JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

MANIPULATION OF IMAGES AND VIDEOS USING DEEPFAKE REALMS

B. RAKESH SAGAR
K SAJITHA

Under the Guidance of

Dr. M. RAMA BAI
Professor & HoD

Dept. of Emerging Technologies
MAHATMA GANDHI INSTITUTE OF TECHNOLOGY (Autonomous)
(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)
GANDIPET, HYDERABAD – 500075, TELANGANA, INDIA
2023-2024

ABSTRACT

This project delves into the sophisticated manipulation of images and videos using deepfake realms, with a focus on both creation and detection of highly realistic talking head models. Our primary objective is to develop a system capable of generating authentic video sequences that replicate the speech expressions and facial movements of specific individuals with high fidelity. Utilizing "few-shot learning," our approach allows the system to produce convincing results using only a handful of photographs, significantly reducing the required training time. The system can generate plausible video sequences from a single photograph, with additional photographs enhancing the personalization and accuracy of the output. By employing advanced meta-learning techniques on an extensive collection of video data, our model generalizes and adapts to new, previously unseen individuals, framing the task as an adversarial training problem with high-capacity generators and discriminators. This enables efficient training with millions of parameters fine-tuned for each individual's unique facial features and expressions.

In addition to the creation of deepfake content, our project places a significant emphasis on detection and mitigation. We develop robust detection algorithms capable of identifying synthetic content, addressing the ethical and security concerns associated with deepfake technology. These algorithms analyze various aspects of video sequences, including inconsistencies in facial movements, lighting, and other subtle cues that may indicate manipulation. Our comprehensive approach not only showcases the potential of deepfake technology for image and video manipulation but also highlights the critical importance of

developing safeguards to prevent misuse. By balancing the innovative capabilities of deepfake creation with advanced detection mechanisms, we aim to contribute to a more secure and ethically responsible use of this powerful technology.

1. INTRODUCTION

In a narrow definition, deepfakes (stemming from "deep learning" and "fake") are created by techniques that can superimpose face images of a target person onto a video of a source person to make a video of the target person doing or saying things the source person does. This constitutes a category of deepfakes, namely faceswap. In a broader definition, deepfakes are artificial intelligence-synthesized content that can also fall into two other categories, i.e., lip-sync and puppet-master. Lip-sync deepfakes refer to videos that are modified to make the mouth movements consistent with an audio recording. Puppet-master deepfakes include videos of a target person (puppet) who is animated following the facial expressions, eye and head movements of another person (master) sitting in front of a camera . While some deepfakes can be created by traditional visual effects or computer-graphics approaches, the recent common underlying mechanism for deepfake creation is deep learning models such as autoencoders and generative adversarial networks (GANs), which have been applied widely in the computer vision domain . These models are used to examine facial expressions and movements of a person and synthesize facial images of another person making analogous expressions and movements . Deepfake methods normally require a large amount of image and video data to train models to create photo-realistic images and videos. As public figures such as celebrities and politicians may have a large number of videos and images available online, they are initial targets of deepfakes. Deepfakes were used to swap faces of celebrities or politicians to bodies in porn images and videos. The first deepfake video emerged in 2017 where face of a celebrity was swapped to the face of a porn actor. It is threatening to world security when deepfake methods can be employed to create videos of world leaders with fake speeches for falsification purposes . Deepfakes therefore can be abused to cause political or religion tensions between countries, to fool public and affect results in election campaigns, or create chaos in financial markets by creating fake news . It can be even used to generate fake satellite images of the Earth to contain objects that do not really exist to confuse military analysts, e.g., creating a fake bridge across a river although there is no such a bridge in reality. This can mislead a troop who have been guided to cross the bridge in a battle . As the democratization of creating realistic digital humans has positive implications, there is also positive use of deepfakes such as their applications in visual effects, digital avatars, snapchat filters, creating voices of those who have lost theirs or updating episodes of movies without reshooting them . Deepfakes can have creative or productive impacts in photography, video games, virtual reality, movie productions, and entertainment, e.g., realistic video dubbing of foreign films, education through the reanimation

of historical figures, virtually trying on clothes while shopping, and so on . However, the number of malicious uses of deepfakes largely dominates that of the positive ones. The development of advanced deep neural networks and the availability of large amount of data have made the forged images and videos almost indistinguishable to humans and even to sophisticated computer algorithms. The process of creating those manipulated images and videos is also much simpler today as it needs as little as an identity photo or a short video of a target individual. Less and less effort is required to produce a stunningly convincing tempered footage. Recent advances can even create a deepfake with just a still image . Deepfakes therefore can be a threat affecting not only public figures but also ordinary people. For example, a voice deepfake was used to scam a CEO out of $243,000. A recent release of a software called DeepNude shows more disturbing threats as it can transform a person to a non-consensual porn . Likewise, the Chinese app Zao has gone viral lately as less-skilled users can swap their faces onto bodies of movie stars and insert themselves into well-known movies and TV clips . These forms of falsification create a huge threat to violation of privacy and identity, and affect many aspects of human lives. Finding the truth in digital domain therefore has become increasingly critical. It is even more challenging when dealing with deepfakes as they are majorly used to serve malicious purposes and almost anyone can create deepfakes these days using existing deepfake tools. Thus far, there have been numerous methods proposed to detect deepfakes . Most of them are based on deep learning, and thus a battle between malicious and positive uses of deep learning methods has been arising.

To address the threat of face-swapping technology or deepfakes, the United States Defense Advanced Research Projects Agency (DARPA) initiated a research scheme in media forensics (named Media Forensics or MediFor) to accelerate the development of fake digital visual media detection methods [30]. Recently, Facebook Inc. teaming up with Microsoft Corp and the Partnership on AI coalition have launched the Deepfake Detection Challenge to catalyse more research and development in detecting and preventing deepfakes from being used to mislead viewers . Data obtained from https://app.dimensions.ai at the end of 2021 show that the number of deepfake papers has increased significantly in recent years . Although the obtained numbers of deepfake papers may be lower than actual numbers but the research trend of this topic is obviously increasing. There have been existing survey papers about creating and detecting deepfakes, presented in . For example, Mirsky and Lee focused on reenactment approaches (i.e., to change a target's expression, mouth, pose, gaze or body), and replacement approaches (i.e., to replace a target's face by swap or transfer methods). Verdoliva separated detection approaches into conventional methods (e.g., blind methods without using any external data for training, one-class sensor-based and model-based methods, and supervised methods with handcrafted features) and deep learning-based approaches (e.g., CNN models). Tolosana et al. categorized both creation and detection methods based on the way deepfakes are created, including entire face synthesis, identity swap, attribute manipulation, and expression swap. On the other hand, we carry out the survey with a different perspective and taxonomy. We categorize the deepfake detection methods based on the data type, i.e., images or videos

With fake image detection methods, we focus on the features that are used, i.e., whether they are handcrafted features or deep features. With fake video detection methods, two main subcategories are identified based on whether the method uses temporal features across frames or visual artifacts within a video frame. We also discuss extensively the challenges, research trends and directions on deepfake detection and multimedia forensics problems.

1.1 About the Project

The project for manipulating videos and images using deepfaking technology delves into the cutting-edge realm of artificial intelligence and computer vision. Deepfakes, a fusion of "deep learning" and "fake," leverage sophisticated neural network architectures to seamlessly swap faces, voices, and even entire contexts within multimedia content.

1.1.1 Technology

The technologies used in creating a face-swapping model with SimSwap include:

- **Deep Learning:** Neural networks, especially Generative Adversarial Networks (GANs), are central to SimSwap for generating realistic images or videos.
- **Facial Detection and Recognition:** Algorithms locate and extract facial regions, helping identify individuals for accurate face swapping.
- **Computer Vision:** Techniques like facial landmark detection and optical flow estimation track movements and align features between source and target images.
- **Image Processing:** Techniques preprocess images and apply transformations for realistic swapping.
- **Hardware Acceleration:** GPUs or TPUs speed up computation for deep learning tasks.
- **Libraries and Frameworks:** Tools like PyTorch, OpenCV, and specialized libraries aid in model training and deployment.

- **Python 3.12.1:** A versatile programming language used for web development, data science, and machine learning.
- **Flask:** A lightweight web framework in Python known for its simplicity and flexibility in building web applications.
- **Visual Studio Code (VS Code):** A source-code editor supporting various programming languages with features like debugging and syntax highlighting.
- **MySQL:** A relational database management system commonly used for data storage and retrieval in web applications.
- **NumPy:** A Python library for working with arrays and matrices, widely used in scientific computing and data analysis.
- **TensorFlow:** An open-source software library for machine learning and artificial intelligence, particularly focused on deep neural networks.
- **OpenCV:** A library for real-time computer vision tasks, providing tools for image and video processing.
- **Scikit-learn (Sklearn):** A library containing efficient tools for machine learning and statistical modeling, including classification and regression algorithms.

1.2 Existing System

Realistic talking head sequence synthesis is notoriously challenging for two reasons. First, the photometric, geometric, and kinematic complexity of human skulls is quite great. This complexity results from modelling the lips, hair, and clothing in addition to modelling faces, for which there are several modelling techniques. The second challenging element is how sensitive the human visual system is to even seemingly insignificant errors in how human heads are modelled (the so-called "uncanny valley effect"). The present predominance of non photorealistic cartoon-like avatars in many actually deployed teleconferencing systems is explained by such a low tolerance for modelling errors.

Deepfake, the ability to automatically replace a person's face in photos and videos using deep learning algorithms, is one of the scariest modern occurrences. The creation of false-face detection algorithms for media is essential. The majority of current techniques surround a face with bounding boxes and assign a fakeness probability to each one. It could be more useful to describe why a model forecasts a certain face as real or fake, such as which features the model thinks are fabricated and where it is looking to produce this prediction.

Problem Definition

Several recent studies have demonstrated how to train convolutional neural networks to generate remarkably realistic human head images. These efforts necessitate training on a large collection of photos of a single person in order to construct a personalised talking head model. However, in many real-life situations, In other cases, such personalised talking head models must be learned from a few image views of a person or simply a single image. A system with such a limited number of shots. It does extensive meta-learning on a big video dataset and then can frame the few and one-shot learning of neural talking head models of previously unknown persons as adversarial training problems with high-capacity generators and discriminators. With the advancement of computer vision and deep learning, there has been an increase in the number of realistic-looking fake face media edited by AI, such as Deepfake, that manipulate facial identities or expressions. The artificial faces were largely made for entertainment purposes, but abuse has generated societal turmoil. It is critical to build models that recognise false faces in the media in order to protect individual privacy as well as social, political, and international security. A convolutional neural network-based integrated face forensics framework that combines the two forensics methodologies to improve manipulation detection performance. We used a public Face2Face dataset and a proprietary Deepfake dataset that we collected ourselves to validate the proposed methodology.

1.3 Proposed System

The process of building customised photo-realistic talking head models, or devices that can create believable video sequences of speech expressions and impersonations of a specific person. More precisely, we take into account the challenge of creating photorealistic personalised head pictures from a collection of facial landmarks, which control the model's motion. Such a capability has real-world uses in the telepresence business, including video conferencing, multiplayer gaming, and the special effects sector. A face forensics methodology that combines the fake face image forensic method with the traditional image forensic method. The suggested model uses two different feature extractors to simultaneously extract content characteristics and trace data from a facial picture. It is a form of convolutional neural network. The feature extractor of a pretrained object recognition model is transferred to and adjusted to train the previously mentioned feature extractor. As a result, the retrieved characteristics are tailored to reflect different facial contents. The latter feature extractor is based on the local relationship between nearby pixels and works by first applying multi-channel constrained convolution to the input image to create a content-excluded image and then extracting the features hierarchically from it.

Objectives

A pretrained object recognition model's feature extractor is transferred to and tweaked to train the previously described feature extractor. As a result, the retrieved traits are adjusted to reflect various facial contents. The latter feature extractor works by first applying multi-channel restricted convolution to the input image to create a content-excluded image and then extracting the features hierarchically from it. The goal of this research is to develop systems that can simulate realistic video sequences of speech expressions and impersonations of a certain person. Training the system to generate deep-fake multimedia using limited datasets.
We also provide a face forensics model for false multimedia identification that combines the traditional image forensic technique with the fake face image forensic approach.

1.4 Requirements Specification

1.4.1 Hardware Requirements

- Processor: Minimum: Intel Core i5 or AMD equivalent
- Memory (RAM): 16 GB
- Storage: Minimum: 512 GB
- Graphics Processing Unit (GPU): Dedicated GPU with at least 2 GB VRAM (NVIDIA GTX 1050 or AMD equivalent)

1.4.2 Software Requirements

- Operating System: Windows 10 (64-bit)
- Development Environment: Python 3.12.1, Visual Studio Code (or any preferred code editor)
- Web Framework: Flask (Python micro web framework)

- Libraries and Dependencies: NumPy (for numerical computations), OpenCV (for image and video processing), TensorFlow (for deep learning-based face swapping), scikit-learn (for machine learning algorithms), Flask-MySQLdb (for MySQL database connection in Flask

2.LITERATURE SURVEY

1. Deepfake and Face2Face Manipulation

In their study, **Keymans Malaysia Sdn Bhd** delves into the burgeoning realm of AI-manipulated fake face media, specifically focusing on the phenomena of Deepfake and Face2Face. They highlight the profound implications these technologies have on privacy and security, sparking concerns within society. While the study provides a thorough examination of face image forensics and general-purpose image forensics, it falls short in offering specific solutions tailored to modern deepfake detection techniques.

2. Deep Vision Technique

BUET (Bangladesh University of Engineering and Technology) introduces an innovative approach to identifying Deepfakes through the analysis of blinking patterns. Their method employs an adversarial network (GANs) model to scrutinize changes in these patterns, which are known to vary significantly among individuals. However, the study's exclusive focus on eye blinking patterns may not encompass all facets of deepfake detection, potentially leaving gaps in its efficacy.

3. Advances in AI for Media Manipulation

Gloria, Padua B. explores the remarkable advancements in AI, machine learning, and deep learning that have facilitated the manipulation of multimedia content. While acknowledging the positive contributions of these technologies, the study also addresses their misuse in activities such as propaganda and blackmail. However, it provides a broad overview of the field without offering specific, actionable detection methodologies, leaving room for further research in this area.

4. High Realism in Multimedia

In their examination of multimedia manipulation techniques, **SEK Socheat, May Sila, and Kong Bunthoeurn** highlight the unprecedented level of realism achieved through deep learning methods like Deepfake. They discuss the diverse applications of these techniques, ranging from enhancing visual effects in media to facilitating the spread of false information. However, the study lacks a detailed analysis of specific deepfake detection methodologies, suggesting a need for further research in this domain.

5. Image Forensics and Manipulations

An anonymous study focuses on the analysis and categorization of modifications in images related to forgery. They propose the manipulation classification network (MCNet), which leverages multi-stream structure to learn forensic features from various domains. However, the study's focus on JPEG images and its failure to address other formats or modern counter-forensic techniques limit its applicability in comprehensive image manipulation detection.

6. Contrast Enhancement Forensics

Another anonymous study delves into contrast enhancement forensic methods and their effectiveness against modern counter-forensic attacks. Their findings suggest promising results, especially in detecting forgery accuracy against such attacks. However, the study's narrow focus on contrast enhancement may restrict its applicability to all types of image manipulations, warranting further investigation.

7. Real-time Manipulation Techniques

Examining real-time generation of manipulated images/videos, an anonymous study explores various technologies facilitating such manipulations. While shedding light on the advancements in real-time manipulation techniques, the study primarily focuses on this aspect, potentially overlooking detection methods for pre-recorded manipulations, suggesting a need for comprehensive detection strategies.

