Deep Learning for Fashion: Compatibility Evaluation, Grading, and Recommendation of Outfits

TOHOKU UNIVERSITY
Graduate School of Information Sciences

Deep Learning for Fashion: Compatibility Evaluation, Grading, and Recommendation of Outfits
(ファッションのための深層学習：服装の統一性評価と格付けおよび推薦)

A dissertation submitted to the department of
System Information Science in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy
in
Information Sciences

by

TANGSENG Pongsate (ID No.: B5ID2503)

January 15, 2019

Deep Learning for Fashion: Compatibility Evaluation, Grading, and Recommendation of Outfits

TANGSENG Pongsate (ID No.: B5ID2503)

Abstract

Clothing and fashion are essential parts of the everyday life of everybody, as good dressing does not only improves comfort but also plays an important role in the image of the wearer. Although fashion is a subjective subject, there are still standards and guidelines to follows according to various factors such as climates and cultures. Such guidelines can be found indirectly from outfit images or posts on the Internet.

Together with a large amount of data, the unprecedented processing power of modern computers and graphic processing units (GPU), and the knowledge of artificial intelligence and machine learning, we could build an intelligent system to study clothing trend faster and more efficient than ever. This leads us to focus on a few key topics in this dissertation - semantic segmentation for fashion to identify the clothing items in the human body, outfit quality measurement to quantify the outfit quality according to the large amount of fashion outfit data which can also be used for outfit recommendation, and, finally, identifying attributes of clothing items that influence the overall quality of a particular outfit.

In the first part of this work, we consider the semantic segmentation on fashion domain, which can be called *clothing parsing*. For clothing parsing, the item categories are one of the clothing items, such as *t-shirt*. Clothing parsing distinguishes itself from the general object or scene segmentation problems in that objects that look locally very similar can be in completely different categories such as *skirt* and a part of *dress*. We extend fully-convolutional neural networks (FCN) with a side-branch network which we refer as outfit encoder to predict a consistent set of clothing labels to encourage combinatorial preference, and with a conditional random field (CRF) to explicitly consider coherent label assignment to the given image. The empirical results using Fashionista and CFPD datasets show that our model achieves state-of-the-art performance in clothing parsing, without additional supervision during training. We also study the qualitative influence of annotation on the current clothing parsing benchmarks, with our Web-based tool for multi-scale pixel-wise annotation and man-

ual refinement effort to the Fashionista dataset. Finally, we show that the image representation of the outfit encoder is useful for dress-up image retrieval application.

For the second part of this work, we consider grading a fashion outfit for recommendations, where we assume that users have a closet of items and we aim at producing a score for an arbitrary combination of items in the closet. The challenge in outfit grading is that the input to the system is a bag of item pictures that are orderless and vary in size. We build a deep neural network-based system that can take variable-length items and predict a score. We collect a large number of outfits from a popular fashion sharing website, *Polyvore*, and evaluate the performance of our grading system. We compare our model with a random-choice baseline, both on the traditional classification evaluation and on people's judgment using a crowdsourcing platform. With over 84% in classification accuracy and 91% matching ratio to human annotators, our model can reliably grade the quality of an outfit. We also build an outfit recommender on top of our grader to demonstrate the practical application of our model for a personal closet assistant.

To increase transparency and trustworthy of the reported outfit quality value, the third part of this work focus on explaining the reason behind the outfit quality prediction. In particular, we proposed a gradient-based method with an interpretable feature extraction to identify the feature of items in the outfit that positively and negatively influences the outfit's overall quality. The challenge is not only on explaining the outfit quality but also on evaluating such explanations. Here, we transform the problem of outfit quality explanation into outfit flaw detection by focusing on item-features that influence outfit's quality in a negative way. Our experiment shows that our system can detect the flaw in our testing samples effectively at 99.52%, 99.48%, and 85.37% item-wise, *shape-and-texture*-wise and color-wise, respectively.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter aims to discuss the motivation and overview of the problems studied in this dissertation and summarizes the contribution of each chapter. Since this dissertation focuses on the applications of deep learning on fashion, this chapter also provides preliminary knowledge in deep learning domain in term of biological motivation and basic modules in artificial neural networks.

1.1 Motivation

Fashion is a very large industry. It is estimated that the value of the global fashion industry in 2016 is 3 trillion US dollars, which occupied 2% of the global GDP [5]. Clothing, in general, is also an essential part of everyday life as good dressing not only improves the comfort of the wearer, it also improves the person's image. Since it is the first thing that people will notice about that person, and the first impression also last forever [6], it is important to dress well.

Automatic outfit recommendation systems can come in handy in this regard. A

system that knows how to dress well can not only help a person to dress nicely, but also save time that could be spent choosing outfits. In addition, this system can help to find new item combinations from existing items which improve reusability, save money, less clothing waste, and create a more sustainable society as a whole.

Although fashion is a subjective topic and subject to change over time, there are still standards or guidelines according to various factors such as occasions, seasons, climates, locations, and cultures. Such guidelines can be studied indirectly from outfit images and other kinds of posts from the Internet.

In this work, we aim to study fashion preferences from online sources and develop an effective outfit recommendation system that can also point out the flaw in an outfit in feature level.

1.2 Overview of the problem

To create a fashion recommendation system that can point out the flaw in the outfit, we study four essential components:

1. Fashion items in outfits.
2. A system to measure the fashionability score of an outfit.
3. A system that can recommend outfits from a pool of items.
4. A system that can point out flaws in an outfit at the feature level, so the outfit can be improved efficiently.

1.2.1 Extracting Fashion Items in an Outfit

For extracting fashion items in an outfit image, we can apply a computer vision technique that used to solve semantic segmentation to the outfit image. Semantic

segmentation is a well-known computer vision problem. The goal of this problem is to segment the image base on the semantic. In the other word, we want to give every pixel in the image a semantic label, such as cars, roads, humans, etc.

Clothing parsing is a specific form of semantic segmentation, where the labels are one of the clothing items, such as *t-shirt*. It has been actively studied in the vision community [7–11] because of its tremendous value in the real-world application. Clothing parsing has a specific property that general object or scene segmentation problems do not have: the fine-grained clothing categories require higher-level judgment based on the semantics of clothing and the deforming structure within an image. What we refer the semantics here is the specific type of clothing combination people choose to wear in daily life. For example, people might wear a *dress* or separate a *top* and a *skirt*, but not both of them together. However, from a recognition point of view, both styles can look locally very similar and can result in false positives in segmentation, as shown in figure 1.1. Such combinatorial preference at the semantic level [12–14] introduces a unique challenge in clothing parsing where a bottom-up approach is insufficient to solve the problem [15].

In this dissertation, we approach the clothing parsing problem using fully convolutional neural networks (FCN). FCN has been proposed for general object segmentation [1] and shown an impressive performance thanks to the rich representational ability of deep neural networks learned from a huge amount of data. To utilize FCN in clothing parsing, we need to take the above clothing-specific challenges into account, as well as care to address the lack of training data for learning large neural networks. Based on the FCN architecture, we propose to extend the parsing pipeline by 1) a side-branch that we call *outfit encoder* to predict the combinatorial preference of gar-

Figure 1.1: Combinatorial preference in clothing parsing: dress and skirt are in the exclusive relationship, yet independent pixel-wise prediction cannot encode such knowledge and results in mixture of patches (FCN-8s [1]). We propose the side-path outfit encoder and CRF alongside the segmentation pipeline to address the issue.

ments for dealing with semantics-level consistency, and 2) conditional random field (CRF) to consider both semantics and appearance-level context in the prediction. Experimental results show that starting from a pretrained network, we are able to learn the outfit encoder and finetune the whole segmentation network with a limited amount of training data, and our model achieves the state-of-the-art performance in the publicly available Fashionista dataset [7] and Colorful Fashion Parsing dataset (CFPD) [11].

We also study the qualitative issue in the current clothing datasets. The existing benchmarks suffer from erroneous annotations due to the limitation in the superpixel-based annotation, as well as from the ambiguity of labels [9]. We develop a Web-based tool to interactively annotate pixels at multiple scales and study how much influence

we have on the performance metrics by manually refining the Fashionista dataset [7].

The outfit encoder learns a compact representation of the combinatorial clothing preference through the training of segmentation pipeline. We find that the resulting internal representation of the encoder is suitable for style retrieval application. The learned representation compactly encodes the gist of the dress-up style of the picture, and when used in retrieval, the representation is able to find semantically similar clothing style (e.g., dress only or shirt + skirt combination), even if the low-level appearance cues such as color or texture look different.

1.2.2 Measuring the Fashionability Score of an Outfit and Outfit Recommendation System

Although Identifying the items that people are wearing is helpful, the statistic of individual items is not the only information we want to learn from data. Since people wear many items together to create outfits, to create an outfit recommendation system, knowing what kind of items that are often worn together can also be very important. Recently, Amazon announced their automatic style assistant called "Echo Look™". Although the underlying mechanism is not published, emerging commercial applications confirm the ripe of computer vision applications in fashion. In this dissertation, we purpose to create an outfit fashionability measurement system that learns from a large number of outfits to know the good and bad item combinations (Figure 1.2).

Previous works in outfit evaluation can be divided into two groups based on the input format: a worn outfit as a full-body picture as in [16–19], and as a set of images of items [20, 21], or a combination of both [22]. Each outfit can have an arbitrary

Figure 1.2: Given an arbitrary number of items, our goal is to evaluate the quality of the outfit combination.

number of items. For examples, in one day, one might prefer a combination of a jacket, a t-shirt, and jeans, while in the another she might want to wear a dress. Our goal is to build a machine learning system that accepts variable numbers of items yet produce a consistent score for any size of combinations.

Although previously we use clothing parsing to extract fashion items from an outfit image, we found that the number of outfit images in the clothing parsing dataset is very low compared to the general object or scene segmentation datasets, and we could use the outfit data where the items in each outfit have its own image. As a result, we collect a large number of outfit data from a popular fashion website `polyvore.com` and attempt to measure the fashionability of an outfit as a bag of fashion items.

Our Polyvore409k dataset consists of 409,776 sets of clothing items from 644,192 unique items. The dataset forms a large bipartite graph of items and outfits. We partition the dataset into training and testing sets such that there is no overlapping nodes and edges between the sets, and use them measure the classification performance. We also conduct a human study using crowdsourcing to assess predicted scores against human judgments and show our model closely resembles human behavior. Using our grader, we build an outfit recommendation system that takes clothing items as input

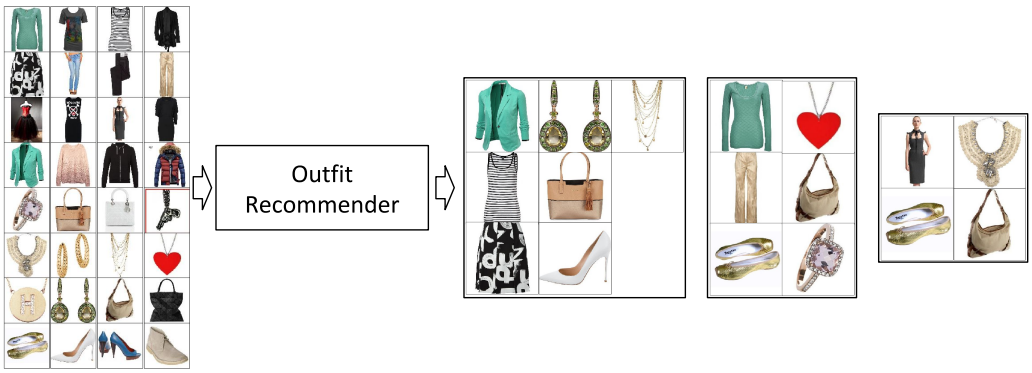


Figure 1.3: Given a pool of items, our goal is to recommend outfits from items in the pool.

Figure 1.4: The goal is to identify the flaw in an outfit in both item-level and feature-level. Each row is an outfit consists of variable number of items. In the first row, the flawed item is identified. In the second and third row, the flaw is identified to the feature-level, so that item can be changed accordingly.

and recommends the best outfits from the given items as shown in figure 1.3.

1.2.3 Feature-Level Outfit Flaw Detection

Although the outfit grader can tell the fashionability of an outfit in a form of the score, to help the user improve the outfit, we aim to identify the flaw in the outfit so

the user can change accordingly. This aim is illustrated in figure 1.4.

Recently there are many approaches to applying computer vision techniques on outfit fashionability measurement that also provide reasons for the given score [23,24]. However, their approaches need to be trained using a large amount of manually annotated data, which is expensive in both time and money, and since their dataset is not publicly available, the usefulness of the approaches is limited. Here, we purpose an outfit fashionability measurement system that can predict the fashionability level reliably, that is also able to explain the reason behind the predicted fashionability score at the interpretable feature-level, by giving the numerical influence level of each feature in each item to the overall outfit quality both positively and negatively (*Item-Feature Influence Value (IFIV)*). In addition, our system is trained on a publicly available dataset without any additional annotation.

Since evaluating the explanation via human experiment is always subjective, to even a greater extent in the fashion topic, we also propose an evaluation protocol as outfit flaw detection, with necessary testing samples from an existing dataset to quantify the performance of the explanation. Because our system gives *Item-Feature Influence Value (IFIV)*, we can use the *negative IFIV* that show how each item-feature negatively influence the outfit quality to identify the flaw in the outfit and use the predicted item-feature to evaluate the performance of the system.

1.3 Contributions of Individual Chapter

We summarize the contribution of each chapter as follows:

Contribution of chapter 2 In this chapter, the side-path outfit encoder for the FCN architecture, and together with CRF to improve segmentation performance in clothing parsing are proposed. The evaluation shows that the proposed model achieves state-of-the-art performance in clothing parsing. A Web-based tool to interactively annotate pixels and study the qualitative influence in the segmentation benchmarks is also developed. Using the tool, the Fashionista benchmark [7] is manually annotated with high-quality and less ambiguous labels, which is referred as Refined Fashionista dataset. The tool and annotation is released to the public for future research. In addition, a preliminary study is conducted as it shows that the outfit encoder is also useful for retrieval since the encoder representation compactly encodes the combinatorial preference of clothing items, and suitable for retrieving semantically similar styles.

Contribution of chapter 3 In this chapter, the Polyvore409k dataset is built. It contains 409,776 outfits and 644,192 items, and every outfit in this dataset is guaranteed to covers the entire body while having a variable numbers of items. An outfit grader that produces a score for fashion outfits with a variable number of items is also purposed. An empirical study shows that the model achieves 84% of accuracy and precision in Polyvore409k dataset. In addition, a human judgment framework on outfit quality is also purposed. It provides a simple and reliable method to verify the reliability of outfit verifiers using a crowdsourcing platform. Finally, based on the outfit grader, a outfit recommendation system that is able to suggests good outfits from a pool of items is build to demonstrate a real-world application of the outfit grader.

Contribution of chapter 4 In this chapter, a method to extract interpretable features from item images in outfits without using any additional annotation is purposed. The outfit fashionability measurement system with item-feature influence values is built. Given an outfit as a set of item with associated outfit parts, this system predicts how each item-feature affects the overall outfit quality as *Item-Feature Influence Values (IFIV)*. These values can be used to identify the flaw in an outfit, which not only explain the reason behind the output outfit quality, but also guide the user to dress better. Finally, to evaluate the outfit flaw detection system, a method to create outfit flaw detection samples is purposed.

1.4 Preliminaries

This section provides preliminary knowledge in deep learning domain including motivation behind artificial neuron network and commonly used modules in constructing more complex neural network architecture.

1.4.1 From Biological Motivation to an Artificial Neuron

The ANN, as the name implies, is inspired by the actual biological neural system in brains. The biological neural system is a network consists of many neurons connected together. Each neuron takes electrical signals from other neurons as inputs via dentrites, then processes them according to learnable synaptic strengths that control the influence magnitude of each input, and fires the neural impulse through a myelinated axon to axon terminals if the total signal strength is strong enough, which pass the impulse to other neurons. Each neuron in ANN consists of inputs x_i, weights w_i which equivalent to learnable synaptic strengths, and an activation function f, as

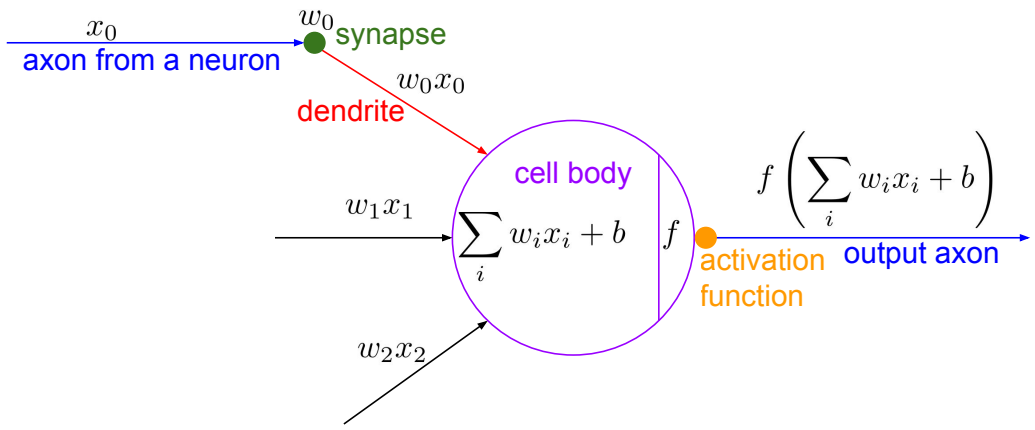


Figure 1.5: Illustration of a mathematical model of an artificial neuron.

illustrated in figure 1.5. The output of an artificial neuron is:

$$y = f\left(\sum_i w_i x_i + b\right) \tag{1.1}$$

1.4.2 Activation Functions

The activation function f takes a single number and perform a fixed mathematical operation on it. The commonly used activation functions are:

- **Sigmoid** $\sigma(x) = 1/(1 + e^{-x})$ which takes a real-valued number and output it into range [0,1].
- **ReLU.**, stands for Rectified Linear Unit, $ReLU(x) = max(0, x)$, is simple threshold the input at zero. It is very popular since Krizhevsky et al. [25] found that it greatly accelerates the learning process of the network.

