AI-Driven Visual Search for E-Commerce: Final Report

KARS Ahmet Kutlug Alperen[1*] YANG Zhenyao[1*] MIGITA Rana Aomi[1*]
Hong Kong University of Science and Technology
akakars@connect.ust.hk zyangbz@connect.ust.hk ramigita@connect.ust.hk

Abstract

E-commerce visual search is crucial for enhancing user experience and product discovery, addressing the limitations of text-based queries in capturing nuanced visual attributes. However, effective visual search solutions remain largely inaccessible to smaller retailers. This paper aims to bridge this gap by developing a content-based visual search model capable of retrieving visually similar clothing products from a database using deep metric learning. We investigate different architectural and loss function strategies, including multi-branch networks with contrastive loss and ResNet50 backbones trained with classical triplet loss and Proxy-Anchor loss, leveraging subsets of the DeepFashion dataset. Our experiments reveal that classical triplet loss faces significant challenges with stability and scalability on datasets with high class fragmentation. In contrast, we demonstrate that the Proxy-Anchor loss provides superior stability, reliable convergence, and scales effectively. This work contributes a practical and robust deep metric learning approach using proxy-based methods, offering a viable solution for fine-grained visual search in large-scale e-commerce applications.

1. Introduction

Text-based search methods in e-commerce often fail to capture nuanced visual attributes, as users struggle to express them through textual queries [2]. This limitation leads to suboptimal search results and missed opportunities for both consumers and retailers. As an alternative, fashion image recognition and retrieval have gained significant attention. However, while major platforms offer visual search, such solutions remain largely inaccessible to smaller retailers due to technical and resource constraints [6].

In this work, we aim to improve e-commerce visual search technology by developing a model capable of retrieving visually similar clothing products from a database. Our approach leverages deep metric learning to learn robust visual representations, investigating different architectural and loss function strategies tailored for this task. By applying these methods, we seek to provide effective visual search functionality, supporting retailers by improving product discoverability and enhancing user experience.

This paper details our system implementation, presents results demonstrating the efficacy of our approach, and discusses potential future work.

2. Problem Statement

Effective visual search is crucial for e-commerce, addressing the limitations of text-based queries in capturing nuanced visual attributes. The core problem we tackle is the development of a content-based visual search model capable of retrieving products that are visually similar to a query image. In current platforms, reliance on keywords often fails to capture detailed appearance, especially with limited product descriptions, making text retrieval imprecise.

Our goal is to enable image-to-image retrieval: given a product image, our system should return a ranked list of similar items based purely on visual content. This problem falls under the field of metric learning, where the objective is to learn an embedding function that maps visually similar items to nearby points in a feature space, while mapping dissimilar items far apart.

Solving this involves technical challenges, including building effective feature extractors that generalize across diverse products and handling significant intra-class variability. We address this problem by designing and evaluating deep metric learning architectures trained on subsets of the DeepFashion dataset.

2.1. Datasets

We use the DeepFashion dataset [3] developed by The Chinese University of Hong Kong's Multimedia Lab. This dataset is designed for cross-domain fashion retrieval, where consumer images are matched with shop images. The dataset contains 239,557 clothing images and includes bounding boxes, fashion landmarks, item labels, and attribute annotations that aid model evaluation.

*Equal contribution

For our project, due to hardware and time constraints, we conducted experiments on **subsets** of the DeepFashion dataset rather than the full collection. The specific subsets used, including their exact sizes and filtering criteria, differed between the primary experimental streams presented in this paper (the MILDNet experiments in Section 4, the Triplet Loss experiments in Section 5, and the Proxy-Anchor experiments in Section 6). Details regarding the dataset subset used for each specific experiment are provided within the corresponding section.

2.2. Expected Results and Evaluation

We expect our visual search model to retrieve visually similar clothing items given a query image, with high Top-K accuracy and well-clustered feature embeddings. In this work, we evaluate performance using quantitative metrics and qualitative visualizations.

Quantitative metrics include:

- **Triplet Accuracy:** Measures the percentage of triplets (A, P, N) satisfying $d(A, P) < d(A, N)$, assessing the model's ability to learn relative distances within batches.
- **Top-K Recall:** Measures retrieval performance by calculating the percentage of queries where a ground-truth positive is found within the top K retrieved images from the gallery. This metric directly reflects the system's effectiveness in a visual search scenario.

