# Supplementary for CVPR-CVFAD 2022 paper "DAtRNet: Disentangling Fashion Attribute Embedding for Substitute Item Retrieval"

## Supplementary overview

This supplementary document for CVPR-CVFAD 2022 submission entitled **DAtRNet: Disentangling Fashion Attribute Embedding for Substitute Item Retrieval** provides additional details for the end-to-end pipeline of the proposed attribute-aware substitute recommendation system which are augmented with the proposed Disentangled Attribute Representation Network (DAtRNet) architecture. Also, in this document, the details of hyper-parameter tuning, t-SNE visualization, visual examples and the class activation maps generated by separately using the height and width attention are given.

We have provided the block diagram describing the end-to-end pipeline for attribute-aware substitute recommendation in Figure 1. As shown, the entire pipeline can be divided into five sub-parts: data pre-processing, deep CNN architecture, attribute representation, manipulated attribute and substitute recommendation. Out of these, the deep CNN architecture consists of DAtRNet designed using ASE module, which are discussed in detail in main manuscript. In supplementary, therefore, we elaborate on four other parts.

In section 1, we discuss the ways to obtain the data for three different search strategies employed in the main manuscript. Since, each of these strategies are based on different kinds of attribute manipulation, the generation of inference data is different for them. Moreover, we explain the method of creating the triplets to train the DAtRNet architecture.

In section 2, we illustrate the attribute-aware feature representation for the substitute recommendation system. Here we describe the process to extract the super-category specific attribute embedding from the style space and the way of incorporating these as the representation of the input image. Hence, we obtain the method of getting an image as an aggregation of a set of attribute features in the style space.

In section 3, we elaborate on the attribute-aware substitute recommendation, emphasizing the way to accommodate the desired attribute instructions in the feature representation in style space and retrieve top-$k$ products.

In section 4, we present the extensive study of several hyper-parameters, In 4, we observed the change in performances with the variations in the image dimension, the kernel size and the number of filters in the first *Conv* layer (stem).

In section 6, we provide the experimental result of DAtRNet without and with attribute manipulation with respect to $k$ and compare with the state-of-the-art methods. In section 9, we demonstrate the impact of number of manipulated attributes on the performance of attribute-aware substitute recommendation.

In section 5, we demonstrate the class activation maps of the fashion images by separately applying the height and width attention. From these visualization, we aim to depict that these attention modules localize different regions of the fashion images, thereby their combination can encapsulate the holistic feature representation. Similarly, in section 10, we depict the t-SNE plots [6] for the six super-categories of DeepFashion dataset [5], obtained using DAtRNet.

In sections 7 and 8, we provide the experimental results of DAtRNet with product-category specific representation and class-specific substitute recommendation results w.r.t. $k$ with one attribute manipulation. Finally, in section 11, we illustrate extensive visual examples and failure cases of all the tasks considered by us.

## 1. Data pre-processing

The attribute-aware fashion search is a novel way of incorporating customer feedback to enhance the recommendation experience. However, there is no existing dataset with the annotations required for this operation. Hence, for our perusal, we have updated the annotations given in the two attribute recognition datasets, namely DeepFashion [5] and Shopping100k [2]. These datasets were chosen for our application due to large number of fine-grained attributes, large number of fashion product images with visually-similar overlapping features and the class-imbalance problem.

