Toward Explainable Fashion Recommendation

Pongsate Tangseng[1] and Takayuki Okatani[1,2]

[1]Graduate School of Information Sciences, Tohoku University
[2]RIKEN Center for AIP
{tangseng,okatani}@vision.is.tohoku.ac.jp

Abstract

Many studies have been conducted so far to build systems for recommending fashion items and outfits. Although they achieve good performances in their respective tasks, most of them cannot explain their judgments to the users, which compromises their usefulness. Toward explainable fashion recommendation, this study proposes a system that is able not only to provide a goodness score for an outfit but also to explain the score by providing reason behind it. For this purpose, we propose a method for quantifying how influential each feature of each item is to the score. Using this influence value, we can identify which item and what feature make the outfit good or bad. We represent the image of each item with a combination of human-interpretable features, and thereby the identification of the most influential item-feature pair gives useful explanation of the output score. To evaluate the performance of this approach, we design an experiment that can be performed without human annotation; we replace a single item-feature pair in an outfit so that the score will decrease, and then we test if the proposed method can detect the replaced item-feature pair correctly using the above influence values. The experimental results show that the proposed method can accurately detect bad items in outfits lowering their scores.

1. Introduction

Recently, there have been many studies of applying computer vision techniques to various problems of fashion, such as quantifying/measuring goodness of outfits [7, 11, 16, 33, 38] and recommending to users outfits from a pool of items [19, 38] or outfits that fit users's personal preferences [13] or location [40]. However, many of the existing studies, particularly the recent ones that employ CNNs, rely on black-box models, which may provide good performance on respective tasks but cannot explain the reason of their judgments [13, 16, 19, 38]. There are a few attempts to develop models that can provide useful explanations [7, 40], but they require a large amount of manually annotated data for supervised training of the models, which is expensive and usually not publicly available.

In this study, we propose a system that is able not only to judge and quantify goodness/badness of an outfit but also to provide a reason(s) of the prediction. Similar to existing methods, our system receives images of multiple items comprising an outfit as inputs and then computes a score quantifying its goodness/badness of the outfit; example inputs are shown in the rows of Fig. 1. This forward computation is done by a part of our system called the outfit grader. To explain the output score, we quantify and use *how large the influence of each item, or of each feature of each item, is on the predicted score.* This enables to identify which item and what feature make the outfit good or bad; examples of the identification are shown in Fig. 1. For this purpose, we represent each item, rigorously its image, with a combination of human-interpretable features, and thereby the identification of the most influential item-feature pair will be a useful explanation of the score.

To measure the influence of item-feature pairs, we employ the multiplication of an individual feature with the gradient of the output score with respect to the feature. This is similar to the methods for visualizing inference of CNNs, such as the multiplication of an input image with its sensitivity map [35, 37] (i.e., the score gradient with respect to image pixels) and Grad-CAM [32]. The values thus computed are averaged and normalized within each feature of each item to yield our measure of the influence of the item-feature pair, which we call its *Item-Feature Influence Value* (IFIV). Note that our method does not need extra training data other than those for training the outfit grader.

It is usually hard to evaluate explanations provided by AI systems, since their quality can theoretically be evaluated only by humans. Human evaluation is generally costly; moreover, in our case, it is difficult to perform and conveys open problems, as the judgments to be explained are often

Figure 1: Our system first predicts a goodness score of an input outfit consisting of multiple items. It then identifies which item and what feature is the cause of, for instance, a low score. It is able not only to perform item-level identification (first row) but also to perform feature-level identification (second and third rows).

subjective. To cope with this difficulty, we employ an automatic evaluation method by designing a test for the evaluation that is based on synthesis of datasets. The basic idea is that i) we first replace a single item or its single feature of an outfit so that the resulting score will decrease and ii) we then test if the proposed method can detect the replaced item by identifying the item-feature pair with the maximum IFIV.

The organization of this paper is as follows. We first discuss the related work in Sec. 2. Next, we describe the proposed method for explaining judgments made by our outfit grader on the quality of input outfits in Sec. 3. Section 4 explains and evaluates the outfit grader that is the target of explanation. Experimental results on the proposed method for explaining its judgments are provided in Sec. 5. Section 6 concludes this study.

2. Related Work

2.1. Measuring Goodness of Outfits

There is a growing interest in the application of computer vision techniques to measure the goodness of outfits. The authors of [33] predicted fashionability scores from an outfit image and tags. The authors of [11] use bidirectional LSTM (Bi-LSTM) [9] to learn the compatibility relationship among fashion items by modeling an outfit as a sequence, whereas fully-connected layers are employed in [16, 38]. In [11, 16, 38], CNNs trained for generic image recognition are used to extract features for their respective purposes. Overall, the proposed methods in these studies work fairly well for measuring the goodness of outfits, i.e., predicting a score for each outfit. However, these methods lack the ability of providing reasons of the predicted scores.