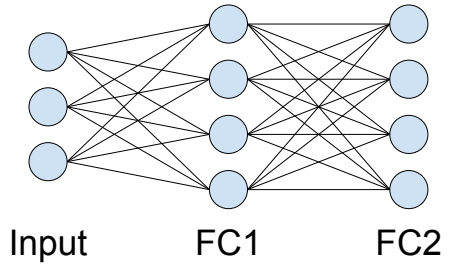


Figure 1.6: An illustration of 2 fully connected layers.

1.4.3 Fully Connected (FC) Layer

A fully connected layer is a layer of neuron. Each neuron in a FC layer takes input from (often called *connected to*) all outputs of the previous layer. Let an n dimensional vector $\mathbf{x} \in \mathbb{R}^n$ be an input variable, where n is also a number of outputs of the previous layer, a FC layer can be formulated by

$$\mathbf{h} = \mathbf{w}\mathbf{x} + \mathbf{b} \tag{1.2}$$

where $\mathbf{h} \in \mathbb{R}^m$ is an output variable, and $\mathbf{w} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are weight and bias parameters, respectively. Figure 1.6 illustrates two fully connected layers with 4 nodes each, where n and m of the first FC layer are 3 and 4, while those of the second layer are 4 and 4, respectively.

1.4.4 Convolutional Layer

In this dissertation, we consider two dimensional convolutional layers (unless specified, *convolution* always indicates a two dimensional convolution operation in this dissertation.) The input to the convolutional layer is a tensor often in three-dimensional, denoted as $\mathbf{X} \in \mathbb{R}^{w \times h \times c}$, where w, h, and c are *width*, *height*, and number of *channel*, respectively. For example, a RGB image of size 224×224 is in $\mathbb{R}^{224 \times 224 \times 3}$.

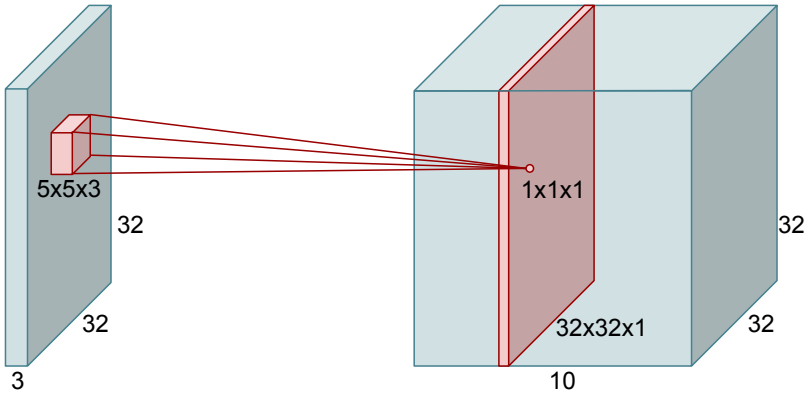


Figure 1.7: An illustration of a convolutional layer with convolutional filter of size 5×5, and an input tensor (left) of size 32×32×3. The size of output tensor (right) depends on stride s, size of filter, and the number of filters (10 in this case).

Instead of matching the entire input tensor with an equal number of weights, each neuron in a convolutional layer matches only number of channels of the input layer, while maintains a predefined size in spatial dimensions, thus reduce number of weight, which commonly called as *convolutional filter*, dramatically. Figure 1.7 shows the input tensor and the size of convolutional filter of each neuron in a convolutional layer, where the predefined size in the spatial dimension is 5, and the size of the input tensor is 32×32×3.

Let the number of output channels be D, the stride be s, the width and height of each convolutional fiter be U and V, respectively. Then, the i-th and j-th element of output tensor $\mathbf{H}$ on its d-th channel is computed by

$$\mathbf{H}^{(i,j,d)} = \sum_{k=1}^{c} \sum_{u=1}^{U} \sum_{v=1}^{V} \mathbf{W}^{(u,v,k,d)} \mathbf{X}^{(s\times i+u, s\times j+v, k)} + \mathbf{b}^{(d)}, \tag{1.3}$$

where $\mathbf{W}$ and $\mathbf{b}$ are convolutional filters and biases.

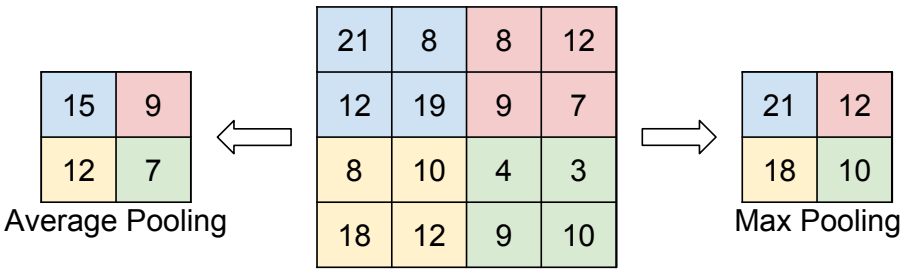


Figure 1.8: An illustration of average and max pooling.

1.4.5 Pooling Layer

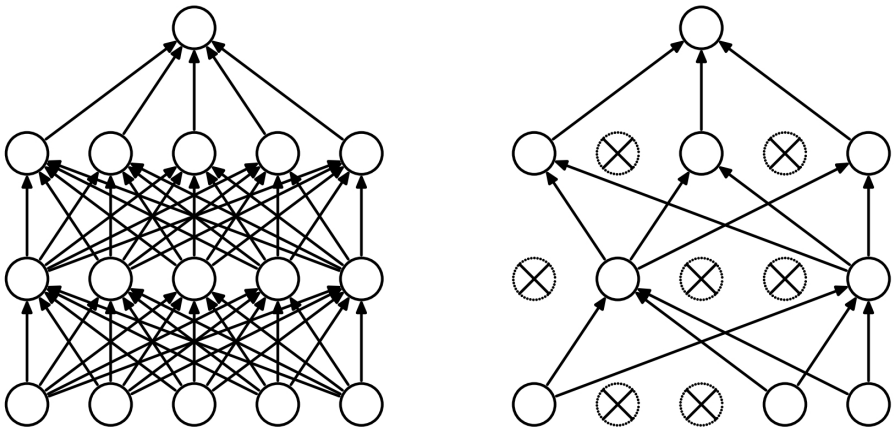
A pooling layer reduces the size of spatial dimensions of input tensor by the factor of the stride s and according to the specified size of receptive field U, while maintain the number of channels K. Given input tensor $\mathbf{X} \in \mathbb{R}^{w \times h \times c}$, where w, h, and c are *width*, *height*, and number of *channel*, respectively, the i-th and j-th element of output tensor $\mathbf{H}$ on its k-th channel is computed by

$$\mathbf{H}^{(i,j,k)} = f(\mathbf{X}^{(s\times i+1, s\times j+1, k)}, \ldots, \mathbf{X}^{(s\times i+1, s\times j+u, k)}, \ldots, \mathbf{X}^{(s\times i+u, s\times j+1, k)}, \ldots, \mathbf{X}^{(s\times i+U, s\times j+U, k)}) \tag{1.4}$$

where f can be either *maximum* or *mean* function, and the layer will be called either *max* or *average* pooling layer, respectively. Figure 1.8 illustrates max and average pooling operation.

1.4.6 Dropout

Dropout is a commonly-used method to help reduce overfitting. It randomly removes (*drops*) the output of each neuron with probability p every training iteration. The hyperparameter p is called *dropout-rate* and typically set to 0.5. It is proved to improve the performance substantially [2]. Figure 1.9 illustrates the idea of dropout.



(a) Standard Neural Net (b) After applying dropout.

Figure 1.9: An illustration of dropout idea. This figure is taken from [2].

1.4.7 Batch Normalization (BN)

Batch Normalization (BN), proposed by [26], aims to reduce the value which the weights of neurons shift around, as known as internal covariance shift, which is caused by the change of distribution of activations (output of the previous layer, input of the current layer) during training. It normalizes the activations by by subtracting and dividing the batch by mean and and standard deviation of that batch as follows:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1.5}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{1.6}$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{1.7}$$

$$y_i = \gamma\hat{x}_i + \beta \tag{1.8}$$

where x_i is the i-th sample in the batch of size n, ϵ is a small constant used to avoid zero division, and γ and β are parameters to be learned.

1.4.8 Softmax

Softmax function is usually used at the end of a classification model, where the goal is to assign label i from K possible labels to each sample. The input and output of this layer is K-dimensional vector, where the input value that represent score of each class is transformed into a real value in [0,1] range. Since the summation of the output values is 1, each *softmax*-ed value y_k is often interpreted as the probability of the output class k, and is calculated as follows:

$$p(\mathbf{y} = k|\mathbf{x}) = \frac{e^{x_k}}{\sum_{i=1}^{K} e^{x_i}} \tag{1.9}$$

where $\mathbf{x}$ is the input vector.

1.4.9 Loss Functions

Loss function is used to measure the error of the output of the ANN, and adjust the weights accordingly. The loss function is chosen according to the tasks and still is an active research area. The commonly used loss functions are:

- **Square Loss** $L = (y - \hat{y})^2$ measures the different between the predicted outcome y to the expected outcome $\hat{y}$ of a sample. It is more commonly used in regression problem where the goal is to predict single output value from the input sample.
- **Softmax Cross Entropy Loss** $L = -\log\left(\frac{e^{x_{\hat{y}}}}{\sum_{i=0}^{K} e^{x_i}}\right)$, where $\hat{y}$ is the ground truth label of the sample out of K possible labels, and $\mathbf{x}$ is the output of the final layer of classification model.

Chapter 2

Looking at Outfit to Parse Clothing

2.1 Introduction

Clothing parsing is a specific form of semantic segmentation, where the categories are one of the clothing items, such as *t-shirt*. Clothing parsing has been actively studied in the vision community [7–11], perhaps because of its unique and challenging problem setting, and also because of its tremendous value in the real-world application. Clothing is an essential part of our culture and life, and a significant research progress has been made with a specific application scenario in mind [14,27–36]. In this chapter, we consider how we can utilize recent deep segmentation models in clothing parsing and discuss issues in the current benchmarks.

* The main contributions in this chapter first appeared in our publication on *"Looking at Outfit to Parse Clothing"*, arXiv preprint arXiv:1703.01386,2017.

Figure 2.1: Combinatorial preference in clothing parsing: dress and skirt are in the exclusive relationship, yet independent pixel-wise prediction cannot encode such knowledge and results in mixture of patches (FCN-8s [1]). We propose the side-path outfit encoder and CRF alongside the segmentation pipeline to address the issue.

Clothing parsing distinguishes itself from general object or scene segmentation problems in that fine-grained clothing categories require higher-level judgment based on the semantics of clothing and the deforming structure within an image. What we refer the semantics here is the specific type of clothing combination people choose to wear in daily life. For example, people might wear *dress* or separate *top* and *skirt*, but not both of them together. However, from recognition point of view, both styles can look locally very similar and can result in false positives in segmentation, as shown in Figure 2.1. Such combinatorial preference at the semantics level [12–14] introduces a unique challenge in clothing parsing where a bottom-up approach is insufficient to solve the problem [15].

In this chapter, we approach the clothing parsing problem using fully-convolutional

neural networks (FCN). FCN has been proposed for general object segmentation [1] and shown impressive performance thanks to the rich representational ability of deep neural networks learned from a huge amount of data. To utilize FCN in clothing parsing, we need to take the above clothing-specific challenges into account, as well as a care to address the lack of training data for learning large neural networks. Based on the FCN architecture, we propose to extend the parsing pipeline by 1) a side-branch that we call *outfit encoder* to predict the combinatorial preference of garments for dealing with semantics-level consistency, and 2) conditional random field (CRF) to consider both semantics and appearance-level context in the prediction. Experimental results show that, starting from a pre-trained network, we are able to learn the outfit encoder and fine-tune the whole segmentation network with a limited amount of training data, and our model achieves the state-of-the-art performance in the publicly available Fashionista dataset [7] and Colorful Fashion Parsing dataset (CFPD) [11].

We also study the qualitative issue in the current clothing datasets. The existing benchmarks suffer from erroneous annotations due to the limitation in the superpixel-based annotation, as well as from the ambiguity of labels [9]. We develop a Web-based tool to interactively annotate pixels at multiple scales, and study how much influence we have on the performance metrics by manually refining the Fashionista dataset [7].

The outfit encoder learns a compact representation of the combinatorial clothing preference through the training of segmentation pipeline. We find that the resulting internal representation of the encoder is suitable for style retrieval application. The learned representation compactly encodes the gist of the dress-up style of the picture, and when used in retrieval, the representation is able to find semantically similar

clothing style (e.g., dress only or shirt + skirt combination), even if the low-level appearance cues such as color or texture look different.

We summarize our contribution in the following.

- We propose the side-path outfit encoder for the FCN architecture, and together with CRF to improve segmentation performance in clothing parsing. The evaluation shows that our model achieves state-of-the-art performance in clothing parsing.
- We develop a Web-based tool to interactively annotate pixels and study the qualitative influence in the segmentation benchmarks. Using the tool, we manually annotate the Fashionista benchmark [7] with high-quality and less ambiguous labels, which we refer Refined Fashionista dataset, and study the impact on the benchmark. We will release the tool and annotation for future research.
- We make a preliminary study showing that the outfit encoder is also useful for retrieval. The encoder representation compactly encodes the combinatorial preference of clothing items, and suitable for retrieving semantically similar styles.

2.2 Related Work

2.2.1 Semantic segmentation

The use of deep convolutional neural networks for semantic segmentation is increasingly becoming popular since the recent success in dense object recognition [1, 37,38]. Various techniques have been proposed to further improve the performance of dense prediction by deep neural networks, including global context information [39,

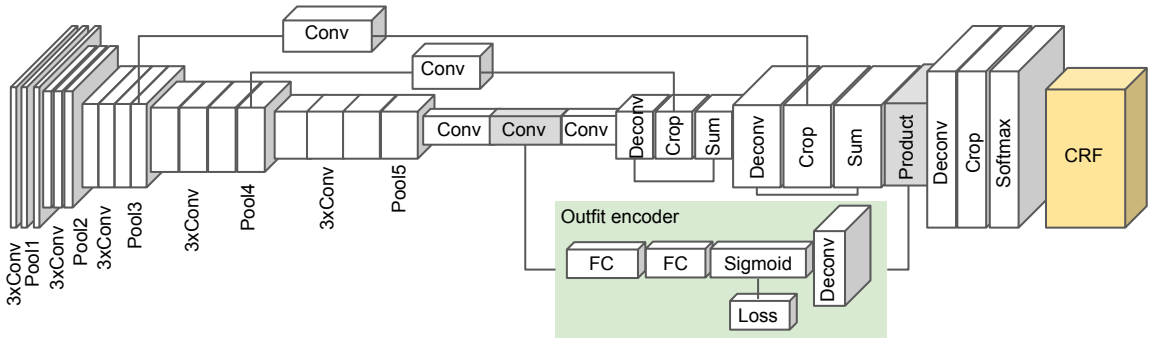


Figure 2.2: Our segmentation model based on FCN. We introduce 1) the outfit encoder to filter inappropriate clothing combination from segmentation, and 2) CRF to assign a visually consistent set of clothing labels.

40], learning deconvolution layers [41], applying conditional random fields as a post-processing [42, 43], or incorporating weakly annotated data in training set [44]. In this chapter, we propose a side-path encoder to predict unique set of consistent labels in segmentation and feed FCN output to fully-connected CRF for addressing combinatorial preference issue in clothing parsing. The side-path can be considered one kind of attention mechanism [45] or gating function to control information flow [46].

2.2.2 Clothing parsing

Clothing parsing has been an active subject of research in the vision community [7, 9–11, 47, 48]. Also, there is a similar variant of parsing problem referred human parsing [15, 39, 49–52]. The major difference between clothing and human parsing are the types of labels; Clothing parsing attempts to identify fine-grained categories of clothing items such as *t-shirt* and *blouse*, whereas human parsing aims at identifying body parts and broad clothing categories, such as *left-leg* or *upper-body clothing*. This difference brings a further challenge in clothing parsing that we have to disambiguate confusing labels, for example, *sweater* and *top*, even if they look similar.

The basic approach attempted in clothing parsing is first to identify human body

configuration in the image, and given the body joints, solve for the best assignment of pixel-wise labels [7, 9, 47, 48, 51]. Often superpixel-based formulation is employed instead of pixel-wise labeling to reduce the computational complexity, though superpixels sometimes harm the final segmentation quality in the presence of textured region [9]. In this work, we rely on the large size of receptive fields in the deep architecture to identify the contextual information from human body, and eliminate the extra pre-processing necessity to explicitly identify human parts in the image. Skipping pose-estimation has an additional advantage of not requiring full-body visible in the image frame.

2.2.3 Clothing retrieval

Retrieval and recommendation is one of the most important applications in clothing recognition, and there have been many efforts in various scenario, such as street-to-shop [27, 34], or style suggestion or matching [14, 30]. The key idea is to learn a meaningful representation to define the distance between style images [18, 32]. In this chapter, we consider retrieving the whole dress-up style rather than looking at a specific item using the outfit encoder representation.

2.3 Our approach

This section describes our FCN model with outfit encoder and fully-connected CRF.

2.3.1 Fully convolutional networks

We build our segmentation model upon the FCN architecture [1]. The FCN model is a convolutional neural network and makes a dense pixel-wise prediction by replacing all fully-connected layers in the classification network with convolutional layers, followed by upsampling filters to recover the original image resolution in the output. The FCN model proposed in [1] can implement a different upsampling strategy. In this chapter, we use the 8-stride variant of the VGG 16-layer net [53] to build our clothing parsing model on.