Qualitative eval Qualitative evaluation through t-SNE visualizations and example retrieval results is also used to assess the learned embedding space and visual similarity capture.

3. Methodology

Our approach focuses on training deep metric learning models to embed clothing images into a low-dimensional space where visual similarity corresponds to geometric proximity. The core idea is to train neural networks using loss functions designed to pull embeddings of similar items closer together while pushing embeddings of dissimilar items farther apart. This involves training neural networks with loss functions designed to pull similar item embeddings closer and push dissimilar ones apart. To investigate different architectural and training strategies, our work is structured around three primary experimental streams:

1. Exploring a multi-branch MILDNet architecture with triplet-based contrastive loss, combining deep and multi-scale features (4).
2. Utilizing a ResNet50 backbone trained with classical triplet loss, examining various embedding heads and sampling strategies (5).
3. Employing a ResNet50 backbone trained with Proxy-Anchor loss to address the training stability and scalability challenges of classical triplet loss (6).

3.1. General Data Preparation

Data preparation involved common steps applied across all experimental streams. This included resizing input images to a fixed resolution 224×224 pixels using interpolation, followed by standard normalization. Image loading was handled efficiently using PyTorch `Dataset` and `DataLoader` classes. Data augmentation techniques were also applied during training in certain experiments to enhance model robustness. Specific details regarding the dataset subsets used and the particular processing pipelines, including unique data augmentation strategies, are provided within their respective sections (4, 5, and 6).

4. MILDNet Experiments

While our main focus shifted towards the Triplet network architecture, we initially explored a Siamese network approach based on the MILDNet multi-branch architecture, inspired by [5]. This architecture was chosen for its ability to extract features at multiple scales and levels of abstraction, combining a deep VGG19 backbone with two shallower convolutional pathways.

4.1. Architecture Details

Our Siamese network implementation follows the structure depicted in Figure 1. It consists of three main branches:

- **VGG19 Branch:** Utilizes the feature extractor of a pre-trained VGG19 model, followed by global average pooling and two fully connected layers with ReLU activations and dropout. This branch captures high-level semantic features.
- **Shallow Branch 1 & 2:** These branches process downsampled versions of the input image through a series of pooling and convolutional layers. They are designed to capture low-level visual attributes like textures, shapes, and patterns.

The outputs of the two shallow branches are flattened, concatenated, and L2-normalized to form a single shallow feature vector. This vector is then concatenated with the L2-normalized output of the VGG19 branch. The combined features are passed through a final fully connected layer and L2-normalized to produce the final embedding. All three branches share weights across both arms of the network.

4.2. Dataset and Triplet Construction

For training and evaluating the Siamese network, we used a subset of the DeepFashion Consumer-to-Shop

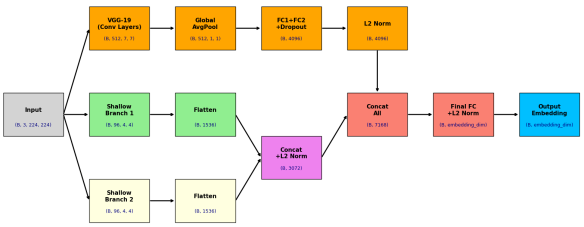

Figure 1. Our multi-branch Siamese network architecture. The three pathways extract features at different levels of abstraction, following the approach of [5].

Clothes Retrieval dataset. Specifically, we focused on images within the `CLOTHING` category, which includes 8 subcategories: T-Shirt, Tank Top, Summer Wear, Polo Shirt, Pants, Jeans, Coat, and Blouse. This subset contains 2,469 unique products and a total of 12,577 images.

To train the network using triplet loss, we constructed triplets (A, P, N), where A is an anchor image, P is a positive image, and N is a negative image.

- A positive pair (A, P) consists of two different images of the *same* product (identified by the product ID).
- A negative pair (A, N) consists of two images of *different* products.

To create challenging negative examples, we sampled negative pairs based on two criteria:

- **In-class negatives:** Images from different products but within the *same* category as the anchor.
- **Out-of-class negatives:** Images from products belonging to a *different* category than the anchor.

