Modular Fashion Item Detection and Semantic Matching for Smart Outfit Retrieval

Mattia Roccatello
University of Padua
mattia.roccatello@studenti.unipd.it

Ulaşcan Akbulut
University of Padua
ulascan.akbulut@studenti.unipd.it

Ayşenur Oya Özen
University of Padua
aysenuroya.ozen@studenti.unipd.it

Abstract

This report presents a modular multi-stage system for fashion item detection and classification across heterogeneous datasets. Motivated by the challenges of incomplete category coverage in fashion datasets, our two-stage approach utilizes YOLOv11x object detectors, each trained on a dedicated dataset, to perform precise item-level bounding box detection for tops, bottoms, shoes, and headwear. In the second stage, a multi-head classification model predicts high-level semantic attributes. Furthermore, we develop a matching procedure that accurately associates clothing items detected from full outfits to different instances of them in other contexts. The model is trained using DeepFashion2, DeepFashion1, Shoes, Headwear, and Fashion Product Images datasets, achieving strong detection accuracy and reliable attribute classification for most of the classes. Results demonstrate that modular design combined with architectural extensibility and intelligent instance matching enables scalable and interpretable outfit parsing in complex visual settings. Notebooks can be found at this link. .

1. Introduction

Fashion item recognition in unconstrained visual scenes is a key challenge at the intersection of computer vision and retail AI. Understanding and parsing fashion images at the item level is fundamental, particularly in applications such as virtual wardrobes, outfit recommendations, and e-commerce systems. Full-body fashion images typically contain multiple garments and accessories with diverse appearances, occlusions, and poses, which makes it difficult for a single model or dataset to achieve comprehensive coverage.

Existing fashion datasets tend to suffer from incomplete category representation and label inconsistency. For example, DeepFashion2 offers detailed annotations for tops, bottoms, and outerwear but lacks coverage for accessories like headwear and footwear. Other datasets may include rich metadata but offer no object-level localization. These limitations hinder the development of scalable systems that aim to identify and describe all clothing components in a given image. To address these challenges, we propose a modular, two-stage pipeline composed of a detection model and a classification model. Each model requires a dedicated dataset setup due to the incompleteness and domain differences between existing fashion datasets. A significant amount of effort was dedicated to identifying, combining, and preprocessing the data to support our modular architecture.

The second stage performs semantic classification using a multi-head architecture. Here, features such as gender, color, fabric, pattern, clothing type, and usage are predicted using the outcomes of the detectors. Each classification head is tailored to a specific label domain (e.g., DeepFashion1 and Fashion Product Images), and an interleaved training strategy is used to deal with label mismatch across datasets.

Finally, we introduce the matching component, which leverages the learned encodings from the trained classifier to associate each detected item in the query outfit with its closest counterpart among stock-style images. This is achieved by comparing feature embeddings, enabling the matcher to identify visually and semantically similar items. For evaluation, we rely on the DeepFashion1 dataset, which provides both worn and unworn views with consistent attribute annotations, allowing us to verify whether the retrieved match aligns with the expected counterpart.

Together, these design choices result in a flexible and scalable architecture that generalizes well across heterogeneous data sources, performing well despite the strong im-

balance of data.

2. Related Work

Research on fashion understanding in computer vision has evolved significantly, focusing on tasks such as detection, segmentation, attribute recognition, and retrieval. Below, we highlight works closely related to ours and explain how our methodology differs.

DeepFashion2 and Match R-CNN Ge et al. [2] introduced the DeepFashion2 dataset and proposed Match R-CNN, a multi-task architecture that performs detection, landmark estimation, and retrieval simultaneously. While comprehensive, this method is tightly coupled with DeepFashion2's label space and lacks flexibility to extend to new categories like shoes or headwear. In contrast, we use modular YOLOv11x detectors trained on distinct datasets, each covering specific clothing types.

ModaNet and Attribute Segmentation ModaNet [6] offers fine-grained segmentation of fashion images, facilitating pixel-level attribute analysis. However, it lacks metadata like gender and usage, limiting its use in commercial attribute modeling. Our work avoids pixel-wise segmentation and instead focuses on bounding box detection followed by classification using rich structured metadata.

Fashionpedia and Ontological Labeling Fashionpedia [3] introduced an ontological taxonomy of garments and attributes. It inspired our multi-label taxonomy design, particularly in structuring hierarchical categories. However, unlike Fashionpedia, we prioritize detection and classification over segmentation, aiming for efficiency and scalability.

DeepFashion1 and Attribute Recognition Liu et al. [5] developed DeepFashion1 with a focus on predicting visual attributes like neckline and sleeve type using full-body images. Our pipeline enhances this by performing object-level cropping via YOLO and retrieving the right attributes of each detected item.

Contrastive Learning with SimCLR Chen et al. [1] proposed SimCLR, a self-supervised contrastive learning framework that improves visual representation quality. We incorporate SimCLR-based pretraining to improve the robustness of our classifier, especially when labeled samples are scarce or imbalanced.

YOLO-based Detection in Fashion YOLO models have been adopted in fashion detection pipelines for their speed and accuracy. For instance, Lee and Lin [4] trained a YOLOv4-based detector on five categories from a unified dataset. In contrast, we employ a multi-model strategy using YOLOv11x detectors trained separately on DeepFashion2, Shoes, and Headwear datasets, ensuring complete coverage and category specialization.

Key Distinctions Our work introduces several novel aspects. First, we use a **multi-detector YOLOv11x setup** to handle incomplete and disjoint label spaces across datasets. Second, we implement a **file-based matching mechanism** to assign labels at the instance level without manual intervention. Third, our classifier combines **global and local features** using a **multi-head architecture** that supports multiple attribute types simultaneously. Finally, we leverage **SimCLR-based contrastive learning** to pretrain our backbone encoder, enhancing generalization in low-supervision settings. These design choices enable robust, scalable, and interpretable fashion item understanding.

3. Dataset

Each stage of the two-part pipeline demands a custom dataset configuration, due to gaps and domain mismatches across available fashion datasets. Considerable work was invested in selecting, merging, and preparing the data to enable our modular approach.

3.1. Detection Data

The detection task (Model 1) aims to localize multiple clothing items in full-body images using the prediction of the bounding box. For this purpose, we employed three datasets: DeepFashion2, a Headwear dataset, and a custom Shoes dataset.

DeepFashion2 was selected as the primary dataset, as it is one of the few large-scale fashion datasets offering bounding box annotations. It contains over 800,000 clothing items in 425k images with labels spanning 13 categories, alongside segmentation masks, keypoints, and clothing attributes. We merged similar categories into five superclasses: top-wear, bottom-wear, dress, outerwear, and vest. Despite its richness, it lacks annotations for certain item types such as shoes and hats. To address these limitations, we included the following independent datasets:

Headwear (Cap) dataset includes bounding boxes for hats, extracted from XML Pascal VOC files and converted to YOLO format. It consists of approximately 671 images with various bounding boxes.

Shoes Dataset was **manually annotated** and includes 12,000 images labeled in YOLO format under a single class of footwear (the original ones were: Boots, Flip Flops, Loafers, Sandals, Sneakers, and Soccer Shoes).

Given that each dataset covers only a subset of relevant clothing categories, we adopted a modular approach where

one YOLOv11x model was trained per dataset. Merging the datasets into a single training set without addressing their annotation gaps would have led to incomplete supervision. As a result, the model would fail to learn consistent detection across all classes, severely hindering its ability to recognize the full range of clothing items. All three models were fine-tuned to our specific label set.

