Data Science and Engineering
https://doi.org/10.1007/s41019-018-0079-z

Learning Discrete Hashing Towards Efficient Fashion Recommendation

Luyao Liu[1] · **Xingzhong Du**[2] · **Lei Zhu**[3] · **Fumin Shen**[4] · **Zi Huang**[2]

Received: 3 August 2018 / Accepted: 20 October 2018

Abstract
In our daily life, how to match clothing well is always a troublesome problem 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 issues, recent studies in the recommender system area have started to infer the fashion matching results automatically. The traditional 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. 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.

Keywords Hashing · Discrete Hashing · Fashion · Fashion recommendation · Fashion coordinates

1 Introduction

With the rapid growth of e-commerce, conventional offline clothing sales have been moving to the online websites [51]. 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 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 labour-consuming. In result, recent research efforts in the recommender system area try to infer the fashion matching results automatically for the customers [40], which has strong potential to provide considerable economic value to the existing online services.

✉ Luyao Liu
luyao.liu@uq.edu.au

Xingzhong Du
x.du@uq.edu.au

Lei Zhu
leizhu0608@gmail.com

Fumin Shen
fumin.shen@gmail.com

Zi Huang
huang@itee.uq.edu.au

[1] The University of Queensland, Room 636, Building 78, St Lucia Campus, Brisbane, Australia

[2] The University of Queensland, Brisbane, Australia

[3] School of Information Science and Engineering, Jinan, Shandong, China

[4] School of Computer Science and Engineering, UESTC, Chengdu, China

Published online: 03 November 2018

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 [24].

Since the ability of visual aware is significantly advanced by the progress in the computer vision area, recent work [18, 26, 39, 40, 51] 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, 40, 51].

- *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 employs the Euclidean distance to calculate the nearest neighbours 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 [37, 38] and Amazon Product Data [41, 52], 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 three real-life fashion datasets with clothing images and professional fashion coordinates advices. These datasets are built up based on websites Net-a-porter,[1] Farfetch[2] and Mytheresa.[3] To the best of our knowledge, this is the first large-scale fashion database with professional advices for fashion recommendation.

This paper is an extension and improvement of our previous work presented in [31]. Firstly, online websites regularly release the newest fashion coordinates as the season changes. To enrich the scale of our database, we keep collecting clothing and matching pairs from the fashion websites. Until the date of submission, we have enlarged the scale of the two existing datasets around 50%. Meanwhile, a new dataset called Mytheresa has been constructed which helps promote our experiments and the overall comparison results with state-of-the-art methods in this paper are based on the updated database. Secondly, to evaluate the time efficiency of our proposed method, we calculate the average

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

[2] www.farfetch.com/au/.

[3] www.mytheresa.com/en-au.

time consumptions at the training and test phases to complement the experiments. In addition, a more comprehensive analysis is applied in the later Sect. 4.3.

The rest of the paper is structured as follows. Section 2 reviews the related work in the field of fashion recommendation and hashing techniques. Details about the proposed methodology are presented in Sect. 3. Section 4 elaborates the detailed descriptions of experiment settings, comparison results with baselines and comprehensive analysis of proposed method. Lastly Sect. 5 concludes the paper and provide a future work discussion.

2 Related Work

In this section, we briefly introduce the most related works on fashion recommendation and hashing techniques.

2.1 Fashion Recommendation

Motivated by the huge impact for e-commerce applications, fashion recommendation [21, 40, 51] has been receiving increasing attentions. Content-based recommender systems [29] attempt to model each user's preference towards particular types of goods. An early work [20] 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 [20, 21, 25], recommendation systems require learning the human notions between outfit collections [47, 51] and mining personal taste [6] with surrounding auxiliary information. In [40], 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 [54]. A feature transformation learning [51] extends the traditional metric learning by utilizing Siamese Convolutional Neural Network (CNN) [14] 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 [24] 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 [7] 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. A fresh work [19] develops sub-modular objective functions to capture the major ingredients of visual compatibility, versatility and user preference.

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 [62]. 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, 34, 55, 57, 60].

Basically, the binary coding or hashing techniques can be roughly categorized into two major families: data-independent and data-dependent. Locality Sensitive Hashing (LSH) [11] is one of representative data-independent methods, which simply exploits random mapping to project the data samples into binary Hamming space. In addition to traditional Euclidean distance, the LSH family has been continuously developed to accommodate diverse distance and similarity measures such as p-norm distance [9], Mahalanobis distance [28] and kernel similarity [27, 42]. However, to achieve satisfactory retrieval performance in practice, LSH usually requires long hash bits and multiple tables so that the storage cost is huge which restricts its practicability.

Recent years, learning-based hashing methods have witness a dramatic growth with available training data. As data-dependent methods, learning compact binary codes can effectively and efficiently index and organize massive data by generating short hash codes. According to the learning dependence on semantic labels, existing learning-based hashing methods can be divided into two groups: unsupervised hashing [12, 23, 33, 34, 36, 56, 58] and supervised hashing [13, 30, 35, 45, 62]. Specifically, a representative of unsupervised hashing methods is Spectral Hashing (SH) [58], which solves a continuously relaxed mathematical function similar to Laplacian Eigenmaps [5] to generate hash codes without any supervision. A later improvement, Anchor Graph Hashing (AGH) [36], learns compact hash

codes by discovering the neighbourhood structural inherent in the data, making the learning process tractable and efficient for large-scale datasets.

By contrast, supervised hashing learns effective binary codes based on the supervised semantic labels. It usually achieves better performance than unsupervised hashing methods since supervised hashing methods can generate more discriminative binary codes effectively preserving the high-level label information instead of the low-level data structures [32].

Generally, hash codes consist of 0 and 1 or -1 and 1. Regardless of the types of the hashing methods, learning this mixed binary-integer codes through objective functions with discrete constraints always suffers an optimization problem [45] which is NP-hard. To overcome this issue and make it tractable, some of the aforementioned methods [33, 36, 56, 58] first simply drop the discrete constraints, converting the NP-hard problem into a relaxed continues embedding problem and then quantize the optimized solution to obtain an approximate binary codes. However, the relaxation scheme makes the hash projection less effective and brings accumulated quantization errors between projected value and hash codes, especially for long code length, which leads to suboptimal results. Iterative Quantization (ITQ) [12] applies an orthogonal rotation on mapped training data to decrease the quantization error based on the pre-computed mapping steps such as PCA [56] and CCA [17]. But the separated learning process still makes the hash codes suboptimal. Kernel-Based Supervised Hashing (KSH) [35] simulates discrete constraints by replacing the sign function with the sigmoid function to catch no-linear manifold hidden structure in the data and it shows effectiveness in generating compact neighbourhood-preserving hash codes. Facing large-scale data, however, the discrete approximation with the sigmoid operation spends an expensive computational price for hash function optimization and the optimized result is still suboptimal. To find more effective hashing scheme, Supervised Discrete Hashing (SDH) [45] proposes an algorithm that directly learns the binary hash codes without relaxing the discrete constraints. Although SDH outperforms previous hashing methods on accuracy, but the optimization process with discrete cyclic coordinates descent (DCC) is time-consuming. To achieve better performance and speed, later work Fast Supervised Discrete Hashing (FSDH) [13] proposes a closed-form solution for hash learning that only requires a single step instead of iteration, which makes an impressive effect on learning speed.

