

Rank-Supervised SKU Fingerprints for Low-Latency Generative Visual Search

10-423/623/723 Generative AI – Final Report

Patrick Chen (MSCV, bochunc@andrew.cmu.edu)

Mingyang Yu (MSME, myu3@andrew.cmu.edu)

Kanlong Ye (MSME, kanlongy@andrew.cmu.edu)

Abstract

We study Stock Keeping Unit (SKU)-level visual search for fashion e-commerce under strict latency and memory constraints. Unlike systems that index multiple vectors per product, we target a *SKU-per-vector* design: each SKU is represented by a single embedding shared by image and text queries. Using DeepFashion2, we first implement strong baselines: a ResNet-50 re-identification model and CLIP-based SKU retrievers with mean-pooled catalog embeddings. We then propose a *rank-supervised SKU fingerprint model* that distills one compact vector per SKU from multiple catalog and generative multi-view images using a Q-Former-style aggregation network supervised directly by search metrics. Our final fingerprint model improves image Recall@1 from 0.587 to 0.865 on seen SKUs, and from 0.587 to 0.602 on all validation SKUs (seen + zero-shot), with similar gains on text queries, while achieving significantly lower CPU latency than both the re-id and CLIP baselines. Offline Stable Diffusion v1.5 multi-view augmentation yields additional gains, and the distilled fingerprints generalize to zero-shot SKUs not seen during training.

1 Introduction

Fashion e-commerce must answer image or text queries with the exact purchasable product under tight latency budgets: a user uploads a noisy phone photo or types a short description (e.g., “a grey trousers from the front view”), and expects the system to return the matching catalog SKU (pair_id, style, category_id) quickly and reliably.

We study a production-friendly **SKU-per-vector** regime where each SKU is mapped to a single embedding shared across image and text queries. This keeps index size and similarity search cost proportional to the number of SKUs, but forces each embedding to be robust to viewpoint, background, lighting, and partial views in user photos. Our approach builds on CLIP-style vision–language pretraining and generative augmentation: we establish strong re-identification and CLIP baselines, then introduce a rank-supervised SKU fingerprint that aggregates multiple catalog and generative views using a Q-Former-style module trained directly on search metrics (Figure 1).

Our main contributions are:

- A search-metric-supervised SKU fingerprint model that aggregates multiple catalog and generative views into a single vector per SKU via best-shot distillation and a lightweight Q-Former-style aggregator.
- An offline Stable Diffusion v1.5 img2img + LoRA pipeline that produces identity-preserving multi-view catalog images without affecting online latency.

- A systematic evaluation of SKU-per-vector retrieval on DeepFashion2 covering image and text queries, seen vs. zero-shot SKUs, and end-to-end CPU latency.

2 Dataset, Task, and Metrics

2.1 Dataset: DeepFashion2

We use DeepFashion2 (DF2) [2] as our only dataset. Each image has image-level labels `source` (shop or user) and `pair_id`; each clothing item has item-level labels `style`, `category_id`, and a bounding box. Here, `pair_id` groups a user photo and its corresponding shop images of the same garment, `style` is a fine-grained outfit/style identifier where `style > 0` marks valid clothing items, and `category_id` is a coarse garment category (e.g., dress, top, trousers) that we treat as the catalog category.

We define a visual SKU as

$$\text{SKU} = (\text{pair_id}, \text{style} > 0, \text{category_id}). \quad (1)$$

We crop clothing items using their bounding boxes and discard items with `style=0`. For each SKU, crops from images with `source = shop` are *catalog* images, and crops from images with `source = user` are *query* images. We follow the original DeepFashion2 train/validation/test split at the SKU level, ensuring that SKUs do not leak across splits, yielding 21,190 SKUs and 198,140 images (catalog + user) in total.

2.2 Tasks

We study two retrieval tasks: **Image-to-SKU retrieval** and **Text-to-SKU retrieval**. Image-to-SKU retrieval is given a cropped user image, retrieve the matching catalog SKU (same (`pair_id`, `style > 0`, `category_id`)). On the other hand, Text-to-SKU is given a short text description, retrieve the matching catalog SKU. DeepFashion2 does not provide text descriptions, so we synthesize prompts from metadata (category name, domain, occlusion, viewpoint, and optional style/SKU tags) using simple templates; examples and generation details are in Appendix D.

