Personalized Compatibility Metric Learning

Meet Taraviya
UCLA
Los Angeles, USA

Anurag Beniwal
Amazon
Seattle, USA

Yen-Liang Lin
Amazon
Palo Alto, USA

Larry Davis
Amazon
New York, USA

ABSTRACT

Recommending sets of items that include both personalized and compatible items is crucial to personalized styling programs such as Amazon's Personal Shopper. There is both an extensive literature on learning generic fashion compatibility and also on personalization in fashion. However, recommending pairs of items that the customer would like to wear together is still less studied as it involves learning a compatibility metric personalized to each customer. We propose a new framework (PSA-Net) to learn compatibility that is personalized to the customer - a customer dependent subspace learning framework where attention weights of subspaces are learnt using customer representations. We evaluate our approach on compatibility data provided directly by customers. Our approach outperforms the non-personalized approach in predicting compatibility preferences of customers. In other words, an approach that learns a common compatibility metric for all customers. In addition, we establish the advantage of data collected directly from customers compared to data collected from human stylists in predicting compatibility for Amazon customers.

CCS CONCEPTS

• Information systems → Recommender systems; **Collaborative filtering**; **• Applied computing → Online shopping**;

KEYWORDS

Neural Networks, Compatibility, Fashion, Attention, Complementarity, Recommendation Systems, Compatibility

ACM Reference Format:
Meet Taraviya, Anurag Beniwal, Yen-Liang Lin, and Larry Davis. 2021. Personalized Compatibility Metric Learning. In *Proceedings of KDD 2021 (KDD'21).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Personal shopper by Prime Wardrobe (PSPW) [1] is a subscription service that provides personalized styling using machine learning with humans in the loop to recommend sets of items to customers. Within the Personal shopper experience, *Love It or Leave It* is a fun and frictionless way for any Amazon customer to share their style likes and dislikes through a simple thumbs up/down interaction. Customer feedback is used to improve understanding of their style, resulting in better recommendations. Customers can vote items individually as well as pairs of items (Figure 1). Votes on individual items are used to learn representations of customers and items to predict their preferences on individual items. Votes on pairs of items are used to learn compatibility of pairs of items for each customer. The data collected from Love it or leave it, in addition to other data sources, is used to train recommendation models. These models along with inputs from fashion experts are used to curate a personalized box for the customers. Customers can choose what they want to try and are only charged for items they keep after trying (See appendix A).

Fashion compatibility is subjective and depends on several factors including the preference of customer for the individual items, and what they like to wear together. Depending on various factors like profession, location and age compatibility could mean different things to different customers, for e.g. for a particular customer black top on a white jeans is more compatible than a pink top on a white jeans and for another customer the opposite could be true. There is significant literature that addresses the problem of recommending compatible items but most learn a common compatibility metric for everyone ignoring personal tastes of individual customers. Most commonly, these models are trained on open source datasets like Maryland Polyvore [8] [7] or datasets labelled by fashion experts and do not take into account customer level feedback on compatibility. Even in papers that address personalized compatibility like [10], co-purchase is often used as a proxy for compatibility which is prone to adding noise to the model as people might buy things together without any intent to wear them together. We propose a subspace attention based architecture that projects item images into multiple embedding subspaces. Different subspaces represent different notions of compatibility and the importance of each subspace in learning a customer's compatibility metric is determined by its attention weight. The attention weights are dependent on customer embeddings learnt using matrix completion-like algorithms. The number of subspaces is a hyperparameter in the model.

Our contributions are two fold 1) A novel architecture that takes into account the customer side information to project item embedding into multiple subspaces using attention to learn a personalized metric of compatibility for each customer 2) A two stage training process that learns customer representations from item level

KDD'21, August 14–18, 2021,
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

preferences in the first stage and then use these learnt customer representations along with pairwise preference data to predict pairwise compatibility in the second stage.

