

1. INTRODUCTION

The recent growth of technology in computer-generated editing programs has made synthesizing and modifying media content easier than ever. The potential for misinformation spread has exploded, especially with the phenomenon known as Deepfake. Deepfake is a technology that uses deep learning to create fake videos, alter existing videos, or even synthesize the speech of someone's voice. This makes it a dangerous tool in malicious applications of spreading fake news and disseminating false or dangerous information. Today, many Deepfake tools are free, open-source, and have many learning resources. The most available are Faceswap, Faceswap-GAN, DeepFaceLab, and DFaker. These tools swap the source person's face with the target face to create a new video with the same action as the source but with the target person's face. The products of these convenient services are notably difficult for humans to distinguish between real and fake. Therefore, developing digital forensic machine learning models and algorithms to detect fakes is more critical for digital security than ever. The two main techniques of Deepfake technology are generation and detection, both of which utilize deep learning. Many models in literature can generate fake videos.

Addressing the challenges posed by Deepfake technology requires a multifaceted approach. While the creation and dissemination of Deepfake content continue to evolve, efforts to develop robust digital forensic tools for detecting fakes have become increasingly critical. These tools leverage machine learning models and algorithms designed to analyze videos and audio for signs of manipulation or synthetic generation. The field of Deepfake detection encompasses a range of techniques, from analyzing facial and vocal cues to identifying anomalies in video and audio data. Researchers have developed a variety of detection models, each with its strengths and limitations. These models often rely on training data composed of both authentic and synthetic media content, allowing them to learn to distinguish between real and fake instances.

Despite the progress made in Deepfake detection, the arms race between creators and detectors continues unabated. As Deepfake technology advances, so too must the capabilities of detection algorithms evolve to keep pace. Moreover, the ethical implications of Deepfake technology raise complex questions surrounding privacy, consent, and the manipulation of digital content. Originally, Deepfake techniques primarily focused on generating realistic-looking facial swaps in videos. However, as the technology has matured, it now encompasses a broader range of applications, including voice synthesis, body movement replication, and even the creation of entirely fabricated scenes. Voice synthesis, in particular, has seen significant progress within the

realm of Deepfake technology. Using deep learning algorithms, it is now possible to generate synthetic speech that closely mimics the cadence, tone, and intonation of a specific individual. This capability has profound implications for impersonation and social engineering, as it allows malicious actors to create audio recordings that sound convincingly like a targeted individual. Furthermore, recent developments in Deepfake technology have expanded beyond mere face-swapping to include full-body manipulation and scene generation. By leveraging generative adversarial networks (GANs) and other deep learning techniques, researchers have demonstrated the ability to create entirely fabricated videos that depict individuals engaging in actions and scenarios that never occurred. These advancements blur the line between reality and fiction, raising concerns about the potential for widespread deception and manipulation.

For instance, Choi et al. [7] proposed a Star-GAN-based deepfake method using a generative adversarial network. The architecture of the method was composed of an encoder, decoder, and discriminator. The encoder consisted of $8 \times 8 \times 512$ output and $64 \times 64 \times 3$ input, and a series of output tensors followed by a self-attention block, a 2D convolution, and up-scaling blocks where activation of leaky ReLU occurs. The main shortcoming of the approach was overreliance swapping on the mouth, nose, and eyes. The look-alike targets were also obtained by averaging features map input, which generated some artefacts making the blending imperfect. This also leads to blurriness in the skin, appearing unnatural. Karras et al. [8] proposed a ProGAN-based method with a generative adversarial network. The generator and discriminator were designed to grow progressively. This speeds the training up and significantly stabilizes, allowing the production of an image with unprecedented quality. However, the poor image micro-structure compared to the results by other state-of-the-art techniques make the method seek some improvement in future research. Karras et al. [9] proposed a styleGAN based on style transfer literature. The proposed architecture yields automated learning, reducing image quality variation, using exponential in the architecture to average pixel values from logarithmic to linear for easy identification results in images with poor features. This is due to the exponential decay of the pixels resulting from the loss of some pixels. Siarohin et al. [10] proposed a first-order motion model for image animation where an object in a source image is not animated based on the motion of a driving video. However, these come with a price; that is, the transfer only allows global object geometry; thus, fine image details are lost.

The accuracy of existing techniques in detecting deepfake videos and images is average 90%. However, this has further reduced to 70% with the introduction of facemasks during the Covid-19 pandemic. This has motivated criminals to use facemasks to elude and cover their criminal activities by editing surveillance videos to escape the criminal justice system. The

proposed system uses algorithms like Inception Resnet v2, VGG19, Convolution Neural Network, Xception with preprocessing stages, feature-based, residual connection, and batch normalization. Among them Xception model increase the detection accuracy of deepfake videos in the presence of facemasks, unlike the traditional methods.

1.1 Motivation

The rise of deepfake technology has ushered in a new era of digital manipulation, raising significant societal concerns. This phenomenon allows for the creation of fabricated images and videos with convincingly replaced or synthesized faces, posing risks such as the spread of misinformation, manipulation of public opinion, and erosion of trust in media sources. The malicious potential of deepfake technology extends to various domains, including the fabrication of electronic evidence, perpetration of digital harassment and fraud, and dissemination of false political news. As society grapples with these challenges, the advent of face masks during the COVID-19 pandemic has added a layer of complexity to the detection and mitigation of deepfake videos. To address this evolving threat, the proposed system introduces a novel approach for detecting deep-fake videos featuring individuals wearing face masks. The motivation behind this endeavor is multifaceted, driven by the pressing need to mitigate the harmful effects of misinformation and deception facilitated by deepfake technology. By using various detection algorithms tailored to deep-fake videos with face masks, this project aims to address the unique challenges posed by the widespread adoption of face coverings during the pandemic.

The proposed system seeks to mitigate the spread of misinformation and deception by enhancing the accuracy of deep-fake detection algorithms. Deepfake technology has the potential to undermine trust in legitimate sources of news and information, manipulate public opinion, and exacerbate societal divisions. By using robust detection algorithms capable of identifying deep-fake videos, particularly those featuring individuals wearing face masks, this can help safeguard against the misuse of this technology for unusual activity purposes. This endeavor represents a critical step in protecting individuals, organizations, and institutions from the harmful effects of deepfake manipulation. The main motive to safeguard the integrity of digital content, protect individuals and organizations from the consequences of manipulated information, and preserve the trust that society places in media and communication channels. As technology evolves, the commitment to advancing digital forensic capabilities becomes crucial in maintaining a secure and reliable digital environment.

1.2 Problem Definition

The rapid growth of Deepfake technology poses a significant threat to the authenticity of digital media, leading to an increased risk of misinformation and malicious content dissemination. The availability of user-friendly, open-source Deepfake tools makes it challenging to distinguish between real and manipulated content, creating a pressing need for robust detection mechanisms. The primary problem addressed is the vulnerability of digital platforms to the harmful impact of Deepfake content. The difficulty in distinguishing between authentic and manipulated media poses risks to individuals, organizations, and society at large. Therefore, the project aims to address this problem by using different digital forensic machine learning models and algorithms specifically designed for effective Deepfake detection, contributing to the overall enhancement of digital security measures in the face of evolving synthetic media technologies.

1.3 Objective

The objective of the proposed system is to use robust and efficient digital forensic machine learning models and algorithms for the detection of Deepfake content. With the proliferation of technology enabling the creation of highly convincing synthetic media through Deepfake tools, there is an urgent need to enhance digital security measures. The main aim of the proposed system is to use the best accurate advanced detection algorithms that can effectively identify manipulated videos, altered images, and synthesized voice recordings. By leveraging deep learning methodologies, the objective is to create a reliable and scalable solution capable of discerning between authentic and manipulated media, mitigating the potential harm caused by the spread of misinformation and fake content. This system also intends to explore novel approaches to stay ahead of evolving Deepfake techniques, ensuring the continued effectiveness of detection mechanisms.

2. LITERATURE SURVEY

Firas Husham Almkhtar's [1] system employs AI, AI programming, and a computer to generate and identify deep fake videos, incorporating Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). However, it lacks technical specifics on AI programming and methodology. Additionally, there is a notable absence of discussions on practical challenges and limitations. Comparing to our system the researchers system exhibits promise in detecting deep fakes through a diverse approach, and its practicality need the more technical depth, real-world insights, and comprehensive evaluation scenarios.

S. R. Ahmed et al., [2] investigate Deepfake technology in their paper, emphasizing the necessity for automated detection methods to mitigate privacy and security risks. While the paper underscores the significance of developing robust detection systems and provides insightful historical and conceptual perspectives, but it lacks specific technical details on proposed methods. Additionally, it may not encompass the latest advancements in Deepfake technology and detection techniques.

The paper by Juan Hu, et al., [3] introduces a two-stream approach for identifying compressed Deepfake videos, showcasing enhanced detection accuracy through frame-level and temporality-level feature analysis. However, the lack of technical details and discussion on real-world challenges raises concerns about practical implementation. While the two-stream method holds promise for improved detection, a more thorough exploration of its intricacies and potential limitations is crucial for ensuring its effectiveness in real-world applications.

Davide Coccomini, et al., [4] present a study on video deep fake detection, employing Vision Transformers and an EfficientNet B0 as a feature extractor. Despite achieving competitive results without distillation or ensemble methods, the paper lacks in-depth technical details. Furthermore, it overlooks potential challenges and limitations in real-world scenarios. Nevertheless, the successful fusion of Vision Transformers and EfficientNet B0 for video deep fake detection holds promise for accurate outcomes, demonstrating efficacy without resorting to intricate methodologies.

Md Shohel Rana, et.al.'s [5] systematic literature review on Deepfake detection categorizes methods into deep learning, classical machine learning, statistics, and blockchain. While evaluating performance and highlighting deep learning's superiority, the paper lacks a thorough discussion of practical implementation challenges and real-world issues. Additionally, it may not capture the latest advancements in Deepfake technology. Despite this, the review offers a valuable overview, emphasizing the need for addressing ethical considerations and practical challenges in detecting malicious Deepfakes.

S. Bommareddy, et.al.'s [6] developed a model in response to the escalating threat of disseminating inaccurate information, the utilization of deep convolutional neural network (CNN) techniques has led to a surge in the proliferation of manipulated images. Addressing this challenge necessitates the development of a reliable and efficient mechanism for detecting false photographs. Their study tackles the issue by employing the V4D architecture to scrutinize manipulated videos, with a specific focus on identifying DeepFake videos within three distinct contexts: (I) recognizing all types of manipulations, (II) pinpointing single manipulations, and (III) detecting cross manipulation to assess the authenticity of videos. The investigation involves the application of diverse manipulation techniques, complemented by the creation of algorithms designed to categorize previously unidentified manipulation methods. The paper lacks while providing valuable insights into combating the manipulation of visual content, the study may benefit from a more comprehensive discussion on practical implementation challenges and an exploration of real-world issues associated with its proposed solutions. Additionally, it is important to acknowledge the dynamic nature of Deepfake technology, and the study's findings may need continual validation against the latest advancements in this rapidly evolving field.

3. ANALYSIS

3.1 Existing System

In literature many researchers introduced a two-stream DNN method by analyzing the frame-level and temporality-level of compressed Deepfake videos. Since the video compression brings lots of redundant information to frames, the proposed frame-level stream gradually prunes the network to prevent the model from fitting the compression noise. And they apply a temporality-level stream to extract temporal correlation features. Another related work, they introduced a Convolutional Vision Transformer for the detection of Deepfakes. It has two components: Convolutional Neural Network (CNN) and Vision Transformer (ViT), in which CNN extracts learnable features while the ViT takes in the learned features as input and categorizes them using an attention mechanism.

3.2 Proposed System

Addressing the escalating societal concern surrounding deepfakes, the system adopts innovative methodologies to enhance the accuracy of deepfake video detection, particularly in scenarios involving face masks. A notable contribution is the introduction of the Deepfake Face Mask Dataset (DFFMD), which fills a critical gap left by conventional methods in tackling the challenges posed by deepfakes amidst the widespread use of face masks.

This leverages a novel Inception-ResNet-v2 approach, augmented with preprocessing stages, feature-based analysis, residual connections, and batch normalization. These elements collectively contribute to a significant improvement in the precision and effectiveness of deepfake detection within the context of face masks, showcasing a sophisticated and thorough methodology. This delves into various deep learning models, including VGG19, CNN, and an extended Xception architecture. This comprehensive exploration of diverse techniques not only demonstrates the versatility of the approach but also establishes a robust foundation for advanced deepfake detection. By examining multiple models, this system provides insights into the strengths and limitations of different methodologies, offering a nuanced understanding of the complexities associated with combating deepfake threats.

In an era dominated by evolving digital challenges, the multifaceted approach presented in this system responds to the dynamic nature of deepfake technology. By contributing to the development of effective detection mechanisms, this plays a crucial role in mitigating the adverse impacts of falsified content, thereby safeguarding the integrity of digital information in contemporary society.

3.3 System Requirement Specification

Software Requirements

- 1) Software: Anaconda
- 2) Primary Language: Python
- 3) Frontend Framework: Flask
- 4) back-end Framework: Jupyter Notebook
- 5) Database: Sqlite3
- 6) Front-End Technologies: HTML, CSS, JavaScript

Hardware Requirements

- 1) Operating System: Windows Only
- 2) Processor: i5 and above
- 3) Ram: 8gb and above
- 4) Hard Disk: 25 GB in local drive

High processing machine for large data set, and libraries such as tensorflow, keras, cv2, numpy, matplotlib are utilized.

3.3.1 Purpose

The purpose of this project is to use advanced digital forensic machine learning models and algorithms with a primary focus on detecting Deepfake content. Given the alarming rise of technology facilitating the creation and manipulation of media through tools like Faceswap, Faceswap-GAN, DeepFaceLab, and DFaker, the potential for malicious dissemination of misinformation is at an unprecedented high. This project aims to address the urgent need for robust detection techniques in the face of increasingly convincing Deepfake videos. By leveraging deep learning methods, the goal is to use sophisticated algorithms capable of accurately distinguishing between authentic and manipulated content, thereby enhancing digital security and combating the spread of fake news.

3.3.2 Scope

The project aims to address the escalating threat of Deepfake technology by using advanced digital forensic machine learning models and algorithms for the detection of manipulated media content. Focusing on both video and voice synthesis, the scope involves researching and implementing cutting-edge techniques to distinguish authentic from manipulated content. The project will explore the vulnerabilities of popular Deepfake tools such as Faceswap, Faceswap-GAN, DeepFaceLab, and DFaker. By enhancing detection capabilities through deep learning, the goal is to contribute to digital security efforts, safeguarding against the dissemination of fake news and misinformation in an era dominated by easily accessible and increasingly sophisticated media manipulation tools.

3.3.3 Overall Description

The rapid advancement of technology, particularly in computer-generated editing programs, has ushered in an era where synthesizing and modifying media content has become more accessible than ever before. This has given rise to a concerning phenomenon known as Deepfake, a technology that utilizes deep learning to create deceptive videos, alter existing ones, and even synthesize realistic speech, posing a significant threat to the spread of misinformation. Deepfake tools have proliferated, with many of them being freely available and supported by extensive learning resources. Notable examples include Faceswap, Faceswap-GAN, DeepFaceLab, and DFaker. These tools excel at swapping the face of a source person onto a target person, producing videos that closely mimic the actions of the source but feature the appearance of the target. The results are often highly convincing, making it challenging for humans to discern between authentic and manipulated content. As a consequence, this project aims to use the best and accurate machine learning models and algorithms to detect deepfakes that has become more critical than ever for ensuring digital security.

These models primarily focus on two key techniques within the realm of Deepfake technology: generation and detection. The generation aspect involves creating realistic fake videos using deep learning, while the detection aspect centers around developing algorithms capable of identifying such deceptive content. Numerous models documented in the literature specialize in generating fake videos, showcasing the sophistication and diversity within the realm of deep learning techniques applied to this technology. However, the simultaneous advancement of detection methods is crucial to counter the potential misuse of deepfakes for spreading fake news or disseminating false and dangerous information. As technology continues to evolve, the battle between the creation and detection of deepfakes is poised to shape the landscape of digital security and trust in media content.

4. DESIGN

Designing a Deepfake Face Mask Dataset (DFFMD) and developing a detection model using the Inception-ResNet-v2 architecture with integration of Convolutional Neural Networks (CNN) and Xception is comprehensive. Here's a breakdown of each step:

1. Data Collection:

Collect a diverse dataset of both real and deepfake videos for training and testing purposes. Annotate the dataset to identify ground truth labels (real or deepfake).

2. Preprocessing:

Extract relevant features from video frames, considering facial landmarks, lip synchronization, and other visual cues. Normalize and standardize the dataset to ensure consistency.

3. Deepfake Detection Model:

Choose a suitable deep learning architecture for detecting deepfake content. Experiment with pre-trained models or design a custom architecture optimized for the task. Implement data augmentation techniques to enhance the model's robustness.

4. Training:

Split the dataset into training, validation, and testing sets. Train the detection model on the training set and fine-tune hyperparameters using the validation set. Monitor and address overfitting issues.

5. Evaluation:

Evaluate the performance of the trained model on the testing set using metrics such as accuracy, precision, recall, and F1 score. Conduct cross-validation to ensure the model's generalization across different datasets.

6. Model Interpretability:

Implement techniques for interpreting the model's decisions, providing insights into the features contributing to deepfake detection.

4.1 SYSTEM ARCHITECTURE

System architecture refers to the high-level structure of a software or hardware system, outlining its components, interactions, and functionalities. It provides a blueprint for designing, implementing, and managing complex systems, ensuring that they meet performance, scalability, and reliability requirements. At its core, system architecture defines the organization of a system's components, including modules, layers, and subsystems, and their relationships with one another. It outlines the flow of data and control within the system, as well as the interfaces through which components interact. This abstraction helps manage system complexity and facilitates communication among stakeholders involved in system development and maintenance. A well-designed system architecture considers various factors, including functional requirements, non-functional requirements (such as performance, security, and usability), and constraints (such as budget, time, and technology limitations). It involves making decisions about the selection of technologies, platforms, and design patterns to achieve the desired system behavior and characteristics.

The development of a robust system architecture for deepfake face mask detection is essential in combating the proliferation of manipulated videos, particularly in the context of face coverings. This project presents a comprehensive system architecture designed to detect deepfake videos featuring individuals wearing face masks. The architecture encompasses data collection, exploratory data analysis (EDA), model training, testing, providing a structured approach to deepfake detection.

Data Collection: Data collection serves as the cornerstone of any machine learning endeavor, including in deepfake face mask detection systems. It encompasses the acquisition of relevant data from diverse sources to train, validate, and test the models utilized for detection. This initial phase is crucial as the quality and diversity of the dataset directly impact the performance and effectiveness of the subsequent detection system. The process begins with the procurement of a comprehensive dataset containing authentic videos, deepfake videos, and videos featuring individuals wearing face masks. These videos should cover a wide range of scenarios, lighting conditions, facial expressions, and demographics to ensure the robustness and generalizability of the trained models. Furthermore, ensuring the quality of the dataset is paramount to the success of the detection system. Data collection involves rigorous quality control measures such as data cleaning and validation to eliminate inconsistencies, biases, or irrelevant samples that may adversely affect the model's performance. This includes verifying the authenticity and relevance of the videos, as well as addressing issues related to dataset imbalance and bias. Imbalanced

datasets, where one class significantly outweighs the other, can lead to skewed model predictions. Thus, strategies like oversampling, undersampling, or synthetic data generation may be employed to mitigate these challenges and ensure balanced representation across classes. Ethical considerations also play a pivotal role in the data collection process, especially concerning deepfake technology. It is imperative to obtain consent from individuals featured in the videos and adhere to privacy regulations and guidelines. Additionally, measures should be implemented to prevent the dissemination of harmful or sensitive content during the data collection process. Moreover, dataset annotation and labeling are essential components of data collection, facilitating supervised learning and model evaluation. Annotating the videos with relevant metadata, such as ground truth labels indicating whether a video is real or manipulated, enables the models to learn from labeled examples and improve detection accuracy. The deep fake face mask detection dataset, sourced from Kaggle, comprises two main folders: "fake" and "real," each containing distinct videos for training and evaluation purposes. The "fake" folder encompasses a total of 1000 videos, specifically curated to simulate instances where facial features are manipulated through the application of deep fake technology, particularly the addition of face masks. On the other hand, the "real" folder consists of 836 videos, capturing authentic facial expressions without the influence of deep fake alterations.

