DEEPFAKE DETECTION AND ATTRIBUTION

A Capstone Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

by

J. SRI VARSHA	**(2303A51L99)**
K. SINDHU	**(2303A51LA0)**
B. JAYANTH	**(2303A51LA7)**
G. MAHENDRA	**(2303A51LA9)**
K. NIKHIL	**(2303A51LB0)**

Under the guidance of

Dr. Ramesh Babu A

Assistant Professor, School of CS&AI.

SR UNIVERSITY

SR University, Ananthsagar,Warangal,Telangana-506371

SR University

Ananthasagar, Warangal.

CERTIFICATE

This is to certify that this project entitled **"DEEPFAKE DETECTION AND ATTRIBUTION"** is the Bonafide work carried out by **J. Sri Varsha, K. Sindhu, B. Jayanth, G. Mahendra, K. Nikhil** as a Capstone Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2025-2026 under our guidance and Supervision.

Dr. Ramesh Babu A
Assistant Professor,
SR University
Ananthasagar, Warangal.

Dr. M. Sheshikala
Professor & Head,
School of CS&AI,
SR University
Ananthasagar, Warangal.

Reviewer-1
Name:
Designation:
Signature:

Reviewer-2
Name:
Designation:
Signature:

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Capstone project guide **Dr. Ramesh Babu A, Assistant Professor** as well as Head of the School of CS&AI, **Dr. M. Sheshikala, Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project coordinator **Mr. Sallauddin Md, Assistant Professor** for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

Acronym	Full Form
AI	Artificial Intelligence
DFDA	Deepfake Detection and Attribution
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MFCC	Mel-Frequency Cepstral Coefficients
GPU	Graphics Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
API	Application Programming Interface
UI	User Interface
TTS	Text-to-Speech
MTCNN	Multi-task Cascaded Convolutional Neural Network
GRU	Gated Recurrent Unit
ML	Machine Learning
OS	Operating System
FFT	Fast Fourier Transform
RGB	Red-Green-Blue Colour Model
DFDC	Deepfake Detection Challenge
DFL	DeepFaceLab
GAN	Generative Adversarial Network
MELSPEC	Mel-Spectrogram
CVPR	Conference on Computer Vision and Pattern Recognition
ICCV	International Conference on Computer Vision
ICML	International Conference on Machine Learning
CSV	Comma-Separated Values
TIMM	PyTorch Image Models (Library)

ABSTRACT

The Deepfake Detection and Attribution (DFDA) is an inclusive and multi-platform project. It helps in identifying the deep fake content in multiple media i.e., video, audio, and image and also identifies unique fingerprints of the AI tool which is used to make that particular deep fake. This project is united with sophisticated deep learning models which are built by using PyTorch. The enhanced XceptionNet architecture is the image detector. The models are trained by DALL-E, Midjourney, StyleGAN, and Face Swap to identify the faces. A focusing mechanism is attached to the model, this focuses in identifying the inconsistencies. The video is divided into separate frames and every frame is moved through XceptionNet framework and gets trained. Here, the temporal clues like abnormal blinking of eyes, inconsistent facial expressions, unstable head gestures. These are mainly used in detecting Deepfake in video. The audio detection uses CNN (Convolutional Neural Network). Firstly, the audio is converted to spectrograms (spectrograms are visual representations of audio), then the spectrogram is trained with classes and saved as a model. It detects Voice Cloning, Voice conversion audios. This application is deployed using streamlit. Deepfake Detection and Attribution is a platform used to detect the fake content and also identifies the source from where the content is generated.

I. INTRODUCTION:

In this digital world, the usage of AI tools is increased drastically. This helps in creating innovative solutions when it comes its merits. But with these AI tools, we have disadvantages too. Some frauds are using this AI tools to morph the image, video, and audio. In today's society, it is the most serious issue we are facing right now. These particular AI tools are generating the fake content and people are misusing this technology. This fake media is using in cybercrimes, politically, and in manipulating social media. Also leading to the spread of misinformation rapidly.

To restrict all the vulnerable acts to some extent we came up with an idea as "Deepfake Detection and Attribution". This Deepfake Detection and Attribution System is mainly used to detect the fake media including audio, video, and image. It not only detects the Deepfake content but also traces the source of the Deepfake media. Our project main goal is to identify the Deepfake media with their attribution i.e., identifying the source which is used in generating the fake content. We aren't limited to a single media Deepfake detection, we are also using multi-modal concept which includes audio, video and image.

Our project is bridge between our academics and practical knowledge. By this project, we are providing you a full-fledged platform for identifying Deepfake media. This allows transparency and also enables us to know the truth behind the misinformation.

II. RELATED WORK

As we discussed earlier, the fast development of AI which resulted in Deepfake media generation is mainly used for spreading false information for manipulating the people using the Deepfake content. Earlier, we had some researches which focused on image Deepfake detection and video Deepfake detection, in which only the architecture is used to detect the pixel level frames. But the introduction of the dataset FaceForensics++ had contributed majorly because it is the collection of facial ideas which are generated by different AI tools like Face2Face, FaceSwap, Dall-E, etc., With the help of this dataset, it became easy to train the models using deep learning algorithms.

The XceptionNet architecture is the most widely used Deepfake detection framework. In the existing researches, there are some limitations that affects the accuracy. XceptionNet architecture, Transformers had developed and their performance is improved for detecting the fake media. By this, we can directly detect the Deepfake images which are generated by the AI tools like DALL-E, StyleGAN and Midjourney which are mainly used in fake image generation.

And with the image research, the audio research was also made. But the audio has given many challenges like inconsistencies of patterns. In past, we had used MFCCs for differentiating the natural speech with artificial speech. In modern days, Spectrogram is used for the detection of audios in which CNN and RNN are trained by the model. So, we made a 4-layer CNN project where we trained on spectrograms which detects the cloning of voice and other voice Deepfakes more accurately.

When it comes to videos, there are many such challenges because videos are a combination of frames. It is difficult to work on each single frame in video when it compared to image.3D CNN, LSTM approaches are discovered for temporal analysis. By carrying all such necessary information, we made our implementation by using XceptionNet model which analysis by frame by frame.

All this research was made on a single platform like they mainly focussed on any one media. So, we developed a multi-modal platform which can detect the Deepfake content in any media like audio, video and image. It not only detects the Deepfake but also identifies the source of the Deepfake from where it is generated.

III. PROBLEM STATEMENT

The development of AI and digital world had made us to generate the realistic media including image, video and audio. We can able to generate the fake content by using some AI tools. So, this technology is being misused by some frauds. They are using this morphed media for cyber-crimes. These Deepfake content is leading to rise the privacy issues along with security issues. This made our digital world as a untrustworthy world. This also used for damaging the reputation of individual personality.

The Deepfakes are made by using the facial expressions, tone of our voice, movement of eyes, skin texture etc., We humans are not able to find the Deepfake content using the naked eye because they are not visible to us directly. The existing solutions have many limitations. Many systems focus on just one media and they are not able to trace the source of the Deepfake content. And many gives the result in binary format only. With this the transparency of the outputs is not so realistic.

So, we came up with a problem which is united, and a reliable system which is able to detect fake content in different media including audio, video and image. Also, we traced the source of the AI tool which is used for generating the fake content. We made the attribution too. Our project is best because:

- The accuracy of detection of Deepfake is 90.
- It is very realistic and performance is very better when compared with existing solutions.
- We can clearly see the results in the dashboard and also the attribution is displayed clearly.

We had built a project, which is very helpful in making the digital platform trustworthy. We made the limitations of the earlier systems to work in our project with high accuracy. Our Deepfake Detection and Attribution (DFDA) is mainly a solution make the world to not believe all the media and to check whether it is fake or not also we made them to know what AI tool is used in making that Deepfake content.

IV. REQUIREMENT ANALYSIS

Our project, Deepfake Detection and Attribution is mainly developed for the detection of fake media like image, video and audio and also to trace the source of fake media which is called attribution. Our project give high accuracy, provides transparency and gives better performance in generating the results.

4.1. PERFORMANCE REQUIREMENTS

1. **SUPPORT OF TRIPLE-MEDIA**
    - Our project is developed to support different media uploading i.e., audio, video, and image.
    - .wav, .mp3, .m4a, .ogg are the formats used to upload for the detection of audio.
    - .mp4, .webm, .mkv are used to upload for the video.
    - .jpg, jpeg, .png, .webp are used to upload for the image.
2. **REPROCESSING PIPELINES**
    - The images should be resized and normalized in preprocessing.
    - The videos should be converted to individual frames.
    - The audio should be converted to spectrograms.
3. **MODEL IDENTIFICATION**
    - Our Deepfake Detection and Attribution is used in detecting the fake content in different media.
    - As we discussed earlier, we also perform the attribution which is nothing but finding the source of the faked content.
    - The models which is used to generate the fake content are DeepFaceLab, FaceSwap, StyleGAN and Stable Diffusion.
4. **REPLICA REVIEW**
    - The model supports the audio uploading, so the audio is reviewed by using the CNN on the spectrograms (Mel-Spectograms).
    - The video is identifying after extracting the individual frames and train by XceptionNet Temporal Analysis.
    - The image is identified by using the XceptionNet modal.

5. **OUTPUTS**
    - We are using a dashboard for displaying the results by using confidence score, probability, and we use confusion matrices, spectrograms for a clear result.

4.2. NON-PERFORMANCE REQUIREMENTS

1. **EXECUTION**
    - Our project is a real-time based system and also provides accuracy above 90%.
2. **SERVICEABLENESS**
    - Our Dashboard is user-friendly and it is very easy to use.
    - We can easily upload the files in the website and can get the results quickly.
3. **SOLIDITY**
    - Our project provides the consistent results and also performs good.
    - It can handle all the three types of files.
4. **SAFETY**
    - In our system, the uploaded files are not stored permanently.
    - The files are deleted automatically after displaying the results.
5. **EXTENSABILITY AND OPERABILITY**
    - This architecture can handle multiple requests at once.
    - Individual changes won't affect the whole system.

