Learning Binary Code for Personalized Fashion Recommendation

Zhi Lu[†], Yang Hu[§][*], Yunchao Jiang[†], Yan Chen[§], Bing Zeng[§]
School of Information and Communication Engineering
University of Electronic Science and Technology of China
[†]{zhilu,jiangyunchao}@std.uestc.edu.cn, [§]{yanghu,eecyan,eezeng}@uestc.edu.cn

Abstract

With the rapid growth of fashion-focused social networks and online shopping, intelligent fashion recommendation is now in great needs. Recommending fashion outfits, each of which is composed of multiple interacted clothing and accessories, is relatively new to the field. The problem becomes even more interesting and challenging when considering users' personalized fashion style. Another challenge in a large-scale fashion outfit recommendation system is the efficiency issue of item/outfit search and storage. In this paper, we propose to learn binary code for efficient personalized fashion outfits recommendation. Our system consists of three components, a feature network for content extraction, a set of type-dependent hashing modules to learn binary codes, and a matching block that conducts pairwise matching. The whole framework is trained in an end-to-end manner. We collect outfit data together with user label information from a fashion-focused social website for the personalized recommendation task. Extensive experiments on our datasets show that the proposed framework outperforms the state-of-the-art methods significantly even with a simple backbone.

1. Introduction

Fashion-focused social networks have become a vibrant realm where millions of individuals share and post daily fashion-related activities. A huge amount of fashion outfits have been created by users in these communities. Mining desirable fashion outfits from this massive data set is very challenging but critical to the development of these online fashion communities. Therefore, there is a great need for intelligent fashion recommendation techniques. The number of possible outfits grows exponentially with the number of items in each garment category. The storage complexity and the retrieval efficiency of the outfits are essential for the deployment of a fashion recommendation system in practice, which however have not been well addressed before.

[*]The corresponding author is Yang Hu. This work was supported by the National Natural Science Foundation of China (61602090) and the 111 Project (B17008).

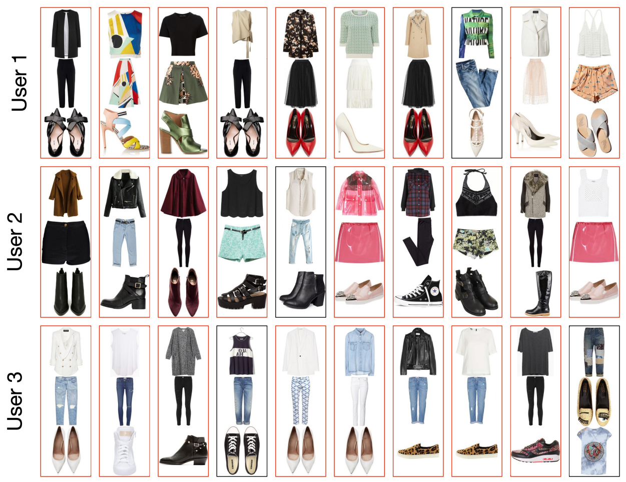


Figure 1. Examples of recommended outfits for three users, where the outfits in red boxes are user-created ones.

Existing related works can be loosely classified into two types based on the way they evaluate how compatible the items in an outfit are. The first type of approaches model pairwise compatibilities between fashion items. Veit *et al.* [28] propose to use a Siamese network to learn the distance between paired items. Hu *et al.* [7] learn a functional factorization and compute the compatibility based on pairwise inner product. Vasileva *et al.* [27] learn type-aware embedding for fashion items and use a fully-connected layer as a generalized distance function for compatibility prediction. The second type of approaches seek to model high order relations among the items of an outfit. Li *et al.* [12] use a recurrent neural network to predict set compatibility. Han *et al.* [3] treat outfits as a sequence of items and compute the compatibility score through LSTM.

The number of items in a fashion inventory is usually very large and the number of outfits that can be composed by these items is orders of magnitude larger. To deploy a practical fashion recommendation system, efficiency becomes an extremely important problem. The items and outfits should be stored in an economical way. The compatibility of an outfit should be evaluated not only accurately but also efficiently. And the most compatible items and outfits need to be retrieved swiftly. These are issues that haven't been well handled by previous works. In this work, we take them into consideration and explore the hash technique for fashion recommendation. Learning to hash has been extensively studies for efficient image retrieval [15]. Some recent works have also combine it with collaborative filtering algorithms to recommend individual items to users [35, 19, 32, 13]. We incorporate it into the task of sets composition problem for personalized outfit recommendation.

We design a neural network for efficient personalized fashion outfit recommendation. The network captures the favors of different users and learn the compatibility between fashion items. It is composed of three components. A feature network is first used to extract content features. Then a set of type-dependent hashing modules convert the features and user taste representations into binary codes. Finally, the overall preference of a user to an outfit is computed by a matching block that conducts pairwise weighted hashing matching. Both visual and textual information are utilized in our model. Since existing datasets are either two small or lacking user information, we collect a new dataset which contains more than 138,000 outfits created by hundreds of online users. We evaluate our method on this large scale dataset. Extensive experiments show that it outperforms the state-of-the-art methods even with a simple backbone and binary representations.

2. Related Work

2.1. Fashion recommendation

Fashion analysis [17], such as clothing recognition [20], latent embedding [24, 5, 10], parsing [2, 31, 14], retrieval [18, 14] and recommendation [16, 8, 21, 28, 6] have attracted many attentions in recent years. Among the plenty of works that studied fashion related problems, we focus on those that related to the recommendation task in the following.

