Towards a Unified Framework for Visual Compatibility Prediction

Anirudh Singhal*†2, Ayush Chopra*‡1, Kumar Ayush*§4, Utkarsh Patel*†3, and Balaji Krishnamurthy[1]

[1]Media and Data Science Research Lab, Adobe
[2]IIT Bombay
[3]IIT Madras
[4]Stanford University

Abstract

Visual compatibility prediction refers to the task of determining if a set of items go well together. Existing techniques for compatibility prediction prioritize sensitivity to type or context in item representations and evaluate using a fill-in-the-blank (FITB) task. We scale the FITB task to stress-test existing methods which highlights the need for a compatibility prediction framework that is sensitive to multiple modalities of item relationships. In this work, we introduce a unified framework for compatibility learning that is jointly conditioned on the type, context, and style. The framework is composed of TC-GAE, a graph-based network that models type & context; SAE, an autoencoder that models style; and a reinforcement-learning based search technique that incorporates these modalities to learn a unified compatibility measure. We conduct experiments on two standard datasets and significantly outperform existing state-of-the-art methods. We also present qualitative analysis and discussions to study the impact of components of the proposed framework.

1. Introduction

The 2018 Fashion United [35] analysis values the global fashion industry at $3 trillion, an approximate 2% of the world GDP. A recent report by Shopify [31] (2018) estimates revenue for the apparel and clothing segment itself to rise by $257.8 billion over the next 2 years [28]. The State of Fashion report by McKinsey [24] in 2018 emphasizes the critical role of intelligent systems in driving this growth.

Consequently, several recent efforts have been directed towards providing smart, intuitive experiences for fashion commerce such as visually similar retrieval [7, 12, 40, 20] , fine-grained product tagging [1, 21], virtual try-on [14, 38, 10, 4, 3] and compatible recommendations [22, 8, 36, 13, 34, 2]. The problem of predicting fashion compatibility refers to determining if a set of items go well together. The problem is particularly challenging due to the complex interplay of human creativity, style expertise, and self-expression involved in the process of transforming a collection of seemingly disjoint items into a cohesive concept.

Initial methods such as [22, 37] learn item representations in independence and then perform pairwise comparisons between items to predict compatibility. Recent state-of-the-art methods focus on incorporating item interactions and differ with respect to their primary modality of focus. [36] focuses on type conditioning in pairwise relationships. In contrast, methods such as [13, 8, 9] prioritize item context, defined by the set of neighboring items it is known to be compatible with when learning representations for compatibility prediction. In consistence with real-world applications, existing methods report performance using a fill-in-the-blank(FITB) evaluation task. The FITB task consists of test questions with an incomplete (partial) outfit and set of 4 candidate items as input with the objective to pick the next-best item recommendation. In contrast, real-world recommender systems may routinely require querying for next-best items over a larger set of candidate choices.

To stress-test the corresponding robustness of existing techniques, we scale the FITB task by linearly increasing the number of candidate choices. Our observations necessitate the requirement for a more efficient compatibility framework. We posit that the sensitivity of learned representations to multiple aspects of item relationships can help achieve improved compatibility prediction. In this work, we present a unified framework for learning fashion compatibility that is jointly conditioned to type, context and style when learning item representations. Our contributions can be summarized as follows:

- We scale the FITB task for compatibility prediction.

*equal contribution
†work done as part of Adobe MDSR internship program
‡corresponding author (ayuchopr@adobe.com)
§work done while at Adobe

Stress testing existing methods using this variant motivates the need for a robust compatibility framework.

- We introduce a type-conditioned graph autoencoder (TC-GAE) to learn a compatibility measure conditioned on type and context.
- We introduce an attentive autoencoder (SAE) that learns a style measure of compatibility through an outfit level style representation.
- We present a reinforcement learning-based search strategy that integrates these multiple modalities: type, context, and style, to learn a unified measure of compatibility.

The related work is presented in Section 2 and task preliminaries are summarized in Section 3. The scaled FITB evaluation in Section 4 is used to motivate the methodology which is detailed in Section 5. The experiments and results are presented in Section 6 and 7 respectively. Extensive comparisons with existing methods on standard datasets highlight the superior performance of the proposed method.

2. Related Work

The present work aligns with existing literature in fashion compatibility prediction and style extraction using images. Our work also finds similarity with search-based techniques for learning composite transformation functions.

Visual Fashion Compatibility Prediction To approach the task of compatibility prediction, [23] learn a compatibility metric on top of CNN extracted visual features, and apply their method to pairs of products such that the learned distance in the embedding space is interpreted as compatibility. Their approach is improved by Veit et al. [37], who instead of using pre-computed features for the images, use an end-to-end siamese network to predict compatibility between pairs of images. A similar end-to-end approach [17] shows that jointly learning the feature extractor and the recommender system leads to better results. Recent state of the art methods seeks to incorporate item-item relationships for compatibility learning. [13] attempts to learn item representations by focusing on item context, other items that it appears alongside in outfits. To achieve the same, they consider a fashion outfit to be an ordered sequence of products using a bidirectional LSTM on top of the CNN-extracted features from the images and semantic information extracted from text in the embedding space. This method was improved by adding a new style embedding for the full outfit [26]. Vasileva et al. [36] also use textual information to improve the product embeddings, along with using conditional similarity networks [37] to produce type conditioned embeddings and learn a metric for compatibility. To efficiently model item-item type relationships, this approach projects each product embedding to a new space, depending on the type of the item pairs being compared. However, outfits are often characterized by more complex relationships that may not be fully encapsulated by visualizing an outfit as an ordered sequence or a combination of pairs of items.

