# Discrete Binary Hashing Towards Efficient Fashion Recommendation

Luyao Liu[1](✉), Xingzhong Du[1], Lei Zhu[2], Fumin Shen[3], and Zi Huang[1]

[1] School of ITEE, The University of Queensland, Brisbane, Australia
{luyao.liu,x.du}@uq.edu.au, huang@itee.uq.edu.au

[2] School of Information Science and Engineering, Shandong Normal University, Jinan, China
leizhu0608@gmail.com

[3] School of Computer Science and Engineering, UESTC, Chengdu, China
fumin.shen@gmail.com

**Abstract.** How to match clothing well is always a troublesome problem in our daily life, especially when we are shopping online to select a pair of matched pieces of clothing from tens of thousands available selections. To help common customers overcome selection difficulties, recent studies in the recommender system area have started to infer the fashion matching results automatically. The conventional fashion recommendation is normally achieved by considering visual similarity of clothing items or/and item co-purchase history from existing shopping transactions. Due to the high complexity of visual features and the lack of historical item purchase records, most of the existing work is unlikely to make an efficient and accurate recommendation. To address the problem, in this paper we propose a new model called Discrete Supervised Fashion Coordinates Hashing (DSFCH). Its main objective is to learn meaningful yet compact high level features of clothing items, which are represented as binary hash codes. In detail, this learning process is supervised by a clothing matching matrix, which is initially constructed based on limited known matching pairs and subsequently on the self-augmented ones. The proposed model jointly learns the intrinsic matching patterns from the matching matrix and the binary representations from the clothing items' images, where the visual feature of each clothing item is discretized into a fixed-length binary vector. The binary representation learning significantly reduces the memory cost and accelerates the recommendation speed. The experiments compared with several state-of-the-art approaches have evidenced the superior performance of the proposed approach on efficient fashion recommendation.

## 1 Introduction

With the rapid growth of e-commerce, traditional offline clothing sales have been moving to the online websites [33]. Facing the eyeful of clothing items available online, customers usually have limited time on fashion matching and are easy to be suffering from selection difficulties. It is a very common scenario that we

J. Pei et al. (Eds.): DASFAA 2018, LNCS 10827, pp. 116–132, 2018.
https://doi.org/10.1007/978-3-319-91452-7_8

feel difficult to decide ‘which trousers would fashionably match this jumper’ or ‘what kind of skirt would go well with this shirt’. Clothing recommendation is now a trending service provided by a number of major online shopping websites. Hand-picked fashion coordinates such as model images which are advised by the fashion insides are presented to customers to assist them choosing a better matching style. However, the hand-picked solution is usually unscalable and labor-consuming. In result, recent research efforts in the recommender system area try to infer the fashion matching results automatically for the customers [25], which has strong potential to provide considerable economic value to the existing online services.

The existing work technically provides the clothing fashion matching automatically in three steps: (1) learning representations of clothing items by high-dimensional vectors with real values based on visual features and matching tuples; (2) calculating the Euclidean distances between the matching target and complementary clothing; (3) selecting the nearest complementary clothing as the matching results [12].

Since the ability of visual aware is significantly advanced by the progress in the computer vision area, recent work [8,14,24,25,33] mainly focus on how to embed the matching relations between clothing items into the embedding vectors. Although the matching accuracy has been improved by recent studies to some extent, the fashion recommendation task still faces three challenges [1,25,33].

- *Inference efficiency.* With the sustainable growth of e-commerce, a large amount of clothing is available online at high speed nowadays. Considering that the existing work need to store a high-dimension real-value vector for each item, the persistent and temporal storage costs for inference are heavy burden due to the massive data scale. In addition, the existing work employ the Euclidean distance to calculate the nearest neighbors for each query target. Given the huge amount of clothing, the inference process would be very slow. As a result, it is necessary to develop a compact feature representation for clothing items to support high efficient and scalable fashion matching with limited storage cost.
- *Label quality.* Precise labels that represent matching relationships are important for constructing an effective learning system. In other words, a matching matrix to carry the relationships (i.e., matched, un-matched, unknown) among clothing items is the essential priori knowledge for the learning process in the recommender system. As fashion matching is subjective without a clear definition, precise matching relationships are generally achieved from fashion expertise. To the best of our knowledge, the existing datasets for fashion matching, i.e., Deep Fashion [22,23] and Amazon Product Data [26,34], construct the matching labels purely according to customers’ shopping carts in single transactions. Obviously, co-purchased items cannot be guaranteed relevant or matched with each other. The matching labels generated in this way is not reliable for fashion matching supervision.

– *Fashion understanding.* Individuals may have different understanding of fashion. Fashion, from the perspective of automatic fashion matching, need to be understood by the learning over user-clothing interactions and visual features. Accordingly, how to design a better learning process to effectively capture the fashion is in high demand for personalization.

In this paper, we propose an efficient fashion recommendation method to learn meaningful yet compact representations of clothing items to capture their intrinsic visual appearances and the matching relationships. The efficiency problem in existing methods is addressed with high competitive recommendation accuracy. Specifically, we design a supervised hashing framework, called Discrete Supervised Fashion Coordinates Hashing (DSFCH), that learns discrete binary representations of clothing items from their visual content features and the matching matrix constructed based on expertise knowledge. The proposed framework guarantees that each clothing item is discretized into a fixed-length binary vector when the training stops. The discretization significantly reduces the memory cost and accelerates the inference speed. Our experiments validate that the learned binary representations effectively facilitate the fashion matching with competitive recommendation accuracy.

