Pattern Recognition Letters 172 (2023) 203–212

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Attribute disentanglement with gradient reversal for interactive fashion retrieval

Giovanna Scaramuzzino[a], Federico Becattini[b,*], Alberto Del Bimbo[a]

[a] University of Florence, Italy
[b] University of Siena, Italy

ARTICLE INFO

Article history:
Received 20 January 2023
Revised 28 May 2023
Accepted 28 June 2023
Available online 29 June 2023

Edited by: Maria De Marsico

MSC:
41A05
41A10
65D05
65D17

Keywords:
Attribute manipulation
Disentanglement
Fashion retrieval

ABSTRACT

Interactive fashion search is gaining more and more interest thanks to the rapid diffusion of online retailers. It allows users to browse fashion items and perform attribute manipulations, modifying parts or details of given garments. To successfully model and analyze garments at such a fine-grained level, it is necessary to obtain attribute-wise representations, separating information relative to different characteristics. In this work we propose an attribute disentanglement method based on attribute classifiers and the usage of gradient reversal layers. This combination allows us to learn attribute-specific features, removing unwanted details from each representation. We test the effectiveness of our learned features in a fashion attribute manipulation task, obtaining state of the art results. Furthermore, to favor training stability we present a novel loss balancing approach, preventing reversed losses to diverge during the optimization process.

1. Introduction

In the field of fashion image analysis, the ability to effectively manipulate and optimize the features extracted from garment images is crucial for the development of accurate and robust models. In particular, attribute manipulation finds large application for online retailers since users can add, remove or transform visual traits of a garment to match a desired style. Being able to decompose and describe an image using a set of attributes also helps in analyzing the images, recognizing patterns and styles at a fine grained level.

Closely related to attribute manipulation is the concept of attribute disentanglement, which focuses on separating the underlying factors of variation in a dataset. In the context of fashion images, this means identifying and isolating the latent features that represent different fashion attributes such as color, texture and pattern. The problem of attribute disentanglement in fashion images is particularly challenging due to the high variability and complexity of garments. For example, the same color can appear in different shades and hues, and the same pattern can be represented in different scales and orientations. Additionally, fashion images often contain multiple attributes that are correlated and intertwined, making it difficult to separate them. To address this problem, several approaches have been proposed in the literature [1–3], using specifically tailored architectures like Generative Adversarial Networks (GANs) [4,5] or Memory Augmented Neural Networks (MANNs) [2,3,6].

In this paper, we propose a novel approach based on gradient reversal to disentangle the attribute feature space in fashion images. Our method is based on training multiple classifiers to predict the value of each attribute, such as color, texture, and pattern. Thanks to a gradient reversal layer, the optimization of the classifiers leads to feature representations that are specifically geared towards particular attributes while ignoring the others. The rationale behind this approach is the following. Training attribute-specific classifiers has been shown to provide attribute disentanglement to a certain extent [2]. In this scenario, the same input image is projected into multiple latent spaces with a multi-branch architecture and attribute-specific classifiers ensure that each latent space is organized well to discriminate the correspondent attribute values. However, in this setting, there is no guarantee that a given attribute feature does not capture aspects relative to other attributes (e.g., the color latent space might still encode information about

* Corresponding author.
E-mail addresses: giovanna.scaramuzzino@stud.unifi.it (G. Scaramuzzino), federico.becattini@unisi.it (F. Becattini), alberto.delbimbo@unifi.it (A. Del Bimbo).

https://doi.org/10.1016/j.patrec.2023.06.018

shape), negatively affecting downstream applications. We build on this idea, but instead of simply learning to classify the value of a specific attribute, we also ensure that the model removes all possible information regarding other attributes. To reach this purpose, we feed each feature from the branched architecture to all the attribute classifiers, yet only one is optimized to correctly classify the attribute values and the others push the backbone to remove unwanted features thanks to the usage of gradient reversal. In fact, reversing the gradient favors weight updates that work against a correct classification for unwanted attributes, thus shaping the latent space accordingly. Experimental validation demonstrates the effectiveness of the proposed approach, obtaining state of the art results on attribute manipulation tasks.

To summarize, the main contributions of this paper are:

- We present a multi-attribute disentanglement strategy based on gradient reversal. Our approach makes use of attribute-specific classifiers combined with gradient reversal layers to learn disentangled representations.
- To address the challenges of training a model with gradient reversal, we propose a loss balancing method that favors training convergence.
- We integrate our extractor of disentangled features into an attribute manipulation framework, obtaining state of the art results on the challenging dataset Shopping100k [7].

2. Related works

With the growth of online shopping, retailers and social networks have gathered large collections of fashion images, that are used on a daily basis by customers to browse and purchase items. For this reason, there is a strong interest in recognizing and modeling fashion items and their attributes automatically [2,3,6,8,9] for different kinds of tasks [10–12]. In this work we focus on understanding how a garment can be characterized as a composition of parts, i.e. a collection of attributes that can describe the item. Detecting and describing such attributes has been a hot research topic [8,9,13]. For instance, Chen et al. [8] proposed an automatic framework to generate a list of attributes, whereas Abdulnabi et al. [13] devised a multi-task approach for multi-attribute classification. Attributes have also been modeled jointly with additional media. In [9], the authors automatically infer attributes based on fashion images and their web-based description. Liu et al. [14] instead modeled attributes with image landmarks introducing a model dubbed FashionNet.

