Personalized Fashion Recommendation with Discrete Content-based Tensor Factorization

Zhi Lu, Yang Hu†, Cong Yu, Yunchao Jiang, Yan Chen, *Senior Member, IEEE,* and Bing Zeng, *Fellow, IEEE*

***Abstract*—Fashion outfit recommendation has attracted lots of attention recently. The problem becomes even more interesting and challenging when considering users' personalized fashion preferences. Although existing works have successfully improved the recommendation accuracy, the efficiency issue of computation and storage is still under-investigated and often ignored. In this paper, we propose a discrete content-based tensor factorization model that maps items and user to binary codes for efficient fashion recommendation. We introduce a probabilistic perspective for learning to hash, where the binary codes are sampled from a set of underlying Bernoulli variables. To demonstrate the effectiveness of our model, we collect a large-scale outfit dataset together with user label information from a fashion-focused social website. Extensive experiments on our dataset show that the proposed model outperforms other state-of-the-art methods.**

***Index Terms*—fashion analysis, personalized recommendation, outfit recommendation, binary code, learn to hash.**

I. Introduction

PERSONALIZED outfit recommendation has attracted lots of attention recently. Since the number of possible outfits explodes with the number of items in each fashion category, the storage complexity and retrieval efficiency of outfits are essential for the deployment of fashion recommendation systems. 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. In this work, we take both accuracy and efficiency into consideration and explore the hash technique for fashion outfit recommendation.

A fashion outfit consists of a set of items from different fashion categories. Let $\boldsymbol{x}_i$ represent the i-th item in an outfit, respectively. We use $f(u, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ to evaluate the compatibility among n items for user u. Our goal is to learn u and $\boldsymbol{x}_i$, which are in binary forms, together with a carefully designed score function f. We give examples of outfits recommended by our approach in Fig. 1. By comparing the outfits created by different users, we can see that different users have different fashion preferences. In this task, we need not only to learn the compatibility among fashion items, but also to capture the diversified tastes of different users.

Fig. 1. Examples of recommended fashion outfits for different users. Outfits in red boxes are from the set of user created outfits and those in black boxes are randomly generated. We can also see that users have diverse tastes for fashion outfits.

Existing works on outfit compatibility prediction are different in the ways they evaluate the compatibility score. Decomposing the multiway relationship into a set of pairwise matches provides an explicit way to model the compatibility. For example, Veit *et al.* [1] use Siamese networks to learn the similarity between two items. In personalized outfit recommendation, Hu *et al.* [2] learn a factorization model based on item-item and item-user pairs. Recent works [1], [3]–[5] improve the pairwise model by using different conditional similarities. Although the decomposed models achieve promising results, recent works are also interested in modeling items in an outfit as a whole to capture the higher-order relationships. Li *et al.* [6] use a feature fusion model to represent an outfit by a single vector to predict the compatibility score. Han *et al.* [7] treat an outfit as a sequence of items and compute the compatibility score through a Bi-LSTM. Besides, graph-based approaches [8]–[10] are also promising candidates to capture underlying relationships among items. Collaborative filtering signal is often required in these approaches. However, in practice, an item only appears in very few outfits, and outfits created by different users seldom overlap, which make collaborative filtering approaches prone to fail. Other graph-based methods may require additional side information, which, however, is not always available. The personalized outfit

Zhi Lu, Cong Yu, Yunchao Jiang and Bing Zeng are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: {zhilu,congyu,jiangyunchao}@std.uestc.edu.cn; eezeng@uestc.edu.cn).

Yang Hu is with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: eeyhu@ustc.edu.cn).

Yan Chen is with the School of Cyber Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: eecyan@ustc.edu.cn).

†Corresponding author: Yang Hu. This work was supported by the National Natural Science Foundation of China under Grant 62172381.

recommendation problem is still under-investigated, especially in the context of computation efficiency.

In nearest neighbor search, hashing [11] is one of the most widely used methods to address the computation and storage efficiency problem, which has been extensively studied in image retrieval [12]–[14]. Since directly optimizing an objective function with the binary constraints is challenging, it is common to relax the problem by using continuous variables [12]. To improve the performance, recent deep hashing approaches usually decompose the original similarity preserving problem into sub-problems [15], [16]. The binary codes and continuous variables are optimized alternatively, where the binary codes are learned with similarity labels and the continuous features are learned with deep neural networks. Recently, the hashing techniques have also been explored for recommendation systems [17]–[21]. But most works have only considered the recommendation of single item. For a set of items, we need to model not only the user-item relationship but also the compatibility between items, which makes incorporating hashing into the system quite challenging.

We present a novel deep hashing approach in this paper that samples binary codes from a set of underlying Bernoulli variables. The motivation of our method is similar to that of local sensitive hashing (LSH) [22], in which the more compatible the items are, the more likely the same binary codes will be sampled. The Bernoulli variable is used to represent the confidence of being encoded to a certain code. As a result, we can sample multiple binary codes for each Bernoulli variable without having to retrain the entire model to improve the performance. Because we treat the binary code as a realization of the underlying distribution, we don't need to include the commonly used quantization step, which would limit the model's performance. We use the feature representations as the underlying distributions directly, allowing the model to take advantage of modern representation learning techniques. Furthermore, because most of the items and outfits would be cold starters during testing, we use a discrete content-based factorization method to decompose the multiway relationships between user and items in this paper.

A preliminary version of this work has been presented in [23]. We extend the original version in the following aspects for completeness. (1) We propose a novel deep hashing approach by treating the binary codes as samples of underlying distributions, which provides a controllable trade-off between efficiency and accuracy; (2) We evaluate more state-of-the-art methods and show more recommendation results; (3) We investigate our model on item suggestion task, which shows its capability to find compatible substitutes; (4) We conduct a time complexity analysis and demonstrate experimentally that the inference time can be reduced by orders of magnitude.

II. Related Work

Fashion related applications [24]–[31] have received significant attention in recent years. Among the numerous works, we focus on those related to the recommendation task.

A. Fashion recommendation

Existing works on outfit recommendation differ in how they evaluate the compatibility score among a set of items. One explicit way is to decompose outfit compatibility into pairwise interactions between items. McAuley *el al.* [32] learn the compatibility of items with personalized weights in a single latent space. Veit *et al.* [33] propose a conditional similarity network (CSN) for different types of items. Vasileva *et al.* [3] learn the type-aware compatibility for item pair in multiple conditioned similarity subspaces. Yang *et al.* [5] consider the conditional similarity using translation-based distance. Tan *et al.* [4] learn different conditional embeddings with different weights for items without using the categorical information. Liu *et al.* [21] learn binary codes for fashion items by solving tractable sub-problems. In personalized outfit recommendation, Hu *et al.* [2] propose a factorization method to model the compatibility under pairwise interactions, in which additional user-item pairs are used to model the personalized preferences.

To capture high-order relationships among items, most of the recently works treat the outfit as a whole. Li *et al.* [6] apply pooling method for items to classify the outfit popularity. Tangseng *et al.* [34] learn outfit embedding by concatenating all items. Han *et al.* [7] propose a LSTM-based approach to model the outfit as a sequence. Yang *et al.* [35] improve the sequence modeling approach by considering categorical information. Besides, Jing *et al.* [36] propose a low-rank representation approach to directly optimize the outfit embeddings. Graph-based approaches have also attracted a lot of interests recently since it is capable to capture the underlying structure among items. Cucurull *et al.* [8] treat item compatibility measurement as a link prediction problem and decomposed the item relationships into pairwise links. They construct item graph from pairs of items in outfits. Cui *et al.* [9] use the frequency of co-occurring fashion categories to build the graph for items and learn the outfit embedding by features aggregation. Based on this graph, Li *et al.* [10] further consider the user-outfit structure to do personalized recommendation. However, the graph-based approaches are computationally intensive and the requirements for building graphs may not be met in practice.

Among the above methods, only a few of them have studied the problem of personalized outfit commendation. Moreover, most of these works have not considered the efficiency issue in fashion recommendation.

B. Learning to hash

The hashing approach enjoys a surge of interest for accelerating nearest neighbor search [11], [37]. Learning to hash aims to maintain the nearest neighbor relation of original space in the hamming space by utilizing properties such as data distribution [38] and manifold structure [39]. Recently, deep learning based hashing methods have shown superiority over traditional ones [12], [13], [40], [41]. Typically, these methods simultaneously learn the feature representation and hash codes. However, due to the continuous relaxation, there exists discrepancy between the optimization and the hashing objectives, which leads to obvious performance degradation.

