False Negative Distillation and Contrastive Learning for Personalized Outfit Recommendation

Seongjae Kim, Jinseok Seol, Holim Lim, Sang-goo Lee
Seoul National University
Republic of Korea
{sjkim,jamie,ihl7029,sglee}@europa.snu.ac.kr

ABSTRACT
Personalized outfit recommendation has recently been in the spotlight with the rapid growth of the online fashion industry. However, recommending outfits has two significant challenges that should be addressed. The first challenge is that outfit recommendation often requires a complex and large model that utilizes visual information, incurring huge memory and time costs. One natural way to mitigate this problem is to compress such a cumbersome model with knowledge distillation (KD) techniques that leverage knowledge from a pretrained teacher model. However, it is hard to apply existing KD approaches in recommender systems (RS) to the outfit recommendation because they require the ranking of all possible outfits while the number of outfits grows exponentially to the number of consisting clothing items. Therefore, we propose a new KD framework for outfit recommendation, called False Negative Distillation (FND), which exploits false-negative information from the teacher model while not requiring the ranking of all candidates. The second challenge is that the explosive number of outfit candidates amplifying the data sparsity problem, often leading to poor outfit representation. To tackle this issue, inspired by the recent success of contrastive learning (CL), we introduce a CL framework for outfit representation learning with two proposed data augmentation methods. Quantitative and qualitative experiments on outfit recommendation datasets demonstrate the effectiveness and soundness of our proposed methods.

KEYWORDS
Personalized Outfit Recommendation, Knowledge Distillation, False Negative Distillation, Contrastive Learning

1 INTRODUCTION
Personalized outfit recommendation is the task of determining the preference of a user to an input outfit that consists of multiple clothing. It has recently attracted attention with the rapid growth of the online fashion industry, and several related studies [22, 24, 25] have been conducted. However, despite the success of existing works, outfit recommendation has two significant challenges that should be addressed. First, recommending outfits often requires a complex and large model that involves the utilization of visual information (i.e., images) [24]. Such a large model incurs high latency and memory costs during the inference phase, making it difficult to apply to real-time services [14]. The second challenge is that outfit recommendation inevitably suffers from the data sparsity problem because the possible pool of outfit data grows exponentially to the number of consisting clothing items [22]. The sparsity problem often leads to poor learning of outfit representation, which hinders achieving satisfactory recommendation performance [40].

To address the first challenge stemming from a large model, one can employ knowledge distillation (KD) techniques that compress a model by transferring knowledge from a large teacher model to a small student model. Accordingly, one may try to apply existing studies [14, 18, 35] of KD available in recommender systems (RS) to the outfit recommendation. However, existing methods leverage predicted ranking of all possible outfits from the teacher model, so they are not applicable in outfit recommendation tasks with explosively large pools. Therefore, we propose a novel KD framework named False Negative Distillation (FND) that does not require the ranking of all outfit candidates. Similar to most outfit recommendation studies [22, 24, 25], FND utilizes a ranking loss that pulls observed (positive) outfits to a user while pushing unobserved (negative) outfits. As illustrated in Fig. 1, FND claims that unobserved is not the same as true-negative and assumes that negative outfits close enough to the user are false-negative outfits. We show through various experiments that FND is effective, and the assumption is reasonable.

The approach for the second challenge is to deal with the problem of poor outfit representation. As learning enhanced representation of entities is one of the core components to achieve high performance in deep learning [2], numerous works [3, 6, 8, 27] from diverse domains accomplished this with self-supervised learning (SSL) techniques. There have been several studies [26, 41, 44] utilizing SSL techniques in RS as well. Among them, more recent works [23, 40, 43] exploit contrastive learning (CL), especially SimCLR [3], which learns meaningful representations by pulling the differentially augmented view of the same data while pushing the others in the batch. Nevertheless, leveraging CL in outfit recommendation is relatively unexplored. Hence, as illustrated in Fig. 2, we introduce...
We also experiment on the cold-start scenario where users have very few outfits interacted, and the trained model does not have any knowledge of those users. To make appropriate recommendations to cold starters, we introduce two practical strategies that do not require additional training of the model.