S.NO	NAME	AUTHORS	DESCRIPTION	LIMITATION
1	Deepfake and Face2Face Manipulation	Keymans Malaysia Sdn Bhd	With the rise of AI-manipulated fake face media, realistic-looking facial identities or emotions are created. Deepfake and Face2Face have led to concerns over privacy and security. The study separates the field into face image forensics and general-purpose image forensics, focusing on handcrafted features.	Does not provide specific solutions for modern deepfake detection techniques.
2	Deep Vision Technique	BUET (Bangladesh University of Engineering and Technology)	Proposes a method to identify Deepfakes using an adversarial network (GANs) model to examine changes in blinking patterns. Blinking, an involuntary motion, varies greatly among individuals. This heuristic method monitors changes in eye blinking patterns to identify deepfakes.	Focuses only on eye blinking patterns, which may not cover all types of deepfake manipulations.
3	Advances in AI for Media Manipulation	Gloria, Padua B.	AI, machine learning, and deep learning advancements have led to tools for altering multimedia. Despite positive	Broad overview, lacks specific,

			uses, these tools have also been misused for illegal activities like propaganda and blackmail. Research has discussed various methods to address deepfake issues, providing an updated overview of the field.	actionable detection methodologies.
4	High Realism in Multimedia	SEK Socheat, May Sila, Kong Bunthoeurn	Describes advancements in multimedia manipulation techniques that ensure high realism, using deep learning techniques like Deepfake. Applications include visual effects in media and illegal activities like spreading false information. Research focuses on deepfake identification using deep neural networks (DNNs).	General description, needs more detailed analysis of DNN-based detection methods.
5	Image Forensics and Manipulations	Anonymous	Focuses on analyzing and categorizing modifications in images related to forgery. Proposes the manipulation classification network (MCNet) that uses multi-stream structure to learn forensic features from spatial, frequency, and compression domains.	Limited to JPEG images, does not address other image formats or modern counter-forensic techniques.
6	Contrast Enhancement Forensics	Anonymous	Discusses CE forensic methods using handcrafted features based on first-and second-order statistics. Highlights difficulties in detecting modern counter-forensic attacks. Experimental results show the proposed method outperforms conventional forensic methods in forgery detection accuracy, especially against counter-forensic attacks.	Focuses on contrast enhancement, may not be applicable to all types of image manipulations.
7	Real-time Manipulation Techniques	Anonymous	Examines real-time generation of manipulated images/videos using technologies like image morphing, Snap-Chat, CGFI, GAN, and Face2Face. Considers two types of manipulation: Source-to-target and Self-reenactment.	Real-time focus may overlook detection methods for pre-recorded manipulations.

3. DESIGN METHODOLOGY

Given a source image and a target image, we present a framework that transfers the identity of the source face into the target face while keeping the attributes of target face unchanged. Our framework extends from an identity-specific face swapping architecture (DeepFakes, 2020) and can be adapted to arbitrary identities. We first discuss the limitation of the original architecture(Sec 3.1). We show how to extend it to a framework for arbitrary identity.Then we present the Weak Feature Matching Loss which helps to preserve target's attributes. Finally, we give out our loss function.

3.1 DEEPFAKE CREATION

3.1.1 ARCHITECTURE

Deepfake technology has revolutionized the landscape of digital imagery, offering a fascinating glimpse into the convergence of creativity and manipulation. At its core lies a complex architecture designed to seamlessly blend reality and fiction, crafting images and videos that challenge our very perception of truth.

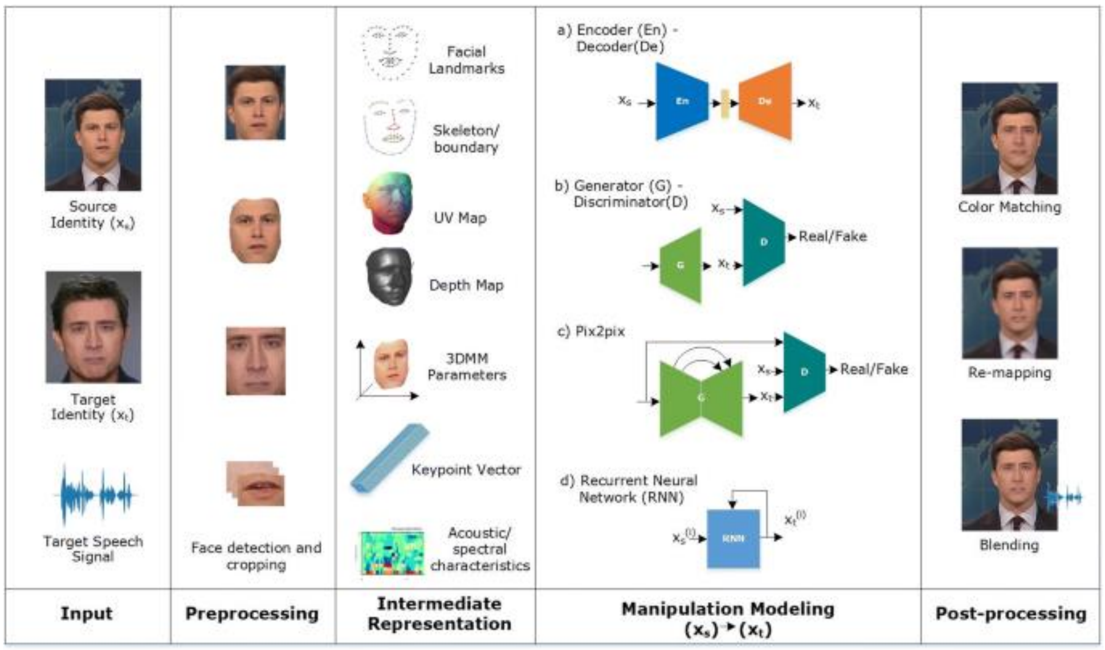


Fig:Architecture of Deepfake Creation

In this digital realm, intricate algorithms serve as the building blocks, meticulously analyzing and reconstructing faces, voices, and movements with astonishing accuracy. These algorithms form the foundation of creation, allowing artists and enthusiasts alike to delve into a world where imagination knows no bounds.

Yet, beneath the surface of this creative tapestry lies the architecture of manipulation. Here, algorithms are honed not only to construct but also to deceive, meticulously crafting deepfakes that blur the lines between authenticity and fabrication. With every frame manipulated, every pixel altered, the boundaries of reality are pushed further, raising profound questions about trust and authenticity in an increasingly digital age.

As we navigate this evolving landscape, it's essential to recognize the dual nature of deepfake architecture — a tool for both artistic expression and potential deception. Understanding its intricacies offers a glimpse into the future of digital creativity, where innovation and responsibility walk hand in hand.

In this dynamic interplay of creation and manipulation, the architecture of deepfaking continues to shape our perceptions, challenging us to explore the boundaries of truth and fiction in an ever-expanding digital canvas

3.1.2 Limitation of the DeepFakes

The architecture of DeepFakes contains two parts, a common Encoder Enc and two identity-specific Decoders $DecS$,$EncT$. In the training stage, the Enc-$DecS$ architecture takes in the warped source images and restore them to the original unwarped source images. The same procedure will be conducted with the target images using the Enc-$DecT$ architecture. In the test stage, a target image will be sent to the Enc-$DecS$ architecture. The architecture will mistake it for a warped source image and produce an image with the source's identity and the target's attributes.

During this process, the Encoder Enc extracts the target's features which contain both identity and attribute information of the target face. Since the Decoder $DecS$ manages to convert the target's features to an image with source's identity, the identity information of the source face must have been integrated into the weights of $DecS$. So the Decoder in DeepFakes can be only applied to one specific identity.

3.1.3 Generalization to Arbitrary Identity

To overcome such limitation, we are seeking a way to separate the identity information from the Decoder so that the whole architecture can be generalized to arbitrary identity. We improve the architecture by adding an additional ID Injection Module between the Encoder and the Decoder. Our framework is shown in Figure 2. Given a target image IT, we pass it through our Encoder to extract its features $FeaT$. Since our task is to swap the target face with the source face, we have to replace the identity information in $FeaT$ with the identity information of source face while keeping the attribute information in $FeaT$ unchanged. However, the identity and attribute information in $FeaT$ are highly coupled and difficult to tell apart. So we directly conduct modifications on the whole $FeaT$ and we are using the training loss to encourage the network to learn implicitly which part of $FeaT$ should be changed and which part should be preserved.

Our ID Injection Module works on changing the identity information in $FeaT$ towards the identity information of the source face. The module is composed of two parts, the identity extraction part and the embedding part. In the identity extraction part, we deal with the input source image IS which contains both identity and attribute information of source face. Since we only need the former, we use a face recognition network (Deng et al., 2019) to extract the identity vector vS from IS. In the embedding part, we are using the ID-Blocks to inject the identity information into the features. Our ID-Block is a modified version of the Residual Block (He et al., 2016) and we are using the Adaptive Instance Normalization(AdaIN) (Huang and Belongie, 2017) to replace the original Batch Normalization (Ioffe and Szegedy, 2015). The formulation of AdaIN in our task can be written as

$$AdaIN(Fea, v_S) = \sigma_S \frac{Fea - \mu(Fea)}{\sigma(Fea)} + \mu_S$$

Here, $\mu(Fea)$ and $\sigma(Fea)$ is the channel-wise mean and standard deviation of the input feature Fea. σS and μS are two variables generated from vS using full connected layers. To guarantee enough identity embedding, we are using a total of 9 ID-Blocks.

After the injection of identity information, we pass the modified features through the Decoder to generate the final result IR. Since source images from different identities are involved in the training, the weights of the Decoder should be unrelated to any specific identity. Our Decoder will just focus on restoring the image from the features and leave the identity modification mission to the ID Injection Module, so we can apply our architecture to arbitrary identities.

During the training process, we extract the identity vector vR from the generated result IR and we use the Identity Loss to minimize the distance between vR and vS. However, the minimization of the Identity Loss can make the network overfitted and only generate front face images with the source's identity while losing all the target's attributes. To avoid such phenomena, we utilize the idea of adversarial training (Goodfellow et al., 2014; Park et al., 2019; Liu et al., 2019; Karras et al., 2019) and use the Discriminator to distinguish results with apparent error. The Adversarial Loss also plays an import role in improving the quality of the generated result. We use the patchGAN (Isola et al., 2017) version of the Discriminator.

3.1.4 Preserving the Attributes of the Target

In the face swapping task, the modification should be only conducted in the identity part and the attributes (*e.g.* expression, posture, lighting etc.) of the target face should remain unchanged. However, since we are directly conducting modifications on the whole $FeaT$ which contains both identity and attribute information of target face, the attribute information is likely to be affected by the identity embedding. To prevent the attribute mismatch, we are using the training loss to constrain them. However, if we choose to constrain all the attributes explicitly, we will have to train one network for each attribute. The whole process should be impractical since there are too many attributes should be considered. So we propose to use the Weak Feature Matching Loss to do the constraining in an implicit way.

The idea of Feature Matching originated in pix2pixHD (Wang et al., 2018a) which used the Discriminator to extract multiple layers of features from the Ground Truth image and the generated output. The original Feature Matching Loss is written as:

$$L_{oFM}(D) = \sum_{i=1}^{M} \frac{1}{N_i} \|D^{(i)}(I_R) - D^{(i)}(I_{GT})\|_1$$

Here $D(i)$ denotes the i-th layer feature extractor of Discriminator D and Ni denotes the number of elements in the i-th layer. M is the total number of layers. IR is the generated output and IGT is its corresponding Ground Truth image.

In our architecture, since there's no Ground Truth in face swapping task, we are using the input target image IT to replace its position. We remove the first few layers and only use the last few layers to calculate our Weak Feature Matching Loss, which can be written as:

$$L_{wFM}(D) = \sum_{i=m}^{M} \frac{1}{N_i} \|D^{(i)}(I_R) - D^{(i)}(I_T)\|_1$$

Here m is the layer where we start to calculate the Weak Feature Matching Loss.

Although the original Feature Matching Loss and Weak Feature Matching Loss share similar formulations, their objectives are totally different. The original Feature Matching Loss is proposed to stabilize the training and the generator is required to produce natural statistics at multiple levels. The features of shallow layers will play the key role since they mainly contain texture information and are able to constrain the results at pixel level. However, in our face swapping task, introducing too much texture information from the input target image will make the result similar to target face and cause difficulty in identity modification, so we remove the first few layers in the original Feature Matching term. Our objective is to constrain the attribute performance. Since the attribute is high semantic information which mainly lies in deep features, we are requiring the result image to align with the input target at deep level, and our Weak Feature Matching Loss is only using the last few layers of the Discriminator to calculate the Feature Matching term. By using such a loss function, even if we are not explicitly constraining the network on any specific attribute, it will implicitly learn how to preserve the attributes of the input target face.

3.1.5 Overall Loss Function

Our Loss function has 5 components, including Identity Loss, Reconstruction Loss, Adversarial Loss, Gradient Penalty and Weak Feature Matching Loss.

Identity Loss Identity Loss is used to constrain the distance between vR and vS. We are using the cosine similarity to calculate the distance, which can be written as:

$$L_{Id} = 1 - \frac{v_R \cdot v_S}{\|v_R\|_2 \|v_S\|_2}$$

Reconstruction Loss If the source face and the target face are from the same identity, the generated result should look the same as target face. We are using the Reconstruction Loss as a regularization term, which can be written as:

$$L_{Recon} = \|I_R - I_T\|_1$$

We set this term to 0 if the source and target faces are from different identities.

Adversarial Loss and Gradient Penalty We are using the Hinge version (Park et al., 2019; Brock et al., 2019; Liu et al., 2019) of the Adversarial Loss. We use multi-scale Discriminators (Wang et al., 2018a) for better performance under large postures. We also utilize the Gradient Penalty term (Arjovsky et al., 2017; Gulrajani et al., 2017) to prevent the Discriminators from gradient explosion.

Weak Feature Matching Loss Since we are using mutli-scale Discriminator, the Weak Feature Matching Loss should be calculated using all Discriminators, which can be written as:

$$L_{wFM\_sum} = \sum_{i=1}^{2} L_{wFM}(D_i)$$

The overall Loss can be written as:

$$\lambda_{Id} L_{Id} + \lambda_{Recon} L_{Recon} + L_{Adv} + \lambda_{GP} L_{GP} + \lambda_{wFM} L_{wFM\_sum}$$

3.2 Detailed design For DeepFake Creation

3.2.1 Use Case Diagram

In the realm of deepfaking, USECASE architecture plays a vital role in both creation and detection. This architecture involves designing and implementing systems tailored to specific scenarios or applications, hence the acronym USECASE (Utilization and Scalable Evaluation of deepfake Content). On the creation side, it enables the development of more sophisticated and targeted deepfake content, such as for entertainment, education, or even artistic expression. Conversely, on the detection side, USECASE architecture focuses on building robust algorithms and tools to identify deepfakes within these specific contexts, enhancing the ability to safeguard against the potential misuse of this technology. Overall, USECASE architecture serves as a framework for both harnessing the creative potential and mitigating the risks associated with deepfake technology in various real-world scenarios.

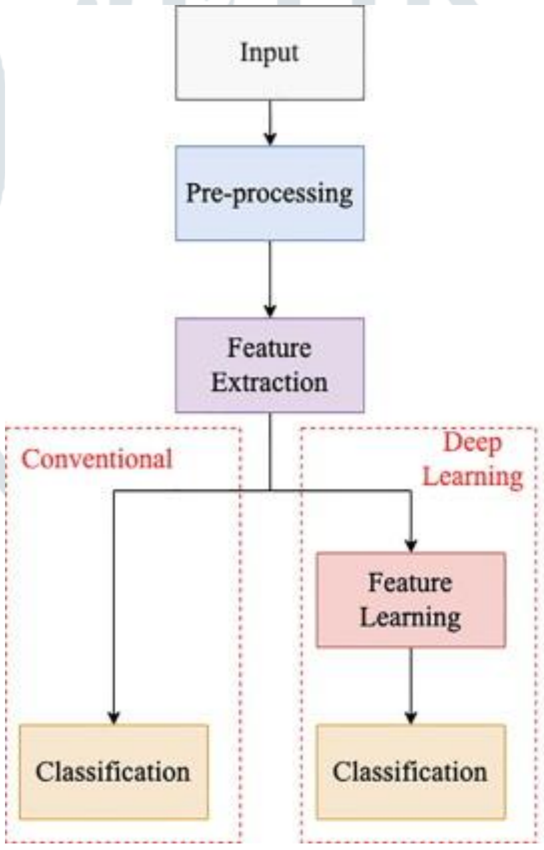


Fig: Use Case Diagram

3.2.2 Sequence Diagram

A sequence diagram for deepfaking images and videos illustrates the step-by-step process involved in creating realistic but fabricated visuals. It typically starts with data collection, where images or videos of the target subjects are gathered. Then, preprocessing steps such as facial recognition and alignment occur to prepare the data for manipulation. Next comes the heart of the process: the deepfake model, which uses sophisticated algorithms like generative adversarial networks (GANs) to generate new images or videos that mimic the appearance of the target subjects. After the deepfake generation, post-processing techniques may be applied to enhance the realism of the result. Finally, the deepfaked images or videos are rendered and distributed. This sequence diagram helps understand the intricate workflow involved in the creation of deepfakes, highlighting the various stages from data input to the production of the final manipulated content.

