Learning Type-Aware Embeddings for Fashion Compatibility

Mariya I. Vasileva, Bryan A. Plummer(✉), Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth

Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, USA
{mvasile2,bplumme2,dusad2,srajpal2,ranjitha,daf}@illnois.edu

Abstract. Outfits in online fashion data are composed of items of many different types (*e.g.* top, bottom, shoes) that share some stylistic relationship with one another. A representation for building outfits requires a method that can learn both notions of *similarity* (for example, when two tops are interchangeable) and *compatibility* (items of possibly different type that can go together in an outfit). This paper presents an approach to learning an image embedding that respects item type, and jointly learns notions of item similarity and compatibility in an end-to-end model. To evaluate the learned representation, we crawled 68,306 outfits created by users on the Polyvore website. Our approach obtains 3–5% improvement over the state-of-the-art on outfit compatibility prediction and fill-in-the-blank tasks using our dataset, as well as an established smaller dataset, while supporting a variety of useful queries (Code and data: https://github.com/mvasil/fashion-compatibility).

Keywords: Fashion · Embedding methods · Appearance representations

1 Introduction

Outfit composition is a difficult problem to tackle due to the complex interplay of human creativity, style expertise, and self-expression involved in the process of transforming a collection of seemingly disjoint items into a cohesive concept. Beyond selecting which pair of jeans to wear on any given day, humans battle fashion-related problems ranging from, *"How can I achieve the same look as celebrity X on a vastly inferior budget?"*, to *"How much would this scarf contribute to the versatility of my personal wardrobe?"*, to *"How should I dress to communicate motivation and competence at a job interview?"*. This paper provides a step towards answering such diverse and logical queries.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01270-0_24) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11220, pp. 405–421, 2018.
https://doi.org/10.1007/978-3-030-01270-0_24

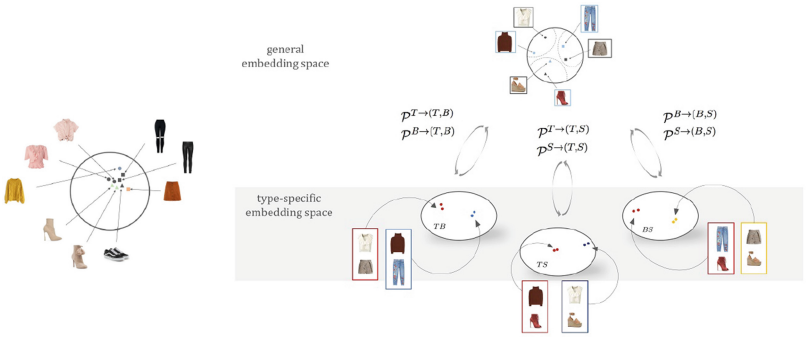

Fig. 1. Left: Conventional embedding strategies embed objects of all types in one underlying space. Objects that are compatible must lie close; as a result, all shoes that match a given top are **obliged** to be close. **Right:** Our type-respecting embedding, using "top", "bottom" and "shoes" as examples. We first learn a single, shared embedding space. Then, we project from that shared embedding to subspaces identified by type. This means that all shoes that match a given top **must** be close in shoe-top space, but can be very different in the general embedding space. This enables us to search for two pairs of shoes that (1) match the same top, and (2) look very different from one another

To learn how to compose outfits the underlying representation must support both notions of *similarity* (*e.g.*, when two tops are interchangeable) and notions of *compatibility* (items of possibly different type that can go together in an outfit). Current research handles both kinds of relationships with an embedding strategy: one trains a mapping, typically implemented as a convolutional neural network, that takes input items to an embedding space (*e.g.* [12,14,34]). The training process tries to ensure that similar items are embedded nearby, and items that are different have widely separated embeddings (*i.e.*the left side of Fig. 1).

These strategies, however, do not respect types (*e.g.* shoes embed into the same space hats do), which has important consequences. Failure to respect types when training an embedding *compresses variation*: for instance, all shoes matching a particular hat are forced to embed close to one another, thus making them appear compatible even if they are not, which severely limits the model's ability to address diverse queries. Worse, this strategy encourages *improper triangles*: if a pair of shoes match a hat, and that hat in turn matches a blouse, then a natural consequence of models without type-respecting embeddings is that the shoes are forced to also match the blouse. This is because the shoes must embed close to the hat to match, the hat must embed close to the shoes to match, thus ensuring the shoes embed close to the blouse as well. Instead, they should be allowed to match in one context, and not match in another. An alternative way to describe the issue is that compatibility is *not* naturally a transitive property, but *being nearby* is. Thus, an embedding that clusters items close together is not a natural way to measure compatibility without paying attention to context. By

learning type-respecting spaces to measure compatibility, as in the right side of Fig. 1, we avoid the issues stemming from using a single embedding.

We begin by encoding each image in a general embedding space which we use to measure item similarity. The general embedding is trained using a visual-semantic loss between the image embedding, and features representing a text description of the corresponding item. This helps ensure that semantically similar items are projected in a nearby space. In addition, we use a learned projection which maps our general embedding to a secondary embedding space that scores compatibility between two item types. We utilize a different projection for each pairwise compatibility comparison, unlike prior work which typically uses a single embedding space for compatibility scoring (*e.g.* [12,34]). For example, if an outfit contains *a hat*, *a blouse*, and *a shoe*, we would learn projections for *hat-blouse*, *hat-shoe*, and *blouse-shoe* embeddings. The embeddings are trained along with a generalized distance metric, which we use to compute compatibility scores between items. Please refer to the supplementary material for a visualization of our approach.