Almost all the aforementioned hashing methods are proposed to achieve Approximate Nearest Neighbour search [44]. However, hashing techniques are not just limited to single-modality retrieval. The inner product of binary codes play an important role on cross-modality retrieval application [32] and supervised hashing [35]. As indicated by the existing studies [35, 44, 46], 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 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 $X = \{x_1, x_2, \ldots, x_n\} \in \Re^{n \times d}$ indicates 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 $Z(x) = \mathrm{sgn}(F(x))$, which maps x from the original space into a Hamming space. Here, $\mathrm{sgn}(\cdot)$ 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 ith, jth row of B, respectively. Formally, the hashing projection loss can be formulated as:

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

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 [32, 35, 59] 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:

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

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:

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

where C_{ij} indicates the precision parameter for S_{ij}. The element-wise product "$\odot$" means we are interesting in the b_j where $S_{ij} = 1$ corresponds to each b_i more than the one where $S_{ij} = 0$, which is targeted to our application. More details will be discussed in Sect. 3.3.4. According to [53], we set C_{ij} a higher value when $S_{ij} = 1$ than when $S_{ij} = 0$,

$$C_{ij} = \begin{cases} pr_a, & \text{if} \quad S_{ij} = 1 \\ pr_b, & \text{if} \quad S_{ij} = 0 \end{cases} \tag{4}$$

where pr_a and pr_b are tuning parameters satisfying $pr_a > pr_b > 0$. Here, we follow the same settings in [53] as $pr_a = 1$, $pr_b = 0.01$.

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 [32]. It will result in crucial representation problems of raw features. Thus, we utilize RBF kernel embedding to achieve better performance [35]. The nonlinear form can be formulated as:

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

where $\phi(x) \in \mathfrak{R}^{1\times m}$ is a m-dimensional row vector obtained by the kernel mapping: $\phi(x) = [\exp(\|x - a_1\|^2/\epsilon), \ldots, \exp(\|x - a_m\|^2/\epsilon)]$, where $\|\cdot\|$ denotes the Frobenius norm operation, $\{a_u\}_{u=1}^{m}$ indicates the randomly selected m anchor points from the training samples and ϵ is the kernel width. The $H \in \mathfrak{R}^{m\times r}$ 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:

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

where λ 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.

3.3.1 Optimizing *F*

If B is fixed in Eq. (3), the projection matrix H is independent to other regularization terms. Therefore, we can easily compute the H by solving:

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

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

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

3.3.2 Optimizing *B*

It is still challenging to optimize B due to the discrete constraints in Eq. (3) 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. (3) as:

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

where L denotes the total loss of each loop, and it will achieve convergence after Kth 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:

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

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

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

We can observe that computing each single bit for 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.

3.3.4 Precision Parameter C_{ij}

In the above section, we have presented the discrete learning algorithm for each bit of hash codes. We have not 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 $pr_a = 1$). In addition, the parameter helps the model balance the weight of loss between matching and unknown cases (by defining $pr_b = 0.01$ when $S_{ij} = 0$). It means the model considers the loss of 100 unknown cases as 1 trust case.

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 number K, parameters λ and ν.

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

3.3.3 Initializing B

Obviously, we should initialize B_0 to start F sub-problem before conducting the K iterations. Inspired by *SH* [58] and *KSH* [35], we tried to initialize the binary codes by thresholding spectral graph decomposition. However, it makes the final result fluctuating on large variations, which leads to unsatisfactory performance on evaluation process. Due to this problem, we utilize random binary codes $B_0 \in \{-1, 1\}^{n \times r}$ and found that random B_0 with uniform distribution only makes the fluctuation within a narrow range around 2% which is an acceptable impact on our experiment. Considering above issues, we choose to use the later strategy which is sufficient to show the effectiveness of our method.

3.3.5 Online Recommendation

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

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

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

4 Experiment

In this section, we evaluate the performance of our proposed DFSCH method by conducting extensive experiments. The configuration of our experiments is illustrated first which includes the datasets, feature extraction, matching matrix, data preprocessing, self-augmentation, evaluation metrics, compared approaches and implementation settings. Then, we analyse the comparison results about our method and several state-of-the-art approaches. In addition, we provide a further testing on discrete strategy. Finally, a comprehensive parameter sensitivity investigation will be given.

4.1 Experimental Settings

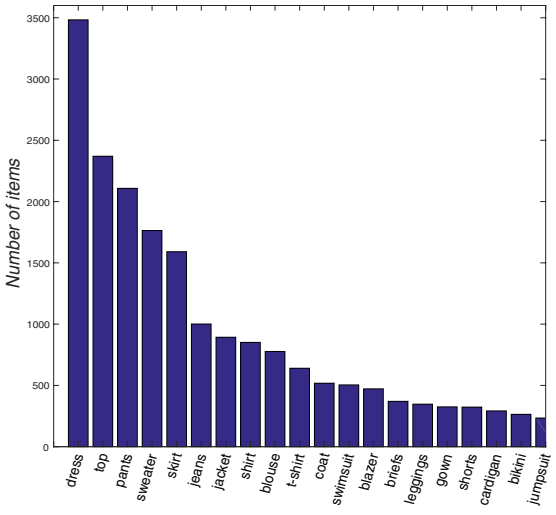
4.1.1 Dataset

As one of the key contributions of this work, three real-life fashion datasets are constructed by crawling meta data from well-known online shopping websites. Those 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 139,000 clothing items have been stored in our fashion database with more than 66,000 professional clothing matching suggestions, which is detailed in Table 1. In this paper, we take advantage of each single clothing picture with clean background and the matching advices among clothings. Seasonally, these popular websites regularly release a plenty of new fashion coordinates to the public. To the best of our knowledge, this is the first large-scale fashion database with professional advices for fashion recommendation. Those professional advices provided by fashion designers will be translated to the matching matrix which supervises our approach intuitively. More details about the matching matrix will be discussed in Sect. 4.3.