It is worthwhile to highlight the key contributions of our proposed method:

– we propose a supervised learning to hash framework that learns the discrete binary representations of clothing items from their visual content features and the matching matrix constructed based on expertise knowledge. An iterative optimization guaranteed with convergence is proposed to effectively solve the optimal binary representation of clothing items. The discretization can significantly reduce the memory cost and accelerate the fashion recommendation speed.
– We construct two real life fashion datasets with clothing images and professional fashion coordinates advices. These datasets are built up based on websites Netaporter[1] and Farfetch[2]. To the best of our knowledge, this is the first large-scale fashion database with professional advices for fashion recommendation.

The rest of the paper is structured as follows. Section 2 reviews the related work. Details about the proposed methodology are presented in Sect. 3. In Sects. 4 and 5, we introduce the experiments. Section 6 concludes the paper.

## 2 Related Work

Due to the limited space here, in this section, we only focus on the most related works on fashion recommendation and hashing techniques.

[1] www.net-a-porter.com/au/.
[2] www.farfetch.com/au/.

### 2.1 Fashion Recommendation

Motivated by the huge impact for e-commerce applications, fashion recommendation [10,25,33] has been receiving increasing attentions. Content-based recommender systems [15] attempt to model each user's preference toward particular types of goods. An early work [9] proposes a probabilistic topic model to learn information about coordinates from visual features by training full-body photographs from fashion magazines. The model finds reference photographs that are similar to the query image based on image content and recommends fashion items that are similar to those in the reference photograph.

Beyond exact matching between user photos and clothing images [9,10,13], recommendation systems require learn the human notions between outfit collections [30,33] and mining personal taste [4] with surrounding auxiliary information. In [25] the authors aim to model human notion of what is visually correlated by investigating a large scale dataset and affluent corresponding information. The model understands human preference more than just the visual similarity between the two. The system suggests people what not to wear and who is more fashionable.

A variety of approaches are proposed to incorporate deep learning into recommender systems [36]. A feature transformation learning [33] extends the traditional metric learning by utilizing Siamese Convolutional Neural Network (CNN) [7] architecture and projects images into a latent fashion style space to express the compatibility of outfit with the help of cross-category labels and user co-purchase data. Similarly, a recent work [12] combines fashion design and image classification by training image representations to achieve personalized fashion recommendation.

Forecasting future fashion trend is also an interesting way [1] to recommend fashion outfits before they occur. A study in [5] investigates the correlation between attributes popular in New York fashion shows versus what is seen later on the street. Another model [1] analyses fine-grained visual styles from large scale fashion data in an unsupervised manner to identify unique style signatures. The model provides a semantic description on key visual attributes to predict the future popularity of the styles.

However, existing fashion recommendation approaches still suffer from the problem of inference efficiency, label quality and fashion understanding.

### 2.2 Hashing

Hashing [2] is an advanced indexing technique that can achieve both high retrieval efficiency and memory saving. With binary embedding of hashing, the original time-consuming similarity computation can be substituted with efficient bit operations. Thus, the similarity search process could be greatly accelerated with constant or linear time complexity [43]. Moreover, binary representation could significantly shrink the memory cost of data samples, and thus accommodate large-scale similarity search with very limited memory. Due to these desirable advantages, hashing has been received great attention in literature [3,20,37,39,42].

According to the learning dependence on semantic labels, existing learning-based hashing methods can be categorized into two major families: unsupervised hashing [6,11,18,20,38,40] and supervised hashing [16,19,27,43]. Supervised hashing learns effective binary codes based on the supervised semantic labels. It usually achieves better performance than unsupervised hashing methods.

The inner product of binary codes play an important role on cross-modality retrieval [17] and supervised hashing [21]. As indicated by the existing studies [21,28,29], it has been proved that code inner product can characterize the similarity of two binary hash codes in Hamming space.

## 3 Methodology

In this section, we will detail our proposed Discrete Supervised Fashion Coordinates Hashing (DSFCH) for efficient fashion recommendation. We develop a unified hashing learning framework. A kernelized feature embedding is employed to efficiently capture the nonlinear structure of the raw feature in original space with a single vector. An inner-product fitting model is designed to preserve the correlation between various images of clothing items into binary hash codes.

### 3.1 Problem Formulation

Let $X = \{x_1, x_2, \ldots, x_n\} \in \Re^{n\times d}$ represent an image representation matrix for the collection of clothing items, $n$ is the number of data samples and $d$ is the dimension of feature representation. As mentioned above, we aim to learn a hash function $Z(x) = sgn(F(x))$, which maps $x$ from the original space into a Hamming space. Here $sgn(\cdot)$ is the signum function which returns 1 if $x \geq 0$, $-1$ if $x < 0$. We will discuss $F(x)$ in Sect. 3.2.