Of particular interest are methods that allow to interact with a database of garment images. The interaction can take place by searching items with certain attributes, possibly allowing specific manipulations to alter their values in a desired way. WhittleSearch [15] is an early search engine that allowed an interactive search using relative attributes such as “more elegant” or “less colorful”. This engine however required several iterations to change from one attribute to another. Such a concept has been also explored in [16]. More recently, some approaches have declined the problem as an image generation task exploiting Generative Adversarial Networks [17], such as GVM [4] or AMGAN [5]. These approaches however make the retrieval of real garments more challenging and dependent on the overall quality of the generated images.

To overcome this limitation, Zhao et al. [6] proposed a memory augmented attribute manipulation network dubbed AMNET, which instead of generating a new image, manipulates attributes in a disentangled latent space and directly performs retrieval. Other approaches have exploited memory augmented networks to store prototypes of disentangled features [2,3,18,19]. In [3] the authors use a contrastive learning approach to separate color and shape attributes and the memory stores different modalities to combine such features to compose an outfit. Variations of such method have been proposed in [20–22] including additional metadata related to style and suitability for social events. Differently from these approaches, which are able to disentangle only style and color, we propose a disentangling method for arbitrarily large set of attributes. More recently, Hou et al. [2] proposed ADDE, a model based on the combination of attribute classifiers to disentangle features and a memory of prototypes to perform manipulations. Our work is closely related to [2], from which we borrow part of the architecture, namely the memory structure. Differently from Hou et al. [2], however, we introduce a new training strategy based on gradient reversal [23] to favor attribute disentanglement. We show how our method outperforms prior work and is able to obtain better separated features and thus better attribute manipulations. Gradient reversal [23] is a training technique that exploits a special layer (the Gradient Reversal Layer, GRL) that acts as an identity during the forward pass of the network and inverts the sign of the gradient during backpropagation. This technique has been introduced for domain adaptation tasks, using a domain classifier which, under the influence of the GRL, is trained to remove domain information from the feature space. A few approaches have recently used gradient reversal to achieve feature disentanglement in different domains such as audio [24] or face recognition [25]. These methods however use the GRL as a sort of binary discriminator, similarly to Ganin and Lempitsky [23]. Instead, we exploit the GRL to perform multi-attribute disentanglement using a set of attribute-wise classifiers and we introduce a novel weighing strategy to avoid divergence during training.

3. Model

3.1. Problem formulation

Let $A = \{a_1, \ldots, a_N\}$ be a set of predefined garment attributes of length N, indexed by the symbol n. Each of these attributes a_n is associated with a list of possible values $a_n = (v_n^1, v_n^2, \ldots, v_n^{J_n})$, where J_n is the number of possible values for that attribute. For example $A = \{a_{color},\ a_{category}\}$ with

$$a_{color} = (red,\ green,\ white,\ black)$$
$$a_{category} = (shirt,\ dress,\ trousers).$$

In this case J_{color} and $J_{category}$ are equal to 4 and 3, respectively, as the number of values the attribute can assume. We indicate with J the total number of values that any attribute can assume: $J = \sum_{n=1,\ldots,N} J_n$.

Any garment image I can be described as a vector $v^I = (v_1^I, v_2^I, \ldots, v_N^I)$, where the jth attribute v_j^I can assume any value in a_j. An attribute manipulation is defined as a retrieval task where, given a query image I_q described by $v^q = (v_1^q, v_2^q, \ldots, v_N^q)$ and a query manipulation $a_k = q$ we want to retrieve an ordered list of target images described by $v^t = (v_1^t, v_2^t, \ldots, v_N^t)$, where $v_k^t = q$ and the remaining attributes are left unchanged.

3.2. Attribute disentanglement with gradient reversal

In order to obtain effective manipulations, it is necessary that the features of each attribute are disentangled and well separate. In this paper we achieve such goal using gradient reversal [23].

Two network architectures were designed and studied to separate the features using gradient reversal. The architectures are distinguished by the number of classifiers used to achieve feature disentanglement: (i) The first configuration (Fig. 1) uses N classifiers, one for each attribute. (ii) The second configuration (Fig. 2) uses N

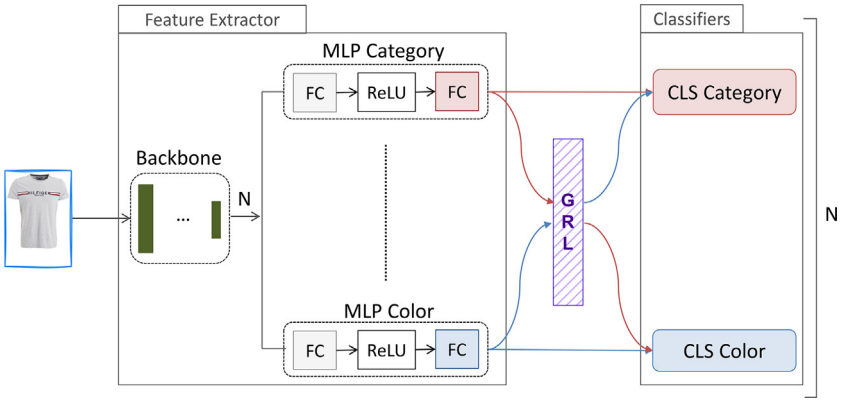

Fig. 1. Network architecture with N classifiers. Every attribute feature is classified by the respective classifier. At the same time, the remaining N-1 classifiers are trained to classify the other attributes under the influence of the Gradient Reversal Layer. This enforces each MLP to learn an attribute-specific latent space where data is organized according to a given attribute, disregarding information about the other.