We choose the FCN architecture for clothing parsing expecting that the receptive fields of mid-to-later layers can cover sufficiently large input regions so that the final pixel-wise prediction makes a proper judgment on different but similar items, such as *coat* and *jacket*, based on the contextual information in the image. Thanks to the deep architecture, the receptive field in the last layer has large coverage of the input frame and is expected to contain sufficient contextual information from human body in the internal representation within the network. Also, there is no restriction on the input image that the image frame must contain the full human body.

2.3.2 Outfit encoder and filtering

We introduce a side path to FCN that encode and predict combinatorial preference of garment items. Figure 2.2 illustrates the architecture of our network. The idea is to predict the *garment combination as a summary of segmentation* using this side encoder path, and feed it into the main segmentation pipeline to filter the prediction on the possible set of labels. Our outfit encoder predicts a binary indicator of existence of each clothing label in the final segmentation, as usual attribute prediction problem.

The outfit encoder inserts two fully-connected (FC) layers and a sigmoid layer to predict a vector of clothing indicators. The first FC layer has 256 dimensions, and the second FC layer has dimensions equal to the number of classes in the dataset. The second layer predicts confidences of existence of each garment, which can be viewed as soft-attention or gating function to the segmentation pipeline. We connect 2nd FC with a sigmoid, then merge this vector back to the FCN segmentation pipeline using element-wise product, similarly to gating functions in LSTM or GRN [46]. This vector multiplies confidences to filter out uncertain labels from the image.

Formally, our outfit filtering is expressed by the following. Let us denote the heat-maps of the FCN by F_i for each label i, and the scalar prediction by our outfit encoder by g_i. Then, we apply a product to obtain the filtered heat-maps G_i:

$$G_i = g_i \cdot F_i.$$

The role of outfit filtering is to encourage or prevent certain clothing combination from appearing at the image-level (e.g., skirt + dress never happens together, but dress-only or skirt + top likely). Such decision requires to look at the whole image instead of local parts, and thus we let the garment encoder predict the existence of all labels at the image-level. The prediction is integrated back to the main segmentation trunk as a bias to the heat-maps produced per label.

The internal representation of the garment encoder makes a compact representation of clothing semantics in the given picture. We show that the representation is also useful in image retrieval scenario in section 2.6.

2.3.3 Conditional random fields

The outfit filtering enforces combinatorial semantics in the segmentation, but still the prediction contains a lot of small regions of incompatible items due to the pixel-wise prediction without explicit modeling of label combinations. Here, we introduce a fully-connected CRF to improve the segmentation quality. CRF has been shown to be effective for segmentation [9,54,55] or used as post-processing step after segmentation using CNN [42–44]. In this chapter, we use the implementation of [54] as a post-processing step to correct predictions*.

Our energy is a fully-connected pairwise function:

$$E(\boldsymbol{x}) = \sum_i \phi(x_i) + \sum_{i<j} \psi(x_i, x_j), \tag{2.1}$$

where $\boldsymbol{x}$ is the label assignment for pixels. The unary potential $\phi(x_i) = -\log P(x_i)$ takes label assignment probability distribution at pixel i, denoted by $P(x_i)$. We use the softmax output of FCN for this probability. The pairwise potential $\psi(x_i, x_j)$ considers contrast and position of two pixels using Gaussian kernels:

$$\psi_p(x_i, x_j) = \begin{cases} w_1 g_1(i,j) + w_2 g_2(i,j) & \text{if } x_i \neq x_j \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

$$g_1(i,j) = \exp\left(-\frac{|p_i - p_j|^2}{\sigma^2_{\text{position}}} - \frac{|I_i - I_j|^2}{\sigma^2_{\text{color}}}\right), \tag{2.3}$$

$$g_2(i,j) = \exp\left(-\frac{|p_i - p_j|^2}{\sigma^2_{\text{smooth}}}\right), \tag{2.4}$$

*We also attempted the end-to-end model [43] but could not reliably learn the network perhaps due to the small data size in our experiment.

where $g_1(i,j)$ is an appearance kernel, $g_2(i,j)$ is a smoothness kernel, p_i is the position of pixel i, and I_i is the RGB color vector of pixel i. Weights of both kernels are controlled by w_1 and w_2. The hyper-parameters σ_{position}, σ_{color}, σ_{smooth} control position scaling, color scaling, and smoothness scaling, respectively. We find the best hyper-parameters by non-linear optimization (L-BFGS) using the validation set, starting from the initial parameters $(w_1, w_2, \sigma_{\text{position}}, \sigma_{\text{color}}, \sigma_{\text{smooth}}) = (10, 10, 30, 10, 3)$ in our experiment. We approximately solve for the optimal label assignment using the algorithm of [54]:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} E(\boldsymbol{x}). \tag{2.5}$$

2.3.4 Training the network

Our assumption is that we do not have a sufficient number of images to learn deep networks from scratch in clothing parsing. We consider transfer-learning from the pre-trained FCN models. In this chapter, we follow the incremental procedure to fine-tune a coarse FCN model [1], and learn our garment encoder after that. We train FCN-32s, FCN-16s, and FCN-8s in sequence, move on to the training of the encoder, then fine-tune the whole model at the end.

We need to train new layers in the outfit encoder from scratch, using binary attribute vector as a ground truth. We can trivially obtain a binary vector from segmentation ground truth by simply taking a unique set of labels. We use sigmoid cross-entropy loss to train the encoder first (Figure 2.2). The encoder path is first trained independently with the binary indicators fixing learning rate for the main segmentation pipeline to zero, then fine-tuned together with segmentation pipeline later to avoid local optima. We implement the neural network using Caffe framework [56].

Figure 2.3: Common annotation error due to inappropriate superpixels. Superpixels can segment object boundary when the region is uniform and distinct from others, but spill over the object boundary when the regions have similar color (left) or produce a lot of small segments on the textured surface (right).

2.4 Interactive pixel annotation

We initially attempted to evaluate our model on the publicly available Fashionista dataset [7] and CFPD [11]. However, we find that both datasets have quite a bit of annotation errors. The annotation errors in both datasets are caused by 1) superpixel errors (Figure 2.3) and 2) ambiguous clothing categories (e.g., *shirt* and *blouse*). These annotation errors lead us to noticeable prediction errors in the final segmentation. Therefore, we decided to manually improve the annotation quality of the Fashionista dataset, and study how much quality improvement we can benefit from the dataset itself.

We have developed a Web-based tool to interactively annotate pixels. Our segmentation tool is based on annotation over superpixels, but we resolve the improper boundary of superpixels by coarse-to-fine interactive segmentation. Figure 2.4 demonstrates the example. Our tool computes SLIC [3] superpixels on the fly inside the Web browser, and the annotator can adjust the resolution of the superpixels as needed to mark smaller segments. Our tool can overcome the limitation of superpixel-based

Figure 2.4: Coarse-to-fine interactive annotation. We compute SLIC [3] superpixels on the fly so that the annotator can select the desired scale.

annotation because the tool does not share the segment boundary across multiple-resolutions. The tool can also apply morphology-based smoothing to remove the artifacts of SLIC superpixels. In a modern Web browser, the tool computes SLIC superpixels in a second, depending on the size of the image.

We merged some of the confusing clothing in the 56 categories (e.g., shirt and blouse) in Fashionista dataset as well as split broad labels (e.g., accessories) to prevent ambiguity in the annotation. We manually annotated all the 685 images in the Fashionista dataset with 25 categories. Note that the mapping of the labels is not unique and unfortunately the labels in one dataset cannot be automatically converted to the other.

We hope to expand the number of fully-annotated images in the future, but we find it still challenging to scale up the dataset due to the required level of expertise for a non-expert user in crowdsourcing service. It is our future work to make our annotation tool as easy as possible for non-expert users so that any type of semantic segmentation can benefit. We release our interactive annotation tool to the community as open-source software, as well as our annotation to Fashionista dataset.

2.5 Experiments

2.5.1 Datasets

We use Fashionista with the original annotation of 56 categories [7] (Fashionista v0.2), our refined annotation with 25 categories (Refined Fashionista), and CFPD [11] with 23 categories. Fashionista consists of 685 front-facing full-body images. Every pixel is given one of 56 fine-grained categories. However, the dataset has only 685 annotated images and some label appears only once or twice in the dataset. This results in some skewed performance metric due to the missing category in the ground truth in the test split. Our Refined-Fashionista reduces the number of clothing categories from 56 to 25 essential labels so as to avoid ambiguous labels. The annotation contains almost no superpixel artifact. CFPD consists of 2,682 annotated images based on superpixels for 23 labels. We divide Fashionista dataset into train-test splits using the same setting to the previous work [7], with 10% of training images leaving for validation, and divide CFPD dataset into (train, validation, test) = (78%, 2%, 20%) ratio. Images in all datasets are 400×600 pixels in RGB color, and we do not change the image format in our experiments. Each image shows a front-facing person with visible full-body.

2.5.2 Evaluation methods

We measure the performance using pixel-wise accuracy and intersection-over-union (IoU). We compare our models against FCN-32s, FCN-16s and FCN-8s [1], as well as against the reported state-of-the-art for each dataset, though some measurements are not available in the respective publication and the experimental condition

could be slightly different. Note that we cannot directly compare the performance against human parsing [39] due to the difference in semantic categories and the difficulty of reproducing the same condition without a publicly available dataset.

2.5.3 Quantitative evaluation

Table 2.1 shows performance of our models compared to various baselines. First, we notice that using FCN already shows the solid performance improvement over the previous state-of-the-art [9, 11, 48] that are not based on deep architecture. Also, as reported in [1], applying finer-scale upsampling (8s) improves the segmentation quality. Our best model achieves the state-of-the-art 88.68% accuracy compared to 84.68% of [48] and 54.65% IoU compared to 42.10% of [11], even with annotation issues in both benchmarks. In our Refined Fashionista dataset, our model marks 51.78% of IoU, which is a significant improvement from 44.72% of the base FCN-8s model.

Our outfit filtering with CRF makes an improvement in all of the datasets. Particularly, CRF shows an improvement in all cases in all datasets compared to the model without CRF. This confirms the lack of joint prediction ability in the plain FCN [42, 43]. The final effect of outfit filtering varies depending on the dataset. In Fashionista v0.2, our outfit filtering has weak effect, and only CRF is showing a noticeable improvement. We suspect this is due to the large number of ambiguous categories in Fashionista v0.2, such as *blazer* and *jacket*. Outfit filtering has weak influence by itself in CFPD, but combined with CRF, achieves the best IoU. In Refined Fashionista, our outfit encoder together with CRF achieves the best performance. We suspect the difference between datasets partly stems the low-quality annotation

Table 2.1: Parsing performance [%].

Dataset	Method	Acc	IoU
Fashionista v0.2 [7]	Paper doll [48]	84.68	-
	Clothelets CRF [9]	84.88	-
	FCN-32s [1]	85.94	29.61
	FCN-16s [1]	87.53	34.26
	FCN-8s [1]	87.51	33.97
	+ CRF	**88.68**	**38.03**
	+ Outfit filter	87.55	34.26
	+ Outfit filter + CRF	88.34	37.23
Refined Fashionista	FCN-32s [1]	88.56	40.88
	FCN-16s [1]	89.74	43.96
	FCN-8s [1]	90.09	44.72
	+ CRF	91.23	49.21
	+ Outfit filter	91.50	46.40
	+ Outfit filter + CRF	**91.74**	**51.78**
CFPD [11]	CFPD [11]	-	42.10
	FCN-32s [1]	90.34	47.65
	FCN-16s [1]	91.27	50.07
	FCN-8s [1]	91.58	51.28
	+ CRF	**92.39**	54.60
	+ Outfit filter	91.52	51.42
	+ Outfit filter + CRF	92.35	**54.65**

in Fashionista v0.2 and CFPD.

Figure 2.5 plots the Intersection-over-union (IoU) of FCN-8s and our model in our Refined Fashionista dataset, with % area of each class in the dataset. We find that our models improve IoU in almost all categories. The exceptions are small items, such as *necklace*, *glasses*, or *bracelet*, and *dress*. Dresses are usually confused with *blouse* and *skirt* combination. The small items tend to be smoothed out by CRF, and

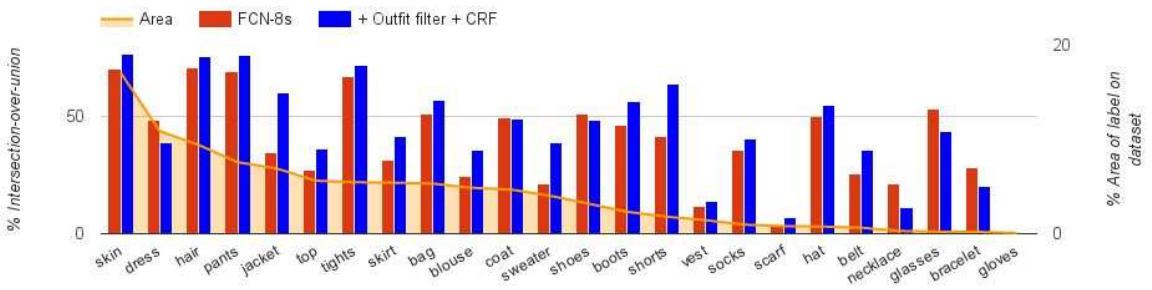


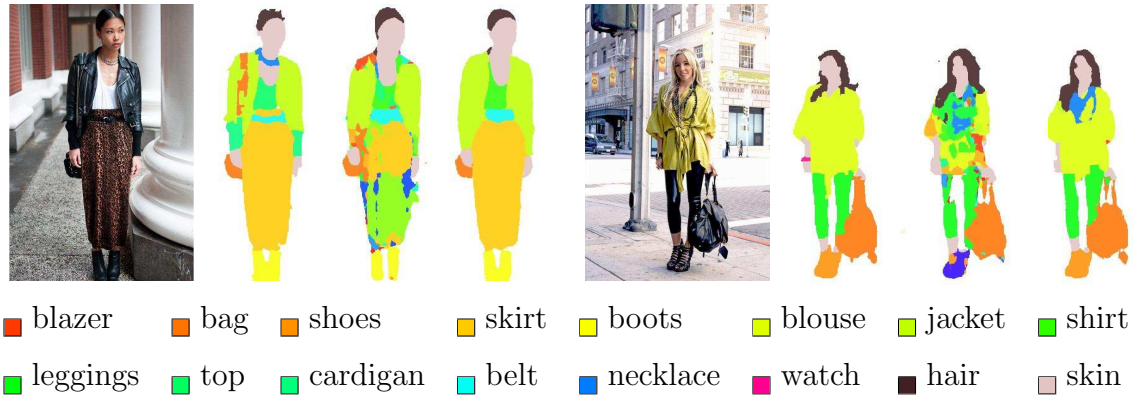
Figure 2.5: Intersection-over-union (IoU) in Refined Fashionista dataset [%].

does not always yield precise segmentation. We also find that it is challenging even for human to give precise annotation on small items, such as bag straps or necklace. Unfortunately, our annotation tool in section 2.4 is unable to provide annotation on such small regions, but we have less practical importance in identifying almost invisible items anyway. All models get 0 IoU for *gloves* due to the extremely limited examples (10 out of 685 images) in the Fashionista dataset.

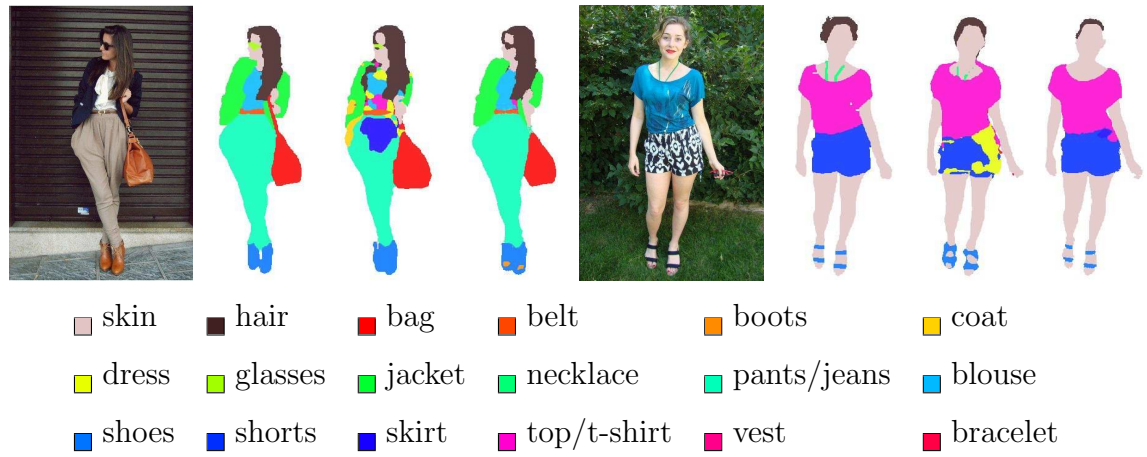
2.5.4 Qualitative evaluation

Figure 2.6 shows successful parsing results in Fashionista v0.2, Refined Fashionista, and CFPD, using the baseline FCN-8s and our outfit filtering with CRF. Our model can produce pixel-wise segmentation that are sometimes more precise than ground-truth annotations based on superpixels. In some case, our model can correctly identify small items that were missing in the ground truth, such as *necklace* in the right half of Figure 2.6a, even though they were counted as mistakes in the benchmark.

Figure 2.7 lists some of the failure cases from CFPD dataset. Failures can happen either because of the model or the dataset. The common error happening in the model-side is the confusion between clothing, such as making a mistake on *dress* vs.