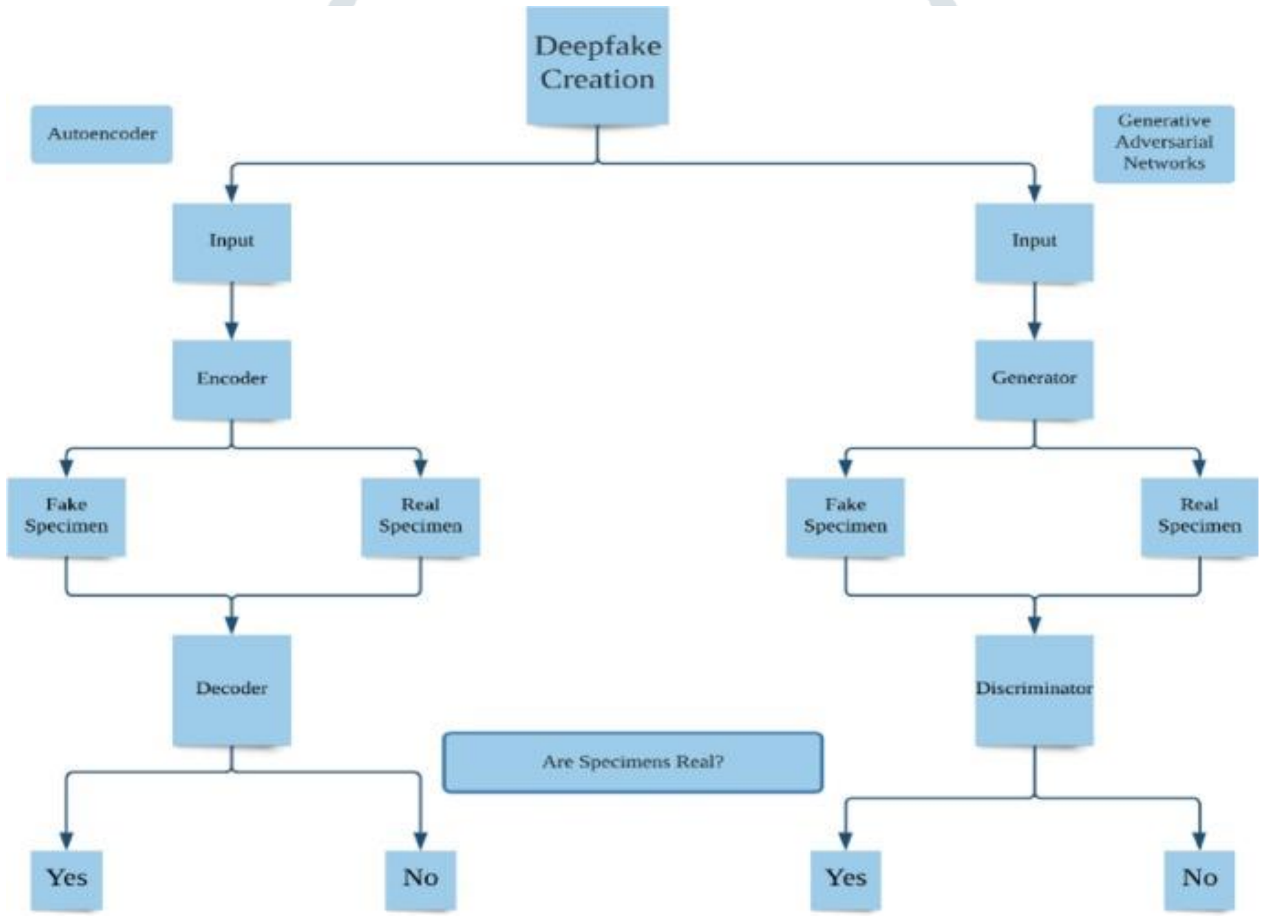


Fig:Sequence Diagram

3.2.3 Activity Diagram

Activity diagrams provide a visual representation of the step-by-step processes involved in deepfaking images and videos. At the heart of this diagram is the intricate dance between data processing and algorithmic manipulation. It starts with data acquisition, where raw images or videos are collected. Then, the preprocessing stage kicks in, where the data is cleaned and prepared for manipulation. Next, the core of deepfake creation unfolds through the application of sophisticated algorithms, which seamlessly blend and alter the visuals to create the desired effect. Post-processing adds the finishing touches, refining the output to enhance realism. Throughout this diagram, the iterative nature of deepfake creation is evident, as adjustments are made based on feedback to achieve the desired result. However, it's crucial to remember the ethical considerations surrounding deepfakes, as they have the potential to deceive and manipulate if used irresponsibly.

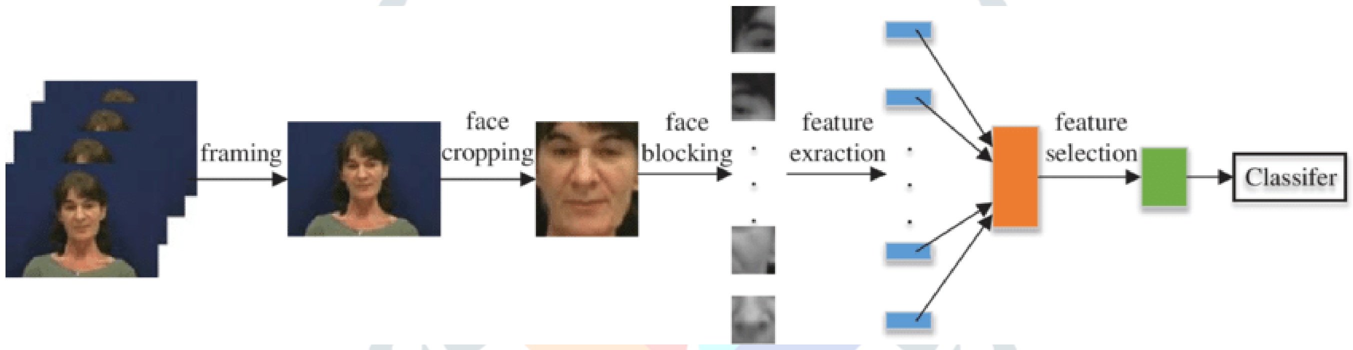


Fig: Activity Diagram

3.2.4 Component Diagram

The component diagram of deepfaking images and videos visually represents the key elements and their interactions within the deepfake creation process. It outlines the various components involved in manipulating digital media, illustrating how they work together to produce convincing but potentially deceptive content

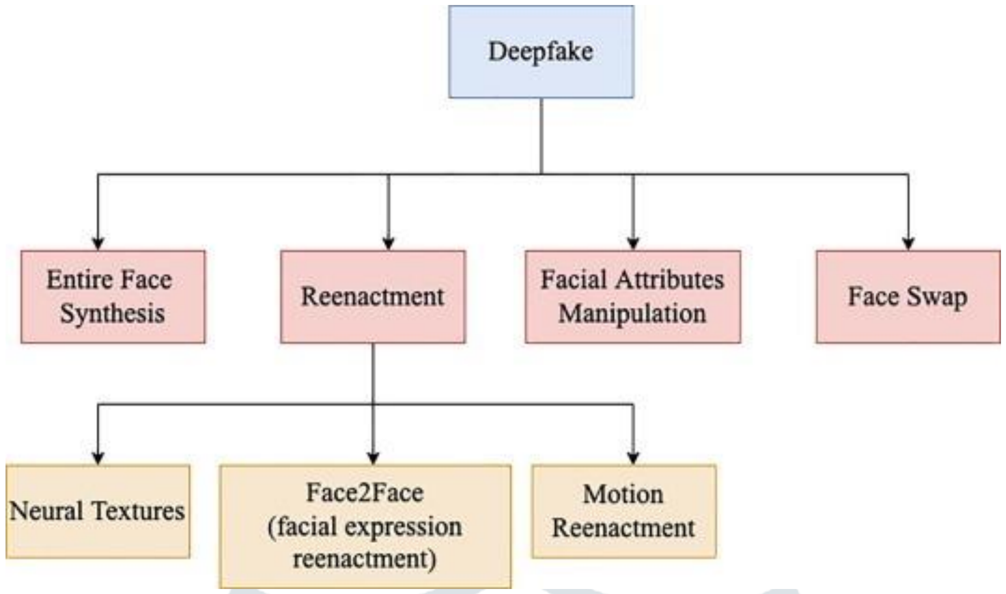


Fig: Component Diagram

3.3 DEEPFAKE DETECTION:

3.3.1 Architecture

Detection architecture for deepfakes is like a digital security guard. It's a smart system that checks photos and videos to see if they're real or fake. Using clever technology, it looks for clues that reveal if something's been changed. Just like how you might spot a fake painting by noticing details that don't look quite right, this system looks for inconsistencies in the pixels and patterns of digital images. Its job is to help us tell fact from fiction in the world of online media.

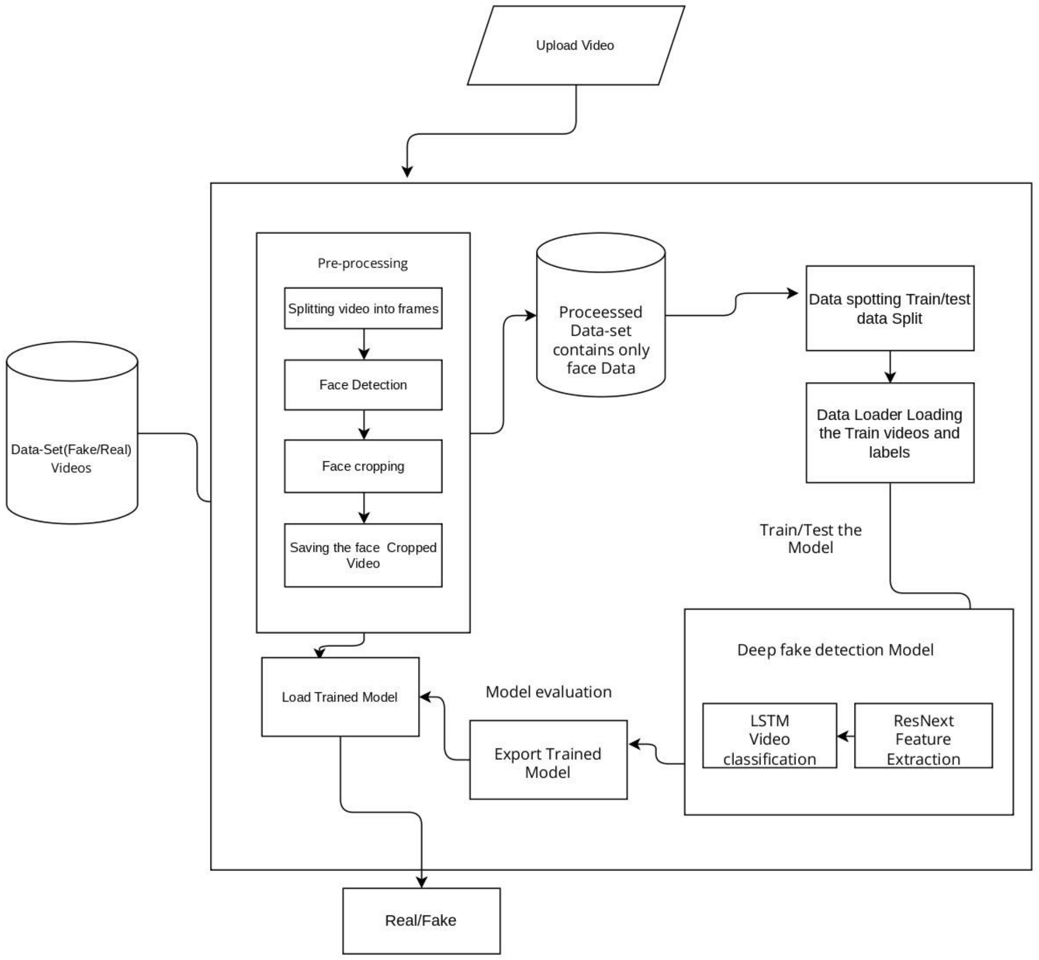

Upload Video
Data-Set(Fake/Real) Videos
Pre-processing
Splitting video into frames
Face Detection
Face cropping
Saving the face Cropped Video
Proceessed Data-set contains only face Data
Data spotting Train/test data Split
Data Loader Loading the Train videos and labels
Train/Test the Model
Deep fake detection Model
LSTM Video classification
ResNext Feature Extraction
Model evaluation
Export Trained Model
Load Trained Model
Real/Fake

Fig: Architecture

3.2.2 Module description:

1 .User module: In the user module the user opens the webpage. 1.Upload video. 2.Start detection. 3.Results. After completing the process it classifies real and fake video.

2. Machine-learning module: The machine learning module comprises of the stress model which takes in image/live detection video of the user as input.After collecting image/video from the user, it detects whether the user is stressed or not stressed.It uses CNN with multiple layers with 20 epochs**.**

3. Feature extraction module: The primary goal of the feature extraction module is to analyze the visual content of a video and extract meaningful features or attributes that can help distinguish between authentic and manipulated videos. These features are usually derived from both spatial and temporal characteristics of the video frames.

4. Classification module: The classification module receives input features that have been extracted from the video frames by the feature extraction module. These features encapsulate various spatial and temporal characteristics of the video, which are indicative of whether the video is authentic or manipulated**.**

3.4 Detailed design for DeepFake Detection:

3.4.1 Use Case Diagram

A use case diagram is a visual representation in Unified Modelling Language (UML) that illustrates the interactions between a system and its external actors, showcasing various ways the system can be utilized. It identifies specific functionalities or "use cases" that the system provides to its users. Actors represent external entities interacting with the system. Relationships between actors and use cases depict how users engage with the system's features. Use case diagrams are valuable for capturing and communicating high-level system behavior and functionality.

