Project Architecture: Deepfake Image Detector

1. Data Pipeline

- Dataset Storage: Images are stored in directories:
 - Train/Real



- Train/Fake
- Validation/Real
- Validation/Fake
- Test/Real
- Test/Fake
- Data Augmentation (for better model generalization):
 - Train Dataset: Rescaled, rotated, shifted, sheared, zoomed, and flipped images.
 - Validation Dataset: Only rescaled images.
 - Test Dataset: Used for final evaluation.

2. Model Architecture

- Model Type: Sequential Convolutional Neural Network (CNN)
- Layers:
 - Conv2D + ReLU Activation (Extracts features)
 - MaxPooling2D (Reduces spatial dimensions)
 - Conv2D + ReLU Activation
 - MaxPooling2D
 - Flatten Layer (Converts 2D features into 1D vector)
 - Fully Connected Dense Layers
 - **Dropout Layer** (Reduces overfitting)
 - Output Layer (Sigmoid Activation): Binary classification (Real or Fake)

3. Training Pipeline

- Class Weighting: Handles dataset imbalance.
- Loss Function: Binary Cross-Entropy.

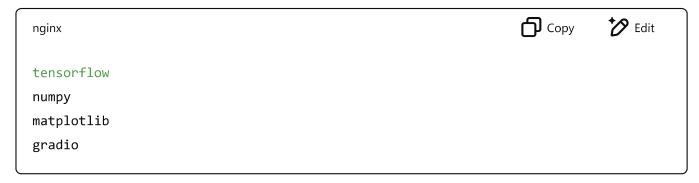
- Optimizer: Adam.
- Metrics: Accuracy.
- Training Process: Model learns to classify images as deepfake or real.

4. Deployment Pipeline

- Model Saving: deepfake_model.h5
- Gradio Web App (for user interaction):
 - Upload an image.
 - Resize and preprocess it.
 - Load the trained model.
 - Predict if the image is real or fake.
 - Display the result.

5. Hosting on Hugging Face Spaces

- Python App (app.py): Contains model loading, prediction, and Gradio UI.
- Requirements (requirements.txt):



6. Execution Flow

- 1. User uploads an image.
- 2. The image is resized and normalized.
- 3. The model makes a prediction.
- 4. The app displays the result with confidence.