RISK ANALYSIS

The Deepfake Detection and Attribution have many risks while developing and deploying the project. The following are some of the risks:

- If the user interface is complex, the user is not able to understand the system, by keeping the dashboard clean we can avoid this risk.
- One of the main issues are limited datasets. We can't find many datasets for Deepfake detection and attribution. With this, the training will be affected.
- While training the models, we can observe overfitting and underfitting. This may affect the validation checks.

- For this project, we require a strong GPU Laptop, so that we can train our models and can get accurate results.
- And the issue rises with privacy, because the user is sharing their sensitive information. So, we can make the automatic deletion.
- We can also see the dependency and system function issues along with the classification problems.

FEASIBILITY ANALYSIS

Analysis of feasibility of the system is successful when it is deployed without any problems.

1. TECHNICAL VIABILITY

- In this project, we mainly used the technologies like Python, Streamlit, OpenCV, Plotly, PyTorch.
- This project requires a system which has good GPU for better analysing of results.
- We have to train the models which is called as Pre-training.

2. PRACTICAL USABILITY

- It is simple and easy to use platform.
- We can use it with less knowledge. Anyone can use this for any operation without any cons.

3. COST EFFECTIVENESS

- We can use the free sources for this project.
- We require only GPU included system, which is very cost effective.
- We can maintain the project simply and neatly.

4. LEGAL VALIDITY

- The files which we are uploading in the system is not stored, which make the user data secured.
- The data will be deleted automatically after displaying the results.

5. CULTURAL VIABILITY

- The main goal of this project development is identifying the faked content and making it clear that from where it is generated.
- By this we can build a trustworthy environment which is safe for us.

V. PROPOSED SYSTEM

Deepfake Detection and Attribution is a multi-model system which is used to identify the Deepfake or morphed content of the media including audio, video, and image. Also, we can able to find the attribution which means we can trace the media of faked content and can find which particular AI tools is used in making that content. This can be possible by identifying the unique fingerprints in the media. We can achieve this by using different dee learning techniques.

1. MULTI-MODAL SYSTEM

In the earlier systems, they are limited to one media only like they are used to detect any particular media faked content. And no other system is made the attribution possible. This model will catch the unique fingerprints and detects the model.

1.1.IDENTIFICATION OF DEEPFAKE IN IMAGE

- The detection of image is done by XceptionNet with an Attention Mechanism united with it.
- The model will identify the texture of the face like skin, abnormality in eyes, and the inconsistencies seen in face.
- It can identify the models like DALL-E, StyleGAN, FaceSwap.

1.2.IDENTIFICATION OF DEEPFAKE IN VIDEO

- From the video, individual frames are extracted and they are used for further processes.
- The individual frames are trained by using CNN and XceptionNet.
- The video Deepfake is detected by using the facial inconsistencies, movement abnormalities of head and body, and skin texture.

1.3.IDENTIFICATION OF DEEPFAKE IN AUDIO

- The audio is converted to Spectrograms, according to the variations in time and frequency.
- We use CNN for the audio detection to the Spectrograms. We mainly use this for cloning of voice, Text to speech conversions etc.,

2. ATTRIBUTION METHOD

Attribution is nothing but tracing the AI tool which is used in generating the faked content. This system is a multi-media platform like audio, video and image. This is used to identify the source of the faked content.

FaceSwap, DeepFacelab, StyleGAN and other AI models are traced by using the attribution.

3. ENVISIONING AND UI

This project is deployed by using Streamlit and it also has following attributes:

- It is very simple to upload the files.
- The files will be deleted after displaying the results.
- In results, the confidence scores, confusion matrices are displayed.

This project provides transparency and makes to build a trustworthy platform.

4. WORKFLOW

i) **Input**

The user will upload the files of the media.

ii) **Preprocessing**

Here, the normalisation, resizing, framing and other techniques are used.

iii) **Detection**

The Deepfake is detected here, whether the media is faked or not, it will display.

iv) **Attribution**

The source of the faked media is detected and traced here.

v) **Results**

The result will be displayed with the confusion matrix, confidence scores.

5. ARCHITECTURE OF SYSTEM

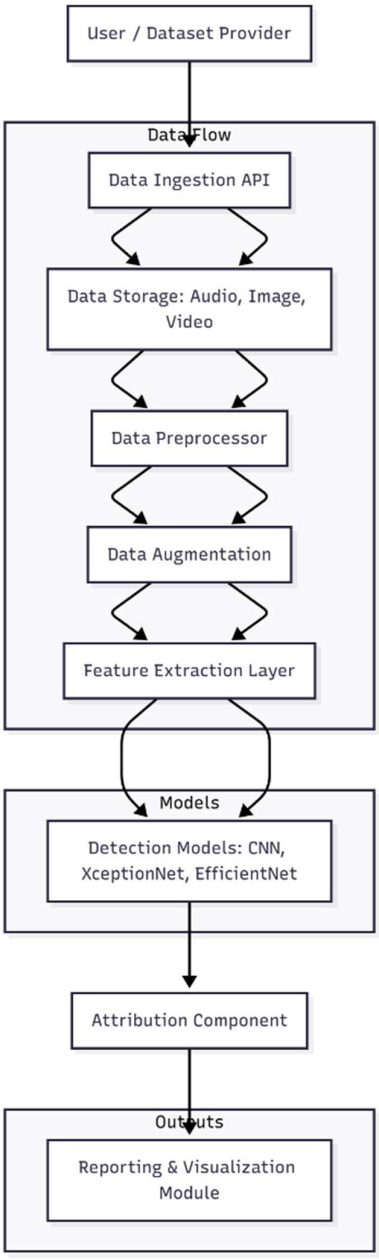
- Each and every module is independent to each other.
- The change in one particular module won't affect the other and results.
- The dashboard will be smooth and no problem will rise while using it.

6. SECURITY

- The uploaded files are not used further for any extra activities. They are deleted automatically after displaying the results.
- This enables a trustworthy platform to upload the media and to believe it.

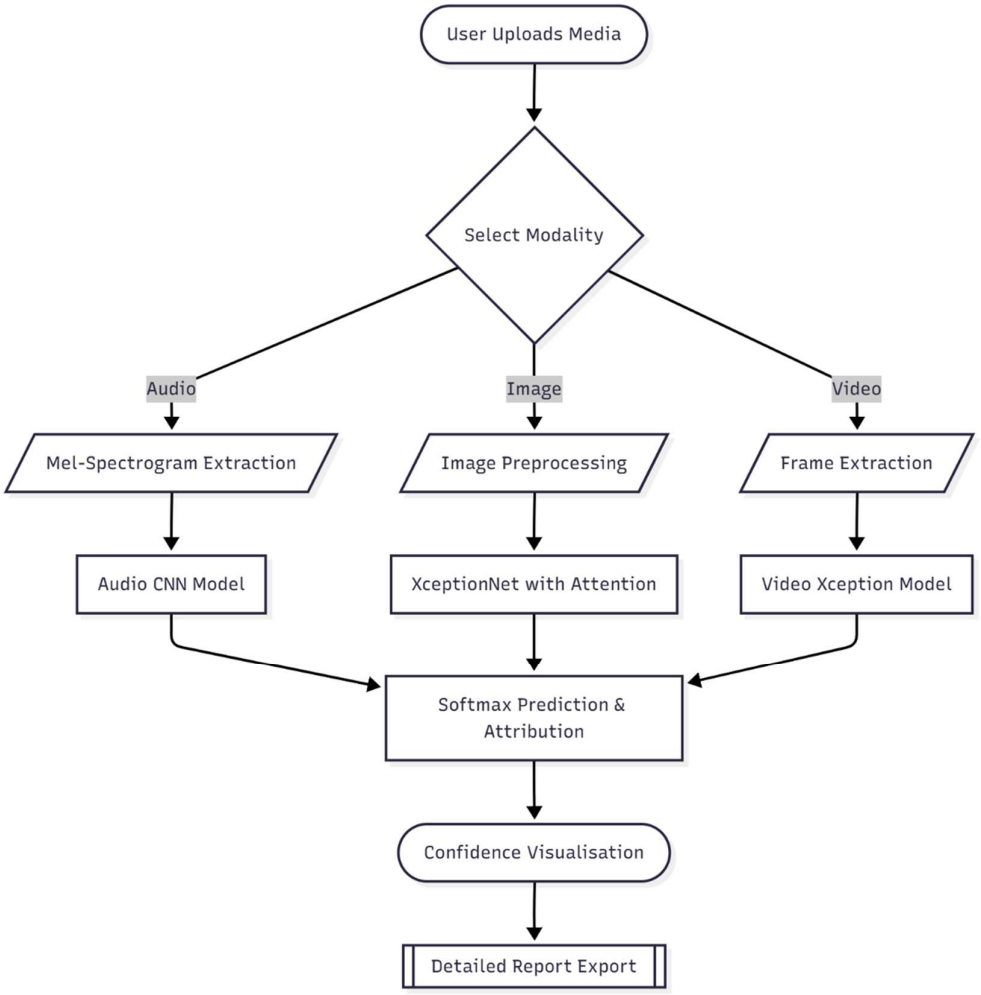
VI. ARCHITECTURE DIAGRAMS, FLOW CHARTS, DFD

ARCHITECTURE DIAGRAM:



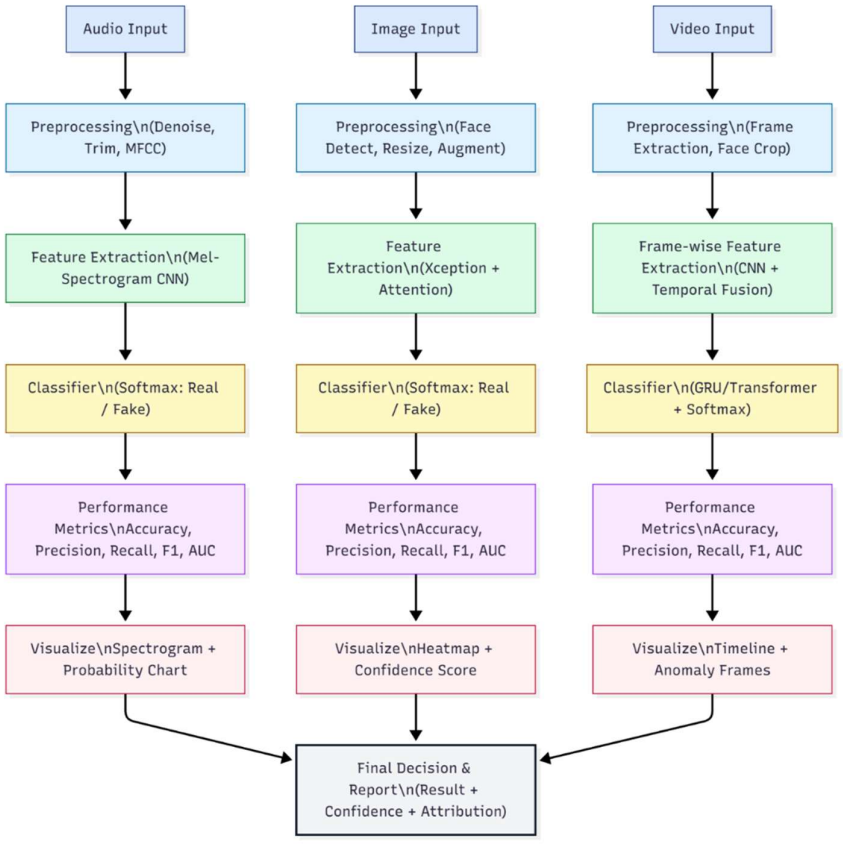
1. **User/Dataset Provider**
    - Provides input media such as audio, image, or video to be analyzed.
2. **Data Ingestion API**
    - Accepts uploaded files, verifies formats, and routes them to the correct processing path.
3. **Data Storage: Audio, Image, Video**
    - Organizes the media inputs by type for efficient access during pre-processing.
4. **Data Preprocessor**
    - Performs initial cleaning, normalizes and converts:
        a. Audio → Mel-Spectrograms
        b. Image → Resized/Normalized
        c. Video → Extracted Frames
5. **Data Augmentation**
    - Applies transformations such as rotation, masking, or noise addition to improve model generalization.
6. **Feature Extraction Layer**
    - Extracts key artifacts and features that indicate manipulations using deep feature mapping.
7. **Detection Models**
    - Classifies media authenticity using specialized models:
        a. CNN for audio
        b. XceptionNet for images
        c. EfficientNet for video
8. **Attribution Component**
    - Identifies the source or technique used for manipulation, e.g., FaceSwap, StyleGAN, TTS.
9. **Reporting & Visualization Module**
    - Displays the results of output, including:
        a. Real/Fake classification
        b. Confidence scores and probability charts
        c. Attribution details and spectrograms

FLOW CHART:



1. **Input Stage:**
    - The system takes input through three media: audio, image, and video, for verification of authenticity.
2. **Preprocessing:**
    - Audio: Denoising, trimming, and MFCC extraction.
    - Image: Face detection, resizing, and data augmentation.
    - Video: Frame extraction and face cropping for sequence analysis.
3. **Feature Extraction:**
    - Audio features extracted using a CNN-based Mel-Spectrogram encoder.
    - Image features derived using Enhanced XceptionNet with attention.
    - Video features captured through CNN with temporal fusion across frames.

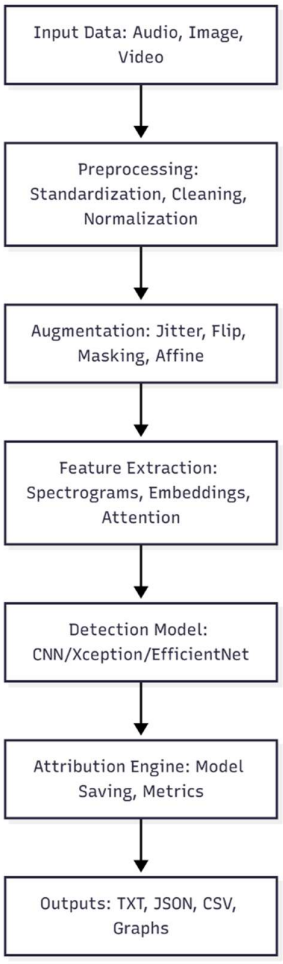
4. **Classification Layer:**
    - Each modality uses a SoftMax-based classifier to classify the media as real or fake.
    - The video model can also include GRU/Transformer layers for sequential learning.
5. **Performance Evaluation:**
    - Models are evaluated based on metrics including accuracy, precision, recall, F1-score, and AUC, which quantify robustness.
6. **Visualization:**
    - Audio: Spectrogram and probability charts.
    - Image: Confidence heatmaps highlighting manipulated regions.
    - Video: Temporal timelines showing anomaly frames.
7. **Final Decision and Reporting:**
    - It consolidates system results from all modalities into one final decision about authenticity, including confidence level and attribution details. Reports are shown on screen and can be exported for further analysis.



DATAFLOW DIAGRAM:

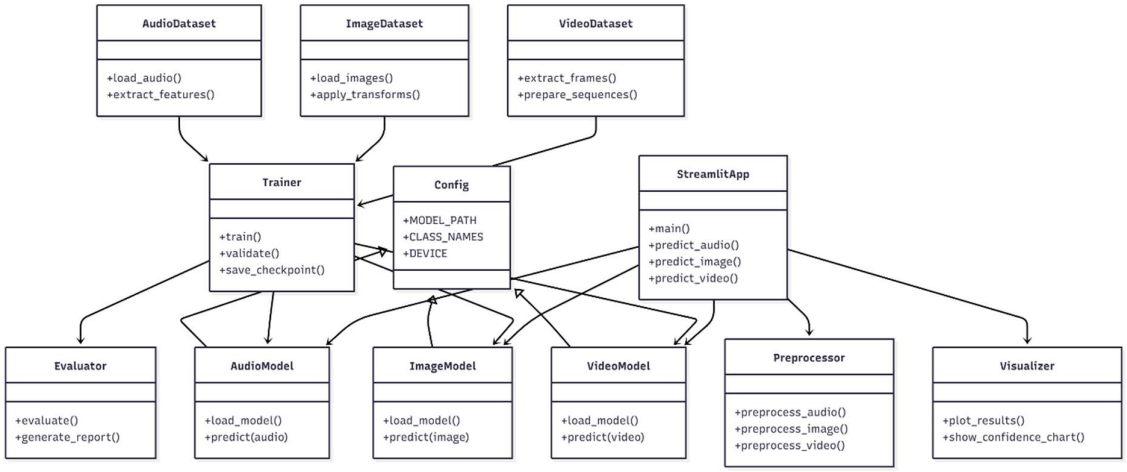
Process Flow Explanation:

1. **Input Data:** Audio, Image, Video
    - The system accepts multimedia inputs of various formats; thus, these inputs act as the basis for detection and analysis.
2. **Preprocessing:** Standardization, Cleaning, Normalization
    - Raw inputs are cleaned and standardized.
    - Resampling of audio, resizing of images, and extracting video frames are done for uniform processing.
3. **Augmentation:** Jitter, Flip, Masking, Affine
    - Data augmentation techniques are applied to increase dataset variability and reduce overfitting.
    - Operations include random flips, brightness jitter, affine transformations, and masking.



4. **Feature Extraction:** Spectrograms, Embeddings, Attention
    - Key features are extracted from each modality:
    - Audio: Mel-Spectrograms
    - Image/Video: Deep feature embeddings with attention mechanisms
5. **Detection Model:** CNN / Xception / EfficientNet
    - Classification is done by special deep learning models:
    - CNN for spectral analysis
    - XceptionNet for image and frame-level detection
    - EfficientNet for lightweight inference
6. **Attribution Engine:** Model Saving, Metrics
    - Specifies the origin or method through which the Deepfake is created, for example, FaceSwap, StyleGAN.
    - Records performance metrics for continuous improvement.
7. **Outputs:** TXT, JSON, CSV, Graphs
    - The final results are produced in various forms, namely in structured reports, table data, and as visual graphs.

CLASS DIAGRAM:



- The AudioDataset class manages the loading and feature extraction of audio data used for Deepfake detection.
- The ImageDataset class handles the loading and transformation of image data in preparation for model training and testing.

- The VideoDataset class is responsible for extracting frames and preparing video sequences for temporal analysis.
- The Trainer class controls the entire process of training: model optimization, validation, and saving checkpoints.
- The Config class makes sure that all global configuration details like model paths, class names, hardware device settings are kept consistent throughout the system.
- The AudioModel, ImageModel, and VideoModel classes represent the trained deep learning models corresponding to each modality to perform media-specific prediction and inference.
- The Preprocessor class standardizes, normalizes, and prepares the input data before passing it to the detection models for analysis.
- The Evaluator class assesses model performance, calculates accuracy metrics, and creates analytical reports to validate the results.
- The Visualizer class is responsible for graphically representing the detection output using confidence charts and plots, along with the visualization of probabilities for better interpretability.
- The StreamlitApp class combines the system's components into an integrated web-based interactive interface. It provides a user-friendly interface for uploading files, executing predictions, and displaying the results in real time.

ACTIVITY DIAGRAM:

1. **IMAGE ACTIVITY DIAGRAM:**

1. **Load Image / Frame**

   The system starts by loading an input image or a single frame extracted from a video.

2. **Face Detection (MTCNN / RetinaFace)**

   Advanced face detection algorithms identify and localize facial regions within the frame.

3. **Face Found?**

   - If a face is detected, the pipeline continues to cropping and alignment.