2.3 Metrics

All metrics are computed at the SKU level by ranking SKU fingerprints according to cosine similarity with the query embedding. We report Recall@K, mean reciprocal rank (MRR), normalized discounted cumulative gain at 10 (NDCG@10), and the 95th-percentile and mean query latency on CPU. Exact formulas appear in Appendix A.

3 Related Work

DeepFashion and DeepFashion2 [2] define the consumer-to-shop retrieval setting we adopt. CLIP [1] provides the joint image–text embedding space we build upon, and product-centric pretraining like Product1M/CAPTURE [5] highlights the importance of product- or SKU-level representations. FashionBERT [6] and FashionIQ [3] explore multimodal fashion retrieval with natural-language descriptions, motivating our text-to-SKU evaluation. On the generative side, Diffusion Transformers (DiT) [4] and Stable Diffusion inspire our use of SD v1.5 + LoRA for multi-view catalog augmentation. Finally, person re-identification work [7, 8] provides strong CNN baselines with classification+metric losses; we replicate this as an image-level SKU re-id baseline.

4 Methods

4.1 End-to-End Pipeline

Figure 1 summarizes our training and inference pipeline. In the **training stage** (top), we first fine-tune a ViT-B/32 CLIP encoder on DF2 SKU labels (red), obtaining strong image/text encoders and best-shot teacher scores. We then LoRA-tune Stable Diffusion v1.5 img2img on catalog crops (magenta) to generate identity-preserving multi-view catalog images. Catalog and generative views are encoded by the fine-tuned CLIP image tower and passed to a Q-Former-style aggregator g_θ (orange), which outputs one fingerprint per SKU and is trained with regression, rank distillation, and SKU classification losses. In parallel, we train a small Visual-Language-Action (VLA) module (green) that learns to select an image-processing action (e.g., smart crop, denoise, sharpen) that can optionally be applied to user photos before CLIP encoding.

In the **inference stage** (bottom), a visual search engine takes an image or text query, optionally enhances user images via the VLA (see details in Appendix E.), encodes the query with the fine-tuned CLIP towers, computes dot products against the precomputed rank-supervised SKU fingerprints, and returns the top-ranked SKUs together with representative catalog images.