Liu *et al.* [16] introduce a latent SVM based model for occasion-oriented clothing recommendation. McAuley *el al.* [21] propose a parametric distance transformation to learn the item compatibility. Veit *et al.* [28] utilize Siamese network to learn embedding for fashion items. These works do not consider the composition of multiple items in an outfit and only focus on the matching between two items. Some methods seek to directly model the high-order relationships between items. Li *et al.* [12] apply multi-modality fusion and RNN-based multi-instance pooling models to classify the outfit quality. Han *et al.* train a bidirectional LSTM as a scorer to predict the compatibility of an outfit. Vasileva *et al.* [27] learn type-aware mapping for different kinds of items and utilize metric layers to learn the compatibility.

As for personalized composition problem. Hu *et al.* [7] make an initial exploration. A functional tensor factorization method is proposed to model the user-item and item-item interactions in multiple latent spaces. Nevertheless, they use hand-crafted features and do not jointly optimize the representation of images. Hsiao *el. al* [6] propose a subset selection model for selecting a minimal set of garments that maximize the compatibility and versatility, which introduce a new recommendation topic. However, those mentioned models do not consider the efficiency of the system.

2.2. Learning to hash

Learning to hash is the task of learning a compact binary code for the input item. It aims to maintain the nearest neighbor relation of the original space in the hamming space. Hashing methods have become a promising and popular technique for efficient similarity search, which also reduce the storage cost of data. The basic idea in learning to hash is similarity preserving [29], i.e., minimizing the gap between the similarity computed in hash-coded space and the similarity in the original space. Most existing hashing methods first introduce some relaxations to their problems by learning real-valued embedding and then take the sign of the values to obtain binary codes [15, 36, 11], which however often suffer from quantization loss. Recently, Cao *et al.* [1] proposed a method to learn hash codes by continuation with convergence guarantees.

There have been some works reported that apply hashing technique to the recommendation problem [35, 13]. Zhou *et al.* [35] learn binary code that preserves the preference of users to items in collaborative filtering. Lian *et al.* [13] propose a discrete content-aware matrix factorization model. However, these methods cannot be applied directly to the outfit composition problem since they only recommend single items while an outfit contains multiple interacted items. In this work, we model outfit compatibility through pairwise interactions and employ the weighted hashing technique [34, 30, 33] for matching users and items.

3. The Proposed Approach

3.1. Problem formulation

Suppose there are N fashion categories (e.g. top, bottom and shoes). The number of items in the n-th category is donated by L_n and the number of users is U. Let $X^{(n)} = \{x_1^{(n)}, \ldots, x_{L_n}^{(n)}\}$ donate all items in the n-th category, where $x_i^{(n)}$ is the i-th item in it. Then an outfit with N items, with each from one category, can be represented as:

$$\mathcal{O}_{\boldsymbol{i}} = \{x_{i_1}^{(1)}, x_{i_2}^{(2)}, \ldots, x_{i_N}^{(N)}\}, \qquad (1)$$

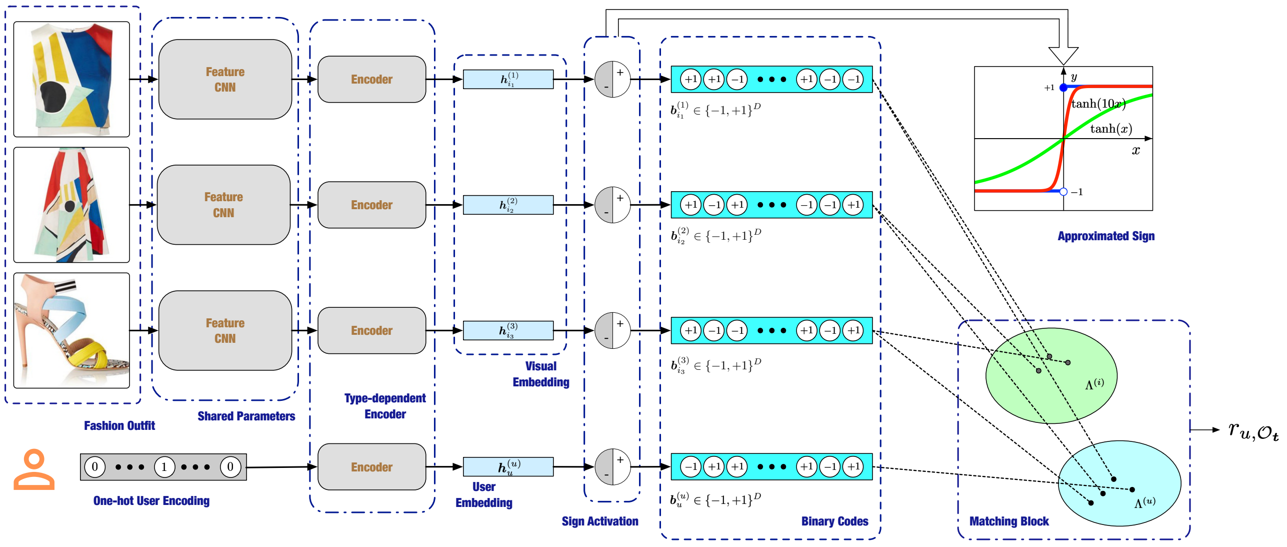


Figure 2. Overall network architecture. All convolutional networks share the same weights. For each type of items, a type-dependent encoder is used to learn binary codes. The matching block is responsible for computing the final preference the user has for the outfit.

where $\boldsymbol{i} = (i_1, \dots, i_N)$ is the index tuple.

The fashion taste can be very different for different users, i.e. the fashion style preference is personal (see Fig. 1). The personalized fashion preference of a certain user can be implicitly represented by outfits the user created or rated in the past. We use $r_{u,\mathcal{O}_i}$ to indicate the preference of user u to outfit $\mathcal{O}_{\boldsymbol{i}}$. The higher the score, the more the user likes that outfit. Our task is to predict $r_{u,\mathcal{O}_i}$ for any user and outfit pair so that the most preferable outfits can be recommended to users.