- In case no face is detected, the image will either be rejected or undergo processing along a full-image detection path.

4. **Crop, Align & Resize**

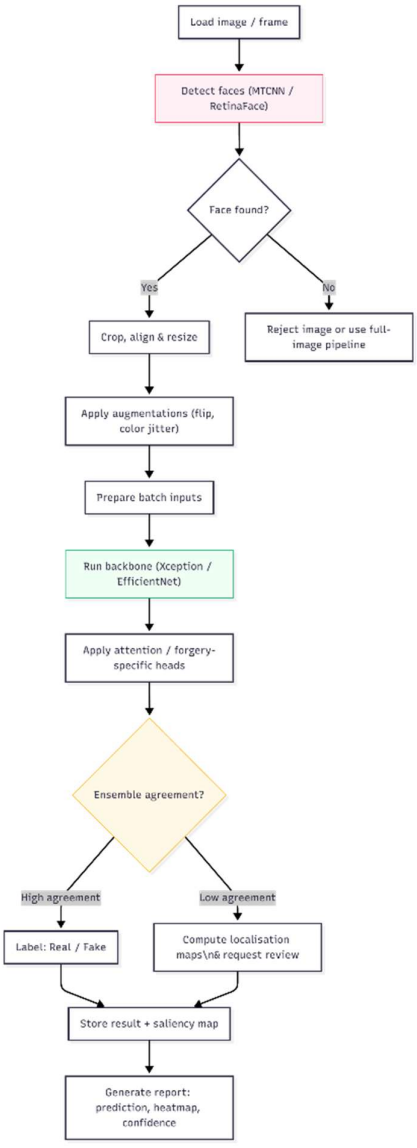
   The face region detected is cropped, geometrically aligned, and resized into a standard input dimension for model consistency.

5. **Apply Augmentations**

   It employs data augmentation techniques like random flipping and color jittering to improve the robustness of the model.

6. **Prepare Batch Inputs**

   Such pre-processed and augmented images are converted into batches for efficient model inference.



7. **Run Backbone Network**

   Xception / EfficientNet The deep learning backbone extracts rich spatial and texture-based features from the input image or frame.

8. **Apply Attention / Forgery-Specific Heads**

   Attention mechanisms highlight manipulated regions and improve the detection of subtle artifacts.

9. **Ensemble Agreement Check**

   Several model predictions are compared; high agreement indicates confident classification, while low agreement triggers additional review.

10. **Final Output Generation**

    It provides a label for Real/Fake, stores the results along with the saliency map, and also generates a detailed report containing the confidence of predictions and their heatmap visualization.

2. **VIDEO ACTVITY DIAGRAM:**

1. **Load Video File**

   The system initiates by loading the input video for analysis.

2. **Extract Key Frames**

   Frames are extracted regularly, for example, every Nth frame, to decrease the computational load without losing temporal context.

3. **Face Detection and Cropping**

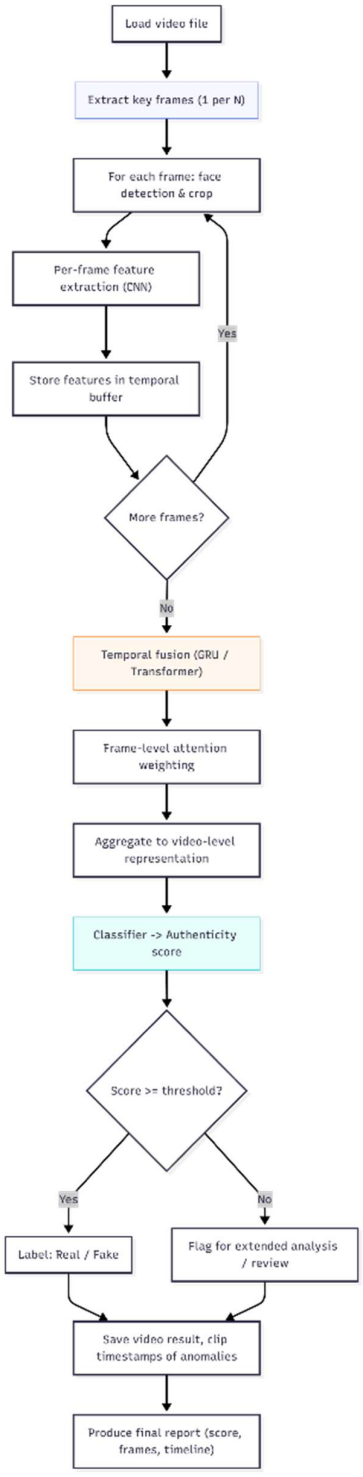
   For each frame, face detection is performed, and the detected faces are cropped and aligned to uniformly feed the model.

4. **Per-Frame Feature Extraction (CNN)**

   Each cropped face frame is passed through a convolutional neural network for extracting spatial and textural features.

5. Store Features in Temporal Buffer

The features extracted from consecutive frames are stored one after another for temporal processing.



6. **Temporal Fusion (GRU / Transformer)**

   When all key frames are processed, a GRU or Transformer module fuses temporal information in order to capture both motion and consistency across frames.

7. **Frame-Level Attention Weighting**

   Attention weights are given to frames by the system, where stronger manipulation evidence is highlighted.

8. **Aggregation to Video-Level Representation**

   Frame-level predictions are aggregated into a unified video-level feature representation.

9. **Classification Authenticity Scoring**

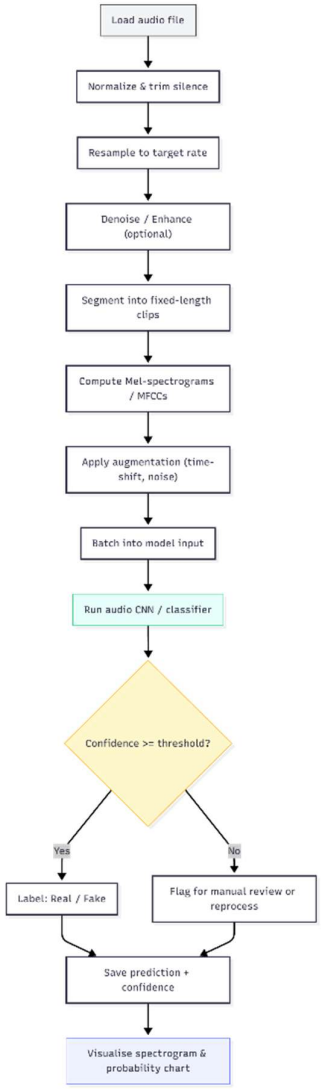
   A classifier predicts an authenticity score indicating whether a video is fake or real.

10. **Final Decision and Reporting**

    If the score is greater than the threshold, the video is labeled as Real/Fake.

    The system then saves the results, timestamps the anomalies, and creates a final report along with the confidence score and frame-level visualization.

3. AUDIO ACTIVITY DIAGRAM:



1. **Load Audio File**

   It starts by loading the input audio file from the user or dataset to be analyzed.

2. **Normalize and Trim Silence**

   This normalizes the amplitude of the audio signal and trims the silent regions.

3. **Resample to Target Rate**

   The audio is resampled to a fixed sampling rate, for example, 16 kHz, for consistent processing by the model.

4. **Denoise / Enhance (Optional)**

   Noise reduction signal enhancement techniques are applied to facilitate clarity.

5. **Segment into Fixed-Length Clips**

   Long audio files are first divided into smaller segments to enable efficient feature extraction and classification.

6. **Compute Mel-Spectrograms / MFCCs**

   Each audio segment is then put into a mel-spectrogram or MFCC representation to capture frequency and time-based patterns.

7. **Apply Augmentation: Time-Shift, Noise**

   Data augmentation methods like time-shifting and noise addition are applied to improve model robustness.

8. **Run Audio CNN / Classifier**

   The pre-processed spectrograms are then fed into a CNN-based model that classifies the audio as real or fake.

9. **Confidence Evaluation**

   If the confidence score of the model is above the set threshold, a final label is assigned; otherwise, the clip will be flagged for review.

VI. SIMULATION SETUP AND IMPLEMENTATION

The Deepfake Detection and Attribution is multi-modal platform, which is used to detect the faked content in different media like audio, video and image using their respective deep learning techniques.

1. EVOLUTION FRAMEWORK

1.1.SOFTWARE STRUCTURE

- Programming Language: 3.11.9
- Frameworks: PyTorch, Streamlit, Timm
- Libraries Used: NumPy, Pandas, OpenCv,TorchVision, Librosa, Plotly, Matplotlib
- Platform: VS Code, Jupyter Notebook
- Operating System: Windows 10/ Windows 11

1.2.HARDWARE STRUCTURE

- GPU: NVIDIA RTX 3050
- Processor: Intel i5/ Ryzen 5
- RAM: 16GB DDR4
- CUDA Tools: Version 12.9

2. DATASET COLLECTION

- The image dataset is generated by DALL-E, Midjourney, StyleGAN, DeepFaceLab and some are real-images.
- The videos is collected from the FaceForensics.
- The audio is collected from the ElevenLabs, Text to Speech conversions and some are human speeches.

3. IMPLEMENTATION

3.1.IMAGE DETECTION

- The images are collected from the different datasets.
- Then they are resized and normalized by using the tools.
- Here, we are using the Enhanced XceptionNet with the integration of the Attention Mechanism for detecting the morphed content.
- We achieved 99.54% accuracy.

3.2.VIDEO DETECTION

- The collected videos from the dataset are pre-processed for the training.
- The frames are extracted from the videos; these are individually extracted and again these are normalized and resized.
- These frames are passed through the XceptionNet based CNN to achieve our results.
- We achieved 90% accuracy for the video Deepfake detection.

3.3.AUDIO DETECTION

- This collected audios are converted to Spectrograms particularly the Mel-Spectrogram.
- Here, we use CNN for the detection of Deepfake.
- WE achieved 98% accuracy for the audio.

4. USER INTERFACE

Here, we are using Streamlit for the deployment of the project.

