

DEEPFAKE DETECTION PROJECT REPORT

ABSTRACT


The rise of Deepfakes, synthetic media generated using Generative Adversarial Networks (GANs), presents significant challenges in integrity verification due to their increasing realism. Traditional detection methods based on pixel anomalies are becoming less effective as Deepfakes evolve. To address this, we propose DeepVision, an algorithm leveraging insights from medicine, biology, brain engineering, and machine learning to detect Deepfakes through analysis of eye blinking patterns.

Human eye blinks are influenced by various factors, including gender, age, activity, and time of day. By collecting and analyzing data on these factors, DeepVision predicts the number and frequency of eye blinks expected under specific conditions. This approach offers a unique solution for integrity verification, as eye blinking patterns are involuntary and reflect cognitive activities and behavioral factors.

Our experimental results demonstrate DeepVision's effectiveness in detecting Deepfakes, with accuracy rates improving when incorporating gender, age, activity, and time data. The incorporation of physiological cues like eye blinking patterns enhances deepfake detection capabilities, addressing challenges posed by advanced Deepfakes and contributing to the development of more robust integrity verification methods.

PLAGIARISM REPORT

We have checked plagiarism for our Project Report for our project a **Turnitin**. We are thankful to our mentor Er. Rahul Agarwal for guiding us at this. Below is the digital receipt. The Plagiarism is approximately 10%.



Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:

Sumit Hanwate

Assignment title:

Mini Project

Submission title:

DeepFake Detection

File name:

Merged_file_2_.pdf

File size:

1.72M

Page count:

39

Word count:

5,954

Character count:

35,206

Submission date:

25-May-2024 07:50AM (UTC+0530)

Submission ID:

2387632868

DEEPPAKE Detection

A PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

Arun Singh (21101024)

Anshu Singh (21101024)


Rahul Kumar (21101179)

Sumit Hanwate (21101147)

Under the supervision of

Mr. Rahul Agarwal

Assistant Professor



Department of Computer Science and Engineering

Dr. B. R. Ambedkar National Institute of Technology Jalandhar

144608, Punjab (India)

May 2024

Copyright 2024 Turnitin. All rights reserved.

Match Overview			×
10%			
<			>
1	Submitted to Dr. B R A... Student Paper	3%	>
2	Submitted to University... Student Paper	1%	>
3	Submitted to University... Student Paper	1%	>
4	Submitted to Harare In... Student Paper	1%	>
5	Submitted to ECPI Coll... Student Paper	1%	>
6	Submitted to Kaplan Pr... Student Paper	1%	>
7	Submitted to Wilmingt... Student Paper	1%	>
8	community.cadence.co... Internet Source	1%	>
9	www.tutorialspoint.com Internet Source	<1%	>

LIST OF FIGURES

Figure number	Description	Page number
2.1	Factors affecting eyeblinks	8
2.2	EAR graph for deepfake video	8
2.3	Architecture of the proposed system	9
2.4	Detecting face using fast-hyperface	10
2.5	Code for finding local landmark	10
2.6	Showing eyelid landmarks	10
2.7	EAR graph	11
2.8	Thresholding code for eyeblink	11
2.9	EAR graph showing multiple eyeblink	11
2.10	Examples and their predicted results	12
2.11	Example of deepfake	12
2.12	Code showing when to use eyeblink detect	13
2.13	Confusion matrix for CNN modal	15
2.14	Showing various CNN modal metrics	15
3.1	Tensorflow	16
3.2	OpenCV	17
3.3	Dlib	17
3.4	Numpy	18
3.5	Moviepy	18
3.6	Matplotlib	18
3.7	React	19
3.8	HTML5	19
3.9	Tailwind	20

3.10	Python	20
3.11	Django	21
3.12	Google Firestore	22
5.1	Choosing File	25
5.2	Showing result	26
5.3	Modal Parameters	27
5.4	EAR graph	28

LIST OF ABBREVIATIONS

GAN: Generative Adversial Networks

EAR: Eye Aspect Ratio

TABLE OF CONTENTS

CANDIDATES' DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
PLAGIARISM REPORT	iv
LIST OF FIGURES	v
LIST OF TABLES	vi
LIST OF ABBREVIATIONS	vii
1. INTRODUCTION	
1. Background of the Problem	1
2. Literature Survey	4
3. Problem Statement	6
4. Motivation	7
5. Feasibility	8
2. PROPOSED SOLUTION	8-16
3. TECHNOLOGY ANALYSIS	17-22
4. ECONOMIC ANALYSIS	23-24
5. RESULT AND DISCUSSION	
1. Usage Instructions	25-28
2. Result	
6. CONCLUSION	29
7. REFERENCES	30

CHAPTER 1

INTRODUCTION

1.1 Background

The proliferation of deepfake videos has emerged as a significant challenge for both individuals and society at large. Deepfakes, created using Generative Adversarial Networks (GANs), can manipulate or fabricate video and audio content to appear convincingly real. This technology is increasingly being used for malicious purposes, including spreading misinformation, creating non-consensual explicit content, and committing fraud. As a result, the need for effective deepfake detection has become critical to ensure the integrity of digital media and protect individuals from potential harm.

The process of detecting deepfakes is complex and requires sophisticated techniques to identify subtle anomalies that distinguish fake content from real. Traditional methods often involve analyzing pixel inconsistencies or other visual artifacts. However, as GANs technology advances, these methods are becoming less effective, necessitating the development of more robust detection mechanisms.

In many regions with underdeveloped technological infrastructure or limited access to advanced detection tools, the identification of deepfakes can be delayed, leading to significant social and personal repercussions. In remote or underserved areas, the lack of resources can further exacerbate the issue, leaving communities vulnerable to the adverse effects of deepfake dissemination. Additionally, many individuals may be unaware of the presence of deepfakes or underestimate their potential impact, resulting in a lack of timely intervention and mitigation.

To address these challenges, we propose an advanced algorithm designed to detect deepfakes by analyzing human eye blinking patterns. Eye blinks are involuntary and spontaneous actions influenced by a variety of factors such as age, gender, cognitive

activities, and time of day. By monitoring these patterns, it can identify anomalies that indicate the presence of deepfakes. This method leverages insights from interdisciplinary research, combining medicine, biology, brain engineering, and machine learning to enhance detection accuracy.

