the datasets they are fed. When we compare training data like SBI in Figure 6.5 with those in Figures 4.11, 4.12 and 4.13, we can not only confirm that SBIs present more challenging data for the detector, but also has a broader range of artefacts that include most of those present in other datasets and real life deepfakes, which makes such a method more practical than having to study and add many datasets which has certainly a limit to it.

6.4 Testing on our Deepfakes

6.4.1 Motivation

After training and evaluating our chosen detector RECCE, we have found that the only way around the overfitting caused by the data is to train it with SBI, a combination that has provided us with some promising results in terms of generalisation. On top of that, SBIs do not follow a specific deepfake representation such as face swapping, but rather represent artefacts present in some of these methods, which makes it even more valuable for this experiment.

To test our method's robustness and detection ability, there is no better way than to test it on external videos generated by unseen forgery types. To check the last box, we collected 15 videos generated by [87], which included five subjects and consisted of only LipSync manipulated videos generated by three different methods that are: DINet [88], GeneFace [44] and Wav2Lip [7].

For a deeper analysis, we selected a state of the art LipSync detector developed by Liu et al. [13] and published in March 2024, which we reproduced the paper's results in order to test out on the mentioned videos and compare with our method.

6.4.2 Results on our Deepfakes

For testing, we set a uniform number of frames to extract for each method, which was:

- For LipFD, we set a maximum of 150 frames aligned with their spectrogram as shown in Figure 6.6, we note that for shorter videos such as those of Zemmour and President Macron, they only included 105 and 135 frames respectively.

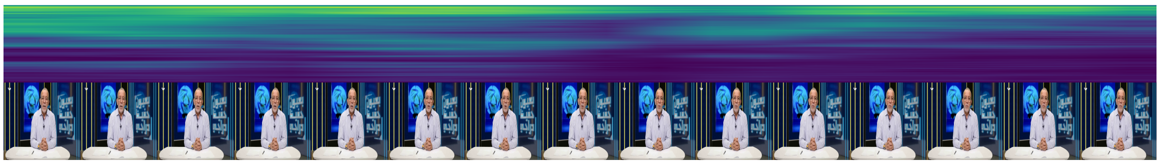


Figure 6.6: Example of a preprocessed result of an external video.

- For RECCE+SBI, we extracted 100 frames per video to stay consistent, with the exception of the Mr. Derradji ones where we sampled 150 frames (the difference was not significant).

To determine whether a video is fake or not, we calculated the level of fakeness which we simply formulated as the mean prediction score of the frames, a video is labeled as fake if the fakeness level is above 0.5, testing results are reported in Table 6.13.

Generation method	Dinet		GeneFace		Wav2Lip	
Detector	LipFD	SBI+RECCE	LipFD	SBI+RECCE	LipFD	SBI+RECCE
Mr. Derradji	0.0002	<u>0.4298</u>	0.0001	<u>0.4409</u>	0.0002	**0.8056**
President Tebboune	0.0151	0.2986	0.0156	0.2669	0.0696	<u>0.4871</u>
Mr. Mbappé	0.3675	0.1448	0.1766	0.1201	**0.5652**	**0.5085**
Mr. Zemmour	0.0754	**0.5151**	0.074	0.1826	0.0949	**0.7529**
President Macron	0.0000	0.3252	0.0000	0.1891	0.0000	**0.6933**

Table 6.13: Testing results on external videos.

The bold values indicate that the deepfake was successfully detected, whereas the underlined values indicate a suspected video (score very close to 0.5).

6.4.3 Analysis and Discussion

When we analyse the results from Table 6.13, we can see the specialised LipSync detector failed miserably in detecting these deepfake illustrated in Figure 6.7, even though they exhibit many visual artefacts and audio-visual inconsistencies from the exception of President Tebboune videos which were the closest to real videos the most, the most absurd result is for the President Macron videos, which were clearly deepfakes but had a fakeness level of the order of 10^{-8}.