For our experiments, we generated 100,000 triplets, with a fixed ratio of 70% in-class negatives and 30% out-of-class negatives within the negative pairs. The full set was then split into training (70%), validation (15%), and testing (15%) sets.

4.3. Loss Function and Metrics

The network is trained using the Contrastive Loss function, adapted for triplets. Given anchor (A), positive (P), and negative (N) embeddings, the loss aims to make $d(A, P)$ small and $d(A, N)$ large:

$$\mathcal{L}(A, P, N) = \frac{max\left(d(A, P)^2 + \max(0, m - d(A, N))^2, 0\right)}{2} \quad (1)$$

where $d(\cdot, \cdot)$ is the Euclidean distance between embeddings and m is the margin hyperparameter (set to 1.0 in our experiments).

Performance was evaluated using Triplet Accuracy and Top-K Recall, as defined in Section 2.2.

4.4. Experiment 1: Baseline Training

Our initial training experiment utilized a set of baseline hyperparameters, including a learning rate of 0.001, weight decay of 1e-5, and a dropout rate of 0.5. The VGG19 backbone was frozen for the first 3 epochs (`FREEZE_UNTIL_EPOCH = 3`), after which it was unfrozen and fine-tuned with a learning rate of 0.0001. Training was run for 10 epochs using an RMSprop optimizer and a `ReduceLROnPlateau` scheduler with a factor of 0.1, monitoring validation loss. No explicit data augmentation beyond basic resizing and normalization was applied during training in this setup.

The training process showed rapid convergence on the training set, with training accuracy quickly approaching 100% (see Figure 2). However, validation accuracy plateaued relatively early, and validation loss started increasing after the backbone was unfrozen, indicating signs of overfitting to the training data.

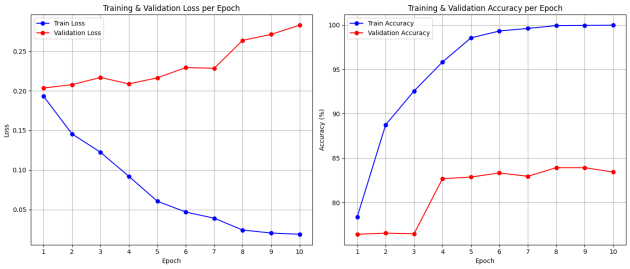

Figure 2. Training and Validation Loss/Accuracy per Epoch for Experiment 1 (Baseline).

The final performance after 10 epochs is summarized in Table 1.

Table 1. Summary of Results for Experiment 1 (Baseline Training)

Metric	Value
Final Test Loss	0.2752
Final Test Triplet Accuracy	83.69%
Recall@10	39.01%
Recall@20	52.30%
Recall@30	59.00%

4.5. Experiment 2: Hyperparameter Tuning and Augmentation

To address the overfitting observed in Experiment 1 and attempt to improve performance, we conducted a second experiment with tuned hyperparameters and data augmen-

tation. Key modifications were implemented across several areas:

Regularization was increased by raising the dropout rate to 0.6 and the weight decay (L2 regularization) to 1e-4. **Training dynamics** were adjusted by extending the initial backbone freezing period to 4 epochs (`FREEZE_UNTIL_EPOCH = 4`), reducing the learning rate for phase 2 (after unfreezing) to 5e-05, and modifying the `ReduceLROnPlateau` scheduler factor to 0.5 for a less aggressive reduction. **Significant data augmentation** was applied during training, including random horizontal flips, color jitter, random rotations, and random affine transformations. Finally, **early stopping** based on validation accuracy with a patience of 5 epochs was introduced to mitigate overfitting.

Figure 3 shows the training dynamics for this experiment. The data augmentation helped to reduce the gap between training and validation accuracy, mitigating overfitting compared to the baseline. The validation accuracy showed steady improvement, reaching a peak before the end of the 10 planned epochs.

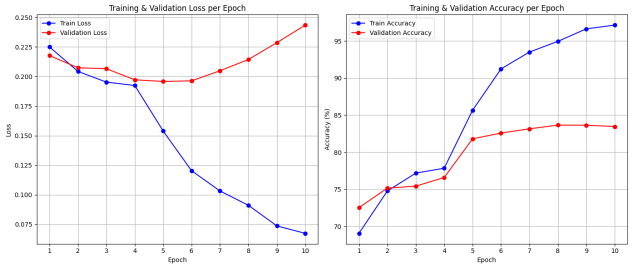

Figure 3. Training and Validation Loss/Accuracy per Epoch for Experiment 2 (Tuned).

