Deepfake Detective:
From Model Development to Web Application Integration

By

Tyler Supersad

Submitted to

The University of Roehampton

In partial fulfilment of the requirements

for the degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Abstract

The advent of deepfake media, produced through artificial intelligence-generated face swapping technology, has raised concerns about the possibility of its exploitation for activities such as deepfake pornography, financial fraud, and political disruption, which can have negative impacts for individuals and organizations alike. Hence, there is a critical need to develop approaches that can identify and mitigate such harmful practices. In this regard, this project leveraged two Convolutional Neural Networks (CNNs) to differentiate between deepfake and authentic media and conducted a comparative analysis. Specifically, the investigation employed the pre-trained CNN EfficientNet-B0 architecture and a custom baseline CNN. The FaceForensics++ dataset, comprising 1,000 authentic and 1,000 deepfake videos, was utilized to train the models and the study used three metrics - accuracy, loss, confusion matric - to evaluate the results. The EfficientNet-B0 model achieved the highest accuracy of 93.3%, whereas the custom baseline model attained an accuracy of 86.9%. Finally, a minimum viable product was developed to explore the possibility of making the EfficientNet-B0 deepfake detection model available for public use.

Declaration

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Tyler Supersad

May 12, 2023

Acknowledgements

I am sincerely grateful to my supervisor, Dr Charles Clarke, for his insightful guidance and support throughout the development of this project. I would also like to express my heartfelt appreciation to my family and my closest friends, Michelle Maldonado, Justin Hughes, and Austin Matthews, for their unwavering support and encouragement during this endeavor.

Table of Contents

1 Introduction

The proliferation of deepfake technology has presented significant challenges across various societal platforms, ranging from politics to individual privacy. Such easily accessible and utilizable digital manipulations threaten the integrity of information and the credibility of public discourse, emphasizing the urgent need for innovative solutions that can accurately detect them and prevent their normalization within the public. As a response, this project explored the effectiveness of two machine learning techniques for comparative analysis, with the intention of achieving accurate detection of deepfakes in visual media. As part of this effort, the project introduced a user-friendly web application that deployed the best-performing technique for public use. The software allowed individuals to easily upload images or videos for analysis and identified their authenticity through the chosen technique. Through extensive research analysis and the development of a simplistic web application, the aspiration was to make a noteworthy contribution to the development of accurate and robust deepfake detection technologies, thus mitigating the negative impacts of deepfake technological usages.

1.1 Research Question

What machine learning techniques can be employed to improve the accuracy of detecting deepfakes in visual media (images and videos)? Furthermore, how can such techniques be effectively deployed for wider accessibility and practical use?

1.2 Aims

This project intended to address the pressing issue of deepfake technology by achieving three specific outcomes:

- Provide a comprehensive overview of deepfakes and explore potential solutions to combat them, including traditional detection methods and advanced techniques using machine learning algorithms such as CNNs.
- Accurately identify deepfakes within a visual medium through the development of deepfake detection models using advanced machine learning techniques for comparative analysis.

- Deploy the most optimal detection model on a user-friendly web application that enables users to upload videos for analysis and receive immediate feedback on whether the video is a deepfake or not.

1.3 Objectives

Aligned with the established aims (see Section 1.2), a series of objectives need to be executed to successfully achieve each aim. Ergo, the subsequent information presented a detailed list of objectives that were necessary to accomplish each aim.

Exploring and Researching Deepfakes and Detection Methods

- Perform a thorough literature review on deepfakes to acquire relevant information on their generation process, and utilization.
- Evaluate the potential ethical, societal, and legal challenges posed by deepfakes.
- Investigate existing advanced methods that employed machine learning algorithms to accurately detect deepfakes.

Accurately Detecting Deepfakes through Advanced Machine Learning

- Select the most appropriate deepfake detection dataset by reviewing existing ones.
- Choose several architectures suitable for the task of deepfake detection.
- Train and test the selected architectures with the chosen dataset to ensure their accuracy.
- Conduct a comparative analysis of the developed deepfake detection architectures to identify the most accurate and effective one.

Constructing a Web-Based Application for Deepfake Detection

- Design an application for users to upload content and receive deepfake classification results.
- Utilize available web construction technologies to implement the design of the web application.
- Integrate the deepfake detection architecture with the best efficiency into the web application.

1.4 Ethical, Social, and Legal Considerations

Deepfake technology is predicated on the ability to deceive through the production of fictional scenarios that are virtually indistinguishable from reality [1]. Although exhibiting such a dangerous quality, the film and television industry has exercised deepfakes as a professional tool to enhance the quality of their content, without malevolent intent. Lucasfilm, a prominent film and television production company, has utilized them to de-age actors or resurrect deceased actors. For instance, in The Book of Boba Fett, a 2021 series, deepfakes were used to depict a younger version of Luke Skywalker, despite the actor who initially portrayed that version of the character being decades older presently [1]. Furthermore, they have also enabled the late Carrie Fisher to reprise her role as Leia Organa in Star Wars: Episode IX, three years after her death [1].

While Hollywood is one of the industries that benefit from deepfakes, their capacity for deception has also led to the creation of highly objectionable and controversial content, including the insertion of individuals into pornographic films and the manipulation of media to influence politics. Such actions have been widely condemned as ethically reprehensible, raising significant concerns about the misuse of the technology. Therefore, an examination of the ethical implications of deepfakes and their potential impact on legal and societal issues was performed.

Ethical Considerations of Deepfake Technology

Deepfake pornography accounts for more than 90% of harmful deepfakes, and is primarily targeted towards women, who comprise of 90% of all cases [2]. High-profile actresses such as Gal Gadot, Emma Watson, and Taylor Swift have been victimized from this form of image-based sexual abuse, with deepfake videos portraying them engaging in offensive, frightening, or exclusively private acts. Such content represents a significant violation of personal autonomy, and poses risks to safety, dignity, reputation, and mental health, according to many experts [3]. Consequently, several online platforms, including Reddit and Pornhub, have banned them due to their non-consensual and destructive nature.

In addition to the concerning prevalence of deepfake pornography, there is a growing apprehension regarding the utilization of deepfakes to misrepresent world leaders in ways that they do not intend. For example, in March 2022, a fake video depicted Ukrainian President Volodymyr Zelensky surrendering to Russia during the Russia-Ukraine War [4]. Despite the video's poor deepfake quality as evidenced in Figure

1, it was widely circulated on social media, raising concerns about the potential dangers of more advanced deepfake technology being maliciously deployed in politics. Such events could have drastic implications, such as election tampering, economic destabilization, or incitement of violence, thus posing a risk to the stability of many political systems.

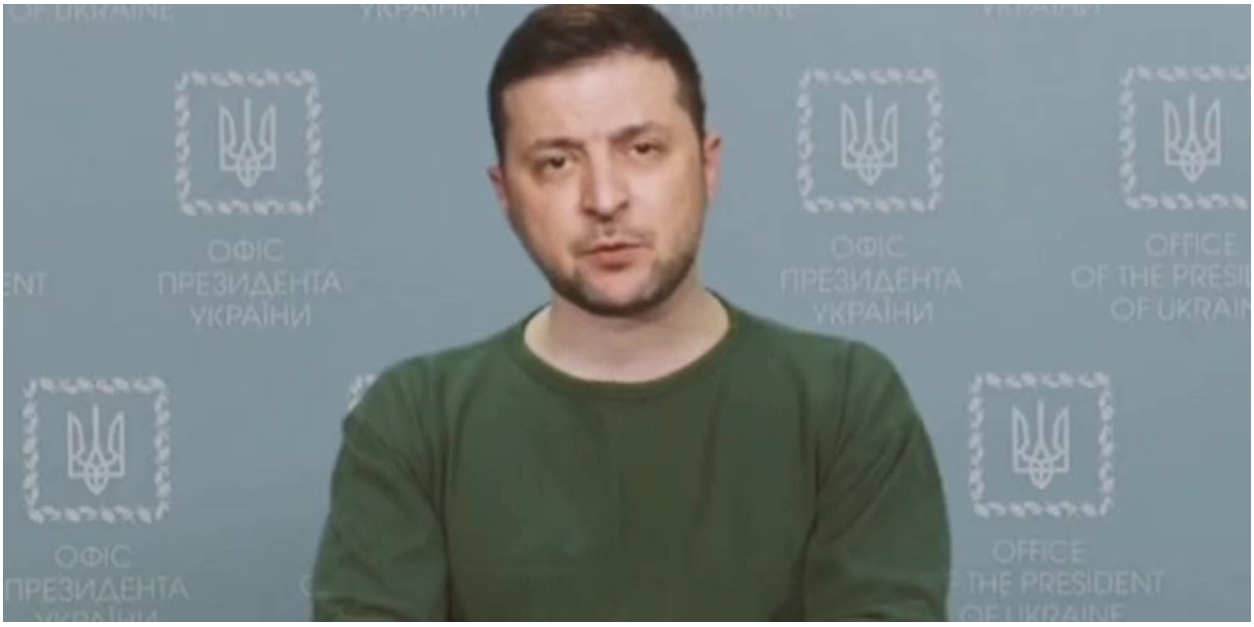

Figure 1. An extracted frame from a deepfake video featuring President Volodymyr Zelensky issuing a surrender statement to Russia during the early phases of the Russia-Ukraine conflict [5].

Legal Considerations of Deepfake Technology

In response to such unethicality, the UK government has taken combative measures by criminalizing non-consensual explicit deepfakes through an amendment to the Online Safety Bill in November 2022 [7]. Although, regulating political deepfakes poses complex predicaments, with critics suggesting that legislation could impinge on free speech and limit the use of deepfake techniques for socially beneficial purposes [8]. Moreover, banning political deepfakes would require lawmakers to provide exemptions to protect creators' rights to express themselves through satire or art that do not intend to deceive others, a challenging task due to the subjectivity of art and satire.

Social Considerations of Deepfake Technology

The availability of deepfake generation tools, including open-source software and personal laptops, has become more widespread, contributing to the democratization of artificial intelligence (AI), which Is

generally beneficial for the field. However, this increased availability also presents challenges, as it enables malicious actors to utilize these tools for harmful purposes as outlined in *Ethical Considerations of Deepfake Technology*. While individual deepfakes can be refuted through careful analysis, their broader impact has resulted in destabilization, psychological distress, and a reduction in public confidence, thereby disrupting and distorting the state of the media landscape [6].

1.5 Background

In recent years, the ability to accurately impersonate another individual's likeness has increased in frequency and sophistication due to advancements derived from artificial intelligence and machine learning. Such advancements have resulted in the emergence of deepfakes, a term conceived from the amalgamation of *deep learning* and *fake*. Deepfakes are created by training extensive models based on pattern and image recognition to comprehend the structure of the human face, ergo allowing accurate and realistic facial replications to be superimposed onto another individual's face [9]. Despite its formal introduction in 2015, deepfake technology can have massive implications upon the world of news and media, particularly in terms of misinformation and the potential for harm. Cybercriminals have exploited its capabilities to conduct financial fraud, spread fictious news, and propagate pornographic material. These occurrences have raised concerns surrounding privacy, reputation, democracy, and national security, leading the Pentagon (US Department of Defense) to perceive it as a consequential threat to democratic institutions. As deepfake technology continues to advance, the ability to differentiate between genuine and manipulated content is expected to significantly decrease, interfering with the capacity to acquire knowledge from any form of media. Consequently, it has become increasingly imperative to develop advanced systems for verification and forgery detection before the ability to distinguish deepfakes from genuine content becomes impossible for the human eye.

Rationale Basis for Selecting Deepfake Technology

The selection of deepfake technology as the focal aspect of this project was influenced by the technology's creative applications displayed on the internet, which began to gain prominence in the mainstream zeitgeist in 2017, two years after its inception [10]. Specifically, the technology gained popularity on various platforms, including YouTube, where it amassed significant attention. One notable example was a video that utilized the technology to enhance the appearance of Grand Moff Tarkin in Rogue One, a 2016 Star Wars anthology film, making him appear more lifelike and closely resembling Peter Cushing, the

late actor who originally played the role in 1977's Star Wars [11]. Subsequent uses on YouTube evolved into the exploration of alternative casting choices and de-aging actors, yielding results comparable to the costly visual effects of major movie studios prior to their incorporation of deepfakes. However, interest and wonder eventually evolved into trepidation over their potential misuses, as individuals and industries alike expressed concerns about their ability to mislead.

Suitability of the Project for a BSc Level Study

Amid concerns about the dispersion of harmful misinformation on their platforms, technological conglomerates such as Meta AI and Google have expressed unease over the increasing threat posed by deepfakes [12]. Consequently, such companies are taking proactive measures to mitigate such a risk by releasing large-scale datasets and conducting competitions aimed at accelerating the development of adept deepfake detection systems. Considering the high demand for deepfake detection systems from various entities, including government agencies and large corporations, it was evident that the development of an advanced deepfake detection system represented a suitable undertaking for a BSc level project. Such an endeavor necessitated a combination of exploring and implementing progressive research, alongside technical competence in programming, data analysis, artificial intelligence, and machine learning. In addition to technical skills, the project also demanded critical thinking abilities and the capacity to expand upon novel approaches to mitigate the detrimental societal impact of deepfake technology.

Overall, the project presents an opportunity to extend and advance the theoretical and practical knowledge acquired from introductory-level courses, such as Software Development I, to more advanced concepts covered in modules such as Machine Learning, towards a real-world predicament with drastic implications.