Human eye blinking patterns are known to vary significantly according to an individual's overall physical conditions, cognitive activities, biological factors, and information processing levels. For instance, gender and age, the time of day, or a person's emotional state or degree of alertness can all influence blinking patterns. DeepVision performs integrity verification by tracking significant changes in these patterns, using a heuristic approach based on research results from various fields.

The proposed method involves analyzing the period, frequency, and duration of eye blinks to detect anomalies. The model has demonstrated an overall average accuracy rate of 80% in detecting deepfakes across various video types, suggesting it can overcome the limitations of traditional pixel-based verification algorithms. By incorporating demographic variables and activity types into the analysis, DeepVision enhances its detection accuracy, acknowledging that blinking patterns fluctuate with age, gender, activity engagement, and diurnal cycles.

DeepVision aims to provide a reliable tool for identifying deepfakes, particularly in areas with limited access to advanced technology or expert analysts. By offering an effective solution for early detection, DeepVision can help mitigate the impact of deepfakes and contribute to the overall integrity of digital media. This represents a significant advancement in the fight against digital misinformation and malicious content, improving the security and trustworthiness of media consumed by the public.

Overall, DeepVision is a significant step toward improving access to reliable deepfake detection, especially in areas where there may be a shortage of experts or limited access to technological resources. By leveraging physiological cues and interdisciplinary research,

DeepVision offers a robust framework for enhancing the integrity and security of digital media.

1.2. Literature Survey

The detection of deepfake videos has become increasingly crucial in today's digital age, where advancements in Generative Adversarial Networks (GANs)[1][5] have made it challenging to discern between real and synthetic content. One promising avenue for deepfake detection lies in analyzing human eye blinking patterns, which are influenced by a myriad of factors such as gender, age, activity, and time.[2] These factors significantly impact blinking frequency, with differences observed between adults and infants, as well as variations throughout the day, particularly peaking around nighttime.

Recognizing the intricate relationship between blinking patterns and contextual variables, our proposed approach, it leverages here insights to detect anomalies indicative of deepfakes. The architecture encompasses a systematic process that integrates gender, age, activity, and time data into the analysis of eye blinking. This multi-faceted approach acknowledges the nuanced nature of blinking behavior and its relevance in discerning natural versus artificial eye movements.[2]

The architecture encompasses a pre-process phase where input data, including gender, age, activity type (dynamic or static), and time (A.M. or P.M.), is fed into the system as critical parameters for verifying changes in eye blinks . This pre-processing stage lays the foundation for subsequent measurements conducted by the Target Detector and Eye Tracker components.[2]

The Target Detector module within the model utilizes the dlib face detection algorithm for landmarks localization, pose estimation, and gender recognition. This algorithm's robustness enhances It's ability to accurately detect facial features, setting the stage for precise eye tracking. The Fast-HyperFace algorithm's high detection performance, coupled

with its gender recognition capabilities, contributes significantly to the comprehensive analysis of eye blinking patterns.[2]

Moreover, the integration of the EAR (Eye Aspect Ratio) algorithm[2] within the Eye Tracker component further refines its ability to detect eye blinks. While the Fast-HyperFace algorithm excels in overall facial detection, the EAR algorithm specializes in eye blinking detection. It synergistically leverages the strengths of both algorithms, effectively enhancing the detection performance and accuracy of identifying deepfakes.

In summary, the model represents a pioneering approach to deepfake detection by capitalizing on the nuanced characteristics of human eye blinking patterns. By integrating contextual variables and leveraging state-of-the-art algorithms, it demonstrates a holistic and effective strategy for combating the proliferation of deepfake content in digital media.

1.3. Problem Statement and its Necessity

The inception of this project is driven by several critical issues in the realm of digital media integrity and trust, including:

1. Lack of Robust Deepfake Detection Solutions:

As deepfake technology evolves, traditional methods of detecting synthetic media based solely on visual artifacts become less effective. There is a pressing need for advanced and robust deepfake detection mechanisms that can accurately identify manipulated content.

2. Mitigating Social Misinformation and Manipulation:

The proliferation of deepfake content poses a significant risk to public trust and can be exploited for spreading misinformation, propaganda, and fake news. Detecting and flagging deepfakes is essential for safeguarding the authenticity and credibility of digital media.

3. Preserving Authenticity in Digital Content:

Deepfake technology can be misused to create forged content that can harm individuals, tarnish reputations, and manipulate public perception. Ensuring the authenticity and integrity of digital content is paramount for upholding ethical standards and trust in digital media platforms.

4. Protecting Against Malicious Use of Deepfakes:

Deepfakes have the potential to be used for malicious purposes, such as creating fake videos for extortion, defamation, or impersonation. Developing robust deepfake detection tools is crucial for mitigating these risks and protecting individuals from harm.

5. Empowering Content Creators and Consumers:

Content creators and consumers need reliable tools to verify the authenticity of media content and distinguish between genuine and manipulated material. Providing accessible and accurate deepfake detection solutions empowers users to make informed decisions and maintain trust in digital communications.

6. Addressing Ethical and Legal Implications:

The rise of deepfake technology raises complex ethical and legal questions regarding privacy, consent, and the manipulation of digital media. Developing effective deepfake detection systems contributes to addressing these concerns and promoting responsible use of technology.

1.4. Motivation

- The motivation behind this deepfake detection project stems from the increasing prevalence and sophistication of deepfake technology, which poses significant threats to digital media integrity and trust.
- As deepfake techniques become more accessible and advanced, there is a growing concern about their potential misuse for spreading misinformation, manipulating

public opinion, and causing harm to individuals and organizations. The need for robust deepfake detection solutions is paramount to mitigate these risks and preserve the authenticity and credibility of digital content.

- Furthermore, the rapid evolution of deepfake technology underscores the urgency of developing proactive measures to detect and counteract its malicious use. By leveraging cutting-edge deep learning algorithms and innovative approaches, this project aims to empower content creators, consumers, and digital platforms with effective tools to identify and mitigate the impact of deepfake content.
- Our goal is to provide accessible, affordable, and user-friendly deepfake detection solutions that can be deployed across various digital platforms and settings. By enhancing media integrity and trust, we aim to safeguard digital ecosystems and uphold ethical standards in digital communications.

1.5. Feasibility: Non-Technical and Technical

As with any successful project, it is very crucial to have a plan of whether a project is feasible from different standpoints. The various possibilities/standpoints can be summarized as follows.