The final performance, evaluated using the best model checkpoint based on validation accuracy, is summarized in Table 2.

Table 2. Summary of Results for Experiment 2 (Tuned Training)

Metric	Value
Final Test Loss	0.1966
Final Test Triplet Accuracy	84.51%
Recall@10	38.42%
Recall@20	49.62%
Recall@30	57.40%

4.6. Retrieval Visualization

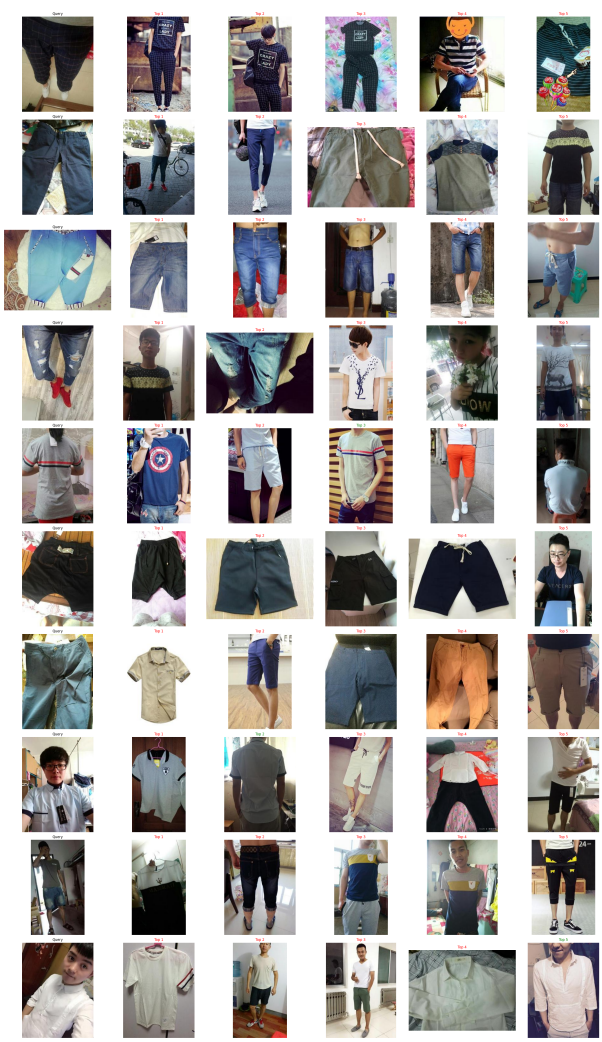

Figure 4. Top 5 retrieval results for sample queries from the test set. The first column shows the query image. Subsequent columns show the top 5 retrieved images from the gallery.

To qualitatively assess the retrieval performance, we visualize the top 5 retrieved images for a selection of query images from the test set. The first column shows the query image, and subsequent columns display the retrieved images ranked by embedding distance. Correct retrievals (images of the same product) will be highlighted with a green title color. (see Figure 4).

4.7. Discussion on Results

The two experiments reveal interesting insights. While Experiment 2, with its stronger regularization and data augmentation, successfully reduced overfitting and achieved a slightly higher triplet accuracy on the test set (84.51% vs 83.69%), Experiment 1 yielded marginally better perfor-

mance in the actual retrieval task (Recall@K). This discrepancy between batch-level triplet accuracy and gallery-level retrieval performance highlights that while the model effectively separates sampled positive-negative pairs within batches, this local ordering doesn't perfectly ensure the desired global structure where all positives are clustered closer to the anchor than all negatives in the full gallery.

The retrieval results themselves (around 38-39% Recall@10) are a solid starting point for a complex dataset like DeepFashion with significant intra-class variation and visual similarity between different products within the same category. Crucially, as illustrated by the retrieval visualization (Figure 4), even when the exact same product is not retrieved within the top-K, the model demonstrates an ability to capture visual similarities such as patterns, shapes, colors, and components. This indicates that the learned embeddings represent visually related items, which is valuable for e-commerce scenarios where recommending visually similar products can enhance user experience and product discoverability, even if they are not identical matches.