1.6 Report Overview

The paper is organized as follows: Section 2 provides a literature review of deepfake technology, popular existing deepfake datasets, and detection efforts, along with a technology review of resources pertaining to potential detection techniques; Section 3 presents the proposed methodology for data handling, and the theoretical basis of the techniques used for comparative analysis, along with the design of a web application aimed at deploying the optimal detection approach; Section 4 details the experimental results

of the proposed methodology and includes visualizations, as well as a comprehensive account of the implementation of the designed web application; finally, Section 5 provides a reflection on the project execution and outlined future work plans.

2 Literature & Technology Review

A literature review was conducted to explore the construction of deepfakes, available deepfake detection datasets, and potential solutions to mitigate their negative impacts. Based on the findings, a technical approach was devised that involved selecting suitable programming languages and a machine learning library, as well as a suitable model deployment method. Such components were critical for establishing a research environment to effectively address the threat of deepfake technology in visual media.

2.1 The Deepfake Generation Process

Presently, there are several facial deepfake video technologies being utilized, including Cycle Consistent Generative Adversarial Networks (Cycle GAN) [13], Face2Face [13], and deepfake technology [14]. However, while Cycle GAN and Face2Face have limited effectiveness, deepfake technology has become increasingly popular due to its effective learning of features through neural networks and face synthesis, resulting in videos that are nearly indistinguishable from authentic ones. Therefore, this study focused on deepfake technology, specifically on deepfakes generated using the Face Identity Swap method [14].

The Face Identity Swap method has garnered prominent recognition, especially in the creation of deepfakes featured in numerous large-scale datasets. Despite variations in implementation, other deepfake generation methods share the same fundamental approach to achieve visual deception. Essentially, all deepfake generation algorithms involve the use of an autoencoder neural network to alter the facial characteristics of individuals depicted in various forms of media [15]. To illustrate, the deepfake generation pipeline for Face Identity Swap is presented in Figure 2.

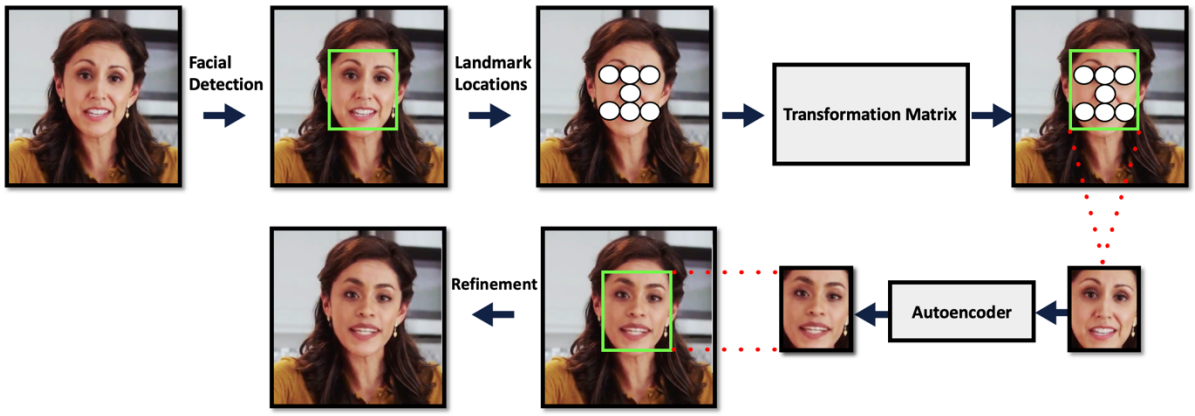

Figure 2. The process of creating a deepfake using the Face Identity Swap method. The source face is detected and extracted, then transformed into the target face using an autoencoder. The transformed face is subsequently warped back onto the original facial landmark locations and refined to achieve a convincing deepfake.

Autoencoders enable relatively simple deepfake video generation techniques by consisting of two interdependent networks: an encoder and a decoder [15]. These networks function collaboratively to compress input data into a smaller representation, which preserves essential information for later reconstruction of the original input. During the training process, an autoencoder utilizes one encoder and two decoders to generate the face-swapping effect, as illustrated in Figure 3. By condensing the input data into a compressed representation of itself, autoencoders can learn and extract salient features such as pose and movement mannerisms. These acquired features can subsequently be applied upon a new subject using an overlay technique, ultimately resulting in the production of a plausible deepfake [15].

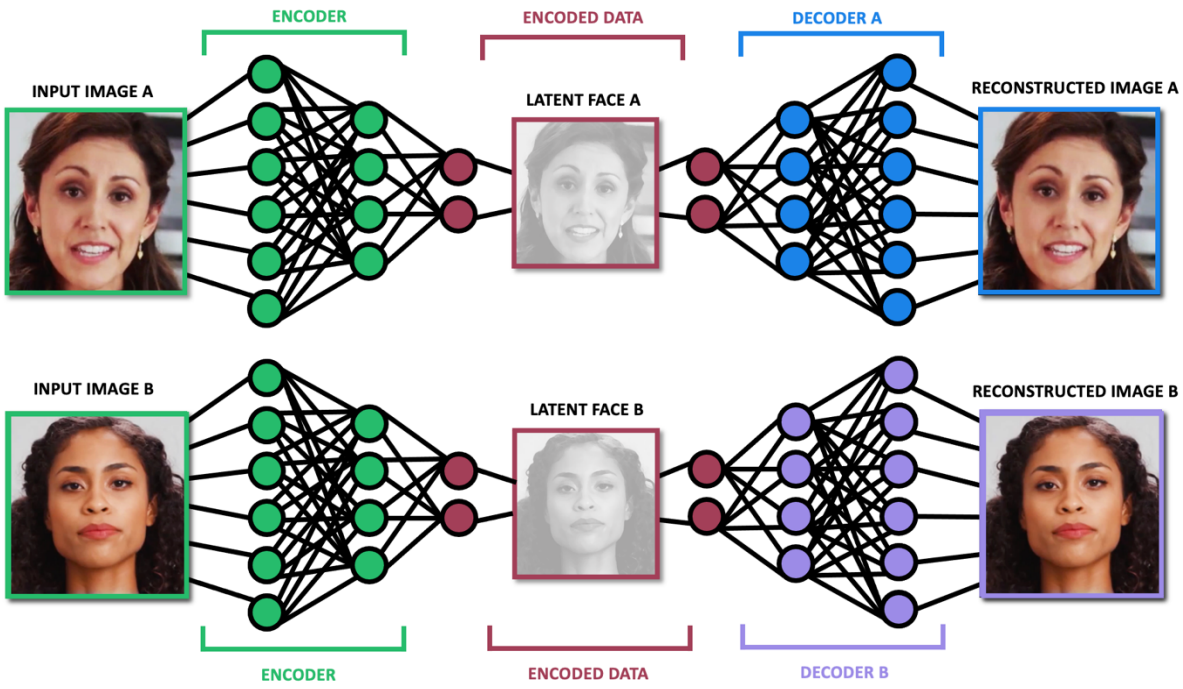

Figure 3. A representation of an autoencoder training process, in which the encoder compresses input images into a condensed representation, while the decoders reconstruct the original input. The encoder is shared for both image types, while each image type has its own decoder.

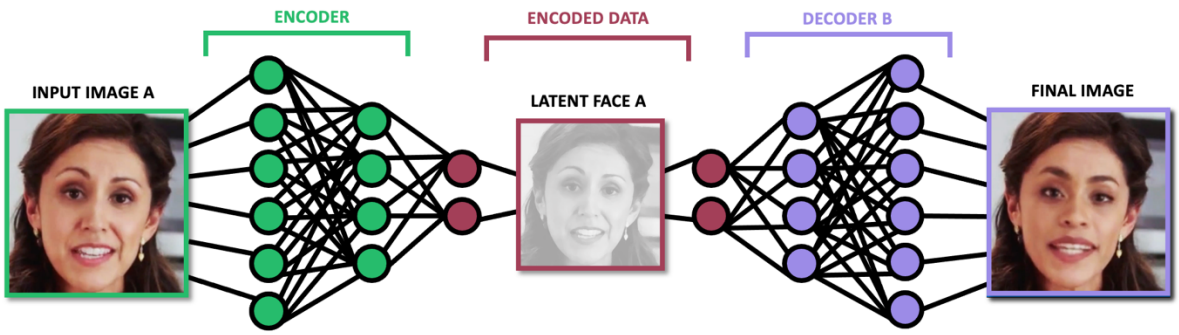

Figure 4. Deepfake Face-swapping generation achieved using trained shared encoder and Image B decoder as depicted in Figure 3.

Although autoencoders can produce acceptable results in deepfake generation, their level of feasibility heavily relies on the similarity between the features of the subjects. As emphasized in Figure 4, the credibility of the resulting image is attributed to the comparable facial structures and skin tones of the featured individuals. However, variations in these characteristics can result in less persuasive deepfakes,

as autoencoder-based techniques may not have the capacity to effectively address such deviations. As a method to combat such limitations, more advanced techniques use Generative Adversarial Networks (GANs) [16] to generate high-quality deepfakes. GANs operate through a two-part network system where the two networks, the generator and discriminator, compete against each other in a game-like manner to enhance the overall quality of the generated content. The generator network aims to create fake data that closely resembles the real data, while the discriminator network distinguishes between real and fake data [16]. A significant aspect of GANs is that the generator network is not exposed to the real data and depends entirely on the feedback received from the discriminator to enhance the quality of the fake image as presented in Figure 5.

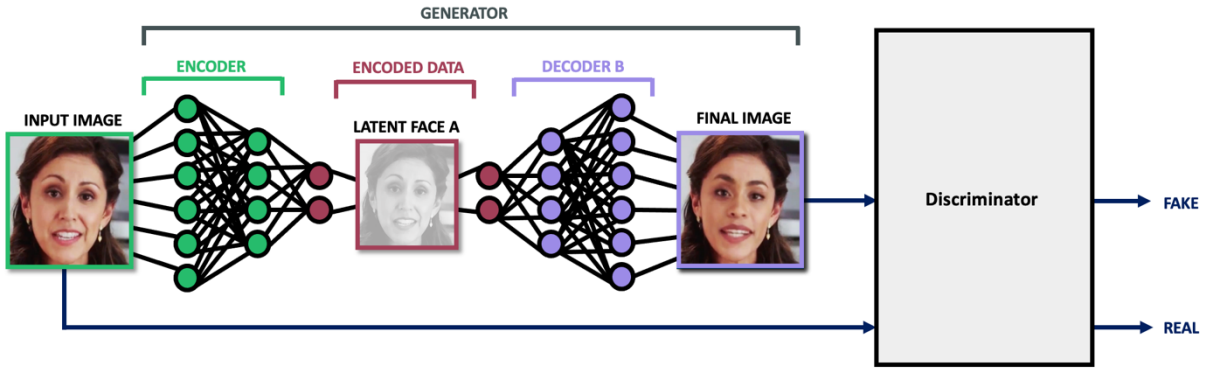

Figure 5. In the context of deepfake creation, the generator network functions as the decoder that produces the fake image, while the discriminator network compares the fake image to the original input image.

The potency of GAN-constructed deepfakes has presented arduous challenges for developing deepfake detection methods, as GANs can generate manipulated content that can evade such detection techniques. Despite this, the computational cost and time required to train them limits their accessibility to advanced users, resulting most users to opt for the conventional autoencoder method as an alternative for generating deepfakes.

2.2 Overview of Deepfake Detection Datasets

In recent years, with the rise of deepfakes since 2015, many datasets consisting of both real and deepfake videos have become accessible, collected from the internet or custom-made using volunteers. Although

there are many deepfake video datasets available, only a few are widely used due to their convenient organization of data. These include datasets of differing scales.

The FaceForensics++ Deepfake Dataset

The FaceForensics++ Deepfake dataset [17], curated by Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, and Christian Riess, is a notable dataset of a large scale. This dataset comprises of an expansive collection of over 1,000 manipulated videos, along with 1,000 authentic videos that are characterized through a range of lighting conditions, facial expressions, and head poses with each video possessing a resolution of 1080p and varied durations.

Figure 6. Frames from videos in the FaceForensics++ Deepfake dataset. The left side displays a frame from a deepfake video, while the right side shows a frame from the authentic video.

The Deepfake Detection Challenge (DFDC) Dataset

Another large-scale collection of data that is widely used for training and evaluating deepfake detection models is the DFDC dataset [18]. This dataset features over 4,000 deepfake videos and 1,000 authentic videos, all with sufficient resolution. It also includes a diverse collection of faces with varying attributes such as gender, race, and age; and was custom assembled with the consent of over fifty participants as part of a deepfake detection challenge organized Facebook and hosted on a web-based data science environment called Kaggle [19].

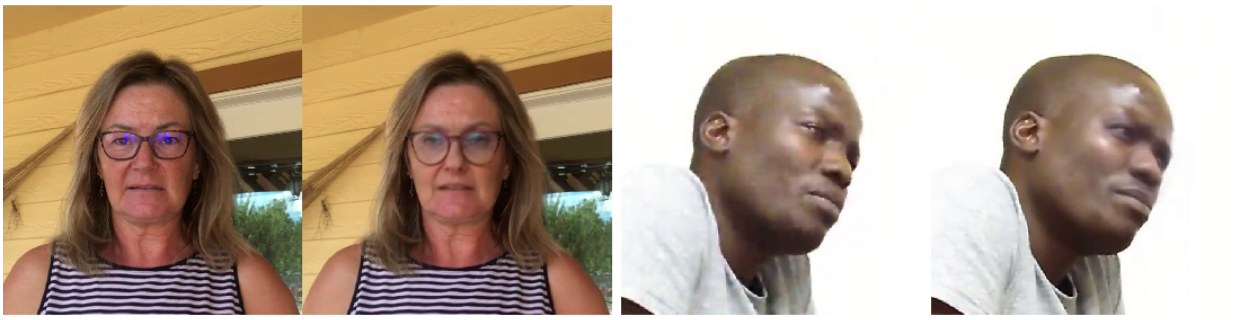

Figure 7. Side-by-side comparison of real and deepfake videos of two participants from the Deepfake Detection Challenge dataset, displaying frames from each. The left half of each participant's portrayal represents a frame from the real video, while the right half presents a frame from the corresponding deepfake video.