2 RELATED WORK

2.1 Complementary Item Recommendation

Existing approaches to complementary item recommendation can be categorized into two types. The first type of approach addresses the pairwise similarities between fashion items [8, 11, 13]. Veit et al. [13] proposed a Siamese network to learn item compatibility using co-purchase or co-view information. Instead of computing the similarity in a single space, several recent approaches [8, 11, 12] explored learning sub-space embeddings to capture different notions of similarities. Vasileva et al. [12] learned type-aware embeddings to compute similarities in sub-spaces. Tan et al. [11] further improved the performance of [12] by learning shared sub-spaces as well as the importance of each sub-space. Lin et al. [8] designed a scalable sub-space attention approach for large-scale retrieval. The pairwise models [8, 11, 12] can be also used to evaluate the compatibility of an entire outfit. The second type of approach models the higher-order relations of items in an outfit. Han et al. [3] modeled an outfit as as sequence of items and used a bidirectional LSTM model to compute the compatibility score. Kipf et al. [6] used graph convolutional networks (GCN) to model item relationship in an outfit. All of these methods consider a generic compatibility model and don't take user preferences into account to personalize compatibility. In contrast, our method utilizes a two stage approach where we first a learn customer representation based on millions of customer feedbacks on individual items. We then use these customer representations along with customer preferences on pairs of items to learn a personalized metric of fashion compatibility for each customer. At a customer level, preferences on pairs of items are relatively fewer as compared to preferences on individual items (On an average, we have 5 times as many preferences on individual items than on pairs of items, as the ratings for individual items are easier to collect). Therefore, single item preferences help in reliably learning customer representation and with the learnt customer representations it becomes possible to learn a personalized compatibility metric with relatively fewer pairwise preferences.

2.2 Personalized Complementary Item Recommendation

Various personalized complementary methods have been proposed to incorporate personalized information into the complementary model. McAuley et al. [10] learned a personalized distance function (a weighting matrix) to measure the pair-wise item distances for a user. The matrix is learned using co-purchase data, which is often noisy. Some recent work learned personalized complementary models at the outfit-level [2, 9, 15]. Lu et al. [9] learned binary codes for each item type and user embeddings. The outfit score for a user is computed based on the compatibility score between each pair of fashion items in an outfit and the user's preference for each fashion item (i.e., distance between user embedding and a fashion item). Zheng et al. [15] utilized user information (e.g., selfie posts) from social-media to recommend outfits. Chen et al. [2] collected a large-scale user clicked set of outfits, and used user clicked items as user preference signals to generate outfits. These methods often use random negative images (or outfits) when learning the embeddings. However, these negative images may not be the "true" negatives to the anchor images (outfits). In practice, good negative images are critical to achieve good performance. Therefore, we collect user ratings on pairs of items (Figure 1b) to obtain positive and negative image pairs for learning embeddings.

In this work, we consider the complementary relationship at the pairwise level, due to its simplicity and scalability to different configurations of product types. It is common to have a different number of categories in an outfit rather than a fixed number of categories [4]. A pairwise model is more suitable in this case, and it can be easily extended to outfit-level similar to [8, 11, 12], but we leave that extension to future work.