Exploratory Data Analysis (EDA): Exploratory Data Analysis (EDA) plays a crucial role in the development of deepfake face mask detection systems, providing insights into the characteristics and distribution of the dataset. This initial phase involves the exploration and visualization of the data to uncover patterns, anomalies, and potential biases that may influence the subsequent steps in the development process. By gaining a deeper understanding of the dataset through EDA, developers can make informed decisions regarding preprocessing, feature selection, and model training. During EDA, various statistical and visualization techniques are employed to analyze the dataset's attributes and distributions. Descriptive statistics, such as mean, median, standard deviation, and quartiles, provide summary measures that characterize the central tendency and variability of the data. Histograms, box plots, and scatter plots are used to visualize the distributions of numerical variables and identify potential outliers or anomalies. Moreover, EDA allows for the exploration of relationships and correlations between different variables in the dataset. Correlation matrices and heatmaps provide insights into the strength and direction of relationships between numerical variables, while cross-tabulations and pivot tables facilitate the analysis of categorical variables. By examining these relationships, developers can identify important features and patterns that may be relevant for deepfake detection. Additionally, EDA helps uncover potential biases or limitations in the dataset that may impact model performance. Disparities in sample sizes across different classes,

imbalances in demographic representation, or inconsistencies in data collection methods are examples of biases that may need to be addressed during preprocessing or modeling. By identifying and addressing these biases early in the development process, developers can ensure the fairness, reliability, and generalizability of the trained models. EDA provides valuable insights into the distribution of deepfake and authentic videos across different classes and subgroups. Understanding the prevalence and distribution of manipulated videos with face masks compared to authentic videos helps inform model training and evaluation strategies. By examining the distribution of videos across different classes, developers can determine the optimal sampling strategies, evaluation metrics, and model evaluation techniques.

Model Training: Model training is a pivotal phase in the deepfake face mask detection systems, where collected data undergoes extensive learning processes with various models to discern patterns and characteristics indicative of manipulated content. Among the models utilized for this purpose are Inception ResNet-v2, VGG19, Convolutional Neural Networks (CNN), and Xception, each offering unique architectures and capabilities in discerning subtle differences between authentic and manipulated videos.

Inception ResNet-v2, a fusion of Google's Inception architecture and residual connections, is renowned for its depth and efficiency in feature extraction. During training, Inception ResNet-v2 scrutinizes the intricate details of facial features and mask placements, leveraging its deep layers to capture nuanced patterns indicative of deepfake manipulation. Its ability to identify subtle variations in facial expressions and movements enables it to distinguish between genuine and manipulated videos with remarkable accuracy.

VGG19, on the other hand, is celebrated for its simplicity and effectiveness in image classification tasks. With a straightforward architecture comprising multiple convolutional layers, VGG19 excels in discerning global patterns and features within images. During training, VGG19 analyzes the spatial relationships between facial features and mask textures, utilizing its hierarchical structure to extract hierarchical representations that facilitate the detection of deepfake anomalies.

Convolutional Neural Networks (CNN) represent a versatile class of deep learning models widely employed in computer vision tasks. Leveraging a series of convolutional and pooling layers, CNNs excel in extracting spatial hierarchies of features from input images. During training, CNNs scrutinize facial features and mask placements, learning discriminative representations that enable them to differentiate between authentic and manipulated videos with high accuracy.

Xception, inspired by the Inception architecture, introduces depthwise separable convolutions to enhance feature learning and reduce computational complexity. During training, Xception dissects

facial features and mask textures with precision, leveraging its depthwise separable convolutions to capture intricate details while minimizing computational overhead. Its ability to extract informative representations from input images enables it to discern subtle discrepancies indicative of deepfake manipulation. Model training with Inception ResNet-v2, VGG19, CNN, and Xception plays a pivotal role in the development of deepfake face mask detection systems. Each model brings unique strengths and capabilities to the table, leveraging its architecture and learning mechanisms to discern patterns and anomalies indicative of manipulated content. Through extensive training with diverse datasets, these models equip detection systems with the ability to accurately differentiate between genuine and manipulated videos, safeguarding against the spread of misinformation and deception in digital media.

Model Testing: The system architecture for testing the deepfake detection is designed to provide a comprehensive and user-friendly experience, while ensuring accurate and efficient analysis of uploaded videos.

User Interface:

- The system begins with a user interface accessible through a web application or desktop software.
- Upon accessing the application, users are presented with a home page that provides navigation options.

User Authentication:

User authentication on signin and signup pages is a fundamental aspect of the deepfake detection system's security and usability. Here's a detailed overview of how user authentication is implemented on these pages:

- **Sign-up Page:** The sign-up page allows new users to create an account and gain access to the system's functionalities.
 - Users are typically prompted to provide necessary information such as:
 - Username: A unique identifier chosen by the user.
 - Email Address: Used for communication and account verification purposes.
 - Password: A securely chosen password that meets specified criteria (e.g., minimum length, complexity requirements).
 - Additional information may be requested depending on system requirements or preferences, such as full name, date of birth, or security questions for account recovery. Upon submission of the sign-up form, the system validates the provided information, ensuring that all required fields

are filled out correctly and that the username and email address are unique and meet validation criteria. If the sign-up process is successful, the user's account is created.

● Sign-in Page: The sign-in page allows registered users to log in to their accounts and access the system's features.

- Users are typically prompted to enter their credentials, which typically include:
- Username: The unique identifier chosen during the sign-up process.
- Password: The securely chosen password associated with the user's account.
- Forgot Password: A link or button to initiate the password reset process in case the user forgets their password.

Upon submission of the sign-in form, the system verifies the user's credentials by comparing the entered username and password against stored records in the system's database. If the authentication is successful, the user is granted access to the system and redirected to the designated landing page. In case of authentication failure (e.g., incorrect username or password), the system displays an error message informing the user of the issue and prompting them to try again.

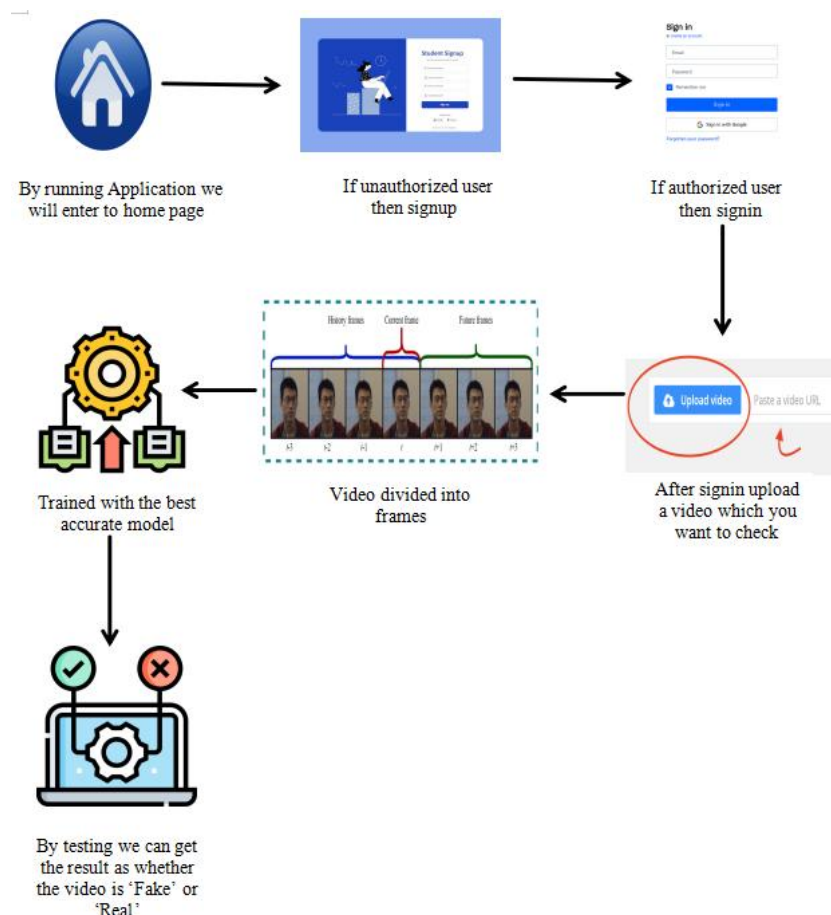


Fig 4.1 System Architecture

Upload Page:After successfully signing into the deepfake detection system, users are directed to the upload page, where they can initiate the process of uploading a video for analysis. Here's how the upload page functions:

- **Accessing the Upload Page:** Upon signing in, users are automatically redirected to the upload page as it is the next step in the user flow. Alternatively, users may navigate to the upload page through the navigation menu or a designated button on the home page.
- **User Interface:** The upload page features a user-friendly interface designed to facilitate the process of uploading videos. Users are presented with clear instructions and prompts guiding them through the upload process.
- **Selecting the Video File:**Users are prompted to select the video file they wish to upload from their local device. The upload page may include a file input field or drag-and-drop functionality to facilitate the selection of the video file.
- **Supported File Formats:**The system specifies the supported file formats for video uploads, ensuring compatibility with the analysis pipeline. Commonly supported formats may include MP4, AVI, MOV, or other widely used video formats.

Segmentating into frames: After the video is successfully uploaded to the deepfake detection system, the next step involves breaking down the video into individual frames for further analysis. Here's how this process typically unfolds:

- **Video Processing Initialization:** Upon completion of the upload, the system initiates the video processing pipeline to handle the uploaded video file. This pipeline may involve dedicated processing modules or services responsible for various tasks, including frame extraction, analysis, and feature extraction.
- **Frame Extraction:** The uploaded video file is parsed and processed to extract individual frames at regular intervals or keyframes. Frame extraction techniques may vary based on factors such as video format, encoding, and resolution. Common methods include using video processing libraries (e.g., OpenCV) to extract frames programmatically or utilizing specialized video processing algorithms.
- **Frame Preprocessing:** Extracted frames may undergo preprocessing steps to enhance their quality and prepare them for analysis. Preprocessing techniques may include resizing, normalization, denoising, or color correction to standardize the appearance of frames and facilitate accurate analysis.

Choose best model: After selecting the Xception model for deepfake detection due to its superior accuracy among the evaluated models, the next step involves training the extracted frames using this chosen model. Here's a detailed description of the training process:

- **Data Preparation:** The frames extracted from the uploaded video are organized into training and validation datasets. Data preprocessing techniques may be applied to standardize the format, resolution, and quality of the frames, ensuring consistency across the dataset. Additionally, data augmentation techniques such as random cropping, rotation, and flipping may be employed to increase the diversity of training samples and improve model generalization.
- **Model Initialization:** The Xception model is initialized with pre-trained weights on a large dataset of images, such as ImageNet. Transfer learning is leveraged to fine-tune the pre-trained Xception model on the specific task of deepfake detection using the extracted frames.
- **Training Procedure:** The training procedure involves feeding the preprocessed frames into the Xception model and adjusting the model's parameters (weights and biases) iteratively to minimize a predefined loss function. Backpropagation and gradient descent algorithms, such as Adam or Stochastic Gradient Descent (SGD), are used to update the model's parameters based on the computed gradients. The training process continues for multiple epochs, with the model learning to distinguish between authentic and deepfake frames by minimizing the classification error on the training data.

Results: After the trained Xception model has been deployed and integrated into the deepfake detection system, users can receive the authenticity result for their uploaded video. Here's a breakdown of how the system generates and presents the result:

- **Submission of Video:** Users navigate to the upload page and select the video they want to analyze for deepfake content. The selected video is then processed and broken down into individual frames as described earlier.
- **Frame Analysis:** Each frame extracted from the uploaded video is passed through the trained Xception model for analysis. The Xception model evaluates each frame based on learned features and patterns to determine the likelihood of it being part of a deepfake.
- **Aggregation of Results:** The results from analyzing individual frames are aggregated to make a holistic assessment of the entire video. This aggregation process may involve calculating the average confidence score or considering the majority vote among the analyzed frames.
- **Result Generation:** Based on the aggregated analysis results, the system generates a final output indicating whether the uploaded video is classified as fake or real. This result is presented to the user through the system's user interface, typically alongside additional details such as confidence levels or supporting evidence.

5. IMPLEMENTATION

In the creation of the Deepfake Face Mask Dataset (DFFMD), a meticulous curation process ensures diversity by incorporating variations in lighting, face mask types, and facial expressions. The dataset is strategically split into training and testing sets for accurate model assessment in real-world scenarios involving face-masked individuals. The model architecture employs the powerful combination of Inception-ResNet-v2 and Xception, enhancing detection accuracy through multi-scale feature extraction and depthwise separable convolutions. Data preprocessing focuses on enhancing video frame quality, normalizing pixel values, and employing augmentation techniques to optimize the model for recognizing faces with masks.

During model training, emphasis is placed on detecting face-mask-enhanced deepfakes, utilizing a suitable loss function and optimizer for optimal convergence. Fine-tuning is performed based on performance metrics to enhance accuracy and effectiveness. Evaluation of the trained model showcases its effectiveness in discerning manipulated content, surpassing benchmarks like InceptionResNetV2 and VGG19. The integration of CNN and Xception further improves image feature extraction, resulting in a robust model for image classification tasks. Performance metrics such as accuracy, precision, recall, and F1 score are employed to assess the model's overall effectiveness in minimizing false positives/negatives and achieving a balance between precision and recall. The algorithms, including Inception-ResNet-v2, VGG19, CNN, and Xception, contribute to the model's state-of-the-art capabilities in deepfake detection, advancing technology in addressing evolving challenges.

5.1 Module

Supervised machine learning: It is one of the ways of machine learning where the model is trained by input data and expected output data. To create such model, it is Necessary to go through the following phases:

1. Deepfake Face Mask Dataset (DFFMD) Creation
2. Model Architecture
3. Data Preprocessing
4. Model Training
5. Evaluation
6. Integration of CNN and Xception
7. Performance Metrics

5.2 Module Description

1. Deepfake Face Mask Dataset (DFFMD) Creation:

Compiling a diverse deepfake video dataset featuring face masks demands meticulous curation. The collection incorporates variations in lighting, encompassing diverse conditions to enhance model robustness. Encompassing an array of face mask types ensures comprehensive coverage, while the inclusion of various facial expressions adds realism and complexity. To foster accurate model assessment, the dataset is judiciously split into training and testing sets. This meticulous approach guarantees a comprehensive and representative dataset for training deepfake detection models, considering the dynamic nature of real-world scenarios involving face-masked individuals. The dataset's compilation involves several unique considerations to capture the diversity and nuances of face mask usage. One key aspect is the inclusion of various face mask types, ranging from surgical masks to cloth masks and N95 respirators, to encompass the wide array of designs and materials encountered in practice. Additionally, variations in lighting conditions are crucially incorporated to simulate the diverse environments individuals may encounter, including indoor, outdoor, daylight, and artificial lighting settings. Moreover, the dataset emphasizes the inclusion of diverse facial expressions to enhance realism and complexity, thereby challenging detection algorithms to accurately discern manipulated content amidst natural variations in human behavior.

Annotating and labeling each video with ground truth information is paramount to ensure accurate model training and evaluation. This process involves meticulous annotation of genuine and deepfake videos, along with metadata detailing mask type, lighting conditions, and facial expressions. Careful curation ensures a balanced distribution of samples across different categories, mitigating biases and facilitating fair evaluation of model performance. Furthermore, data augmentation techniques are employed to enrich the dataset with synthetic variations, enhancing model generalization and robustness. Ethical considerations play a pivotal role, necessitating measures to obtain consent from individuals appearing in the videos and protect their identities through anonymization techniques.

Continuous updates and maintenance are essential to keep the dataset relevant and reflective of evolving deepfake techniques and real-world challenges. Regular additions of new samples and reevaluation of existing ones ensure that the dataset remains up-to-date and representative of current trends in deepfake detection research. In summary, the creation of DFFMD is a comprehensive endeavor that requires careful consideration of diverse face mask types, lighting conditions, facial expressions, annotation processes, ethical considerations, and

continuous updates. By addressing these unique considerations, DFFMD serves as a valuable resource for training and evaluating deepfake detection models in real-world scenarios involving face-masked individuals.

2. Model Architecture:

The Inception-ResNet-v2 model, featuring preprocessing, feature-based analysis, residual connections, and batch normalization, is implemented. Integration of Convolutional Neural Networks (CNN) is enhanced with Xception to elevate detection accuracy. This architecture combines the strengths of Inception-ResNet-v2's multi-scale feature extraction and Xception's depthwise separable convolutions, fostering a robust and efficient model for image analysis tasks. The inclusion of residual connections and batch normalization further optimizes training, ensuring superior performance in image recognition and object detection applications. Furthermore, the inclusion of residual connections in the architecture plays a pivotal role in optimizing training and improving model convergence. Residual connections mitigate the vanishing gradient problem commonly encountered in training deep neural networks, allowing for more efficient gradient propagation and faster convergence. This results in a more stable and robust training process, ultimately leading to superior performance in image recognition and object detection tasks.

Additionally, batch normalization is incorporated into the architecture to further enhance training stability and accelerate convergence. By normalizing the activations within each mini-batch during training, batch normalization reduces internal covariate shift and accelerates training convergence, enabling the model to learn more effectively from the data. This ensures that the model can achieve higher accuracy levels with fewer training iterations, reducing the computational resources required for training while improving overall performance. Moreover, the combination of Inception-ResNet-v2 and Xception architectures allows for a flexible and adaptable model that can be tailored to various image analysis tasks. Whether it's image classification, object detection, semantic segmentation, or other tasks, this hybrid architecture can be fine-tuned and optimized to achieve state-of-the-art performance across different domains. This versatility makes it a valuable tool for researchers and practitioners in fields such as computer vision, medical imaging, autonomous driving, and more.

3. Data Preprocessing :

Enhance video frame quality for facial features, accounting for face masks. Normalize pixel values and implement data augmentation techniques to enhance the training dataset for improved model robustness in recognizing faces, even with the presence of masks, ensuring optimal performance in facial recognition applications.

Data preprocessing is a critical step in preparing video frames for deepfake detection, particularly in scenarios involving face masks. Beyond enhancing frame quality for facial features, the preprocessing pipeline encompasses various techniques aimed at standardizing and augmenting the training dataset to enhance model robustness. One key aspect involves normalizing pixel values across frames to ensure consistency in intensity levels, thereby mitigating potential biases during model training. By standardizing pixel values, the model becomes more adept at discerning subtle variations in facial features, even when obscured by masks, leading to improved accuracy in facial recognition applications. Moreover, data augmentation techniques play a crucial role in enriching the training dataset with diverse variations of face-masked individuals. Augmentation strategies such as rotation, translation, scaling, and flipping are applied to video frames to simulate variations commonly encountered in real-world scenarios. This augmentation not only increases the diversity of training samples but also helps the model generalize better to unseen data, ultimately enhancing its ability to recognize faces under different conditions and orientations. Additionally, techniques such as random cropping and resizing are employed to further diversify the dataset and expose the model to a wide range of facial configurations and mask types.