The Deepfake-TIMIT Dataset

In addition to large-scale datasets, the use of smaller-scale datasets can also aid in the development of deepfake detection models. One such dataset is the Deepfake-TIMIT dataset [20], which was one of the first datasets to include both authentic and deepfake videos. The dataset includes a subset of sixteen matched pairs of individuals from the open-source VidTIMIT database, with 620 manipulated videos created using two different models with varying quality levels. It has been used as a benchmark for various deepfake detection models (see Section 2.3.2) due to its unique feature of distinctive facial positions of the subjects in its videos, which simplifies face localization for pre-processing. However, many of the deepfakes provided contain visible artifacts in the face boundary, which are indicative of a low quality deepfake, as illustrated in Figure 8C.

Figure 8. Example frames from the Deepfake-TIMIT dataset showing the original donor (8A), the original target (8B), and the resulting face-swapped deepfake (8C) [20].

2.3 Challenges & Initiatives in Deepfake Detection

Mitigating the adverse effects of deepfake technology requires the development of effective techniques for identifying deepfakes and distinguishing them from authentic media. In low-quality deepfakes, visual inconsistencies are often apparent, such as a sense of an “uncanny valley” where the deepfake replicates most humanistic characteristics but contains noticeable flaws. Other signs may include a lack of synchronicity between the words spoken and the facial expression displayed, facial warping or discoloration, or the absence of blinking. However, high-quality deepfakes can be much difficult to detect as they may not contain the same visual artifacts as low-quality deepfakes.

To address such challenges, researchers are actively engaged in developing deepfake detection algorithms to detect deepfakes, akin to the discriminator network used in GANs. One notable initiative was the Deepfake Detection Challenge launched by Facebook in 2019, which provided a dataset (see Section 2.2) for participants to develop algorithms for detecting deepfakes. Although promising, the ever-evolving nature of deepfake technology presents ongoing difficulties. Specifically, newer versions of deepfakes may outpace previous detection models that were effective on earlier versions due to the swift pace at which deepfakes are being generated and improved. Therefore, as detection models become more efficient, deepfakes are likely to become more advanced.

2.3.1 Enhancing Deepfake Detection with Deep Learning

Given the rapid pace of advancements in deepfake technology, researchers are also investigating alternative approaches to enhance the reliability of detection algorithms. Among these approaches is the utilization of deep learning, a subset of machine learning that involves the use of neural networks to identify complex patterns [13]. These networks employ many hidden layers that extract high-level information from raw input data. The number of hidden layers required is determined by the complexity of the training data. Hence, as the complexity of the data increases, additional layers become necessary to effectively capture the underlying patterns, facilitating the acquisition of more abstract features and leading to more dependable outcomes [13].

Recently, deep learning techniques have gained prominence due to their high effectiveness in various fields such as natural language processing and computer vision, exceeding traditional machine learning methods. As a result, researchers have utilized deep learning’s abstract feature learning capabilities to

develop more robust deepfake detection algorithms that address the continuously evolving state of deepfake technology. Many scholars have conducted research on this topic, utilizing techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and more. Therefore, a concise review of previous work on deepfake detection using deep learning techniques was executed.

2.3.2 Related Work on Deepfake Detection Using Deep Learning

Y. Li, S. Lyu [21] presented a deep learning-based approach to identify deepfake videos by capitalizing on a limitation of deepfake generation algorithms, which produce face images of a fixed size and necessitate affine warping to match the source's face configuration. Consequently, a CNN was employed to detect artifacts caused by resolution inconsistencies between the resulting warped face area and its surrounding context. The model was trained by simulating such inconsistencies, and the experimental outcomes showcased that the proposed method was effective, with a best-case accuracy of 96.6% on the Deepfake-TIMIT dataset.

D. Guera and J. Delp, [22] presented a deep learning recognition pipeline to automatically detect deepfakes, using a two-step analysis. Firstly, the proposed method extracted features at the frame level using CNN. Secondly, a temporally aware RNN network captured aberrant frames introduced due to the face-swapping process. The performance of the proposed method using a dataset of 600 videos obtained from various online sources was evaluated, reporting a high accuracy of 94.0%.

B. Bayar and M. C. Stamm, [23] proposed a unique approach for deepfake detection using deep learning. Specifically, their approach employed a new type of convolutional network layer, called a constrained convolutional layer, that can learn deepfake detection features directly from training data. This universal forensic approach eliminated the need for pre-selected features or pre-processing steps. Ultimately, their experiments demonstrated the approach's ability to automatically detect various types of deepfake manipulations with high accuracy. On private web data, the approach achieved an accuracy of 99.1%.

Cozzolino et al, [24] introduced a method that utilized a 3D morphable model to extract facial features and an adversarial learning strategy to focus on temporal behaviour. The method was thoroughly analyzed and achieved an accuracy of 77.2% on the DFDC dataset.

Rahmouni et al, [25] presented a deep learning algorithm that utilized a CNN with a customized pooling layer to enhance the feature extraction process of computer-generated images, which was extended to detecting deepfake images. The method was tested on recent photo-realistic computer graphics and yielded better results than state-of-the-art methods for both local and full image classification.

Y. Li, MC. Chang, and S. Lyu, [26] proposed a novel approach for detecting deepfakes by exploiting an unrealistic eye blinking patterns present in most deepfake videos, caused by the limited number of training images with the subject's eyes closed in the deepfake generation process. Therefore, their approach involved pre-processing videos to locate the face region and used a combination of CNN and LSTM neural networks to analyze the eye region across a sequence of frames. The method displayed an excellent performance on a small deepfake dataset of 100 videos, with an accuracy of 99.0%.

2.4 Model Construction Tools

In the project, the selection of a suitable programming language and library was crucial for developing deepfake detection techniques. Hence, Python [27] was chosen due to its clear syntax, language familiarity, and rich libraries for data processing and model training. Among the several libraries within Python, TensorFlow [28] and PyTorch [29] were considered due to their popularity, documentation, and community support.

After carefully assessing both of their qualities, TensorFlow was selected due to its scalability and ease of use in a research environment, given prior experience with it and its high-level API, Keras. Ergo, this choice aided the reduction of the learning curve and ensured that the project requirements were met.

Selection of Face Extraction Tool

Upon the selection of Python and TensorFlow for development machine learning models, further research was conducted into pre-processing tools, with a focus on face detection as a necessary pre-processing step for deepfake detection. Two main approaches were considered, including face_recognition [30] and Multi-Task Cascaded Convolutional Networks (MTCNN) [31]. Face_recognition, a user-friendly library with a high-level API for face recognition tasks, was noted for its decent accuracy in recognizing faces in images and videos; while MTCNN, a deep learning-based face detection library, was recognized for its ability to detect facial landmarks at high accuracy in computer vision applications.

Ultimately, MTCNN was selected due to its facial landmark detection capabilities, resulting in more precise extractions. Although MTCNN required more computational resources than face_recognition, accuracy was prioritized over convenience and was factored into the project management approach.

2.5 Model Deployment Tools

A comprehensive investigation was conducted to determine the most efficient method of deploying the deepfake detection model for public use, considering important factors such as user-friendliness and accessibility. After assessing several deployment options, a basic web application was selected, which allowed users to upload videos or images for deepfake analysis. The decision to opt for a web application over mobile app alternatives was based on its compatibility with the Pythonic backend structure selected. Developing a web application also enabled a prompt creation of a prototype, thereby expediting the prototype development timeline suitable for the project's scope.

To create the web application, a conventional client-server architecture was adopted that consisted of two primary factors: the frontend (client) and the backend (server). The frontend component of the applications was responsible for user interaction and provided a visual representation of the application's functionality. Conversely, the backend was responsible for managing requests from the frontend and ensuring the efficient operation of the application behind-the-scenes.

Evaluation of Backend Options

In the process of conceiving the backend implementation of the application, two prospective frameworks, Django [32] and Flask [33], were carefully evaluated for their practicality with the Pythonic deepfake detection technique. Django is an intricate web framework tailored for Python that can support complex web applications and facilitate full-stack development. With an extensive toolkit, it also allows for building robust and secure applications. In contrast, Flask is a lightweight web framework that is also well-suited for Pythonic backend-frontend configurations. Its popularity amongst web developers is largely due to its simplicity, versatility, and flexibility in developing smaller web applications.

Upon careful evaluation, Flask was selected as the preferred framework for the web application based on its suitability for the intended purpose. While Django possesses a sophisticated framework and extensive toolset, the simplicity and flexibility offered by Flask were better suited for a small-scale application. Given that the deepfake detection web application does not require a complex infrastructure, Flask's quick

development capabilities also enable the construction of an application within the confines of the project timeline, further affirming its superiority over Django in this instance.

Evaluation of Frontend Options

To effectively implement a user interface that showcased the deepfake detection technique capabilities, a thorough evaluation of two frontend options were conducted. React [34], a popular JavaScript framework known for its proficiency in developing dynamic web applications, was compared to Vanilla JavaScript [35], a simplistic approach that utilizes Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript. React's advantages include scalability and the use of Virtual DOM, which enhances the performance of applications that require frequent updates, whereas, Vanilla JavaScript, though less advanced, is highly effective and user-friendly, making it a suitable option for beginners.

Ultimately, Vanilla JavaScript was chosen to develop the frontend of the application, primarily due to prior experience and familiarity with the core technological languages involved, as well as its user-friendliness. However, it is worth noting that the React framework was also a viable option for the scope of the project and could have been a suitable option to utilize for the application's UI.

3 Methodology & Design

In this section, a comprehensive overview was presented on the development process and techniques used to construct two deep learning CNN architectures for deepfake detection, along with the design of their deployment as a web application. As such, details relating to data collection, data pre-processing, model architectures, and web application design were outlined to provide a clear understanding of the intended deepfake detection system.

3.1 Data Collection

To establish the foundation for constructing reliable deepfake detection algorithms, a preliminary literature review was conducted to acquire a high-quality dataset for training and verification purposes (see Section 2.2). The review examined multiple public deepfake detection datasets available from the internet, considering various factors such as the number of videos, resolution, and quality of deepfakes. The analysis included large scale datasets such as FaceForensics++ and DFDC, as well as a small-scale dataset, Deepfake-TIMIT.

Upon review, it was observed that many deepfake videos from the DFDC and Deepfake-TIMIT dataset displayed suboptimal resolution and an abundant number of artifacts that failed to accurately represent the deepfakes currently circulating on the internet. Hence, the study presented the utilization of the FaceForensics++ Deepfake dataset, which included manipulated videos of an adequate quality for the development and evaluation of deepfake detection algorithms. The dataset was sourced from Kaggle and can be found at https://www.kaggle.com/datasets/sorokin/faceforensics.

3.2 Procedures for Data Pre-Processing

With the careful selection of the FaceForensics++ dataset, a series of transformative preprocessing steps were executed to prepare the video-level data for optimal model training. Specifically, these steps encompassed data conversion, face image extraction, data splitting, and data formatting to ensure that it can be fed into the intended CNN architectures properly.

Figure 9. A diagram of the pre-processing steps applied to an individual video (excluding data splitting as it pertains to the entire dataset).

3.2.1 Data Conversion & Extraction Process

In the initial step of data preparation, the original video-level format of the FaceForensics++ dataset was converted to an image-level format. Such a process involved the extraction of individual frames from the videos, which was necessary for the CNN models under consideration, as they required image-level data as input. To facilitate this process in an organized manner, the videos were split into designated folders based on their authenticity (real or deepfake), and a Pythonic pipeline was employed to extract frames from each video at one-second intervals. As a result, the pipeline was able to yield a comprehensive dataset consisting of 19,727 images, despite the original dataset comprising of 2,000 videos.

However, a significant challenge arose due to the varying durations attributed by the videos within the FaceForensics++ dataset, leading to an unequal distribution of frames between the authentic and deepfake classes. Precisely, 9,864 authentic images and 9,863 deepfake images were obtained, resulting in a one-image disparity between the two categories. While it was assumed that the videos in each class were fundamentally identical, differing only in the presence or absence of deepfake features; the minute variations in video duration underscored a flaw in the extraction pipeline, ultimately resulting in a non-proportional frame extraction. Despite this flaw, the extracted frames were still deemed suitable for model training and were annotated to their respective folders for further processing.

3.2.2 Face Extraction & Data Curation

To introduce accuracy and eliminate noisy data in the newly converted image data, intricate face recognition and cropping tasks were performed. In particular, the advanced MTCNN algorithm was utilized for face detection (see Section 2.4), known for its accuracy and efficiency in detecting facial

landmarks such as the eyes, nose, and mouth [31]. Subsequently, such landmarks were used to precisely crop the image, creating a square image with the face in the center. This precision ensured that the face was the focal point of the image, eliminating any extraneous elements that could potentially interfere with the models' ability to learn the intricate characteristics of the human face.

Following the filtering process, a refined dataset of 17,071 images (comprising of 8,554 authentic images and 8,517 deepfake images) was obtained. Although this represented over a 13% reduction from the initial number of images that were extracted, it was imperative to ensure the data's quality by reducing excessive noise.