The projected binary codes are defined as $B = \{b_1, b_2, \ldots, b_n\} \in \{-1,1\}^{n\times r}$, where $r$ denotes the hash code length and $b_i^T, b_j^T \in \{-1,1\}^{1\times r}$ denote the $i_{th}, j_{th}$ row of $B$, respectively. Formally, the hashing projection loss can be formulated as:

$$\begin{gathered}\min_{B,F} \frac{1}{2}\sum_{i=1}^{n}(b_i - F(x_i))^2 \\ s.t.\ b_i \in \{-1,1\}^r.\end{gathered} \tag{1}$$

We introduce a fashion matching matrix $S \in \{0,1\}^{n\times n}$ to semantically guide the hash code learning process. The matrix records each pairwise similarity $S_{ij}$ as 1 if two clothing items are correlated, and 0 if their matching relations are unknown. As mentioned above, existing studies [17,21,41] have approved that the inner product of binary codes can characterize their similarity in Hamming space. In this paper, to preserve the fashion matching relations in binary codes, we try to solve the following optimization problem:

$$\begin{gathered}\min_{B,F} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(S_{ij} - \frac{1}{r}b_j^T \cdot b_i)^2 + \frac{1}{2}\nu\sum_{i=1}^{n}(b_i - F(x_i))^2 \\ s.t.\ b_i, b_j \in \{-1, 1\}^r\end{gathered} \tag{2}$$

where $\nu > 0$ is the parameter to balance regularization terms. Considering that the elements of $S$ are comprised of 0 and 1, and the binary quantization loss between each $b$ and $F(x_i)$ can be minimized by imposing the binary constraints on $B$. Therefore we rewrite the problem (2) as:

$$\begin{gathered}\min_{B,F} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij} \odot (S_{ij} - \frac{1}{r}b_j^T b_i)^2 + \frac{1}{2}\nu\sum_{i=1}^{n}(b_i - F(x_i))^2 \\ s.t.\ b_i, b_j \in \{-1, 1\}^r\end{gathered} \tag{3}$$

where $C_{ij}$ indicates the precision parameter for $S_{ij}$. The element-wise product "$\odot$" means we are only interested in the $b_j$ where $S_{ij} = 1$ corresponds to each $b_i$, which is targeted to our application. According to [35], we set $C_{ij}$ a higher value when $S_{ij} = 1$ than when $S_{ij} = 0$,

$$C_{ij} = \begin{cases} a, & if\ S_{ij} = 1 \\ b, & if\ S_{ij} = 0 \end{cases} \tag{4}$$

where $a$ and $b$ are tuning parameters satisfying $a > b > 0$. Here we follow the same settings in [35] as $a = 1, b = 0.01$.

## 3.2 Kernelized Feature Embedding

Large-scale real-world data contains a lot of noises which negatively affect the accuracy of the projections. Specifically, the learned hash codes will be affected unavoidably by variances, redundancies and noises [17]. It will result in in crucial representation problems of raw features. Thus we utilize RBF kernel embedding to achieve better performance [21]. The nonlinear form can be formulated as:

$$F(x) = \phi(x) \cdot H \tag{5}$$

where $\phi(x) \in \Re^{1\times m}$ is a $m$-dimensional row vector obtained by the kernel mapping: $\phi(x) = [exp(\|x - a_1\|^2/\epsilon, \cdots, exp(\|x - a_m\|^2/\epsilon)]$, where $\|\cdot\|$ denotes the Frobenius norm operation, $\{a_u\}_{u=1}^m$ indicates the randomly selected $m$ anchor points from the training samples and $\epsilon$ is the kernel width. The $H \in \Re^{m\times r}$ is the projection matrix which maps the original image feature into the low dimensional space. Once the kernelized feature embedding is obtained, we derive the overall objective formulation as:

$$\begin{gathered}\min_{B,F} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij} \odot (S_{ij} - \frac{1}{r}b_j^T b_i)^2 + \frac{1}{2}\nu\sum_{i=1}^{n}(b_i - F(x_i))^2 + \lambda\|H\|^2 \\ s.t.\ b_i, b_j \in \{-1, 1\}^r\end{gathered} \tag{6}$$

where $\lambda$ denotes the penalty parameter. The next step is to optimize the hash functions and find the optimal solution.

## 3.3 Optimization

Directly solving the minimization problem in Eq.(3) is NP-hard. Thus, we propose an iterative approach to convert this problem into a few sub-problems with each solving one variable when fixing all other variables. For each sub-problem, it is tractable and able to get the optimal solution.

**Optimizing *F*.** If B is fixed in Eq. (6), the projection matrix $H$ is independent to other regularization terms. Therefore we can easily compute the $H$ by solving:

$$\min_{H} \|B - \phi(X)H\|^2 + \lambda\|H\|^2 \tag{7}$$

Eq. (5) can be solved by linear regression. The optimal $H$ can be derived as:

$$H = (\phi(X)^T\phi(X) + \lambda I)^{-1}\phi(X)^T B. \tag{8}$$

**Optimizing *B*.** It is still challenging to optimize $B$ due to the discrete constraints in Eq. (6) which is NP-hard problem. So we try to find a closed-form solution for each single $b_i$ by fixing all other bits $\{b_j\}_{j\neq i}^{n}$ during optimization. We can rewrite the each iteration step of Eq. (6) as:

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} \odot (S_{ij} - \frac{1}{r}b_j^T b_i)^2 + \nu\sum_{i=1}^{n}(b_i - F(x_i))^2 + \lambda\|H\|^2 \tag{9}$$

where $L$ denotes the total loss of each loop, and it will achieve convergence after $K_{th}$ iteration. It should be noticed that when we apply another embedded iteration to solve each single $b_i$ of $B$, we relax the discrete constraint. When $i$ is ranged from 1 to $n$, we calculate the partial derivatives of loss term $l_i$ with respect to the output $b_i$. The partial derivation process can be written as:

$$\begin{aligned}\frac{\partial l_i}{\partial b_i} &= 2\sum_{j=1}^{n} C_{ij} \odot (S_{ij} - \frac{1}{r}b_j^T b_i)\frac{\partial(-\frac{1}{r}b_j^T b_i)}{\partial b_i} + 2\nu(b_i - F(x_i))\frac{\partial b_i}{\partial b_i} \\ &= (\sum_{j=1}^{n} C_{ij} \odot \frac{1}{r^2}S_{ij}b_j b_j^T + \nu I)b_i - (\sum_{j=1}^{n} C_{ij} \odot \frac{1}{r}S_{ij}b_j + \nu F(x_i)).\end{aligned} \tag{10}$$

Due to Eq. (4), $C_{ij} \odot S_{ij} = S_{ij}$. Let $\nabla l_i = 0$, the optimized solution can be calculated as:

$$b_i = sgn((\sum_{j=1}^{n} \frac{1}{r^2}S_{ij}b_j b_j^T + \nu I)^{-1}(\sum_{j=1}^{n} \frac{1}{r}S_{ij}b_j + \nu F(x_i))). \tag{11}$$

We can observe that computing single bit binary codes for each data point relies on the rest of pre-learned $(n-1)$ binary codes. It is also noted that $b_j^k$ should be selected from the previous iterative round of pre-learned $B^{k-1}$ corresponding to each $b_i$. Thus, we need to learn and update $b_i$ for $n$ times in each iteration to obtain the final optimized $B$. The iteration complexity here is $O(knr + knr^3)$ where $k, r \ll n$. More importantly, we still keep the discrete constrains for $B$ outside the embedded iteration.

**Algorithm 1.** *Discrete Supervised Fashion Coordinates Hashing (DSFCH)*

```
Input: Training data $X = \{x_1, x_2, \ldots, x_n\} \in \Re^{n \times d}$ , matching matrix $S$, precision
    parameter $C$, code length $r$, number of anchor points $m$, maximum iteration num-
    ber $K$, parameters $\lambda$ and $\nu$.
Output: Binary codes $B \in \{-1, 1\}^{n \times r}$, hashing projection matrix $H$.

    Randomly select $m$ samples $\{a_u\}_{u=1}^{m}$ from the training data and map the training
    data via the RBF kernel function $\phi(x)$
    Initialize $B_0 \in \{-1, 1\}^{n \times r}$.
    repeat
      Optimizing $F$:
         Calculate $H$ using Eq.(8):
            $H = (\phi(X)^T \phi(X) + \lambda I)^{-1} \phi(X)^T B$
      Optimizing $B$:
         Calculate each $b_i$ of $B$ using Eq.(11):
            $b_i = sgn((\sum_{j=1}^{n} \frac{1}{r^2} S_{ij} b_j b_j^T + \nu I)^{-1} (\sum_{j=1}^{n} \frac{1}{r} S_{ij} b_j + \nu F(x_i)))$
      Calculate the loss of each iteration using Eq.(9):
            $L = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \odot (S_{ij} - \frac{1}{r} b_j^T b_i)^2 + \sum_{i=1}^{n} \nu (b_i - F(x_i))^2 + \lambda \|H\|^2$
    until Convergence
```

**Initializing *B*.** Obviously we should initialize $B_0$ to start $F$ sub-problem before conducting the $K$ iterations. Inspired by $SH$ [40] and $KSH$ [19], we tried to initialize the binary codes by thresholding spectral graph decomposition. However, the performance was unsatisfactory and considering the time consumption, we choose to use random binary codes $B_0 \in \{-1, 1\}^{n \times r}$ which is sufficient to show the effectiveness of our method.

**Precision Parameter $C_{ij}$.** In the above section, we have presented the discrete learning algorithm for each bit of hash codes. We haven't discussed the influence of the $C_{ij}$ which is a precision parameter for rating the correlation matrix $S_{ij}$. Without $C_{ij}$, our model will compute all of the 0 labels (unknown cases) same as the ones with 1 labels, which dramatically reduces the learning effectiveness. With considering $C_{ij}$, we trust the labelled cases more than the unknown cases when $C_{ij}$ is high (e.g. here we define it as $a = 1$). In addition, the parameter helps the model balance the weight of loss between matching and unknown cases (by defining $b = 0.01$ when $S_{ij} = 0$). It means the model considers the loss of 100 unknown cases as 1 trust case.

**Online Recommendation.** Once we get the optimized projection matrix $H$, for given a query $q$, the predicted binary code can be simply computed by a signum function on a linear embedding. The formula is $B_q = Z(x) = sgn(\phi(q) \cdot H)$. We use inner product to calculate ranking score which is formulated as:

$$score = \frac{1}{r} B \cdot B_q^T. \tag{12}$$

The ranking score will be sorted in descend order and the larger value get the better recommending priority.