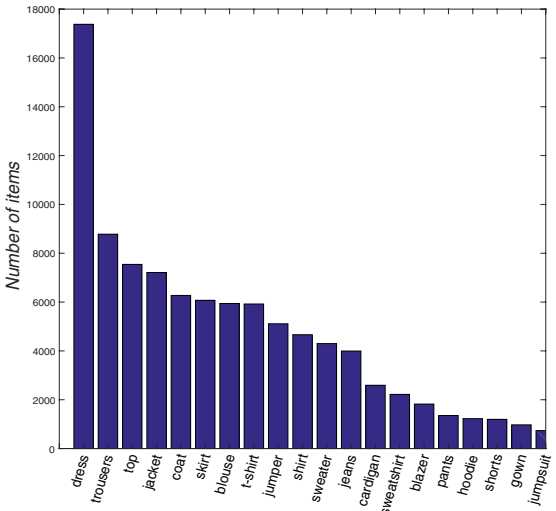
We do not leverage the existing datasets for fashion matching learning such as Deep Fashion [38] or Amazon Product Data [41, 52] because the matching labels in those

Table 1 General statistics of the three fashion datasets

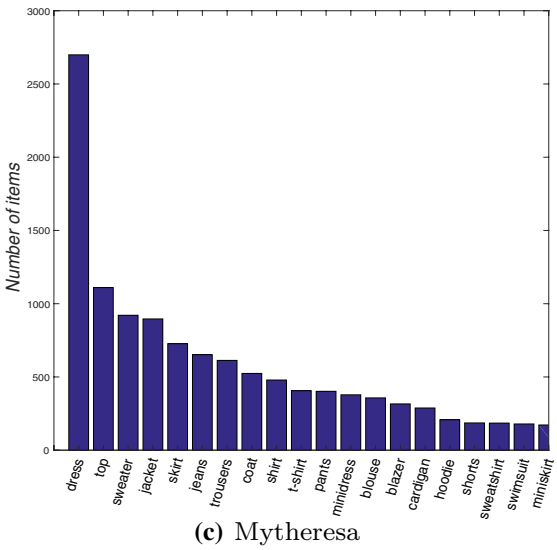
# of	Netaporter	Farfetch	Mytheresa
Total instances	20,868	105,864	12,932
Selected items	17,488	31,788	7548
Categories	58	202	60
Matching pairs	27,490	28,978	10,193



(a) Netaporter



(b) Farfetch



(c) Mytheresa

Fig. 1 The Number of clothing in top 20 categories for Netaporter, Farfetch and Mytheresa

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

datasets are constructed purely according to customers' shopping carts in single transactions so that the relationship between items highly relies on the co-purchased history. Obviously, co-purchased items cannot be guaranteed relevant or matched with each other which results in the matching labels generated in this way are not reliable for fashion matching supervision.

In addition, the real-world data on the website contains a lot of noises such as typo, wrong labels, ambiguity of name, strange id numbers and offline 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 offline 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. Furthermore, we only focus on the items being labelled. Before training, we select those positive records which at least are labelled with another item.

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 in later Sect. 4.1.4.

The statistic of items in top 20 categories for Netaporter, Farfetch and Mytheresa datasets is shown in Fig. 1 and the detailed descriptions are as follows:

Netaporter is a real-life dataset originally consisting of 20,868 clothing items which are collected from www.Netaporter.com/au website. We pick up 17,488 instances which have at least one matched clothing. Totally, there are 27,490 matching pairs and they all belongs to 58 categories.

Farfetch is an accumulation of 105,864 clothing images which are released from www.farfetch.com/au website. After selecting valid records which have at least one matched pair, 31,788 instances are left. Totally, there are 28,978 matching pairs and 202 categories.

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

Mytheresa consists of 12,932 records collected from www.mytheresa.com/en-au website. Among these, 7,548 records are valid and total number of pairs reaches 10,193 which belongs to 60 categories.

4.1.2 Feature Extraction

The deep Convolutional Neural Networks (CNNs) [48] is employed in this work to capture the visual appearance of clothing items based on VGG-16. Specifically, VGG-16 is a powerful pre-trained network for classification on the dataset ImageNet [10] which has more than a million images belonging to 1000 object categories. VGG-16 model contains 13 convolutional layers including 5 max-pooling layers and three fully connected layers. To make use of this pre-trained model, 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 self-augmentation of matching matrix.

The tool we utilized to extract feature is Convolutional Neural Networks for MATLAB (MatConvNet) [50] which is a MATLAB toolbox implementing CNNs for computer vision applications. The version of MatConvNet is 1.0-beta25.

4.1.3 Description of Matching Matrix

The matching matrix indicates the identified matching items based on both the professional advices and the self-augmented relationships Fig. 2. The original clothing items are divided into different categories, such as T-shirt, pants, skirt, 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 advices that are provided by the websites. All these advices are hand-picked (i.e. manually generated) and obviously quite limited. Due to this, the matching matrix is very sparse.

4.1.4 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.

4.1.5 Baselines and Implementation Details

We compare our DSFCH with four state-of-art supervised hashing methods, including Supervised Hashing with Kernels (KSH) [35], Inter-media hashing (IMH) [49], Iterative Quantization based on Canonical Correlation Analysis (CCA-ITQ) [12] and Supervised Discrete Hashing (SDH) [45]. For baselines, we follow the suggested or default parameters provided by the authors and report the best results in 5 runs for each code length on different datasets.

KSH simulates discrete constraints by replacing the sign function with the sigmoid function to catch no-linear manifold hidden structure in the data and it shows effectiveness in generating compact neighbourhood-preserving hash codes. We assign the same anchor number for KSH and DSFCH.

IMH learns linear hash functions for mapping features in different views into a common Hamming space by preserving the inter-view and intra-view consistency. We set both views as the same training data and the number of shared image with tags $n2$ equals to training data size. Both parameters λ and β are assigned as 1 suggested by the corresponding author.

CCA-ITQ applies an orthogonal rotation on mapped training data to decrease the quantization error based on the pre-computed mapping step Canonical Correlation Analysis (CCA) [17]. CCA is a classic supervised approach to learn a common latent subspace for images from different modalities. In the subspace, CCA maximizes the correlation between matched images which is widely used for cross-modal retrieval. MATLAB provides the public code of CCA. We followed the default settings and provided the original CNN features as input.

SDH directly learns the binary hash codes without relaxing the discrete constraints. The optimization process utilizes discrete cyclic coordinates descent. Detailed settings for SDH are: maximum iterations number is 5, $\lambda = 1$, $\nu = 1e-5$, same anchor number with DSFCH for all the three datasets, *RBF* kernel and *L2* loss are selected.

The proposed DSFCH has two trade-off parameters λ and ν which balance the regularization terms. We empirically set λ and ν to be 10^{-2} for experiment on three datasets which leads to good performance. In the later Sect. 4.3.3, we provide a parameter sensitivity study about λ and ν. The self-augmentation parameter n is set to be 3 for all three datasets and the density scale of matching pairs is justified in later Sect. 4.3.1. The maximum iteration number K is defined as 15 for all three datasets, and a discussion about the convergence speed is given in later Sect. 4.3.2.

To summarize the training and query time, we perform 5 runs for DSFCH and the baselines with the various code lengths. Then, the average time consumption is calculated for comparison. In experiments, hash code length on all the datasets is varied in the range of bits [16, 32, 64, 128] to observe the performance. Some different settings are discussed below:

(1) For Netaporter, we split this dataset into a training set of 16,488 instances and a query set of 1000 instances with the same initial seed. The number of anchor points m is defined as 1000. The kernel width ϵ equals to $2 * 275^2$.
(2) For Farfetch, We pick up 1,000 instances as the query set with the same initial seed and make the rest of the dataset being the training set and testbed which contains 30,788 clothing items. The number of anchor points $m = 2000$. The kernel width ϵ equals to $2 * 278^2$.
(3) For Mytheresa, We take 500 records as the query set with the same initial seed and the remaining 7,048 as the training set and retrieval set. The number of anchor points m is 500. The kernel width ϵ equals to $2 * 256^2$.