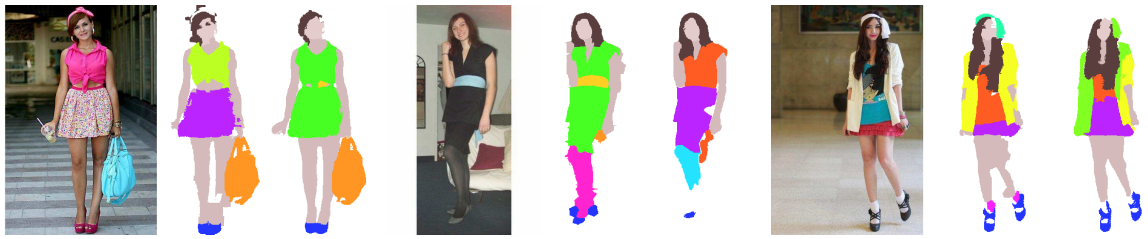
(a) Fashionista v0.2



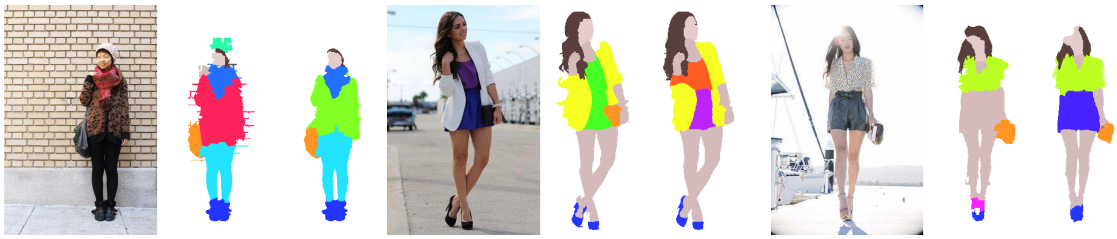
(b) Refined Fashionista

(c) CFPD

Figure 2.6: Successful cases in (a) Fashionista v0.2, (b) Refined Fashionista, and (c) CFPD. The figure shows an input image, a ground truth, the output of FCN-8s and our outfit filtering with CRF from left to right respectively.



(a) Prediction errors



(b) Dataset problem

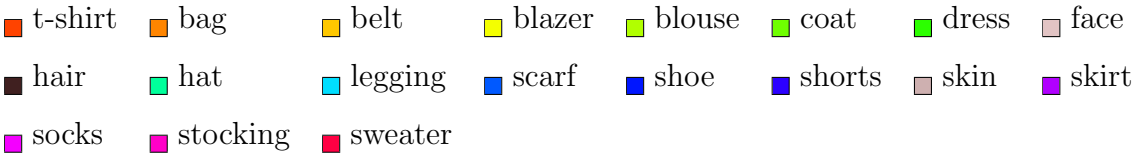


Figure 2.7: Failure cases in CFPD. Each triple shows an input image, a ground truth, and the output of our outfit filtering with CRF respectively. Failure is either caused by (a) the model (prediction error), or (b) the dataset (clothing ambiguity or incorrect annotation).

top and *skirt* combination. Such confusion can happen together in the same image, and produces a mixture of incompatible segments in the foreground region (right in figure 2.7a). We observe the common confusion among: outers (jacket, blazer, coat, sweater), inners (t-shirt, tops, shirt, blouse, dress), long-sleeves (tops, sweater), bottoms (jeans, pants, leggings), leggings (leggings, stockings, tights, long socks), or (boots, high sneakers). Some of the items are even difficult for humans to distinguish depending on the visibility. Our outfit filtering with CRF disambiguate this clothing-specific confusion and improves the segmentation quality, but some prediction errors still remain.

There is a noticeable error due to the annotation quality. The bottom row of

Table 2.2: Average performance of outfit prediction [%].

Dataset	Accuracy	Precision	Recall	F1
Fashionista v0.2	88.66	25.21	26.57	25.46
Refined Fashionista	73.36	39.29	46.06	40.00
CFPD	88.57	66.70	68.06	65.50

Figure 2.7 depicts cases when the ground-truth annotation is incorrect, even if our model can predict appropriate labels. The major reason of the annotation error is from 1) the inability of superpixel algorithms to respect object boundaries (bottom-center and right) and 2) human mistakes between the ambiguous categories (bottom-left). Such annotation issue introduces unreasonable artifact in the existing benchmarks.

2.5.5 In-depth analysis

How well outfit prediction performs?

Our outfit encoder learns to predict the set of applicable clothing categories as binary classification. Here, we report the performance of this prediction after the side-path encoder training. Table 2.2 summarizes the results in terms of average accuracy, precision, recall, and F1, for three datasets. Note that there are categories that always exist in the image (e.g., skin or background), but we do not exclude such labels in the evaluation.

The average accuracy in Fashionista v0.2 is 88.66% while in Refined Fashionista the average accuracy is 73.36%, despite the better performance in all other metrics. This counter-intuitive result comes from the fact that Fashionista v0.2 has much larger number of less frequent labels, and thus the large number of true negatives contributing to accuracy, because all other metrics ignore true negatives.

Table 2.3: Training and testing segmentation performance of outfit encoder [%IoU].

Dataset	Training	Testing
Refined Fashionista	84.72	46.40
CFPD	76.95	51.42

CFPD results have much better accuracy, precision, recall and F1 than both versions of Fashionista datasets. We suspect that the smaller number of images (685) in Fashionista compared to CFPD (2,682) is somehow causing overfitting and makes the overall performance lower in Fashionista than in CFPD.

The drawback of our side-path architecture is the increased risk of overfitting against small datasets, because our outfit encoder must be trained to predict an image-level category and thus the available training size is restricted to the number of images but not pixels. However, we are able to mitigate overfitting by fine-tuning the entire network towards the segmentation loss at the end (section 2.3.4). Using external weakly-annotated data [48,51] to learn the side-path is an alternative option.

How much overfitting happens in segmentation?

We evaluate how overfitting is happening in the final segmentation by comparing the training and testing performance. Table 2.3 summarizes the IoU performance of our outfit filtering without CRF on the training and testing splits in Refined Fashionista and CFPD. There is clearly a discrepancy between the performance in training and testing splits in both datasets, and the gap is more significant in Fashionista dataset. Indeed, we observed the training and testing discrepancy in all of the models including baseline FCNs. Our model achieves the state-of-the-art, but the result also suggests that we need a larger benchmark to properly evaluate clothing parsing.

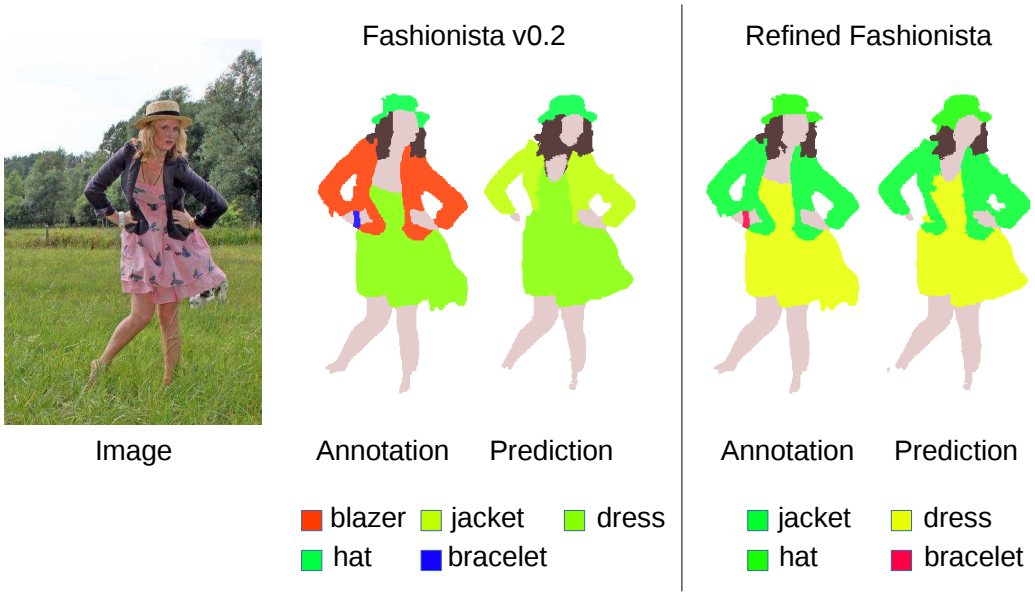


Figure 2.8: Comparison of annotation and prediction.

Qualitative effect of refining annotation

Figure 2.8 shows an illustrative example of how our refinement to Fashionista dataset disambiguates the segmentation. The Refined Fashionista merges some ambiguous labels, as shown in the figure that the confusion between *blazer* and *jacket* in Fashionista dataset disappeared in the refined dataset. The category-ambiguity is an inherent problem in clothing recognition, and is perhaps impossible to completely resolve. One approach might be to re-formulate the problem to allow multiple labels to be assigned to each pixel instead of the current exclusive label assignment, though that might require more annotation efforts.

2.6 Application: outfit retrieval

In this section, we demonstrate the application of the outfit encoder to image retrieval. The internal representation in the outfit encoder encompasses the gist of

(a) Using outfit encoding

(b) Using generic fc7 feature from VGG16

Figure 2.9: Image retrieval on Refined Fashionista dataset using (a) our outfit encoding, and (b) generic features from VGG16. Notice our encoding retrieves consistent clothing combination (jacket + top + shorts), while the generic feature pays attention to the background (road).

combinatorial clothing preference. This compact representation makes an ideal use for retrieving images of certain clothing combination (outfit). We have a preliminary study of how our encoding performs in the retrieval scenario. In this study, we do not make a quantitative evaluation and instead make a small qualitative analysis, due to the challenge in defining the ground-truth in fashion similarity [32].

Using Fashionista dataset, we first split the data into query and retrieval sets, and extract 256 dimensional representation from the fully-connected layer in the outfit encoder. As a baseline, we also compute the generic image feature from fc7 layer in the pre-trained VGG16 network. Our retrieval is based on Euclidean distance.

Figure 2.9 shows an example of 5 closest images retrieved using our encoder rep-

resentation and the baseline fc7 feature. In the figure, images retrieved using our method contain the same set of garments as in query, such as jacket, top, or shorts, without concerning colors. Retrieved images from the baseline feature seem to pay more attention on color, and the results contain the same gray and green background, but the dress-up style look rather random. We observed similar trend in other images.

The result indicates that our encoder representation clearly captures the combinatorial semantics of garments. The learned gist representation would benefit in fashion-focused applications such as outfit search or recommendation. We emphasize that the encoder does not require extra annotation for training, and the image representation comes for free in the training of the segmentation pipeline.

2.7 Conclusion and future work

This chapter proposed an extension to FCN architecture to solve the clothing parsing problem. The extension includes the side-path outfit encoder to predict a set of labels in the image, and CRF to produce a consistent label assignment both in terms of clothing semantics and structure within and image. Our model can learn from a pre-trained network and the existing annotated dataset without additional data. In addition, the learned image representation in the outfit encoder is useful for similar dress-up styles thanks to the internal representation that encompasses combinatorial clothing semantics. This chapter also introduced an refined annotation to Fashionista dataset for better benchmarking of clothing parsing, built with a Web-based tool to create a high-resolution pixel-based annotation. The empirical study using the Fashionista and CFPD dataset shows that our method achieves state-of-the-art parsing performance.

In the future, we wish to scale up datasets for clothing recognition with our Web-based tool, as well as to further investigate an approach to better incorporate the semantics of clothing in the prediction, for example by integrating CRF into the network [43]. Also, we wish to study how human pose estimation [57, 58] relates to clothing parsing under fully-convolutional architecture.

Chapter 3

Recommending Outfits from Personal Closet

3.1 Introduction

There have been growing interests in applying computer vision to fashion, perhaps due to the rapid advancement in computer vision research [7–11, 19, 32, 59–61]. One of the popular fashion applications is item recommendation [20–22, 62], where the objective is to suggest items to users based on user's and/or society's preference. Computer vision is used in various fashion applications such as e-commerce and social media. Recently, Amazon announced their automatic style assistant called "Echo Look™". Although the underlying mechanism is not published, emerging commercial applications confirm the ripe of computer vision applications in fashion.

* The main contributions in this chapter first appeared in our publication on *"Recommending Outfits from Personal Closet"*, in *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, vol. 00, pp. 269–277, Mar 2018.

Figure 3.1: Given an arbitrary number of items, our goal is to evaluate the quality of the outfit combination.

Measuring the quality of outfit is essential in building fashion recommendation system. In this chapter, we consider the problem of grading arbitrary combination of items as a whole (Figure 3.1). Previous works in outfit evaluation can be divided into two groups based on the input format: a worn outfit as a full-body picture as in [16–19], and as a set of images of items [20, 21], or a combination of both [22]. Each outfit can have an arbitrary number of items. For examples, in one day, one might prefer a combination of a jacket, a t-shirt, and jeans, while in the another she might want to wear a dress. Our goal is to build a machine learning system that accepts a variable numbers of items yet produce a consistent score for any size of combinations.

In this chapter, we view an outfit as a bag of fashion items and utilize deep neural networks to produce a score for a fixed-length representation of outfits. Unlike style recognition [16, 18, 19, 61], we take item images in isolation, not on human body, as seen on e-commerce sites or catalogs. We collect a large number of outfit data from a popular fashion website `polyvore.com`, and evaluate our approach based on standard classification metrics and human judgment. Our Polyvore409k dataset consists of 409,776 sets of clothing items from 644,192 unique items. The dataset forms a large bipartite graph of items and outfits. We partition the dataset into training and

testing sets such that there is no overlapping nodes and edges between the sets, and use them measure the classification performance. We also conduct a human study using crowdsourcing to assess predicted scores against human judgment, and show our model closely resembles human behavior. Using our grader, we build an outfit recommendation system that takes clothing items as an input and recommends the best outfits from the given items, to demonstrate the usefulness in a real-world scenario of personal outfit assistant. The contributions of the chapter are summarized below:

1. We build Polyvore409k dataset containing 409,776 outfits and 644,192 items. Every outfit covers the entire body with a variable numbers of items.
2. We propose an outfit grader that produces a score for fashion outfits with a variable number of items. Our empirical study shows that our model achieves 84% of accuracy and precision in Polyvore409k dataset.
3. We propose a human judgment framework on outfit quality, which provides a simple and reliable method to verify the reliability of outfit verifiers using a crowdsourcing platform.
4. We demonstrate that our outfit grader can build a recommendation system that suggests good outfits from a pool of items.

3.2 Related Work

3.2.1 Outfit Modeling

The use of computer vision techniques to study fashion is gaining popularity. Some early studies work on outfit images [16–18]. Although these studies can use

the appearances of outfit on a real subject, accurately identifying items in an outfit image is still an open problem. The manual annotation is costly, and the automatic detection and segmentation of fashion items in an outfit image [7–11, 63] are still not reliable. For example, "dress" and "top with skirt" are often incorrectly segmented, and the small objects like shoes and accessories are often missed. In addition, the importance of each item to the overall style may or may not be related to the scale of the item in an outfit image. In this chapter, we aim to study outfits as a combination of items, where each item has its own image.

Some studies treat outfits as combinations of item images as well [12, 20–22]. In [22], items in an outfit are recommended according to the requested occasion and the existing items in that outfit. The work by [20] focuses on learning personal preference on fashion based on accounts and associated outfits from `polyvore.com`. A study of pairwise relationship between fashion items was explored in [12] using co-purchase data. The work by [21] also uses data from `polyvore.com` to learn outfits as combinations of items based on item image, name, and category. They create an item recommendation system that suggests an item to match with other manually selected items.

Outfits have a natural structure based on human body, but [20–22] consider outfits with fixed number of items without considering variation in the structure. Outfits in [20] consist of one top, one bottom, and a pair of shoes without considering full-body items such as a dress, nor accessories. In [22], recommendation is made for either whole outfit, or upper-lower body pairs. Likewise, outfits in [21] consist of 4 items, regardless of item role. [21] does not guarantee the completeness of outfits. Since an outfit can be a collection of any items, it is possible to have incomplete

Table 3.1: Comparison of outfit datasets

	[22]	[21]	[20]	**Ours**
Number of images in dataset	24,417	347,339	85,252	644,192
Annotation method	Crowd-sourcing	Metadata from polyvore.com		
Outfit labels	Occasions and attributes	Based on number of votes	User-created vs. Randomly created outfits	
Number of items in outfits	2	4	3	Variable, up to 8
With respect to body parts	2 parts	No	3 parts	6 parts
Train/test item separation	Not verified	Verified	Not verified	Verified
Evaluated by human	No	No	No	Yes

outfits that do not cover whole body, such as an outfit consisting only of four pairs of boots.

In this work, we view outfits as collections of items from `polyvore.com`, similar to [20, 21]. We arrange the outfit data such that each outfit covers the entire body by considering the body part covered by each item, with variable number of items in the outfit.

3.2.2 Fashion Datasets

The number of fashion datasets is growing. Each image in some datasets [8,10,11, 17] is an outfit images, which contains many items. In other datasets [12, 20, 21, 62], each image contains only one item. Some datasets [22, 35] are combinations of both type. In [12], item combinations come in pairwise format from amazon co-purchase data. However, items that are bought together do not necessarily mean that they look good together as an outfit.

There are segmentation datasets [7, 10, 11] that seem suitable for our problem setup, because the datasets provide outfit images with the boundary of each item, but the number of samples is too few to learn a reasonable predictive model. Although [20] and [21] use datasets with combination of images as outfits and each item has its own image, the dataset is not publicly available. For the above reasons, we collect and build a new dataset, Polyvore409k dataset, which we describe in section 3.3.

In fashion outfit problem, each sample is an outfit which is a combination of items. We have to cleanly separate training data from testing data both for a set and individual items. [22] and [20] do not describe the detail on separation. [21] constructs a graph dataset, where each node represents an outfit, and a connection between any two nodes is formed if these two outfits have a common items. After that, the graph is segmented based on connected components. In this work, we use an efficient alternative approach to split a graph, which we describe in section 3.3.4.

3.3 Polyvore409k Dataset

This section describes our Polyvore409k dataset that consists of variable-length sets of items. Our Polyvore409k dataset has 409k outfits consisting of 644k item images. The comparison of outfit datasets to the previous work is shown in table 3.1. We plan to release the metadata of items and outfits, including the URLs to the images, to the public.