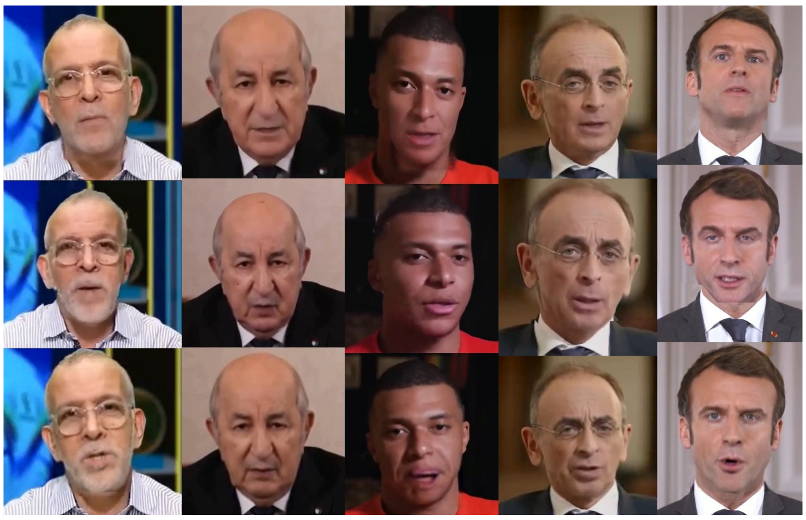


Figure 6.7: Samples from our deepfakes, with DINet on top, GeneFace in the middle, and Wav2Lip on the bottom for each subject [87].

On the other hand, our method which is the combination RECCE+SBI has had some promising results, detecting 1/3 of the total of videos, with additional three videos on the verge of detection. While this might sound underwhelming, we remind that this detector was trained on samples that mimic artefacts present in face swapping and face reenactment deepfakes, while lip syncing deepfakes are a totally new and unseen manipulation type, which makes these results more impressive. We can see that most detected videos are those generated from Wav2Lip which exhibit visually more artefacts and have a lower quality (bottom row of Figure 6.7).

To further test these two detectors, we evaluated them on a deepfake collected from social media as shown in Figure 6.8, it has been generated by an unknown LipSync method. The fakeness scores are the following:

- **LipFD**: 0.0129. (Frames extracted: 320).
- **RECCE+SBI**: 0.5367. (Frames extracted: 100).

Figure 6.8: Deepfake video collected from social media [89].

All of the previous results on new unseen videos external from existing datasets confirm even more our assumptions on the generalisation deepfake detectors lack, especially when trained on conventional datasets (even in this case a private dataset), which brings us to the conclusion that this field could improve way better with a different approach to its training as it was done in SBIs [12] which proved to be the best existing method, that could be improved more by including some artefacts from LipSync deepfakes, on top of having to update the data constantly to catch up to the rapid development of deepfake generation.

6.5 Conclusion

To conclude, we have studied through this chapter the generalisation ability of deepfake detectors through two main parts: The first one included facing the selected detector RECCE [9] with multiple dataset combinations to study the famous effect of having more diversified data leading to a better performance, which was not the case for every combination due to the broad difference between deepfakes and generation methods, which also led to an overfitting that we have tried to tackle using cross-testing and data augmentation. The second part was motivated by the high performance exhibited by SBI [12] that proposes challenging data through its pre-processing step, we have then combined this pre-processing of SBI with RECCE to obtain the best generalisation on testing datasets while never including any of their samples. Finally, we tested the last method which was the best along with a lip syncing detector in LipFD [13] on 16 lip syncing deepfakes, in which we outperformed the specialised detector, highlighting even more on the importance of training on harder and more general data for a better performance in real life scenarios.

Conclusion and Perspectives

Conclusion and Perspectives

In conclusion, our study delved deep into the assessment of deepfake detectors, first by setting the theoretical basis of deepfakes and deepfake detection, where we presented our approach on the subject by choosing some promising state of the art detectors alongside datasets and evaluation metrics to enable us to effectively train and evaluate these methods.

