IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 7, July 2025

DOI: 10.17148/IJIREEICE.2025.13713

Deepfake Detection System using ResNet50 (based lightweight for static images) – Deep Dect

Mrs. Vedhapriya P[1], Vyshali M[2], Yashoda N[3], V Lavanya[4], Kusuma J M[5]

Asst. Professor, Department of Computer Science and Design, Mysore University School of Engineering, Mysuru, India[1]

Student, Department of Computer Science and Design, Mysore University School of Engineering, Mysuru, India[2-5]

Abstract: The rise of synthetic media, commonly known as deepfakes, has created serious threats to digital authenticity. This paper presents a simple yet effective deepfake detection system for static facial images. It uses transfer learning with the ResNet50 CNN to distinguish between real and manipulated images by detecting subtle differences in facial features. A user-friendly interface was created using Streamlit and deployed through Hugging Face Spaces, allowing for real-time classification. While the system achieves moderate accuracy (about 61%), it shows the potential for accessible and deployable AI tools in digital forensics.

I. INTRODUCTION

Deepfakes are synthetic media created with machine learning techniques, and they pose serious risks in areas like fake news, financial fraud, and political propaganda. The term "deepfake" combines "deep learning" and "fake," highlighting the use of models such as CNNs and Generative Adversarial Networks (GANs) to produce realistic altered content. This project aims to meet the urgent need for automated detection systems since manual inspection is subjective and impractical for large-scale verification.

II. HISTORY OF DEEPFAKES

The idea of deepfakes appeared in late 2017 when a Reddit user posted videos made with deep learning algorithms that changed celebrity faces. Early methods used autoencoders, which often led to blurring or unnatural motion. In 2014, the arrival of GANs made a big difference in realism, making it possible to create synthetic videos that were almost indistinguishable by 2019.

III. ORIGIN OF DETECTION SYSTEMS

The rise of deepfake technology created a need for strong detection systems. Around 2018, researchers started to spot issues in fake content, like inconsistent facial shapes and lighting problems. Early heuristic methods were replaced by deep learning, especially CNNs, which were great at learning features from pixels. Datasets like FaceForensics++ and Celeb-DF helped train CNNs to tell real content apart from fake. This project uses a ResNet50-based CNN to classify static images.

IV. PROBLEM OVERVIEW

The main challenge being tackled in this project is building a reliable, user-friendly interface that detects whether a static image is a deepfake or a genuine photograph - accessed through a simple web interface. Manual detection is limited due to the advancement of the sophistication of deepfakes today.
The error rates from manual detection are no longer able to keep up with increasingly more sophisticated techniques.

V. OBJECTIVES

The project has three objectives:

1. **Create a Robust Classifier:** Build a deep learning model using ResNet50 architecture, tuned specifically with real and manipulated images in order to classify these images based on minute visual discrepancies.
2. **Create an Intuitive Front End:** Create an easy to use Streamlit web interface to allow image uploads and prediction in real-time; with the goal of developing a tool that people without a technical background can use.
3. **Deploy on a Public Platform:** Deploy the entire system to a public platform such as Hugging Face Spaces - allowing users to access the system without having to set up anything, or make any local installations.

Hardware and Software Requirements

Hardware Recommendations:

1. **Processor:** Intel Core i5 (or AMD Ryzen 5) or above
2. **Memory (RAM):** Minimum 8 GB (16 GB preferred)
3. **Storage:** Minimum 2 GB free disk space
4. **GPU (Optional):** NVIDIA GPU with CUDA support (e.g., GTX 1060 or better) for accelerated training

Software Requirements:

1. **Operating Systems:** Windows 10/11, Linux (Ubuntu 20.04+), macOS 10.15+
2. **Programming Language:** Python 3.8 to 3.10
3. **Key Python Libraries:**
   - TensorFlow 2.11.0: For model loading and execution
   - Streamlit 1.20.0: For front-end interface development
   - Pillow 9.4.0: For image loading and format conversion
   - NumPy 1.24.0: For numerical computation and image array manipulation
4. **Deployment Platform:** Hugging Face Spaces with Streamlit SDK

VI. SYSTEM DESIGN AND IMPLEMENTATION

The project's goal is to create a deepfake detection system that can accurately tell the difference between real and fake images. It will also provide immediate feedback through a user-friendly interface.