Since many of the current fashion datasets either do not contain outfit compatibility annotations [20], or are limited in size and the type of annotations they provide [12], we collect our own dataset which we describe in Sect. 3. In Sect. 4 we discuss our type-aware embedding model, which enables us to perform complex queries on our data. Our experiments outlined in Sect. 5 demonstrate the effectiveness of our approach, reporting a 4% improvement in a fill-in-the-blank outfit completion experiment, and a 5% improvement in an outfit compatibility prediction task over the prior state-of-the-art.

2 Related Work

Embedding methods provide the means to learn complicated relationships by simply providing samples of positive and negative pairs. These approaches tend to be trained as a siamese network [3] or using triplet losses [23] and have demonstrated impressive performance on challenging tasks like face verification [26]. This is attractive for many fashion-related tasks which typically require the learning of hard-to-define relationships between items (*e.g.* [10–12,14,34]). Veit *et al.* [34] demonstrate successful similarity and compatibility matching for images of clothes on a large scale, but do not distinguish items by type (*e.g.* top, shoes, scarves) in their embedding. This is extended in Han *et al.* [12] by feeding the visual representation for each item in an outfit into an LSTM in order to jointly reason about the outfit as a whole. There are several approaches which learn multiple embeddings (*e.g.* [2,5,11,30,33]), but tend to assume that instances of distinct types are separated (*i.e.*comparing bags only to other bags), or use knowledge graph representation (*e.g.* [36]). Multi-modal embeddings appear to reveal novel feature structures (*e.g.* [10,12,24,25]), which we also take advantage of in Sect. 4.1. Training these type of embedding networks remains difficult because arbitrarily chosen triples can provide poor constraints [35,43].

Table 1. Comparison in dataset statistics. Our dataset's variants (last two rows) contains more outfits than related datasets along with detailed descriptions and fine-grained semantic categories

Dataset	#Outfits	#Items	Max Items/ Outfit	Text available?	Semantic category?
Maryland polyvore [12]	21,889	164,379	8	Titles Only	–
Polyvore outfits-D	32,140	175,485	16	Titles & Descriptions	✓
Polyvore outfits	68,306	365,054	19	Titles & Descriptions	✓

Fashion Studies. Much of the recent fashion related work in the computer vision community has focused on tasks like product search and matching [6,9, 11,20,41], synthesizing garments from word descriptions [42], or using synthesis results as a search key [41]. Kiapour *et al.* [16] trained an SVM over a set of hand-crafted features to categorize items into clothing styles. Vaccaro *et al.* [32] trained a topic model over outfits in order to learn latent fashion concepts. Others have identified clothing styles using meta-data labels [28] or learning a topic model over a bag of visual attributes [15]. Liu *et al.* [19] measure compatibility between items by reasoning about clothing attributes learned as latent variables in their SVM-based recommendation system. Others have focused on attribute recognition [4,8,37], identifying relative strength of attributes between items (*i.e.*a shoe is more/less pointy than another shoe) [29,38–40], or focused on predicting the popularity of clothing items [1,27].

3 Data

Polyvore Dataset: The Polyvore fashion website enables users to create outfits as compositions of clothing items, each containing rich multi-modal information such as product images, text descriptions, associated tags, popularity score, and type information. Han *et al.*supplied a dataset of Polyvore outfits (referred to as the Maryland Polyvore dataset [12]). This dataset is relatively small, does not contain item types or detailed text descriptions (see Table 1), and has some inconsistencies in the test set that make quantitative evaluation unreliable (additional details in Sect. 5). To resolve these issues, we collected our own dataset from Polyvore annotated with outfit and item ID, fine-grained item type, title, text descriptions, and outfit images. Outfits that contain a single item or are missing type information are discarded, resulting in a total of 68,306 outfits and 365,054 items. Statistics comparing the two datasets are provided in Table 1.

Test-train splits: Splits for outfit data are quite delicate, as one must consider whether a garment in the train set should be allowed to appear in unseen test outfits, or not at all. As some garments are "friendly" and appear in many outfits, this choice has a significant effect on the dataset. We provide two different versions of our dataset with respective train and test splits. An "easier" split contains 53,306 outfits for training, 10,000 for testing, and 5,000 for validation,

whereby no outfit appearing in one of the three sets is seen in the other two, but it is possible that an item participating in a train outfit is also encountered in a test outfit. A more "difficult" split is also provided whereby a graph segmentation algorithm was used to ensure that no garment appears in more than one split. Each item is a node in the graph, and an edge connects two nodes if the corresponding items appear together in an outfit. This procedure requires discarding "friendly" garments, or else the number of outfits collapses due to superconnectivity of the underlying graph. By discarding the smallest number of nodes necessary, we end up with a total of 32,140 outfits and 175,485 items, 16,995 of which are used for training and 15,145 for testing and validation.

4 Respecting Type in Embedding

For the i'th data item $\mathbf{x}_i$, an embedding method uses some regression procedure (currently, a multilayer convolutional neural network) to compute a nonlinear feature embedding $\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i; \theta) \in \mathbb{R}^d$. The goal is to learn the parameters θ of the mapping $\mathbf{f}$ such that for a pair of items $(\mathbf{x}_i, \mathbf{x}_j)$, the Euclidean distance between the embedding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ reflects their compatibility. We would like to achieve a "well-behaved" embedding space in which that distance is small for items that are labelled as compatible, and large for incompatible pairs.

