Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

School of Computer Science and Statistics

The Tell-Tale Heart: The Detection of Deepfakes Using Biological Signals

Ryan O'Neill

Project Supervisor: Kenneth Dawson-Howe

April 15th, 2024

A Dissertation submitted in partial

fulfilment of the requirements for the

degree of Master of Arts in Computer

Engineering (MAI)

Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

_____

First Last

_____

Month Day, Year

Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Abstract

This project aims to create an application which can detect deepfake videos. A video is dubbed a deepfake when the appearance of the subject in said video has been manipulated by Artificial Intelligence (AI) in some way. There are a multitude of examples of deepfake technology being used for malicious purposes. These include spoofing videos of politicians making statements that they never made, fabricating audio of specific individuals to trick people into thinking they are talking to someone that they are not, and AI generated pornography, to name a few.

Since the introduction of deepfakes in 2017, many different methods have been purposed in order detect when a video has been manipulated by AI in some way. Many of these detection methods have become more difficult to use as time has moved forward and deepfakes have become more realistic. The objective is then to find some information that is present in real videos that would be difficult for deepfake generation tools to replicate both now and in the future. It appears biological signals, which can be calculated from video footage of real people, is difficult for deepfake generation tools to replicate, and thus a distinction between real and fabricated media can be made using these types of signals. This project attempts to discriminate between real and deepfake videos by extracting information related to the heart rate of subjects in video footage.

This project modifies an existing method [49] that successfully distinguished real from deepfake videos using data related to the heart rate of the subjects in the videos. This modification was done to shorten this method's execution time. This project created a method that was 5x faster than the existing method it was derived from in an effort to make the application more accessible for use by the general public. This project's method performed worse overall than the original method when attempting to distinguish 2 different types of deepfake videos from real videos. The results for 1 of these 2 deepfake types were comparable to the original method, the results of the other tested deepfake type were much worse than the original method and were more akin to the model randomly guessing which video was real and which was a deepfake.

Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Acknowledgements

A special thank you to Leah, Val, and my family for helping me when times were difficult.

I would also like to thank my supervisor, Dr. Kenneth-Dawson Howe, who's advice was invaluable during the completion of this project.

Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Contents

1 Introduction 1

1.1 Objective 1

1.2 Motivation 2

1.3 Executive Summary 3

2 Background 6

2.1 Overview of the Deepfake Technology 6

2.1.1 Applications of Deepfakes 6

2.1.2 Generative AI Models 9

2.1.3 Deepfake Generation Applications 13

2.1.4 Deepfake Detection 14

2.2 Related Work 18

2.2.1 Biological Signal Extraction from Video Footage 18

2.2.2 Methods of Deepfake Detection Using Biological Signals 21

2.3 Datasets for Deepfake Detection 23

2.3.1 This Project's Dataset 24

3 Explanation of the Deepfake Detection Application 26

3.1 Previous Methodology 26

3.2 Proposed Methodology 32

4 Explanation of Extracted Features 35

4.1 Summary 35

4.2 Features Used in the Original Method 36

4.2.1 Measurements of Signal Complexity 36

4.2.2 Measurements of Signal Similarity 40

4.3 Features Used in This Project's Method 42

5 Development of the Deepfake Detection Application 49

6 Results 55

6.1 Performance Metrics 55

6.2 Raw Dataset 58

6.2.1 Results 58

6.2.2 Discussion 61

6.3 Lower Quality Datasets 62

6.3.1 Results 62

6.3.2 Discussion 63

7 Conclusion 64

7.1 Review and Critique 64

7.2 Future Work 65

Bibliography 67

List of Figures

2.1 The deepfake video of the President of Ukraine, Volodymyr Zelenskyy. The deepfake video did not originally possess the red and white 'FAKE' label, that was added after the video was found to be illegitimate. Adapted from [3] 7

2.2 An advertisement which consisted of the appearance of the players on the men's French national football team being superimposed onto the women's French national football team using deepfake technology. Adapted from [6] 8

2.3 Deepfake generation using a deep autoencoder. Adapted from a larger figure from [9] 10

2.4 Generative Adversarial Network architecture. Adapted from [11] 11

2.5 A number of input and output images from CycleGAN. Adapted from a larger figure from [12] 12

2.6 The average deepfake detection pipeline. Adapted from [1] 16

2.7 The process of measuring BVP signals via PPG. Adapted from [40] 19

2.8 The process of extracting the skin pixels from the face of a subject in a video in order to analyse the change in colour of these skin pixels, and thus calculate the BVP signals of the subject in the video 20

2.9 An example for both a facial reenactment deepfake and a facial replacement deepfake. Adapted from a larger figure from [56] 25

3.1 An example of splitting a video into multiple overlapping video clips. Every 2 second overlapping interval of the 6 second video is analysed and split up into 5 videos clips. The still image of a subject of a video was adapted from [56] 27

3.2 The pipeline of extracting BVP signals from video footage. The face from every frame of the video is extracted. The mouth and eye regions are identified and removed from this facial image because movement from these regions causes erroneous results when using rPPG. This modified facial image is then split up

into different ROIs. The pixel's RGB values within each ROI are then analysed, and the BVP signal is derived from these values via rPPG methods ……28

3.3 The extraction of signal complexity related features using the existing method. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Features are then calculated from all of these BVP signals directly. Each feature value is then averaged to create a singular feature value that represents the entire video clip ……29

3.4 The extraction of signal comparison related features using the original method with the Magnitude Squared Coherence Estimate. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Features are then calculated, comparing each BVP signal to every other BVP signal. Each of these signals are compared in the frequency domain. The result of the feature calculations is an array of values representing the similarities of each BVP signal at different frequencies. Each of these values are then averaged to achieve a singular scalar value of each BVP signal pair comparison. Each of these feature values are then averaged to create a singular feature value that represents the feature measurement for the current video clip ……30

3.5 The extraction of signal comparison related features using the original method with Mutual Information. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Features are then calculated, comparing each BVP signal to every other BVP signal. Each of these signals are compared in the time domain. The result of these feature calculations is a singular value which provides a measurement of the similarity of each BVP pair in the time domain. The standard deviation of these feature values is then calculated to create a singular feature value that represents the feature measurement for the current video clip ……31

3.6 The extraction of signal complexity related features using this project's method. For each video clip, the modified facial image is split up into a number of ROIs. A

BVP signal is derived for each ROI. Each of these signals are then averaged to generate a singular BVP signal that represents the current video clip. Features are then calculated from this BVP signal to output a singular feature value that represents that particular feature for the current video clip ……33

5.1 BMP over time of a real and deepfake person from video footage. These signals were extracted using built in functions from the *pyVHR* framework ……..50

5.2 An example command line output of the statistics reported from this project's deepfake detection model ……52

6.1 The ROC curve generated from testing this project's method on the *Deepfakes* category. The AUC score is generated by calculating the area under this curve 57

6.2 The PR curve generated from the testing this project's method on the *Deepfakes* category ……57

List of Tables

6.1 All 13 features tested on the *Deepfakes* category ……58

6.2 Features excluded one at a time and tested on the *Deepfakes* category……59

6.3 All 13 features tested on all FF++ categories……60

6.4 Results of testing different hyperparameters on the underperforming categories ……60

6.5 Final results for both methods tested on the raw FF++ dataset ……61

6.6 This project's method tested on all categories in the HQ FF++ dataset ……63

6.7 This project's method tested on all categories in the LQ FF++ dataset ……63

1 Introduction

1.1 Objective

A deepfake refers to an image or video of a subject whose appearance has been manipulated by Artificial Intelligence (AI) in some way. The objective of this project was to create a deepfake detection model. This model is intended to distinguish between real and deepfake videos by extracting Blood Volume Pulse (BVP) signals from the subjects present in video footage. This deepfake detector is aimed to be accessible for a member of the general public to use. The metrics used to achieve this goal of accessibility are as follows:

1. The deepfake detection method is to consist of a relatively fast execution time compared to other detection methods.
2. The deepfake detection method is to consist of a simple method to distinguish real from deepfake media which can be easily replicated.

These accessibility goals were the main priority when creating this project's deepfake detector. This project modifies an existing method by G. Boccignone et al. [49] that was purposed as a simple deepfake detection method. This existing method achieved an average accuracy score of 94.45% when differentiating real and deepfake videos from a publicly available deepfake detection dataset, the FaceForensics++ (FF++) dataset [56]. This method takes approximately 100 hours to execute on all 6000 videos present in the FF++ dataset. This project strives to modify this method to improve this execution time to process all of these videos in under 24 hours. Given this goal to reduce the execution time of this method, it is estimated that the results of this project's deepfake detection application will be worse than the existing method. This project will strive to reach an average classification accuracy of at least 80% on the FF++ dataset. However, approximately 83.33% of this dataset consists of deepfake videos, hence the deepfake detector would achieve an accuracy score of around 83.33% if it simply classified every

video it encountered as a deepfake. To counteract this issue, the existing method included AUC and F1 scores in order to gauge how well their model predicted real videos. The average of these 2 scores were 89.75% and 96.64% respectively. This method strives to achieve at least 80% in both of these scores as well.

1.2 Motivation

In the short time period from the creation of deepfakes in 2017 to the present day, generative AI technologies have produced a number of seemingly realistic images and videos. The videos and images these generative AI models produce have the potential to cause great harm. From creating videos of politicians stating things they never said [3], to inserting a person's face onto an actor's body in a pornographic film [1, 25], many methods exist for deepfakes to be abused. As such, a plethora of methods have been recently introduced in order to distinguish between real and fabricated media as the possibility of the general public not being able to trust the legitimacy of the videos and images they encounter is becoming a reality.

As a result of advancements in deep learning architecture, deepfakes have become more realistic since their introduction. Due to the evolving nature of deepfakes, it is desired to find information present in real videos and images that is difficult for deepfake generation tools to replicate. It is possible for different kinds of biological signals to be extracted from video footage. Deepfake generation tools have seemly been unable to replicate the biological information that can be extracted from videos of real people [44, 45, 46, 47, 48, 49, 50, 52]. Thus, the deepfake detection method used in this project will use information related to the Heart Rate (HR) of subjects in video footage to distinguish between real and deepfake videos.

1.3 Executive Summary

In this paper, a new deepfake detection model is proposed as an alternative to existing detection methods. The importance of these types of applications has been made apparent by the various malicious acts committed with the aid of deepfake technology. Section 2 discusses some examples of these cases, as well as explaining some of the positive use cases of deepfake technology, which mostly revolve around advertisements that promote some sort of charity or cause. This section also discusses many of the deep learning architectures used to generate deepfakes. The first architecture that was used for this purpose was the deep autoencoder architecture [8], however as technology progressed, different variations of the Generative Adversarial Network (GAN) [10] architecture has been more commonly used for deepfake generation in the present day. Some AI chatbots such as DALL·E [17] and Sora [18] are used to generate fake images; these tools utilise the deep learning Transformer [16] architecture to take text prompts from the user and generate images from those prompts. This chapter also introduces many of the deepfake detection methods that have been proposed, as well as the main publicly available datasets that are commonly used to train these deepfake detection methods. The FaceForensics++ (FF++) dataset [56] was chosen for this project. Finally, this chapter introduces detection methods which distinguish real from deepfake videos by extracting biological signal data from the subjects in the video footage. These types of methods are used in order to find information within video footage that separates real from deepfake videos, while also finding information that seems difficult for deepfake generation tools to learn and replicate. This project also aims to distinguish between real and deepfake videos using biological data; namely by using Blood Volume Pulse (BVP) signals.

BVP is a method of estimating the Heart Rate (HR) of a subject. This is traditionally measured using a wearable senor (that would be placed on somewhere such as the wrist) which calculates BVP signals using a method known as photoplethysmography (PPG). This method would use a light source and a photodetector, which would be present in a wearable sensor. The light would shine onto the subject's skin and a

photodetector would measure the amount of light that is reflected off the subject's skin. This measurement from the photodetector would provide an indicator of the volume of blood that is under the skin and passing the sensor at that point, thus achieving an estimate for BVP signals. From these signals, information about the subject's HR can be derived. This project uses a version of PPG called remote PPG (rPPG) which allows the BVP signals of subjects to be calculate directly from video footage. rPPG accomplishes this by analysing how the Red-Green-Blue (RGB) values of the skin pixels of the subject's face change over the course of the video in order to estimate how light is reflected off the subject's skin [41, 42].

This project's method of deepfake detection using BVP signals is a modified version of an existing method purposed by G. Boccignone et al. [49]. This existing method addressed some of the issues present in almost every detection method that uses BVP signals for video classification. These issues are related to the lack of a simple detection method in the field. The majority of detection methods that utilise BVP signal information use deep learning to achieve the best classification results; however, when using deep learning, some barriers are present in terms of how accessible and simple the method is to recreate. This is inherent in deep learning applications as they often include high training times and intensive computational and hardware requirements [61]. The existing method, which this project modifies, uses no deep learning in an effort to provide a simple and explainable method, which is in line with this project's goal of creating an accessible deepfake detection application. Upon implementing this existing method, it was found that this method took approximately 100 hours to process all 6000 videos in the FF++ dataset. Since this project's goal was to provide a simple and accessible detection method, this project set out to reduce the computation time of this existing method in order to make it more accessible for use by members of the general public. Section 3 discusses how the existing method, and this project's method differ.

Both the existing method and this project's method extract features from the BVP signals derived from each video. Section 4 describes the purpose of the features used in the existing method and this project's method. This project aimed to carry out as few

feature calculations as possible in order to reduce computation time. The existing method calculated 10,300 features per video, while this project's method reduced this to 13 feature calculations per video.

Section 5 details the creation of this project's deepfake detection method. This project used the Python package *pyVHR* [62] to extract the BVP signals from the videos in the chosen dataset. The Python package *antroPY* [78] was used to calculate various features that were extracted from each BVP signal. Once these features were extracted, a Support Vector Machine (SVM) was adopted to classify each video in the FF++ dataset as real or a deepfake.

The results of this method's implementation are compared to the results achieved by the existing method in section 6. This project's method processes all 6000 videos in the FF++ dataset in approximately 20 hours, 5x faster than the 100 hours the original method approximately took. On average, the existing method took 60 seconds to process a single video, while this project's method took 12 seconds. This project's method performed comparably well with the existing method for deepfakes that consisted of the face of a subject in a video being completely replaced by a face that was not their own. However, it performed poorly when attempting the identify deepfakes that consisted of the facial expression of the subject in the video being manipulated in some way (i.e., the mouth region of the person in a video being manipulated to make it appear as though they are saying something in the video in which they never said in the original recording).

2 Background

2.1 Overview of Deepfake Technology

This section provides an overview of modern deepfake technology (for both deepfake generation and deepfake detection) in order to provide background on some of the important topics related to this research field before explaining the details related to this project. First, examples of various applications of deepfakes are discussed in section 2.1.1, before discussing some of the main architectures that have been used to generate deepfakes in section 2.1.2. Some of the deepfake generation applications that have been used by the public are mentioned in section 2.1.3. This section then concludes by discussing the deepfake detection research field in section 2.1.4.

2.1.1 Applications of Deepfakes

Deepfake technology has been used in a variety of different applications, many of them malicious. An example of a deepfake being used for a malicious purpose was shown with the first deepfake ever created, which was posted by a user of Reddit in 2017 [1]. This user posted a pornographic video, where the face of the actress in the video was replaced with the face of a celebrity.

Deepfakes do not strictly manipulate visual media, audio deepfake technology also exists. In 2019 a company lost approximately €220,000 from a single phone call [2]. The person who contacted the company created deepfake audio of the company's CEO and convinced the employees to transfer this amount of money into their bank account.

A more recent example of the misuse of deepfake technology was seen when an AI generated video was created which consisted of the President of Ukraine, Volodymyr Zelenskyy, commanding Ukrainian soldiers to cease fighting and surrender to the Russian army [3]. Videos such as this have the dangerous potential of causing people to doubt the validity of the media they receive from their country's leaders in times of war.