More recently, [9, 8] visualize outfits as an unordered sequence and utilize graph neural networks to efficiently encapsulate item context. [9] proposes to represent an outfit using a type-level graph where each node represents a type and each edge represents the interaction between two types. Accordingly, each outfit can be represented as a sub-graph by putting the items into their corresponding types of nodes. A node-wise graph neural network is introduced to model node interactions and learn node representations. [8] uses an item-level graph to represent clothing items and their pairwise compatibility relationships. In the graph, each vertex represents a clothing item and edges connect pairwise of items that are compatible.

In this work, we introduce TC-GAE, a type-conditioned graph autoencoder to jointly model context and type when learning compatibility.

Style Extraction In context of fashion, an outfit, when visualized as a whole, has its own style. The ability to effectively model the same can be a particularly valuable modality for recommendations. Preliminary attempts to incorporate style in fashion compatibility have been largely focused on using text data [13, 36] which is difficult to obtain and may not be a good representative of the visual style. Leveraging visual cues for style can yield more robust representations. Takegi et al. [33] introduced a supervised framework to encode visual style. However, the subjectivity of formulation along with the absence of labeled data makes a supervised approach to the problem particularly challenging. Consequently, Hsiao et al. [16] presented an unsupervised methodology for style extraction of outfits. Another recent unsupervised approach was introduced in [5] that leveraged an aspect extraction method to train an autoencoder for style representation.

To incorporate sensitivity to outfit style in learning compatibility, we introduce SAE, an attention-based style autoencoder network.

Learning Unified Representations To learn a unified measure of compatibility, we use an automated search technique to discover composite functions in order to integrate context, type and style modalities for compatibility measure prediction. We use deep reinforcement learning based search mechanisms that have been used for discovering neural optimization methods [6], neural activation functions [29] and neural architecture designs [41]. For example, [41] uses a recurrent neural network to generate the model descriptions of neural networks and train this RNN with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set. We follow a similar approach and use an RNN to generate functions (to be used for element-wise transformation of feature embeddings) and use reinforcement learning to train the RNN to maximize the accuracy of compatible item selection. We verify the effectiveness of our approach by conducting an empirical evaluation with the discovered transformation function.

3. Preliminaries

In this section, we introduce the datasets used and baselines we perform comparisons against. We conclude the section by detailing standard evaluation tasks used to compare performance of models for compatibility prediction.

Datasets We conduct experiments on two standard datasets, *Maryland Polyvore* and *UIUC Polyvore*. *Maryland Polyvore* dataset was introduced by Han et al [13]. It consists of 21,889 outfits containing 164,379 items, which is split into three non-overlapping sets - 17316 for training, 1,497 for validation and 3,076 for testing. The *UIUC Polyvore* dataset was introduced by Vasileva et al. [36]. The dataset is split into two versions - disjoint and non-disjoint. The disjoint version is more "difficult" as no garment appears in more than one of the train-val-test splits. In this work, we have only used the disjoint UIUC Polyvore dataset. It has a total of 32,140 outfits containing 175,485 items and is split into two non-overlapping sets, 16,995 for training, 15,145 for validation and testing. Both datasets include rich multi-modal information about fashion images such as text descriptions, associated tags, and type information. However, we use only the type-information. For UIUC Polyvore, we use the type taxonomy as defined in [36]. While Polyvore Maryland provides fine-grained categories, we use the 14 coarse types defined in [36].

Baselines We compare our approach with recent state-of-the-art methods including [13, 36, 8]. [13] focuses on the utilization of item context when learning representations and models an outfit as an ordered sequence using Bidirectional LSTMs. In contrast, [8] models an outfit as an unordered sequence and uses a graph neural network-based framework to incorporate item context. [36] uses conditional similarity networks to learn type conditioned item representations. We leverage implementations of [13], [8], and [36] which are available at [39], [11], and [25] respectively.

Evaluation Tasks To evaluate learned representations, [13] introduced the task of fill-in-the-blank (FITB) in fashion recommendation. In this task, given a set of fashion items and a blank, the aim is to find the most compatible item from the candidate set to fill in the blank. One FITB question is defined for each test outfit. Each question consists of a set of products that form a partial outfit, and a set of possible choices $c_1, ..., c_M$ that includes a correct answer and $M-1$ randomly chosen products, often from different categories. And the task is evaluated by measuring whether or not the correct item was selected from the list of choices. Vasileva et al. [36] proposed a *resampled FITB* evaluation where the incorrect options are sampled from the same category as the correct option. A sample FITB evaluation question is presented in Figure 1.