2.2. Explaining Inference of Models

Recent advances in deep learning have dramatically improved accuracy of many computer vision tasks, such as image classification [12, 34, 36], object detection [30], object segmentation [5, 21], Visual-Question Answering (VQA) [1, 8, 22, 29], etc. These progresses have left behind explanation and understanding of what the deep neural networks have learned as well as how they make inference/judgments. Thus, there is a growing concern particularly about life-critical applications [18]. A number of studies have been conducted to resolve this so far; [2,31,32,41] to name a few. LIME [31] is a method for explaining the prediction of a machine learning model for an input, which estimates a linear model that locally approximates the model at the neighborhood of the input, and then uses it for explanation. There are many studies of visualization of inference made by CNNs. The authors of [41] proposed the Class Activation Map (CAM) for a particular class of CNN models, which shows the region in the input image that is responsible for the prediction. This is later extended to Grad-CAM [32], which is be applicable to more general CNN models, including image captioning [4, 15, 39], and Visual Question Answering (VQA) [1, 8, 22, 29].

2.3. Explainable Models for Fashion

The aforementioned computer vision systems for fashion [13, 16, 19, 23, 24] employ black-box models, too, which show fairly good performance for the respective tasks but lack ability of providing reason of inference/judgment. It is not straightforward to apply the above generic methods for explaining machine learning and deep learning models to these systems for fashion, because the problems are basically more complicated (e.g., multiple items contained in an outfit, stratified factors affecting the goodness/badness of an outfit etc.)

There are a few studies that attempt to provide useful explanation on model's evaluation of outfits [7, 17]. The method proposed in [7] relies on a massive amount of annotated data to train a multi-category attribute predictor and create a composition graph based on pairwise co-occurrence of those predicted attributes in outfits. On the other hand, the method proposed in [17] provides an upper-lower matching recommendation with textual explanation by utilizing comments provided by users of polyvore.com. Although this method does not require manual annotation, it can deal with only two items in each outfit.

3. Explaining Goodness of Outfit

Figure 2 shows an overview of the proposed system. It employs the outfit grader developed in [38], which classifies an input outfit either as positive (a good outfit) or negative (a bad outfit). We wish to explain judgment made by

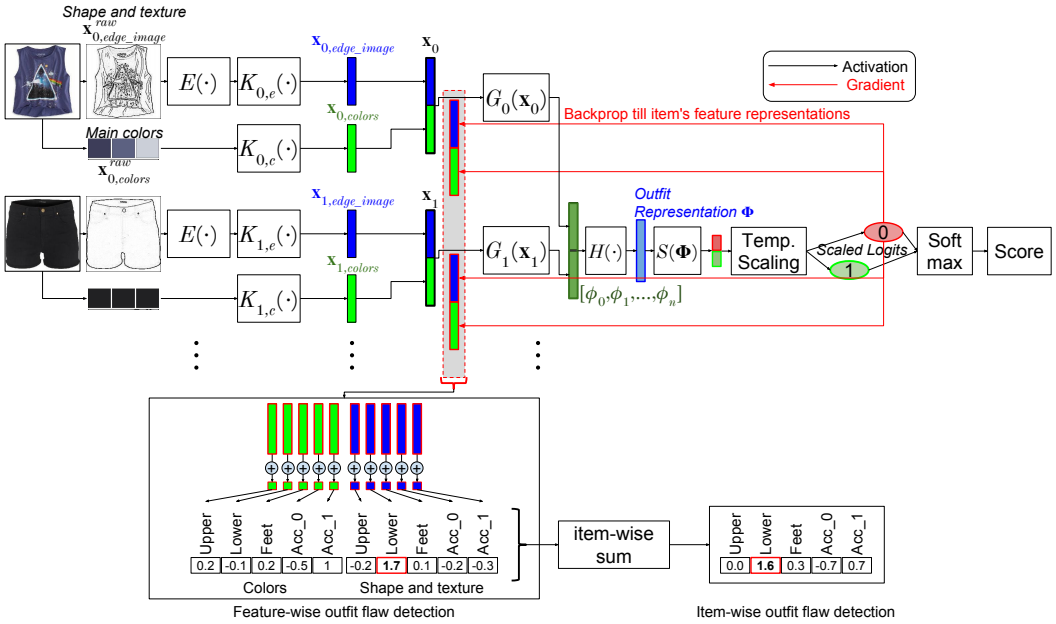

Figure 2: The overview of the proposed system. Given an outfit as a set of items, it extracts *edge_image* and main colors of each item. The *edge_image* is forward-propagated through a pretrained CNN E, then the output and main colors are forward-propagated through a series of concatenation and fully connected layers with ReLU (i.e., K, G and H) to obtain the score. The system also computes the gradient of the score (rigorously, the logit before softmax) with respect to the representation of each item through backpropagation. The gradients are multiplied with the corresponding features, yielding *Item Feature Influence Value* (IFIV). There is a single IFIV for each item-feature pair.

the grader for an outfit, i.e., why it classifies an input outfit as positive or as negative. For this purpose, we evaluate influence of each item and its features on the grader's judgment. The former (i.e., the influence of each item) provides item-level explanations, e.g., *this outfit is bad because of the inclusion of this particular item*. For this, we use the internal features (i.e., penultimate layer activation) that the grader uses. To further enable to obtain deeper explanations, we use human-interpretable features for the purpose, e.g., shape, texture, and colors extracted from the item images comprising the input outfit. To do this, we redesign the grader so that it can make judgments solely from these features.