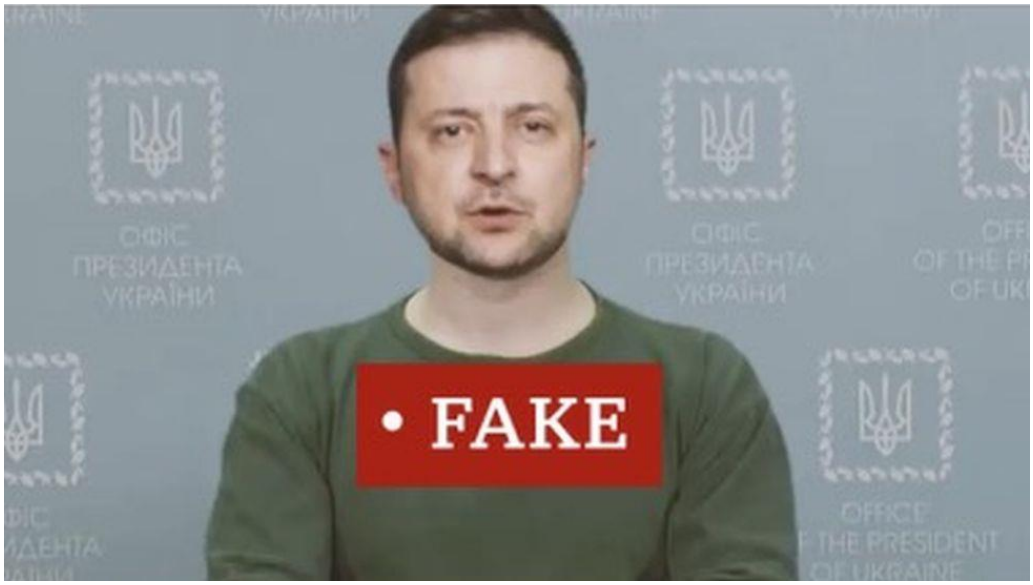

Figure 2.1: The deepfake video of the President of Ukraine, Volodymyr Zelenskyy. The deepfake video did not originally possess the red and white 'FAKE' label, that was added after the video was found to be illegitimate. Adapted from [3]

Not all deepfakes are created with malicious intent. Many of them are created with the intent of comedy. There are numerous comedic deepfake videos that can be found on websites such as YouTube. These videos are usually not trying to be realistic and can consist of celebrities saying things they would not say or being present in a movie that they did not act in. The comedy stems from the obvious absurdity of the video's content [4].

Aside from comedy, deepfakes have also been used in advertisements to generate support for various organisations. The campaign *Malaria Must Die* created an advertisement starring David Beckham. In the advertisement, he begins speaking English in his own voice, then the organisation used deepfake technology to manipulate David Beckham's mouth region in the video so that it appeared as though he spoke 9 different languages fluently [5]. The audio of each language was recorded by different people affected by malaria in order to show how the disease is still a problem all over the world.

There was also a French advertisement which displayed the impressive technical ability of their men's national football team. It did this by showing multiple short video clips of the team performing well in real football matches. It is then revealed mid-way through the advertisement that the video clips were actually taken from football games that were played by the woman's national team [6]. The player's appearances from men's national team were placed onto the players from the women's national team with the use of deepfake technology. This was carried out in order to promote women's football and challenge the public's perception of the sport.

Figure 2.2: An advertisement which consisted of the appearance of the players on the men's French national football team being superimposed onto the women's French national football team using deepfake technology. Adapted from [6]

While these examples of the ethical use cases of deepfake technology benefit important organisations, they do not outweigh the destructive potential that this technology possesses. Creating fabricated images and videos of a person committing immoral or illegal acts may wrongly damage their reputation. This worry is heightened in the case of

fake media being created of politicians. Creating media of a political figure committing an illegal or immoral act may cause the reputation of an entire political party to suffer, and thus sway the political opinions of the public. Another reason for this concern is that a lot of images and videos exist of politicians on the internet, which can be used to train a deepfake generation model. Once a deepfake video has been generated, it has never been easier to spread misinformation to the public by using social media platforms. For example, some have used AI generated images and videos of celebrities to promote different products [7]. This was done in order to deceive the public into thinking that their favourite celebrities were associated with certain organisations, which in reality they were not. The celebrities in question felt the need to warn the members of their fanbase about such practices, indicating that some of these members may have been successfully tricked into buying these products due to the false advertisements.

2.1.2 Generative AI Models

When deepfake technology was first introduced, these fabricated images and videos were created using deep autoencoders [8] for the sole purpose of swapping the face of a person in the image or video with the face of another person. A deep autoencoder uses an encoder-decoder architecture to create deepfakes. This encoder-decoder network consists of a universal encoder which encodes an image with face *A* into a lower-dimensional representation. This lower dimensional representation of the image could be decoded by decoder *A* in order to attempt to revert the image back to its original state. However, the image is instead decoded by decoder *B*, which was trained to decode features from Face *B*. The output of this network would include face *A* being recreated and replacing the face of the target in image *B*.

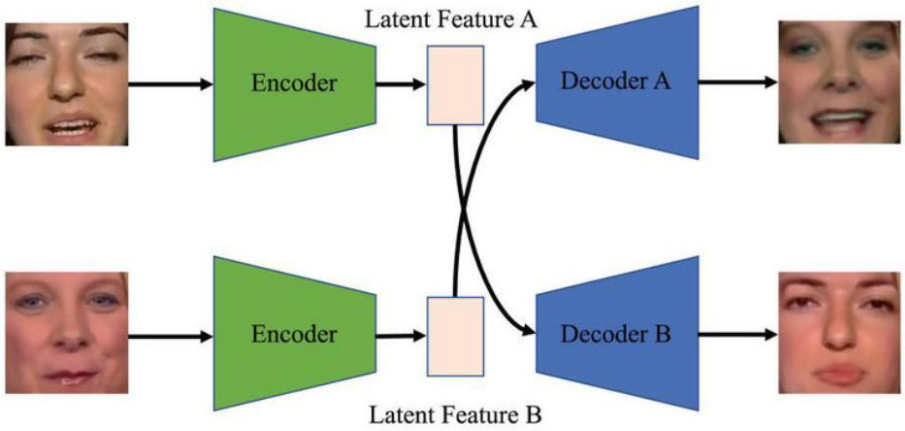

Figure 2.3: Deepfake generation using a deep autoencoder. Adapted from a larger figure from [9]

The uncanny realism of deepfakes in the present day is largely due to generative AI models moving on from deep autoencoder architecture and instead adopting the architecture of Generative Adversarial Networks (GANs). GANs were introduced in 2014 [10] and consist of two different networks: the generator, and the discriminator. The generator is tasked with generating realistic fake images. It learns the distribution of the data the network has been trained on (this could consist of real images or videos) and then takes random noise as input. It then attempts to create realistic outputs that fit the distribution of the training data from this input noise. The output of the generator is a fabricated image or video of a person who does not exist. The discriminator takes images or videos as its input. This network is then tasked with classifying which images/videos are real and which have resulted from the generator. The discriminator outputs a probability of whether a given input is fabricated or real from 0 to 1. An output of 0 represents that the image is certainly fake, whereas an output of 1 indicates that the image is certainly real. The generator then learns how the discriminator is detecting its output images and uses that information to create more realistic images. The generator is always learning how to trick the discriminator into misclassifying its fabricated output as real, while the discriminator is always trying to classify all fake and

real images correctly. These two networks are adversaries that are always attempting to outperform one another.

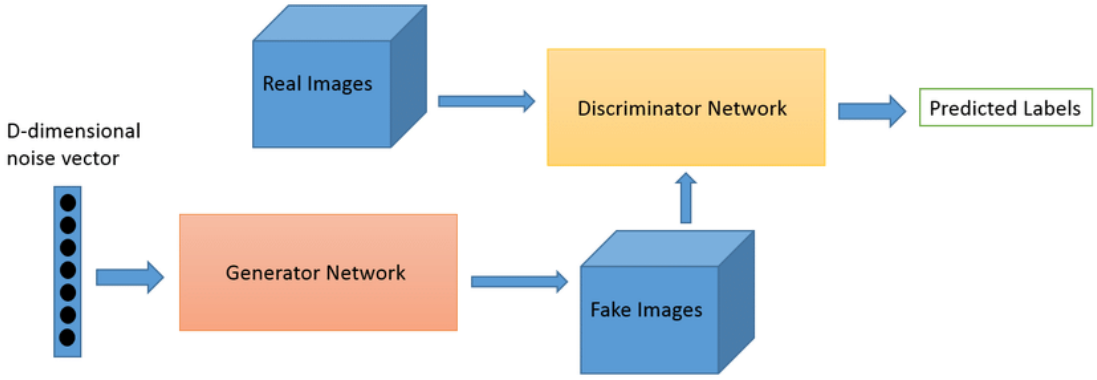

Figure 2.4: Generative Adversarial Network architecture. Adapted from [11]

Since its inception, multiple variations of GANs have been created. There once existed a large research area where researchers attempted to figure out how to translate an image from one domain to another domain using the GAN architecture. This research was related to deepfake technology as an image domain could represent the hair colour of a person in an image, this could then be swapped with the hair colour of a target person in a different image, thus creating a deepfake image. One of the first networks which figured out how to perform this operation was CycleGAN in 2017 [12].

CycleGAN introduced 2 generators and 2 discriminators. The goal of the first generator was to create a realistic fake image in the target domain from an input image. The goal of the second generator was to translate the output image from the first generator back to the source domain of the original image. This network used a *cycle-consistency loss* function which penalised the network if the output of the second generator did not resemble the input image. This was done in order to ensure that the input image was accurately translated to a different domain, without changing the core content of the

input image. Each generator had its own corresponding discriminator network which classified the generated image in each domain as real or fake.

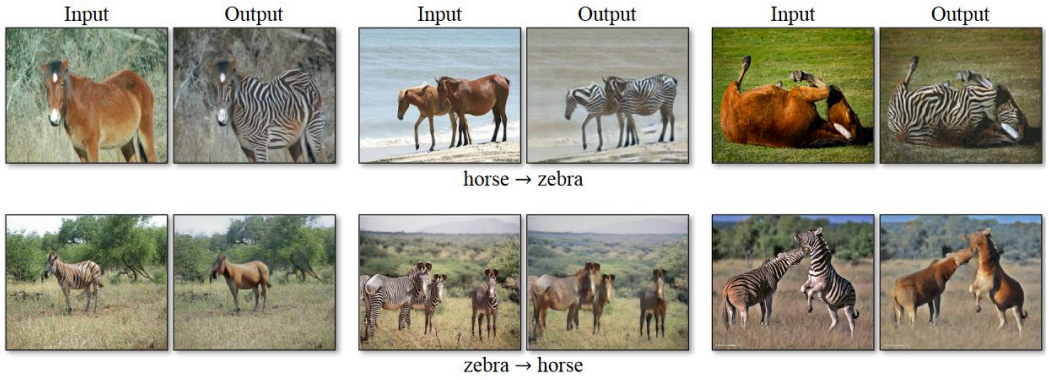

Figure 2.5: A number of input and output images from CycleGAN. Adapted from a larger figure from [12]

Many other networks have been developed that advance the architecture of CycleGAN. The first of these major advancements was found in StarGAN in 2017 [13]. A drawback of CycleGAN was that it required training 2 generators and 2 discriminators every time an image was translated from one domain to another. StarGAN introduced a method of performing all possible image domain translations while only training a singular generator and a singular discriminator. This model learned the attribute mapping for all domains, which allowed an image to be flexibly translated from one domain to another.

Nvidia introduced their own GAN in 2018 that allowed users to control how the style of a subject in an input image (age, hair colour, skin colour, etc.) was expressed in the output image. This network is referred to as StyleGAN [14] and it possessed vastly different architecture than GANs that attempted to perform the same operations at the time. One key element of the architecture of StyleGAN was that noise could be added to each layer of the network, not just at the beginning. This method is referred to as mixing regularisation and is the reason why the user had so much control over how the features

of a person in the input image could be manipulated and changed in the output image.

StyleGAN is an impressive generative neural network, however it possessed a major flaw. Many deepfake images and videos produced by this network possessed artifacts in the form of water-like blobs in the image. Another issue included facial features staying in the same location after a face had moved in a deepfake video. These errors were attributed to the progressive growing technique that StyleGAN employed. This technique refers to training a GAN to output low resolution images and then slowly increasing the output resolution until the output is at the highest resolution. This method was changed to an architecture with skipped connections (skipping some layers in the deep learning model) in Nvidia's updated model, StyleGAN2 [15]. This updated GAN produced deepfake images and videos without the previously mentioned erroneous artifacts.

Moving on from GAN models, members of the public have begun to adopt AI image generation tools which create images based off of text prompts from the user. These image generation tools make use of the Transformer architecture [16], which was originally developed by Google to further research in the field of Natural Language Processing (NLP), which is a field that strives to teach computer algorithms to understand and generate text. From this Transformer architecture, many tools have been created to generate images based on the text prompts that are received from a user. These tools are widely available to the public, some are free to use, and some require a subscription to use. Examples of these AI image generation tools are DALL·E [17] and Sora [18].

2.1.3 Deepfake Generation Applications

The technology required to create deepfakes is widely available to the public. There are mobile and web applications that exist for the sole purpose of generating deepfake images and videos, eliminating the need for a person to create and train their own deepfake generation model. A number of these applications were introduced soon after

the first deepfake video was uploaded to Reddit in 2017. These applications include FaceSwap [19], FaceApp [20], and DeepFaceLab [21] to name a few.

Some applications which were developed more recently do not produce realistic results due to the creator's intent to create deepfakes for the sake of entertainment, and not for malicious intentions; an example of this practice is shown in the application Deepfakes Web [22].

The public appears to have a great fascination with deepfake technology. This can be seen with the release of the mobile application Zao in China [23]. It was the most downloaded free app in China the month it released in 2019 [24]. The application allowed users to upload an image from their phone and from this image they could create deepfake images of themselves with different hairstyles, swap faces with celebrities, and more.

At the time of writing, AI image generation tools, such as the previously mentioned DALL·E and Sora, have seen some use by members of the public. Many of these tools have been used to generate images of fake people, or real people in scenarios that never occurred. A recent example of these types of tools being used by a member of the public occurred when a user of the social media platform X (formerly known as Twitter) posted sexually explicit, AI generated content of pop star Taylor Swift [25]. The post was viewed by 10s of millions of other users before it was removed by the social media platform. This event demonstrated the ease in which members of the public can now use deepfakes for malicious purposes with the availability of these AI image generation tools. Because of this, the creation of formidable deepfake detection tools has never been more important.

2.1.4 Deepfake Detection

When deepfake technology was first introduced, the fabricated images and videos produced were not convincingly realistic. Early deepfake detection methods took

advantage of the limitations of generative AI tools at the time. These included creating networks that investigated the image or video for warping artifacts [26, 27]. An example of a warping artifact is blurring around a person's face in an image or video as a result of attempting to blend the subject's original face with that of another person. They would also search for things such as the head movements of the subject in a video not matching the orientation of their face [28]. The colour of objects in the images and videos would also be investigated for irregularities, as a limitation of GANs during the emergence of deepfake technology was that they would convey colour space information in a different way a real image or video that was taken by a Red-Green-Blue (RGB) camera would [29]. These methods were able to successfully find a clear distinction between real and fabricated media.

Since their emergence in 2017, deepfakes have become uncanny in terms of their realism, which has caused the deepfake detection community to pursue more sophisticated detection methods.

All deepfake detection methods mostly follow the same detection pipeline. The current methodology of detecting deepfake images and videos includes a pre-processing step, where the face of the person in the image or video frame is extracted. This facial image is then entered into a Neural Network (NN), which extracts features from the facial image. Alternatively, handcrafted features are extracted directly from the facial image without the use of NNs. The information from these features is then inputted into a classifier. The image or video is then classified as real or fake. Most detection methods use supervised learning, where they are trained with data that have labels discerning whether they are fake or real samples.

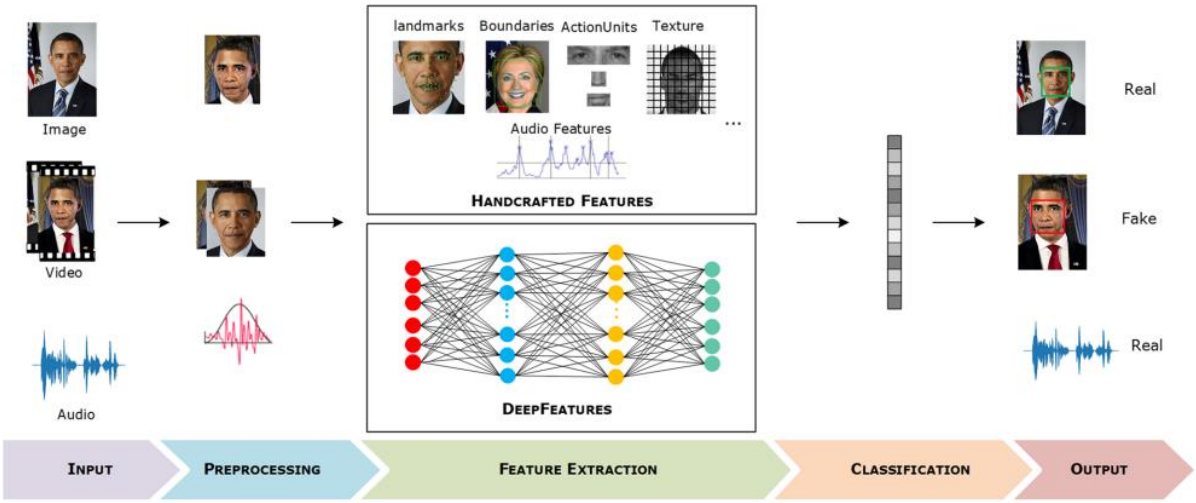

Figure 2.6: The average deepfake detection pipeline. Adapted from [1]

Many of the NN models that were first used for deepfake detection were repurposed Convolution Neural Networks (CNNs) that were originally used for other types of image classification, such as identifying an object in a scene. XceptionNet [30] is a deepfake detection model that is still sometimes used when comparing the performance of newer deepfake detection models. This model is based on Google's Xception image classification network [31].