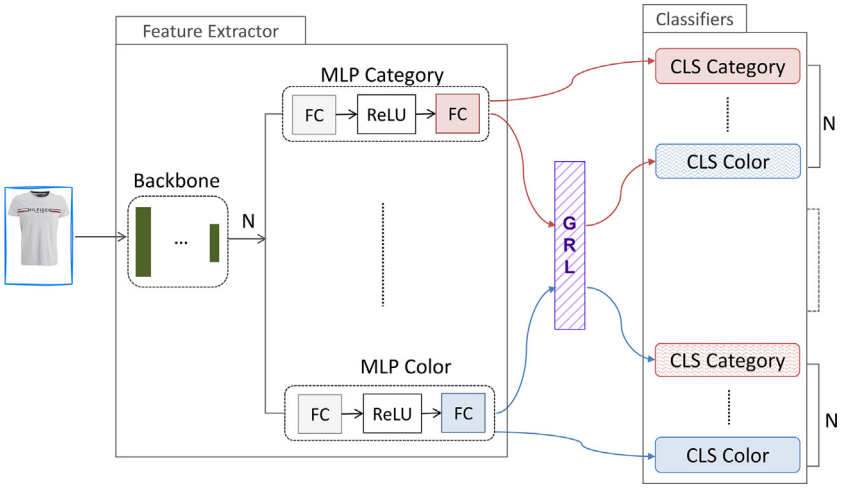

Fig. 2. Network architecture with $N * N$ classifiers. Every attribute feature is classified by N different classifiers. All the classifiers work under the influence of the Gradient Reversal Layer with the exception of the ones corresponding to the same attribute of the input features. This enforces each MLP to learn an attribute-specific latent space where data is organized according to a given attribute, disregarding information about the other.

attribute classifiers plus *N*(N-1)* additional classifiers preceded by a Gradient Reversal Layer (GRL).

The two network architectures have in common the use of a convolutional backbone and N different Multi-Layer Perceptrons (MLP). Following prior work [2], we used Alexnet [26] as backbone. The backbone network maps the input image into a latent representation ϕ. This representation is then used by the N MLPs to obtain separated features for each attribute.

Now we will analyze the other elements of the two implemented architectures. The first configuration involves the use of N classifiers, one for each attribute, each of which, given the feature extracted from the corresponding MLP, must classify the corresponding attribute. The nth classifier will therefore have as many outputs as the possible number of values J_n for the correspondent attribute. Referring to our previous example, the color classifier will receive as input the feature extracted from the MLP dedicated to color and will have to predict the color among the possible values a_{color}.

To optimize the classifier, the cross entropy loss was used as objective function, as this is a multi-label attribute classification problem. The objective function of the classifier is defined as:

$$L_{\text{cls}} = -\sum_{i=1}^{M}\sum_{n=1}^{N}\log(p(y_{i,n}|\hat{y}_{i,n})), \tag{1}$$

where $y_{i,n}$ is the ground-truth label of image I_i for attribute n, $\hat{y}_{i,n}$ is the output of the model, and M is the number of examples in a training batch.

Whereas such approaches have been used before for attribute disentanglement [2], optimizing only an attribute classifier loss does not yield an optimal separation of the descriptors. In fact, the model might exploit biases that create correlations between attributes such as gender and color.

To avoid these issues, we introduced the usage of a Gradient Reversal Layer (GRL) for attribute disentanglement. We augmented the network by adding gradient reversal layers between the attribute feature extractors (Alexnet + MLP) and the N-1 classifiers dedicated to classify all the other attributes. More formally, we define $N \times N$ pathways in the network separately connecting the N features with the N classifiers. The pathway between feature n and classifier n is left unchanged, whereas pathways between feature j and classifier k are connected through a gradient reversal layer when $j \neq k$.

The idea is that the classifiers, for each training batch, continue to classify the features extracted from the correspondent MLP, generating the correspondent attribute value. At the same time, the classifiers must also attempt to classify the features extracted from all the other N-1 MLPs, which are passed through the GRL. The forward pass will not be affected by gradient reversal, whereas during the backward pass each classifier will propagate a signal to the

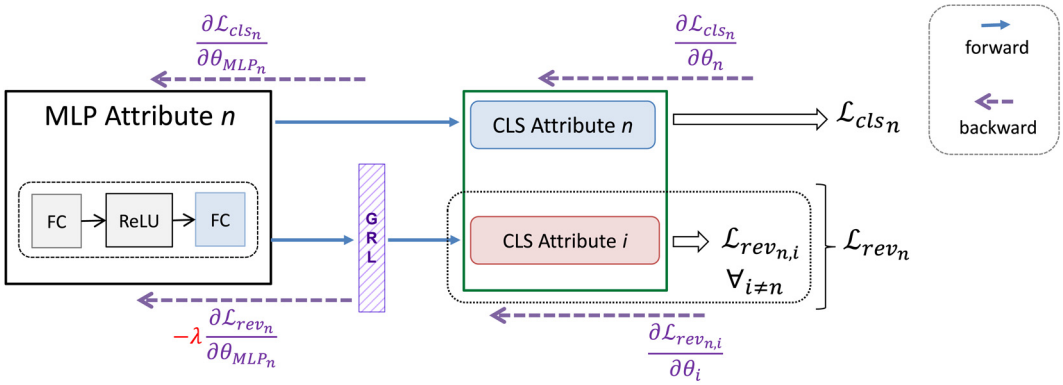

Fig. 3. Training scheme. Each MLP generates a different feature and is optimized to classify the correspondent attribute with separate classifiers. At the same time each feature is fed to the other classifiers to classify the other attributes under the influence of the gradient reversal layer.

features of the other attributes in order to remove information that is not attribute-specific. Overall, during training, each MLP will receive gradient updates from N distinct sources, one for each type of attribute (and thus, one for each classifier) as shown in Figs. 1 and 2. Among these, only the gradient of the *i*th classifier will be used *as is* to optimize the *i*th MLP. The remaining N-1 gradients will in fact be reversed, pushing the network to work poorly when classifying other attributes. This procedure shares some resemblance with adversarial training typical of GANs, where a "positive" and a "negative" loss are optimized jointly towards establishing an equilibrium. Such an idea has been exploited in literature to obtain disentangled representations both in GANs [27] and VAEs [28]. Disentanglement stems from the fact that, to satisfy all losses, the network must learn a latent space that allows an effective classification of the principal task (the "positive" one, i.e. the one without the GRL), while not allowing the other classifiers to perform their task.