3.1. Interpretable Item Features

The idea is to represent each item in terms of its *attributes* that are human-interpretable. We also rebuild the grader so that it can judge an input outfit from its attribute representation, and then attempt to explain its judgments according to influence of each attribute on the final score.

There are many candidate for this purpose, such as item type, brand, color, shape, texture, style etc. However, it may be a difficult task even for fashion experts to define such attributes determining the goodness of outfit. Moreover, we also need to be able to accurately predict those attributes

Figure 3: Item with their *edge_image* and three main colors.

from input item images, which will require costly annotation for training a proper model (e.g., a CNN). Additionally, the attributes need to be sufficiently rich so that the grader can properly judge goodness of outfits only from them.

Considering these requirements, we choose primitive image features that can be easily extracted from the item images: shape, texture, and colors. To be specific, we first divide contents of item images into color and non-color information. For the former, we extract three dominant colors from each image by finding clusters of pixels in color space. For non-color information, we first convert the image into gray-scale and then extract edges, which are expected to maintain shape and texture of the item. Figure 3 shows examples of original images, their *edge_image*, and three dominant colors. Their details are given below.

For colors, after removing background from the item im-

age, we apply K-mean clustering [20] to cluster all the pixels in the item image into three main colors in RGB color space. We use their centroids as three dominant colors of the item, yielding a 9-dimensional vector (3 colors $\times$ 3 RGB color values) for each item image. We denote it by $\mathbf{x}^{raw}_{i,colors}$, where the subscript i indicate that this is the color of the item that occupies i-th outfit part. In addition, since we use a zero-vector to represent absence of an outfit part, to enable to deal with outfits with a variable number of items, as in [38], we add 1 to all color values to avoid the conflict of a zero-vector with black color, resulting in the shift of the color value range from [0,1] to [1,2].

For shape and texture, we extract features in the following way. Let I be the input item image. We first apply the Canny edge detector [3] to I to obtain an edge map I_{e_1}. In parallel, we also apply a simple 3×3 filter f to I as $I_{e_2} = I * f$; f is defined as

$$f = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{1}$$

We add these two edge-like maps to obtain

$$I_e = I_{e_1} + I_{e_2}. \tag{2}$$

We call its black-white inverted version (i.e., $I_e \leftarrow 255 - clip(I_e, 0, 255)$) *edge_image* of I. We then use a pre-trained convolutional neural network (CNN) to extract an n-dimensional embedding of *edge_image*, which we denote by $\mathbf{x}^{raw}_{i,edge\_image}$, as

$$\mathbf{x}^{raw}_{i,edge\_image} = E(edge\_image) \tag{3}$$

where E is the CNN (up to its penultimate layer). We will use this as the representation of shape and texture of the item occupying the i-th outfit part.

The features $\mathbf{x}^{raw}_{i,colors}$ and $\mathbf{x}^{raw}_{i,edge\_image}$ obtained as above are transformed by a trainable item-feature encoders $K_{i,c}$ and $K_{i,e}$ into item-feature encodings $\mathbf{x}_{i,colors}$ and $\mathbf{x}_{i,edge\_image}$ respectively. We use a stack of a few fully-connected layers for $K_{i,c}$ and $K_{i,e}$ each. Finally, we concatenate them together and denote the resultant vector by $\mathbf{x}_i = [\mathbf{x}^\top_{i,edge\_image}, \mathbf{x}^\top_{i,colors}]^\top$, which gives a representation of an item.

3.2. Outfit Grader

Our outfit grader is basically the same as the one proposed in [38] except the representation of items described above. We summarize its design here. The input is an outfit consisting of n items, each of which occupies a different part. Given the feature of an i-th part item as mentioned above, our grader first transforms it by a trainable item encoder G_i as

$$\phi_i = G_i(\mathbf{x}_i). \tag{4}$$

We use a stack of a few fully-connected layers for G_i. The representations of n items are then concatenated and transformed to the representation $\mathbf{\Phi}$ of the entire outfit as

$$\mathbf{\Phi} = H([\phi_0, \phi_1, \ldots, \phi_n]), \tag{5}$$

where H is a trainable outfit encoder, for which we employ a single fully-connected layer (followed by BN and ReLU).