Another CNN based network that has been adopted for deepfake detection is EfficientNet [32]. EfficientNets are a family of CNN models whose purpose was to introduce a standardised way to scale up CNN models instead of manually and arbitrarily tuning parameters like depth, width or image resolution. This led to increased accuracy and efficiency when scaling up these networks. In order to achieve this, the creators introduced a 'compound scaling' method which balances the 3 previously mentioned parameters when they are being scaled up. The result was a family of models that were 8.4x smaller and 6.1x faster than existing CNNs at the time of the network's release. The models also showed good transfer learning accuracy, which explains why the deepfake detection community began utilising these networks in their models. Kaggle, in association with companies such as Meta and Microsoft, hosted a Deepfake Detection

Challenge (DFDC) competition in 2020 where competitors submitted models that differentiated between real and deepfake videos [33]. 5 winners were announced from that competition and of those 5 winning submissions, 2 utilised EfficientNet, and 2 utilised XceptionNet architecture.

Other notable CNN network architectures that are used for deepfake detection are ResNet [34] and MesoNet [35]. ResNet is a CNN architecture that was introduced in 2015 and its aim was to decrease the training time of deeper NNs. MesoNet was introduced in 2018 and was specifically created for the purpose of deepfake detection.

There have been attempts to improve the performance of deepfake detection models by adding to existing CNN architectures with more modern deep learning architectures. Researchers have found success in using deep learning architectures such as Vision Transformers (ViTs) [36], and different Attention mechanisms [37], in addition to CNN architectures to produce good results in frequently used publicly available deepfake detection datasets.

In addition to deep learning methods, there have been many deepfake detection methods that have been purposed that do not utilise deep learning and instead extract handcrafted features from videos in order to distinguish between real and deepfake content. These methods include analysing the head and facial movements of the subjects in video footage, and searching for manipulation traces such as blurring artifacts around the face region that are a result of attempting to replace the face of a subject in a video with the face of another person, which were both mentioned at the beginning of this section.

Some deepfake detection methods have found success in using biological signal data extracted from video footage in order to distinguish between real and deepfake videos. This project also uses biological signal data to discriminate between real and deepfake videos. The type of biological data that is used for this detection task is Blood Volume Pulse (BVP) signals, which are used to estimate the Heart Rate (HR) of a subject. The

method of extracting this biological data from video footage is explained in section 2.2.1.

2.2 Related Work

This section will discuss some of the research that has been carried out to investigate the detection of deepfakes using BVP signals. It is worthy to note that extracting BVP signals from video data is not the only method in which researchers have attempted to distinguish between real and deepfake videos by using biological signal data. Research has been presented that has shown the detection of deepfakes by exploiting the lack of natural eye-blinking information present in deepfake videos [38, 39]. While impressive, this method becomes unreliable when the subject in the video is suffering from mental illness, or with problems related to their nerve conduction pathways as their eye blinking frequency can become much more varied in these cases [39]. Thus, BVP signals were used for this project's detection application as deepfake detection models which utilise these signals don't suffer the same drawback that models which utilise eye-blinking rate do. Furthermore, due to the involuntary nature of biological signal data (humans can't deliberately control their HR or eye-blinking rate, for example), it is estimated that it would be difficult for deepfake generation tools to replicate this information.

Section 2.2.1 details what BVP signals are and how they are extracted from video footage. Section 2.2.2 discusses some of the deepfake detection applications that have been created using BVP signals.

2.2.1 BVP Signal Extraction from Video Footage

This project focuses on analysing BVP signals extracted from subjects in videos to distinguish between real and deepfake content. BVP is a biological signal that can be used to derive the HR of a subject by measuring the volume of blood passing through blood vessels in the area of the subject that is being analysed (in this case, this area is

the face). These signals are traditionally measured through a wearable device (that would be worn in an area such as the wrist) via a method known as photoplethysmography (PPG). This method includes a light source and a photodetector in the wearable device. The light source shines onto the skin and a photodetector measures the amount of light that has been reflected off of the skin. The measurement from the photodetector is used as indicator of the volume of blood that is passing the sensor at that moment in time, hence it is used to estimate the BVP signals of the subject.

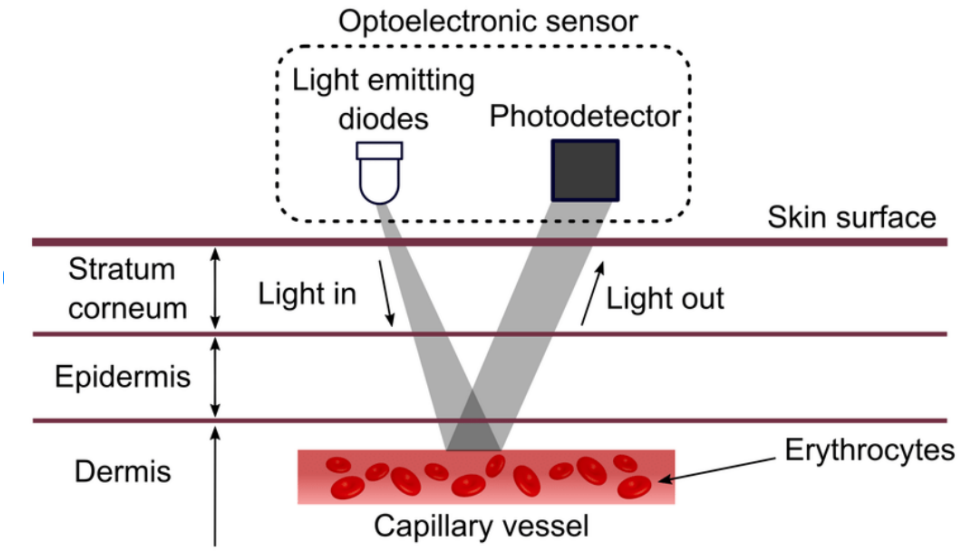

Figure 2.7: The process of measuring BVP signals via PPG. Adapted from [40]

For the purposes of deepfake detection, it is desired to extract BVP signals directly from video footage. This can be achieved via a variation of PPG known as remote PGG (rPPG). This method extracts BVP signals from a subject in a video by analysing the way in which light is reflected off the video subject's skin. This is carried out by measuring the change in the Red-Green-Blue (RGB) values in the subject's skin pixels. As this method uses colour changes to estimate the BVP signals of the subject in the video, the background of the video needs to be removed as any change in the colour in the background of the

scene would introduce error into the measurement. Thus, the subject's face in each frame of the video is located and extracted. From these facial images, the movement of the mouth and eye regions of the face could still introduce errors to the measurement due to these movements causing a change in the RGB values of the pixels in these regions. As a result of this, these regions of the facial image are located and removed. The resulting image contains only the skin pixels belonging to the face of the subject in the video. The RGB values of the skin pixels are then directly used to estimate the BVP signals of the subject in the video.

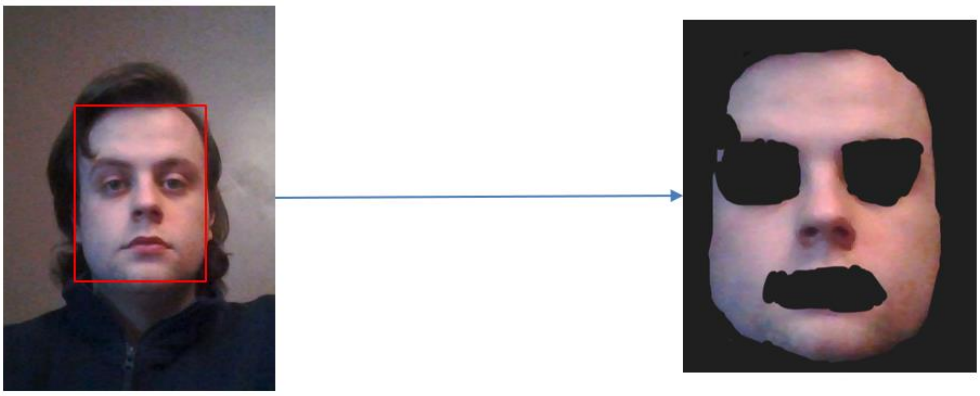

Figure 2.8: The process of extracting the skin pixels from the face of a subject in a video in order to analyse the change in colour of these skin pixels, and thus calculate the BVP signals of the subject in the video.

Many different rPPG methods exist in order to measure this change in RGB pixel values. For example, the green colour channel can be used as an estimate for the subject's BVP signal as the signal of the green channel closest resembles the BVP signals of the subject. This method is known as GREEN rPPG [41]. There also exists rPPG methods which utilise information from all 3 colour channels. An example of one of these methods is Position Orthogonal-to-Plane (POS) rPPG. POS rPPG calculates biological signals by projecting the RGB signals from the skin pixels of the subject's face, onto a plane that is orthogonal to

the skin tone of the subject [42].

rPPG was originally created to measure a medical patient's vital signs from a distance, instead of using a more invasive, physical measuring device [43]. It has since been adopted for use to distinguish between real and fake video footage as it is believed that BVP signals extracted from current deepfake videos will differ from BVP signals extracted from real videos. This belief has been validated as many deepfake detection methods have been able to successfully distinguish between real and fake videos by analysing BVP signal data.

2.2.2 Methods of Deepfake Detection Using BVP Signals

A number of researchers have successfully created methods which distinguish between real and deepfake videos using rPPG [44, 45, 46, 47, 48]. All of these aforementioned methods incorporated the use of deep learning in some capacity in order to produce the best possible classification results. This is the case for the majority of deepfake detection models purposed in this research field; only a singular method was found that did not utilise any deep learning algorithms [49]. The creators of this method point out the lack of explainable methods in this field, as when deep learning is used, the ability to fully explain what features differentiate real from deepfake videos is lost. This led them to create their own simpler, explainable detection method.

A deepfake detection model that uses rPPG based methods to distinguish real videos from fake videos is *FakeCatcher* [50]. This method extracted 126 different features from their calculated biological signals and used a Support Vector Machine (SVM) for classification. This method achieved an accuracy score in the region of 76.78% when tested on their own dataset, which consisted of over 140 videos sourced from the internet. This was achieved by using a 60/40 training/testing split. It also achieved an accuracy score of 75% when using the same training/testing split on the publicly available FaceForensics (FF) dataset [30], which consisted of over 1000 videos. The SVM was eventually replaced with a CNN classifier for the final model, which achieved a final

accuracy score of 96% and 91.07% on the FF dataset and their private dataset respectively. In 2022, Intel adopted this *FakeCatcher* model [51].

Accessibility of methods in this research field is a major issue. When conducting research for this project, no method was found which publicly provided its codebase; thus, anyone who wishes to make their own deepfake detection application based on rPPG methods introduced in the literature must implement these methods themselves. At the time of writing, there is currently one rPPG based deepfake detection method which made its pre-trained deep learning model publicly available on GitHub. This model is the DeepFakesON-Phys model [52] and is based off the DeepPhys architecture [53], which is a deep learning model that was trained to extract biological signal data from real subjects in videos by using a Convolutional Attention Network (CAN). DeepFakesON-Phys was tested on 2 widely used deepfake datasets, the Celeb-DF [54] and Deepfake Detection Challenge (DFDC) Preview [33] datasets (the content of these datasets is explained in section 2.3); it achieved an AUC and accuracy score of 99.9% and 98.7%, and 98.2% and 94.4% on each of these datasets respectively for frame level video predictions.

It is great that a model which achieved such good results is publicly available, however there are some caveats if one wishes to use this model to classify a video as real or a deepfake. Firstly, this video requires a large pre-processing step. This model takes 2 inputs, the first input is a normalized image of the current frame, and the second input is a normalized difference image of the current frame and the next frame. These inputs must be extracted from every frame of a video and saved to the user's local machine before the detection model can be used to classify a video as real or fake. This is a minor issue when the model is tested with a small number of videos of relatively short duration and low frame rate, but when this model is tested on a large dataset, this pre-processing step could take up a lot of time and space. The second caveat is that the predictions of this model are made per frame of the video, so it is up to the user to manually compute the video level predictions from this model. While it is good that the creators of this method made their pre-trained model available for the public to use as

they wish, their method could be modified to be more accessible for the average person to use. There are applications in which people can use in order to create deepfakes from their personal computer, or even mobile phone, with ease; the same standard of accessibility should ideally be applied to deepfake detection applications created in all fields of research, including this one.

2.3 Datasets for Deepfake Detection

Many of the datasets researcher's use to train and test their deepfake detection models were created by the researchers themselves and were not made publicly available. However, there are a number of publicly available datasets that were created by a number of different researchers. This section will introduce and provide an overview of some of these publicly available datasets.

The Celeb-DF [54] dataset contains 590 real and 5639 deepfake videos. The subjects in the videos are diverse in terms of sex, age, and ethnic background. The dataset consists of YouTube videos of Celebrities. The real videos in this dataset were sourced directly from YouTube and the deepfake videos were created by swapping the faces of different pairs of subjects present in the real videos. The average length of a video in this dataset is 13 seconds at 30 frames per second.

The WildDeepfake [55] dataset is a small dataset that can be used in addition to other datasets. It contains 707 deepfake videos completely extracted from the internet. The technique of image forgery is not explicitly stated in the research paper associated with this dataset as the videos are sourced from deepfake videos on the internet.

The Deepfake Detection Challenge (DFDC) [33] dataset was created for the DFDC challenge which was hosted on Kaggle in association with companies like Meta and Microsoft. Multiple methods were incorporated when generating this dataset's deepfake videos so that when a model is trained with this dataset, the model can identify multiple different deepfake techniques. The dataset includes over 100,000 video

clips of 3,426 paid actors.

2.3.1 This Project's Dataset

The FaceForensics++ (FF++) dataset was chosen for this project [56]. This dataset was chosen because this project's method of deepfake detection is a modification of an existing detection method, and this method was trained and tested this dataset. Thus, in order to provide a comprehensive comparison between this project's method and the existing method, the same dataset was used for this project.

The FF++ dataset was introduced in 2019 and was an update to the original FaceForensics (FF) dataset that was introduced in 2018 [30]. It contained 5000 videos, 1000 of which were real videos, and the remaining 4000 videos were deepfake videos. These fake videos were created using 4 different deepfake techniques, each technique was used to create 1000 deepfake videos for this dataset. These techniques were *Deepfakes* [57], *Face2Face* [58], *FaceSwap* [19], and *NeuralTextures* [59]. This dataset is available in 3 different video qualities. These qualities are (in order of highest quality to lowest quality):

1. The highest quality (raw), which possesses a lossless compression factor.
2. high-quality (HQ), which possesses a compression factor of 23.
3. low-quality (LQ), which possess a compression factor of 40.

An additional 1000 deepfake videos, created by using the *FaceShifter* [60] method, were added to this dataset after its initial release.

This dataset contains 2 types of deepfakes. The first type of deepfake is facial replacement deepfakes, which attempt to replace the face of a source person with the face of a target person. The *Deepfakes*, *FaceSwap*, and *FaceShifter* categories in the FF++ dataset are all facial replacement deepfakes. The second type of deepfake is facial

reenactment deepfakes, which attempt to transfer the facial expression of a source face onto a target face. The *Face2Face* and *NeuralTextures* categories in the FF++ dataset are both facial reenactment deepfakes.