3.3.1 Data Collection

We collect Polyvore409k dataset from the fashion-based social media website `polyvore.com`. Each outfit, or *set* in Polyvore's terminology, consists of a title, items in the set, a composed image, and behavioral data such as likes and comments from other users.

3.3.2 Data Preprocessing

Data Cleansing The collected sets can contain non-clothing items or items that cannot be worn such as logo, background image for presentation purpose, or cosmetic items. We remove the item if its name does not contain clothing categories, then each item in a set is categorized into one of 6 outfit parts according to its categories:

1. Outer: coat, jacket, parka, etc.
2. Upper-body: blouse, shirt, polo, etc.
3. Lower-body: pants, jeans, skirt, joggers, etc.
4. Full-body: dress, gown, jumpsuit, robe, etc.
5. Feet: shoes, boots, flats, clutches, etc.
6. Accessory: bag, glove, necklace, earring, etc.

Table 3.2: Number of unique items in each outfit part

Part	Outer	Upper	Lower	Full	Feet	Accessory
Train	11,168	21,760	16,287	11,523	26,574	60,760
Test	6,656	12,744	11,089	8,871	17,564	37,988

Our definition of an outfit is a set that covers both upper and lower part of body, each of first 5 categories has at most one item, and at most three items for accessory. Sets that do not cover the whole body, e.g. missing lower body, are removed. At the end, we obtained 409,776 valid outfits which are composed of 644,192 unique items.

We consider only two layers on the upper body (outer and upper) because of the visibility, as the layers under two outermost layers are usually covered. We process sweaters, knitwear, and the likes as outer-upper hybrids. They will be considered as an upper if the outfit has other outer, and as an outer if the outfit does not have. The list of item categories and respective outfit parts is included as supplementary.

3.3.3 Quality Measurement

Measuring the quality of an outfit is a challenging task due to the subjective nature of judging visual appearance. The approach of [21] directly uses the number of votes (or *like* in `polyvore.com`'s terminology) of the outfit on the website as a quality measurement. However, some studies [17, 18, 64] argue that the number of votes from social media does not directly reflect the quality of the outfit, because the number of clicks is affected by a variety of factors, such as the topology of the social networks or the time when the outfit was published. In [20], the quality is defined by the preference of each user: outfits created by the user are treated as positive samples, and outfits created by randomly pick items are treated as negative. Given

Algorithm 1: Disjoint Set Sampling

input : All outfits $\mathbf{O}$
output: Set A, B, and C containing outfits such that items in outfits in A is not in B and vice versa

$A \leftarrow B \leftarrow C \leftarrow \emptyset$;
$A \cup \{\mathbf{O}_0\}$;
for $i \leftarrow 1$ **to** $|\mathbf{O}|$ **do**
  $O \leftarrow \mathbf{O}_i$;
  $items_A \leftarrow$ items in outfits in A;
  $items_B \leftarrow$ items in outfits in B;
  $items_O \leftarrow$ items in O;
  $sec_{AO} \leftarrow intersection(items_A, items_O)$;
  $sec_{BO} \leftarrow intersection(items_B, items_O)$;
  if $|sec_{AO}| > 0$ *and* $|sec_{BO}| > 0$ **then** $C \cup \{O\}$;
  else if $|sec_{AO}| > 0$ **then** $A \cup \{O\}$;
  else if $|sec_{BO}| > 0$ **then** $B \cup \{O\}$;
  else
    if $|A|/2 > |B|$ **then** $B \cup \{O\}$;
    else $A \cup \{O\}$;
  end
end

these insights, we take the following strategy.

Positive Samples Each Polyvore409k outfit has an associated *like*s that Polyvore users provide. Although the number of *like* might not directly reflect the quality of the outfit, it still shows that some people like the outfit. As a result, we use 212,623 outfits that has least one *like* as positive samples. In the future, as we obtain more data, we wish to increase the number of *like* threshold.

Negative Samples Similar to [20], we use outfits created by picking items randomly as negative samples. We believe that there are some preferred combinations of colors, textures, or shape of items in an outfit, and we assume that a randomly created outfit has very small chance to match those preferences.

Table 3.3: Number of outfits in each train and test partition

Number of outfits	Train	Test
Positive samples	66,434	26,813
Negative samples	132,868	53,626
Total	199,302	80,439
Ratio positive:negative	1:2	1:2

For each positive sample, we create two identical samples as negative samples, because the number of preferred combinations is expected to be significantly lower than random combinations. Then, we replace items in those two negative samples with random items of the same parts from the same train/test item pool. Although this sampling strategy is not *i.i.d.*, this approach guarantees the disjoint set property between training and testing sets, and tends to produce *hard* negative examples that shares some items with positive counterpart. Also, the distribution of number of items and existences of outfit parts in samples are preserved.

Table 3.2 shows the number of items in each part of outfit. Figure 3.2 shows the distribution of number of items in an outfit in train and test splits for both positive and negative samples. Table 3.3 shows the numbers of positive and negative samples in each split.

3.3.4 Evaluation Data

The set-item relationship constitutes a bipartite graph, where nodes are outfits or items, and edges represent inclusion relationship. For performance evaluation using Polyvore409k, we have to split the bipartite graph such that the training and testing

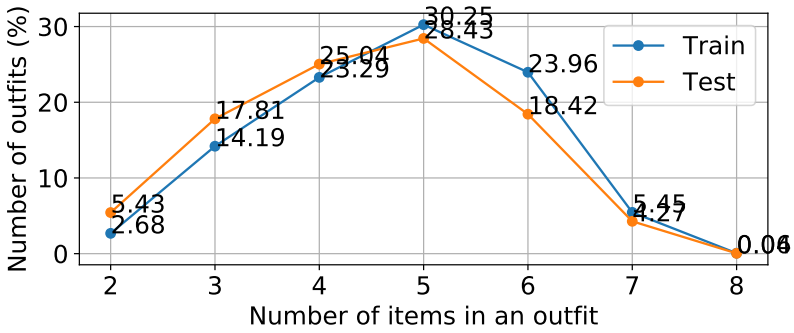


Figure 3.2: Distribution of number of items in outfits in training and testing partition are similar.

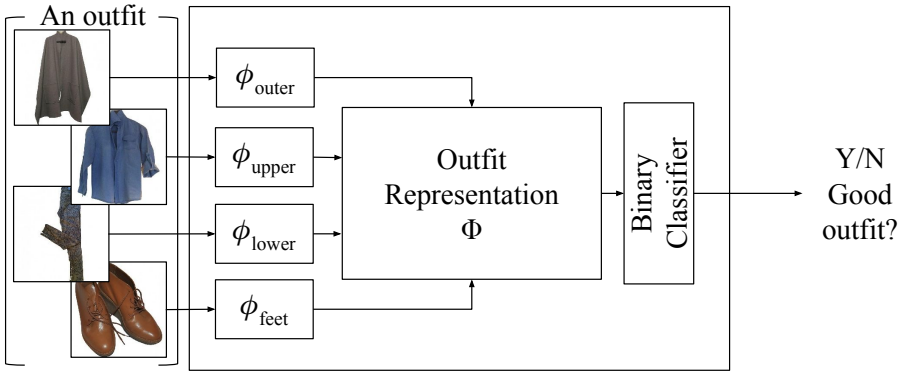


Figure 3.3: Outfit Grader

splits do not share any item or outfits. We use Algorithm 1 to separate training and testing splits.

3.4 Outfit Grader

3.4.1 Problem Formulation

We formulate the outfit grading as a binary classification problem. Given an outfit $O \equiv \{x_{\text{outer}}, x_{\text{upper}}, \cdots, x_{\text{accessory3}}\}$, where x_{part} is an item image, the goal is to learn a mapping function: $F : O \mapsto y$ to predict the outfit's quality $y \in \{0, 1\}$. Once we learn the mapping F, we are able to sort arbitrary combinations of items according

to the prediction score.

The challenge is how to represent an outfit O with a variable number of items. Luckily, the number of visible items is limited even though an outfit can contain a variable number of items. Therefore, we assign each item into one of the six categories and concatenate the item representations to produce the outfit representation. Figure 3.3 shows our grader. Our grader takes a bag of images and convert them to feature representations, then concatenates the individual features according to the item's category to produce the fixed-length representation. We describe details below.

3.4.2 Item Representation

We convert the image of each item in the outfit to a feature representation $\phi_{\text{part}}(x_{\text{part}})$, using a convolutional network. In this chapter, we use ImageNet-pretrained ResNet-50 [65], and extract the 2,048-dimensional embedding from `pool5` layer as an item representation. We extract features for 5 item parts and up to 3 accessories. For missing parts, we give a mean image to obtain features which is equal to zero-input to the convolutional network.

3.4.3 Outfit Representation

After we extract features from each item, we concatenate all features in the fixed order to form an outfit representation $\Phi(O) \equiv [\phi_{\text{outer}}, \phi_{\text{upper}}, \cdots, \phi_{\text{accessory2}}]$. Note that we allow accessories to appear multiple times in the outfit, and we simply concatenate all the accessory features ignoring the order. Outfits with less than 3 accessories get mean images as well to the other part. We have 5 item parts and 3 accessories per outfit, resulting in a 16,384 dimensional representation as the outfit representation.

3.4.4 Scoring Outfits

From the outfit representation Φ, we learn a binary classifier and predict a score. We utilize a multi-layer perceptron (MLP) to learn the mapping function. In this chapter, we compare 3 MLPs with various configurations to see the effect of number and size of fully-connected (FC) layers on this problem. The models we used are:

1. one_fc4096: one 4096-d FC layer
2. one_fc128: one 128-d FC layer
3. two_fc128: two 128-d FC layers

Each of fully-connected layers are followed by batch normalization and rectified linear activation (ReLU) with dropout. One 2-d linear layer followed by soft-max activation is added to every models to predict a score. We use multinomial logistic loss to learn the parameters of the grading model.

3.5 Performance Evaluation

3.5.1 Evaluation Setup

We learn the grading model from the training split of Polyvore409k dataset, and evaluate the binary classification measures on the testing split. The performance is measured against the ground truth. In this chapter, we report the performance of our models without fine-tuning the parameters of the convolutional network for the item feature extraction. We implement the neural network using Caffe framework [56]. We choose cross entropy as a loss function. We train the models for 400,000 iterations using stochastic gradient descent with momentum, where the initial learning rate and momentum are set to 10^{-4} and 0.9, respectively. We measure accuracy, precision, and

Table 3.4: Accuracy, average precision, and average recall of our outfit graders at 400,000 iterations.

	Accuracy	Avg. Precision	Avg. Recall
one_fc4096	**84.51**	**83.66**	**80.62**
one_fc128	80.14	81.25	72.79
two_fc128	82.11	82.14	76.36

Table 3.5: Precision, recall, and F1 value of both classes from one_fc4096 model at 400,000 iterations.

	Testing			Training
	Negative	Positive	Average	Average
Precision	85.60	81.73	83.66	99.25
Recall	92.29	68.95	80.62	99.31
F1	88.82	74.80	81.81	99.28

recall to evaluate the performance. The prediction is counted as correct if it matches the ground truth.

3.5.2 Quantitative Results

The accuracy, average precision, and average recall of all models are displayed in table 3.4. According to the table, one_fc4096 ,which has 84.51% accuracy, 83.66% average precision, and 80.62% average recall, is clearly the best among the three models. The precision, recall and f1 value of both classes from one_fc4096 model are shown in table 3.5. Top 8 positive and negative samples from the model are shown in figure 3.4. Qualitatively, preferred outfits contain items with consistent colors and styles, whereas low-scoring outfits tend to have less common visual elements between

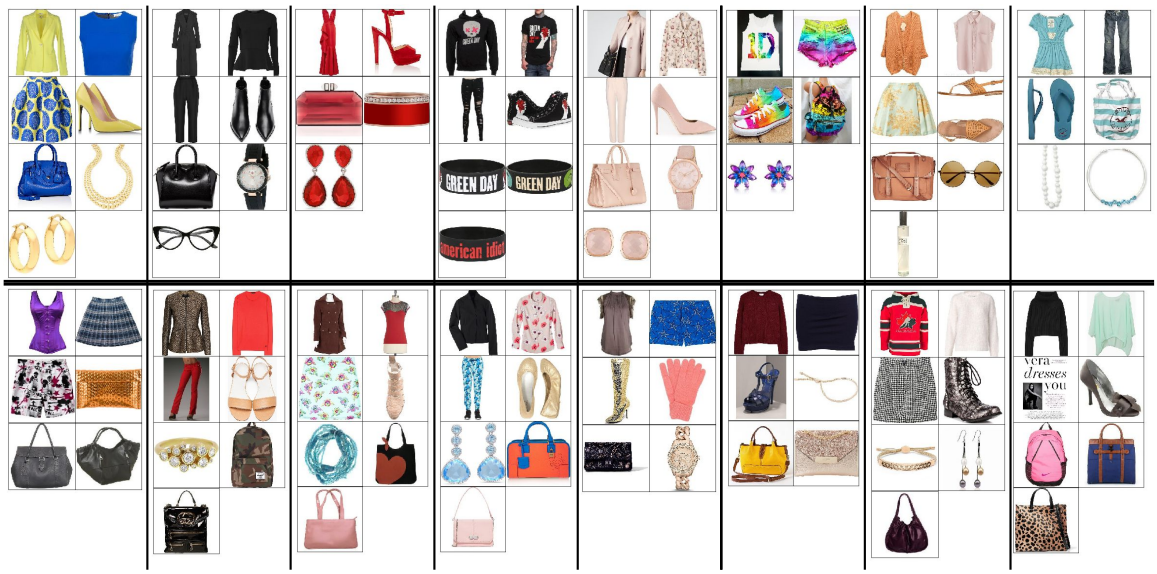


Figure 3.4: Eight best (top row) and worst (bottom row) outfits judged by our outfit grader

items.

From table 3.5, 92.29% recall for negative class shows that the model is very reliable for pointing out the bad outfit. However, 68.95% for the positive one shows that it tends to judge positive outfit as a negative one as well. When considering that the training performance is almost 100% correct as shown in table 3.5, we can conclude that the model overfits the training data.

3.5.3 Color and Item Type Analysis

We conduct another set of experiments to analyze the effect of various features on grading performance. We train one_fc4096 for 100,000 iterations using 5 features: (1) item type, (2) 4-color palette, (3) (1)+(2), (4) ResNet-50 features from grayscale images, and (5) ResNet-50 features from RGB images. Item types are extracted from item name, and 4-color palettes are extracted from item image.

The result in table 3.6 shows that the item type and color represent the items

Table 3.6: Performances of one_fc4096 outfit grader trained by different features: (1) item type, (2) 4-color palette, (3) (1)+(2), (4) ResNet-50 features from grayscale images, (5) ResNet-50 features from RGB images

Feature	Accuracy	Avg. Precision	Avg. Recall
(1) Item types	74.33	71.77	67.02
(2) 4-color palettes	74.53	72.71	66.35
(3) (1)+(2)	78.93	77.57	73.03
(4) ResNet-50 grayscale	81.31	79.35	77.69
(5) ResNet-50 RGB	84.26	83.06	80.73

equally, and the combination of them gives a better representation. However, the composite feature from ResNet-50 outperforms both primitive features, even without the color information. Finally, the color information in the ResNet-50 features affects the performance of outfit grader by 3% classification accuracy.

3.6 Human Evaluation

Outfit quality is a subjective topic. An outfit that looks chic to one person may look ugly to another. Although an evaluation on testing samples is important, we argue that it might be insufficient to verify the reliability of the approach. We conduct a large-scale human perception evaluation to further assess our model. We use the predictions from one_fc4096 to do evaluations on human perception using Amazon Mechanical Turk (AMT).

3.6.1 Geographic Trends

People from different regions have different tastes in fashion. Since the one_fc4096 model learned from data from `polyvore.com`, in order to show that the model successfully learned the compatibilities between fashion items, the model's predictions should be judged by people from the same region as the training data. After we inspect metadata of all 93,247 outfits that are used as positive samples, we found that they come from 39,590 different users. Around half of them (21,413 users ,54%) did not provide the country. For the remaining 18,177 users, which come from 175 countries, most of them come from United States (8,167 users, 45%), followed by Canada (872 users, 5%), and other countries. As a result, our model's predictions will be judged by Americans.

3.6.2 Evaluation Protocol

We setup the experiment as choosing the better outfit from each pair to minimize the effect of absolute bias or personal preference from human subjects. In addition, outfits in each pair must have exactly same outfit parts at the same location in the outfit image, so that only the compatibility of items affects the judgments, not the outfits' configuration nor number of items in the outfits.

Our hypothesis is, if outfits in the pair have similar quality, people will choose both outfits equally. On the other hand, if the outfit has a large gap in quality, people will definitely choose one over the other. The quality score of each outfit come from our outfit grader. If our outfit grader can reliably judge the quality of the outfit, our hypothesis will be true.

To verify the hypothesis, we select a number of best outfits as references, denoted

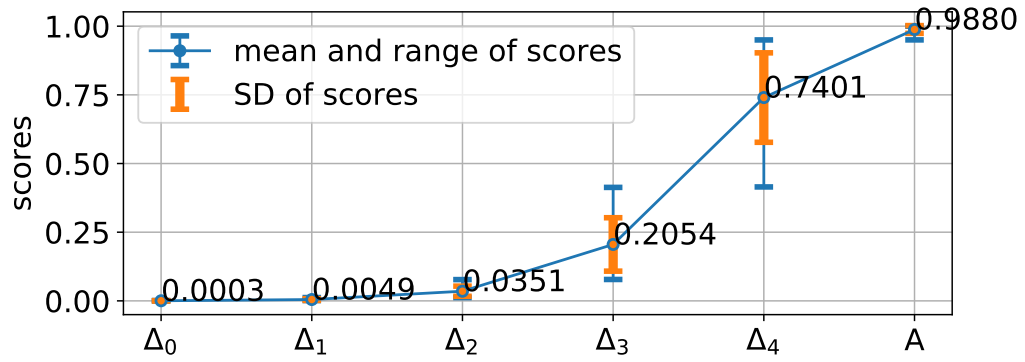


Figure 3.5: Mean, range, and SD of scores in each samples group **Δ** and **A**

as *Alpha (***A***)*. Then, we select other outfits of different qualities, denoted as *Delta (***Δ***)*, and pair them up with best outfits. After that, we show these pairs to human annotators. For each pair, we tell the annotators to choose the better of the two.