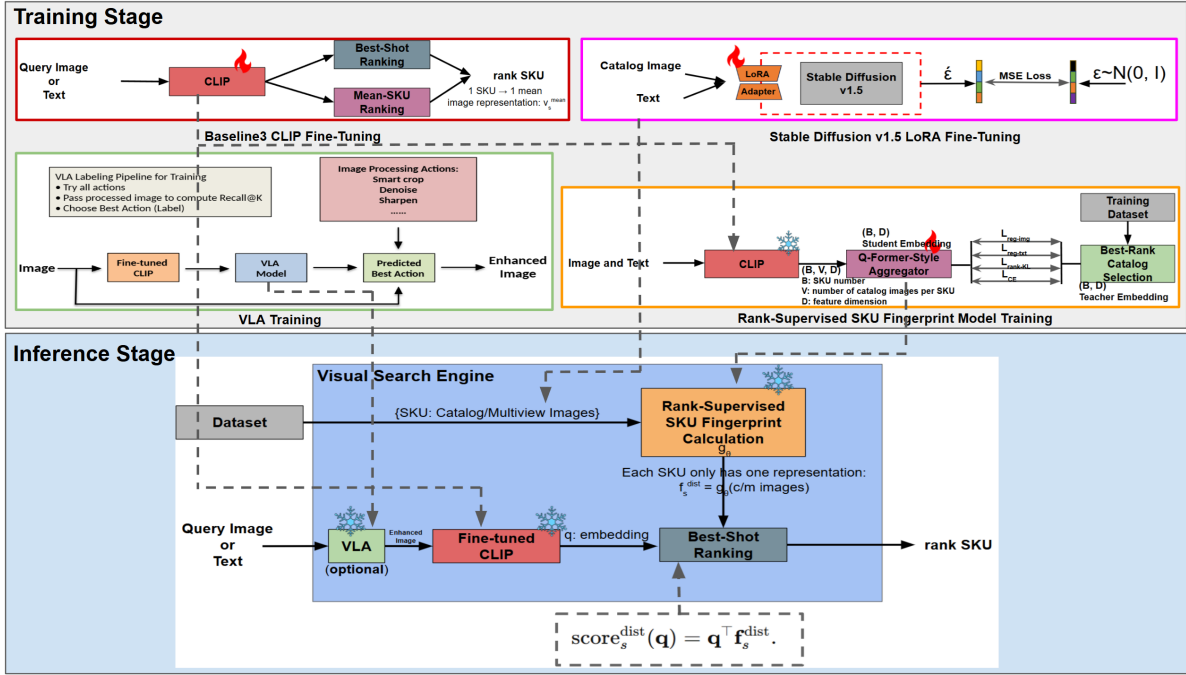


Figure 1: End-to-end pipeline. *Training:* CLIP is fine-tuned on DF2 SKUs (red); Stable Diffusion v1.5 is LoRA-tuned to generate identity-preserving multi-view catalog images (magenta); a Q-Former-style aggregator g_θ learns rank-supervised SKU fingerprints from CLIP embeddings of catalog and generative views (orange), with an optional VLA module for image enhancement (green). *Inference:* an image or text query is encoded by CLIP, compared to one precomputed fingerprint per SKU, and the top-ranked SKUs are returned.

4.2 Baselines (High-Level)

We compare against three baselines. Baseline 1 is a DF2 re-identification model: ResNet-50 with BNNeck trained using cross-entropy and triplet loss over SKU IDs, evaluated with an image-level gallery and per-SKU max pooling. Baseline 2 is frozen CLIP with SKU-mean pooling: a pretrained CLIP ViT-B/32 where

each SKU is represented by the mean of its catalog embeddings. Baseline 3 is fine-tuned CLIP with SKU-mean pooling: the same CLIP architecture fine-tuned on DF2 SKU labels for both image and text encoders, still evaluated with mean catalog embeddings. Losses, sampling strategies, and full training details are given in Appendix B.

4.3 Rank-Supervised SKU Fingerprint Model

Our main model learns an aggregation function

$$g_\theta : \mathbb{R}^{V \times D} \rightarrow \mathbb{R}^D, \quad (2)$$

that maps the V catalog (and multi-view) embeddings of a SKU into one L2-normalized *SKU fingerprint* $f_s^{\text{dist}} \in \mathbb{R}^D$. Using fine-tuned CLIP (Baseline 3) as teacher, we embed all catalog views and validation queries and, for each SKU s , identify a *best-shot teacher* embedding t_s by testing each catalog view as the sole representation and selecting the view that yields the best validation retrieval metric. Given CLIP embeddings $X_s \in \mathbb{R}^{V_s \times D}$, a small cross-attention transformer with K learnable query tokens (2 layers in our final model) attends over views, mean-pools the output tokens, applies a linear projection, and L2-normalizes the result to form f_s^{dist} ; this keeps g_θ lightweight while allowing it to focus on informative views.

We train g_θ on image and text queries with three signals. First, L2 regression pulls f_s^{dist} toward image and (optionally) text best-shot teachers. Second, rank distillation matches teacher and student rankings: for each query q we compute teacher scores $q^\top t_s$ and student scores $q^\top f_s^{\text{dist}}$ over SKUs and minimize the KL divergence between their temperature-scaled softmax distributions. Third, a cross-entropy loss over SKU labels encourages each query to give its ground-truth fingerprint the highest score. The total loss is a weighted sum of these terms; exact equations and hyperparameters are detailed in Appendix C. At deployment, we discard teacher-specific components and store only the fingerprints $\{f_s^{\text{dist}}\}$ in the index.

