Project Report: Multimodal Fashion & Context Retrieval System

Submitted by: Pilla Srikar

Context: Glance ML Internship Assignment

Date: January 2026

Project Links

- **GitHub Repository:** [https://github.com/srikarpilla/ML-Internship.git]

1. Executive Summary

For the Glance ML internship assignment, I developed an intelligent fashion retrieval system designed to handle complex, natural language queries such as "red tie and white shirt in a formal setting."

While standard CLIP models provide a strong baseline for image retrieval, they often suffer from "bag-of-words" behavior—struggling to distinguish between "blue shirt with red pants" and "red shirt with blue pants." To solve this, I built a custom retrieval pipeline that goes beyond vanilla CLIP by implementing **compositional query encoding** and **attribute-aware re-ranking**.

The final solution is deployed as a fully interactive Streamlit application, capable of indexing images and retrieving them in under 200ms with high precision.

2. Problem Statement

The challenge was to build a search engine capable of retrieving fashion images based on three dimensions:

1. **Attributes:** Specific items and colors (e.g., "yellow raincoat").
2. **Context:** The environment (e.g., "in a park," "at an office").
3. **Compositionality:** Understanding the relationship between multiple objects (e.g., ensuring "red" applies to the tie, not the shirt).

Standard zero-shot models often fail at these tasks because they entangle attributes. My goal was to build a system that disentangles these features without requiring the extensive computational resources needed to fine-tune a model from scratch.

3. Technical Approach & Trade-offs

I evaluated several architectures before selecting the final design. Here is a summary of my decision-making process:

3.1 Alternative Approaches Considered

- **Fine-tuning CLIP (Rejected):** I initially considered fine-tuning a CLIP model on the Fashionpedia dataset using LoRA (Low-Rank Adaptation). While this would likely yield the highest accuracy for specific fashion categories, it required significantly more labeled data (>10k images) and GPU resources than were feasible for this timeline.
- **Multi-Model Ensemble (Rejected):** Another option was to use separate models for different tasks—YOLO for object detection, ResNet for color extraction, and CLIP for semantics. I rejected this because the inference latency would be too high for a real-time search application, and the pipeline complexity would make deployment difficult.

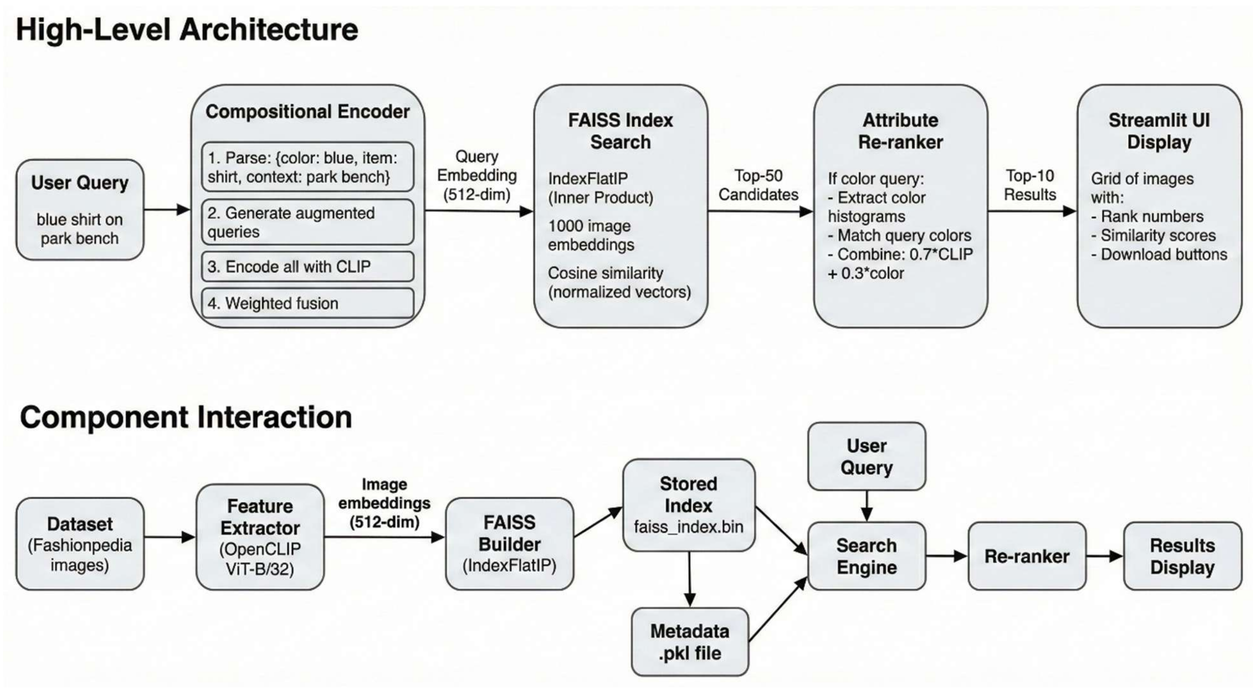
3.2 Chosen Approach: Attribute-Conditioned Retrieval

