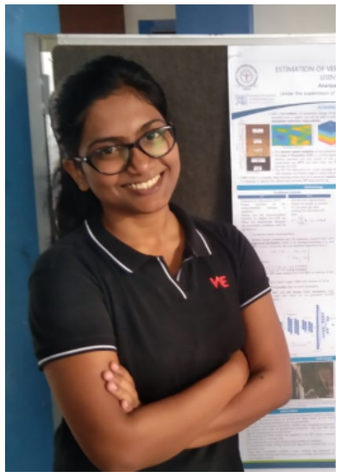
DEEPFAKE DETECTION

Meet Our Team

Akansha Madavi
(2103104)

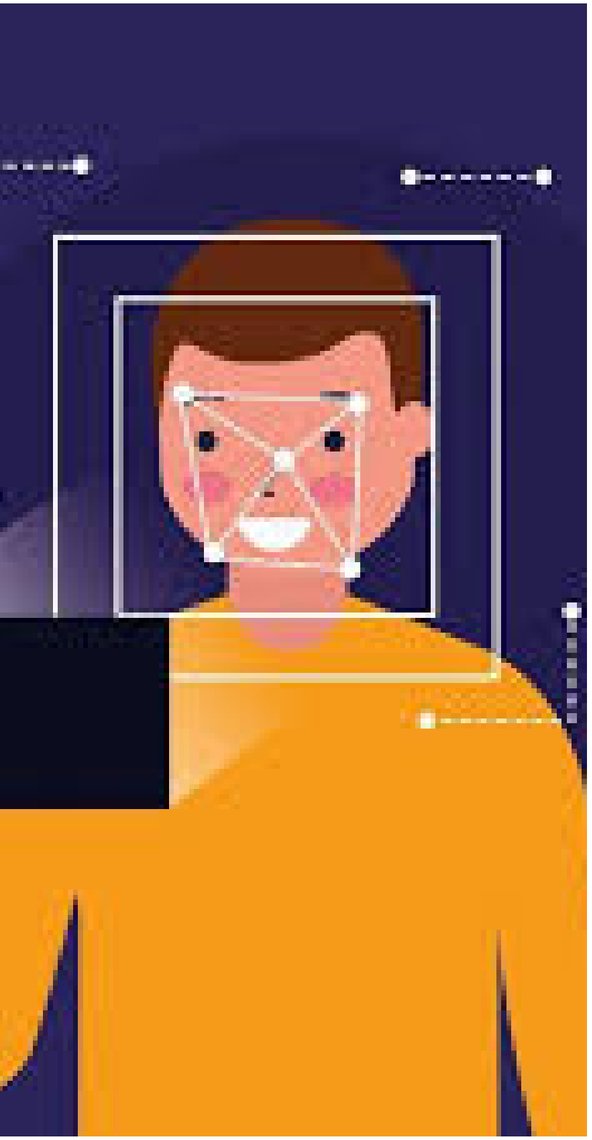


Ananya Purkait
(2103303)

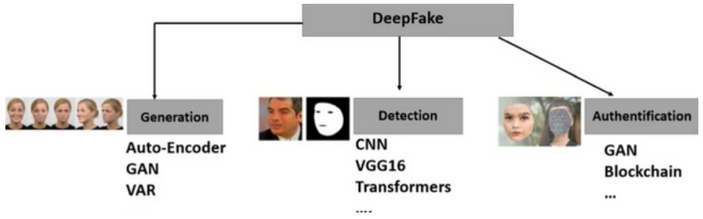
Shreya Marda
(2103321)

TABLE OF CONTENTS

What are DEEPFAKES?



- AI-generated synthetic media (videos, images, audio)
- Manipulated or created using Artificial Intelligence (AI)
- Origin of "**Deepfake**": Combines "**deep learning**" + "**fake**"



How are They Made?