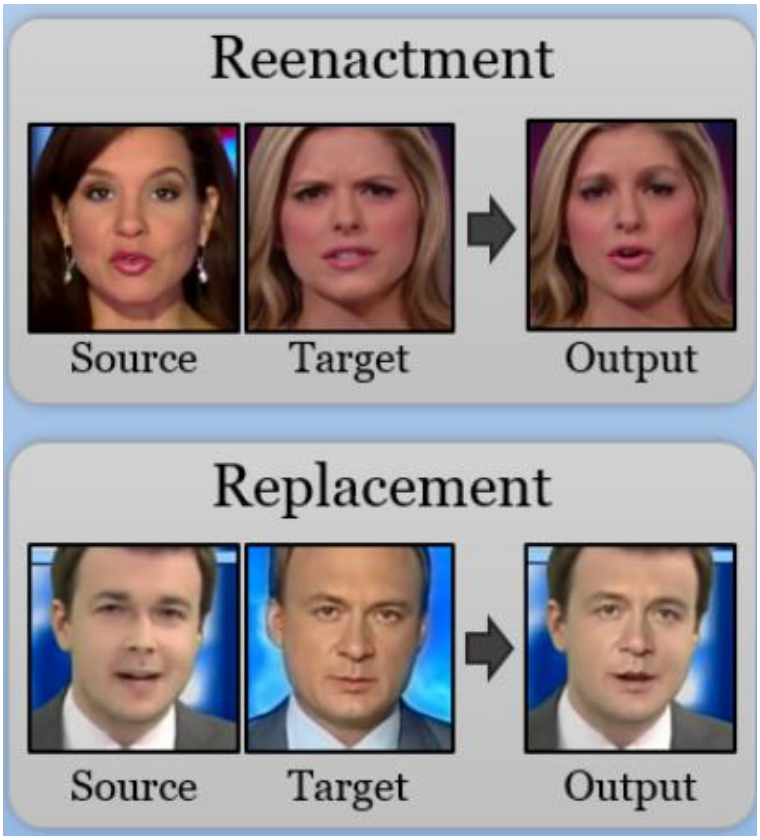

Figure 2.9: An example for both a facial reenactment deepfake and a facial replacement deepfake. Adapted from a larger figure from [56]

3 Explanation of the Deepfake Detection Application

As mentioned previously, this project's method of deepfake detection is a modified version of an existing detection method. Before the details of the methodology of this project are explained in section 3.2, the previously existing methodology will be explained in section 3.1 in order to provide the context behind this project's methodology.

3.1 Previous Methodology

As discussed in section 2.2.2, the majority of deepfake detection methods that use rPPG methods to differentiate between real and deepfake videos utilise deep learning algorithms to do so. While researching this field, only a singular detection method that did not utilise deep learning was found. To reiterate, the goal of this project is to create a deepfake detection model that is accessible for the public to use and recreate. Detection methods that used deep learning were largely not considered for this project due to the hardware requirements and long training times associated with many deep learning networks [61].

The researcher's motivation behind creating the deepfake detection method that this project is derived from was the lack of explainable methods in this research field. The researchers explained that once a method includes deep learning, the method loses its explainability as there is no way to gauge what is happening when the NN is making its calculations. This explainable method was created by G. Boccignone et al. [49] and it uses the *pyVHR* python framework [62] in order to manually extract BVP signals from all videos present in the FF++ dataset.

Firstly, each video in the dataset is split up into multiple overlapping time frames (also referred to as video clips throughout this paper). The number of time frames that a

video is split up into depends on the length of the video, as well as the chosen window size. The larger the window size, the smaller the number of time frames that are analysed. This is carried out in order to provide the classification model with more data in the hopes that the classification model will find a better separation between real and deepfake videos. This results in the classification model outputting predictions for each time frame, rather than each video. The predictions for each video were retrieved by selecting the most frequently predicted label of the video's time frames.

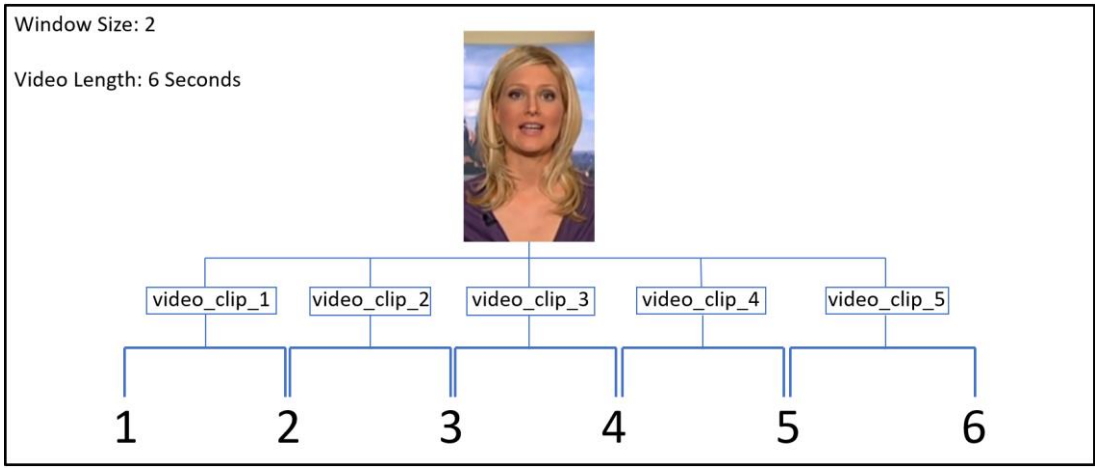

Figure 3.1: An example of splitting a video into multiple overlapping video clips. Every 2 second overlapping interval of the 6 second video is analysed and split up into 5 videos clips. The still image of a subject of a video was adapted from [56]

The skin pixels from the face of the subject in each time frame were extracted at every frame. These skin pixels were extracted using a *Convex-Hull* extractor [62], which treats the skin of the face of the subject as a number of convex-hulls, which are polygons that enclose all of the facial landmarks in a region of the face. The convex-hulls that were calculated using facial landmarks from the mouth and eye regions of the face were subtracted from face of the subject in the video. This resulted in an image consisting of only the skin pixels of the face. Different Regions of Interest (ROIs) were analysed from these skin pixels. BVP signals were then extracted from each ROI using the POS rPPG

method. Approximately 100 ROIs were analysed at each time frame, and a BVP signal was present at each of these ROIs.

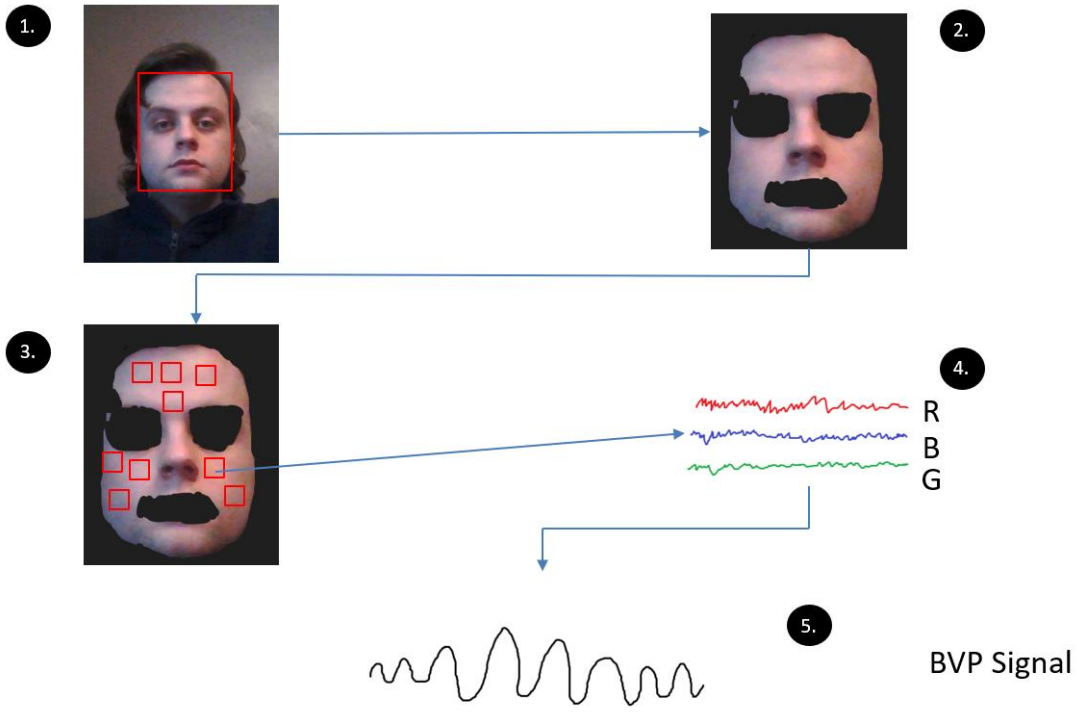

Figure 3.2: The pipeline of extracting BVP signals from video footage. The face from every frame of the video is extracted. The mouth and eye regions are identified and removed from this facial image because movement from these regions causes erroneous results when using rPPG. This modified facial image is then split up into different ROIs. The pixel's RGB values within each ROI are then analysed, and the BVP signal is derived from these values via rPPG methods

A total of 6 features were extracted from each of these BVP signals at every time frame (each of these features are further explained in section 4). 4 of these features measured to complexity of each BVP signal's shape. 2 of these complexity related features were variations of the Fractal Dimension (FD) and 1 was related to the Entropy of the signal. These 4 complexity related features were the Petrosian FD [63], the Higuchi FD [64], the

Singular Value Decomposition (SVD) Entropy [65], and the Zero-Crossing Rate of the signal. The remaining 2 features analysed the similarity of each unique BVP signal pair. This was carried out by comparing every unique pair of BVP signals from each ROI at every time frame, resulting in 4,950 calculations made per feature for each time frame. The signal similarity related features were the Mutual Information and the Magnitude Squared Coherence Estimate (which is dubbed by the creators of this method as the "Spectral Similarity").

All complexity related features were calculated identically. For each complexity related feature, the BVP signal from each ROI on the face was analysed and the feature was computed directly from the BVP signal. As there were 100 ROIs present on the face, this resulted in each feature having 100 values for each video time frame, 1 for each ROI. Each of these 100 features were then averaged in order to achieve a singular value of the feature that represented the entirety of the time frame.

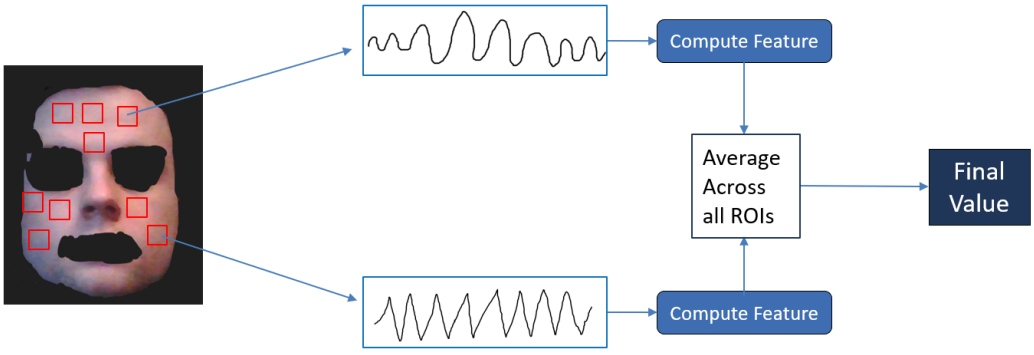

Figure 3.3: The extraction of signal complexity related features using the existing method. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Features are then calculated from all of these BVP signals directly. Each feature value is then averaged to create a singular feature value that represents the entire video clip

For each signal comparison related feature, the BVP signal at each ROI was analysed. These BVP signals were then compared with every BVP signal in every other ROI that made a unique signal pair. The similarity of each signal in the frequency domain was calculated with the use of the Magnitude Squared Coherence Estimate. This feature measured the similarity of each pair of BVP signals at multiple frequencies, meaning that an array of values was produced when analysing each BVP signal pair. These values were averaged together in order to produce a singular scalar value that represented the spectral similarity of each signal pair. This was carried out for all unique BVP signal pairs. The resulting measurements from these calculations were all averaged together to achieve a singular value that represented this feature for the entire video clip.

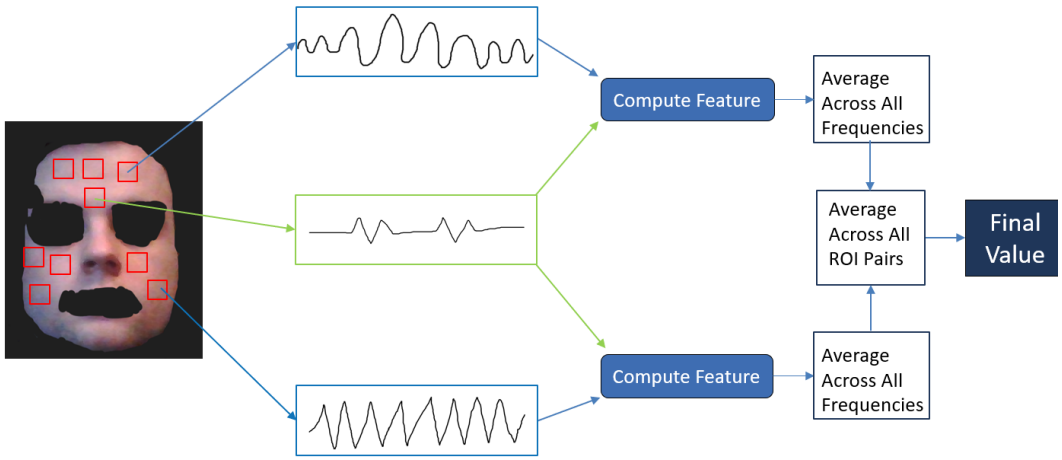

Figure 3.4: The extraction of signal comparison related features using the original method with the Magnitude Squared Coherence Estimate. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Features are then calculated, comparing each BVP signal to every other BVP signal. Each of these signals are compared in the frequency domain. The result of the feature calculations is an array of values representing the similarities of each BVP signal at different frequencies. Each of these values are then averaged to achieve a singular scalar value of each BVP signal pair comparison. Each of these feature values are then

averaged to create a singular feature value that represents the feature measurement for the current video clip

The similarity of each signal in the time domain was calculated with the use of Mutual Information. This measurement produced a singular scalar value which represented the similarity of each unique pair of BVP signals. The standard deviation of these resulting measurements was calculated in order to achieve a singular value that represented this feature for the entire video clip.

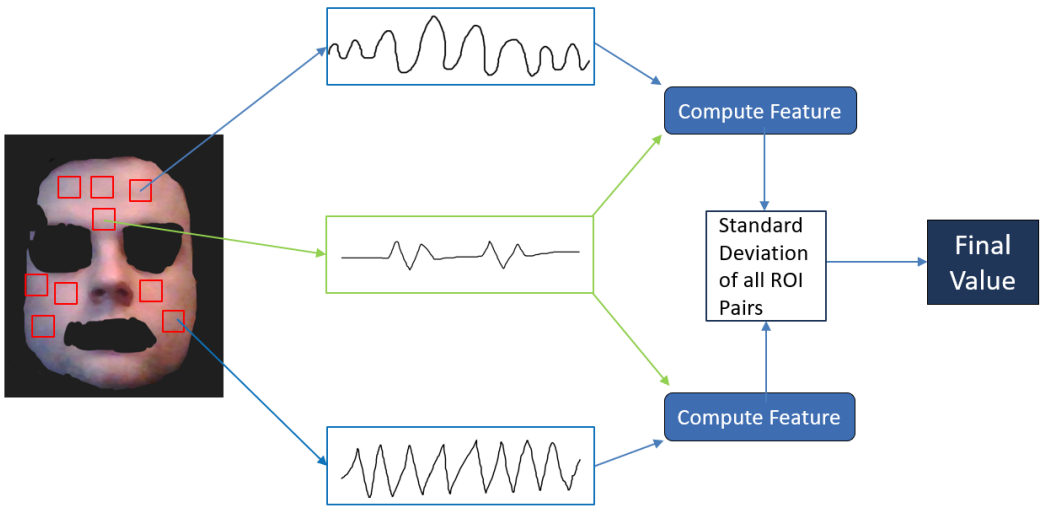

Figure 3.5: Extraction of signal comparison related features using the original method with Mutual Information. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Features are then calculated, comparing each BVP signal to every other BVP signal. Each of these signals are compared in the time domain. The result of these feature calculations is a singular value which provides a measurement of the similarity of each BVP pair in the time domain. The standard deviation of these feature values is then calculated to create a singular feature value that represents the feature measurement for the current video clip

The result of all of these feature calculations was a 6-D feature vector that was used to train a Support Vector Machine (SVM) on video data at the time frame level. The predictions of the SVM were at a time frame level, and the video level predictions were calculated by selecting a given video's most frequent time frame level prediction. This method produced an average accuracy, AUC, and F1 score of 94.45%, 89.75%, and 96.64% respectively on the highest quality version of the FF++ dataset.

This detection pipeline culminated in a long computation time. When attempting to recreate this methodology, the model was trained on 40% of the videos from the FF++ dataset and the program completed its execution approximately 40 hours after it began. Therefore, it is estimated that this method would require 100 hours to process all videos in the FF++ dataset. The goal of this project was to modify this approach in order to reduce this execution time, while preserving the simplicity and good results of the method as much as possible.

3.2 Proposed Methodology

As stated previously, this project attempts to modify the method created by G. Boccignone et al. in order to lower its computation time and preserve its simplicity and performance as much as possible. The long computation time is partially related to the method of calculating the 4 complexity related features at every ROI, and then averaging those feature values to get a singular value for each feature at every time frame. However, it is mainly due to the 2 signal similarity related features that compare every unique pair of BVP signals. As the face is split up into 100 ROIs, this leads to 4,950 calculations per feature at each time frame, or 9,900 calculations made in total per time frame.