Furthermore, preprocessing techniques specifically tailored to handle face masks are incorporated into the pipeline to address the unique challenges posed by mask-wearing individuals. This may include techniques for enhancing contrast and sharpness in regions of interest corresponding to facial features, such as eyes, nose, and mouth, to improve their visibility despite mask occlusion. Additionally, specialized preprocessing steps may be applied to mitigate artifacts introduced by the presence of masks, such as reflections, shadows, or fabric textures, which could potentially confound the model's ability to accurately recognize faces. In addition to enhancing frame quality and addressing mask-related challenges, data preprocessing also involves partitioning the dataset into training, validation, and test sets. This ensures that the model is trained and evaluated on diverse samples representative of real-world scenarios, thereby fostering optimal generalization and performance. Careful consideration is given to maintain class balance and prevent data leakage between partitions, ensuring unbiased model evaluation and reliable performance estimation.

4. Model Training:

Train the model with a focus on detecting deepfakes featuring face masks, employing a suitable loss function and optimizer for optimal convergence. Fine-tune model parameters based on performance metrics to enhance accuracy and effectiveness. This approach ensures the model

is adept at identifying deepfake content specifically involving face masks, providing a robust solution for addressing evolving challenges in deepfake detection. Model training is a crucial phase in the development of deepfake detection systems, particularly when focusing on detecting deepfakes featuring face masks. During training, the model learns to discern between authentic and manipulated content, with a specific emphasis on identifying deepfakes with face masks accurately. To accomplish this, a suitable loss function is selected to quantify the disparity between predicted and ground truth labels, guiding the model towards optimal parameter adjustments. Common loss functions for classification tasks, such as cross-entropy loss, are often employed to measure the discrepancy between predicted probabilities and actual class labels, facilitating effective gradient-based optimization.

In addition to the choice of loss function, the selection of an appropriate optimizer plays a crucial role in ensuring optimal convergence during training. Optimizers such as stochastic gradient descent (SGD), Adam, or RMSprop are commonly utilized to update model parameters iteratively, minimizing the loss function and improving classification accuracy. Fine-tuning of model parameters based on performance metrics, such as accuracy, precision, recall, and F1 score, is performed iteratively to enhance the model's effectiveness in detecting deepfakes featuring face masks. This iterative refinement process involves adjusting hyperparameters, including learning rates, regularization strengths, and batch sizes, to achieve the desired balance between model complexity and generalization ability. Furthermore, model training involves careful monitoring of performance metrics on validation datasets to prevent overfitting and ensure robust model generalization. Early stopping criteria may be employed to halt training when performance on the validation set ceases to improve, thereby preventing the model from memorizing training data and improving its ability to generalize to unseen examples. Regularization techniques, such as dropout and weight decay, may also be applied to mitigate overfitting and improve model generalization. Throughout the training process, the model's performance is evaluated on diverse datasets encompassing a wide range of face-masked individuals and scenarios. This ensures that the model learns to generalize effectively across different variations of face masks, lighting conditions, facial expressions, and backgrounds encountered in real-world applications. By focusing on detecting deepfakes specifically featuring face masks and fine-tuning model parameters based on performance metrics, the trained model becomes adept at accurately identifying manipulated content in the presence of face masks, providing a robust solution for addressing evolving challenges in deepfake detection.

5. Evaluation:

The trained model's accuracy in detecting face-mask-enhanced deepfake videos on the test

dataset was assessed, revealing its effectiveness in discerning manipulated content. Comparative analysis with renowned methods like InceptionResNetV2 and VGG19 demonstrated the model's competitive performance, showcasing its potential as a robust solution for identifying deepfakes with face-mask enhancements. The results signify the model's contribution to advancing the state-of-the-art in deepfake detection technology.

In addition to assessing the accuracy of the trained model in detecting face-mask-enhanced deepfake videos, comprehensive evaluation metrics were employed to provide a nuanced understanding of its performance. Beyond accuracy, metrics such as precision, recall, F1 score, and area under the receiver operating characteristic curve were calculated to gauge the model's effectiveness across different aspects of classification performance. Precision measures the proportion of true positive predictions among all positive predictions made by the model, providing insight into its ability to avoid false positives. Recall, on the other hand, quantifies the proportion of true positive predictions among all actual positive instances in the dataset, indicating the model's sensitivity to detecting true positive cases. The F1 score, which is the harmonic mean of precision and recall, offers a balanced assessment of the model's performance, particularly in scenarios where class imbalances exist.

6. Integration of CNN and Xception:

Integrating Convolutional Neural Networks (CNN) with an extension using Xception enhances the model's capacity for image feature extraction. To capitalize on the strengths of both architectures, fine-tuning is performed on the combined model using the training dataset. This synergistic approach leverages CNN's foundational image processing capabilities and Xception's deep learning advancements, resulting in a more robust and effective model for image classification tasks.

The integration of Convolutional Neural Networks (CNN) with an extension using Xception represents a significant advancement in the field of deep learning for image analysis tasks. By combining the foundational image processing capabilities of CNN with the deep learning advancements of Xception, the integrated model achieves a synergistic effect that enhances its capacity for image feature extraction. This integration allows the model to leverage the strengths of both architectures, capitalizing on CNN's ability to capture spatial dependencies in images and Xception's efficiency in learning complex patterns through depthwise separable convolutions. During fine-tuning on the combined model using the training dataset, parameters are adjusted to optimize performance specifically for image classification tasks. This fine-tuning process involves updating the weights and biases of the network layers based on the error gradient

computed during backpropagation. By iteratively adjusting these parameters, the model learns to extract discriminative features from input images, enabling more accurate classification of image content. One key advantage of integrating CNN with Xception is the improved robustness and effectiveness of the resulting model. CNNs are renowned for their ability to capture hierarchical features from input images, making them well-suited for a wide range of image processing tasks. Xception, on the other hand, introduces depthwise separable convolutions, which significantly reduce the number of parameters and computational complexity while preserving representational capacity. By combining these architectures, the integrated model achieves a balance between efficiency and effectiveness, making it more suitable for deployment in resource-constrained environments or real-time applications where speed and accuracy are paramount.

Furthermore, the integration of CNN with Xception facilitates the development of a more versatile and adaptable model architecture. This hybrid approach allows researchers and practitioners to leverage the strengths of both architectures for various image analysis tasks, including image classification, object detection, semantic segmentation, and more. By fine-tuning the integrated model on specific datasets and tasks, it can be tailored to achieve state-of-the-art performance across different domains, making it a valuable tool for a wide range of applications in computer vision and beyond.

7. Performance Metrics:

Valuate model effectiveness by measuring key performance metrics like accuracy, precision, recall, and F1 score. These metrics provide insights into the model's overall performance, helping assess its ability to make correct predictions, minimize false positives/negatives, and achieve a balance between precision and recall.

Performance metrics serve as critical indicators of a model's effectiveness and reliability in making predictions and discerning between different classes of data. Among the key performance metrics used to evaluate the model's performance are accuracy, precision, recall, and F1 score. Accuracy measures the proportion of correct predictions made by the model out of the total predictions, providing an overall assessment of its correctness. Precision, on the other hand, evaluates the model's ability to minimize false positives by measuring the proportion of true positive predictions among all positive predictions made by the model. Similarly, recall assesses the model's sensitivity in detecting true positive cases by measuring the proportion of true positive predictions among all actual positive instances in the dataset. Additionally, the F1 score, which is the harmonic mean of precision and recall, offers a balanced evaluation of the model's performance, particularly in scenarios where class imbalances exist. It provides a single value that

combines both precision and recall, offering insights into the model's ability to achieve a balance between minimizing false positives and false negatives. By considering these diverse performance metrics, the evaluation process offers a comprehensive assessment of the model's effectiveness in making correct predictions and minimizing errors.

Furthermore, performance metrics play a crucial role in guiding model optimization and fine-tuning efforts. By analyzing the model's performance across different metrics, researchers and practitioners can identify areas for improvement and adjust model parameters accordingly to enhance its overall performance. Moreover, performance metrics provide valuable insights into the model's strengths and weaknesses, helping stakeholders make informed decisions about its deployment and usage in real-world applications.

5.3 Introduction To Technologies Used

Deep Learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

How deep learning works ?

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data. Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

Another process called backpropagation uses algorithms, like gradient descent, to calculate errors

in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

Algorithms Used:

Inception-ResNet-v2: Inception-ResNet v2 is a deep convolutional neural network architecture that combines elements from the Inception architecture and ResNet (Residual Network) architecture. It was introduced by Google researchers in the context of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2016. The architecture builds upon the Inception family of models, known for using inception modules that allow for the extraction of features at multiple spatial scales. Inception modules are characterized by the use of parallel convolutional filters of different sizes, enabling the network to capture information at various levels of abstraction.

The integration of residual connections, inspired by ResNet, is a key aspect of Inception-ResNet v2. Residual connections facilitate the training of very deep networks by addressing the vanishing gradient problem. These connections allow the network to learn residual functions, making it easier to train deeper architectures. Inception-ResNet v2 incorporates a series of these inception-residual blocks, each containing multiple inception modules with residual connections. The overall architecture is designed to achieve a balance between computational efficiency and high performance in image classification tasks. It has shown strong performance in terms of accuracy and generalization, making it suitable for a variety of computer vision applications. It's worth noting that Inception-ResNet v2 is part of a family of Inception models, including Inception v1, Inception v3, and others, each with variations and improvements over the previous versions. These models are often used as feature extractors or pre-trained models for transfer learning in various deep learning applications. Inception-ResNet-v2 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. There are further two sub-versions of Inception-ResNet, namely v1 and v2. Inception-ResNet v2 has higher accuracy and computational cost as compared to Inception-ResNet v1. They are different. Inception-ResNet V2 model, with weights pre-trained on ImageNet. This model can be built both with 'channels_first' data format(channels, height, width) or 'channels_last' data format(height, width, channels). The default input size for this

model is 299x299.

A distinctive feature of Inception-ResNet-v2 lies in its utilization of dual pathways within each module. These pathways encompass both Inception and ResNet paths, allowing for parallel feature extraction. By concurrently leveraging the advantages of both architectures, Inception-ResNet-v2 can extract rich hierarchical representations of input images while maintaining computational efficiency. This dual-pathway design underscores the architecture's versatility and adaptability to diverse datasets and tasks. To address concerns regarding computational complexity and memory requirements associated with its increased depth, Inception-ResNet-v2 employs various optimization techniques. These include factorized convolutions, dimensionality reduction, and efficient pooling operations. By judiciously optimizing network architecture and operations, Inception-ResNet-v2 achieves a balance between model complexity and computational efficiency, making it suitable for deployment in resource-constrained environments. Furthermore, Inception-ResNet-v2 adopts bottleneck blocks akin to those found in ResNet architectures. These blocks consist of a sequence of 1x1, 3x3, and 1x1 convolutions, facilitating the reduction of parameters and computational cost while preserving representational capacity. This strategic use of bottleneck blocks underscores Inception-ResNet-v2's commitment to efficiency without compromising on model performance or expressiveness.

Inception-ResNet-v2 culminates its network architecture with global average pooling (GAP) as the final layer. GAP averages the feature maps spatially to produce a single vector representation for each feature map. This pooling operation aids in reducing overfitting, enhancing regularization, and accommodating variable input sizes. By incorporating GAP, Inception-ResNet-v2 further refines its ability to generalize across diverse datasets and input variations. In terms of training and optimization, Inception-ResNet-v2 typically relies on stochastic gradient descent (SGD) with momentum or its variants. Additional techniques such as batch normalization, dropout, and weight regularization may be employed to improve convergence and generalization. This meticulous training methodology ensures that Inception-ResNet-v2 learns discriminative features effectively while minimizing overfitting and maximizing performance. In practice, Inception-ResNet-v2 has found wide-ranging applications in various computer vision tasks, including image classification, object detection, and semantic segmentation. Its versatility and robustness have enabled it to achieve state-of-the-art performance on benchmark datasets such as ImageNet, COCO, and PASCAL VOC. Moreover, the availability of pre-trained models and open-source implementations within frameworks like TensorFlow has democratized access to Inception-ResNet-v2, empowering researchers and practitioners to

leverage its capabilities for their own projects and applications. Inception-ResNet-v2 stands as a testament to the continuous evolution of deep learning architectures in the pursuit of higher performance and efficiency. By seamlessly integrating the strengths of Inception and ResNet architectures, Inception-ResNet-v2 represents a formidable tool for advancing the state of the art in computer vision and deep learning research. Its innovative design, coupled with its practical applications and open-source availability, cements its position as a cornerstone in the field of computer vision.

VGG19: VGG-19 is a convolutional neural network that is 19 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. VGG19, a member of the Visual Geometry Group (VGG) family of models, stands out as a prominent deep convolutional neural network architecture developed by researchers at the University of Oxford in 2014. Serving as an extension of the original VGG16 architecture, VGG19 derives its name from the total number of weight layers it encompasses. Comprising 19 layers in total, the architecture includes 16 convolutional layers and 3 fully connected layers, making it a robust model for image classification tasks.

One of the distinctive characteristics of VGG19 lies in its consistent use of small 3x3 convolutional filters across the entire network. These filters, coupled with Rectified Linear Unit (ReLU) activation functions after each convolutional layer, introduce non-linearity and facilitate the learning of complex features. Additionally, VGG19 incorporates max-pooling layers with 2x2 filters and a stride of 2 pixels, contributing to spatial dimension reduction while expanding.

The architecture concludes with three fully connected layers, culminating in a softmax layer for classification in the output. VGG19's design philosophy involves increasing the number of channels or filters in deeper layers, allowing the model to capture progressively complex features. Originally crafted for image classification, VGG19 has been a benchmark in computer vision tasks, notably participating in the ImageNet Large Scale Visual Recognition Challenge.

Beyond its standalone performance, VGG19 is frequently employed in transfer learning scenarios. The model's popularity and straightforward architecture make it a compelling choice for leveraging pre-trained features on large datasets, followed by fine-tuning for specific tasks with smaller datasets. It's important to note, however, that while VGG19 has demonstrated strong performance, its relatively high number of parameters renders it computationally demanding. As the field of deep learning has evolved, newer architectures such as ResNet and EfficientNet have

surpassed VGG19 in terms of efficiency and accuracy, yet the model remains a valuable and well-understood tool in the deep learning landscape.

One of the defining characteristics of VGG19 is its uniform architecture, where each convolutional layer is followed by a max-pooling layer, resulting in a straightforward and easy-to-understand structure. This uniformity contributes to its popularity as it allows for easy implementation, modification, and experimentation in various deep learning projects. VGG19 is known for its deep stack of convolutional layers, which enables it to capture intricate features and patterns in images with multiple levels of abstraction. By employing a series of smaller convolutional filters (typically 3x3), VGG19 is capable of learning complex representations of visual features, leading to impressive performance in image recognition tasks. Despite its effectiveness, VGG19 has certain limitations, particularly in terms of computational efficiency and model size. The extensive depth of the network results in a large number of parameters, making it computationally expensive to train and deploy, especially on resource-constrained devices. Additionally, the uniform architecture of VGG19 may lead to issues with overfitting, especially when dealing with smaller datasets.

However, VGG19 remains a valuable asset in the deep learning toolkit, particularly as a benchmark architecture for evaluating the performance of newer, more complex models. Its simplicity and effectiveness make it a popular choice for researchers and practitioners working on various computer vision tasks, including image classification, object detection, and segmentation. Furthermore, VGG19's pre-trained weights, trained on large-scale image datasets such as ImageNet, can be leveraged for transfer learning. This means that developers can fine-tune the model on domain-specific datasets with relatively small amounts of labeled data, thereby accelerating the development of custom deep learning solutions for specific applications.

CNN: A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. Convolutional Neural Networks (CNNs) represent a class of deep neural networks specifically designed for tasks related to computer vision, image analysis, and pattern recognition. CNNs have revolutionized the field by introducing specialized layers and operations that effectively capture spatial hierarchies and local patterns within visual data. The fundamental building blocks of CNNs include convolutional layers, pooling layers, and fully connected layers.

At the core of CNNs are convolutional layers, which employ convolutional operations to extract features from input data. These layers consist of learnable filters or kernels that slide

across the input, capturing local patterns and spatial relationships. The application of non-linear activation functions, typically Rectified Linear Units (ReLU), introduces non-linearity, enabling the network to learn complex representations. Pooling layers play a crucial role in reducing the spatial dimensions of the feature maps generated by convolutional layers. Commonly, max pooling is employed to downsample the spatial resolution while retaining essential features. This spatial hierarchy allows CNNs to focus on increasingly abstract and hierarchical features as they process deeper layers.

Fully connected layers at the end of a CNN transform the high-level learned features into class scores or predictions. These layers connect every neuron to every neuron in the subsequent layer, creating a global perspective that combines the localized features extracted earlier in the network. CNNs excel in tasks such as image classification, object detection, and facial recognition due to their ability to automatically learn hierarchical representations from raw pixel data. Transfer learning is a prevalent technique with CNNs, involving the use of pre-trained models on large datasets and fine-tuning for specific tasks with limited data, enabling efficient learning even in scenarios with limited labeled examples.

Beyond traditional computer vision tasks, CNNs have found applications in diverse fields such as natural language processing and medical image analysis. The success of CNNs underscores their adaptability and effectiveness in capturing and understanding complex patterns within multi-dimensional data, making them a cornerstone in modern machine learning and artificial intelligence.

XCEPTION: Xception, short for "Extreme Inception," is a deep convolutional neural network architecture that represents a variant of the Inception architecture. Introduced by Google researchers in 2017, Xception is designed to push the boundaries of convolutional neural networks (CNNs) in terms of efficiency and performance. The key innovation of Xception lies in its departure from traditional convolutional approaches by replacing standard convolutional layers with depthwise separable convolutions. This modification significantly reduces the number of parameters and computations, resulting in a more efficient and lightweight network while maintaining or even enhancing its ability to capture complex hierarchical features.

In Xception, each convolutional operation is decomposed into two separate steps: a depthwise convolution, which operates on each input channel independently, and a pointwise convolution, which combines the results across channels. This separation of spatial and cross-channel information helps in capturing fine-grained details and global context efficiently.

Xception has demonstrated its effectiveness in various computer vision tasks, achieving competitive performance on benchmarks like ImageNet. Its architecture highlights the importance of rethinking the conventional convolutional design, providing a valuable contribution to the development of more efficient and powerful deep learning models.

Xception is a convolutional neural network that is 71 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. Traditional CNNs typically use standard convolutions followed by pointwise convolutions to extract features from input data. However, Xception replaces these standard convolutions with depthwise separable convolutions. In depthwise separable convolutions, each spatial location is convolved separately, followed by a pointwise convolution that combines information across channels. This separation significantly reduces the number of parameters and computational complexity while preserving model performance.

The architecture of Xception is based on a series of depthwise separable convolution blocks, each consisting of depthwise convolutions, batch normalization, activation functions (e.g., ReLU), and pointwise convolutions. These blocks are stacked together to form a deep neural network capable of learning hierarchical representations from input data efficiently. One of the key advantages of Xception is its ability to achieve state-of-the-art performance on image classification tasks while being computationally efficient. By reducing the number of parameters and computational operations, Xception enables faster training and inference times compared to traditional CNN architectures.