3.2. Classification Data

For the classification task (Model 2), which aims to predict attributes of the clothing items, we used a combination of the following datasets:

DeepFashion Dataset (DF) includes detailed visual attributes (e.g., fabric and pattern), but it does not include labels regarding usage or color of the items.

Fashion Product Images Dataset (FP) complements this by providing more than 44,000 product-level images sourced from e-commerce websites, labeled with structured metadata fields such as gender, articleType, baseColour, and season.

The final set of selected features to be predicted by Model 2 is summarized in the table below.

Feature	Source Dataset
Gender	FP & DF
Clothing Type	FP & DF
Usage	FP
Color	FP
Fabric	DF
Pattern	DF

Table 1. Final selected features for Model 2 and their source datasets.

3.3. Data Volume and Preprocessing

In total, the detection task was trained on approximately 200,000 images from DeepFashion2 (364,676 annotated items), 670 from the Headwear dataset, and 12,000 manually annotated samples from the Shoes dataset. For classification, around 194,000 filtered samples from DeepFashion2 and Fashion Product Images dataset were used.

A unified preprocessing pipeline was applied to all datasets. For detection, bounding boxes were converted to YOLO format with normalized coordinates. Samples with corrupted or low-resolution content were removed. Each dataset was randomly split into training (80%) and test (20%) subsets, ensuring category balance.

For classification, metadata files were parsed and merged with image paths. Labels were normalized across datasets (e.g., merging "t-shirt" and "tee"), and rare or inconsistent entries were discarded. Images were resized to 256×256 pixels to prevent GPU memory issues.

Dataset	# Images
DeepFashion2	≈400 000
Headwear	≈671
Shoes	≈12 000
Detection Total	≈437,000
Fashion Product Images	≈44,000
DeepFashion	≈10 000
Classification Total Filtered	≈51 000

Table 2. Summary of image counts used in the detection and classification stages.

4. Method

4.1. Pipeline

In this part, the methodology is explained with respect to models and datasets used in clear order.

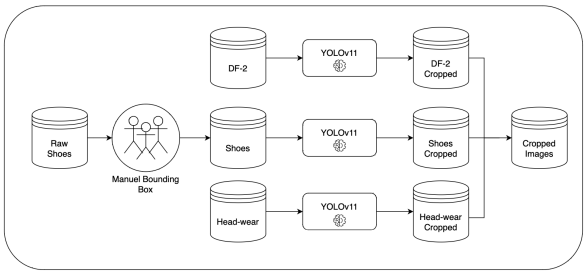


Figure 1. Pipeline Part 1

4.1.1. Part I: YOLOv11 Bounding Box Detection

The first stage of our pipeline consists of training and evaluating the three detector models, each trained on the single datasets DeepFashion2 (Top-wear, Bottomwear, Outerwear, Dress, and Vest categories), Cap-dataset (Headwear), and Shoes-dataset (Footwear). All detectors use the YOLOv11x (the biggest available) architecture from Ultralytics, fine-tuned with the same training configuration, since it's optimized to deal with various datasets. YOLOv11 is a fast, accurate, single-stage detector with a CNN backbone, FPN neck, and multi-head prediction layer, well-suited for real-time clothing detection. Augmentations and learning schedules were handled automatically by the YOLO training pipeline.

4.1.2. Part II: Dataset Preparation for the Classifier

This step sets a bridge between the first part and the classifier. Here, the dataset is being prepared for the classification algorithm, which is explained in the next step.

Component	Configuration
Model	YOLOv11x (pretrained on COCO)
Image Size	256×256
Epochs	57 (DeepFashion2), 20 (Cap - dataset), 150 (Shoes - dataset)
Batch Size	32 (auto-adjusted for memory)
Optimizer	SGD (momentum = 0.937)
LR Scheduler	Cosine Annealing
Losses	CIoU (bbox), BCE (objectness), CE (classes)
Validation Metrics	Accuracy, precision, recall, mAP@0.5, mAP@0.5-0.9
Data Augmentation	Mosaic, HSV, Flip, Affine
Logging	Weights&Biases
Early Stopping	Enabled

Table 3. YOLOv11 training configuration used across all datasets.

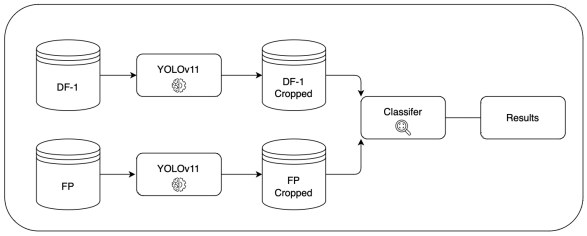


Figure 2. Pipeline Part 2-3

To train the classifier, a curated dataset of cropped single clothing items was created by applying the three YOLOv11 detectors to the two selected datasets: Fashion Products Dataset (FP) and DeepFashion1 (DF1). Each image crop was paired with its corresponding semantic labels (e.g., gender, fabric, usage), requiring specific handling for each dataset due to differences in structure and annotation quality.

In FP, each image corresponds to a single clothing item, which simplifies the label association. However, we still performed the described checks to ensure precise alignment: for each detection, we compared the predicted clothing type (e.g., "Topwear", "Bottomwear") with the dataset's articleType label to avoid mismatches between bounding box location and semantic label. If multiple detections matched the article type, we retained the one with the highest confidence score. Some fields were discarded or merged to ensure consistency and reduce class imbalance: Year and Season have been discarded because considered unreliable and noisy; masterCategory and subCategory have been discarded because too coarse for fine-graned classification (articleType was kept and called Clothing Type); Gender had Unisex, Boy and Girl filtered out to align with DF's binary gender classification; Color label had broader color categories merged to reduce imbalance; Usage was cleaned out of rare and ambiguous categories like Home and Travel.

Label association in DF1 required a more advanced strategy, as each image often depicts multiple clothing items with overlapping annotations: If only one item was detected in the image, we ensured the predicted label (e.g., "bottomwear") was semantically compatible with the label in the dataset (e.g., rejecting a "t-shirt" label for a detected "Bottomwear" bounding box); When several detections had the same predicted class (e.g., multiple tops), we selected the one with the highest confidence. In cases where multiple detections predicted different clothing types, we used semantic similarity to associate each detection with the most plausible label in the annotations. Detections that could not be reliably matched to a label were discarded to maintain label integrity.

Once the DF1 and FP datasets were cleaned, processed, and merged, final statistics were generated to understand the distribution of features across the full dataset. These statistics, also available in the notebook, provide insight into any class imbalance and help validate the quality of the merged dataset. For training purposes, the full dataset was split into training (80%) and validation (20%) sets.

4.1.3. Part IIIa: The Classifier

This part introduces the classifier algorithm for the cropped image dataset that is created in Part 2 (see Fig. 2). Here, the idea is to classify cropped images using a dedicated multi-label neural network. Several experiments, briefly summarized in the next section, have been carried out to select the best strategy to achieve our goal. The major issue we have faced was the tendency of the model to overfit since the very first iterations. In terms of average accuracy across all the classes to be predicted, here is a summary of the best model we came up with.

Our final model is based on a ResNet101 backbone, pretrained with self-supervised learning, used to extract deep features from each cropped item. A unique feature of our approach is that we concatenate the predicted clothing type label (obtained from the detector) as an additional input alongside image features. This allows the model to condition its predictions on semantic cues.

The model includes a fully connected multi-layer block followed by independent heads, one for each semantic label (e.g., gender, fabric, usage). A flexible interleaved learning scheme was implemented: for each item, only the relevant heads are updated based on the source dataset (DF or FP). To handle strong class imbalance, we combine masked cross-entropy and Focal Loss, and apply class-specific weighting.