Moreover, a curation process was carried out to ensure a balanced number of images within the newly formed dataset, given that there was a slight overrepresentation of authentic images by 0.43% in comparison to deepfake images. Such a process was necessary to ensure that the models would be trained on data that accurately represented an unbiased distribution of both authentic and deepfake images. As a result, a subset of 17,000 images was carefully curated from the dataset of 17,071 images, comprising of precisely 8,500 authentic images and an equal number of deepfake images. In the end, the pre-processed dataset represented a trustworthy and well-calibrated compilation, achieved through curation that minimized discrepancies, rendering it optimal for training the models. Table 1 provides an overview of the number of images at each pre-processing step for further clarification.

PRE-PROCESSING ACTION	# REAL IMAGES	# DEEPFAKE IMAGES	# TOTAL IMAGES
Initial Frame Extraction	9,864	9,863	19,727
Frame Filtering with MTCNN	8,554	8,517	17,071
Balancing of Classes	8,500	8,500	17,000

Table 1. Results from the initial data pre-processing steps.

3.2.3 Data Splitting & Formatting

After performing data cleaning and filtering, significant information such as file location, label (real or deepfake) and facial landmarks were established for each image as previously indicated; the data was

organized and saved into a csv file to be easily accessible as a reference point for dividing the available data into three distinct subsets, particularly, the training set, validation set, and testing set. Specifically, a random 70:15:15 ratio was implemented to partition the data into these subsets. This commonly selected partitioning scheme was implemented to enable effective execution of key stages of model development, such as model training, monitoring, and performance evaluation [36]. Explicitly, the training set of 11,900 images was used to train our model, while the validation set of 2,550 images was utilized to assess the model's performance during training, and the testing set of 2,550 images was reserved for evaluating the finalized predictive performance of our model. For vivid context, the count of each set is represented graphically in Figure 10.

Figure 10. Graphical representation of the count of images from each set of a sample dataset. Since the original dataset was biased towards authentic images, an equal proportion of real and deepfake images were selected for the sample dataset.

In the concluding phase of the image pre-processing pipeline, all images from each set were subjected to a sequence of modifications. These include conversion to the BGR2RGB color format, resizing to dimensions of 224 * 224 * 3, transformation into float32 NumPy arrays, and normalization to conform to the input format specifications of the proposed CNN architectures (see Section 3.3).

3.3 Proposed Network

Convolutional Neural Networks (CNNs) have been instrumental in advancing computer vison, particularly in image classifications tasks [13]. For deepfake detection, CNNs have become the preferred approach due to their scalability, capacity, and feature extraction capabilities. In this project, two distinct CNN architectures were proposed to fulfil such a task: a baseline architecture and a pre-trained CNN architecture, EfficientNet-B0. By training both models on a shared subset of pre-processed data from the FaceForensics++ dataset, the performance of the two models could be measured and compared for future analysis (see Section 4.1).

3.3.1 The Baseline CNN Architecture

In the context of machine learning, employing a baseline model serves as a frame of reference for comparing the performance of more complex models [37]. Therefore, to evaluate the potential performance improvement offered by the pre-trained model, a comparative analysis was launched, involving a custom baseline CNN architecture. While both proposed models share similar CNN-based structures, the baseline model adopted a simpler approach, without scaling techniques to balance depth, width, and resolution. Given such an approach, the baseline model was expected to have limited predictive power compared to the more intricate pre-trained model, making it a suitable candidate for comparison to ascertain its reliability.

To construct a baseline model of minimal complexity, Pythonic machine learning libraries such as TensorFlow and Keras were utilized (see Section 2.4). The final architecture comprised of a Keras Sequential object with seven Conv2D and MaxPooling2D layers, followed by a Flatten layer and two Dense layers that used ReLU and sigmoid activation functions, respectively, for classification. To decrease the fit for each convolutional layer, a Dropout layer with a rate of 0.5 was included as the final layer. Overall, the baseline model was essential for providing a benchmark for comparing the performance of the pre-trained

EfficientNet-B0 model and was recommended as the initial model to be developed to initiate such comparisons.

3.3.2 The Pre-trained Architecture

In contrast to the baseline architecture, a pre-trained CNN model can be utilized to accelerate the training process and improve feature extraction for more accurate deepfake classifications. Among the pre-trained networks available, the EfficientNet architecture [38] has emerged as a popular choice due to its ability to provide state-of-the-art performance using fewer parameters than comparable models. Such efficiency is achieved through a scaling method that balances depth, width, and resolution, resulting in a network that is both adept and effective, a notable advantage over the baseline architecture. Therefore, to address the need for a model scaling approach that takes both speed and accuracy into consideration for the computationally intensive task of deepfake detection, the EfficientNet series network was proposed.

To commence, the EfficientNet network processes a three-channel color image with a specified width (W) and height (H) through a series of convolutions to learn relevant features, as represented in Figure 11A. Figures 11B-11D demonstrates the three ways the network can be unilaterally scaled; Figure 11B involves expanding or reducing the input image size to improve resolution and learning adequacy; Figure 11C modifies the number of channels per layer to extract more information from the input; Figure 11D adjusts the number of layers to capture intricate and complex features and optimize computational efficiency. Overall, the EfficientNet network applies composite parameters, as presented in Figure 11E, to concurrently perform the scaling of these three techniques and achieve optimal performance. Given its improved capabilities, the finalized architecture of the EfficientNet network served as a basis for the selected model choice: EfficientNet-B0 [39].

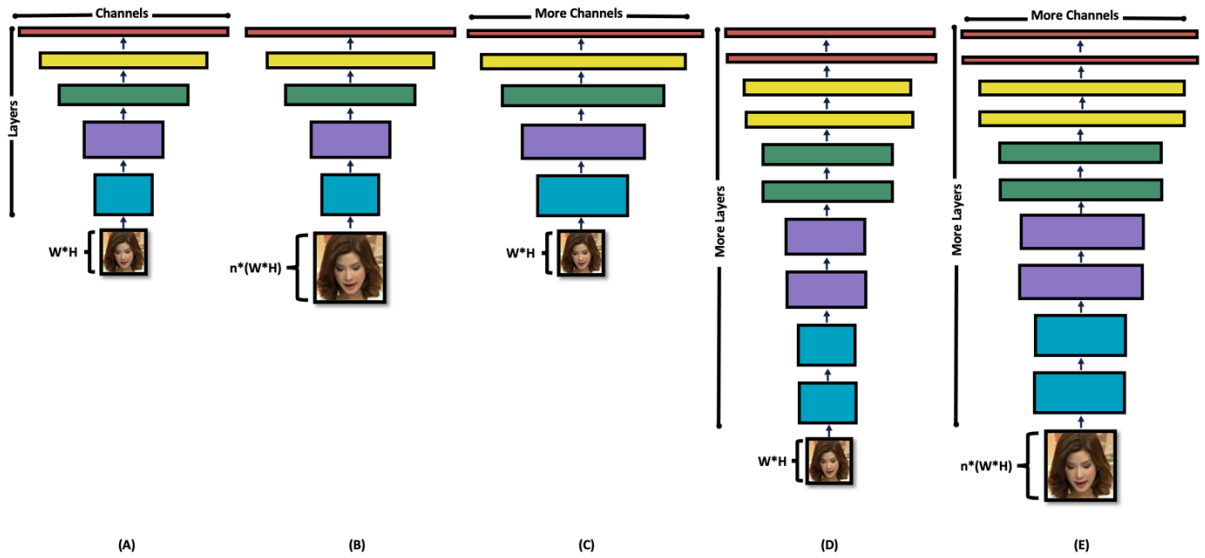

Figure 11. A comparison of different methods for model scaling. Panel A depicts an example of a baseline network, while panels B-D represent conventional scaling methods that increase only one dimension of network width, depth, or resolution. Panel E shows a proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio [40].

EfficientNet-B0 is an accurate and highly efficient deep learning framework for image classification tasks within the EfficientNet network family [41]. While larger models within the family, like EfficientNet-B1 to B7, offer higher accuracy, they are less practical for the scope of this project due to increased parameters and computational cost. However, EfficinetNet-B0 provided a balance between model size and accuracy, with only 11 million trainable parameters, enabling lightweight and efficient deployment while maintaining high accuracy [39]. Figure 2 presents the network framework of EfficientNet-B0, which is divided into nine stages, each consisting of a distinct number of layers, input resolution, and output channels specified in each row.

Stage	Operator	Resolution (H * W)	# of Channels	# of Layers
1	Conv3x3	224 * 224	32	1
2	MBConv1, k3x3	112 * 112	16	1
3	MBConv6, k3x3	112 * 112	24	2
4	MBConv6, k5x5	56 * 56	40	2
5	MBConv6, k3x3	28 * 28	80	3
6	MBConv6, k5x5	14 * 14	112	3
7	MBConv6, k5x5	14 * 14	192	4
8	MBConv6, k3x3	7 * 7	320	1
9	Conv1x1 & Pooling & FC	7 * 7	1280	1

Table 2. The EfficientNet-B0 framework [42].

As illustrated in Table 2, the EfficientNet-B0 network architecture is primarily composed of the Mobile Inverted Residual Bottleneck (MBConv) block, which consists of several stages, including squeeze & excitation convolutions [41]. Ultimately, the increasing number of MBConv blocks enhance the model's depth and complexity. The architecture starts with an initial stage (Stage 1), consisting of a 3x3 convolutional layer, followed by batch normalization and the Swish activation function. This stage serves the dual purpose of extracting of low-level features while simultaneously reducing the spatial size of the input [40]. In subsequent stages (Stages 2-8) multiple MBConv blocks are stacked upon each other to extract increasingly abstract features that Stage 1 was unable capture. The final stage (Stage 9) includes 1x1 convolutional layers, global average pooling, and a fully connected layer, each with batch normalization and the Swish activation function. The Swish function Is used as the final activation function for producing predictions. The expression of the Swish activation function, defined as $f(x) = x * \text{sigmoid}(\beta x)$, introduces non-linearity to facilitate learning of intricate patterns and input-output relationships. By setting β as a trainable parameter, the model can optimize it during training for better performance [40].

To incorporate the careful composition of such a framework into a Pythonic environment, the same machine learning libraries as those used to build the baseline CNN architecture, such as TensorFlow, and Keras were utilized. Moreover, to augment the model's ability to extract valuable features from images, pre-trained weights from the ImageNet database were integrated. Additional layers were also included to

enhance its performance by enabling a more nuanced analysis of input data. In detail, these layers consisted of a GlobalAveragePooling2D layer and a fully connected Dense layer with a sigmoid activation function. The GlobalAveragePooling2D layer served as a pivotal asset in reducing the dimensionality of the feature maps produced by the EfficientNet-B0 base model, which prevented overfitting by reducing the number of parameters in the architecture; whereas the fully connected Dense layer with a sigmoid activation function predicted the likelihood of the input image being either authentic or deepfake.

3.4 Model Training Pipeline

The training of machine learning models is a multifaceted and iterative procedure that necessitates the consideration of various factors, such as image augmentation techniques, compilation parameters, and model fitting metrics. Upon the completion of data pre-processing and model designing, three precisely formatted sets of data were obtained: training, validation, and testing (see Section 3.2.3). These sets of data were critical components of the model training pipeline and were employed in tandem with several techniques to optimize the performance of the two models.

3.4.1 Data Augmentation Techniques for Improved Model Performance

To enhance the performance of both models, a widely used technique in deep learning, data augmentation, was employed to expand the training set size by introducing arbitrary modifications to the input images during training. The implementation aimed to introduce greater variety in the training data, thus compelling the models to learn invariant features that were not affected by the transformations. To accomplish this within the study, the *ImageDataGenerator* class was utilized in Keras, enabling a range of transformations to be applied to the images in the training set. These transformations included random rotation, width and height shifts, shear, zoom, and horizontal flipping. Additionally, the *fill_mode* parameter was set to *nearest* to preserve the pixel values of the original images, ergo reducing the risk of introducing artifacts in the image data. By introducing such techniques, the models were encouraged to learn robust features that were generalizable to new examples while avoiding overfitting to the training data.

3.4.2 Model Learning Process Configuration

Moreover, prior to the commencement of model training, it was essential to configure the learning process by specifying a loss function, an optimizer, and an evaluation metric.

- Loss function: To assess the efficacy of the models in distinguishing between authentic and deepfake images, the binary cross-entropy loss function was employed. It calculated the scalar loss value of the two labels accurately by measuring the difference between predicted probabilities and actual binary labels, which was crucial for the deepfake detection classification task.
- Optimizer: An optimizer was utilized to minimize the loss function by updating the weights using gradients. In this study, the Adam optimizer [43] was chosen for both the baseline CNN and EfficientNet-B0 models with low learning rates of 1e-3 and 1e-5, respectively, to ensure that the models converged to an optimal solution without overfitting the training data. The Adam optimizer provides individual adaptive learning rate for each parameter [43] and is renowned for its adaptive learning rate optimization algorithm, making it an excellent choice for optimizing the ImageNet weights of the pre-trained EfficientNet-B0 network.
- Evaluation metric: The accuracy was measured for both models to provide a simple and intuitive measure of performance by determining the proportion of correct predictions over the total number of predictions.