TECHNICAL:

With the availability of powerful high-level programming languages such as Python, along with comprehensive support for Machine Learning algorithms.

SOCIAL:

As of now, there is a noticeable gap in the availability of widely adopted solutions specifically targeting deepfake detection. This project addresses a critical need in the digital media landscape, making it socially relevant and potentially impactful in combating misinformation and preserving media integrity.

ECONOMICAL FEASIBILITY:

The project's economic feasibility is high due to the utilization of open-source libraries, publicly available datasets, and cloud-based computing resources. These factors significantly reduce development costs, making the project financially viable even for individuals or organizations with limited resources.

SCOPE:

The scope of this project extends to both professional users, such as media organizations and content creators, as well as general consumers concerned about the authenticity of digital content. By offering a reliable and accessible deepfake detection solution, the project aims to enhance trust in digital media platforms and empower users to make informed decisions regarding the content they consume and share

1.6 Research Objectives

- Deepfake Detection project aims to revolutionize deepfake detection by harnessing advanced deep learning technologies, with a specific focus on improving accessibility for users in regions with limited resources for combating digital manipulation.
- By implementing a specialized algorithm focused on analyzing eye blinking patterns, the project aims to achieve high accuracy and reliability in detecting deepfake videos. This innovative approach enhances the integrity verification process, contributing to more effective detection of manipulated videos.

CHAPTER 2

PROPOSED SOLUTION

Eye Blink Detection for Deepfake Detection

Introduction:

Deepfake videos often lack natural human behavior, such as blinking. Our proposed solution aims to leverage this by detecting and analyzing eye blinks in videos to identify potential deepfakes. Blinking is a spontaneous and involuntary action, and its patterns can reveal anomalies that may indicate manipulated content.

Human blinking patterns are influenced by various factors, including age, gender, cognitive activities, and physiological conditions. These factors contribute to the unique and unpredictable nature of blinking, making it challenging for deepfake algorithms to accurately replicate natural blinking patterns.

TYPE OF INPUT DATA IN PRE-PROCESS	
Category	Input
Gender	Male, Female
Age	<20, 20-30, 30-40, 40-50, 50-60, 65+
Activity	Dynamic / Static
Time	A.M / P.M

Fig 2.1 Factors which affects eye blinks

By analyzing the frequency, duration, and timing of eye blinks in a video, our solution can detect anomalies that deviate from typical human blinking behavior. These anomalies may be indicative of deepfake content, where the blinking patterns have been artificially generated or manipulated.

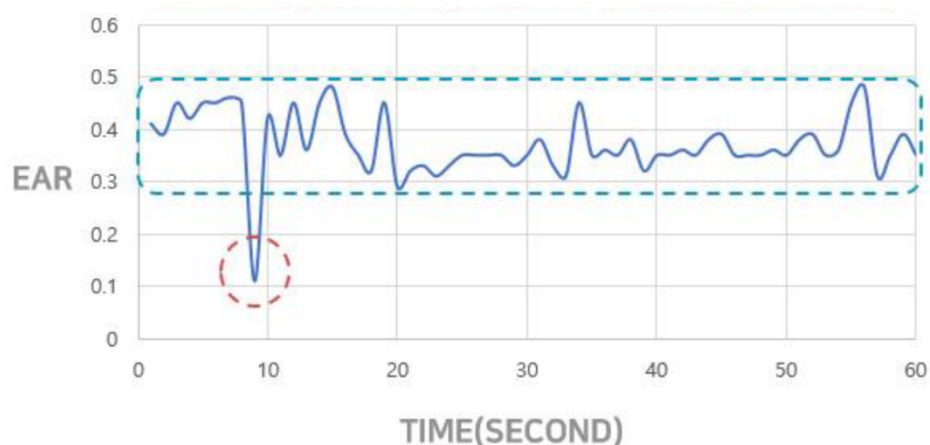


Fig 2.2 Example of artificial generated(deepfake) video as there is only one eyeblink in a minute. Additionally, the proposed solution takes into account the context of the video, such as the subject's age, gender, and activity (e.g., reading, conversing, or performing physical tasks). This contextual information allows for more accurate comparisons with expected blinking patterns, further enhancing the detection capabilities.

The eye blink detection approach offers a non-intrusive and computationally efficient method for deepfake detection, as it does not require extensive training data or resource-intensive deep learning models. Instead, it relies on well-established computer vision techniques and statistical analysis of blinking patterns.

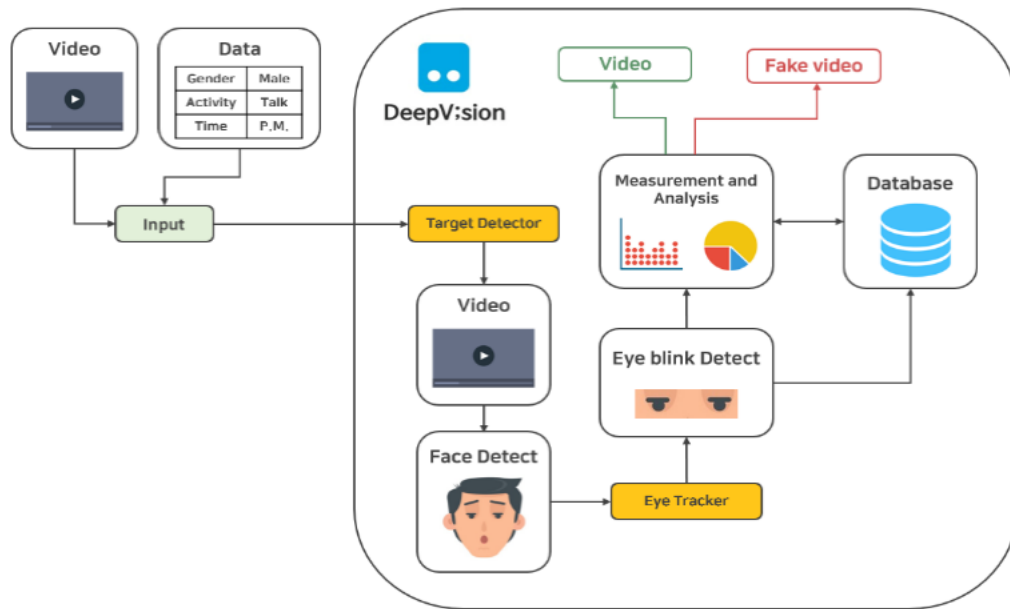


Fig 2.3 Architecture of proposed system[2]

Methodology:

1. Eye Blink Detection

- Utilize the Facial Landmark Detection algorithm to locate facial features, including eyes.
- Calculate the Eye Aspect Ratio (EAR) to determine blink occurrences. A significant decrease in EAR indicates a blink.