4.4 Generative Multi-View Catalog Augmentation

To enrich catalog views without increasing online latency, we build an offline multi-view pipeline based on Stable Diffusion v1.5 img2img with LoRA adapters. For each catalog crop, we condition SD v1.5 on prompts describing category, visibility, and a neutral studio background, run img2img at low noise strength, and apply LoRA adapters trained on a subset of SKUs so that generated views preserve garment identity. Generated candidates are filtered using CLIP cosine similarity, HSV color distance, and NSFW checks; accepted images are encoded by CLIP and concatenated with original catalog embeddings as additional views for X_s . Figure 2 shows that LoRA improves identity preservation, as it fixes the weird contour around lens of camera and the arms of the human over the frozen counterparts, while still providing pose and background variation useful for learning robust fingerprints.



Figure 2: Stable Diffusion v1.5 multi-view examples. From left to right: original catalog image, two views generated by the frozen model, and two views generated after LoRA fine-tuning. LoRA improves identity preservation and consistency of garment details while still varying pose and background.

5 Experiments

5.1 Experimental Setup

All models are trained on DF2 training SKUs and evaluated on the validation SKUs. For the fingerprint model, we report metrics on two subsets: *Seen-SKU* queries whose SKUs also appear in training, and *zero-shot* SKUs that never appear in training, which requires generalizing fingerprints to unseen products. We implement models in PyTorch using `open_clip` for CLIP encoders, use AdamW with cosine learning-rate schedules for 20–40 epochs and batch size 128–256, and measure latency on CPU via batched matrix multiplications between query embeddings and the SKU index, reporting p95 and mean latency per query.

5.2 Quantitative Results

Table 1: Image-to-SKU retrieval on DeepFashion2 validation set. “Seen-SKU” restricts to SKUs that appear in training; “Seen+Zero” uses all validation SKUs. Latencies are measured on CPU.

Method	Setting	R@1	R@5	R@10	MRR	NDCG@10	p95 (ms)	Mean (ms)
Baseline1 (ReID)	Seen+Zero	0.571	0.806	0.864	0.677	0.718	2.619	2.125
Baseline2 (CLIP, mean-SKU)	Seen+Zero	0.280	0.463	0.540	0.371	0.403	0.32	0.29
Baseline3 (FT-CLIP, mean-SKU)	Seen+Zero	0.587	0.807	0.866	0.688	0.727	0.37	0.30
Proposed fingerprint	Seen-SKU	0.857	0.978	0.991	0.908	0.929	0.13	0.14
Proposed-M (+multi-view)	Seen-SKU	0.865	0.975	0.990	0.914	0.932	0.09	0.10
Proposed fingerprint	Seen+Zero	0.596	0.817	0.873	0.696	0.736	0.39	0.33
Proposed-M (+multi-view)	Seen+Zero	0.602	0.823	0.879	0.702	0.742	0.29	0.23

Table 2: Text-to-SKU retrieval on DeepFashion2 validation set using synthetic metadata-based queries. Baseline1 does not support text queries.

Method	Setting	R@1	R@5	R@10	MRR	NDCG@10	p95 (ms)	Mean (ms)
Baseline1 (ReID)	Seen+Zero	–	–	–	–	–	–	–
Baseline2 (CLIP, mean-SKU)	Seen+Zero	0.005	0.019	0.031	0.019	0.016	0.40	0.32
Baseline3 (FT-CLIP, mean-SKU)	Seen+Zero	0.004	0.015	0.028	0.017	0.013	0.30	0.27
Proposed fingerprint	Seen-SKU	0.061	0.237	0.379	0.155	0.192	0.12	0.13
Proposed-M (+multi-view)	Seen-SKU	0.071	0.220	0.352	0.160	0.187	0.08	0.10
Proposed fingerprint	Seen+Zero	0.005	0.032	0.056	0.029	0.026	0.45	0.35
Proposed-M (+multi-view)	Seen+Zero	0.005	0.027	0.051	0.026	0.023	0.29	0.23