3 PROPOSED APPROACH

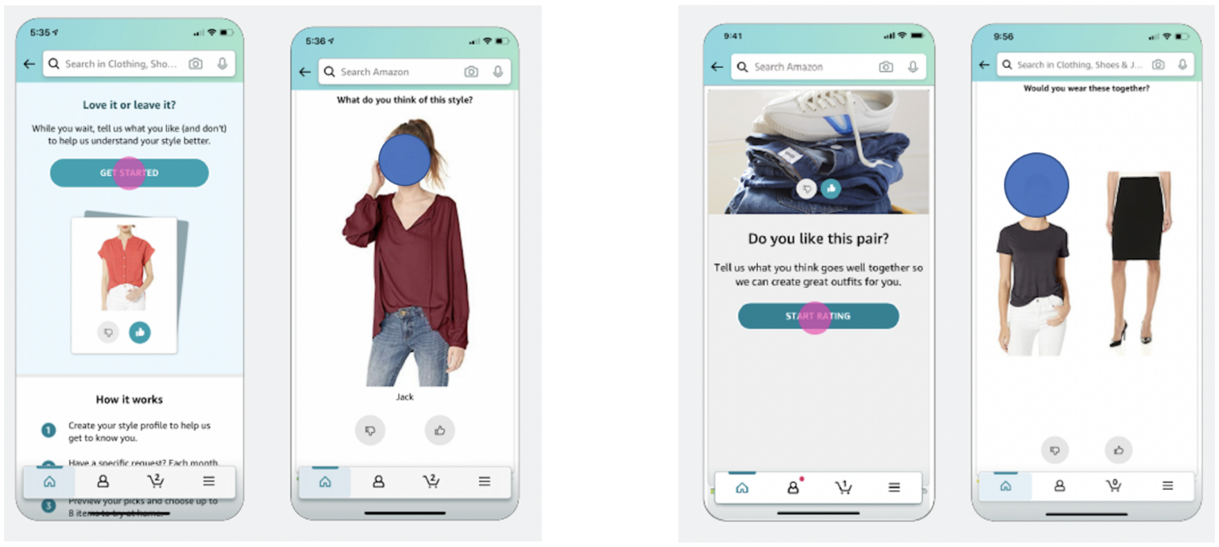
Figure 2 illustrates the system overview of our framework. Our framework has three inputs: images of top and bottom items forming a pair, the customer's provided rating on the pair, and the customer's embedding. The input images are passed through a CNN to extract image feature vectors. Similar to prior work [8] [12], a set of masks are applied to the image feature vector to generate multiple subspace embeddings. They correspond to different notions of compatibility that are dependent on the customer. The customer embedding is fed into a sub-network to predict the attention weights, which in turn select the appropriate subspace embeddings for the final embedding computation. The selected subspace embeddings for a particular customer would encode the compatibility relationship that is personalized to that customer. The details of the subspace attention network are presented in a later section.

Our network is trained using contrastive loss on pairs of items. It takes in a training sample $\{\langle \mathbf{A_i}, \mathbf{B_i}, c, r_\mathbf{i} \rangle | i \in [n]\}$, where r_i is the customer c's rating on compatibility of items corresponding to the pair $\langle \mathbf{A_i}, \mathbf{B_i} \rangle$ of images. The loss of the i^{th} data point is calculated using the positive distance $d(\mathbf{A_i}, \mathbf{B_i})$ and the label r_i. Details of the contrastive loss are given in a later section.

3.1 Customer embeddings

We learn customer representation using a class of algorithms called matrix completion. We experiment with different algorithms including Partial Dual (a matrix completion algorithm) and a neural network to learn customer style (Stynet) (Figure 3). We use customer preferences on individual items (upvotes/down votes on *"Love it or Leave it"*), item features (visual features, brand, color etc.) and other customer side information as input to these algorithms. These embeddings are used to compute attention weights in the personalized compatibility model (Figure 2).

3.1.1 Partial Dual. This is a class of matrix completion algorithm where a low rank representation $\mathbf{W}$ of the rating matrix $\mathbf{Y}$ is learnt. $\mathbf{W}$ can be factorized into customer and item embedding matrix. The customer embedding matrix is used to compute subspace attention weights in PSA-Net. $\mathbf{Y} \in \mathbb{R}^{d \times T}$ is the rating matrix on individual items and $L : \mathbb{R}^{d \times T} \times \mathbb{R}^{d \times T} \rightarrow \mathbb{R}$ is a convex loss function, $\|\|_*^2$ is a nuclear norm regularizer, $\lambda > 0$ is a cost parameter and D is a set of constraints. The W matrix is a low rank matrix that is learnt by solving the following optimization problem as described in [5]:



Love it or leave it?

While you wait, tell us what you like (and don't) to help us understand your style better.

How it works

1 Create your style profile to help us get to know you.

Have a specific request? Each month

3 Preview your picks and choose up to 8 items to try at home.

What do you think of this style?

