Transformer-based Graph Neural Networks for Outfit Generation

Federico Becattini, Federico Maria Teotini, and Alberto Del Bimbo

Abstract—Suggesting complementary clothing items to compose an outfit is a process of emerging interest, yet it involves a fine understanding of fashion trends and visual aesthetics. Previous works have mainly focused on recommendation by scoring visual appeal and representing garments as ordered sequences or as collections of pairwise-compatible items. This limits the full usage of relations among clothes. We attempt to bridge the gap between outfit recommendation and generation by leveraging a graph-based representation of items in a collection. The work carried out in this paper, tries to build a bridge between outfit recommendation and generation, by discovering new appealing outfits starting from a collection of pre-existing ones. We propose a transformer-based architecture, named TGNN, which exploits multi-headed self attention to capture relations between clothing items in a graph as a message passing step in Convolutional Graph Neural Networks. Specifically, starting from a seed, i.e. one or more garments, outfit generation is performed by iteratively choosing the garment that is most compatible with the previously chosen ones. Extensive experimentations are conducted with two different datasets, demonstrating the capability of the model to perform seeded outfit generation as well as obtaining state of the art results on compatibility estimation tasks.

Index Terms—Transformer, Graph Neural Networks, Outfit Generation

✦

1 INTRODUCTION

Fashion is one of the most important industries in the world, moving very large amounts of money with thousands of customers and field operators. Choosing garments to create a good looking outfit is a very important task in the fashion field, but it is a difficult one: it involves many complex concepts like style and visual composition expertise, creativity, cultural and social understanding, trends, etc., and they all need to be balanced to make sure that the resulting outfit is indeed aesthetically appealing.

Outfit creation is an ubiquitous task and, with the rapid growth of online fashion retailers and fashion related social networks (e.g. Instagram), it became a fundamental task, often carried out by entire company departments of fashion experts.

Every architecture needs to support two notions in order to put together a good outfit:

- The *similarity* notion, that is when two garments are similar to each other and possibly interchangeable.
- The *compatibility* notion, meaning that the fashion items composing the same outfit should aesthetically be compatible with each other.

Previous studies [1], [2] focused on learning compatibility metrics between pairwise items (Figure 1 **a**); these kind of architectures cannot model the complex relations between outfit items since each pair is treated independently. Some works [3] attempted to represent an outfit as an ordered sequence of garments (Figure 1 **b**) and using Recurrent Neural Network (RNN) to model compatibility. This kind of representation, however, is a forcing as it is reasonable to
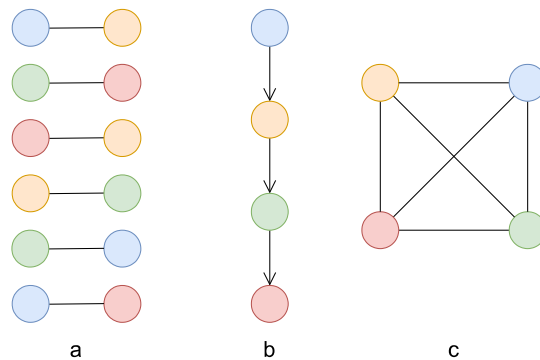
- *F. Becattini, F. M. Teotini and A. Del Bimbo are with the Media Integration and Communication Center (MICC) of the University of Florence, Italy.*
  E-mail: name.surname@unifi.it

Fig. 1. Different representations for an outfit composed by four different items: (**a**) the pairwise one, (**b**) the ordered sequence one, and (**c**) the graph one.

consider an outfit as a set, thus without a specified order, instead of as a list. To overcome these representation limitations, further architectures [4]– [5] explored the concept of representing an outfit as a graph (Figure 1 **c**).

In this paper a new architecture is proposed, *Transformer-based Graph Neural Network* (TGNN), aimed at generating new outfits starting from a garment or a set of garments and thus allowing even inexperienced people to create their own outfit. TGNN is based on two complementary architectures: the Transformer [6] and Graph Neural Network [7], [8]. Transformer is a kind of encoder-decoder architecture [9], [10] born in the Natural Language Processing (NLP) field and it is the base architecture for many state-of-art NLP models. GNN proved successful in modeling complex relations in very large graphs [11], [12] and is one of the most researched fields at the time of writing.

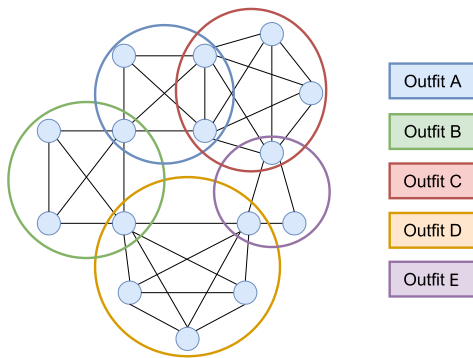The GNN part of TGNN is aimed at learning context

Fig. 2. The item relation graph. There are five different highlighted outfits, each connected to another at least through one or more shared items. In particular, each item is linked to all the other ones appearing in the same outfits.

features from this graph. The Transformer architecture, on the other hand, is employed for its outstanding performance on sequences modeling. Born to overcome the recurrent nature of architectures like LSTM [13] and GRU [9], the Transformer is capable of predicting the next item in a sequence, given the previous ones; this ability is a perfect fit for the outfit generation task starting from a seed: every predicted garment has to be compatible with the seed and the already predicted garments.
It has been proved that, under some constraints, GNNs can be seen as Transformer encoders [14]. TGNN uses this concept to integrate these two architectures and to successfully model the compatibility in fashion outfit generation.

To summarize, the main contributions of this work are:

- The proposal of Transformer-based Graph Neural Network (TGNN), a new architecture aimed at seeded outfit generation, which can better capture the complex relations among multiple items in an outfit.
- The development of an hybrid between Graph Neural Networks and Transformer encoders, treating the nodes' neighborhoods as interlinked unordered sequences.
- Experimental results on widely used evaluation tasks demonstrate the effectiveness of the developed architecture over other previous state-of-the-art ones.
- The introduction of a new evaluation task, Seeded Items Prediction (SIP), to test how good a model is at iteratively reconstructing a given outfit starting from one outfit item (the seed).

2 RELATED WORK

Fashion recommendation is a hot topic, for which several aspects have been studied. The problem has been declined in many forms, ranging from compatibility estimation [15], [16], [17], [18], [19], outfit generation [20] to plain simple recommendation [21], [22] and try-on [23], [24].

Many early works [1], [25]– [26] focus on calculating a pairwise compatibility metric between garments: for example, *McAuley et al.* [1] extracts visual features to model human visual preference for a pair of items, *Veit et al.* [27] develops a Siamese based network that estimates pairwise

compatibility based on co-occurrence in large-scale user behavior data, while *Lu et al* [26] aggregates user preferences on each item and integrates them with pairwise compatibility scores. These approaches lack efficiency and accuracy in real-world usage.

The first work considering a fashion compatibility between a set of items is *Han et al.* [3]: here an outfit is considered as an ordered sequence and fed to a bidirectional LSTM network to predict a compatibility score. In this work the Polyvore dataset was also introduced. *Vasileva et al.* [2] trains pairwise embedding spaces to learn different representations for different pairs of categories, that, however, is not feasible when the number of categories is high. Furthermore, they also extended and enriched the Polyvore dataset introducing more challenging evaluation sets.