Figure 1. The Fill-in-the-blank Evaluation Task (taken from [8])

4. Scaled FITB Task

When evaluating using the FITB task, existing state-of-the-art methods report performance by freezing the number of candidate choices to 4, in each question, for both original and resampled strategies. In FITB original, where candidates are sampled randomly from all classes, few options may be trivially eliminated. Even when using the resampled strategy, the likelihood of encountering the hard negatives is low when searching over a small number of candidates. Additionally, real-world fashion recommendation applications would routinely require selection of the next-best item from a larger set of candidate choices.

To stress-test robustness of representations learnt by existing methods, we scale the FITB task by linearly increasing number of candidate choices. We evaluate all the baselines under consideration, [13] (Bi-LSTM, green curve), [36] (Type-Aware, orange curve) and [8] (Context-Aware, blue curve) by linearly scaling number of candidate choices from 4 to 10. Figure 2 and 3 present performance on the Maryland Polyvore dataset using these scaled variants of the original and resampled FITB tasks respectively. For future correspondence, we will call these configurations as *Scaled FITB-Original* and *Scaled FITB-Resampled*.

[8] (blue curve) achieves highest performance on both FITB configurations, which highlights the benefit of capturing item context when learning item representations. However, [8] shows the highest rate of fall with its performance decreasing by almost 50% on both evaluation configurations when the number of candidates increase from 4 to 10. The performance of [13] (green curve), which models an outfit as an ordered sequence to capture item context in representations, falls drastically when switching to resampled FITB evaluation along with increasing number of candidates. While [36] (orange curve), which focuses on type conditioning, does not achieve absolute best accuracy, it's performance is similar in both original and resampled FITB tasks with a relatively small rate of fall in accuracy as the number of candidates increase.

Our observations necessitate the need for a compatibility prediction framework that respects both item context and type in order to learn robust representations. Additionally, we posit that style can be a useful modality as well when learning representations for compatibility prediction.

In the next section, we describe our efforts towards a unified framework for compatibility learning that is sensitive to context, type, and style.

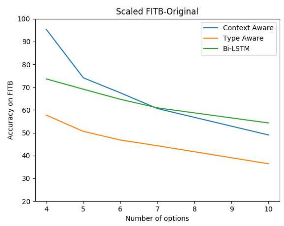

Figure 2. Performance of baseline methods on the Scaled FITB-Original task on Maryland Polyvore dataset.

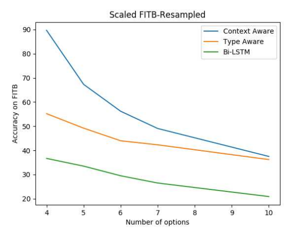

Figure 3. Performance of baseline methods on the Scaled FITB-Resampled task on Maryland Polyvore dataset.

5. Methodology

In this section, we detail our efforts towards a unified framework conditioned on type, context and style. First, we present TC-GAE, a type-conditioned graph autoencoder network to incorporate type and context. Next, we introduce, SAE, an attentive autoencoder to model style. Finally, we present a reinforcement learning-based search strategy to integrate measures from TC-GAE and SAE to output a final measure of compatibility.

5.1. Type Conditioned Graph Autoencoder

Building on [8], TC-GAE uses an item-level graph representation where vertices are catalog items and edges connect pairs of items that belong to the same outfit.

Let $G = (V, E)$ be an undirected graph with N nodes where $edge(i, j) \in E$ connect pairs of nodes $i, j \in V$. Each node in the graph is represented with a vector of features, $\vec{x_i} \in R^F$, and $X = \{\vec{x_0}, \vec{x_1}, ..., \vec{x_{N-1}}\}$ is a $R^{N\times F}$ matrix that contains features for all nodes. Each node i in the graph has category information c_i. Using the training metadata, we calculate the co-occurence frequency $Count_{c_i,c_j}$ of categories c_i and c_j. Then, the graph G is represented by a weighted adjacency matrix $A \in R^{N\times N}$, where $A_{i,j} = Count_{c_i,c_j}$, if an edge exists between nodes i and j, and $A_{i,j} = 0$ otherwise. The particular impact of using this co-occurence weighted adjacency matrix is discussed in Section 7.1.2.

In this framework, the encoder get as input an incomplete graph and produces embedding for each node. Then, the node embeddings are used by the decoder to predict missing edges in the graph. The implementation setup is presented in detail in the training paragraph. The network architecture is summarized in Figure 4.

Encoder For a node i in the graph, the encoder transforms its initial feature $\vec{x}_i$ into a latent representation $\vec{h}_i$. $\vec{x}_i$ contains information about the particular node item and is computed using technique discussed in Section 6. We want the encoded representation of each node, $\vec{h}_i$, to capture information not only about itself, but also about its ***context***, which is defined by its neighbors $\vec{N}_i$, where $\vec{N}_i = \{j \in V | A_{i,j} \neq 0\}$. Hence, the encoding is formulated as $\vec{h}_i = f_{enc}(\vec{x}_i, \vec{N}_i)$. The encoder f_{enc} is implemented as a deep Graph Convolutional Network [19] with multiple hidden layers. At layer $l + 1$, the hidden state H^{l+1} is represented as,

$$H^{(l+1)} = ReLU\left(\sum_{s=0}^{S} \tilde{\boldsymbol{A}}_s \boldsymbol{H}^{(l)} \boldsymbol{\Theta}_s^{(l)}\right)$$

where A is the type-co-occurence weighted adjacency matrix, $\tilde{A}_s$ is the normalized s-th step adjacency matrix (see [8]), S is the context depth and $\boldsymbol{\Theta}^{(l)}$ contains the trainable parameter for layer l. For our experiments, we set $S = 1$ to consider only immediate neighbours.

