

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | | Intern Recruitment Challenge20-03-25 |  | | |
|  |  | |  | |  | |
|  |  | | Phase 1 Report:  Deepfake Detection System | |  | |
|  |  | | | | |  |
|  | **Project Overview** The Deepfake Detection System aims to classify videos as real or fake using advanced computer vision techniques. This report focuses on **Phase 1: Data Collection & Preprocessing**, covering dataset collection, feature extraction, and the challenges encountered during the process. **Phase 1: Data Collection & Preprocessing****Step 1: Diverse Deepfake Dataset Collection** To develop a robust deepfake detection model, I gathered deepfake datasets from various sources:   * **DeepFake Detection Challenge (DFDC)**: A large-scale dataset containing real and deepfake videos. * **FaceForensics++**: A benchmark dataset with different compression levels of fake videos. * **Celeb-DF**: A dataset featuring realistic deepfakes of celebrities. * **DF-TIMIT**: A speech-driven deepfake dataset focusing on manipulated audio and video. * **GAN-Generated Deepfakes**: I generated synthetic deepfake samples using **StyleGAN, DeepFaceLab, and First Order Motion Model** to create adversarial attacks for model robustness.  **Challenges Faced & Solutions**  1. **Dataset Availability & Access**    * Challenge: Some datasets required specific permissions, and obtaining high-quality deepfake videos was difficult.    * Solution: Applied for research access where needed and leveraged publicly available deepfake datasets. 2. **Large Storage & Processing Power**    * Challenge: Handling large datasets required significant computational resources.    * Solution: Used cloud storage (Google Drive, Kaggle Datasets) and Google Colab for preprocessing and model training. 3. **Data Cleaning & Labeling**    * Challenge: Some datasets lacked proper labeling or contained noise.    * Solution: Implemented **OpenCV-based preprocessing** to clean datasets and ensured correct labeling of real vs. fake samples.  **Step 2: Multi-modal Feature Extraction** To detect deepfake patterns, I extracted essential features from both **video and audio** sources. ****Video-Based Feature Extraction****  * **Facial Landmarks**: Used **Dlib and OpenCV** to detect inconsistencies in facial movements. * **Micro-Expressions & Eye Movements**: Applied **OpenFace and EmoReact** models to track facial expressions and blinking irregularities. * **Temporal Inconsistencies**: Implemented **frame interpolation** to detect abrupt changes between frames.  ****Audio-Based Feature Extraction****  * **MFCC (Mel-Frequency Cepstral Coefficients)**: Extracted voice features to identify unnatural voice modulations. * **Spectrogram Analysis**: Used **Librosa** to visualize and analyze fake voice spectrograms.  **Challenges Faced & Solutions**  1. **Feature Extraction Complexity**    * Challenge: Extracting meaningful features from both video and audio required advanced preprocessing techniques.    * Solution: Utilized pre-trained models such as **Dlib for face tracking** and **Librosa for voice analysis**. 2. **Performance Bottlenecks**    * Challenge: Running facial landmark detection on large datasets was computationally expensive.    * Solution: Used **batch processing** and optimized code execution with multiprocessing. 3. **Audio-Video Synchronization**    * Challenge: Ensuring extracted audio features aligned with corresponding video frames.    * Solution: Implemented **FFmpeg-based synchronization** to maintain frame-to-audio accuracy.  **Conclusion & Future Steps**As a frontend developer and a UI/UX Designer, the entire concept of the data science was new to me before I had opted for this internship. Throughout this internship period, I had successfully completed all the projects and tasks assigned to me by our mentor and I was able to learn a lot under her guidance. On the basis of my performance I am now a part of this Intern Recruitment Challenge where I am currently working on a major project- “Deepfake Detection System”.A lot of challenges were faced by me while I was working on the Phase 1 part of the project but I faced all those challenges and was not only able to complete the Phase 1 task before the deadline date but was also able to learn a lot of new things in the field of data science. Phase 1 involved extensive dataset collection, preprocessing, and feature extraction. Overcoming challenges related to data availability, storage, and computational complexity was crucial in ensuring high-quality training data for the deepfake detection model. The next step is Phase 2, where I will implement **Vision Transformers (ViTs) and 3D-CNN models** for robust deepfake classification. **Phase 2 Report:** Project Overview Following the successful completion of Phase 1, which focused on dataset collection and preprocessing, Phase 2 emphasizes model development and adversarial training. This phase involves implementing multi-modal deepfake detection techniques using Vision Transformers (ViTs), 3D-CNNs, and self-supervised learning methods to enhance model robustness against adversarial deepfake attacks. Phase 2: Model Development & Adversarial TrainingStep 3: Multi-Stream Neural Network for Deepfake Detection To build a robust deepfake detection system, I implemented a multi-stream neural network capable of analyzing both spatial and temporal features in videos. ****Model Architectures Implemented:****  * **Vision Transformers (ViTs)**: Used for spatial analysis of facial inconsistencies in video frames. ViTs offer improved performance over traditional CNNs for feature extraction. * **3D Convolutional Neural Networks (3D-CNNs)**: Applied for analyzing temporal variations in deepfake videos by detecting motion anomalies. * **Self-Supervised Learning Models (DINO, MAE)**: Integrated for generalization, reducing dependence on large labeled datasets. * **Contrastive Learning**: Employed contrastive loss to enhance model differentiation between real and fake patterns.  ****Challenges Faced & Solutions:****  1. **High Computational Requirements**    * Challenge: Training ViTs and 3D-CNNs required significant computational power.    * Solution: Leveraged Google Colab Pro’s TPUs and optimized batch sizes to improve training efficiency. 2. **Model Overfitting**    * Challenge: Training on a limited dataset led to overfitting issues.    * Solution: Applied data augmentation techniques and dropout layers to improve generalization. 3. **Limited Temporal Feature Representation**    * Challenge: 2D-CNNs failed to capture sequential inconsistencies in video frames.    * Solution: Introduced 3D-CNNs to extract motion-based features effectively.  Step 4: Detecting Advanced Deepfake Attacks This step involved enhancing model robustness against adversarial deepfake techniques and low-quality compressed deepfakes. ****Techniques Implemented:****  * **GAN-generated Adversarial Deepfakes**: Included deepfakes from StyleGAN3 and Stable Diffusion-generated video manipulations. * **Low-Quality Compressed Deepfake Training**: Simulated real-world distortions by training the model on heavily compressed videos. * **Forensic Watermarking Detection**: Used frequency-based analysis to identify watermark inconsistencies in fake videos.  ****Challenges Faced & Solutions:****  1. **Adversarial Robustness**    * Challenge: GAN-generated deepfakes continuously evolved, bypassing detection models.    * Solution: Trained the model on a diverse dataset of adversarial attacks to improve detection capabilities. 2. **Handling Video Compression Artifacts**    * Challenge: Compression noise sometimes led to false positives.    * Solution: Applied pre-processing filters to remove unnecessary artifacts while preserving deepfake characteristics. 3. **Forensic Watermarking Complexity**    * Challenge: Identifying subtle watermark patterns required specialized techniques.    * Solution: Used Fourier Transform-based methods to detect tampering traces in GAN-generated videos.  Conclusion & Future Steps Phase 2 focused on model development and adversarial training to improve deepfake detection accuracy. By integrating ViTs, 3D-CNNs, and self-supervised learning, the model achieved significant improvements in detecting both high-quality and low-quality deepfakes. The next step is Phase 3, where I will focus on deploying the system, implementing real-time monitoring, and developing APIs for live deepfake detection. **Phase 3 Report:** **Deployment & Real-time Streaming Analysis****Overview** Phase 3 of the Deepfake Detection System project aimed to operationalize the trained deepfake detection models into real-world environments. The focus was on developing a real-time scanning system with fast inference capabilities and accessible deployment mechanisms. Although Step 5 was successfully implemented, Step 6 could not be completed due to several practical constraints, which are detailed in this report. **Step 5: Real-time Deepfake Scanning & Monitoring****Objectives:**  * Deploy trained models (ViT + 3D-CNN) into a real-time detection pipeline. * Enable video-level scanning with latency under 50ms per frame. * Integrate edge deployment capabilities using TensorFlow Lite. * Build a Gradio-based interface to demonstrate real-time detection.  **Implementation Highlights:**  * **ONNX Export:** The final models were successfully exported to ONNX format, enabling optimized inference and compatibility with lightweight deployment targets. * **Gradio Interface:** A browser-based real-time detection interface was developed using Gradio. Users can upload or stream video, and the system classifies segments as Real or Fake. * **Model Inference:** Leveraged onnxruntime with GPU acceleration to achieve fast inference times, averaging around **42–45ms** per frame. * **Edge Deployment (TF Lite):** Successfully converted the EfficientNetB0 + attention-based spatial model to TensorFlow Lite. Mobile inference was tested locally on a simulated Android environment with **acceptable performance (<100ms/frame)**.  **Challenges and Solutions:**  | **Issue** | **Solution** | | --- | --- | | Model inference time exceeded 50ms due to temporal model complexity (3D-CNN + Bi-LSTM). | Reduced input sequence size and optimized the model using quantization-aware training. | | Gradio didn’t support true video streaming in real-time. | Implemented frame-by-frame simulation with short delays to mimic live feed processing. | | TensorFlow Lite conversion failed initially for 3D-CNN due to unsupported ops. | Only the spatial stream (EfficientNetB0) was converted for edge deployment; the full pipeline remains on server/cloud. |  **Step 6: Social Media & Threat Monitoring** (Not Implemented)**Proposed Objectives:**  * Develop Telegram/Discord bots for deepfake monitoring. * Implement correlation models linking deepfake content to fake news. * Build a browser plugin for detecting YouTube/Twitter deepfakes.  **Reasons for Non-Implementation:**  | **Barrier** | **Explanation** | | --- | --- | | **Lack of Real-Time Public Dataset APIs** | Social platforms (YouTube, Twitter) limit API access for video content due to privacy and rate limits, making it infeasible to automatically scan content at scale. | | **High Infrastructure Demand** | Implementing continuous monitoring (e.g., a bot or plugin that watches for deepfakes 24/7) requires dedicated backend services, GPU inference nodes, and significant cloud resources, which were beyond current scope. | | **Legal and Privacy Concerns** | Deploying bots that analyze user-uploaded content on messaging platforms raises privacy flags. User consent, data processing agreements, and content moderation guidelines would need to be handled. | | **Time Constraints** | Given the internship timeline, prioritizing real-time scanning and edge deployment was more feasible and aligned with project goals. Social threat monitoring was an advanced extension not essential for core detection functionality. |  **Conclusion** Phase 3 successfully demonstrated a real-time deepfake detection system with deployable models and a practical user interface. The system meets performance benchmarks for latency and modularity. While Step 6 presents promising future directions, limitations in infrastructure, legality, and public platform APIs made it impractical to implement within the current project scope. **Future Work:**  * Explore scalable infrastructure (e.g., AWS Lambda + GPU) for continuous monitoring bots. * Partner with fact-checking platforms or social media moderation teams for ethical deployment. * Develop a lightweight browser plugin using WebAssembly for passive deepfake scanning. | | | | |  |
|  | |