Development Phase 1

Detecting Synthetic Faces Using Convolutional Neural Networks

Group 7
Govt Model Engineering College

06/03/2025

Contents

TEST PLAN

Objective of Testing

- Ensure the system accurately detects deepfake and real faces.
- Validate the performance, reliability, and security of the detection model.
- Identify and fix bugs to improve system robustness.
- Ensure smooth user experience with UI/UX testing.

Scope

- **Covered:**
- Image-based deepfake detection.
- CNN-based feature extraction and classification.
- User interface and API functionality testing.
- **Not Covered:**
- Audio or video deepfake detection.
- Real-time surveillance integration.
- Deepfake generation techniques.

Testing Approach

- Manual Testing: UI, usability, and exploratory testing.
- Automated Testing: Model validation, API testing, performance testing.
- Black-Box Testing: Validates input/output accuracy without internal knowledge.
- White-Box Testing: Verifies internal CNN model performance and logic.

Test Environment

- **Hardware:**
- GPU-enabled system (NVDIA RTX 3060 or higher)
- Minimum 16GB RAM, 512GB SSD storage
- **Software and Dependencies**
- Python, TensorFlow
- Web interface- React.js
- Dataset- kaggle

Risks and Mitigation Strategy

- **False Positives/Negatives:**
- Improve dataset quality, fine-tune CNN model.
- **Performance Lag :**
- Optimize model inference, use batch processing.
- **Security Issues:**
- Implement API authentication, secure database access.
- **Dataset Bias:**
- Use diverse datasets, apply augmentation techniques.

TEST SCENARIO

TEST SCENARIO

- **Image upload and File Handling:**
- Ensure the system handles large image files efficiently.
- Test drag-and-drop and manual file selection.
- **Results and Visualization :**
- Ensure the system correctly displays "Real" or "Fake".

TEST CASES

Test Cases

- **Test Case ID:TC001**
- Load and preprocess an image for deep fake detection
- **Test Case ID:TC002**
- Extract features from the input image
- **Test Case ID:TC003**
- Classify an image as real or fake

	Predicted Real (0)	Predicted Fake (1)
Actual Real (0)	7	3
Actual Fake (1)	4	6

Figure: confusion matrix

- Accuracy = (6 + 7) / (6 + 7 + 3 + 4) = 13 / 20 = 0.65 (65Precision = 6 / (6 + 3) = 6 / 9 = 0.67 (67Recall = 6 / (6 + 4) = 6 / 10 = 0.6 (60F1 Score = 2 * (0.67 * 0.6) / (0.67 + 0.6)=0.63(63

PROJECT PLAN CREATION

Project Goals And Objectives

Goals and Objectives

- The primary objective of this project is to develop a robust, real-time deepfake detection system that can accurately classify facial images as real or fake while maintaining scalability and security.
- Ensure the detection system performs accurately on various datasets by training on a wide range of real and deepfake images. This improves adaptability to different deepfake generation techniques and reduces bias in classification.

What problem does the project solve?

This project solves the deepfake detection problem — identifying whether an uploaded image is real or fake (AI-generated or manipulated), Why is this important?

- Misinformation: Deepfakes can spread false information, making it hard to trust what we see online.
- Cybersecurity: Fraudsters use deepfakes for identity theft, bypassing facial recognition systems.
- Legal concerns: Deepfakes are increasingly used in criminal activities, making detection crucial for law enforcement.

Expected Outcome

The expected outcome of this project is a fully functional deepfake detection website with the following features:

- Image Upload: A simple interface to upload any image.
- Prediction Result: Instant feedback showing whether the image is Real or Fake.
- Confidence Score: A percentage (like 87 Real or 92 Fake) to show how confident the model is about its prediction.
- Clear Visualization: Display the uploaded image along with the result.

Scope of work and Functionalities

Scope

- Image upload and preview.
- Real-time predictions (asynchronous requests).
- Confidence score display for each prediction.
- Error handling for invalid file types or size constraints

Features to be implemented

- PredictionResult: Displays the result and confidence score.
- ErrorMessage: Shows errors if any.
- LoadingSpinner: Indicates when the model is processing the image.

Technologies Used

- Frontend: React
- Backend: Flask, Python, TensorFlow/Keras
- Api integration done using Flask
- Model: CNN architecture

Timelines and Milestones

- Phase 1: frontend, basic user image uploading and binary prediction(real or fake) - Completed
- Phase 2: Additional information and score prediction on the predicted image, Deadline: 13/03/2025

Resource Allocation

Team members and roles

- Jaimy- Model integration with frontend using Flask Api
- Vrishti - Model Training on various data set
- Lakshmi - Fine tuning the model
- Sneha- Frontend

Tools and software required

- Flask — for model training, testing, and visualization
- VS Code — for writing and organizing React (frontend) and Flask (backend) code.
- Postman — for testing Flask API endpoints.
- Git — for version control.
- GitHub — for hosting the project's code repository.