Xception has found widespread application across various computer vision tasks beyond image classification. It has been successfully employed in tasks such as object detection, semantic segmentation, and, notably, deepfake detection. Its ability to capture intricate features and patterns from input data makes it well-suited for discerning between authentic and manipulated media content. In the realm of deepfake detection, Xception serves as a powerful tool for identifying subtle artifacts and inconsistencies present in manipulated videos. Its depthwise separable convolutions allow it to learn discriminative features effectively, enabling accurate detection of deepfake content even in the presence of sophisticated manipulation techniques. Despite its effectiveness, Xception is not without limitations. Training Xception models from scratch can require substantial computational resources due to the depth and complexity of the network. Additionally, fine-tuning pre-trained Xception models for specific tasks may still necessitate

considerable data and compute resources. Overall, Xception represents a groundbreaking advancement in deep learning architecture, offering a compelling combination of performance, efficiency, and versatility. Its adoption in various applications, including deepfake detection, underscores its significance in pushing the boundaries of artificial intelligence and computer vision.

LIBRARIES/PACKGES :

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem.

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

DATABASE:

A database is a structured collection of organized data that is stored, managed, and accessed electronically. It serves as a central repository for information, allowing users to efficiently store, retrieve, and manipulate data. Databases are crucial components in various applications and systems, providing a systematic way to organize and manage large volumes of information. They use specific data models, such as relational, document-oriented, or graph-based, to define the structure and relationships within the data. The primary functions of a database include storing data in tables or documents, ensuring data integrity through relationships and constraints, and offering query languages for efficient data retrieval and manipulation. Databases play a pivotal role in business, technology, and research, serving as a foundation for applications ranging from

simple websites to complex enterprise systems.

SQLite3

SQLite3 is a lightweight, embedded relational database management system (RDBMS) that is widely used for embedded systems, mobile applications, and small-scale database-driven applications. One of the key features of SQLite is its self-contained nature, as it operates without the need for a separate server process and relies on a single ordinary disk file for data storage. This simplicity and portability make SQLite suitable for scenarios where a full-scale database server might be impractical or unnecessary.

SQLite3 supports the SQL query language and provides a range of features typical of relational databases, including tables, indexes, transactions, and triggers. Despite its lightweight footprint, SQLite is ACID-compliant (Atomicity, Consistency, Isolation, Durability), ensuring data integrity and reliability. It is also known for its ease of integration, with many programming languages and platforms offering built-in support for SQLite. Developers often choose SQLite for applications with moderate data storage needs, especially in mobile and embedded environments where resources are constrained.

Another notable aspect of SQLite is its widespread adoption. It is employed in various software applications, including popular web browsers, mobile apps, and embedded systems. The ease of use, minimal setup requirements, and consistent performance contribute to SQLite's reputation as a versatile and reliable database solution for a range of development projects.

Hyper Text Markup Language (HTML)

HTML, or HyperText Markup Language, is the standard markup language used for creating and structuring content on the World Wide Web. Developed by the World Wide Web Consortium (W3C), HTML provides a set of elements and tags that define the structure and presentation of web documents. It forms the backbone of web development, serving as the building blocks for creating web pages and applications.

HTML uses a tag-based syntax where elements are enclosed within angled brackets. Each HTML document typically consists of an opening ``<html>`` tag, followed by a ``<head>`` section for metadata and a ``<body>`` section for the actual content. Common elements include headings (`<h1>` to `<h6>`), paragraphs (`<p>`), lists (``, ``, ``), links (`<a>`), images

(``), forms (`<form>`), and more. These elements provide structure and semantics to the content, facilitating accessibility and search engine optimization.

HTML works in conjunction with Cascading Style Sheets (CSS) and JavaScript to create dynamic and visually appealing web pages. CSS is used for styling and layout, while JavaScript adds interactivity and enhances the functionality of web applications. Modern web development often involves the use of frameworks and libraries to streamline the process of creating responsive and feature-rich websites.

As HTML has evolved, newer versions have been released, with HTML5 being the latest standard. HTML5 introduces new elements, attributes, and APIs (Application Programming Interfaces) that support multimedia, offline web applications, and enhanced semantics. With its simplicity and ubiquity, HTML remains a fundamental language for web development, providing a foundation for creating content that is accessible, well-structured, and compatible across different browsers and devices.

Cascading Style Sheet (CSS)

CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation and formatting of a document written in HTML or XML. Developed by the World Wide Web Consortium (W3C), CSS enables web developers to control the visual appearance of web pages, ensuring separation between the structure/content (HTML) and presentation/style (CSS). This separation enhances maintainability, scalability, and the ability to apply consistent styling across multiple pages.

CSS works by applying rules to HTML elements. Each rule consists of a selector (indicating which HTML elements to style) and a declaration block containing one or more property-value pairs specifying the desired styles. Properties control various aspects such as color, size, spacing, font, and layout. CSS rules can be embedded directly within an HTML document, placed in a separate external stylesheet (.css file), or applied inline within individual HTML elements.

Selectors in CSS can target specific HTML elements, classes, IDs, or even complex relationships between elements, allowing for fine-grained control over styling. The term "Cascading" in CSS refers to the order of priority when conflicting styles are applied. Styles can be inherited from

parent elements, overridden by more specific selectors, or influenced by the order of rule declarations. CSS3, the latest version of CSS, introduces numerous features, including transitions, animations, flexbox, grid layout, and responsive design capabilities. These additions enhance the possibilities for creating visually appealing, interactive, and responsive web pages. CSS plays a crucial role in modern web development, complementing HTML and JavaScript to create engaging and user-friendly online experiences.

CSS is an integral part of the front-end development process, influencing the user interface and overall user experience. It allows developers to create aesthetically pleasing designs, adapt layouts for different screen sizes and devices, and ensure accessibility. Responsive web design, a key concept in modern web development, is achieved through CSS media queries, enabling the adaptation of styles based on factors like screen width, height, and orientation.

One of the strengths of CSS lies in its ability to apply styles consistently across a website. By defining styles in an external stylesheet, changes made to the design are instantly reflected across all pages that link to that stylesheet. This consistency simplifies maintenance, reduces redundancy, and promotes a cohesive visual identity for a website.

CSS preprocessors like Sass and Less extend the capabilities of CSS by introducing features such as variables, nesting, and mixins. These preprocessors enhance code organization, improve maintainability, and provide a more efficient workflow for developers. Additionally, CSS frameworks like Bootstrap and Tailwind CSS offer pre-designed components and utilities, further accelerating development and ensuring a standardized look and feel.

The evolving landscape of CSS continues to adapt to the demands of modern web development. CSS Grid and Flexbox, for instance, offer powerful layout options, allowing developers to create complex, responsive designs with ease. As the web continues to advance, CSS remains a dynamic tool, empowering developers to bring creativity and functionality to the visual aspect of web applications and websites.

5.4 Sample Code

Training code:

```
import dlib
import cv2
import os
import re
import json
from pylab import *
from PIL import Image, ImageChops, ImageEnhance
import pandas as pd
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import os
import splitfolders
import PIL
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import Model
import glob
import cv2
import seaborn as sns
from tensorflow.keras import datasets, layers, models, losses
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Flatten, Dense, Dropout,
BatchNormalization, LSTM
import tensorflow.keras.backend as K
from sklearn.metrics import confusion_matrix
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3, ResNet50
from tensorflow.keras.layers import Input, Flatten, Dense, Concatenate, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow import keras
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import os, shutil
import sys
import shutil
import glob as gb
import warnings
warnings.filterwarnings('ignore')
```

```

real_videos_folder = 'Dataset/Real'
fake_videos_folder = 'Dataset/Fake'

def process_videos(video_folder, output_folder):
    list_of_data = [f for f in os.listdir(video_folder) if f.endswith('.mp4')]
    detector = dlib.get_frontal_face_detector()

    for vid in list_of_data:
        count = 0
        cap = cv2.VideoCapture(os.path.join(video_folder, vid))
        frameRate = cap.get(5)

        while cap.isOpened():
            frameId = cap.get(1)
            ret, frame = cap.read()

            if not ret or frame is None or frame.size == 0:
                break

            if frameId % ((int(frameRate) + 1) * 1) == 0:
                face_rects, _, _ = detector.run(frame, 0)

                for i, d in enumerate(face_rects):
                    x1 = d.left()
                    y1 = d.top()
                    x2 = d.right()
                    y2 = d.bottom()
                    crop_img = frame[y1:y2, x1:x2]

                    if crop_img.size != 0:
                        resized_img = cv2.resize(crop_img, (128, 128))
                        cv2.imwrite(os.path.join(output_folder, vid.split('.')[0] + '_' + str(count) + '.png'),
resized_img)
                        count += 1
        process_videos(real_videos_folder, 'data/real')
        process_videos(fake_videos_folder, 'data/fake')

class config:

    data_path = 'data/'

    path_train = "./output/train"

```

```

path_test = "./output/test"
splitfolders.ratio(config.data_path, output="output", seed=101, ratio=(.8, .2))
BATCH_SIZE = 64
IMAGE_SHAPE = (128, 128)
TRAIN_PATH = "output/train"
VAL_PATH = "output/val"
datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)

```

```

train_gen = datagen.flow_from_directory(directory = TRAIN_PATH,
                                       class_mode="categorical",
                                       target_size = IMAGE_SHAPE,
                                       batch_size = BATCH_SIZE,
                                       color_mode='rgb',
                                       seed = 1234,
                                       shuffle = True)

```

```

val_gen = datagen.flow_from_directory(directory = VAL_PATH,
                                       class_mode="categorical",
                                       target_size = IMAGE_SHAPE,
                                       batch_size = BATCH_SIZE,
                                       color_mode='rgb',
                                       seed = 1234,
                                       shuffle = True)

```

INCEPTION RESNET:

```

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

```

```

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

```

```

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

```

```

    return 2*((precision*recall)/(precision+recall+K.epsilon()))
inc= tf.keras.applications.inception_resnet_v2.InceptionResNetV2(include_top=False,
weights='imagenet', input_shape=(128, 128, 3), pooling='max')

```

```

x31 = Flatten()(inc.output)
predictions = Dense(2, activation='softmax')(x31)
model = Model(inputs = inc.inputs, outputs = predictions)
model.summary()
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=3)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', f1_score,
recall_m, precision_m])
history = model.fit(train_gen, validation_data=val_gen, epochs=25,
steps_per_epoch=len(train_gen), validation_steps=len(val_gen),
callbacks=[early_stopping_callback])
model.save('models/Inceptionresnet_v2.h5')

```

VGG19 :

```

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))
inc = tf.keras.applications.vgg19.VGG19(include_top=False, weights='imagenet',
input_shape=(128, 128, 3), pooling='max')
x31 = Flatten()(inc.output)
predictionss = Dense(2, activation='softmax')(x31)
model = Model(inputs = inc.inputs, outputs = predictionss)
model.summary()
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=3)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', f1_score,
recall_m, precision_m])
history = model.fit(train_gen, validation_data=val_gen, epochs=50,
steps_per_epoch=len(train_gen), validation_steps=len(val_gen), callbacks=[early_stopping_callback])
model.save('models/vgg19.h5')

```

CNN :

```

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

```



```

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))

model = models.Sequential()
model.add(Conv2D(filters=32, kernel_size=3, strides=1, padding='same', activation='relu',
input_shape=[128, 128, 3]))
model.add(MaxPooling2D(2, ))
model.add(Conv2D(filters=64, kernel_size=3, strides=1, padding='same', activation='relu'))
model.add(MaxPooling2D(2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.summary()
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=3)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', f1_score,
recall_m, precision_m])
history = model.fit(train_gen, validation_data=val_gen, epochs=15,
steps_per_epoch=len(train_gen), validation_steps=len(val_gen), callbacks=[early_stopping_callback])
model.save('models/cnn.h5')

```

XCEPTION :

```

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))

```

```

inc      =      tf.keras.applications.xception.Xception(include_top=False,      weights='imagenet',
input_shape=(128, 128, 3), pooling='max')
x31 = Flatten()(inc.output)
predictionss = Dense(2, activation='softmax')(x31)
model = Model(inputs = inc.inputs, outputs = predictionss)
model.summary()
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=3)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', f1_score,
recall_m, precision_m])
history      =      model.fit(train_gen,      validation_data=val_gen,      epochs=50,
steps_per_epoch=len(train_gen),      validation_steps=len(val_gen),
callbacks=[early_stopping_callback])
model.save('models/Xception.h5')
results = {'Accuracy': [a,a1,a2,a3],
'Recall': [r,r1,r2,r3],
'Precision': [p,p1,p2,p3],
'F1 Score' : [f,f1,f2,f3]}
index = ['InceptionResnet V2','VGG19','CNN','Xception']
results = pd.DataFrame(results,index=index)
fig = results.plot(kind='line',title='Comaprison of models',figsize =(19,19)).get_figure()
fig.savefig('Final Result.png')

```

Testing:

```

import tensorflow as tf
import dlib
import cv2
import os
import numpy as np
from PIL import Image, ImageChops, ImageEnhance
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from flask import Flask, render_template, request, redirect, send_file, url_for, Response
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import sqlite3
app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
model_path2 = 'models/xception.h5'
custom_objects = {
    'f1_score': f1_score,
    'recall_m': recall_score,
    'precision_m': precision_score
}
model = load_model(model_path2, custom_objects=custom_objects)
@app.route("/index")
def index():
    return render_template("index.html")

```

```

@app.route('/')
@app.route('/home')
def home():
    return render_template('home.html')
@app.route('/detection_results')
def detection_results():
    return render_template('detection_results.html')
@app.route('/upload')
def upload():
    return render_template('upload.html')
@app.route('/logon')
def logon():
    return render_template('signup.html')
@app.route('/login')
def login():
    return render_template('signin.html')
@app.route('/note')
def note():
    return render_template('note.html')
@app.route("/signup")
def signup():
    username = request.args.get('user', '')
    name = request.args.get('name', '')
    email = request.args.get('email', '')
    number = request.args.get('mobile', '')
    password = request.args.get('password', '')
    con = sqlite3.connect('signup.db')
    cur = con.cursor()
    cur.execute("insert into `info` (`user`,`email`,`password`,`mobile`,`name`)
VALUES(?, ?, ?, ?, ?)",(username,email,password,number,name))
    con.commit()
    con.close()
    return render_template("signin.html")
@app.route("/signin")
def signin():
    mail1 = request.args.get('user', '')
    password1 = request.args.get('password', '')
    con = sqlite3.connect('signup.db')
    cur = con.cursor()
    cur.execute("select `user`, `password` from info where `user` = ? AND `password`
= ?",(mail1,password1,))
    data = cur.fetchone()
    if data == None:
        return render_template("signin.html")
    elif mail1 == 'admin' and password1 == 'admin':
        return render_template("index.html")
    elif mail1 == str(data[0]) and password1 == str(data[1]):
        return render_template("index.html")
    else:
        return render_template("signup.html")
@app.route("/notebook")

```

```

def notebook():
    return render_template("notebook.html")
@app.route("/predict", methods=["GET", "POST"])
def predict_img():
    if request.method == "POST":
        if 'file' in request.files:
            f = request.files['file']
            filename = f.filename
            input_shape = (128, 128, 3)
            pr_data = []
            detector = dlib.get_frontal_face_detector()
            print("@@ Input posted = ", filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            f.save(file_path)
            cap = cv2.VideoCapture(file_path)
            frameRate = cap.get(5)
            predictions = []
            while cap.isOpened():
                frameId = cap.get(1)
                ret, frame = cap.read()
                if ret != True:
                    break
                if frameId % ((int(frameRate) + 1) * 1) == 0:
                    face_rects, scores, idx = detector.run(frame, 0)
                    for i, d in enumerate(face_rects):
                        x1 = d.left()
                        y1 = d.top()
                        x2 = d.right()
                        y2 = d.bottom()
                        crop_img = frame[y1:y2, x1:x2]
                        data = img_to_array(cv2.resize(crop_img, (128, 128))).flatten() / 255.0
                        data = data.reshape(-1, 128, 128, 3)
                        prediction = model.predict(data)
                        prediction_class = np.argmax(prediction, axis=1)
                        predictions.append(prediction_class)
            predictions = np.array(predictions)
            result = np.bincount(predictions.flatten()).argmax()
            if result == 0:
                ans = 'Fake'
            else:
                ans = 'Real'
            return render_template('display_image.html', result=ans)
    return render_template('index.html')
if __name__ == '__main__':
    app.run(debug=True)

```

HTML:

Home.html

```
<!DOCTYPE html>
```

```

<html lang="en">

  <!-- Basic -->
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <!-- Mobile Metas -->
  <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-
scale=1.0, user-scalable=no">

  <!-- Site Metas -->
  <title>Home</title>
  <meta name="keywords" content="">
  <meta name="description" content="">
  <meta name="author" content="">

  <!-- Site Icons -->
  <link rel="shortcut icon" href="static/images/favicon.ico" type="image/x-icon" />
  <link rel="apple-touch-icon" href="static/images/apple-touch-icon.png">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="static/css/bootstrap.min.css">
  <!-- Site CSS -->
  <link rel="stylesheet" href="static/style.css">
  <!-- Colors CSS -->
  <link rel="stylesheet" href="static/css/colors.css">
  <!-- ALL VERSION CSS -->
  <link rel="stylesheet" href="static/css/versions.css">
  <!-- Responsive CSS -->
  <link rel="stylesheet" href="static/css/responsive.css">
  <!-- Custom CSS -->
  <link rel="stylesheet" href="static/css/custom.css">

  <!-- Modernizer for Portfolio -->
  <script src="static/js/modernizer.js"></script>

  <!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
    <script src="https://oss.maxcdn.com/libs/respond.static/js/1.4.2/respond.min.js"></script>

  <![endif]-->
  <style>
#home {
/* background-image: url('static/uploads/slider-01.jpg'); */
background-repeat: no-repeat;
background-size: cover;
background-position: center;
max-width: 100%;
height: auto;

```

```

/* You can also set background-position to center if needed */
/* Example: */

```

```

}
</style>
</head>
<body class="host_version">

```

```

<!-- LOADER -->
<div id="preloader">
  <div class="loading">
    <div class="finger finger-1">
      <div class="finger-item">
        <span></span><i></i>
      </div>
    </div>
    <div class="finger finger-2">
      <div class="finger-item">
        <span></span><i></i>
      </div>
    </div>
    <div class="finger finger-3">
      <div class="finger-item">
        <span></span><i></i>
      </div>
    </div>
    <div class="finger finger-4">
      <div class="finger-item">
        <span></span><i></i>
      </div>
    </div>
    <div class="last-finger">
      <div class="last-finger-item"><i></i></div>
    </div>
  </div>
</div>
<!-- END LOADER -->

<header class="header header_style_01">
  <nav class="megamenu navbar navbar-default">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>

```

```

        </button>
        <!-- <a class="navbar-brand" href="index.html"></a> -->
    </div>
    <div id="navbar" class="navbar-collapse collapse">
        <h1 style="font-size:300%;">DFFMD</h1><h3 style="font-
size:160%;"> A Deepfake Face Mask Dataset For Infectious Disease Era With Deepfake
Detection Algorithms</h3>
        <ul class="nav navbar-nav">
            <li><a class="active" href="/logon">SignUp</a></li>
            <li><a class="active" href="/login">Signin </a></li>

        </ul>

    </div>
</div>
</nav>
</header>