For robustness, we introduced a LightDegradation data augmentation module, simulating JPEG compression, blur,

Component	Description
Backbone	ResNet101 pretrained with self-supervised learning
Input	Cropped images + predicted label from detectors
FC Layers	[1024 + 1 $\rightarrow$ 512 $\rightarrow$ 512] with BatchNorm, LeakyReLU, Dropout, where +1 is the predicted label value from the applied YOLO detectors
Task Heads	6 linear heads for: gender (2), type (32), usage (4), color (13), fabric (7), pattern (7)
Loss	Focal Loss + Masked Cross Entropy with class weights
Augmentation	Blur, brightness/contrast shift, JPEG compression
Training Strategy	Interleaved learning based on dataset origin (FP or DF)
Optimizer	Adam: LR = $1e{-4}$ (heads/FC), $5e{-5}$ (backbone)
Epochs	20
Logging	W&B + per-task confusion matrices

Table 4. Classifier model summary.

brightness/contrast shifts to improve generalization under real-world noise.

One of the most critical challenges was improving validation accuracy for the Clothing Type class, which is particularly difficult due to its large number of fine-grained, imbalanced categories. To support the model in distinguishing between visually similar yet semantically distinct types, we enriched the input by combining the cropped item image with its predicted category label from the YOLO detector. This auxiliary input serves as contextual guidance, injecting prior knowledge derived from the original full-outfit detection phase. Additional difficulties arose with the Color, Fabric, and Pattern classes, which rely heavily on subtle visual cues within the image. Pre-trained backbones such as ResNet and ConvNeXt, despite their general visual competence, struggled to extract sufficiently rich representations for these appearance-centric attributes. To overcome this limitation, we trained a ResNet101 encoder from scratch in a self-supervised manner, tailored specifically to our domain—a process detailed in the following section.

4.1.4. Part IIIb: The Encoder

Residual connections in ResNet101 architectures are particularly effective for learning features such as color, Fabric, and texture, as they allow gradients to propagate deeply through the network without vanishing and preserve high-frequency details, essential when distinguishing textures or subtle color shifts. SimCLR (Simple Contrastive Learning of Representations, [1]) was chosen to allow the model to learn semantic similarity without supervision. It achieves this by projecting augmented image pairs into a shared representation space using a two-layer MLP head, trained with the NT-Xent loss, which pulls positive pairs (different augmentations of the same image) together while pushing apart all other images in the batch.

To ensure the final representations preserved color fidelity, augmentations involving color jitter and grayscale conversion were explicitly disabled, unlike standard SimCLR pipelines. The model was trained for 30 epochs using 256×256 image crops and a batch size of 32, with two augmentations per image, including random horizontal flip and rotation. Once trained, the projection head is discarded, and the ResNet backbone is used to generate embeddings. These were evaluated using K-Nearest Neighbors in the latent space, confirming that visually similar garments clustered closely, even without labels.

Component	Value
Backbone	ResNet101 (ImageNet, headless)
Proj. Head	MLP: 2048 $\rightarrow$ 2048 $\rightarrow$ 512
Loss	NT-Xent (contrastive, temp. scaled)
Batch Size	32
Image Size	256×256
Epochs	30
Augmentations	Flip, rotation, masking
Color Aug.	**Disabled** (to retain color cues)
Optimizer	SGD (lr=2e-2, mom=0.9, wd=5e-4)
Scheduler	Cosine annealing
Validation	KNN on frozen embeddings
Framework	PyTorch Lightning + W&B

Table 5. Self-supervised training of the encoder weights

The saved weights have then been loaded into the backbone of the classifier, and slow learning (small learning rate) was allowed in order to adapt the encoder to the new supervised task.

4.1.5. Part IV: The matching strategy

The final stage of our pipeline is a semantic matcher that associates the cropped clothing items detected in a query outfit with their corresponding images in a user's virtual wardrobe. The intuition is that visually similar garments should lie close in a learned embedding space. Initially, the self-supervised SimCLR encoder alone provided meaningful representations, but to incorporate semantic cues, we enhanced the model with supervision by utilizing the last 512-dimensional embedding layer of the multi-task classifier. Matching is performed via cosine distance in this learned feature space. To validate matching, we leverage the structure of the DeepFashion1 dataset, where the same clothing item is photographed in multiple poses. Specifically, we use images labeled with pos = 'flat' as virtual wardrobe references and those with pos different from 'flat' (e.g., on models in different poses) as query outfit items. To extract these images from the datasets, we exploited their

naming structure: images in DF1 are uniquely identified by a triplet: (id, n1, n2). A single outfit on a model is indexed by (id, n1), while positions vary by n2. Ground truth associations are defined between flat and worn versions using (id + n1 + type) as keys. After resolving these associations, we extracted 227 unique clothing items with both flat and worn versions, and computed embeddings for each using the enhanced classifier. For each query item (worn pose), we searched for its true wardrobe counterpart (flat pose) among the k nearest neighbors in the entire encoding set. The success rate improves as k increases:

Top-k	**Accuracy**
1	37.36%
2	48.28%
3	56.90%
4	63.79%
5	67.24%

Table 6. Matcher performance by varying the number of neighbors k considered to find the ground-truth encoding of the pos = 'flat' counterpart of query items.

5. Experiments

Training the YOLO models was relatively straightforward due to the availability of robust pre-trained weights. The primary experimentation focused on tuning computational parameters—such as batch size and image resolution—to avoid GPU memory overflow. For the Caps dataset in particular, we ran additional trials to determine an optimal number of training epochs, aiming to prevent overfitting. The final choice was guided by consistency across multiple validation metrics. A summary of the selected configuration is provided in Table 3.

The design and tuning of the classifier architecture required significantly more experimentation. While the foundational structure was clear from the beginning, multiple iterations were conducted to refine it. To keep the presentation concise, we report only the average validation accuracy across all classes in the table below, although detailed evaluations—including per-class precision, recall, loss curves, and confusion matrices—were analyzed at each step. Each row in the table represents a key architectural or training modification from the configuration in the row above. Only the most relevant configurations are shown to avoid unnecessary clutter. The starting structure was composed of: double input, whole image and cropped item, to inject contextual information (later proved to be irrelevant and slowed down the processing), augmentation - not for colors -, Resnet18 backbone, 2 FC layers, no regularization, learning rate of 1e-5, standard Cross Entropy loss, multi-head.

Architecture	**Val. avg. acc.**
Starting structure	0.35
ResNet50, 3 FC layers, Dropout	0.44*
BatchNorm, masked loss	0.46*
SimCLR ResNet101 for custom pre-training, 2 FC layers, no Dropout, Input: cropped images	0.29
3 FC layers, Dropout, weight decay	0.31
Input: cropped images + YOLO predicted label	0.33
Backbone unfreezed: slow training	0.38
Reduced regularization (Dropout and L2)	0.41

Table 7. Summary of the tried configurations for the classifier, metric: average validation accuracy. The accuracies with a * refer to models whose accuracy was higher, but due to the outperforming class of Gender with a near 1.0 accuracy, but characterized by strong overfitting on the others. With the custom pre-trained backbone and stronger regularization, overfitting was mitigated, but the overall performance lowered.