Our main contributions can be summarized as follows:

- We propose a new knowledge distillation framework that can be utilized in outfit recommendation tasks without requiring the ranking of all outfit candidates in the system.
- We propose two novel outfit data augmentation methods to leverage contrastive learning in outfit recommendation.
- We introduce two practical strategies to deal with the cold-start problem.
- We demonstrate the effectiveness and soundness of our approaches with comprehensive experiments on fashion outfit recommendation datasets.

2 RELATED WORK

2.1 Outfit Recommendation

Based on whether the individual preference is neglected or not, existing outfit recommendation studies can be classified into two categories: non-personalized [4, 10, 20, 32, 34, 37, 38, 42] and personalized [12, 19, 22, 24, 25] outfit recommendation. Lu et al. [25] used pairwise scores, and they employed the weighted hashing technique to tackle the efficiency problem. Lin et al. [22] utilized an attention mechanism to estimate the preference score, weighting items in an outfit differently. Lu et al. [24] exploited Set Transformer [17], the state-of-the-art model for set-input problems, to capture the high-order interactions among fashion items. They also disentangled each user into multiple anchors to accommodate the variety of preferences. Note that methods based on graph neural networks [4, 19, 32] or predicting distribution over whole clothing items [42] require the test items to be in the training set.

2.2 Knowledge Distillation

Knowledge distillation is a model-agnostic compression strategy for generating efficient models. Since the early success of KD in image recognition [11, 30], KD has been widely accepted in other fields. In recommendation tasks, several works [14, 18, 35] have employed KD. They rank all items with the teacher model and utilize the items of high rank when training the student model. Tang et al. [35] considered top-K items as false-negatives and differentiated their relative importance based on their rankings. Lee et al. [18] trained the student to mimic the predicted probabilities of the teacher on the sampled items of high rank. Kang et al. [14] achieved state-of-the-art performance by transferring both the prediction and latent knowledge of the teacher.

2.3 Contrastive Learning

Contrastive learning is a framework for obtaining high-quality representations to boost the performance of downstream tasks and was first introduced in computer vision [3]. CL enhances representations by maximizing agreement between two differently augmented views of the same data. A few works [23, 40, 43] applied CL to RS, and they showed notable success. In sequential recommendation, Xie et al. [40] applied CL by applying three augmentation methods (crop/mask/reorder) to user interaction history. Yao et al. [43] focused on large-scale item recommendations and employed a two-stage augmentation consisting of masking and dropout. Liu et al. [23] utilized CL for graph neural network based RS by randomly removing some edges.

3 APPROACH

We recommend outfits to users based on their preference score. To compute the preference score, we use user embeddings and vector representations of outfits. Due to the set-like nature of fashion outfits, the representation model requires two conditions. First, the outfit representation should be invariant to the order of comprising fashion items. Second, the model should be able to process input outfits of any size. To this end, we borrow the architecture from LPAE [24] model, which uses Set Transformer [17] module designed to address these set-input problems.

3.1 Background: Computing the Preference Score to an Outfit

As illustrated in Fig. 3, an outfit \( o_j \) with \( n \) items is a tuple of fashion item images: \( o_j = (x_{j1}, x_{j2}, \cdots, x_{jn}) \) ∈ \( \mathcal{I}^n \). Let \( f : \mathcal{I} \rightarrow \mathbb{R}^d \) be a...
Convolutional Neural Network (CNN) with fully connected layers that encodes \( x_{jk} \) into an item feature vector \( x_{jk} = f(x_{jk}) \), where \( d \) is a feature dimension. Through Set Transformer \( T : \mathbb{R}^{nxd} \rightarrow \mathbb{R}^d \), we obtain an outfit representation \( o_j = T(X_j) \) from item features \( X_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]^T \). Then, we compute the preference score with user embedding \( u_i \in \mathbb{R}^d \), where:

\[
    o_j = \cos(u_i, o_j)
\]

The obtained \( o_j \in \mathbb{R}^d \) is a single compact vector representation of an outfit \( o_j \), holding compatibility relationships among consisting fashion items.

\[ r_{ij} = \cos(u_i, o_j) \equiv \frac{u_i^T o_j}{||u_i|| ||o_j||}, \]  

3.2 False Negative Distillation

Large models generally show relatively higher recommendation performance compared to their smaller counterparts. However, employing a small-sized model is necessary to reduce latency and memory costs during the inference phase. Therefore, we propose a novel knowledge distillation framework named False Negative Distillation (FND) that transfers false-negative information extracted from a well-trained large teacher model to a small student model. As illustrated in Fig. 1, in the user-outfit space of a trained teacher model, we assume that negative (i.e., unobserved) outfits close enough to the user are false-negative outfits.