Some works [4], [5], [28] focused on representing an outfit as a graph: in NGNN [4], *Cui et al.* represented an outfit as a graph to model complex relations among items which has been demonstrated to be more effective than pairwise and sequence representations. In our work we represent outfits as graphs, treating fashion items as nodes, connected to each other if they can be combined to compose an outfit. Differently from prior work, we process such graphs with a model harnessing the effectiveness of Graph Convolutions as well as multi-headed self-attention, typical of transformers [6]. To this end, we integrate the message passing strategy of Graph Convolutions as a form of attention in the encoder structure of a transformer model. Transformers have also been used recently in literature to generate outfit level representations for compatibility prediction tasks [29].

In addition to visual features, some take advantage of other kind of information like textual representations [2], [4], [20], categories [2], [30], and user preferences [30], [26], [31]. Lately, there have been studies [32] aimed at operating in semi-supervised settings, as creating fully labeled datasets is expensive and often requires fashion experts' knowledge.

3 OVERVIEW

Transformer-based Graph Neural Network (TGNN) is a novel architecture whose objective is to discover new aesthetically appealing outfits by picking and combining items from a collection. When referring to an outfit we refer to a composition of several garments or accessories of different categories that can be worn together. For instance, an outfit can be composed by combining a t-shirt, a skirt, a scarf, shoes, a bag and a necklace. Starting from an initial garment seed, TGNN is capable of generating unseen outfits complementing the given item. TGNN will iteratively chose garments from a candidate set that are compatible with the seed and with the other previously chosen garments, until a stop condition is reached. We make the assumption that the collection of garments from which to make a recommendation is fixed.

The TGNN architecture is based on a Transformer network: the encoder block operates on a collection of predefined outfits to learn the complex relations standing between garments, while the decoder carries out the outfit generation, conditioning the output with the seed and the contextual information provided by the encoder.

Following prior work, we represent an outfit as a graph clique. Since a single garment can belong to more than one outfit, the collection of pre-defined outfits is naturally modeled as a graph, connecting such cliques through nodes belonging to more outfits. The encoder blocks in TGNN are therefore a special kind of Graph Neural Network that adapts the standard transformer encoder to graph data structures.

3.1 Notation and Problem Definition

Let $\mathcal{O}$ be a collection of pre-defined outfits. Each outfit $o_i \in \mathcal{O}$ is composed of a combination of n_i garments $g_{i,j}$ with $j = 1, ..., n_i$ belonging to a set $\mathcal{G}$, i.e. $o_i = \{g_{i,1}, \cdots, g_{i,n_i}\}$. Each item belongs to a distinct fashion category, which we identify as $C(g_{i,j})$. Since the same garment can belong to several different outfits, we denote with $O(g_i) := \{o_j \mid g_i \in o_j\}$ the set of outfits in which the item appears. Every outfit might be composed of a different number of garments, depending on how many fashion categories have been used.

In order to build the data structure representing the collection of outfits, we provide the following definitions.

Definition 3.1 (Outfit Relation Graph). An Outfit Relation Graph (ORG) is an undirected graph whose nodes are outfits and edges exist between two nodes if the linked outfits share at least one garment. Formally, $G_{\mathcal{O}} = (V, \mathcal{E})$ such that

$$V \subseteq \mathcal{O}$$

$$\mathcal{E} = \{(o_i, o_j) \mid o_i \cap o_j \neq \emptyset, i \neq j\}$$

Definition 3.2 (Item Relation Graph). An Item Relation Graph (IRG) is an undirected graph whose nodes are garments and edges exist between two nodes if the linked garments belong to the same outfit. Formally, $G_{\mathcal{G}} = (V, \mathcal{E})$ such that

$$V \subseteq \mathcal{G}$$

$$\mathcal{E} = \{e_{i,j} := (g_i, g_j) \mid O(g_i) \cap O(g_j) \neq \emptyset\}$$

When dealing with an IRG, the direct neighborhood $\mathcal{N}_i = \{g_j \mid g_j \in V, e_{i,j} \in \mathcal{E}\}$ of node g_i is assumed to contain g_i itself, that is, an IRG is an undirected graph with self edges. An example of Item Relation Graph is shown in Fig. 2.

Definition 3.3 (ORG to IRG induction). An ORG can induce an IRG, meaning, an IRG can be built starting from an ORG. Formally, given an ORG $G_{\mathcal{O}} = (V, \mathcal{E})$, its induction $\overline{G_{\mathcal{O}}} = (\overline{V}, \overline{\mathcal{E}})$ is an IRG such that

$$\overline{V} = \{g_i \mid O(g_i) \cap V \neq \emptyset\}$$

$$\overline{\mathcal{E}} = \{e_{i,j} := (g_i, g_j) \mid g_i, g_j \in \overline{V}, O(g_i) \cap O(g_j) \neq \emptyset\}$$

Based on these concepts, we define the problem of Seeded Item Prediction as the task aimed at generating complementary items to compose an outfit, based on a garment seed and on an IRG $G_{\mathcal{G}}$. A garment seed is a sequence of garments appearing in $G_{\mathcal{G}}$, such that they do not belong to the same outfit, i.e., they are not linked. Formally

$$\varphi = \{g_{s,1}, \cdots, g_{s,n} \mid g_i \in V, e_{i,j} \notin \mathcal{E} \quad \forall i \neq j\}$$
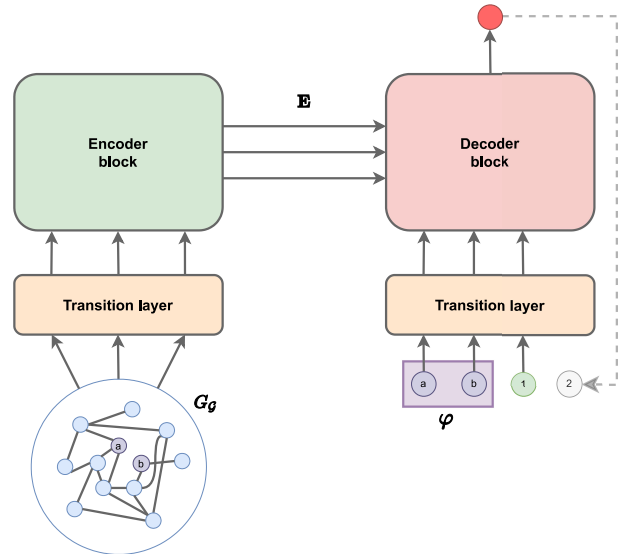

Fig. 3. The architecture of TGNN, here visualized during computation at time step $t = 2$ with a garment seed φ of size 2. The garment seed with the previous generated garment $\widehat{g}_1$ and $\mathbf{E}$ are fed into the decoder part. The results will become the next outfit garment $\widehat{g}_2$.

4 METHODOLOGY

Let $G_{\mathcal{G}}$ be an IRG, φ a garment seed, and $\mathcal{C} \subset \mathcal{G}$ the garment candidate set with $\varphi \cap \mathcal{C} = \emptyset$ and $\overline{g_\omega} \in \mathcal{C}$, where $\overline{g_\omega}$ is a fake garment indicating end-of-outfit, to be used as a stop sign during generation.