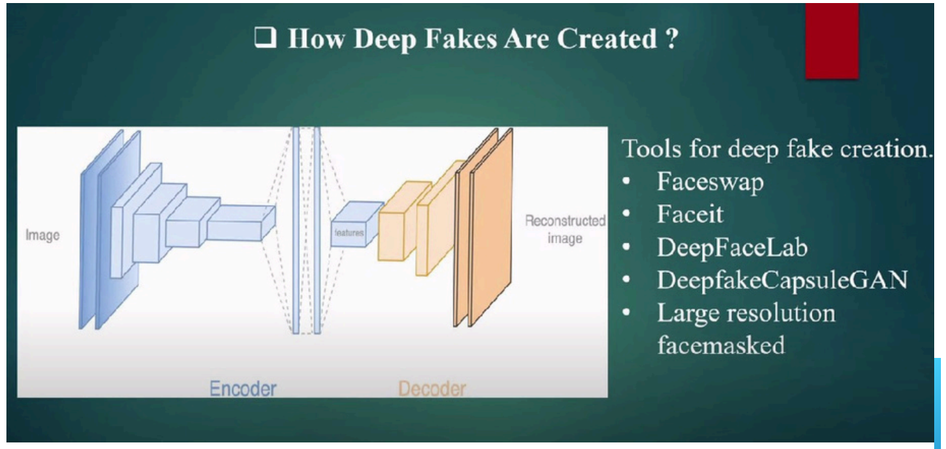
1.Generative Adversarial Networks (GANs):

- Two neural networks (Generator & Discriminator)
- Create realistic fake content

2.Autoencoders:

- Extract facial features
- Map and synthesize altered media

3.Techniques: Face swapping, motion transfer



Purpose of the Project

- Build a machine learning model to detect deepfakes
- Protect societal trust and prevent misuse of technology
- Ensure authenticity of digital content
- Stay ahead of evolving threats in AI manipulation

Types of DEEPFAKES?

1. Video Deepfakes

2. Audio Deepfakes

3. Image Deepfakes

4.Text-bsed-Deepfakes

5. Lip-Sync Deepfakes

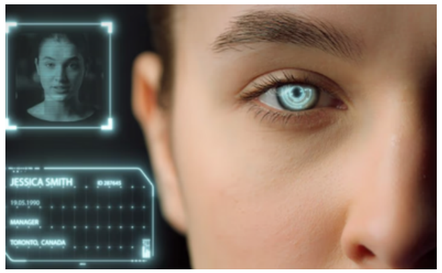
6. Face Swap Deepfakes

7.Gestures and body movement manipulation

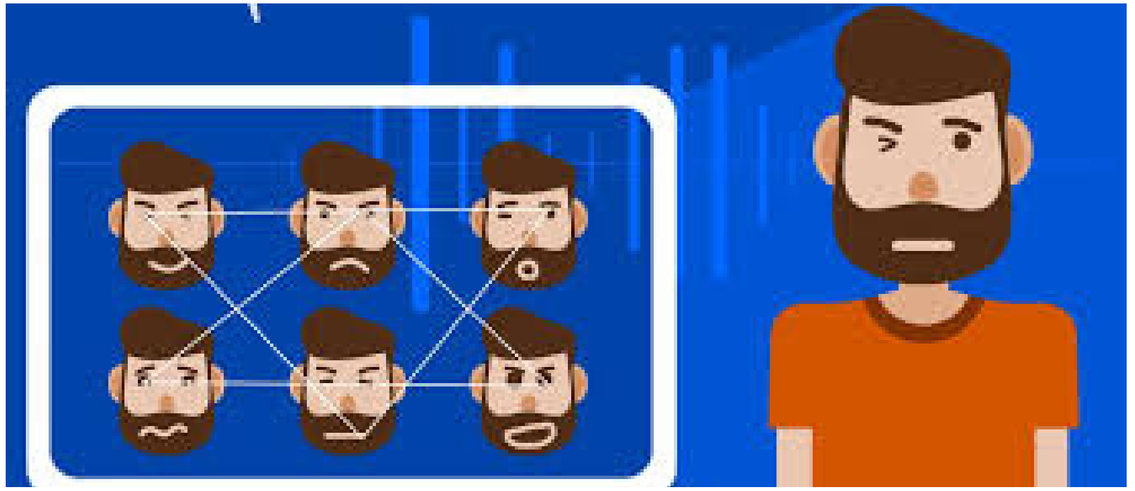
8. Digital Doppelgangers

9. Object Manipilation

10. Generative Adversarial Network (GAN)-Based Deepfakes



IMPORTANCE OF DEEPFAKE DETECTION



Impact on Society
Significant threats to individuals, organizations, and society

01

Misinformation:

- Fuels fake news and propaganda
- **Example**: Altered political speeches, historical events

02

Cybersecurity Risks:

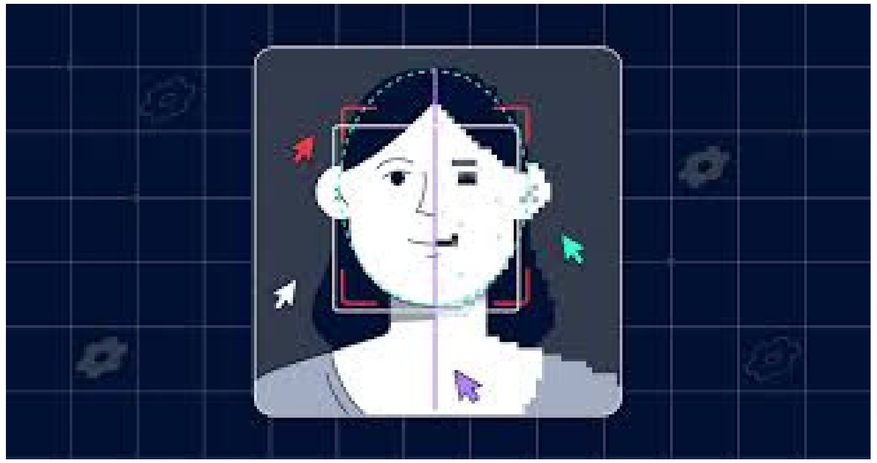
- Identity theft through mimicked appearances or voices
- **Example**: Fake CEO videos for fraudulent fund transfers

03

Erosion of Trust:

- Questions authenticity of videos, images, audio
- Leads to **"truth decay"** where even real evidence is doubted

CHALLENGES IN DETECTION



Detecting deepfakes is an ever-evolving challenge due to advancements in AI and synthesis techniques:

01

High Realism :

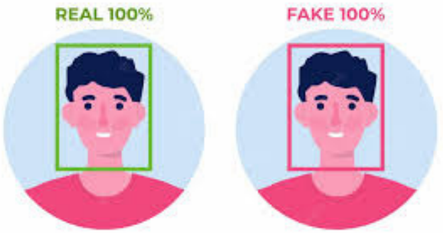
- Near-perfect replication of expressions and emotions
- GANs improve constantly, narrowing the gap between fake and real

02

Continuous Evolution:

- Rapid advancements in deepfake techniques
- AI models becoming more accessible and efficient
- Countermeasures developed to bypass detection tools

Why Detection Matters



- Protects democratic processes, financial security, and personal safety
- Maintains trust in digital media
- Safeguards against malicious AI us

Methodology

01

Experimentation with ConvNeXt and ViT for Classification Tasks:

- Dataset: **100,000** samples (**80,000** train, **20,000** test).
- Train-Test Split: Custom **fraction-based sampling.**

- Models Used: **ConvNeXt, ViT**

- Scenarios Explored:

1. **No Fine-Tuning**
2. **Fine-Tuning**
3. **AutoAugment**
4. **RandAugment**
5. **Combined AutoAugment and RandAugment.**

- Metrics Evaluated:

1. Training Loss
2. Accuracy
3. F1 Score.

- Result:

1. Model 1 is trained and being used for next part of work.
2. Model 2 is running on its 6(/10)th epoch.

Methodology

02

Pre & Post Processing and Saliency Map:

- Model: Load pretrained **ConvNeXt** Tiny with modified classifier for **binary** output (Fake/Real classification).

- Preprocessing: **Frames** resized to 224x224, normalized, and converted to tensors.

- **Saliency Maps**:

1. Gradients highlight influential regions for predictions.
2. Overlaid on frames for visualization.

- WorkFlow:

1. Reads frames from the uploaded video.
2. Processes every **10th frame** to reduce computation.
3. Classifies each frame as "Fake" or "Real" using the model's output probability (**threshold: 0.5**).
4. Adds the **generated saliency map**s to the output video.
5. Accuracy Calculation based on label
6. **Streamlit** Integration

Methodology

03 Dependencies & Libraries Used

Core Dependencies:
- Torch & Torchvision
- Timm
- Scikit-learn
- Tensorflow

Data Processing & Visualization:
- Pandas
- Matplotlib & Seaborn
- Pillow

Additional Tools:
- Streamlit
- Tqdm
- glob
- cv2 (OpenCV)

etc.