\[ L_{\text{N-pair}} = \frac{1}{N} \sum_{(u_i, o_j) \in B} \log \frac{\exp(r_{ij}/\tau_{\text{FND}})}{\exp(r_{ij}/\tau_{\text{FND}}) + \sum_{k \neq i} \exp(r_{ij}/\tau_{\text{FND}})}, \]  

where \( \tau_{\text{FND}} > 0 \) is a temperature hyper-parameter.
To obtain more enhanced outfit representations, we propose a novel data augmentation method to leverage SimCLR [3] framework in outfit recommendation. Specifically, we suggest two data augmentation methods: erase and replace. Both augmentations randomly alter comprising items from an outfit while preserving the semantic context. We treat the augmentation set as a hyper-parameter and fix them at the beginning of the training. Note that if two identical augmentations are applied, we alter different items from the input outfit to obtain distinct views.

3.3.1 Erase. Randomly erasing components from the input is a common data augmentation method in diverse domains. In sequential recommendation, for example, Xie et al. [40] randomly crop items from user interaction history. In natural language processing, Wu et al. [39] erase or replace randomly selected words in a sentence. In computer vision, DeVries et al. [7] cut out contiguous sections of an input image, inspired by the object occlusion problem. When it comes to outfit recommendation, a subset of an outfit may imply or even determine the semantic information. Motivated by this, we randomly remove one item from the outfit to generate an augmented view.

3.3.2 Replace. Our proposed model computes preference scores based solely on visual information (i.e., images), and this assumption is helpful if access to other metadata is limited. In this situation, the semantic information of an outfit is derived from the appearance of consisting items. Accordingly, we claim that the visual similarity of consisting items leads to the semantic similarity of an outfit. Based on the claim, we generate an augmented view by randomly replacing one item from the outfit with a visually similar item from the same category. To this end, we train a CNN autoencoder model and retrieve similar items through their latent features.

3.4 Final Objective: FND-CL

The proposed losses $L_{\text{FND}}$ and $L_{\text{CL}}$ can be used independently; hence we can take advantage of both methods. Therefore, our final objective is to minimize the weighted sum of both losses as follows:

$$L_{\text{FND-CL}} = L_{\text{FND}} + \lambda L_{\text{CL}},$$

where $\lambda$ is a loss weight hyper-parameter.
users, or cold starters, have relatively few interactions in general, and the deployed model does not have any prior knowledge of those users. In practice, fine-tuning the model for them is a time-consuming process; thus, cold starters might starve for the recommendation until the next iteration of deployment. Therefore, it is necessary to have an alternative recommendation method that exploits the already deployed model with no additional training. In personalized outfit recommendation, only a few works [24] handled the cold-start problem without fine-tuning the model. Here, we introduce two strategies to compute preference scores of cold starters, analogous to memory-based collaborative filtering.

Given a cold starter \(u_c\), we define a neighborhood \(N_c\) from the set of non-cold users \(U\) as follows: with a set of indices of positive outfits \(J^+\) for \(u_c\),

\[
s_{ci} = \frac{1}{|J^+|} \sum_{j' \in J^+} r_{ij'},
\]

\(i \in N_c = \{u_i \in U \mid s_{ci} > \delta \lor i = i_c\},\)

where \(s_{ci}\) represents the asymmetric similarity from \(u_c\) to \(u_i\), \(\delta\) is a similarity threshold, and \(i_c = \arg\max s_{ci}\) denotes the index of the most similar user, which ensures at least one neighbor for each \(u_c\). To compute the preference score \(r_{cij}^{\text{cold}}\) of the cold starter \(u_c\) to a given outfit \(o_j\), we aggregate the preference scores of neighbors to the outfit. Here we use two aggregation strategies: Average and Weighted Average.