6. Conclusion

By training individual YOLOv11x models on dataset-specific clothing categories, we address the issue of incomplete and disjoint annotations across standard datasets. On top of detection, we explored multiple strategies to address the complex task of attribute classification. While the final classifier performs robustly on core categories, it still struggles with visually subtle features such as color and fabric, which remain open challenges due to their ambiguity and dependence on lighting and image quality. The matching component, powered by learned feature embeddings, proves effective when using multiple neighbors for similarity retrieval. However, its performance remains inherently tied to the quality of attribute classification, reinforcing the need for strong foundational predictions.

Together, these modules compose a flexible and scalable framework for fashion item detection and retrieval, paving the way for advanced applications such as smart wardrobes, visual outfit search, and personalized fashion recommendations. Future improvements may include integrating segmentation-aware modules and exploring transformer-based backbones to further enhance visual understanding.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119, pages 1597–1607, 2020.

[2] Yuying Ge, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5337–5345, 2019.

[3] Zhenyun Jia, Yujun Wang, Yinan Liu, Jimei Lin, Xiaohui Shen, Xin Yang, Jonathan Brandt, and Alan L. Yuille. Fashionpedia: Ontology, segmentation, and an attribute localization dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11537–11546, 2020.

[4] Kuan-Chuan Lee and Hsin-Ying Lin. A two-stage approach for clothing detection using yolov4. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2, 2021.

[5] Ziwei Liu, Ping Luo, Sheng Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1096–1104, 2016.

[6] Sijie Zheng, Liang Zhang, Jiaxin Song, Zhe Huang, Xinyu Cao, and Chen Change Loy. Modanet: A large-scale street fashion dataset with polygon annotations. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1670–1678, 2018.

Appendices

A. Supplementary Figures

Here we collect a list of supplementary figures useful to elucidate some aspects mentioned in the main text. Their purpose is to show how datasets are composed and to better explain the choices we made in selecting the models.

A.1. Overview of the Train Datasets used for the YOLO Detectors

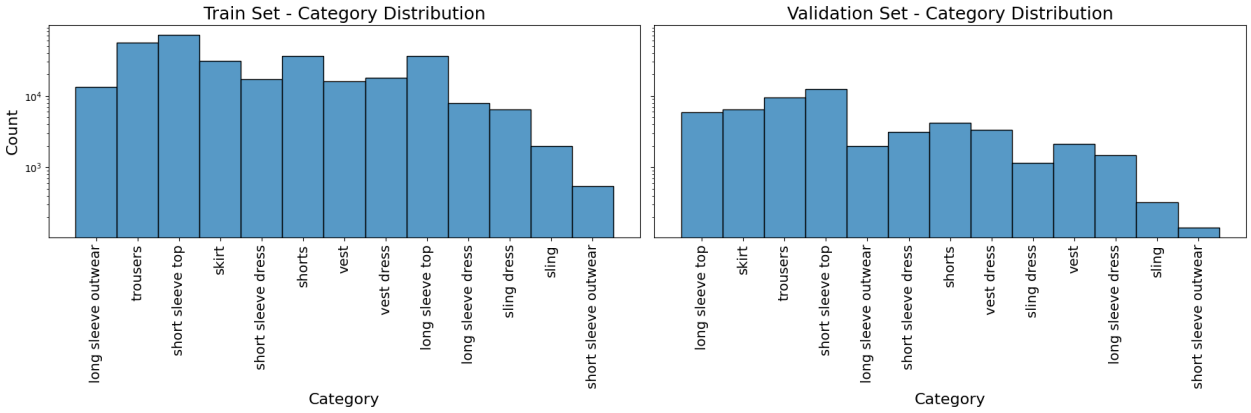


Figure 3. Labels Distribution of DeepFashion2 dataset for training and validation set

Figure 4. An example of the annotated bounding boxes - golden labels

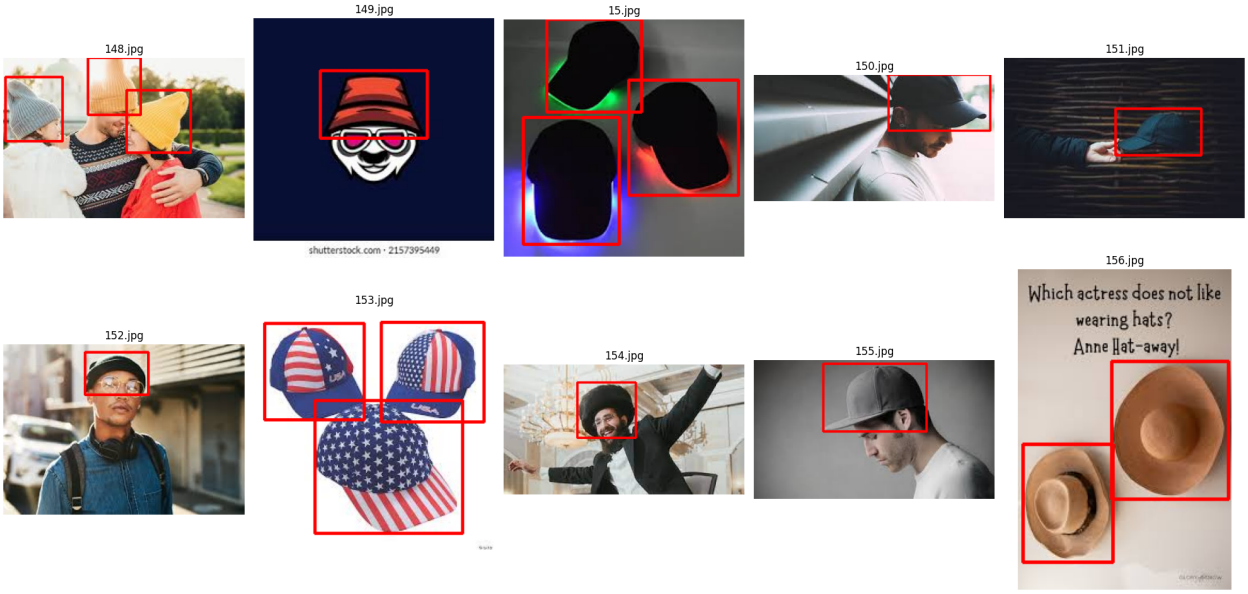


Figure 5. An example of the annotated bounding boxes - golden labels for the Caps-dataset

Figure 6. An example of the annotated bounding boxes - golden labels (manually placed) for the Shoes-dataset