- We can easily upload the media (audio, video, image).
- It automatically detects the media type.
- Then the visualization is made for the process.
- The methods for respective media is done.
- Probabilities of the results is generated.
- The results will be displayed on the dashboard.

DASHBOARD:

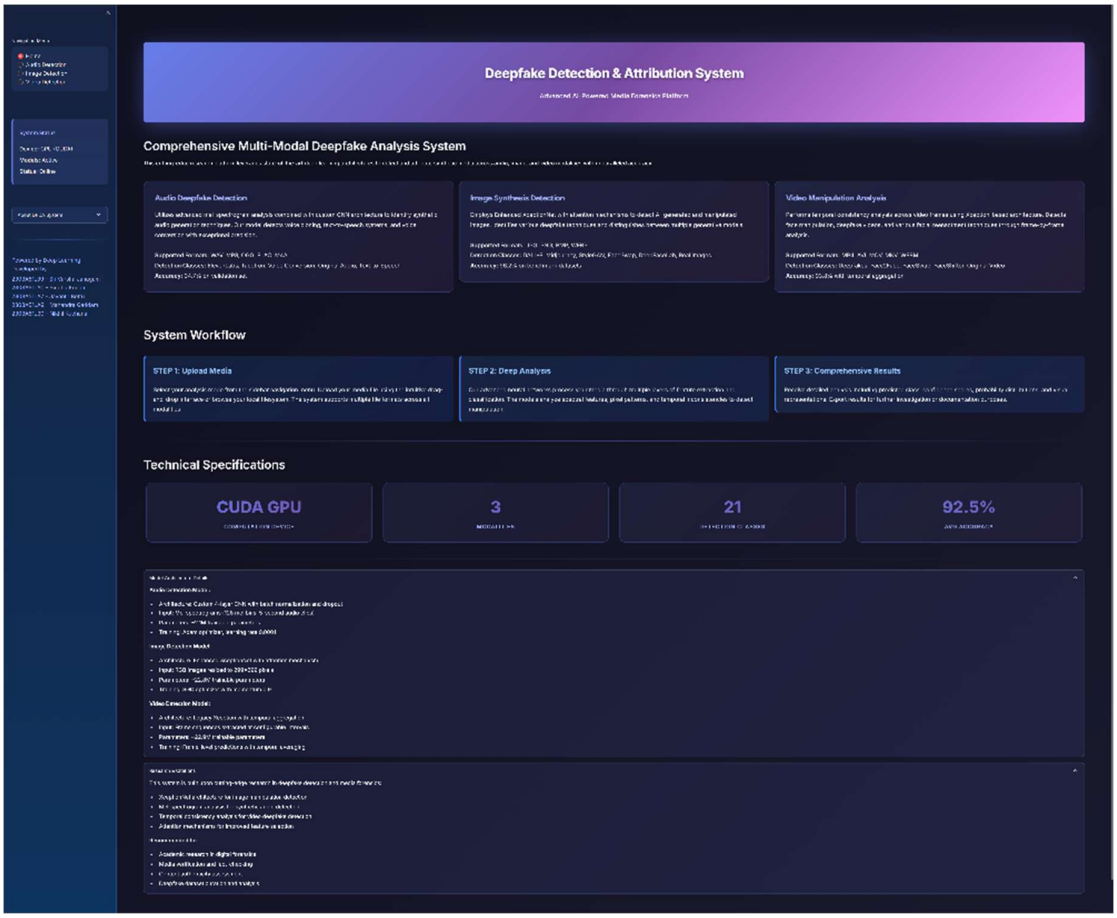
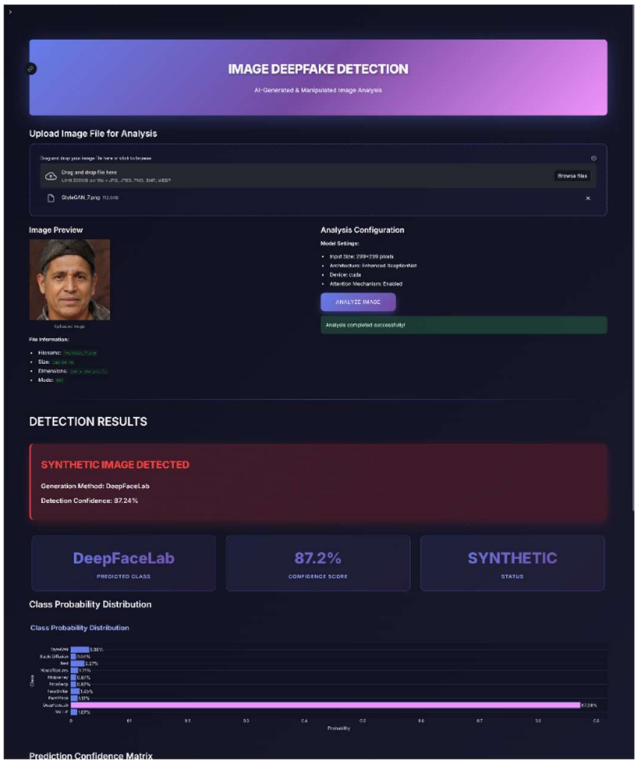
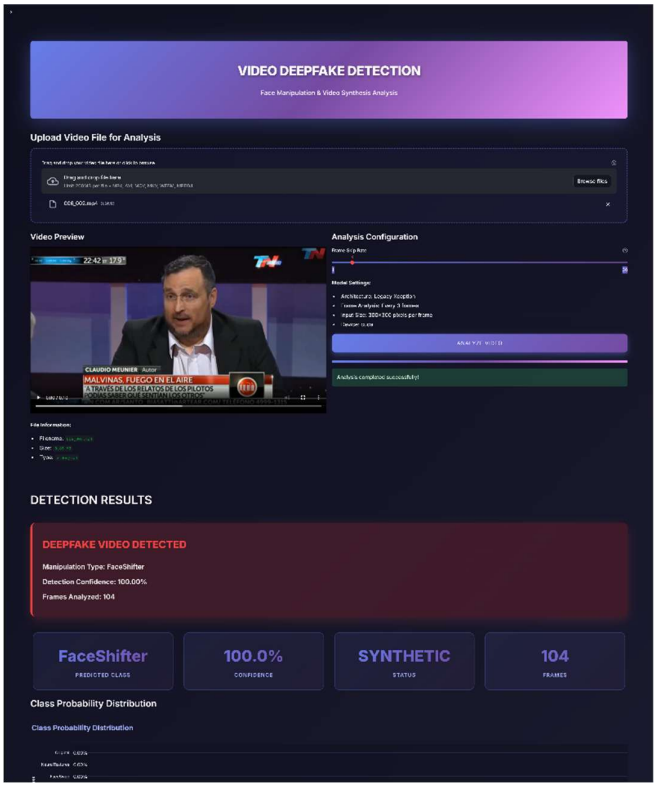


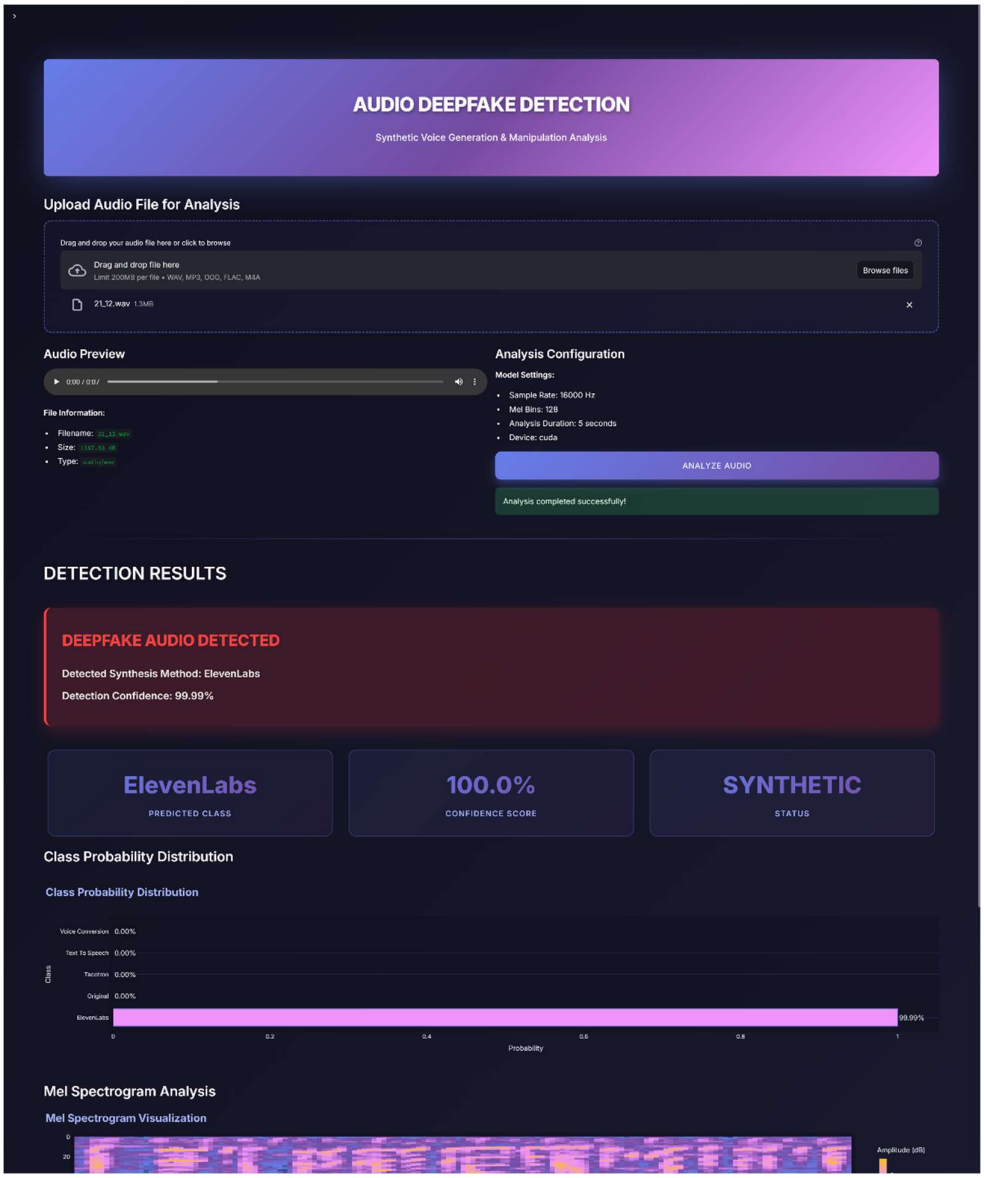
IMAGE DEEPFAKE DETECTION AND IMPLEMENTATION:



VIDEO DEEPFAKE DETECTION AND IMPLEMENTATION:



AUDIO DEEPFAKE DETECTION AND IMPLEMENTATION:



Code Implementation:

AudioModelTraining.py

```
import torch

import torch.nn as nn

import torch.optim as optim

from torch.utils.data import DataLoader


def train_model(model, train_loader, val_loader, num_epochs=30, learning_rate=0.0001, device='cuda'):

    criterion = nn.CrossEntropyLoss()

    optimizer = optim.Adam(model.parameters(), lr=learning_rate, weight_decay=1e-5)

    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=5)

    best_val_acc = 0.0

    patience_counter = 0

    max_patience = 4

    for epoch in range(num_epochs):

        model.train()

        train_loss = 0

        train_correct = 0

        train_total = 0

        for inputs, labels in train_loader:

            inputs = inputs.to(device)

            labels = labels.to(device)

            optimizer.zero_grad()
```

```
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        _, predicted = outputs.max(1)
        train_total += labels.size(0)
        train_correct += predicted.eq(labels).sum().item()
    train_loss = train_loss / len(train_loader)
    train_acc = 100.0 * train_correct / train_total
    model.eval()
    val_loss = 0
    val_correct = 0
    val_total = 0
    with torch.no_grad():
        for inputs, labels in val_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            val_loss += loss.item()
            _, predicted = outputs.max(1)
            val_total += labels.size(0)
            val_correct += predicted.eq(labels).sum().item()
```

```
    val_loss = val_loss / len(val_loader)
    val_acc = 100.0 * val_correct / val_total
    scheduler.step(val_loss)
    if val_acc > best_val_acc:
        best_val_acc = val_acc
        patience_counter = 0
        torch.save({
            'epoch': epoch,
            'model_state_dict': model.state_dict(),
            'optimizer_state_dict': optimizer.state_dict(),
            'val_acc': val_acc,
        }, 'best_model.pth')
    else:
        patience_counter += 1
    if patience_counter >= max_patience:
        break
return best_val_acc
```

ImageModelTraining.py

```
def train_image_model(model, train_loader, val_loader, num_epochs=20, learning_rate=0.0001, device='cuda'):
    criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
    optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate, weight_decay=0.01)
    scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
        optimizer, mode='max', patience=3, factor=0.5, min_lr=1e-7)
    scaler = torch.cuda.amp.GradScaler(enabled=(device=='cuda'))
    best_val_f1 = 0.0
    patience_counter = 0
    max_patience = 5
    for epoch in range(num_epochs):
        model.train()
        running_loss, correct, total = 0.0, 0, 0
        all_preds, all_labels = [], []
        for images, labels, _ in train_loader:
            images, labels = images.to(device), labels.to(device)
            with torch.cuda.amp.autocast(enabled=(device=='cuda')):
                outputs = model(images)
                loss = criterion(outputs, labels)
            optimizer.zero_grad()
            scaler.scale(loss).backward()
            scaler.unscale_(optimizer)
            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
```

```
        scaler.step(optimizer)
        scaler.update()
        running_loss += loss.item() * images.size(0)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())
    epoch_loss = running_loss / total
    epoch_acc = correct / total
    from sklearn.metrics import f1_score
    epoch_f1 = f1_score(all_labels, all_preds, average='weighted')
    model.eval()
    val_loss, val_correct, val_total = 0.0, 0, 0
    val_preds, val_labels = [], []
    with torch.no_grad():
        for images, labels, _ in val_loader:
            images, labels = images.to(device), labels.to(device)
            with torch.cuda.amp.autocast(enabled=(device=='cuda')):
                outputs = model(images)
                loss = criterion(outputs, labels)
            val_loss += loss.item() * images.size(0)
            _, predicted = torch.max(outputs.data, 1)
            val_total += labels.size(0)
```

```
            val_correct += (predicted == labels).sum().item()
            val_preds.extend(predicted.cpu().numpy())
            val_labels.extend(labels.cpu().numpy())
    val_loss = val_loss / val_total
    val_acc = val_correct / val_total
    val_f1 = f1_score(val_labels, val_preds, average='weighted')
    scheduler.step(val_f1)
    if val_f1 > best_val_f1:
        best_val_f1 = val_f1
        patience_counter = 0
        torch.save({
            'epoch': epoch,
            'model_state_dict': model.state_dict(),
            'optimizer_state_dict': optimizer.state_dict(),
            'val_f1': val_f1,
            'val_acc': val_acc,
        }, 'best_model_img.pth')
    else:
        patience_counter += 1
    if patience_counter >= max_patience:
        break
return best_val_f1
```

VideoModelTraining.py

```
import torch

import torch.nn as nn

import torch.optim as optim

from torch.utils.data import DataLoader


def train_and_save_audio_model(model, train_dataset, val_dataset, batch_size, epochs, lr, device):

    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

    val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

    criterion = nn.CrossEntropyLoss()

    optimizer = optim.Adam(model.parameters(), lr=lr, weight_decay=1e-5)

    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, factor=0.5, patience=2)

    best_val_acc = 0.0

    patience = 5

    patience_counter = 0

    train_history = []

    val_history = []

    for epoch in range(epochs):

        model.train()

        running_loss = 0.0

        running_correct = 0

        running_total = 0

        for inputs, labels in train_loader:
```

```
        inputs = inputs.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        _, predicted = torch.max(outputs, 1)
        running_correct += predicted.eq(labels).sum().item()
        running_total += labels.size(0)
    train_loss = running_loss / running_total
    train_acc = running_correct / running_total
    train_history.append((train_loss, train_acc))
    model.eval()
    val_loss = 0.0
    val_correct = 0
    val_total = 0
    with torch.no_grad():
        for inputs, labels in val_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
```

```
            val_loss += loss.item() * inputs.size(0)
            _, predicted = torch.max(outputs, 1)
            val_correct += predicted.eq(labels).sum().item()
            val_total += labels.size(0)
    epoch_val_loss = val_loss / val_total
    epoch_val_acc = val_correct / val_total
    val_history.append((epoch_val_loss, epoch_val_acc))
    scheduler.step(epoch_val_loss)
    if epoch_val_acc > best_val_acc:
        best_val_acc = epoch_val_acc
        patience_counter = 0
        torch.save(model.state_dict(), "AudioTrainedModel.pt")
    else:
        patience_counter += 1
    if patience_counter >= patience:
        break
return {
    "train": train_history,
    "val": val_history,
    "best_val_acc": best_val_acc
}
```

app.py (Main UI)

```
import streamlit as st

import torch

import torchaudio

import torchvision.transforms as transforms

from PIL import Image

import numpy as np


st.set_page_config(page_title="Deepfake Model Testing", layout="wide")


st.title("Deepfake Model Evaluation UI")


MODEL_OPTIONS = ["Audio", "Image", "Video"]


model_choice = st.sidebar.selectbox("Choose Model Type", MODEL_OPTIONS)


def load_model(model_path, model_class, device):
    model = model_class()
    state = torch.load(model_path, map_location=device)
    model.load_state_dict(state)
    model.to(device)
    model.eval()
    return model
```

```
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")


if model_choice == "Audio":
    st.header("Test Audio Deepfake Detection Model")
    uploaded = st.file_uploader("Upload an audio (.wav) file...", type=["wav"])
    if uploaded:
        waveform, sr = torchaudio.load(uploaded)
        # assuming preprocessing: mono, resample, trim/pad, mel-spectrogram
        # apply same as in training
        waveform = torch.mean(waveform, dim=0, keepdim=True)
        # Example: resample
        resample = torchaudio.transforms.Resample(sr, 16000)
        waveform = resample(waveform)
        # Example: mel-spec
        mel = torchaudio.transforms.MelSpectrogram(
            sample_rate=16000,
            n_fft=2048,
            hop_length=512,
            n_mels=128
        )(waveform)
        mel_db = torchaudio.transforms.AmplitudeToDB(top_db=80)(mel)
        mel_db = (mel_db - mel_db.mean()) / (mel_db.std() + 1e-8)
        model = load_model("AudioTrainedModel.pt", YourAudioModelClass, DEVICE)
        with torch.no_grad():
```

```
            mel_db = mel_db.unsqueeze(0).to(DEVICE)  # add batch dim
            output = model(mel_db)
            pred = output.argmax(dim=1).item()
        st.success(f"Prediction: {pred}")

elif model_choice == "Image":
    st.header("Test Image Deepfake Detection Model")
    img = st.file_uploader("Upload an image...", type=["png", "jpg", "jpeg"])
    if img:
        image = Image.open(img).convert("RGB")
        transform = transforms.Compose([
            transforms.Resize((299, 299)),
            transforms.ToTensor(),
            transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
        ])
        x = transform(image).unsqueeze(0)
        model = load_model("ImageTrainedModel.pt", YourImageModelClass, DEVICE)
        with torch.no_grad():
            x = x.to(DEVICE)
            output = model(x)
            pred = output.argmax(dim=1).item()
        st.success(f"Prediction: {pred}")

elif model_choice == "Video":
```

```
    st.header("Test Video Deepfake Detection Model")
    vid = st.file_uploader("Upload a video frame/image...", type=["png","jpg","jpeg"])
    if vid:
        frame = Image.open(vid).convert("RGB")
        transform = transforms.Compose([
            transforms.Resize((300, 300)),
            transforms.ToTensor(),
            transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
        ])
        x = transform(frame).unsqueeze(0)
        model = load_model("VideoTrainedModel.pt", YourVideoModelClass, DEVICE)
        with torch.no_grad():
            x = x.to(DEVICE)
            output = model(x)
            pred = output.argmax(dim=1).item()
        st.success(f"Prediction: {pred}")


st.sidebar.markdown("---")
st.sidebar.write("Upload input, models must be in working directory. Replace `YourAudioModelClass`, `YourImageModelClass`, `YourVideoModelClass` with actual model classes.")
```