3.5.1 Average (avg). A basic aggregation strategy is simply averaging the preference scores:

\[
r_{cij}^{\text{cold}} = \frac{1}{|N_c|} \sum_{u_i \in N_c} r_{ij},
\]

3.5.2 Weighted Average (w-avg). We further utilize the similarity between the cold starter and its neighbors as aggregation weights using cross-entropy with temperature:

\[
\bar{s}_{ci} = \frac{\exp(s_{ci}/T_{\text{w-avg}})}{\sum_{u_i \in N_c} \exp(s_{ci}/T_{\text{w-avg}})},
\]

\[
r_{cij}^{\text{cold}} = \sum_{u_i \in N_c} \bar{s}_{ci} r_{ij},
\]

where \(T_{\text{w-avg}} > 0\) is a temperature hyper-parameter, and note that \(\sum_{u_i \in N_c} \bar{s}_{ci} = 1\) holds. We considered other methods for deriving the aggregation weights \(\bar{s}_{ci}\) from \(s_{ci}\); however, the suggested method empirically showed the best and stable results, especially in terms of robustness to hyper-parameters.

### 4 EXPERIMENT

#### 4.1 Experimental Design

4.1.1 Datasets. We use datasets collected from the Polyvore website: Polyvore-U [25], where \(U \in \{630, 519, 53, 32\}\) denotes the number of users. Polyvore-U contains outfits posted by users, each consisting of three categories: top, bottom, shoes. Outfits in Polyvore-\{630, 53\} have a fixed number of items: one item for each category. Polyvore-\{519, 32\} include outfits with a variable number of items (i.e., some outfits may have two tops). We use Polyvore-\{630, 519\} for most of the experiments and Polyvore-\{53, 32\} for cold starter tasks. Statistics of the datasets are provided in Table 1. Following previous works [24, 25], we define user-posted outfits as positive outfits for each user and category-wise random mixtures of items as negative outfits. We also discuss the results of hard negative outfits (i.e., random samples of positive outfits of other users) separately in Sec. 4.4. In the evaluation phase, we set the ratio between positive and negative outfits to 1:10 for each user. We split training, validation, and test sets to 9:2:2, and we further split the validation set into two halves, one for the teacher model and the other for the student model. As [25] affirmed, there are no duplicate items between the training and the test sets for each user.

4.1.2 Evaluation metrics. We evaluate the ranking performance via Area Under the ROC curve (AUC) and Normalized Discounted Cumulative Gain (NDCG), similar to previous works [24, 25]. For each user, we rank the test outfits by the predicted preference score of the model. We report the performance averaged over all users.

4.1.3 Considered methods. We compare our methods with the following state-of-the-art non-personalized [10, 34, 37] and personalized [22, 24, 25] outfit recommendation models. Type-Aware [37] projects pairs of items onto the type-specific subspaces. Compatibility is then measured in these subspaces and learned through the triplet loss. SCE-Net [34] learns conditional embeddings and their weights using an attention mechanism. Each conditional embedding is implicitly encouraged to encode different semantic subspaces via the triplet loss. Bi-LSTM [10] considers an outfit as a sequence of items and uses a bidirectional LSTM to learn the compatibility. The model is trained by predicting the next and previous items in the sequence through cross-entropy loss. OutfitNet [22] consists of two stages to capture both general compatibility and personal taste. The objective of both stages is to maximize the difference between positive and negative scores, similar to BPR. FHN [25] uses pairwise scores to compute outfit compatibility and personal preference simultaneously. We train FHN with BPR without the binarization step, following the previous work [24]. LPAE [24] includes two models LPAE-u and LPAE-g, which mainly handles the cold-start problem using multiple anchors for each user. Both models utilize BPR loss, and LPAE-g has additional general anchors to model non-personalized compatibility. For a more fair comparison, we apply temperature scaling when using BPR or cross-entropy loss.

4.1.4 Implementation details. Similar to the previous work [25], we use AlexNet [16] pretrained on ImageNet [5] as a backbone CNN. We define two versions of AlexNet to experiment the knowledge distillation. One is AlexNet-large, which is the original AlexNet.
We conduct experiments 10 times and report the average results in \( \tau = 0.8 \). As shown in Table 2, our proposed FND outperforms baseline methods under all datasets and metrics. Furthermore, the performance of FND-CL shows the effectiveness of the outfit CL framework. Recall that LPAE-u adopts multiple anchors for representing each user, and LPAE-g further leverages non-personalized compatibility. Comparison with LPAE models shows that FND can effectively achieve improved performance without auxiliary parameters and structures.