The method proposed for this project removes both of these signal similarity related features in the hopes of drastically decreasing the execution time of this method. To

further lower this execution time, instead of calculating the 4 complexity related features at each ROI, and then averaging the values of these features at each ROI to result in a singular scalar value that represents a singular time frame, each of the BVP signals at each ROI were averaged together, resulting in a singular BVP signal that represented a given time frame. The complexity related features were then extracted from this BVP signal at each time frame. Reducing the number of complexity related features calculated at each time frame from 400, to just 4. Overall, the number of features calculated per time frame was decreased from 10,300 to 4.

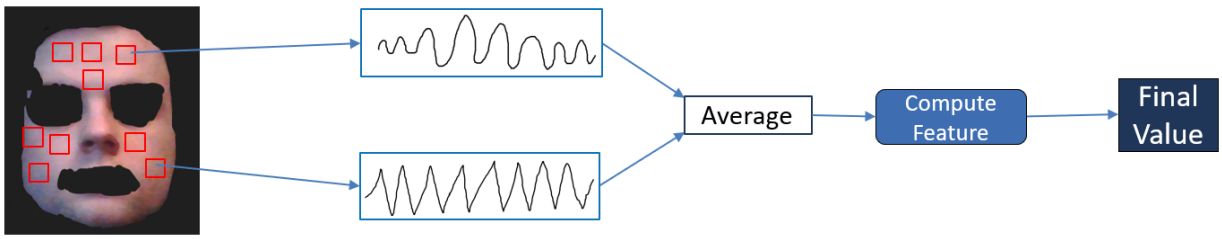

Figure 3.6: The extraction of signal complexity related features using this project's method. For each video clip, the modified facial image is split up into a number of ROIs. A BVP signal is derived for each ROI. Each of these signals are then averaged to generate a singular BVP signal that represents the current video clip. Features are then calculated from this BVP signal to output a singular feature value that represents that particular feature for the current video clip

To compensate for the loss of information when transitioning to this method, 9 more features were added to measure the complexity of the BVP signals. These features were the Katz FD [66], Detrend Fluctuation Analysis (DFA), Spectral Entropy, Permutation Entropy, Approximate Entropy, Sample Entropy, Hjorth Activity, Hjorth Mobility, and the Hjorth Complexity of the signal (these features are further explained in section 4). These features were chosen as 8 of them were also used in the existing method purposed by G. Boccignone et al. before they optimized their method during testing to find the 6

features which contributed the most to the classification accuracy of their model. The only feature that is included in this project's method but not the existing method was the Hjorth Activity. This feature is simply the variance of a signal, and thus it was added to this project due to its simple implementation in Python in order to use all 3 Hjorth parameters for this project's detection method.

Once these features were extracted, an SVM was adopted for the task of classification. This SVM was trained on all but 1 of the deepfake categories present in the FF++ dataset at a time, as well as 80% of the real videos found in this dataset. It was then tested on the videos created by the deepfake category that was left out of training, as well as the remaining 20% of the real videos. This was also done by the original method in order to demonstrate how well the model generalised when new types of deepfakes were introduced, as this method would ideally be used when new types of deepfakes are discovered in the real world. The hyperparameters for this SVM were chosen via Randomized Search Cross Validation. Like the original method, the predictions of this SVM were at a time frame level, and the video level predictions were calculated by selecting the video's most frequent time frame level prediction.

4 Explanation of Extracted Features

In this section, the various features that were extracted from the BVP signals by the detection method purposed by G. Boccignone et al., as well as the detection method proposed in this project, are explained. The majority of the features that are discussed in this section provide a measurement for the complexity of the shape of the BVP signals, while the remainder measure the how similar each unique pair of BVP signals are to each other. A summary of the purpose of each of the features is provided in section 4.1. If the reader does not wish to delve into the mathematical methods by which these features are calculated, the reader can read this summary, skip sections 4.2 and 4.3, and move on to subsequent sections of this paper without the risk of not understanding the content of these subsequent sections.

4.1 Summary

The Zero-Crossing Rate provides a measure of the complexity of a signal's shape by calculating the number of times the signal crosses the x-axis. Each of the remaining features that provide measurements for the complexity of each signal can be placed into 1 of 3 categories. The first of these categories is the Fractal Dimension (FD). FDs are used to describe the complexity of a given signal and is a measurement that has been frequently utilised when analysing biological signals as they compute biomarkers [67]. These biomarkers indicate the condition of an organism's health. The features that are included in this category are Detrend Fluctuation Analysis (DFA), Petrosian FD, Higuchi FD, and Katz FD. The second category is Entropy. These features provide a measurement of the complexity of a signal by measuring the amount of uncertainty, or randomness present in the signal. The features that are included in this category are Spectral Entropy, Approximate Entropy, Sample Entropy, Singular Value Decomposition (SVD) Entropy, and Permutation Entropy. The last of these categories are the Hjorth parameters, which provide statistical properties of a signal in the time domain using different measurements of the variance of the signal. These parameters are the Hjorth

Activity, Hjorth Mobility, and Hjorth Complexity of the signal.

There were 2 features which were measurements of signal similarity that were used by the method purposed by G. Boccignone et al. These features were the Magnitude Squared Coherence Function, which measures the similarity of a signal pair in the frequency domain, and Mutual Information, which measures the similarity of a signal pair in the time domain.

4.2 Features Used in the Original Method

This section will discuss the features that were extracted from the BVP signals in the method proposed by G. Boccignone et al. There were 2 types of features that were used in this method:

1. Measurements of signal complexity
2. Measurements of signal similarity

The features that were used for measuring signal complexity were the Petrosian Fractal Dimension (FD), the Higuchi FD, the Singular Value Decomposition (SVD) Entropy, and the Zero-Crossing Rate of the signal.

The features that were used for measuring signal similarity were the Mutual Information, and the Magnitude Squared Coherence Estimate (or the "Spectral Similarity", as the researchers of the original method referred to it).

4.2.1 Measurements of Signal Complexity

Fractal Dimensions are used to describe the complexity of a given signal and is a measurement that has been frequently utilised when analysing biological signals as they compute biomarkers [67]. These biomarkers indicate the condition of an organism's health. Many different methods exist to calculate the FD of a signal. These methods provide a measurement of self-similarity and disorder in a given signal.

Petrosian Fractal Dimension

The Petrosian FD calculates the FD of the signal by converting the signal into a binary sequence. There are different ways to compute this binary sequence; for example, it can be created by having a 1 signify the difference between 2 data points exceeding a standard deviation magnitude, and a 0 if the inverse is true. The formula for this calculation is then [67]:

$$\frac{\log_{10} n}{\log_{10} n + \log_{10} n\left(\frac{n}{n+0.4N_{\Delta}}\right)} \quad \textbf{(1)}$$

Where:

- n is the length of the binary sequence
- N_{Δ} is the number of sequential bit changes in the binary sequence.

Higuchi Fractal Dimension

Higuchi FD is derived from analysing the signal as a time sequence $x(1), x(2), \ldots x(N)$. From these time sequences, a time series can be calculated [68]:

$$X_k^m : x(m), x(m+k), x(m+2k), \ldots, x\left(m + int\left[\frac{(N-k)}{k}\right]k\right) \quad \textbf{(2)}$$

Where:

- m is the initial time value (m = 1, 2, 3, …, k)
- k is the time interval between consecutive data points (k = 1, 2, 3, …, k_{max})
- k_{max} is a free parameter
- $int[r]$ is the integer part of the real number r
- N is the length of the time series X

The length of the signal $L_m(k)$ is then calculated for each k time series using the following equation:

$$L_m(k) = \frac{1}{k}\left[\left(\sum_{i=1}^{int\left[\frac{(N-k)}{k}\right]} |x(m+ik) - x(m+(i-1)k)| \right) \frac{N-1}{int\left[\frac{(N-k)}{k}\right]} \right] \quad \textbf{(3)}$$

Where:

- $\frac{N-1}{int\left[\frac{(N-k)}{k}\right]}$ is a normalization factor

An array of the mean values of $L_m(k)$, $L(k)$ is then calculated using the following equation:

$$L(k) = \frac{\sum_{m=1}^{k} L_m(k)}{k} \quad \textbf{(4)}$$

Finally, the Higuchi FD is calculated by using the line of best fit in a plot of $\ln(L(k))$ and $\ln\left(\frac{1}{k}\right)$:

$$Higuici\ FD = \frac{\ln(L(k))}{\ln\left(\frac{1}{k}\right)} \quad \textbf{(5)}$$

Singular Value Decomposition Entropy

Entropy features can also be used to measure the complexity of data. Entropy is a measurement of the amount of disorder, of randomness present in the signal. The SVD

Entropy of the signal provides the number of eigenvectors required in order to reproduce the signal. To measure this, i delay vectors are created [65, 69]:

$$y(i) = [x_i, x_{i+\tau}, \dots, x_{i+(d_E-1)\tau}] \quad \textbf{(6)}$$

Where:

- τ is the delay factor
- d_E is the embedding dimension

A matrix is Y then created:

$$Y = [y(1), y(2), \dots, y(N-(d_E-1)\tau]^T \quad \textbf{(7)}$$

SVD is then performed on matrix Y to produce M singular values $\bar{\sigma}_1, \dots, \bar{\sigma}_M$. The SVD of an mxn matrix Y is defined as factorizing the matrix Y into 3 matrices, such that:

$$Y = U\Sigma V^T \quad \textbf{(8)}$$

Where:

- U is an mxm matrix of the orthonormal eigenvectors of YY^T
- V^T is the transpose of the nxn matrix of the orthonormal eigenvectors of Y^TY
- Σ is an mxn diagonal matrix with M values

The matrix Σ produces the M singular values $\bar{\sigma}_1$, $\bar{\sigma}_2 \cdots$, $\bar{\sigma}_M$ on its diagonal. Information about the matrix, such as its rank, can be derived from these singular values.

The SVD entropy is then defined as:

$$SVD\ Entopy = -\sum_{i=1}^{M} \bar{\sigma}_i \log_2 \bar{\sigma}_i \quad \textbf{(9)}$$

Zero-Crossing Rate

The Zero-Crossing Rate of a signal is the number of times the signal crosses the x-axis, which can be measured by counting the number sign changes between sequential points in the signal.

4.2.2 Measurements of Signal Similarity

G.Boccignone et al. also used features in order to measure the similarity of each unique pair of BVP signals. These features were the Mutual Information, and the Magnitude Squared Coherence Estimate (or "Spectral Similarity", as the researchers of the original method referred to it).

Magnitude Squared Coherence Estimate

The Magnitude Squared Coherence Estimate (MSCE) measures the similarity of a pair of signals, X and Y, in the frequency domain. It does this by using the following equation [70]:

$$MSCE\ (X,Y) = \frac{|P(X,Y)|^2}{P(X)P(Y)} \quad \textbf{(10)}$$

Where:

- $P(X,Y)$ is the cross-spectral density of the signals X and Y
- $P(X)$ and $P(Y)$ is the power spectral densities of the signal X and Y respectively

This calculation is computed at multiple frequencies, so the similarity of each signal pair is compared at each frequency, and a scalar value is returned for each measurement.

Mutual Information

Mutual Information (MI) measures the similarity of a pair of signals, X and Y, in the time domain. MI uses entropy to calculate information about how much one variable influences the other in the time domain. Given that $X = \{x_i\}$ and $Y = \{y_i\}$ represent a set of random observations, a priori measurement of information regarding the amount of uncertainty in X can be calculated using the Shannon entropy as such [71]:

$$H(X) = -\sum_{x_i} P(x_i)\, log_2\, P(x_i) \quad \textbf{(11)}$$

Where:

- $P(x_i)$ is the probability distribution amplitude of x_i

The same equation can be applied to Y. The equation for MI is then given as:

$$MI(X,Y) = H(X) - H(X|Y) \quad \textbf{(12)}$$

Where:

- $H(X|Y)$ is the amount of conditional information that can be measured from X, given Y

This equation can be rewritten as:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \quad \textbf{(13)}$$

Where:

$$H(X,Y) = -\sum_{x_i,y_i} P(x_i,y_i)\, log_2 P(x_i,y_i) \quad \textbf{(14)}$$

4.3 Features Used in This Project's Method

No features that measure signal similarity were used in this project. Instead, additional features that measured signal complexity were used in this project. The complexity related features that were used in this project, in addition to the previously mentioned features were Katz FD, Detrend Fluctuation Analysis (DFA), the Spectral Entropy, the Permutation Entropy, the Approximate Entropy, the Sample Entropy, the Hjorth Activity, the Hjorth Mobility, and the Hjorth Complexity of the Signal.

Katz Fractal Dimension

The Katz Fractal Dimension can be computed directly from the signal by using the following equation [67]:

$$\frac{\log_{10} L}{\log_{10} d} \quad \textbf{(15)}$$

Where:

- L is the total length of the time series
- d is the Euclidean distance that separates the first point from the point furthest away from it

The normalized version of this expression was used in this project. This normalized version of the Katz FD equation is:

$$\frac{\log_{10} L/a}{\log_{10} d/a} \quad \textbf{(16)}$$

Where:

- a is the average number of steps in the time series

Detrend Fluctuation Analysis

DFA, like the FD methods previously discussed, has been used to measure the health an organism by applying it to biological signals. It is used to find long term statistical dependencies of a signal. It does this by analysing a property that fractals possess, self-affinity. This refers to a smaller part of a fractal structure being similar or identical to the whole structure and is a special case of self-similarity. A process is said to be self-affine if a windowed section of the structure has the same statistical distribution as the whole structure. The standard deviation is used as a measure for this. Thus, the self-affinity of a signal, Y, is estimated by [72]:

$$Y(Lt) \equiv L^H Y(t) \quad \textbf{(17)}$$

Where:

- $Y(Lt)$ and $Y(t)$ are the values of Y at time windows of length Lt and t respectively.
- L is the length of the window
- H is the Hurst parameter, a measure of self-affinity
- $\equiv$ is the operator which states that the standard deviation on both sides of the equation are equal

DFA provides an estimate for H. It does this by performing a number of steps. First, a cumulative sum of the signal is calculated in order to produce a signal profile. This signal

profile is then split into T equally spaced time windows. These equally spaced time windows overlap by 50%. For each portion of the signal that is present in each of these overlapping time windows, the linear trend (pattern) is removed using a least-squares line fit. The standard deviation of each resulting windowed signal is then calculated. The fluctuation of the original signal is computed as the mean standard deviation of all windowed signals. The log scale of the fluctuation value is plotted against the log scale of different values of time window length. The slope of this plotted line, α, is the estimated value for H.

Spectral Entropy

The Entropy features are used to measure the complexity of data. The Spectral Entropy is defined as the Shannon Entropy of the Power Spectral Density (PSD) of the signal. [69, 73].

$$Spectral\ Entropy = -\sum_{f=0}^{fs/2} P(f) \log_2 P(f) \quad \textbf{(18)}$$

Where:

- P is the power spectral density
- fs is the sampling frequency

Permutation Entropy

Permutation Entropy is a complexity measure of a time-series. It is computed in a number of steps. Firstly, a time series is defined as follows [69, 74]:

$$x(N) = (x_1, x_2, \dots, x_N) \quad \textbf{(19)}$$

This time series is then converted into a matrix using 2 parameters. The time delay, τ, and the embedding dimension, d_E. τ denotes the periods of overlap between each column vector in the matrix, and d_E describes the length of each of the column vectors. Therefore, if the time series $x(N)$ contained 5 variables, with τ and d_E were given the values of 1 and 3 respectively, the resulting matrix would be:

$$X = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_3 & x_4 \\ x_3 & x_4 & x_5 \end{bmatrix} \quad \textbf{(20)}$$

From this matrix, the column vectors are mapped to unique permutations that describe the ordinal rankings of the data within each column vector. There are $d_E!$ possible permutations (ordinal patterns). An example of one of these ordinal patterns is in the following column vector:

$$\pi = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad \textbf{(21)}$$

This ordinal pattern describes the ranking of the magnitude within a column vector. Using the first column vector in the matrix X for example, this ordinal pattern would occur if $x_1 < x_2 < x_3$.