Decoder The decoder is trained to predict the probability of two nodes in the graph being connected. In TC-GAE, the decoder function is formulated to be ***type*** respecting when comparing two nodes (items). For two nodes i, j with latent representations $\vec{h}_i, \vec{h}_j$ and types c_i, c_j respectively, the edge probability p predicted by the decoder is defined as,

$$p = \sigma\left(\left|\vec{h}_i - \vec{h}_j\right| \vec{\omega}^T_{c_i,c_j} + b_{c_i,c_j}\right)$$

Here $|\cdot|$ is absolute value operator, and $\vec{\omega}_{c_i,c_j} \in \mathbb{R}^{F'}$ and $b_{c_i,c_j} \in \mathbb{R}$ are learnable parameters where F' is the dimension of the hidden state embedding. $\sigma(\cdot)$ is the sigmoid function that maps a scalar value to a valid probability $\in (0, 1)$.

Training The model is trained to predict compatibility among the products with A being the adjacency matrix for the graph of items. For training, after every N_{random} epochs, we randomly remove a subset of edges and randomly sample a set of negative edges E^- to generate an incomplete adjacency matrix A. The set of edges removed is denoted by E^+, as they represent positive edges, i.e., pairs

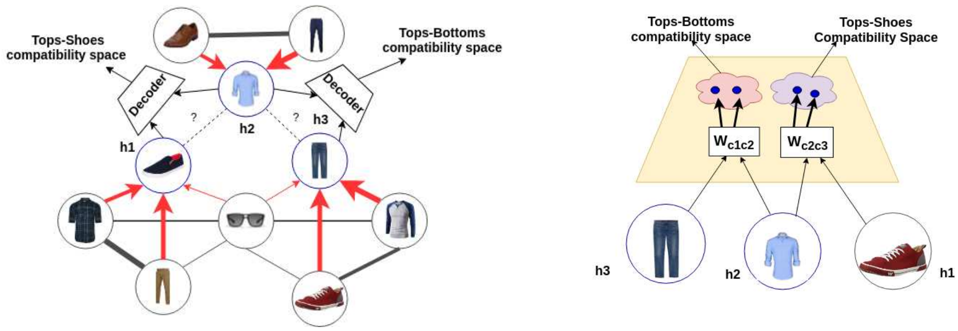

Figure 4. The left figure shows the TC-GAE architecture. In the figure, h_1, h_2, and h_3 are the node representations. The edges in TC-GAE are weighted and the width in the graph is proportional to weightage in the adjacency matrix. The directional red arrows depict message passing from neighbors in an item's context. The figure on right is the enlarged view of TC-GAE decoder.

of nodes (i, j) such that $A_{i,j} \neq 0$ and E^- is the set of negative edges which represent pairs of nodes (i, j) that are not connected, i.e., products that are not compatible. The model is trained to predict the edges $E_{\text{train}} = (E^+, E^-)$ that contains both positive and negative edges (we keep the classes balanced). Therefore, given the incomplete adjacency matrix A and the initial features for each node X, the decoder predicts the edges defined in E_{train}, and the model is optimized by minimizing the cross entropy loss between the predictions and their ground truth values, which is 1 for the edges in E^+ and 0 for the edges in E^-.

5.2. SAE: Attentive Style Autoencoder

Previous methods such as [13, 36] attempted to learn style representations through meta-data such as text descriptions, which are often noisy, incomplete and difficult to obtain. We introduce an attentive autoencoder to learn an outfit style representation using visual cues only. We leverage the TC-GAE decoder to attend over outfit items when learning the style representation. Next we detail our approach for learning outfit style which is similar to [27] with certain differences.

Consider an outfit $\mathbb{O}$ of with N_o number of items. First, for each item $i \in \mathbb{O}$, the latent node embedding $\vec{h}_i$ is transformed into a node style embedding, y_i, such that

$$y_i = W_s \cdot \vec{h}_i$$

where W_s is a learnable style transformation matrix. Next, the outfit style attention of each item, α_i, is computed as

$$\alpha_i = \frac{e^{d_i}}{\sum_{j \in \mathbb{O}} e^{d_j}}$$

where,

$$d_i = \frac{1}{N_o - 1} \sum_{j \in \mathbb{O} \setminus i} \sigma\left(\left|\vec{h}_i - \vec{h}_j\right| \vec{\omega}^T_{c_i,c_j} + b_{c_i,c_j}\right)$$

The style vector $z^{\mathbb{O}}$ for an outfit $\mathbb{O}$ is defined as,

$$z^{\mathbb{O}} = \sum_{i=1}^{N_o} \alpha_i y_i \tag{1}$$

where y_i is the item style embedding and α_i is outfit attention for item i.

Now, through the process of compressing and reconstructing the style vector, we aim to obtain a basis of styles observed in a variety of outfits. Assuming such a basis exists, then outfits can be represented as a linear combination of elements of the basis. For example, given a style basis that has two elements, casual and formal, outfits can be labeled as casual, formal, or their mixture.