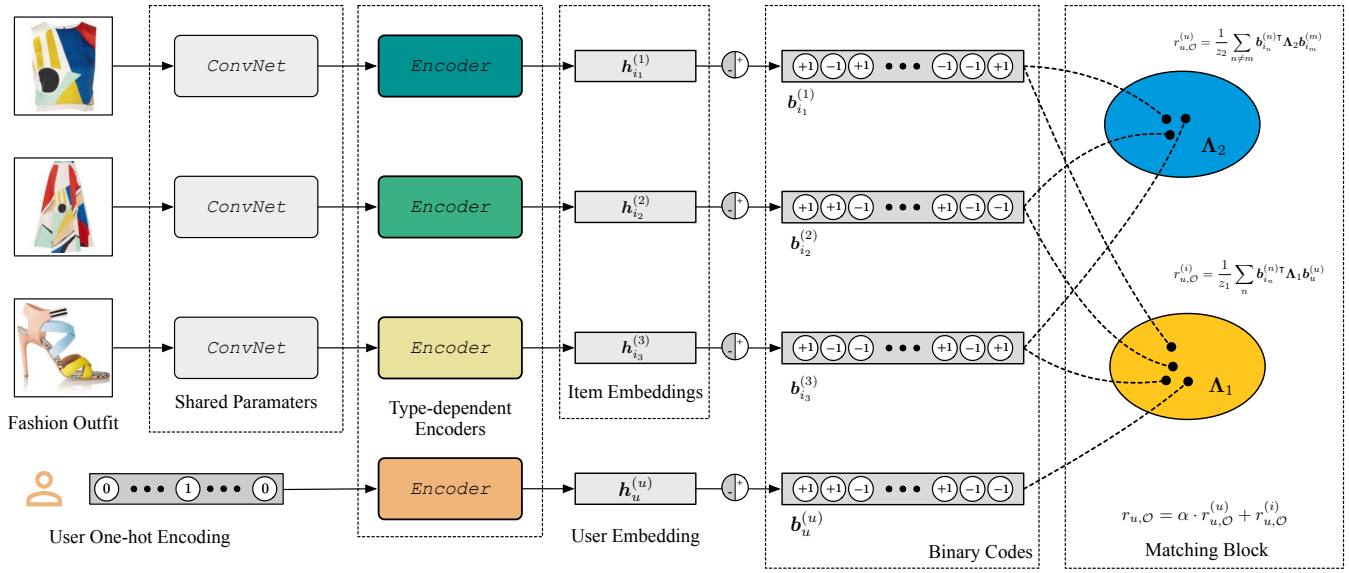


Fig. 2. Overall network architecture of our approach. The model consists of three components: feature network, type-dependent encoders and matching block. We use convolutional networks to extract the features from item images. All convolutional networks share the same weights. For each type of items and users, there is a type-dependent encoder for binary code learning. The input to item encoder is the features of each item. And the input to user encoder is an index which is represented with one-hot encoding. Given the binary codes of user and items, the matching block is responsible for computing the final preference score for this user outfit pair.

To improve the performance, recent deep hashing approaches usually decompose the original similarity preserving problem into sub-problems [15], [16]. The binary codes and continuous variables are optimized alternatively, where the binary codes are learned with similarity labels and the continuous features are learned with deep neural networks.

Existing hashing algorithms can be extended to learn the user-item relationships. And there are works that have applied hashing technique to collaborative filtering [19], [20], [42], [43]. Zhou *et al.* [42] solve the optimization problem by first relaxing the solution to $[-1, 1]$ and then conducting value rounding. To improve the code quality, they propose orthogonal transformations for value rounding, which outperforms the closest rounding strategy. Zhang *et al.* [18] learn the hash codes by minimizing binary reconstructive error with the code balance and un-correlation constraints. The authors propose an alternating optimization approach to solve the relaxed problem. Lian *et al.* [43] propose a discrete content-based matrix factorization model in the scenario that users and items have their content information. Liu *et al.* [21] apply hash technique in fashion dataset with the features extracted from CNNs. However, these methods cannot be applied directly to the outfit composition problem since they only consider one-to-one retrieval while an outfit contains multiple interacted items. In this work, we learn the outfit compatibility through pairwise interactions and employ the weighted hashing [44] technique for matching users and items.

III. Discrete Content-based Tensor Factorization

A. Overall framework

Let N be the number of fashion categories. The number of items in the n-th category is denoted as L_n and the number of users is U. Without loss of generality, we assume that an outfit is composed of N items from different categories, i.e.

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

where $\boldsymbol{i} = (i_1, \ldots, i_N)$ and $x_{i_n}^{(n)}$ is the i_n-th item in the n-th category. The user-outfit pair is represented as an index tuple $(u, \mathcal{O}_{\boldsymbol{i}})$. The total number of possible user-outfit pairs is $U \times L_1 \times \cdots \times L_N$, which is extremely large in practice. Therefore, we need to compute the compatibility $r_{u,\mathcal{O}_{\boldsymbol{i}}}$ in a more feasible way so that efficient search techniques can be applied.

In general, our model is composed of three components: a feature network, a set of type-dependent hashing modules and a matching network as shown in Fig. 2. The feature network is used to extract compact features of fashion items. The type-dependent hashing modules are nonlinear hash functions that convert the features of fashion items and users into a set of underlying distributions for sampling the binary codes. The overall preference score $r_{u,\mathcal{O}}$ of a user to an outfit is computed by a matching block that conducts pairwise weighted hash matching [44]–[46].

B. Compatibility score

Tensor factorization is a natural way to model the high-order relationships, and there exist approaches for fast approximate maximum search for factorized tensors [47], [48]. Formally, let $\boldsymbol{b}_u^{(u)} \in \{\pm 1\}^D$ be the binary code of the u-th user and $\boldsymbol{b}_{i_n}^{(n)} \in \{\pm 1\}^D$ be the binary code of the i_n-th item in the n-th category. A widely used tensor factorization is the CANDECOMP/PARAFAC (CP) model [49], which is defined as follows:

$$r_{u,\mathcal{O}_{\boldsymbol{i}}}^{CP} = \sum_{k=1}^{D} b_{u,k}^{(u)} b_{i_1,k}^{(1)} \cdots b_{i_N,k}^{(N)}. \tag{2}$$

Due to the sparsity problem, the high-order relationship $r^{CP}_{u,\mathcal{O}_i}$ could be difficult to learn [50]. Moreover, in practice, each item only appear in very few outfits, which makes the sparsity problem even worse. The fact that most of the items in the test set are cold-start items that do not appear in the training set also adds the difficulty. As a result, we decompose the compatibility into pairwise interactions using a content-based tensor factorization method. The overall score for outfit $\mathcal{O}_i$ with respect to user u is computed by:

$$r_{u,\mathcal{O}_i} = \frac{1}{z_1}\sum_n \boldsymbol{b}_{i_n}^{(n)\intercal}\boldsymbol{b}_u^{(u)} + \frac{1}{z_2}\sum_{n\neq m}\boldsymbol{b}_{i_n}^{(n)\intercal}\boldsymbol{b}_{i_m}^{(m)} \tag{3}$$

where z_1 and z_2 are the number of pairs involved in summation and are used for normalization. We further use weighted hashing [44] to improve performance by re-weighting each bit based on its importance, i.e.

$$r_{u,\mathcal{O}_i} = \frac{1}{z_1}\sum_n \boldsymbol{b}_{i_n}^{(n)\intercal}\boldsymbol{\Lambda}_1\boldsymbol{b}_u^{(u)} + \frac{1}{z_2}\sum_{n\neq m}\boldsymbol{b}_{i_n}^{(n)\intercal}\boldsymbol{\Lambda}_2\boldsymbol{b}_{i_m}^{(m)} \tag{4}$$

where $\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2 \in \mathbb{R}^{D\times D}$ are learnable diagonal matrices for user-item and item-item interactions, respectively. And we assume that the weights are non-negative.

There are two terms contributing to the final score. We use $r^{(i)}$ for user-item pairs and $r^{(u)}$ for item-item pairs. The $r^{(i)}$ term only considers the general compatibility between items while the $r^{(u)}$ term takes users' taste into consideration. The overall score function can be obtained by:

$$r_{u,\mathcal{O}_i} = \alpha \cdot r^{(u)}_{u,\mathcal{O}_i} + r^{(i)}_{u,\mathcal{O}_i} \tag{5}$$

where α is a scalar to balance the two terms. In practice, we can simply let α be a learnable parameter.