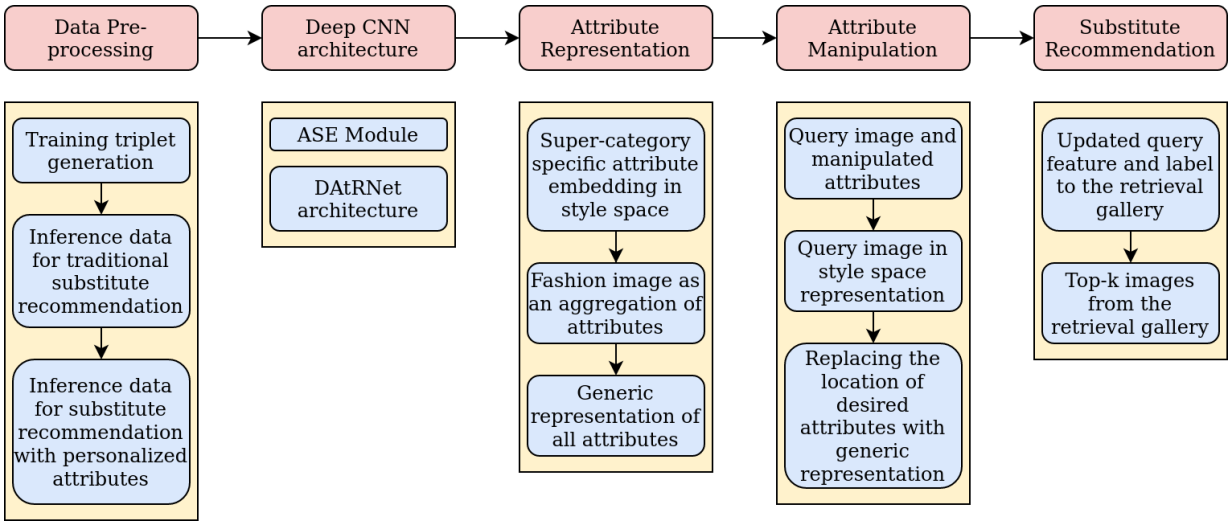



Figure 1: The end-to-end pipeline for substitute recommendation with attribute manipulations using DAtRNet as the network. Here, the blocks at the top represent the five primary parts of the pipeline. Each of these parts are further sub-divided into multiple secondary steps given below these blocks. The arrow between the primary and secondary steps depict the flow of operation.

### 1.1. Training triplet generation

The strategy of training triplet generation for our application is to ensure the images with same attributes to come closer and images with different attributes to push apart, irrespective of the visual similarity for attribute-specific style embedding generation. To perform this, we generate separate set of triplets for each super-category to embed the features from the corresponding super-category. Also, the positive and negative images of an anchor is generated in random to ensure that the DAtRNet can ignore the visual similarity detail of two products and instead focus on the attribute-level similarity. For example, the network can obtain one blue colored T-shirt as anchor, one blue trouser as positive and one red T-shirt as negative. This is done to ensure that the network learns the features pertaining to the color attributes, *not* the features of the entire shirt as a whole. Also, we generate multiple triplets considering each image as an anchor, which enables to create more presence for images containing minority classes.

### 1.2. Inference data for traditional substitute recommendation

For traditional substitute recommendation, we need a set of query image with the ground-truth attribute labels. For our experiment, we have considered 4,000 query images. During inference, we consider the $768-$dimensional feature vector of the query image with the ground-truth attributes and find the nearest $k$-images from the retrieval gallery. Also, note that, we consider attribute-level similarity for our application of substitute recommendation, unlike the image-level similarity performed by traditional substitute recommendation. Hence, our network does not look for the entire image, rather check the individual attributes to find similar products, much like the human visual system.

### 1.3. Inference data for substitute recommendation with one manipulated attribute

For our application, we have used the publicly available large-scale DeepFashion and Shopping100k datasets. However, none of these datasets are equipped with the attribute manipulation instructions. Also, we aim to obtain the super-category specific performance of our DAtRNet architecture. To accomplish this, we follow the steps given here. Firstly, we choose a super-category and an image with its ground-truth. Secondly, we alter the ground-truth of the super-category of the image with some other random value existing in the super-category. For example, if a super-category has 100 fine-grained attributes and the ground-truth of this category of the image is, suppose 10, then we consider a random number ranging from 1 to 100, except 10. Thirdly, to ascertain that the target ground-truth label is present in the entire dataset, we check the retrieval gallery and consider the desired attribute valid only if the target ground-truth is present in the gallery. This is important, because if the image with the target at-

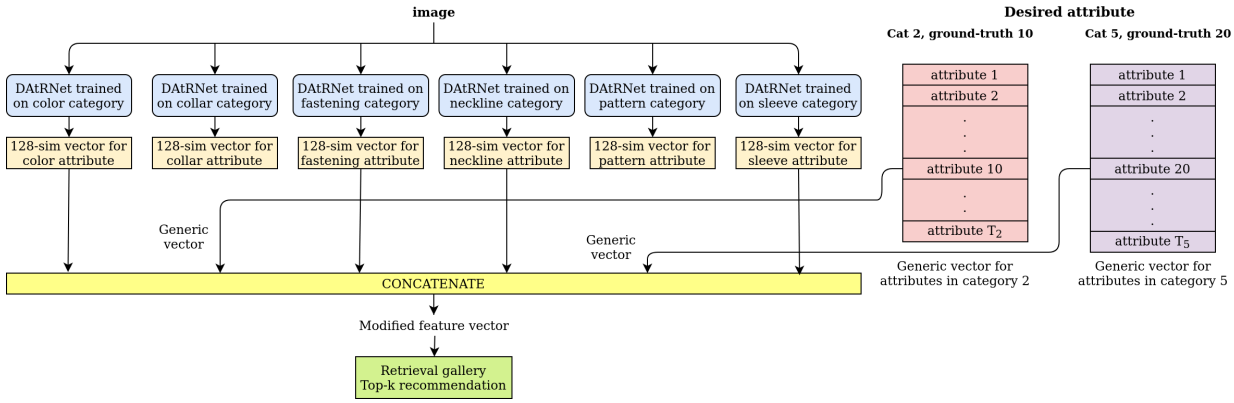



Figure 2: The method to generate the 768-dimensional fashion embedding with attribute manipulation. Here, the blue blocks represent the DAtRNet architecture, each trained using one super-category, pink and purple blocks represent the generic feature representation for each of the fine-grained attributes for category 2 (Collar) and category 5 (pattern), respectively.