04 Data Augmentation & Experimental Protocol

Employ advanced data augmentation techniques:

AutoAugment RandAugment Combination of both

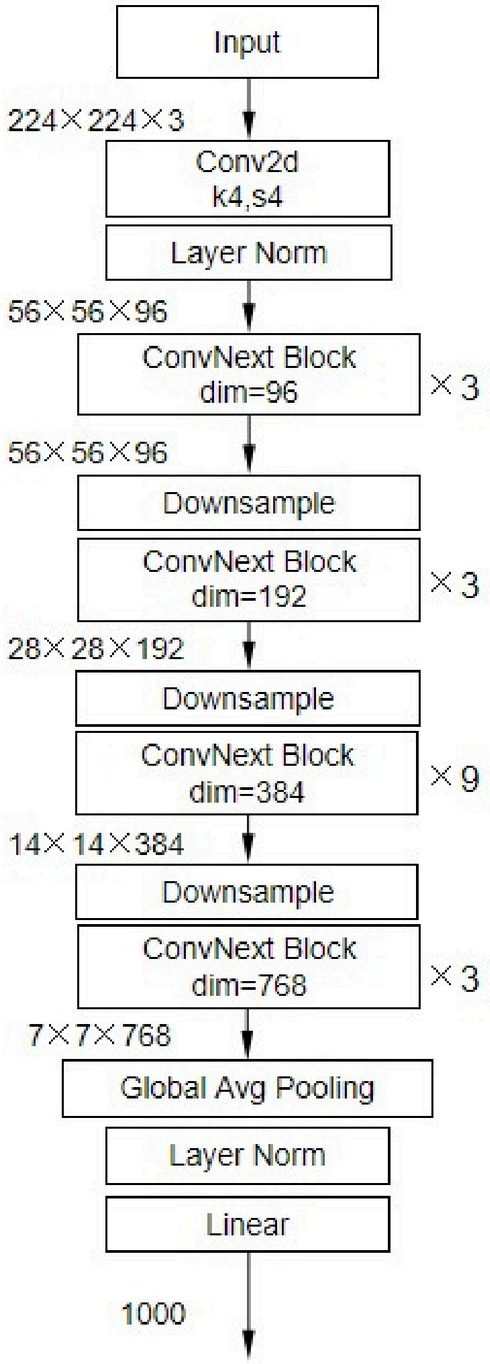
With Fine Tuning Without Fine Tuning

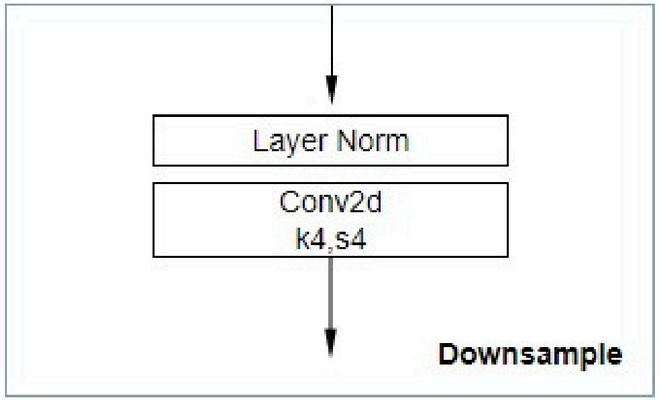
- **Goal**: Improve model performance and generalization using diverse, enriched datasets.