2. Blink Rate Calculation

- Count the total number of blinks over the duration of the video.

- Compute the blink rate (blinks per minute) to assess the naturalness of the blinking pattern.

Implementation Steps:

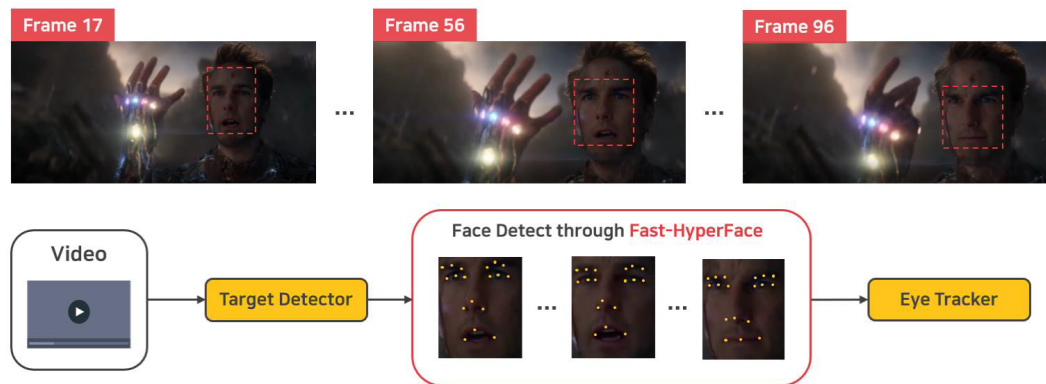


Fig 2.4 Detecting Face using Fast-HyperFace[2]

1. Facial Landmark Detection

- Employ the dlib library for facial landmark detection.
- Identify key facial landmarks, including eye corners and eyelids.

```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

Fig 2.5 Code for finding facial landmarks

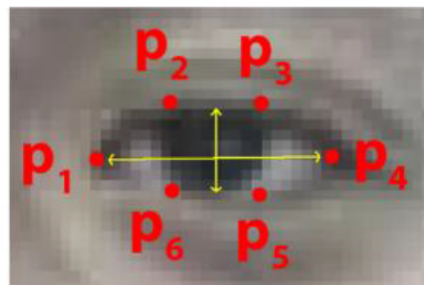


Fig 2.6 Showing eyelid landmarks

2. EAR Calculation

- Calculate the EAR using the formula, where p1-p6 are specific landmarks.

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2 ||p_1 - p_4||}$$

- Detect blinks based on predefined EAR thresholds.

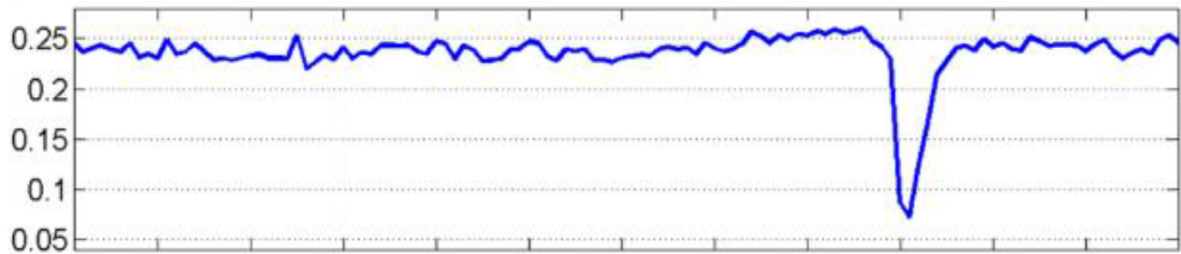


Fig 2.7 EAR graph overtime

3. Blink Rate Analysis

- Count the number of blinks detected throughout the video.

```
if ear < cfg.EYE_AR_THRESH:
    COUNTER += 1
```

Fig 2.8 Thresholding code for eyeblink

- Calculate the blink rate to determine the frequency of blinks.

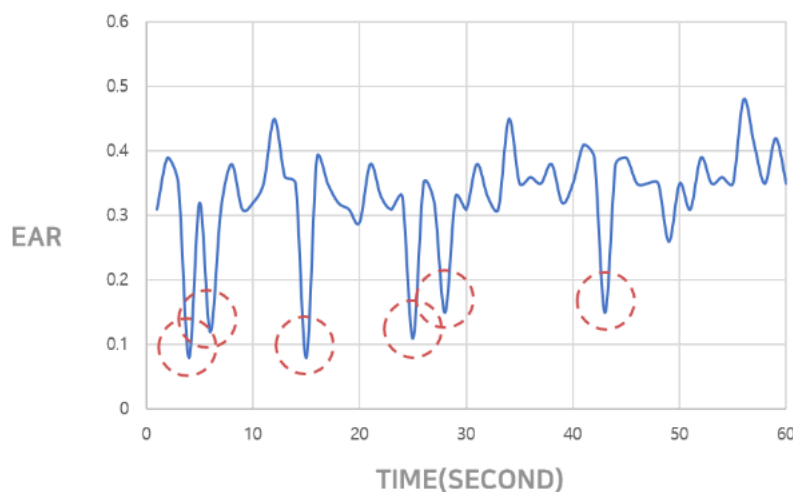


Fig 2.9 EAR graph showing multiple eyeblinks

Case	Data of Measured Eye Blinks			Prediction
	Repeated Number	Period	Elapsed Time	
1	-/min	-/sec	-/sec	Fake
2	1/min	-/sec	0.1135/sec	Fake
3	6/min	6.3333/sec (average)	0.1321/sec (average)	Fake

Fig 2.10 Some example cases and their predicted results

4. Classification

Once the eye blinks have been detected and the blink rate calculated, the next critical step is to classify the video based on the observed blink patterns. This classification process involves determining whether the video is likely to be real or a deepfake based on established thresholds and patterns of natural human behavior.

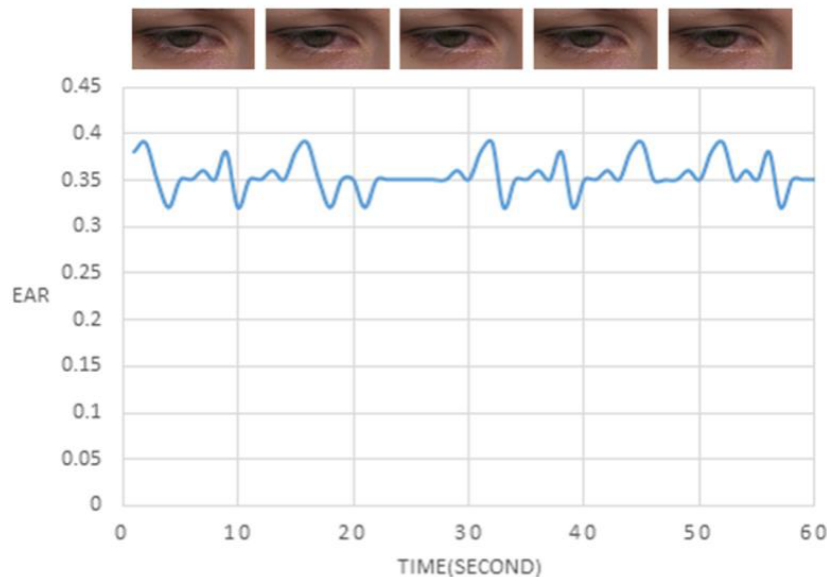


Fig 2.11 Definitely an Deepfake there are no blink in a minute

5. Limitations

However, we found some limitations in the experiment. The number of eye blinks was correlated with a mental illness closely connected with dopamine activity. The study

results revealed that the number (27 times/min) of blinking in patients with schizophrenia was considerably higher than that of normal people (17 times/min), and that their blinking count was uninfluenced by the medicine administered. To overcome to this, we can shift to CNN Modal to check the deepfake possibility.

Use Case:

- Long Videos which are greater than or equal to 15 sec in length.