IV. Learning to Hash

Without loss of generality, we use $\boldsymbol{b}$ to represent the binary code for items or users. Since directly optimizing an objective function with the binary constraint is challenging, it is common to relax the problem by using continuous variable $\boldsymbol{h}$ to replace $\boldsymbol{b}$. And the quantization error, i.e., $\|\boldsymbol{h}-\boldsymbol{b}\|_1$, is minimized [12] so that the model can produce binary codes for different input data. Recently, Cao *et al.* propose the ScaleTanH [51] activation function that approximates the signum operation. This method avoids explicitly tackling the quantization error. Suppose $\boldsymbol{h}_t$ is the continuous relaxation for binary code $\boldsymbol{b}_t$ at the t-th step. They use $\hat{\boldsymbol{b}}_t = \tanh(\beta_t \cdot \boldsymbol{h}_t)$ to approximate the binary code $\boldsymbol{b}_t$, where β_t is a scalar that grows with t. The scaled activation function is a smooth approximation of the signum function and converge to the signum function as $t \to \infty$. However, as β_t increases, the output saturates and the gradients may not be sufficient to update the weights. A small quantization error does not guarantee good performance. Most importantly, when we want to increase the length of the binary codes for better performance, the model needs to be retrained, which is not efficient. To address these problems, we propose a sampling-based approach for learning to hash.

A. Bernoulli variables

The motivation of local sensitive hashing [22] is to map the data into hash buckets, such that the shorter the original distance, the higher the probability that they fall into the same hash bucket. In our approach, we follow the same idea and map the items into a set of underlying Bernoulli variables where the binary codes are sampled from, so that the more compatible two items are, the higher the probability that they have the same binary codes. In particular, suppose an item is mapped into D underlying Bernoulli variables $\boldsymbol{p} = (p_1, \ldots, p_D)$, such that the k-th code is $+1$ with probability p_k and -1 with probability $1-p_k$. Thus, the probability of items i and j having the same k-th codes is:

$$P(b_{ik} = b_{jk}) = p_{ik}p_{jk} + (1-p_{ik})(1-p_{jk}) \tag{6}$$

Let $\boldsymbol{b} \in \{\pm 1\}^D$ be the D-dimensional binary code, where each bit is sampled according to the distribution $\boldsymbol{p}$. For item i and j, the expectation of the weighted similarity computed by binary codes is:

$$\begin{aligned}\mathbb{E}[\boldsymbol{b}_i^\intercal\boldsymbol{\Lambda}\boldsymbol{b}_j] &= \mathbb{E}[\sum_{k=1}^D \lambda_k b_{ik}b_{jk}] = \sum_{k=1}^D \lambda_k \mathbb{E}[b_{ik}b_{jk}] \\ &= \sum_{k=1}^D \lambda_k P(b_{ik}=b_{jk}) - \lambda_k P(b_{ik}\neq b_{jk}) \\ &= \sum_{k=1}^D 2\lambda_k(2p_{ik}p_{jk} - p_{ik} - p_{jk}) + \mathrm{tr}(\boldsymbol{\Lambda})\end{aligned} \tag{7}$$

where $\boldsymbol{\Lambda} = \{\lambda_1, \ldots, \lambda_D\}$ are non-negative weights for the binary codes. The expectation suggests that we can directly use the extended vector,

$$\boldsymbol{h} = [\boldsymbol{p}, 1-\boldsymbol{p}] \in \mathbb{R}^{2D} \tag{8}$$

as the feature representation for items. The compatibility between items $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ is defined as follows:

$$\begin{aligned}c_{ij} &= \boldsymbol{p}_i^\intercal\boldsymbol{\Lambda}\boldsymbol{p}_j + (1-\boldsymbol{p}_i)^\intercal\boldsymbol{\Lambda}(1-\boldsymbol{p}_j) \\ &= \frac{1}{2}\left(\mathbb{E}[\boldsymbol{b}_i^\intercal\boldsymbol{\Lambda}\boldsymbol{b}_j] + \mathrm{tr}(\boldsymbol{\Lambda})\right),\end{aligned} \tag{9}$$

which shows that $(\boldsymbol{b}_i^\intercal\boldsymbol{\Lambda}\boldsymbol{b}_j + \mathrm{tr}(\boldsymbol{\Lambda}))/2$ is an unbiased estimator for the compatibility score c_{ij}. For the non-weighted case, the compatibility score c_{ij} is the inner product between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$. Thus, we can directly optimize the feature representation $\boldsymbol{h}$ without introducing the quantization constraint, and the binary codes are sampled afterwards.

For implementation, we encode each item into a latent vector $\boldsymbol{v} \in \mathbb{R}^{2D}$ and normalize it into a set of D probability distributions with the Softmax function. For example, the k-th Bernoulli variable is:

$$(h_k, h_{D+k}) = \mathrm{Softmax}(v_k/\tau, v_{k+D}/\tau), k \leq D \tag{10}$$

where τ is the temperature parameter that controls the sharpness of the Bernoulli distribution. Previous work [52] has also treated the binary code as a set of Bernoulli distributions in variational auto-encoder. And ScaleTanH [51] can be seen as a special case with simulated annealing.

B. Binarization

Deterministic binarization, i.e., signum function, is the common binarization operation in most existing works. In this work, from the perspective of Bernoulli variables, we introduce a stochastic binarization that samples binary codes for each Bernoulli variable. The estimator in Eq. (9) indicates that stochastic binarization can be used to estimate the compatibility score, and approach the performance before binarization by sampling more codes for each bit.

Suppose n_s is the number of bits sampled for each Bernoulli variable, the binary code for item i is extended to

$$[\boldsymbol{b}_{i1}, \ldots, \boldsymbol{b}_{in_s}] \in \{\pm 1\}^{n_s \times D} \tag{11}$$

where $\boldsymbol{b}_{il} \in \{\pm 1\}^D$ is th l-the sampled binary code for item i. For stochastic binarization, the compatibility score is evaluated with the extended binary codes.

Let $Y = \sum_{l=1}^{n_s} Y_l$ where $Y_l = (\boldsymbol{b}_{il}^\intercal \boldsymbol{\Lambda} \boldsymbol{b}_{jl} + \text{tr}(\boldsymbol{\Lambda}))/2$. Since we assume that $\boldsymbol{\Lambda}$ is non-negative, then $Y_l \in [0, \text{tr}(\boldsymbol{\Lambda})]$. According to Hoeffding's inequality, given any $\delta > 0$ we have:

$$\Pr(|Y/n_s - c_{ij}| \geq \delta c_{ij}) \leq 2 \exp\left(\frac{-2n_s c_{ij}^2 \delta^2}{\text{tr}(\boldsymbol{\Lambda})^2}\right) \tag{12}$$

where c_{ij} is the compatibility score between $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$. With the increase of n_s, the extended binary codes can recover the compatibility score c_{ij} with lower error. Since the computing of the similarity between binary codes is lightweight, a relatively large n_s is affordable. The quality of feature representations $\boldsymbol{h}_i, \boldsymbol{h}_j$ determines the upper bound of the performance, and we can approach the upper bound by sampling more codes for each bit without re-training the model. This provides a controllable trade-off between accuracy and efficiency.

Besides, a hybrid binarization that utilizes both stochastic binarization and deterministic binarization can also be used for completeness.

C. Objective function

Given a pair of outfits, our network is trained to predict which one is preferable to a user. The training set contains a set of outfit pairs:

$$\mathcal{P} \equiv \{(u, \boldsymbol{i}, \boldsymbol{j}) | r_{u,\mathcal{O}_{\boldsymbol{i}}} > r_{u,\mathcal{O}_{\boldsymbol{j}}}\}, \tag{13}$$

where $(u, \boldsymbol{i}, \boldsymbol{j})$ indicates that user u prefers outfit $\mathcal{O}_{\boldsymbol{i}}$ over $\mathcal{O}_{\boldsymbol{j}}$. We adapt the Bayesian personalized ranking (BPR) [53] criterion to maximize the posterior probability of model parameters. The objective can be expressed as:

$$\mathcal{L}_{rank} = \sum_{(u,\boldsymbol{i},\boldsymbol{j}) \in \mathcal{P}} \log\left(1 + \exp(-(r_{u,\mathcal{O}_{\boldsymbol{i}}} - r_{u,\mathcal{O}_{\boldsymbol{j}}}))\right). \tag{14}$$

where the score function $r_{u,\mathcal{O}_{\boldsymbol{i}}}$ is computed with continuous features. Since we eliminate the binary constraint, the loss function can be directly optimized.