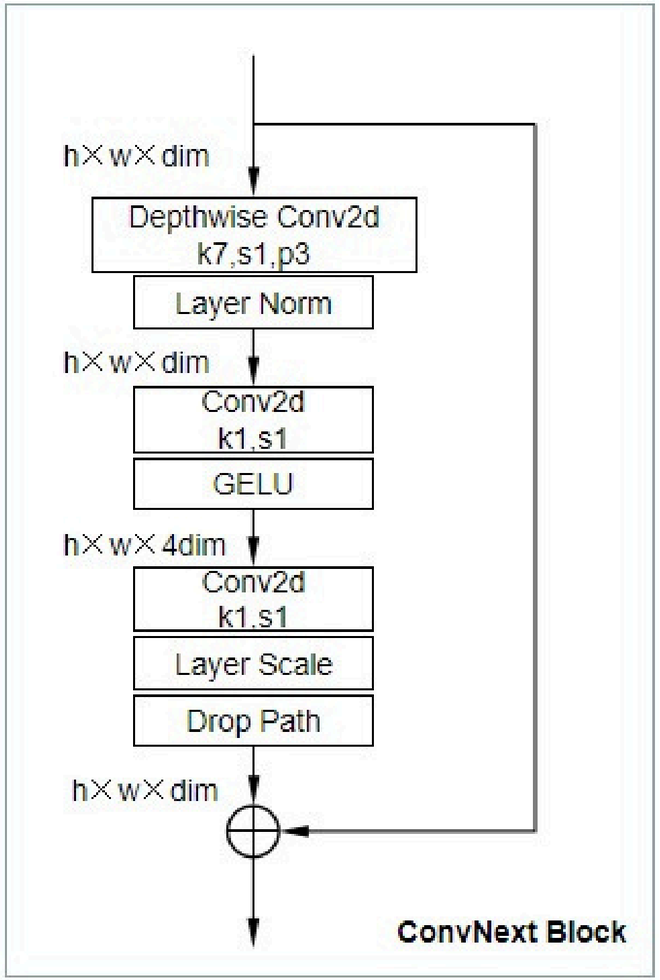
ConvNeXT:

- ConvNeXt modernizes traditional CNNs by integrating principles from Vision Transformers (ViTs).
- It replaces BatchNorm with LayerNorm, uses larger kernels, and simplifies activation functions for better performance.
- Achieves state-of-the-art results in image classification and other vision tasks.
- Introduced in the paper **"A ConvNet for the 2020s"**

ViT: Processes images as a sequence of patches using Transformer encoders







Future Scope :

- Audio Deepfakes
- Customized Deepfakes
- Biometric Implementation
- Digital Doppelgangers
- Deepfake Detection Techniques
- Deepfake Frauds
- Ethical and Legal Debates