The relative frequency (or probability of the occurrence), p_i, of each ordinal pattern, π is found by calculating the number of times the ordinal pattern occurs in the matrix, divided by the total number of column vectors, d_E. These probabilities are used to calculate the Permutation Entropy for the time series via the following equation:

$$Permutation\ Entropy = -\sum_{i=1}^{n!} p_i \log_2 p_i \quad \textbf{(22)}$$

Approximate Entropy

The Approximate Entropy of the signal is used to measure the uncertainty in the fluctuations of the signal. It is measured by first taking a time series such as [75, 76]:

$$u(N) = (u_1, u_2, \dots, u_N) \quad \textbf{(23)}$$

From this time series, a sequence of vectors can be created, $x_1, x_2, \dots, x_{N-d_E+1}$ in $\boldsymbol{R}^{d_E}$ where d_E is the embedding dimension, and any x_i is defined as:

$$x_i = \left(u_i, u_{i+!}, \dots, u_{i+d_E-1}\right) for\ i = 1, \dots, N - d_E + 1 \quad \textbf{(24)}$$

The distance between two vectors x_i and x_j is defined as the maximum absolute variation between their scalar components:

$$d[x_i, x_j] = \max_{k=1,2,\dots,d_E} (\left|u_{i+k-1} - u_{j+k-1}\right|) \quad \textbf{(25)}$$

Parameters $C_i^{d_E}(r)$ are defined as the number of j (where $j \leq N - d_E + 1$) where $d[x_i, x_j] \leq r$, where r is a tolerance parameter. The mean values of $C_i^{d_E}(r)$, $\phi^{d_E}(r)$, are defined as:

$$\phi^{d_E}(r) = (N - d_E + 1)^{-1} \sum_{i=0}^{N-d_E+1} \log C_i^{d_E}(r) \quad \textbf{(26)}$$

The Approximate Entropy of a finite signal is then defined as:

$$Approximate\ Entropy = \phi^{d_E}(r) - \phi^{d_E+1}(r) \quad \textbf{(27)}$$

Sample Entropy

Sample Entropy is a modification of Approximate Entropy [75, 76]. The distance between 2 vectors x_i and x_j is computed the same was as Approximate Entropy. From this point, 2 variables are defined.

The first is $b^{d_E}(r)$. For a given x_i, $b_i^{d_E}(r)$ is calculated by multiplying $(N - d_E - 1)^{-1}$ by the number of vectors x_j that are a distance of r away from vectors x_i. Here, j ranges from 1 to $(N - d_E)$ and j does not equal i. With these conditions, $b^{d_E}(r)$ is defined as:

$$b^{d_E}(r) = (N - d_E)^{-1} \sum_{i=1}^{N-d_E} b_i^{d_E}(r) \quad \textbf{(28)}$$

The second is $a^{d_E}(r)$. By increasing the embedding dimension from d_E to $d_E + 1$, for a given x_i, $a_i^{d_E}(r)$ is calculated by multiplying $(N - d_E - 1)^{-1}$ by the number of vectors x_j that are a distance of r away from vectors x_i. Here, j ranges from 1 to $(N - d_E)$ and j does not equal i. With these conditions, $a^{d_E}(r)$ is defined as:

$$a^{d_E}(r) = (N - d_E)^{-1} \sum_{i=1}^{N-d_E} a_i^{d_E}(r) \quad \textbf{(29)}$$

$b^{d_E}(r)$ is the probability that 2 vectors are similar for d_E samples, while $a^{d_E}(r)$ provides the probability that 2 vectors are similar for $d_E + 1$ samples. The Sample Entropy is then defined for a finite signal as:

$$Sample\ Entropy = \ln\left[\frac{a^{d_E}(r)}{b^{d_E}(r)}\right] \quad \textbf{(30)}$$

Hjorth Parameters

The Hjorth parameters provide statistical properties of the signal in the time domain. These parameters are the Activity, Mobility, and Complexity of the signal [77]. The Hjorth Activity is simply the variance of the signal:

$$Hjorth\ Activity = var(x(t)) \quad \textbf{(31)}$$

The Hjorth Mobility represents the mean frequency of the signal's power spectrum. This is measured by calculating the square root of the Hjorth Activity of the first derivative of the signal divided by the Hjorth Activity of the signal.

$$Hjorth\ Mobility = \sqrt{\frac{Hjorth\ Activity(x\prime(t))}{Hjorth\ Activity\ (x(t))}} \quad \textbf{(32)}$$

The Hjorth Complexity estimates the bandwidth of the signal. This is measured by calculating the Hjorth Mobility of the first derivative of the signal divided by the Hjorth Mobility of the signal.

$$Hjorth\ Complexity = \frac{Hjorth\ Mobility(x\prime(t))}{Hjorth\ Mobility\ (x(t))} \quad \textbf{(33)}$$

5 Development of the Deepfake Detection Application

Before an attempt could be made to recreate the method purposed by G. Boccignone et al., it was required to download the *pyVHR* Python framework [62]. This framework is publicly available on GitHub and allows users to extract BVP signals from video footage using rPPG methods with Python. The original method's researchers used this framework to create their deepfake detection method, so the same was done for this project's method in order to allow for a more direct comparison between both methods.

Once the *pyVHR* framework was downloaded, the FaceForensics++ (FF++) dataset was retrieved. This dataset is available in 3 different video qualities:

1. The highest quality (raw), which possesses a lossless compression factor.
2. high-quality (HQ), which possesses a compression factor of 23.
3. low-quality (LQ), which possess a compression factor of 40.

The HQ version of this dataset was initially chosen to test the functionality of the detection model because the raw dataset required approximately 600GB of free space, whereas the HQ dataset only required approximately 12GB. Multiple videos were selected from the FF++ dataset for testing. These tests consisted of using a built-in function from the *pyVHR* framework which calculated the Heart Rate (HR) of a subject directly from video footage. Once the HR of a subject in singular video was successfully calculated, pairs of videos were then analysed. These video pairs consisted of a video of a real person, and a deepfake version of that same video. The HRs of the subjects in both videos were plotted on graphs in order to investigate the differences between the HR of the real and deepfake subjects.

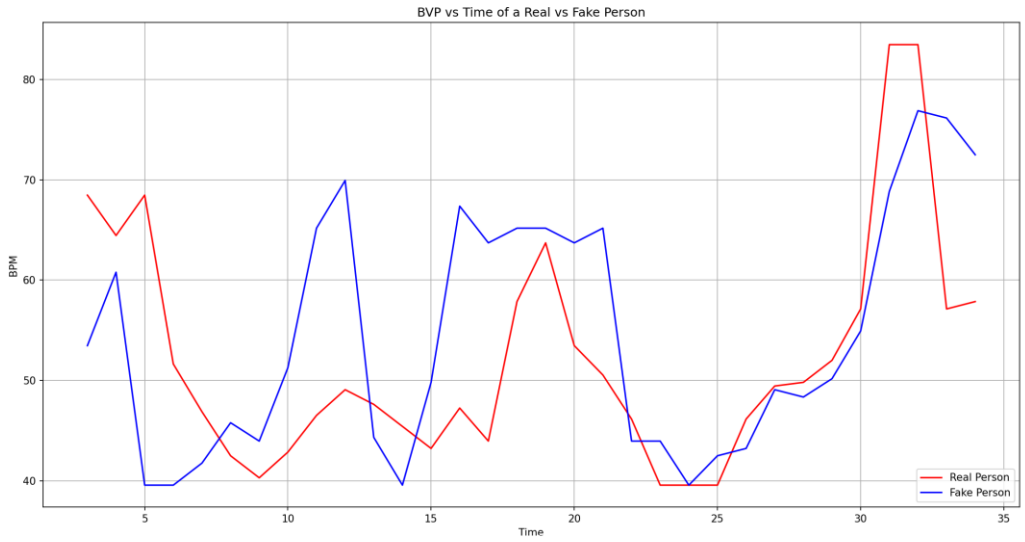

Figure 5.1: BMP over time of a real and deepfake person from video footage. These signals were extracted using built in functions from the pyVHR framework

Once a clear difference between the HR of the subjects from each video pair was observed, the method created by G. Boccignone et al. was then attempted to be recreated. As stated previously, the founders of this method did not publicly provide their codebase; as a result, this method had to be completely independently created based on the information provided in the original research paper.

The implementation of the original method began by replacing the built-in function that automatically calculated HR signals directly from video footage, with a self-made pipeline that calculated the BVP signals used to derive the HR of the subjects in the videos. This was achieved by using methods included in the *pyVHR* framework in order to have more control over how these biological signals were extracted from the video footage. The skin pixels of the subject were extracted from the video using the *Convex-Hull* extraction method. Once the BVP signals were extracted from these skin pixels using the POS rPPG method, each video, and its corresponding BVP signal, were split up into multiple time frames. A window size of 6 was used as this was the window size that was used in the original method.

The features related to measuring the complexity of the shape of each signal were extracted directly using the *antroPY* Python package [78]. The Magnitude Squared Coherence estimate was calculated using the *scipy* Python package [79]. Mutual Information was calculated using Python functions which were sourced from GitHub [80].

Once the BVP signal and feature extraction pipeline was working as intended, this method was then tested on 1% (60) of the videos in the dataset in order test that an SVM had been correctly implemented to work with the time frame level data. Once the implementation of the SVM was verified, the model was then tested on 5% (300) of the videos in the dataset. It was here that the lengthy execution time first became apparent. It was taking approximately 10 hours for the program to complete its execution at this stage in the testing. It was then decided that the model would be trained on 40% (2400) of the videos in the dataset to examine how long the execution time would last, and approximately 4 days (96 hours) later, the program completed its execution. From these tests, it is estimated that this method would have taken 10 days (240 hours) to process all 6000 videos in the FF++ dataset. As a result of this long runtime time, it was decided that this method would be altered in order to decrease its execution time and make it more easily usable.

The *pyVHR* framework includes pre-trained deep learning models that were created to extract BVP signals directly from video footage. One of these models was used to predict the BVP signals from the 6000 videos present in the FF++ dataset. This model consisted of an MTTS-CAN network [81], which predicted the BVP signals of a subject directly from the video footage. As this model was pre-trained, the drawback of the long training times that accompany deep learning methods was not an issue. The inclusion of this deep learning model resulted in some loss of information as only 1 ROI was analysed (the full frame) instead of multiple ROIs; however, with this change, the model completed its execution on all 6000 videos present in the FF++ dataset in approximately 7 hours; a substantial improvement when compared to the estimated 240 hours of the original method. Due to the faster execution time, this deep learning model stayed

present in the deepfake detection pipeline as it sped up the time required to validate the design of the application during testing. As the ROI information was no longer available and only a singular BVP signal was present at each time frame, the 13 signal complexity related features discussed in previous sections were extracted from these signals directly. These features were then used to train an SVM on time frame level data.

The ground truth and SVM predictions were used to derive 3 statistics which provided a way to gauge how well the SVM was classifying the data. These metrics were the accuracy, AUC, and F1 score. The original method used these metrics, so they were used for this project's method in order to allow for a direct comparison between both methods. These statistics were printed out onto the command line after every test completed its execution. The purpose of each of these metrics are explained in section 6.1. Approximately 83.33% of the data in the FF++ dataset consisted of deepfake videos. Thus, the SVM would produce an accuracy score of 83.33% if it simply predicted that every video in the dataset was a deepfake. Therefore, during testing, the AUC score of the model was used in order to gauge how well the model was classifying the data. An AUC score of 0% indicated that the model always made incorrect predictions, an AUC score of 50% indicated that the model's classification accuracy was equivalent to random guessing, and an AUC score of 100% indicated that the model always made correct predictions.

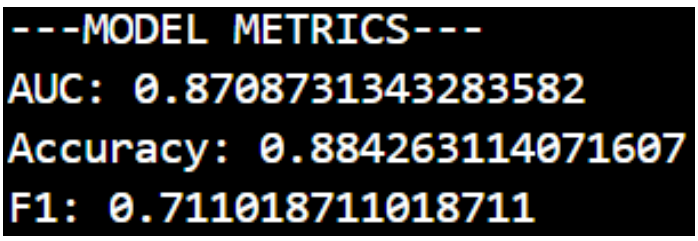

Figure 5.2: An example command line output of the statistics reported from this project's deepfake detection model

When the SVM was tested on 5% (300) of the videos found in the FF++ dataset, the

model showed a positive separation between real and fake videos by reporting an AUC score of 63%. Due to this positive result, the model was executed on all of the 6000 videos present in the FF++ dataset.

The design of the files changed at this point in the testing. Instead of extracting the BVP signals and training the SVM in the same Python file, these operations were split up into 2 separate files. The features extracted from the BVP signals in the first file were saved to .csv files that were loaded in to the second file in order to train the SVM. This change was made in order to test the performance of multiple SVM configurations, without the need to spend 7 hours extracting the BVP signals from all 6000 videos before every test. BVP signals were extracted from each of the 6 categories in the FF++ dataset and saved to their own respective .csv file when all videos from a specific category were analysed. This had the added benefit in the case that the program was suddenly interrupted, as the program would then only be required to start from whatever category it was analysing at the time of the interruption, instead of restarting from the first video in the dataset.

The results from the initial execution after this change were poor and the AUC score reverted to 50%.

Upon researching possible issues with this implementation, it was discovered that there are many factors that affect the quality of the results when using rPPG methods; one of these factors being video compression [82]. Therefore, the raw version of the FF++ dataset was retrieved and replaced the HQ dataset during testing. After the dataset quality changed, the AUC of the model remained at 50% after analysing all 6000 videos in the FF++ dataset. At this point the MTTS-CAN model was replaced by the POS rPPG extraction method that was used in the original method. After this change was made, the execution time of the model increased from 7 hours to approximately 20 hours when analysing all 6000 videos in the FF++ dataset, and the AUC score of the model increased from 50% to 68%. This increase in results could be due to the fact that the MTTS-CAN model was trained to extract BVP signals from videos of exclusively real

people and may have been failing to accurately portray the difference in BVP signals found in real and deepfake videos.

Once this separation of real and fake videos was found, Randomised Search Cross Validation was used in order to select the optimal hyperparameters for the SVM. The predictions were also converted from time frame level predictions to video level predictions by selecting the most frequent time frame level prediction for each video as the video level prediction.

Approximately 2 weeks before the submission of this paper, an error was found with the implementation of the deepfake detection method purposed by G. Boccignone et al. In the paper where this detection method was introduced, the authors explain that the features which measure signal similarity are calculated for every possible BVP signal pair. It was assumed that this meant that if the face of the subject in the video was split up into 100 ROIs, then each BVP signal in each ROI would be compared with every other BVP signal in every other ROI. This would result in 9,900 calculations being computed per feature for each time frame. Upon researching the mathematics behind these features which provide a measurement of signal pair similarity, it was discovered that the order of signals does not matter when calculating these features. Meaning that the similarity measurement of a signal *A* and a signal *B*, would be the same as the measurement between signal *B* and signal *A*. Given this, it would only be necessary for the existing detection method to compare every possible unique BVP signal pair, and not all pairs of BVP signals. This reduces the number of calculations per feature for each time frame from 9,900 to 4,950. This change also reduces the time it takes for the existing method to process all 6,000 videos in the FF++ dataset from 240 hours to 100 hours. This issue highlights the importance of publicly releasing the codebase of deepfake detection methods, as this would clarify existing methods for future researchers attempting to add to existing work. Once this error was rectified, the existing method and this project's method could be more accurately compared in terms of execution time.

6 Results

Each of the 5 deepfake categories present in the FaceForensics (FF++) dataset were tested individually. For each category that was tested, a Support Vector Machine (SVM) was trained on all data from the other 4 categories and 80% of the data from the real videos. The SVM was then tested on all the data from the chosen category, and the remaining 20% of the real data. This method of testing was chosen in order to make a direct comparison with the existing method which this project's method is derived from. This was originally carried out in order to analyse how the detection method would perform when new types of deepfakes were introduced, in order to gauge if this method could be used to detect fake videos when new types of deepfakes are eventually created in the real world.

Section 6.1 explains the purpose of the performance metrics used to gauge how well the SVM classified the data. Section 6.2 discusses the performance of this project's method when tested on the highest quality (raw) version of the FF++ dataset, and how this method compares to the existing method purposed by G. Boccignone et al. Section 6.3 then presents how well this project's method performed on lower quality versions of the FF++ dataset.

6.1 Performance Metrics

The chosen performance metrics for the SVM were the accuracy score, AUC score, and F1 score. These metrics were used in the existing method by G. Boccignone et al. and were thus also used in this project in order to provide a direct comparison between the performance of both methods. This section will discuss the purpose of each of these metrics.

The accuracy score provides a measure of how many of the SVM's predictions are correct; however, only 16.67% of the videos in the dataset are real videos. Meaning that an accuracy score of around 83.33% could be achieved if the SVM simply predicted that

every video in the dataset was a deepfake. Because of this, the AUC score and F1 score were introduced to provide a measure of how well the SVM classified the real videos within the dataset.