Next, for our experiments we re-implemented the chosen unimodal detectors: F3Net, SPSL, CORE, and RECCE and compared their results to come out with the best detector which is RECCE for the generalisation study. During the reproduction step, we identified the problem and our interest which was the generalisation ability which deepfake detectors lacked. In order to tackle the latter, we adopted many deep learning techniques from adding data to the train set to expose the network to more diverse samples, on top of using regularisation techniques to avoid or reduce the overfitting we faced, which has worked slightly until we added more techniques nearly leading to an underfitting. Then, to prove our assumptions on the reason abstracting our learning which we hypothesized was the data, we have combined RECCE with SBI pre-processing step that provides challenging and more general data for the detector, a combination that has provided us with an improvement over the baseline of 9.01%, 4.91%, 1.83%, 15.58%, and 1.38% on CDF1, CDF2, DFD, FSh, and DFDCP respectively. Finally, to test the robustness and generalisation ability to unseen forgeries, we tested the best method which was the detector combination RECCE+SBI on 16 LipSync deepfakes where it outperformed a state of the art LipSync detector, proving even more the alarming data problem deepfake detection faces.

To sum up, our contribution has been to assess the generalisation ability of deepfake detector by modifying the training process which has not attracted sufficient attention by the community to the best of our knowledge, where we tested out all dataset combinations possible to study the influence of datasets on each-other, after we debugged these datasets to delve deeper and explain the performances. Next, we combined two deepfake detectors to gain in performance and generalisation ability while also reducing the computational needs for the training. On top of that, we studied the effects of data augmentation on deepfake detection models, before finally testing on the highly overlooked LipSync deepfakes, where we outperformed a state of the art LipSync detector introduced by Liu et al. [13] and obtained a 37.5% accuracy on 16 external deepfakes.

Although we have made a huge step towards the development of deepfake detectors where we identified its biggest issue, there are still improvements and new fields to investigate that could be summarised as follows:

- We have tested out more of spatial and frequency based detectors, while still having the possibility of testing more temporal based detectors, which we deemed were more suitable for LipSync detection.
- Talking about LipSync, we didn't have resources (unavailability of LipSync based datasets) to concentrate on them, as only few detectors exist also, from which we chose the most recent and the top performer.
- We tested RECCE+SBI which was the most promising method of our study on external LipSync deepfakes without even training them on LipSync artefacts, so it would be more interesting to study this deepfake type more in order to be able to mimic its artefacts, and adding these artefacts to SBI to push its boundaries even further.
- We have noticed throughout our study that the deepfake detectors need a lot of memory and computational power to work properly, we propose to study the possibility of optimising the algorithms to reduce those needs.

Bibliography

1. *. Mahajan, V. (2024, March 19). 100+ YouTube statistics in 2024: Users, revenue more.* . [https://www.notta.ai/en/blog/youtube-statistics]. [N.d.].
2. WESTERLUND, Mika. The emergence of deepfake technology: A review. *Technology innovation management review*. 2019, vol. 9, no. 11.
3. BALDI, Pierre. Autoencoders, unsupervised learning, and deep architectures. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.
4. GOODFELLOW, Ian; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua. Generative adversarial nets. *Advances in neural information processing systems*. 2014, vol. 27.
5. NIRKIN, Yuval; KELLER, Yosi; HASSNER, Tal. Fsgan: Subject agnostic face swapping and reenactment. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 7184–7193.
6. WANG, Yuxuan; SKERRY-RYAN, RJ; STANTON, Daisy; WU, Yonghui; WEISS, Ron J; JAITLY, Navdeep; YANG, Zongheng; XIAO, Ying; CHEN, Zhifeng; BENGIO, Samy, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*. 2017.
7. PRAJWAL, KR; MUKHOPADHYAY, Rudrabha; NAMBOODIRI, Vinay P; JAWAHAR, CV. A lip sync expert is all you need for speech to lip generation in the wild. In: *Proceedings of the 28th ACM international conference on multimedia*. 2020, pp. 484–492.
8. AFCHAR, Darius; NOZICK, Vincent; YAMAGISHI, Junichi; ECHIZEN, Isao. Mesonet: a compact facial video forgery detection network. In: *2018 IEEE international workshop on information forensics and security (WIFS)*. IEEE, 2018, pp. 1–7.
9. CAO, Junyi; MA, Chao; YAO, Taiping; CHEN, Shen; DING, Shouhong; YANG, Xiaokang. End-to-end reconstruction-classification learning for face forgery detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4113–4122.