The grader performs binary classification on the representation $\mathbf{\Phi}$ of the input outfit O. To do this, the outfit representation is transformed by a single fully-connected layer S to two logits $\mathbf{s} = [s_{pos}, s_{neg}]$ as $\mathbf{s} = S(\mathbf{\Phi})$. Then they are normalized by softmax to yield scores for positive and negative classifications. Denoting the score for O being positive by $F(O)$, it is given by

$$F(O) = \sigma_{pos}(\mathbf{s}) = \frac{\exp(s_{pos})}{\exp(s_{pos}) + \exp(s_{neg})}. \tag{6}$$

For the CNNs extracting item features (e.g., $\mathbf{x}_{edge\_image}$), we use those pretrained on other tasks such as object recognition. Thus, the learnable parameters in the grader are in $K_{i,e}$, $K_{i,c}$, G_i, H, and S. They are learned by minimizing a cross-entropy loss on training data consisting of pairs of outfit O and the ground-truth label (i.e., positive or negative).

Calibration of Outfit Scores It is known [10] that modern deep neural networks employing softmax for multi-class classification tend to be over-confindent, that is, the score of the predicted class, or *confident* (i.e., the max of softmax outputs), tends to be large and even close to one, even if the prediction is wrong. We found that this is exactly the case with our implementation of the outfit grader [38]. A simple but effective method to alleviate this overconfidence is to perform calibration of the softmax outputs using temperature scaling [10,28]. To be specific, we replace $\mathbf{s}$ in the softmax (6) with $\mathbf{s}/T$. T is determined using validation samples so that the resulting score $F(O)$ is as close to classification accuracy as possible; then the score will better represent confidence of the prediction. We use $\hat{q} = 100 \cdot F(O)$ (in percent) as the fashionability score of an outfit O.

3.3. Item Feature Influence Value (IFIV)

Suppose that we input an outfit to the above grader and receive its judgment. To explain the judgment, we evaluate influence of each feature of each item. If the judgment is negative and a particular feature of an item has large influence on it, we regard that feature of the item to be the reason for the negativity; the same is true for a positive judgment.

To be specific, we define the influence on the logit s_c ($c \in \{neg, pos\}$) of a feature $f (\in \{edge\_image, colors\})$ of i-th item, denoted by $\mathbf{x}_{i,f}$, as follows. We first compute

$$\mathbf{g}_{i,f} = \mathbf{x}_{i,f} \odot \frac{\partial s_c}{\partial \mathbf{x}_{i,f}}, \tag{7}$$

where $\odot$ is element-wise multiplication. Note that the logit s_c here is the temperature-scaled version mentioned above.

A similar method is used for visualization of CNNs for object classification, where the pixel-wise multiplication of an input image and the gradient of a class score with respect to its pixels is used to show which part positively or negatively affects the score and which part has no influence on it. As we consider influence of only each feature, not its element, we compute the sum over all its elements as

$$IFIV_i = \sum_f IFIV_{i,f}, \tag{8a}$$

where

$$IFIV_{i,f} = \sum_k g_{i,f,k}, \tag{8b}$$

where $g_{i,f,k}$ is the k-th element of $\mathbf{g}_{i,f}$. Figure 2 shows the diagram explaining how *Item Feature Influence Value (IFIV)* of each item feature is computed.

4. Evaluation of the Outfit Grader

4.1. Prediction Accuracy vs. Interpretability

We redesign the outfit grader for the purpose of improved explanability. The original model [38] is designed to be an end-to-end model receiving raw item images as inputs, aiming at the best prediction accuracy of outfit quality. Our redesigned model receives hand-engineered features extracted from item images for the sake of explanability. This will sacrifice accuracy of outfit quality prediction. We conducted experiments to examine this.

Model architecture We compare two models that differ only in the item representation $\mathbf{x}$. One is the model we described in Sec. 3. The other is a baseline model, which uses a CNN feature directly extracted from RGB item images; to be specific, the feature of the i-th part item is given by $\mathbf{x}_i = E(RGB\_image)$, where E is a pretrained CNN that is the same as the one used to extract $\mathbf{x}^{raw}_{i,edge\_image}$. The configurations and parameters that are shared by the two models are as follows:

- For the feature extractor E, we employ ImageNet-pretrained InceptionV3 [36]. The activation of *pool5* layer for an input item image is used for $\mathbf{x}$, which forms a 2048-dimensional vector.
- An identity function is used for item-feature encoders $K_{i,e}$, $K_{i,c}$ and item encoder G_i.
- A single fully-connected layer with 4096 units is used for the outfit encoder H, followed by batch normalization [14] and ReLU [26] activation function.
- The both models are trained for 50 epochs with learning rate $1e-4$ and batch size 256 on Polyvore409k dataset [38].

Table 1: Accuracy and average f1 of a baseline and an interpretable outfit grader on Polyvore409k dataset [38].