Jack

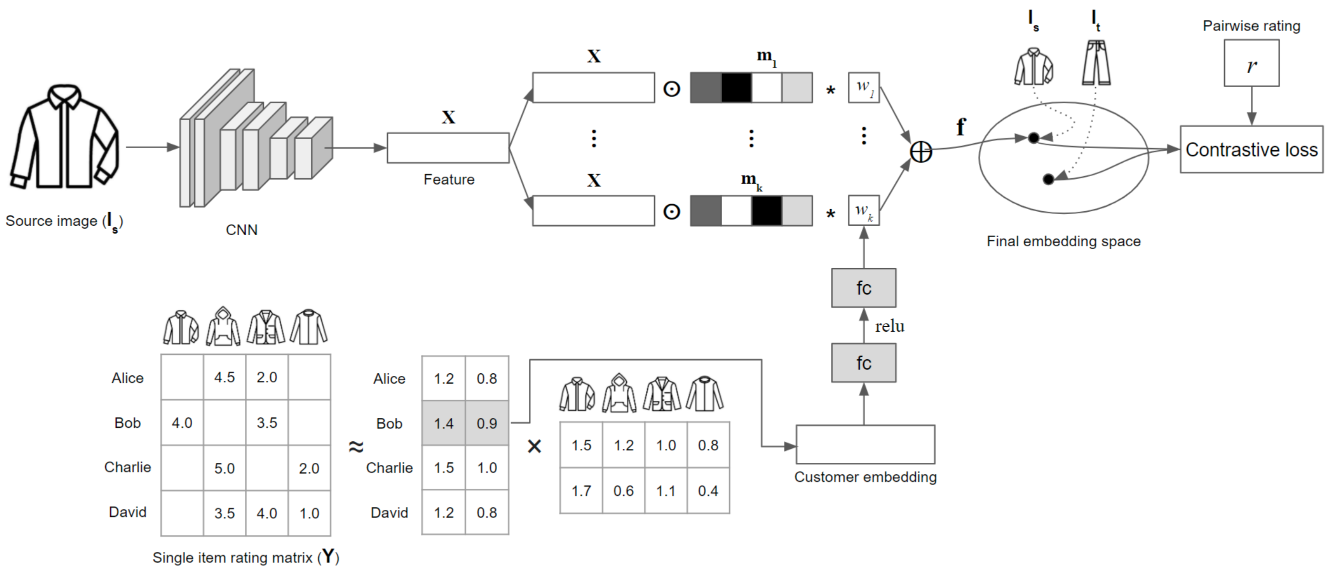
Do you like this pair?

Tell us what you think goes well together so we can create great outfits for you.

Would you wear these together?

(a) Item level feedback **(b) Feedback on pairs**

Figure 1: Love it or leave it



Alice		4.5	2.0	
Bob	4.0		3.5	
Charlie		5.0		2.0
David		3.5	4.0	1.0

Alice	1.2	0.8
Bob	1.4	0.9
Charlie	1.5	1.0
David	1.2	0.8

1.5	1.2	1.0	0.8
1.7	0.6	1.1	0.4

Figure 2: PSA-Net: An overview of our proposed approach

$$\begin{aligned}&\min_{\mathbf{W}\in\mathbb{R}^{d\times T}} \lambda L(\mathbf{Y},\mathbf{W}) + \|\mathbf{W}\|_*^2,\\ &\text{subject to } \mathbf{W}\in D\end{aligned} \tag{1}$$

W can be factorized into customer and item embedding matrix denoted by **E** and **M** respectively. Emebedding of a customer c is given by $\mathbf{E}_c$. The customer embeddings learnt using this method are then used to learn attention weights for compatibility subspaces for the personalized compatibility model. The loss function used is mean squared loss between the predicted ratings matrix **W** and actual rating matrix **Y** and is optimized using Riemannian conjugate gradient and trust-region algorithms.

3.1.2 Stynet. StyNet is a deep neural-network based recommendation model motivated from [14] which is able to generalize to both cold-start customers and items by learning the interactions between customer and item features in a latent embedding space. In this architecture, customer embeddings and item embeddings are learnt through customer network and an item network respectively. Customer Id, Customer survey features and customer's past preferences on the *"Love it or leave it"* serve as the input layer to the customer neural net while item catalog features like itemId, color, category, brand and item visual embeddings (learned separately by a convolutional neural network) serve as the input layer to the item neural net. Two embeddings are concatenated and a sigmoid function is applied to the concatenated embedding to transform it into the predicted rating of an item. The network is optimized in an end to end fashion using a mean squared loss. Architecture details are shown in figure 3b.

3.2 Subspace attention network

In this section, we describe our personalized subspace attention network (PSA-Net). Instead of computing similarity in a single space, we utilize style subspaces [8, 11, 12] to capture multiple dimensions of compatibility. This is important in learning a personalized notion of compatibility for each customer.