At each time step t, TGNN predicts the next outfit garment $\widehat{g}_t$, given $G_{\mathcal{G}}$, φ and all the previous generated garments $\{\widehat{g}_i\}_{i=1}^{t-1}$. The element $\widehat{g}_t$ is chosen among the elements of $\mathcal{C}^{(t)} = \mathcal{C}/\{g \in O(\widehat{g}_i)\}_{i=1}^{t-1}$. The outfit generated by TGNN at the end of step t is:

$$\widehat{o}^{(t)} = \varphi \cup \{\widehat{g}_i\}_{i=1}^t \qquad \widehat{g}_i \in \mathcal{C}, \widehat{g}_i \neq \widehat{g}_j \tag{1}$$

This process continues until the model outputs the stop sign $\overline{g_\omega}$. The final outfit is therefore $\widehat{o}^{(T-1)}$, where T is the total number of generation steps.

The full TGNN architecture is shown in Fig. 3. The node embeddings of $G_{\mathcal{G}}$ are first passed to a transition layer whose responsibility mirrors the embedding layer of a standard Transformer. Moreover it adapts the d_e-dimensional embeddings to the hidden dimension d_m of the model. Then, the encoder block operates on this modified graph to generate a new representation $\mathbf{E}$ of all the nodes/garments.

For the decoder part, at first, $\widehat{o}^{(t)}$ is passed through another transition layer, then it is fed to the decoder block along with $\mathbf{E}$. The result is a vector $\mathbf{h}^{(t)} \in \mathbb{R}^{d_m}$. Then $\widehat{g}_t$ is obtained as:

$$\widehat{g}_t = \arg \max_{g_c \in \mathcal{C}^{(t)}} \frac{\exp(\mathbf{h}^{(t)} \cdot \tau(\mathbf{g}_c))}{\sum_{g_i \in \mathcal{C}^{(t)}} \exp(\mathbf{h}^{(t)} \cdot \tau(\mathbf{g}_i))} \tag{2}$$

that is, the garment g_c of the candidate set, whose transition representation $\tau(\mathbf{g}_c)$ results in the highest similarity with $\mathbf{h}^{(t)}$, among all the candidate set items.

Note that in principle the number of candidates could become arbitrarily large. However, each element is independently fed to the transition layer and compared to the output of the decoder, as detailed in Eq. 2. This operation can be
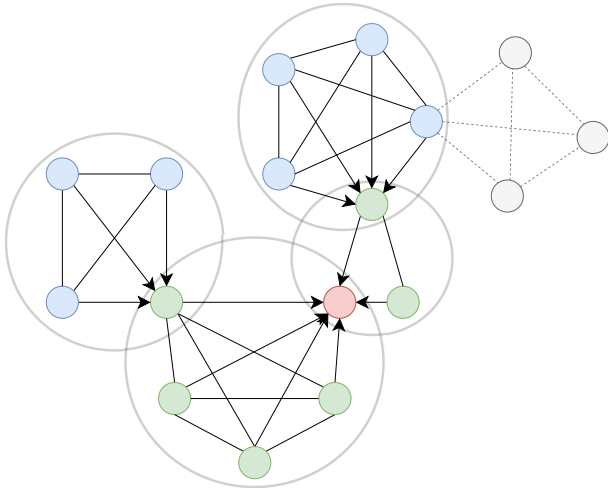
Fig. 4. An example illustration of how attention flows towards the red node in a $K_{enc} = 2$ encoder block. The information first flows from the green nodes to the red one, then from the blue nodes, through the green ones, up to the red one. Since $K_{enc} = 2$, no attention information coming from the grey nodes can arrive to the red one.

done easily for large candidate sets, since both operations can be carried out in parallel. If the number of candidate items is so high to make parallel computation unfeasible, an index (e.g. FAISS [33]) could be used to approximate the similarity computation.

In addition, in this paper we assume to have a closed collection of garments from which to make recommendations. Adding new garments would require to update the graph, which could be done at any time without requiring any further training.

4.1 Initial Garment Embeddings

Unlike Transformers for NLP [6], which deal with words, it is not possible to build a garment "vocabulary" containing all the existing garments. On the other hand, a garment can be associated with an image and/or a title. This means that instead of learning the initial embeddings as it is done with words, in the case of garments, they can be pre-computed as feature vectors.

In this work, each garment image is fed into an ImageNet pretrained ResNet50, and the outputs of the last layer before the classification block are then taken as garment embeddings. Since these embeddings are computed by ResNet50, their dimension is 2048. To compact the features we use PCA and project them to a lower dimension d_e.

4.2 Transition Layer

As shown in Fig. 3, there are two transition layers, one for the encoder block and one for the decoder block. These are simply feed-forward layers followed by a ReLU activation, that serve the purpose to learn a meaningful and compact representation to be fed to each block of the model. In particular, for the decoder, it also allows to map items in the candidate set into a semantic space suitable for identifying complementary items according to Eq. 2.

4.3 Transformer Encoder as ConvGNN

The encoder block operates on a graph containing a pre-defined collection of outfits, specifically an Item Relation Graph. The traditional encoder of a transformer is modified only in its self-attention layer: the idea, adapted from GAT [34], is to treat each node along with its direct neighbors as an unordered sequence and to apply to each of them the multi-head scaled dot-product attention mechanism. These sequences are actually outfits and, as outfits can share garments, these sequences are linked together. By stacking multiple of these modified encoder modules, the message passing nature of spatial ConvGNNs can be leveraged, meaning that a single garment can attend not only to the garments with which it shares a common outfit, but also to other garments belonging to linked outfits.

In Fig. 4, an example of how attention information is passed between different but linked outfits in an IRG, is shown.

Formally, referring to the MultiHead attention function adopted in transformers [6], the multi-head scaled dot-product attention mechanism calculated for a node $\mathbf{g}_i \in \mathbb{R}^d$ is defined as:

$$\text{MultiHead}(\mathbf{g}_i, \mathcal{N}_i, \mathbf{g}_i) \tag{3}$$

that is, the query Q and the value V are set to be equal to $\mathbf{g}_i$ while the key K is a $|\mathcal{N}_i| \times d$ matrix containing the feature vectors of the neighboring nodes of $\mathbf{g}_i$.

4.4 Objective Function

Let $o = \{g_1, \cdots, g_n\}$ be a predefined outfit and let $G_{\mathcal{G}} = (V, \mathcal{E})$ be an IRG where $g_1, \cdots, g_n \in V$ but $e_{i,j} \notin \mathcal{E} \; \forall \, g_i, g_j \in o$, and $\mathcal{C} \subset \mathcal{G}$ the garments candidate set with $o \subset \mathcal{C}$. Thus, for the triplet $(o, G_{\mathcal{G}}, \mathcal{C})$ the objective function of TGNN can be written as

$$\mathcal{L}_{(o, G_{\mathcal{G}}, \mathcal{C})} = -\frac{1}{n} \sum_{t=1}^{n} \log P(g_{t+1}|g_1, \cdots, g_t, G_{\mathcal{G}}, \mathcal{C}, \Theta) \tag{4}$$

where Θ denotes the model parameters, g_{n+1} is assumed to be the end-of-outfit fake garment $\overline{g_\omega}$, and $P(\cdot)$ is the probability of choosing the next correct garment conditioned on the previously predicted outfit garments, the IRG and the candidate set.