10. HALIASSOS, Alexandros; MIRA, Rodrigo; PETRIDIS, Stavros; PANTIC, Maja. Leveraging real talking faces via self-supervision for robust forgery detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14950–14962.
11. YAN, Zhiyuan; ZHANG, Yong; FAN, Yanbo; WU, Baoyuan. Ucf: Uncovering common features for generalizable deepfake detection. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 22412–22423.
12. SHIOHARA, Kaede; YAMASAKI, Toshihiko. Detecting deepfakes with self-blended images. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18720–18729.
13. LIU, Weifeng; SHE, Tianyi; LIU, Jiawei; WANG, Run; YAO, Dongyu; LIANG, Ziyou. Lips Are Lying: Spotting the Temporal Inconsistency between Audio and Visual in Lip-Syncing DeepFakes. *arXiv preprint arXiv:2401.15668*. 2024.
14. KARRAS, Tero; LAINE, Samuli; AILA, Timo. A style-based generator architecture for generative adversarial networks. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.
15. KARRAS, Tero; LAINE, Samuli; AITTALA, Miika; HELLSTEN, Janne; LEHTINEN, Jaakko; AILA, Timo. Analyzing and improving the image quality of stylegan. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.
16. KARRAS, Tero; AITTALA, Miika; HELLSTEN, Janne; LAINE, Samuli; LEHTINEN, Jaakko; AILA, Timo. Training generative adversarial networks with limited data. *Advances in neural information processing systems*. 2020, vol. 33, pp. 12104–12114.
17. . *Face generator – generate faces online using AI*. [https://generated.photos/face-generator]. [N.d.].
18. STEHOUWER, Joel; DANG, Hao; LIU, Feng; LIU, Xiaoming; JAIN, Anil. On the detection of digital face manipulation. *arXiv*. 2019, arXiv–1910.
19. NEVES, Joao C; TOLOSANA, Ruben; VERA-RODRIGUEZ, Ruben; LOPES, Vasco; PROENÇA, Hugo; FIERREZ, Julian. Ganprintr: Improved fakes and evaluation of the state of the art in face manipulation detection. *IEEE Journal of Selected Topics in Signal Processing*. 2020, vol. 14, no. 5, pp. 1038–1048.
20. MALIK, Asad; KURIBAYASHI, Minoru; ABDULLAHI, Sani M; KHAN, Ahmad Neyaz. DeepFake detection for human face images and videos: A survey. *Ieee Access*. 2022, vol. 10, pp. 18757–18775.
21. . *Marekkowalski/FACESWAP: 3D face swapping implemented in Python* [https://github.com/MarekKowalski/FaceSwap]. [N.d.].
22. . *Deepfakes/faceswap: Deepfakes Software for all* [https://github.com/deepfakes/faceswap]. [N.d.].

23. CHOI, Yunjey; CHOI, Minje; KIM, Munyoung; HA, Jung-Woo; KIM, Sunghun; CHOO, Jaegul. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8789–8797.

24. . *Deepfakes. (2019, March 6). FakeApp 2.2 - download for PC free. Malavida.* [https://www.malavida.com/en/soft/fakeapp/]. [N.d.].

25. THIES, Justus; ZOLLHOFER, Michael; STAMMINGER, Marc; THEOBALT, Christian; NIESSNER, Matthias. Face2face: Real-time face capture and reenactment of rgb videos. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2387–2395.

26. THIES, Justus; ZOLLHÖFER, Michael; NIESSNER, Matthias. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)*. 2019, vol. 38, no. 4, pp. 1–12.

27. . *Face morphing: Photos de stock (1156 images). Shutterstock.* [https://www.shutterstock.com/fr/search/face-morphing]. [N.d.].

28. SUN, Zongji. *Face de-identification for privacy protection*. 2018. PhD thesis. University of Hertfordshire.

29. SUWAJANAKORN, Supasorn; SEITZ, Steven M; KEMELMACHER-SHLIZERMAN, Ira. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (ToG)*. 2017, vol. 36, no. 4, pp. 1–13.

30. . *Macdonald, F. (2024, March 14). How a 19th-century portrait of Abraham Lincoln was later revealed to be a fake. BBC News.* [https://www.bbc.com/culture/article/20240313-how-a-19th-century-portrait-of-abraham-lincoln-was-later-revealed-to-be-a-fake]. [N.d.].