All the experiments are implemented on MATLAB R2016b platform installed on a workstation which has 4-core Intel

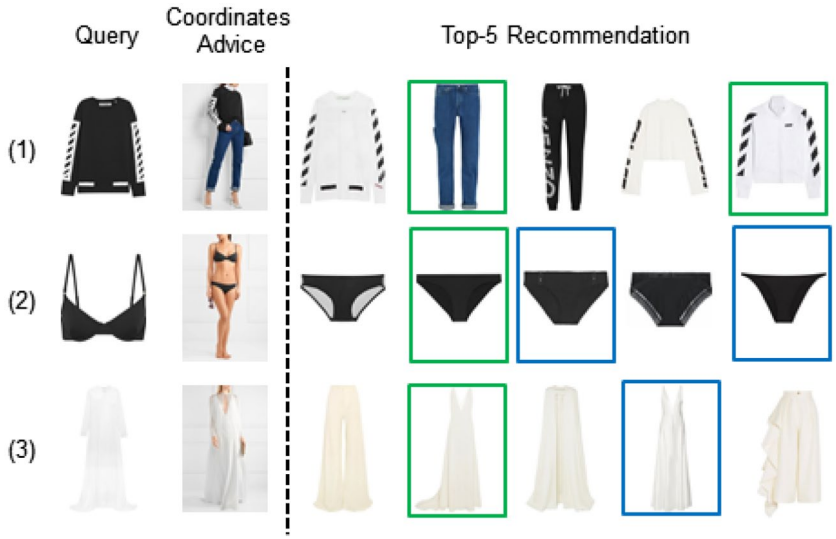


Fig. 4 Example of fashion recommendation by DSFCH with top-5 returned candidates. The returned clothing items that are the same as the recommended ones by professionals are highlighted with green frame. The items bounded in blue boxes are the same as the matched ones identified by the proposed self-augmentation 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

Core i7-4790k 4.00 GHZ CPUs with 28 GB RAM and 64-bit Windows 10 operating system.

4.1.6 Evaluation Metrics

In our experiments, our major task is to recommend a series of matching images by given a query image. However, different to the conventional classification or content-based retrieval tasks, our proposed approach is to utilize professional supervision to find out latent matching image pairs that might be very dissimilar to each other on visual appearance. Learning this intrinsic relationship between related items is a classic problem in recommender system but it faces challenges such as time efficiency and cold start issues [40]. We translate this recommendation problem into a retrieval task to overcome above difficulties and take advantage of hashing technique to enhance the efficiency. In other words, finding the ranking score of the correlative probabilities for recommender system is transformed into finding ranked distance in Hamming space for hash learning.

For our proposed DSFCH and KSH, the ranking score is calculated by inner product of query and database in binary codes. For the rest compared approaches, the negative value of Hamming distance is adopted to measure the correlative probabilities between binary codes of query and database. Videlicet the less Hamming distance it has, the higher ranking score it will be.

To evaluate the recommendation performance of each model, we compute a widely used performance measurement in recommender system, namely AUC (Area Under the ROC curve). 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:

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

where $\delta(\cdot)$ is an indicator function that counts valid conditional cases and 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. Note that the label information used to validate is augmented by augmentation process, where details of n refers to Sect. 4.3.1 and the example of label usage refers to Fig. 4.

4.2 Overall Comparison with Baselines

To provide an overall evaluation for the proposed DSFCH and all compared methods, we discuss both the AUC and time consumption performance in the Table 2 on the datasets Netaporter, Farfetch and Mytheresa. A query example of coordinates recommendation is shown in Fig. 4.

AUC results of all comparison approaches are reported in the first segment of each table in Table 2. For an overview of the result, our DSFCH outperforms the compared baselines in most cases and the performance increases significantly along with the growth of the binary code length on all the

Table 2 AUC and time consumption results of all approaches with various code lengths on datasets Netaporter, Farfetch and Mytheresa