In one training sample, the i-th ground truth Γ_i is the next garment to be produced, that is $\Gamma_i = g_{i+1}$, as shown in Fig. 5, and $\Gamma_n = \overline{g_\omega}$.

In this setting, the conditioned probability of Equation (4) becomes

$$P(g_{t+1}|g_1, \cdots, g_t, G_{\mathcal{G}}, \mathcal{C}, \Theta) = \frac{\exp(\mathbf{h}^{(t)} \cdot \tau(\Gamma_t))}{\sum_{g_i \in \mathcal{C}} \exp(\mathbf{h}^{(t)} \cdot \tau(\mathbf{g}_i))} \tag{5}$$

5 EXPERIMENTS

In this section we outline the experimental validation of the proposed method and the analysis of the obtained results. In addition, both the network setup and the hyperparameter values used throughout the experiments are explained.
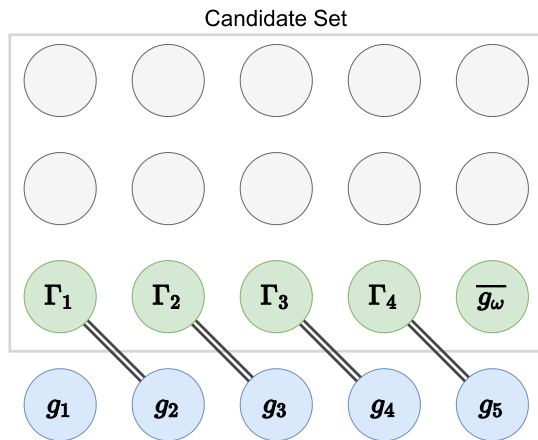
Fig. 5. Example illustration of a training sample outfit with five garments (in blue) and their corresponding ground truth items belonging to the candidate set. The ground truth for a given item is equal to the next one, while for the last is the end-of-outfit fake garment.

Fig. 6. An outfit example from the Polyvore dataset [2]

6 DATASET

To perform experimental validation of our approach, we adopt different versions of the Polyvore dataset [2].

The dataset has been built from the Polyvore website, a popular social network hub where fashion enthusiasts could upload, tag and title garment images and create custom outfits as compositions of these. The crafted outfits, then, could be shared, commented, liked, but also used as basis for other new outfits.

An eary version of the dataset was proposed by Han et al. [3], which is now referred to as the Maryland Polyvore dataset. This dataset was relatively small (around 20k outfits and 160k items), did not contain item categories, and had some inconsistencies in the test set. To resolve these issues, Vasileva et al. [2] produced another outfit dataset based on Polyvore data, referred to as the Polyvore dataset, with both coarse and finer grained item categories (defined respectively as "semantic" and "leaf" categories), titles and descriptions; this dataset is, also, larger than the Maryland dataset as it contains more than 60K outfits and 360K items. Moreover, this dataset provides carefully tailored test-train splits: as some garments appear in many different outfits, the choice of letting these garments appear in unseen test outfits has a significant effect. The Polyvore Dataset comes in two versions:

- **Polyvore-S** (Standard) – This version is easier as no outfit appearing in the training set or in the test set can appear in the other one, while it is possible that an item belonging to a training outfit is seen in a test one.

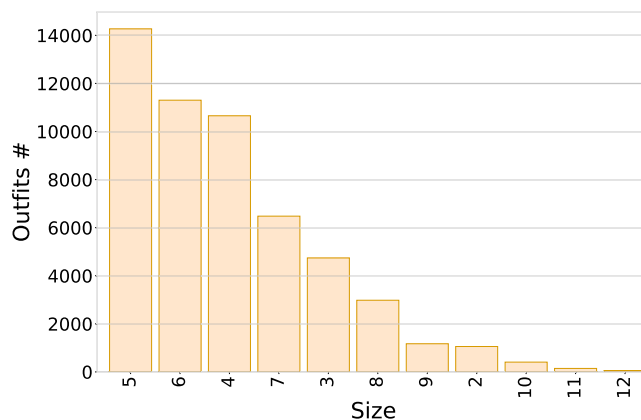| Version	#Outfits	#Items	Train Outfits	Val Outfits	Test Outfits
Polyvore-S	68306	365054	53306	5000	10000
Polyvore-D	32140	175485	16995	3000	15145

Fig. 7. The outfit size distribution of Polyvore-S train split

- **Polyvore-D** (Disjoint) – A more challenging version, where even garments are not allowed to appear in more than one set.

In Tab. 1, the outfit and item numbers, along with the split sizes, are summarized for both versions. In Fig. 7, the outfit size distribution for the Polyvore-S train split, is visualized.

7 TGNN SETUP

First, each garment image is distilled in a feature vector, as described in Section 4.1, by feeding it to a pretrained ResNet50 [35], and then through a PCA resulting in $d_e = 128$ dimensional feature vectors.

Let $G_{\mathcal{G}}$ be the Item Relation Graph of the dataset, as in Definition 3.2. Generally speaking, every dataset of interest will have an IRG too large to handle. In Tab. 2, some graph statistics for the Polyvore IRGs are shown. For example, the Polyvore-S train split IRG contains more than 200k nodes and more than 680k edges.

These numbers do not allow operating on the entire IRG at once, as it will saturate the GPU memory. To overcome this issue, we first split the graph into clusters and only retain relevant partitions. Let $G_{\mathcal{O}}$ be the dataset ORG, as in Definition 3.1. Based on the idea of Cluster-GCN [36], this ORG is partitioned, using the METIS clustering algorithm [37], in $\mathcal{P}$ subgraphs such that each of them contains about ϕ nodes (i.e. outfits). We refer to the p-th partition of the ORG as $G_{\mathcal{O},p}$. In our experiments we use $\phi = 50$.

7.1 Training Setup

Given an outfit $o = \{g_1, \cdots, g_n\}$, a training example is a triplet $(\hat{o}, G_{\mathcal{G},o}, \mathcal{C}_{\hat{o}})$ where $\hat{o}$ is a random permutation of the original outfit o. This means that the model sees a different shuffled version of the same outfit at every epoch, acting as a form of data augmentation and making the model invariant

TABLE 2
Graph statistics for both the train and test IRGs of the Polyvore-S and Polyvore-D datasets

	Polyvore-S		Polyvore-D	
	Train	Test	Train	Test
#Nodes	204679	47854	71967	70035
#Edges	685024	129496	192545	165313
Avg. Degree	6.693	5.412	5.35	4.721
Median Degree	5	5	5	4
Conn. Components	9226	4994	4837	11224
Transitivity	0.328	0.732	0.654	0.806
Avg. Cluster Coeff.	0.88	0.945	0.916	0.955