```
if video_duration < 15:
    video_frames = extract_frames_from_video(video_content)
    result_label = process_video(video_frames)
    print(result_label)
    return JsonResponse({'result': result_label})
else:
    total_blinks, blinks_per_minute, ear_values = detect_blinks(temp_filename)
    os.remove(temp_filename)
    print(total_blinks, blinks_per_minute)
```

Fig 2.12 Code showing when to use eye blink detection and when to use CNNs

Deep Learning-based Image and Video Analysis

Introduction:

Deepfake detection is a challenging task due to the rapid advancement of generative adversarial networks (GANs) and other deep learning techniques used to create realistic manipulated content. Our deep learning-based solution utilizes state-of-the-art convolutional neural networks (CNNs) to analyze images and video frames for accurate deepfake detection.

One of the key advantages of this approach is its ability to learn and extract complex features from the data itself, rather than relying on handcrafted features or specific artifacts. By training on a diverse dataset of authentic and manipulated videos, the CNN model can learn to distinguish subtle patterns and anomalies that may be imperceptible to human observers.

Furthermore, the deep learning-based solution can leverage transfer learning techniques, where pre-trained models on large-scale datasets are fine-tuned on the deepfake detection task. This approach can significantly improve training efficiency and model performance, especially when dealing with limited data.

The proposed solution is designed to be highly scalable and adaptable, capable of handling various types of deepfake manipulations, including face swapping, lip-syncing, and synthetic video generation. As new deepfake techniques emerge, the model can be retrained on updated datasets, ensuring its continued effectiveness.

Moreover, by analyzing individual video frames and aggregating the results, our solution can provide frame-level and video-level classifications, allowing for detailed analysis and localization of manipulated regions within the video.

Methodology:

The proposed deep learning-based solution for deepfake detection leverages the power of convolutional neural networks (CNNs) to analyze images and video frames. The methodology consists of the following key steps:

1. **Data Collection and Preprocessing:** A diverse dataset comprising both authentic and manipulated videos will be collected. The videos will undergo preprocessing steps, including resizing, cropping, and pixel value normalization, to standardize the input data for the model.
2. **Model Training:** A CNN model will be trained from scratch on the preprocessed dataset to classify videos as either real or fake. The model architecture will be carefully designed, incorporating convolutional layers for feature extraction, pooling layers for downsampling, and dense layers for classification.
3. **Prediction:** Individual frames from the video will be analyzed using the trained CNN model to obtain frame-level predictions (real or fake). These frame-level predictions will then be aggregated using techniques such as majority voting or temporal smoothing to determine the overall classification of the video as either real or manipulated.

Implementation Steps:

The proposed solution will be implemented following these steps:

1. **Data Collection and Preprocessing:**
 - We gathered a diverse dataset comprising both authentic and manipulated videos.
 - The collected videos underwent preprocessing steps, including resizing, cropping, and normalization of pixel values.
 - We split the dataset into training, validation, and testing sets to facilitate model training and evaluation.

- Data augmentation techniques, such as rotation, flipping, and shifting, were applied to the training data to increase diversity and improve model generalization.

2. Model Architecture:

- We designed a CNN model architecture tailored specifically for image classification tasks.
- The model incorporates convolutional layers, pooling layers, and dense layers to enable effective feature extraction and classification.

3. Model Training:

- We compiled the CNN model with an appropriate loss function (e.g., binary cross-entropy) and optimization algorithm (e.g., Adam, SGD).
- Training commenced from scratch on the training set, with constant monitoring of performance using the validation set to prevent overfitting.
- Techniques such as early stopping and model checkpointing were implemented to ensure the preservation of the best-performing model during training.

4. Evaluation:

- We evaluated the trained model's performance on the previously unseen test set, which was not utilized during training.
- Various performance metrics, including accuracy, precision, recall, and F1-score, were computed to gauge the model's effectiveness in detecting deepfakes.
- Misclassified examples and potential failure cases were meticulously analyzed to pinpoint areas for improvement.