Proposed Solution

A three-layer solution was created:

1. Backend: A deep learning model based on the ResNet50 architecture.
2. Interface: A front-end built with Streamlit.
3. Deployment: Hosted on Hugging Face Spaces.

Dataset Details

The dataset came from Kaggle's "deepfake-and-real images" repository and is organized into Train, Validation, and Test folders. Each folder contains "Real" and "Fake" subfolders. The images were resized to 256x256 pixels and normalized to a range of [0,1]. Data augmentation was applied to the training set.

Mathematical Model and Algorithm

The model uses transfer learning with the ResNet50 architecture. It includes a Global Average Pooling layer, a Dense layer (1024 units, ReLU), Dropout (rate 0.5), and a Sigmoid output layer. The loss function is Binary Cross-Entropy, and the Adam optimizer is used with a learning rate of 0.0001 for optimization.

The algorithm involves:

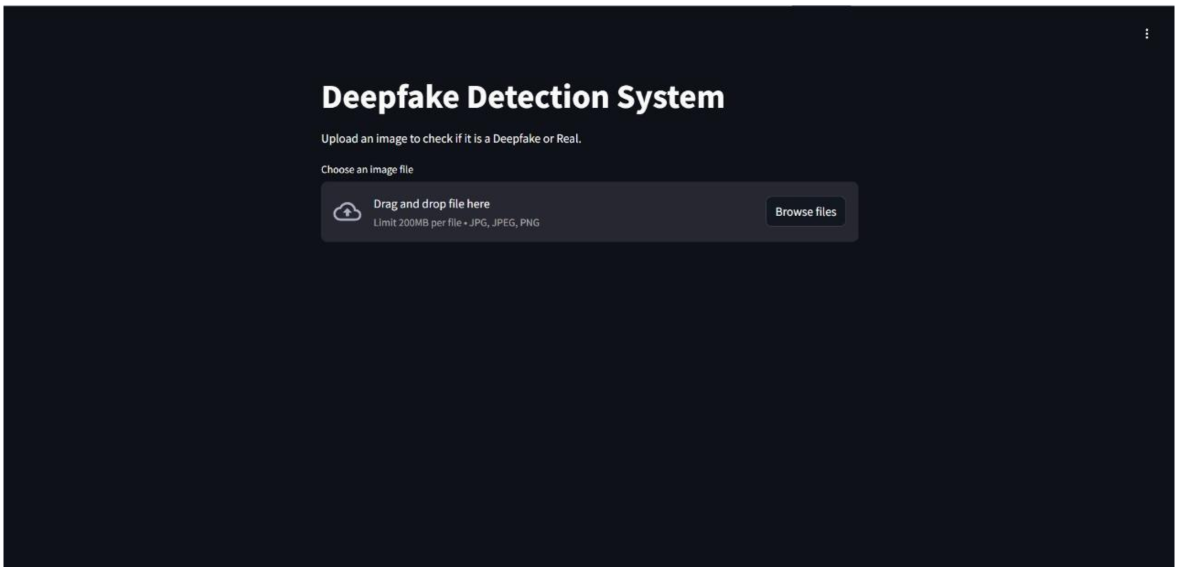
1. Loading the ResNet50 base model and freezing its layers.
2. Adding a custom classification head.
3. Compiling the model with binary_crossentropy loss and Adam optimizer.
4. Training and validating the model, then saving the best version.
5. Using the model within the Streamlit application.

User Interface Development

The Streamlit-based interface takes image input, preprocesses it, makes predictions using the ResNet50 model, and shows results with a confidence score. It is designed to be simple and doesn't require any installation.

Interface Algorithm

1. Load the trained model.
2. Launch the Streamlit application.
3. Accept image input through a file uploader.
4. Resize the image to 256x256 and normalize pixel values.
5. Pass the image to the model for prediction.
6. Display the predicted label (Real or Deepfake) and confidence score.