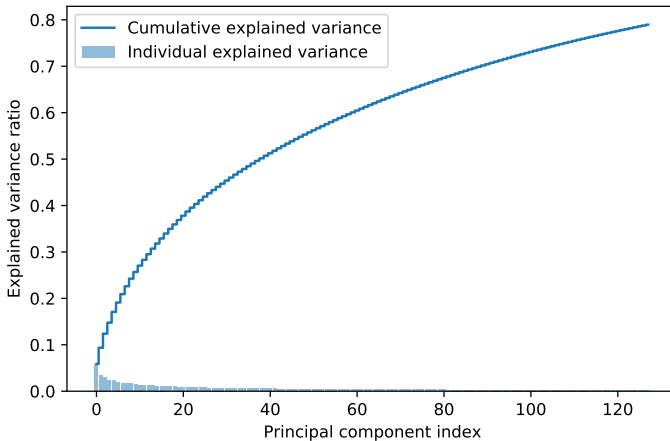Fig. 8. Explained variance for PCA computed on ResNet50 features extracted from Polyvore.

to garment ordering. To build the partitioned IRG $G_{\mathcal{G},o}$, relative to outfit o, we adopt the following steps:

1) Find the partition $G_{\mathcal{O},p} = (V_p, \mathcal{E}_p)$ such that $o \in V_p$.
2) From $G_{\mathcal{O},p}$, build its IRG induction $\overline{G_{\mathcal{O},p}} = (\overline{V_p}, \overline{\mathcal{E}_p}) = G_{\mathcal{G},p}$ as in Definition 3.3.
3) Remove all the edges $e_{i,j} \in \overline{\mathcal{E}_p}$ such that $g_i, g_j \in o, g_i \neq g_j$, i.e. all edges connecting elements of o.

Finally, for each garment g_i in the outfit that has to be predicted, we build a candidate set C_i containing $(n_C + n_R + 1)$ garments:

- n_C random negatives $g_j \in G_{\mathcal{G},o}$ such that $g_j \notin o$ and $C(g_j) = C(g_i)$, i.e. belonging to the same category of the ground truth item.
- n_R random distractors $g_j \in G_{\mathcal{G},o}$ such that $g_j \notin o$.
- 1 garment representing the ground truth item Γ_i.

We refer to $\mathcal{C}_{\hat{o}} = \{\mathcal{C}_1, \cdots, \mathcal{C}_n\}$ as the set of candidate sets for each step. In our experiments we use, unless expressly stated otherwise, $n_C = 3$ and $n_R = 5$.

The model is trained to output the sequence of garments in the outfit, starting from a garment seed and optimizing the loss defined in Eq. 4 at each generation step.

7.2 Training Details and Hyperparameters

If not otherwise stated, the following hyperparameter values are used throughout all the experiments. We set the number of modules in the encoder block, i.e. the aggregation
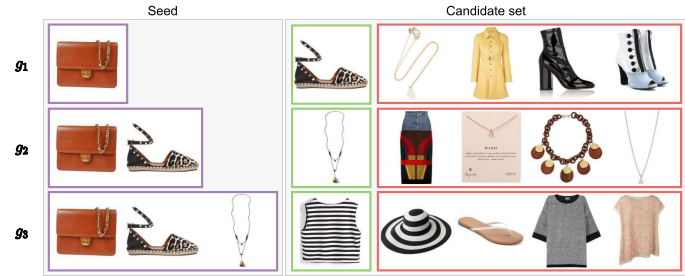

Fig. 9. SIP is a task in which a model has to predict, for each item in a given outfit, the next one. Here is an example for the first 3 items of an outfit, along with their candidate sets from which to choose the next correct item. This task objective is to find the correct garment $\Gamma_i = g_{i+1}$ given the seed $(g_1, \cdots, g_i)$.

radius, equal to $K_{enc} = 2$. As for the decoder, we use a number of modules $K_{dec} = 4$. Garment images are projected using PCA to an embedding of dimension $d_e = 128$. The explained variance for the PCA is depicted in Fig. 8. Such embeddings are projected by the transition layer into a hidden space with $d_m = 256$ dimensions. For multi-head attention, we use $H = 8$ heads.

The network is optimized with the Adam [38] optimizer with learning rate $5 \cdot 10^{-4}$ and weight decay $5 \cdot 10^{-5}$. The learning rate is reduced by a 0.1 factor each time the validation loss reaches a plateau. Dropout is applied to each dense layer and with value 0.35. The model is trained for a maximum of 1000 epochs with an early stopping strategy on the validation loss with a 10 epoch patience.

The GPU used during training is a Nvidia Titan RTX with 24GB memory. Half-precision was employed to lower memory requirements and training time.

8 EVALUATION TASKS OVERVIEW

The performance of TGNN is evaluated on three different tasks. Our main focus is on outfit generation, for which we formalize an evaluation protocol named Seeded Item Prediction. However, we also apply our model to standard compatibility tasks, namely Fill-In-The-Blank and Compatibility Prediction, which are widely adopted in literature [2], [3], [32]. In the following we provide a brief overview of such tasks and relative evaluation metrics.

Seeded Item Prediction (SIP) is a task where the model is presented with a partial outfit and it has to incrementally predict the next complementary item given the previous ones from a collection of possible candidates (Fig. 9). The performance is evaluated using the accuracy in choosing the correct item.

Fill-In-The-Blank (FITB) is a task in which the model is presented with an incomplete outfit missing one item. Along with the outfit, four candidate items (three incorrect and one correct), are provided as possible answers. The task is then to choose the item between the candidates that is the most compatible with the given incomplete outfit (Fig. 10). The results are evaluated as overall accuracy. It must be noted that FITB is a special case of the SIP task: the two tasks are the same when the seed of SIP is the entire outfit minus one item and the possible answer set is composed by
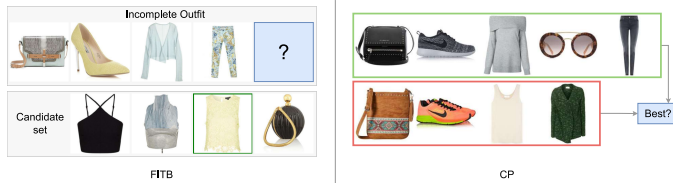
Fig. 10. Visual illustration of the FITB (left) and the CP (right) tasks. **Left** – In FITB, the task has to choose an item from a candidate set that is most compatible with a given incomplete outfit; the performance is evaluated by the accuracy of choosing the correct item, here visualized as the green one. **Right** – In CP, the task scores how much a candidate outfit is compatible; to do this, it compares pairs of compatible (green) and incompatible (red) outfits, trying to assign a higher score to the compatible one.

four items. In this sense, the commonly adopted FITB task is an oversimplification of SIP.

Compatibility Prediction (CP) is a task where the model has to predict whether a candidate outfit is compatible or not (Fig. 10). Outfits are scored to assess whether their constituting items are compatible with each other. The task is performed by feeding to the model two different outfits, namely a positive ground truth outfit, which is known to be compatible, and a negative random outfit. The model is requested to score the two outfits and the task is considered successful when the positive one is scored higher than the negative one. The performance is evaluated using the area under the receiver operating characteristic curve (AUROC).

8.1 Test Setup

Since the TGNN architecture works with triplets containing an outfit, a graph partition and a candidate set in the form of $(o, G_{\mathcal{G},o}, \mathcal{C}_o)$, like the ones described in Section 7.1, these must be carefully constructed for each evaluation task.