```
Confusion Matrix
[[5058  434]
 [1062 4351]]
```

Fig 2.13 Confusion Matrix for Modal

```

Classification Report - Model 2:
              precision    recall  f1-score   support

    Fake       0.83       0.92       0.87       5492
    Real       0.91       0.80       0.85       5413

 accuracy              0.86       10905
 macro avg       0.87       0.86       0.86       10905
weighted avg       0.87       0.86       0.86       10905

```

Fig 2.14 Showing various performance metrics

5. Prediction:

- Upon receiving new video inputs, we subjected the video frames to the same preprocessing steps as the training data.
- Each video frame underwent prediction by passing through the trained model to obtain frame-level predictions (real or fake).
- Techniques such as majority voting or temporal smoothing were employed to aggregate frame-level predictions, thus determining the overall video classification.

CHAPTER 3

TECHNOLOGY ANALYSIS

Tech stack used in project consist of following technologies :

3.1 Deep Learning Framework

3.1.1 TensorFlow (Keras)

TensorFlow is a powerful and widely-used open-source deep learning framework developed by Google. It provides a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers and developers build and deploy machine learning-powered applications.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. Keras allows for easy and fast prototyping, supports both convolutional networks and recurrent networks, and runs seamlessly on both CPU and GPU.

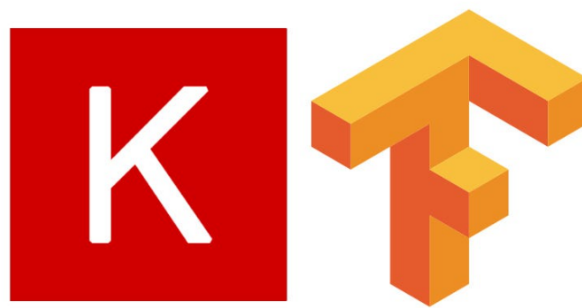


Figure 3.1 : TensorFlow

In this Deepfake detection project, TensorFlow (with Keras) is used for designing and training deep learning models. Its extensive support for neural network operations and easy-to-use syntax makes it an ideal choice for implementing complex architectures required for detecting deepfakes.

3.2 Model Architecture

3.2.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep neural networks, most commonly applied to analyzing visual imagery. They are particularly effective for tasks involving image and video data due to their ability to capture spatial hierarchies in the data.

In this project, CNNs are used to detect deepfakes by learning and identifying patterns and features that distinguish real images or videos from manipulated ones. Their capability to automatically learn spatial hierarchies from the input data is critical for accurately identifying subtle artifacts introduced during the creation of deepfakes.

3.3 Libraries and Tools

3.3.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It contains a comprehensive set of tools for image processing and computer vision tasks.

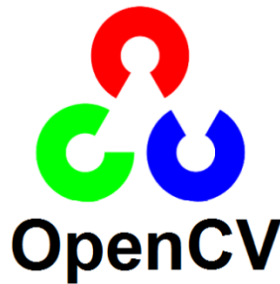


Figure 3.2 OpenCV

In this project, OpenCV is utilized for pre-processing the video frames, such as resizing, color space conversion, and frame extraction, which are essential steps before feeding the data into the neural network for deepfake detection.

3.3.2 dlib

dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software. It includes a wide range of functionality for face detection, object detection, and more.



Figure 3.3 : dlib

dlib is used in this project for face detection and facial landmark extraction, which are critical preprocessing steps for ensuring that the regions of interest (faces) are accurately analyzed for deepfake detection.

3.3.3 NumPy

NumPy is a fundamental package for scientific computing with Python. It provides support for arrays, matrices, and many mathematical functions.



Figure 3.4 : NumPy

NumPy is used for handling numerical data and performing operations on arrays, which is essential for manipulating the image data and performing mathematical operations required during the data preprocessing and model evaluation phases.

3.3.4 moviepy

moviepy is a Python library for video editing, which allows for basic operations on videos, such as cutting, concatenating, and processing.



Figure 3.5 : MoviePy

In this project, moviepy is used for handling video files, including extracting frames from videos, which are then processed and analyzed by the deep learning models.

3.3.5 matplotlib

matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It is used for creating static, animated, and interactive visualizations.



Figure 3.6 : matplotlib

matplotlib is employed in this project for visualizing the results of the deep learning models, such as plotting training and validation accuracies, losses, and displaying example frames of detected deepfakes.

3.4 Frontend Technologies

3.4.1 React

React is a JavaScript library for building user interfaces, particularly single-page applications where data changes over time. Developed and maintained by Facebook, it enables developers to create large web applications that can update and render efficiently in response to data changes.

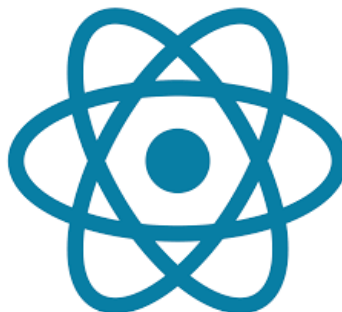


Figure 3.7 : React

React is used in this project to build the frontend interface, providing a responsive and interactive user experience for users to upload videos, receive analysis results, and visualize the detection process.

3.4.2 HTML

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages. It provides the structure for web content.



Figure 3.8 : HTML5

HTML is utilized in conjunction with React to define the structure of the web pages, ensuring that the content is semantically organized and accessible.

3.4.3 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that provides a low-level, utility-based approach to styling web pages. It allows developers to build custom designs without leaving their HTML.



Tailwind CSS

Figure 3.9 : Tailwind

Tailwind CSS is used in this project for styling the frontend interface, providing a clean, modern, and responsive design with minimal effort.

3.5 Backend Technologies

3.5.1 Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used in web development, data science, artificial intelligence, and more.



