Multimodal Clothing Style Classifier

1. Project Overview

This project is a **hybrid machine learning system for clothing style classification**, combining both **Convolutional Neural Networks (CNNs)** and **Transformer-based models (such as CLIP or BERT)** to achieve highly accurate and context-aware predictions of outfit styles (e.g., casual, classic, sporty, formal, streetwear, etc.).

The system is deployed as an **interactive Streamlit web application**, allowing users to upload images of clothing or full outfits. The app then analyzes the image, extracts visual features, processes semantic associations using text-image encoders, and outputs the predicted style category.

The project is designed to run both **locally** and **in Google Colab**, making it flexible for development and testing. When running on Colab, the app is exposed publicly through **LocalTunnel**.

2. Key Features

- **Image Upload Interface:** Users can upload JPG/PNG outfit images directly from the UI.
- **CNN-Based Image Classification:** Extracts visual features from the image using a trained TensorFlow/Keras CNN model.
- **Transformer/CLIP Semantic Analysis:** Enhances classification accuracy using text embeddings and multimodal comparisons.
- **Hybrid Prediction Engine:** Fuses both CNN and semantic predictions to enhance performance.
- **Real-Time Output:** Predictions are generated in seconds and displayed in a clean UI.
- **Google Colab Support:** Works in environments without local Python installations.
- **Error Handling & Logging:** Ensures a smooth user experience even with invalid images.
- **Model Modularity:** Supports replacing or upgrading models without changing the UI.

3. System Architecture

The architecture consists of three major layers:

3.1 Frontend (Streamlit Application)

- Handles file upload
- Displays results, probabilities, and model insights
- Manages UI layout and user interaction

3.2 Backend Processing

A. CNN Model

- Trained TensorFlow model
- Extracts visual features (patterns, colors, shapes, garment structure)
- Outputs initial class probabilities

B. Transformer Model (CLIP or BERT)

- Generates semantic embeddings for clothing categories
- Compares embeddings with extracted image features
- Enhances classification accuracy when categories appear visually similar

C. Hybrid Fusion Engine

- Combines CNN prediction vector and semantic similarity scores
- Applies weighted averaging or rule-based fusion
- Outputs final class prediction

3.3 Deployment Layer

- **Local Machine**
- **Google Colab** using LocalTunnel to create a public URL

4. Technologies Used

Programming Languages

- Python

Frameworks & Libraries

- Streamlit (UI)
- TensorFlow/Keras (CNN model)
- PyTorch (optional, for CLIP)
- Transformers (HuggingFace)
- NumPy, Pandas
- OpenCV & PIL (image preprocessing)

Tools & Platforms

- Google Colab
- Visual Studio Code
- LocalTunnel (app hosting)

5. Installation & Setup

Requirements

- Python 3.10+ recommended
- Pip package installer
- Node.js (for LocalTunnel)

6. Usage Instructions

1. Launch the Streamlit app.
2. Upload an image of an outfit.
3. The system preprocesses the image (resizing, normalization).
4. CNN model predicts the initial category.
5. Transformer model aligns predictions with semantic meaning.
6. Hybrid model outputs the final prediction.
7. Results and probability scores are displayed.

7. Dataset & Models

7.1 Dataset Description

The dataset used for training may include:

- Public fashion datasets (DeepFashion, FashionMNIST, custom labeled datasets)
- Manually curated images for specific style categories

7.2 Preprocessing Pipeline

- Normalize pixels to [0, 1]
- Augment (optional): rotation, flipping, brightness adjustment

7.3 CNN Model Architecture

Common architectures used:

- MobileNetV2
- EfficientNet
- Custom CNN with Conv2D, MaxPooling, Dense layers

7.4 Transformer / CLIP Model

- Converts style category labels to vector embeddings
- Aligns text and image vectors for improved classification

7.5 Hybrid Model Fusion

Fusion strategies:

- Weighted sum of CNN and CLIP outputs
- Rule-based decision layer
- Softmax normalization across combined scores

8. Streamlit UI Structure

Main UI Components

- Header & project description
- File uploader widget
- Image preview section
- Prediction output with confidence bars
- Expanded section for model explanations

Optional Sections

- Model info
- Processing logs
- Downloadable results

9. Testing

Manual Testing

- Upload images of different clothing styles
- Check consistency of predictions
- Test failure cases (low-light images, partial images)

Automated Testing

- Evaluate CNN model accuracy on validation set
- Evaluate CLIP semantic alignment
- Measure hybrid model improvements

11. Results

- CNN model accuracy typically above 96**%**, depending on dataset size
- Hybrid model improves accuracy especially for overlapping categories
- sample outputs:
    - Input image
    - Top 3 predicted styles
    - Confidence scores

12. Future Improvements

- Add recommendation engine (suggest outfits)
- Train with larger datasets for more categories
- Add color/style detection
- Support video input (outfit analysis from camera)
- Publish as mobile-friendly web app

13. TEAM MEMBERS

• Sohaib Mahmoud Mashaly
•Abanoub Ayman Nakhla
•Ali Ahmed Adel El-Gamal
•Samir Nabil Samir
•Sabry Wael Sabry
•Mohamed Ahmed Adel