3.4.3 Model Training Process & Evaluation Metrics

The training process of the models required a precise alignment with the available sets to produce accurate predictions. Upon defining and compiling the models in this study, the *fit* method was used to initiate the training process. The models were trained with the training and validation set, using a batch size of thirty-two, an epoch limit of twenty-five, an early stopping callback to prevent overfitting, and a MacBook Pro (13-inch 2017, Two Thunderbolt 3 ports) laptop with a 2.3 GHz Dual-Core i5 processor serving as the experimental setup. To accelerate the process, Google Colab Pro was utilized to compile the model training code, offering ample cloud storage capabilities and substantial GPU support [44]. Specifically, the model performances were evaluated at each epoch, taking several minutes to complete, with the training process subjected to halt if the validation loss remained unchanged for five consecutive epochs. The resulting training history was saved for analysis, and the trained models were tested on the testing set to evaluate their predictive capability (see Section 4.1).

3.5 Model Deployment: Web Application Design

To deploy the best-performing model for practical use through a web application, a simple and user-friendly demo was designed. With an intent to have a user upload an image or video for deepfake analysis, the user interface (UI) was kept minimalist and unobtrusive, with limited interactive elements that included an image and video upload button, as well as a prediction result display. Users could upload files in common formats such as JPEG, JPG, PNG, and MP4 to the platform using the upload button, as shown in Figure 12. To ensure an intuitive upload process, transparent error messages were implemented to address any issues with invalid file formats or additional limitations. The UI was also developed to present the prediction result in a comprehensible format that indicated the presence of deepfakes in the uploaded media.

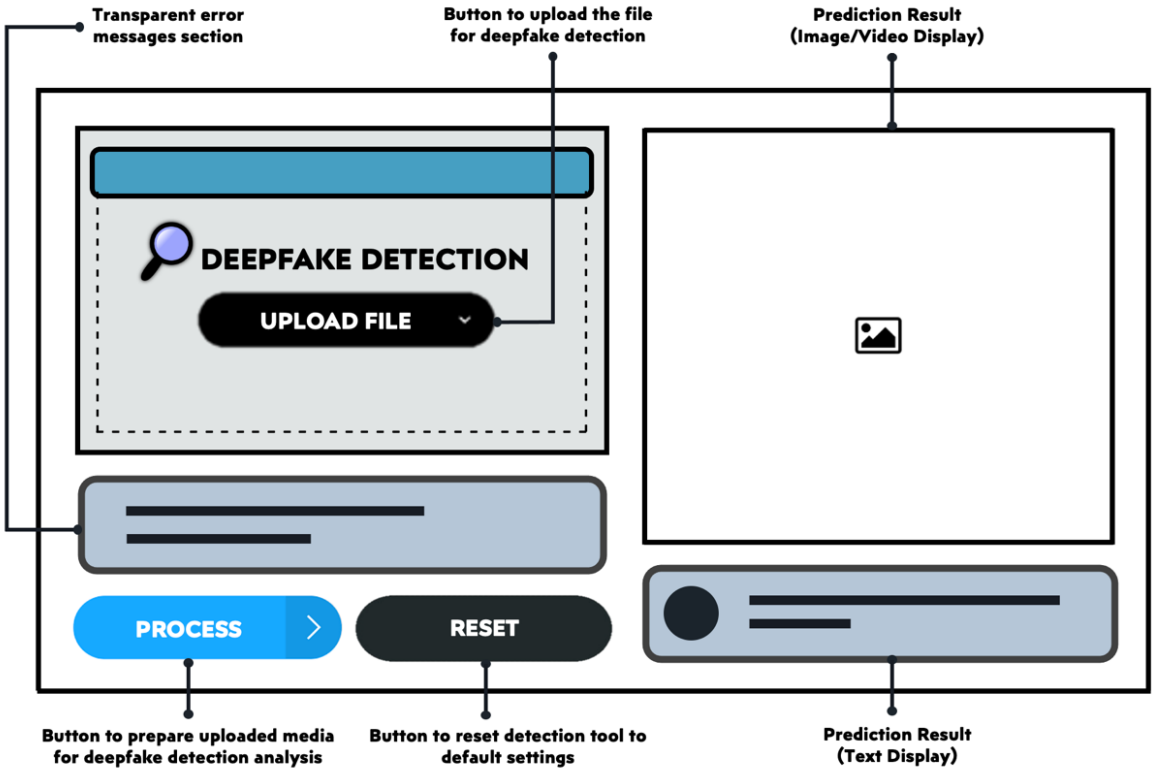

Figure 12. Deepfake Detective web application concept.

4 Results & Implementation

This section provides an overview of the results obtained from the methodology utilized to construct two CNN models for deepfake detection using the FaceForensics++ dataset (see Sections 3.1, 3.2, 3.3, and 3.4). A comparative analysis was subsequently conducted to determine the most optimal model between the two. Additionally, a detailed account of the deepfake detection web application design (see Section 3.5) implementation was presented, including its architecture, components, functionalities, and integration with the best-performing model.

4.1 Performance Evaluation of the Models

Following the completion of data pre-processing, model designing, and training (see Sections 3.2, 3.3, and 3.4), the baseline CNN model and pre-trained EfficientNet-B0 model were subjected to evaluation using three commonly utilized performance metrics: accuracy, loss, and confusion matrix. Such metrics were employed to monitor the performance of the models during training and to assess their finalized predictive capabilities on the testing set. Therefore, a proper analysis using such metrics enabled the determination of the effectiveness of the models in making accurate deepfake detection classifications.

A comparative analysis was performed between the models, employing additional metrics based on their corresponding test set confusion matrix outcomes, including *True Positive* (TP), *False Negative* (FN), and *False Positive* (FP). Precision and recall were calculated using these metrics, which were then utilized to compute the F1 score [45]. The formulae for precision, recall, and F1 score were as follows:

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 score = 2 * ((Precision * Recall) / (Precision + Recall))

4.1.1 Performance Evaluation of Baseline CNN Model

In accordance with standard practice, the performance evaluation of the baseline CNN model was conducted prior to the evaluation of the pre-trained model, as it served as a benchmark for comparison (see Section 3.3.1). Figure 13 presents the performance of the baseline CNN network on the pre-processed FaceForensics++ dataset, including accuracy and loss decay curves for the training and validation sets.

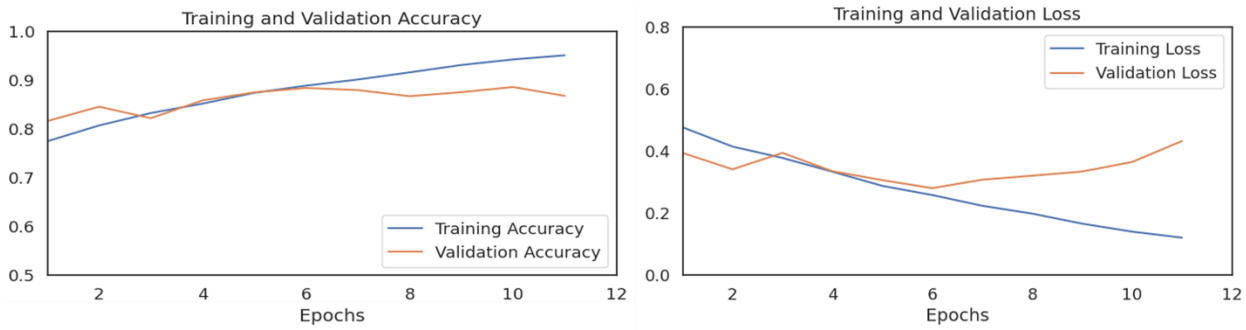

Figure 13. The performance of the baseline CNN network on FaceForensics++ training & validation sets with early stopping employed at epoch 12. The best validation accuracy of 86.8% was achieved at epoch 7, of which the weights from the end of this epoch would be restored for the finalized model.

During the training process, depicted in Figure 13, the accuracy and loss curves of the baseline CNN model demonstrated temporary convergence and subsequent stabilization in different directions throughout epochs. The real-time curve of the training set appeared to be smoother and followed a consistent slope compared to the validation set, which fluctuated and faltered to a near-constant state throughout the later stages of epochs. Notably, the validation accuracy attained values of approximately 88.6% and 86.7%, while the loss value decayed to approximately 0.281 and 0.307, respectively. These findings suggest that the baseline CNN model was successful in accurately classifying deepfake images, with high accuracy and low loss values.

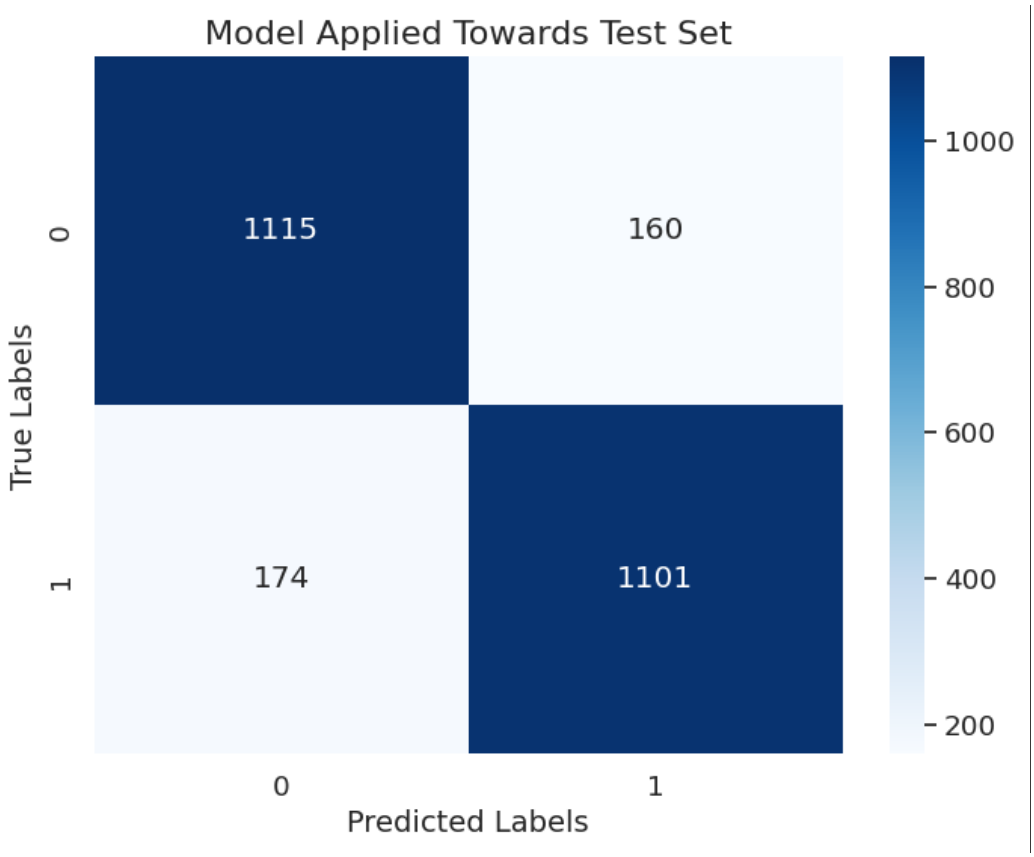

Figure 14. Confusion matrix of the baseline CNN model test set predictions against the actual test set classifications.

To further evaluate the efficacy of the baseline CNN model, it was applied on the testing set, and its predictions were assessed using a confusion matrix as presented in Figure 14. The results indicated a high-level performance, with an accuracy rate of 86.9% and correct classification of 2,216 out of 2,550 total images. With an accuracy rate surpassing the 80% threshold, a significant precedent was established for the EfficientNet-B0 model to either meet or exceed.

4.1.2 Performance Evaluation of EfficientNet-B0 Model

Subsequent to the evaluation of the baseline CNN model, the training performance of the EfficientNet-B0 model was also examined. Based on the graphs, depicted in Figure 15, the validation accuracy curve exhibited a fluctuating rise throughout the epochs, ultimately reaching a plateau of 94.2%, whereas the validation loss curve demonstrated a gradual but noisy decline, attaining a minimum value of approximately 0.169. In contrast, the training curves displayed a consistent and quick trajectory towards optimization, which may be indicative of the pre-trained features of the model already being trained to

detect various features. As a result, the validation curves were unable to keep pace, leading to an absence of convergence and a slight disparity between the training and validation curves. Nonetheless, the overall findings were encouraging, with the pre-trained model achieving a desirable learning rate with an ideal slope [46], suggesting its capabilities of classifying deepfakes with a high degree of precision.

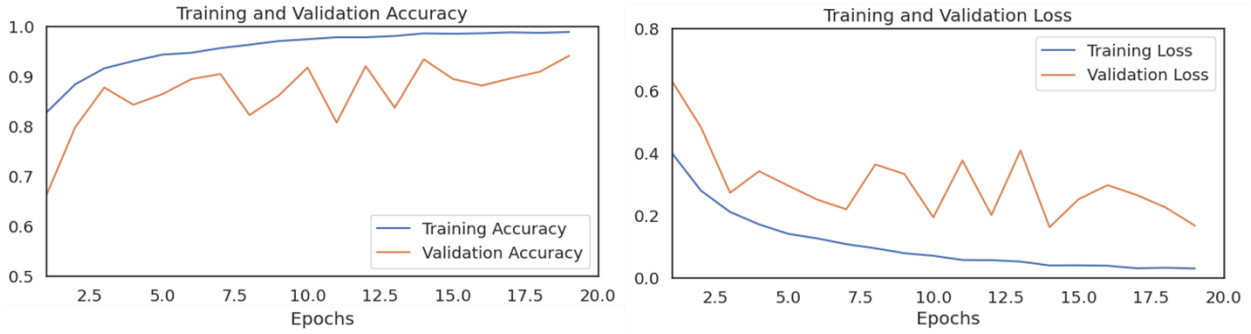

Figure 15. The performance of the EfficientNet-B0 network on FaceForensics++ training & validation sets with early stopping employed at epoch 20. The best validation accuracy of 94.2% was achieved at epoch 15, of which the weights from this epoch would be restored for the finalized model.