Rank-supervised fingerprints substantially outperform all baselines on image retrieval while offering lower CPU latency than both the ReID and CLIP baselines. Fine-tuned CLIP alone (Baseline 3) already closes much of the accuracy gap to ReID, but learning one fingerprint per SKU turns the system into a compact matrix–vector product while preserving or improving accuracy. Multi-view augmentation (Proposed-M) consistently improves Recall@1 and latency: generative views give the aggregator more pose and background variation, allowing fingerprints to focus on garment identity and reducing the need for a large image-level gallery. Note that we only augment 5.5k SKUs, which is only 30% of the full dataset. If we do the augmentation on the full dataset, the improvement is expected to be better. Text retrieval remains weaker because our text inputs are short synthetic prompts derived from metadata rather than rich

natural language; CLIP was pre-trained on web-scale captions, so metadata-style prompts under-utilize its language capacity, and the fingerprint model only sees text via distillation from the CLIP text tower. Seen-SKU performance is much higher than Seen+Zero because fingerprints for SKUs present in training can directly leverage supervised rank signals; this behavior is desirable for an online system, where new SKUs could continuously receive additional views and incremental training. In contrast, zero-shot SKUs are never observed during training, making lower performance expected and analogous to retrieving items that do not exist in the training dataset at all.

5.3 Qualitative Analysis

Figure 3 illustrates typical retrieval behavior. The first two rows show successful image-to-SKU retrieval: despite changes in background and pose, the model correctly ranks the ground-truth shorts and vest SKUs at the top. The third row shows a text query failure: both the ground-truth and retrieved SKUs are short-sleeve tops with similar color and logo style, highlighting how coarse text descriptions (no specific style and color pattern description) can be ambiguous even when category is correct.



Figure 3: Qualitative retrieval examples. Columns show user inputs (left), ground-truth catalog images (middle), and model outputs (right) for image queries (top two rows) and a text query (bottom row). The first two rows are correct matches; the bottom row shows a text query where the retrieved SKU is visually plausible but not the exact ground-truth SKU.

6 Conclusion and Future Work

We presented a SKU-per-vector visual search system for DeepFashion2 that combines CLIP-based encoders, Stable Diffusion multi-view augmentation, and a rank-supervised SKU fingerprint model. Our distilled fingerprints substantially improve both image and text retrieval accuracy while achieving very low CPU latency and matching or exceeding a strong re-id baseline. Future work includes richer natural-language queries, more sophisticated generative augmentations under strict identity constraints, and extensions to multi-attribute retrieval where multiple SKUs may be acceptable answers.

A Appendix: Metric Definitions

For completeness we present the exact formulas used to compute our retrieval metrics. Let Q denote the set of all queries, $q \in Q$ a query, s a candidate SKU with fingerprint f_s , and q a normalized query embedding. We define a similarity score

$$\text{score}_s(q) = q^\top f_s, \quad (3)$$

rank SKUs in descending $\text{score}_s(q)$, and let r_q be the rank of the first correct SKU for query q (with $r_q = \infty$ if it never appears up to the cutoff).

Recall@K.

$$\text{Recall@K} = \frac{1}{|Q|} \sum_{q \in Q} \mathbf{1}[r_q \leq K] = \frac{\#\{\text{queries where correct SKU is in top-}K\}}{|Q|}. \quad (4)$$

Mean Reciprocal Rank (MRR).

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{\mathbf{1}[r_q < \infty]}{r_q}. \quad (5)$$

NDCG@K. For each query we assign $rel_i = 1$ if rank i contains the correct SKU and $rel_i = 0$ otherwise. Discounted cumulative gain at cutoff K is

$$\text{DCG@K} = \sum_{i=1}^K \frac{2^{rel_i} - 1}{\log_2(i + 1)}. \quad (6)$$

In our single-label SKU setting, the ideal DCG is

$$\text{IDCG@K} = \frac{2^1 - 1}{\log_2(1 + 1)} = 1, \quad (7)$$

so

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} = \text{DCG@K} \in [0, 1]. \quad (8)$$

Latency. We measure end-to-end query time from a query embedding to a ranked SKU list using CPU-only matrix multiplications and reductions, and report the 95th percentile and mean latency in milliseconds over all validation queries.