Figure 3.10 : Python

Python is the primary language used for implementing the backend of this project, leveraging its extensive libraries and frameworks to handle data processing, model training, and server-side logic.

3.5.2 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It provides a robust and scalable foundation for building web applications.



Figure 3.11 : django

Django is used in this project to develop the backend server, manage API endpoints, handle user requests, and interface with the machine learning models for processing and returning the deepfake detection results.

3.6 Database

3.6.1 Google Firestore

Google Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It provides real-time data synchronization and supports offline mode.



Figure 3.12 : Google Firestore

In this project, Google Firestore is used to store and manage data such as uploaded images, uploaded videos, and analysis results. Its real-time capabilities ensure that users receive immediate feedback and updates on the status of their deepfake detection requests.

CHAPTER 4

ECONOMIC ANALYSIS

Our Deepfake detection project has been developed with a focus on economic efficiency and accessibility. By leveraging a suite of free and secure technologies, we have ensured that the project incurs minimal costs while maintaining high standards of functionality and performance. This strategic choice allows us to deliver a robust solution without burdening users with additional financial expenses.

1. Free and Open-Source Technologies:

We have utilized a variety of free and open-source software in our development process, including TensorFlow (Keras) for deep learning, OpenCV for image processing, and dlib for facial recognition. These technologies are not only cost-effective but also come with extensive community support and continuous updates, ensuring that our application remains up-to-date with the latest advancements without incurring additional costs.

2. Development Stacks and Tools:

The development stacks and tools employed in this project, such as NumPy, moviepy, and matplotlib, are freely available and widely used in the machine learning and data science communities. This significantly reduces the overall cost of development and maintenance.

3. Frontend and Backend Technologies:

For the frontend, we have chosen React, HTML, and Tailwind CSS—all of which are free to use. React provides a powerful framework for building dynamic user interfaces, while Tailwind CSS offers a modern approach to styling, all without any licensing fees. On the backend, Python and Django form the core of our application, both of which are open-source and free.

4. Database Management:

Our project uses Google Firestore for data storage and management, which also offers a free tier with sufficient capacity for our application's requirements. Firestore's real-time data synchronization capabilities enhance user experience without adding costs, as the free tier covers the basic needs of our project.

5. Overall Cost Efficiency:

By strategically selecting these free and open-source technologies, we have minimized the economic barriers associated with developing our Deepfake detection application. This ensures that our solution is affordable and accessible to a broad audience, including individuals and organizations that may have limited budgets. The absence of licensing fees and reliance on free tiers of cloud services contribute to a highly cost-effective solution.

6. Long-term Sustainability:

The sustainability of our project is further enhanced by the active communities supporting the open-source technologies we use. Continuous improvements and updates from these communities ensure that our project remains secure, efficient, and cutting-edge, without the need for significant financial investment in proprietary software or tools.

CHAPTER 5

RESULT AND DISCUSSION

5.1. Instructions of Usage

Hardware and software requirements: The frontend website is designed to be user-friendly and accessible, requiring only basic hardware to function effectively. To ensure smooth operation, users need a minimum of a dual-core processor, such as an Intel Core i3 or equivalent, while a quad-core processor like an Intel Core i5 or equivalent is recommended for better performance. Additionally, the system should have at least 4 GB of RAM, with 8 GB being preferable for handling video frames more efficiently.

Stable internet connectivity is crucial for uploading and downloading videos seamlessly. A minimum internet speed of 5 Mbps is required, but a high-speed connection of at least 20 Mbps is recommended for optimal performance.

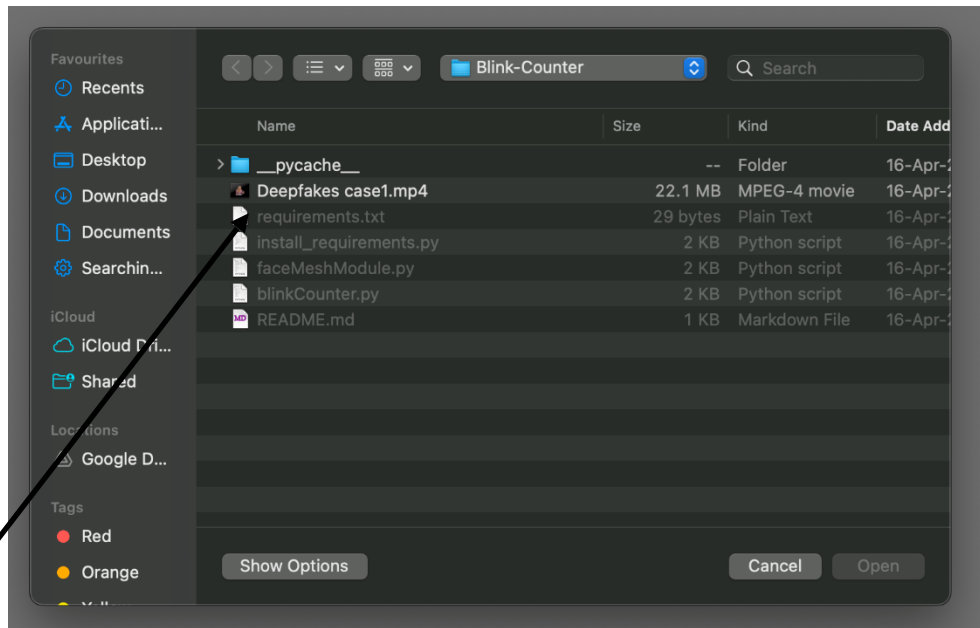
The website is compatible with various operating systems, including Windows 10 or later, macOS, and Linux distributions with kernel version 4.15 or later. To maintain computational efficiency, the website uses pre-trained weights to make predictions, which helps reduce processing time. The code's get function is optimized to sequentially take frames until the video is complete, minimizing lag and enhancing the user experience. Users are advised to use devices that meet or exceed the recommended specifications to ensure smooth operation, especially when processing longer videos.

With these basic hardware requirements, users can effectively utilize the frontend website for Deepfake detection. Meeting or exceeding the recommended specifications will result in improved performance and a smoother user experience.

Steps for Deepfake detection:



Fig 5.1 Choosing a file



Choose the mp4/jpg file for input

Check the input by clicking on the given button

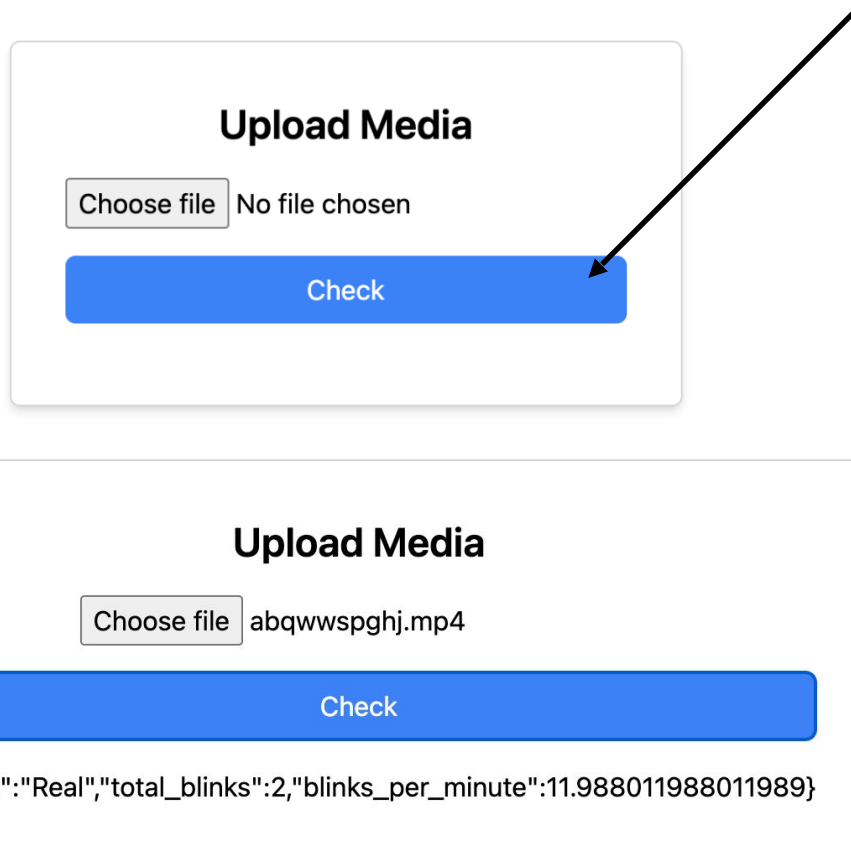


Fig 5.2 Showing result

Result:

CNN Model:

Trained on a extensive dataset, expected to exhibit improved generalization and performance on unseen data.