tribute set is not present in the retrieval gallery, then the model will always incur failure in terms of exact similar-attribute search and it is not necessarily the fault of the proposed architecture but the shortage of products in the retrieval gallery. Finally, we perform this operation multiple times on each image for each super-category to obtain a large set of inference data. The large inference data can also enable to obtain a generalized performance.

### 1.4. Inference data for substitute recommendation with multiple manipulated attributes

The aim to observe the performance of DAtRNet using multiple attribute manipulation is to check its robustness under multiple changes in the feature representation. This work can be performed as a series of recommendation using a single attribute instruction, however the latency involved in the process does not reflect a very suitable solution, where the time consumed for recommendation increases linearly with the number of manipulated attributes. Therefore, we follow these steps to obtain ground truth with multiple attribute manipulations.

Firstly, we arbitrarily choose $k$ number of super-categories out of the six categories considered for each dataset and one image. Secondly, we visit the ground truth of the image selected by us in previous step and alter the ground-truths of the $k$ categories previously selected. The alteration for each of these $k$ categories is done by selecting one random value existing in the super-category, except the one already present. Thirdly, we check if any image with the target attribute set is present in the retrieval gallery. This is done to avoid getting false results due to absence of target image. Finally, we perform these steps for several times considering several $k$ values to ensure the creation of large number of inference data for substitute recommendation with multiple manipulated attributes.

## 2. Attribute Representation

Feature extraction is an important process for image recommendation methods. After feature extraction, representation of visual information in an images is a crucial step, in which we compile the information across all categories. This is a challenging step because this greatly affects the retrieval accuracy in recommendation systems. In this section, we explain how we extract category-level features and how we represent an image using these features.

### 2.1. Super-category specific attribute embedding in style space

Conventional feature extraction methods employ CNNs' in a single style space. However, for fine-grained super-category specific feature extraction this is not enough. Hence, we extract the features across multiple style spaces. For each image in the datasets, i.e. DeepFashion and Shopping 100k, a super-category specific unique model extracts a $128-$ dimensional feature vector. This feature vector contains the category-specific information present in the image. Feature extraction at super-category specific style space ensures to separate visually similar intra-category attributes much apart. Using independent models for each category is a crucial step in retrieval process, because it

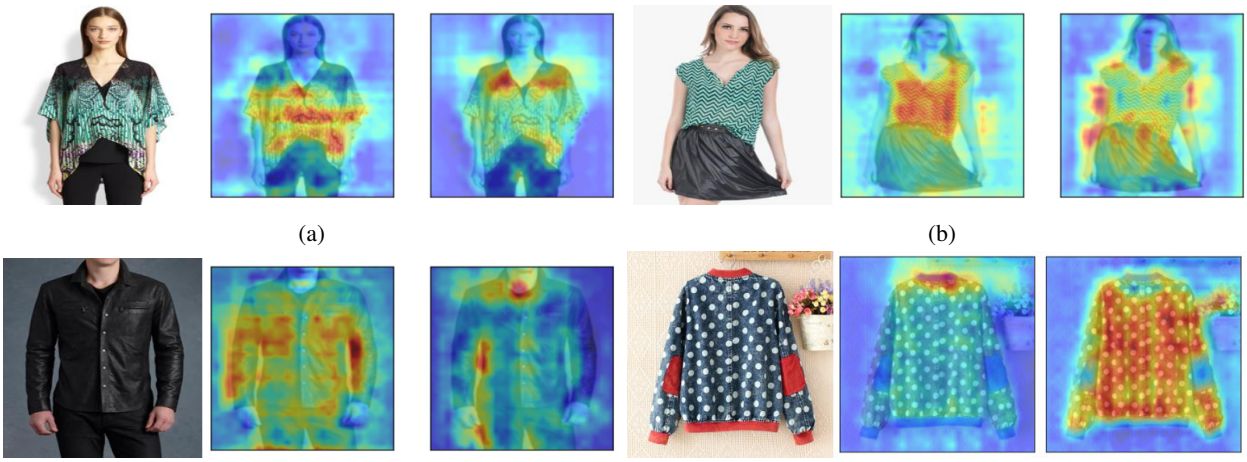



Figure 3: Class activation maps generated from the sample images from DeepFashion dataset. Here, the first image in the sequence of images (a)-(d) is the fashion product image, second image is the class activation map generated by considering width attention, third image is the class activation map generated by considering height attention.