We use $p \in \mathbb{R}^{\kappa}$ to denote the following mixture ratio

$$p^{\mathbb{O}} = \text{softmax}\left(W_z \cdot z^{\mathbb{O}} + b_z\right)$$

where W_z and b_z are the weight matrix and bias vector used to map a style vector to a mixture ratio. Since p is assumed to be a mixture ratio, the softmax function is applied so that each element of p is non-negative and the sum of the elements is 1. Also, κ represents the number of elements of the style basis.

Next, the style vector $z^{\mathbb{O}}$ for outfit $\mathbb{O}$ is reconstructed as

$$r^{\mathbb{O}} = W_p^T \cdot p^{\mathbb{O}}$$

where W_p is a style-embedding matrix and $r^{\mathbb{O}}$ is the reconstructed style vector. The training objective for the autoencoder is formulated as a reconstruction triplet loss and an orthoganalization loss which are defined as

$$L_R(r^o, z^o, z') = \max\left(0, m_r - d(r^o, z^o) + d\left(r^o, z'\right)\right)$$

$$L_O\left(W_p\right) = \left\|W_{pn} W_{pn}^T - I\right\|$$

$$L_{train} = L_R + L_O$$

where $d(r, z)$ is the cosine similarity of vector representations r, z ,W_{pn} is normalized W_p and I is the identity matrix. Here, z' is the style vector for a outfit different than outfit $\mathbb{O}$.

When evaluating style compatibility using FITB task with M options, $M+1$ style mixture ratios (vector of probabilities) are computed for the partial (input) outfit and for the M complete outfits formed by introducing each candidate option to the partial outfit. The compatibility score for each of the M candidates items is defined as inverse of the decrease in uncertainty of the outfit mixture ratio on adding the candidate item to the outfit.

5.3. Learning Unified Representations

Consider an FITB task with an input (partial) outfit of K elements and M candidate options. TC-GAE is used to predict a compatibility score of each of the M options by averaging pairwise decoder similarity of candidate with each item from the input (partial) outfit. Also, as described previously, the style autoencoder, SAE, also predicts a compatibility score. Next, in this section, we present a reinforcement learning based strategy to learn a unified measure of compatibility. Leveraging work in RL based search mechanisms [6, 29, 41], we learn a transformation function that weight scores from the two compatibility measures to output a final compatibility score.

As shown in Figure 5, the function is constructed by repeatedly composing the "core unit". A core unit first selects two operands ($op1$ and $op2$), then two unary functions ($u1$ and $u2$) to apply on the operands and finally a binary function b that combines the outputs of the two unary functions. The resulting $b(u1(op1), u2(op2))$ then becomes an operand that can be selected in the next group of predictions. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

Given the search space, the goal of the search algorithm is to find effective choices for the unary and binary functions. We use an RNN controller [41]. At each timestep, the controller predicts a single component of the function. The prediction is fed back to the controller in the next timestep, and this process is repeated until every component of the function is predicted. Once a candidate function has been generated by the search algorithm, the unified compatibility score is estimated and corresponding FITB accuracy is used as a reward signal to train the RNN Controller.

6. Experimental Setup

The TC-GAE and SAE networks are trained using an NVIDIA Titan-X GPU with 12 GB memory. The RNN controller used to learn the composite scoring function is trained on a CPU machine with 64 GB RAM.

We trained TC-GAE for 4,000 iterations (with early stopping), using Adam [18] optimizer, and a learning rate of 0.001. We experiment with input visual features (x_i) from two different models: a) ResNet-50 [15] pretrained on ImageNet [30] b) StyleNet [32]. For each item, the ResNet-50 model generates a visual feature of 2048-dim while StyleNet model generates a visual feature of 128-dim. The TC-GAE encoder model consists of 3 graph convolutional layers. For quantitative results reported in Section 7.1, the dimensions of the hidden states are [350, 350, 350] when using ResNet-50 features and [128, 128, 128] for StyleNet features. In Section 7.2, we present a discussion to study the impact of dimensionality of the hidden states.

When training the RNN controller, only 2000 outfits are used from the dataset. The operands, unary and binary functions accessible to the controller are presented below:

- Operands using the compatibility scores from TC-GAE and Attentive Style Autoencoder, x and y: x, y, $x+y$
- Unary functions: x, $-x$, x^2, $|x|$, x^3, $\sqrt{(|x|)}$, e^x, $\sin x$, $\cos x$, $\sinh x$, $\cosh x$, $\tanh x$, $erfx$, $\tan^{-1} x$, $\sigma(x)$, $\max(x, 0)$, $\min(x, 0)$, $\log_e(1 + e^x)$
- Binary functions: $x_1 + x_2$, $x_1 - x_2$, $x_1 * x_2$, $\max(x_1, x_2)$, $\min(x_1, x_2)$, $\sigma(x_1) * x_2$

7. Results and Discussions

In this section, we extensively evaluate the performance of the proposed framework. On the Maryland Polyvore dataset, we report performance comparisons against [8, 36, 13] and On the Polyvore UIUC-D dataset, we report comparisons against [8, 36]. For UIUC-D, comparisons are not reported with [13] since the dataset was not used in the original work.