31. . *It's widely known that Stalin was known for doctoring pictures to remove people he did not like. how did we find out he was doing this? Quora.* [https://www.quora.com/It-s-widely-known-that-Stalin-was-known-for-doctoring-pictures-to-remove-people-he-did-not-like-How-did-we-find-out-he-was-doing-this]. [N.d.].

32. . *Is a bear chasing National Geographic Photographers Real?. Quora.* [https://www.quora.com/Is-a-bear-chasing-National-Geographic-photographers-real]. [N.d.].

33. MASOOD, Momina; NAWAZ, Mariam; MALIK, Khalid Mahmood; JAVED, Ali; IRTAZA, Aun; MALIK, Hafiz. Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward. *Applied intelligence*. 2023, vol. 53, no. 4, pp. 3974–4026.

34. BREGLER, Christoph; COVELL, Michele; SLANEY, Malcolm. Video rewrite: Driving visual speech with audio. In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, pp. 715–722.

35. . *Iperov/deepfacelab: Deepfacelab is the leading software for creating deepfakes.* [https://github.com/iperov/DeepFaceLab]. [N.d.].

36. . *Gillespie, D. (2020, December 25). Deepfake Luke Skywalker is way more convincing in the mandalorian. ScreenRant.* [https://screenrant.com/mandalorian-luke-skywalker-mark-hamill-cgi-deepfake-video/]. [N.d.].

37. . *Artificial-intelligence voice is used in a theft - The Washington Post* [https://www.washingtonpost.com/technology/2019/09/04/an-artificial-intelligence-first-voice-mimicking-software-reportedly-used-major-theft/]. [N.d.].

38. . *YouTube. (2022, March 17). Deepfake video of Volodymyr Zelensky surrendering surfaces on social media.* [https://www.youtube.com/watch?v=X17yrEV5sl4]. [N.d.].

39. LIU, Kunlin; PEROV, Ivan; GAO, Daiheng; CHERVONIY, Nikolay; ZHOU, Wenbo; ZHANG, Weiming. Deepfacelab: Integrated, flexible and extensible face-swapping framework. *Pattern Recognition*. 2023, vol. 141, p. 109628.

40. LI, Lingzhi; BAO, Jianmin; YANG, Hao; CHEN, Dong; WEN, Fang. Faceshifter: Towards high fidelity and occlusion aware face swapping. *arXiv preprint arXiv:1912.13457*. 2019.

41. BARATTIN, Simone; TZELEPIS, Christos; PATRAS, Ioannis; SEBE, Nicu. Attribute-preserving face dataset anonymization via latent code optimization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8001–8010.

42. DOUKAS, Michail Christos; ZAFEIRIOU, Stefanos; SHARMANSKA, Viktoriia. Headgan: One-shot neural head synthesis and editing. In: *Proceedings of the IEEE/CVF International conference on Computer Vision*. 2021, pp. 14398–14407.

43. WANG, Jiadong; QIAN, Xinyuan; ZHANG, Malu; TAN, Robby T; LI, Haizhou. Seeing what you said: Talking face generation guided by a lip reading expert. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14653–14662.

44. YE, Zhenhui; JIANG, Ziyue; REN, Yi; LIU, Jinglin; HE, Jinzheng; ZHAO, Zhou. Geneface: Generalized and high-fidelity audio-driven 3d talking face synthesis. *arXiv preprint arXiv:2301.13430*. 2023.

45. LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. *nature*. 2015, vol. 521, no. 7553, pp. 436–444.

46. SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 2014.

47. SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent; RABINOVICH, Andrew. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

48. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

49. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. *Advances in neural information processing systems*. 2017, vol. 30.

50. CHOLLET, François. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

51. TAN, Mingxing; LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

52. DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

53. YAN, Zhiyuan; ZHANG, Yong; YUAN, Xinhang; LYU, Siwei; WU, Baoyuan. Deepfakebench: A comprehensive benchmark of deepfake detection. *arXiv preprint arXiv:2307.01426*. 2023.

54. NI, Yunsheng; MENG, Depu; YU, Changqian; QUAN, Chengbin; REN, Dongchun; ZHAO, Youjian. Core: Consistent representation learning for face forgery detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12–21.