helps to extract attribute information in a unique feature space thereby allowing to extract fine-grained attribute information for images. By separating intra-category specific attributes much apart we ensure to extract rich high-level features which help in providing fine-grained attribute information of the query image. This is achieved by using triplet-based training process. By minimizing the triplet loss we bring similar looking images together in category-specific sub-spaces. After this operation, we aggregate the information for images across all super-categories in the dataset. In Figure 2, we observe the feature extraction for all the super-categories from an image separately in order to obtain the feature embedding.

### 2.2. Fashion Image as aggregation of attributes

Aggregation of attribute information for images is a crucial step in retrieval process. The category-level attribute embeddings extracted for an image represents the visual information present in it. Compilation of these features determine how efficient the retrieval process is. After extracting feature embeddings at category-level we need a method which incorporated the image information across all super-categories. For this we adopt concatenation operation. We concatenate all the category-specific feature vectors resulting in a $768-$dimensional vector which is used as feature representation for an input image. This vector is used to find the most similar looking apparel in the retrieval process. The concatenation operation with/without the attribute manipulation is illustrated in Figure 2.

### 2.3. Generic representation of all attributes

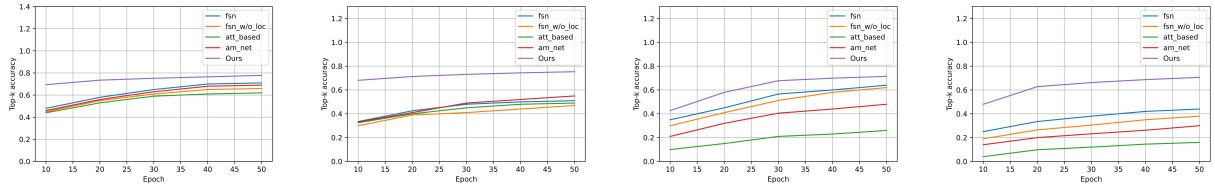
For interactive search to recommend substitute products, a generic representation of each attribute is required. Whenever a manipulation is employed, the generic feature representation of the required attribute replaces the feature representation of query attribute at the mentioned location. Instead of using complex additional memory blocks to achieve this, we simplify this by using a simple averaging process for generic representation. For each specific attribute in the super-category, we find images containing the attribute and average all the $128-$dimensional vectors from the corresponding model. After extracting all the average vectors, one for each attribute, we use it as a generic representation for the corresponding attribute. When a manipulation instruction is specified for an attribute in the query image, we replace the generic representation of the required attribute at its corresponding location. More details on attribute manipulation and image retrieval is explained in Section 3.

## 3. Attribute manipulation and substitute recommendation

After the generation of super-category specific attribute information and the generic representation of fine-grained attributes, the next imminent step for consideration becomes the manipulated attribute representation of the fashion product depending on user's feedback. This section constitutes the fourth and the fifth part of the end-to-end pipeline of the substitute recom-

Table 1: The performance comparison of DAtRNet (ours) w.r.t. DAtRNet with product category-specific embedding (Metric: Top-30 Retrieval accuracy)

| Method | Texture | Sleeve | Length | Neckline | Category | Shape | Overall |
|---|---|---|---|---|---|---|---|
| DAtRNet | 0.636 | 0.727 | 0.554 | 0.768 | 0.667 | 0.652 | 0.662 |
| DAtRNet with product category-specific embedding | 0.581 | 0.742 | 0.546 | 0.732 | 0.671 | 0.643 | 0.638 |




(a) (b) (c) (d)

Figure 4: Comparison for top-$k$ retrieval accuracy for search by query and search by query and attribute manipulation. (a). Search by query for Shopping100k dataset, (b). Search by query for DeepFashion dataset, (c). Search by query and attribute manipulation for Shopping100k dataset, (b). Search by query and attribute manipulation for DeepFashion dataset.

mendation with attribute manipulation feedback which sheds light on the interactive fashion product feature generation and process to perform substitute recommendation with it.