### 4.2 Performance Comparison

As shown in Table 2, our proposed FND outperforms baseline methods under all datasets and metrics. Furthermore, the performance of FND-CL shows the effectiveness of the outfit CL framework. Recall that LPAE-u adopts multiple anchors for representing each user, and LPAE-g further leverages non-personalized compatibility. Comparison with LPAE models shows that FND can effectively achieve improved performance without auxiliary parameters and structures.

### 4.3 Performance on Cold Starters

We evaluate the ranking performance of recommending to the cold starters. Concretely, we test a scenario where a model is trained in Polyvore-[630, 519], and cold starters in Polyvore-[53, 32] desire recommendations. Following the previous work [24], we experiment on the circumstances that each cold starter has only 1 or only 5 interacted outfits. We compare our methods with non-personalized and LPAE methods, which do not require additional training of the model. For LPAE methods, we use the anchor-search [24], which is known to be the most effective strategy in the cold-start case. In FND and FND-CL, we evaluate both avg and w-avg strategies. We conduct experiments 10 times and report the average results in Table 3. The results show that our approaches consistently outperform baseline methods even though the primary purpose of LPAE methods is to deal with the cold-start problem. The w-avg strategy is more effective than the avg strategy in both FND and FND-CL, implying the importance of considering neighbors differently based on similarity rather than treating them equally.

### 4.4 Performance on Hard Negative Outfits

We test a more challenging case where negative outfits in evaluation are composed of positive outfits of other users (i.e., hard negatives). We only compare with personalized methods since non-personalized methods cannot distinguish users. Following the previous work [25], we set half of the negative outfits to hard negative outfits. We only compare with personalized methods since non-personalized methods cannot distinguish users. Following the previous work [25], we set half of the negative outfits to hard negative outfits. SGD with momentum [28], and the number of heads \( h = 8 \). We set \( \alpha \) to 1.25 for Polyvore-630 and 1.5 for Polyvore-519. When it comes to CL, the pair of augmentations are (\( e_r \), \( e_p \)) for Polyvore-630 and (\( e_r \), \( e_p \)) for Polyvore-519. SGD with momentum [28] is used to train all methods, and the batch size is set to 32. For each method, we report the test performance with their optimal hyper-parameters searched via the validation set unless otherwise specified.
4.5 Performance with Different $\alpha$

We evaluate the performance of FND under various $\alpha$ (see Eq. 12). The results are reported in Fig. 4, and we also show the performance of $N$-pair and BPR in the same figure. As mentioned in Sec. 3.2.1, $N$-pair overcomes the partial shortcoming of BPR by considering multiple negative outfits in each update and thus clearly outperforms BPR. Different datasets tend to have different optimal $\alpha$, but given adequate value, FND can surpass the performance of a strong $N$-pair.

4.6 Performance with Different Augmentations

To test the performance of FND-CL for all possible augmentation methods, we put the identity function into the set of augmentations. The results are shown in Fig. 5, and we also report the performance of FND in the same figure. As mentioned in Sec. 3.2.1, $N$-pair shares the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

Detailed information on inference efficiency is measured for each model of different sizes and reported in Table 5. We conduct experiments using FND for the student model and $N$-pair for the teacher model. Since FND affects only the training step, FND and $N$-pair share the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

4.7 Performance with Different Model Sizes

We study the impact of model size on performance, especially in the case of the student model. We consider student models with three different sizes (i.e., XS, S, M) and the teacher model. XS uses AlexNet-small, and S and M use a downsized version of AlexNet that output dimensions of all fully-connected layers are reduced to 1/4 and 1/2, respectively. Fig. 6 shows the results. It is clear that the larger the size, the better the performance. Moreover, the fact that FND-CL outperforms FND and FND outperforms $N$-pair is consistent regardless of the size of the model, supporting that our approaches are meaningful.