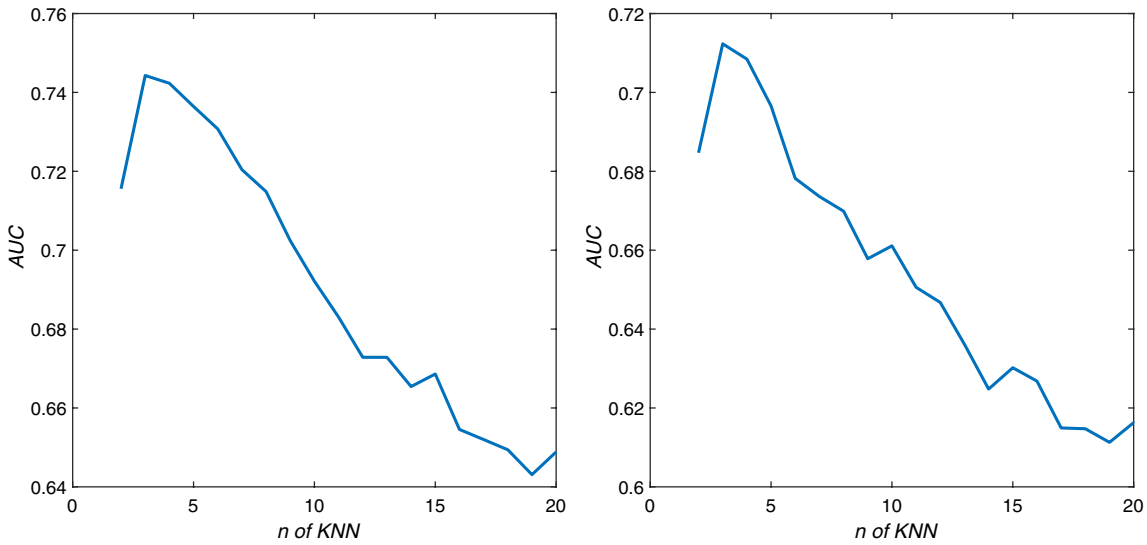
	Bits	KSH	IMH	CCA-ITQ	SDH	DSFCH
(a) Netaporter						
AUC	16	**0.5746**	0.4559	0.5033	0.4771	0.5575
	32	0.5460	0.4714	0.5038	0.4115	**0.5707**
	64	0.5363	0.4854	0.5018	0.4531	**0.6475**
	128	0.5237	0.4933	0.5076	0.4958	**0.7220**
Training	16	(8.4 ± 2.0)e2	1.8e3 ± 25.3	1.3e3 ± 0.9	(5.0 ± 0.2)e2	87.7 ± 2.3
	32	(1.6 ± 0.3)e3	1.8e3 ± 13.3	1.3e3 ± 3.6	(1.3 ± 0.1)e3	93.9 ± 2.7
	64	(3.5 ± 1.1)e3	1.7e3 ± 66.9	1.3e3 ± 2.5	4.4e3 ± 43.1	1.3e2 ± 1.9
	128	(9.5 ± 1.8)e3	1.6e3 ± 9.2	1.3e3 ± 3.5	9.5e3 ± 77.7	1.6e2 ± 1.4
Test	16	(5.7 ± 0.5)e−2	4.0 ± 0.06	2.6 ± 0.7	0.96 ± 0.03	(7.7 ± 0.2)e−2
	32	(6.0 ± 0.1)e−2	6.1 ± 0.2	3.9 ± 0.1	1.7 ± 0.2	(8.1 ± 0.3)e−2
	64	(6.8 ± 0.3)e−2	9.9 ± 0.4	8.2 ± 0.2	3.4 ± 1.6e−2	(9.5 ± 0.3)e−2
	128	0.82 ± 9.0e−3	19.0 ± 1.0	17.4 ± 0.2	7.6 ± 4.2e−2	0.12 ± 8.9e−3
(b) Farfetch						
AUC	16	**0.5536**	0.4629	0.5079	0.4954	0.5354
	32	**0.5620**	0.4807	0.5111	0.4845	0.5614
	64	0.5604	0.4905	0.5077	0.4723	**0.6311**
	128	0.6131	0.5001	0.5027	0.5205	**0.7258**
Training	16	(2.9 ± 0.05)e3	4.5e3 ± 25.0	3.1e3 ± 1.0	(1.9 ± 0.1)e3	(7.2 ± 1.0)e2
	32	(5.7 ± 0.09)e3	4.5e3 ± 9.2	3.1e3 ± 1.8	(4.8 ± 0.1)e3	(8.3 ± 0.8)e2
	64	(1.2 ± 0.08)e4	4.6e3 ± 1.4e2	3.2e3 ± 1.6	(1.3 ± 0.01)e4	(8.1 ± 1.0)e2
	128	(2.3 ± 0.01)e4	4.8e3 ± 12.5	3.2e3 ± 1.2	(3.2 ± 0.01)e4	(9.1 ± 0.6)e2
Test	16	0.25 ± 0.1	4.9 ± 1.5	3.1 ± 0.2	2.5 ± 0.3	2.9 ± 8.8e−2
	32	0.24 ± 0.2	7.4 ± 1.0	6.0 ± 0.2	3.3 ± 0.1	2.9 ± 6.4e−2
	64	0.47 ± 5.4e−3	21.3 ± 6.8	12.8 ± 0.5	7.4 ± 0.2	2.9 ± 9.1e−2
	128	0.49 ± 2.2e−3	36.9 ± 5.1	26.7 ± 1.2	16.03 ± 0.4	3.0 ± 0.2
(c) Mytheresa						
AUC	16	0.5806	0.4765	0.5001	0.5477	**0.5830**
	32	0.5516	0.4794	0.4959	0.5698	**0.6507**
	64	0.5301	0.4954	0.5005	0.5095	**0.7173**
	128	0.5086	0.5024	0.5115	0.5513	**0.7539**
Training	16	40.7 ± 1.4	1e2 ± 69.5	210.8 ± 0.4	93.9 ± 7.3	17.0 ± 0.6
	32	78.0 ± 2.3	1e2 ± 70.5	249.3 ± 2.5	249.1 ± 7.3	19.8 ± 0.9
	64	1.5e2 ± 2.9	1e2 ± 69.0	297.3 ± 2.7	(6.4 ± 0.1)e2	32.2 ± 0.6
	128	3.5e2 ± 8.9	1e2 ± 73.5	221.2 ± 1.3	(1.7 ± 0.03)e3	44.5 ± 1.6
Test	16	(1.6 ± 0.8)e−2	0.12 ± 0.1	0.1 ± 6.4e−3	(8.6 ± 3.9)e−2	(1.7 ± 0.1)e−2
	32	(1.3 ± 0.1)e−2	0.61 ± 0.4	0.9 ± 1.5e−2	(3.9 ± 1.3)e−2	(1.8 ± 0.2)e−2
	64	(1.4 ± 0.2)e−2	1.15 ± 0.8	1.7 ± 6.8e−2	0.69 ± 9.3e−3	(2.1 ± 0.2)e−2
	128	(1.9 ± 0.2)e−2	2.31 ± 2.3	3.6 ± 0.2	1.6 ± 0.5	(3.0 ± 0.3)e−2

The training and testing times are in seconds. The best AUC result in each row is highlighted with bold

three datasets. By contrast, other methods show less sensitivity of code length and long code length does not help them improve the performance well. We can conclude a point that the length of learned binary codes plays a significant role of preserving latent semantic similarities for our proposed method.

For the Netaporter dataset, our method DSFCH exceeds at 32, 64 and 128 code bits. The largest improvement appears on 128 bits about 37.9% than the second best baseline KSH. KSH shows a better performance on 16 bits. For Farfetch dataset, our method DSFCH outperforms at 64 and 128 code bits. DASFCH has very close results to KSH on 16 and 32 bits and slightly less than KSH. For Mytheresa dataset, DSFCH outperforms the competitors on all cases. In particular, the largest enhancement happens on 128 bits about 36.7% than the second best approach SDH.

Furthermore, we provide the running time of the proposed DSFCH in comparison with the baselines, including



(a) Netaporter **(b)** Farfetch

Fig. 5 Comparison of AUC curves for different Self-augmentation parameter n on 128 bits for two datasets Netaporter and Farfetch

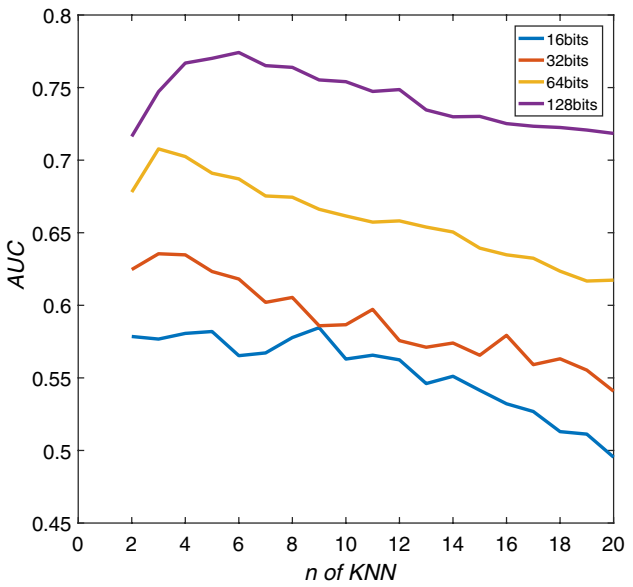


Fig. 6 Comparison of AUC curves for different Self-augmentation parameter n along with various code lengths on the datasets Mytheresa

the training and testing time of each method. In the last part of Sect. 3.3.2, we mentioned that the time complexity of our proposed method is $O(knr + knr^3)$ where $k, r \ll n$. Generally, DSFCH has the least training time in all cases. CCA-ITQ and IMH cost the most training time on 16 bit and have similar time consumption on 32 bits with SDH and KSH. However CCA-ITQ spends a large portion of time on pre-step training (i.e. CCA) which is only involved with training data size and similar as IMH. Therefore, there is not distinct growth along the increase in code length. On the contrary, KSH spends high price on sigmoid operation and the complexity of SDH is highly related code length due to the DCC (solving binary codes bit by bit) so that the time consumption dramatically boosts on 64 and 128 bits. Our algorithm is also exponentially related to code length (r^3) but the basic computational complexity is low (linear n) which mitigates the negative impact of bit growth ($r \ll n$). Finally, it is worth to mention that hash codes have efficient retrieval speed by Hamming distance. In our test phase, inner product of binary codes exceeds the way that directly calculates Hamming distance in most of the cases.