B Appendix: Baseline Details

Baseline 1: DF2 Re-Identification (Image-Level Index)

Our replication baseline follows Bag-of-Tricks [7] and Hermans triplet loss [8]. We train an ImageNet-initialized ResNet-50 with BNNeck and a 512-dim L2-normalized embedding. The label space is SKU ID:

$$\text{sku_id} = (\text{pair_id}, \text{style} > 0, \text{category_id}). \quad (9)$$

We combine cross-entropy classification over SKU IDs with batch-hard triplet loss (margin 0.3). At test time we compute and store embeddings for all shop crops, form an image-level gallery matrix, compute dot products between each query and all gallery images, group gallery images by SKU, and take the max similarity per SKU to form final rankings.

Baseline 2: Frozen CLIP with SKU-Mean

We use a pretrained CLIP ViT-B/32 model [1] and freeze both image and text towers. Each catalog crop is encoded into a 512-dim embedding. For each SKU, we compute the mean of its catalog embeddings and L2-normalize it to obtain a single vector per SKU. Image and text queries are encoded by the corresponding CLIP tower and compared against SKU vectors by cosine similarity.

Baseline 3: Fine-Tuned CLIP with SKU-Mean

Baseline 3 uses the same CLIP architecture but fine-tunes it on DeepFashion2 using cross-entropy classification over SKUs for both image and text encoders. We keep the SKU-mean evaluation scheme. The model is fine-tuned for several epochs with a cosine-annealed learning rate and image augmentations (random resize-crop, horizontal flip, color jitter). Compared to Baseline 2, Baseline 3 slightly improves image retrieval accuracy and stabilizes the SKU-mean vs. best-shot gap, while maintaining low latency.

C Appendix: Fingerprint Loss Details

For completeness, we provide the full loss used to train the SKU fingerprint model.

For each SKU s we build a tensor $X_s \in \mathbb{R}^{V_s \times D}$ of CLIP embeddings from catalog and generative views. The Q-Former-style aggregator g_θ produces fingerprints f_s^{dist} . We define:

Embedding regression.

$$\mathcal{L}_{\text{reg-img}} = \frac{1}{|S|} \sum_s \|f_s^{\text{dist}} - t_s^{\text{img}}\|_2^2, \quad (10)$$

$$\mathcal{L}_{\text{reg-txt}} = \frac{1}{|S|} \sum_s \|f_s^{\text{dist}} - t_s^{\text{txt}}\|_2^2. \quad (11)$$

Rank distillation. For each query q we compute teacher scores $\alpha_s = q^\top t_s$ and student scores $\beta_s = q^\top f_s^{\text{dist}}$ and minimize

$$\mathcal{L}_{\text{rank-KL}} = \frac{1}{|Q|} \sum_q \text{KL}(\text{softmax}(\alpha/\tau) \parallel \text{softmax}(\beta/\tau)), \quad (12)$$

with temperature τ .

Classification contrastive loss.

$$\mathcal{L}_{\text{CE}} = -\frac{1}{|Q|} \sum_q \log \frac{\exp(q^\top f_{s(q)}^{\text{dist}}/\tau')}{\sum_{s'} \exp(q^\top f_{s'}^{\text{dist}}/\tau')}, \quad (13)$$

where $s(q)$ is the ground-truth SKU for query q .

Total loss.

$$\mathcal{L} = \lambda_{\text{img}} \mathcal{L}_{\text{reg-img}} + \lambda_{\text{txt}} \mathcal{L}_{\text{reg-txt}} + \lambda_{\text{rank}} \mathcal{L}_{\text{rank-KL}} + \lambda_{\text{CE}} \mathcal{L}_{\text{CE}}, \quad (14)$$

with $\lambda_{\text{img}} = \lambda_{\text{rank}} = \lambda_{\text{CE}} = 1$ and $\lambda_{\text{txt}} = 0.5$ in our final model.

D Appendix: Synthetic Text Prompt Construction

DeepFashion2 does not provide natural-language descriptions. We therefore construct synthetic prompts from metadata using scripts such as `dataset/build-deepfashion2-text-prompts.py`. For each crop we use the category name, domain (catalog vs. query), occlusion, and viewpoint to build a short sentence (e.g., “A user photo of a person wearing a partially occluded dress from the front.”), and optionally append supervision tokens “(style X , SKU Y)” during training with dropout. Validation and test prompts use only natural-language descriptions without SKU tokens.