The network learns a non-linear function $\psi(\mathbf{I_s}, c)$ that takes a source image $\mathbf{I_s}$ and a customer embedding $\mathbf{E}_c$ as inputs and generates a feature embedding $\mathbf{f}$. The image is first passed through a CNN to extract its visual feature vector (denoted as $\mathbf{X}$). The customer embedding is used to predict the subspace attention weights $(w_1, ..., w_k)$ (k is the number of subspaces) using a sub-network. Then, a set of learnable masks $(\mathbf{m}_1, ..., \mathbf{m}_k)$ that have the same dimensionality as the image feature vector are applied to the image feature vector via Hadamard product. These masks are learned so as to project the features into a set of subspaces that encode different compatibility substructures. The final embedding constructed by the network is a weighted sum of the subspace embeddings:

$$\mathbf{f} = \sum_{i=1}^{k} w_i * (\mathbf{X} \odot \mathbf{m}_i), \tag{2}$$

where k is the number of subspaces, $\mathbf{X}$ is the image feature vector after the base CNN, $\mathbf{m}_i$ are learnable masks, w_i are attention weights, and $\mathbf{f}$ is the final embedding.

3.3 Ranking Losses

We define $\mathbf{I_s}$ as image from the source category (Tops in our case) and $\mathbf{I_t}$ as image from target category (Bottoms in our case). Given the embeddings $\mathbf{f}_s = \psi(\mathbf{I_s}, c)$ and $\mathbf{f}_t = \psi(\mathbf{I_t}, c)$ for the image pair $\langle \mathbf{I}_s, \mathbf{I}_t \rangle$, and the customer c's rating r on the pair, we evaluate the neural network's loss. We consider two loss functions, as discussed below.

3.3.1 Triplet loss. Since we collect data on pairs, we need to sample a third item from the target category. If the rating r is positive for a pair then we would randomly sample a negative item from list of items from target category and assume that randomly sampled item would be less compatible than the item from target category (bottom in this case) in the pair. Likewise, if the rating r is negative for a pair then we would randomly sample a positive item from list of items from target categories and assume that randomly sampled item would be more compatible than the bottom in the pair.

This loss requires a triplet $\langle \mathbf{f}_a, \mathbf{f}_p, \mathbf{f}_n \rangle$ of embeddings. It is defined as:

$$\ell = \max(||\mathbf{f}_a - \mathbf{f}_p|| - ||\mathbf{f}_a - \mathbf{f}_n|| + m, 0), \tag{3}$$

where a is anchor image, p is positive image and n is negative image.

3.3.2 Contrastive loss. We noticed that sampling of item to create a triple added noise (positive and negative images are randomly sampled) to the training process. We experimented with contrastive loss which can be computed directly for the pair. We found that contrastive loss outperformed the triplet loss. See section 7.2.2 for more details. Contrastive loss is defined as:

$$\ell = \mathbb{I}_{r=+1} \cdot ||\, \mathbf{f}_s - \mathbf{f}_t||^2 + \mathbb{I}_{r=-1} \cdot \max\left(0, m - ||\mathbf{f}_s - \mathbf{f}_t||\right)^2, \tag{4}$$

where $\mathbb{I}_{r=+1}$ is an indicator function for a positive label, which takes the value 1 if the label for a pair is positive else 0. Likewise, $\mathbb{I}_{r=-1}$ is an indicator function for a negative label, which takes the value 1 if the label for a pair is negative else 0.

4 DATA

While we have items from multiple categories in Amazon catalogue, we consider only Tops (Shirts, Sweaters) and Bottoms (Pants, Jeans, Skirts, Shorts) categories for this paper.