D. Multi-modalities

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 as shown in previous works [3], [7]. Suppose $\boldsymbol{v}, \boldsymbol{f}$ are binary codes for items from the two modalities, and their similarity is denoted by $s(\boldsymbol{v}, \boldsymbol{f}) = \boldsymbol{v}^\intercal \boldsymbol{f}$. The visual-semantic loss is also used to align the representations between two modalities:

$$\begin{aligned}\mathcal{L}_{vse} = &\sum_{\boldsymbol{v},k} \max\{0, c - \boldsymbol{v}^\intercal \boldsymbol{f} + \boldsymbol{v}^\intercal \boldsymbol{f}_k\} + \\ &\sum_{\boldsymbol{f},k} \max\{0, c - \boldsymbol{v}^\intercal \boldsymbol{f} + \boldsymbol{v}_k^\intercal \boldsymbol{f}\},\end{aligned} \tag{15}$$

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

Different from previous works that use the textual information as a regularization term, we further incorporate the preference scores computed from different modalities. We simply use the addition of the scores computed from the two modalities as the final score, where the same user embeddings are shared for different modalities. We use the same framework to convert textual features into binary codes and compute the corresponding preference scores as in Eq. (3) or Eq. (4).

V. Outfit Dataset

A. The Polyvore-Us dataset

TABLE I
STATISTICS OF OUR POLYVORE-US DATASETS.

Split	Polyvore-630		Polyvore-53	
	#Outfits	#Items	#Outfits	#Items
Train	127,326	159,729	10,712	20,230
Test	23,054	45,505	1,944	4,437
Split	Polyvore-519		Polyvore-32	
	#Outfits	#Items	#Outfits	#Items
Train	83,416	146,475	5,133	14,594
Test	14,654	39,085	898	2,797

The Polyvore-based datasets [3], [7] are widely used for fashion outfit recommendation tasks. However, these datasets do not contain user information and cannot be used for our personalized outfit recommendation problem. Thus, we collect a new dataset with user information from the Polyvore website. We constrain each outfit to consist of items from three primary categories, i.e. top, bottom, and shoes. We create 4 different versions of the datasets denoted by Polyvore-U, where U is the number of users. Since an outfit may contain more than one top, e.g a T-shirt and a jacket, we first create two versions of the datasets from the crawled data. The Polyvore-630/53 are created by fixing the number of items, i.e., each outfit has one and only one item from each category. If a user created outfit has more than one top, we generate multiple outfits from it, i.e. one for each top. On the contrary, the Polyvore-519/32 have varying number of tops, i.e, one or 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

TABLE II
COMPARISON OF DIFFERENT METHODS ON POLYVORE-US. WE USE "T" TO REPRESENT THE MODEL USING TEXTUAL FEATURES, "V" TO REPRESENT THE MODEL USING VISUAL FEATURES, AND "VSE" TO REPRESENT A MULTI-MODALITY MODEL.

Methods	Polyvore-630			Polyvore-519		
	FITB	AUC	NDCG	FITB	AUC	NDCG
SiameseNet [33]	0.5501 ± 0.0042	0.8123 ± 0.0004	0.6623 ± 0.0016	0.5621 ± 0.0049	0.8315 ± 0.0003	0.6736 ± 0.0020
Bi-LSTM [7]	0.5610 ± 0.0031	0.8211 ± 0.0004	0.6943 ± 0.0017	0.5977 ± 0.0028	0.8569 ± 0.0004	0.7153 ± 0.0014
Type-Aware [3]	0.5576 ± 0.0026	0.8166 ± 0.0004	0.6756 ± 0.0019	0.5600 ± 0.0041	0.8234 ± 0.0005	0.6801 ± 0.0022
SCE-Net [4]	0.5858 ± 0.0026	0.8400 ± 0.0005	0.7161 ± 0.0012	0.5763 ± 0.0034	0.8348 ± 0.0005	0.6951 ± 0.0019
FHN-VSE w/ Eq. (3)	0.6241 ± 0.0014	0.9000 ± 0.0003	0.8288 ± 0.0009	0.6234 ± 0.0047	0.9033 ± 0.0003	0.8372 ± 0.0023
FHN-VSE w/ Eq. (4)	**0.6628** ± 0.0021	**0.9237** ± 0.0002	**0.8674** ± 0.0009	**0.6575** ± 0.0048	**0.9282** ± 0.0003	**0.8722** ± 0.0015
FHN-T (ScaleTanH)	0.5620 ± 0.0031	0.8686 ± 0.0004	0.7786 ± 0.0019	0.5450 ± 0.0026	0.8666 ± 0.0005	0.7711 ± 0.0021
FHN-T (Deterministic)	0.5718 ± 0.0019	0.8739 ± 0.0003	0.7903 ± 0.0013	0.5655 ± 0.0036	0.8773 ± 0.0005	0.7929 ± 0.0018
FHN-T ($n_s = 3$)	0.5729 ± 0.0021	0.8747 ± 0.0004	0.7936 ± 0.0011	0.5674 ± 0.0043	0.8793 ± 0.0004	0.7969 ± 0.0021
FHN-V (ScaleTanH)	0.6145 ± 0.0022	0.8979 ± 0.0004	0.8178 ± 0.0015	0.6027 ± 0.0024	0.8974 ± 0.0004	0.8106 ± 0.0012
FHN-V (Deterministic)	0.6172 ± 0.0019	0.8992 ± 0.0003	0.8199 ± 0.0015	0.6145 ± 0.0043	0.9032 ± 0.0003	0.8205 ± 0.0014
FHN-V ($n_s = 3$)	0.6263 ± 0.0021	0.9036 ± 0.0004	0.8294 ± 0.0013	0.6191 ± 0.0045	0.9068 ± 0.0006	0.8272 ± 0.0015

the user generalization ability of our models. The statistic of our data sets is shown in Table I.

B. Negative outfits

Since the datasets only contain positive outfits, we need to create negative outfits from existing items. To make the negative outfits more reasonable and challenging, we use the category information of the items to guide the generation of the negatives. Specifically, given a positive outfit, we replace each item in it by sampling an item from the same category to get a negative outfit. Another type of negatives are outfits sampled from other users. This type of negative outfits are more difficult. Outfit composition methods that do not consider the personalization issue usually fail to distinguish them from positive outfits. 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 with that in a positive outfit in each dataset. We ensure that there is no overlap of items between the training and testing sets for each user.

C. Implementation details

In our experiments, we use *AlexNet* [54] as the default backbone. To handle images with arbitrarily sizes, we replace all *FC* layers in *AlexNet* with convolutional layers and an average pooling layer is added to get a fixed feature dimension of 4096. The textual features are obtained from the descriptions and tags of the items using *seq2seq* [55]. The dimension for textual feature is 2400. The type-dependent hashing modules for items consist of two *FC* layers, and the encoder for users contains one *FC* layer. We implement our model and other deep learning based baselines using PyTorch [56]. Instead of fixing the negative sets during training, we re-sample negative outfits in each epoch.

VI. EXPERIMENTAL RESULTS

A. Evaluation metric

We conduct experiments on two different recommendation tasks. The first task 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) [2]. The second is the fill-in-the-blank (FITB) [7] fashion recommendation. The goal is to select the right item from a set of candidate items given an incomplete outfit. Following the previous works, the number of candidate items are set to 4, and the performance is measured by accuracy of the answers. For each task, we report the average results over all users.

B. Comparison methods

We compare our model with the following state-of-the-art methods.

1) **SiameseNet** [33] utilizes a SiameseNet to learn feature representations that maintain the matching relationship for pairs of items. The score of an outfit is obtained by averaging the pairwise similarities.
2) **Bi-LSTM** [7] uses bidirectional LSTM to learn the compatibility of an outfit by considering the items as a sequence.
3) **Type-Aware** [3] maps pairs of items into type-specific embedding spaces, where compatibility is measured. An extra distance metric, i.e. a weighted inner product, is also learned to replace the Euclidean distance.
4) **SCE-Net** [4] learns multiple conditional embeddings for items. The weights for each conditional embedding are computed by attention mechanism.
5) **FPITF** [2] use handcrafted features to learn a functional pairwise interaction model based on tensor factorization.
6) **NGNN** [9] utilizes a fashion graph that was built upon the co-occurrence of fashion categories to improve the performance.
7) **OutfitNet** [57] is an attention-based approach that learns different weights for fashion items.
8) **FHN** is our fashion hashing net. We use weighted hashing with $D = 128$ by default. Besides, we also consider FHN using the ScaleTanH [51] as a counterpart.

TABLE III
COMPARISON WITH DIFFERENT METHODS ON IQON [59] AND POLYVORE-180 [2].

