**DeepFake Detection**

**PROBLEM STATEMENT**

Deepfake technology uses AI to generate hyper-realistic fake videos that can be weaponized to spread **dangerous misinformation** (e.g., fake political speeches), enable **financial fraud** (e.g., impersonating CEOs), or facilitate **identity theft** (e.g., forged biometric data). While these threats grow, existing detection tools often suffer from **slow processing speeds**, **complex interfaces** , or **opaque decision-making**. To combat this, there is an urgent demand for a **user-friendly web application** that offers: Real-time analysis , Intuitive design.

**PROJECT OVERVIEW**

This **Django-based web application** detects deepfakes in uploaded videos using a hybrid approach:

1. **Computer Vision**: Analyzes facial inconsistencies (e.g., unnatural blinking, lighting) via OpenCV and face\_recognition.
2. **Deep Learning**: Employs PyTorch models (e.g., ResNet, EfficientNet) trained on datasets like FaceForensics++ to flag AI-generated artifacts.

**User-Friendly Interface**: Allows non-technical users to upload media, track progress with TQDM, and receive instant results with visual explanations (heatmaps, confidence scores).

**SOLUTION OFFERED**

**Automated DeepFake Detection:** The system processes uploaded videos, extracts frames, detects faces, and classifies authenticity using deep learning.

**Real-time Feedback:** Users receive real-time results with confidence scores to assess video authenticity.

**Advanced Deep Learning Models:** Incorporates CNNs and RNNs (including EfficientNet & MobileNet) for high-accuracy classification.

**User-Friendly Web Application:** A simple and intuitive interface for seamless video upload and result visualization.

**END USERS**

* **Journalists/Media:** Verify authenticity of viral content.
* **Legal Teams:** Analyze evidence for forensic investigations.
* **Social Media Platforms:** Automatically flag Check personal media credibility,deepfakes.
* **General Public:** Check personal media credibility.

**TECHNOLOGIES USED**

**Programming Language** -> Python

**Web Framework** -> Django (backend), HTML/CSS/JS (frontend)

**Deep Learning** -> PyTorch, Torchvision (pretrained models)

**Computer Vision** -> OpenCV, face\_recognition (facial analysis)

**Data Handling** -> Pandas, NumPy, scikit-learn, PIL

**System Interaction** -> OS module, UUID (file management)

**Progress Tracking** -> TQDM (user-facing progress bars)

**CONCLUSION**

This project bridges the gap between cutting-edge AI research and practical, user-centric tools. By combining **Django’s scalability** with **PyTorch’s deep learning capabilities** and **OpenCV’s real-time processing**, it delivers an accessible, robust solution to combat deepfakes. Future enhancements could include adversarial training for evolving threats, mobile integration, or blockchain-based verification for tamper-proof audits.