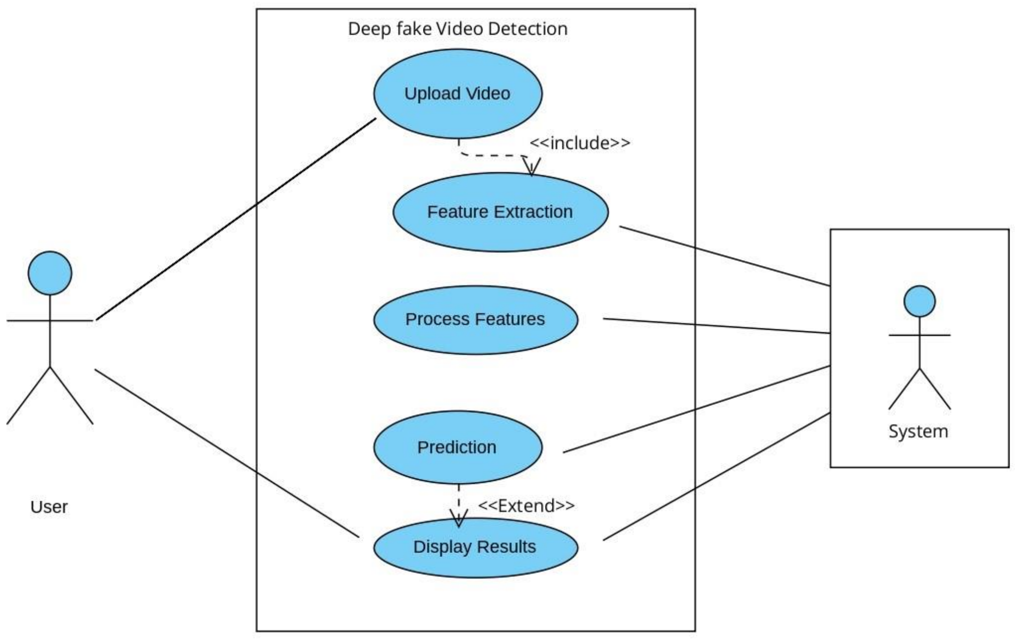


Figure : Use Case diagram

3.4.2 Sequence Diagrams

A sequence diagram is a type of UML diagram that illustrates interactions between objects or components in a system over time. It represents the flow of messages and the order of events in a dynamic view of a system. Lifelines, represented by vertical lines, depict the participants in the interaction. Arrows indicate the messages exchanged between lifelines, showing the chronological order of interactions. Sequence diagrams help visualize and understand the behaviours of a system during runtime**.**

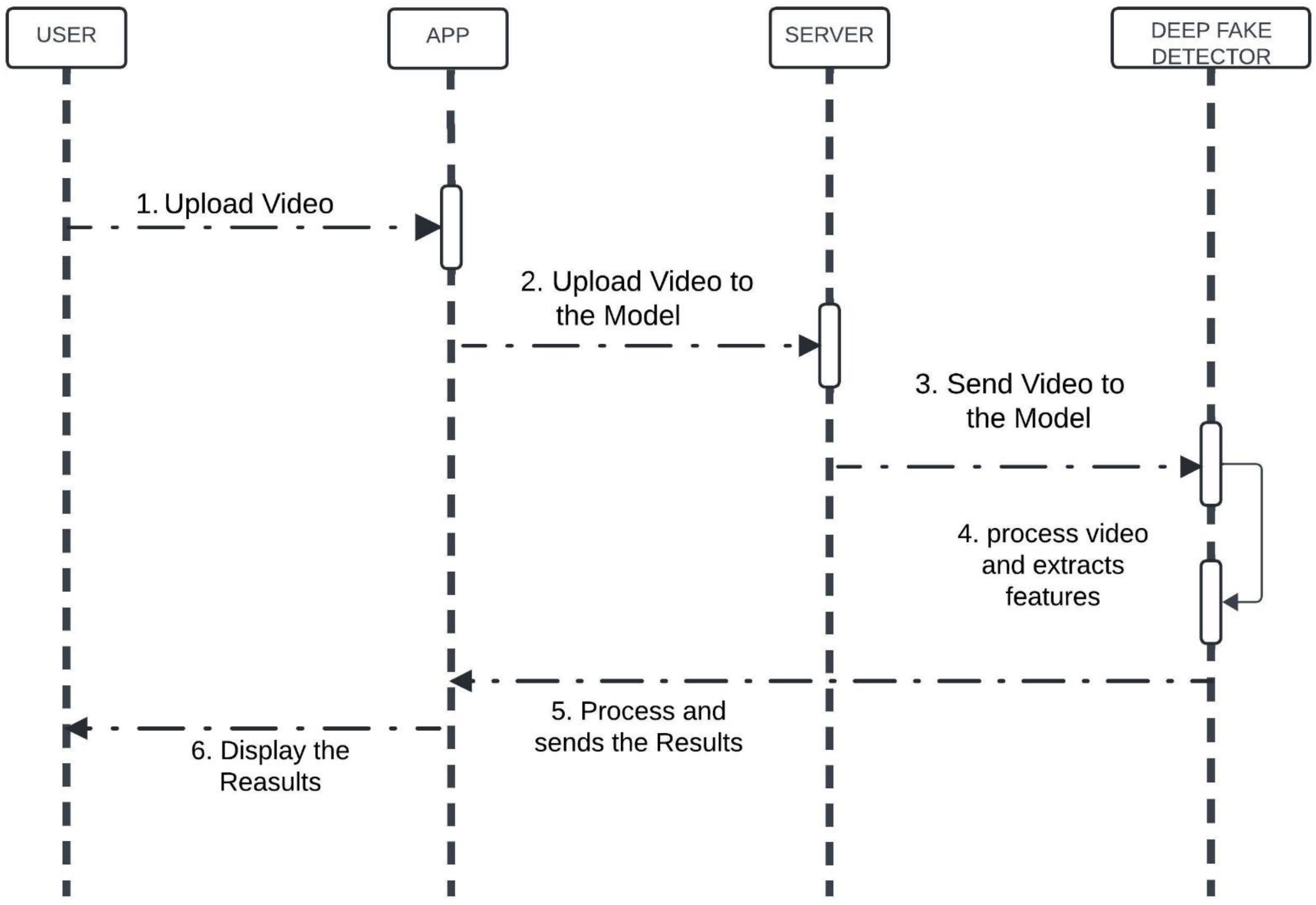

USER
APP
SERVER
DEEP FAKE DETECTOR
1. Upload Video
2. Upload Video to the Model
3. Send Video to the Model
4. process video and extracts features
5. Process and sends the Results
6. Display the Reasults

Figure :Sequence Diagram

3.4.3 Activity Diagrams

An activity diagram is a UML (Unified Modelling Language) diagram that depicts the flow of activities within a system or process. It visually represents actions, decision points, and the sequential order of tasks. Nodes and arrows illustrate the workflow, making it easy to understand complex processes. This may be used to show responsibilities among different entities. Activity diagrams are valuable for modelling business processes, software algorithms, and workflow scenarios.

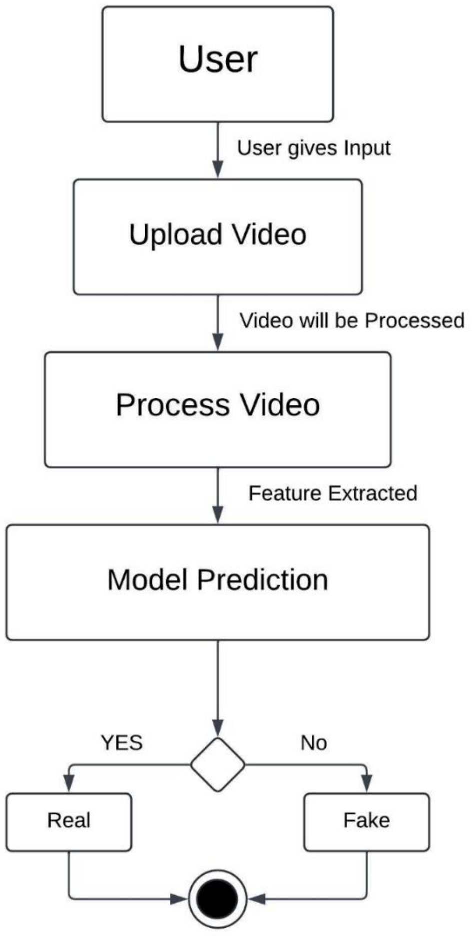


Figure :Activity diagram

4.IMPLEMENTATION

4.1 DEEPFAKE CREATION

Code for swapping faces:

```
!cd SimSwapClone & python test_video_swapsingle.py --crop_size 224 --use_mask --name people --Arc_path arcface_model/arcface_checkpoint.tar--pic_a_path./videoswap/siactress.jpg--video_path ./videoswap/girlsmile(f8).mp4 --output_path ./videoswap/outputsiact.mp4 --temp_path ./temp_results
```

Test Case 1:

Source image: The source image is the canvas upon which truth is painted and fiction takes flight.

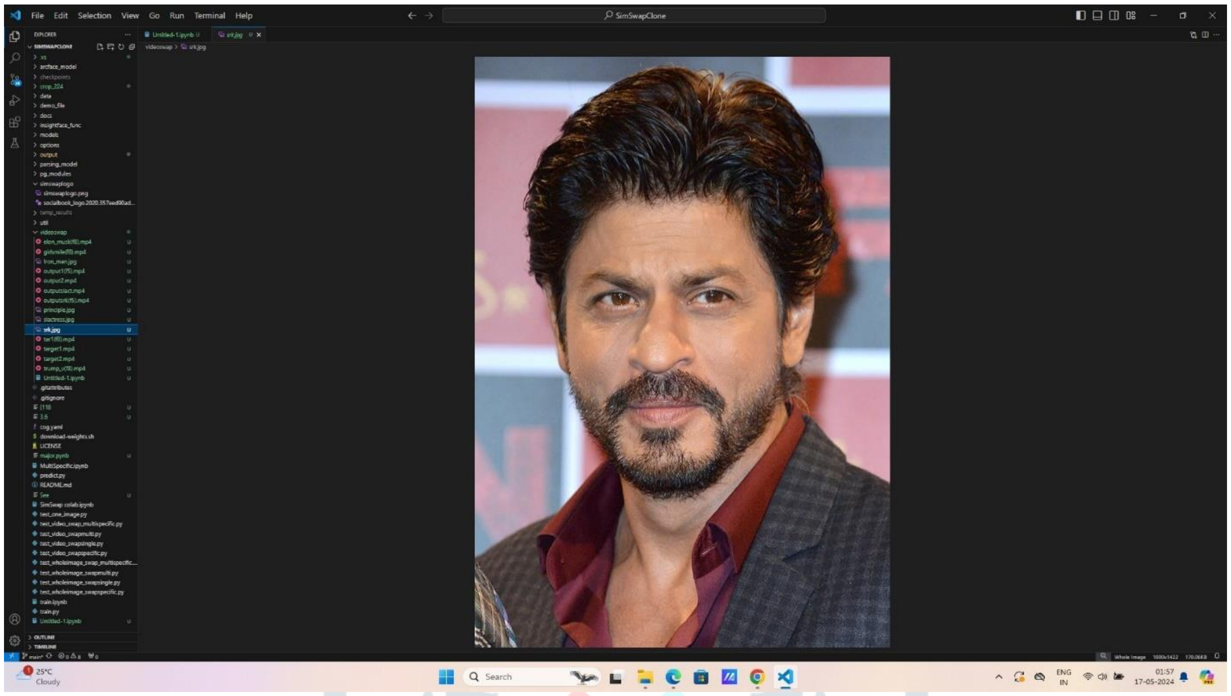
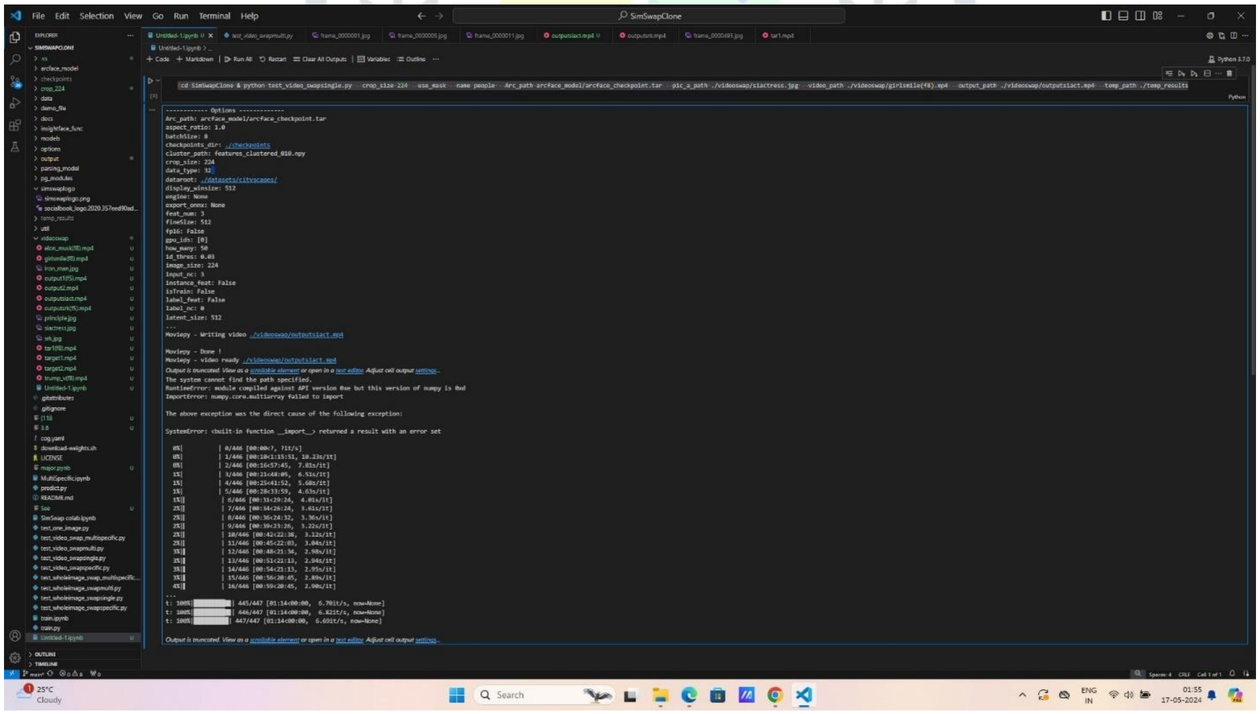