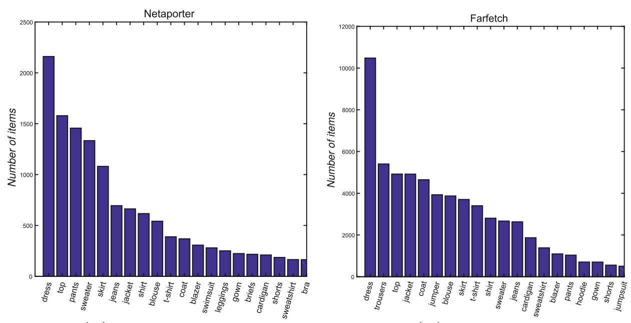
## 4 Experimental Dataset

### 4.1 Fashion Dataset

As one of the key contributions of this work, two real-life fashion datasets are constructed by crawling clothing data from two well-known online shopping websites Netaporter and Farfetch. These two websites demonstrate millions of clothing images, where each item is associated with detailed descriptions such as category, brand, price, similar items, matching advice and groups of pictures taken from different views. At the current stage, more than 80,000 clothing items have been stored in our fashion database with more than 30,000 professional clothing matching suggestions, which is detailed in Table 1. In this paper, we are only focused on the clothing visual appearance and matching advice (Fig. 1).

**Table 1.** Statistics of the fashion dataset

| # of | Netaporter | Farfetch |
|---|---|---|
| Items | 13,190 | 68,563 |
| Categories | 57 | 174 |
| Matching pairs | 15,476 | 18,244 |



(a) Netaporter (b) Farfetch

**Fig. 1.** Number of clothing in top 20 categories for both datasets.

## 4.2 Feature Extraction

The deep convolutional Neural Network (CNN) [31] is employed in this work to capture the visual appearance of clothing items. We extract the 4096 dimensional visual features from the second fully-connected layer (i.e. FC7). These features are used as the input of our learning model and also for the matching matrix self-augmentation (Fig. 2).

## 4.3 Description of Matching Matrix

The matching matrix indicates the identified matching items based on both the professional advices and the self-augmented relationships. The original clothing items are divided into different categories, such as T-shirt, pants, etc. Possible matching relationships are not limited to the items from different categories. In reality, matched pairs may from the same category, where one example is shown in Fig. 3 This fact clearly points out the difference between our work of fashion recommendation and the conventional visual similarity based clothing retrieval, where the later one is limited to finding the similar items from the same category. The initial matching matrix is constructed based on the professional advice that is provided by the websites. All these advice is hand-picked (i.e., manually generated) and obviously quite limited. Due to this, the matching matrix is very sparse (Fig. 3).

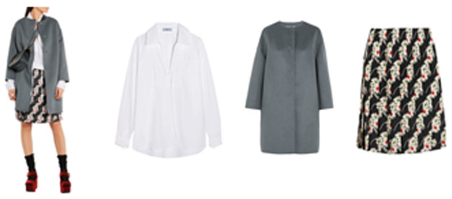


**Fig. 2.** Example of professional advices for matching. These three clothing match each other.

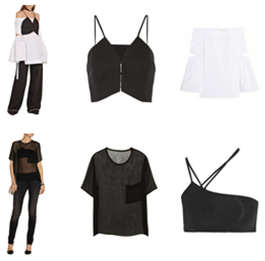


**Fig. 3.** Example of matched pairs from the same category "top".

## 4.4 Data Preprocessing

The real world data on the website contains a lot of noises such as typo, wrong labels, ambiguity of name, strange id numbers and off-line items which bring negative impact on recommendation model training. Therefore we made a lot of efforts on correcting and eliminating those noises to get pure valid pair labels. In particular, for off-line items, if the items are still stored in the image database and do have pair matchings with other items, we still save them as valid records.

In addition, we only focus on the items being labelled. Before training, we select those positive records which at least are labelled with another item. The valid dataset size of Netaporter and Farfetch are 10,793 and 18,753 respectively. For Netaporter dataset, we select 500 records as query set, the remaining 10,293 records are determined as the training and retrieval set. For Farfetch dataset, we select 1000 records as query set, the remaining 17,753 records as the training and retrieval set.

After we separate the whole data into training and testing parts, some of the records which belong to the training part will lose their pair labels due to the sparsity of the matching matrix. For example, if one record only has one matching pair which is selected into the test part by accident, this record becomes invalid. In this paper, we propose an effective self-augmentation process to alleviate the problem.

### 4.5 Matching Matrix Self-augmentation

Due to the sparsity of the initial matching matrix, we conduct a self-augmentation process to enrich the density of the matching relationships.

Firstly, we directly calculate the Euclidean distance of CNN features between each clothing in order to find the K-Nearest Neighbour similar items. Then we find all of the matched items for each clothing by the matching matrix. Finally we assign each matched item with the most $n$ similar neighbours of the clothing as matching pair. In other words, if two items $x_i$ $x_j$ are labelled with 1 (i.e. $S_{ij} = 1$), we find K-NN samples $x_{ik}$ and $x_{jk}$ where $x_{ik}, x_{jk} \in X$ and $x_{ik} \neq x_i$, $x_{jk} \neq x_j$. Then assign $S_{ik,j} = S_{jk,i} = 1$. As a result, the scale of density is multiplied by $n$.