A.2. Detectors performances

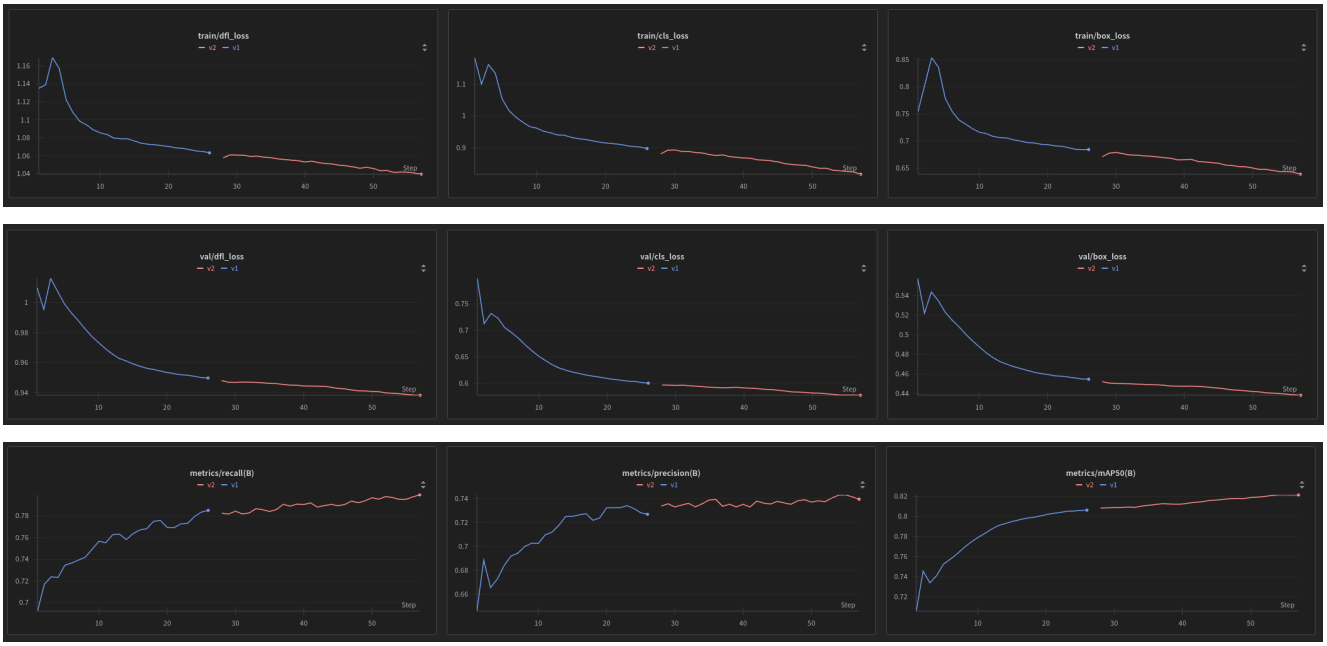


Figure 7. Performance YOLO detector trained on DeepFashion2 dataset. Top row: training losses; Central row: validation losses; bottom row: validation metrics. The two lines indicate that the training has been interrupted and continued at a second moment.

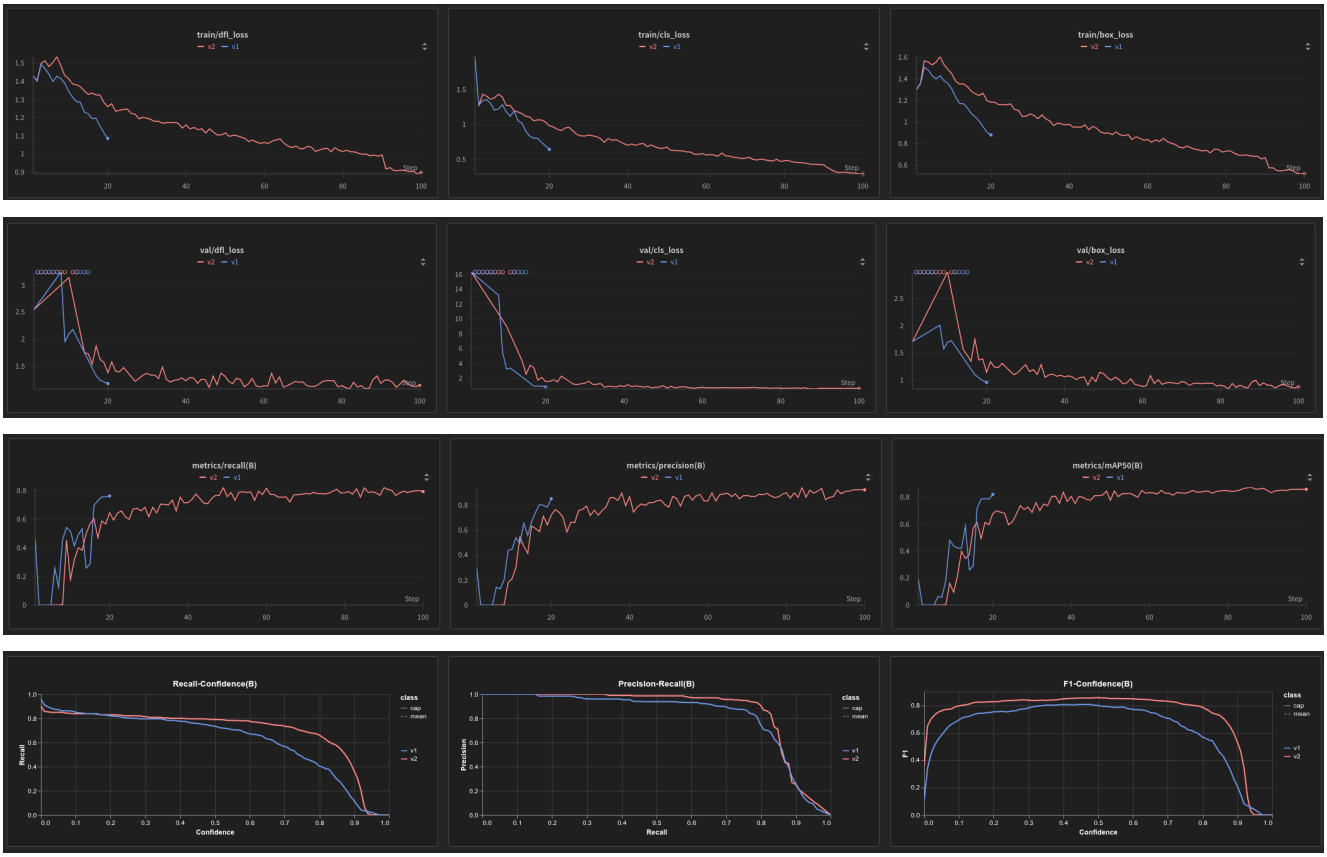


Figure 8. Performance of the YOLO detector trained on the Caps dataset. Top row: training losses. Second row: validation losses. Third and fourth rows: validation metrics. The red model, outperforming the other one for all metrics and not overfitting, is the chosen one.