4.1 Labelled data from human stylists

We use an existing dataset of human stylists curated outfits within Amazon ($\sim 70K$). Since our task is to learn pairwise compatibility, we produce pairs from the curated outfits dataset. We cannot use rejected outfits to generate negative pairs since some of those pairs can still be compatible even when the entire outfit is not. We use this data to train a pairwise non personalized compatibility model based on the state of the art approach in [8]. This baseline model is compared against a model trained on customer pairwise preference data from *Love it or leave it.*

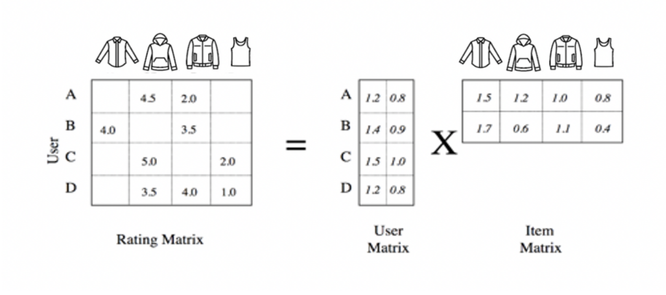
4.2 Personalized ratings obtained from customers

4.2.1 Preferences on pairs. Lack of negatively labelled data and the variance in compatibility labels provided by human stylists encouraged us to collect pairwise compatibility data directly from customers to learn a compatibility metric personalized to them. We get customer feedback on pairs of items through the *Love it or Leave it* experience where a customer upvotes a pair if they think they can wear the two items together and downvote it otherwise (see Figure 1a). We get data on pairs from each customer (~100 on average) and ~500K pairs across all customers. This data is used to train the personalized compatibility model.

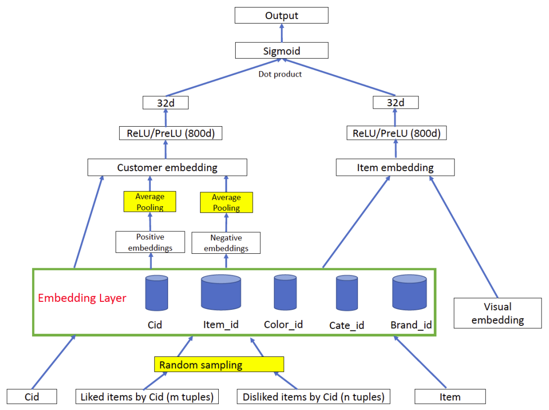
4.2.2 Preferences on individual items. We collect data on customer style preferences on individual items through the *Love it or Leave it* experience (see Figure 1b). A customer rates approximately 200 items to let us know their style preferences. We use this data to learn customer embeddings that are used in the personalized compatibility model to learn subspace attention weights.

5 PREPROCESSING

Item images sometimes show the model wearing pieces of clothing other than the item of interest, which might add noise to the training. To correct for this, images are cropped before feeding into our models. A YOLO3 object detection model trained on Amazon



(a) Matrix completion



(b) Stynet

Figure 3: Customer representation learning based on single item ratings

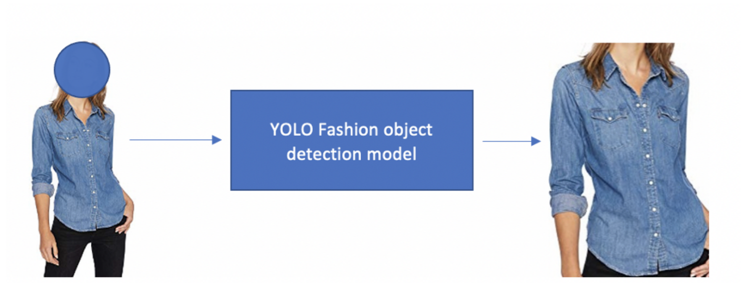


Figure 4: Cropping using YOLO3 fashion object detection model

fashion images that identifies the bounding boxes of all the entities present in the image is used to extract the part of the image which represents the item, as shown in (Figure 4).

6 EVALUATION

We evaluate our model by computing different offline metrics like customer level AUC, precision@K and recall@K and compare it with three baseline models, two of them do not take into account the customer embeddings(one trained on fashion expert labelled data and another trained on customer labelled data but which doesn't used customer embeddings to personalize compatibility) and the third one uses customer embeddings but still learns a single compatibility embedding sub-space for all customers. Since most literature on compatibility either does not take personal tastes of the customer into account or collect dataset in a way that is different than ours, there is no external baseline to compare our result against. Works like [4] and [2] include personalization but they work on data-sets consisting of entire outfits and hence are not directly compare-able.