Methods	IQON			Polyvore-180		
	FITB	AUC	NDCG	FITB	AUC	NDCG
SiameseNet [33]	0.4778 ± 0.0048	0.8054 ± 0.0006	0.6532 ± 0.0028	0.5163 ± 0.0041	0.7921 ± 0.0005	0.6219 ± 0.0022
Bi-LSTM [7]	0.5139 ± 0.0037	0.8127 ± 0.0006	0.6650 ± 0.0016	0.5269 ± 0.0011	0.7871 ± 0.0007	0.6267 ± 0.0030
Type-Aware [3]	0.4581 ± 0.0024	0.7910 ± 0.0007	0.6301 ± 0.0035	0.5071 ± 0.0042	0.7776 ± 0.0008	0.6030 ± 0.0027
SCE-Net [4]	0.4877 ± 0.0029	0.8232 ± 0.0005	0.6820 ± 0.0013	0.5148 ± 0.0056	0.7921 ± 0.0010	0.6328 ± 0.0037
NGNN [9]	0.4933 ± 0.0028	0.8636 ± 0.0006	0.7615 ± 0.0021	0.4806 ± 0.0048	0.7753 ± 0.0011	0.5958 ± 0.0023
OutfitNet [57]	0.4291 ± 0.0042	0.8308 ± 0.0005	0.6956 ± 0.0026	0.5451 ± 0.0035	0.8841 ± 0.0006	0.7798 ± 0.0031
FHN (ScaleTanH)	0.5404 ± 0.0062	0.8961 ± 0.0004	0.8018 ± 0.0019	0.5700 ± 0.0020	0.8907 ± 0.0008	0.7827 ± 0.0034
FHN (Deterministic)	0.5582 ± 0.0048	0.9115 ± 0.0003	0.8299 ± 0.0017	0.6187 ± 0.0051	0.9154 ± 0.0006	0.8407 ± 0.0032
FHN ($n_s = 3$)	**0.5637** ± 0.0041	**0.9149** ± 0.0004	**0.8374** ± 0.0013	**0.6220** ± 0.0055	**0.9156** ± 0.0004	**0.8426** ± 0.0033

TABLE IV
WINING RATE (%) OF FHN OVER OTHER METHODS.

Dataset	SiameseNet	Bi-LSTM	Type-Aware	SCE-Net
Polyvore-630	99.2	98.6	98.9	97.3
Polyvore-519	96.7	97.1	97.3	97.3

C. Personalized outfit recommendation

1) Recommendation accuracy: We first use Polyvore-630 and Polyvore-519 to compare the recommendation accuracy. The results are shown in Table II where each method is evaluated 10 times and the negative outfits are regenerated in each run. For baseline methods, we use ResNet-18 [58] as the backbone. To show the contribution of each modality, we train the weighted model with each modality separately. Besides, we also show the corresponding results using ScaleTanH [51] to learn binary codes. From the results, we can see that our FNH-VSE models outperform the state-of-the-arts methods under all metrics. Even with only visual features, our model still works better than other methods with multi-modalities. And the proposed hashing approach outperforms the counterpart using ScaleTanH. By comparing results of the two modalities, we find that the visual information is more helpful than textual information in this task. And combining the two modalities leads to better performance than only using one of them. Besides, the weighted hashing can notably improve the performance as expected. In the following, we only consider the weighted model and make it the default setting for analysis.

To further compare our model with other methods, we compute AUC for each user and show histograms of AUC scores computed by different methods in Fig. 3a. Besides, we show the cumulative histogram of AUC in Fig. 3b which shows that around 80% of the users have the AUC greater than 0.9 in our method. By comparing the winning rate (i.e. the percentage of users that our method has the highest AUC) in Table IV, we notice FHN has better ranking results for at least 96.7% users. The results also show that capturing the user preference is critical for improving the recommendation accuracy. We further show the top-10 recommendation results for two different users in Fig. 4, and our model has notably better ranking results.

In addition, we compare our model with more baselines using pre-trained visual features. We use the Polyvore-180 [2] and IQON [59] datasets for the evaluation of performance. Polyvore-180 is a much smaller Polyvore-based dataset and IQON contains more items in an outfit. The results are shown in Table III. As we see, our models outperform the baselines significantly. Moreover, both results show that our new hashing method outperforms the FHN with ScaleTanH.

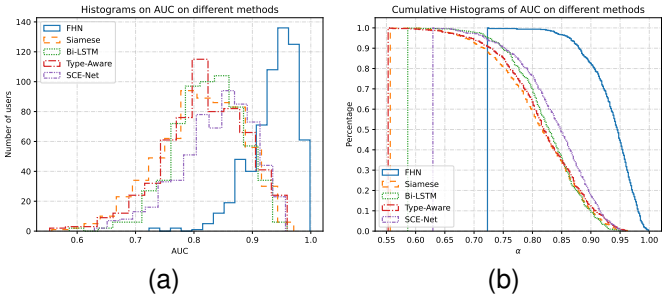


Fig. 3. Histograms of AUC for different methods on Polyvore-630. (a) Histograms of user AUC. (b) Cumulative histograms of user AUC. The points show how many users admit AUC greater than α.

TABLE V
PERFORMANCE ON NEW USERS PROFILING TASK.

Methods	Polyvore-53		Polyvore-32	
	AUC	NDCG	AUC	NDCG
SiameseNet	0.7802	0.5818	0.7891	0.5894
Bi-LSTM	0.8253	0.6740	0.8525	0.7235
Type-Aware	0.8015	0.6367	0.8030	0.6512
SCE-Net	0.8207	0.6570	0.8125	0.6692
FHN	**0.9211**	**0.8688**	**0.9233**	**0.8608**

We further compare our method with FPITF [2], and use the original negative outfits to test the performance. For a fair comparison, we set the length of the binary code to be the same dimension of the latent features in FPITF, i.e. $D = 30$. The NDCG of our method is 0.6231, which achieves 12% performance improvement compared to 0.5576 in FPITF.

2) Learning binary codes for cold-start users: Cold-start is a common problem in recommendation systems. New users join constantly in a social network. It will be unaffordable 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

Fig. 4. Top-10 recommendation results for different users. We show the results on Polyvore-630. We compute the scores of outfits in test set and select outfits with the highest scores computed by different methods. The outfits with red border are positive outfits and those with blank borders are negative ones. Compared with other methods, our method can rank the positive examples better.

TABLE VI
RESULTS ON HARD NEGATIVE OUTFITS. FHN-H UTILIZES HARD NEGATIVE OUTFITS DURING TRAINING, WHILE FHN-R DOES NOT INCLUDE HARD NEGATIVE OUTFITS DURING TRAINING.

Methods	Polyvore-630-H		Polyvore-519-H	
	AUC	NDCG	AUC	NDCG
SiameseNet	0.4993	0.2808	0.4997	0.2731
Bi-LSTM	0.4992	0.2817	0.4990	0.2739
Type-Aware	0.5000	0.2790	0.4995	0.2740
SCE-Net	0.4994	0.2810	0.4996	0.2738
FHN-R	0.7654	0.5552	0.7550	0.5369
FHN-H	0.8440	0.6869	0.8361	0.6685

computed very efficiently. We evaluate a scenario where the system has built a model for 630/519 users, and then 53/32 new users join in. The size of the dataset has grown around 7%. The results are reported in Table V. Compared with the performance on Polyvoe-630/519, we find that our method can maintain the performance by only fine-tuning the new users' representations.

3) Performance on hard outfits: As mentioned in Sec. V-B, 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 current user in evaluation. This setting is different from that of previous works, where all user created outfits are taken as positive ones. We show the comparison results in Table VI. FHN-R indicates results obtained without including hard negative outfits during training. And FHN-H involves hard negative outfits in the training set. Since the baseline methods do not consider the user information, the performance behaves like random guessing. Our method works much better with the same training set. By including hard negatives during training, the data distribution discrepancy between the training and test set is eliminated, and the performance is improved.

D. Personalized item suggestion

In this section, we show the results for the item suggestion task [60]. In this task, given an incomplete outfit, the goal is to fill in the missing item with items in the test set. Some example results are shown in Fig. 5. Query outfits are shown on the left and the items omitted from the outfits are shown on the right. The items in the middle are top 10 suggested results over all items in the test set. Our approach gives a list of compatible items that match the style of the query outfit.

