Face Recognition using Triplet Loss & EfficientNetB3

■ Dataset

Dataset Used: Mini VGGFace2 (Resized Version)
CSV File: triplets_train.csv
Each row contains Anchor, Positive, Negative images.
Path Example:
C:\Users\Shahzaib\Desktop\ai document aqsa\archive (2)\mini_train_resized

■■ Preprocessing

1. Loaded all image paths from triplets_train.csv
2. Converted paths to local paths
3. Resized all images to 300x300x3
4. Normalized pixel values between 0 and 1
5. Generated 3 arrays: anchors, positives, negatives

■ Model Architecture

Model: EfficientNetB3 (Pretrained + Fine-tuned)
Framework: TensorFlow / Keras
Loss Function: Triplet Loss
Optimizer: Adam (lr=1e-4)
Embedding Dimension: 256

■ Model Training

Epochs: 15 | Batch Size: 32 | LR: 0.0001 | Validation Split: 20%
Training Log Example:
Epoch 1/15 - loss: 0.4012 - val_loss: 0.3000
Epoch 15/15 - loss: 0.3215 - val_loss: 0.2996
Model Saved: face_embedding_efficientnetb3_highacc.h5

■ Results

Model Comparison:
EfficientNetB3 - 91.84%
ResNet50 - 87.52%
MobileNetV2 - 81.13%
Best Model: EfficientNetB3

■ Verification Logic

Used cosine similarity between embeddings:
If sim(anchor, positive) > threshold → Same Person
If sim(anchor, negative) < threshold → Different Person

■ Technologies Used

Python 3.10, TensorFlow, Keras, NumPy, Pandas, scikit-learn, tqdm

■ Future Improvements

• Add FaceNet-style hard triplet mining
• Apply L2 normalization on embeddings
• Convert model to TensorFlow Lite
• Deploy using Streamlit / Flask

■■■ Author

Aqsa Abdul Rasheed
Business Analyst & AI Enthusiast
Pakistan

Model	Accuracy (%)
EfficientNetB3	91.84
ResNet50	87.52
MobileNetV2	81.13