Expanding upon its training performance, the finalized EfficientNet-B0 model was also evaluated on its predictive ability on the testing set. The results were analyzed and presented in the form of a confusion matrix, as shown in Figure 16. The analysis revealed that the EfficientNet-B0 model achieved an impressive accuracy of 93.3%, accurately predicting 2,378 out of 2,550 test set images, with a relatively low error rate of 6.7%. These results indicate that the EfficientNet-B0 model was able to successfully leverage its pre-trained functionalities and generalize its learned patterns to unseen data with precision that exceeded the commonly accepted industry-standard benchmark of 90%.

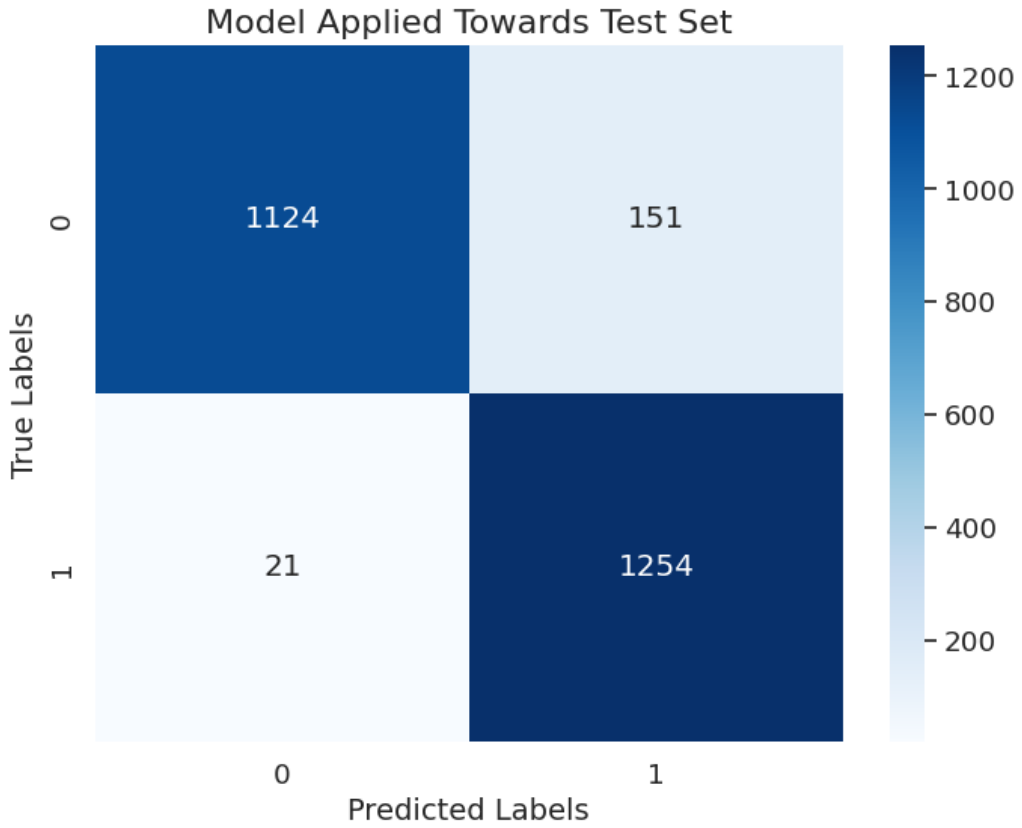

Figure 16. Confusion matrix of the EfficicentNet-B0 model test set predictions against the actual test set classifications.

4.1.3 Comparative Analysis of EfficientNet-B0 & Baseline CNN Performances

Upon gathering the performance results based on the test sets from the confusion matrices presented in Figure 14 and 16, the precision, recall, accuracy, and F1 score for both the baseline CNN model and EfficientNEt-B0 model were computed and compiled in Table 3 for easy comparison.

MODEL	ACCURACY (%)	PRECISION (%)	RECALL (%)	F1 SCORE (%)
EfficientNet-B0	93.3	89.3	98.4	93.6
Baseline CNN	86.9	87.3	86.4	86.8

Table 3. Results obtained from model test set performances.

Based on the information presented in Table 3, it was apparent that the performance of the EfficientNet-B0 model demonstrated superior results across all metrics, outperforming the baseline CNN model. Despite possessing a satisfactory accuracy, the baseline CNN model exhibited a higher misclassification rate of 334 images, whereas the EfficientNet-B0 model had only 172 misclassifications. With EfficientNet-B0's capacity to reduce the number of misclassified images by nearly 50% contributed significantly to the observed 6.4% increase in accuracy, reaching 93.3%. Overall, this outcome aligned with the anticipated capability of the EfficientNet-B0 architecture and indicated that it was a viable approach for deepfake detection, with promising performance results.

For further evaluation, the study included a comparative analysis of EfficientNet-B0 with other state-of-the-art networks (see Section 2.3.2) that also utilized the FaceForensics++ dataset. Table 4 summarized the results of this comparison, with each method ranked in ascending order based on accuracy, and the best score presented in bold. The study's approach (EfficientNet-B0) outperformed other methods, including Bayar and Stamm, Cozzolino et al., Rahmouni et al., and MesoNet [47]. Notably, EfficientNet-B0 achieved the second highest accuracy in the ranking, surpassing the previous second-ranked method, MesoNet, by 5.9% (see Table 4). However, XceptionNet's performance remained superior [48], with EfficientNet-B0 falling short by approximately 3.1% to equate to its accuracy.

DEEPFAKE DETECTION NETWORK	ACCURACY (%)
Bayar and Stamm	84.6
Cozzolino et al.	85.5
Rahmouni et al.	85.5
MesoNet	87.3
EfficientNet-B0	93.3
XceptionNet	**96.4**

Table 4. The performance of the FaceForensics++ dataset on various state-of-the-art networks [11].

4.1.4 Limitations of the EfficientNet-B0 Model

Despite achieving an encouraging outcome, the EfficientNet-B0 deepfake detection approach had several limitations:

- Due to time constraints, the technique was limited to analyzing images or frames containing only a single visible subject. As a result, accommodating multiple individuals within a single frame was not possible.
- The method relied on appropriate lighting conditions and the subject facing frontward for accurate detection by the MTCNN face detector tool. Hence, the subject's eyes, nose, and mouth had to be visible during the face extraction process, making it challenging for the approach to comprehend complex facial angles.
- The technique required a close proximity of the subject to the frame and adequate quality and resolution of the frame to prevent the MTCNN face detector from encountering difficulties in extracting the necessary facial features for effective deepfake analysis.
- The limited representation of a video through the frame extraction pipeline (extracting one frame per second), may result in the misclassification of a video, as certain deepfake videos from external sources may be infused with authentic footage intended to deceive detection systems. The approach was not trained on such instances, and thus, misclassification is plausible.

Despite the identified limitations, the study successfully demonstrated the potential of EfficientNet-B0 on the FaceForensics++ dataset, resulting in a highly effective deepfake detection model. The model achieved a binary detection accuracy score of 93.3% on the test set, comparable to other advanced models such as MesoNet. Although the primary objective of developing a robust detection model was achieved, the accuracy set by XceptionNet (96.4%) on similar data was not surpassed. Therefore, future research could investigate advanced methodologies to improve model performance and overcome its limitations, aiming to outperform XceptionNet's benchmark performance.

4.2 Web Application Implementation

In the development of our web application, a conventional client-server architecture was adopted to demonstrate the deepfake detection proficiency of the best performing model, EfficientNet-B0. Specifically, the frontend was constructed utilizing Vanilla JavaScript, while the backend was implemented using the Flask (see Section 2.5). This section details the implementation of Flask and its integration with the frontend for responsive communication and data processing.

Flask Application Structure

The Flask application was structured into several components, including the main application package directory, configuration classes, static files, HTML web page templates, the optimal detection model (EfficientNet-B0), and a module containing the DeepfakeDetective class for detecting deepfakes in images and videos.

Routes & Views

Flask's routing system was utilized in the web application to create routes that aided in structuring, comprehending, debugging, and extending the application. The application consisted of three routes, including the Home route and two Deepfake Detection Tool routes.

The Home route functioned as the initial landing page for the applications and was accessed through the root URL ("/") and handled GET and POST HTTP methods and was implemented using a view function and an HTML template for rendering the landing page. Figure 17 displays the resulting landing page as seen by the user.

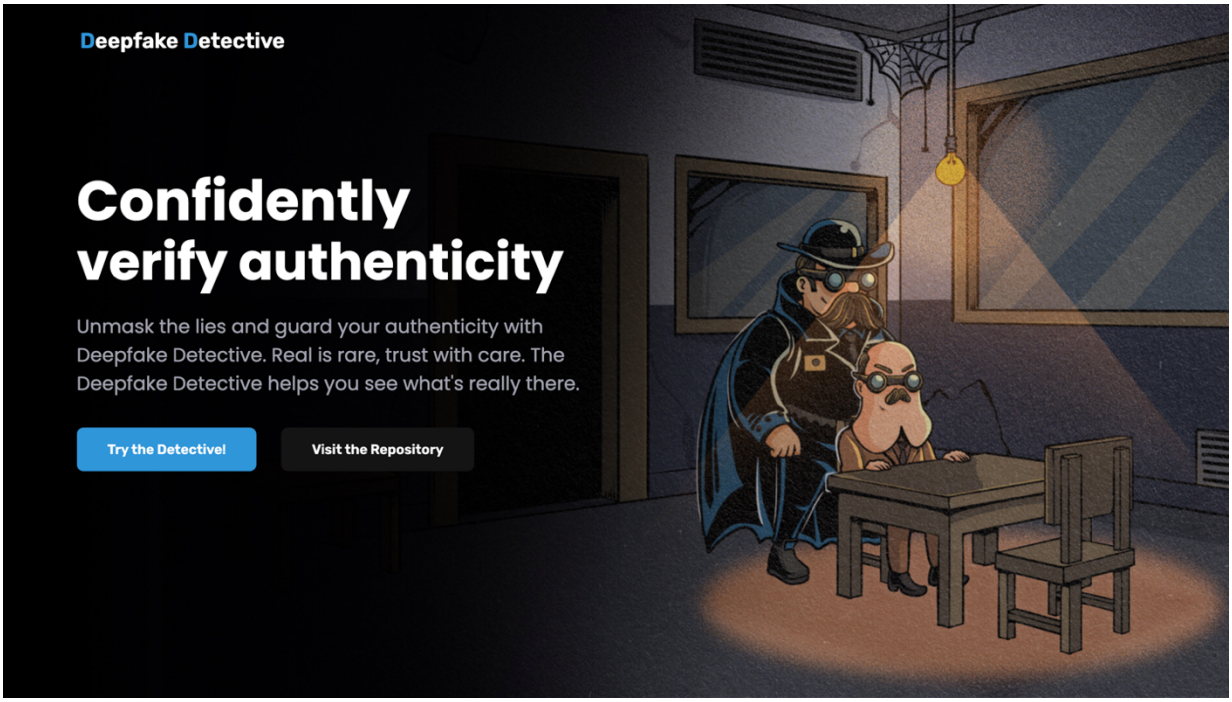

Figure 17. Home page of the Deepfake Detective web application.

The Deepfake Detective Tool routes were designated for the handling and analysis of both image and video files. In particular, one route managed image uploads while the other managed video uploads. Both routes allowed for GET and POST HTTP methods and used the DeepfakeDetective class to detect any deepfakes present in the uploaded files. Figure 18 presents the finalized Deepfake Detective tool as seen by the user.

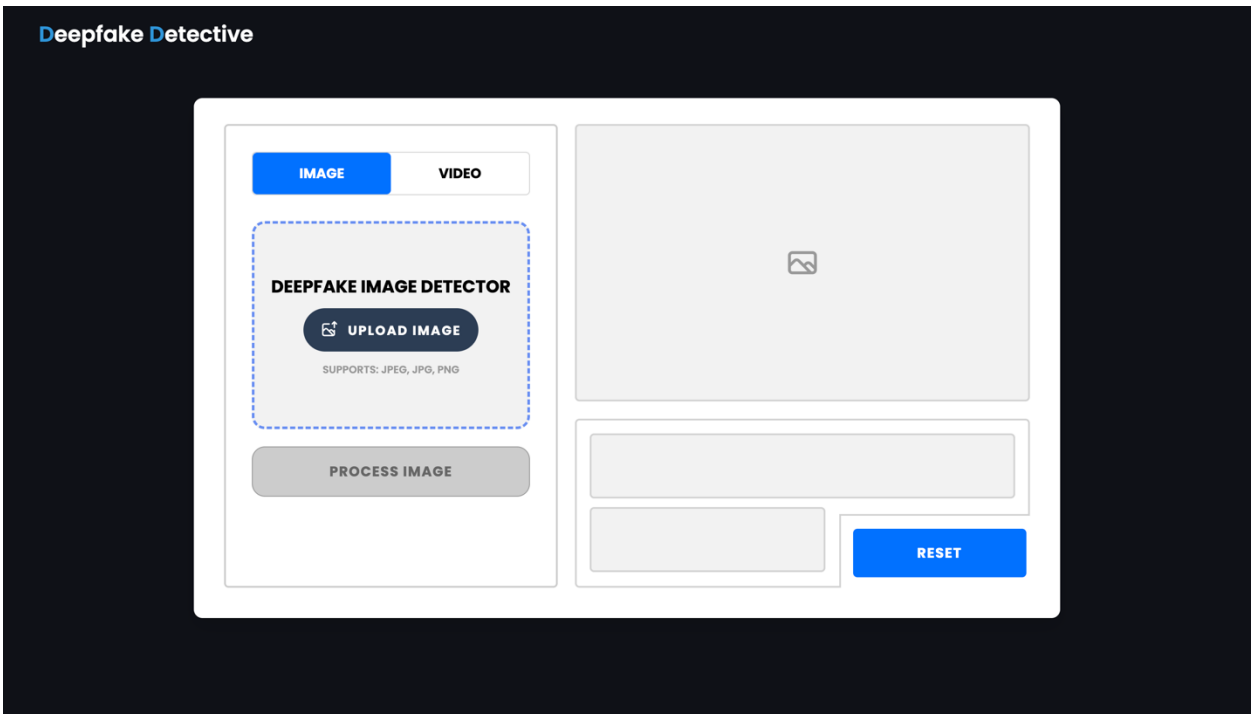

Figure 18. Upload file page of the Deepfake Detective web application.

