Robust Face and Gender Recognition in Challenging Environments
Technical Summary
Team BYTEBash – Comsys Hackathon 2025

🧠 Summary

This Model is a dual-task deep learning system designed to perform accurate **Gender Classification** and **Face Recognition (Matching)** under challenging real-world conditions. The system combines **CNN-based classification**, **triplet-loss-based metric learning**, and a **Flask-powered frontend** to deliver real-time evaluation via a web portal.

🔹 Task A – Gender Classification

✅ Approach

- Binary classification problem: **Male** vs **Female**
- Data is structured using standard ImageFolder format under data/Task_A/{train,val}/
- Trained using BCE loss on gender labels

🧠 Architecture

- Backbone: **ResNet18** (pretrained on ImageNet)
- Final FC layer replaced with a 1-node Sigmoid output head
- Only gender head used (no multitask mode)

Input (224x224x3)

↓

ResNet18 Backbone

↓

Fully Connected Layer (1)

↓

Sigmoid → Binary Gender Output

🌟 Innovations

- Clean modular YAML-based config
- Robust evaluation pipeline showing Accuracy, Precision, Recall, F1-score
- Dynamic path override support for flexible test data

🔹 Task B – Face Recognition (Matching)

✅ Approach

- Face verification/matching using **Triplet Loss**
- Learning embeddings such that same identity faces are closer in latent space

- Matching is done via **Cosine Similarity**
- Evaluation is based on correct identity match and confidence score threshold

🧠 Architecture

- Backbone: **FaceNet (InceptionResNetV1)** from facenet-pytorch
- Embedding size: 512-D vector
- Matching Logic:
  - Reference embedding per identity from 1 clean image
  - Compare distorted test images using cosine similarity
  - Apply identity threshold (0.65) to validate match

    Test Image → EmbeddingModel (FaceNet) → 512D Embedding

    Reference → EmbeddingModel (FaceNet) → 512D Embedding

    ↓

    Cosine Similarity

    ↓

    Best Match + Threshold

    ↓

    Match (Label = 1 or 0)

🌟 Innovations

- Semi-hard triplet mining during training for better convergence
- Full pipeline for test-time matching + CSV logging
- Live frontend with progress bar using matcher_progress.txt
- Auto-generation of evaluation metrics (Top-1 Accuracy, Macro-F1)

🌐 Web Frontend

- Developed using **HTML + JavaScript + Flask backend**
- Accepts user-defined test path for Task A and B
- Dynamically runs respective evaluation scripts
- Live **loading spinner** and **progress bar (Task B)** with percent tracking
- Displays evaluation metrics instantly on page

✅ Highlights

- ✅ Modular architecture: training, evaluation, embedding, matching decoupled cleanly
- ✅ Dynamic data path input and validation
- ✅ Google Drive-based checkpoint loading
- ✅ Fully reproducible with detailed README & frontend