Assume we have a taxonomy of $\mathcal{T}$ types, and let us denote the type of an item as a superscript, such that $\mathbf{x}_i^{(\tau)}$ represents the i'th item of type τ, where $\tau = 1, \ldots, \mathcal{T}$. A triplet is defined as a set of images $\{\mathbf{x}_i^{(u)}, \mathbf{x}_j^{(v)}, \mathbf{x}_k^{(v)}\}$ with the following relationship: the anchor image $\mathbf{x}_i$ is of some type u, and both $\mathbf{x}_j$ and $\mathbf{x}_k$ are of a different type v. The pair $(\mathbf{x}_i, \mathbf{x}_j)$ is compatible, meaning that the two items appear together in an outfit, while $\mathbf{x}_k$ is a randomly sampled item of the same type as $\mathbf{x}_j$ that has not been seen in an outfit with $\mathbf{x}_i$. Let us write the standard triplet loss in the general form

$$\ell(i, j, k) = \max\{0, d(i, j) - d(i, k) + \mu\} \,, \tag{1}$$

where μ is some margin.

We will denote by $\mathcal{M}^{(u,v)}$ the type-specific embedding space in which objects of types u and v are matched. Associated with this space is a projection $\mathcal{P}^{u \to (u,v)}$ which maps the embedding of an object of type u to $\mathcal{M}^{(u,v)}$. Then, for a pair of data items $(\mathbf{x}_i^{(u)}, \mathbf{x}_j^{(v)})$ that are compatible, we require the distance $||\mathcal{P}^{u \to (u,v)}(\mathbf{f}(\mathbf{x}_i^{(u)}; \theta)) - \mathcal{P}^{v \to (u,v)}(\mathbf{f}(\mathbf{x}_j^{(v)}; \theta))||$ to be small. This *does not* mean that the embedding vectors $\mathbf{f}(\mathbf{x}_i^{(u)}; \theta)$ and $\mathbf{f}(\mathbf{x}_j^{(v)}; \theta)$ for the two items in the general embedding space are similar - the differences just have to lie close to the kernel of $\mathcal{P}^{u \to (u,v)}$.

This general form requires the learning of 2 $(d \times d)$ matrices per pair of types for a d-dimensional general embedding. In this paper, we investigate two simplified versions: (a) assuming diagonal projection matrices such that $\mathcal{P}^{u \to (u,v)} = \mathcal{P}^{v \to (u,v)} = \text{diag}(\mathbf{w}^{(u,v)})$, where $\mathbf{w}^{(u,v)} \in \mathbb{R}^d$ is a vector of learned

weights, and (b) the same case, but with $\mathbf{w}^{(u,v)}$ being a fixed binary vector chosen in advance for each pairwise type-specific space, and acting as a gating function that selects the relevant dimensions of the embedding most responsible for determining compatibility. Compatibility is then measured with

$$d_{ij}^{uv} = d(\mathbf{x}_i^{(u)}, \mathbf{x}_j^{(v)}, \mathbf{w}^{(u,v)}) = ||\mathbf{f}(\mathbf{x}_i^{(u)}; \theta) \odot \mathbf{w}^{(u,v)} - \mathbf{f}(\mathbf{x}_j^{(v)}; \theta) \odot \mathbf{w}^{(u,v)}||_2^2 \ , \quad (2)$$

where $\odot$ denotes component-wise multiplication, and learned with the modified triplet loss:

$$\mathcal{L}_{\text{comp}}(\mathbf{x}_i^{(u)}, \mathbf{x}_j^{(v)}, \mathbf{x}_k^{(v)}, \mathbf{w}^{(u,v)}; \theta) = \max\{0, d_{ij}^{uv} - d_{ik}^{uv} + \mu\} \ , \quad (3)$$

where μ is some margin.

4.1 Constraints on the Learned Embedding

To regularize the learned notion of compatibility, we further make use of the text descriptions accompanying each item image and feed them as input to a text embedding network. Let the embedding vector outputted by that network for the description $\mathbf{t}_i^{(u)}$ of image $\mathbf{x}_i^{(u)}$ be denoted $\mathbf{g}(\mathbf{t}_i^{(u)}; \phi)$, and substitute $\mathbf{g}$ for $\mathbf{f}$ in ℓ as required; the loss used to learn similarity is then

$$\mathcal{L}_{\text{sim}} = \lambda_1 \ell(\mathbf{x}_j^{(v)}, \mathbf{x}_k^{(v)}, \mathbf{x}_i^{(u)}) + \lambda_2 \ell(\mathbf{t}_j^{(v)}, \mathbf{t}_k^{(v)}, \mathbf{t}_i^{(u)}) \ , \quad (4)$$

where λ_{1-2} are scalar parameters.

We also train a visual-semantic embedding in the style of Han *et al.* [12] by requiring that image $\mathbf{x}_i^{(u)}$ is embedded closer to its description $\mathbf{t}_i^{(u)}$ in visual-semantic space than the descriptions of the other two images in a triplet:

$$\mathcal{L}_{\text{vse}i} = \ell(\mathbf{x}_i^{(u)}, \mathbf{t}_i^{(u)}, \mathbf{t}_j^{(v)}) + \ell(\mathbf{x}_i^{(u)}, \mathbf{t}_i^{(u)}, \mathbf{t}_k^{(v)}) \quad (5)$$

and imposing analogical constraints on $\mathbf{x}_j^{(v)}$ and $\mathbf{x}_k^{(v)}$.