For the Fill-In-The-Blank task, given the original complete outfit $o = \{g_1, \cdots, g_b, \cdots, g_n\}$, where g_b is the blank garment placeholder corresponding to the item to be predicted, the triplet will be $(\hat{o}, G_{\mathcal{G},o}, \mathcal{C}_{\hat{o}})$ where:

- $\hat{o}$ is the incomplete outfit, that is, o without the blank item g_b.
- $G_{\mathcal{G},o}$ is a graph partition built in the same way as described in Section 7.1. However, instead of identifying the partition that exactly contains o, we retrieve the most similar garments in the training set and take the partition containing the maximum number of such garments. This is done because, in general, a test outfit might not be present in the training set. By doing so, we assume that test garments have some sufficiently similar items in the training graph, which we believe to be a reasonable assumption. Other methods from the state of the art not depending on a garment graph may not make this assumption.
- $\mathcal{C}_{\hat{o}} = \mathcal{C}_b$ containing the four possible answers, among which there is the correct garment. Note that there is a single candidate set, since in the FITB task we only need to perform one prediction step to complete the outfit.

For the Compatibility Prediction task, instead, we need to identify a graph partition for both the positive and neg-

ative outfits. Both partitions are taken looking for the most similar items in the training set, as in the FITB configuration.

We then generate a random partition of the garments and ask the model to recreate the outfit from a single-element seed. We average the likelihood of the correct garment at each generation step to obtain the final compatibility prediction score. In order to do so, we need to define a candidate set $\mathcal{C}_i$ for each step. Each one is composed of the ground truth item Γ_i and three other random garments belonging to $G_{\mathcal{G},o}$.

Finally, for the Seeded Item Prediction task, the setup is the same as Section 7.1 without the initial random permutation. Also in this case, the graph partition is obtained by taking the one with the maximum number of most similar items to the partial outfit in the training set.

8.2 Baselines

To validate the hypothesis that contextual information, i.e. the Item Relation Graph, and self-attention play an important role for compatibility learning, TGNN is compared with previously reported approaches on the Polyvore dataset:

- **Siamese Network** - The approach described by *Veit et al.* [27] that estimates pairwise compatibility based on co-occurrence in large-scale user behavior data.
- **Bi-LSTM** - *Han et al.* [3] was the first work to consider an outfit as a whole, representing it as an ordered sequence. Taking multi-modal data as input, it can calculate the compatibility scores of outfits by iteratively predicting the next item.
- **CSN T1:1** - Learns a pairwise category-dependent transformation using the approach of *Veit et al.* [39] to project a general embedding to a single type-specific space, measuring compatibility between two item categories.
- **Type Aware** - This method, described in *Vasileva et al.* [2], focuses on learning item image embeddings that respect the item type by measuring pairwise item similarity in different semantic subspaces.
- **SCE-Net** - Introduced in *Tan et al.* [40], it jointly learns pairwise different representation subspaces for different similarity conditions without explicit supervision.
- **CSA-Net** - Approach introduced by *Lin et al.* [41] that models outfit compatibility by encoding category pairs information to each item embedding vector.
- **Pseudo-Label** - Described in *Revanur et al.* [32], this method, developed for semi-supervised settings, learns item embeddings by generating pseudo positive and negative pairs on-the-fly during each training epoch. These pairs can be based on different characteristics, like colors, shapes, etc. while discarding the item category notion.
- **ADDE-O** - The method [42] is based on learning disentangled features for specific attribute classes. This allows to perform targeted manipulations exploiting a memory bank of attribute prototypes.
- **MUFIN** - Recently proposed in [43], the method addresses the task of Extreme Classification, i.e. a

TABLE 3
Accuracy on the Seeded Item Prediction task for both Polyvore-D and Polyvore-S datasets.

	Polyvore-D	Polyvore-S
Random	11.0	11.2
Type Aware [2]	43.3	42.0
TGNN	51.3	55.6

Fig. 11. Qualitative results obtained by TGNN on Polyvore-D. Images marked as *seed* are garments belonging to the same outfit, whereas *generated* ones are items suggested by TGNN to complete the outfit.

training strategy involving millions of labels leveraging both visual and textual descriptors.

- **OutfitTransformer** - This paper [29] proposes to obtain outfit-level representations with transformers to address the tasks of compatibility prediction and fill in the blank, as well as a complementary outfit retrieval taks.

We also report variants of some of the methods, referred to as *VSE*, which add a visual-semantic embedding, as described in [2], jointly learned with the compatibility embedding.

9 RESULTS

We first evaluate our model on the newly introduced task of Seeded Item Prediction (SIP). The task was created to evaluate the outfit generation capabilities of TGNN. As illustrated in Fig. 9, given an outfit $o = \{g_1, \cdots, g_n\}$, the task objective is to correctly predict the next item $\Gamma_i = g_{i+1}$ from a candidate set, given the previous items $\{g_1, \cdots, g_i\}$. Tab. 3 shows how TGNN performs on this task, along with two baselines, by measuring the accuracy of choosing the correct item. Since SIP is a novel evaluation task proposed in this work, there are no available results from previous works. To provide a solid baseline, the task was evaluated with the publicly released code of the Type Aware architecture[1]. To generate the results, we trained the model with the same hyperparameter values as described in the original

1. Original code available at https://github.com/mvasil/fashion-compatibility

Fig. 12. Failure cases. In the first example, the model has failed to complete the outfit, picking only a bottom and a bag without choosing a top. In the second one instead TGNN has built an outfit with a different style than one ground truth one.

paper [2]. We provide also a random baseline as a lower bound.

TGNN performs well on this difficult task, showing that it is capable of correctly generating outfits. There is a gap between the accuracy obtained with the two Polyvore versions. We attribute this to the attention mechanism, whose characteristic is to learn how different items are related among themselves. Therefore, the presence, or absence, in the test set of items seen during the training phase could most likely influence this mechanism, and, in turn, the whole model performance.

In Fig. 11 we report qualitative results of correctly generated outfits by TGNN in the Seeded Item Prediction setting. It can be seen that TGNN is able to generate outfits with a variable number of outfits, starting from a variable number of seeds. We also show a few failure cases in Fig. 12. In the first failure case, the model emits the stop sign before completing the outfit, lacking a top garment to complete the outfit. In the second example, the model generates an outfit with a style different from the ground truth one. This behaviour is easily avoidable by adding more seed items to the input.

We then evaluate TGNN for the tasks of Fill-In-The-Blank (FITB) and Compatibility Prediction (CP). As shown in Tab. 4, TGNN outperforms previous state of the art methods on both tasks. These results suggest that the graph structure provides valuable information for this kind of task. In particular for FITB it is indeed to be expected that, within an IRG, the missing garment (or one similar enough) will be found close to its complementary items. At the same time, for Compatibility Prediction we believe that the multi-head self-attention mechanisms of TGNN plays an important role: by learning to model compatibilities between an item and all the other ones within the same outfit, the model implicitly learns a global outfit compatibility notion by combining all of these item compatibilities.