5. Triplet Loss and Embedding Experiments

Triplet loss is a fundamental technique in deep metric learning, particularly well-suited for tasks such as visual similarity and retrieval. Rather than training a model to classify images into discrete categories, the goal of triplet loss is to project images into an embedding space where semantically similar items are positioned closely, and dissimilar ones are pushed farther apart. This idea was popularized in the FaceNet paper by Schroff et al. [4], where the authors applied triplet loss for face recognition and clustering tasks.

A triplet consists of an anchor image, a positive image from the same class, and a negative image from a different class. The loss function ensures that the distance between the anchor and the positive is smaller than the distance between the anchor and the negative by at least a margin α:

$$\mathcal{L}(A, P, N) = \max\left(d(A, P) - d(A, N) + \alpha, 0\right), \quad (2)$$

where $d(\cdot, \cdot)$ denotes the distance metric, which we take to be cosine distance in all our experiments.

Our dataset is based on the DeepFashion Consumer-to-Shop Clothes Retrieval dataset. Each clothing item is assigned a unique identifier of the form `id_XXXXXX`, which we use as a class label. To ensure meaningful triplet sampling, we only retain classes with at least two image samples.

5.1. Experiment 1: Small Subset of Clothing Categories

Our first experiment focuses on a small subset of the dataset. We used images from the `CLOTHING` category, specifically `T_Shirt` and `Polo_Shirt`. After filtering, the dataset contained 885 valid images (640 for training and 48 for testing). The base model consists of a ResNet50 backbone pretrained on ImageNet, followed by an embedding head projecting to a 128-dimensional feature space.

We experimented with five embedding head variants (`modelver1` to `modelver5`), each incorporating different architectural strategies such as deeper MLPs, residual-style blocks, or Transformer-inspired components. The best performing model used a residual-style head (`modelver4`), achieving a Top-10 accuracy of 79.2%, as shown in Table 3.

Table 3. Top-K retrieval accuracy on small dataset (T-Shirt and Polo_Shirt only)

Model	Top-1	Top-5	Top-10
`modelver1` (Linear)	0.229	0.583	0.750
`modelver2` (Deep MLP)	0.313	0.604	0.771
`modelver3` (Funnel)	0.188	0.500	0.667
`modelver4` (Residual)	**0.333**	0.604	**0.792**
`modelver5` (Transformer-style)	0.271	**0.688**	0.771

We also tested hard triplet mining on `modelver4`, but the Top-10 accuracy decreased to 58.3%. We attribute this to the limited number of valid triplets per batch due to the small dataset size and class sparsity. In many cases, no valid positive or negative could be found for an anchor within the same batch, rendering hard mining ineffective.

5.2. Qualitative analysis of the learned embedding

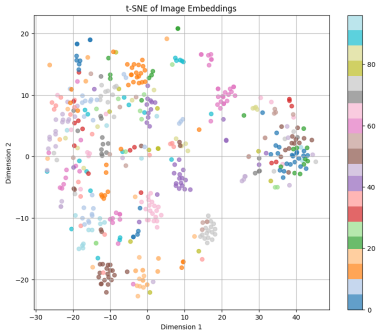

Figure 5. t-SNE visualization of 128-dimensional image embeddings from the trained model (modelver4).

Figure 5 shows a two-dimensional t–SNE projection of the 128-D embeddings produced by the network after training on the *small* dataset (809 training images, 229 test images). Even with so few examples per class, the triplet-loss network forms well-defined, compact clusters: points that belong to the same *id_xxxxxx* label are tightly grouped, while clusters from different items are clearly separated by

empty space. This structure confirms that the model has internalised a semantic notion of visual similarity rather than memorising individual images. The tight intra-cluster cohesion combined with large inter-cluster margins is exactly what triplet loss is designed to encourage.

5.3. Experiment 2: Scaling to a Larger Dataset

To test the scalability of our approach, we expanded the dataset to include images from both the `TOPS` and `CLOTHING` categories, ensuring that no `T_Shirt` images overlapped across sources. The expanded dataset contained 7,953 images (6,206 for training and 580 for testing).