Without loss of generality, let's assume that an outfit is composed of N items from N categories Then the total number of possible user-outfit pairs is $U \times L_1 \times \cdots \times L_N$, which is an extremely large number in practice. Therefore, we need to compute $r_{u,\mathcal{O}_i}$ in a more feasible way so that some efficient search technique can be applied. In this work, we explore the binary embedding technique, i.e. we seek to represent users and fashion items with binary codes, from which the preference scores are computed.

3.2. Overall framework

We propose a fashion hashing network (*FHN*) to predict the preference of users to different outfit compositions. The overall architecture is shown in Fig. 2. It consists of three types of components, a feature network for feature extraction, several type-dependent hashing modules to learn binary codes, and a matching block to predict the preference score. For input, each user is represented by a one-hot vector, indicating the index of the user. We use a convolutional network to extract visual features for images. The feature networks for different item categories share the same parameters. The specific structure of this feature network is optional to us. Note that textual information can also be incorporated which we will discuss later. Items from different categories and the users are considered as different types. Each hashing module consists of some fully-connected layers with a *sign* function for binarization. The last component is a matching block to compute the preference score given the binary codes. There are two terms contributing to the final score. One only considers item compatibilities and the other takes users' taste into consideration. The detailed formulation is discussed in the following subsections.

3.3. The matching block

The compatibility among the items of an outfit and the fitness to a user are multiway relationships. Although many efforts have been made to the model high order relationships in various applications, the most successful way to handle them is still to decompose it into pairwise relationships, which is easier to learn and has been proven to be a good approximation. Considering the efficiency of computing the preference scores and retrieving compatible outfits for recommendation, we also build our model based on pairwise interaction relations.

Let $\boldsymbol{b}_i, \boldsymbol{b}_j \in \{-1, +1\}^D$ be the binary codes of two objects, which can be a fashion item or a user, their compatibility is measured by

$$m_{ij} = \boldsymbol{b}_i^\intercal \boldsymbol{\Lambda} \boldsymbol{b}_j, \tag{2}$$

where $\boldsymbol{\Lambda}$ is a weighting matrix which we constrain it to be diagonal, i.e. $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_D)$. $\boldsymbol{\Lambda}$ is introduced to better capture the relationship between objects while maintaining the computational efficiency of binary codes. And the score for outfit $\mathcal{O}_{\boldsymbol{i}}$ with respect to user u is computed by

$$r_{u,\mathcal{O}_{\boldsymbol{i}}} = \alpha \cdot r^{(u)}_{u,\mathcal{O}_{\boldsymbol{i}}} + r^{(i)}_{u,\mathcal{O}_{\boldsymbol{i}}}, \tag{3}$$

where

$$r^{(u)}_{u,\mathcal{O}_{\boldsymbol{i}}} = \frac{1}{z_1}\sum_{n} \boldsymbol{b}^{(n)\intercal}_{i_n}\boldsymbol{\Lambda}^{(u)}\boldsymbol{b}^{(u)}_{u}, \tag{4}$$

$$r^{(i)}_{u,\mathcal{O}_{\boldsymbol{i}}} = \frac{1}{z_2}\sum_{n}\sum_{m} \boldsymbol{b}^{(n)\intercal}_{i_n}\boldsymbol{\Lambda}^{(i)}\boldsymbol{b}^{(m)}_{i_m}. \tag{5}$$

In the above formulations, $\boldsymbol{\Lambda}^{(u)}$ and $\boldsymbol{\Lambda}^{(i)}$ are the weighting matrices for user-item and item-item pairs respectively. The Eq. (4) models user's preference to the fashion items and Eq. (5) measures the compatibility between pairs of fashion items. They are normalized by the number of pairs involved, i.e. z_1 and z_2. The scalar α is used to balance the contributions of the two terms. Although α can be absorbed into $\boldsymbol{\Lambda}^{(u)}$, we keep it in Eq. (3) for later discussion of not using weighted hashing, i.e, $\boldsymbol{\Lambda}^{(u)}, \boldsymbol{\Lambda}^{(i)}$ are set to identity matrix $\mathbf{I}$.

3.4. Learn to hash

Directly optimizing an objective function with the binary constraint on $\boldsymbol{b}^{(n)}_{i_n}, \boldsymbol{b}^{(u)}_{u}$ is challenging. It is common to relax the problem by using continuous variables to replace them during optimization. The binary codes are obtained by taking the sign of the continuous variables, i.e.,

$$\boldsymbol{b}^{(n)}_{i_n} = \text{sign}(\boldsymbol{h}^{(n)}_{i_n}), \boldsymbol{b}^{(u)}_{u} = \text{sign}(\boldsymbol{h}^{(u)}_{u}), \tag{6}$$

where $\text{sign}(x) = 1$ if $x \geq 0$ and -1 otherwise.