First, we report comparative performance using the TC-GAE network. Next, we study the additional impact of incorporating style using SAE.

7.1. Type-Conditioned Graph Autoencoder

In this section, we evaluate the quantitative performance of TC-GAE on the Scaled FITB-Original and Scaled FITB-Resampled evaluations. We also present discussions to analyze the particular impact of various design constraints adopted when training TC-GAE.

7.1.1 Quantitative Results

For the scope of our analysis, we consider 4, 5, 6, 7 and 10 candidate options. We report performance for TC-GAE trained using both ResNet-50 features (reported as TC-GAE (R)) and StyleNet features (reported as TC-GAE (S)). While [8] originally used only ResNet-50 features, we additionally perform experiments using StyleNet [26] features. We first present results on the Scaled FITB-Original evaluation and then on the Scaled FITB-Resampled evaluation.

Scaled FITB Original In this configuration, the FITB candidate options are sampled randomly from different categories. Results on Maryland Polyvore and Polyvore UIUC-D datasets are reported in Table 1 and Table 2 respectively. On both datasets, the best performance is achieved

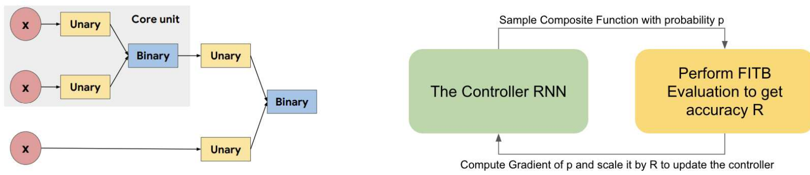

Figure 5. The left figure (taken from [29]) shows an example composite function structure. The function is composed of multiple repetitions of the "core unit", which consists of two inputs, two unary functions, and one binary function. Unary functions take in a single scalar input and return a single scalar output. Binary functions take in two scalar inputs and return a single scalar output. The figure in right shows an overview of the search mechanism.

by TC-GAE with an average improvement of 3% over the next best network. The gain increases with the number of candidate options which highlights improved robustness of the representations. For instance, on the UIUC-D dataset (Table 2), the gain increased from 3.13% with 5 candidate options to 5.70% with 10 candidates. In most cases, using ResNet-50 features results in better performance that StyleNet for both TC-GAE and [8].

Method	Number of Options				
	4	5	6	7	10
TC-GAE (R)	93.59	**78.57**	**69.47**	**64.04**	**53.31**
TC-GAE (S)	94.47	71.19	60.89	52.79	38.98
[8] (R)	95.32	76.49	67.49	60.6	49.02
[8] (S)	**96.7**	67.17	54.19	45.55	33.49
[36]	57.7	50.6	46.8	44.3	36.4
[13]	73.6	69.08	64.66	60.92	54.29

Table 1. Scaled FITB-Original on the Maryland Polyvore Dataset.

Method	Number of Options				
	4	5	6	7	10
TC-GAE (R)	**91.31**	**78.77**	**71.33**	**65.7**	**55.45**
TC-GAE (S)	88.15	62.85	52.38	45.90	33.57
[8] (R)	85.79	75.64	67.21	61.48	49.75
[8] (S)	89.58	60.77	49.94	42.43	30.33
[36]	55.8	50.9	46.8	43.6	36.09

Table 2. Scaled FITB-Original on Polyvore UIUC-D Dataset

Scaled FITB Resampled In this configuration, FITB options are sampled from the same category (type) as the correct solution. Results for Maryland Polyvore and Polyvore UIUC-D datasets are reported in Table 3 and Table 4 respectively. On both datasets, the best performance is obtained by TC-GAE with the improvement being more pronounced as the number of options increase. For instance, on Maryland dataset (Table 3) the gain for TC-GAE (R) over the next best model increases from 5.74% with 6 options to 7.24% with 10 options. Also, better gains in the resampled evaluation, when options are from same category, than in original evaluation further validates the hypothesis of jointly modeling type with context. Even here, in most cases, using ResNet-50 features results in better performance than StyleNet for both TC-GAE and [8].

Method	Number of)ptions				
	4	5	6	7	10
TC-GAE (R)	93.4	**74.18**	**61.92**	**56.43**	**44.74**
TC-GAE (S)	94.87	67.85	57.02	49.46	39.11
[8](R)	89.67	67.32	56.19	49.06	37.5
[8](S)	**96.74**	65.18	51.66	43.4	31.51
[36]	55.2	49.2	43.98	42.3	36.2
[13]	36.68	33.49	29.5	26.5	20.9

Table 3. Scaled FITB-Resampled on Maryland Polyvore dataset.

Method	Number of Options				
	4	5	6	7	10
TC-GAE (R)	**91.11**	**75.27**	**66.41**	**59.81**	**47.84**
TC-GAE (S)	86.88	63.49	52.15	45.18	33.57
[8] (R)	88.59	72.44	62.49	56.08	44.562
[8] (S)	89.92	62.49	49.67	42.43	30.33
[36]	57.5	51.8	47.6	43.95	36.1

Table 4. Scaled FITB-Resampled on Polyvore UIUC-D dataset

7.1.2 Discussion

Next, we present ablation studies to analyze the impact of a few TC-GAE design constraints namely the type-weighted adjacency matrix and the dimensionality of the encoder hidden state.