Despite the increased data size, retrieval accuracy dropped significantly. The best model (`modelver4`) achieved only 44.3% Top-10 accuracy with random triplet sampling, and just 13.6% with hard triplet mining. We suspect this is due to label fragmentation: many new classes contained only 2-3 images, making generalization difficult. Additionally, the chance of finding multiple samples of the same class within a batch decreased as the number of classes grew, reducing the effectiveness of both sampling and mining strategies.

6. Proxy-Anchor Loss

After extensive experimentation we found that the classical triplet-loss objective often fails to converge in our large-scale setting: enumerating $O(B^3)$ anchor-positive-negative triples per batch leads to exploding gradients and high gradient variance once hard-mining is applied; with thousands of product-IDs and only a few examples per ID, most mini-batches contain no valid (A, P, N) triplets, causing updates to vanish or become unstable; and the margin parameter α together with the mining schedule prove highly sensitive, requiring painstaking tuning to avoid divergence or collapse.

To sidestep these issues, we adopt the *Proxy-Anchor* loss of Kim *et al.* [1], which replaces explicit pair mining with a set of learnable *proxy vectors* $\{\mathbf{p}_c\}_{c=1}^{C}$, one per class. Each embedding $\mathbf{f}_i \in \mathbb{R}^D$ is compared directly to all proxies in the batch, yielding only $O(B \times C)$ interactions and bounded gradient variance, and by aggregating over all positive and negative proxies in two logarithmic terms, the objective naturally emphasizes hard examples without an external mining policy.

Concretely, let $\tau > 0$ be a temperature and $m \geq 0$ an angular margin. For a given sample $\mathbf{f}_i$ of class y_i, denote

$$\mathcal{P} = \{\mathbf{p}_{y_i}\}, \qquad \mathcal{N} = \{\mathbf{p}_c : c \neq y_i\}.$$

The Proxy-Anchor loss is defined as

$$\mathcal{L}_{\text{ProxyAnchor}}(\mathbf{f}_i) = \underbrace{\log\Big(1 + \sum_{\mathbf{p}^+ \in \mathcal{P}} \exp\big(-\tfrac{1}{\tau}(\mathbf{f}_i^\top \mathbf{p}^+ - m)\big)\Big)}_{\text{pull positives together}} + \underbrace{\log\Big(1 + \sum_{\mathbf{p}^- \in \mathcal{N}} \exp\big(\tfrac{1}{\tau}(\mathbf{f}_i^\top \mathbf{p}^- + m)\big)\Big)}_{\text{push negatives apart}}. \tag{3}$$

Here, the first term attracts each feature toward its own proxy by at least margin m, while the second term repels it from all other proxies. Empirically this objective converges 3-4x faster than triplet loss and remains stable even with modest batch sizes, making it ideally suited for our in-shop clothing retrieval task.

6.1. Evaluation of the Initial 40-Epoch Run

Table 4 summarizes the key retrieval and classification metrics observed over 40 epochs on the 117-class / 723-train / 52-test split:

Table 4. Best-seen and final metrics for the 40-epoch Proxy-Anchor model with split ratio of 0.1.

Metric	Best observed	At epoch 40
Recall@1	0.269 (epoch 35)	0.231
Recall@5	0.442 (epoch 25)	0.404
Recall@10	0.596 (epoch 20/40)	0.596
Class-Acc@1	0.540 (epoch 15)	0.460
Class-Acc@5	0.750 (epoch 40)	0.750

Classification accuracy, measured by direct proxy-based top-k matching, peaked at 0.540 for top-1 in epoch 15 and then oscillated down to 0.460 by epoch 40, whereas top-5 accuracy improved steadily throughout training to 0.750. The divergence between proxy classification accuracy and image-image retrieval (e.g. Class-Acc@1 vs. Recall@1) highlights that while the network became increasingly confident in assigning embeddings to their class proxies, the fine-grained nearest-neighbor relationships among individual images exhibited greater sensitivity to over-tuning.

Finally, the pronounced metric fluctuations, particularly for Recall@1, are amplified by the small test set (52 images), where the movement of one or two nearest neighbors can shift the recall by nearly 0.02. Overall, the model converged reliably under the proxy-anchor loss, but attained its best generalization performance at approximately 15-35 epochs, after which marginal over-fitting on the small test set caused a slight performance decline.

