Multimodal Product Retrieval System Using Contrastive Learning and Residual Adapters

Section D, Team 06

Doan Le; Jiongyang Song; Xin'ai Hao; Xinyao Wang; Yanze Liu

December 7, 2025

Abstract

This project details the development of a multimodal text-to-image retrieval system designed for e-commerce. Leveraging the pre-trained CLIP (Contrastive Language-Image Pre-Training) architecture, we processed a dataset of approximately 20,000 Amazon product listings. We followed an iterative experimental approach: initially training a naive Two-Tower MLP (Experiment A), comparing it against a zero-shot baseline (CLIP Raw), and finally developing a Residual Projection Head adaptation (Stage A). Our investigations revealed that naive fine-tuning caused catastrophic misalignment, where Recall@10 dropped to 36.3%, whereas the un-tuned baseline achieved 58.7%. The final "Stage A" model, employing residual connections and InfoNCE loss, resolved these issues, achieving a Recall@10 of 69.7%—an 11% absolute improvement over the baseline. Furthermore, we discuss the deployment strategy, identifying operational risks and ethical considerations regarding bias in foundation models.

1 Motivation and Data Understanding

In the rapidly evolving landscape of e-commerce, a significant disconnect exists between user search queries (e.g., ”cute small orange plushie”) and keyword-based indexing (e.g., ”Toy, Polyester, 5-inch”). Traditional retrieval systems rely on lexical overlap and often fail to capture the visual essence of a user’s intent. Vision-Language Models (VLMs) offer a solution by bridging the modality gap. This project aims to build a specialized retrieval system aligning textual descriptions with product images in a shared latent space. While the primary problem is technical—adapting pre-trained models without ”catastrophic forgetting”—the business relevance is profound, impacting user engagement and conversion rates.

To support this, we utilized a dataset of Amazon product listings. The data sources include unstructured text fields (titles, descriptions) and image URLs. Understanding this data revealed inherent noise: broken links, keyword stuffing, and weak correlation between titles and visual appearance, necessitating a deep learning approach capable of extracting robust semantic features.

2 Data Preparation

Integrating noisy real-world data into a format suitable for deep learning required a rigorous pipeline.

2.1 Data Integration and Cleaning

To ensure robust model training, we implemented a highly efficient image acquisition pipeline using a multi-threaded `ThreadPoolExecutor`. This system was designed to handle large-scale concurrent requests, achieving a throughput of approximately 40 images per second. Crucially, we integrated a strict filtering mechanism within this pipeline. Any image downloaded that was smaller than 200 bytes or triggered a file corruption error during loading was automatically

discarded. This preprocessing step was vital to prevent empty or corrupted tensors from destabilizing the training loop.

2.2 Formatting for Deep Learning

Product titles alone often lack sufficient descriptive power. To address this, we engineered a composite text feature by merging the `title`, `features`, `description`, `categories`, and `details` into a single "combined text" field. This concatenation strategy ensures that the text encoder receives a dense semantic signal that includes material details, color specifications, and usage contexts, which are essential for aligning with fine-grained visual features.

2.3 Feature Caching Strategy

Training deep learning models on raw images is computationally expensive. To optimize our resource usage, we adopted a "freeze-and-cache" strategy. We passed all 19998 valid image-text pairs through the pre-trained `openai/clip-vit-base-patch32` model exactly once. The resulting 512-dimensional embeddings were serialized and saved to disk. This approach effectively decoupled the heavy feature extraction process from the lightweight training of the projection heads, reducing the time required for each training epoch from hours to mere minutes.

3 Modeling

We adopted a systematic, iterative modeling approach. Our process began with a hypothesis that domain-specific projections were necessary (Experiment A), followed by a diagnostic phase using the raw model (Baseline), and concluded with a hybrid architecture (Experiment B) that synthesized the strengths of both approaches.

3.1 Experiment A: Naive Two-Tower MLP

Our initial approach was driven by the assumption that the generic features from CLIP needed to be completely re-mapped to fit the Amazon product distribution. We constructed a ”Two-Tower” architecture consisting of two symmetrical Multi-Layer Perceptrons (MLPs). Each tower transformed the 512-dimensional CLIP input through a non-linear sequence: Linear ($512 \rightarrow 512$), ReLU, Dropout ($p = 0.2$), and a final Linear layer ($512 \rightarrow 256$). We trained this model using Margin Ranking Loss.