Backend & Frontend Processing of User-Submitted Files

The Vanilla JavaScript-built frontend utilizes AJAX to transmit a POST request to the Flask API backend upon submission of a file with extensions such as PNG, JPG, JPEG, for images, or MP4 for videos. Upon the receipt of the file, the appropriate view function is employed based on the file type. Thereafter, the backend utilizes a Python-based extraction pipeline (see Section 3.2) and applies the MTCNN face recognition tool to extract the subject's face, assuming that only one subject is present in the content. For image uploads, a single face frame is extracted, whereas multiple frames of faces are extracted for video uploads. Once facial frame extraction is completed, the frame is subjected to pre-processing techniques (see Section 3.2.3) to adhere to the input format requirements of the EfficientNet-B0 model. The processed frame is then passed through the model to compute classification results and determine the

file's authenticity. Such results are then returned to the client-side JavaScript for rendering, with a button spinner functionality indicating the progress of the deepfake analysis. Figure 19 provides a diagram illustrating this process.

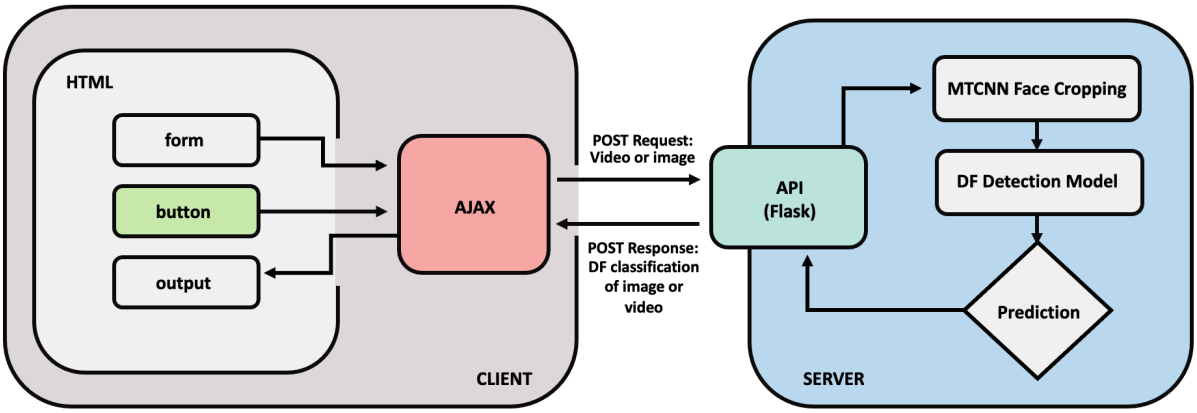

Figure 19. Standard client-server architecture comprising of a frontend consisting of Vanilla JavaScript and a backed developed using Python and Flask.

5 Conclusion

Overall, the project effectively utilized two CNN architectures to detect the authenticity of content trained from the FaceForensics++ dataset, achieving its machine learning objectives with reliable results. Furthermore, the best performing model, EfficientNet-B0, was deployed in a web application to showcase its capabilities to users.

Despite such successes, the project encountered challenges related to planning and management that hindered motivation towards tasks such as pre-processing and web application development. Such challenges were primarily attributed to a computationally inadequate laptop and a steep learning curve, resulting in a reduced dataset size, slower detection analysis, and inability to further enhance the web application design. Upon reflection, the importance of better planning, motivation, achievable goals, task division, and suitable resource selection were recognized as crucial for optimal project development, and their implementation would have been beneficial. Applying such lessons learned will enable the project to address the identified issues and improve future developments.

5.1 Future Work

Automated detection techniques necessitate advancements to keep up with the continuous evolution of deepfake generation methods. While this project yielded an effective model with evident limitations (see Section 4.1.4), its long-term efficacy is uncertain, prompting the development of innovative intelligent techniques to address the negative implications of realistic deepfakes.

Future research should aim to construct more robust and dynamic deepfake detection models capable of identifying and extracting new abstract features by utilizing diverse datasets beyond FaceForensics++ and exploring complex architecture variations other than EfficientNet-B0 to enhance detection accuracy. Moreover, the use of a more advanced facial detection tool can improve the practicality of the technique by overcoming limitations presented by the MTCNN face recognition tool employed in the extraction pipeline, facilitating accurate detection of faces even in challenging settings. Other improvements include reducing detection processing time, analyzing multiple subjects in a frame, and improving the web application’s user experience.

6 References

[1] S. Irani, “On the moral gray area of Star Wars’ use of deepfake technology,” *The Michigan Daily*, Mar. 20, 2023. https://www.michigandaily.com/arts/b-side/this-is-not-the-luke-skywalker-youre-looking-for/ (accessed March 11, 2023).

[2] A. Middleton, “A Twitch streamer was caught watching deepfake porn of women gamers. Sexual images made without consent can be traumatic and abusive, experts say — and women are the biggest victims.,” *Insider*, Feb. 10, 2023. https://www.insider.com/atrioc-caught-qtcinderella-ai-picture-twitch-deepfake-controversy-streamer-trauma-2023-2

[3] 16i- https ://www.16i.co.uk, “Deepfakes and their impact on women,” *DAC Beachcroft*. https://www.dacbeachcroft.com/en/gb/articles/2021/august/deepfakes-and-their-impact-on-women/

[4] J. Wakefield, “Deepfake presidents used in Russia-Ukraine war,” *BBC News*, Mar. 18, 2022. Available: https://www.bbc.co.uk/news/technology-60780142

[5] “Ukraine war: Deepfake video of Zelenskyy telling Ukrainians to ‘lay down arms’ debunked,” *Sky News*. https://news.sky.com/story/ukraine-war-deepfake-video-of-zelenskyy-telling-ukrainians-to-lay-down-arms-debunked-12567789

[6] 16i- https ://www.16i.co.uk, “Political Deepfakes: social media trend or genuine threat?,” *DAC Beachcroft*. https://www.dacbeachcroft.com/en/gb/articles/2022/september/political-deepfakes-social-media-trend-or-genuine-threat/

[7] A. Hern, “Online safety bill will criminalise ‘downblousing’ and ‘deepfake’ porn,” *The Guardian*, Nov. 25, 2022. Available: https://www.theguardian.com/technology/2022/nov/24/online-safety-bill-to-return-to-parliament-next-month

[8] “Snapshot Paper - Deepfakes and Audiovisual Disinformation,” *GOV.UK*. https://www.gov.uk/government/publications/cdei-publishes-its-first-series-of-three-snapshot-papers-ethical-issues-in-ai/snapshot-paper-deepfakes-and-audiovisual-disinformation

[9] M. Koopman, A. M. Rodriguez, and Z. Geradts, “Detection of Deepfake Video Manipulation,” Aug. 31, 2018. https://www.researchgate.net/profile/Zeno-Geradts/publication/329814168_Detection_of_Deepfake_Video_Manipulation/links/5c1bdf7da6fdccfc705da03e/Detection-of-Deepfake-Video-Manipulation.pdf (accessed Feb. 06, 2023).

[10] M. Westerlund, “The Emergence of Deepfake Technology: A Review,” *Technology Innovation Management Review*, vol. 9, no. 11, Nov. 2019, Available: https://timreview.ca/article/1282

[11] P. Suciu, “Deepfake Star Wars Videos Portent Ways the Technology Could Be Employed for Good and Bad,” *Forbes*. https://www.forbes.com/sites/petersuciu/2020/12/11/deepfake-star-wars-videos-portent-ways-the-technology-could-be-employed-for-good-and-bad/ (accessed Feb. 19, 2023).

[12] L. Finger, “Deepfakes - The Danger of Artificial Intelligence That We Will Learn To Manage Better,” *Forbes*. https://www.forbes.com/sites/lutzfinger/2022/09/08/deepfakesthe-danger-of-artificial-intelligence-that-we-will-learn-to-manage-better/

[13] T. Nguyen, C. Nguyen, D. Nguyen, T. Duc, Nguyen, and S. Nahavandi, “Deep Learning for Deepfakes Creation and Detection.” Available: https://arxiv.org/pdf/1909.11573.pdf

[14] A. Abdulreda and A. Obaid, “A landscape view of deepfake techniques and detection methods,” *Int. J. Nonlinear Anal. Appl*, vol. 13, no. 1, pp. 2008–6822, 2022, Doi: https://doi.org/10.22075/ijnaa.2022.5580.

[15] A. Zucconi, “Understanding the Technology Behind Deepfakes - Alan Zucconi,” *Alan Zucconi*, Mar. 14, 2018. https://www.alanzucconi.com/2018/03/14/understanding-the-technology-behind-deepfakes/

[16] I. D. A. in S. Valley, “Deepfakes and the world of Generative Adversarial Networks,” *Medium*, Dec. 26, 2019. https://medium.com/@lennartfr/deepfakes-and-the-world-of-generative-adversarial-networks-bf6937e70637

[17] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images.” Available:

https://openaccess.thecvf.com/content_ICCV_2019/papers/Rossler_FaceForensics_Learning_to_Detect_Manipulated_Facial_Images_ICCV_2019_paper.pdf

[18] B. Dolhansky *et al.*, "The Deepfake Detection Challenge Dataset," *arXiv:2006.07397 [cs]*, Jun. 2020, Available: https://arxiv.org/abs/2006.07397

[19] Kaggle, "Kaggle: Your Home for Data Science," *Kaggle.com*, 2022. https://www.kaggle.com/

[20] admin, "DeepfakeTIMIT," *Idiap Research Institute, Artificial Intelligence for Society*. https://www.idiap.ch/en/dataset/deepfaketimit

[21] Y. Li and S. Lyu, "Exposing Deepfake Videos by Detecting Face Warping Artifacts," *arXiv:1811.00656 [cs]*, May 2019, Available: https://arxiv.org/abs/1811.00656

[22] D. Guera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Nov. 2018, doi: https://doi.org/10.1109/avss.2018.8639163.

[23] M. Belhassen Bayar and Stamm, "Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection." Accessed: May 04, 2023. [Online]. Available: https://misl.ece.drexel.edu/wp-content/uploads/2018/04/BayarStammTIFS01.pdf

[24] D. Cozzolino, A. Rössler, J. Thies, M. Nießner, and L. Verdoliva, "ID-Reveal: Identity-aware Deepfake Video Detection." Available: https://openaccess.thecvf.com/content/ICCV2021/papers/Cozzolino_ID-Reveal_Identity-Aware_DeepFake_Video_Detection_ICCV_2021_paper.pdf

[25] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, "Distinguishing Computer Graph-ics from Natural Images Using Convolution Neural Networks," 2017. Accessed: May 04, 2023. [Online]. Available: https://hal.science/hal-01664590/file/without_IEEE_logo.pdf

[26] wer deepfakers, “FaceForensics++ & Survey of Multi-Modal techniques,” *Medium*, Oct. 17, 2019. https://medium.com/@werdeepfakers/faceforensics-survey-of-multi-modal-techniques-7b637fc161d0 (accessed Feb. 27, 2023).

[27] “Python (in Machine Learning),” *Corporate Finance Institute*. https://corporatefinanceinstitute.com/resources/data-science/python-in-machine-learning/

[28] J. Brownlee, “Introduction to the Python Deep Learning Library TensorFlow,” *Machine Learning Mastery*, May 04, 2016. https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/#:~:text=TensorFlow%20is%20a%20Python%20library

[29] “Why PyTorch Is the Deep Learning Framework of the Future,” *Paperspace Blog*, Oct. 30, 2019. https://blog.paperspace.com/why-use-pytorch-deep-learning-framework/

[30] “Face Recognition with Python face_recognition and OpenCV,” *Datagen*. https://datagen.tech/guides/face-recognition/face-recognition-with-python/

[31] “Face Recognition with FaceNet and MTCNN,” *arsfutura.com*. https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/

[32] “What is Django? | IBM,” *www.ibm.com*. https://www.ibm.com/topics/django (accessed Mar. 01, 2023).

[33] Agrawal, “What is Flask | A Comprehensive Guide on using Flask for Data Science,” *Analytics Vidhya*, Oct. 27, 2021. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-using-flask-for-data-science/

[34] S. Surve, “Why You Should Use React.js For Web Development,” *freeCodeCamp.org*, Feb. 18, 2021. https://www.freecodecamp.org/news/why-use-react-for-web-development/

[35] S. Azam, “What is Vanilla JavaScript?” https://linuxhint.com/what-is-vanilla-javascript/ (accessed Mar. 01, 2023).