The performance gap between FND and $N$-pair appears to shrink with the increasing size of the student model. Such a tendency implies that the effectiveness of FND depends on the performance gap between the teacher and the student model, which implicitly emphasizes the importance of utilizing the superior teacher model. On the other hand, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

Detailed information on inference efficiency is measured for each model of different sizes and reported in Table 5. We conduct experiments using FND for the student model and $N$-pair for the teacher model. Since FND affects only the training step, FND and $N$-pair share the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

Detailed information on inference efficiency is measured for each model of different sizes and reported in Table 5. We conduct experiments using FND for the student model and $N$-pair for the teacher model. Since FND affects only the training step, FND and $N$-pair share the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

Detailed information on inference efficiency is measured for each model of different sizes and reported in Table 5. We conduct experiments using FND for the student model and $N$-pair for the teacher model. Since FND affects only the training step, FND and $N$-pair share the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

Detailed information on inference efficiency is measured for each model of different sizes and reported in Table 5. We conduct experiments using FND for the student model and $N$-pair for the teacher model. Since FND affects only the training step, FND and $N$-pair share the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.

Detailed information on inference efficiency is measured for each model of different sizes and reported in Table 5. We conduct experiments using FND for the student model and $N$-pair for the teacher model. Since FND affects only the training step, FND and $N$-pair share the same inference efficiency. Note that if CL is added, only the number of parameters increases by about 0.03M. In all inference tests, we use PyTorch with CUDA from Tesla P100 SXM2 GPU and Xeon E5-2690 v4 CPU. From the results, we can see that the effectiveness of the CL framework is hardly affected by the size of the model, as expected.
Table 5: Model compactness and inference efficiency. “Time” denotes model inference time for making recommendations to every user in each dataset, and we report the mean and standard deviation of 10 runs. “Memory” represents GPU memory usage. “Ratio” indicates the relative parameter size of the student model compared to the teacher model.

| Dataset  | Model | Time   | Memory  | # Params | Ratio |
|----------|-------|--------|---------|----------|-------|
| Polyvore-630 | Teacher | 80.5±1.7 | 2.89GB  | 65.99M | -  |
| XS       | 74.1±1.7 | 2.64GB  | 2.85M  | 4.3%    |       |
| S        | 75.2±2.9 | 2.68GB  | 13.88M | 21.0%   |       |
| M        | 76.3±3.3 | 2.75GB  | 28.10M | 42.6%   |       |
| Polyvore-519 | Teacher | 65.4±2.7 | 3.05GB  | 65.97M | -  |
| XS       | 58.3±3.0 | 2.80GB  | 2.83M  | 4.3%    |       |
| S        | 61.6±2.7 | 2.84GB  | 13.86M | 21.0%   |       |
| M        | 64.4±1.9 | 2.91GB  | 28.09M | 42.6%   |       |

Figure 7: Comparison of different batch sizes on Polyvore datasets.

sufficient number of negative samples per each update when using the ranking loss.

4.9 Visualization of the User-Outfit Space

We visualize the user-outfit space of the teacher model to support the intuition of FND (see Fig. 1). The visualization uses t-SNE [36] and shows three users and their positive and negative outfits from the training set. We focus on the training phase since the approach of FND is to distill knowledge from the teacher model when training the student model. The results are shown in Fig. 8. Recall that negative outfits are randomly generated, and thus a positive outfit from the test set can appear as a negative sample by pure chance in the training step. With the help of the teacher model, the student model can treat such samples as false-negatives, denoted as a dash-bordered rectangle in the figure. Moreover, other negative outfits close to a user share a similar style with positive outfits, showing the possibility of being false-negatives. Hence, we can conclude that the approach of FND that utilizes false-negative information from the teacher model is reasonable.

5 CONCLUSION

In this paper, we study how to leverage knowledge distillation (KD) and contrastive learning (CL) framework for personalized outfit recommendation. We propose a new KD framework named False Negative Distillation (FND) that does not require the ranking of all possible outfits. We also propose two novel data augmentation methods to make use of the CL framework in outfit recommendation. Quantitative experiments show that our FND and CL achieve notable success in outfit recommendation tasks. In detail, FND outperforms the state-of-the-art methods under fair conditions and achieves improved performance than without using FND in the same model. The outfit CL framework also contributes to the recommendation performance by allowing the model to obtain a more meaningful outfit representation. We support the soundness of our FND by visualizing the user-outfit space of the teacher model. One interesting future work is to apply a contrastive learning framework in a supervised manner by treating each user as a class.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450 (2016).
[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35, 8 (2013), 1798–1828.
[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In International conference on machine learning. PMLR, 1597–1607.
[4] Guillem Cucurull, Perouz Taslakian, and David Vazquez. 2019. Context-aware visual compatibility prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 12617–12626.
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition. Ieee, 248–255.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
[7] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017).