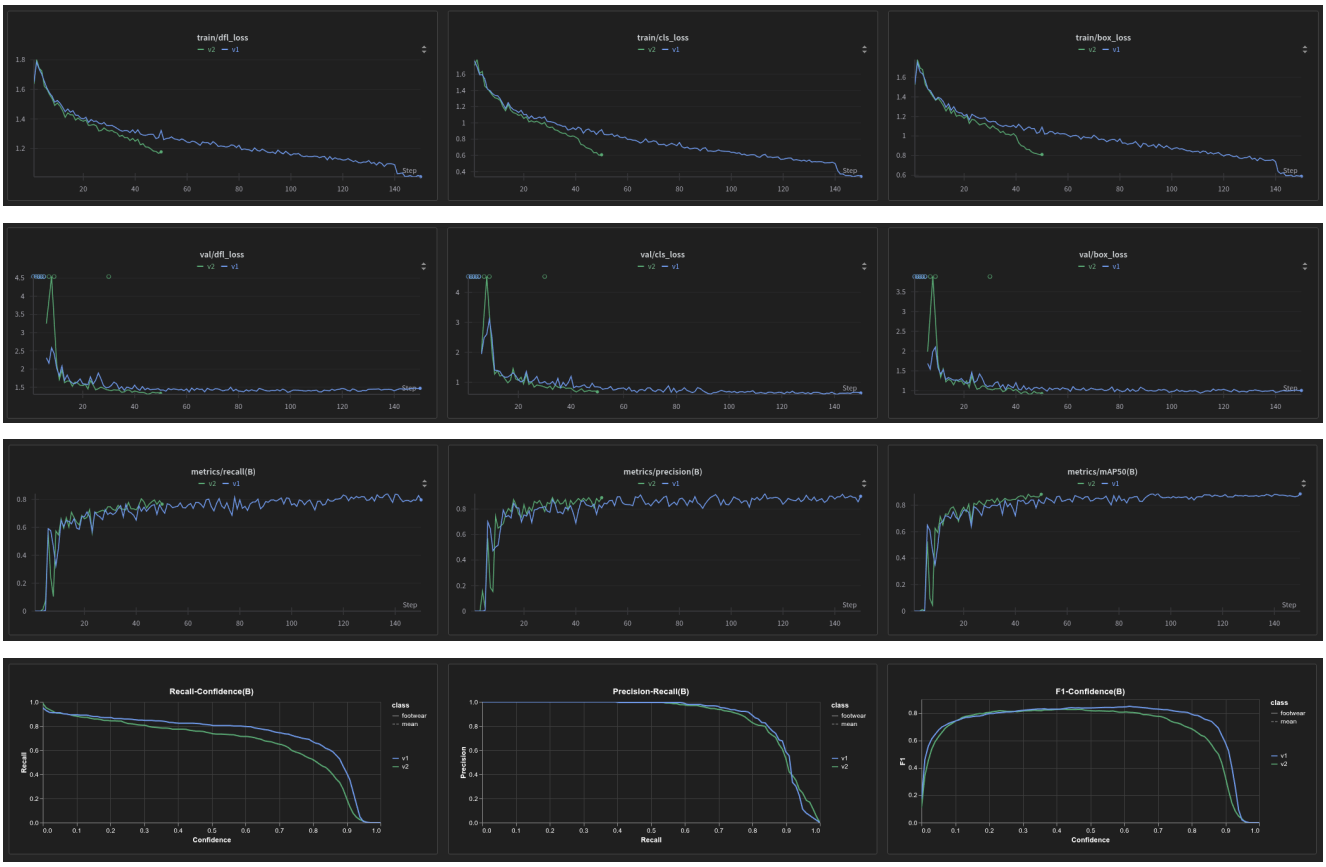


Figure 9. Performance of the YOLO detector trained on the Shoes dataset. Top row: training losses. Second row: validation losses. Third and fourth rows: validation metrics. The blue model is the one chosen.

A.3. Classifier Dataset Preparation (Fashion Product Dataset)

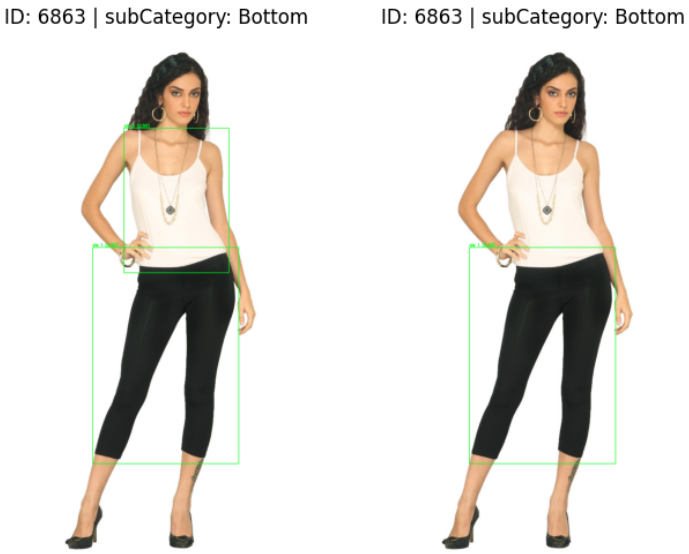
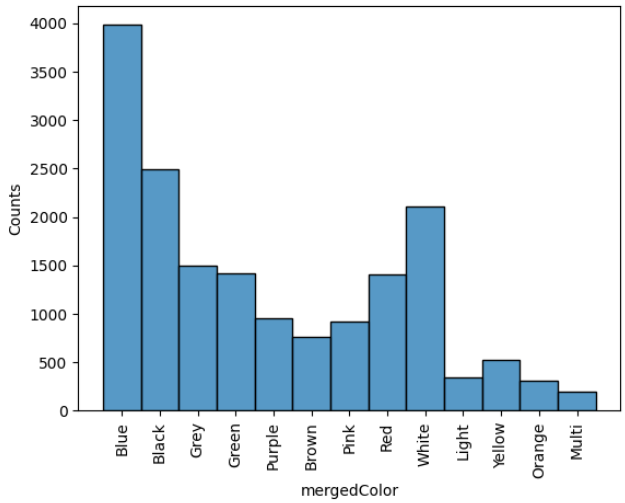
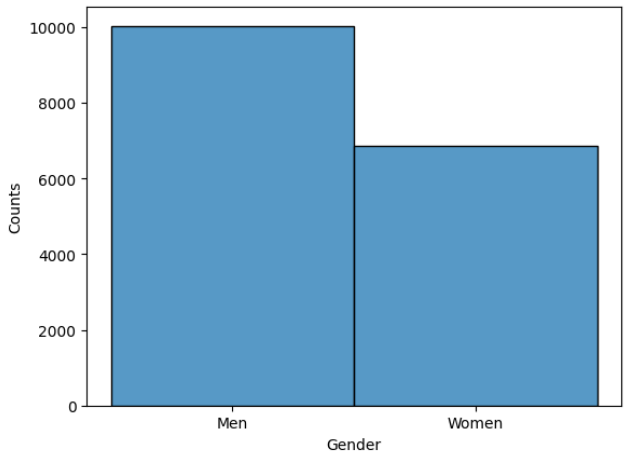


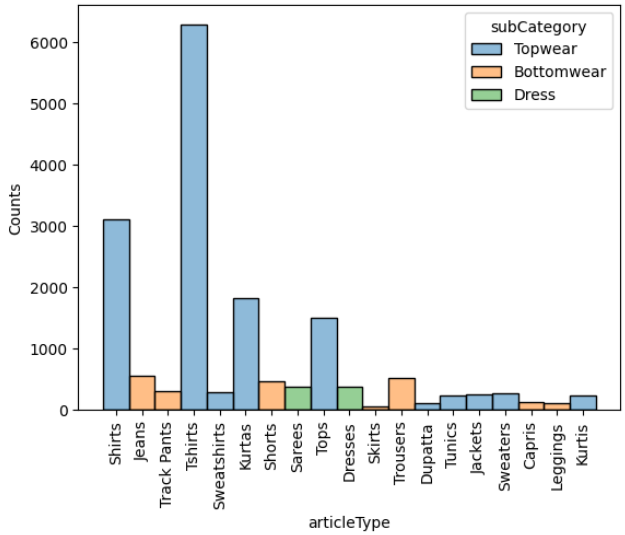
Figure 10. An example from the **Fashion Product Dataset**, showing a double detected bounding box. Matching the predicted label with the subcategory led to the correct labeling of the image.