```

```

<div id="bootstrap-touch-slider" class="carousel control-round indicators-line" data-
ride="carousel" data-pause="hover" data-interval="false" >
    <!-- Indicators -->
    <ol class="carousel-indicators">
        <li data-target="#bootstrap-touch-slider" data-slide-to="0" class="active"></li>
        <li data-target="#bootstrap-touch-slider" data-slide-to="1"></li>
        <li data-target="#bootstrap-touch-slider" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner" role="listbox">
        <div class="item active">
            <div id="home" class="first-section" style="background-
image:url('https://akm-img-a-
in.tosshub.com/businessoday/images/story/202311/untitled_design_-_2023-11-09t103355-
sixteen_nine.jpg?size=948:533');">
                <div class="container">
                    <div class="row">
                        <div class="col-md-12 col-sm-12 text-center">
                            <div class="big-tagline">

                                </div>

                            </div>
                        </div><!-- end row -->
                    </div><!-- end container -->
                </div><!-- end section -->
            </div>
            <div class="item">
                <div id="home" class="first-section" style="background-
image:url('https://www.livelaw.in/h-upload/2023/11/21/1600x960_505235-deepfake-
technology.jpg');">
                    <div class="container">
                        <div class="row">

```

```

        <div class="col-md-12 col-sm-12 text-center">
            <div class="big-tagline">

                </div>
            </div>
        </div><!-- end row -->
    </div><!-- end container -->
</div><!-- end section -->
</div>
<div class="item">
    <div id="home" class="first-section" style="background-
image:url('https://lawtrend.in/wp-content/uploads/2023/11/ai-deep-fake.jpeg');">
        <div class="container">
            <div class="row">
                <div class="col-md-12 col-sm-12 text-center">
                    <div class="big-tagline">

                        </div>
                    </div>
                </div><!-- end row -->
            </div><!-- end container -->
        </div><!-- end section -->
    </div>
    <!-- Left Control -->
    <a class="left carousel-control" href="#bootstrap-touch-slider" role="button"
data-slide="prev">
        <span class="fa fa-angle-left" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
    </a>

    <!-- Right Control -->
    <a class="right carousel-control" href="#bootstrap-touch-slider" role="button"
data-slide="next">
        <span class="fa fa-angle-right" aria-hidden="true"></span>
        <span class="sr-only">Next</span>
    </a>
</div>
</div>

<div id="overviews" class="section wb">
    <div class="container">
        <div class="section-title row text-center">
            <div class="col-md-8 col-md-offset-2">
                <h3 class="about">About</h3>
                <p class="lead">The advent of deepfake technology, allowing the creation of
fabricated images and videos with convincingly replaced or synthesized faces, has raised
significant societal concerns. This phenomenon poses risks, including the malicious generation of
false political news, dissemination of misleading information, fabrication of electronic evidence,
and perpetration of digital harassment and fraud. The utilization of face masks during the COVID-

```


19 pandemic has exacerbated the challenge, making it easier to create deepfakes while simultaneously complicating their detection. To address this evolving threat, this paper proposes a pioneering Deepfake Face Mask Dataset (DFFMD) and introduces a novel approach based on Inception-ResNet-v2, incorporating preprocessing stages, feature-based analysis, residual connections, and batch normalization. The study's results, compared with state-of-the-art methods, demonstrate heightened accuracy in detecting face-mask-enhanced deepfake videos, surpassing traditional methods like InceptionResNetV2 and VGG19. Furthermore, this research advocates for the integration of Convolutional Neural Networks (CNN) and an extension using Xception, underscoring their efficacy in enhancing deepfake detection accuracy. Future work should focus on evaluating the accuracy through subsequent experimental iterations, emphasizing the continued development of robust methods for increased detection of deepfakes with facemasks in the ever-evolving technological landscape.</p>

</div><!-- end title -->

</div><!-- end section -->

<footer class="footer">

<div class="container">

<div class="row">

<p>Copyright @ 2024</p>

</div><!-- end row -->

</div><!-- end container -->

</footer><!-- end footer -->

<!-- ALL JS FILES -->

<script src="static/js/all.js"></script>

<!-- ALL PLUGINS -->

<script src="static/js/custom.js"></script>

</body>

</html>

Signup.html

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags-->

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="Colorlib Templates">

<meta name="author" content="Colorlib">

<meta name="keywords" content="Colorlib Templates">

<!-- Title Page-->

<title>Signup</title>

```

<!-- Icons font CSS-->
<link href="static/assets/vendor/mdi-font/css/material-design-iconic-font.min.css"
rel="stylesheet" media="all">
<link href="static/assets/vendor/font-awesome-4.7/css/font-awesome.min.css" rel="stylesheet"
media="all">
<!-- Font special for pages-->
<link
href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
500i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

<!-- Vendor CSS-->
<link href="static/assets/vendor/select2/select2.min.css" rel="stylesheet" media="all">
<link href="static/assets/vendor/datepicker/daterangepicker.css" rel="stylesheet" media="all">

<!-- Main CSS-->
<link href="static/assets/css/main.css" rel="stylesheet" media="all">
<style>
    a:link, a:visited {
        background-color: rgb(196, 39, 136);
        color: rgb(245, 243, 244);
        border: 2px solid rgb(235, 238, 238);
        padding: 10px 20px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
    }

    a:hover, a:active {
        background-color: rgb(196, 39, 136);
        color: white;
    }
</style>
</head>

<body>
<div class="page-wrapper bg-gra-02 p-t-130 p-b-100 font-poppins">
    <div class="wrapper wrapper--w680">
        <div class="card card-4">
            <div class="card-body">
                <h2 class="title">Registration Form</h2>
                <form method="GET" action="/signup">
                    <div class="row row-space">
                        <div class="col-2">
                            <div class="input-group">
                                <label class="label">Username</label>
                                <input class="input--style-4" type="text" name="user" required>
                            </div>
                        </div>
                        <div class="col-2">
                            <div class="input-group">
                                <label class="label">Name</label>

```

```

        <input class="input--style-4" type="text" name="name" required>
    </div>
</div>
</div>

<div class="row row-space">
    <div class="col-2">
        <div class="input-group">
            <label class="label">Email</label>
            <input class="input--style-4" type="email" name="email" required>
        </div>
    </div>
    <div class="col-2">
        <div class="input-group">
            <label class="label">Phone Number</label>
            <input class="input--style-4" type="text" name="mobile" required>
        </div>
    </div>
</div>

<div class="row row-space">
    <div class="col-2">
        <div class="input-group">
            <label class="label">Password</label>
            <input class="input--style-4" type="password" name="password"
required>
        </div>
    </div>
</div>

<input class="form-row-last" type="submit" value="Register" class="btn btn-
block btn-primary"><br>

<div class="form-row-last">
    <p class="text-1">Click here for Signin
    </p>
    <a href="/login">Signin</a>
</div>
</form>
</div>
</div>
</div>
</div>

<!-- JQuery JS-->
<script src="static/assets/vendor/jquery/jquery.min.js"></script>
<!-- Vendor JS-->

```

```

<script src="static/assets/vendor/select2/select2.min.js"></script>
<script src="static/assets/vendor/date picker/moment.min.js"></script>
<script src="static/assets/vendor/date picker/daterangepicker.js"></script>

<!-- Main JS-->
<script src="static/assets/js/global.js"></script>

</body>
</html>
<!-- end document-->

```

Signin.html

```

<!DOCTYPE html>
<html lang="en">

<head>
  <!-- Required meta tags-->
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="Colorlib Templates">
  <meta name="author" content="Colorlib">
  <meta name="keywords" content="Colorlib Templates">

  <!-- Title Page-->
  <title>Signin</title>

  <!-- Icons font CSS-->
  <link href="static/assets/vendor/mdi-font/css/material-design-iconic-font.min.css"
rel="stylesheet" media="all">
  <link href="static/assets/vendor/font-awesome-4.7/css/font-awesome.min.css" rel="stylesheet"
media="all">
  <!-- Font special for pages-->
  <link
href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
500i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!-- Vendor CSS-->
  <link href="static/assets/vendor/select2/select2.min.css" rel="stylesheet" media="all">
  <link href="static/assets/vendor/date picker/daterangepicker.css" rel="stylesheet" media="all">

  <!-- Main CSS-->
  <link href="static/assets/css/main.css" rel="stylesheet" media="all">
  <style>
    a:link, a:visited {
      background-color: rgb(196, 39, 136);
      color: rgb(245, 243, 244);
      border: 2px solid rgb(235, 238, 238);
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;

```

```

        display: inline-block;
    }

    a:hover, a:active {
        background-color: rgb(196, 39, 136);
        color: white;
    }
</style>
</head>

<body>
    <div class="page-wrapper bg-gra-02 p-t-130 p-b-100 font-poppins">
        <div class="wrapper wrapper--w680">
            <div class="card card-4">
                <div class="card-body">
                    <h2 class="title">Login Form</h2>
                    <form method="GET" action="/signin">
                        <div class="row row-space">
                            <div class="col-2">
                                <div class="input-group">
                                    <label class="label">USERNAME:</label>
                                    <input class="input--style-4" type="text" name="user">
                                </div>
                            </div>
                            <div class="col-2">
                                <div class="input-group">
                                    <label class="label">PASSWORD</label>
                                    <input class="input--style-4" type="password" name="password">
                                </div>
                            </div>
                        </div>

                        <div class="p-t-15">
                            <button class="btn btn--radius-2 btn--blue" type="submit">Submit</button>
                        </div>
                    </form>
                </div>
                <div class="form-row-last">
                    <p class="text-1">Click here for SignUp
                        <a href="/logon">Signup</a>
                    </p>
                </div>
            </div>
        </div>
    </div>
    <div class="p-t-15">
        <button class="btn btn--radius-2 btn--blue" type="submit">Submit</button>
    </div>
</br>

    <div class="form-row-last">
        <p class="text-1">Click here for SignUp
            <a href="/logon">Signup</a>
        </p>
    </div>
</form>
</div>
</div>
</div>
</div>
</div>

<!-- JQuery JS-->
<script src="static/assets/vendor/jquery/jquery.min.js"></script>
<!-- Vendor JS-->

```

```

<script src="static/assets/vendor/select2/select2.min.js"></script>
<script src="static/assets/vendor/date picker/moment.min.js"></script>
<script src="static/assets/vendor/date picker/daterangepicker.js"></script>

<!-- Main JS-->
<script src="static/assets/js/global.js"></script>

</body>
</html>
<!-- end document-->

```

Index.html

```

<!DOCTYPE html>
<html lang="en">

  <!-- Basic -->
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <!-- Mobile Metas -->
  <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-
scale=1.0, user-scalable=no">

  <!-- Site Metas -->
  <title>Home</title>
  <meta name="keywords" content="">
  <meta name="description" content="">
  <meta name="author" content="">

  <!-- Site Icons -->
  <link rel="shortcut icon" href="static/images/favicon.ico" type="image/x-icon" />
  <link rel="apple-touch-icon" href="static/images/apple-touch-icon.png">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="static/css/bootstrap.min.css">
  <!-- Site CSS -->
  <link rel="stylesheet" href="static/style.css">
  <!-- Colors CSS -->
  <link rel="stylesheet" href="static/css/colors.css">
  <!-- ALL VERSION CSS -->
  <link rel="stylesheet" href="static/css/versions.css">
  <!-- Responsive CSS -->
  <link rel="stylesheet" href="static/css/responsive.css">
  <!-- Custom CSS -->
  <link rel="stylesheet" href="static/css/custom.css">

  <!-- Modernizer for Portfolio -->
  <script src="static/js/modernizer.js"></script>

  <!--[if lt IE 9]>

```

```

    <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
    <script src="https://oss.maxcdn.com/libs/respond.static/js/1.4.2/respond.min.js"></script>
<![endif]-->

<style>
    #home {
        background-image: url('static/uploads/slider-01.jpg');
        background-repeat: no-repeat;
        background-size: cover;
        /* You can also set background-position to center if needed */
        /* Example:
        background-position: center;
        */
    }
</style>

</head>
<body class="host_version">

    <!-- LOADER -->
    <div id="preloader">
        <div class="loading">
            <div class="finger finger-1">
                <div class="finger-item">
                    <span></span><i></i>
                </div>
            </div>
            <div class="finger finger-2">
                <div class="finger-item">
                    <span></span><i></i>
                </div>
            </div>
            <div class="finger finger-3">
                <div class="finger-item">
                    <span></span><i></i>
                </div>
            </div>
            <div class="finger finger-4">
                <div class="finger-item">
                    <span></span><i></i>
                </div>
            </div>
            <div class="last-finger">
                <div class="last-finger-item"><i></i></div>
            </div>
        </div>
    </div>
    <!-- END LOADER -->

```

```

<header class="header header_style_01">
  <nav class="megamenu navbar navbar-default">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <!-- <a class="navbar-brand" href="index.html"></a -->
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li><a class="active" href="notebook">Notebook</a></li>
          <li><a href="login">Logout</a></li>

        </ul>

      </div>
    </div>
  </nav>
</header>

```

```

<div id="bootstrap-touch-slider" class="carousel bs-slider fade control-round indicators-line"
data-ride="carousel" data-pause="hover" data-interval="false" >
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#bootstrap-touch-slider" data-slide-to="0" class="active"></li>
    <li data-target="#bootstrap-touch-slider" data-slide-to="1"></li>
    <li data-target="#bootstrap-touch-slider" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner" role="listbox">
    <div class="item active">
      <div id="home" class="first-section" style="background-
image:url('static/uploads/slider-01.jpg');">
        <div class="container">
          <div class="row">
            <div class="col-md-12 col-sm-12 text-center">
              <div class="big-tagline">

            </div>
          </div>
        </div><!-- end row -->
      </div><!-- end container -->
    </div><!-- end section -->
  </div>
<div class="item">

```



```

        <div id="home" class="first-section" style="background-
image:url('static/uploads/slider-02.jpg');">
            <div class="container">
                <div class="row">
                    <div class="col-md-12 col-sm-12 text-center">
                        <div class="big-tagline">

                                </div>
                            </div>
                        </div><!-- end row -->
                    </div><!-- end container -->
                </div><!-- end section -->
            </div>
            <div class="item">
                <div id="home" class="first-section" style="background-
image:url('static/uploads/slider-03.jpg');">
                    <div class="container">
                        <div class="row">
                            <div class="col-md-12 col-sm-12 text-center">
                                <div class="big-tagline">

                                        </div>
                                    </div>
                                </div><!-- end row -->
                            </div><!-- end container -->
                        </div><!-- end section -->
                    </div>
                    <!-- Left Control -->
                    <a class="left carousel-control" href="#bootstrap-touch-slider" role="button"
data-slide="prev">
                        <span class="fa fa-angle-left" aria-hidden="true"></span>
                        <span class="sr-only">Previous</span>
                    </a>

                    <!-- Right Control -->
                    <a class="right carousel-control" href="#bootstrap-touch-slider" role="button"
data-slide="next">
                        <span class="fa fa-angle-right" aria-hidden="true"></span>
                        <span class="sr-only">Next</span>
                    </a>
                </div>
            </div>
            <center>
                <form class="form-signin" method=post action="/predict" enctype=multipart/form-data
name="form1">
                    <!--  -->
                    <h1 class="h3 mb-3 font-weight-normal">Upload video here!</h1>

```

```

<input type="file" name="file" class="form-control-file" id="inputfile" >

<br/>

<button type="submit">Upload</button>

</form>
</center>

<!-- detected video display using opencv-->

<footer class="footer">
<div class="container">
<div class="row">

</div><!-- end row -->
</div><!-- end container -->
</footer><!-- end footer -->
</body>
</html>

```

Display.html

```

<!DOCTYPE html>
<html lang="en">

  <!-- Basic -->
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <!-- Mobile Metas -->
  <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum
scale=1.0, user-scalable=no">

  <!-- Site Metas -->
  <title>Home</title>
  <meta name="keywords" content="">
  <meta name="description" content="">
  <meta name="author" content="">

  <!-- Site Icons -->
  <link rel="shortcut icon" href="static/images/favicon.ico" type="image/x-icon" />
  <link rel="apple-touch-icon" href="static/images/apple-touch-icon.png">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="static/css/bootstrap.min.css">
  <!-- Site CSS -->
  <link rel="stylesheet" href="static/style.css">
  <!-- Colors CSS -->
  <link rel="stylesheet" href="static/css/colors.css">
  <!-- ALL VERSION CSS -->

```

```

<link rel="stylesheet" href="static/css/versions.css">
<!-- Responsive CSS -->
<link rel="stylesheet" href="static/css/responsive.css">
<!-- Custom CSS -->
<link rel="stylesheet" href="static/css/custom.css">

<!-- Modernizer for Portfolio -->
<script src="static/js/modernizer.js"></script>

<!--[if lt IE 9]>
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
  <script src="https://oss.maxcdn.com/libs/respond.static/js/1.4.2/respond.min.js"></script>
<![endif]-->

<style>
  #home {
    background-image: url('static/uploads/slider-01.jpg');
    background-repeat: no-repeat;
    background-size: cover;
    /* You can also set background-position to center if needed */
    /* Example:
    background-position: center;
    */
  }
</style>

</head>
<body class="host_version">

  <!-- LOADER -->
  <div id="preloader">
    <div class="loading">
      <div class="finger finger-1">
        <div class="finger-item">
          <span></span><i></i>
        </div>
      </div>
      <div class="finger finger-2">
        <div class="finger-item">
          <span></span><i></i>
        </div>
      </div>
      <div class="finger finger-3">
        <div class="finger-item">
          <span></span><i></i>
        </div>
      </div>
      <div class="finger finger-4">
        <div class="finger-item">

```

```

        <span></span><i></i>
      </div>
    </div>
    <div class="last-finger">
      <div class="last-finger-item"><i></i></div>
    </div>
  </div>
</div>
<!-- END LOADER -->

<header class="header header_style_01">
  <nav class="megamenu navbar navbar-default">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <!-- <a class="navbar-brand" href="index.html"></a> -->
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li><a class="active" href="index">Try Again?</a></li>
          <li><a class="active" href="notebook">Notebook</a></li>
          <li><a href="login">Logout</a></li>

        </ul>

      </div>
    </div>
  </nav>
</header>

<div id="bootstrap-touch-slider" class="carousel bs-slider fade control-round indicators-line"
data-ride="carousel" data-pause="hover" data-interval="false" >
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#bootstrap-touch-slider" data-slide-to="0" class="active"></li>
    <li data-target="#bootstrap-touch-slider" data-slide-to="1"></li>
    <li data-target="#bootstrap-touch-slider" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner" role="listbox">
    <div class="item active">
      <div id="home" class="first-section" style="background-
image:url('static/uploads/slider-01.jpg');">
        <div class="container">
          <div class="row">
            <div class="col-md-12 col-sm-12 text-center">

```

```

<div class="big-tagline">

</div>
</div>
</div><!-- end row -->
</div><!-- end container -->
</div><!-- end section -->
</div>
<div class="item">
    <div id="home" class="first-section" style="background-
image:url('static/uploads/slider-02.jpg');">
        <div class="container">
            <div class="row">
                <div class="col-md-12 col-sm-12 text-center">
                    <div class="big-tagline">

</div>
</div>
</div><!-- end row -->
</div><!-- end container -->
</div><!-- end section -->
</div>
<div class="item">
    <div id="home" class="first-section" style="background-
image:url('static/uploads/slider-03.jpg');">
        <div class="container">
            <div class="row">
                <div class="col-md-12 col-sm-12 text-center">
                    <div class="big-tagline">

</div>
</div>
</div><!-- end row -->
</div><!-- end container -->
</div><!-- end section -->
</div>
<!-- Left Control -->
<a class="left carousel-control" href="#bootstrap-touch-slider" role="button"
data-slide="prev">
    <span class="fa fa-angle-left" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
</a>

<!-- Right Control -->
<a class="right carousel-control" href="#bootstrap-touch-slider" role="button"