The AUC score measures the area under the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate (the number of real videos that are correctly labelled as real) against the false positive rate (the number of fake videos that are incorrectly labelled as real) of the SVM's predictions. The F1 score is a measurement of the harmonic mean of the true positive rate (also called recall) and precision (the probability that a positive prediction is actually positive.) Both scores analyse how the SVM predicts positive samples but do so in different ways. Research has been presented which showed that the ROC curve (and thus the AUC score that's derived from it) is unaffected by skewed data, while the Precision-Recall (PR) curve (and thus the F1 score that's derived from it) is affected by skewed data [83]. This research also showed that ROC curves may hide poor performance as different PR curves can be associated with the same ROC curve. The data of the FF++ dataset is heavily skewed as there are 5 times more fake videos than real videos. The AUC score considers all the possible classification thresholds whereas the F1 score does not; however, the F1 score provides an indication of how the results are affected by the skewed data present in the FF++ dataset. Thus, both scores are used as they provide different and useful measurements regarding the positive class (real videos) for this classification task.

These 3 metrics were given equal importance in the classification task due to the class imbalance present in the dataset. The AUC score and F1 score are measurements of how well the SVM predicts the minority class (real videos), while the accuracy metric indicates how well the model can predict the majority class (fake videos).

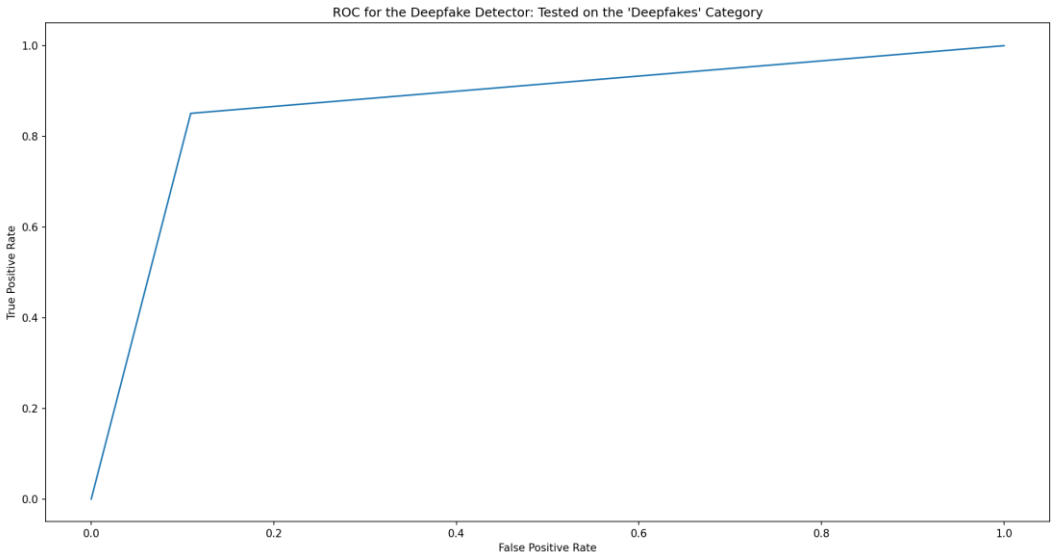

Figure 6.1: The ROC curve generated from testing this project's method on the Deepfakes category. The AUC score is generated by calculating the area under this curve

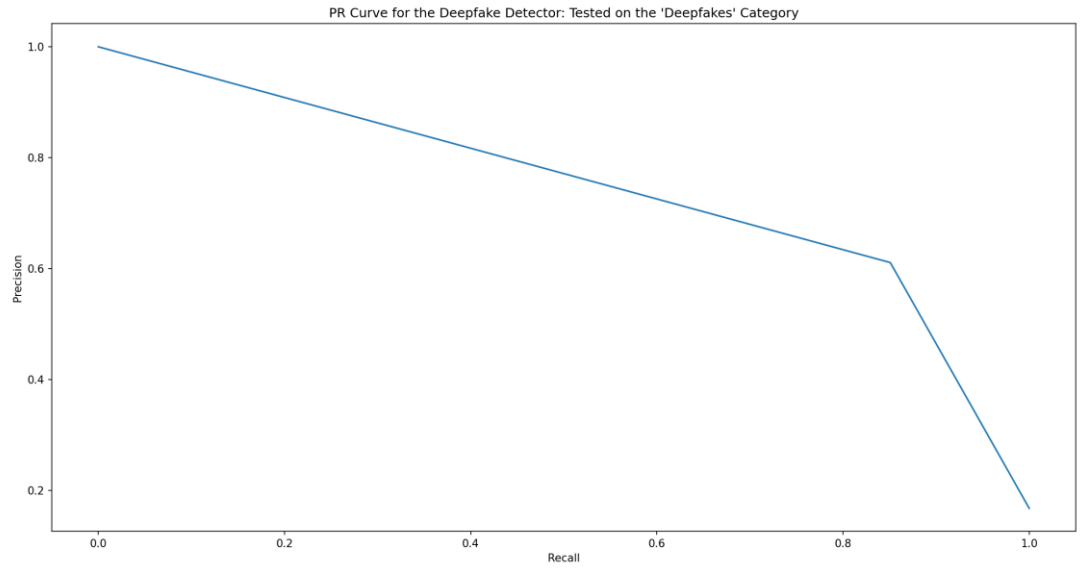

Figure 6.2: The PR curve generated from the testing this project's method on the Deepfakes category

6.2 Raw Dataset

This section discusses the results retrieved from the highest quality (raw) version of the FF++ dataset, which is compared with the results that were achieved by the method purposed by G. Boccignone et al.

6.2.1 Results

The tests results of this project's method when executed with all of the 13 complexity related features on the *Deepfakes* category is shown in Table 6.1.

Table 6.1: All 13 features tested on the *Deepfakes* category

Accuracy	AUC	F1
88.43%	87.09%	71.10%

In order to achieve an understanding of which features contributed the most to the performance of the SVM, each of the 13 features were excluded one at a time and the SVM was tested with the remaining 12 features. The results of each of these tests are shown in Table 6.2.

Table 6.2: Features excluded one at a time and tested on the *Deepfakes* category

Excluded Feature	Accuracy	AUC	F1
Higuchi FD	88.59%	85.80%	70.54%
Petrosian FD	83.76%	83.29%	63.00%
Katz FD	87.68%	86.44%	69.67%
DFA	87.68%	86.24%	69.55%
Zero Crossing Rate	74.94%	81.77%	55.14%
SVD Entropy	86.51%	85.74%	67.73%
Permutation Entropy	85.68%	86.03%	66.92%
Approximate Entropy	87.43%	85.89%	68.99%
Sample Entropy	87.76%	86.29%	69.69%
Spectral Entropy	88.59%	86.98%	71.28%
Hjorth Activity	78.68%	76.67%	53.62%
Hjorth Mobility	84.93%	82.60%	63.73%
Hjorth Complexity	87.68%	87.63%	70.40%

As can be seen from these tables, all features had some kind of positive influence on the SVM's performance. Some features caused an increase in 1 performance metric, but a decrease in the remaining 2. As a result of all 3 performance metrics being given equal importance, and the fact that no feature's removal caused an increase in all 3 of these performance metrics, all 13 features were kept for this classification task. The most important feature appeared to be the Hjorth Activity as this was the only feature that caused the AUC score to fail to reach a score of at least 80% upon its exclusion. This leads to the question of how well the method purposed by G. Boccignone et al. would have performed if this feature was included in their detection method. Each test took approximately 3 hours to finish its execution, except when the Zero Crossing Rate feature was removed. In this case, the test finished its execution in a matter of seconds.

Table 6.3 presents the test results when all categories of the FF++ dataset were tested with all 13 features.

Table 6.3: All 13 features tested on all FF++ categories

	Facial Replacement deepfakes			Facial Reenactment deepfakes	
Metric	**Deepfakes**	**FaceSwap**	**FaceShifter**	**Face2Face**	**NeuralTextures**
Accuracy	88.43%	94.00%	95.34%	28.73%	20.73%
AUC	87.09%	87.46%	89.05%	51.63%	47.03%
F1	71.10%	81.25%	85.11%	28.79%	26.77%

This method produces good results for 3 out of the 5 categories. The hyperparameters for the SVM were chosen via Randomized Search Cross Validation. It was possible that when the data from the *Face2Face* and *NeuralTextures* categories were removed, the distribution of the data changed, and thus different hyperparameters may have been required. The hyperparameters of the SVM were then manually changed and tested on these 2 categories. The best results from these tests are shown in Table 6.4.

Table 6.4: Results of testing different hyperparameters on the underperforming categories

Category	Accuracy	AUC	F1
Face2Face	43.88%	53.98%	29.20%
NeuralTextures	61.53%	50.27%	22.48%

The hyperparameters of the SVM could not be configured to improve the results of these 2 categories enough to become comparable to the other 3 categories. The final results of this method are displayed and compared against the results from the method purposed by G. Boccignone et al. in Table 6.5.

Table 6.5: Final results for both methods tested on the raw FF++ dataset

		Facial Replacement deepfakes			Facial Reenactment deepfakes		
Method	**Metric**	**Deepfakes**	**FaceSwap**	**FaceShifter**	**Face2Face**	**NeuralTextures**	**Avg.**
This Project	Acc.	88.43%	94.00%	95.34%	43.88%	61.53%	76.64%
	AUC	87.09%	87.46%	89.05%	53.98%	50.27%	73.57%
	F1	71.10%	81.25%	85.11%	29.20%	22.48%	57.83%
Boccignone et al. [49]	Acc.	94.53%	95.66%	96.98%	94.48%	90.61%	94.45%
	AUC	89.82%	88.59%	91.52%	88.97%	89.76%	89.75%
	F1	96.72%	97.44%	98.21%	96.70%	94.14%	96.64%

6.2.2 Discussion

The *Face2Face* and *NeuralTextures* categories contain different types of deepfakes than the *Deepfakes*, *FaceShifter* and *FaceSwap* categories. The later 3 deepfake techniques are facial replacement deepfakes, while the former 2 deepfake techniques are facial reenactment deepfakes.

These results indicate that this project's method works well at detecting when a subject's entire face has been replaced with the face of another, however it is not fit to detect when a subject's face is making facial expressions that were artificially added to the video. The method purposed by G. Boccignone et al. may perform much better on these 2 categories due to its use of the patch-similarity related features that were not used in this project's method. These features measured the similarity of the BVP signal at each ROI on the video subject's face. These measurements may have allowed the deepfake detector to understand when only one region of the face was manipulated in the video footage, instead of requiring the majority (or possibly all) of the video subject's face to be manipulated.

The removal of these patch-similarity related features significantly reduced the runtime of this method; however, this has been achieved seemingly at the expense of the ability to detect facial expression manipulations. The method proposed by G. Boccignone et al. performs better in every deepfake category; however, this project's method completes its execution time in 20 hours, approximately 5x quicker than the method purposed by G. Boccignone et al. Overall, this project's method may be a suitable replacement to the method purposed by G. Boccignone et al. when attempting to detect facial replacement deepfakes due to the decreased runtime and comparable results of this project's method, however it is not suitable for the task of identifying facial reenactment deepfakes.

6.3 Lower Quality Datasets

The method proposed by G. Boccignone et al. only tested the highest quality (raw) version of the FF++. This section tests this project's method on both the lower quality versions of the FF++ dataset; these being the HQ and LQ versions of the dataset.

6.3.1 Results

The performance of rPPG methods is heavily affected on a number of factors. One of these factors is video compression [82]. Thus, the HQ and LQ datasets were also tested with this project's method, and the results of these tests are shown in Table 6.6 and Table 6.7 respectively.

Table 6.6: This project's method tested on all categories in the HQ FF++ dataset

	Facial Replacement deepfakes			Facial Reenactment deepfakes	
Metric	**Deepfakes**	**FaceSwap**	**FaceShifter**	**Face2Face**	**NeuralTextures**
Accuracy	40.55%	53.87%	62.61%	39.63%	44.80%
AUC	56.35%	65.94%	69.00%	56.99%	59.10%
F1	31.08%	37.89%	41.31%	31.54%	32.83%

Table 6.7: This project's method tested on all categories in the LQ FF++ dataset

	Facial Replacement deepfakes			Facial Reenactment deepfakes	
Metric	**Deepfakes**	**FaceSwap**	**FaceShifter**	**Face2Face**	**NeuralTextures**
Accuracy	42.21%	46.21%	43.55%	38.05%	45.21%
AUC	52.58%	54.58%	51.19%	49.09%	53.98%
F1	28.31%	29.48%	27.10%	26.19%	29.09%

6.3.2 Discussion

The standard of classification results diminished as the quality of the dataset lowered. This suggests that if one wishes to create deepfakes that would avoid detection via rPPG methods, one should create deepfakes of poor video quality. The quality of the BVP signals produced by the colour pixels from the video footage of a subject's skin appears to become less accurate with the lowering of the video quality. The human eye may also struggle to differentiate real from fake videos if the video quality was sufficiently poor, due to being unable to make out a clear image of the subject in a video.

7 Conclusion

7.1 Review and Critique

As was mentioned in the beginning of this paper, the objective of this project was to create a deepfake detection model that was accessible to use. This project set out to achieve this objective by reducing the time it took for the detection method presented by G. Boccignone et al. to execute, while preserving the simplicity of this method. This project's method successfully achieved this goal by processing all videos in the FaceForensics++ (FF++) dataset in under 24 hours, while also computing the features used for classification in a much simpler way. This led to a singular video being processed in approximately 12 seconds, when it took the existing method approximately 60 seconds to perform the same operation, resulting in a 5x decrease in computation time. If this method was used by a social media platform, where users would be given an option to verify the legitimacy of videos they came across on the platform, the users would be more likely to use this application if the verification process took 12 seconds, rather than taking a full minute.

This decrease in execution time did come with its drawbacks. This project's method performed worse than the method purposed by G. Boccignone et al. on average for the FF++ dataset. However, this project's method is a suitable alternative to the existing method when analysing facial replacement deepfakes due to their comparable performance on these types of deepfakes, and this project's method's decreased execution time. While this project's method still lowered the computation time of processing each video, the classification performance when tested on facial reenactment deepfakes was only marginally better than that of randomly guessing. Thus, this project's method is not a suitable replacement for the existing method when analysing facial reenactment deepfakes, nor is it a suitable method for detecting these types of deepfakes in general.

As is evident from the results of this paper, there is a trade-off between computation

time and the performance of a deepfake detection model. As previously mentioned, this project achieved its objective of creating a deepfake detection application which is simple to recreate and quicker to execute than many other deepfake detection methods. The other objective of this project was to preserve the results of the method purposed by G. Boccignone et al. as much as possible. It was endeavoured to achieve average accuracy, AUC, and F1 scores of 80% each on the FF++ dataset. This project achieved average scores of 77%, 74%, and 58% for each of these scores respectively, which concludes that this goal was not met. However, this project's method is a viable replacement for the existing method when identifying facial replacement deepfakes. When analysing if this project's goal of preserving the quality of the results was achieved on the facial replacement categories, the average accuracy, AUC, and F1 scores of this method were 93%, 88%, and 79% respectively, showing that this project met its goal for 2 of these 3 performance metrics when analysing facial replacement deepfakes. Since this project's method fails to accurately identify facial reenactment deepfakes, the average performance metric scores for the entire dataset were brought down by the results stemming these types of deepfakes.

This project's deepfake detection method demonstrates a possible drawback of using rPPG based detection methods. The performance of this method appears to be correlated to the quality of the videos being analysed. This results in low quality deepfakes not being reliably detected.

7.2 Future Work

It appears that there is not a lot of research regarding the creation of simple deepfake detection methods that are easy to replicate. This subject should be of concern to future researchers because it is now easier than it ever has been for an ordinary person with no knowledge of deepfake generation to create a deepfake and post it on social media to spread misinformation or inflict some other sort of societal harm. This research field should focus on creating simple detection methods and make the codebase for these methods publicly available in order to enable members of the public to create their own

deepfake detection solutions to help combat the potential malicious uses of deepfakes.

The dataset used in this paper is not the most complex, or new, publicly available dataset. It was used in this paper for the sake of comparing this project's method with the one it is derived from. Once simpler deepfake detection methods have been created, they should be tested on more modern deepfake detection datasets. Furthermore, the most recent publicly available dataset that is commonly used to train deepfake detection models is the DFDC dataset, which was created in 2020. Since then, new deepfake generation tools have been released, many in the form of AI chatbots such as DALL·E and Sora. A new deepfake detection dataset should be created with more modern deepfakes in order to show how well previous detection methods detect these new types of deepfakes.

Deepfake generation tools are able to produce more convincing images and videos more easily the more time moves forward. Thus, creating easy to use and easy to replicate deepfake detectors may become more difficult to achieve as deepfakes become more convincing. However, an attempt should still be made for the reasons that have already been listed.

With the tools to create AI generated media being widely available to the public, the responsibility falls onto the owners of social media platforms in order to verify that information in the form of images and videos uploaded to the platform is legitimate and not AI generated as this could lead to the quick spread of misinformation across the internet. Deepfake detection tools could aid in this mission. Social media companies such as *Meta* have included fact checking systems to minimize the spread of misinformation on these platforms since 2016 [84]. However, this has not seemed to be sufficient given that many people have still been deceived by the false information that is still present on these platforms, such as advertisements consisting of AI generated videos of celebrities that are not associated with the advertised products. Thus, it is evident that more work is still required before users of these social media platforms can fully trust the information present on these platforms.