Our expectation is that, the difference in quality between outfits in the **A** group and each of **Δ** group is directly related to the probability that the annotators will choose outfits in the **A** group given the outfits in **Δ**. Since this experiment is set as pairwise comparisons, we believe that the mentioned probability should be approximated as

$$p(s_\alpha|s_\delta) = s_\alpha/(s_\alpha + s_\delta) \tag{3.1}$$

where s_α and s_δ are positive probability calculated by our model of outfits in **A** and **Δ**, respectively.

3.6.3 Implementation detail

Our outfit's *score* is the positive probability from the outfit grader. We randomly select 1,000 outfits with the score more than 95% as "**A**" group. After that, we

Table 3.7: Number of "Unable to decide" answers and ties from the experiments comparing outfits in $\mathbf{A}$ to $\mathbf{\Delta_j}$ for $j \in 0, 1, 2, 3, 4$

j	0	1	2	3	4
Number of "Unable to decide" out of 5,000 questions	688	820	924	696	422
Number of ties Out of 1,000 pairs	63	80	114	111	80

sort outfits with score less than 95% in an ascending order, then divide them into 5 groups, from "$\mathbf{\Delta_0}$" which is the group of outfits with the lowest scores, to "$\mathbf{\Delta_4}$" which is outfit with the highest scores but still less than 95%.

The experiment consists of 5,000 pairs of outfits. We use outfits from $\mathbf{A}$ group as "good" outfits, and $\mathbf{\Delta_j}$ for $j \in \{0, 1, 2, 3, 4\}$ as "bad" outfit. For each $\alpha_i \in \mathbf{A}$, we randomly select an outfit $\delta_{j,i} \in \mathbf{\Delta_j}$ for each $j \in \{0, 1, 2, 3, 4\}$ that has exactly same outfit parts as α_i. Our hypothesis is, the visible difference in outfit quality in $(\alpha_i, \delta_{0,i})$ pairs is more than in $(\alpha_i, \delta_{4,i})$. We denote pair $(\alpha_i, \delta_{j,i})$ as $p_{j,i}$ for $j \in \{0, 1, 2, 3, 4\}$ and $i \in \{0, 1, .., 999\}$. We show in figure 3.5 the mean, range, standard deviation of scores in each $\mathbf{\Delta_j}$ for $j \in \{0, 1, 2, 3, 4\}$ and $\mathbf{A}$, and the difference of mean of scores of each group to $\mathbf{A}$.

We ask 5 annotators to vote each pair $p_{j,i}$. Each annotator selects the better outfit in each pair, or select "Unable to decide" if the annotator thinks that the outfits looks equally good (or bad). The total number of questions in our experiment equals to: 5 annotators $\times$ 1,000 questions $\times$ 5 δs = 25,000 questions. We show examples of outfits in figure 3.6. For each row, 5 pairs of outfits are created by pairing an outfit in $\mathbf{\Delta_j}$ for $j \in \{0, 1, 2, 3, 4\}$ with $\mathbf{A}$. An example questionnaire is shown in figure 3.7.

Figure 3.6: Comparison of outfits used in human evaluation. Each row shows outfits in different quality groups but have the same outfit configuration.

3.6.4 Evaluation Metrics

We use the term *Matching Ratio* to describe the ratio that human annotators select α_i in pair $(\alpha_i, \delta_{j,i})$ as the better-looking outfit. We also remove the "Unable to decide" votes, shown in table 3.7, from the calculation. There are two metrics, matching ratio by individual answer, and by majority vote on each pair. For the latter, we also remove ties, shown in table 3.7, from the calculation.

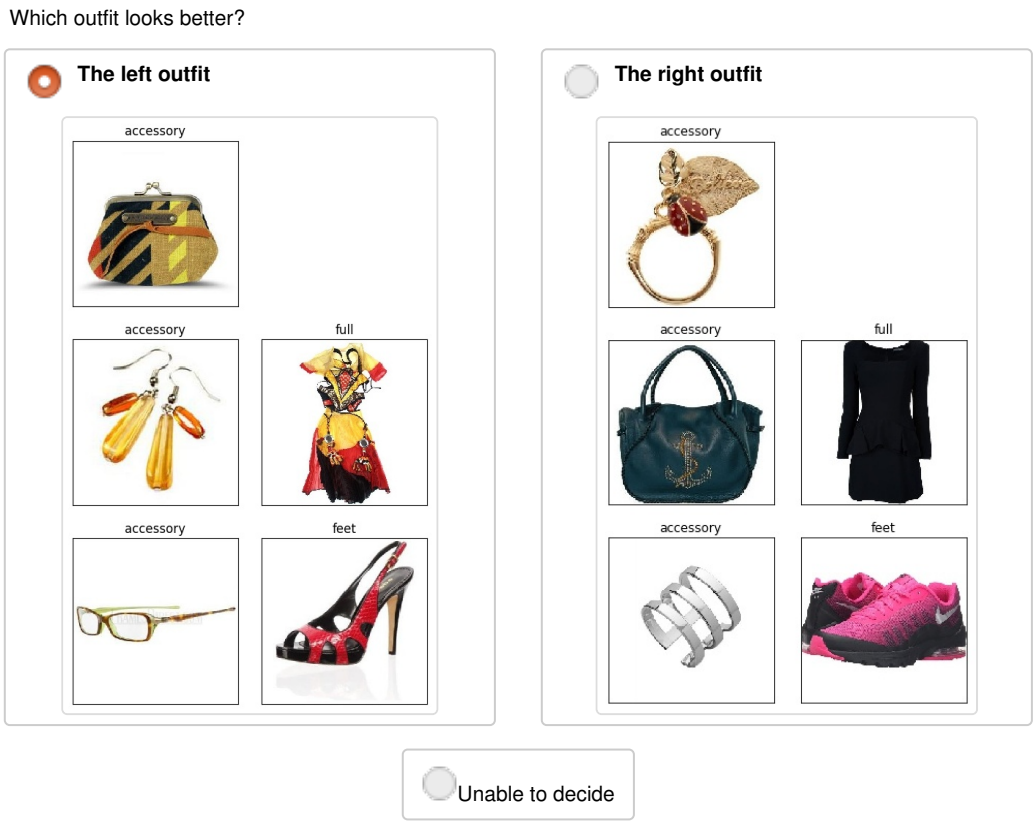


Figure 3.7: An example of questionnaires used in human evaluation, with associated outfit part of each item

3.6.5 Results

The results, with our expectation as explained in section 3.6.2, are shown in figure 3.8. The 91.25% matching ratio by voting shows that the human annotators agree with predictions from our model. Although not perfectly matched, the result has similar trend with our expectation. The result indicates that that the value of our positive probability (score) properly resembles the quality of the outfit.

Regarding the gap between human votes and our model, we have to remind that, the reliability of the human evaluations is not the absolute. As said earlier, fashion is a subjective topic. We might be able to use some small sets of questions to verify the ability of annotators, although this approach introduces absolute bias to the

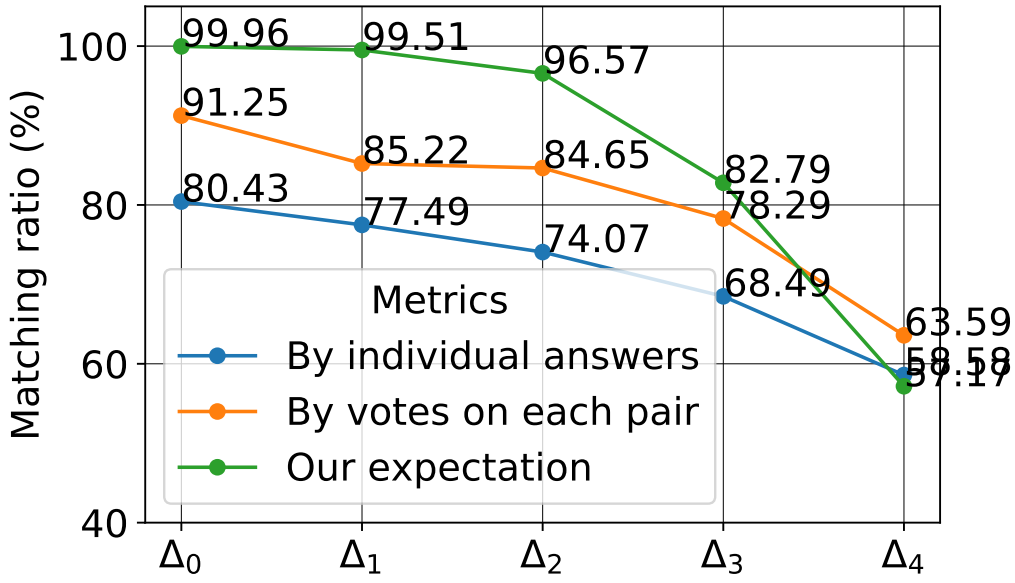


Figure 3.8: Matching ratio of the prediction to human judgment in each samples group $\boldsymbol{\Delta}$ with our expectation

evaluation since people have different tastes in fashion.

3.7 Application: Outfit Recommendation

If the number of items is not very large, as is often the case with a personal closet, our outfit grader can directly be used as an outfit recommender by generating multiple outfits and ranking them. To demonstrate this usage, we conduct experiments as follows.

3.7.1 Outfit generation

For generating outfits, we consider four outfit configurations: (1) outer layer with upper- and lower-body, (2) only upper- and lower-body, (3) outer layer with full-body, and (4) full-body only. All configurations include a footwear and at most three optional accessories.

Although it may be reasonable to assume that there are a modest number of clothes, there could be a large number of accessories and generating all possible item combinations is computationally expensive. We test four methods for generating outfits that take efficiency into consideration. The first method (*Ordered Beam Search*) is to regard outfit generation as a sequence generation problem, and employ a beam search algorithm. To be specific, in each item step t, the items that belongs to $part_t$ are the possible item extensions, and our "one_fc4096" outfit grader is used as the scoring function. The beam search starts from each item in the pool and considers all outfit configurations applicable to the item. It stops when all parts of the outfit are added according to its configuration. We then remove the duplicated outfits and recommend the best outfits based on the score from our outfit grader.

The second method (*Orderless Beam Search*) uses the entire item pool as the possible item extensions at all time steps, while the rest is the same as the first one. The third method (*Partial Beam Search*) generates all possible combination of main parts (outer, upper, lower, full, feet) of the four outfit configurations, from which ten best outfits (based on score from our outfit grader) per outfit configuration per item are kept as base outfits. We then use the beam search to add accessories to those base outfits. The fourth method (*Baseline*) creates 100 outfits per outfit configuration in a random manner. Then, the duplicates are removed and the best ones are recommended.

Each of the four methods outputs 10 best outfits based on the scores from our outfit grader. In the experiments, for all the methods, we set the beam width for beam search to three and include a null item in the item pool as an accessory to give a choice to the beam search to add nothing to an outfit in each "accessory" time

steps.

3.7.2 Evaluation

A good outfit recommender should be able to find sets of well-coordinate items in a pool of apparently random items. To test each recommendation method in terms of this property, we created 957 test cases, each of which contains items from one positive, denoted as P, and two negative samples. These samples are randomly drawn from the testing partition of Polyvore409k dataset. From those items, the recommended outfit, denoted as R, should be similar to the positive samples P. To measure the performance of the recommender, we use four conditions as (1) $P = R$, (2) $P \subset R$ (3) $R \subset P$, (4) $(P = R) \cup (P \subset R) \cup (R \subset P)$. For each method, we regard a recommendation (i.e., top ten recommended outfits) as successful if the condition is met by one of the ten recommended outfits.

Table 3.8 shows the results. It is seen that *Partial Beam Search* outperforms the baseline in every metrics. The reason why Ordered and Orderless Beam Search perform worse than Partial Beam Search is because our outfit grader is trained using complete outfits, while the early steps of beam search rely on the score of partial outfits, which our outfit grader is not trained for. Figure 3.9 shows successful and unsuccessful recommendations. We argue that the recommended outfits in the failure case are even better than the target positive sample. This is due to the nature of weakly-supervised data.

Figure 3.9: Recommended outfits from Partial Beam Search. Each row shows one test case, where 5 outfits on the right are generated from items in 3 outfits on the left. Outfits with blue border are positive, red are negative, green are *exact match*, cyan are $P \subset R$, and orange are $R \subset P$. The others are recommended outfits that do not meet the conditions.

Table 3.8: Performance of outfit recommendation by the proposed outfit grader combined with four outfit creation methods. The metrics are (1) $P = R$, (2) $P \subset R$ (3) $R \subset P$, (4) $(P = R) \cup (P \subset R) \cup (R \subset P)$, where P and R denote the positive sample and recommended outfits, respectively.

Approaches	(1)	(2)	(3)	(4)
Ordered Beam Search	11.39	14.11	19.64	32.29
Orderless Beam Search	14.84	20.79	9.40	29.89
Partial Beam Search	**34.38**	**41.80**	**22.68**	**59.77**
Baseline	8.88	21.53	14.11	36.36

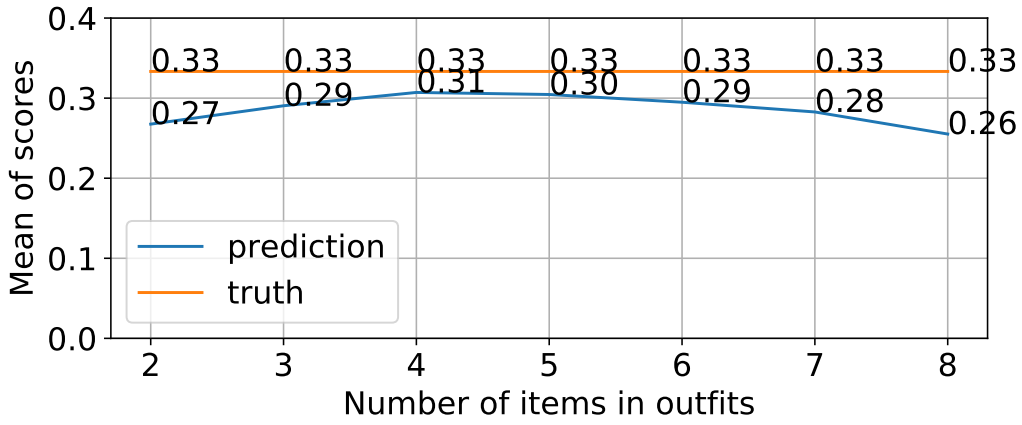


Figure 3.10: Mean of outfit score by number of items according to prediction and truth

3.8 Conclusion

In this chapter, we study outfits as combinations of items by developing outfit graders and outfit recommenders. Given a combination of items as an outfit, our best model can judge if the outfit looks good or not at over 84% accuracy on testing samples, and at 91% matching ratio on evaluations by human annotators. In addition, user can just give a pool of items that user have to our outfit recommender, and it will recommend outfits from the item pool. We also collect a large clothing dataset consisting of over 600,000 clothing items and over 400,000 outfits, and use the dataset to learn and evaluate the outfit graders and recommenders.

3.9 Appendix

3.9.1 On Biases from the Creation of Negative Samples

In section 3.3.3, we claimed that our negative sample creation method can prevent additional artificial bias between positive and negative samples. There are two biases

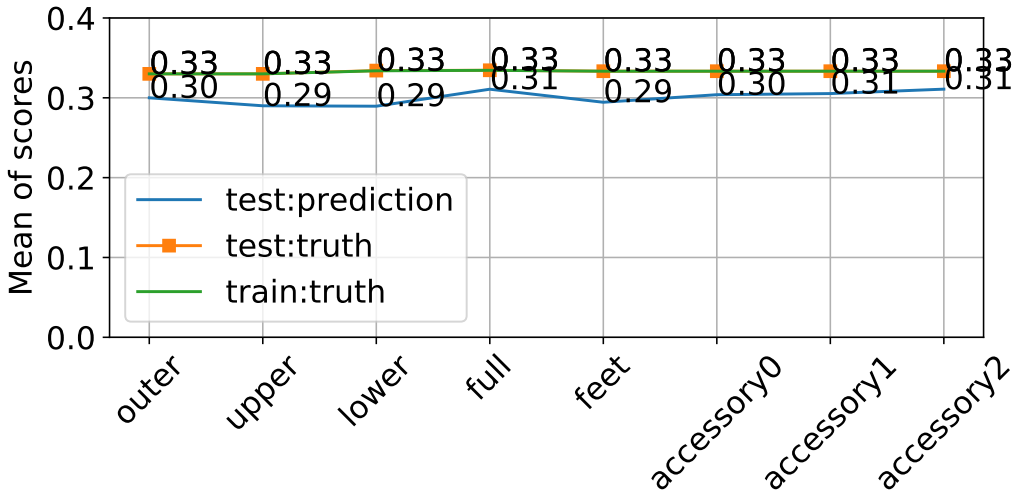


Figure 3.11: Mean of scores of outfits containing each outfit part. Note that the train:truth line lies on top of the test:truth line perfectly.

that we tried to prevent:

- Bias from number of items in outfits
- Bias from existence of parts

To support the claim, we analyze our model's predictions as follows:

Bias from number of items in outfits To see if the number of items affects the outfit quality judged by our model or not, we plot the average score of testing samples for different number of items in samples in figure 3.10.

The mean score from the truth of samples is 0.33 because the ratio between number of positive (the score is one) and negative (the score is zero) samples is 1:2. However, our model gives lower score to all cases, while outfits with 4-5 items got higher scores than others. We notice that the curve is similar to the distribution of outfits by number of items in figure 3.2. We suspect that the bias come from the overfitting of our model to training samples.