To encourage sparsity in the learned weights $\mathbf{w}$ so that we achieve better disentanglement of the embedding dimensions contributing to pairwise type compatibility, we add an ℓ_1 penalty on the projection matrices $\mathcal{P}^{\cdot \to (\cdot,\cdot)}$. We further use ℓ_2 regularization on the learned image embedding $\mathbf{f}(\mathbf{x}; \theta)$. The final training loss therefore becomes:

$$\mathcal{L}(\mathbf{X}, \mathbf{T}, \mathcal{P}^{\cdot \to (\cdot,\cdot)}, \lambda, \theta, \phi) = \mathcal{L}_{\text{comp}} + \mathcal{L}_{\text{sim}} + \lambda_3 \mathcal{L}_{\text{vse}} + \lambda_4 \mathcal{L}_{\ell_2} + \lambda_5 \mathcal{L}_{\ell_1} \quad (6)$$

where $\mathbf{X}$ and $\mathbf{T}$ denote the image embeddings and corresponding text embeddings in a batch, $\mathcal{L}_{\text{vse}} = \mathcal{L}_{\text{vse}_i} + \mathcal{L}_{\text{vse}_j} + \mathcal{L}_{\text{vse}_k}$, and λ_{3-5} are scalar parameters. We preserve the dependence on $\mathcal{P}^{\cdot \to (\cdot,\cdot)}$ in notation to emphasize the type dependence of our embedding. As Sect. 5 shows, this term has significant effects.

5 Experiment Details

Following Han *et al.* [12] we evaluate how well our approach performs on two tasks. In the *fashion compatibility* task, a candidate outfit is scored as to whether its constitute items are compatible with each other. Performance is evaluated using the average under a receiver operating characteristic curve (AUC). The second task is to select from a set of candidate items (four in this case) in a fill-in-the-blank (FITB) fashion recommendation experiment. The goal is to select the most compatible item with the remainder of the outfit, and performance is evaluated by accuracy on the answered questions.

Datasets. For experiments on the Maryland Polyvore dataset [12] in Sect. 5.1 we use the provided splits which separate the outfits into 17,316 for training, 3,076 for testing, and 1,407 for validation. For experiments using our dataset in Sect. 5.2 we use the different version splits described in Sect. 3. We shall refer to the "easier" split as *Polyvore Outfits*, and the split containing only disjoint outfits down to the item level as *Polyvore Outfits-D*.

Implementation. We use a 18-layer Deep Residual Network [13] which was pretrained on ImageNet [7] for our image embedder with a general embedding size of 64 dimensions unless otherwise noted. Our model is trained with a learning rate of $5e^{-5}$, batch size of 256, and a margin of 0.2. For our text representation, we use the HGLMM Fisher vector encoding [17] of word2vec [22] after having been PCA reduced down to 6000 dimensions. We set $\lambda_3 = 5e^{-1}$ from Eq. (6) for the Maryland Polyvore dataset ($\lambda_3 = 5e^{-5}$ for experiments on our dataset), and all other λ parameters from Eqs. (4) and (6) to $5e^{-4}$.

Sampling Testing Negatives. In the test set provided by Han *et al.* [12], a negative outfit for the compatibility experiment could end up containing only tops and no other items, and a fill-in-the blank question could have an earring be among the possible answers when trying to select a replacement for a shoe. This is a result of sampling negatives at random without restriction, and many of these negatives could simply be dismissed without considering item compatibility. Thus, to correct this issue and focus on outfits that cannot be filtered out in such a manner, we take into account item category when sampling negatives. In the compatibility experiments, we replace each item in a ground truth outfit by randomly selecting another item of the same category. For the fill-in-the-blank experiments, our incorrect answers are randomly selected items from the same category as the correct answer.

Comparative Evaluation. In addition to performance of the state-of-the-art methods reported in prior work, we compare the following approaches:

- **SiameseNet (ours).** The approach of Veit *et al.* [34] which uses the same ResNet and general embedding size as used for our type-specific embeddings.
- **CSN, T1:1.** Learns a pairwise type-dependent transformation using the approach of Veit *et al.* [33] to project a general embedding to a type-specific space which measures compatibility between two item categories.

- **CSN, T4:1.** Same as the previous approach, but where each learned pairwise type-dependent transformation is responsible for four pairwise comparisons (instead of one) which are assigned at random. For example, a single projection may be used to measure compatibility in the (shoe-top, bottom-hat, earrings-top, outwear-bottom) type-specific spaces. This approach allows us to assess the importance of having distinct learned compatibility spaces for each pair of item categories versus forcing the compatibility spaces to "share" multiple pairwise comparisons, thus allowing for better scalability as we add more fine-grained item categories to the model.
- **VSE.** Indicates that a visual-semantic embedding as described in Sect. 4.1 is learned jointly with the compatibility embedding.
- **Sim.** Along with training the model to learn a visual-semantic embedding for compatibility between different categories of items as done with the VSE, the same embeddings are also used to measure similarity between items of the same category as described in Sect. 4.1.
- **Metric.** In the triplet loss, rather than minimizing Euclidean distance between compatible items and maximizing the same for incompatible ones, an empirically more robust way is to optimize over the inner products instead. To generalize the distance metric, we take an element-wise product of the embedding vectors in the type-specific spaces and feed it into a fully-connected layer, the learned weights of which act as a generalized distance function.