55. LIU, Honggu; LI, Xiaodan; ZHOU, Wenbo; CHEN, Yuefeng; HE, Yuan; XUE, Hui; ZHANG, Weiming; YU, Nenghai. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 772–781.

56. LI, Lingzhi; BAO, Jianmin; ZHANG, Ting; YANG, Hao; CHEN, Dong; WEN, Fang; GUO, Baining. Face x-ray for more general face forgery detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 5001–5010.

57. PEI, Gan; ZHANG, Jiangning; HU, Menghan; ZHAI, Guangtao; WANG, Chengjie; ZHANG, Zhenyu; YANG, Jian; SHEN, Chunhua; TAO, Dacheng. Deepfake Generation and Detection: A Benchmark and Survey. *arXiv preprint arXiv:2403.17881*. 2024.

58. QIAN, Yuyang; YIN, Guojun; SHENG, Lu; CHEN, Zixuan; SHAO, Jing. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In: *European conference on computer vision*. Springer, 2020, pp. 86–103.

59. RÖSSLER, Andreas; COZZOLINO, Davide; VERDOLIVA, Luisa; RIESS, Christian; THIES, Justus; NIESSNER, Matthias. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*. 2018.

60. LI, Yuezun; YANG, Xin; SUN, Pu; QI, Honggang; LYU, Siwei. Celeb-df: A large-scale challenging dataset for deepfake forensics. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 3207–3216.

61. DOLHANSKY, Brian; BITTON, Joanna; PFLAUM, Ben; LU, Jikuo; HOWES, Russ; WANG, Menglin; FERRER, Cristian Canton. The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397*. 2020.

62. KIM, Dong-Keon; KIM, Kwang-Su. Generalized facial manipulation detection with edge region feature extraction. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 2828–2838.

63. DONG, Shichao; WANG, Jin; JI, Renhe; LIANG, Jiajun; FAN, Haoqiang; GE, Zheng. Implicit identity leakage: The stumbling block to improving deepfake detection generalization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 3994–4004.

64. KHODABAKHSH, Ali; RAMACHANDRA, Raghavendra; RAJA, Kiran; WASNIK, Pankaj; BUSCH, Christoph. Fake face detection methods: Can they be generalized? In: *2018 international conference of the biometrics special interest group (BIOSIG)*. IEEE, 2018, pp. 1–6.

65. . *NVlabs/FFHQ-dataset: Flickr-Faces-HQ Dataset (FFHQ)* [https://github.com/NVlabs/ffhq-dataset]. [N.d.].

66. . *Fyu/LSUN: LSUN dataset documentation and demo code* [https://github.com/fyu/lsun]. [N.d.].

67. DANG, Hao; LIU, Feng; STEHOUWER, Joel; LIU, Xiaoming; JAIN, Anil K. On the detection of digital face manipulation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern recognition*. 2020, pp. 5781–5790.

68. . *Large-scale celebfaces attributes (celeba) dataset* [http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html]. [N.d.].

69. ROSSLER, Andreas; COZZOLINO, Davide; VERDOLIVA, Luisa; RIESS, Christian; THIES, Justus; NIESSNER, Matthias. Faceforensics++: Learning to detect manipulated facial images. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1–11.

70. TODA, Tomoki; CHEN, Ling-Hui; SAITO, Daisuke; VILLAVICENCIO, Fernando; WESTER, Mirjam; WU, Zhizheng; YAMAGISHI, Junichi. The Voice Conversion Challenge 2016. In: *Interspeech*. 2016, vol. 2016, pp. 1632–1636.

71. WU, Zhizheng; YAMAGISHI, Junichi; KINNUNEN, Tomi; HANILÇI, Cemal; SAHIDULLAH, Mohammed; SIZOV, Aleksandr; EVANS, Nicholas; TODISCO, Massimiliano; DELGADO, Hector. ASVspoof: the automatic speaker verification spoofing and countermeasures challenge. *IEEE Journal of Selected Topics in Signal Processing*. 2017, vol. 11, no. 4, pp. 588–604.

72. LI, Yuezun; CHANG, Ming-Ching; LYU, Siwei. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In: *2018 IEEE International workshop on information forensics and security (WIFS)*. IEEE, 2018, pp. 1–7.