### 3.1. Query data and feature generation

With the concatenated feature of the retrieval images and the corresponding ground-truth attributes in hand, we hereby bring the query image data with/without manipulated attribute information for inference operation. We consider the query image and then extract the features for all six attribute categories, as given in Figure 2. Here, we give an example of an inference data: containing one query image and two manipulated attributes. The query image undergoes through all six super-categories considered for Shopping100k dataset, namely color, collar, fastening, neckline, pattern and sleeve. Also, in this step, we convert the manipulated attribute names to the corresponding ground-truth vector for that super-category. For the example in Figure 2, the desired attributes for category 2 and 5 transforms to $10^{th}$ and $20^{th}$ attribute in their respective super-category.

### 3.2. Manipulated feature generation

The generated features and the desired attribute ground-truths are then used to generate the desired feature representation of a fashion product. For the substitute recommendation without manipulated attributes, the six vectors are concatenated without any modification, as given in section 2.2. However, for one/multiple manipulated attribute(s), we use the generic feature representation as discussed in section 2.3. For all manipulated attribute instruction, we take the generic representation of the corresponding ground-truth and replace them with the vector extracted by the DAtRNet with same super-category. Figure 2 describes this operation in detail. In this example, we consider two manipulated attributes from category 2 and 5, having the ground truth value of 10 and 20, respectively. Hence, we consider the $10^{th}$ generic vector from category 2 in place of the $128-$dimensional vector obtained from collar attribute and the $20^{th}$ generic vector from category 5 in place of the $128-$dimensional vector obtained from pattern attribute. Finally, we concatenate the vectors, keeping $1^{st}$, $3^{rd}$, $4^{th}$ and $6^{th}$ directly from DAtRNet and $2^{nd}$ and $5^{th}$ from the generic representation. Note that, the generic vectors for category 2 and 5 in Figure 2 and this discussion is for example, we have created generic representation for all attributes present in all super-categories.

### 3.3. Top-$k$ images from the retrieval gallery

The updated query feature after concatenation and the target ground-truth label is then used for the computation involved with the features in retrieval gallery. We compute the $L_2$ distance between the vectors present in retrieval gallery and the update query feature vector. Based on this, we select top-$k$ images having smallest distance from the query image and their corresponding ground-truth attributes.

Table 2: Top-$k$ retrieval accuracy and NDCG@$k$ for all 6 categories of DeepFashion datasets for search by query and attribute manipulation.

| | k = 10 | | k = 20 | | k = 30 | | k = 40 | |
|---|---|---|---|---|---|---|---|---|
| | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** |
| **Texture** | 0.431 | 0.931 | 0.588 | 0.923 | 0.636 | 0.922 | 0.652 | 0.935 |
| **Sleeve** | 0.580 | 0.934 | 0.703 | 0.928 | 0.727 | 0.928 | 0.749 | 0.929 |
| **Length** | 0.363 | 0.932 | 0.515 | 0.924 | 0.554 | 0.923 | 0.577 | 0.924 |
| **Neckline** | 0.605 | 0.937 | 0.739 | 0.931 | 0.768 | 0.930 | 0.784 | 0.930 |
| **Category** | 0.485 | 0.928 | 0.631 | 0.921 | 0.667 | 0.920 | 0.698 | 0.921 |
| **Shape** | 0.443 | 0.924 | 0.613 | 0.917 | 0.652 | 0.916 | 0.676 | 0.918 |

Table 3: Top-$k$ retrieval accuracy and NDCG@$k$ for all 6 categories of Shopping100k dataset for search by query and attribute manipulation.

| | k = 10 | | k = 20 | | k = 30 | | k = 40 | |
|---|---|---|---|---|---|---|---|---|
| | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** |
| **Color** | 0.511 | 0.861 | 0.655 | 0.849 | 0.738 | 0.846 | 0.775 | 0.847 |
| **Collar** | 0.429 | 0.881 | 0.626 | 0.857 | 0.638 | 0.854 | 0.669 | 0.855 |
| **Fastening** | 0.208 | 0.887 | 0.358 | 0.866 | 0.503 | 0.861 | 0.504 | 0.860 |
| **Neckline** | 0.368 | 0.884 | 0.531 | 0.867 | 0.641 | 0.868 | 0.675 | 0.868 |
| **Pattern** | 0.398 | 0.863 | 0.563 | 0.842 | 0.674 | 0.838 | 0.678 | 0.840 |
| **Sleeve** | 0.589 | 0.915 | 0.693 | 0.905 | 0.761 | 0.905 | 0.775 | 0.904 |

Table 4: Performance of DAtRNet with the variation of image dimension, kernel size and number of filters.