Partition	Metric	Outfit Grader	
		Baseline	Interpretable
Train	Acc.	98.41	99.04
	Avg. F1	98.20	98.92
Validation	Acc.	83.19	80.23
	Avg. F1	81.86	79.06
Test	Acc.	79.19	76.36
	Avg. F1	74.11	71.42

Figure 4: Eight best (upper) and worst (lower) outfits from testing partition of Polyvore409k dataset according to our outfit grader.

Results Table 1 shows the results. Accuracy indicates that of binary classification, where a prediction is considered to be correct if it matches the ground truth. As expected, the baseline model shows better performance than the interpretable model by 2.83% accuracy and 2.69% average f1. This is a noticeable gap but is arguably not so large to make the explanation by the interpretable model meaningless.

Configuration of Outfit Grader To recover the performance drop as much as possible and further achieve better prediction accuracy, we tested a number of configurations of the interpretable grader. To be specific, we tested different configurations of the item-feature encoder $K_{i,c}$ and $K_{i,e}$, the item encoder G_i and the outfit encoder H. The configurations and their performance on testing samples are shown in Table 2. Since the model #3 has the best performance, we will use this model for the experiments on explainability using feature influence values. Figure 4 shows examples of judgments of the grader; outfits with the highest score and those with the lowest scores.

4.2. Effect of Calibration of Score (Confidence)

As mentioned in Sec. 3.2, we employ the temperature scaling to calibrate the outfit score (or confidence) $\hat{q}$. Figure 5 shows the reliability diagrams [6, 27] before and after

Table 2: Testing accuracy and average f1 of various configurations of outfit grader after training for 50 epochs of Polyvore409k dataset [38]. Each cell in the "Item-feature Encoder $K_{i,c}$, $K_{i,e}$", "Item Encoder G_i", and "Outfit Encoder H" columns specify the size of the fully-connected layer The $\times$ indicates a stack of multiple layers.

#	Item-feature Encoder $K_{i,c}$, $K_{i,e}$	Item Encoder G_i	Outfit Encoder H	Acc.	Avg. F1
1	-	-	4096	76.36	71.42
2	128	1024	2048	80.19	75.76
3	**1024**	**1024**	**2048**	**80.75**	**76.76**
4	128	128	128	77.56	71.61
5	128×64	512×256	2048	80.05	75.70
6	128×64×32	512×256	2048	79.04	75.84

the calibration. Searching for the best value for the temparature T on the validation samples yielded $T = 6.77$. To do this, we split all the testing samples into 10 bins with an equal width, using which we plot the expected accuracy of samples in each bin against the average confidence from the outfit scores. A perfectly calibrated model will yield an identity relation between them. We also calculated expected calibration error (ECE) [25], the difference in expectation between confidence and accuracy. ECE is reduced from 11.32 and 14.97 before the calibration to 0.92 and 0.46 after calibration for validation and testing partition of Polyvore409k dataset [38] respectively. Figure 6 shows distributions of outfit scores for samples with positive labels and those with negative labels. The distributions with the temperature scaling clearly have a much wider spread, making the score more meaningful. We can conclude from Figs. 5 and 6 that the temperature scaling is able to calibrate the outfit scores.

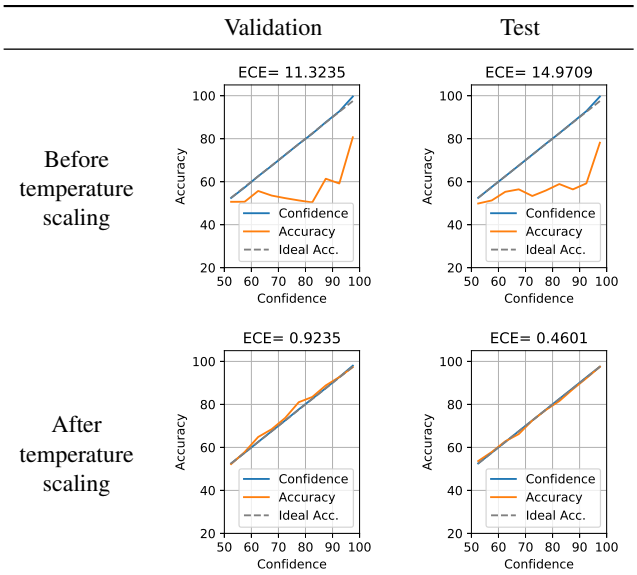

Figure 5: Reliability diagrams and ECE values before and after temperature scaling for validation and testing partition of Polyvore409k dataset [38]. Confidence is equivalent to the outfit score.