Intuitively, it can be understood that if a white long-sleeve shirt is labelled with a jeans and there is another white long-sleeve shirt which is super close to the previous shirt on visual content, we can infer that the second shirt is also well matching the jeans. But we do not label them crossly and it is expected that our model is able to learn those intrinsic relationships.

## 5 Experiment

### 5.1 Evaluation Metrics

We compute the AUC (Area Under the ROC curve) to evaluate the recommendation performance of each model. The AUC measures the quality of a ranking based on pairwise comparisons. Higher value of AUC means higher performance on recommendation. The AUC is formulated as:

$$AUC = \frac{1}{|\mathcal{Q}|} \sum_{(q,i,j) \in \mathcal{Q}} \delta(x_{q,i} > x_{q,j}),$$

where $\delta(\cdot)$ is an indicator function and $\mathcal{Q}$ is the fraction of the data withheld for testing. In other words, we are counting the fraction of times the model correctly ranks $i$ higher than $j$.

## 5.2 Compared Approaches

We compare our DSFCH with four state-of-art hashing methods, including Supervised Hashing with Kernels (KSH) [21], Inter-media hashing(IMH) [32], Canonical Correlation Analysis (CCA) [6] and Supervised Discrete Hashing (SDH) [27].

## 5.3 Implementation Settings

In experiments, hash code length on all datasets is varied in the range of [16, 32, 64, 128] to observe the performance. (1) For Netaporter, number of anchor points $m = 500$, maximum iteration number $K = 50$, parameters $\lambda = 10^{-2}$ and $\nu = 10^{-2}$, Self-augmentation $n = 3$, query size=500 with the same initial seed. (2) For Farfetch, number of anchor points $m = 1000$, maximum iteration number $K = 50$, parameters $\lambda = 10^{-2}$ and $\nu = 10^{-2}$. Self-augmentation $n = 3$, query size=1000 with the same initial seed.

## 5.4 Experiment Results

We report AUC results of all compared methods in Table 2. A query example of coordinates recommendation is shown in Fig. 4. For the Netaporter dataset, we can easily find that our method DSFCH outperforms the competitors on all

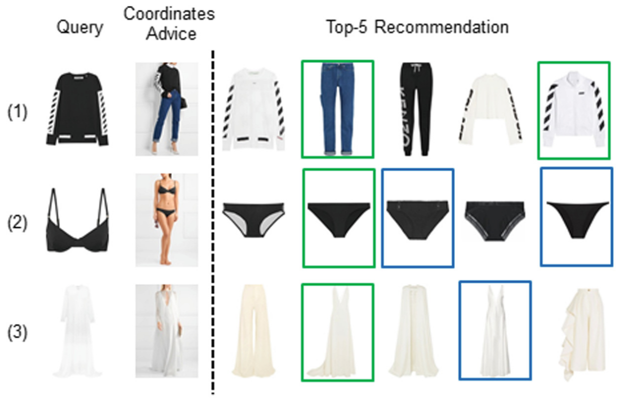


**Fig. 4.** Example of fashion recommendation by DSFCH with top-5 returned candidates. The returned clothing items that are the same as the recommended ones by professionals are highlighted with green frame. The items bounded in blue boxes are the same as the matched ones identified by the proposed self-augmentation process. (1) Given a query of "top", the recommended results are jeans, pants, top and jackets. The 5th is a matching advice of the query (https://www.net-a-porter.com/au/en/product/735424). The 1st and 4th are visual similar to the 5th. (2) Given a query of "bra", the recommended results are briefs. (3) Given a query of "cape", the recommended results are pants, gown and culottes.

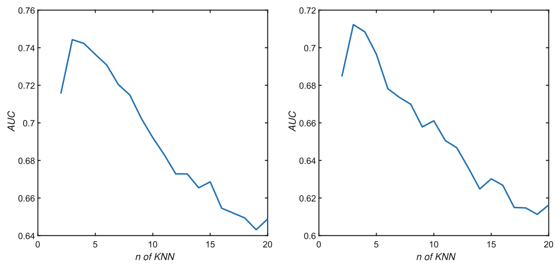
cases. The largest improvement appears on 128 bits about 40.5% than the second best approach. For the Farfetch dataset, our method DSFCH is outperform at 64 and 128 code bits. In particular, DSFCH can achieve 0.5228 AUC on 16 bits for Farfetch if the Self-augmentation parameter setting is $n = 2$, which shows that different lengths of binary code accommodate different $n$. In addition, there is also an interesting finding from the experimental results that the AUC increases significantly along with the increase of the hash code length on both two datasets.