| | **Only query** | **Query + Attribute** |
|---|---|---|
| **Image dimension** | | |
| Dimension: 160 | 0.873 | 0.834 |
| Dimension: 192 | 0.869 | 0.838 |
| Dimension: 256 | **0.894** | **0.846** |
| **Kernel size** | | |
| Kernel: (2,2) | 0.873 | 0.813 |
| Kernel: (3,3) | **0.894** | **0.846** |
| Kernel: (4,4) | 0.889 | 0.797 |
| **Number of filters** | | |
| Number: 16 | 0.885 | 0.823 |
| Number: 24 | 0.887 | 0.839 |
| Number: 32 | **0.894** | **0.846** |

# 4. Hyper-parameter tuning for DAtRNet architecture

We perform an extensive set of experiments to estimate the hyper-parameters, *e.g.* the image dimension, the kernel size and the number of filters in the first *Conv* layer. These experiments justify our choice of the said hyper-parameters in the main manuscript. For all these experiments, we have considered the DeepFashion dataset with 20,000 triplets for texture and sleeve super-categories.

Firstly, we observe the variation of the model performance by varying the image dimension. We considered three image dimensions, *i.e.* 160, 192 and 256, keeping kernel size as $3 \times 3$ and number of filters in the first layer as 32. From the results in Table 4, we observe that the model with image dimension of 256 gives best performance, hence this value is considered for further evaluation.

Then, we check the model performance by varying the kernel size from $2 \times 2$ to $4 \times 4$, keeping image dimension as 256 and number of filters in the first layer as 32. From the results in Table 4, we observe that the kernel size of $3 \times 3$ yields best performance.

Finally, we observe the optimum number of filters in the first layers, keeping image dimension as 256 and kernel size as $3 \times 3$. For this analysis, we considered 16, 24 and 32 as the numbers of the filters in the first layer. The results in Table 4 dictate that the proposed DAtRNet gives best performance with the number of filters as 32.

# 5. Axis-specific Class Activation Maps

In DAtRNet, concurrent axial attention has been proposed to separately extract discriminatory local features across three axes. The features obtained by the three heads are supposed to develop a holistic representation of several fine-grained attributes present in the image. To demonstrate the individual contribution of the height and width attention, we generate class activation maps (CAMs) by using any one of the width or height attention modules.

The generated CAMs of four sample images from DeepFashion dataset are given in Fig. 3. In the first image (Fig. 3(a)), the width attention branch effectively captures the pattern and the texture of the cloths with the sleeve length, whereas the height attention branch captures the neckline region properly. In the second image (Fig. 3(b)), the width attention branch focuses on the apparel pattern, while height attention highlights the neckline, sleeve region and the pleats in the skirt. In the third image (Fig. 3(c)), width and height attention focuses on sleeve length and neckline, respectively.

Table 5: Top-$k$ retrieval accuracy for Shopping100k and DeepFashion datasets for search by query and one attribute manipulation, where $k \in \{10, 20, 30, 40, 50\}$.

| Methods | Shopping100k | | | | | DeepFashion | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Top-10** | **Top-20** | **Top-30** | **Top-40** | **Top-50** | **Top-10** | **Top-20** | **Top-30** | **Top-40** | **Top-50** |
| Attribute-based [4] | 0.102 | 0.152 | 0.216 | 0.232 | 0.260 | 0.046 | 0.098 | 0.124 | 0.145 | 0.163 |
| AMNet [7] | 0.256 | 0.361 | 0.429 | 0.477 | 0.516 | 0.141 | 0.193 | 0.229 | 0.255 | 0.276 |
| FashionSearchNet w/o Loc [2] | 0.311 | 0.415 | 0.512 | 0.585 | 0.628 | 0.192 | 0.265 | 0.305 | 0.351 | 0.386 |
| FashionSearchNet [2] | 0.384 | 0.474 | 0.572 | 0.616 | 0.667 | 0.252 | 0.335 | 0.381 | 0.426 | 0.447 |
| ADDE-M [3] | 0.412 | 0.529 | 0.598 | 0.641 | 0.673 | 0.236 | 0.286 | 0.315 | 0.340 | 0.359 |
| **DAtRNet (Ours)** | **0.634** | **0.651** | **0.677** | **0.691** | **0.745** | **0.479** | **0.627** | **0.662** | **0.687** | **0.706** |

Table 6: Top-$k$ retrieval accuracy and NDCG@$k$ for Shopping100k and DeepFashion datasets for search by query.

| | Shopping100k | | DeepFashion | |
|---|---|---|---|---|
| | **Acc@k** | **NDCG@k** | **Acc@k** | **NDCG@k** |
| $k = 10$ | 0.694 | 0.874 | 0.681 | 0.936 |
| $k = 20$ | 0.735 | 0.857 | 0.713 | 0.930 |
| $k = 30$ | 0.752 | 0.856 | 0.731 | 0.929 |
| $k = 40$ | 0.765 | 0.856 | 0.744 | 0.929 |

Finally, in Fig. 3(d), width attention localizes the neckline and height attention encapsulates the pattern and the sleeves.