Besides, we quantitatively compare all methods in Table VII where we calculate the normalized position of the original item in the ranked list given the other two items. We report the average results over $1,000$ randomly selected outfits. The smaller the value, the better the results. For each fashion

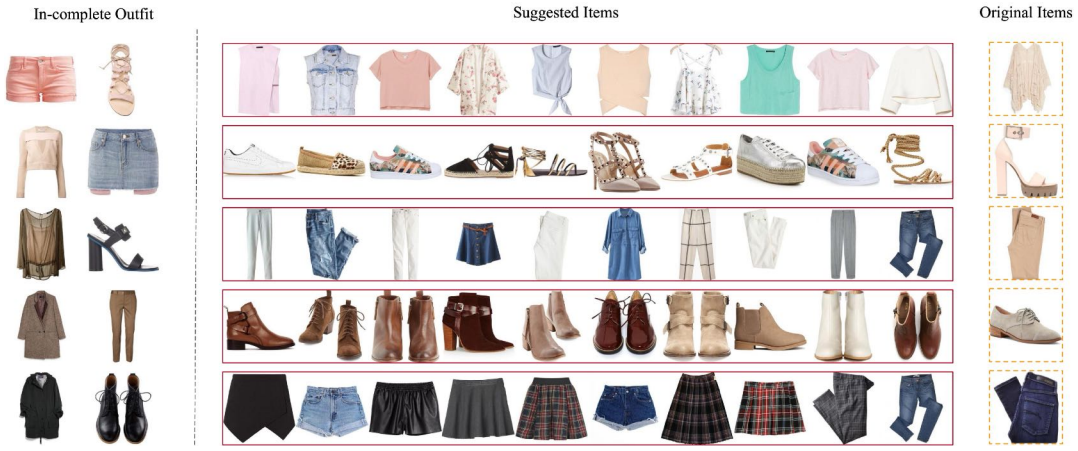


Fig. 5. Examples for item suggestion results of our method. Query outfits are shown on the left where original omitted items are on the right. The items in the middle are top 10 suggested results over all items in the test set. Our approach gives a list of compatible items that match the query outfit.

TABLE VII
NORMALIZED POSITION OF THE ORIGINAL ITEM IN THE ITEM SUGGESTION TASK.

Item	Siamese	Bi-LSTM	Type-Aware	SCE-Net	FHN
Top	26.91	22.38	25.49	23.28	17.88
Bottom	26.81	23.71	26.76	24.16	17.83
Shoe	25.88	22.50	24.01	23.07	17.33

TABLE VIII
THE CONTRIBUTION OF EACH TERM IN COMPATIBILITY SCORE.

Methods	Polyvore-630			Polyvore-519		
	FITB	AUC	NDCG	FITB	AUC	NDCG
Train w/ $r^{(i)}$	0.62	0.86	0.75	0.61	0.86	0.75
Test w/ $r^{(i)}$	0.60	0.85	0.73	0.60	0.85	0.74
Train w/ $r^{(u)}$	0.56	0.88	0.80	0.55	0.89	0.80
Test w/ $r^{(u)}$	0.55	0.87	0.78	0.54	0.88	0.79
$r^{(u)}$ + $r^{(i)}$	0.66	0.92	0.87	0.66	0.93	0.87

category, our method shows a significant improvement over other methods.

E. Ablation analysis

1) On different compatibility terms: As shown in Eq. (5), the compatibility consists of item-item and item-user interactions. The $r^{(i)}$ term only considers the general compatibility between items, while the $r^{(u)}$ term takes users' taste into consideration. To evaluate the contribution of each term, we do ablation study by training with only one term. Besides, we also evaluate the performance by using only one term of a full model. The results are shown in Table VIII. We can see that retraining the model with each term always improve the performance and no term overtakes the other. We can see that all results are worse than the full FHN model. Another interesting finding is that the $r^{(u)}$ term performs better in the ranking task but worse in the FITB task. This demonstrates that every term is indispensable in the model. By combining the two terms, the performance can be greatly improved.

2) On the sharpness of Bernoulli distribution: In Eq. (10), we use τ to define the sharpness of the Bernoulli distribution.

TABLE IX
PERFORMANCE WITH DIFFERENT CHOICE OF τ IN EQ. (10). β_t STANDS FOR OUR METHOD THAT USES SCALETANH [51].

τ	IQON			Polyvore-180		
	AUC	NDCG	$\|\boldsymbol{h}-\boldsymbol{b}\|_1$	AUC	NDCG	$\|\boldsymbol{h}-\boldsymbol{b}\|_1$
0.1	0.8627	0.7417	**0.0101**	0.9118	0.8324	**0.1306**
0.5	**0.9115**	**0.8299**	0.1690	**0.9154**	**0.8407**	0.2021
1.0	0.9107	0.8284	0.1915	0.9114	0.8310	0.2324
2.0	0.9083	0.8233	0.2087	0.9060	0.8219	0.2612
β_t	0.8961	0.8018	0.0139	0.8907	0.7827	0.0036

We train our model with different τ and show the results in Table IX. As τ decreases, the sampled binary code becomes more deterministic and thus the quantization error, i.e. $\|\boldsymbol{h}-\boldsymbol{b}\|_1$, becomes smaller. However, a small quantization error does not necessarily mean good performance. Besides, a large τ may prevent the model to converge. For reference, we show the results of our model that uses ScaleTanH [51] for learning to hash in the last row. This approach always achieves low quantization errors due to the use of an approximation to the signum function. However, the performance is inferior to our stochastic method.

3) On stochastic binarization: A large quantization error suggests that there is a large performance gap between $\boldsymbol{h}$ and $\boldsymbol{b}$. The bound in Eq. (12) shows that the gap can be reduced by sampling more codes for each Bernoulli distribution. Fig. 6 shows the performance with different number of codes sampled from each Bernoulli variable using stochastic binarization and hybrid binarization. We further show the performance of deterministic binarization and that before binarization. As expected, with the increase of n_s for each bit, the performance approaches the upper performance bound before binarization. Note that to get longer binary code we do not need to retrain the model. This is an inbuilt advantage over other methods in which the length of the code needs to be fixed before training.

4) On different length of codes: Usually, with the increase of code length, the performance becomes better. Here, we illustrate the influence of code length on our approach. We evaluate a wide range of code lengths, i.e. $\{16, 32, 64, 128, 256\}$,

TABLE X
RUNNING TIME OF EACH METHOD. WE SHOW THE SPEEDUP OF FHN-OPT OVER OTHER METHODS IN THE SECOND ROW.

	FHN-Opt	FHN	SiameseNet	Bi-LSTM	Type-Aware	SCE-Net
Time	$1.66 \pm 0.07\mu s$	$27.2 \pm 3.07\mu s$	$38.9 \pm 1.90\mu s$	$1.23 \pm 0.04ms$	$59.1 \pm 3.17\mu s$	$192 \pm 8.78\mu s$
Speedup	-	$\sim 16.4\times$	$\sim 23.4\times$	$\sim 741.0\times$	$\sim 35.6\times$	$\sim 115.7\times$

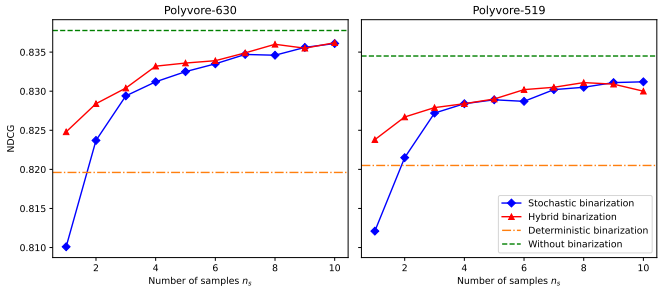


Fig. 6. Performance of stochastic binarization on different number of bits (n_s) sampled for each Bernoulli distribution.

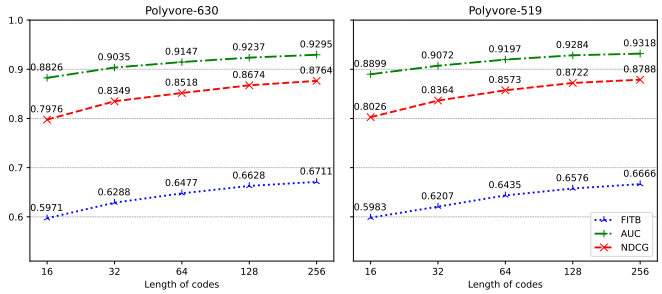


Fig. 7. Performance on different length of codes on Polyvore datasets.

k	v
0	0
1	w_1
2	w_2
3	$w_1 + w_2$
...	...
254	$w_2 + \cdots + w_8$
255	$w_1 + \cdots + w_8$

Fig. 8. Lookup-table for fast computation of the weighted Hamming distance for 8 bits binary code. The table is built given the weight w_i of each bit. k is the look-up key and v is the corresponding value.

and show their performance comparison in Fig. 7. Taking the AUC for example, the improvement is roughly proportional to the log of the code length. Too short code gets poor result. A hashing code with D bits can distinguish at most 2^D objects. 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 items in the datasets we use. Thus, the accuracy drops significantly with $D = 16$.

F. Running Time

In this subsection, we compare the running time of different algorithms for computing the compatibility score of one outfit. Fan *et al.* [61] present a look-up-table strategy for efficiently computing of the weighted hamming distance. Given two 8 bits binary codes $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^8$ and the corresponding weights $\boldsymbol{w} \in [-1, 1]^8$, the bit-wise XOR is conducted to obtain a byte-type value $k = \boldsymbol{a} \otimes \boldsymbol{b}$. Since k has 256 possible values, we can build the corresponding weighted distance v for each possible k in advance using the weight $\boldsymbol{w}$. (as shown in Fig. 8). Thus, to compute the similarity between any two binary codes, we can get the results by one bit-wise XOR operation and table lookup operation, which is significantly faster than inner product between two vectors.

