

PROJECT REPORT

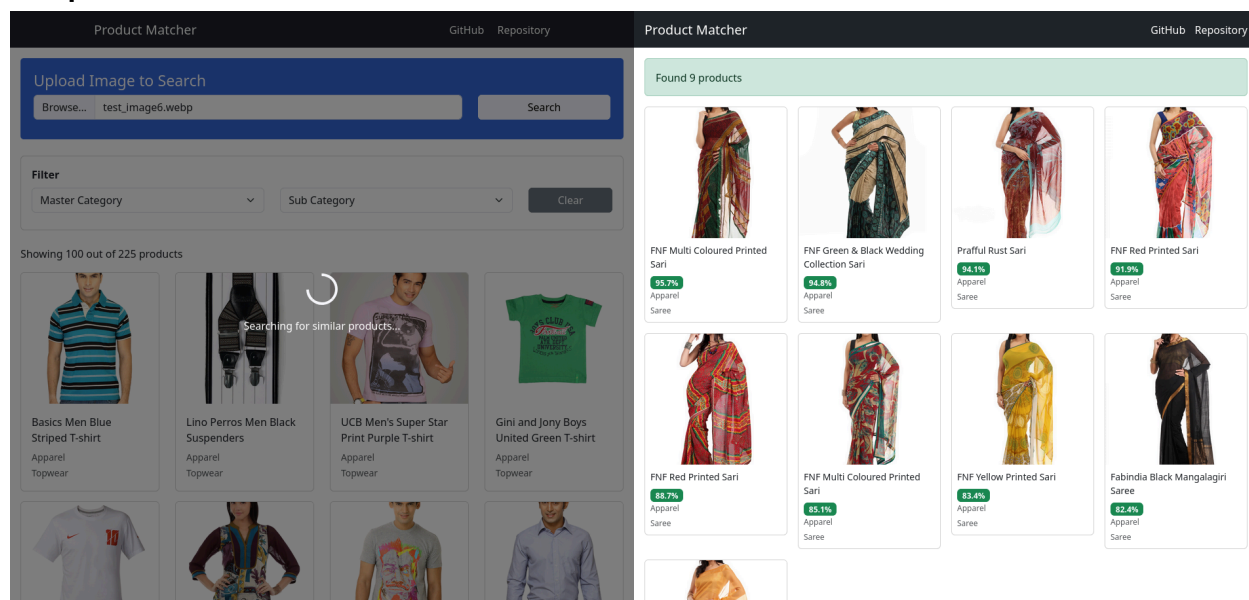
Submission by: ANIK HALDER 24MCA0251

[Visual Product Matcher Build](#) is a web-based fashion product catalogue and search application built with Flask, FastAPI, Bootstrap 5 and MongoDB Atlas. The application employs a custom built, fine-tuned and hosted hybrid machine learning architecture combining ResNet50 and OpenAI-CLIP models, outperforming traditional standalone CLIP implementations by more than 10% with 90% visual similarity and 95% semantic accuracy.

Link to deployed Application : <https://visual-product-matcher-1hkn.onrender.com>

Link to Github Repository: https://github.com/Anikrage/Visual_Product_Matcher

Sample Screenshot:



Model Architecture:

Unlike conventional single-model approaches, the custom built model uses:

1. Three-stage ranking algorithm that first identifies the visual matches using ResNet50 and CLIP image embeddings using cosine similarity.
2. Followed by a semantic filtering using CLIP text embeddings to eliminate gender/category confusions.
3. and then finally combining the scores to create a hybrid score and ranking the results.

This approach addresses the common failures in simpler systems that often mismatch men's and women's products.

Full Stack Implementation:

The frontend uses Flask templating with Bootstrap 5 and vanilla javascript for responsive UI, while the backend FastAPI service handles the ML inference. The database is hosted online with MongoDB Atlas that stores product metadata, pre-computed embeddings, links to product image hosted via fast CDN using Cloudinary, with indexed queries for fast filtering.

Optimizations:

The application follows a separation of concern approach, Flask for presentation, FastAPI for ML inference and MongoDB for persistence, enabling independent scaling. Further optimizations are done via :

1. pre-computed embeddings eliminating real-time inference
2. Numpy vectorized similarity calculations
3. heap-based selection for $O(n \log k)$ complexity
4. lazy loading with async pagination
5. CDN-cached resources

Cloud Deployment:

1. The custom built model is hosted via HuggingFace Spaces that provides an API for connectivity and inference
2. The frontend and backend is hosted via Render with automatic configuration for continuous integration and deployment.

This approach results in a production system that demonstrates complete software engineering from data preprocessing and model training to API deployment,frontend integration and cloud deployment, achieving 1-2 second response times.