As mentioned in [2], the neuron network is prone to overfit training data if the

number of sample is not large enough. Because we have more outfits with, for examples, 4 and 5 items than 2 or 7, and we have more negative than positive samples at the ratio of 2:1. As a result, our model overfits these samples, especially with 2 or 7 items, which leads to lower scores in said samples.

To conclude, the result shows that our method to create negative samples can prevent the bias from number of items in outfits.

Bias from existence of parts To see if the existence of outfit parts affects the score given by our model or not, we plot the average score of training and testing samples containing each outfit part in figure 3.11.

As shown in figure 3.11, the mean of scores of training samples containing each part is exactly 0.33 because the ratio between number of positive and negative samples is 1:2. In testing samples, the means are around 0.3 perhaps because of overfitting. The figure also shows that the average score of outfits containing each part are almost identical, means that the score does not depend on the existence of any particular outfit part, thus confirm that this bias does not exist in our dataset.

Chapter 4

Identifying Feature-Level Flaws in Outfits via Gradient-Based Method

4.1 Introduction

Recently, there are many approach on applying computer vision technique on fashion, either as an item recommendation based on one's purchase history or personal preference [20], recommending outfits from a pool of item [4, 22], measuring outfit fashionability [4, 17, 21, 24, 66], and recommending outfits according to the location [23]. However, many of these works rely on black-box model that, while giving a very good numerical result on testing samples, do not explain the reason behind the prediction that is useful in term of fashion [4, 20–22]. For the works that provide reasons [23, 24], the model usually needs to be trained on a large amount of manually

* The main contributions in this chapter first appeared in our publication on *"Toward Explainable Fashion Recommendation"*, arXiv preprint arXiv:1901.04870,2019.

Figure 4.1: The goal of the this work is to identify the flaw in an outfit in both item-level and feature-level. Each row is an outfit that consists of variable number of items. In the first row, the flawed item is identified. In the second and third row, the flaw is identified to the feature-level.

annotated data, which is expensive in both time and money, and usually not publicly available. Here, we purpose an outfit fashionability measurement system that can predict outfit fashionability level reliably, is able to explain the reason behind the predicted fashionability score at the interpretable feature-level by giving the numerical influence level of each feature in each item to the overall outfit fashionability both positively and negatively (*Item-Feature Influence Value (IFIV)*), and is trained on publicly available dataset without any additional annotation.

Since evaluating the explanation via human experiment is always subjective, especially in the fashion topic, we create our testing samples from an existing dataset and use them to quantify the performance of the explanation as an outfit flaw de-

tection. Because our system gives *Item-Feature Influence Value (IFIV)*, we can use the *negative IFIV*, that show how each item-feature *negatively* influences the outfit fashionability, to identify the flaw in the outfit and use the predicted item-feature to evaluate the performance of the system. The goal of this work is illustrate in the figure 4.1.

The main contributions of this chapter are as follows:

1. A method to extract interpretable features from item images in outfits without using any additional annotation.

2. The outfit fashionability measurement system with item-feature influence values that, given an outfit as a set of item with associated outfit parts, predicts how each item-feature affects the overall outfit fashionability. These values can be used to identify the flaw in an outfit, which not only explain the reason behind the output outfit fashionability, but also guide the user to dress better.

3. A method to create outfit flaw detection samples which can be used to evaluate the outfit flaw detection system.

The organization of this chapter is as follows. We discuss the related work in section 4.2. Next, we talk about our purposed method to explain outfit fashionability at item-feature level in section 4.3. To reliably explain the outfit fashionability, we first evaluate the outfit fashionability measurement system in section 4.4, then use outfit flaw detection as an evaluation method for item-feature influence prediction at section 4.5. Finally, we provide a conclusion of our work in this chapter in section 4.6.

4.2 Related Work

4.2.1 Outfit Fashionability Measurement and Recommendation

There is a growing interest in the application of computer vision techniques to measure the fashionability of an outfit. The authors of [17] predicted fashionability scores from an outfit image and tags. The authors of [66] used bidirectional LSTM (Bi-LSTM) [67] to learn the compatibility relationships among fashion items by modeling an outfit as a sequence. In contrast, the authors of [4, 21] use fully-connected layers instead of Bi-LSTM. In [4, 21, 66], the authors used fashion item representations from deep models that is trained for generic image recognition to measure the fashionability of outfits. This approach works well for measuring the fashionability of an outfit by giving the score. However, since it cannot provide the reason to support the score, the usefulness of the score is limited. In addition, fashion is a subjective. Without providing any reason, user may just discard the score entirely.

4.2.2 Explaining Artificial Intelligence

Recent advances in deep learning have dramatically improved neural network accuracy in image classification [53, 65, 68], object detection [69], object segmentation [1, 70], Visual-Question Answering (VQA) [71–74], etc. Despite the impressive performances, the lack of explanation and understanding raise concerns from public, especially in life-critical applications [75]. To this end, there are several works trying to explain the decision of machine learning models [76–79]. The author of [76] explained the prediction from a complex model of any particular sample by examining

the local perturbed neighbors of that sample, then created a linear model that approximates the complex model locally to the sample being explained. The authors of [77] proposed the Class Activation Map (CAM) which shows the region in an image that responsible for the prediction. Unfortunately, it is only applicable to image classification models that do not contain any fully-connected layer. The authors of [78] generalized and extended [77] into a new method called *Grad-CAM*, which applicable to wide variety of CNN model-families including CNN with fully-connected layers (e.g. [53]), image captioning [80–82], and Visual Question Answering (VQA) [71–74].

4.2.3 Explaining Artificial Intelligence for Fashion Outfit

In recent years, there are several studies on popularity of fashion styles and fashion recommendation as mentioned in section 4.2.1 [12, 20–22, 83]. Many of these works employed the black-box scheme that gives very high predictive performance but also uninterpretable. Although there are several attempts on explaining black box models as mentions in section 4.2.2, fashion outfits consist of sets of items, which those approaches are not applicable.

Recently, there are two attempts to explain the popularity on fashion styles [24,84]. The author of [24] relied 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 an outfit. This method is very costly because of the required manual annotations. This model also considers only pairwise relationships between items in an outfit. On the other hand, the authors of [84] created an upper-lower matching recommendation with a textual explanation from outfits and comments crawled from `polyvore.com`. Although this work does not

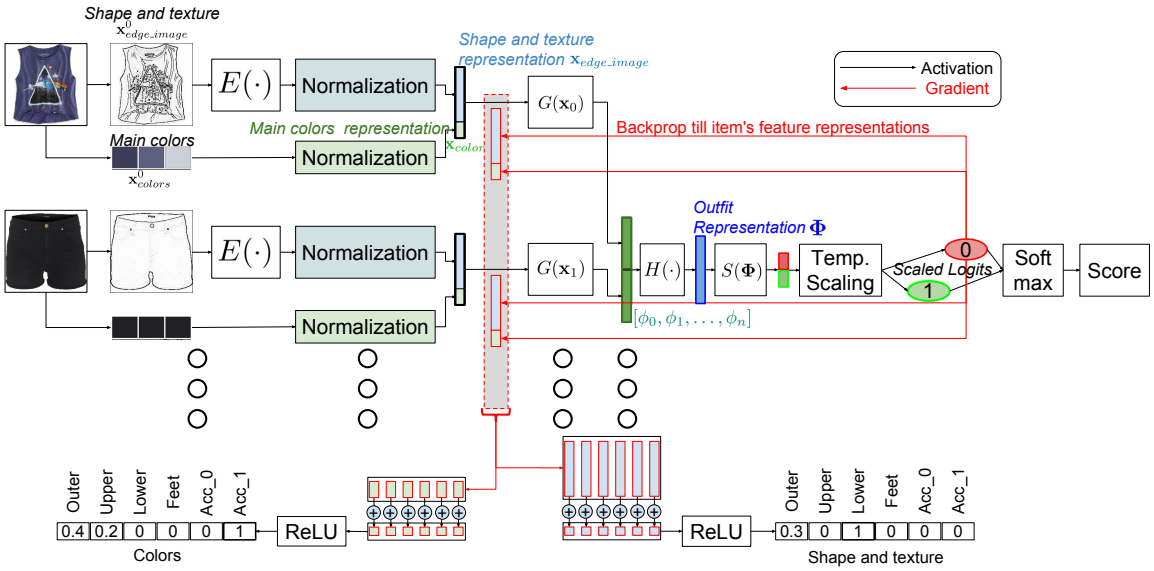


Figure 4.2: The overview of our outfit fashionability measurement system with feature influence values. Given an outfit as a set of item, we extract *edge_image* and main colors of each item, then forward propagate them through a pretrained CNN, normalization, concatenation, and fully connected layers with ReLU and batch normalization to obtain the raw score for the *bad* outfit label. We then backpropagate the gradient of the *bad* outfit label back to the representation of each item. The values of gradient at the item representation are separated by feature and, summed, rectified, and scaled to [0,1] range. We call these values *Item Feature Influence Value (IFIV)*. Finally, the *IFIV* of each item feature is used to identify the flaw of the outfit.

require manual annotation, the size of outfits is limited to only 2 items, and we argue that the comments from a social media website are considered weak.

In this work, we create an outfit fashionability measurement system that takes an outfit that consists of a variable number of items that is able to quantify the influence that each item in the outfit has on the outfit fashionability in the feature level. In addition, we also purpose an evaluation protocol to assess the ability of our method to identify the flaw of the outfit.

4.3 Explaining Outfit fashionability

In [4], the outfit grader classifies an outfit as a positive (good) and negative (bad) outfit as in the binary classification problem and use the probability of positive prediction as the outfit score. Motivated by *Grad-CAM* [78]'s ability to identify the section of feature map that corresponds to a specified class accurately while using only image-level annotation in training, we develop a gradient-based method that can identify the degree that each item affects the *goodness* and *badness* of an outfit.

In addition to item-level influence, we also develop a method to extract human-interpretable features from each item, which allow us to identify how each feature of each item influences the fashionability of the outfit compared to the same feature of other items. As a result, we purpose a system architecture that can separate each item in an outfit into shape with texture and main colors, measure the outfit's fashionability, then use a gradient-based method to see how each feature of each item affect the fashionability of the outfit.

An overview of our outfit fashionability measurement system with feature influence values is illustrate in figure 4.2

4.3.1 Extraction of Interpretable Item Features

Intuitively, attributes that associate with fashion items are item type, material, color, shape, and texture. However, obtaining such attributes requires experts in the fashion domain, and there is no clear definition of various attributes such as item type and shape of the item. As a result, we purpose to use the three explainable features that are already embedded in the item image and can be extracted easily: shape, texture, and colors, as the features of each item in an outfit.

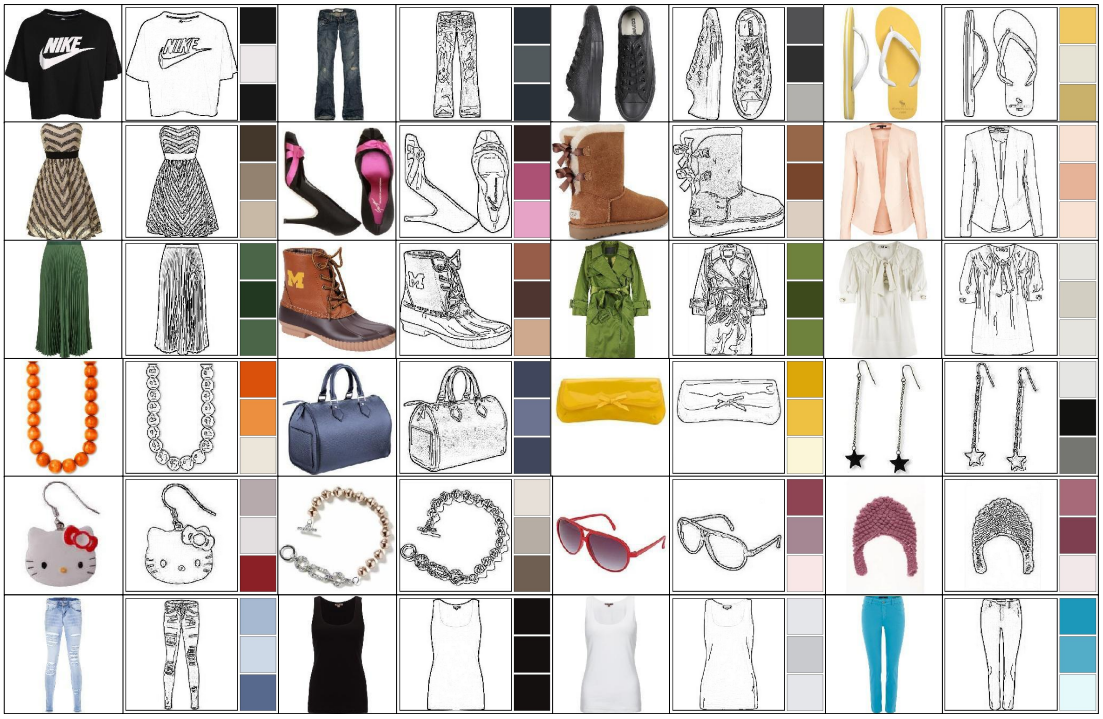


Figure 4.3: Item images with their *edge_image* and main colors.

Shape, texture, and colors of an item can be extracted from an item image using conventional image processing techniques. In this work, we extract main colors using K-mean clustering [85]. For shape and texture, we apply two operations to an item image and combine the result together as shown in Algorithm 2. We call the output of this process *edge_image*.

For an item with a rougher texture, the *edge_image* output contains more small dots over the area. On the other hand, the *edge_image* of an item with smoother texture contains less number of dots. The operation also preserves creases and patterns in the item.

From the *edge_image*, we then use a pretrained convolutional neural network (CNN) as a feature extractor E to extract an n-d embedding from *edge_image*, de-

Algorithm 2: *edge_image* extraction

input : RGB Image I
output: *edge_image*

$$f \leftarrow \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$I_{e_1} \leftarrow canny(I, \sigma = 1.1)$ [86];
$I_{e_2} \leftarrow I * f$;
$I_e \leftarrow I_{e_1} + I_{e_2}$;
$edge\_image \leftarrow 255 - clip(I_e, 0, 255)$;

noted as $\mathbf{x}^0_{edge\_image}$ and use as a representation of shape and texture of an item.

$$\mathbf{x}^0_{edge\_image} = E(edge\_image) \tag{4.1}$$

For colors, we first remove the background from the item image, then apply K-mean clustering [85] to cluster the color value of all pixels in the item image into 3 clusters. We then use the centroids of those 3 clusters as 3 dominant colors of the item, results in 9-d color representation (3 colors $\times$ 3 RGB color values) denoted as $\mathbf{x}^0_{colors}$. The examples of original images, their *edge_image*, and 3 dominant colors are shown in figure 4.3.

4.3.2 Item Representation

To create an item representation from n-d *edge_image* representation and 9-d color representation, we normalize both representations separately by

$$\mathbf{x}_f = \frac{\mathbf{x}^0_f - \mu(\mathbf{X^0_f})}{\sigma(\mathbf{X^0_f})}, f \in \{edge\_image, colors\} \tag{4.2}$$

where $\mathbf{x}_f^0$ and $\mathbf{x}_f$ are a raw and normalized representation of a feature f of an item respectively, while $\mu(\mathbf{X}_f^0)$ and $\sigma(\mathbf{X}_f^0)$ are mean and standard deviation vector of the raw representation of feature f of all items in training outfits.

Finally, we concatenate $\mathbf{x}_{edge\_image}$ and $\mathbf{x}_{colors}$ together as the representation of an item, denoted as $\mathbf{x}$.

$$\mathbf{x} = [\mathbf{x}_{edge\_image}, \mathbf{x}_{colors}] \tag{4.3}$$

Since we have multiple item in an outfit, and each item occupy a specific outfit part, we denote the representation of item that occupy part i as $\mathbf{x}_i$.

4.3.3 Outfit Representation and Encoding

Firstly, for each item in an outfit, we encode its representation $\mathbf{x}$ as ϕ by

$$\phi_{\mathbf{i}} = G(\mathbf{x}_i) \tag{4.4}$$

where G is a trainable item encoder and $\phi_{\mathbf{i}}$ is the item encoding of an item that occupy i^{th} part of human body.

Based on [4], our outfit representation $\mathbf{\Phi}$ of an outfit O is defined as

$$\mathbf{\Phi} = H([\phi_0, \phi_1, \ldots, \phi_n]) \tag{4.5}$$

where n is a maximum number of items that an outfit can have, which is 8 in the polyvore409k dataset [4], and H is a trainable outfit encoder.

4.3.4 Outfit Grader as a Binary Classifier

Given an outfit representation $\mathbf{\Phi}$ of an outfit O, a binary classification model F is used to evaluate the fashionability of the outfit:

$$\mathbf{s} = S(\mathbf{w}^T\mathbf{\Phi} + \mathbf{b}) \tag{4.6}$$

$$F(O) = \sigma(\mathbf{s}) \tag{4.7}$$

where $\mathbf{w}^T$ and $\mathbf{b}$ are the weights and bias of the linear model, respectively. $F(\cdot)$ is the entire system, and $\sigma(\cdot)$ is the softmax function:

$$\sigma(\mathbf{s})_k = \frac{\exp(\mathbf{s}_k)}{\sum_{m=0}^{1}\exp(\mathbf{s}_m)}, k \in \{0, 1\} \tag{4.8}$$

where we are using the notation $\mathbf{s}_k$ to refer to the k-th element of the vector of class scores $\mathbf{s}$ of outfit O. To learn the parameters (G, H, S) in the framework, a loss function $L(O, y)$, is defined on the training data (O, y) based on softmax loss that has the form:

$$L(O, y) = -\log\left(\frac{\exp(\mathbf{s}_y)}{\sum_{m=0}^{1}\exp(\mathbf{s}_m)}\right) \tag{4.9}$$

Temperature scaling Since we use the positive probability $\sigma(\mathbf{s})_1$ as an outfit score, and it is an output of the softmax function (equation 4.8), scores of outfits are accumulated at either end of probability range, which is less ideal when using it to compare the fashionability of outfits. To elevate this, we apply the temperature scaling [87], which is the simplest extension of Platt scaling [88], to calibrate the outfit

score $\hat{q}$ as follows:

$$\hat{q} = \sigma_{SM}(\mathbf{s}/T)^{(1)} * 100 \tag{4.10}$$

where T is a trainable parameter called the *temperature*. T is optimized with respect to the negative log-likelihood on the validation samples. We use $\hat{q}$ as the fashionability score of an outfit O.