The cross-entropy loss was used also for the classification of features passed through the gradient reversal layer. For simplicity we denote this loss as L_{rev}, and define it as:

$$L_{rev} = -\sum_{i=1}^{M}\sum_{n=1}^{N}\sum_{j\in N\setminus\{n\}} \mathcal{L}(F_n(f_j), y). \tag{2}$$

Here, $\mathcal{L}$ is the cross entropy, F_n is the classifier of attribute n, f_j is the feature of attribute j, y is the ground truth and M the number of samples in the batch.

The total loss is computed as the sum of the two losses defined in Eqs. (1) and (2):

$$L_{tot} = w_{cls}L_{cls} + w_{rev}L_{rev}. \tag{3}$$

The symbols w_{cls} and w_{rev} denote hyper-parameters weighing the losses.

As explained above, the GRL during forward propagation will act as an identity function. During backpropagation instead it will multiply the gradient of the loss L_{rev} to a negative scalar. At the same time, the classification loss L_{cls} is going to be minimized.

The weights of the feature extractors θ_{MLP_n} during backpropagation will therefore be updated according to:

$$\frac{\partial L}{\partial \theta_{MLP_n}} = \frac{\partial L_{cls_n}}{\partial \theta_{MLP_n}} - \lambda \frac{\partial L_{rev_n}}{\partial \theta_{MLP_n}}. \tag{4}$$

However, all the classifiers are trained to minimize L_{tot}. This means that the loss L_{rev} will eventually be maximized so that other attributes will not be deducible from a given feature, leading to a better feature disentanglement.

Subsequently, for the second configuration of our architecture, we took the architecture described so far to the extreme by using $N \times N$ classifiers. For each attribute n, N separate classifiers were created: one classifier must classify the features received from the *n*th MLP and the other N-1 classifiers must classify the features passed by the gradient reversal layer and extracted from the N-1 MLPs corresponding to the other attributes.

The loss to be minimized is the same as that reported in Eq. (3). In this case, therefore, all classifiers are trained to minimize their respective losses, in order to learn to classify the features. The presence of gradient reversal, among certain classifiers and the MLPs, means that each feature extractor is trained to maximize the loss L_{rev}. But since there are also attribute classifiers without GRL, the MLPs are also trained to minimize L_{cls} at the same time. Overall, the two model configurations optimize the same amount of losses to obtain the same outcome. The difference lies in the number of trainable parameters, since the first configuration has N shared classifiers, whereas the second one has $N \times N$ independent classifiers. The overall training scheme, shared by both architectures, is shown in Fig. 3.

3.3. On training with the gradient reversal layer

Training a model with multiple losses is not straightforward since weighing individual losses is cumbersome, yet cross-validation can help to establish suitable parameters. On the contrary, defining the λ hyper-parameter of the Gradient Reversal Layer is a much more sensible matter. In fact, when optimizing the L_{rev} losses we are minimizing the classification error but at the same time we are inverting the gradient, thus removing from the feature maps useful information that can be exploited for classification. This has the effect of pushing parts of the network to make a mistake. The λ parameter in the GRL regulates the strength of this "push" and, if not accurately tuned, will likely lead the model to diverge. The main issue is that the error function is bounded from below (when the loss is equal to zero), but is not bounded from above. Thus, when two losses push the same weights to achieve opposite outcomes it might be hard to find an equilibrium: as soon as L_{cls} becomes small enough, the importance of L_{rev} can quickly overcome such equilibrium, making the network diverge.

To avoid this issue, we introduce an adaptive reversal weight that constantly keeps a balance between the actual classification loss and the reversal one. The main idea is to let the gradient reversal layer affect weight updates by following the direction of the (reversed) gradient, but normalizing its magnitude. We set $\lambda = 1$ in the GRL, thus simply reversing the gradient without affecting its weight, and we balance the two losses L_{cls} and L_{rev} with Eq. (3), regulating the ratio between the importance of L_{cls} and L_{rev} with a

scalar β:

$$w_{rev} = \left(\frac{L_{cls} w_{cls}}{L_{rev}}\right)\beta. \quad (5)$$

In our experiments, we set $\beta = 0.1$ to keep an order of magnitude between the effect of the standard classification loss and the reversal one. In this way, when the classification loss tends towards zero, the update due to L_{rev} will become increasingly smaller. This approach has been adopted for both of our network configurations, with N or $N * N$ classifiers.