4.3 Comprehensive Analysis on DSFCH

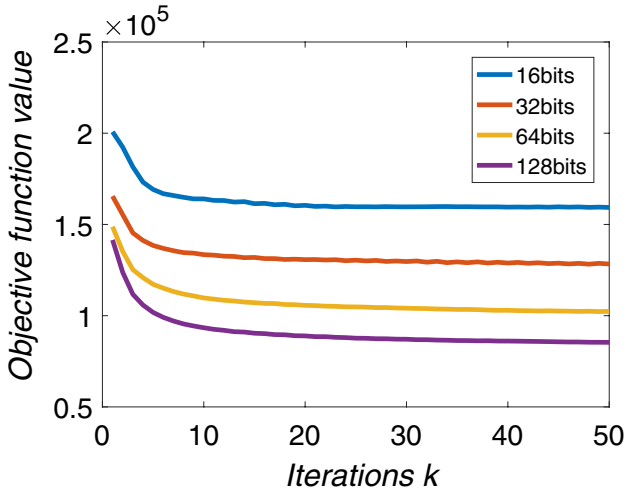
4.3.1 Self-Augmentation Study

We are trying to enrich the density of the matching matrix by KNN search, the experiment result on three datasets Netaporter, Farfetch and Mytheresa with 128 bits. is shown below in Fig. 5. Testing range of n is along with $\{2, 3, \ldots, 20\}$. We can observe that, when the nearest neighbours number $n = 3$, the performance peaks at the highest value in both Netaporter and Farfetch. Along with the number n increasing, the performance is dropping sharply. The reason is that, the raw features of these two datasets in the original space is very distributed and the distances among CNN features only guarantee items resembling each other on visual appearance in a tiny set. If n is large, there is not enough real similar items on visual content supporting the learning process which leads to over-fitting.

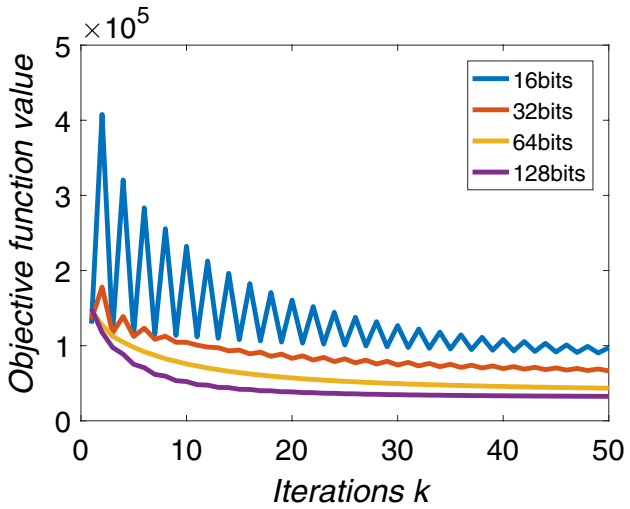
Table 3 Comparative performance between discrete or relaxed methods on Netaporter dataset.

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

The higher AUC result in each column is highlighted with bold



(a) Discrete



(b) Relaxed

Fig. 7 Objective function value variations with the number of iterations on discrete and relaxed methods

A special case is Mytheresa, which requires different level of n variations with code length. Firstly, we observe that the AUC on Mytheresa reaches the peak at $n = 6$ on 128 bits. Therefore, we conduct extra experiments on Mytheresa along with code lengths $\{16, 32, 64, 128\}$ which are shown in Fig. 6. Compared to the other two datasets, the same situation happens on 32 and 64 bits that $n = 3$ outperforms the other competitors. When the code length equals to 16 bits, the performance is fluctuating before $n = 10$, peaking at $n = 9$. Then, it decreases significantly. As it can be seen from the figure, augmentation process does enhance the qualities of the learning binary codes and it works effectively near $n = 3$ for most of the cases. Therefore, we apply this value $n = 3$ for all the comparison approaches which is sufficient to show the effectiveness and efficient of our proposed method.

4.3.2 Discrete and Convergence Study

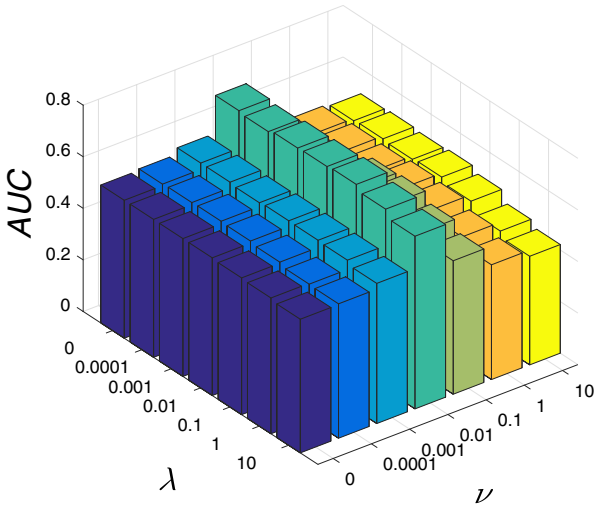
In practical, we compare the performance in two situations: (1) we keep the binary constraint of B in Eq. (11) during training and (2) we relax the discrete constraints to get a continuous B and threshold it at last. In most cases, we notice that discrete method is better than relaxed method. Keeping 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).

Figure 7 shows the convergence procedures in discrete and relaxed respectively. It can be seen from the figure that discrete method is much faster to get convergent than the other one. Usually, discrete method could converge within 15 rounds of iterations, which is set as the maximum iteration number in previous experiments. In addition, the convergence curve of discrete is more stable than the relaxed one because directly thresholding the optimized value at last brings accumulated quantization loss into the optimization process. Although each iteration tries to mitigate the loss repeatedly, some of the efforts are wasted. As a consequence, discrete learning process wins the better prize, applied to all the experiments.