Fig: Source image



```
cd SimSwapClone & python test_video_swapsingle.py --crop_size 224 --use_mask --name people --Arc_path arcface_model/arcface_checkpoint.tar --pic_a_path ./videoswap/siactress.jpg --video_path ./videoswap/girlsmile(f8).mp4 --output_path ./videoswap/outputsiact.mp4 --temp_path ./temp_results
```

```
------------ Options -------------
Arc_path: arcface_model/arcface_checkpoint.tar
aspect_ratio: 1.0
batchSize: 8
checkpoints_dir: ./checkpoints
cluster_path: features_clustered_010.npy
crop_size: 224
data_type: 32
dataroot: ./datasets/cityscapes/
display_winsize: 512
engine: None
export_onnx: None
feat_num: 3
fineSize: 512
fp16: False
gpu_ids: [0]
how_many: 50
id_thres: 0.03
image_size: 224
input_nc: 3
instance_feat: False
isTrain: False
label_feat: False
label_nc: 0
latent_size: 512
...
Moviepy - Writing video ./videoswap/outputsiact.mp4

Moviepy - Done !
Moviepy - video ready ./videoswap/outputsiact.mp4
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
The system cannot find the path specified.
RuntimeError: module compiled against API version 0xe but this version of numpy is 0xd
ImportError: numpy.core.multiarray failed to import

The above exception was the direct cause of the following exception:

SystemError: <built-in function __import__> returned a result with an error set

  0%|          | 0/446 [00:00<?, ?it/s]
  0%|          | 1/446 [00:10<1:15:51, 10.23s/it]
  0%|          | 2/446 [00:16<57:45,  7.81s/it]
  1%|          | 3/446 [00:21<48:05,  6.51s/it]
  1%|          | 4/446 [00:25<41:52,  5.68s/it]
  1%|          | 5/446 [00:28<33:59,  4.63s/it]
  1%|▏         | 6/446 [00:31<29:24,  4.01s/it]
  2%|▏         | 7/446 [00:34<26:24,  3.61s/it]
  2%|▏         | 8/446 [00:36<24:32,  3.36s/it]
  2%|▏         | 9/446 [00:39<23:26,  3.22s/it]
  2%|▏         | 10/446 [00:42<22:38,  3.12s/it]
  2%|▏         | 11/446 [00:45<22:03,  3.04s/it]
  3%|▎         | 12/446 [00:48<21:34,  2.98s/it]
  3%|▎         | 13/446 [00:51<21:13,  2.94s/it]
  3%|▎         | 14/446 [00:54<21:13,  2.95s/it]
  3%|▎         | 15/446 [00:56<20:45,  2.89s/it]
  4%|▎         | 16/446 [00:59<20:45,  2.90s/it]
...
t: 100%|██████████| 445/447 [01:14<00:00,  6.70it/s, now=None]
t: 100%|██████████| 446/447 [01:14<00:00,  6.82it/s, now=None]
t: 100%|██████████| 447/447 [01:14<00:00,  6.69it/s, now=None]

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Features extracted from source image

Fig: Features Extraction

Target Video: Every pixel in the target image is a canvas awaiting the brush of manipulation, where subtlety meets strategy in the art of visual deception.

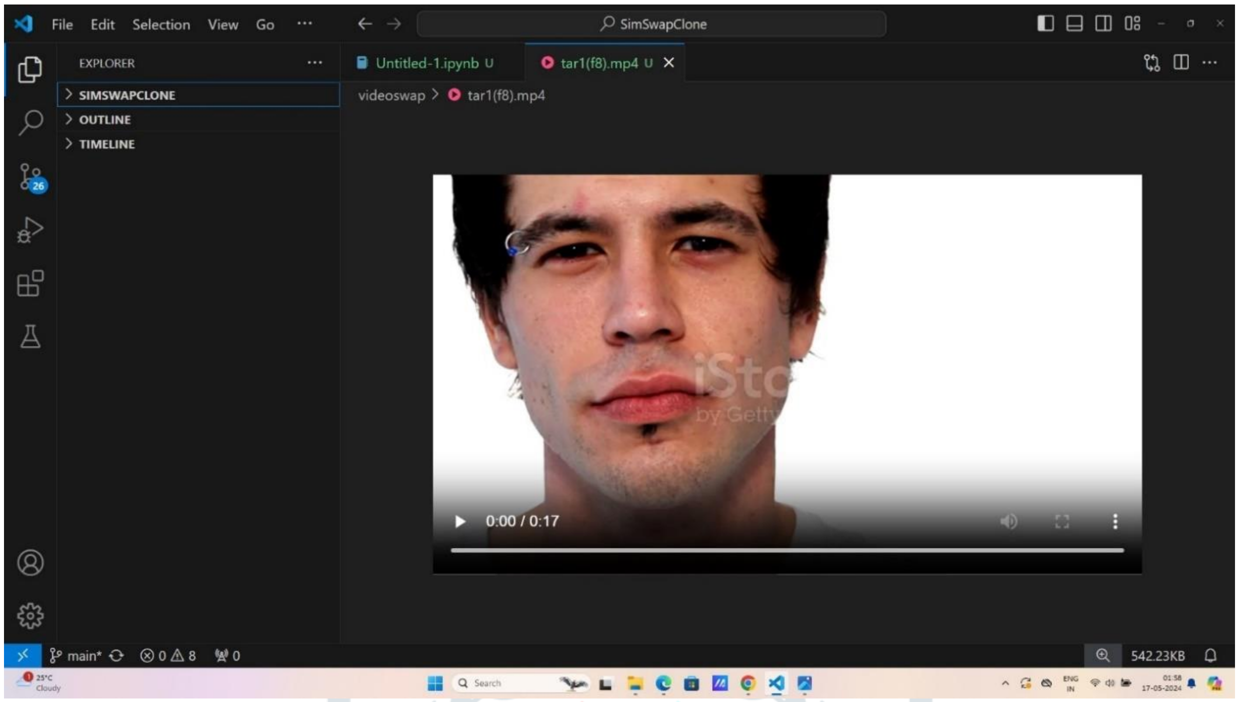
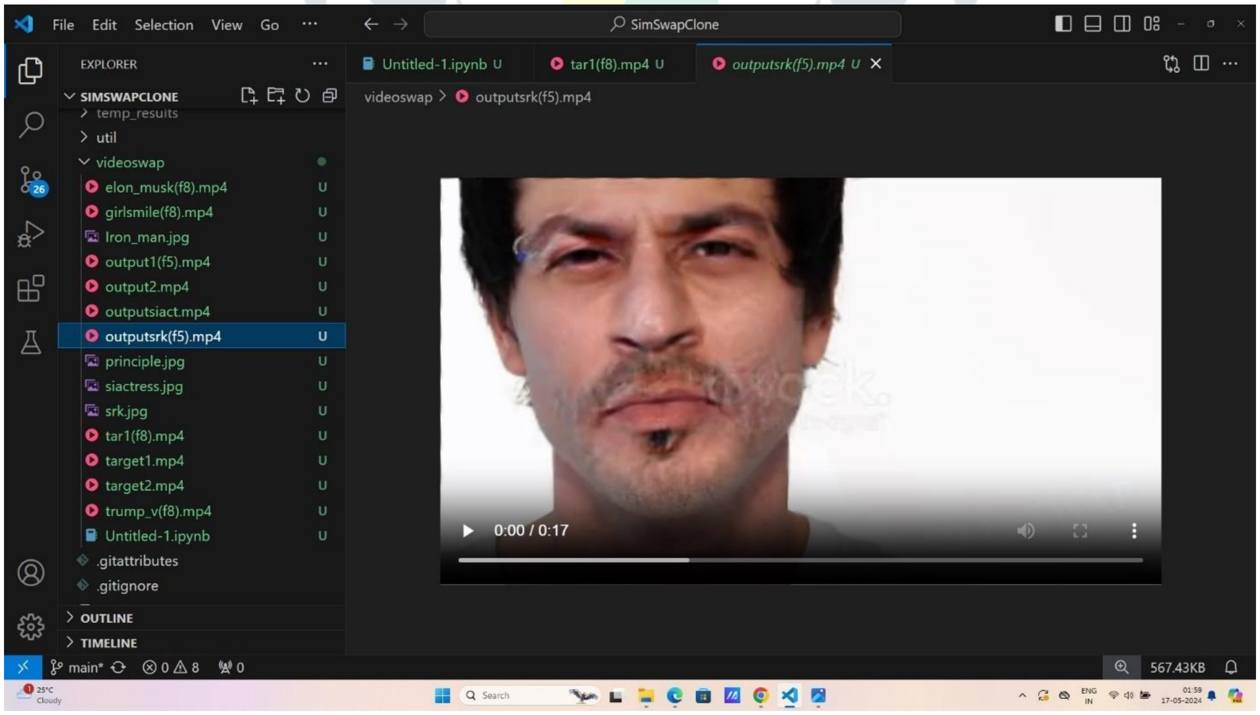


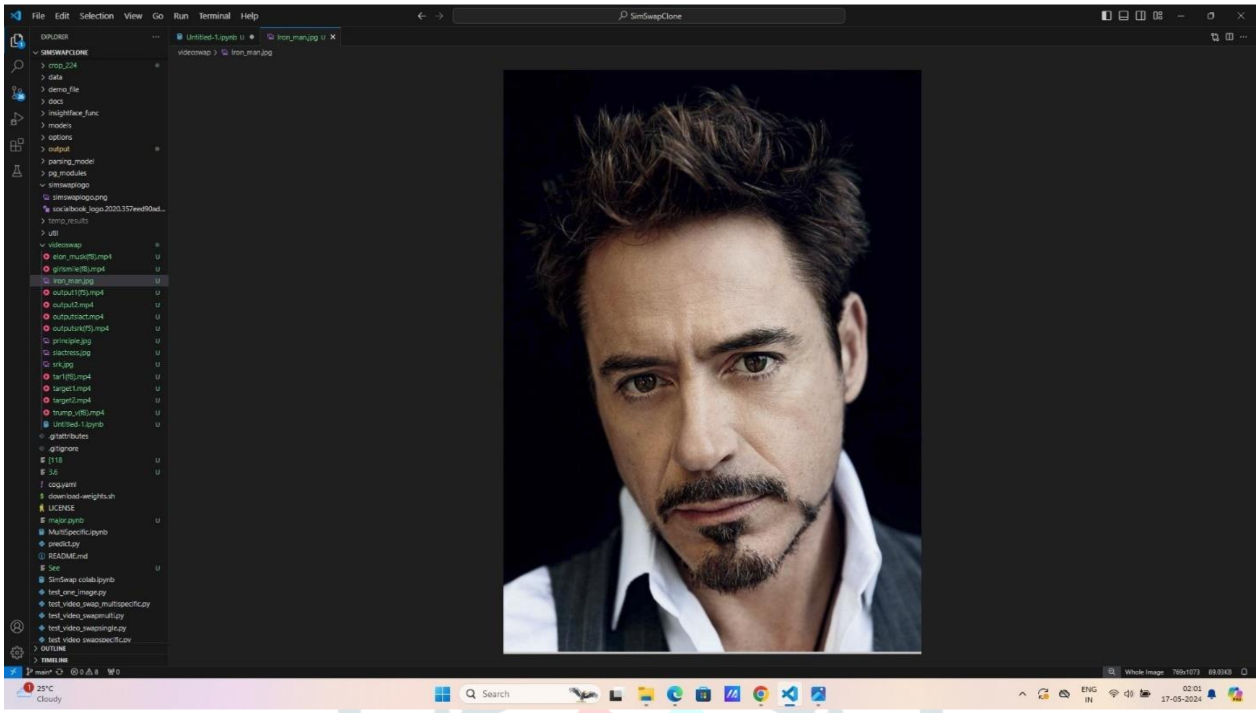
Fig Target Video

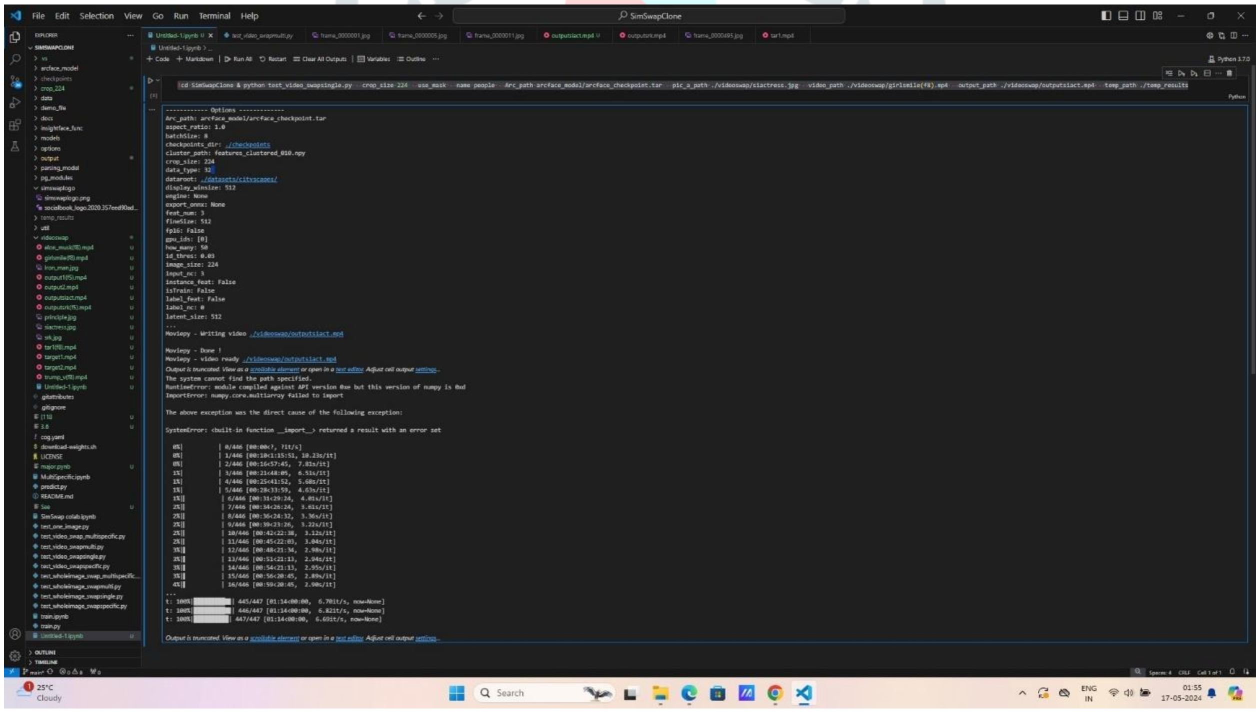
Result: Within the pixels of a deepfaked image, reality and fantasy intertwine, casting shadows of doubt upon the canvas of perception



Case 2:

Source image: The source image is the canvas upon which truth is painted and fiction takes flight.



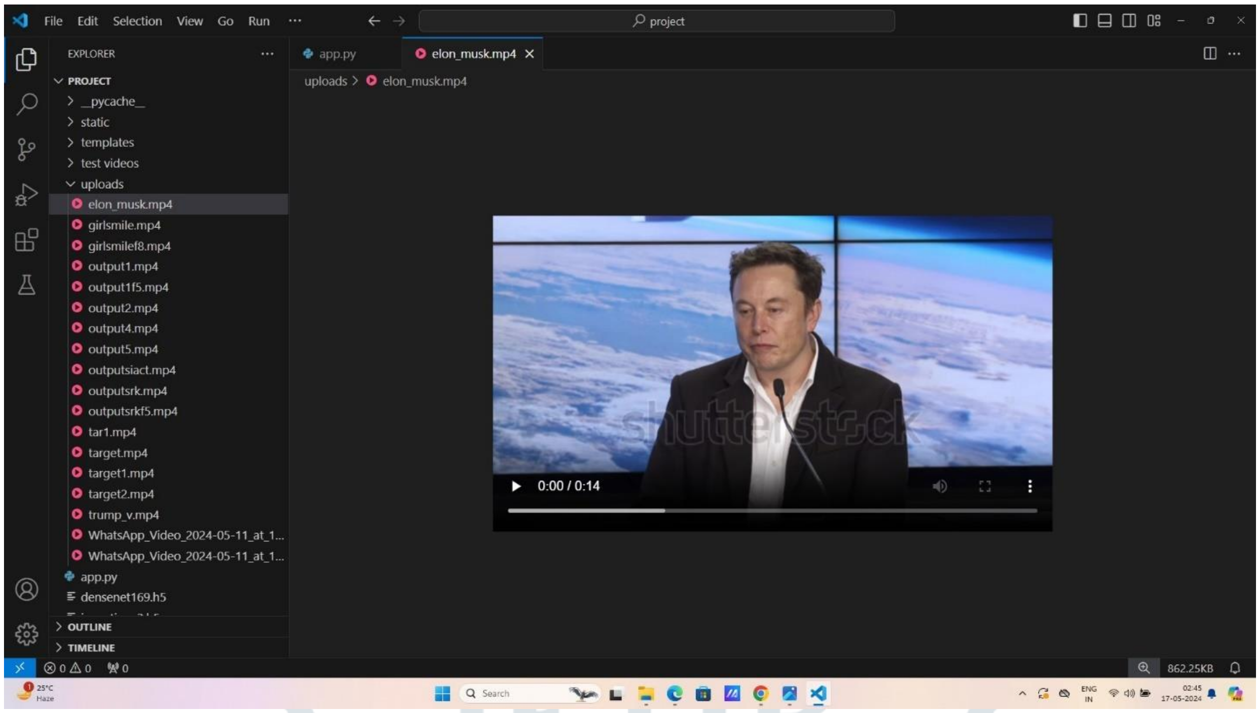


```
!cd SimSwapClone & python test_video_swapsingle.py --crop_size 224 --use_mask --name people --Arc_path arcface_model/arcface_checkpoint.tar --pic_a_path ./videoswap/siactress.jpg --video_path ./videoswap/girlsmile(f8).mp4 --output_path ./videoswap/outputsiact.mp4 --temp_path ./temp_results
```

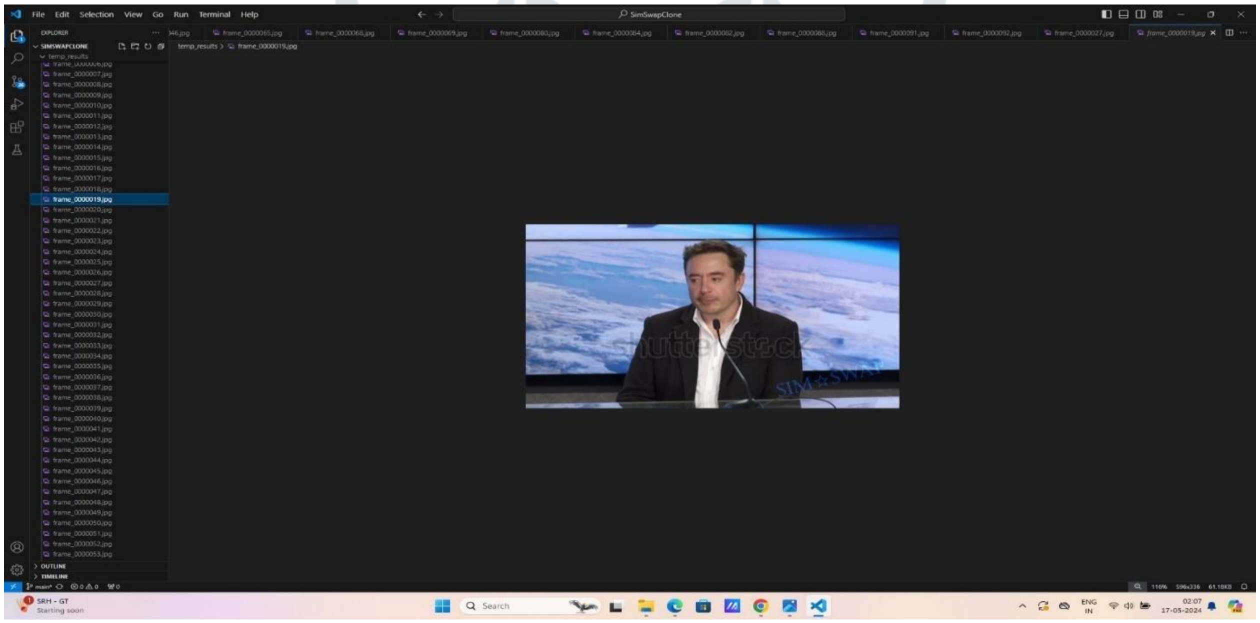
```
------------ Options -------------
Arc_path: arcface_model/arcface_checkpoint.tar
aspect_ratio: 1.0
batchSize: 8
checkpoints_dir: ./checkpoints
cluster_path: features_clustered_010.npy
crop_size: 224
data_type: 32
dataroot: ./datasets/cityscapes/
display_winsize: 512
engine: None
export_onnx: None
feat_num: 3
fineSize: 512
fp16: False
gpu_ids: [0]
how_many: 50
id_thres: 0.03
image_size: 224
input_nc: 3
instance_feat: False
isTrain: False
label_feat: False
label_nc: 0
latent_size: 512
...
Moviepy - Writing video ./videoswap/outputsiact.mp4