3.4. Attribute manipulation

To perform attribute manipulation using our disentangled features we rely on an architecture inspired by Hou et al. [2]. A memory block $\mathcal{M} \in \mathbb{R}^{N \cdot d \times J}$ is used, allowing to store prototype features of all attributes. N indicates the number of different attribute types present in the dataset, J indicates the total number of attribute values, and d indicates the specific embedding size of each attribute. Prototypes are stored for each attribute value. For example, for the *Color* attribute, there will be a prototype stored for each possible color (red, green, etc.) present in the dataset. The prototypes, for each attribute value, are saved in the columns of the memory block, according to:

$$\mathcal{M} = \begin{bmatrix} e_1^1 & \dots & e_1^{J_1} & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & e_2^1 & \dots & e_2^{J_2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 0 & e_N^1 & \dots & e_N^{J_N} \end{bmatrix}. \quad (6)$$

Element e_n^j are the embedding of the jth value for attribute n. Each prototype corresponds to the mean of the disentangled embeddings obtained with the encoder for each image sharing the same attribute value.

To perform an attribute manipulation, we define as $\mathbf{i} = (i^1, i^2, \dots, i^J)$, with $i \in \{-1, 1, 0\}$, the corresponding manipulation vector which specifies which attributes should be added, removed or left unaltered. In particular, every vector can specify a single manipulation, i.e. all values are set to zero with the exclusion of a single $+1$ and -1 corresponding to the value to be added and the value to be removed, respectively. Such values must correspond to the same attribute type (e.g., a color can be changed only into another color and not into a shape or a style). A manipulation is then obtained by performing

$$\mathbf{r}' = \mathbf{r_q} + \mathcal{M}\mathbf{i} \quad (7)$$

where $\mathbf{r_q}$ is the attribute disentangled feature of the query image. The obtained representation r' should be as close as possible to the feature of the target image r_t.

We finetune the memory block following the setting proposed in [2], thus adding $\mathcal{M}$ to the trainable parameters. The overall model architecture, including attribute manipulation is shown in Fig. 4.

4. Experiments

Dataset We performed experimental validation of our approach on the Shopping100k dataset [7]. The dataset contains 101,021 garment images with 12 attributes for a total of 151 unique attribute values. Table 1 shows the attributes and some examples of corresponding values. Differently from prior fashion datasets which were scarcely annotated with attribute labels such as [29], each image in Shopping100k has at least five attributes, providing a fine-grained description of the garment. Images have a dimension of 762 × 1100px.

Table 1
Attributes and example of values from Shopping100k [7].

Attributes	Values	Total
Category	Shirt, Dress, Trousers, Coat, ...	16
Color	Black, Pink, White, Green, ...	19
Pattern	Animal, Plain, Photo, Print, ...	16
Fit	Skinny, Regular, Loose, Oversize, ...	15
Sleeve	Long, Short, Sleeveless, Strapless, ...	9
Pocket	Side, Sleeve, Zip, Flap, ...	7
Neckline	Boat, Backless, Round, Square, ...	11
Fastening	Zip, Belt, Covered, Button, ...	10
Collar	High, Round, Hood, Lape, ...	17
Fabric	Denim, Canvas, Lace, Leather, ...	14
Sport	Basketball, Hiking, Swim, Tennis, ...	15
Gender	Male, Female	2

Table 2
Architecture details of our proposed approach.

Component	Structure
Backbone	Alexnet (Conv1 to Conv5 + Linear 1)
MLPs ϕ_n	Linear(4096, 340) + ReLU + Linear(340, 340)
Attribute predictor	Linear(340, number of attribute values)
Attribute predictor with Gradient Reversal	Linear(340, number of attribute values)

We used the same split as prior work [2,18], comprising 80,586 images for training and 20,000 for testing. Among these, 2000 images are selected as query images for the attribute manipulation task. Target attributes for the manipulations are generated so that at least an image exists in the dataset that shares the exact attribute values.

Network Architecture

Following prior work, we used an AlexNet [26] as backbone for our model. We adopt this solution, first of all, to provide comparable results with other works such as [2], which use the same architecture. In addition, AlexNet has a simple backbone where the only trainable layers are convolutional ones. This makes the usage of gradient reversal in our work more controllable and less inclined to yield unpredictable behaviors due to, for instance, skip-connections as in ResNet-like architectures.[1] The backbone provides 4096-dimensional features, which are then fed to all the MLP attribute encoders. The MLPs, composed of two fully connected layers separated by a ReLU activation, generate latent features in $\mathbb{R}^{340}$. Finally, such features are classified by a fully connected classifier with softmax activation. Every classifier has a different number of outputs, depending on the number of values the corresponding attribute can assume (see Table 1). Table 2 summarizes the details of the network architecture.

The memory block has a size of 4080×151. The first dimension stems from the concatenation of 12 (one per attribute) 340-dimensional features, whereas 151 is the total number of attribute values. The model has been trained on an Nvidia Tesla K80 using the Adam optimizer with learning rate 0.0001.

Attribute Manipulation Given a query image and a desired attribute manipulation, we use our feature extractors to obtain disentangled attribute specific representations. Through the usage of the memory module we compute the residual feature and, as specified in Eq. (7), we perform the manipulation in the feature space. To obtain a set of relevant garment images with the desired attribute values we pose the problem as a retrieval task. To this end

[1] We leave the study of gradient reversal on such architectures as future work.