73. . *Contributing data to Deepfake Detection Research* [https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html]. [N.d.].

74. DOLHANSKY, Brian; HOWES, Russ; PFLAUM, Ben; BARAM, Nicole; FERRER, Cristian Canton. The deepfake detection challenge (dfdc) preview dataset. *arXiv preprint arXiv:1910.08854*. 2019.

75. ZAKHAROV, Egor; SHYSHEYA, Aliaksandra; BURKOV, Egor; LEMPITSKY, Victor. Few-shot adversarial learning of realistic neural talking head models. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9459–9468.

76. JIANG, Liming; LI, Ren; WU, Wayne; QIAN, Chen; LOY, Chen Change. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2889–2898.

77. ZI, Bojia; CHANG, Minghao; CHEN, Jingjing; MA, Xingjun; JIANG, Yu-Gang. Wilddeepfake: A challenging real-world dataset for deepfake detection. In: *Proceedings of the 28th ACM international conference on multimedia*. 2020, pp. 2382–2390.

78. KHALID, Hasam; TARIQ, Shahroz; KIM, Minha; WOO, Simon S. FakeAVCeleb: A novel audio-video multimodal deepfake dataset. *arXiv preprint arXiv:2108.05080*. 2021.

79. JIA, Ye; ZHANG, Yu; WEISS, Ron; WANG, Quan; SHEN, Jonathan; REN, Fei; NGUYEN, Patrick; PANG, Ruoming; LOPEZ MORENO, Ignacio; WU, Yonghui, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*. 2018, vol. 31.

80. ALTUNCU, Enes; FRANQUEIRA, Virginia NL; LI, Shujun. Deepfake: definitions, performance metrics and standards, datasets and benchmarks, and a meta-review. *arXiv preprint arXiv:2208.10913*. 2022.

81. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 2020.

82. TAUD, Hind; MAS, Jean-Franccois. Multilayer perceptron (MLP). *Geomatic approaches for modeling land change scenarios*. 2018, pp. 451–455.

83. ZHOU, Yang; HAN, Xintong; SHECHTMAN, Eli; ECHEVARRIA, Jose; KALOGERAKIS, Evangelos; LI, Dingzeyu. Makelttalk: speaker-aware talking-head animation. *ACM Transactions On Graphics (TOG)*. 2020, vol. 39, no. 6, pp. 1–15.

84. GONG, Liang Yu; LI, Xue Jun. A Contemporary Survey on Deepfake Detection: Datasets, Algorithms, and Challenges. *Electronics*. 2024, vol. 13, no. 3, p. 585.

85. DAS, Sowmen; SEFERBEKOV, Selim; DATTA, Arup; ISLAM, Md Saiful; AMIN, Md Ruhul. Towards solving the deepfake problem: An analysis on improving deepfake detection using dynamic face augmentation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3776–3785.

86. HALIASSOS, Alexandros; VOUGIOUKAS, Konstantinos; PETRIDIS, Stavros; PANTIC, Maja. Lips don't lie: A generalisable and robust approach to face forgery detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 5039–5049.
87. TAIBAOUI, Abdellah; DJABER, Achref. Assessment of Deepfake Generation Methods. *Ecole Nationale Polytechnique*. 2024.
88. ZHANG, Zhimeng; HU, Zhipeng; DENG, Wenjin; FAN, Changjie; LV, Tangjie; DING, Yu. Dinet: Deformation inpainting network for realistic face visually dubbing on high resolution video. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2023, vol. 37, pp. 3543–3551. No. 3.
89. . *Keep your alive. Facebook.* [https://www.facebook.com/share/v/9vQR6S3K3tZBKmye/?mibextid=SphRi8]. [N.d.].

Appendix

Definitions

Temporal Flickering: Refers to inconsistent brightness and color changes in adjacent frames in a video which causes a distracting effect.

Temporal Jitter: Refers to the inconsistencies and fluctuations appearing in the facial expressions or other high semantic elements across consecutive frames in a video, making the transitions look unnatural and the video unstable.