```

```

data-slide="next">
    <span class="fa fa-angle-right" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
</div>
<center>
<h1>For the given input video is {{result}}</h1>
</center>
<footer class="footer">
<div class="container">
  <div class="row">

  </div><!-- end row -->
</div><!-- end container -->
</footer><!-- end footer -->
</body>
</html>

```

Style.css

```

@import url(css/animate.css);
@import url(css/animate.min.css);
@import url(css/pr_loading.css);
@import url(css/bootstrap-touch-slider.css);
@import url(css/flaticon.css);
@import url(css/prettyPhoto.css);
@import url(css/owl.carousel.css);
@import url(css/font-awesome.min.css);

/*-----
  SKELETON
  -----*/

body {
  color: #999;
  font-size: 15px;
  font-family: 'PT Sans', sans-serif;
  line-height: 1.80857;
}

body.demos .section {
  background: url(images/bg.png) repeat top center #f2f3f5;
}

body.demos .section-title img {
  max-width: 280px;
  display: block;
  margin: 10px auto;
}

```

```
body.demos .service-widget h3 {
  border-bottom: 1px solid #ededed;
  font-size: 18px;
  padding: 20px 0;
  background-color: #ffffff;
}
```

```
body.demos .service-widget {
  margin: 0 0 30px;
  padding: 30px;
  background-color: #fff
}
```

```
body.demos .container-fluid {
  max-width: 1080px
}
```

```
a {
  color: #1f1f1f;
  text-decoration: none !important;
  outline: none !important;
  -webkit-transition: all .3s ease-in-out;
  -moz-transition: all .3s ease-in-out;
  -ms-transition: all .3s ease-in-out;
  -o-transition: all .3s ease-in-out;
  transition: all .3s ease-in-out;
}
```

```
h1,
h2,
h3,
h4,
h5,
h6 {
  letter-spacing: 0;
  font-weight: normal;
  position: relative;
  padding: 0 0 10px 0;
  font-weight: normal;
  line-height: 120% !important;
  color: #1f1f1f;
  margin: 0
}
```

```
h1 {
  font-size: 24px
}
```

```
h2 {
  font-size: 22px
}
```

```

h3 {
  font-size: 18px
}

h4 {
  font-size: 16px
}

h5 {
  font-size: 14px
}

h6 {
  font-size: 13px
}

h1 a,
h2 a,
h3 a,
h4 a,
h5 a,
h6 a {
  color: #212121;
  text-decoration: none !important;
  opacity: 1
}

h1 a:hover,
h2 a:hover,
h3 a:hover,
h4 a:hover,
h5 a:hover,
h6 a:hover {
  opacity: .8
}

a {
  color: #1f1f1f;
  text-decoration: none;
  outline: none;
}

a,
.btn {
  text-decoration: none !important;
  outline: none !important;
  -webkit-transition: all .3s ease-in-out;
  -moz-transition: all .3s ease-in-out;
  -ms-transition: all .3s ease-in-out;
  -o-transition: all .3s ease-in-out;

```



```

    transition: all .3s ease-in-out;
}

.btn-custom {
    margin-top: 20px;
    background-color: transparent !important;
    border: 2px solid #ddd;
    padding: 12px 40px;
    font-size: 16px;
}

.lead {
    font-size: 18px;
    line-height: 30px;
    color: #767676;
    margin: 0;
    padding: 0;
}

blockquote {
    margin: 20px 0 20px;
    padding: 30px;
}

/*-----
WP CORE
-----*/

.first {
    clear: both
}

.last {
    margin-right: 0
}

.alignnone {
    margin: 5px 20px 20px 0;
}

.aligncenter,
div.aligncenter {
    display: block;
    margin: 5px auto 5px auto;
}

.alignright {
    float: right;
    margin: 10px 0 20px 20px;
}

```

```

.alignleft {
    float: left;
    margin: 10px 20px 20px 0;
}

a img.alignright {
    float: right;
    margin: 10px 0 20px 20px;
}

a img.alignnone {
    margin: 10px 20px 20px 0;
}

a img.alignleft {
    float: left;
    margin: 10px 20px 20px 0;
}

a img.aligncenter {
    display: block;
    margin-left: auto;
    margin-right: auto;
}

.wp-caption {
    background: #fff;
    border: 1px solid #f0f0f0;
    max-width: 96%;
    /* Image does not overflow the content area */
    padding: 5px 3px 10px;
    text-align: center;
}

.wp-caption.alignnone {
    margin: 5px 20px 20px 0;
}

.wp-caption.alignleft {
    margin: 5px 20px 20px 0;
}

.wp-caption.alignright {
    margin: 5px 0 20px 20px;
}

.wp-caption img {
    border: 0 none;
    height: auto;
    margin: 0;
    max-width: 98.5%;
}

```

```

padding: 0;
width: auto;
}

.wp-caption p.wp-caption-text {
font-size: 11px;
line-height: 17px;
margin: 0;
padding: 0 4px 5px;
}

/* Text meant only for screen readers. */

.screen-reader-text {
clip: rect(1px, 1px, 1px, 1px);
position: absolute !important;
height: 1px;
width: 1px;
overflow: hidden;
}

.screen-reader-text:focus {
background-color: #f1f1f1;
border-radius: 3px;
box-shadow: 0 0 2px 2px rgba(0, 0, 0, 0.6);
clip: auto !important;
color: #21759b;
display: block;
font-size: 14px;
font-size: 0.875rem;
font-weight: bold;
height: auto;
left: 5px;
line-height: normal;
padding: 15px 23px 14px;
text-decoration: none;
top: 5px;
width: auto;
z-index: 100000;
/* Above WP toolbar. */
}

/*-----
HEADER
-----*/

.megamenu .nav,
.megamenu .collapse,
.megamenu .dropup,
.megamenu .dropdown {
position: static;

```

```

}

.megamenu .container-fluid {
    position: relative;
}

.megamenu .dropdown-menu {
    left: auto;
}

.megamenu .megamenu-content {
    padding: 20px 30px;
}

.megamenu .dropdown.megamenu-fw .dropdown-menu {
    left: 0;
    right: 0;
}

.megamenu .list-unstyled {
    min-width: 200px;
}

.header_style_01 {
    background-color: rgba(255, 255, 255, 0.1);
    box-shadow: 0 0 8px 0 rgba(0,0,0,.12);
    display: block;
    left: 0;
    padding: 20px 40px !important;
    position: absolute;
    right: 0;
    top: 0;
    width: 100%;
    z-index: 111;
}

.header_style_01 .navbar-default {
    background-color: transparent;
    border: none;
    border-radius: 0;
    /* padding: 15px 32px;
    display: inline-block;
    color: red; */
}

.header_style_01 .navbar,
.header_style_01 .navbar-nav,
.header_style_01 .navbar-default,
.header_style_01 .nav {
    margin-bottom: 0 !important;
}

```

```

.header_style_01 .navbar-brand {
  padding: 2px 15px 0 15px;
}

.header_style_01 .navbar-default .navbar-nav > li > a {
  color: #ffffff;
  font-size: 15px;
  font-style: normal;
  font-weight: 400;
  text-transform: capitalize;
}

.geneartive{
  font-size: 50px;
  font-weight: bolder;
  position: relative;
}

.header_style_01 .navbar-default .navbar-nav > li a {
  background-color: transparent;
  padding: 7px 7px;
  background-color: rgb(51, 177, 190);
  display: inline-block;
  border: 4px;
  color: #000;
  border-radius: 3px;
  margin-right: 10px;
  justify-content: space-between;
}

body.host_version .header_style_01 .navbar-default .navbar-nav > li: hover a,
body.host_version .header_style_01 .navbar-default .navbar-nav > li: focus a {
  color: #000;
}

body.host_version .header_style_01 .navbar-default .navbar-nav > li a.active{
  color: #000;
}

.header_style_01 .navbar-right > li {
  margin-top: 2px;
  -webkit-transition: all .3s ease-in-out;
  -moz-transition: all .3s ease-in-out;
  -ms-transition: all .3s ease-in-out;
  -o-transition: all .3s ease-in-out;
  transition: all .3s ease-in-out;
}

body.host_version .header_style_01 .navbar-right > li .log: hover{

```

```

    background: #00aeef;
    color: #fff !important;
    border-color: #00aeef !important;
}

.header_style_01 .navbar-right > li > a {
    padding-bottom: 10px;
    padding-top: 10px;
}

li.social-links {
    margin: 0 8px;
}

li.social-links a {
    padding: 13px 0 !important;
}

.affix-top {
    overflow: hidden;
    visibility: visible;
    opacity: 1;
    top: -100%;
}

.affix {
    top: 0;
    left: 0;
    right: 0;
    width: 100%;
    padding: 20px 40px;
    background-color: #1f1f1f !important;
    -webkit-transition: visibility 0.95s ease-in-out, opacity 0.95s ease-in-out, bottom 0.95s ease-in-out, top 0.95s ease-in-out, left 0.95s ease-in-out, right 0.95s ease-in-out;
    -moz-transition: visibility 0.95s ease-in-out, opacity 0.95s ease-in-out, bottom 0.95s ease-in-out, top 0.95s ease-in-out, left 0.95s ease-in-out, right 0.95s ease-in-out;
    -o-transition: visibility 0.95s ease-in-out, opacity 0.95s ease-in-out, bottom 0.95s ease-in-out, top 0.95s ease-in-out, left 0.95s ease-in-out, right 0.95s ease-in-out;
    transition: visibility 0.95s ease-in-out, opacity 0.95s ease-in-out, top 0.95s ease-in-out, bottom 0.95s ease-in-out, left 0.95s ease-in-out, right 0.95s ease-in-out;
}

.navbar-nav li {
    position: relative;
}

.navbar-nav span {
    font-size: 24px;
    position: absolute;
    right: 2px;
    top: 13px;
}

```

```
}
```

```
.fixed-menu .navbar-default{  
  position: fixed;  
  visibility: hidden;  
  left: 0px;  
  top: 0px;  
  width: 100%;  
  padding: 0px 0px;  
  background: #ffffff;  
  z-index: 0;  
  transition: all 500ms ease;  
  -moz-transition: all 500ms ease;  
  -webkit-transition: all 500ms ease;  
  -ms-transition: all 500ms ease;  
  -o-transition: all 500ms ease;  
  z-index: 999;  
  opacity: 1;  
  visibility: visible;  
  -ms-animation-name: fadeInDown;  
  -moz-animation-name: fadeInDown;  
  -op-animation-name: fadeInDown;  
  -webkit-animation-name: fadeInDown;  
  animation-name: fadeInDown;  
  -ms-animation-duration: 500ms;  
  -moz-animation-duration: 500ms;  
  -op-animation-duration: 500ms;  
  -webkit-animation-duration: 500ms;  
  animation-duration: 500ms;  
  -ms-animation-timing-function: linear;  
  -moz-animation-timing-function: linear;  
  -op-animation-timing-function: linear;  
  -webkit-animation-timing-function: linear;  
  animation-timing-function: linear;  
  -ms-animation-iteration-count: 1;  
  -moz-animation-iteration-count: 1;  
  -op-animation-iteration-count: 1;  
  -webkit-animation-iteration-count: 1;  
  animation-iteration-count: 1;  
}
```

```
.fixed-menu .navbar-default{  
  padding: 15px 0px;  
  box-shadow: 0 0 8px 0 rgba(0,0,0,.12);  
}
```

```
/*-----  
  SECTIONS  
-----*/
```

```
.parallax {
```

```

background-attachment: fixed;
background-size: cover;
height: 100%;
padding: 120px 0;
position: relative;
width: 100%;
}

.parallax.parallax-off {
background-attachment: fixed !important;
display: block;
height: 100%;
min-height: 100%;
overflow: hidden;
position: relative;
background-position: center center;
vertical-align: sub;
width: 100%;
z-index: 2;
}

.no-scroll-xy {
overflow: hidden !important;
-webkit-transition: all .4s ease-in-out;
-moz-transition: all .4s ease-in-out;
-ms-transition: all .4s ease-in-out;
-o-transition: all .4s ease-in-out;
transition: all .4s ease-in-out;
}

.section {
display: block;
position: relative;
overflow: hidden;
padding: 100px 0;
}
/* .about {
font-size: 10px;
font-weight: bolder;
} */
.noover {
overflow: visible;
}

.noover .btn-dark {
border: 0 !important;
}

.nopad {
padding: 0;
}

```



```

.nopadtop {
  padding-top: 0;
}

.section.wb {
  background-color: #ffffff;
}

.section.lb {
  background-color: #f2f3f5;
}

.section.db {
  background-color: #1f1f1f;
}

.section.color1 {
  background-color: #448AFF;
}


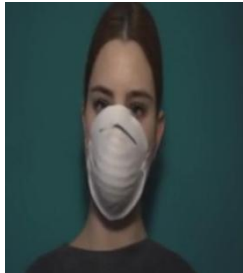


.section.cl {
  background-color: #2d3032;
}

.first-section {
  display: 100%;
  position: relative;
  overflow: hidden;
  padding: 16em 0 13em;
  background-size: cover;
  height: auto;
}

.first-section::before {
  content: "";
  position: absolute;
  height: 100%;
  width: 100%;
  top: 0px;
  left: 0px;
  display: 100%;
  /* background: linear-gradient(to right,rgba(255,255,255,0.99) 20%,rgba(255,255,255,0.7)
70%,rgba(255,255,255,0) 95%); */
}

```

6. TESTCASES

| Test Case ID | Input (Video as input) | Expected Output | Actual Output | Rate |
|--------------|---|-------------------------------|-------------------------------|---------|
| 1 | Login Credentials(Authorized) | Login Access | Login Access | Success |
| 2 | Login Credentials (Unauthorized) | Login Denied | Login Denied | Success |
| 2 |  | The given video is 'Real' | The given video is 'Real' | Success |
| 3 |  | The given video is 'Fake' | The given video is 'Fake' | Success |
| 4 |  | Without mask can't predict | Without mask can't predict | Success |
| 5 |  <small>shutterstock.com · 1679563561</small> | Upload video not image | Upload video not image | Success |

7. SCREENSHOTS

7.1 Dataset

The deep fake face mask detection dataset, sourced from Kaggle, comprises two main folders: "fake" and "real," each containing distinct videos for training and evaluation purposes. The "fake" folder encompasses a total of 1000 videos, specifically curated to simulate instances where facial features are manipulated through the application of deep fake technology, particularly the addition of face masks. On the other hand, the "real" folder consists of 836 videos, capturing authentic facial expressions without the influence of deep fake alterations. This dataset is designed to facilitate the development and evaluation of algorithms and models for detecting the presence of face masks in videos, a task that holds significant relevance in the context of public health and safety, particularly during periods of widespread mask-wearing. Researchers and practitioners in the field of computer vision and deep learning can leverage this dataset to enhance the accuracy and robustness of face mask detection systems, contributing to the ongoing efforts to address public health challenges.

Dataset related to fake videos i.e Deep Fake Face Mask Videos:

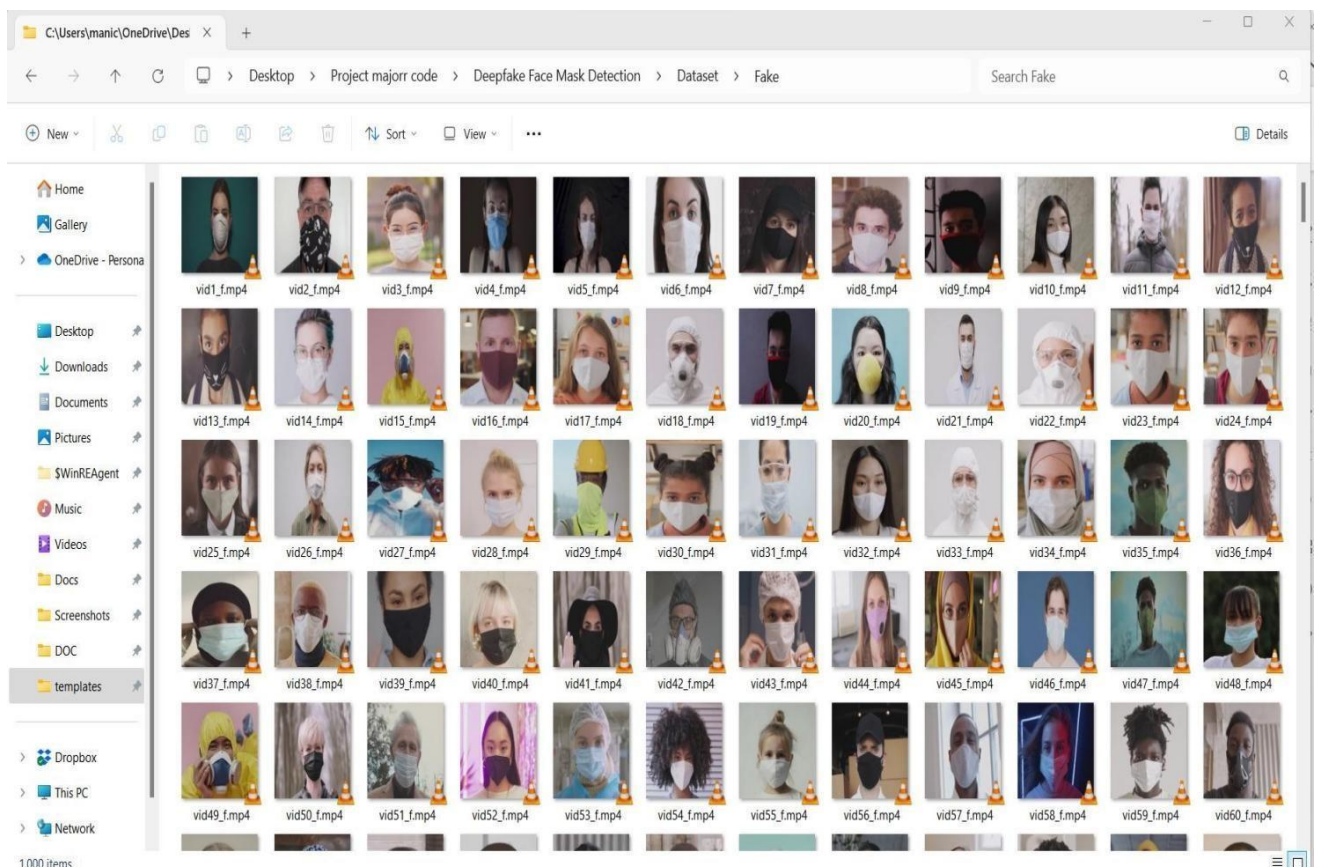


Fig 7.1.1 Sample Dataset of Fake Videos

Dataset related to real videos:

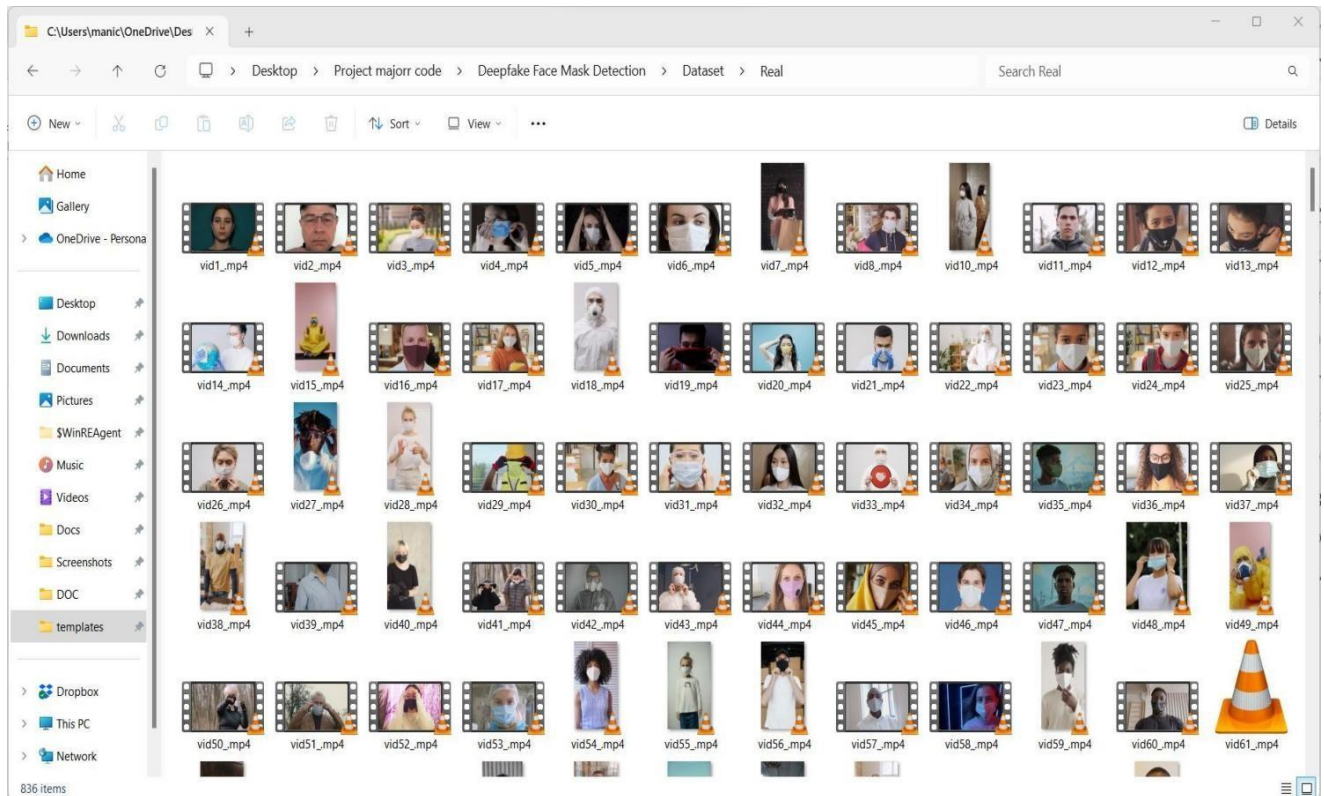


Fig 7.1.2 Sample Dataset of Real Videos

7.2 Testing Output

Home Page: In home page we will be having 2 buttons i.e, SignUp and Signin (7.2.1a) and the introduction information about the project(7.2.1b).

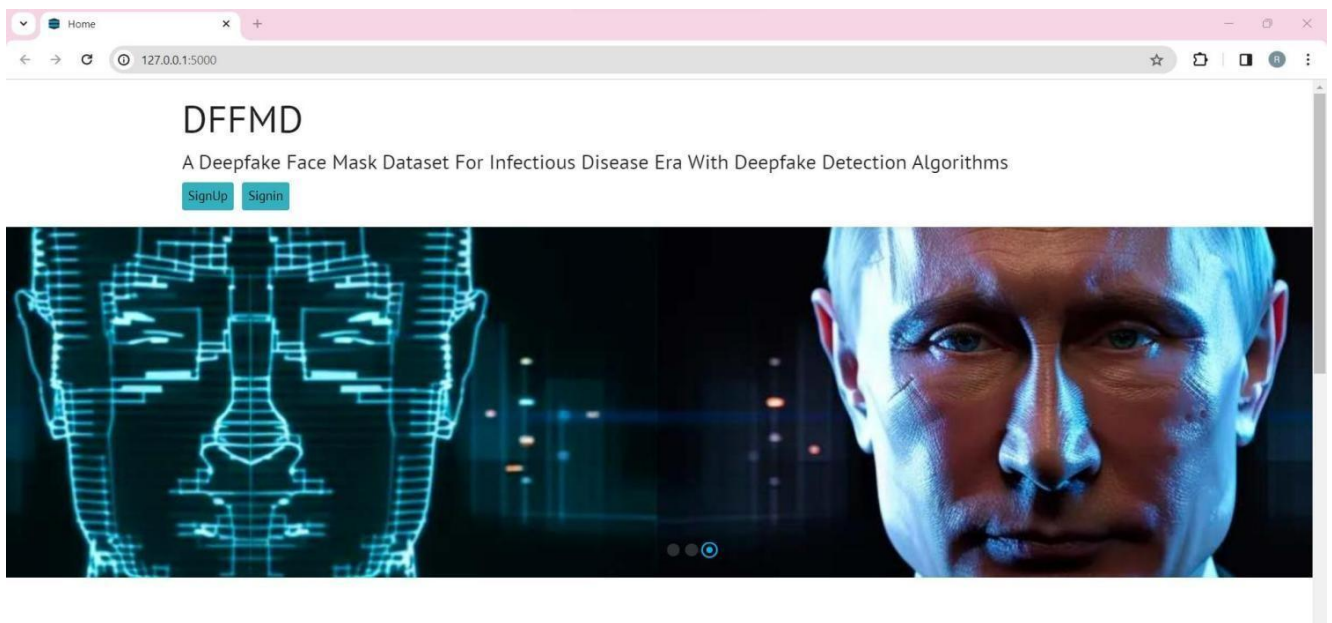


Fig 7.2.1a Home Page

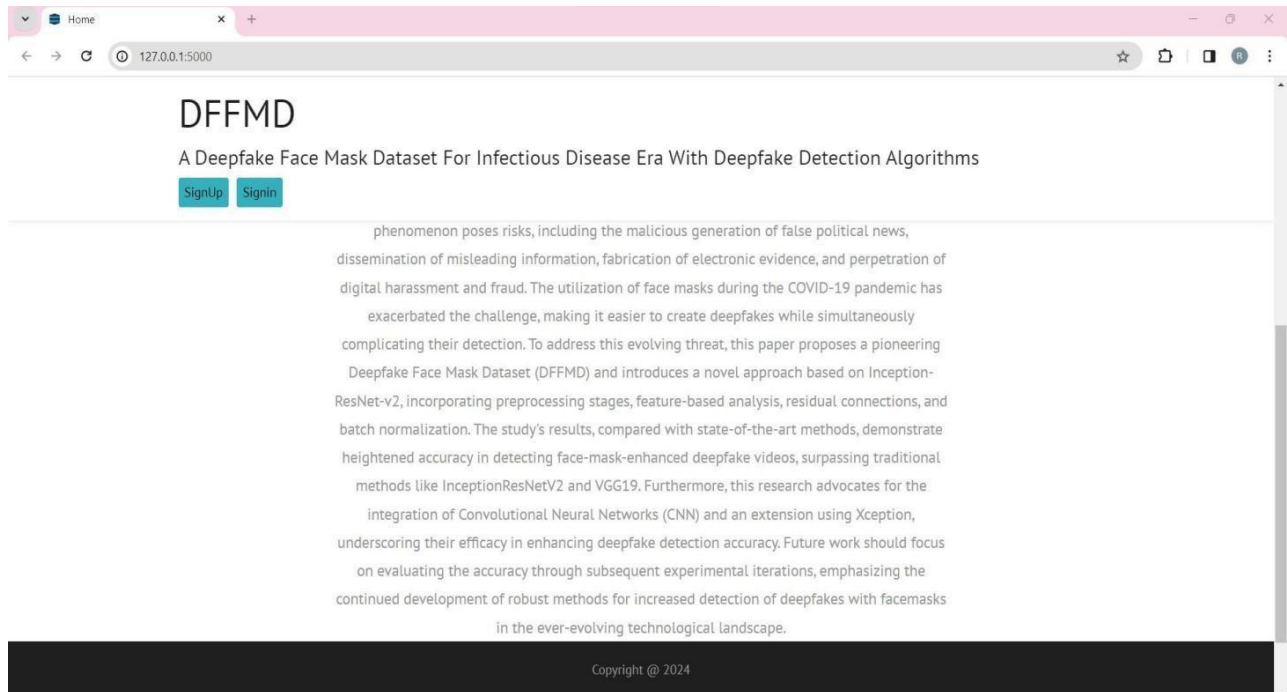


Fig 7.2.1b Home Page

Signup Page: In signup page we will have username, name, email, phone number, password. We should create all of them and registered using a register button

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/login'. The page has a blue and purple gradient background. A white registration form is centered, containing the following fields and buttons:

- Username:
- Name:
- Email:
- Phone Number:
- Password:
- Register:
- Click here for Signin: [Click here for Signin](#)
- SignIn:

Fig 7.2.2 SignUp Page

SignIn Page: In Signin page we will have username and password. We should enter correctly whatever we created in Signup page or else it will not open the upload page i.e, Access will denied.

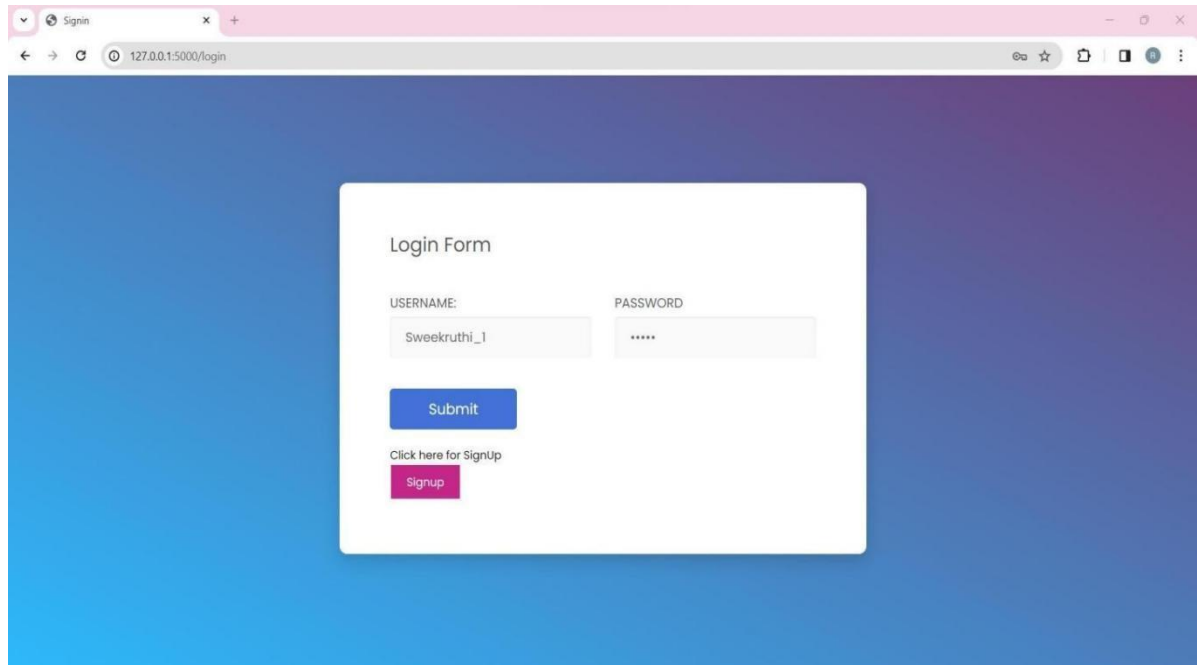


Fig 7.2.3 SignIn Page

Upload Page: In Upload page we have upload button where we have to upload a video.

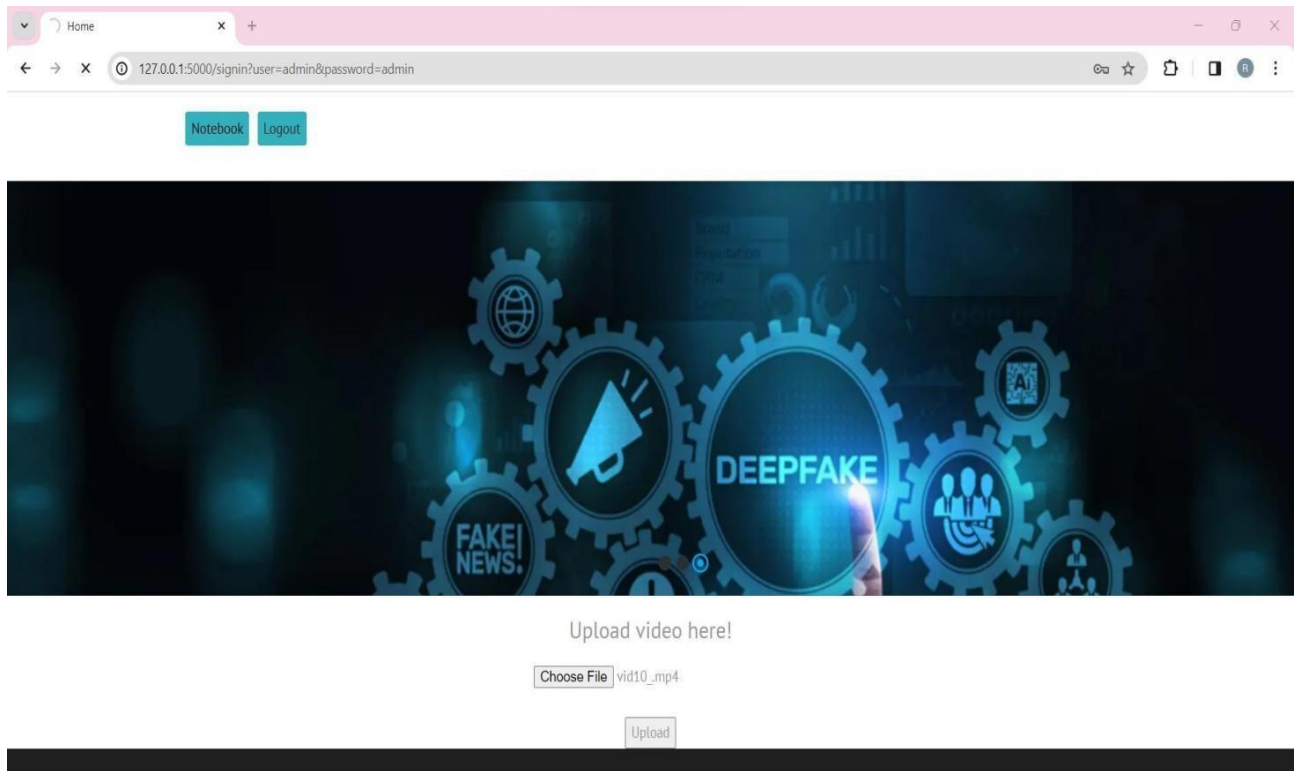


Fig 7.2.4 Upload Page

Result Page: If the uploaded video is “REAL” then the output will be shown like 7.2.5a, or if it is “FAKE” then the result will be shown like 7.2.5b.