Surprisingly, this model performed poorly. By treating the CLIP embeddings merely as raw numerical inputs and projecting them through random initializations, the model effectively ”ignored” the rich semantic alignment already present in the pre-trained image vectors. The non-linear transformation destroyed the pre-existing geometry where ”image of a bear” and ”text of a bear” were already close. Instead of adapting, the model had to learn visual-semantic alignment from scratch using only our small dataset, leading to severe overfitting and catastrophic forgetting.

3.2 Baseline Model: Zero-Shot CLIP

The failure of Experiment A prompted us to pivot and evaluate the raw, unmodified weights of the CLIP model (`vit-base-patch32`) to understand the intrinsic quality of the image vectors. We computed the dot product between the normalized image and text embeddings directly from our cached features.

The raw baseline significantly outperformed our trained MLP (Experiment A). This was a crucial finding as it proved that the original image information—specifically the pre-trained alignment between visual features and textual concepts—was highly valuable and robust. It demonstrated that any successful model must *preserve* this foundational visual-semantic structure rather than discarding it in favor of a new, learned projection. This realization became the cornerstone for our final

architecture.

3.3 Experiment B: Residual Projection Head

Incorporating the lessons from the previous steps, we designed ”Experiment B” (Stage A) to balance two conflicting needs: the need to respect the strong pre-trained image alignment proven by the Baseline, and the need to adapt to e-commerce nuances.

We implemented a Residual Projection Head defined by $y = x + \alpha \cdot f(x)$. Here, x is the original CLIP embedding carrying the crucial image information. The term $\alpha \cdot f(x)$ represents a small, learnable correction, where $\alpha = 0.2$ scales the contribution of the adapter network. This architecture ensures that the model respects the image information by default (since $y \approx x$ at initialization), preventing the information loss seen in Experiment A. Simultaneously, it allows the model to fine-tune the space for domain-specific vocabulary via the InfoNCE loss function, effectively synthesizing the generalization of the baseline with the specificity of a trained model.

4 Implementation Challenges and Solutions

Implementing these models presented specific technical hurdles. A major challenge was the instability of the Margin Ranking Loss in Experiment A, which led to slow convergence and poor generalization. Our solution was the transition to InfoNCE loss in Stage A, which stabilized training by leveraging the entire batch for negative sampling. Additionally, to handle the custom architecture of the Residual Head efficiently, we avoided high-level trainer APIs and instead built the training loop from scratch using PyTorch. This allowed us to implement the specific caching mechanism and temperature scaling logic required for our ”Stage A” model.

5 Experimental Setup

The training infrastructure was built upon a single NVIDIA GPU to ensure consistent computation. The optimization process was managed by the AdamW optimizer, with weight decay set to $1e^{-2}$ to prevent overfitting. We carefully tuned the learning rates, assigning $1e^{-4}$ for the projection heads and a higher $1e^{-3}$ for the logit scale parameter to allow for rapid adaptation of the temperature scaling. These rates were modulated by a Cosine Annealing scheduler over a course of 10 epochs. To further enhance training throughput and minimize memory usage, we employed Mixed Precision (AMP) training.

6 Results and Evaluation

To comprehensively assess the effectiveness of our proposed solution, we conducted a multi-faceted evaluation. This involved a rigorous quantitative comparison using standard Information Retrieval metrics, followed by a detailed qualitative analysis of visual retrieval results to diagnose model behavior. Finally, we translated these technical findings into a concrete business case.

6.1 Quantitative Performance Metrics

First, we benchmarked the retrieval accuracy of our three experimental configurations. Using Recall@K (R@1, R@5, R@10) and Mean Reciprocal Rank (MRR), we measured the probability of the correct product appearing in the top K results. The table below illustrates the progression from our failed initial attempt (Exp A) to the final successful model (Stage A).

Table 1: Comparative Performance Metrics

Model	R@1	R@5	R@10	MRR
MyCustomMatcher (Exp A)	0.100	0.262	0.363	0.186
CLIP Raw (Baseline)	0.273	0.490	0.587	0.378
Stage A (Exp B)	**0.289**	**0.574**	**0.697**	**0.420**

The data quantitatively confirms the superiority of the Residual Adapter approach. While naive fine-tuning (Exp A) degraded performance significantly, "Stage A" improved R@10 by 11% (58.7% to 69.7%) over the robust zero-shot baseline. This indicates that the residual connection successfully allowed the model to learn domain-specific features without losing general semantic knowledge.