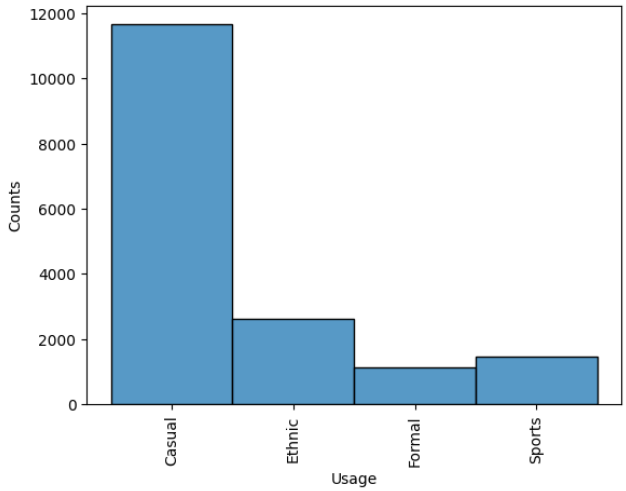
(a) (Merged)Color labels distribution



(b) (Cleaned) Gender labels distribution



(c) (Merged) Clothing type label distribution

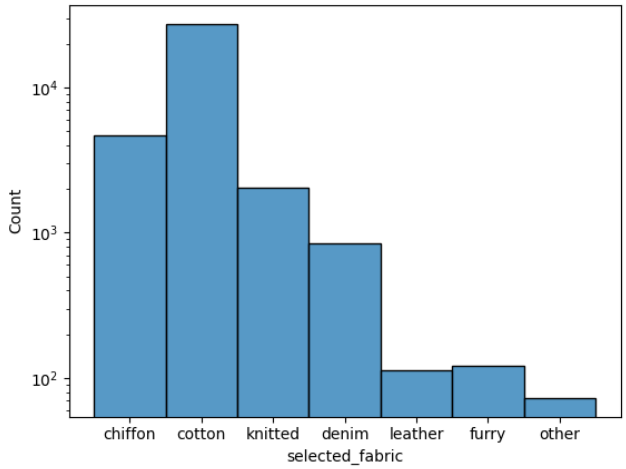


(d) (Cleaned) Usage label distribution

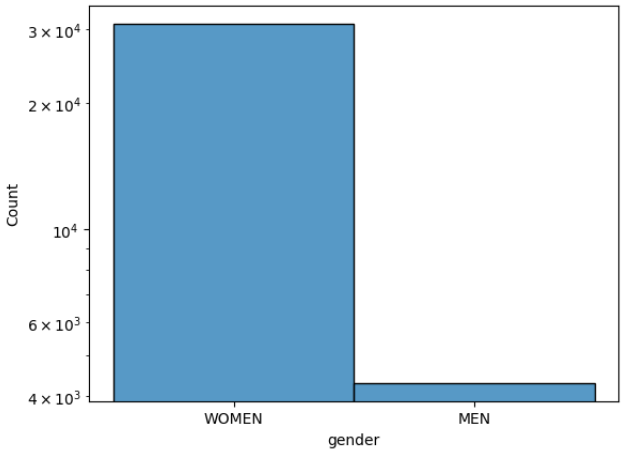
Figure 11. Label distributions retrieved from the cleaned **Fashion Product Dataset**.

A.4. Classifier Dataset Preparation (DeepFashion Dataset)

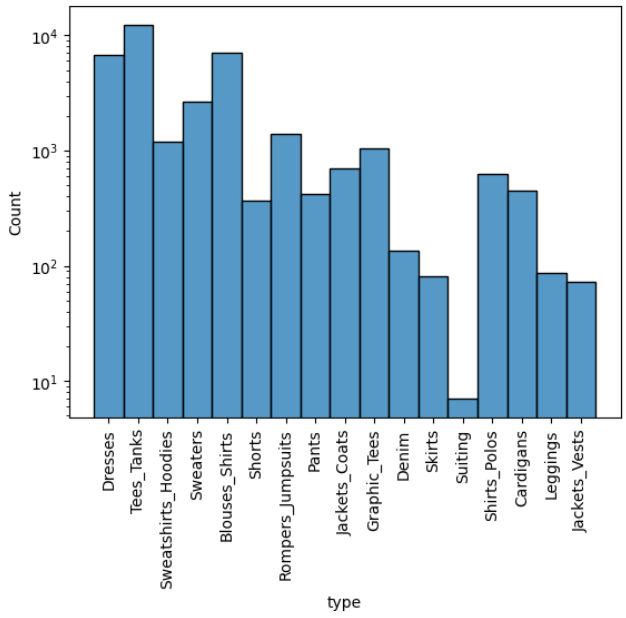
Figure 12. An example from the **DeepFashion Dataset**. Only one label ('Blouses-Shirts') corresponds with the shown bounding boxes. For each column, only the label associated with the top image was retrieved.



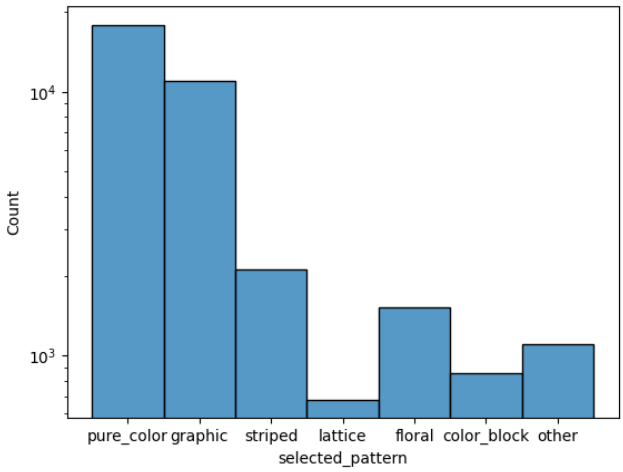
(a) Fabric label distribution



(b) Gender label distribution



(c) (Merged) Clothing type label distribution



(d) Pattern label distribution

Figure 13. Label distributions retrieved from the cleaned **DeepFashion Dataset**.

A.5. Classifier performances

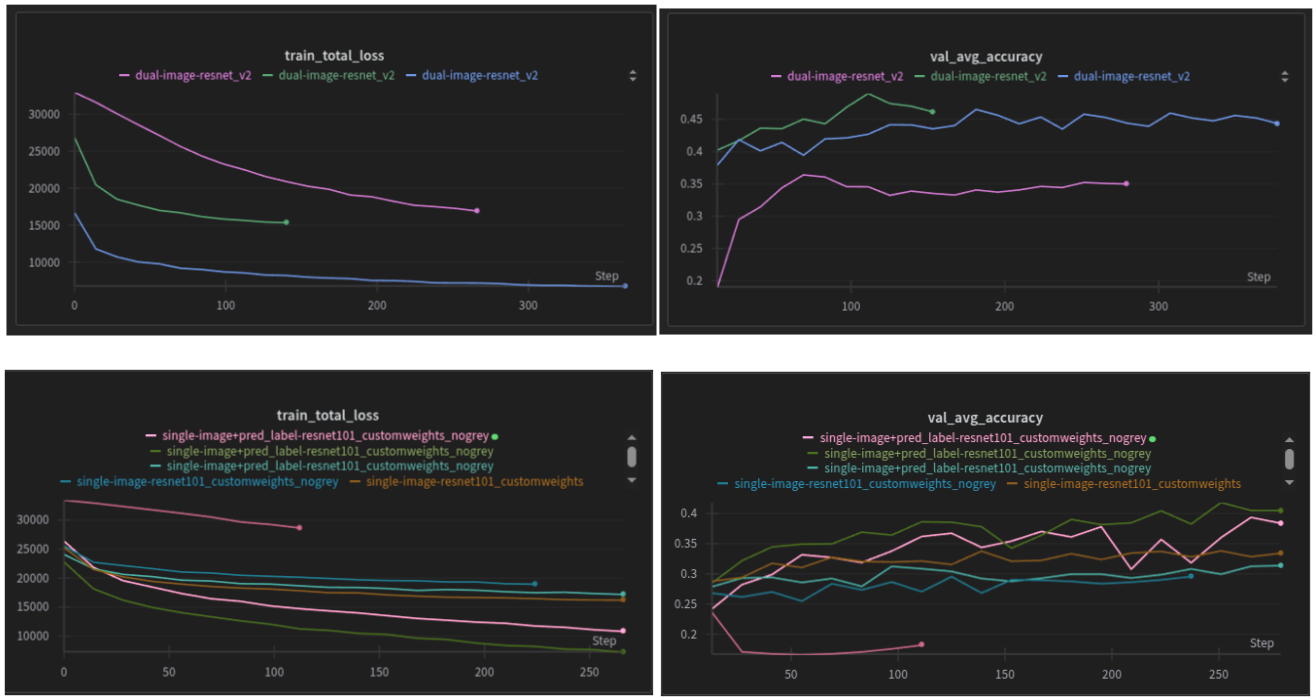


Figure 14. Performance of the classifiers tested. With reference to Table 7, we have as top images the total losses and average accuracies corresponding to the first 3 rows of the table (in order: pink, blue, and green). The images at the bottom show losses and average accuracies for the models receiving as input only the cropped images, and, eventually, the predicted label from the detector. The best model is the one in green. To avoid unnecessary cluttering, the other metrics are not shown here.