For D-dimensional case, we can firstly divide each binary code into $s = \lceil D/8 \rceil$ partitions and build s lookup-table. And the weighted Hamming distance is simply the sum of all partitions, i.e. $\sum_{i=1}^{s} v_i$ where v_i is the output of the i-th lookup-table. Since the lookup-tables are only built once, the operation is efficient. Besides, increasing the number of bits will only introduce a small additional cost.

We adopt this algorithm and use FHN-Opt and FHN to indicate the methods with and without the use of [61] respectively. All methods are run without parallelism, and the running time for computing one outfit is shown in Table X. We do not include the time consumed for feature extraction. Since pair-wise methods are similar in complexity, their running time is similar. The Bi-LSTM method is the slowest one since it needs to compare all existing items. With a simplified implementation, we get more than 16x speedup with weighted hashing.

VII. CONCLUSION

In this paper, we studied the problem of efficient personalized fashion outfit recommendation. We proposed a discrete content-based tensor factorization method to measure the compatibility of outfits. To properly handle the problem in hashing optimization, we proposed a novel deep hashing approach by treating the binary codes as samples from a set of Bernoulli variables. We directly used features of user and item as the underlying distributions, so that the whole framework was trained in an end-to-end manner. Meanwhile, we used a simple approach that combines multi-modality information to improve performance. A lookup-table approach was used for fast computation of pairwise similarities. Through extensive experiments on large scale Polyvore datasets, we showed the superiority of the proposed method over the state-of-the-art methods even with a simple backbone and binary representation.

REFERENCES

[1] A. Veit, S. Belongie, and T. Karaletsos, "Conditional Similarity Networks," in *CVPR*, 2017, pp. 830–838.

[2] Y. Hu, X. Yi, and L. S. Davis, "Collaborative Fashion Recommendation: A Functional Tensor Factorization Approach," in *ACM MM*, 2015, pp. 129–138.

[3] M. I. Vasileva, B. A. Plummer, K. Dusad, S. Rajpal, R. Kumar, and D. A. Forsyth, "Learning Type-Aware Embeddings for Fashion Compatibility," in *ECCV*, 2018, pp. 390–405.

[4] R. Tan, M. I. Vasileva, K. Saenko, and B. A. Plummer, "Learning Similarity Conditions Without Explicit Supervision," in *ICCV*, 2019, p. 10.

[5] X. Yang, Y. Ma, L. Liao, M. Wang, and T.-S. Chua, "TransNCFM: Translation-Based Neural Fashion Compatibility Modeling," in *AAAI*, 2019, pp. 403–410.

[6] Y. Li, L. Cao, J. Zhu, and J. Luo, "Mining Fashion Outfit Composition Using an End-to-End Deep Learning Approach on Set Data," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1946–1955, 2017.

[7] X. Han, Z. Wu, Y.-G. Jiang, and L. S. Davis, "Learning Fashion Compatibility with Bidirectional LSTMs," in *ACM MM*, 2017, pp. 1078–1086.

[8] G. Cucurull, P. Taslakian, and D. Vazquez, "Context-Aware Visual Compatibility Prediction," in *CVPR*, 2019.

[9] Z. Cui, Z. Li, S. Wu, X.-Y. Zhang, and L. Wang, "Dressing as a Whole: Outfit Compatibility Learning Based on Node-Wise Graph Neural Networks," in *WWW*, 2019, pp. 307–317.

[10] X. Li, X. Wang, X. He, L. Chen, J. Xiao, and T.-S. Chua, "Hierarchical Fashion Graph Network for Personalized Outfit Recommendation," in *SIGIR*, 2020.

[11] X. Luo, C. Chen, H. Zhong, H. Zhang, M. Deng, J. Huang, and X. Hua, "A Survey on Deep Hashing Methods," *arXiv*, 2020.

[12] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep Supervised Hashing for Fast Image Retrieval," *International Journal of Computer Vision*, vol. 127, no. 9, pp. 1217–1234, 2019.

[13] T.-T. Do, T. Hoang, D.-K. Le Tan, A.-D. Doan, and N.-M. Cheung, "Compact Hash Code Learning With Binary Deep Neural Network," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 992–1004, 2020.

[14] H. Liu, R. Wang, S. Shan, and X. Chen, "Learning Multifunctional Binary Codes for Personalized Image Retrieval," *International Journal of Computer Vision*, pp. 1–20, 2020.

[15] Q.-Y. Jiang and W.-J. Li, "Asymmetric Deep Supervised Hashing," in *AAAI*, 2018.

[16] Y. Chen, Z. Lai, Y. Ding, K. Lin, and W. K. Wong, "Deep Supervised Hashing with Anchor Graph," in *ICCV*, 2019, pp. 9796–9804.

[17] X. Liu, J. He, C. Deng, and B. Lang, "Collaborative Hashing," in *CVPR*, 2014, pp. 2139–2146.

[18] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete Collaborative Filtering," in *SIGIR*, 2016.

[19] H. Wang, N. Shao, and D. Lian, "Adversarial Binary Collaborative Filtering for Implicit Feedback," in *AAAI*, vol. 33, 2019, pp. 5248–5255.

[20] D. Lian, X. Xie, and E. Chen, "Discrete Matrix Factorization and Extension for Fast Item Recommendation," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019.

[21] L. Liu, X. Du, L. Zhu, F. Shen, and Z. Huang, "Discrete Binary Hashing Towards Efficient Fashion Recommendation," in *DASFAA*, vol. 10827, 2018, pp. 116–132.

[22] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *STOC*, 1998, pp. 604–613.

[23] Z. Lu, Y. Hu, Y. Jiang, Y. Chen, and B. Zeng, "Learning Binary Code for Personalized Fashion Recommendation," in *CVPR*, 2019, pp. 10 562–10 570.

[24] W.-C. Kang, E. Kim, J. Leskovec, C. Rosenberg, and J. McAuley, "Complete the Look: Scene-based Complementary Product Recommendation," in *CVPR*, 2019.

[25] H. Dong, X. Liang, X. Shen, B. Wang, H. Lai, J. Zhu, Z. Hu, and J. Yin, "Towards Multi-Pose Guided Virtual Try-on Network," in *ICCV*, 2019, pp. 9026–9035.

[26] W. Wang, Z. Zhang, S. Qi, J. Shen, Y. Pang, and L. Shao, "Learning Compositional Neural Information Fusion for Human Parsing," in *ICCV*, 2019, pp. 5702–5712.

[27] C. Yu, Y. Hu, Y. Chen, and B. Zeng, "Personalized Fashion Design," in *ICCV*, 2019, p. 10.

[28] S. C. Hidayati, T. W. Goh, J.-S. G. Chan, C.-C. Hsu, J. See, W. L. Kuan, K.-L. Hua, Y. Tsao, and W.-H. Cheng, "Dress with Style: Learning Style from Joint Deep Embedding of Clothing Styles and Body Shapes," *IEEE Transactions on Multimedia*, 2020.

[29] Z. Ma, J. Dong, Z. Long, Y. Zhang, Y. He, H. Xue, and S. Ji, "Fine-Grained Fashion Similarity Learning by Attribute-Specific Embedding Network," in *AAAI*, vol. 34, 2020, pp. 11 741–11 748.

[30] W.-H. Cheng, S. Song, C.-Y. Chen, S. C. Hidayati, and J. Liu, "Fashion Meets Computer Vision: A Survey," *ACM Computing Surveys*, 2021.

[31] Z. Lu, Y. Hu, Y. Chen, and B. Zeng, "Outlier Item Detection in Fashion Outfit," in *ICIAI*, 2022, pp. 166–171.

[32] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-Based Recommendations on Styles and Substitutes," in *SIGIR*, 2015, pp. 43–52.

[33] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. Belongie, "Learning Visual Clothing Style with Heterogeneous Dyadic Co-Occurrences," in *ICCV*, 2015, pp. 4642–4650.

[34] P. Tangseng, K. Yamaguchi, and T. Okatani, "Recommending Outfits from Personal Closet," in *ICCV*, 2017, pp. 2275–2279.

[35] X. Yang, D. Xie, X. Wang, J. Yuan, W. Ding, and P. Yan, "Learning Tuple Compatibility for Conditional Outfit Recommendation," in *ACM MM*, 2020, pp. 2636–2644.