In a follow-up experiment we modified the train/test split from 90/10 to 80/20 (652 train vs. 151 test images). Although this reduced the training set by 71 im-

ages, peak Recall@1 rose from 0.269 to 0.325 and Recall@5 increased likewise, while the larger gallery produced markedly smoother evaluation curves. This outcome indicates that our model was not yet constrained by the amount of training data, but benefited more from a more representative test/gallery distribution that reduces the variance in nearest-neighbour recall estimates.

6.2. Hyperparameter Tuning

To identify the most effective configuration for our Proxy-Anchor model, we conducted a series of controlled sweeps over five key hyperparameters while holding the others at their baseline values: batch size $B \in \{8, 16, 32, 64, 128\}$, embedding dimension $D \in \{64, 128, 256, 512\}$, angular margin $m \in \{0.1, 0.2, 0.3, 0.5\}$, temperature $\tau \in \{0.1, 0.2, 0.3, 0.5\}$, and initial learning rate $\eta \in \{1 \times 10^{-3}, 3 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}\}$. Each sweep recorded the peak Recall@1 on the 80/20 split. Table 5 summarizes these results alongside the next-best alternatives.

Table 5. Optimal hyperparameters and their immediate competitors.

Hyperparameter	Best	Runner-up
Batch size B	16	8
Embedding dim. D	256	512
Margin m	0.5	0.3
Temperature τ	0.1	0.2
LR η	1×10^{-3} (stop@20)	3×10^{-4} (full 40)

These settings guided our subsequent large-scale run on 777 classes.

6.3. Qualitative analysis of the learned embedding

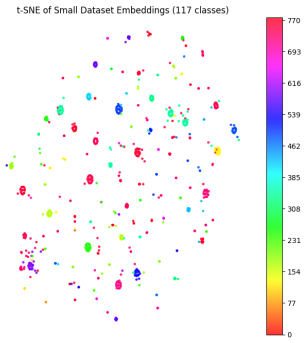

Figure 6. t-SNE visualization of Proxy-Anchor embeddings for 117 classes.

In the t-SNE plot (Figure 6), most of the 117 product classes form tight, well-separated clusters, demonstrating strong intra-class compactness. A few clusters overlap or fragment into multiple islands, indicating that visually similar items (e.g. red shirts of different cuts) remain insufficiently distinguished and some nuisance factors (pose, background) perturb the embedding. Overall, the embedding achieves good coarse retrieval performance, most same-class points lie closer to one another than to other classes, while highlighting specific class pairs for targeted refinement.

6.4. Evaluation on the 777-Class Split

In this experiment, we trained the Proxy-Anchor model using a ResNet-50 backbone with a 256-dimensional embedding head for 40 epochs. We set the batch size to 16, learning rate to 3×10^{-4}, angular margin $m = 0.5$, and temperature $\tau = 0.1$. The dataset comprised 777 distinct classes, with 5,575 images for training and 1,401 images for testing.

Figure 7 shows the average training loss over 40 epochs. We observe a steep descent from 16.3 at epoch 1 to around 10.8 by epoch 15, after which the curve flattens and oscillates between 10.4 and 10.9. This plateau indicates that the proxies and embeddings have reached a near-equilibrium: further updates mostly reorder points on the hypersphere rather than improving discriminability.

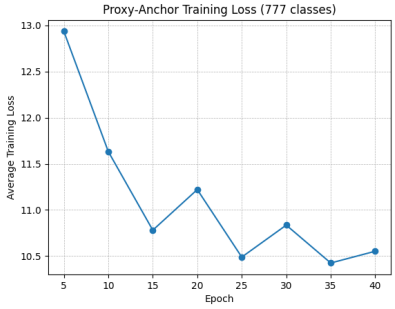

Figure 7. Proxy-Anchor training loss on the 777-class split.

Retrieval performance (Recall@1, @5, @10, @20, @30) is plotted in Figure 8. All five metrics climb rapidly in the first 15 epochs (e.g. Recall@1 rises by +0.075 absolute), then suffer a dip between epochs 16-20 despite continued loss reduction, a classic sign of mild over-fitting. A secondary peak appears around epoch 35 (Recall@1 = 0.443), but by epoch 40 the metrics have regressed to values near those at epoch 10, confirming that prolonged training eventually harms nearest-neighbour structure.