[36] V. R. Joseph, “Optimal ratio for data splitting,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, pp. 531–538, Apr. 2022, doi: https://doi.org/10.1002/sam.11583

[37] A. Nair, “Baseline Models: Your Guide for Model Building,” *Medium*, Apr. 04, 2022. https://towardsdatascience.com/baseline-models-your-guide-for-model-building-1ec3aa244b8d

[38] A. Sarkar, “Understanding EfficientNet — The most powerful CNN architecture,” *MLearning.ai*, May 08, 2021. https://medium.com/mlearning-ai/understanding-efficientnet-the-most-powerful-cnn-architecture-eaeb40386fad

[39] D. Coccomini, N. Messina, C. Gennaro, and F. Falchi, “Combining EfficientNet and Vision Transformers for Video Deepfake Detection,” *ResearchGate*, Jul. 14, 2021. https://www.researchgate.net/publication/353042472_Combining_EfficientNet_and_Vision_Transformers_for_Video_Deepfake_Detection (accessed Mar. 04, 2023).

[40] L. Deng, H. Suo, and D. Li, “Deepfake Video Detection Based on EfficientNet-V2 Network,” *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–13, Apr. 2022, doi: https://doi.org/10.1155/2022/3441549.

[41] A. Borad, “Image Classification with EfficientNet: Better performance with computational efficiency,” *Medium*, Jul. 10, 2021. https://datamonje.medium.com/image-classification-with-efficientnet-better-performance-with-computational-efficiency-f480fdb00ac6

[42] “EfficientNet: Theory + Code – LearnOpenCV,” *learnopencv.com*, Jul. 02, 2019. https://learnopencv.com/efficientnet-theory-code/ (accessed May 02, 2023).

[43] Jason Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning,” *Machine Learning Mastery*, Jul. 02, 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[44] *Shiksha.com*, 2020. https://www.shiksha.com/online-courses/articles/how-to-use-google-colab-for-machine-learning-projects/ (accessed Feb. 26, 2023).

[45] V. Jayaswal, "Performance Metrics: Confusion matrix, Precision, Recall, and F1 Score," *Medium*, Sep. 15, 2020. https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262#:~:text=Confusion%20matrix%2C%20precision%2C%20recall%2C%20and%20F1%20score%20provides%20better

[46] G. V. Jose, "Useful Plots to Diagnose your Neural Network," *Medium*, Oct. 03, 2019. https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45

[47] Z. Xia, T. Qiao, M. Xu, X. Wu, L. Han, and Y. Chen, "Deepfake Video Detection Based on MesoNet with Preprocessing Module," *Symmetry*, vol. 14, no. 5, p. 939, May 2022, doi: https://doi.org/10.3390/sym14050939.

[48] J. Abdelkhalki, M. Ahmed, and A. Boudhir, "DEEPFAKE DETECTION BASED ON THE XCEPTION MODEL," vol. 100, no. 1, 2022, Accessed: May 12, 2023. [Online]. Available: http://www.jatit.org/volumes/Vol100No1/19Vol100No1.pdf

7 Appendices

Appendix A: Project Proposal Form

The Project Proposal Form, which was approved by the project supervisor, Dr. Charles Clarke, provided a comprehensive overview of the project's initial aims, objectives, and methodology. The form can be accessed via the following link: https://drive.google.com/file/d/1UY7e-B_2QtLCrBQ24mzAHzGiA0EZL6r-/view?usp=sharing.

Appendix B: Mid-Project Review Form

The Mid-Project Review Form served as a progress report on the project, outlining the developments made during the middle stage of project development (February 1, 2023). The progress was showcased through an organized meeting with the second supervisor, Dr. Yuanlin Gu, which included a practical demonstration of the technical implementation, as well as evidence of project management tools being utilized, and progress made on the project report. The form can be accessed via the following link: https://drive.google.com/file/d/1rxAck6S76Br7W4tdUHc4VLRXVM-GyfPi/view?usp=sharing.

Appendix C: Technical Output Access Description

To access the Deepfake Detective application, users should first visit the provided GitHub repository link: https://github.com/tyeborg/deepfake-detective. Subsequently, the repository can be cloned by executing the command "git clone https://github.com/tyeborg/deepfake-detective.git" in the terminal. Users should then access the flaskapp folder by navigating to it using the command: "cd flaskapp".

Within the flaskapp directory, users should ensure that the Docker Desktop application is open on their system before proceeding (if not already installed, it can be downloaded here: https://www.docker.com/products/docker-desktop/. To prevent conflicts, it is also essential to ensure that there are no other running containers on the the system. Typing the command "docker ps" in the terminal can be used to verify this. If any running containers exist, they should be stopped before proceeding. The Docker container can be built by executing the command "docker-compose up --build". Upon successful completion of the container build process, users can access the Deepfake Detective app via http://localhost:3000 on their web browser.

Appendix D: Model Deployment Screenshots

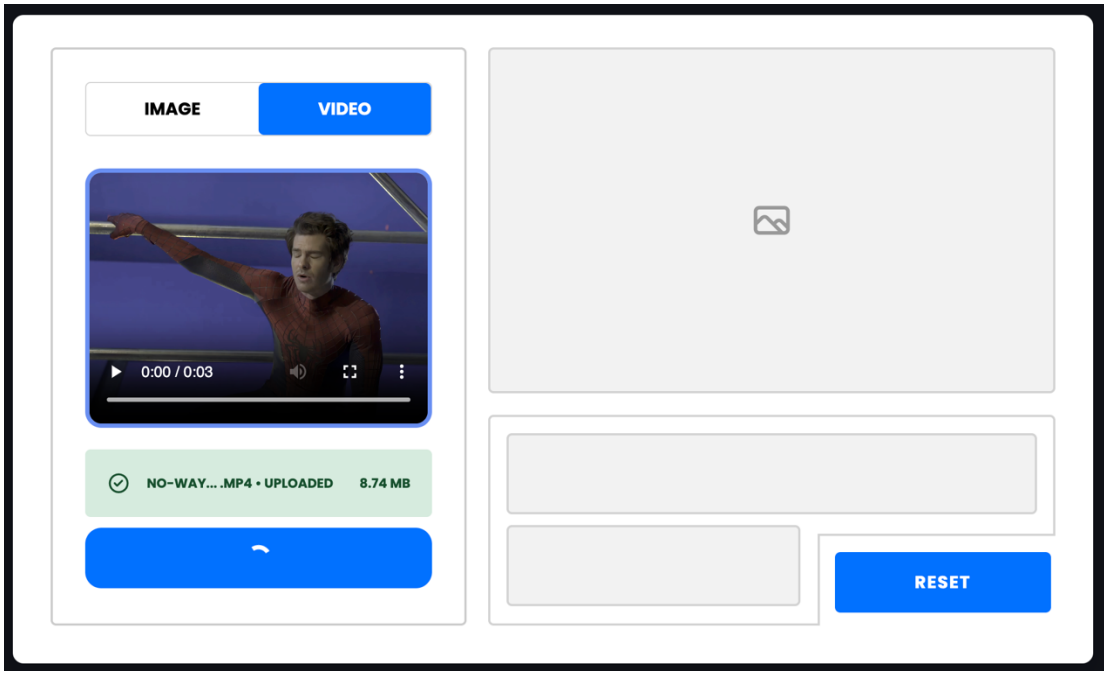

Figure D.1: Screenshot of the Deepfake Detective tool performing the processing an input video for deepfake detection.

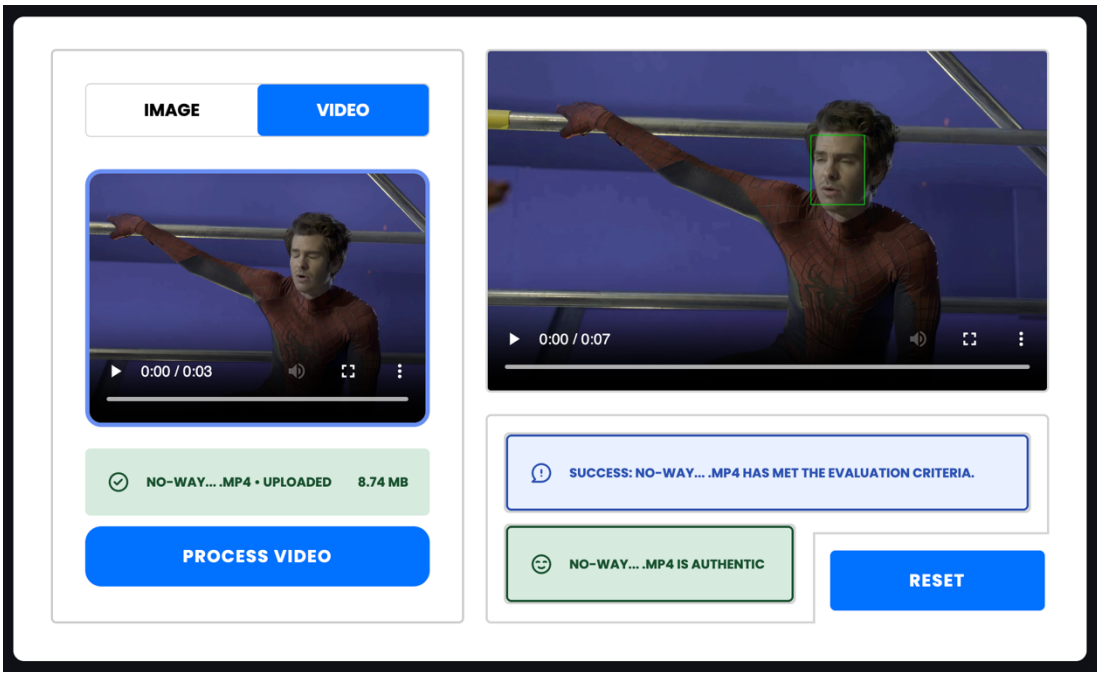

Figure D.2: Screenshot of the Deepfake Detective tool displaying the output resulting from processing an input video.

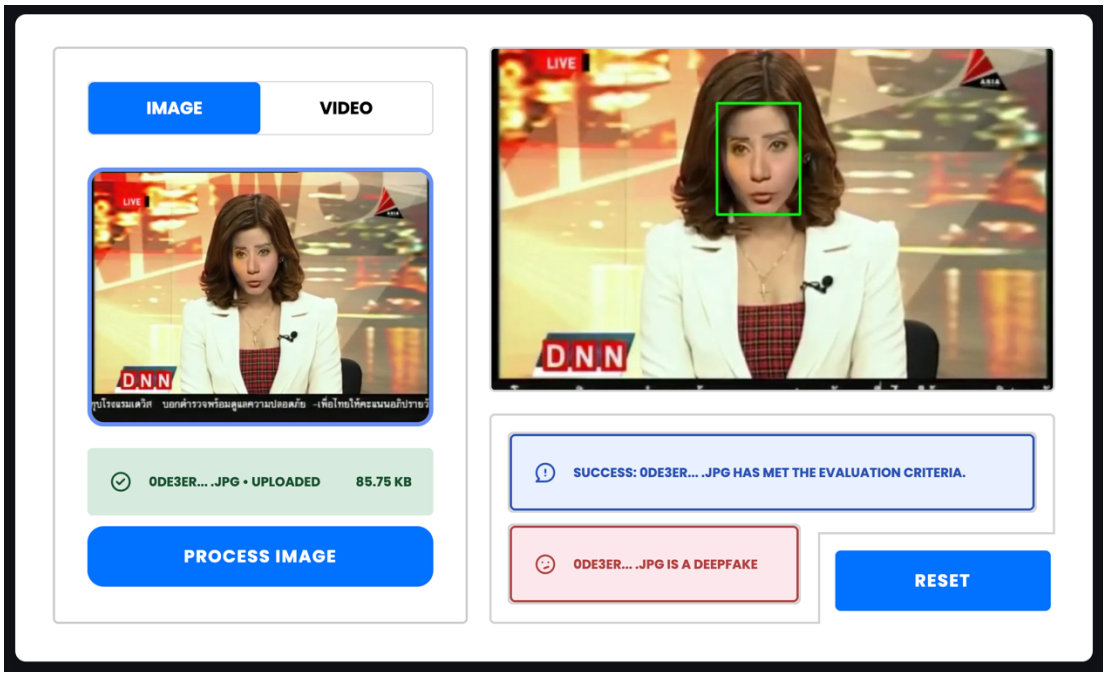

Figure D.3: Screenshot of the Deepfake Detective tool displaying the output resulting from processing an input image.

Appendix E: Deployment Diagram

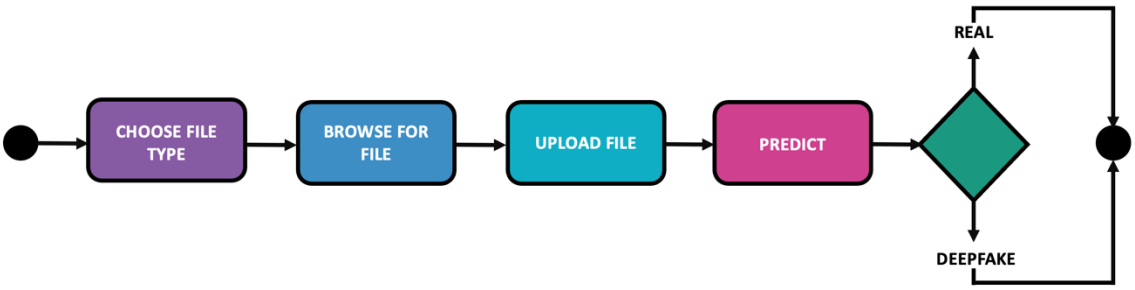

Figure E.1: Deepfake Detective web application activity diagram.