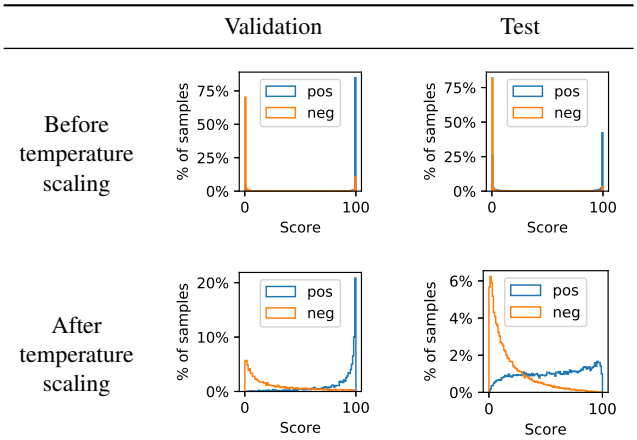

Figure 6: Distribution of outfit scores before and after temperature scaling for positive and negative samples in validation and testing partition of Polyvore409k dataset [38].

5. Experimental Results

We conducted experiments to evaluate the proposed method for explaining judgment of the outfit grader. For the grader, we used the *1024-1024-2048* model from Table 2.

5.1. Experimental Design

Suppose that an outfit is bad (i.e., not fashionable) due to a single item contained in it. There should also be a reason why the item does not match the outfit and makes it bad, e.g., because of its incompatible color or its unmatched shape and texture. We want to identify the item as well as the reason for the bad outfit.

Based on the proposed framework, this is formulated as a task of identifying the item-feature pair that has the most negative influence on an input outfit. We apply the proposed method to this task and evaluate its performance.

For this purpose, we create a set of negative outfits from positive ones in the dataset in the following way. For a positive outfit, we choose an item from those contained in it and then replace its feature $f(\in \{edge\_image, colors\})$ and ensure that the replacement does decrease the outfit score. Note that we are interested here not in the correctness of the judgment of the outfit grader but in how well its judgment can be explained, more precisely, accuracy of the proposed method identifying the item-feature pair lowering the score. Detailed procedures for the creation of data are as follows:

1. 1,000 base outfits with the highest scores are chosen

Table 3: Statistics of the base samples and the negative samples created from them. The three types of negative samples, i.e., *edge_image*-wise, *colors*-wise, and item-wise, have identical statistics by their construction.

Sample type	Number of samples containing following			
	outfit parts		number of items	
Base sample	Outer	205	3 items	14
	Upper	682	4 items	98
	Lower	715	5 items	396
	Full	330	6 items	383
	Feet	967	7 items	107
	Accessory0	986	8 items	2
	Accessory1	901	**Total**	**1,000**
	Accessory2	691		
Outfit flaw detection sample	Outer	2,050	3 items	420
	Upper	6,820	4 items	3,920
	Lower	7,150	5 items	19,800
	Full	3,300	6 items	22,980
	Feet	9,670	7 items	7,490
	Accessory0	9,860	8 items	160
	Accessory1	9,010	**Total**	**54,770**
	Accessory2	6,910		

from the test partition of Polyvore409k dataset [38]. Their average score is 98.37 (out of 100).

2. For each item and its feature f in each base outfit, we create 10 mod samples in the following way:

   2.1 500 mod samples are first created by changing the item-feature f in the base sample. In the case of $edge\_image$, we replace it with that of other item occupying the same part of an outfit randomly chosen from the test partition of the dataset. In the case of $colors$, we replace it with random colors.

   2.2 Their scores are computed by the outfit grader and the worst ten samples are selected and all the others are discarded.

Step 2.2 ensures that the grader gives low scores to the created outfits with a replaced item-feature pair. For the two features of *edge_image* and *colors*, the above procedure produces two datasets, which we call *edge_image*-wise and *colors*-wise samples, respectively. Additionally, we create "item-wise" samples by replacing the entire item in Step 2.1. An example of created negative samples is shown in Fig. 8. The statistics of the base samples and the three types of negative samples are shown in Table 3. The distributions of scores for these samples are shown in Fig. 7.

5.2. Results

We apply our method to the three types of samples created as explained above. To be specific, inputting each

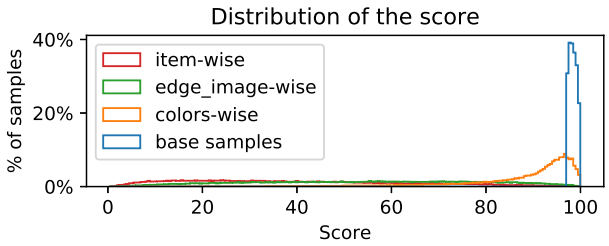

Figure 7: The distribution of scores of each type of samples.

sample to the grader, which yield a lower score as explained above, we compute *IFIV*s for the score defined in (8). We then find the part with the minimum *IFIV*, or equivalently, that the maximum negative *IFIV* over all features $f(\in \{edge\_image, colors\})$ as

$$i^* = \arg\max_{i,f}(-IFIV_{i,f}). \tag{9}$$

We regard the prediction i^* as correct if it matches the true item, which is the replaced one when creating the negative sample. Figure 8 shows examples of IFIVs for different types of samples. It is seen that the replaced item-feature pairs yield high negative IFIVs, meaning that our method can successfully detect the item lowering the outfit score with the reason why it is bad (i.e., the feature lowering the outfit score).