[36] P. Jing, J. Zhang, L. Nie, S. Ye, J. Liu, and Y. Su, "Tripartite Graph Regularized Latent Low-rank Representation for Fashion Compatibility Prediction," *IEEE Transactions on Multimedia*, 2021.

[37] J. Wang, T. Zhang, N. Sebe, and H. T. Shen, "A Survey on Learning to Hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2017.

[38] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," in *NeurIPS*, 2008.

[39] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with Graphs," in *ICML*, 2011.

[40] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, "Deep Quantization Network for Efficient Image Retrieval," in *AAAI*, 2016, pp. 3457–3463.

[41] W.-J. Li, S. Wang, and W.-C. Kang, "Feature Learning Based Deep Supervised Hashing with Pairwise Labels," in *IJCAI*, 2016, pp. 1711–1717.

[42] K. Zhou and H. Zha, "Learning Binary Codes for Collaborative Filtering," in *KDD*, 2012, pp. 498–506.

[43] D. Lian, R. Liu, Y. Ge, K. Zheng, X. Xie, and L. Cao, "Discrete Content-aware Matrix Factorization," in *KDD*, 2017, pp. 325–334.

[44] Q. Wang, D. Zhang, and L. Si, "Weighted Hashing for Fast Large Scale Similarity Search," in *CIKM*, 2013, pp. 1185–1188.

[45] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian, "Binary Code Ranking with Weighted Hamming Distance," in *CVPR*, 2013, pp. 1586–1593.

[46] J. Zhang and Y. Peng, "Query-Adaptive Image Retrieval by Deep-Weighted Hashing," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2400–2414, 2018.

[47] G. Ballard, T. G. Kolda, A. Pinar, and C. Seshadhri, "Diamond Sampling for Approximate Maximum All-Pairs Dot-Product (MAD) Search," in *ICDM*, 2015, pp. 11–20.

[48] Z. Lu, Y. Hu, and B. Zeng, "Sampling for Approximate Maximum Search in Factorized Tensor," in *IJCAI*, 2017, pp. 2400–2406.

[49] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[50] S. Rendle and L. Schmidt-Thieme, "Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation," in *WSDM*, 2010, pp. 81–90.

[51] Z. Cao, M. Long, J. Wang, and P. S. Yu, "HashNet: Deep Learning to Hash by Continuation," in *ICCV*, vol. 2017-October, 2017, pp. 5609–5618.

[52] D. Shen, Q. Su, P. Chapfuwa, W. Wang, G. Wang, L. Carin, and R. Henao, "NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing," in *ACL*, 2018.

[53] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback," in *UAI*, 2009.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NeurIPS*, 2012.

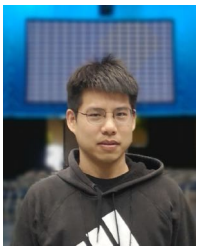
[55] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *NeurIPS*, 2014.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, "Pytorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS*, 2019, pp. 8024–8035.

[57] Y. Lin, M. Moosaei, and H. Yang, "OutfitNet: Fashion Outfit Recommendation with Attention-Based Multiple Instance Learning," in *WWW*, 2020, pp. 77–87.

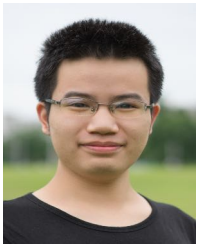
[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016, pp. 770–778.

[59] X. Song, X. Han, Y. Li, J. Chen, X.-S. Xu, and L. Nie, "GP-BPR: Personalized Compatibility Modeling for Clothing Matching," in *ACM MM*, 2019, pp. 320–328.
[60] Y.-L. Lin, S. Tran, and L. S. Davis, "Fashion Outfit Complementary Item Retrieval," in *CVPR*, 2020, pp. 3311–3319.
[61] B. Fan, Q. Kong, X. Yuan, Z. Wang, and C. Pan, "Learning Weighted Hamming Distance for Binary Descriptors," in *ICASSP*, 2013, pp. 2395–2399.

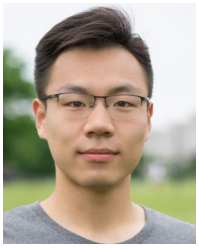


Zhi Lu received the B.S. and M.Phil. degrees from University of Electronic Science and Technology of China, Chengdu, China, in 2015 and 2018 respectively. He is currently pursuing the Ph.D. degree at the School of Information and Communication Engineering, University of Electronic Science and Technology of China. His current research interests include computer vision, machine learning and multimedia signal processing.

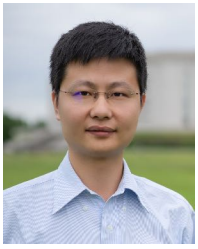
Yang Hu received the B.S. and Ph.D. degrees in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2004 and 2009 respectively. She was with the University of Maryland Institute for Advanced Computer Studies as a research associate from 2010 to 2015. She is currently an associate professor with the School of Information Science and Technology at the University of Science and Technology of China. Her current research interests include computer vision, machine learning and multimedia signal processing.



Cong Yu received B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2019. He is currently pursuing the Ph.D. degree at the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include wireless sensing, image synthesis, and adversarial learning.

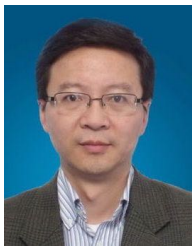


Yunchao Jiang received B.S. and M.Phil. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2018 and 2021 respectively. His current research interests include deep learning, recommendation system, and fine-grained image recognition.



Yan Chen (SM'14) received the bachelor degree from the University of Science and Technology of China in 2004, the M.Phil. degree from the Hong Kong University of Science and Technology in 2007, and the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2011. He was with Origin Wireless Inc. as a Founding Principal Technologist. From Sept. 2015 to Feb. 2020, he was a Professor with the School of Information and Communication Engineering at the University of Electronic Science and Technology of China. He is currently a Professor with the School of Cyber Science and Technology at the University of Science and Technology of China.

Dr. Chen's research interests include multimodal sensing and imaging, multimedia signal processing, and wireless multimedia. He is a co-author of "Reciprocity, Evolution, and Decision Games in Network and Data Science" (Cambridge University Press, 2021) and "Behavior and Evolutionary Dynamics in Crowd Networks: An Evolutionary Game Approach" (Springer, 2020), as well as co-author of over 200 technical papers including more than 100 IEEE journal papers. He is the Associate Editor for IEEE Transactions on Network Science and Engineering (TNSE) and IEEE Transactions on Signal and Information Processing over Networks (TSIPN). He is the Chair for APSIPA Signal and Information Processing Theory and Methods (SIPTM) Technical Committee, a Distinguished Lecturer for APSIPA, the Secretary-General for the CES Young Scientist Network Multimedia Technical Committee. He is an Organizing Co-Chair of PCM 2017, a Special Session Co-Chair of APSIPA ASC 2017, the 10K Best Paper Award Committee Member of ICME 2017, the Multimedia Communications Symposium Lead Chair of WCSP 2019, an Area Chair for ACM Multimedia 2021, a TPC Co-Chair of APSIPA ASC 2021. He was the recipient of multiple honors and awards, including an Excellent Editor for IEEE TNSE in 2021, the best paper award at the APSIPA ASC in 2020, the best student paper award at the PCM in 2017, the best student paper award at the IEEE ICASSP in 2016, the best paper award at the IEEE GLOBECOM in 2013.



Bing Zeng received the B.E. and M.Sc. degrees in Electronic Engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983 and 1986, respectively, and the Ph.D. degree in Electrical Engineering from the Tampere University of Technology, Tampere, Finland, in 1991. He worked as a Postdoctoral Fellow with the University of Toronto from September 1991 to July 1992 and as a Researcher with Concordia University from August 1992 to January 1993. Then he joined the Hong Kong University of Science and Technology (HKUST). After 20 years of service, he returned to the UESTC in the summer of 2013, through Chinas 1000-Talent-Scheme. At UESTC, he leads the Institute of Image Processing to work on image and video processing, 3-D and multiview video technology, and visual big data. During his tenure with the HKUST and UESTC, he graduated more than 30 Master's and Ph.D. students, received about 20 research grants, filed 8 international patents, and published more than 250 papers. He served as an Associate Editor for the IEEE TCSVT for 8 years and received the Best Associate Editor Award in 2011. He was General Co-Chair of the IEEE VCIP-2016, held in Chengdu, China, in November 2016. He is currently on the Editorial Board of Journal of Visual Communication and Image Representation and serves as General Co-Chair of PCM-2017. He was the recipient of a 2nd Class Natural Science Award (the first recipient) from the Ministry of Education of China in 2014 and was elected as an IEEE Fellow in 2016 for contributions to image and video coding.