Since several methods adopt a ResNet18 backbone [35] rather than a ResNet50 like ours, we also show results for a

TABLE 4
Comparison of TGNN against state-of-the-art methods. Some of these rely on additional supervision such as text embeddings and explicit category supervision. FITB accuracy and Compatibility AUROC are reported. *Higher is better.*

Method	Text labels	Explicit category	Polyvore-D		Polyvore-S	
			FITB acc	CP AUROC	FITB acc	CP AUROC
Siamese Network [2]	×	×	51.8	0.81	52.9	0.81
Bi-LSTM + VSE [3]	✓	×	39.4	0.62	39.7	0.65
CSN T1:1 [39]	×	✓	52.5	0.82	54.0	0.83
CSN T1:1 + VSE [39]	✓	✓	53.0	0.82	54.5	0.84
Type Aware [2]	✓	✓	55.65	0.84	58.83	0.86
SCE-Net (avg) [41]	✓	×	53.67	0.82	59.07	0.88
Pseudo-Label [32]	×	×	54.6	0.84	57.9	0.89
CSA-Net [41]	×	✓	59.26	0.87	63.73	0.91
ADDE-O [42]	×	×	60.53	0.88	65.16	0.93
MUFIN [43]	✓	×	64.17	-	-	-
OutfitTransformer [29]	✓	×	59.48	0.88	67.10	0.93
TGNN-18	×	×	65.36	0.93	68.72	0.95
TGNN	×	×	**65.74**	**0.94**	**69.03**	**0.96**

TABLE 5
TGNN results for Seeded Item Prediction and Compatibility Prediction, calculated for different ϕ values.

	Polyvore-D		Polyvore-S	
	SIP	CP	SIP	CP
$\phi = 25$	45.7	0.88	52.4	0.89
$\phi = 50$	**51.3**	**0.94**	**55.6**	**0.96**
$\phi = 100$	44.0	0.78	49.7	0.80
$\phi = 200$	34.8	0.78	42.9	0.80

TABLE 6
TGNN performance on Polyvore-D for different K_{enc} values.

	Polyvore-D		Polyvore-S		
	SIP	CP	SIP	CP	Train time
$K_{enc} = 2$	35.9	0.77	32.6	0.83	3h 27m
$K_{enc} = 3$	42.5	0.86	39.0	0.91	5h 50m
$K_{enc} = 4$	**51.3**	0.94	**55.6**	**0.96**	10h 22m
$K_{enc} = 5$	**51.3**	**0.95**	55.5	0.94	17h 31m

variant of TGNN using ResNet18 features to establish a fair comparison. Apart from the backbone the whole training and evaluation procedure is left unchanged. Indeed, using ResNet18 features there is a slight performance drop but TGNN still manages to achieve state of the art results.

9.1 Ablation Studies

To better analyze the TGNN behavior, it is useful to correlate the performance with those hyperparameters peculiar to the proposed architecture, namely ϕ – the one controlling the partition sizes – and K_{enc} – the one defining how far from a node the information gets aggregated into the said node, i.e. the aggregation radius.

9.1.1 Partition Size

In Tab. 5, the TGNN results for different ϕ values are shown. It is clear that, when increasing ϕ, i.e. when the partitions $G_{\mathcal{G},o}$ grow larger, TGNN performance decreases. The main reasons are:

1) The encoder block is no longer able to learn, and thus to integrate into the nodes useful relational information from the surrounding context. In other words, there is too much information and the encoder block fails to handle it.

2) The *Encoder-Decoder Attention* of each decoder module relates each item $g_i \in o$ with all the nodes of $G_{\mathcal{G},o}$; with higher ϕ values, the number of nodes grows exponentially, thus, making the Encoder-Decoder Attention layer increasingly complex.

On the contrary, when lowering the partition size too much, the accuracy decreases since not enough context is given to the model.

9.1.2 Aggregation Radius

In Tab. 6, the results obtained by TGNN are shown, at increasing values of the aggregation radius K_{enc}. The experiments were carried out with all the hyperparameter values fixed to the ones listed in Sec. 7.2. It is clear that increasing K_{enc} yields better results. In fact, K_{enc} controls how many encoder modules are used in the encoder block, which, in turn, affects how far from a given node the information gets aggregated, i.e. from the K_{enc}-neighborhood of the said node. Increasing K_{enc} does indeed help the encoder block at managing the information in the graph, while learning useful relationships between loosely coupled outfits. Eventually, the benefits of increasing K_{enc} saturate, bringing negligible or no benefits, yet requiring a much longer training time (Tab. 6).

10 CONCLUSIONS

This work focused on the problem of outfit generation starting from a garment seed, which consists in proposing a set of garments that, combined with the seed, assembles a compatible outfit. To this aim, *Transformer-based Graph Neural Network* is proposed as a novel architecture which combines the capabilities of Graph Neural Networks with the powerful Transformer model. The proposed method is capable of learning the complex relations existing between items and outfits by employing graph representations and attention mechanisms. Through message propagation and self-attention, an item compatibility notion is refined by aggregating the interaction information derived from its neighbors. This knowledge is then used in the generation phase to iteratively choose the item that is most compatible with the seed and the previously chosen ones. Extensive

experiments on the Polyvore dataset have demonstrated the rationality and the effectiveness of TGNN, which outperforms previous models in outfit compatibility tasks.

11 FURTHER DEVELOPMENTS

Since the iterative generation mechanism is inspired to the one employed in many NLP models, we plan to improve the generation performance with search strategies inspired by the field of NLP, such as Beam Search. Another matter worth investigating is making the architecture work with a dynamic IRG. In other words a graph that can change over time by adding new nodes or edges. Being able to handle such real-world situations could prove fundamental to evolve a research model in a production ready one. Furthermore, to resolve the Encoder-Decoder Attention issue when the ϕ value increases, further studies could focus on creating masks in order to intelligently limit the number of partition nodes to which any outfit item is related to. Finally, an assessment from domain experts as an additional term of comparison between methods could help understanding which aspects are taken into account while composing the outfit, such as style, color, etc.

ACKNOWLEDGMENTS

REFERENCES

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

[2] M. I. Vasileva, B. A. Plummer, K. Dusad, S. Rajpal, R. Kumar, and D. Forsyth, "Learning Type-Aware Embeddings for Fashion Compatibility," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11220 LNCS, 2018, pp. 405–421.

[3] X. Han, Z. Wu, Y. G. Jiang, and L. S. Davis, "Learning fashion compatibility with bidirectional LSTMs," in *MM 2017 - Proceedings of the 2017 ACM Multimedia Conference*, 2017, pp. 1078–1086.

[4] 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," in *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, 2019, pp. 307–317.

[5] G. Cucurull, P. Taslakian, and D. Vazquez, "Context-aware visual compatibility prediction," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, 2019, pp. 12609–12618.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[7] M. Gori, G. Monfardini, and F. Scarselli, "A New Model for Learning in Graph Domains," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, 2005, pp. 729–734. [Online]. Available: https://www.researchgate.net/publication/4202380

[8] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 2014.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 4, 2014, pp. 3104–3112.