Audio_to_Spectrogram.py:

```
waveform, sr = torchaudio.load(str(audiopath))

if waveform.shape[0] > 1:

    waveform = torch.mean(waveform, dim=0, keepdim=True)

if sr != samplerate:

    resampler = T.Resample(sr, samplerate)

    waveform = resampler(waveform)

maxlen = samplerate * maxlength

if waveform.shape[1] < maxlen:

    waveform = torch.nn.functional.pad(waveform, (0, maxlen - waveform.shape[1]))

else:

    waveform = waveform[:, :maxlen]

melspec = T.MelSpectrogram(samplerate, n_fft=nfft, hop_length=hoplength, n_mels=nmels, power=2.0)(waveform)

melspecdb = T.AmplitudeToDB(top_db=80)(melspec)

melspecdb = (melspecdb - melspecdb.mean()) / (melspecdb.std() + 1e-8)
```

EnhancedXceptionNet (Main Snippet Block):

```
class EnhancedXceptionNet(nn.Module):
    def __init__(self, num_classes, dropout_rate=0.3, use_attention=True):
        super(EnhancedXceptionNet, self).__init__()
        self.base_model = timm.create_model('xception', pretrained=False)
        in_features = self.base_model.get_classifier().in_features
        self.base_model.reset_classifier(0)
        self.use_attention = use_attention
        if use_attention:
            self.attention = AttentionBlock(in_features)
        self.classifier = nn.Sequential(
            nn.Dropout(dropout_rate),
            nn.Linear(in_features, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),
            nn.Dropout(dropout_rate/2),
            nn.Linear(512, num_classes)
        )
    def forward(self, x):
        features = self.base_model(x)
        if self.use_attention:
            features = self.attention(features)
        output = self.classifier(features)
        return output
```

RESULT COMPARISON AND ANALYSIS

For every project, the performance plays the key role in estimating the success rate. Our Deepfake Detection and Attribution is supporting three media i.e., video, audio, and image. The outcome dashboard will display confusion matrix, confidence scores, accuracy, precision, F1-score, recall for examining the performance.

1. IMAGE DETECTION EVALUATION

We used XceptionNet which is Enhanced version by uniting it with Attention mechanism to get accurate results.

- We achieved the accuracy for image Deepfake detection is 99%.
- And we gained 0.995 F1-Score for the image detection criteria.

We use the Attention mechanism to improve our results like it can easily detect the following features.

- Inconsistent facial expressions.
- Abnormality in eyes.
- Change in skin texture.

We can use this model for detecting the Deepfakes which are generated by the DALL-E, Midjourney, StyleGAN, FaceSwap and other AI tools which are mainly used for the generation of Deepfakes.

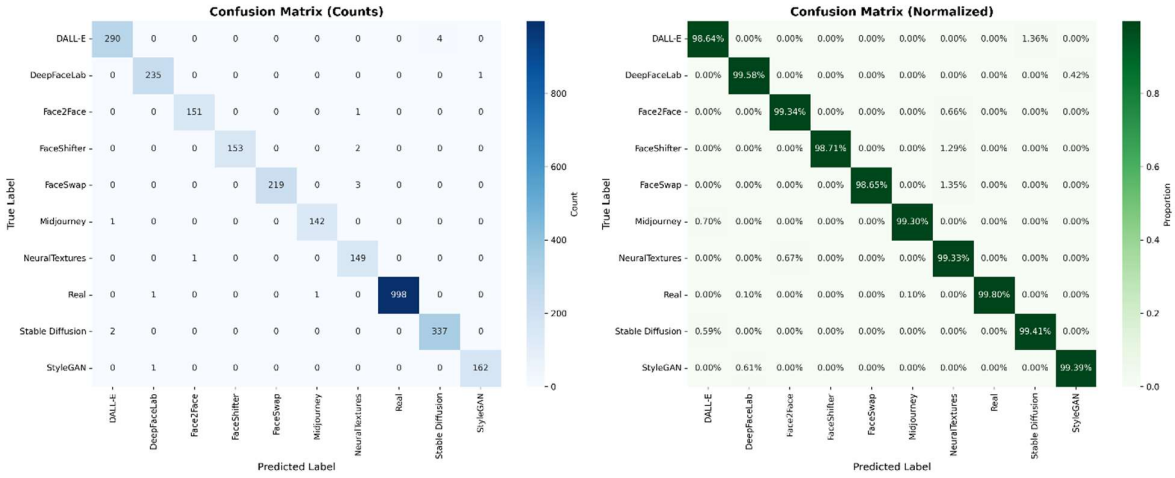


Image DFDA Confusion Matrix

2. VIDEO DETECTION EVALUATION

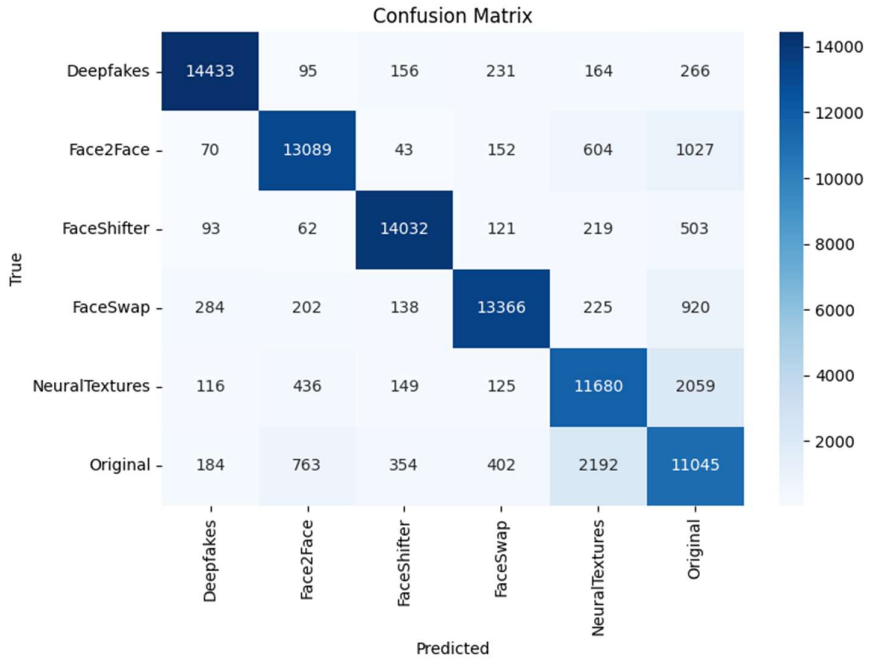
The video detection is mainly based on the XceptionNet classification with the combination of temporal dynamics. We can get the successful results for this to; the following are the factors we achieved:

- This model achieved the validity accuracy of 90% for the Deepfake detection.
- Also, it successfully accomplished the Precision of 0.91 and recall of 0.90.

This model can detect the video including the limitations of the earlier systems. It can successfully detect the following:

- Facial inconsistencies.
- Skin texture.
- Eye movement identification.
- Movement inconsistency.

We can predict the AI content of the video which is generated by AI, by analysing the temporal dynamics.



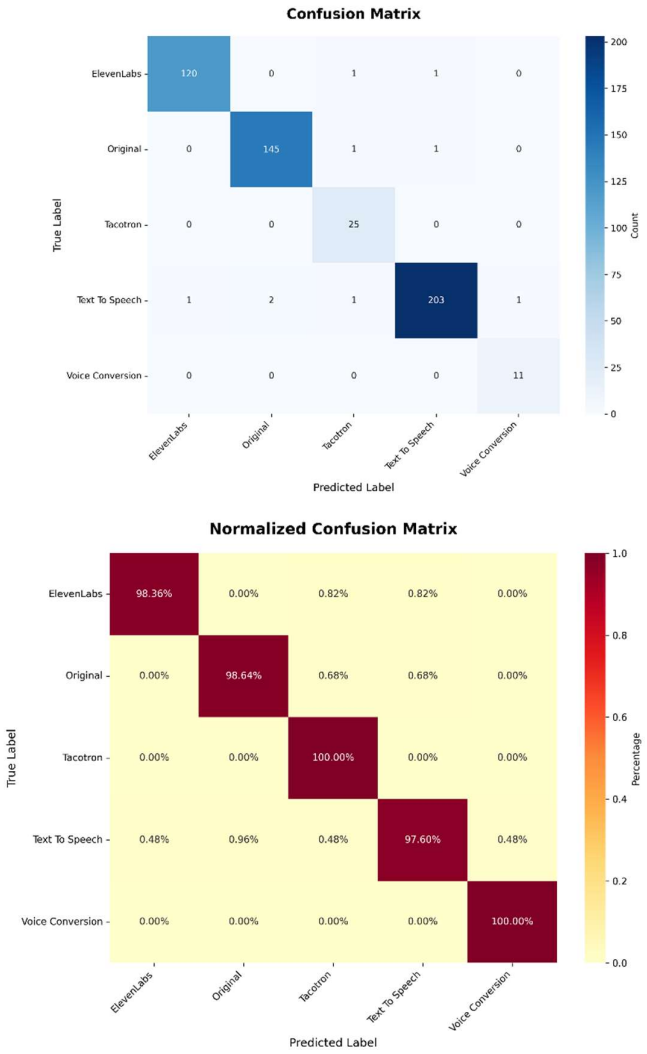
Video DFDA Confusion Matrix

3. AUDIO DETECTION EVALUATION

The audio which is collected from different data sources are moved through a CNN which is 4-layered model for training. And then they are classified into Mel-Spectrograms for the evaluation.