Table 2. Comparison of different methods on the Maryland Polyvore dataset [12] using their unrestricted randomly sampled negatives on the fill-in-the-blank and outfit compatibility tasks. "All Negatives" refers to using their entire test split as is, while "Composition Filtering" refers to removing easily identifiable negative samples. The numbers in **(a)** are the results reported from Han *et al.* [12] or run using their code, and **(b)** reports our results

		All negatives		w/Composition filtering	
	Method	FITB accuracy	Compat. AUC	FITB accuracy	Compat. AUC
(a)	SetRNN [18]	29.6	0.53	–	–
	SiameseNet [34]	52.0	0.85	–	–
	Bi-LSTM (512-D) [12]	66.7	0.89	–	–
	Bi-LSTM + VSE (512-D) [12]	68.6	0.90	**81.5**	0.78
(b)	SiameseNet (ours)	54.2	0.85	72.3	0.81
	CSN, T1:1	51.6	0.83	74.9	0.83
	CSN, T1:1 + VSE	52.4	0.83	73.1	0.83
	CSN, T1:1 + VSE + Sim	51.5	0.82	75.1	0.79
	CSN, T4:1 + VSE + Sim + Metric	84.2	0.90	75.7	**0.84**
	CSN, T1:1 + VSE + Sim + Metric	**86.1**	**0.98**	78.6	**0.84**

Table 3. Comparison of different methods on the Maryland Polyvore Dataset [12] on the fill-in-the-blank and outfit compatibility tasks using our category-aware negative sampling method. **(a)** contains the results of prior work using their code unless otherwise noted, and **(b)** contains results using our approach

	Method	FITB accuracy	Compat. AUC
(a)	Bi-LSTM + VSE (512-D) [12]	64.9	**0.94**
	SiameseNet (ours)	54.4	0.85
(b)	CSN, T1:1	57.9	0.87
	CSN, T1:1 + VSE	58.1	0.88
	CSN, T1:1 + VSE + Sim	59.0	0.87
	CSN, T4:1 + VSE + Sim + Metric	59.9	0.90
	CSN, T1:1 + VSE + Sim + Metric	61.0	0.90
	CSN, T1:1 + VSE + Sim + Metric (512-D)	**65.0**	0.93

5.1 Maryland Polyvore

We report performance using the test splits of Han *et al.* [12] in Table 2 where the negative samples were sampled completely at random without restriction. The first line of Table 2(b) contains our replication of the approach of Veit *et al.* [34], using the same convolutional network as implemented in our models for a fair comparison. We see on the second line of Table 2(b) that performance on both tasks using all the negative samples in the test split is actually reduced, while after removing easily identifiable negative samples it is increased. This is likely due to how the negatives were sampled. We only learn type specific embeddings to compare the compositions of items which occur during training. Thus, at test time, if we are asked to compare two tops, but no outfit seen during training contained two tops, we did not learn a type-specific embedding for this case and are forced to compare them using our general embedding. This also explains why our performance drops using the negative samples of Han *et al.*when we include the similarity constraint on the third line of Table 2(b), since we explicitly try to learn similarity in our general embedding rather than compatibility. The same effect also accounts for the discrepancy when we train our learned metric shown in the last two lines of Table 2(b). Although we report much better performance than prior work using all the negative samples, our performance is much closer to the LSTM-based method of Han *et al.* [12] after removing the easy negatives.

Since many of the negative samples of Han *et al.* [12] can be easily filtered out, and thus make it difficult to evaluate our method due to their invalid outfit compositions, we report performance on the fill-in-the-blank and outfit compatibility tasks where the negatives are selected by replacing items of the same category in Table 3. The first line of Table 2(b) shows that using our type-specific embeddings, we obtain a 2-3% improvement over learning a single embedding to compare all types. In the second and third lines of Table 3(b), we see that includ-

Table 4. Effect the embedding size has on performance on the fill-in-the-blank and the outfit compatibility tasks on the Maryland Polyvore dataset [12] using our negative samples

Method	FITB accuracy	Compat. AUC
CSN, T1:1 + VSE + Sim + Metric (32-D)	55.7	0.88
CSN, T1:1 + VSE + Sim + Metric (64-D)	61.0	0.90
CSN, T1:1 + VSE + Sim + Metric (128-D)	62.4	0.92
CSN, T1:1 + VSE + Sim + Metric (256-D)	62.8	0.92
CSN, T1:1 + VSE + Sim + Metric (512-D)	**65.0**	**0.93**

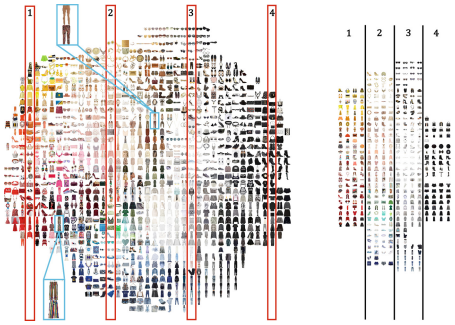

Fig. 2. Left: t-SNE of the learned general embedding space on Polyvore Outfits. We see the learned embedding respects color variations and for types where shape is a unique identifier (*e.g.*, pants and sunglasses) items are more closely grouped together. **Right:** Overlapping items for each cell of the highlighted four columns in the t-SNE plot. Note that each row contains items that are very similar to each other, which suggests a well-behaved embedding. Best viewed in color at high resolution (Color figure online)

ing our visual semantic embedding, along with training our general embedding to explicitly learn similarity between objects of the same category, provides small improvements over simply learning our type-specific embeddings. We also see a pronounced improvement using our learned metric, resulting in a 3-4% improvement on both tasks over learning just the type-specific embeddings. The last line of Table 3(b) reports the results of our approach using the same embedding size as Han *et al.* [12], showing that we obtain similar performance. This is particularly noteworthy since Han *et al.*uses a more powerful feature representation (Inception-v3 [31] vs. ResNet-18), and takes into account the entire outfit when making comparisons, both of which would likely further improve our model. A full accounting of how the dimensions of the final embedding affects the performance of our approach is provided in Table 4.