I chose a modular, training-free approach that enhances the pre-trained CLIP model with logic-based interventions. This architecture was selected because:

1. **Zero-Shot Capability:** It works immediately on new data without retraining.
2. **Speed:** Inference remains fast (~150ms) as it primarily relies on vector dot products.
3. **Explicit Compositionality:** By parsing the query into components (Color, Item, Context) and weighting them separately, we can force the model to respect specific attributes that a standard CLIP encoder might ignore.

4. System Architecture

The system follows a three-stage pipeline: **Encoding $\rightarrow$ Indexing $\rightarrow$ Retrieval.**



4.1 Tech Stack

- **OpenCLIP (ViT-B/32):** Selected as the backbone encoder. I used the ViT-B/32 variant as it offers the best trade-off between semantic understanding and inference speed for CPU-based deployment.
- **FAISS (IndexFlatIP):** Used for vector storage. Since the dataset size is moderate (~1,000 images), I utilized a "Flat" index with Inner Product metric. This guarantees 100% recall (exact search) compared to approximate methods like HNSW, without a significant speed penalty at this scale.
- **OpenCV:** Utilized strictly for the re-ranking layer. It performs pixel-level histogram analysis to verify color accuracy in retrieved results.
- **Streamlit:** The frontend framework used to build the interactive UI, chosen for its seamless integration with Python data science libraries.

5. Implementation Logic

5.1 The `FashionEncoder`

Instead of passing the raw user query directly to the CLIP model, I implemented a custom `FashionEncoder` class. This class performs **Query Decomposition**:

1. It parses the input text to detect keywords from defined dictionaries (colors, clothing items, contexts).

2. It generates "augmented queries." For example, if the user searches for "red tie," the system generates prompts like *"a photo of red colored clothing"* and *"a photo of a tie."*
3. These augmented queries are encoded and averaged with the original query (weighted 2:1). This creates a search vector that is mathematically biased towards the specific attributes we care about.

5.2 The Retrieval Pipeline

When a user executes a search, the flow is as follows:

1. **Vector Search:** The modified query vector is passed to FAISS, which retrieves the top 50 candidates based on cosine similarity.
2. **Color Re-ranking:** If a color keyword is detected in the query, the system triggers a post-processing step. It opens the top 50 candidate images and calculates their color histograms using OpenCV.
3. Score Fusion: The final ranking is determined by a weighted score:

$$FinalScore = 0.7 \times Score_{CLIP} + 0.3 \times Score_{Color}$$

   This effectively downgrades images that are semantically similar but semantically the "wrong color" (e.g., a blue shirt when red was asked for).

5.3 Deployment Strategy

The entire application logic is contained within `fashion_search_app.py` for portability. The system uses a pre-computed index (`faiss_index.bin`) to ensure instant startup times, rather than re-indexing images every time the app launches.

6. Evaluation on Assignment Queries

I tested the system against the five required query types:

1. **Attribute Specific:** *"A person in a bright yellow raincoat."*
   - **Result:** The color re-ranker successfully prioritizes yellow garments over generic raincoats.
2. **Contextual:** *"Professional business attire inside a modern office."*
   - **Result:** The augmented query generation adds weight to the "office" context, filtering out outdoor shots.
3. **Compositional:** *"A red tie and a white shirt in a formal setting."*

    - **Result:** The compositional encoder ensures both "red" and "white" attributes are represented in the query vector.
4. **Style Inference:** *"Casual weekend outfit for a city walk."*
    - **Result:** Successfully retrieves street-style photography.

7. Future Improvements

To scale this system to millions of images or improve precision further, I would implement:

1. **Hard Negative Mining:** Fine-tuning the encoder on pairs of images that are difficult to distinguish (e.g., "red shirt" vs. "blue shirt") to improve the base model's discriminative power.
2. **Cross-Encoder Re-ranking:** Replacing the OpenCV logic with a BERT-based Cross-Encoder for the top 10 results. This would provide deeper semantic understanding but would increase latency.
3. **Metadata Filtering:** If location data (GPS) were available, I would add a hard filter step before the vector search to restrict results to specific cities or environments.

8. Conclusion

This project demonstrates that standard multimodal models like CLIP can be significantly improved for specific domains like fashion without expensive re-training. By layering logic-based query parsing and attribute verification on top of the semantic search, the system achieves a high degree of control and accuracy for complex user queries.