Many hashing methods learn the hash code by minimizing the quantization error after binarization. Recently, Cao *et al.* [1] propose a *HashNet* method which approximates the $sign$ operation through activation in the neural network. This method avoids explicitly tackling the quantization error. Following this work, we rewrite Eq. (4) and Eq. (5) as follows:

$$r^{(u)}_{u,\mathcal{O}_{\boldsymbol{i}}} = \frac{1}{z_1}\sum_{n} \hat{\boldsymbol{b}}^{(n)\intercal}_{i_n,t}\boldsymbol{\Lambda}^{(u)}\hat{\boldsymbol{b}}^{(u)}_{u,t}, \tag{7}$$

$$r^{(i)}_{u,\mathcal{O}_{\boldsymbol{i}}} = \frac{1}{z_2}\sum_{n}\sum_{m} \hat{\boldsymbol{b}}^{(n)\intercal}_{i_n,t}\boldsymbol{\Lambda}^{(i)}\hat{\boldsymbol{b}}^{(m)}_{i_m,t}, \tag{8}$$

where

$$\hat{\boldsymbol{b}}^{(n)}_{i_n,t} = \tanh(\beta_t \boldsymbol{h}^{(n)}_{i_n}),\ \hat{\boldsymbol{b}}^{(u)}_{u,t} = \tanh(\beta_t \boldsymbol{h}^{(u)}_{u}), \tag{9}$$

and β_t is a scalar that increases with the iteration t during optimization. When β_t is large, Eq. (9) is a close approximation of Eq. (6) (see the diagram in Fig. 2). So $\hat{\boldsymbol{b}}^{(n)}_{i_n,t}$ and $\hat{\boldsymbol{b}}^{(u)}_{u,t}$ converge to the binary codes $\boldsymbol{b}^{(n)}_{i_n}$ and $\boldsymbol{b}^{(u)}_{u}$ when the optimization ends.

3.5. Objective function

Besides images, the fashion items are usually also depicted by some textual descriptions when exhibited online. These textual descriptions provide semantic information for the images, which would be very helpful for compatibility modeling. We therefore also use features extracted from textual content in out model.

We use the same way to convert textual features into binary codes and compute the corresponding preference scores. Suppose binary codes from different modalities are donated by $\boldsymbol{b}^{(n)}_{v,i_n}, \boldsymbol{b}^{(n)}_{f,i_n}$, where v and f indicate visual and textual respectively. The overall score with multi-modality information is computed by adding the scores of each modality:

$$r_{u,\mathcal{O}_{\boldsymbol{i}}}(\{\boldsymbol{b}^{(n)}_{v,i_n}\}, \boldsymbol{b}^{(u)}_{u}) + r_{u,\mathcal{O}_{\boldsymbol{i}}}(\{\boldsymbol{b}^{(n)}_{f,i_n}\}, \boldsymbol{b}^{(u)}_{u}). \tag{10}$$

The scores of each modality is computed by Eq. (3) and the same binary codes of users are used for different modalities.

We use ranking loss to learn the parameters of our model, i.e. the network is trained so that given a pair of outfits, it can predict which one is preferable to a user. The training set contains a set of outfit pairs:

$$\mathcal{P} \equiv \{(u, \boldsymbol{i}, \boldsymbol{j}) | r_{u,\mathcal{O}_{\boldsymbol{i}}} > r_{u,\mathcal{O}_{\boldsymbol{j}}}\}, \tag{11}$$

where $(u, \boldsymbol{i}, \boldsymbol{j})$ indicates that user u prefers outfit $\mathcal{O}_{\boldsymbol{i}}$ over $\mathcal{O}_{\boldsymbol{j}}$. We adapt the BPR [23] optimization criterion to maximize the posterior probability of model parameters. The objective can be expressed as:

$$\ell_{BPR} = \sum_{(u,\boldsymbol{i},\boldsymbol{j})\in\mathcal{P}} \log\left(1 + \exp(-(r_{u,\mathcal{O}_{\boldsymbol{i}}} - r_{u,\mathcal{O}_{\boldsymbol{j}}}))\right) \tag{12}$$

Following previous work [3], we also add constraints to make the embeddings of visual and textual information more consistent to each other. For simplicity, suppose $\boldsymbol{v}, \boldsymbol{f}$ are binary codes for items from the two modalities. Their similarity is donated by $s(\boldsymbol{v}, \boldsymbol{f}) = \boldsymbol{v}^\intercal \boldsymbol{f}$. We require that $\boldsymbol{v}$ and $\boldsymbol{f}$ that correspond to the same item should be more similar than those for different items. This is achieved by minimizing the following loss:

$$\begin{aligned}\ell_{VSE} = &\sum_{\boldsymbol{v},k} \max\{0, c - s(\boldsymbol{v}, \boldsymbol{f}) + s(\boldsymbol{v}, \boldsymbol{f}_k)\} + \\ &\sum_{\boldsymbol{f},k} \max\{0, c - s(\boldsymbol{v}, \boldsymbol{f}) + s(\boldsymbol{v}_k, \boldsymbol{f})\},\end{aligned} \tag{13}$$

where $(\boldsymbol{v}, \boldsymbol{f})$ are for the same item. $(\boldsymbol{v}, \boldsymbol{f}_k)$ are for different items and so is $(\boldsymbol{v}_k, \boldsymbol{f})$.

Overall, the objective of the proposed method is:

$$\min_{\Theta} \mathbb{E}(\ell_{BPR} + \lambda \ell_{VSE}), \tag{14}$$

	Polyvore-630		Polyvore-53	
Splits	#Outfits	#Items	#Outfits	#Items
Train	127,326	159,729	10,712	20,230
Test	23,054	45,505	1,944	4,437

	Polyvore-519		Polyvore-32	
Splits	#Outfits	#Items	#Outfit	#Items
Train	83,416	146,475	5,133	14,594
Test	14,654	39,085	898	2,797

Table 1. Statistics of our Polyvore datasets.

where Θ are parameters for the network and λ is a weighting parameter. Θ consists of $\{\Theta_{nn}, \Theta_v, \Theta_f, \Theta_u\}$ where Θ_{nn} are parameters of the feature network and $\Theta_v, \Theta_f, \Theta_u$ are parameters of the encoders for the two modalities and for the users respectively. The optimization problem is solved with continuous relaxation as discussed in Sec. 3.4.