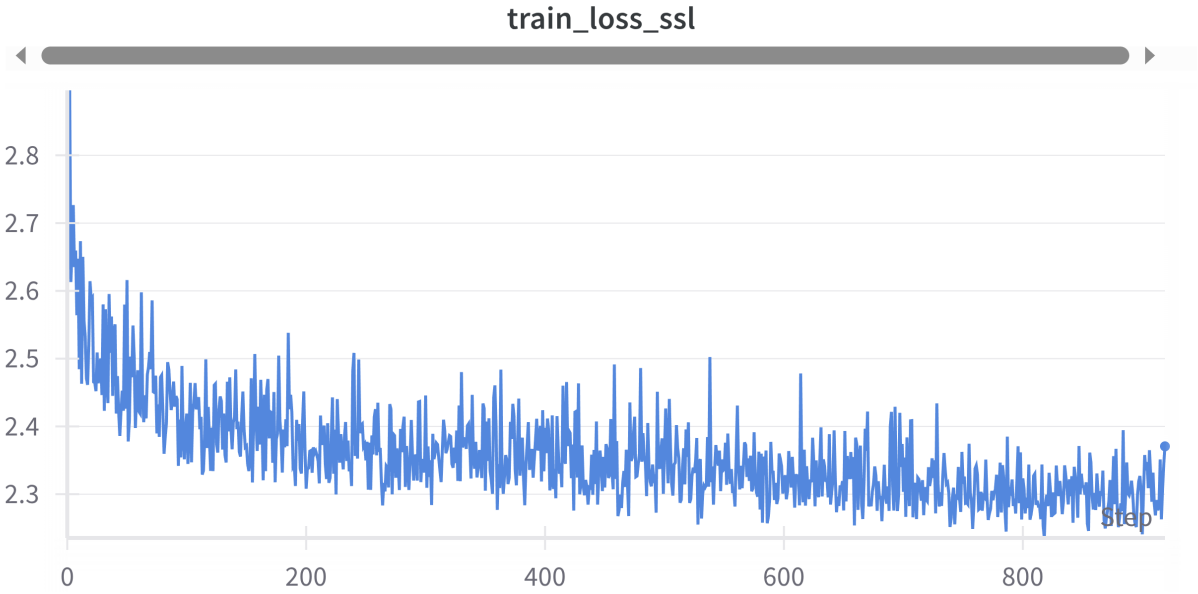


Figure 15. Loss function of the ResNet101 architecture training in a self-supervised way using the SimCLR implementation.

A.6. Matcher performance

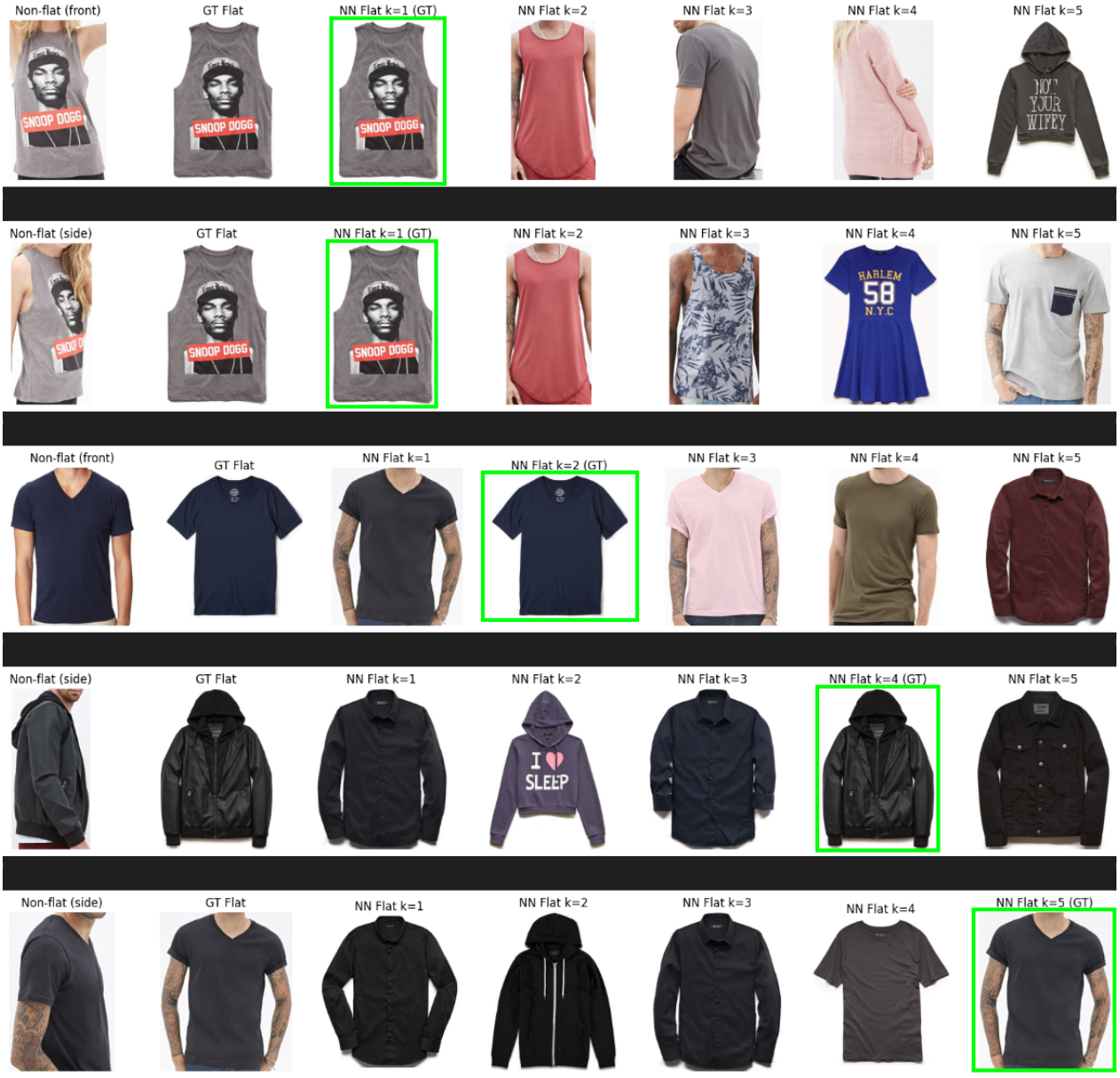


Figure 16. Example of the matcher performance when done on the DeepFashion Dataset as explained in the main text. Row-wise explanation: query image (cropped), ground-truth 'flat' version, first to fifth nearest neighbors in the encoding space. Highlighted in green are the retrieved images that correspond to the expected golden image.