Real / Fake

Real

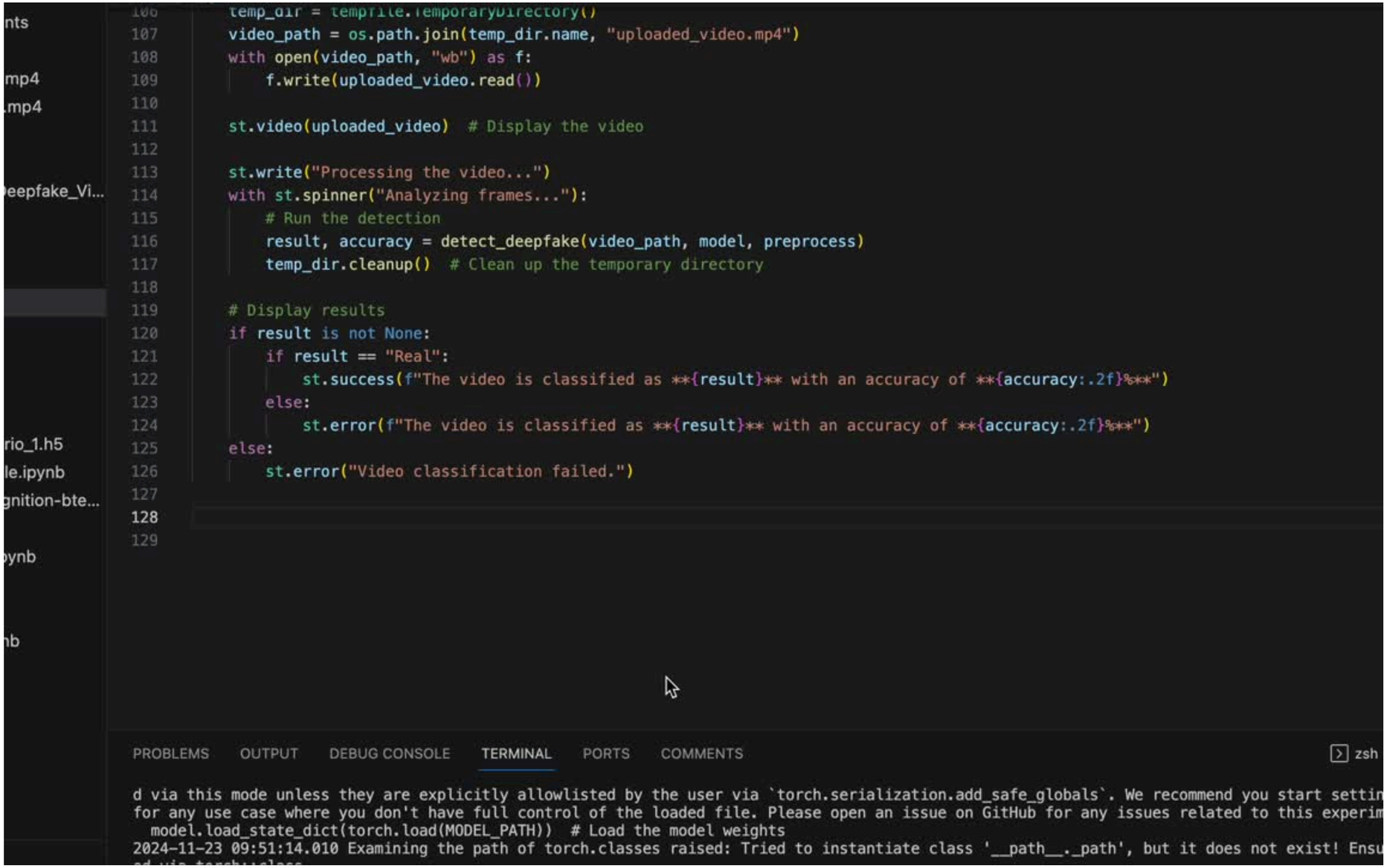
Fake

FAKE

```python
import streamlit as st
import cv2
import tempfile
import os
import torch
import torchvision.models as models
from torchvision import transforms
import torch.nn as nn
import numpy as np

# Define the app title
st.title("Deepfake Detection")

# Load the trained model
Tabnine | Edit | Test | Explain | Document | Ask
@st.cache_resource
def load_model():

    BASE_DIR = "/Users/ananyapurkait/Study Files and Folders/Semester Study/Sem VII/CS435/Project/"  # Replace with the actual pa
    MODEL_PATH = os.path.join(BASE_DIR, "convnext_scenario_1.h5")

    # Assuming you have a pre-trained ConvNeXt model structure
    model = models.convnext_tiny(pretrained=False)  # Initialize the ConvNeXt model
    model.load_state_dict(torch.load(MODEL_PATH))  # Load the model weights

    # Modify the classifier for binary classification
    model.classifier[2] = nn.Linear(model.classifier[2].in_features, 1)
    model.eval()  # Set the model to evaluation mode
    return model

# Preprocessing function for video frames
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS  zsh
ananyapurkait@Ananyas-MacBook-Air Project % stream
```

REAL


temp_dir = tempfile.TemporaryDirectory()
video_path = os.path.join(temp_dir.name, "uploaded_video.mp4")
with open(video_path, "wb") as f:
f.write(uploaded_video.read())
st.video(uploaded_video) # Display the video
st.write("Processing the video...")
with st.spinner("Analyzing frames..."):
Run the detection
result, accuracy = detect_deepfake(video_path, model, preprocess)
temp_dir.cleanup() # Clean up the temporary directory
# Display results
if result is not None:
if result == "Real":
st.success(f"The video is classified as **{result}** with an accuracy of **{accuracy:.2f}%**")
else:
st.error(f"The video is classified as **{result}** with an accuracy of **{accuracy:.2f}%**")
else:
st.error("Video classification failed.")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
zsh
d via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start settin
for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experim
model.load_state_dict(torch.load(MODEL_PATH)) # Load the model weights
2024-11-23 09:51:14.010 Examining the path of torch.classes raised: Tried to instantiate class '__path__._path', but it does not exist! Ensu

Thank You For Watching !