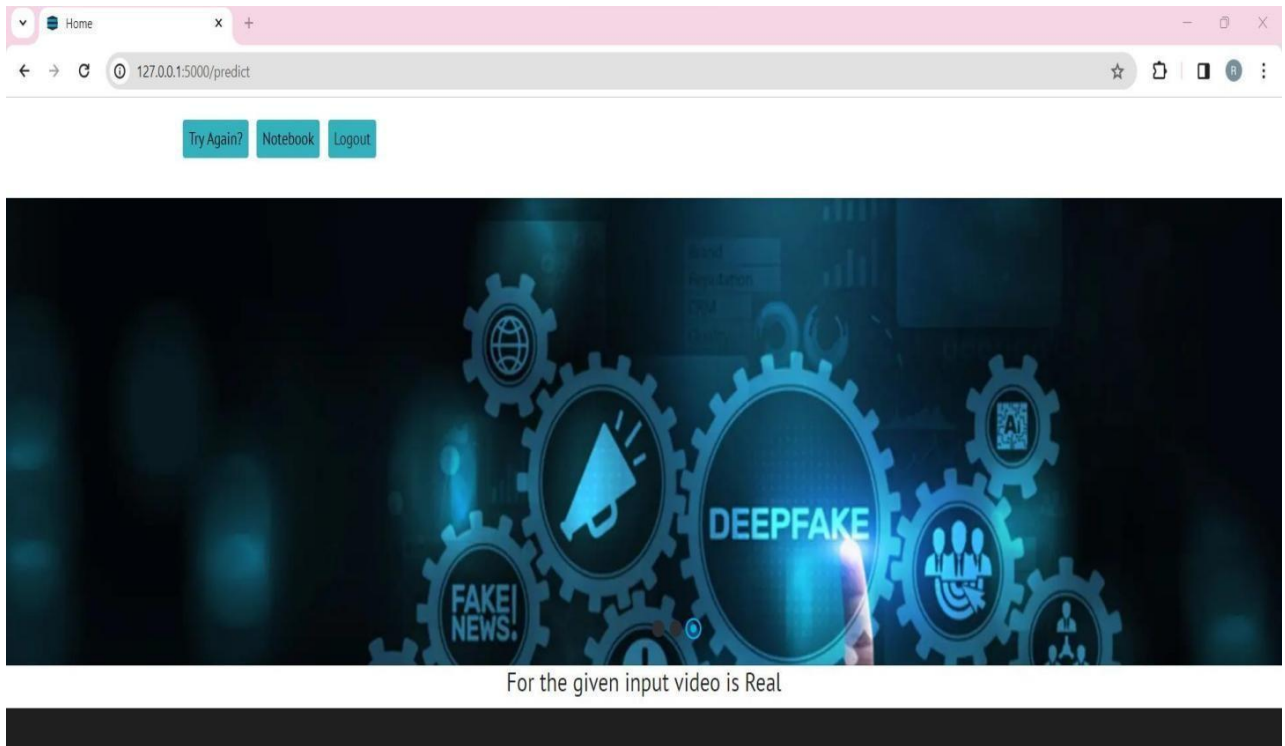


Fig 7.2.5a Result Page

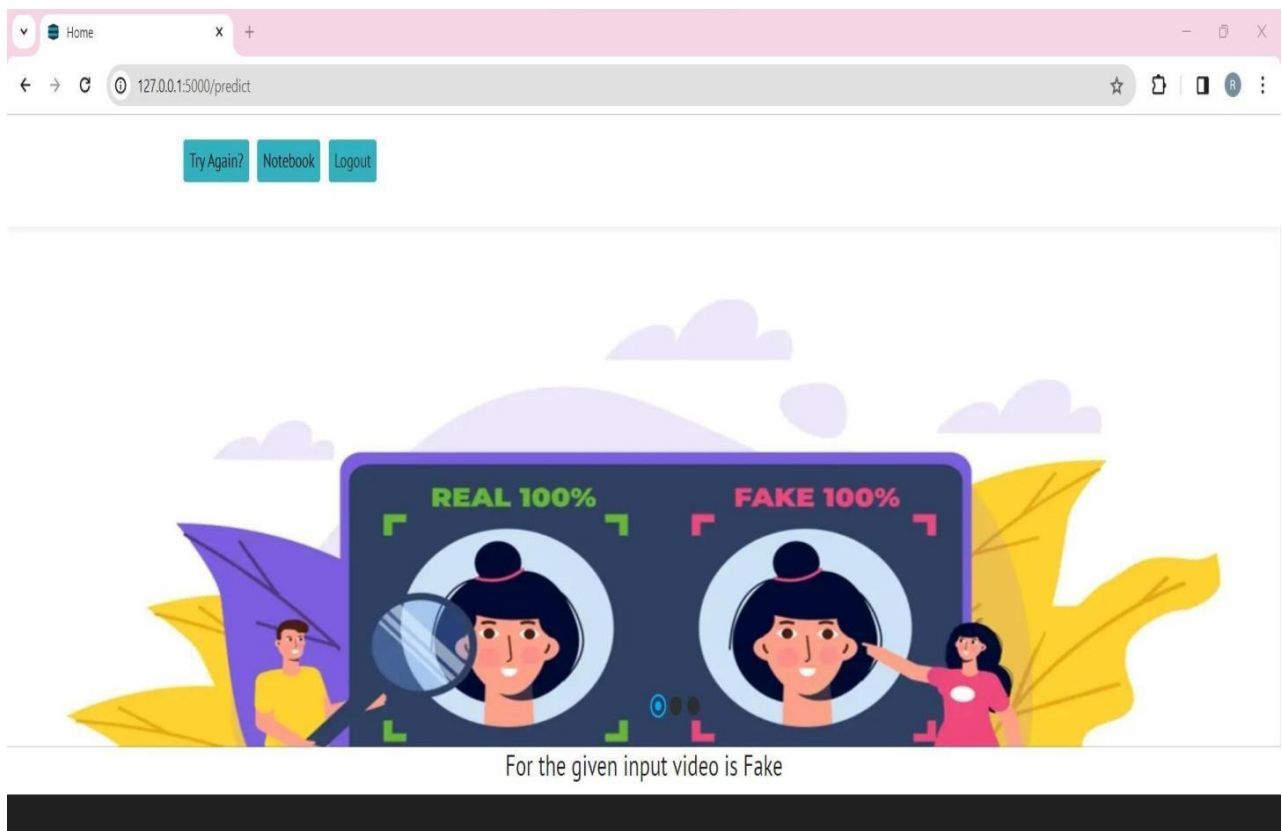


Fig 7.2.5b Result Page

7.3 Comparison Graphs

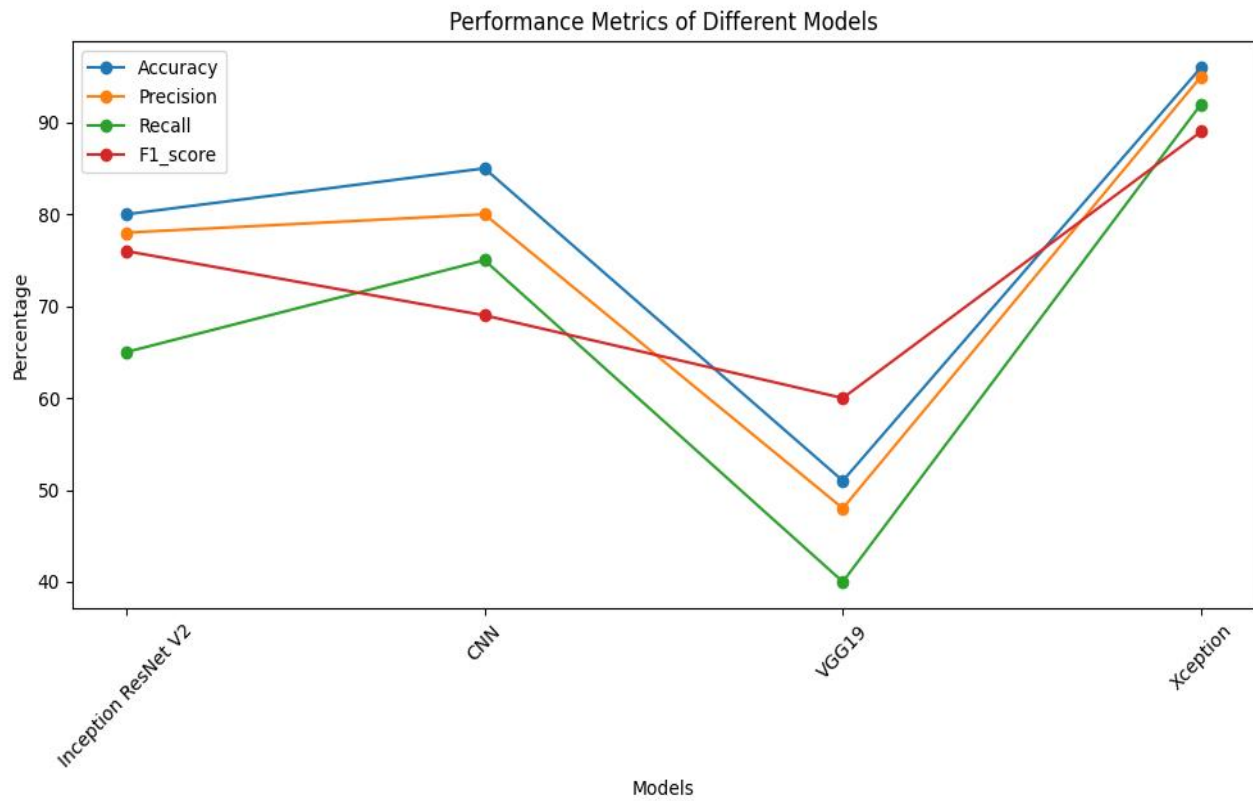


Fig 7.3 Overall Models Comparison Graph

7.3.1 Inception ResNet V2

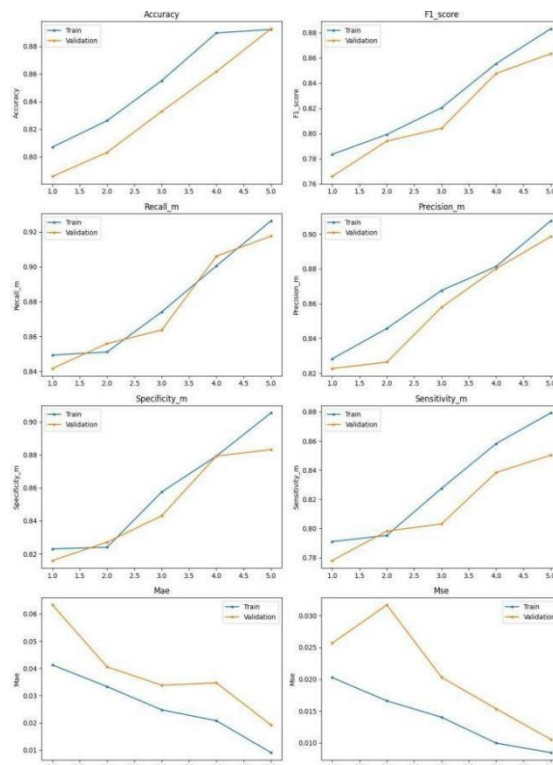


Fig 7.3.1 Inception ResNet V2 Model Graph

7.3.2 VGG 19

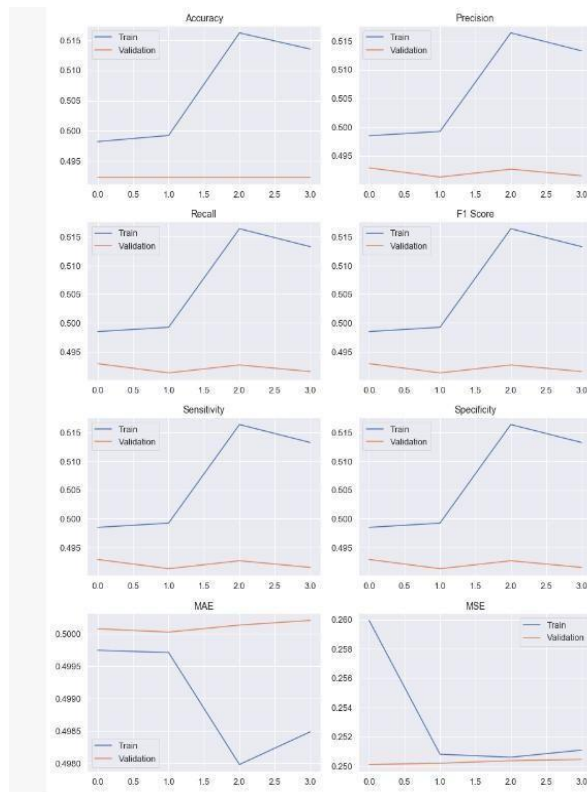


Fig 7.3.2 VGG 19 Model Graph

7.3.3 CNN

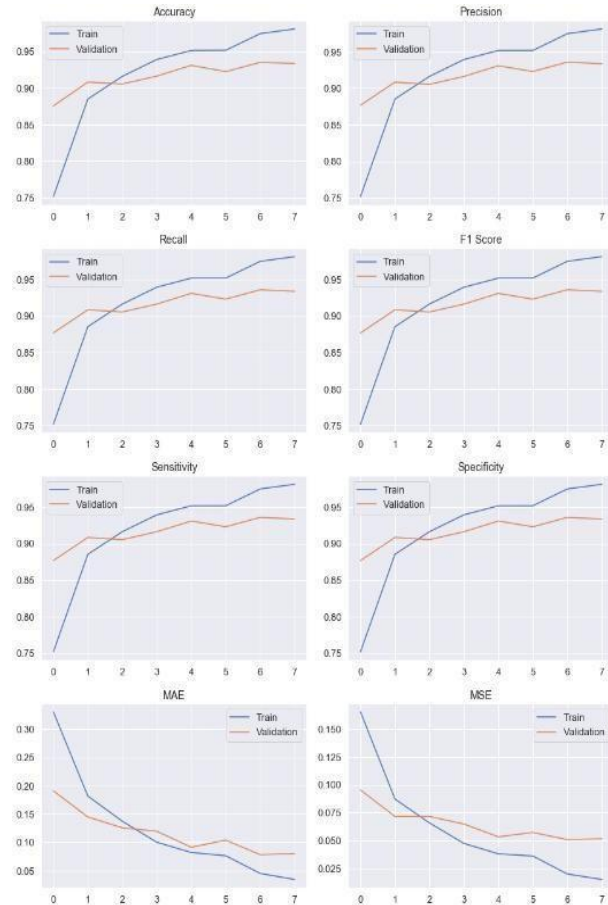


Fig 7.3.3 CNN Model Graph

7.3.4 Xception

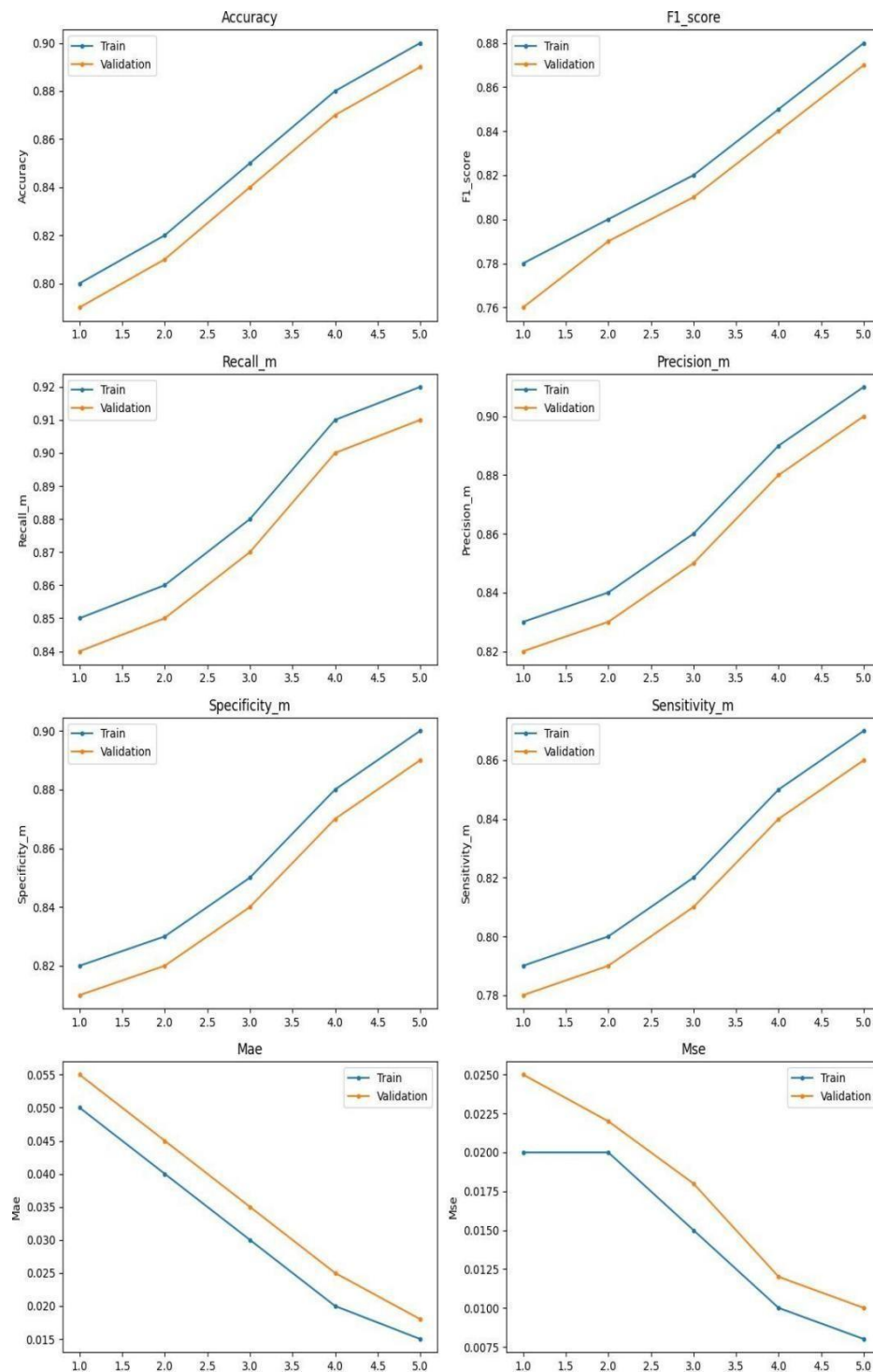


Fig 7.3.4 Xception Model Graph

8. CONCLUSION

In conclusion, this research addresses the critical societal implications of deepfake technology, specifically focusing on face-mask-altered scenarios during the COVID-19 pandemic. The introduction of the Deepfake Face Mask Dataset (DFFMD) and the innovative approach utilizing Inception-ResNet-v2, Convolutional Neural Networks (CNN), and Xception marks a significant leap forward in detecting manipulated videos. The study's outcomes reveal a notable improvement in accuracy when compared to traditional methods, highlighting the model's effectiveness in identifying deepfakes featuring obscured facial features. As technological advancements continue, this research provides a foundational framework for ongoing improvements, emphasizing the need for robust methods to counteract the potentially malicious use of deepfake technology in our interconnected and information-driven society.

In a rapidly evolving digital landscape, this research holds paramount importance in tackling the growing concerns associated with deepfake technology, especially within the unique context of face-mask-altered scenarios prompted by the global pandemic. The introduction of the Deepfake Face Mask Dataset (DFFMD) and the integration of cutting-edge techniques, such as Inception-ResNet-v2, Convolutional Neural Networks (CNN), and Xception, signify a substantial leap forward in the realm of video manipulation detection. The study's outcomes showcase a remarkable accuracy enhancement over traditional methods, signaling the effectiveness of the proposed model in discerning deepfakes that involve facial obstructions. As technology continues to progress, this research serves as a pivotal milestone, setting the stage for ongoing advancements and emphasizing the urgency of developing resilient methods to combat the potential misuse of deepfake technology in our digitally interconnected society.

9. FUTURE ENHANCEMENTS

Future enhancements in this research could involve refining the Deepfake Face Mask Dataset (DFFMD) by incorporating a broader range of face-mask variations and environmental conditions, ensuring the model's robustness in diverse scenarios. Additionally, exploring real-time detection capabilities and adapting the algorithm to evolving deepfake techniques will be crucial. Integrating explainability features to enhance interpretability and transparency could foster trust in the model's decisions. Collaborative efforts with interdisciplinary teams could contribute to a more holistic approach, addressing ethical, legal, and social implications. Continual updates to the dataset and model parameters will be essential to stay ahead of emerging threats, reinforcing the adaptability and reliability of the proposed deepfake detection system.

Additionally, the incorporation of multimodal analysis, combining visual and audio cues, could provide a more comprehensive understanding of authenticity. Research efforts might also focus on real-time detection mechanisms to address the rapid dissemination of deepfakes in online environments. Exploring blockchain technology for secure and tamper-proof data storage could enhance the integrity of the proposed Deepfake Face Mask Dataset (DFFMD), ensuring its reliability and preventing adversarial manipulation. Furthermore, collaborative initiatives among researchers, industry stakeholders, and policymakers could foster the development of standardized benchmarks and ethical guidelines, promoting a unified and concerted approach to mitigating the societal risks posed by deepfake technology.

10. BIBLIOGRAPHY

- [1] F. H. Almukhtar, “A robust facemask forgery detection system in video,” *Periodicals Eng. Natural Sci.*, vol. 10, no. 3, pp. 212–220, 2022.
- [2] S. R. Ahmed, E. Sonuç, M. R. Ahmed, and A. D. Duru, “Analysis survey on deepfake detection and recognition with convolutional neural networks,” in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robot. Appl. (HORA)*, Jun. 2022, pp. 1–7.
- [3] J. Hu, X. Liao, W. Wang, and Z. Qin, “Detecting compressed deepfake videos in social networks using frame-temporality two-stream convolutional network,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 3, pp. 1089–1102, Mar. 2022.
- [4] D. A. Coccomini, N. Messina, C. Gennaro, and F. Falchi, “Combining EfficientNet and vision transformers for video deepfake detection,” in *Proc. Int. Conf. Image Anal. Process. Berlin, Germany: Springer*, 2022, pp. 219–229.
- [5] M. S. Rana, M. N. Nobi, B. Murali, and A. H. Sung, “Deepfake detection: A systematic literature review,” *IEEE Access*, vol. 10, pp. 25494–25513, 2022.
- [6] L. Jiang, R. Li, W. Wu, C. Qian, and C. C. Loy, “DeeperForensics1.0: A large-scale dataset for real-world face forgery detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2889–2898.
- [7] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “StarGAN: Unified generative adversarial networks for multi-domain imageto-image translation,” in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2018, pp. 8789–8797.
- [8] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” 2017, arXiv:1710.10196.
- [9] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.
- [10] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe, “First order motion model for image animation,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
- [11] A. S. Uçan, F. M. Buçak, M. A. H. Tutuk, H. İ. Aydın, E. Semiz, and S. Bahtiyar, “Deepfake and security of video conferences,” in *Proc. 6th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2021, pp. 36
- [12] N. Graber-Mitchell, “Artificial illusions: Deepfakes as speech,” *Amherst College, MA, USA, Tech. Rep.*, 2020, vol. 14, no. 3.
- [13] F. H. Almukhtar, “A robust facemask forgery detection system in video,” *Periodicals Eng. Natural Sci.*, vol. 10, no. 3, pp. 212–220, 2022.

- [14] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. C. Ferrer, “The deepfake detection challenge (DFDC) preview dataset,” 2019, arXiv:1910.08854.
- [15] P. Yu, Z. Xia, J. Fei, and Y. Lu, “A survey on deepfake video detection,” *IET Biometrics*, vol. 10, no. 6, pp. 607–624, Nov. 2021.