Face Recognition System

📌 Objective:

To develop a user-friendly, real-time face recognition system using a deep learning model (ResNet18) integrated with a **Streamlit web interface** that supports:

- Image upload.
- Live webcam detection.

🛠 Tools & Technologies:

- **Python 3.9+**
- **PyTorch**
- **TorchVision**
- **OpenCV**
- **Streamlit**
- **PIL**
- **ResNet18**
- **Scikit-Learn (LabelEncoder with joblib)**

🔧 Model Details:

- Model used: **ResNet18 (pretrained=False)**
- Customization: The final layer of ResNet18 was adjusted to match the number of face classes by:

  model.fc = nn.Linear(model.fc.in_features, len(le.classes_))

- Trained using a custom dataset of face images.
- Labels encoded using **Scikit-Learn LabelEncoder**, saved as label_encoder.pkl.

🧪 Workflow:

1. **Image Upload:**
   - Users can upload .jpg, .jpeg, .png images.

    - The image is passed through the trained ResNet18 model.
    - The predicted face label is shown with confidence.
2. **Live Webcam Detection:**
    - Users can activate their webcam inside the Streamlit app.
    - The model continuously predicts the face in the webcam feed.
    - Results are overlaid live on the video feed.

⚙ Technical Highlights:

- Used **PyTorch's torch.device** for automatic GPU/CPU detection.
- Used **TorchVision transforms** for resizing and normalizing images.
- **Streamlit's file_uploader and button widgets** made the UI interactive.
- **OpenCV (cv2.VideoCapture)** was used for webcam live feed handling.

🔍 Results:

- The system successfully recognized faces both from uploaded images and live webcam.
- The UI was clean, user-friendly, and interactive.
- Predictions were fast and worked well even on CPU.
- Limitation: The system assumes a **clear, single face** in frame.

✅ Future Scope:

- Integrate **FaceNet embeddings for higher accuracy**.
- Add **face detection step before recognition**.
- Support for **multiple faces recognition in a single frame**.
- Logging of recognized faces with timestamps.
- Deployment to **Cloud/Local Network for remote access**.