3.6. Implementation details

The structure of the feature network is optional for us. In our experiments, we simply use *AlexNet* [9] as the backbone. To handle images with arbitrarily sizes, we replace all fully-connected layers in *AlexNet* with convolutional layers and an average pooling layer is added to get a fixed feature dimension of 4096. The dimension for textual feature is 2400. The type-dependent hashing modules for items consist of two fully-connected layers. The encoder for users contains one fully-connected layer. Our methods are implemented in PyTorch [22].

4. Polyvore Dataset

4.1. Polyvore-U

Since existing datasets [7, 3, 27] are either too small or lacking the user information, they cannot be used for our personalized fashion outfit recommendation problem. We collect a new dataset from the Polyvore website. We let each outfit contain items from three categories, i.e. top, bottom, and shoes. We created 4 versions of the datasets denoted by Polyvore-U, where U is the number of users. Two datasets i.e. Polvvore-630 and Polyvore-53, contain outfits with a fixed number of items, i.e., each outfit has one and only one item from each category. In the other two datasets, Polvvore-519 and Polyvore-32, the number of items in each outfit is varying, i.e., some outfits may have two tops. The two larger datasets, Polvvore-630 and Polvvore-519 are used for most experiments. Polyvore-53 and Polyvore-32 are reserved to test the user generalization ability of our models. The statistic of our data sets is shown in Table 1.

4.2. Data preparation

We take outfits created by a user as positive outfits for that user. The negative outfits for him/her come from two sources. One is random mixtures of items, and the other is random samples of other users' positive outfits. The second type of negative outfits are more difficult than the first one. Outfit composition methods that do not consider the personalization issue usually fail to distinguish them from positive outfits. For fair comparison, we first only include the easy negative outfits when comparing the performance of different methods. We discuss the results with the hard negative outfits separately in Sec. 5.3. The ratio between negative and positive outfits is set to 10:1 for each user. The number of items in a negative outfit is kept consistent to that in a positive outfit in each dataset. We ensure that this is no overlap of items between the training and testing sets for each user.

5. Experiment Results

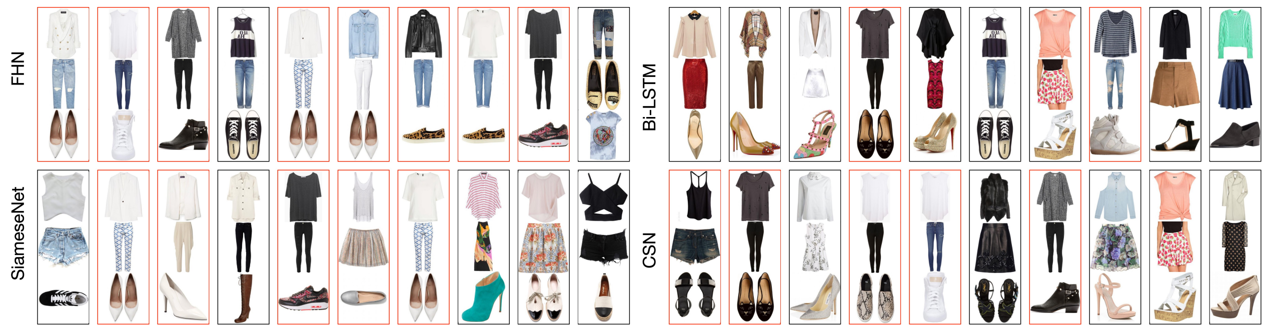
5.1. Evaluation metric

We conduct experiments on two recommendation tasks. The first is outfit recommendation, i.e. for each user we rank the testing outfits in descending order of their compatibility scores. The ranking performance is evaluated by Area Under the ROC curve (AUC) and Normalized Discounted Cumulative Gain (NDCG). The second is the fill-in-the-blank (FITB) fashion recommendation experiment. The goal is to select an item from a set of candidate items (four in our experiments) that is most compatible with the remaining items of the outfit. The ground truth item is the correct answer and the performance is measured by accuracy of the answers. For each experiment, we report the average results over all users.

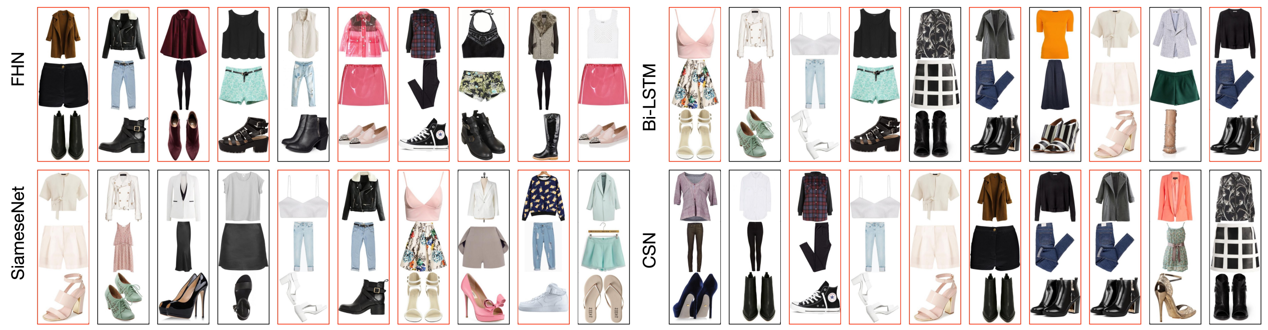
5.2. Performance comparison

We compare variant versions of our models with three state-of-the-art methods:

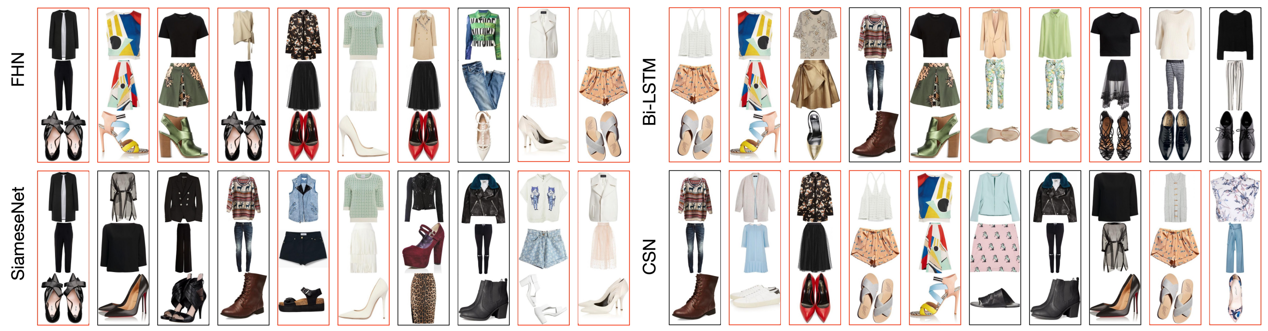
- **SiameseNet** [28] utilizes a *Siamese CNN* to learn a feature transformation to the latent style space, which maintains the matching relationship for pairs of items. The score of an outfit is obtained by averaging the pairwise similarities. The embedding size is set to 512.
- **Bi-LSTM** [3] uses bidirectional *LSTM* to learn the compatibility of an outfit by considering the items as a sequence. Image features are extracted from *Inception-V3* [26] and transformed into representation with 512 dimension before fed into *LSTM*.
- **CSN** [27] maps pairs of items into type-specific embedding spaces. Compatibility is measured by distances in these spaces. An extra distance metric, i.e.



(a) User 1.



(b) User 2.



(c) User 3.

Figure 3. Top-10 outfits with the highest scores computed by different methods on Polyvore-630. The outfits in red boxes are positive outfits and those in black boxes are negative ones.

a weighted inner product, is also learned to replace the Euclidean distance. We use 512 for embedding size and their backbone is *ResNet-18* [4].

- **FHN** is our fashion hashing net. We use *AlexNet* as backbone and use weighted hashing to compute the compatibility between items and users. The length of binary code D is set to 128 for all schemes. We evaluate four different types of weighting schemes:
  - **FHN-T0**: set $\mathbf{\Lambda}^{(u)} = \mathbf{\Lambda}^{(i)} = \mathbf{I}$.
  - **FHN-T1**: set $\mathbf{\Lambda}^{(i)} = \mathbf{I}, \alpha = 1$.
  - **FHN-T2**: set $\mathbf{\Lambda}^{(u)} = \mathbf{I}, \alpha = 1$.
  - **FHN-T3**: set $\alpha = 1$.

We use the two larger datasets Polyvore-630 and Polyvore-519 for the comparison of different methods. The results are shown in Table 2. Here, we also evaluate the contribution of each modality in our model. The textual features are obtained from the descriptions and tags of the items using *seq2seq* [25]. To show the contribution of each modality, we train FHN-T3 with each modality separately and show the results in part (b). And all methods in part (c) employ both visual and textual information. From the results, we can see that all our full-version methods outperform the state-of-the-arts methods under all metrics. Even with only vision features, our model still works better than other methods. By comparing results of the two modalities, we find that the visual information is more helpful than tex-

	Polyvore-630			*Polyvore-519*		
Methods	FITB	AUC	NDCG	FITB	AUC	NDCG
(a) SiameseNet [28]	0.5103	0.7703	0.6109	0.5304	0.8026	0.6648
Bi-LSTM [3]	0.5515	0.8102	0.6629	0.5232	0.7746	0.6210
CSN [27]	0.5536	0.8187	0.6744	0.5617	0.8215	0.6703
(b) FHN-T3 (Textual)	0.5144	0.8441	0.7343	0.4857	0.8188	0.6953
FHN-T3 (Visual)	0.6052	0.8942	0.8090	0.6035	0.8845	0.7784
(c) FHN-T0	0.6066	0.8989	0.8213	0.5770	0.8761	0.7821
FHN-T1	0.6159	0.9016	0.8251	0.6062	0.8892	0.8014
FHN-T2	0.6451	0.9027	0.8296	0.6283	0.8975	0.8184
FHN-T3	**0.6461**	**0.9176**	**0.8541**	**0.6386**	**0.9137**	**0.8448**

Table 2. Comparison of different methods on Polyvore-630 and Polyvore-519.

Dataset	SiameseNe	Bi-LSTM	CSN
Polyvore-630	97.78	94.60	94.76
Polyvore-519	97.11	97.50	96.15

Table 3. Wining rate (%) of FHN-T3 over other methods.

tual in this task. And combining the two modalities leads to better performance than only using one of them.

Overall, the proposed methods with multi-modality get $6.6\% \sim 12.1\%$ improvement in AUC and $19.56\% \sim 26.65\%$ improvement in NDCG when compared with the best results of other methods on the two datasets. To visualize the ranking quality, we show top-10 outfits of three users with the highest scores in Fig. 3. FHN usually has better ranking results. To show how good FHN is when compared to other methods, we define the winning rate as the percentage of users one method outperforms the other in mean NDCG. We show the comparisons in Table 3. It shows that FHN has better ranking results for at least 94% users. The improvement mainly comes from the personalized modeling of users' fashion preferences. It is also beneficial to use the BPR optimization criterion which explicitly takes ranking into consideration. FHN-T0 uses unweighted hashing in Eq. (3), which leads to a relatively poor performance as shown in Table 2. And as expected, adding weighting to hashing improves the results. Without loss of generality, in the following, we only consider FHN-T3 and make it the default setting for analysis.