Moviepy - Done !
Moviepy - video ready ./videoswap/outputsiact.mp4
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
The system cannot find the path specified.
RuntimeError: module compiled against API version 0xe but this version of numpy is 0xd
ImportError: numpy.core.multiarray failed to import

The above exception was the direct cause of the following exception:

SystemError: <built-in function __import__> returned a result with an error set

  0%|          | 0/446 [00:00<?, ?it/s]
  0%|          | 1/446 [00:10<1:15:51, 10.23s/it]
  0%|          | 2/446 [00:16<57:45,  7.81s/it]
  1%|          | 3/446 [00:21<48:05,  6.51s/it]
  1%|          | 4/446 [00:25<41:52,  5.68s/it]
  1%|          | 5/446 [00:28<33:59,  4.63s/it]
  1%|▏         | 6/446 [00:31<29:24,  4.01s/it]
  2%|▏         | 7/446 [00:34<26:24,  3.61s/it]
  2%|▏         | 8/446 [00:36<24:32,  3.36s/it]
  2%|▏         | 9/446 [00:39<23:26,  3.22s/it]
  2%|▏         | 10/446 [00:42<22:38,  3.12s/it]
  2%|▏         | 11/446 [00:45<22:03,  3.04s/it]
  3%|▎         | 12/446 [00:48<21:34,  2.98s/it]
  3%|▎         | 13/446 [00:51<21:13,  2.94s/it]
  3%|▎         | 14/446 [00:54<21:13,  2.95s/it]
  3%|▎         | 15/446 [00:56<20:45,  2.89s/it]
  4%|▎         | 16/446 [00:59<20:45,  2.90s/it]
...
t: 100%|██████████| 445/447 [01:14<00:00,  6.70it/s, now=None]
t: 100%|██████████| 446/447 [01:14<00:00,  6.82it/s, now=None]
t: 100%|██████████| 447/447 [01:14<00:00,  6.69it/s, now=None]

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Features extracted from source image

Target video: Every pixel in the target image is a canvas awaiting the brush of manipulation, where subtlety meets strategy in the art of visual deception.



Result:



4.2 DEEP FAKE DETECTION

4.2.1 Data set description and analysis:

Tool uesd:kaggle

Deepfake techniques, which present realistic AI-generated videos of people doing and saying fictional things, have the potential to have a significant impact on how people determine the legitimacy of information presented online. These content generation and modification technologies may affect the quality of public discourse and the safeguarding of human rights— especially given that deepfakes may be used maliciously as a source of misinformation, manipulation, harassment, and persuasion. Identifying manipulated media is a technically demanding and rapidly evolving challenge that requires collaborations across the entire tech industry and beyond

AWS, Facebook, Microsoft, the Partnership on AI's Media Integrity Steering Committee, and academics have come together to build the Deepfake Detection Challenge (DFDC). The goal of the challenge is to spur researchers around the world to build innovative new technologies that can help detect deepfakes and manipulated media.

Challenge participants must submit their code into a black box environment for testing. Participants will have the option to make their submission open or closed when accepting the prize. Open proposals will be eligible for challenge prizes as long as they abide by the open source licensing terms. Closed proposals will be proprietary and not be eligible to accept the prizes. Regardless of which track is chosen, all submissions will be evaluated in the same way. Results will be shown on the leaderboard.

The PAI Steering Committee has emphasized the need to ensure that all technical efforts incorporate attention to how the resulting code and products based on it can be made as accessible and useful as possible to key frontline defenders of information quality such as journalists and civic leaders around the world. The DFDC results will be a contribution to this effort and building a robust response to the emergent threat deepfakes pose globally.

4.3 Testing methodologies and results

Test case 1:web page for uploading the video

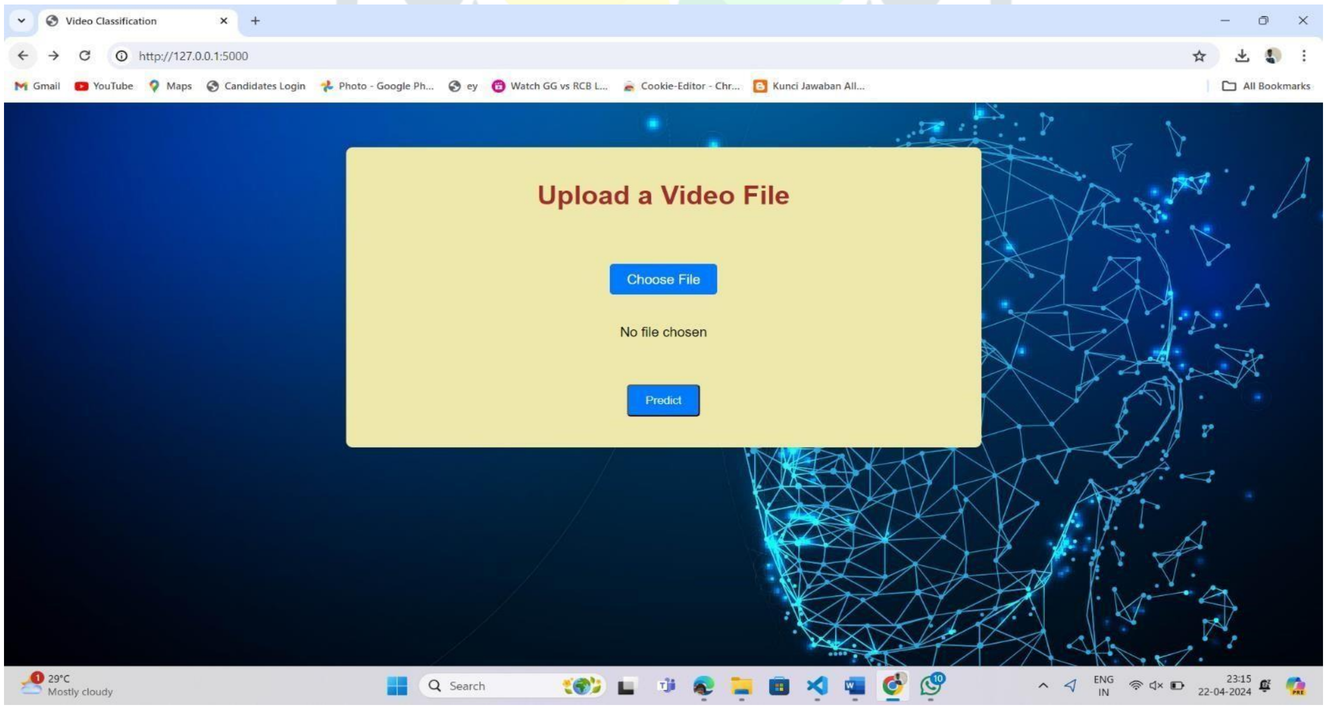


Fig Directing to Web Page

Result:Web page opened successfully

Test case 2: After uploading video

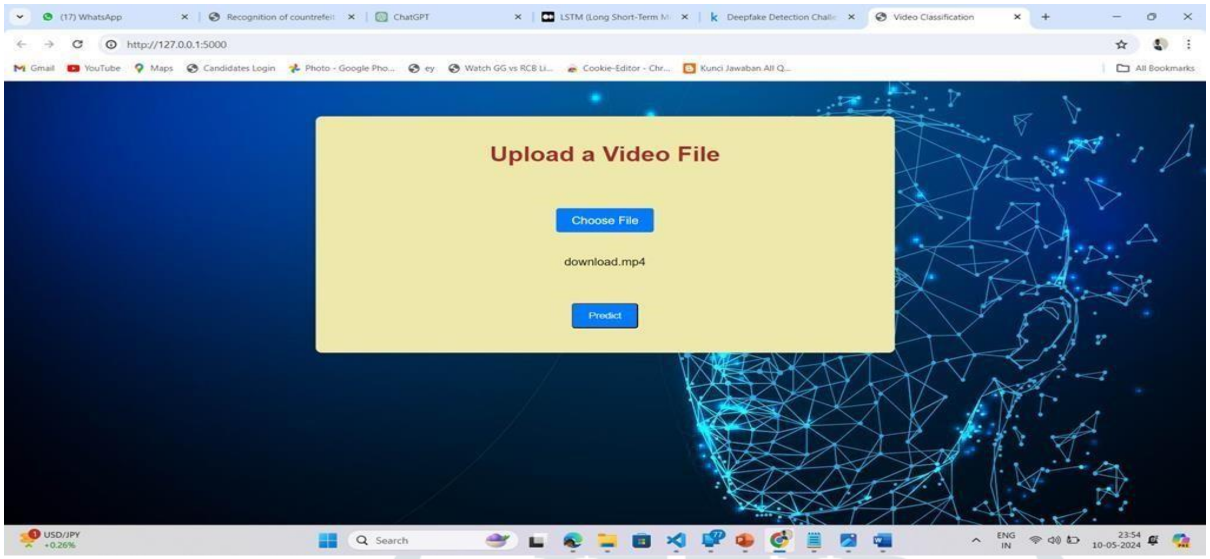
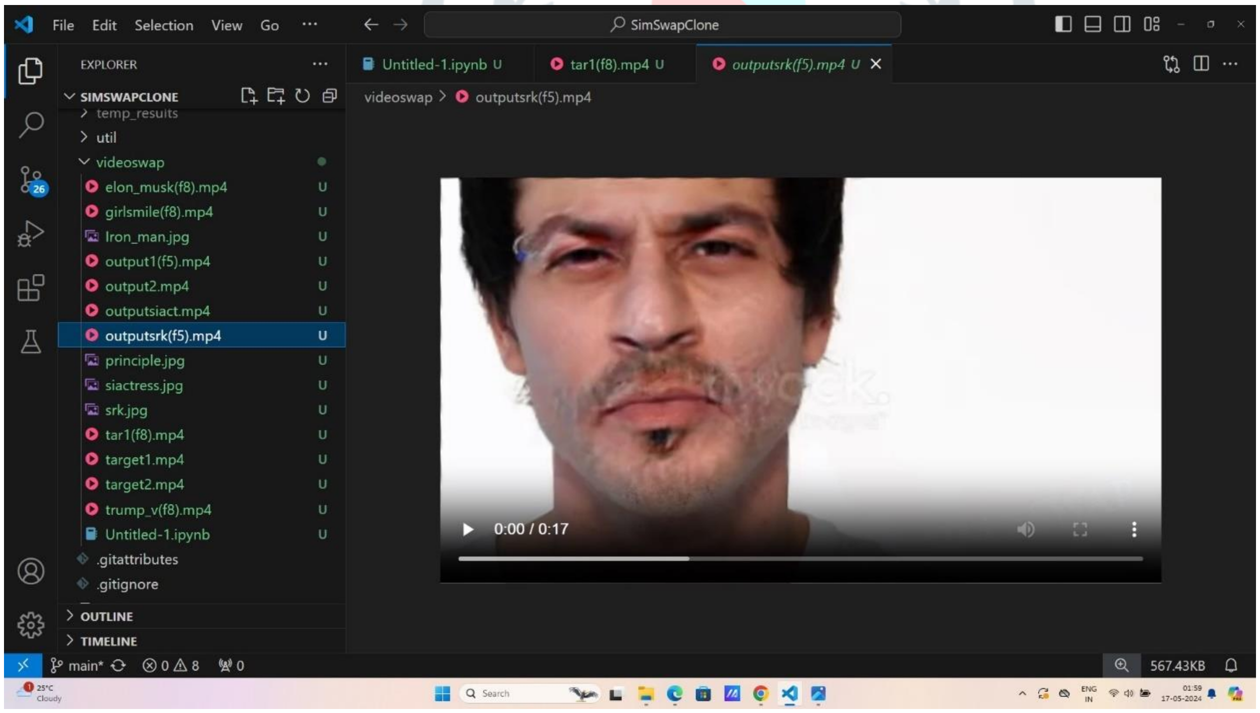
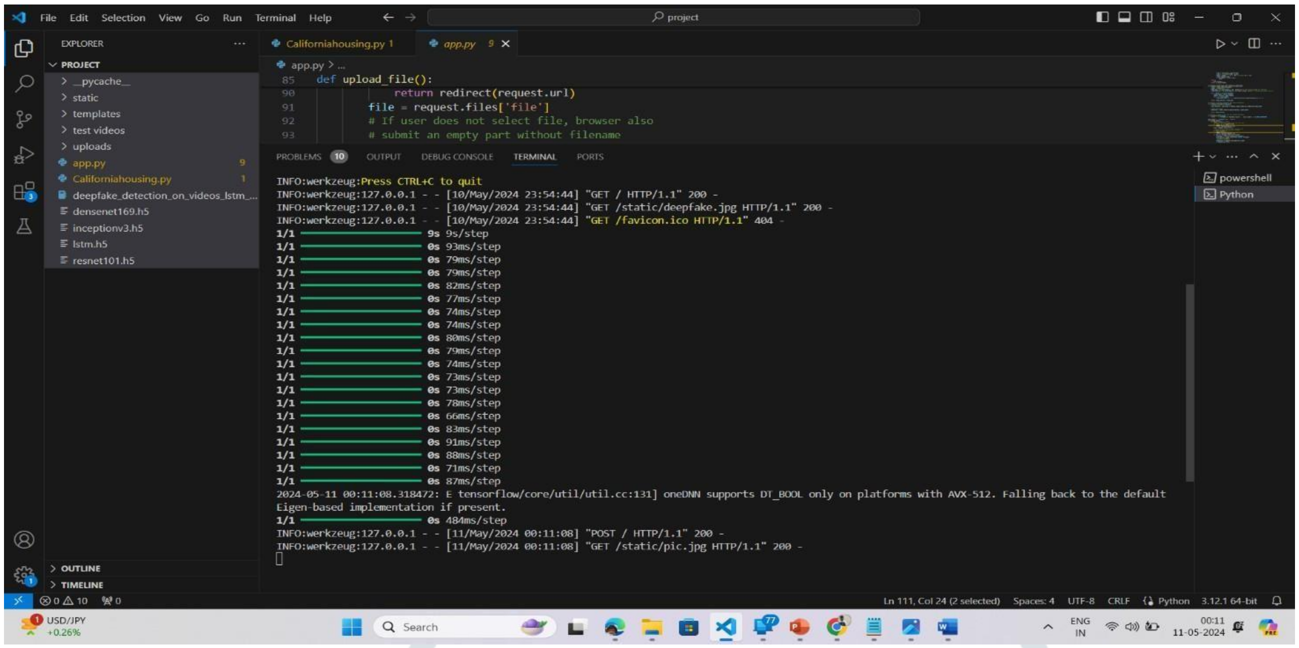


Fig : After Uploading Video

Testcase 3:Extracting Features from the uploaded video





```
def upload_file():
            return redirect(request.url)
        file = request.files['file']
        # If user does not select file, browser also
        # submit an empty part without filename
```

```
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [10/May/2024 23:54:44] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [10/May/2024 23:54:44] "GET /static/deepfake.jpg HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [10/May/2024 23:54:44] "GET /favicon.ico HTTP/1.1" 404 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 9s 9s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step
2024-05-11 00:11:08.318472: E tensorflow/core/util/util.cc:131] oneDNN supports DT_BOOL only on platforms with AVX-512. Falling back to the default
Eigen-based implementation if present.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 484ms/step
INFO:werkzeug:127.0.0.1 - - [11/May/2024 00:11:08] "POST / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [11/May/2024 00:11:08] "GET /static/pic.jpg HTTP/1.1" 200 -
```