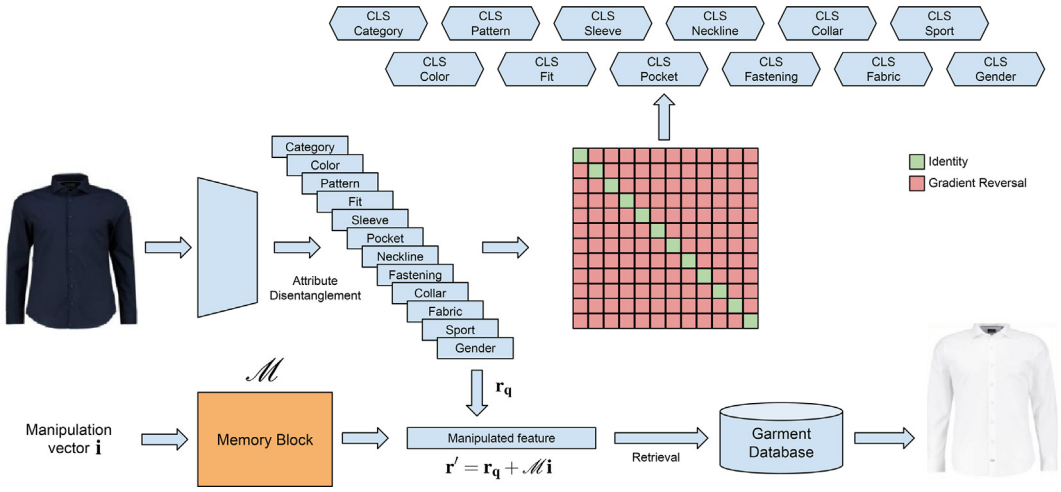

Fig. 4. Overall model architecture. The backbone extract features which are disentangled thanks to the usage of attribute classifiers and gradient reversal at training time. At inference time, the network receives a manipulation vector which is used to extract the corresponding prototype vectors from the memory block and alter the current garment. Manipulated garments can then be retrieved from a predefined database.

Fig. 5. Attribute classification accuracy, testing each of the N classifiers against each of the extracted feature vectors. Thanks to attribute disentanglement, it can be seen how only the ith feature can be correctly classified by the ith classifier. The only exception is the *gender* classifier, which is the hardest to separate from the other attributes since it is based on shared characteristics (collar, pattern, category, etc).

we use a K-Nearest Neighbor with L2 distance to identify garments with similar feature vectors to the one of the manipulated query. To evaluate the attribute manipulation task we use top-K retrieval accuracy by counting the fraction of retrieved garments that share the same attributes of the target among the K retrieved items.

Results

We first tested our feature extractor capability, without using the memory or any attribute manipulation. In particular, we have measured the attribute prediction accuracy of the classifiers and compared it to the results obtained by prior work that followed a similar approach. In Table 3 we report the results for attribute classification.

Table 3
Comparison of the proposed approach against the state of the art for attribute classification on Shopping100k.

Method	Accuracy
AMNet [6]	78.30
ADDE-M [2]	79.30
$N * N$ classifiers (Ours)	79.58
N classifiers (Ours)	**79.61**

Fig. 6. Top-5 results obtained retrieving garments after attribute manipulation on Shopping100k. Perfect matches are highlighted in green. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Both our models with N and $N*N$ classifiers obtain an accuracy higher than prior works. This suggests that the usage of gradient reversal has a beneficial effect on attribute separation in the latent space. The two variants obtain similar results, with a slight improvement for the model with N shared classifiers.

In Fig. 5 we report the confusion matrix obtained by predicting attribute values from the disentangled features generated by each MLP, using all the classifiers from the model with N classifiers. It can be seen how the ith classifier manages to classify well the correspondent ith feature, while it is unable to extract meaningful information from the other features. Note that such matrix does not sum to 100% either row-wise nor column-wise since each pair of feature-classifier is evaluated independently. Furthermore, each classifier at training time observes all features, yet N-1 of them are subject to the effect of the GRL layer.

Moving to the attribute manipulation task, we measure the top-K retrieval accuracy obtained by our model. This metric is computed as the number of hits, i.e. retrieved garments that have the same exact attributes of the target, divided by the number of retrieved garments.

In Table 4 we report the top-K retrieval accuracies for our two proposed models, compared against results from the state of the art. As the number of retrieved items increases, the accuracy improves for both models, reaching more than 68% for K=50. A slight difference is present between the two models yet the architecture with N shared classifiers consistently outperforms the $N*N$ classifiers variant. In particular, for low values of K we have the bigger improvements, meaning that we are able to retrieve more precise items in the first positions of the rankings. Moreover, since we have less trainable parameters, and thus a more lightweight model, for the remainder of the paper we will present results obtained with the first model.

Table 4
Comparison between our model and the state of the art on Shopping100k. For each method we report Top-K accuracy, varying the number of retrieved items.

Method	Top-10	Top-20	Top-30	Top-40	Top-50
AMNet [6]	25.62	36.13	42.94	47.71	61.64
FSN [18]	38.41	47.44	57.17	61.62	66.70
ADDE-M [2]	41.17	52.93	59.81	64.10	67.29
DAtRNet [1]	-	-	**67.70**	-	-
$N*N$ classifiers (Ours)	41.78	53.44	60.36	64.90	68.27
N classifiers (Ours)	**43.02**	**54.13**	60.76	**65.18**	**68.53**

Interestingly, thanks to the usage of the Gradient Reversal Layer and the disentangled representations, we are also able to outperform prior works. As baselines we use AMNet [6], FSN [19] and ADDE-M [2]. Like our models, all methods use an AlexNet [26] backbone to extract image features, making the comparison fair. In addition, despite adopting the same memory-based manipulation of ADDE-M we report an improvement of 1.85% in top-10 accuracy. We can ascribe this improvements to the usage of a better pre-training and better attribute feature disentanglement, gained thanks to our gradient reversal training strategy.

In addition to top-K retrieval accuracy, we also measures Normalized Discounted Cumulative Gain (NDCG@k) [30], which quan-

Fig. 7. Failure cases. None of the top-5 retrieved images match with the target attributes after the manipulation. Whereas the manipulation is often carried out correctly, some other attributes have changed as well.