In a dataset of n ratings on pairs of items the i^{th} observation can be represented as $\langle \boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{c}_i, r_i \rangle$ where $\boldsymbol{c}_i$ is a customer, $\langle \boldsymbol{A}_i, \boldsymbol{B}_i \rangle$ constitute the pair being queried and $r_i \in \{+1, -1\}$ is the label obtained from customer's interaction. Let d_i be the distance between the embeddings of $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ as obtained from the model, when using embeddings of $\boldsymbol{c}_m$. Without loss of generality, assume $d_1 \leq d_2 \leq \ldots \leq d_n$. Hence, the ratings are arranged according to the model's belief about their compatibility (for that specific customer). Let $\textsc{ratings}(\boldsymbol{c})$ be the set of ratings by customer $\boldsymbol{c}$.

6.1 Overall metrics

For overall metrics, we treat each rating as an independent point of data and calculate the metric directly. We consider the following metrics:

(1) **Recall**: Instead of reporting $\textsc{Recall}@k$, we report $\textsc{Recall}@x\%$ which is normalized for the number of ratings n. It is the same as $\textsc{Recall}@k$ where $k = \frac{xn}{100}$.
(2) **Precision**
(3) **Area under ROC (ROC AUC)**: Calculated by plotting TPR vs FPR and calculating the area under this curve[1].
(4) **Area under PR curve (AUPRC)**: Calculated by plotting precision against recall and calculating area the under it[1].

6.2 Customer level metrics

Likewise, we can limit our set of ratings to those of a specific customer $\boldsymbol{c}$, and calculate these metrics in the same way. Once we have these metrics for each customer, we can simply average over

[1]We normalize area under curve to 100%. So an area of 1.0 will be reported as 100%.

customers to get a single metric that is indicative of the model's overall performance. This means that performance of the model on each customer's data contributes equally to the final metric unlike the overall metrics.

If C is the set of all customers, the customer-average of a metric m is defined as:

$$\bar{m} = \frac{\sum_{c \in C} m(\textsc{ratings}(c))}{|C|}$$

7 EXPERIMENTS

7.1 Comparision with baseline models

7.1.1 Comparison with non-personalized model on customer ratings on pairs of items. For this, we consider the *non-personalized model* wherein the attention weights w_i are directly tunable parameters of our model which do not depend on the customer embeddings. This is equivalent to discarding the customer information associated with each rating item. This model achieves a customer-averaged ROC AUC which is 3.12% less than that of the *personalized model* which uses customer embeddings. The impact on other performance metrics is shown in Table 1. This confirms our hypothesis that leveraging the customer embeddings to predict pairwise compatibility leads to improved offline metrics in compatibility prediction tasks for customers as compared to a model where we do not leverage them on the same dataset.

7.1.2 Comparison with non-personalized model on fashion expert ratings on pairs of items. We also tested the impact of the source of training data on testing performance. We compared the performance of a *non-personalized model* trained using data from *fashion expert labelled data* to PSA-Net, which uses ratings from customers. Using customer provided labels improves the offline recommendation performance metrics over the model trained from labels provided by fashion experts. Specifically, we see an improvement of 18.26% in customer level ROC AUC. This establishes that compatibility data collected directly from customers is more useful in compatibility prediction tasks compared to data collected from fashion experts in training models.

7.1.3 Comparison with personalized model on customer ratings on pairs of items, without subspace attention. To test the impact of the subspace attention network, we compare the performance of PSANet against one which does not utilize subspace attention and hence can only learn a single notion of compatibility. Removing subspace attention lowers the offline recommendation performance metrics of the model. Specifically, we see a 5.68% reduction in customer level ROC AUC. This establishes the importance of the subspace attention framework and multiple notions of compatibility. We also compared the performance when using different number of subspaces and found that using 3 subspaces is optimal.

7.2 Ablation studies

The model described in Section 3 has several components which can be tweaked. We discuss the effect of changing individual components, while keeping the rest same.

7.2.1 Effect of source of customer embeddings. Customer embeddings are learnt using customer ratings on individual items using two different methods: Partial Dual and Stynet as described in 3.1. We experiment with customer representations learnt from both these methods in computing attention weights of different subspaces. We found that using embeddings learnt from Partial Dual(a state of the art matrix completion algorithm) leads to better performance in predicting personalized compatibility. Using Stynet embeddings instead of Partial Dual embeddings reduced the model's customer level ROC AUC by 5.86%.