Table 4 show the performance over all the samples. The proposed method can detect the replaced items for *item*-wise samples with 99.51% accuracy and those for *edge_image*-wise samples with 98.99% accuracy, respectively. The accuracy for *colors*-wise samples is 81.83% and is lower than the others. This is due to the fact that the scores of the *colors*-wise samples tend to be higher and their gap to the original outfits are smaller than the other two types, as shown in Fig. 7. That said, this is fairly good considering the chance rate. Note that for the samples of *edge_image*-wise and *colors*-wise, it is necessary to predict both the feature and the item correctly.

Table 5 shows accuracy values for different numbers of items. They are quite consistent for *item*- and *edge_image*-wise samples, except for the outfit with eight items. Note that there is only two out of 1,000 base samples that has eight items, as shown in Table 3, and thus the performance for eight items could be statistically unreliable. For *colors*-wise samples, there is a tendency that the accuracy decreases as the number of items increases.

Table 6 shows accuracy values calculated for each part of outfits. It is seen that for *item*- and *edge_image*-wise samples, the performance are almost the same across all outfit parts, except the full outfit part showing slightly lower accuracy. For *colors*-wise samples, the accuracies are lower the other two types and are somewhat different for different parts.

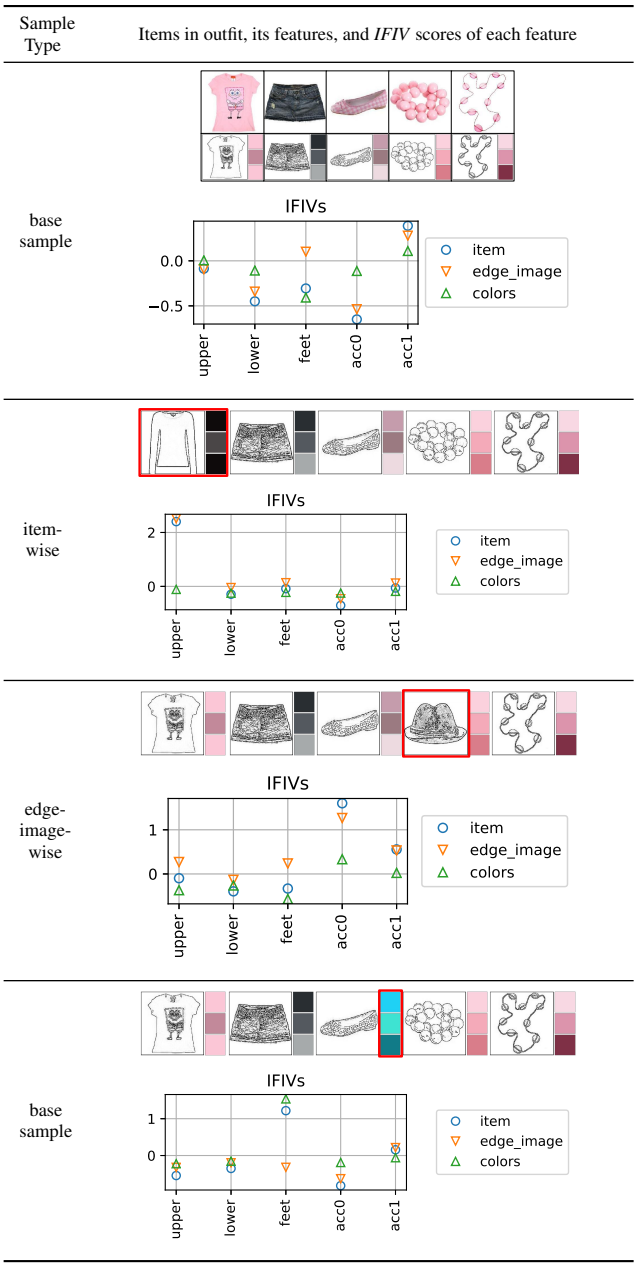

Figure 8: An example of computation of IFIVs. The red boxes indicate the replaced entities from the original high-quality outfits, which makes the new outfits have low outfit scores. "IFIV score" means negative IFIV value.

6. Conclusion

In this paper, we have proposed a novel method for item-feature-wise explanation of outfits. The method can quantify the effect of interpretable features of each item on the goodness of an outfit with the proposed *Item Feature Influence Value (IFIV)*. It does not need any item-level attribute annotation. Using the *IFIV* of each item-feature pair in an outfit, we can detect the bad item in an outfit lowering its score by finding the item-feature pair with the maximum negative *IFIV*. The experiments have shown that our method can detect the bad items at 99.51, 98.99, and 81.83%, for datasets of item-wise, *edge_image*-wise, and *colors*-wise samples, respectively.