Table 5
Comparison between our model and the state of the art on Shopping100k. We measure NDCG@30 for attribute manipulation.

Method	NDCG@30	$NDCG_{target}$@30	$NDCG_{others}$@30
AMNet [6]	71.48	40.10	75.71
ADDE-M [2]	73.67	43.05	77.79
N classifiers (Ours)	**74.26**	**44.00**	**78.32**

Table 6
Normalized Discounted Cumulative Gain on Shopping100k, varying the number of retrieved items K. For our method we use the *N* classifiers versions.

	NDCG		$NDCG_{target}$		$NDCG_{others}$	
K	ADDE-M [2]	Ours	ADDE-M [2]	Ours	ADDE-M [2]	Ours
@1	80.18	**80.97**	45.65	**47.66**	84.82	**85.45**
@2	78.67	**79.36**	46.93	**48.82**	82.96	**83.49**
@3	77.93	**78.55**	47.09	**48.81**	82.10	**82.58**
@4	77.40	**78.04**	47.00	**48.71**	81.51	**82.00**
@5	77.02	**77.64**	46.85	**48.55**	81.11	**81.58**
@10	75.84	**76.42**	45.83	**47.46**	79.90	**80.34**
@20	74.55	**75.10**	44.27	**45.53**	78.63	**79.09**
@30	73.67	**74.26**	43.05	**44.00**	77.79	**78.32**
@40	72.95	**73.60**	42.08	**42.77**	77.09	**77.73**
@50	72.30	**73.07**	41.25	**41.75**	76.47	**77.25**

tifies the ranking quality and is defined as $\frac{1}{Z}\sum_{j=1}^{k}\frac{2^{rel(j)-1}}{\log(j+1)}$ where $rel(j)$ is the attribute relevance score for the jth retrieved item, defined as the fraction of correct attributes, and Z is a normalization constant ensuring that correct results have a score equal to 1. Following [2] we report also $NDCG_{target}$ and $NDCG_{others}$. These variants of NDCG adopt different formulations for rel(j): $NDCG_{target}$ simply checks if the manipulated attribute is correct, whereas $NDCG_{others}$ considers the attributes that should be kept fixed.

We compared our model with *N* classifiers with the state of the art in Table 5, for K=30 as done in prior work. We do not report results for FSN [18] since the authors do not provide results for NDCG in the paper. Our method obtains better results in all three metrics. In particular, the larger gain is obtained for $NDCG_{target}$, underlining the effectiveness of the proposed method for manipulating attributes. To provide a more in depth analysis, we also compare our method against the best performing competitor, ADDE-M, varying the number of retrieved items K. In Table 6 we report such a comparison for the three metrics NDCG, $NDCG_{target}$ and $NDCG_{others}$. Interestingly, the proposed approach obtains higher results in all settings, also for low values of K which represent the most challenging setting. In particular, for $NDCG_{target}$ we report approximately a 2% improvement at K=1. On average, we observe an improvement of 0.65% for NDCG, 1.04% for $NDCG_{target}$ and 0.54% for $NDCG_{others}$.

Overall, we conclude that thanks to the usage of gradient reversal we can obtain a better disentanglement of fashion attributes in the latent space, yielding state of the art performance in attribute manipulation, both concerning the target attributes as well as the original attributes that should be left unchanged.

Ablation Study Here we discuss the importance of components in the model, in particular attribute classifiers and the usage of the GRL layer. Attribute-specific classifiers and gradient reversal are two entwined concepts. If we remove gradient reversal then it does not make sense to feed each of the N attribute features to all the N classifiers or we would obtain the opposite of what we are trying to achieve, i.e. we would obtain N general features that capture all the attributes at once without any disentanglement. What we can instead demonstrate is that if we remove gradient

Fig. 8. Comparison of our method and ADDE-M [2].

Table 7
Ablation study. We compare our proposed approach with and without the usage of gradient reversal layer. GRL proves to be highly important for obtaining more disentangled features and thus perform well of the downstream task of attribute manipulation.

Method	Top-10	Top-20	Top-30	Top-40	Top-50
N classifiers	**43.02**	**54.13**	**60.76**	**65.18**	**68.53**
N classifiers w/o GRL	40.39	49.87	55.76	61.44	63.18

reversal but we still keep attribute-specific classifiers (each classifier is used only for its corresponding feature, as in ADDE-M [2]), then the disentanglement gets worse and the manipulation accuracy drops. In Table 7 we show this behavior.

Qualitative Results

Here we report qualitative results showing results for attribute manipulation. In Fig. 6 we show the top-5 retrieved items after a manipulation. Correct matches are highlighted in green. The model is capable of correctly performing the manipulations, providing at least a correct result among the first results. It can be seen that even when the match is not perfect, the target attribute has been correctly changed.

Similarly, in Fig. 7 we show some failure cases. In these examples, none of the retrieved items matches perfectly with the target. Nonetheless, it has to be noticed that most garments indeed have the correct value for the manipulated attribute, yet they differ for some other aspect. For example, in the first row, the style attribute has been modified from *photo* to *striped* correctly for all retrieved items, but the other attributes of the query have been altered.

In addition, we compare our results to the ones obtained with ADDE-M in Fig. 8. In the first row, we can see that both models were capable of providing a good match, however in the ranking yielded by our model the correct garment is the first whereas for ADDE-M the third. In other examples instead ADDE-M failes to provide a correct match, contrarily to our method.