Bibliography

[1] M. Masood *et al*. "Deepfakes Generation and Detection: State-of-the-art, open challenges, countermeasures, and way forward," *Applied Intelligence*, vol. 53, Jun. 2022, pp. 1-53, doi: 10.1007/s10489-022-03766-z.

[2] C. Stupp. "Fraudsters Used AI to Mimic CEO's Voice in Unusual Cybercrime Case." The Wall Street Journal. https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402 (accessed Oct. 02, 2023).

[3] J. Wakefield. "Deepfake presidents used in Russia-Ukraine war." BBC News. https://www.bbc.com/news/technology-60780142 (accessed Oct. 02, 2023).

[4] Burrito Glasses. *Dr Phil but everyone is Dr Phil [Deepfake]*. (May. 16, 2020). Accessed: Apr. 09, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=f2YbT5SGmGA

[5] Zero Malaria Britian. *David Beckham speaks nine languages to launch Malaria Must Die Voice Petition*. (Apr. 09, 2019). Accessed: Oct. 02, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=QiiSAvKJIHo

[6] A. Monteil. "This Viral Ad Is Challenging Gendered Assumptions About the Women's World Cup." them. https://www.them.us/story/womens-world-cup-viral-french-ad-deepfake (accessed Mar. 05, 2024).

[7] A. Kaur. "Celebrities warn followers not to be duped by AI deepfakes." The Washington Post. https://www.washingtonpost.com/entertainment/2023/10/03/tom-hanks-ai-ad-deepfake (accessed Mar. 07, 2024)

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *ICLR*, Dec. 2014. [Online]. Available: https://arxiv.org/abs/1312.6114

[9] M. M. Rashid *et al*., "Blockchain Technology for Combating Deepfake and Protect Video/Image Integrity," *Journal of Korea Multimedia Society*, vol. 24, Aug. 2021, pp. 1044–1058, doi: 10.9717/kmms.2021.24.8.1044.

[10] I. Goodfellow *et al*., "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[11] J.Bruner and A.Deshpande. “Generative Adversarial Networks for beginners.” O’Reilly Media. https://www.oreilly.com/content/generative-adversarial-networks-for-beginners/ (accessed Dec. 19, 2023).

[12] J.-Y. Zhu *et al*., "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 2242-2251, doi: 10.1109/ICCV.2017.244.

[13] Y. Choi *et al*., "StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 8789-8797, doi: 10.1109/CVPR.2018.00916.

[14] T. Karras *et al*., "A Style-Based Generator Architecture for Generative Adversarial Networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 4396-4405, doi: 10.1109/CVPR.2019.00453.

[15] T. Karras *et al*., "Analyzing and Improving the Image Quality of StyleGAN," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 8107-8116, doi: 10.1109/CVPR42600.2020.00813.

[16] A. Vaswani *et al*., “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[17] “DALL·E 3.” OpenAI. https://openai.com/dall-e-3 (accessed Apr. 05, 2024)

[18] “Sora.” OpenAI. https://openai.com/sora (accessed Apr. 05, 2024)

[19] M. Kowalski. “FaceSwap.” https://github.com/MarekKowalski/FaceSwap (accessed Jan. 18, 2024).

[20] FaceApp. https://www.faceapp.com (accessed Oct. 02, 2023).

[21] I. Perov *et al*. “DeepFaceLab.” https://github.com/iperov/DeepFaceLab (accessed Oct. 02, 2023)

[22] Deepfakes Web. https://deepfakesweb.com (accessed Oct. 02, 2023).

[23] Zao. https://apps.apple.com/cn/app/zao/id1465199127

[24] Z. Doffman. “Chinese Deepfake App ZAO Goes Viral, Privacy Of Millions 'At Risk'.” Forbes. https://www.forbes.com/sites/zakdoffman/2019/09/02/chinese-

best-ever-deepfake-app-zao-sparks-huge-faceapp-like-privacy-storm/ (accessed Oct. 02, 2023)

[25] E. Napolitano. “X blocks searches for "Taylor Swift" after explicit deepfakes go viral.” CBS News. https://www.cbsnews.com/news/taylor-swift-deepfakes-x-search-block-twitter/ (accessed Apr. 05, 2024)

[26] G. Lee and M. Kim, “Deepfake Detection Using the Rate of Change between Frames Based on Computer Vision,” *Sensors*, vol. 21, no. 21, Nov. 2021, p. 7367, doi: 10.3390/s21217367.

[27] J. Kang *et al.*, "Detection Enhancement for Various Deepfake Types Based on Residual Noise and Manipulation Traces," in *IEEE Access*, vol. 10, 2022, pp. 69031-69040, doi: 10.1109/ACCESS.2022.3185121.

[28] X. Yang *et al.*, "Exposing Deep Fakes Using Inconsistent Head Poses," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 8261-8265, doi: 10.1109/ICASSP.2019.8683164.

[29] S. McCloskey and M. Albright, “Detecting GAN-generated Imagery using Color Cues,” *arXiv*, 2018, [Online]. Available: https://arxiv.org/pdf/1812.08247.pdf

[30] A. Rössler *et al.*, “FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces,” *arXiv*, 2018. [Online]. Available: https://arxiv.org/pdf/1803.09179.pdf

[31] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.

[32] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *arXiv*, 2019. [Online]. Available: https://arxiv.org/pdf/1905.11946.pdf

[33] B. Dolhansky *et al.*, “The DeepFake Detection Challenge (DFDC) Dataset.,” *ArXiv*, 2020. [Online]. Available: https://arxiv.org/pdf/2006.07397.pdf

[34] K. He *et al.*, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[35] D. Afchar *et al.*, "MesoNet: a Compact Facial Video Forgery Detection Network," *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*,

Hong Kong, China, 2018, pp. 1-7, doi: 10.1109/WIFS.2018.8630761.

[36] D. Wodajo and S. Atnafu, “Deepfake Video Detection Using Convolutional Vision Transformer,” *arXiv*, 2021, [Online]. Available: https://arxiv.org/pdf/2102.11126.pdf

[37] H. Zhao *et al*., "Multi-attentional Deepfake Detection," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 2021, pp. 2185-2194, doi: 10.1109/CVPR46437.2021.00222.

[38] Y. Li *et al*., "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking," *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, Hong Kong, China, 2018, pp. 1-7, doi: 10.1109/WIFS.2018.8630787.

[39] T. Jung *et al*., "DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern," in *IEEE Access*, vol. 8, 2020, pp. 83144-83154, doi: 10.1109/ACCESS.2020.2988660.

[40] D. Zhao *et al*., “SFST: A robust framework for heart rate monitoring from photoplethysmography signals during physical activities,” *Biomedical Signal Processing and Control*, vol. 33, 2017, pp. 316–324, doi: 10.1016/j.bspc.2016.12.005.

[41] W. Verkruysse *et al*., “Remote plethysmographic imaging using ambient light,” *Optics express*, vol. 16, Dec. 2008, pp. 21434–21445, doi: 10.1364/OE.16.021434.

[42] W. Wang *et al*., "Algorithmic Principles of Remote PPG," in *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 7, Jul. 2017, pp. 1479-1491, doi: 10.1109/TBME.2016.2609282.

[43] T. Wu *et al*., “Photoplethysmography imaging: a new noninvasive and noncontact method for mapping of the dermal perfusion changes,” in *Optical Techniques and Instrumentation for the Measurement of Blood Composition, Structure, and Dynamics*, vol. 4163, 2000, pp. 62–70, doi: 10.1117/12.407646.

[44] U. A. Ciftci *et al*., "How Do the Hearts of Deep Fakes Beat? Deep Fake Source Detection via Interpreting Residuals with Biological Signals," *2020 IEEE International Joint Conference on Biometrics (IJCB)*, Houston, TX, USA, 2020, pp. 1-10, doi: 10.1109/IJCB48548.2020.9304909.

[45] J. Wu *et al*., "Local attention and long-distance interaction of rPPG for deepfake detection," *The Visual Computer*, vol. 40, Mar. 2023, doi: 10.1007/s00371-023-02833-x.

[46] X. Jin *et al*., "Countering spoof: towards detecting deepfake with multidimensional biological signals," *Security and Communication Networks 2021*, Apr. 2021, doi: 10.1155/2021/6626974.

[47] J. Liang and W. Deng, "Identifying Rhythmic Patterns for Face Forgery Detection and Categorization," *2021 IEEE International Joint Conference on Biometrics (IJCB)*, Shenzhen, China, 2021, pp. 1-8, doi: 10.1109/IJCB52358.2021.9484400.

[48] H. Qi *et al*., "DeepRhythm: Exposing DeepFakes with Attentional Visual Heartbeat Rhythms," in *Proceedings of the 28th ACM International Conference on Multimedia*, Oct. 2020, pp. 4318–4327, doi: 10.1145/3394171.3413707.

[49] G. Boccignone *et al*., "DeepFakes Have No Heart: A Simple RPPG-Based Method to Reveal Fake Videos*," Image Analysis and Processing – ICIAP 2022: 21st International Conference, Lecce, Italy, May 23–27, 2022, Proceedings, Part II*, May. 2022, pp. 186–195, doi: 10.1007/978-3-031-06430-2_16.

[50] U. A. Ciftci *et al*., "FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals*,"* in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, doi: 10.1109/TPAMI.2020.3009287.

[51] "Intel Introduces Real-Time Deepfake Detector." Intel. https://www.intel.com/content/www/us/en/newsroom/news/intel-introduces-real-time-deepfake-detector.html (accessed Apr. 06, 2024).

[52] J. Hernandez-Ortega *et al*., "DeepFakes Detection Based on Heart Rate Estimation: Single- and Multi-frame," *arXiv*, 2020. [Online]. Available: https://arxiv.org/pdf/2010.00400.pdf

[53] W. Chen and D. McDuff, "DeepPhys: Video-Based Physiological Measurement Using Convolutional Attention Networks," in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, Sep. 2018, pp. 356–373. doi: 10.1007/978-3-030-01216-8_22.

[54] Y. Li *et al*., "Celeb-DF: A Large-Scale Challenging Dataset for DeepFake Forensics," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 3204-3213, doi: 10.1109/CVPR42600.2020.00327.

[55] B. Zi *et al.*, "WildDeepfake: A Challenging Real-World Dataset for Deepfake Detection," *Proceedings of the 28th ACM International Conference on Multimedia*, Oct. 2020, pp. 2382-2390, doi: 10.1145/3394171.3413769.

[56] A. Rössler *et al.*, "FaceForensics++: Learning to Detect Manipulated Facial Images," *in International Conference on Computer Vision (ICCV)*, 2019.

[57] "deepfakes_faceswap." https://github.com/deepfakes/faceswap (accessed Jan. 18, 2024).

[58] Justus Thies *et al.*, "Face2Face: Real-Time Face Capture and Reenactment of RGB Videos," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 2387-2395, doi: 10.1109/CVPR.2016.262.

[59] Justus Thies *et al.*, "Deferred neural rendering: Image synthesis using neural textures," *ACM Transactions on Graphics 2019 (TOG)*, vol. 38, no.4, 2019, pp. 1-12, doi: 10.1145/3306346.3323035.

[60] L. Li *et al.*, "FaceShifter: Towards High Fidelity And Occlusion Aware Face Swapping," *arxiv*, 2019. [Online]. Available: https://arxiv.org/abs/1912.13457

[61] W. Fu and T. Menzies, "Easy over hard: a case study on deep learning," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, Aug. 2017, pp. 49–60, doi: 10.1145/3106237.3106256.

[62] G. Boccignone et al., "pyVHR: a Python framework for remote photoplethysmography," *PeerJ Computer Science*, vol. 8, Apr. 15, 2022, doi: 10.7717/peerj-cs.929.

[63] A. Petrosian, "Kolmogorov complexity of finite sequences and recognition of different preictal EEG patterns," *Proceedings Eighth IEEE Symposium on Computer-Based Medical Systems*, Lubbock, TX, USA, 1995, pp. 212-217, doi: 10.1109/CBMS.1995.465426.

[64] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," *Physica D: Nonlinear Phenomena*, vol. 31, no. 2, 1988, pp. 277–283, doi: 10.1016/0167-2789(88)90081-4.

[65] S. J. Roberts *et al.*, "Temporal and spatial complexity measures for electroencephalogram based brain-computer interfacing," *Medical & biological engineering & computing*, vol. 37, 1999, pp. 93–98, doi: 10.1007/BF02513272.

[66] M. J. Katz, "Fractals and the analysis of waveforms," *Computers in Biology and Medicine*, vol. 18, no. 3, 1988, pp. 145–156, doi: 10.1016/0010-4825(88)90041-8.

[67] C. Goh *et al*., "Comparison of Fractal Dimension Algorithms for the Computation of EEG Biomarkers for Dementia," *2nd International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2005)*, Jun. 2005.

[68] S. Kesić and S. Z. Spasić, "Application of Higuchi's fractal dimension from basic to clinical neurophysiology: A review," *Computer Methods and Programs in Biomedicine*, vol. 133, Sep. 2016, pp. 55–70, doi: 10.1016/j.cmpb.2016.05.014.

[69] R. Vallat. *AntroPY v0.1.6* (2023). Accessed: Nov. 12, 2023. [Online]. Available: https://raphaelvallat.com/antropy/build/html/index.html

[70] S. Malekpour *et al*., "Measures of generalized magnitude-squared coherence: Differences and similarities," *Journal of the Franklin Institute*, vol. 355, no. 5, Mar. 2018, pp. 2932–2950, doi: 10.1016/j.jfranklin.2018.01.014.

[71] P. Ramanand *et al*., "Mutual Information Analysis of EEG Signals Indicates Age-Related Changes in Cortical Interdependence during Sleep in Middle-aged vs. Elderly Women," *Journal of clinical neurophysiology*, vol. 27, no. 4, Aug. 2010, pp.274–284, doi: 10.1097/WNP.0b013e3181eaa9f5.

[72] R. Hardstone *et al*., "Detrended Fluctuation Analysis: A Scale-Free View on Neuronal Oscillations," *Frontiers in Physiology*, vol. 3, Nov. 30, 2012, doi: 10.3389/fphys.2012.00450.

[73] T. Inouye *et al*., "Quantification of EEG irregularity by use of the entropy of the power spectrum," *Electroencephalography and Clinical Neurophysiology*, vol. 79, no. 3, Sep. 1991, pp. 204–210, doi: 10.1016/0013-4694(91)90138-T.

[74] C. Bandt and B. Pompe, "Permutation Entropy: A Natural Complexity Measure for Time Series," *Physical Review Letters,* vol. 88, no. 17, Apr. 29, 2002, doi: 10.1103/PhysRevLett.88.174102.

[75] A. Delgado-Bonal and A. Marshak, "Approximate Entropy and Sample Entropy: A Comprehensive Tutorial," *Entropy*, vol. 21, no. 6, May. 28, 2019, doi: 10.3390/e21060541.

[76] A. Howedi *et al*., "Exploring Entropy Measurements to Identify Multi-Occupancy in Activities of Daily Living," *Entropy*, vol. 21, no. 4, Apr. 19, 2019, doi: 10.3390/e21040416.

[77] F. Galvão *et al*., "Predicting Exact Valence and Arousal Values from EEG," *Sensors*, vol. 21, no.10, May. 2021, doi: 10.3390/s21103414.

[78] R. Vallat. "AntroPY." https://github.com/raphaelvallat/antropy (accessed Nov. 12, 2023)

[79] P. Virtanen *et al*., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, 2020, pp. 261–272, doi: 10.1038/s41592-019-0686-2.

[80] M. Saint-Antoine. https://github.com/mikesaint-antoine/Comp_Bio_Tutorials/blob/main/numerical_methods/mutual_info.py (accessed Nov. 13, 2023)

[81] X. Liu *et al*., "Multi-Task Temporal Shift Attention Networks for On-Device Contactless Vitals Measurement," *arXiv*, 2020. [Online]. Available: https://arxiv.org/pdf/2006.03790.pdf

[82] A. D'Amelio *et al*., "On Using rPPG Signals for DeepFake Detection: A Cautionary Note," *Image Analysis and Processing – ICIAP 2023*, 2023, pp. 235–246, doi: 10.1007/978-3-031-43153-1_20.

[83] L. A. Jeni *et al*., "Facing Imbalanced Data--Recommendations for the Use of Performance Metrics," *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, Geneva, Switzerland, 2013, pp. 245-251, doi: 10.1109/ACII.2013.47.

[84] "How Facebook's third-party fact-checking program works." Meta. https://www.facebook.com/formedia/blog/third-party-fact-checking-how-it-works (accessed Apr. 07, 2024).