Table 4: Overall accuracy (%) of detection of replaced item-feature pairs.

Method	Sample type	Prediction accuracy
Random	item-wise	18.26
	feature-wise	9.13
Proposed method	item-wise	99.51
	edge_image-wise	98.99
	colors-wise	81.83

Table 5: Accuracy (%) of replaced item-feature detection for different numbers of items contained in each outfit. The *By chance* column shows the chance rate for feature-wise samples.

Number of items	By chance	Proposed method (by sample type)		
		item	*edge_image*	*colors*
3	16.67	95.71	95.71	76.43
4	12.50	99.90	97.37	86.91
5	10.00	99.72	98.94	85.39
6	8.34	99.51	99.26	79.57
7	7.15	99.39	99.57	76.92
8	6.25	80.00	86.25	86.25

Table 6: Accuracy (%) of replaced item-feature detection classified by different outfit parts. Note that there are eight outfit parts in Polyvore409k dataset; the *By chance* column shows the chance rate for feature-wise samples.

Outfit part	By chance	Proposed method (by sample type)		
		item	*edge_image*	*colors*
outer	7.77	100.00	99.66	58.93
upper	8.58	99.75	99.96	57.95
lower	8.59	99.40	99.36	68.20
full	9.93	96.36	87.70	66.36
feet	8.87	99.65	99.38	90.91
accessory0	8.88	99.68	99.69	94.07
accessory1	8.72	99.76	99.99	89.39
accessory2	8.49	100.00	99.99	93.70

Acknowledgements

This work was partly supported by JSPS KAKENHI Grant Number JP15H05919 and JP19H01110 and JST CREST Grant Number JPMJCR14D1.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015. 2

[2] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. *arXiv preprint arXiv:1704.05796*, 2017. 2

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986. 4

[4] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 2

[5] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 2

[6] M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of forecasters. *The Statistician*, pages 12–22, 1983. 5

[7] Z. Feng, Z. Yu, Y. Yang, Y. Jing, J. Jiang, and M. Song. Interpretable partitioned embedding for customized multi-item fashion outfit composition. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 143–151, 2018. 1, 2

[8] H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, pages 2296–2304, 2015. 2

[9] A. Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012. 2

[10] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017. 4

[11] X. Han, Z. Wu, Y.-G. Jiang, and L. S. Davis. Learning fashion compatibility with bidirectional lstms. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 1078–1086, 2017. 1, 2

[12] 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*, pages 770–778, 2016. 2

[13] Y. Hu, X. Yi, and L. S. Davis. Collaborative fashion recommendation: a functional tensor factorization approach. In *Proceedings of the 23rd ACM International Conference on Multimedia*, pages 129–138, 2015. 1, 2

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[15] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4565–4574, 2016. 2

[16] Y. Li, L. Cao, J. Zhu, and J. Luo. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Transactions on Multimedia*, 2017. 1, 2

[17] Y. Lin, P. Ren, Z. Chen, Z. Ren, J. Ma, and M. de Rijke. Explainable fashion recommendation with joint outfit matching and comment generation. *arXiv preprint arXiv:1806.08977*, 2018. 2

[18] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016. 2

[19] S. Liu, J. Feng, Z. Song, T. Zhang, H. Lu, C. Xu, and S. Yan. Hi, magic closet, tell me what to wear! In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 619–628, 2012. 1, 2

[20] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 4

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 2

[22] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–9, 2015. 2

[23] K. Matzen, K. Bala, and N. Snavely. StreetStyle: Exploring world-wide clothing styles from millions of photos. *arXiv preprint arXiv:1706.01869*, 2017. 2

[24] 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, 2015. 2

[25] M. P. Naeini, G. F. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2901–2907, 2015. 6

[26] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010. 5

[27] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 625–632, 2005. 5

[28] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999. 4

[29] M. Ren, R. Kiros, and R. Zemel. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961, 2015. 2

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015. 2

[31] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016. 2

[32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017. 1, 2

[33] E. Simo-Serra, S. Fidler, F. Moreno-Noguer, and R. Urtasun. Neuroaesthetics in fashion: Modeling the perception of fashionability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2015. 1, 2

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2

[35] D. Smilkov, N. Thorat, B. Kim, F. Vigas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 1

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 2, 5

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1

[38] P. Tangseng, K. Yamaguchi, and T. Okatani. Recommending outfits from personal closet. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, pages 269–277, 2018. 1, 2, 4, 5, 6, 7

[39] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015. 2

[40] X. Zhang, J. Jia, K. Gao, Y. Zhang, D. Zhang, J. Li, and Q. Tian. Trip outfits advisor: Location-oriented clothing recommendation. *IEEE Transactions on Multimedia*, 19(11):2533–2544, 2017. 1

[41] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016. 2