6.2 Visual Retrieval Analysis

To gain deeper insight into the models' semantic understanding, we visualized the top retrieval results for the challenging query: *"a small orange plush toy with big eyes"*. The figure below displays the top-5 images returned by each model, highlighting the differences in how each architecture interprets visual attributes like color, texture, and shape.

Figure 1: Top-5 retrieval results for the query ”a small orange plush toy with big eyes” across three models: CLIP Raw, My_Model (MLP), and Stage A (Residual).

6.3 Detailed Error Analysis and Model Behavior

Based on the visual results shown in Figure 1 and the corresponding query, we performed a granular analysis of each model’s failure modes and successes.

CLIP Raw

CLIP Raw successfully captures the high-level concept of ”cute, plush toys” but struggles with specific attributes like ”orange” and ”big eyes.” This occurs because the model prioritizes global image-text alignment over fine-grained visual details. Consequently, items lacking these specific features still rank highly. A lightweight post-filtering step focusing on color or key attributes could align retrieval more closely with user intent without modifying the core model.

My_Model (Naive MLP)

This model performs poorly, often overlooking ”big eyes” and latching onto simpler features like ”orange” or ”plush,” sometimes returning unrelated items. This failure stems from the training strategy: using only randomly sampled negatives failed to provide signals distinguishing visually similar items. To fix this, hard negative mining—contrasting specific attributes like eye size—is required to force the model to learn fine-grained distinctions.

Stage A (Residual Adapter)

Stage A performs best, consistently placing accurate matches (orange plush with big eyes) at the top. However, lower-ranked results sometimes drift toward generic plush toys as strong general traits overshadow fine attributes in the final scoring. Since the top results are excellent, a simple re-ranking step for lower candidates could further stabilize the full retrieval list.

6.4 Developing a Business Case

Our technical improvements directly translate to business value by bridging the ”vocabulary gap” between abstract user intent (e.g., ”vibes”) and technical product metadata. By improving Recall@10 by 11% for complex queries, the Stage A model effectively unlocks long-tail inventory that lacks precise keyword tags. This enhancement reduces the frequency of ”zero results” pages and increases the visibility of niche products, ultimately driving higher Gross Merchandise Value (GMV).

7 Deployment and Ethical Considerations

7.1 Deployment Strategy

We propose a scalable microservice architecture for deployment. Offline, the trained Image Encoder vectorizes the catalog into a high-performance vector database (e.g., Milvus). Online, user queries are processed by the Text Encoder and Residual Head to retrieve top matches via Approximate Nearest Neighbor (ANN) search, ensuring low-latency responses.

7.2 Risks and Ethical Considerations

Operational risks include data drift due to evolving trends, which we mitigate via a continuous learning pipeline that periodically re-trains the lightweight adapters. Ethically, foundation models may harbor societal biases; we address this through rigorous fairness audits on sensitive queries and human-in-the-loop reviews to prevent reinforcing harmful stereotypes.

8 Conclusion

This project illustrates an iterative engineering process. We started by training a standard MLP, discovered it caused catastrophic forgetting, and pivoted to a Residual Adapter architecture. This solution successfully refined the pre-trained features, achieving superior retrieval performance (69.7% Recall@10). The final system offers a scalable, cost-effective solution for modern e-commerce search, provided that deployment risks regarding data drift and bias are carefully managed.

Appendix: Project Contributions

The following table details the specific contributions of each team member to this project.

Table 2: Team Member Contributions

Team Member	**Primary Contributions**
Doan Le	Conducted market analysis on the ”vocabulary gap”; Defined the business motivation and potential impact on Gross Merchandise Value (GMV).
Jiongyang Song	Designed and trained both Experiment A (MLP) and Stage A (Residual) models; Implemented InfoNCE loss; Developed evaluation metrics and visualization scripts.
Xin’ai Hao	Analyzed operational costs and proposed the cost-effective model maintenance strategy; Evaluated infrastructure requirements for deployment.
Xinyao Wang	Developed the real-time deployment strategy; Identified operational risks (data drift) and defined the ethical auditing framework for bias mitigation.
Yanze Liu	Architected the data ingestion pipeline; Implemented the ”freeze-and-cache” feature extraction workflow; Engineered composite text features.