Hence, from these visualizations, we can say that these different attention layers localizes different regions of the fashion image, which in turn aggregates holistic representation when added together.

## 6. Variation in performance of substitute recommendation using DAtRNet with respect to *k*

We provide the results of substitute fashion search with query image in Table 6. Here, we report the top-$k$ retrieval accuracy and NDCG@$k$ values in Table 6 for $k = 10, 20, 30, 40$. Also, it should be noted that attribute-level similarity is the focus and not the super-category level similarity, which makes this problem more challenging than traditional substitute recommendation that consider super-category information. Table 5 compares the performances with respect to $k$ by state-of-the-art methodologies with one attribute manipulation using both the datasets. Here, we observe that the proposed DAtRNet outperforms the existing solutions by a significant margin.

## 7. Inference with product category specific generic embedding

For all our experiments, we have considered a generic representation of every attribute to impose personalization during substitute product search. However, these generic feature vectors were created by averaging all vectors of that fine-grained category. Another method to generate these average feature is to make it product-specific, *e.g.,* the generic feature representing collar attribute of a T-shirt will be different than that of a blazer. To observe the variations in performance, we conduct this experiment for all six super-categories of DeepFashion dataset. For this operation, we have considered the same data as used for all experiments. Also, we considered 50 different product categories to facilitate product-specific embedding.

Table 7: Impact of the number of manipulated attributes on the performance of Shopping100k and DeepFashion datasets.

| Number of attribute | Shopping100k | | DeepFashion | |
|---|---|---|---|---|
| | Acc@30 | NDCG@30 | Acc@30 | NDCG@30 |
| 1 | 0.677 | 0.878 | 0.662 | 0.928 |
| 2 | 0.648 | 0.867 | 0.573 | 0.921 |
| 3 | 0.484 | 0.875 | 0.647 | 0.922 |
| 4 | 0.511 | 0.848 | 0.645 | 0.925 |

The results of these experiments are given in Table 1. From these results, we observe that the product-category specific generic representation yields slightly improved performance for sleeve and category attributes, whereas, it gives poorer performance for other four attributes. Overall, the performance of DAtRNet with product category specific embedding is less compared to DAtRNet with generic embedding. Hence, we can hypothesize that our model captures the features irrespective of the product present.

## 8. Category-specific performance of DAtRNet with variations in $k$

The consistency in the performance of DAtRNet with the variations in attribute super-category and the number of retrieved product is necessary to understand the robustness of the system. In Tables 2 and 3, we report the variation of top-$k$ retrieval accuracy of DAtRNet for substitute recommendation across all attribute super-categories. In Table 2, we provide the results for DeepFashion dataset with texture, sleeve, dress length, neckline, category and shape attributes. In Table 3, we explore these values for Shopping100k dataset having color, collar, fastening, neckline, pattern and sleeve at-

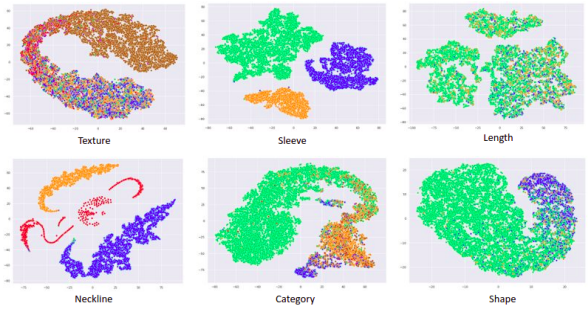



Figure 5: The t-SNE visualization of attribute embedding of six super-categories of the DeepFashion dataset in style space.

tributes. For both the cases, $k$ values are considered to be $\{10, 20, 30, 40\}$. Here, we observe that the values increase consistently with the increment in the values of $k$ for all the super-categories. Also, we have given the plots in Figure 4 to compare the performance of DAtRNet with the state-of-the-art methods by varying the $k$ value. For comparison, we have considered attribute-based method [4], AMNet [7] and FashionSearchNet [1, 2].


Figure 6: Visual examples of DAtRNet to perform fashion search with no attribute manipulation. Here, the first image in each row is considered to be the query image for search and the next four images are retrieved by DAtRNet. The images with green box are the ones with same set of attributes as the query image.