(a) (b) (c)

Fig. 3. t-SNE of the learned type-specific embedding space on our Polyvore dataset for: **(a)** tops and bags; **(b)** shoes and sunglasses; **(c)** scarves and jewelery. As hypothesized, respecting type allows the embedding to specialize to features that dominate compatibility relationships for each pair of types: for example, color seems to matter more in (a) than in (c), where shape is an equally important feature, with a concentration of long pendants in the lower right and smaller pieces towards the top. Best viewed in color at high resolution (Color figure online)

Fig. 4. Examples of learned compatibility relationships by our model. The query item (top row) is randomly pulled from some type u. Rows shaded in gray show our model's suggestions for items compatible with the query item of a randomly selected type v. Bottom row shows items of type v sampled at random

Fig. 5. Examples of learned similarity relationships by our model. Column outlined in yellow shows the query image for each row. Rows contain similar items of the same type as the query item. Note that the learned notion of similarity is more abstract than matching simply based on color or shape. For example, in the second row, the model is perfectly happy suggesting as alternatives both short- and long-leg pants as well as skirts, so long as the general light colors, flowery patterns and flowy materials are present. Similarly, in the third row, all similar shoes are black but they vastly differ in style (e.g., platform vs. sandal vs. bootie vs. loafer) and all have a unique statement element just like the straps detail of the query item: laces, a golden clasp, yellow detail, metal bridge (Color figure online)

Table 5. Comparison of different methods on the two versions of our dataset on the fill-in-the-blank and outfit compatibility tasks using our category-aware negative sampling method. **(a)** contains the results of prior work using their code unless otherwise noted, and **(b)** contains results using our approach

		Polyvore outfits-D		Polyvore outfits	
	Method	FITB accuracy	Compat. AUC	FITB accuracy	Compat. AUC
(a)	Bi-LSTM + VSE (512-D) [12]	39.4	0.62	39.7	0.65
	SiameseNet (ours)	51.8	0.81	52.9	0.81
(b)	CSN, T1:1	52.5	0.82	54.0	0.83
	CSN, T1:1 + VSE	53.0	0.82	54.5	0.84
	CSN, T1:1 + VSE + Sim	53.4	0.82	54.7	0.85
	CSN, T4:1 + VSE + Sim + Metric	53.7	0.82	55.1	0.85
	CSN, T1:1 + VSE + Sim + Metric	54.1	0.82	55.3	**0.86**
	CSN, T1:1 + VSE + Sim + Metric (512-D)	**55.2**	**0.84**	**56.2**	**0.86**

Table 6. Effect the embedding size has on performance on the fill-in-the-blank and the outfit compatibility tasks using the two versions of our dataset

	Polyvore outfits-D		Polyvore outfits	
Method	FITB accuracy	Compat. AUC	FITB accuracy	Compat. AUC
CSN, T1:1 + VSE + Sim + Metric (32-D)	53.2	0.81	53.9	0.85
CSN, T1:1 + VSE + Sim + Metric (64-D)	54.1	0.82	55.3	**0.86**
CSN, T1:1 + VSE + Sim + Metric (128-D)	54.3	0.83	55.2	**0.86**
CSN, T1:1 + VSE + Sim + Metric (256-D)	54.8	**0.84**	55.6	**0.86**
CSN, T1:1 + VSE + Sim + Metric (512-D)	**55.2**	**0.84**	**56.2**	**0.86**

Fig. 6. Examples of outfit generation by recursive item swaps. The top row represents a valid (*i.e.*, human-curated) outfit. At each step, we replace an item from the starting outfit with one that is of the same type and equally compatible with the rest of the outfit, but different from the removed item. For full figure, refer to the supplementary. Best viewed in color (Color figure online)

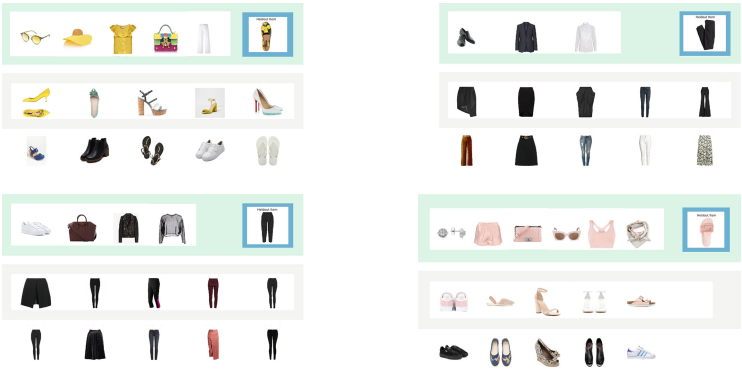

Fig. 7. Examples of item swaps and outfit diversification. Row shaded in green represents a human-curated outfit. Highlighted in blue is a randomly selected heldout item from the outfit to be replaced. Rows shaded in gray displays alternatives that are all equally compatible with the rest of the outfit as the heldout item. The bottom row shows a random selection of alternatives of the same type as the heldout item. The suggested alternatives made by our model in the middle row, although equally compatible with the rest of the items in the outfit, are not forced to be similar to each other but differ vastly in color, style, shape and fine-grained type. Best viewed in color (Color figure online)

5.2 Polyvore Outfits

We report our results on the fill-in-the-blank and outfit compatibility experiments using our own dataset in Table 5. The first line of Table 5(b) shows that learning our type specific embeddings gives a consistent improvement over training a single embedding to make all comparisons. We note that our relative performance using the entire dataset is higher than our disjoint set, which we attribute to likely being due to the additional training data for learning each type-specific embedding. Analogous to the Maryland dataset, the next three lines of Table 5 show a consistent performance improvement as we add in the remaining pieces of our model.