5. Conclusions

In this paper we presented an attribute disentanglement strategy using attribute-specific classifiers along with several gradient reversal layers. Since training such a network with lots of classifiers, most of which under the effect of gradient reversal, makes the training process challenging, we also presented a novel loss balancing approach to avoid the reversed losses to diverge. The presented method achieves state of the art results and improved feature quality compared to prior work. As an application, we used the learned features for attribute manipulation, a task that allows user to interactively retrieve garments with custom modifications based on a query fashion item.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

This work was partially supported by the Italian MIUR within PRIN 2017, Project Grant 20172BH297: I-MALL - improving the customer experience in stores by intelligent computer vision.

References

[1] G. Bhattacharya, N. Kilari, J. Gubbi, A. Pal, et al., DAtRNet: disentangling fashion attribute embedding for substitute item retrieval, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 2283–2287.

[2] Y. Hou, E. Vig, M. Donoser, L. Bazzani, Learning attribute-driven disentangled representations for interactive fashion retrieval, in: The International Conference on Computer Vision (ICCV), 2021.
[3] L. De Divitiis, F. Becattini, C. Baecchi, A.D. Bimbo, Disentangling features for fashion recommendation, ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) (2022).
[4] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, A.A. Efros, Generative visual manipulation on the natural image manifold, in: ECCV, 2016, pp. 597–613.
[5] K.E. Ak, J.H. Lim, J.Y. Tham, A.A. Kassim, Attribute manipulation generative adversarial networks for fashion images, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 10541–10550.
[6] B. Zhao, J. Feng, X. Wu, S. Yan, Memory-augmented attribute manipulation networks for interactive fashion search, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6156–6164.
[7] K.E. Ak, J.-H. Lim, J.Y. Tham, A.A. Kassim, Efficient multi-attribute similarity learning towards attribute-based fashion search, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 1671–1679.
[8] H. Chen, A. Gallagher, B. Girod, Describing clothing by semantic attributes, in: European Conference on Computer Vision, Springer, 2012, pp. 609–623.
[9] S. Vittayakorn, T. Umeda, K. Murasaki, K. Sudo, T. Okatani, K. Yamaguchi, Automatic attribute discovery with neural activations, in: European Conference on Computer Vision, Springer, 2016, pp. 252–268.
[10] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, R. Cucchiara, Dress code: high-resolution multi-category virtual try-on, in: Proceedings of the European Conference on Computer Vision, 2022.
[11] F. Becattini, X. Song, C. Baecchi, S.-T. Fang, C. Ferrari, L. Nie, A. Del Bimbo, PLM-IPE: a pixel-landmark mutual enhanced framework for implicit preference estimation, in: ACM Multimedia Asia, 2021, pp. 1–5.
[12] A. Baldrati, M. Bertini, T. Uricchio, A. Del Bimbo, Conditioned image retrieval for fashion using contrastive learning and clip-based features, in: ACM Multimedia Asia, 2021, pp. 1–5.
[13] A.H. Abdulnabi, G. Wang, J. Lu, K. Jia, Multi-task CNN model for attribute prediction, IEEE Trans. Multimedia 17 (11) (2015) 1949–1959.
[14] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, DeepFashion: powering robust clothes recognition and retrieval with rich annotations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1096–1104.
[15] A. Kovashka, D. Parikh, K. Grauman, WhittleSearch: image search with relative attribute feedback, in: CVPR, 2012, pp. 2973–2980.
[16] D. Parikh, K. Grauman, Relative attributes, in: ICCV, 2011, pp. 503–510.
[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).
[18] K.E. Ak, A.A. Kassim, J.-H. Lim, J.Y. Tham, Learning attribute representations with localization for flexible fashion search, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7708–7717.
[19] K.E. Ak, J.-H. Lim, Y. Sun, J.Y. Tham, A.A. Kassim, FashionSearchNet-v2: learning attribute representations with localization for image retrieval with attribute manipulation, 2021.
[20] L. De Divitiis, F. Becattini, C. Baecchi, A. Del Bimbo, Garment recommendation with memory augmented neural networks, in: International Conference on Pattern Recognition, Springer, 2021, pp. 282–295.
[21] L. De Divitiis, F. Becattini, C. Baecchi, A. Del Bimbo, Style-based outfit recommendation, in: 2021 International Conference on Content-Based Multimedia Indexing (CBMI), IEEE, 2021, pp. 1–4.
[22] F. Becattini, L. De Divitiis, C. Baecchi, A.D. Bimbo, Fashion recommendation based on style and social events, Multimed. Tools Appl. (2023) 1–16.
[23] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: Proceedings of the 32nd International Conference on Machine Learning, PMLR, 2015, pp. 1180–1189.
[24] N. Hou, C. Xu, E.S. Chng, H. Li, Learning disentangled feature representations for speech enhancement via adversarial training, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 666–670.
[25] Z. Zhang, C. Yu, H. Li, J. Sun, F. Liu, Learning distribution independent latent representation for 3D face disentanglement, in: 2020 International Conference on 3D Vision (3DV), IEEE, 2020, pp. 848–857.
[26] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[27] Z. Zheng, L. Sun, Disentangling latent space for VAE by label relevant/irrelevant dimensions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12192–12201.
[28] M. Patacchiola, P. Fox-Roberts, E. Rosten, Y-autoencoders: disentangling latent representations via sequential encoding, Pattern Recognit. Lett. 140 (2020) 59–65.
[29] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, DeepFashion: powering robust clothes recognition and retrieval with rich annotations, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1096–1104.
[30] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (2002) 422–446, doi:10.1145/582415.582418.