5.3. Performance on hard outfits

As mentioned in Sec. 4.2, there are two types of negative outfits. A challenging case is to use the outfits that are posted by other users as negative outfits for the current user in evaluation. This setting is different from that of previous work where all user created outfits are taken as positive ones. We learn users' preferences through the first term in Eq. (3). Note that only 128 extra bits are introduced to characterize the users. The results are shown in Table 4.

	Polyvore-630-H		Polyvore-519-H	
Methods	AUC	NDCG	AUC	NDCG
SiameseNet	0.4993	0.2808	0.4997	0.2731
Bi-LSTM	0.4992	0.2817	0.4990	0.2739
CSN	0.5000	0.2790	0.4995	0.2740
FHN	0.7654	0.5552	0.7550	0.5369
FHN-H	0.8440	0.6869	0.8361	0.6685

Table 4. Results on hard negative outfits. FHN-H utilizes hard negative outfits during training while FHN does not include hard negative outfits during training.

FHN indicates results obtained without including hard negative outfits during training. And FHN-H involves hard negative outfits in the training set. We can see that the baseline methods perform poorly in this experiment since they regard all hard negatives as positive ones. Our method works much better with the same training set. By including hard negatives during training, the performance can be further improved.

5.4. Learning hashing codes for cold-start users

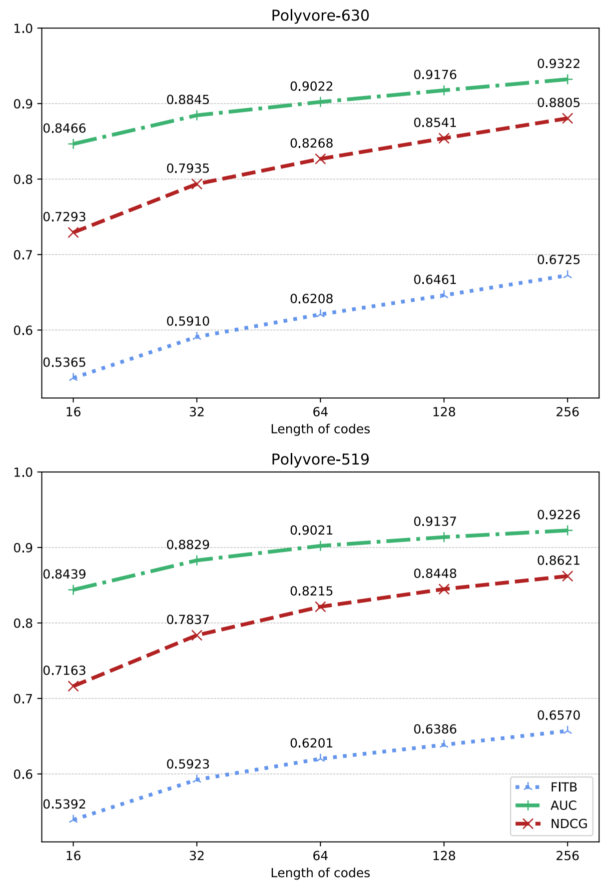
Code start is a common problem in recommendation systems. New users joins constantly in a social network. It will be un-affordable to retrain the whole network for each new user. To tackle this problem, we keep the feature network for items fixed, and only retrain the user representations for newcomers, which is only a 128 bits binary code in our method and can be computed very efficiently. We evaluate a scenario where the system have built a model for 630/529

	Polyvore-53	Polyvore-32
FITB	0.5998	0.5780
AUC	0.8890	0.8911
NDCG	0.8211	0.7970

Table 5. Results of learning hash codes for new users.

	Polyvore-630			Polyvore-519		
Methods	FITB	AUC	NDCG	FITB	AUC	NDCG
FHN w/ Eq. (5)	0.5530	0.8180	0.6637	0.4836	0.8289	0.6949
FHN d/ Eq. (4)	0.5733	0.8333	0.7037	0.5747	0.8334	0.7006
FHN w/ Eq. (4)	0.5062	0.8463	0.7274	0.5578	0.8243	0.6717
FHN d/ Eq. (5)	0.5302	0.8571	0.7609	0.5170	0.8519	0.7449
FHN-T3 in Table 2	0.6461	0.9176	0.8541	0.6386	0.9137	0.8448

Table 6. The contribution of each term in Eq. (3). FHN w/ Eq. (5) is trained only using Eq. (5) and FHN w/ Eq. (4) is trained only using Eq. (4). FHN d/ Eq. (4) and FHN d/ Eq. (5) drop the corresponding term of a trained FHN model.



Length of codes	16	32	64	128	256
Polyvore-630 AUC	0.8466	0.8845	0.9022	0.9176	0.9322
Polyvore-630 NDCG	0.7293	0.7935	0.8268	0.8541	0.8805
Polyvore-630 FITB	0.5365	0.5910	0.6208	0.6461	0.6725
Polyvore-519 AUC	0.8439	0.8829	0.9021	0.9137	0.9226
Polyvore-519 NDCG	0.7163	0.7837	0.8215	0.8448	0.8621
Polyvore-519 FITB	0.5392	0.5923	0.6201	0.6386	0.6570

Figure 4. Comparison of different lengths of codes on Polyvore datasets.

users in the past, and 53/32 new users come. The results are reported in Table 5. Compared with the performance on Polyvoe-630/519, the performance drops are acceptable, i.e., our method can maintain the performance by only fine-tuning the new users' representations even when the size of the dataset has grown around 7%.

5.5. Performance with different lengths of codes

A hashing code with D bits can distinguish at most 2^D objects. Usually, with the increase of code length, the performance will be promoted. Here, we illustrate the influence of code length on our approach. We evaluate a wide range of code lengths, i.e. $\{16, 32, 64, 128, 256\}$, and show their performance comparison in Fig. 4. Taking the AUC for example, the improvement is roughly proportional to the log of the code length. Too short code gets poor result. For example, when $D = 16$, the maximum number of items it can represent is $2^{16} = 65,536$, which is smaller than the number of the items in the datasets we use. Thus, the accuracy drops significantly with $D = 16$.