```
Confusion Matrix
[[5058  434]
 [1062 4351]]
```

```
Model (Larger Dataset) Evaluation:
Loss: 0.3420152962207794
Accuracy: 0.8628152012825012
```

Classification Report				
	precision	recall	f1-score	support
Fake	0.83	0.92	0.87	5492
Real	0.91	0.80	0.85	5413
accuracy			0.86	10905
macro avg	0.87	0.86	0.86	10905
weighted avg	0.87	0.86	0.86	10905

Fig 5.3 Modal Parameters

Definition of some terms:

- Precision: The ratio of true positive predictions to the total number of positive predictions made (i.e., true positives divided by the sum of true and false positives).
- Recall: The ratio of true positive predictions to the total number of actual positive instances (i.e., true positives divided by the sum of true positives and false negatives).
- F1 Score: The harmonic mean of precision and recall, providing a single metric that balances both concerns.

- Support: The number of actual occurrences of each class in the dataset (i.e., the true positive cases for each class).

Eye - Blink Detection:

The proposed algorithm consistently showed a significant possibility of verifying the integrity of Deepfakes and normal videos, accurately detecting Deepfakes in six out of eight videos (75%) for GAN developed datasets (FaceForensics++, OpenForensics)

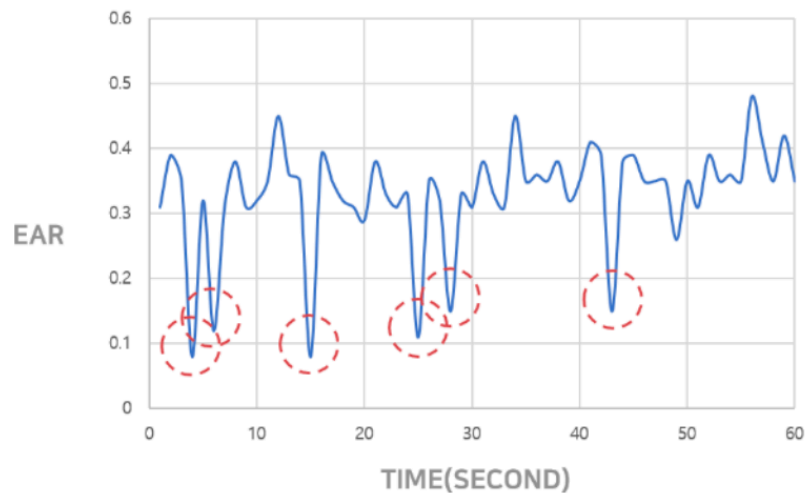


Fig 5.4 EAR graph

CHAPTER 6

CONCLUSION

In this study, we proposed and developed a method to analyze significant changes in eye blinking, which is a spontaneous and unconscious human function, as an approach to detect Deepfakes generated using the GANs model. Blinking patterns vary according to an individual's gender, age, and cognitive behavior, and fluctuate based on the time of day. Thus, the proposed algorithm observed these changes using machine learning, several algorithms, and a heuristic method to verify the integrity of Deepfakes.

To enhance the effectiveness of our method, we implemented different approaches based on the length of the videos. For videos greater than 15 seconds, we utilized a comprehensive algorithm that incorporated results from various previous studies. For videos less than 15 seconds, we employed a CNN-based weighted model to ensure accurate detection. Additionally, we integrated the detection of both images and videos in our algorithm.

The proposed algorithm consistently showed a significant possibility of verifying the integrity of Deepfakes and normal videos, accurately detecting Deepfakes in six out of eight videos averaging to (80%). However, a limitation of the study is that blinking is also correlated with mental illness and dopamine activity. The integrity verification may not be applicable to people with mental illnesses or problems in nerve conduction pathways.

Despite this limitation, our method can be improved through a number of measures as cybersecurity attack and defense evolve continuously. The proposed algorithm suggests a new direction that can overcome the limitations of integrity verification algorithms performed only on the basis of pixels.

REFERENCES

- [1] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies and M.Niener., "FaceForensics++: Learning to Detect Manipulated Facial Images", Jan. 2019. [Online] Available: <https://arxiv.org/abs/1901.08971>[9]
- [2] DeepVision: Deepfakes detection using human eye blinking pattern TackHyun Jung¹ , SangWon Kim², and KeeCheon Kim³
- [3] S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back, "Face recognition: A convolutional neural-network approach", IEEE Trans. Neural Netw., vol. 8, no. 1, pp. 98-113, Jan. 1997
- [4] Kaggle, "Eye Blinking Prediction", CompOmics 2018 summercompetition. [Online] Available: <https://www.kaggle.com/c/compomicssummer2018/data>
- [5] I. Goodfellow et al., "Generative adversarial nets", Proc. Adv. Neural Inf. Process. Syst., pp. 2672-2680, 2014.
- [6] Li, Y., Chang, M. C., and Lyu, S., "In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking", Jun. 2018. [Online] Available: <https://arxiv.org/abs/1806.02877>
- [7] <https://github.com/takhyun12/Dataset-of-Deepfakes>
- [8] <https://zenodo.org/records/5528418#.YpdlS2hBzDd> (OpenForensics Dataset)