Fig: Extracting features from video

Result :Features extracted from the uploaded video

Test case 4:Video classification

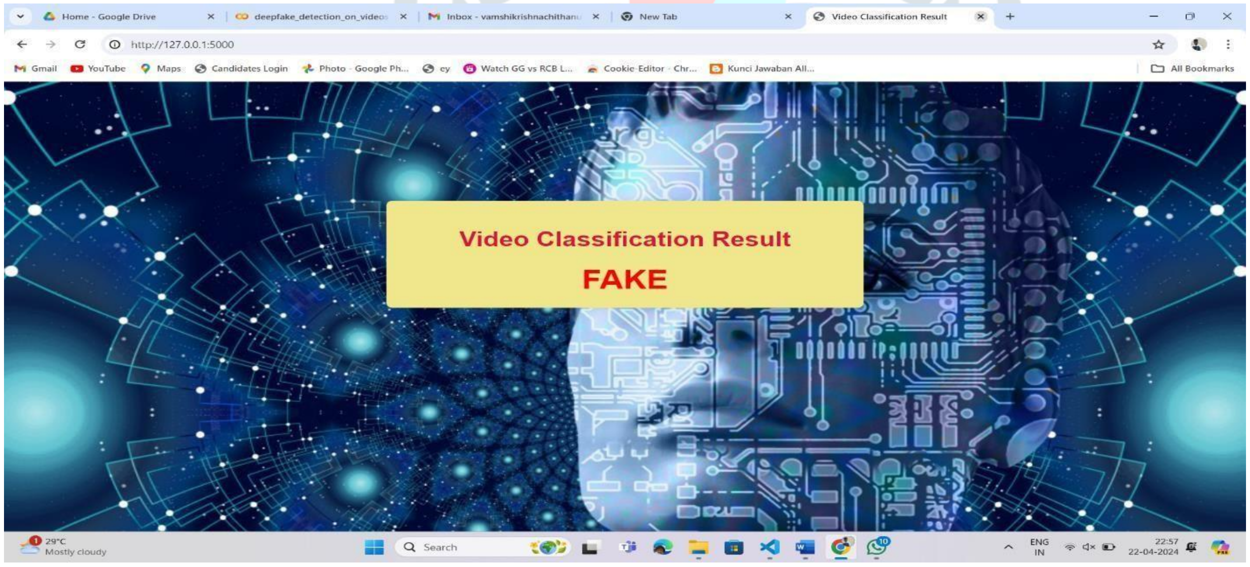
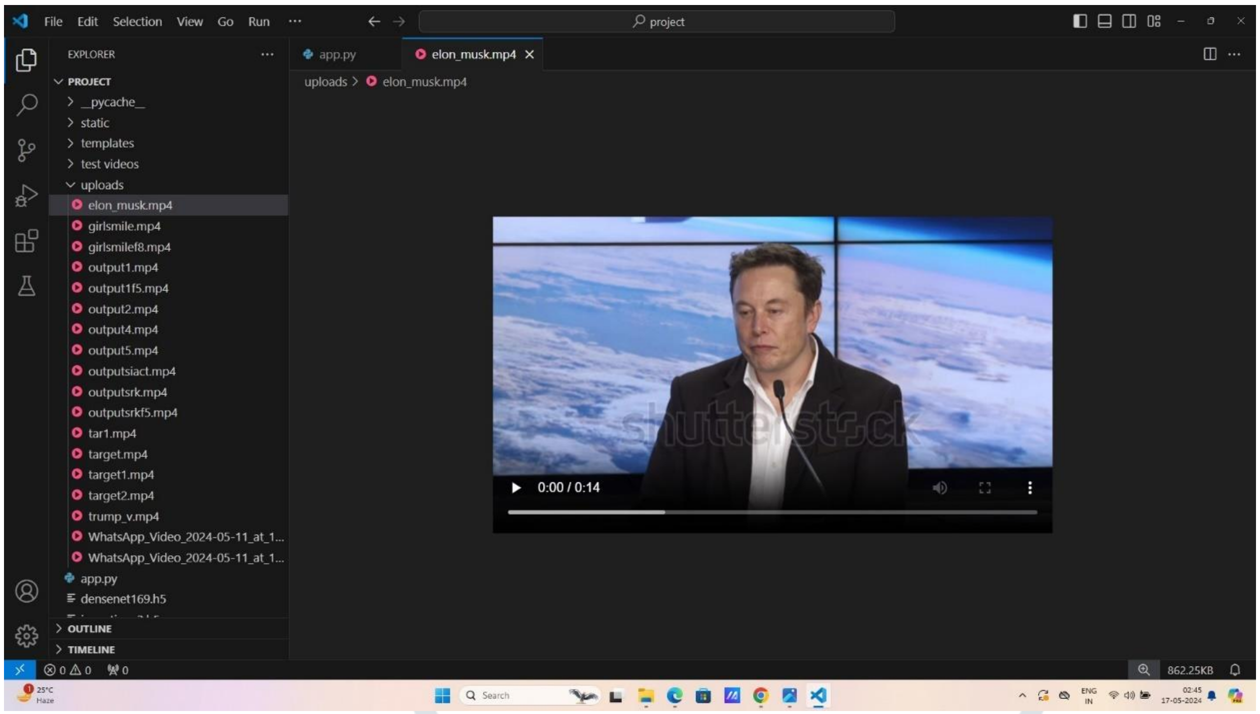


Fig video classification

Result: Uploaded video it is classified as a FAKE video

Test case 5:Video classification



Result :Features extracted from the uploaded video

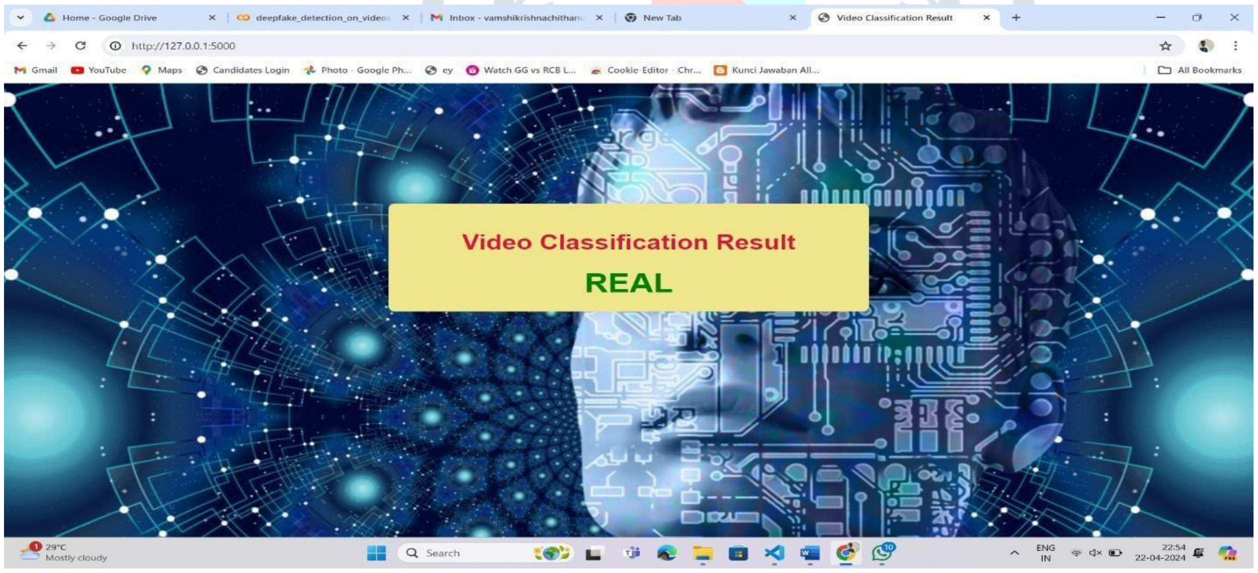


Fig video classification

Result:Uploaded video is classified as a REAL vide

4.2.3 RESULTS

Result 1: Real video

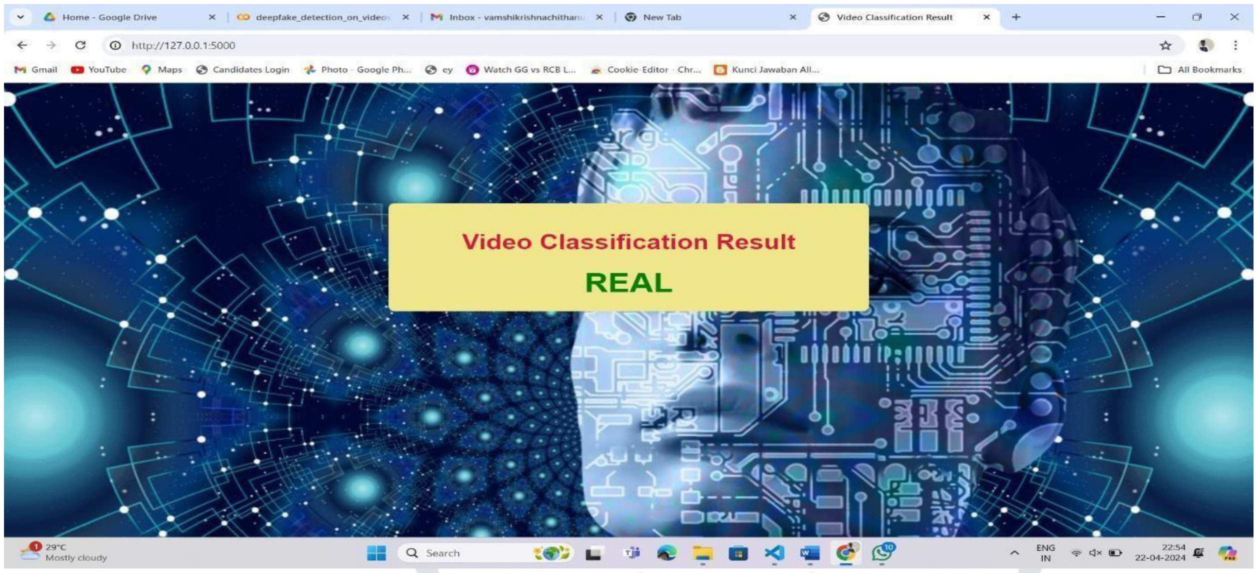


Figure :Video classification

Result 2: Fake video

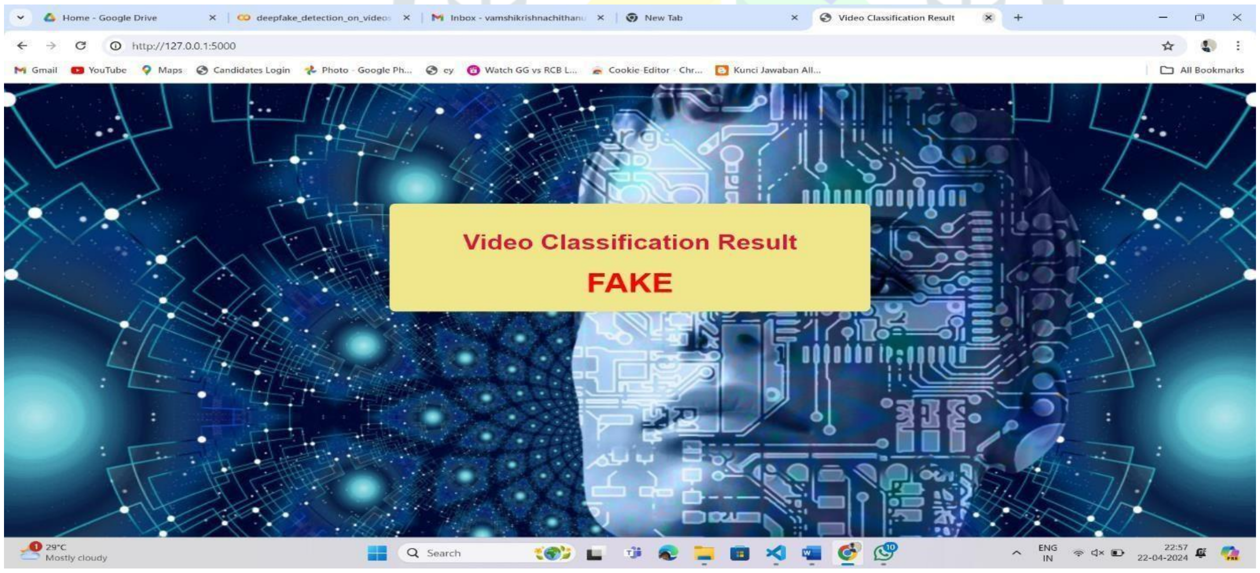


Figure Video Classifficatio

5.CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

The manipulation of images and videos through deepfaking realms represents a profound evolution in the intersection of technology and creativity, ushering in both awe-inspiring possibilities and pressing concerns. As we delve deeper into this digital landscape, it becomes increasingly apparent that the power to reshape reality comes with immense responsibility.

In the realm of creation, deepfake technology offers unprecedented avenues for artistic expression, storytelling, and entertainment. It allows creators to push the boundaries of imagination, breathing life into worlds previously confined to the realms of fiction. However, this boundless creativity also raises ethical questions about consent, authenticity, and the blurring lines between fact and fiction.

Conversely, the emergence of deepfake detection architecture underscores society's urgent need to fortify itself against the potential pitfalls of manipulation. As malicious actors exploit this technology for deception and misinformation, the imperative to develop robust safeguards becomes paramount. Yet, even as we strive to detect and mitigate the harms of deepfakes, we must remain vigilant against overreaching censorship and the suppression of legitimate expression.

Ultimately, the manipulation of images and videos using deepfaking realms presents us with a pivotal moment of reckoning. It challenges us to navigate the delicate balance between innovation and responsibility, creativity and accountability. By embracing this technology with cautious optimism and a steadfast commitment to ethical principles, we can harness its transformative potential while safeguarding the integrity of our digital discourse and the sanctity of truth itself.

5.2 FUTURE SCOPE

- The future scope of manipulation of images and videos using deepfake realms is both promising and challenging, reflecting the duality of technological advancement. Here are some key considerations:
- Advancements in Realism: As deepfake algorithms continue to improve, the fidelity and realism of manipulated content will likely reach unprecedented levels. This could revolutionize industries such as entertainment, advertising, and virtual reality, offering immersive experiences that blur the line between the real and the artificial.
- Ethical and Legal Implications: The proliferation of deepfake technology raises complex ethical and legal questions regarding privacy, consent, and intellectual property. There will be an increasing need for robust regulations and frameworks to govern the responsible use of deepfake technology and protect individuals from malicious manipulation.
- Enhanced Detection Techniques: In response to the growing threat of deepfake manipulation, there will be significant investments in developing more sophisticated detection methods and tools. These advancements may involve the integration of artificial intelligence, machine learning, and blockchain technology to verify the authenticity of digital media.

- Education and Awareness: As deepfake technology becomes more accessible, there will be a heightened need for public awareness and education initiatives to help individuals recognize and discern manipulated content. This includes media literacy programs, cybersecurity training, and campaigns to promote digital hygiene practices.
- Applications in Forensics and Security: Deepfake technology has the potential to revolutionize forensic analysis and security protocols by enabling more accurate and efficient authentication of digital evidence. Law enforcement agencies and cybersecurity firms may increasingly rely on deepfake detection tools to combat fraud, cyberattacks, and disinformation campaigns.
- Creative Expression and Innovation: Despite the risks associated with manipulation, deepfake technology also offers unprecedented opportunities for creative expression and innovation. Artists, filmmakers, and content creators may explore new storytelling techniques, visual effects, and interactive experiences that harness the power of deepfake technology in ethical and imaginative ways.