**Table 2.** AUC of all approaches on two datasets. The best result in each column is marked with bold.

| Methods | Netaporter | | | | Farfetch | | | |
|---|---|---|---|---|---|---|---|---|
| | 16 | 32 | 64 | 128 | 16 | 32 | 64 | 128 |
| KSH | 0.5124 | 0.5016 | 0.4874 | 0.5252 | **0.5495** | **0.5621** | 0.5731 | 0.5720 |
| IMH | 0.4650 | 0.4650 | 0.4650 | 0.4650 | 0.5070 | 0.5070 | 0.5070 | 0.5070 |
| CCA | 0.5067 | 0.5070 | 0.4981 | 0.4953 | 0.5073 | 0.5046 | 0.5037 | 0.5114 |
| SDH | 0.4624 | 0.4796 | 0.5114 | 0.5174 | 0.4369 | 0.4931 | 0.4322 | 0.5436 |
| **DSFCH** | **0.5468** | **0.5911** | **0.6622** | **0.7379** | 0.4815 | 0.5209 | **0.6262** | **0.7105** |

## 5.5 Self-augmentation Study

We are trying to enrich the density of the matching matrix by KNN search, the testing result on 128 bits is shown below in Figure 3. We tested both Netaporter and Farfetch dataset. We found that, when the nearest neighbours number $n = 3$, the performance is peaking at highest value. Along with the number $n$ increasing, the performance is dropping sharply (Fig. 5).



(a) Netaporter (b) Farfetch

**Fig. 5.** Comparison of AUC curves for different Self-augmentation parameter $n$ on 128 bits with two datasets.

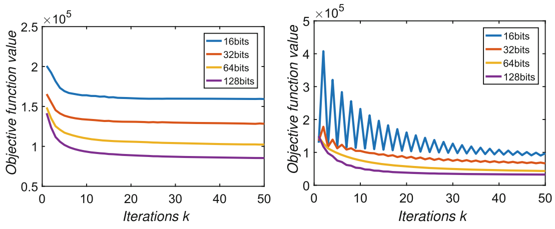
## 5.6 Discrete or Not?

In practical, we compare the performance in two situations: (1) we keep the binary constraint of $B$ in Eq. (11) during training; (2)we relax the discrete constraints to get a continuous B and threshold it at last. In most cases, we notice that discrete method is better than relaxed method. Keeping the binary constraints is getting better and better performance along with the code length increasing. Which can be understood that short code length suffers more penalty from quantization loss. All these two experiment is tested on Netaporter dataset, maximum iteration number $K = 50$, self-augmented neighbours $n = 3$, with the same query set and initial seed (Table 3).

**Table 3.** Comparative performance between discrete or relaxed methods on Netaporter dataset.

| | Constraint | 16 bits | 32 bits | 64 bits | 128 bits |
|---|---|---|---|---|---|
| AUC | Discrete | **0.5702** | 0.5898 | **0.6684** | **0.7500** |
| | Relaxed | 0.5663 | **0.6004** | 0.6655 | 0.7134 |

Figure 6 shows the convergence procedures in discrete and relaxed respectively. It can be seen from the figures that discrete method is much faster to get convergent than the other one. Also the convergence curve of discrete is more stable than the relaxed one.



(a) Discrete (b) Relaxed

**Fig. 6.** Objective function value variations with the number of iterations on discrete and relaxed methods.

## 5.7 Parameter Sensitivity Experiment

In practical, we notice that the parameter $\nu$ has a significant impact on learning performance, so a parameter study is applied shown below in Fig. 7. Testing range is $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}\}$ on 128 bits, argumentation $n = 3$.

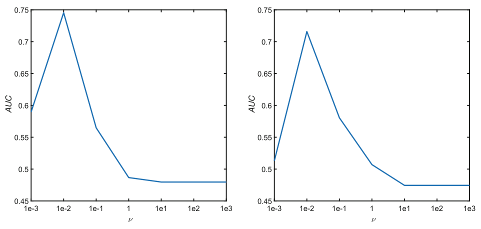


**Fig. 7.** Comparison of AUC for different $\nu$ on 128 bits with two datasets.

## 6 Conclusion

In this paper, we propose an effective model, dubbed as Discrete Supervised Fashion Coordinates Hashing (DSFCH), to learn meaningful yet compact visual features of clothing items, and thus support large-scale fashion recommendation. The learning process is supervised by a clothing matching matrix, which is initially constructed based on the limited pre-known matching pairs with self-augmentation. The proposed model jointly learns the intrinsic matching patterns from the matching matrix and the discrete binary representations from the images of clothing items. The binary representation significantly reduces the memory cost and accelerates the fashion recommendation. Extensive experiments have been conducted to provide comprehensive performance studies on different parameter settings. The comparisons with the-state-of-the-arts methods have evidenced the superior performance of the proposed approach for fashion recommendation.

## References

1. Al-Halah, Z., Stiefelhagen, R., Grauman, K.: Fashion forward: forecasting visual style in fashion. In: ICCV, October 2017
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1), 117–122 (2008)
3. Andoni, A., Razenshteyn, I.: Optimal data-dependent hashing for approximate near neighbors. In: STOC, STOC 2015, pp. 793–801. ACM (2015)
4. Bracher, C., Heinz, S., Vollgraf, R.: Fashion DNA: merging content and sales data for recommendation and article mapping. CoRR abs/1609.02489 (2016)
5. Chen, K., Chen, K., Cong, P., Hsu, W.H., Luo, J.: Who are the devils wearing Prada in New York city? In: Proceedings of the 23rd ACM international conference on Multimedia, pp. 177–180. ACM (2015)
6. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. TPAMI **35**(12), 2916–2929 (2013)
7. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. CVPR **2**, 1735–1742 (2006)