Impact of Weighted Adjacency Matrix When training TC-GAE encoder to capture item context in representations, we weight the adjacency matrix with the type co-occurrence counts for the items. To analyze the impact of this type conditioning, we contrast the performance with when TC-GAE is trained using a binary (unweighted) adjacency matrix. To further validate the impact of type-conditioning in the encoder, we also train the graph network in [8] with the weighted adjacency matrix. We conduct experiments on the Polyvore Maryland dataset. Results for Scaled FITB-Original evaluation presented in Table 5 indicate that using the type-weighted adjacency matrix results in better performance for both networks.

Impact of Encoder Hidden States Dimension The GCN encoder in the proposed TC-GAE is composed of three hid-

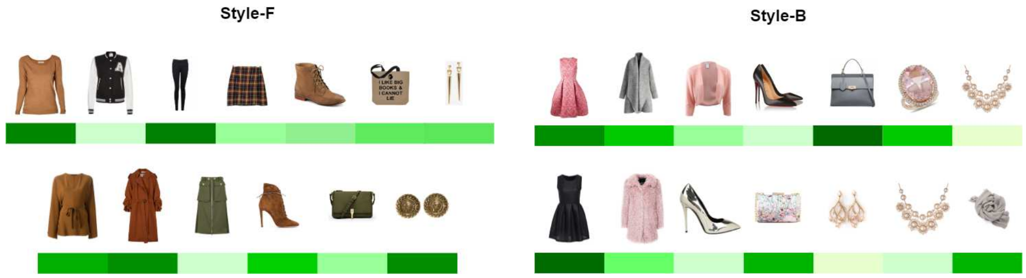

Figure 6. Qualitative Analysis of SAE: Sample outfits from two style clusters in the Maryland Polyvore dataset. The intensity of green for each item represents the attention weight of the item in determining the style of the outfit it belongs to.

Method	Number of Options				
	4	5	6	7	10
TC-GAE (UW)	91.35	77.08	68.10	62.52	53.09
TC-GAE (W)	**93.59**	**78.57**	**69.47**	**64.04**	**53.31**
[8] (UW)	95.32	76.49	**67.49**	60.6	**49.02**
[8] (W)	**95.74**	**77.6**	67.23	**61.9**	48.8

Table 5. Performance comparison on training TC-GAE and [8] with (W) and without (UW) the type-weighted adjacency matrix on the Maryland Polyvore dataset.

den layers. In this study, we analyze the impact of varying dimensionality of these hidden layers. Owing to the computational complexity associated with training graph neural networks, finding an optimal dimensionality of the hidden states can be useful. Table 6 summarize performance on Scaled FITB-Original evaluation task when training on Maryland Polyvore dataset using StyleNet features.

Size	Number of Options				
	4	5	6	7	10
128, 128, 128	94.87	67.85	57.02	49.46	39.11
128, 96, 80	94.22	67.59	56.69	49.32	38.84
96, 80, 64	93.33	66.27	55.57	48.6	38.51
80, 80, 80	92.41	66.79	55.37	47.87	39.2
48, 48, 48	84.57	63.33	52.25	46.56	36.37

Table 6. Effect of size of hidden states in TC-GAE encoder on Scaled FITB-Original evaluation (Maryland Polyvore Dataset).

7.2. Style Extraction

Following extensive analysis of the TC-GAE network, now we study the impact of SAE, the style extraction network. First, we qualitatively analyse the learned style representations. Next, we quantify the benefit of using style as an additional component of compatibility.

7.2.1 Qualitative Analysis

As an initial experiment, we trained SAE on the Maryland Polyvore dataset. To evaluate quality of the learned style representations, the learned mixture ratios for outfits in the dataset were used to bin the outfits into 6 distinct clusters (found empirically). Figure 6 includes sample outfits from two of the learned style clusters (Style F and Style B). The corresponding attention weights for each item in an outfit are presented via intensity of green color in their respective bars.

Method	Number of Options				
	4	5	6	7	10
TC-GAE	93.59	78.57	69.47	64.04	53.31
TC-GAE + SAE	**94.20**	**79.15**	**69.95**	**65.02**	**53.88**

Table 7. Incorporating style as a compatibility measure results in improved performance on the Scaled FITB evaluation task.

7.2.2 Learning Unified Representations

As presented in Section 5, compatibility measures are obtained independently using both TC-GAE and SAE. These are then unified using a learnt composite scoring function to obtain a final compatibility score. When training on the Maryland Polyvore dataset, the learnt compatibility measure for each item is defined as

$$\text{score} = e^{y} - \text{relu}(e^{(-|y-\sin(x)|)})$$

where, x is the TC-GAE item compatibility score and y is the style compatibility score for each item using SAE. Table 7 presents performance on the Scaled FITB-Original evaluation using the learnt scoring function. As highlighted by the results, using the unified measure conditioned on both TC-GAE and SAE results in improved performance even with increasing number of candidates .