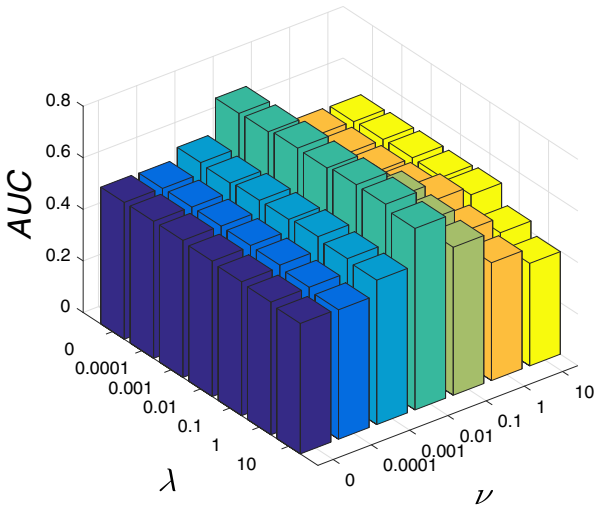
4.3.3 Parameter Sensitivity Study

In previous experiments, we empirically set the two involved parameters λ and ν in the objective function of DSFCH (i.e. Eq.6) as 10^{-2}. The λ is the penalty parameter of H to avoid over-fitting of the binary codes. The parameter ν is the trade-off parameter used to balance the matching pattern loss and discrete binary representation. In this subsection, we analyse their effects on the qualities of the learned binary codes.

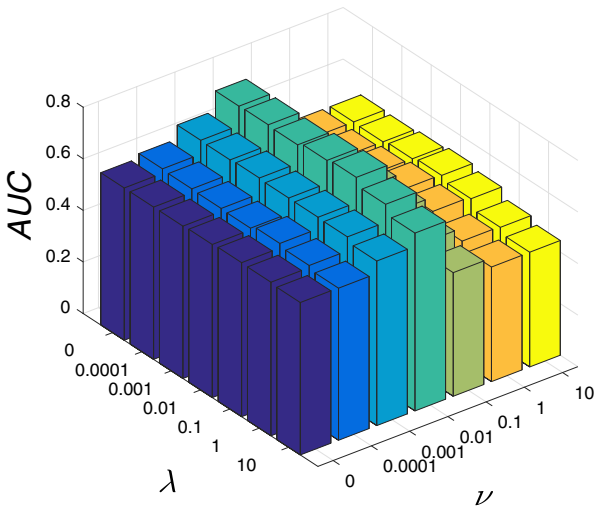
By prefixing the code length as 128 bits and augmentation level n as 3, we vary both λ and ν along with $\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}\}$ on all three datasets. The evaluation is conducted by changing one parameter while



(a) Netaporter



(b) Farfetch



(c) Mytheresa

Fig. 8 DSFCH AUC performance variations with parameters λ and ν when binary code length is 128, augmentation number n is 3, on datasets Netaporter, Farfetch and Mytheresa

fixing the other. Note that when $\nu = 0$, DSFCH learns the binary code only depending on the supervision of matching label information, neglecting of the similarity preserving of content feature representation.

The AUC variations with parameters λ and ν on three datasets are reported in Fig. 8. As it can be observed from the diagrams, ν brings a significant impact on the learning process to achieve the specific retrieval task of this application, while λ is effective but much less than ν. DSFCH peaks the best performance at $\lambda = \nu = 10^{-2}$. Without supervision of matching pattern learning (i.e. ν is extremely large) or only left this part (i.e. ν equals to 0), the qualities of learned binary codes are getting deteriorated dramatically. If there is not any supervision part, the learning process becomes a traditional discrete unsupervised hashing which might be able to work on a retrieval task for single class but definitely failing on cross-categories. If ignoring the discrete binary representation part, obviously the system will get confusing during training because it hardly maps too many dissimilar items together. As a consequence, the two learning parts are very complementary, each with its own sphere of competence and helps each other accomplish the specific application.

5 Conclusion and Future Work

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.

The current work will continue along fashion coordinates retrieval forward for further investigation. A variety of approaches are proposed to incorporate deep learning [15, 43, 61] to learn more discriminative representation of clothing to enhance the performance of fashion recommendation [37]. Deep Convolutional Neural Networks (CNN) [4] models have achieved significant accuracy improvements in computer vision areas [8]. However, they are suffering expensive computational complexity and training phase is

typically time-consuming. In addition, most of them focus on retrieval tasks by introducing auxiliary information such as user purchase history and semantic attributes but discarding professional advices from fashion insiders. More important, fashion coordinates recommendation requires learning and inferring the visual compatibility relationships between different items in an outfit rather than just classifications or feature representations. Based on our previous outcomes, we will keep the professional fashion matching matrix for modelling the compatibility relationships of fashion matching items and utilize hashing technique for efficiency enhancement. Inspired by [16, 22], we plan to generate a deep supervised hashing method which integrates the feature learning and hash function learning into the end-to-end deep learning framework. As another key contributions of our research, we construct real-life fashion datasets with professional matching advices of fashion coordinates and will keep update and maintain our datasets by auto-processing scripts. In addition, we will manually purify the contents of data to improve the quality of the datasets, such as removing damage images and correcting ambiguous descriptions. Finally, this datasets will be released for academic use.

Acknowledgements The work is partially supported by ARC FT130101530.