Rendering Network: A neural network designed to generate or synthesize images or videos, often from a set of input parameters or conditions. These networks are commonly used in applications that require realistic image generation, computer graphics, and virtual or augmented reality. Rendering networks leverage deep learning techniques to model complex visual patterns and produce high-quality, photorealistic outputs.

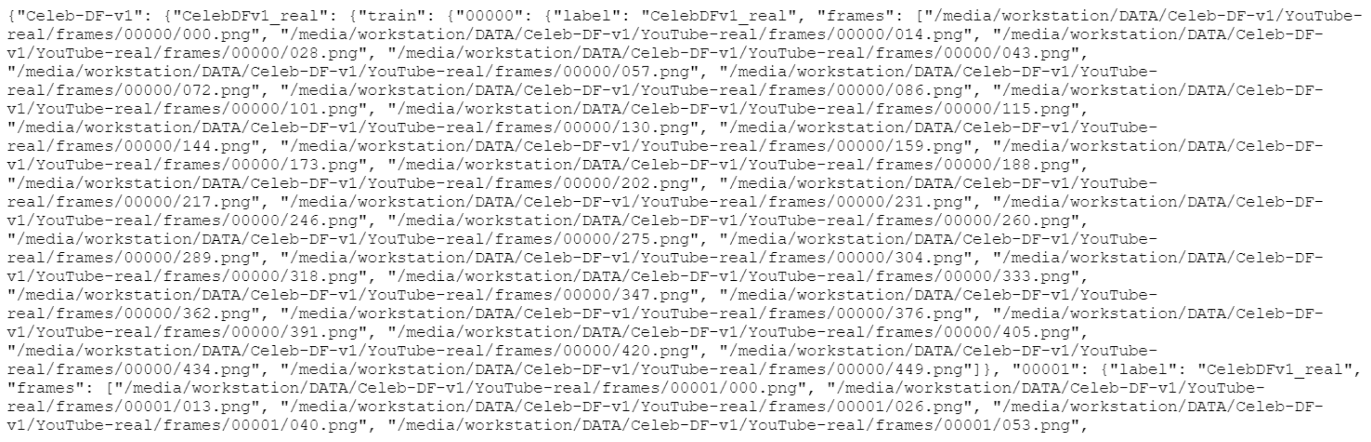
Attention Map: They are a visual representation used to highlight the region in a frame that's the most relevant for the decision making of a network. It helps understand which part is the model focusing on when analysing the inputs.

FLOPs (Floating point operations per second): Refer to the measure of floating-point operations a model computes during inference or training, they are used to determine the computational complexity and efficiency of a model.

Feature Richness Measurement: Used to evaluate the importance or contribution of different feature channels in a neural network. It refers to quantifying how informative or significant each channel of the feature representation is, which can then be used to modulate the aggregation of information from different sources (e.g., encoder and decoder features).

JSON (JavaScript Object Notation): JSON is a lightweight data-interchange format represented under a text format that's independent of programming languages, readable by humans and simple for machines to parse and generate. Figure 9 iluustrates an example of the content of a JSON file.

Convex Hull: The convex hull is the smallest convex polygon or polyhedron that encloses a given set of points. In the context of deepfakes and computer vision, it is the geometric shape that follows the landmarks of the face to give out its shape.



```
{"Celeb-DF-v1": {"CelebDFv1_real": {"train": {"00000": {"label": "CelebDFv1_real", "frames": ["/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/000.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/014.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00000/028.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/043.png",
"/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/057.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/072.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/086.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00000/101.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/115.png",
"/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/130.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/144.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/159.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00000/173.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/188.png",
"/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/202.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/217.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/231.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00000/246.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/260.png",
"/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/275.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/289.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/304.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00000/318.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/333.png",
"/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/347.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/362.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/376.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00000/391.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/405.png",
"/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/420.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00000/434.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00000/449.png"]}, "00001": {"label": "CelebDFv1_real",
"frames": ["/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00001/000.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-
real/frames/00001/013.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00001/026.png", "/media/workstation/DATA/Celeb-DF-
v1/YouTube-real/frames/00001/040.png", "/media/workstation/DATA/Celeb-DF-v1/YouTube-real/frames/00001/053.png",
```

Figure 9: Example of a JSON content for the CDF1 dataset.