We note that the LSTM-based method of Han *et al.* [12] works relatively poorly on our dataset, which suggests that the limited number of items in the outfits in the Maryland dataset (the maximum length of an outfit sequence is fixed to 8 items) may play an important role in obtaining good performance with their approach. The last line of Table 5(b) reports the performance of our model using the same embedding size as the Han *et al.* [12]. The full effect the embedding dimension has on our approach is provided in Table 6. Additional results including an ablation study can be found in supplementary material.

Interestingly, the two splits of our data obtain similar performance, with the better results using the easy version of our dataset only a little better than on the version where all items in all outfits are novel. This suggests that having unseen outfits in the test set is more important than ensuring there are no shared

items between the training and testing splits, and hence in reproductions of our experiments, using the larger version of our dataset is a fair approach.

Why does respecting type help? We visualize the global embedding space and some type-specific embedding spaces with t-SNE [21]. Figure 2 shows the global embedding space; Fig. 3 shows three distinct type-specific embedding spaces. Note how the global space is strongly oriented toward color matches (large areas allocated to each range of color), but for example the scarf-jewelery space in Fig. 3(c) is not particularly focused on color representation, preferring to encode shape (long pendants vs. smaller pieces). As a result, local type-specific spaces can specialize in different aspects of appearance, and so force the global space to represent all aspects fairly evenly.

Geometric Queries. In light of the problems pointed out in the introduction, we show that our type-respecting embedding is able to handle the following geometric queries which previous models are unable or ill-equipped to perform. SiameseNet [34] is not able to answer such queries by construction, and it is not straightforward how the approach of Han *et al.* [12] would have to be repurposed in order to handle them. Our model is the first to demonstrate that this type of desirable query can be successfully addressed.

- Given an item $\mathbf{x}_i^{(u)}$ of a certain type, show a collection of items $\{\mathbf{x}_j^{(v)}\}_{j=1}^N$ of a different type that are all compatible with $\mathbf{x}_i^{(u)}$ but dissimilar from each other (see Fig. 4).
- Given an item $\mathbf{x}_i^{(u)}$ of a certain type, show a collection of items $\{\mathbf{x}_j^{(u)}\}_{j=1}^N$ of the same type that are all interchangeable with $\mathbf{x}_i^{(u)}$ but have diverse appearance (see Fig. 5)
- Given a valid outfit $\mathcal{S} = \{\mathbf{x}_k^{(\tau)}\}_{k=1,\tau=1}^{K,\mathcal{T}}$, replace each item $\mathbf{x}_k^{(\tau)}$ in turn with an item $\tilde{\mathbf{x}}^{(\tau)}$ of the same type which is different from $\mathbf{x}_k^{(\tau)}$, but compatible with the rest of the outfit $\mathcal{S}_{\setminus\{\mathbf{x}_k^{(\tau)}\}}$ (see Fig. 6).
- Given an item $\mathbf{x}_i^{(u)}$ from a valid outfit $\mathcal{S} = \{\mathbf{x}_k^{(\tau)}\}_{k=1,\tau=1}^{K,\mathcal{T}}$, show a collection of *replacement* items $\{\mathbf{x}_j^{(u)}\}_{j=1}^N$ of the same type that are all compatible with the rest of the outfit $\mathcal{S}_{\setminus\{\mathbf{x}_i^{(u)}\}}$ but visually different from $\mathbf{x}_i^{(u)}$ (see Fig. 7).

6 Conclusion

Our qualitative and quantitative results show that respecting type in embedding methods produces several strong and useful effects. First, on an established dataset, respecting type produces better performance at established tasks. Second, on a novel, and richer, dataset, respecting type produces strong performance improvements on established tasks over previous methods. Finally, an embedding method that respects type can represent both *similarity* relationships (whereby

garments are interchangeable - say, two white blouses) and *compatibility* relationships (whereby a garment can be combined with another to form a coherent outfit). Visualizing the learned embedding spaces suggests that the reason we obtain significant improvements on the fill-in-the-blank and outfit compatibility tasks over prior state-of-the-art is that different type-specific spaces specialize in encoding different kinds of appearance variation. The resulting representation admits new and useful queries for clothing. One can search for item replacements; one can find a set of possible items to complete an outfit that has high variation; and one can swap items in garments to produce novel outfits.

Acknowledgements:. This work is supported in part by ONR MURI Award N00014-16-1-2007, in part by NSF under Grant No. NSF IIS-1421521, and in part by a Google MURA Award and an Amazon Research Faculty Award.