Deepfake Detection System

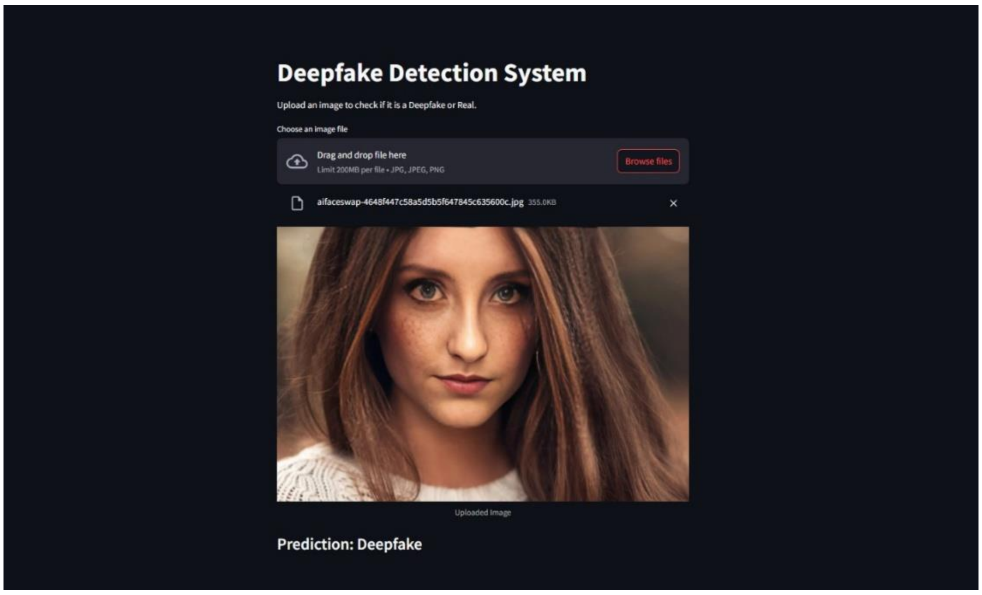
Upload an image to check if it is a Deepfake or Real.

Deployment and Public Accessibility

The system runs on Hugging Face Spaces. This allows for public access without needing installation or a special computing environment. The deployment package contains the trained model, Streamlit code, dependencies, and configuration files. The application loads the model automatically at startup and gives real-time predictions. Where users can access by link: https://share.google/6ukn8OL3SuZ7ahW4y

VII. RESULTS AND DISCUSSION

The model reached about 61.89% training accuracy and 60.84% validation accuracy. Inference time was less than 200 ms per image on CPU. The system successfully identified most real and fake images under ideal conditions. However, accuracy dropped with highly compressed or visually distorted images. The Streamlit integration made it easy for users to interact and evaluate images. The deployment worked well across different devices and browsers, providing quick response times and constant availability.


Deepfake Detection System

Upload an image to check if it is a Deepfake or Real.

aifaceswap-4648f447c58a5d5b5f647845c635600c.jpg 355.0KB

Prediction: Deepfake

VIII. CONCLUSION AND FUTURE WORK

The deepfake detection system uses a trained ResNet50 model along with a responsive Streamlit interface and deployment on Hugging Face Spaces. It effectively meets its main goal of detecting deepfakes through a web interface. Future improvements include:

1. Enhancing model accuracy through fine-tuning and training on more diverse datasets.
2. Adding real-time webcam input or video frame analysis.
3. Including Grad-CAM or other tools for explainable AI to help with visual interpretation.
4. Supporting batch image uploads for bulk verification.
5. Localizing the interface for multiple languages.
6. Logging predictions and feedback to improve continuously and monitor accuracy.
7. Integrating user authentication for personalized use.

REFERENCES

[1]. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: Learning to Detect Manipulated Facial Images.
[2]. Li, Y., Chang, M.-C., & Lyu, S. (2020). In Ictu Oculi: Exposing AI-Created Fake Videos by Detecting Eye Blinking.
[3]. Celeb-DF: A Large-scale Dataset for DeepFake Detection.
[4]. DeepFake Detection Challenge Dataset, provided by Facebook AI.
[5]. WildDeepfake: A Benchmark Dataset for DeepFake Detection in the Wild.
[6]. TensorFlow Documentation. https://www.tensorflow.org/
[7]. Keras Documentation. https://keras.io/
[8]. PyTorch Documentation. https://pytorch.org/
[9]. OpenCV Documentation. https://docs.opencv.org/
[10]. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.
[11]. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.
[12]. Emerging Trends in Deepfake Detection: A Survey. IEEE Access (2024).
[13]. Deep Learning in Forensics: New Techniques for Deepfake Analysis (2024).
[14]. Advanced CNN Architectures for Image Manipulation Detection (2024).