8. Conclusion

In this study, we focus on the task of fashion compatibility learning. When learning item representations for compatibility prediction, existing methods prioritize sensitivity to item type or context. We stress-test these methods using a scaled FITB task. Following our observations, we introduce a framework to learn compatibility measures conditioned on type, context and style. Finally, we also present a reinforcement learning-based search strategy that integrates these modalities to learn a robust compatibility measure. Extensive empirical analysis highlights the superiority of the proposed technique over existing methods.

References

[1] K. E. Ak, A. A. Kassim, J. Hwee Lim, and J. Yew Tham. Learning attribute representations with localization for flexible fashion search. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[2] K. Ayush. Context aware recommendations embedded in augmented viewpoint to retarget consumers in v-commerce. 1

[3] K. Ayush, S. Jandial, A. Chopra, M. Hemani, and B. Krishnamurthy. Robust cloth warping via multi-scale patch adversarial loss for virtual try-on framework. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1

[4] K. Ayush, S. Jandial, A. Chopra, and B. Krishnamurthy. Powering virtual try-on via auxiliary human segmentation learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1

[5] P. Baldi. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11, pages 37–50. JMLR.org, 2011. 2

[6] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le. Neural optimizer search with reinforcement learning. *arXiv preprint arXiv:1709.07417*, 2017. 2, 6

[7] A. Chopra, A. Sinha, H. Gupta, M. Sarkar, K. Ayush, and B. Krishnamurthy. Powering robust fashion retrieval with information rich feature embeddings. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 1

[8] G. Cucurull, P. Taslakian, and D. Vázquez. Context-aware visual compatibility prediction. *CoRR*, abs/1902.03646, 2019. 1, 2, 3, 4, 6, 7, 8

[9] Z. Cui, Z. Li, S. Wu, X. Zhang, and L. Wang. Dressing as a whole: Outfit compatibility learning based on node-wise graph neural networks. *CoRR*, abs/1902.08009, 2019. 1, 2

[10] H. Dong, X. Liang, B. Wang, H. Lai, J. Zhu, and J. Yin. Towards multi-pose guided virtual try-on network. *ArXiv*, abs/1902.11026, 2019. 1

[11] gcucurull. Code for Context Aware Fashion Compatibility. https://github.com/gcucurull/visual-compatibility. 3

[12] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE international conference on computer vision*, pages 3343–3351, 2015. 1

[13] X. Han, Z. Wu, Y. Jiang, and L. S. Davis. Learning fashion compatibility with bidirectional lstms. *CoRR*, abs/1707.05691, 2017. 1, 2, 3, 5, 6, 7

[14] X. Han, Z. Wu, Z. Wu, R. Yu, and L. Davis. Viton: An image-based virtual try-on network. 11 2017. 1

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 6

[16] W.-L. Hsiao and K. Grauman. Creating capsule wardrobes from fashion images. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7161–7170, 2017. 2

[17] W. Kang, C. Fang, Z. Wang, and J. J. McAuley. Visually-aware fashion recommendation and design with generative image models. *CoRR*, abs/1711.02231, 2017. 2

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 6

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. 4

[20] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016. 1

[21] K. Mahajan, T. Khurana, A. Chopra, I. Gupta, C. Arora, and A. Rai. Pose aware fine-grained visual classification using pose experts. pages 2381–2385, 10 2018. 1

[22] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015. 1

[23] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. *CoRR*, abs/1506.04757, 2015. 2

[24] McKenzie, 2018. 1

[25] mvasil. Code for Type Aware Fashion Compatibility. https://github.com/mvasil/fashion-compatibility. 3

[26] T. Nakamura and R. Goto. Outfit generation and style extraction via bidirectional LSTM and autoencoder. *CoRR*, abs/1807.03133, 2018. 2, 6

[27] T. Nakamura and R. Goto. Outfit generation and style extraction via bidirectional lstm and autoencoder. *arXiv preprint arXiv:1807.03133*, 2018. 5

[28] PwC, 2018. 1

[29] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. 2018. 2, 6, 7

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 6

[31] Shopify, 2018. 1

[32] E. Simo-Serra and H. Ishikawa. Fashion style in 128 floats: Joint ranking and classification using weak data for feature extraction. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 298–307, 2016. 6

[33] M. Takagi, E. Simo-Serra, S. Iizuka, and H. Ishikawa. What makes a style: Experimental analysis of fashion prediction. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2247–2253, Oct 2017. 2

[34] K. Tanmay and K. Ayush. Augmented reality based recommendations based on perceptual shape style compatibility with objects in the viewpoint and color compatibility with the background. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1

[35] F. United, 2018. 1

[36] M. I. Vasileva, B. A. Plummer, K. Dusad, S. Rajpal, R. Kumar, and D. A. Forsyth. Learning type-aware embeddings for fashion compatibility. *CoRR*, abs/1803.09196, 2018. 1, 2, 3, 5, 6, 7

[37] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. J. Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. *CoRR*, abs/1509.07473, 2015. 1, 2

[38] B. Wang, H. Zheng, X. Liang, Y. Chen, and L. Lin. Toward characteristic-preserving image-based virtual try-on network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 589–604, 2018. 1

[39] Xthan. Code for BiDirectional LSTM for Fashion Compatibility. `https://github.com/xthan/polyvore`. 3

[40] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 814–823. IEEE, 2017. 1

[41] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2, 6