- The validation accuracy we achieved for the Deepfake detection of the audio is 98%.
- It also achieved the F1-Score of 0.97% for the detection of the Deepfake.

The audios are collected from the Voice Cloning, text to Speech conversions, and other sources. Then this will be passed through the model and classified into Spectrograms. Here, the inconsistencies in the pitch of the voice are detected by the spectral analysis.



Audio DFDA Implementation

4. DETECTION SPEED

Deepfake Detection and Attribution System is a multi-modal which supports three media i.e., audio, video and image Deepfake identification. But all the three won't give same performance. The three media will be evaluated in three methods which will affect on the performance speed of their respective model. The following are the representations:

Media	**Processing Speed**
Image	2-3 sec
Video	3-5 sec
Audio	2-3 sec

5. PERFORMANCE OF DIFFERENT MEDIA

Media	**System Architecture**	**Validation Accuracy**	**Observation**
Image	It is a combination of both Enhanced XceptionNet and Attention Mechanism.	99%	Overall performance is fast and it is very accurate.
Video	It is the combination of the XceptionNet with the temporal dynamics.	90%	The performance is strong for this media.
Audio	It is passed through a 4-layered CNN and made into Mel-Spectrograms.	98%	Very accurate results are generated.

The overall performance of the Deepfake Detection and Attribution (DFDA) achieved the accuracy of 95% which is a multi-media platform consisting of three media i.e., audio, image and video.

The challenges like overfitting, reduction of classification confidence and displaying binary numbers as results are the limitations of the existing systems. We made them to achieve all the limitations of the earlier systems.

Finally, we achieved the overall validation accuracy of 95 for our Deepfake Detection and Attribution System. It is real-time project which is used for the identification of the Deepfakes and also traces its source from where that particular Deepfake is generated and displays the accuracy, confidence score, confusion matrix for the faked content with respective to the media we are uploading.

VII. LEARNING OUTCOME

By developing and testing the Deepfake Detection and Attribution (DFDA) system, the following major skills and learnings were achieved by our project team:

1. Understanding Deepfake:

In this project we developed a clear understanding of how artificial intelligence can generate and manipulate multimedia content such as voices, images and videos.

2. Model Design and Integration:

We learned how to design and integrate multiple deep learning architectures into one single system.

- **Audio**: CNN (Convolutional Neural Network).
- **Image**: Enhanced XceptionNet with Attention layers.
- **Video**: Legacy Xception model with temporal aggregation.

Making these all these models together into one framework improved our understanding of multi model AI systems.

3. Feature Engineering Skills:

Our project increased our ability to perform feature extraction and preprocessing for different types of input data.

- **Audio:** Converted sound waves into **Mel-Spectrograms**, capturing frequency and time-based information.
- **Image:** Processed **RGB images** for spatial pattern recognition and manipulation detection.
- **Video:** Extracted **frame sequences** using OpenCV and performed temporal analysis to detect inconsistencies across frames.

These skills taught us how crucial feature representation is for deep learning model performance.

4. Implementation with Modern Tools:

We gained a lot of hands-on experience with AI tools and data processing libraries, including:

- **PyTorch** for building and training deep learning models
- **Streamlit** for developing an interactive user interface
- **Librosa** for audio signal analysis
- **OpenCV** for video and image processing
- **TIMM** (PyTorch image models) for using advanced pre-trained model architectures

By these libraries we gain a practical experience in end- to-end AI system development, from backend model training to frontend deployment.

5. Evaluation and Visualization:

By our project we learned to assess model performance scientifically using classification metrics such as accuracy, precision, recall and F1-score. These reports made our results more transparent.

6. Practical Problem-Solving:

Development process improved our real-world problem-Solving skills. We learned the importance of debugging, optimization, and experimentation in deep learning projects.

7. Team Collaboration:

As we worked as a team we improved our communication, coordination, and project management skills. We learned how collaborative development leads to efficient progress and higher-quality outcomes.

VIII. CONCLUSION WITH CHALLENGES

By this we conclude that our project The Deepfake Detection and Attribution (DFDA) provides an impact solution for identifying and analyzing manufactured media across audio, image and video. By applying deep learning models such as CNN, XceptionNet, and attention mechanisms which helps us in identifying manipulated media. We used different advanced deep learning techniques and frameworks such as PyTorch, Streamlit, and TIMM, our system successfully detected AI-generated content with more accuracy and reliability.

Our project strengthens our knowledge of machine learning, model integration, and data processing. Deepfake detection system contributes to building a safer digital environment by helping users identify misleading or manipulated media. Our project made us, to be aware of ethical responsibilities involved in artificial intelligence and the importance of using technology to maintain truth and trust in the digital world.

Our project not only achieved its technical goals but also improved our skills in deep learning, teamwork, and research-oriented development.

CHALLENGES FACED:

- Gathering large and diverse dataset.
- Training deep models, mainly video-based networks, required important GPU resources and time. Enhancing model performance under limited hardware conditions.
- Combining different models into a single framework required complex data flow management.
- Verifying alignments and normalization across modalities took most of the time and effort.
- Processing large video files or high-quality audio clips slowed down detection speed, making real-real Deepfake detection challenging.

IX. REFERENCES

[1] Author: Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus

Title: FaceForensics++: Learning to Detect Manipulated Facial Images

Journal: IEEE/CVF International Conference on Computer Vision (ICCV)

Year: 2019

Link: https://arxiv.org/abs/1901.08971

[2] Author: François Chollet

Title: Xception: Deep Learning with Depth wise Separable Convolutions

Journal: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Year: 2017

Link: https://arxiv.org/abs/1610.02357

[3] Author: Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, Cristian Ferrer

Title: The Deepfake Detection Challenge (DFDC) Dataset

Journal: arXiv Preprint

Year: 2020

Link: https://arxiv.org/abs/2006.07397

[4] Author: Thanh Thi Nguyen, Cuong M. Nguyen, Dinh Thai Nguyen, Duc Thanh Nguyen

Title: Deep Learning for Deepfakes Creation and Detection: A Survey

Journal: arXiv Preprint

Year: 2019

Link: https://arxiv.org/abs/1909.11573

[5] Author: Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, Hao Li

Title: Detecting Deep-Fake Videos from Phoneme-Viseme Mismatches

Journal: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)

Year: 2020

Link:

https://openaccess.thecvf.com/content_CVPRW_2020/html/WIFS/Agarwal_Detec

ting_Deep-Fake_Videos_from_Phoneme-Viseme_Mismatches_CVPRW_2020_paper.html

[6] Author: Jennifer Frank, Tobias Eisenhofer, Lars Schönherr, Asja Fischer, Dorothea Kolossa, Thorsten Holz

Title: Leveraging Frequency Analysis for Deep Fake Image Recognition

Journal: International Conference on Machine Learning (ICML) Workshops

Year: 2020

Link: https://arxiv.org/abs/2003.08685

[7] Author: Pavel Korshunov, Sébastien Marcel

Title: Speaker Inconsistency Detection in Tampered Video

Journal: IEEE International Conference on Biometrics Theory, Applications and Systems (BTAS)

Year: 2018

Link: https://ieeexplore.ieee.org/document/8698536

[8] Author: Tanmay Mittal, Uttaran Bhattacharya, Ruchika Chandra, Aniket Bera, Dinesh Manocha

Title: Emotions Don't Lie (ok): A Deepfake Detection Method Using Emotional Inconsistency

Journal: ACM International Conference on Multimedia (ACM MM)

Year: 2020

Link: https://dl.acm.org/doi/10.1145/3394171.3413534

[9] Author: Luigi Guarnera, Oliver Giudice, Sebastiano Battiato

Title: Deepfake Detection by Analyzing Convolutional Traces

Journal: IEEE/CVF Conference on Computer Vision and Pattern Recognition

Year: 2020

Link:https://openaccess.thecvf.com/content_CVPRW_2020/html/WIFS/Guarnera_Deepfake_Detection_by_Analyzing_Convolutional_Traces_CVPRW_2020_paper.html

[10] Author: Jie Zhang, Zhenyu Wu, Shilei Liu

Title: A Survey on Deepfake Audio Detection: Challenges and Progress

Journal: IEEE Access

Year: 2022

Link: https://ieeexplore.ieee.org/document/9865790

PROJECT ARTIFACTS AND SUPPORTING LINKS

Source Code Repository

https://github.com/jayanthbottu/Deepfake-Detection-And-Attribution

Demo:

https://huggingface.co/spaces/jayanthbottu/DFDA

Datasets

https://www.kaggle.com/datasets/jayanthbottu/audio-Deepfake

https://www.kaggle.com/datasets/jayanthbottu/labeled-Deepfake-image-collection

https://www.kaggle.com/datasets/fatimahirshad/faceforensics-extracted-dataset-c23

Technical Blog Article

https://medium.com/@jayanthbottu/Deepfake-detection-attribution-a1a70045a213

Social Media & Outreach

https://www.linkedin.com/posts/jayanthbottu_dfda-deeplearning-teamwork-activity-7387918771216297984-eU9r?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEA-G7kBjfFLQBmdZbRhWJ3_RbOyYs3wHGI

https://www.linkedin.com/posts/jayanthbottu_Deepfakedetection-artificialintelligence-activity-7394065947256549376-CwPK?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEA-G7kBjfFLQBmdZbRhWJ3_RbOyYs3wHGI

https://x.com/jayanthbottu/status/1988302973152514261?s=20

Walkthrough Video

https://youtu.be/4l3FkZkIstA