Figure 7: Visual examples of DAtRNet to perform fashion search with one attribute manipulation. Here, the first image in each row is considered to be the query image for search and the next four images are retrieved by DAtRNet. The arrow provides the information regarding the change in the attributes and the desired attributes. The images with green box are the ones with same set of attributes as desired by the user to replace the query image.

## 9. Impact of number of manipulated attributes on the performance of attribute-aware substitute recommendation

The challenges involved in recommending substitute fashion products increase when the number of desired attributes increases. Only one method in literature considers two attribute manipulation study [7]. To address the robustness of the network, we provide up to four desired attribute manipulation and observe the performance for both the datasets in Table 7. Here, we report the top-30 accuracy and NDCG@30, after providing $N$ desired attributes where $N = 1, 2, 3, 4$. From the results in Table 7, it is clear that the proposed network is well-suited for incorporating multiple attribute instructions without significantly degrading the performance.

## 10. The t-SNE visualization

The DAtRNet architecture generates discriminatory attribute embedding in style space to provide effective fashion search. The performance of substitute recommendation depends upon the separateness of fine-


Figure 8: Visual examples of DAtRNet to perform fashion search with two attribute manipulation. Here, the first image in each row is considered to be the query image for search and the next four images are retrieved by DAtRNet. The arrow between the query and retrieval images denote the attribute manipulation instructions. The images with green box are the ones with same set of attributes after the manipulation.

(a) Fashion Search with query image (no attribute manipulation)


(b) Fashion Search with query image and one attribute manipulation


(c) Fashion Search with query image and two attribute manipulation

Figure 9: Visual examples of failure cases for all the tasks considered by us. Here, we demonstrate set of results where we do not get the *exact* set of attributes in the retrieval product, although, for all of them, majority of the attributes are matching. For (d), green box denotes the ground truth and the red box denotes the model prediction.

grained attributes. To qualitatively estimate different attribute separation, we generate t-SNE visualization for six attribute categories of DeepFashion dataset in style space in Figure 5. From the t-SNE plots, we observe

that the fine-grained attribute categories are quite separable to enable the discrimination between them which amounts to superior performance with the attribute-aware recommendation. This is yet another evidence of effective feature disentanglement of attributes in style spaces which leverage on the improvement in performance in attribute-aware substitute recommendations.

## 11. Visual examples and Failure cases

Here, we provide more visual examples of all the tasks as well as the failure cases to explain the ability of the network to perform the substitute recommendation and the failure cases. For all cases, we considered equal number of male and female products to avoid gender bias for recommendation. In Figure 6, we provide the visual examples of fashion search without attribute manipulation. From the results, we can observe that the network is able to look at the individual attributes by recommending items with same set of attributes as present in the query images.

Similarly, in Figure 7 and in Figure 8, we provide the visual examples of DAtRNet to perform fashion search with one and two attribute manipulations, respectively. For all these cases, it can be noted that the DAtRNet are able to understand most of the correct attributes for all the retrieved items, which shows the ability of DAtRNet to efficiently disentangle and process attribute information for better recommendation.

The attribute-aware recommendation is a complex operation with small error resonating in wrong prediction. To validate the results obtained by us in an unbiased manner, we have provided failure examples of these use-cases in Figure 9. Here, we can observe that for all these cases, the failure is happening due to not preserving all existing attribute information or by considering one or more extra attribute information. in future work, we will aim to address this.

## 12. Human in the loop experiment

In order to validate the proposed method, an experiment with "human judges in the loop" is performed. Ten human judges were selected and twenty sample images were shown to each of them. Based on their desired attribute instruction, ten fashion products were recommended by each of these two methods. The recommendations were shown in a random order to avoid a trend. Out of all votes obtained through this experiment, DAtRNet received 184 of 200 votes, whereas FashionSearchNet received sixteen votes.

## References


[1] Kenan E Ak, Ashraf A Kassim, Joo Hwee Lim, and Jo Yew Tham. Fashionsearchnet: Fashion search with attribute manipulation. In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, pages 45–53, 2018.

[2] Kenan E Ak, Joo Hwee Lim, Jo Yew Tham, and Ashraf A Kassim. Efficient multi-attribute similarity learning towards attribute-based fashion search. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1671–1679, 2018.

[3] Yuxin Hou, Eleonora Vig, Michael Donoser, and Loris Bazzani. Learning attribute-driven disentangled representations for interactive fashion retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12147–12157, 2021.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25 (NIPS)*, pages 1097–1105, 2012.

[5] 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)*, pages 1096–1104, 2016.

[6] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research (JMLR)*, 9(11):2579–2605, 2008.

[7] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. Memory-augmented attribute manipulation networks for interactive fashion search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1520–1528, 2017.