8. He, R., Packer, C., McAuley, J.: Learning compatibility across categories for heterogeneous item recommendation. In: ICDM, pp. 937–942. IEEE (2016)
9. Iwata, T., Wanatabe, S., Sawada, H.: Fashion coordinates recommender system using photographs from fashion magazines. In: IJCAI, vol. 22, p. 2262 (2011)
10. Jagadeesh, V., Piramuthu, R., Bhardwaj, A., Di, W., Sundaresan, N.: Large scale visual recommendations from street fashion images. In: SIGKDD, KDD 2014, pp. 1925–1934. ACM (2014)
11. Jiang, Q.Y., Li, W.J.: Scalable graph hashing with feature transformation. In: IJCAI, IJCAI 2015, pp. 2248–2254. AAAI Press (2015)
12. Kang, W.C., Fang, C., Wang, Z., McAuley, J.: Visually-aware fashion recommendation and design with generative image models. arXiv preprint arXiv:1711.02231 (2017)
13. Kiapour, M.H., Han, X., Lazebnik, S., Berg, A.C., Berg, T.L.: Where to buy it: matching street clothing photos in online shops. In: ICCV, pp. 3343–3351 (2015)
14. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
15. Lew, M.S., Sebe, N., Djeraba, C., Jain, R.: Content-based multimedia information retrieval: state of the art and challenges. TOMM **2**(1), 1–19 (2006)
16. Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: CVPR, pp. 2475–2483 (2015)
17. Liu, L., Lin, Z., Shao, L., Shen, F., Ding, G., Han, J.: Sequential discrete hashing for scalable cross-modality similarity retrieval. TIP **26**(1), 107–118 (2017)
18. Liu, L., Zhu, L., Li, Z.: Learning robust graph hashing for efficient similarity search. In: Huang, Z., Xiao, X., Cao, X. (eds.) ADC 2017. LNCS, vol. 10538, pp. 110–122. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68155-9_9
19. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: CVPR, pp. 2074–2081 (2012)
20. Liu, W., Mu, C., Kumar, S., Chang, S.F.: Discrete graph hashing. In: NIPS, NIPS 2014, pp. 3419–3427. MIT Press (2014)
21. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: CVPR, pp. 2074–2081. IEEE (2012)
22. Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X.: DeepFashion: powering robust clothes recognition and retrieval with rich annotations. In: CVPR (2016)
23. Liu, Z., Yan, S., Luo, P., Wang, X., Tang, X.: Fashion landmark detection in the wild. In: ECCV (2016)
24. McAuley, J., Pandey, R., Leskovec, J.: Inferring networks of substitutable and complementary products. In: SIGKDD, pp. 785–794. ACM (2015)
25. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: SIGIR, SIGIR 2015, pp. 43–52. ACM (2015)
26. McAuley, J., Yang, A.: Addressing complex and subjective product-related queries with customer reviews. In: Proceedings of the 25th International Conference on World Wide Web, pp. 625–635. International World Wide Web Conferences Steering Committee (2016)
27. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing. In: CVPR, pp. 37–45 (2015)
28. Shen, F., Liu, W., Zhang, S., Yang, Y., Tao Shen, H.: Learning binary codes for maximum inner product search. In: ICCV, pp. 4148–4156 (2015)
29. Shrivastava, A., Li, P.: Asymmetric LSH (ALSH) for sublinear time maximum inner product search (mips). In: NIPS, pp. 2321–2329 (2014)
30. Simo-Serra, E., Fidler, S., Moreno-Noguer, F., Urtasun, R.: Neuroaesthetics in fashion: modeling the perception of fashionability. In: CVPR, pp. 869–877 (2015)

31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
32. Song, J., Yang, Y., Yang, Y., Huang, Z., Shen, H.T.: Inter-media hashing for large-scale retrieval from heterogeneous data sources. In: SIGMOD, SIGMOD 2013, pp. 785–796. ACM (2013)
33. Veit, A., Kovacs, B., Bell, S., McAuley, J., Bala, K., Belongie, S.: Learning visual clothing style with heterogeneous dyadic co-occurrences. In: ICCV, pp. 4642–4650 (2015)
34. Wan, M., McAuley, J.: Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In: ICDM, pp. 489–498. IEEE (2016)
35. Wang, C., Blei, D.M.: Collaborative topic modeling for recommending scientific articles. In: SIGKDD, pp. 448–456. ACM (2011)
36. Wang, H., Wang, N., Yeung, D.Y.: Collaborative deep learning for recommender systems. In: SIGKDD, pp. 1235–1244. ACM (2015)
37. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: CVPR, pp. 3424–3431 (2010)
38. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for large-scale search. TPAMI **34**(12), 2393–2406 (2012)
39. Wang, J., Xu, X.S., Guo, S., Cui, L., Wang, X.L.: Linear unsupervised hashing for ANN search in Euclidean space. Neurocomputing **171**, 283–292 (2016)
40. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) NIPS, pp. 1753–1760. Curran Associates, Inc. (2009)
41. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. AAAI **1**, 2156–2162 (2014)
42. Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., Yu, N.: Complementary hashing for approximate nearest neighbor search. In: ICCV, pp. 1631–1638 (2011)
43. Zhang, P., Zhang, W., Li, W.J., Guo, M.: Supervised hashing with latent factor models. In: SIGIR, SIGIR 2014, pp. 173–182. ACM (2014)