E Appendix: VLA Policy Module Details

The Visual-Language-Action (VLA) module is a lightweight policy that selects one of 7 deterministic image-processing operations (e.g., different crops, denoising, sharpening, or exposure adjustments) to improve downstream CLIP features before retrieval. For each input image we compute (i) its CLIP visual embedding and (ii) a set of handcrafted quality features (sharpness, brightness, contrast, etc.). These are concatenated into a feature vector z that serves as input to the VLA policy.

The policy is a small neural network that maps z to a categorical distribution over the 7 processes. Training proceeds in two stages. First, we generate pseudo-labels by exhaustively applying all 7 processes to each training image, encoding each processed image with CLIP, and computing a top-SKU retrieval score with respect to the current SKU fingerprints. The process that yields the highest retrieval score is treated as the “best” action a^* for that image. Second, we train the VLA as a supervised classifier to predict a^* from z using a standard cross-entropy loss

$$\mathcal{L}_{\text{VLA}} = -\log p_{\theta}(a^* | z),$$

where $p_{\theta}(\cdot | z)$ is the softmax over the 7 actions produced by the policy network. Intuitively, the policy learns to predict which processing operation will most improve retrieval quality given the image’s appearance and quality statistics.

At inference time, given a user image we compute its CLIP embedding and quality features, form z , and use the VLA policy to select the most probable process $\hat{a} = \arg \max_a p_{\theta}(a | z)$. We apply the corresponding image-processing operation, re-encode the processed image with CLIP, and use the resulting embedding as the query vector passed to the SKU fingerprint index for retrieval.

F Appendix: Additional Implementation Details

Before running the project, you need to download the original DeepFashion2 dataset from here: <https://github.com/switchablenorms/DeepFashion2>. You need to follow the process to get password from the dataset author and un-compress the dataset files using that password after downloading them. Our codebase separates datasets, models, training, and evaluation, please see `./RUN.md` for details:

- `./scripts/prepare-deepfashion2-sku.sh`: builds SKU-level crops and image-text pairs, with optional flags for generative multi-view augmentation.
- `./scripts/prepare_df2_reid_splits.sh`: constructs the DeepFashion2 ReID-style splits used for training and evaluating the Baseline 1 ReID ResNet-50 model.
- `./scripts/train_baseline1_reid.sh` and `./scripts/eval_baseline1_reid_val.sh`: train and evaluate the Baseline 1 ReID ResNet-50 model.

- `./scripts/train_clip_sku_df2.sh` and `./scripts/eval_clip_sku_df2.sh`: train and evaluate the Baseline 2 and Baseline 3 CLIP-SKU models.
- `./scripts/train_sd_lora_df2.sh`: trains LoRA on SD v1.5 Img2Img model.
- `./scripts/gen_bestshot_teachers_df2.sh`: generates teacher embedding for fingerprint distillation.
- `./scripts/gen_sd_aug_df2_light5p5k_sdloraftep3.sh`: generates augment multi-view images using fine-tuned SD v1.5 Img2Img model.
- `./scripts/train_sku_fingerprint_student_multiview5p5k_sd3ep.sh`: train student aggregator model using all catalogs and generated multi-view images from fine-tuned SD v1.5 Img2Img model.
- `./VLA_model`: contains the VLA policy implementation and training code described in Appendix E.
- `./scripts/eval_sku_fingerprint_student_demo.sh`: evaluate the distilled SKU fingerprint model performance.