5.6. Ablation analysis

As presented in Eq. (3), we argue that the ranking score consists of both item-item and user-item compatibilities. To evaluate the contribution of each term, we do ablation study by training with only one term. If we only use Eq. (5), it degenerates to an unpersonalized outfit composition method. And if only using Eq. (4), it only captures user's preference to individual items, and the compatibility between items would not be modeled. Besides, we also evaluate the performance after dropping one term after a full FHN model is trained. This is to make sure that no term overtakes the other in FHN. The results are shown in Table 6. We can see that all results are worse than the full FHN model. This demonstrates that every term is indispensable in the model.

6. Conclusion

In this paper, we study how to utilize the hashing technique for efficient personalized fashion outfit recommendation. Although there are numerous ways to represent the compatibility of outfits, this problem needs to be well handled to fit into hashing optimization. We propose a formulation based on weighted pairwise relations. We design category-dependent hashing mapping for items and users, and train the whole framework in an end-to-end manner. Meanwhile, we use a simple way to combine multi-modality information to improve the performance. Through extensive experiments on a large scale Polyvore dataset, we show the superiority of the proposed method over the state-of-the-art methods even with a simple backbone and binary representation.

References

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. HashNet: Deep Learning to Hash by Continuation. In *ICCV*, 2017.

[2] Jian Dong, Qiang Chen, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Towards Unified Human Parsing and Pose Estimation. In *CVPR*, 2014.

[3] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning Fashion Compatibility with Bidirectional LSTMs. In *ACM MM*, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[5] Wei-Lin Hsiao and Kristen Grauman. Learning the Latent "Look": Unsupervised Discovery of a Style-Coherent Embedding from Fashion Images. In *ICCV*, 2017.

[6] Wei-Lin Hsiao and Kristen Grauman. Creating Capsule Wardrobes from Fashion Images. In *CVPR*, 2018.

[7] Yang Hu, Xi Yi, and Larry S Davis. Collaborative Fashion Recommendation: A Functional Tensor Factorization Approach. In *ACM MM*, 2015.

[8] Vignesh Jagadeesh, Robinson Piramuthu, Anurag Bhardwaj, Wei Di, and Neel Sundaresan. Large Scale Visual Recommendations From Street Fashion Images. In *KDD*, 2014.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*, 2012.

[10] Hanbit Lee, Jinseok Seol, and Sang-goo Lee. Style2Vec: Representation Learning for Fashion Items From Style Sets. *arXiv*, 2017.

[11] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature Learning Based Deep Supervised Hashing with Pairwise Labels. In *IJCAI*, 2016.

[12] Yuncheng Li, Liangliang Cao, Jiang Zhu, and Jiebo Luo. Mining Fashion Outfit Composition Using an End-to-End Deep Learning Approach on Set Data. *TMM*, 2017.

[13] Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. Discrete Content-aware Matrix Factorization. In *KDD*, 2017.

[14] Xiaodan Liang, Liang Lin, Wei Yang, Ping Luo, Junshi Huang, and Shuicheng Yan. Clothes Co-Parsing Via Joint Image Segmentation and Labeling With Application to Clothing Retrieval. *TMM*, 2016.

[15] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep Supervised Hashing for Fast Image Retrieval. In *CVPR*, 2016.

[16] Si Liu, Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan. Hi, Magic Closet, Tell Me What to Wear! In *ACM MM*, 2012.

[17] Si Liu, Luoqi Liu, and Shuicheng Yan. Fashion Analysis: Current Techniques and Future Directions. *IEEE MultiMedia*, 2014.

[18] Si Liu, Zheng Song, Guangcan Liu, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. Street-to-Shop: Cross-Scenario Clothing Retrieval via Parts Alignment and Auxiliary Set. In *CVPR*, 2012.

[19] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. Collaborative Hashing. In *CVPR*, 2014.

[20] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *CVPR*, 2016.

[21] Julian J McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*, 2015.

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*, 2009.

[24] Edgar Simo-Serra and Hiroshi Ishikawa. Fashion Style in 128 Floats: Joint Ranking and Classification Using Weak Data for Feature Extraction. In *CVPR*, 2016.

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*, 2014.

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016.

[27] Mariya I Vasileva, Bryan A Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David A Forsyth. Learning Type-Aware Embeddings for Fashion Compatibility. In *ECCV*, 2018.

[28] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning Visual Clothing Style with Heterogeneous Dyadic Co-Occurrences. In *ICCV*, 2015.

[29] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A Survey on Learning to Hash. *TPAMI*, 2017.

[30] Qifan Wang, Dan Zhang, and Luo Si. Weighted Hashing for Fast Large Scale Similarity Search. In *CIKM*, 2013.

[31] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Retrieving Similar Styles to Parse Clothing. *TPAMI*, 2015.

[32] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete Collaborative Filtering. In *SIGIR*, 2016.

[33] Jian Zhang and Yuxin Peng. Query-Adaptive Image Retrieval by Deep-Weighted Hashing. *TMM*, 2018.

[34] Lei Zhang, Yongdong Zhang, Jinhu Tang, Ke Lu, and Qi Tian. Binary Code Ranking with Weighted Hamming Distance. In *CVPR*, 2013.

[35] Ke Zhou and Hongyuan Zha. Learning Binary Codes for Collaborative Filtering. In *KDD*, 2012.

[36] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep Hashing Network for Efficient Similarity Retrieval. In *AAAI*, 2016.