7.2.2 Effect of loss function. Contrastive loss works by moving the embeddings of complementary items closer together and those of non-complementary items farther apart. Alternatively, we can use triplet loss for training the model. For calculating triplet loss, we need a triplet of images. Since we have ratings only on pairs of images, the third image is sampled uniformly at random from all the images to generate a triplet. The customer-averaged ROC AUC of the model when using triplet loss in this manner was found to be 19.06% lower than when using contrastive loss. This large difference can be accounted to the fact that triplet loss requires sampling, which introduces noise to the metric.

7.2.3 Effect of masking operation. The masking operation refers to calculating the final embedding by combining different subspace embeddings taking into account attention weights. In addition to the Hadamard masking operation described in a previous section, we also tried using a fully connected layer to calculate the contribution of each subspace embedding. However, that reduced the model's customer level ROC AUC by 3.68%.

8 FUTURE WORK

- Learn customer embeddings end to end while training and not using pre-trained customer embeddings from matrix completion
- Extend the approach to handle multiple categories of items together
- Include features that takes into account popularity of individual items. We have noticed in our internal customer surveys that the decision of a customer on a pair of items is also influenced by their preferences for individual items in the pair.

9 ACKNOWLEDGEMENT

We would like to thank our colleagues - John Park, Andrew Zoerb, Narayanan Sadagopan, Girish Subramanian, Hemant Balakrishnan, Samantha Dubow, Antonio Kirson, Kailash Gopalan, Yuan Zhang, Prafful Agrawal for useful discussion and support.

	Customer level(Δs)				**Overall**(Δs)			
	ROC AUC	**AUPRC**	**Recall@50%**	**Precision@10**	**ROC AUC**	**AUPRC**	**Recall@50%**	**Precision@10**
Baseline 1: **Fashion expert data + non-personalized model** + 3 sub-spaces	-18.27	-15.11	-15	-16.87	-21.35	-22.28	-18.76	-20
Baseline 2: Customer data + **non-personalized model** + 3 sub-spaces	-3.12	-3.65	-1.85	-4.07	-4.15	-6.39	-3.26	-30
Baseline 3: Customer data + personalized model + **1 sub-space**	-5.68	-6.25	-3.25	-7.91	-6.4	-11.39	-3.95	-30
PSA-Net: Customer data + personalized model + 3 sub-spaces	0	0	0	0	0	0	0	0

Table 1: Comparison of performance of PSA-Net against different baselines. All reported numbers are relative to PSA-Net.

REFERENCES

[1] 2021. Personal Shopper. https://www.amazon.com/b?node=19190471011.

[2] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion. In *KDD*.

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

[4] Tong He and Yang Hu. 2018. FashionNet: Personalized Outfit Recommendation with Deep Neural Network.

[5] Pratik Jawanpuria and Bamdev Mishra. 2018. A Unified Framework for Structured Low-rank Matrix Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 2254–2263.

[6] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification With Graph Convolution Networks. In *ICLR*.

[7] Kedan Li, Chen Liu, Ranjitha Kumar, and David A. Forsyth. 2019. Coherent and Controllable Outfit Generation. *CoRR* (2019).

[8] Yen-Liang Lin, Son Tran, and Larry S. Davis. 2020. Fashion Outfit Complementary Item Retrieval. In *CVPR*.

[9] Zhi Lu, Yang Hu, Yunchao Jiang, Yan Chen, and Bing Zeng. 2019. Learning Binary Code for Personalized Fashion Recommendation. In *CVPR)*.

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

[11] Reuben Tan, Mariya I. Vasileva, Kate Saenko, and Bryan A. Plummer. 2019. Learning Similarity Conditions Without Explicit Supervision. In *ICCV*.

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

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

[14] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System. *Comput. Surveys* 52, 1 (Feb 2019), 1–38. https://doi.org/10.1145/3285029

[15] Haitian Zheng, Kefei Wu, Jong-Hwi Park, Wei Zhu, and Jiebo Luo. 2020. Personalized Fashion Recommendation from Personal Social Media Data: An Item-to-Set Metric Learning Approach. In *ACM MM*.