Please refer to `RUN.md` for detailed commands to reproduce our experiments. Before running, download the following pretrained checkpoints:

1. Fine-tuned CLIP (Baseline 3) weights: https://drive.google.com/file/d/1mHehmiGhy_4SgWXDGWIpeIXNRGnAS4uQ/view?usp=sharing
2. Student fingerprint aggregator weights: <https://drive.google.com/file/d/18s3T73SfMycJpnw35B7/view?usp=sharing>
3. Trained VLA policy weights: <https://drive.google.com/file/d/1qqoGZwNzz2JhDF9SLvlhwJY70Q510p/view?usp=sharing>
4. Fine-tuned SD v1.5 w/ LoRA weights: <https://drive.google.com/file/d/1uLRqlPOqFq3jvEAbpRPJ/view?usp=sharing>

After downloading and running the `./scripts/prepare_deepfashion2_sku.sh`, then set the following fields in `./scripts/eval_sku_fingerprint_student_demo.sh`, then you can reproduce the results of our proposed method as described in Table 1 and Table 2 by running:

`./scripts/eval_sku_fingerprint_student_demo.sh`.

1. Set "SKU_ROOT" to your root of generated SKU dataset using `./scripts/prepare_deepfashion2_sku.sh` (the "SKU_ROOT" variable inside).
2. Set "CLIP_SKU_CKPT" to the Fine-tuned CLIP weights file you download from above link 1.
3. Set "STUDENT_CKPT" to the Student fingerprint aggregator weights file you download from above link 2.
4. Set "-vla_checkpoint" (optional) to the trained VLA policy weights file you download from above link 3
5. Set "VAL_IMAGE_TEXT" to the generated jsonl file using `./scripts/prepare_deepfashion2_sku.sh`, usually under SKU_ROOT and named `validation_image_text.jsonl`

G Appendix: Timeline, Research Log, and Compute

Timeline and roles

- Weeks 1–2: processed DF2, implemented initial CLIP baselines, and set up SKU-level cropping and text prompt generation.
- Weeks 3–4: trained the ReID baseline and fine-tuned CLIP, and built the metric/latency evaluation pipeline.
- Weeks 5–6: designed and trained the rank-supervised SKU fingerprint model, including best-shot teachers and Q-Former-style aggregation.
- Week 7: implemented LoRA-based Stable Diffusion multi-view generation and filtering.
- Week 8: ran final experiments, integrated the VLA module, and produced the report and poster.
- Roles: Patrick led CLIP baselines, fingerprint modeling, ReID baseline, and generative model fine-tuning. Mingyang focused on VLA; Kanlong focused on generative model visualizations.

Research log (high level)

- Frozen CLIP with simple SKU-mean pooling was surprisingly competitive, motivating a learned SKU-level aggregator rather than a heavy reranker.
- Early fingerprint variants using simple MLP pooling were unstable; switching to a Q-Former-style attention module gave consistent gains.
- Pure L2 regression to teacher embeddings sometimes hurt ranking; adding KL-based rank distillation improved NDCG and stabilized training.
- Generative augmentation required CLIP/HSV filtering and LoRA to avoid off-identity artifacts; mild viewpoint/background changes helped, while aggressive edits (e.g., color changes) hurt exact-SKU retrieval.
- Latency measurements were initially noisy on GPU; standardizing on CPU matrix multiplications provided a fair, reproducible comparison.

Compute and thought experiment

- We used roughly 150 GPU-hours (about \$225 at \$1.50/hour) across ReID, CLIP baselines, fingerprint training, LoRA/multi-view generation, and VLA training.
- With an extra \$450 (about 300 GPU-hours), we would distill fingerprints from larger encoders (e.g., CLIP ViT-L/14), explore deeper/wider aggregators and loss ablations (including stronger text supervision), and generate multi-views for all 21K SKUs to analyze seen vs. zero-shot behavior per category.

References

- [1] A. Radford et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

- [2] Y. Ge et al. DeepFashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *CVPR*, 2019.
- [3] R. Wu et al. FashionIQ: A new dataset towards retrieving images by natural language feedback. In *CVPR*, 2021.
- [4] J. Peebles and S. Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.
- [5] Y. Zhang et al. Cross-modal pretraining for product retrieval with 1M real-world images. In *ICCV*, 2021.
- [6] Z. Tan et al. FashionBERT: Text and image matching with adaptive loss for cross-modal retrieval. In *KDD*, 2020.
- [7] H. Luo et al. Bag of Tricks and a Strong Baseline for Deep Person Re-Identification. In *CVPR Workshops*, 2019.
- [8] A. Hermans, L. Beyens, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. arXiv:1703.07737, 2017.