References

1. Al-Halah, Z., Stiefelhagen, R., Grauman, K.: Fashion forward: forecasting visual style in fashion. In: ICCV (2017)
2. Bell, S., Bala, K.: Learning visual similarity for product design with convolutional neural networks. ACM Trans. Graph. (SIGGRAPH) **34**(4) (2015)
3. Bromley, J., et al.: Signature verification using a "siamese" time delay neural network. In: IJPRAI (1993)
4. Chen, H., Gallagher, A., Girod, B.: Describing clothing by semantic attributes. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7574, pp. 609–623. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33712-3_44
5. Chen, Q., Huang, J., Feris, R., Brown, L.M., Dong, J., Yan, S.: Deep domain adaptation for describing people based on fine-grained clothing attributes. In: CVPR (2015)
6. Corbiere, C., Ben-Younes, H., Rame, A., Ollion, C.: Leveraging weakly annotated data for fashion image retrieval and label prediction. In: ICCV Workshops (2017)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
8. Di, W., Wah, C., Bhardwaj, A., Piramuthu, R., Sundaresan, N.: Style finder: fine-grained clothing style detection and retrieval. In: CVPR Workshops (2013)
9. Garcia, N., Vogiatzis, G.: Dress like a star: retrieving fashion products from videos. In: ICCV Workshops (2017)
10. Gomez, L., Patel, Y., Rusinol, M., Karatzas, D., Jawahar, C.V.: Self-supervised learning of visual features through embedding images into text topic spaces. In: CVPR (2017)
11. Hadi Kiapour, M., Han, X., Lazebnik, S., Berg, A.C., Berg, T.L.: Where to buy it: matching street clothing photos in online shops. In: ICCV (2015)
12. Han, X., Wu, Z., Jiang, Y.G., Davis, L.S.: Learning fashion compatibility with bidirectional LSTMS. In: ACM MM (2017)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
14. He, R., Packer, C., McAuley, J.: Learning compatibility across categories for heterogeneous item recommendation. In: International Conference on Data Mining (2016)

15. Hsiao, W.L., Grauman, K.: Learning the latent "look": unsupervised discovery of a style-coherent embedding from fashion images. In: ICCV (2017)
16. Kiapour, M.H., Yamaguchi, K., Berg, A.C., Berg, T.L.: Hipster wars: discovering elements of fashion styles. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 472–488. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_31
17. Klein, B., Lev, G., Sadeh, G., Wolf, L.: Fisher vectors derived from hybrid Gaussian-Laplacian mixture models for image annotation. In: CVPR (2015)
18. Li, Y., Cao, L., Zhu, J., Luo, J.: Mining fashion outfit composition using an end-to-end deep learning approach on set data. IEEE Trans. Multimed. **19**(8), 1946–1955 (2017)
19. Liu, S., et al.: Hi, magic closet, tell me what to wear! In: ACM MM (2012)
20. Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X.: DeepFashion: powering robust clothes recognition and retrieval with rich annotations. In: CVPR (2016)
21. van der Maaten, L., Hinton, G.E.: Visualizing high-dimensional data using t-SNE. JMLR **9**, 2579–2605 (2008)
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)
23. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: NIPS (2003)
24. Rubio, A., Yu, L., Simo-Serra, E., Moreno-Noguer, F.: Multi-modal embedding for main product detection in fashion. In: ICCV Workshops (2017)
25. Salvador, A., et al.: Learning cross-modal embeddings for cooking recipes and food images. In: CVPR (2017)
26. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: CVPR (2015)
27. Simo-Serra, E., Fidler, S., Moreno-Noguer, F., Urtasun, R.: Neuroaesthetics in fashion: modeling the perception of fashionability. In: CVPR (2015)
28. Simo-Serra, E., Ishikawa, H.: Fashion style in 128 floats: joint ranking and classification using weak data for feature extraction. In: CVPR (2016)
29. Singh, K.K., Lee, Y.J.: End-to-End localization and ranking for relative attributes. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 753–769. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_45
30. Song, Y., Li, Y., Wu, B., Chen, C.Y., Zhang, X., Adam, H.: Learning unified embedding for apparel recognition. In: ICCV Workshops (2017)
31. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
32. Vaccaro, K., Shivakumar, S., Ding, Z., Karahalios, K., Kumar, R.: The elements of fashion style. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology (2016)
33. Veit, A., Belongie, S., Karaletsos, T.: Conditional similarity networks. In: CVPR (2017)
34. Veit, A., Kovacs, B., Bell, S., McAuley, J., Bala, K., Belongie, S.: Learning visual clothing style with heterogeneous dyadic co-occurrences. In: ICCV (2015)
35. Wu, C.Y., Manmatha, R., Smola, A.J., Krahenbuhl, P.: Sampling matters in deep embedding learning. In: ICCV (2017)
36. Xiao, H., Huang, M., Zhu, X.: SSP: semantic space projection for knowledge graph embedding with text descriptions. In: AAAI (2017)
37. Yamaguchi, K., Okatani, T., Sudo, K., Murasaki, K., Taniguchi, Y.: Mix and match: joint model for clothing and attribute recognition. In: BMVC (2015)

38. Yu, A., Grauman, K.: Fine-grained visual comparisons with local learning. In: CVPR (2014)
39. Yu, A., Grauman, K.: Just noticeable differences in visual attributes. In: ICCV (2015)
40. Yu, A., Grauman, K.: Semantic jitter: dense supervision for visual comparisons via synthetic images. In: ICCV (2017)
41. Zhao, B., Feng, J., Wu, X., Yan, S.: Memory-augmented attribute manipulation networks for interactive fashion search. In: CVPR (2017)
42. Zhu, S., Urtasun, R., Fidler, S., Lin, D., Loy, C.C.: Be your own prada: fashion synthesis with structural coherence. In: ICCV (2017)
43. Zhuang, B., Lin, G., Shen, C., Reid, I.: Fast training of triplet-based deep binary embedding networks. In: CVPR (2016)