[11] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 2017, pp. 1025–1035.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] J. Chaitanya, "Transformers are Graph Neural Networks — NTU Graph Deep Learning Lab," *The Gradient*, 2020. [Online]. Available: https://graphdeeplearning.github.io/post/transformers-are-gnns/

[15] X. Song, F. Feng, J. Liu, Z. Li, L. Nie, and J. Ma, "Neurostylist: Neural compatibility modeling for clothing matching," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 753–761.

[16] X. Song, X. Han, Y. Li, J. Chen, X.-S. Xu, and L. Nie, "Gp-bpr: Personalized compatibility modeling for clothing matching," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 320–328.

[17] L. D. Divitiis, F. Becattini, C. Baecchi, and A. D. Bimbo, "Garment recommendation with memory augmented neural networks," in *International Conference on Pattern Recognition*. Springer, 2021, pp. 282–295.

[18] L. De Divitiis, F. Becattini, C. Baecchi, and A. D. Bimbo, "Disentangling features for fashion recommendation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2022.

[19] F. Becattini, L. De Divitiis, C. Baecchi, and A. Del Bimbo, "Fashion recommendation based on style and social events," *arXiv preprint arXiv:2208.00725*, 2022.

[20] W. Chen, P. Huang, J. Xu, X. Guo, C. Guo, F. Sun, C. Li, A. Pfadler, H. Zhao, and B. Zhao, "POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion," *arXiv:1905.01866 [cs]*, May 2019. [Online]. Available: http://arxiv.org/abs/1905.01866

[21] Y. Lin, P. Ren, Z. Chen, Z. Ren, J. Ma, and M. De Rijke, "Explainable outfit recommendation with joint outfit matching and comment generation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1502–1516, 2019.

[22] L. De Divitiis, F. Becattini, C. Baecchi, and A. Del Bimbo, "Style-based outfit recommendation," in *2021 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2021, pp. 1–4.

[23] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, and R. Cucchiara, "Dress code: High-resolution multi-category virtual try-on," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2231–2235.

[24] N. Zheng, X. Song, Z. Chen, L. Hu, D. Cao, and L. Nie, "Virtually trying on new clothing with arbitrary poses," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 266–274.

[25] 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*, 2016, pp. 1096–1104. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Liu_DeepFashion_Powering_Robust_CVPR_2016_paper.html

[26] Z. Lu, Y. Hu, Y. Jiang, Y. Chen, and B. Zeng, "Learning Binary Code for Personalized Fashion Recommendation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10562–10570. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Lu_Learning_Binary_Code_for_Personalized_Fashion_Recommendation_CVPR_2019_paper.html

[27] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. Belongie, "Learning Visual Clothing Style With Heterogeneous Dyadic Co-Occurrences," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4642–4650. [Online]. Available:

https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Veit_Learning_Visual_Clothing_ICCV_2015_paper.html

[28] X. Yang, X. Du, and M. Wang, "Learning to Match on Graph for Fashion Compatibility Modeling," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 287–294, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5362

[29] R. Sarkar, N. Bodla, M. Vasileva, Y.-L. Lin, A. Beniwal, A. Lu, and G. Medioni, "Outfittransformer: Outfit representations for fashion recommendation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2263–2267.

[30] X. Li, X. Wang, X. He, L. Chen, J. Xiao, and T. S. Chua, "Hierarchical Fashion Graph Network for Personalized Outfit Recommendation," in *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 159–168.

[31] F. Becattini, X. Song, C. Baecchi, S.-T. Fang, C. Ferrari, L. Nie, and A. Del Bimbo, "Plm-ipe: A pixel-landmark mutual enhanced framework for implicit preference estimation," in *ACM Multimedia Asia*, 2021, pp. 1–5.

[32] A. Revanur, V. Kumar, and D. Sharma, "Semi-Supervised Visual Representation Learning for Fashion Compatibility," in *Fifteenth ACM Conference on Recommender Systems*. Amsterdam Netherlands: ACM, Sep. 2021, pp. 463–472. [Online]. Available: https://dl.acm.org/doi/10.1145/3460231.3474233

[33] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *arXiv:1710.10903 [cs, stat]*, Feb. 2018. [Online]. Available: http://arxiv.org/abs/1710.10903

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

[36] W. L. Chiang, Y. Li, X. Liu, S. Bengio, S. Si, and C. J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 257–266. [Online]. Available: https://doi.org/10.1145/3292500.3330925

[37] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, Jan. 1998. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/S1064827595287997

[38] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. [Online]. Available: http://arxiv.org/abs/1412.6980

[39] A. Veit, S. Belongie, and T. Karaletsos, "Conditional Similarity Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 830–838. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Veit_Conditional_Similarity_Networks_CVPR_2017_paper.html

[40] R. Tan, M. I. Vasileva, K. Saenko, and B. A. Plummer, "Learning Similarity Conditions Without Explicit Supervision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10 373–10 382. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/html/Tan_Learning_Similarity_Conditions_Without_Explicit_Supervision_ICCV_2019_paper.html

[41] Y.-L. Lin, S. Tran, and L. S. Davis, "Fashion Outfit Complementary Item Retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3311–3319. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Lin_Fashion_Outfit_Complementary_Item_Retrieval_CVPR_2020_paper.html

[42] Y. Hou, E. Vig, M. Donoser, and L. Bazzani, "Learning attribute-driven disentangled representations for interactive fashion retrieval," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 147–12 157.

[43] A. Mittal, K. Dahiya, S. Malani, J. Ramaswamy, S. Kuruvilla, J. Ajmera, K.-h. Chang, S. Agarwal, P. Kar, and M. Varma, "Multimodal extreme classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 393–12 402.

Federico Becattini obtained his PhD in 2018 from the University of Florence under the supervision of Prof. Alberto Del Bimbo and Prof. Lorenzo Seidenari. Currently he is a PostDoc at the University of Florence, where he is involved in numerous collaborations, mostly focusing on Autonomous Driving, Scene Understanding and Fashion Recommendation. He attended several international conferences both as speaker and volunteer, as well as summer schools. He served to the scientific community as a reviewer for scientific journals and conferences. he is Associate Editor of the International Journal of Multimedia Information Retrieval.

Federico Maria Teotini received a master degree cum laude in 2022 in computer engineering from the University of Florence with the thesis "Graph-Transformer Network for Outfit Generation".

Alberto Del Bimbo is a Full Professor of Computer Engineering, and the Director of the Media Integration and Communication Center with the University of Florence. His scientific interests are multimedia information retrieval, pattern recognition, and computer vision. From 1996 to 2000, he was the President of the IAPR Italian Chapter and the Member-at-Large of the IEEE Publication Board from 1998 to 2000. He was the General Co-Chair of ACMMM2010 and ECCV2012. He was nominated ACM Distinguished Scientist in 2016. He received the SIGMM Technical Achievement Award for Outstanding Technical Contributions to Multimedia Computing, Communications and Applications. He is an IAPR Fellow, and Associate Editor of Multimedia Tools and Applications, Pattern Analysis and Applications, Journal of Visual Languages and Computing, and International Journal of Image and Video Processing, and was Associate Editor of Pattern Recognition, IEEE Transactions on Multimedia, and IEEE Transactions on Pattern Analysis and Machine Intelligence.