References

1. Al-Halah Z, Stiefelhagen R, Grauman K (2017) Fashion forward: forecasting visual style in fashion. In: ICCV
2. Andoni A, Indyk P (2008) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun ACM 51(1):117–122
3. Andoni A, Razenshteyn I (2015) Optimal data-dependent hashing for approximate near neighbors. In: STOC, STOC '15. ACM, pp 793–801
4. Arbib MA (2003) The handbook of brain theory and neural networks. MIT press, Cambridge
5. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
6. Bracher C, Heinz S, Vollgraf R (2016) Fashion DNA: merging content and sales data for recommendation and article mapping. CoRR **(abs/1609.02489)**
7. Chen K, Chen K, Cong P, Hsu WH, Luo J (2015) Who are the devils wearing prada in New York city? In: Proceedings of the 23rd ACM international conference on multimedia. ACM, pp. 177–180
8. Chen Q, Huang J, Feris R, Brown LM, Dong J, Yan S (2015) Deep domain adaptation for describing people based on fine-grained clothing attributes. In: CVPR, pp 5315–5324.
9. Datar M, Immorlica N, Indyk P, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: ISCG. ACM, pp 253–262
10. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: CVPR. IEEE, pp 248–255
11. Gionis A, Indyk P, Motwani R et al (1999) Similarity search in high dimensions via hashing. VLDB 99:518–529
12. Gong Y, Lazebnik S, Gordo A, Perronnin F (2013) Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. TPAMI 35(12):2916–2929
13. Gui J, Liu T, Sun Z, Tao D, Tan T (2018) Fast supervised discrete hashing. TPAMI 40(2):490–496
14. Hadsell R, Chopra S, LeCun Y (2006) Dimensionality reduction by learning an invariant mapping. In: CVPR, Vol 2, pp 1735–1742
15. Han X, Wu Z, Huang PX, Zhang X, Zhu M, Li Y, Zhao Y, Davis LS (2017) Automatic spatially-aware fashion concept discovery. arXiv preprint arXiv:1708.01311
16. Han X, Wu Z, Jiang YG, Davis LS (2017) Learning fashion compatibility with bidirectional lstms. In: ACM MM. ACM, pp 1078–1086
17. Hardoon DR, Szedmak S, Shawe-Taylor J (2004) Canonical correlation analysis: an overview with application to learning methods. Neural Comput 16(12):2639–2664
18. He R, Packer C, McAuley J (2016) Learning compatibility across categories for heterogeneous item recommendation. In: ICDM. IEEE, pp 937–942
19. Hsiao WL, Grauman K (2018) Creating capsule wardrobes from fashion images. In: CVPR, pp 7161–7170
20. Iwata T, Wanatabe S, Sawada H (2011) Fashion coordinates recommender system using photographs from fashion magazines. In: IJCAI, vol 22, p 2262
21. Jagadeesh V, Piramuthu R, Bhardwaj A, Di W, Sundaresan N (2014) Large scale visual recommendations from street fashion images. In: SIGKDD, KDD '14. ACM, pp 1925–1934
22. Jiang Q, Li W (2018) Asymmetric deep supervised hashing. In: AAAI
23. Jiang QY, Li WJ (2015) Scalable graph hashing with feature transformation. In: IJCAI, IJCAI'15. AAAI Press, pp 2248–2254
24. Kang WC, Fang C, Wang Z, McAuley J (2017) Visually-aware fashion recommendation and design with generative image models. arXiv preprint arXiv:1711.02231
25. Kiapour MH, Han X, Lazebnik S, Berg AC, Berg TL (2015) Where to buy it: matching street clothing photos in online shops. In: ICCV, pp 3343–3351
26. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
27. Kulis B, Grauman K (2009) Kernelized locality-sensitive hashing for scalable image search. In: ICCVl conference on computer vision. IEEE, pp 2130–2137
28. Kulis B, Jain P, Grauman K (2009) Fast similarity search for learned metrics. TPAMI 31(12):2143–2157
29. Lew MS, Sebe N, Djeraba C, Jain R (2006) Content-based multimedia information retrieval: state of the art and challenges. TOMM 2(1):1–19
30. Liong VE, Lu J, Wang G, Moulin P, Zhou J (2015) Deep hashing for compact binary codes learning. In: CVPR, pp 2475–2483
31. Liu L, Du X, Zhu L, Shen F, Huang Z (2018) Discrete binary hashing towards efficient fashion recommendation. In: DSFAA. Springer, pp 116–132
32. Liu L, Lin Z, Shao L, Shen F, Ding G, Han J (2017) Sequential discrete hashing for scalable cross-modality similarity retrieval. TIP 26(1):107–118
33. Liu L, Zhu L, Li Z (2017) Learning robust graph hashing for efficient similarity search. In: ADC. Springer, pp 110–122

34. Liu W, Mu C, Kumar S, Chang SF (2014) Discrete graph hashing. In: NIPS, NIPS'14. MIT Press, pp 3419–3427
35. Liu W, Wang J, Ji R, Jiang YG, Chang SF (2012) Supervised hashing with kernels. In: CVPR, pp 2074–2081
36. Liu W, Wang J, Kumar S, Chang SF (2011) Hashing with graphs. In: Getoor L, Scheffer T (eds) ICML. ACM, pp 1–8
37. Liu Z, Luo P, Qiu S, Wang X, Tang X (2016) Deepfashion: powering robust clothes recognition and retrieval with rich annotations. In: CVPR, pp 1096–1104
38. Liu Z, Yan S, Luo P, Wang X, Tang X (2016) Fashion landmark detection in the wild. In: ECCV
39. McAuley J, Pandey R, Leskovec J (2015) Inferring networks of substitutable and complementary products. In: SIGKDD. ACM, pp 785–794
40. McAuley J, Targett C, Shi Q, van den Hengel A (2015) Image-based recommendations on styles and substitutes. In: SIGIR, SIGIR '15. ACM, pp 43–52
41. McAuley J, Yang A (2016) Addressing complex and subjective product-related queries with customer reviews. In: WWW. International world wide web conferences steering committee, pp 625–635
42. Raginsky M, Lazebnik S (2009) Locality-sensitive binary codes from shift-invariant kernels. In: Advances in neural information processing systems, pp 1509–1517
43. Shankar D, Narumanchi S, Ananya H, Kompalli P, Chaudhury K (2017) Deep learning based large scale visual recommendation and search for e-commerce. arXiv preprint arXiv :1703.02344
44. Shen F, Liu W, Zhang S, Yang Y, Tao Shen H (2015) Learning binary codes for maximum inner product search. In: ICCV, pp 4148–4156
45. Shen F, Shen C, Liu W, Shen HT (2015) Supervised discrete hashing. In: CVPR, pp 37–45
46. Shrivastava A, Li P (2014) Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In: NIPS, pp 2321–2329
47. Simo-Serra E, Fidler S, Moreno-Noguer F, Urtasun R (2015) Neuroaesthetics in fashion: modeling the perception of fashionability. In: CVPR, pp 869–877
48. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv :1409.1556
49. Song J, Yang Y, Yang Y, Huang Z, Shen HT (2013) Inter-media hashing for large-scale retrieval from heterogeneous data sources. In: SIGMOD, SIGMOD '13. ACM, pp 785–796
50. Vedaldi A, Lenc K (2015) Matconvnet—convolutional neural networks for matlab. In: ACM MM
51. Veit A, Kovacs B, Bell S, McAuley J, Bala K, Belongie S (2015) Learning visual clothing style with heterogeneous dyadic co-occurrences. In: ICCV, pp 4642–4650
52. Wan M, McAuley J (2016) Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In: ICDM. IEEE, pp 489–498
53. Wang C, Blei DM (2011) Collaborative topic modeling for recommending scientific articles. In: SIGKDD. ACM, pp 448–456
54. Wang H, Wang N, Yeung DY (2015) Collaborative deep learning for recommender systems. In: SIGKDD. ACM, pp 1235–1244
55. Wang J, Kumar S, Chang SF (2010) Semi-supervised hashing for scalable image retrieval. In: CVPR, pp 3424–3431
56. Wang J, Kumar S, Chang SF (2012) Semi-supervised hashing for large-scale search. TPAMI 34(12):2393–2406
57. Wang J, Xu XS, Guo S, Cui L, Wang XL (2016) Linear unsupervised hashing for ANN search in euclidean space. Neurocomputing 171:283–292
58. Weiss Y, Torralba A, Fergus R (2009) Spectral hashing. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) NIPS. Curran Associates, Inc, pp 753–1760
59. Xia R, Pan Y, Lai H, Liu C, Yan S (2014) Supervised hashing for image retrieval via image representation learning. AAAI 1:2156–2162
60. Xu H, Wang J, Li Z, Zeng G, Li S, Yu N (2011) Complementary hashing for approximate nearest neighbor search. In: ICCV, pp 1631–1638
61. Yu W, Zhang H, He X, Chen X, Xiong L, Qin Z (2018) Aesthetic-based clothing recommendation. In: WWW. International world wide web conferences steering committee, pp 649–658
62. Zhang P, Zhang W, Li WJ, Guo M (2014) Supervised hashing with latent factor models. In: SIGIR, SIGIR '14. ACM, pp 173–182