Risk Analysis

- Model Inaccuracy: The CNN model might produce incorrect classifications due to overfitting or insufficient training data.
- Model Deployment Failures: The trained model may face errors during deployment due to compatibility issues between TensorFlow, Flask, and other libraries.
- Frontend Bugs: React components may break due to invalid API responses or unhandled user input.

Potential challenges and contingency plans

- Model Overfitting:
  Challenge: The CNN might perform well on training data but fail on new data. Contingency Plan: Use techniques like data augmentation, dropout layers and cross-validation.
- Malicious File Uploads:
  Challenge: Users might upload harmful files (e.g., scripts or corrupted data). Contingency Plan: Validate file types (allow only images: .jpg, .png). Add file size limits and scan uploads for security threats.
- The CNN might struggle to make accurate predictions, resulting in low accuracy scores:
  Optimize the model: Adjust hyperparameters like learning rate, batch size, and number of epochs.

OVERVIEW

OVERVIEW

Definition:

- Developed a deepfake detection system using a pre-trained CNN model, currently in the training phase to improve accuracy.
- Designed a user-friendly frontend with a landing page and image upload functionality, allowing users to submit images for analysis and receive predictions.

FEATURES DEVELOPED

FEATURES DEVELOPED

We have implemented features like:

- Deepfake Image Analysis – Uses a pre-trained CNN model to analyze uploaded images and determine if they are real or fake.
- User-Friendly Frontend – A clean and intuitive landing page with an image upload feature for easy interaction.
- Real-Time Prediction – Provides immediate results after processing the uploaded image.

CHALLENGES FACED

CHALLENGES FACED

- **Challenge: Model Accuracy and False Positives/Negatives**
- **Solution:** Fine-tuning the model with additional datasets and increased dataset diversity by including high-quality deepfake images from multiple sources.
- **Challenge: Frontend and Backend Integration**
- **Solution:** Built the backend server using Flask to process image uploads, run model inference, and return results to the frontend via an API.

ARCHITECTURE AND DESIGN

ARCHITECTURE AND DESIGN

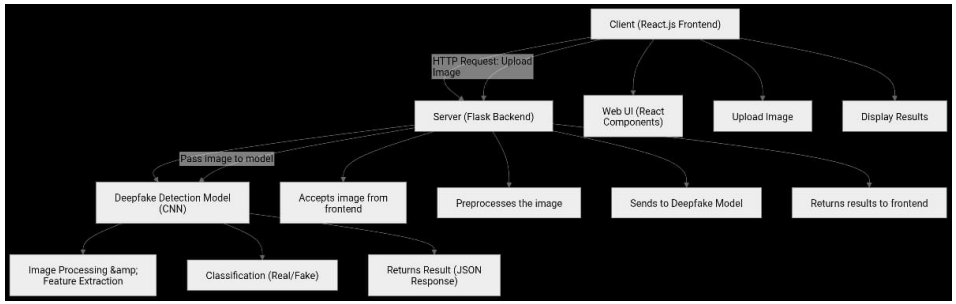

Figure: deployment diagram

IMPLEMENTATION DETAILS

IMPLEMENTATION DETAILS

Tech Stack Used

- **Frontend:**React.js
- **Backend:** Flask(Python)
- **Model:** CNN Model For Deepfake Detection
- **API:** Flask-based REST API for image processing and prediction

IMPLEMENTATION DETAILS

Code Structure and Key Modules

- **Frontend (Web Interface):** Handles user interactions, image uploads, and result display.
- **Backend (Flask Server):** Processes uploaded images, runs deepfake detection, and returns results.
- **API Endpoints:** Flask routes for handling image uploads and sending predictions
- **Utilities:** Functions for preprocessing images, handling errors, and logging system performance.

COMPLETED TASKS

COMPLETED TASKS

- Developed and trained(undergoing) a deepfake detection model using CNN.
- Built a Flask-based backend to handle image uploads and predictions.
- Designed a frontend with a landing page and image upload feature.
- Integrated backend and frontend for smooth data exchange.

PENDING TASKS

PENDING TASKS

- Improve model accuracy with additional training and fine-tuning.
- Implement a detailed analysis report for better understanding of the prediction.
- Implement error handling for invalid uploads.

Thank you!