Figure 9 compares proxy-based classification accuracy (top-1 and top-5). Unlike retrieval, Class-Acc@1 continues to improve until epoch 25 (reaching 0.541) before plateauing, whereas Recall@1 fluctuates strongly. This gap after epoch 25 indicates that embeddings remain well aligned to their class proxies but lose fine-grained separability among

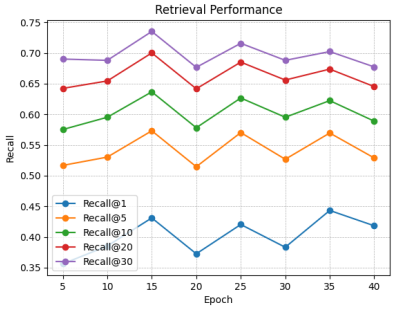

Figure 8. Retrieval performance (Recall@{1,5,10}) over 40 epochs.

individual images.

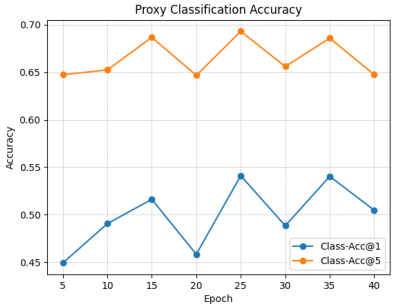

Figure 9. Proxy-based classification accuracy at top-1 and top-5.

Compared to the smaller 117-class split, peak Recall@1 improves from 0.269 to 0.443 and Recall@5 from 0.442 to 0.573, demonstrating that the model scales positively with more classes and images. Moreover, with 1 401 test images the recall curves are far smoother, showing that observed oscillations (±0.02-0.05) now reflect true optimisation dynamics rather than sampling noise.

The two natural checkpoints are epoch 15; where loss, retrieval, and classification performance converge at their first optimum, and epoch 35, which yields the highest Recall@1 but follows a mid-training dip. Given the small absolute difference (≈ 0.012), epoch 15 is the safer choice for early stopping, as it precedes any over-fitting and lies on the steeper, more stable portion of the loss curve.

Overall, this run confirms that Proxy-Anchor scales effectively, achieving Recall@1 ≈ 0.44 with only seven shots per class, but that constant-rate training beyond 15 epochs yields diminishing returns. Implementing early stopping is therefore key to locking in optimal performance.

7. Future Work

One direction that may lead the future work is using the bounding boxes in the DeepFashion dataset to better isolate the main product in each image. Currently, our model processes the full image, which can include irrelevant parts from background, accessories, or other clothing items. This additional visual noise might be affecting how well our embeddings capture the core product features.

To solve this issue, we could first train a model to detect and crop the specific clothing item we care about before generating the embeddings. By focusing only on the relevant region, we could feed the model with cleaner input and potentially get better retrieval results. Even though we did not have the sufficient time to explore this feature throughout the course of the project, this pipeline stands as a natural next step in our direction and it could improve the robustness of our model.

8. Conclusion

This study investigated deep metric learning techniques for in-shop clothing retrieval, exploring multi-branch architectures with contrastive loss and ResNet50 backbones with classical triplet loss and Proxy-Anchor loss. We found that classical triplet loss struggled with optimization and scalability, particularly on datasets exhibiting high class fragmentation. In contrast, the Proxy-Anchor loss demonstrated superior stability and effectiveness. Using the Proxy-Anchor method, our experiments achieved strong retrieval performance, confirming reliable convergence and positive scaling with dataset size. This work establishes proxy-based metric learning as a practical and robust approach for fine-grained visual search in e-commerce.

References

[1] Sungyeon Kim, Donghyeon Kim, Moonsu Cho, and Nojun Kwak. Proxy anchor loss for deep metric learning. 2020. 6

[2] Fengzi Li, Shashi Kant, Shunichi Araki, Sumer Bangera, and Swapna Samir Shukla. Neural networks for fashion image classification and visual search, 2020. 1

[3] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[4] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. 5

[5] Rishab Sharma and Anirudha Vishvakarma. Retrieving similar e-commerce images using deep learning, 2019. 2, 3

[6] Fan Yang, Ajinkya Kale, Yury Bubnov, Leon Stein, Qiaosong Wang, Hadi Kiapour, and Robinson Piramuthu. Visual search at ebay. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 2101–2110. ACM, aug 2017. 1