DeepFake Video Detection Project

Table of Contents:

Overview: The DeepFake Video Detection Project is designed to detect manipulated or “deepfake” videos using advanced machine learning techniques. By analyzing subtle cues within video frames, the system differentiates between genuine and artificially altered content. The project combines computer vision with deep learning models—including Convolutional Neural Networks and Recurrent Neural Networks—to analyze and classify video content. This document provides guidelines on setting up, running, and extending the project.

Features: • Automated Detection: Scans videos and identifies potential deepfakes automatically. • Frame Extraction: Processes videos by extracting frames for analysis. • Deep Learning Models: Utilizes state-of-the-art CNN and RNN architectures. • Real-Time Analysis: Capable of processing video streams with near real-time performance. • User-Friendly Interface: Simple command-

line interface to initiate detection tasks. • Scalable Design: Modular codebase designed to allow future enhancements and additional features.

System Requirements: • Operating System: Linux, macOS, or Windows (with minor modifications) • Python Version: 3.7 or later • Hardware:

- CPU with at least 4 cores (for testing)
- GPU (NVIDIA recommended) for accelerated training and inference • RAM: Minimum 8 GB (16 GB recommended for larger datasets)

Installation:

1. Clone the Repository:
    - Use the Git command to clone the repository and navigate to the project folder.
2. Create a Virtual Environment:
    - Create and activate a Python virtual environment.
3. Install Dependencies:
    - Install required packages as listed in the requirements file.
4. Download Pre-trained Models (Optional):
    - For a quicker setup, download the provided pre-trained models from the Releases page and place them in the models directory.

Usage: Running the Detection Script: • To analyze a video for deepfake content, run the detection script by providing the input video file path and output results file path. • The script will extract frames from the video, process each frame through the deep learning model, and output a file with detection probabilities and timestamps for potential deepfake segments.

Command-Line Options: • --input: Path to the input video file. • --output: Destination file for detection results. • --model: (Optional) Specify a custom model file. • --threshold: (Optional) Set a confidence threshold (default is 0.5).

Running in Batch Mode: • For processing multiple videos, place your videos in a designated folder and run the batch detection script with appropriate input and output directory paths.

Data Preparation: • Video Data:

- Ensure videos are in a supported format (e.g., MP4, AVI) and have consistent frame rates. • Annotations (Optional):
- For training and fine-tuning, prepare a CSV file with video file names and corresponding labels (real or deepfake). For example: filename,label video1.mp4,real video2.mp4,deepfake • Frame Extraction:
- The detection script automatically extracts frames from videos.
- A separate utility is provided for manual frame extraction if needed.

Code Structure: The repository is organized as follows: • models/ – Pre-trained models and checkpoints. • data/ – Example datasets and annotations. • scripts/ – Helper scripts for data processing and training. • detect.py – Main script for video detection. • batch_detect.py – Script for batch processing of videos. • train.py – Training script for model fine-tuning. • utils.py – Utility functions (e.g., frame extraction, preprocessing). • requirements.txt – List of Python package dependencies.

Each module is designed to be modular and easily extendable, allowing developers to integrate new features or swap out model components as needed.

Model Training & Evaluation: Training the Model: • To fine-tune the deep learning model on your custom dataset, run the training script with parameters for the data directory, number of epochs, and batch size. • Key parameters include:

- --data_dir: Directory containing training and validation data.
- --epochs: Number of training epochs.
- --batch_size: Batch size for training.

Evaluation Metrics: • Accuracy – Overall correctness of the model. • Precision & Recall – Detailed metrics for detecting deepfakes. • Confusion Matrix – Visual representation of true vs. predicted classes. • ROC Curve – Performance visualization across different threshold settings. Evaluation outputs are saved in the results directory, with plots and metric summaries provided.