References:

1. Fake Face Detection Methods: Can They Be Generalized? Bromme, C. Busch, A. Dantcheva, C. Rathgeb, A.Uhl 28 Sept 2021

2. A novel contrast enhancement forensics based on convolutional neural networksJee- Young Sun, Seung-Wook Kim,Sang-Won Lee, Sung-JeaKo 01April 2020

3. Data Augmentation Generative Adversarial Networks Antreas Antoniou, Amos Storkey, Harrison Edwards 8 Mar 2022

4. Anatomically-aware Facial Animation from a Single Image Albert Pumarola , Antonio Agudo , Aleix [5] M. Martinez , Alberto Sanfeliu , Francesc Moreno-Noguer 28 Aug 2021

5. Convolutional Neural Network Based on Diverse Gabor Filters for Deepfake Recognition Ahmed H. Khalifa ,Nawal A. Zaher, Abdallah S. Abdallah And Mohamed Waleed Fakhr February 16, 2022

6. Oleg Alexander, Mike Rogers, William Lambeth, Jen-Yuan Chiang, Wan-Chun Ma, Chuan-Chang Wang, and Paul De bevec. The Digital Emily project: Achieving a photorealistic digital actor. IEEE Computer Graphics and Applications, 30(4):20–31, 2022

7. Manjesh R, Ninada D, Neema Jain V, Namitha M, Abhilasha H A, Autism Spectrum Disease Prediction using Deep Learning, International Journal of Advances in Engineering Architecture Science and Technology, June 2023, Volume-1, Issue-3, pp. 14-22, DOI: https://doi-ds.org/doilink/06.2023-25292924/0004052023.

8. Pushpaveni H P, Anil Kumar Yadav, Santosh Kumar Yadav, Sulav Narayan Adhikari, Detection of Skin Cancer using Neural Architecture Search with Model Quantization, International Journal of Advances in Engineering Architecture Science and Technology, May 2023, Volume-1, Issue-2, pp. 19-34, DOI: https://doids.org/doilink/06.2023-12687848/0006052023.

9. Gelli kavya, Parasuraman Harshitha, S. Sanjana, Y. L. Malathi Latha, Lung Cancer Prediction from CT Scan Images Using Dynamic Deep Neural Networks, International Journal of Advances in Engineering Architecture Science and Technology, June 2023, Volume-1, Issue-3, pp. 46-53, DOI: https://doi-ds.org/doilink/06.2023-

91952249/0018062023.

10. Deepfake Detection using Spatiotemporal Convolutional Networks. Oscar de Lima, Sean Franklin, Shreshtha Basu, Blake Karwoski, Annet George 2020.

11. .Deepfake Detection through Deep Learning. Deng Pan, Lixian Sun, Rui Wang, Xingjian Zhang, Richard O. Sinnott 2020.

12..Deepfake Video Detection Using Convolutional Neural Network. Aarti Karandikar, Vedita Deshpande, Sanjana Singh, Sayali Nagbhidkar, Saurabh Agrawal 2020.

APPENDIX:

DEEP FAKE CREATION:

Algorithm Implementation and Pseudo Code

Code for single swap:

```
import cv2

import torch

import fractions

import numpy as np

from PIL import Image

import torch.nn.functional as F

from torchvision import transforms

from models.models import create_model

from options.test_options import TestOptions

from insightface_func.face_detect_crop_single import Face_detect_crop

from util.videoswap import video_swap

import os


def lcm(a, b): return abs(a * b) / fractions.gcd(a, b) if a and b else 0


transformer = transforms.Compose([

        transforms.ToTensor(),

        #transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

    ])


transformer_Arcface = transforms.Compose([

        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

# detransformer = transforms.Compose([
#        transforms.Normalize([0, 0, 0], [1/0.229, 1/0.224, 1/0.225]),
#        transforms.Normalize([-0.485, -0.456, -0.406], [1, 1, 1])
#    ])



if __name__ == '__main__':
    opt = TestOptions().parse()

    start_epoch, epoch_iter = 1, 0
    crop_size = opt.crop_size

    torch.nn.Module.dump_patches = True
    if crop_size == 512:
        opt.which_epoch = 550000
        opt.name = '512'
        mode = 'ffhq'
    else:
        mode = 'None'
    model = create_model(opt)
    model.eval()
    app = Face_detect_crop(name='antelope', root='./insightface_func/models')
    app.prepare(ctx_id= 0, det_thresh=0.6, det_size=(640,640),mode=mode)
    with torch.no_grad():
        pic_a = opt.pic_a_path
        # img_a = Image.open(pic_a).convert('RGB')
        img_a_whole = cv2.imread(pic_a)
        img_a_align_crop, _ = app.get(img_a_whole,crop_size)
```

```
img_a_align_crop_pil=Image.fromarray(cv2.cvtColor(img_a_align_crop[0],cv2.COLOR_BGR2RGB))
        img_a = transformer_Arcface(img_a_align_crop_pil)
        img_id = img_a.view(-1, img_a.shape[0], img_a.shape[1], img_a.shape[2])
        # pic_b = opt.pic_b_path
        # img_b_whole = cv2.imread(pic_b)
        # img_b_align_crop, b_mat = app.get(img_b_whole,crop_size
        # img_b = transformer(img_b_align_crop_pil)
        # img_att = img_b.view(-1, img_b.shape[0], img_b.shape[1], img_b.shape[2])

        # convert numpy to tensor
        img_id = img_id.cuda()
        # img_att = img_att.cuda()

        #create latent id
        img_id_downsample = F.interpolate(img_id, size=(112,112))
        latend_id = model.netArc(img_id_downsample)
        latend_id = F.normalize(latend_id, p=2, dim=1)

        video_swap(opt.video_path,latend_id,model,app,opt.output_path,temp_results_dir=opt.temp_pa,\
            no_simswaplogo=opt.no_simswaplogo,use_mask=opt.use_mask,crop_size=crop_size)
```

code for mutli swap:

```
import cv2
import torch
import fractions
import numpy as np
from PIL import Image
import torch.nn.functional as F
from torchvision import transforms
from models.models import create_model
from options.test_options import TestOptions
from insightface_func.face_detect_crop_multi import Face_detect_crop
from util.videoswap import video_swap
import os
def lcm(a, b): return abs(a * b) / fractions.gcd(a, b) if a and b else 0
transformer = transforms.Compose([
        transforms.ToTensor(),
        #transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
transformer_Arcface = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

# detransformer = transforms.Compose([
#         transforms.Normalize([0, 0, 0], [1/0.229, 1/0.224, 1/0.225]),
#         transforms.Normalize([-0.485, -0.456, -0.406], [1, 1, 1])
#     ])
if __name__ == '__main__':
    opt = TestOptions().parse()

    start_epoch, epoch_iter = 1, 0
    crop_size = opt.crop_size
    torch.nn.Module.dump_patches = True
    if crop_size == 512:
        opt.which_epoch = 550000
        opt.name = '512'
        mode = 'ffhq'
    else:
        mode = 'None'
    model = create_model(opt)
    model.eval()
    app = Face_detect_crop(name='antelope', root='./insightface_func/models')
    app.prepare(ctx_id= 0, det_thresh=0.6, det_size=(640,640),mode = mode)

    with torch.no_grad():
        pic_a = opt.pic_a_path
        # img_a = Image.open(pic_a).convert('RGB')
        img_a_whole = cv2.imread(pic_a)
        img_a_align_crop, _ = app.get(img_a_whole,crop_size)
        img_a_align_crop_pil = Image.fromarray(cv2.cvtColor(img_a_align_crop[0],cv2.COLOR_BGR2RGB))
        img_a = transformer_Arcface(img_a_align_crop_pil)
        img_id = img_a.view(-1, img_a.shape[0], img_a.shape[1], img_a.shape[2])

        # pic_b = opt.pic_b_path
        # img_b_whole = cv2.imread(pic_b)
        # img_b_align_crop, b_mat = app.get(img_b_whole,crop_size)
        #img_b_align_crop_pil=Image.fromarray(cv2.cvtColor(img_b_align_crop,cv2.COLOR_BGR2RGB))
        # img_b = transformer(img_b_align_crop_pil)
        # img_att = img_b.view(-1, img_b.shape[0], img_b.shape[1], img_b.shape[2])

        # convert numpy to tensor
        img_id = img_id.cuda()
        # img_att = img_att.cuda()
        #create latent id
        img_id_downsample = F.interpolate(img_id, size=(112,112))
        latend_id = model.netArc(img_id_downsample)
        latend_id = F.normalize(latend_id, p=2, dim=1)
        video_swap(opt.video_path,latend_id,model,app,opt.output_path,temp_results_
dir=opt.temp_path,\
            no_simswaplogo=opt.no_simswaplogo,use_mask=opt.use_mask,crop_size=crop_size)
```

DEEPFAKE DETECTION

Algorithm Implementation and Pseudo Code

```
import numpy as npimport cv2
from tensorflow import keras

# Function to crop center square of a framedef
crop_center_square(frame):
    y, x = frame.shape[0:2]min_dim = min(y, x)
    start_x = (x // 2) - (min_dim // 2)start_y = (y // 2) - (min_dim
    // 2)
    return frame[start_y: start_y + min_dim, start_x: start_x +
    min_dim]

# Function to load video frames and preprocess them def
load_video(path, max_frames=0, resize=(224, 224)):
    cap = cv2.VideoCapture(path)frames = []
    try:
        while True:
            ret, frame = cap.read()if not ret:
                break
            frame = crop_center_square(frame)frame = cv2.resize(frame,
            resize)
            frame = frame[:, :, [2, 1, 0]] # Convert BGR to RGB
            frames.append(frame)
            if len(frames) == max_frames:break
    finally:
        cap.release() return np.array(frames)

# Function to build ResNet101 feature extractordef
build_feature_extractor():
    feature_extractor = keras.applications.ResNet101V2(
        weights='imagenet',
        include_top=False, pooling='avg', input_shape=(224, 224, 3)
    )
    preprocess_input = keras.applications.resnet_v2.preprocess_input

    inputs = keras.Input((224, 224, 3)) preprocessed =
    preprocess_input(inputs)
    outputs = feature_extractor(preprocessed)
    return keras.Model(inputs, outputs, name='feature_extractor')

# Extract features from video framesdef prepare_all_videos(df,
root_dir):
    num_samples = len(df) video_paths = list(df.index)
    labels = (df['label'] == 'FAKE').astype(int).values

    frame_masks = np.zeros(shape=(num_samples, 20), dtype='bool')
    frame_features = np.zeros(shape=(num_samples, 20, 2048),
dtype='float32')

    for idx, path in enumerate(video_paths):
        frames = load_video(os.path.join(root_dir, path))frames =
        frames[None, ...]

        temp_frame_mask = np.zeros(shape=(1, 20), dtype='bool')
        temp_frame_features = np.zeros(shape=(1, 20, 2048),
dtype='float32')

        for i, batch in enumerate(frames):video_length =
            batch.shape[0] length = min(20, video_length)for j in
            range(length):
                temp_frame_features[i, j, :] =
feature_extractor.predict(batch[None, j, :])
            temp_frame_mask[i, :length] = 1

        frame_features[idx, ] = temp_frame_features.squeeze()
        frame_masks[idx, ] = temp_frame_mask.squeeze()

    return (frame_features, frame_masks), labels

# Load the pre-trained ResNet101 feature extractorfeature_extractor =
build_feature_extractor()

# Split dataset into train and test sets
Train_set, Test_set = train_test_split(train_sample_metadata,
test_size=0.1, random_state=42,
stratify=train_sample_metadata['label'])

# Prepare data for training and testing
train_data, train_labels = prepare_all_videos(Train_set, "train")
test_data, test_labels = prepare_all_videos(Test_set, "test")

# Define input shapes
frame_features_input = keras.Input((20, 2048))mask_input =
keras.Input((20,), dtype='bool')
```

```
# Define the model architecture x = keras.layers.GRU(16,
return_sequences=True)(frame_features_input, mask=mask_input) x =
keras.layers.GRU(8)(x) x = keras.layers.Dropout(0.4)(x) x =
keras.layers.Dense(8, activation='relu')(x) output = keras.layers.Dense(1,
activation='sigmoid')(x)

# Compile the model model_resnet = keras.Model([frame_features_input,
mask_input], output) model_resnet.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy']) model_resnet.summary()

# Train the model checkpoint_resnet =
keras.callbacks.ModelCheckpoint('./resnet101.weights.h5',
save_weights_only=True, save_best_only=True) history_resnet = model_resnet.fit(
    [train_data[0], train_data[1]], train_labels,
    validation_data=([test_data[0], test_data[1]], test_labels),
    callbacks=[checkpoint_resnet], epochs=20, batch_size=64
)

 # Save the model model_resnet.save("resnet101.h5")
```

```
from tensorflow import keras

# Define inputs
frame_features_input = keras.Input((MAX_SEQ_LENGTH, NUM_FEATURES))
mask_input = keras.Input((MAX_SEQ_LENGTH,), dtype="bool")

# LSTM layers
x = keras.layers.LSTM(16, return_sequences=True)(frame_features_input,
mask=mask_input)
x = keras.layers.LSTM(8)(x)

# Additional layers
x = keras.layers.Dropout(0.4)(x)
x = keras.layers.Dense(8, activation="relu")(x)

# Output layer
output = keras.layers.Dense(1, activation="sigmoid")(x)

# Create the model
model_lstm = keras.Model([frame_features_input, mask_input], output)

# Compile the model model_lstm.compile(loss="binary_crossentropy",
optimizer="adam", metrics=["accuracy"])

# Display model summary model_lstm.summary()

# Model training checkpoint_lstm =
keras.callbacks.ModelCheckpoint('./model_lstm.weights.h5',
save_weights_only=True, save_best_only=True)
history_lstm = model_lstm.fit([train_data[0], train_data[1]],
    train_labels,
    validation_data=([test_data[0], test_data[1]], test_labels),
    callbacks=[checkpoint_lstm],
    epochs=EPOCHS,batch_size=BATCH_SIZE
)
```

Code for deepfake detection:

from flask import Flask, flash, render_template, request, redirect, url_for

from werkzeug.utils import secure_filename

import os

import numpy as np

import cv2

from tensorflow import keras

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

app = Flask(__name__)

# Define paths
UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'mp4'}

# Load the pre-trained DenseNet169 model
model = keras.models.load_model("densenet169.h5")

# Load the pre-trained feature extractor (DenseNet169)
feature_extractor = keras.applications.DenseNet169(
    weights='imagenet',
    include_top=False,
    pooling='avg',
    input_shape=(224, 224, 3)
)
# Function to crop center square of a frame
def crop_center_square(frame):
    y, x = frame.shape[0:2]
    min_dim = min(y, x)
    start_x = (x // 2) - (min_dim // 2)
    start_y = (y // 2) - (min_dim // 2)
    return frame[start_y: start_y + min_dim, start_x: start_x + min_dim]

# Function to load video frames and preprocess them
def load_video(path, max_frames=0, resize=(224, 224)):
    cap = cv2.VideoCapture(path)
    frames = []
    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break
            frame = crop_center_square(frame)
            frame = cv2.resize(frame, resize)
            frame = frame[:, :, [2, 1, 0]]  # Convert BGR to RGB
            frames.append(frame)
            if len(frames) == max_frames:
                break
    finally:
        cap.release()
    return np.array(frames)

# Function to prepare a single video for prediction
def prepare_single_video_for_prediction(video_path):
    frames = load_video(video_path)
    frames = frames[None, ...]
    frame_mask = np.ones(shape=(1, 20), dtype='bool') # No masking needed for inference
    frame_features = np.zeros(shape=(1, 20, 1664), dtype='float32') # DenseNet169 output features

    for i, batch in enumerate(frames):
        video_length = batch.shape[0]
        length = min(20, video_length)
        for j in range(length):
            frame_features[i, j, :] = feature_extractor.predict(batch[None, j, :])

    return frame_features, frame_mask
```

```
# Function to perform inference and return prediction
def predict_with_model(video_path):
    # Prepare inputs for prediction
    input_features, input_mask = prepare_single_video_for_prediction(video_path)

    # Make predictions
    prediction = model.predict([input_features, input_mask])

    return prediction[0]

# Function to check if the file has allowed extension
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        # Check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # If user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file_path = os.path.join(UPLOAD_FOLDER, filename)
            file.save(file_path)
            # Perform prediction
            prediction = predict_with_model(file_path)
            if prediction >=0.5:
                result = "FAKE"
            else:
                result = "REAL"
            return render_template('result.html', result=result)
    return render_template('index.html')


if __name__ == '__main__':
    app.run(debug=True)
```