4.3.5 Item Feature Influence Value (IFIV)

As shown in figure 4.2, *Item Feature Influence Value (IFIV)* of an item feature indicates the degree of item feature's influence to outfit fashionability relative to the same feature of other items. To obtain the value, first, we forward propagate the outfit through the model to obtain logit of class c, y^c. We then compute the gradient of y^c with respect to the representation of each item in the outfit $\mathbf{x_i}$ (section 4.3.2), *i.e.* $\frac{\partial y^c}{\partial \mathbf{x_i}}$. The gradient values are element-wise multiplied with the item representation. And since our item representation is a concatenation of feature representations (equation 4.3), the item-feature gradient vector $\mathbf{g}_{i,f}$ of item-feature representation $\mathbf{x}_{i,edge\_image}$ and $\mathbf{x}_{i,colors}$ of outfit part i can be obtained as:

$$\mathbf{g}_{i,f} = \mathbf{x}_{i,f} \circ \frac{\partial y^c}{\partial \mathbf{x}_{i,f}}, f \in \{edge\_image, colors\}$$

To get the *Item Feature Influence Value (IFIV)* of feature f and outfit part i, we sum $\mathbf{g}_{i,f}$ of each item i to get item feature influence vector $\mathbf{v}_f$, ,and scale it to range

[0,1] as follows:

$$v_{i,f} = ReLU\left(\sum_{k} \mathbf{g}_{i,f,k}\right) \tag{4.11}$$

$$IFIV_{i,f} = \frac{v_{i,f}}{max(\mathbf{v}_f)} \tag{4.12}$$

4.4 Evaluation of Outfit fashionability Measurement

4.4.1 Interpretability vs. Classification Performance

Naturally, there exists a trade-off between the interpretability and performance of the model: a more interpretable architecture (e.g. decision tree) usually has lower performance than an opaque architecture (e.g. artificial neural network). In our case, we create a more interpretable model by separating the RGB image into colors and *edge_image* and use them to examine the features separately. Here, we compare the performance of our models that are trained on interpretable item representations in section 4.3.2 to a similar model (as a baseline) that is trained on uninterpretable item representations [4].

Model architecture The architecture of both models is very similar since the only difference is the item representation $\mathbf{x}$ while other parts of the model are identical.

Based on the architecture shown in figure 4.2 and described in section 4.3, the item representation $\mathbf{x}$ of the baseline model is a feature vector extracted directly from the RGB item image using the feature extractor E. In other words, we replace these

equations in section 4.3:

$$\mathbf{x}^0_{edge\_image} = E(edge\_image) \tag{4.13}$$

$$\mathbf{x}_f = \frac{\mathbf{x}^0_f - \mu(\mathbf{X^0_f})}{\sigma(\mathbf{X^0_f})}, f \in \{edge\_image, colors\} \tag{4.14}$$

$$\mathbf{x} = [\mathbf{x}_{edge\_image}, \mathbf{x}_{colors}] \tag{4.15}$$

with

$$\mathbf{x} = E(RGB\_image) \tag{4.16}$$

The other parameters are as follows:

- The feature extractor E is the ImageNet-pretrained InceptionV3 [68] up to *pool5* layer which $\mathbb{R}^{2048}$ feature vector.
- The item encoder G is an identity function.
- The outfit encoder H is a 4096-d fully-connected (FC) layer with batch normalization [26] and ReLU [89] activation function.
- We train both models for 30 epochs with learning rate $1e-4$ and batch size 100 on Polyvore409k dataset [4].

Evaluation We define the accuracy as in traditional binary classification problems. The prediction is counted as correct if it matches with the ground truth.

Result Table 4.1 shows the results. Accuracy indicates that of binary classification, where a prediction is considered to be correct if it matches the ground truth. As

Table 4.1: Training, validation, and testing accuracy and average f1 of both outfit graders after training for 30 epochs on Polyvore409k dataset [4].

Partition	Train		Validation		Test	
Model	Acc.	Avg. F1	Acc.	Avg. F1	Acc.	Avg. F1
Baseline	98.41	98.20	83.19	81.86	79.19	74.11
Interpretable model	99.62	99.57	80.78	79.48	76.69	71.11

expected, the baseline model shows better performance than the interpretable model by 2.50% accuracy and 3.00% average f1. This is a noticeable gap but is arguably not so large to make the explanation by the interpretable model meaningless.

Performance of various configurations In addition to the model configuration we mentioned above, we also develop a number of configurations of the interpretable model to find a better model to be used to explain the fashionability of an outfit. The differences between these models and the one mentioned above are item encoder G and outfit encoder H.

Both item encoder G and outfit encoder H are implemented as a series of FC block. Each FC block consists of 3 layers: fully connected, batch normalization, and ReLU. The differences between models are the size and number of blocks that the item encoder G and outfit encoder H has. The configurations and their performance on testing samples are shown in table 4.2. Since the model#4 has the best performance, we will use this model in our outfit fashionability measurement system with feature influence values.

Table 4.2: Testing accuracy and average f1 of various configurations of outfit grader after training for 30 epochs of Polyvore409k dataset [4]. Each cell in the Item Encoder G and Outfit Encoder H column specify the size of FC layer in the FC block. The $\times$ indicates multiple FC blocks.

#	Item Encoder G	Outfit Encoder H	Acc.	Avg. F1
1	-	4096	76.69	71.11
2	1024	4096	78.93	73.43
3	512×256	4096	79.34	75.19
4	**512×256**	**2048**	**79.45**	**75.76**
5	256×128	4096	79.00	74.47
6	256×128	2048	78.78	74.70

4.4.2 The Effect of Temperature Scaling

As mentioned in section 4.3.4, The temperature scaling is used to calibrate the score of the outfit $\hat{q}$. Here, we use the reliability diagram [90,91], Expected Calibration Error [92], and the distribution of outfit scores to visualize the effect of temperature scaling on the scores of outfits.

To visualize the effect of temperature scaling, we split out testing samples into 10 interval bins, $S_0, S_1, \ldots, S_9$, by the scores (0-100) of testing samples from our model. For a set of samples, S_i, and their sample score, $Q_i = \{\hat{q}_s\}; s \in S_i$ in the bin i, the reliability diagram plots the expected accuracy of samples:

$$acc(S_i) = \frac{1}{|S_i|} \sum_{s \in S_i} 1(\hat{y}_s = y_s) \tag{4.17}$$

against the average confidence from the outfit scores:

$$\bar{Q}_i = \frac{1}{|S_i|} \sum_{s \in S_i} max(\hat{q}_s, 1 - \hat{q}_s) \tag{4.18}$$

If the model is perfectly calibrated, the classification accuracy of a set S_i should match its confidence, $acc(S_i) = \bar{Q}_i$.

We also use the Expected Calibration Error [92], which compute the difference in expectation between confidence and accuracy, as a scalar summary statistic of calibration, i.e.

$$\mathop{\mathbb{E}}_{\hat{P}} \left[\left| \mathbb{P}\left(\hat{Y} = Y | \hat{P} = p\right) - p \right| \right] \tag{4.19}$$

which, in our work, is estimated as:

$$ECE = \sum_{i=1}^{10} \frac{|S_i|}{N} \left| acc(S_i) - \bar{Q}_i \right| \tag{4.20}$$

where N is the total number of testing samples.

As a result of optimization on validation samples, the temperature T in equation 4.10 is set to 6.968055. The reliability diagram and outfit score distribution on each partition of Polyvore409k dataset [4], before and after temperature scaling, along with ECE values, are show in figure 4.4 and 4.5, respectively.

Both figure 4.4 and 4.5 show that the temperature scaling works well for calibrating the outfit scores, as the confidences match the prediction accuracies almost perfectly and ECE is reduced from 13.90 and 16.02 before the calibration to 1.10 and 2.16 after calibration, for validation and testing partition of Polyvore409k dataset [4]

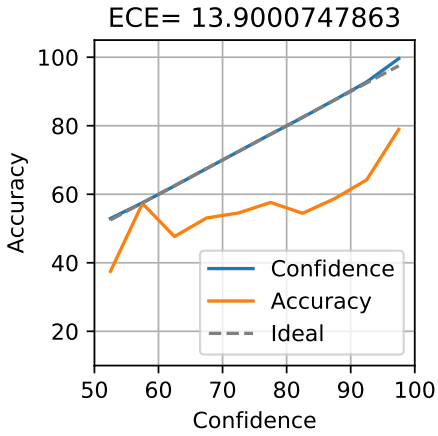
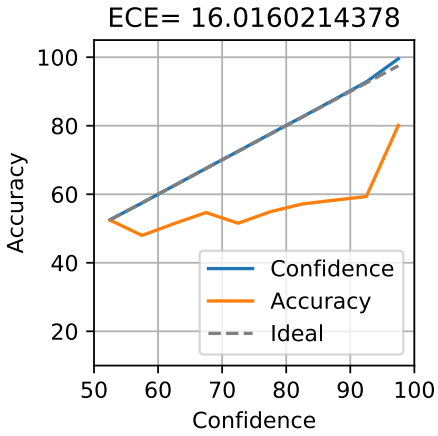
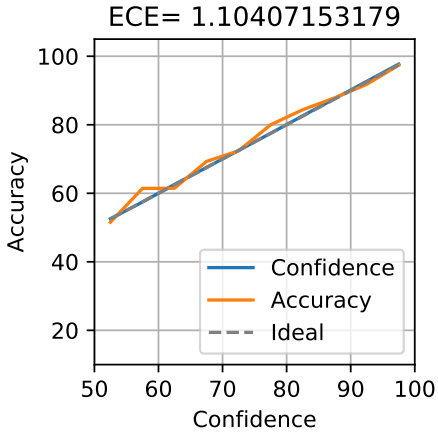
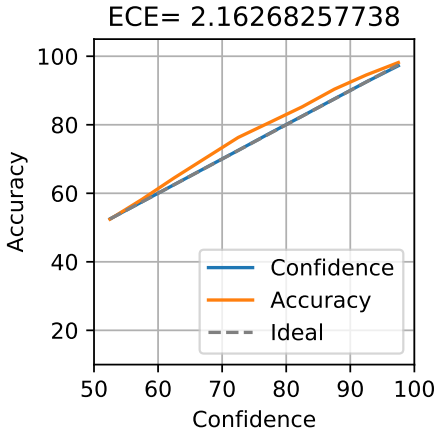
Partition	Validation	Test
Before temperature scaling		
After temperature scaling		

Figure 4.4: Reliability diagrams and ECE values before and after temperature scaling for validation and testing partition of Polyvore409k dataset [4]. Confidence is equivalent to the outfit score. Note that the ideal lines and confidence lines are almost the same.

respectively. The distribution of outfit scores also spreads more evenly compare to before the application of temperature scaling. The 8 best and worst outfits according to our outfit grader are shown in figure 4.6.

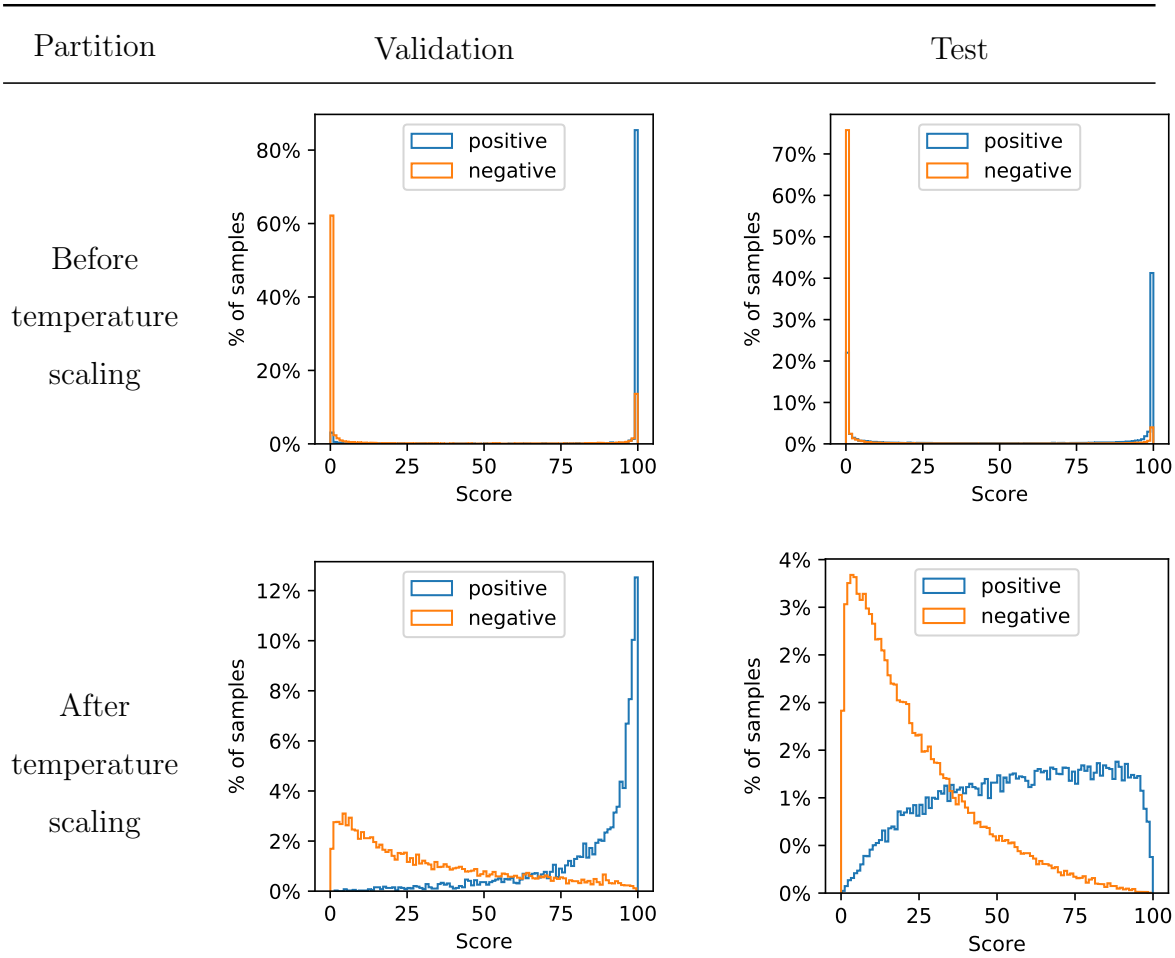


Figure 4.5: Distribution of outfit scores before and after temperature scaling for positive and negative samples in validation and testing partition of Polyvore409k dataset [4].

4.5 Outfit Flaw Detection as Item Feature Influence Evaluation

In this section, we evaluate our model's ability to output item feature influence values (IFIV) accurately. Since the IFIV estimates how each item affects the fashionability of overall outfit, both positively and negatively, we can evaluate the system by using it to identify the item-feature that negatively affect the outfit the most.

Figure 4.6: 8 best and worst outfits from testing partition of Polyvore409k dataset according to our outfit grader.

4.5.1 Evaluation Method

To evaluate our system in outfit flaw detection, we need outfit samples that contain a flaw which can be identified by the outfit part $\hat{i}$ that associated with the flawed item. We then use our system to obtain *negative IFIV* as in equation 4.12, that is, how each item-feature influences the outfit fashionability negatively. The outfit flaw prediction is the item-feature that the most negatively affects the outfit as:

$$i^* = \arg\max_i IFIV_{i,f} \tag{4.21}$$

where i^* is the outfit part associated with the predicted flawed item and f is the item-feature of interest. We consider the prediction i^* as correct if it matches with the ground truth flaw of the outfit $\hat{i}$.

4.5.2 Outfit Flaw Detection Samples

Since our goal here is to detect the flaw of an outfit and measure the accuracy, each sample may contain more than one flaw, but one of the flaws must completely dominate the others. Ideally, we would like to have an outfit that has no flaw, we then inject a major flaw into the outfit by replacing one of the item-feature in the outfit with other. To ensure that the substituted item-feature is the flaw, the overall fashionability of the outfit must be decreased. Since our system can also measure the outfit fashionability, we create the outfit flaw detection samples as follows:

1. Consider only 92,428 items and 26,813 positive outfit samples from the testing partition of Polyvore409k dataset [4].

2. Obtain 1,000 base samples by:

    2.1 Score all samples with the *512×256-2048* outfit grader from table 4.2.

    2.2 To ensure that the samples do not have any major flaw in the first place, we take only 1000 samples with the highest score from step 2.1 and use them as base samples. The average score of these base samples is 97.16 (out of 100).

3. For each outfit part i in each base sample, we obtain 10 mod samples by:

    3.1 Create 500 mod samples by replacing the item-feature f in the base sample with the one from a random item of the same outfit part.

    3.2 Use the same *512×256-2048* outfit grader to score those 500 mod samples.

    3.3 Use only 10 worst samples to ensure that the substituted item-feature is the flaw, not a complement to the base sample.

Since we are considering 2 interpretable features; *edge_image* and *colors*, we create 3 types of outfit flaw detection samples: *edge_image*-wise, *colors*-wise, and item-wise outfit flaw detection samples. For the first type, "*edge_image*-wise" , we replace only *edge_image* in step 3.1. For the second type, "*colors*-wise", only *colors* is replaced. And the third type, "item-wise", the entire item is replaced. The statistic of the base samples and the 3 types of outfit flaw detection sample, including number of samples by outfit part and by number of item in the sample for both base samples and outfit flaw detection samples, are shown in table 4.3. The distribution of score of base samples and those 3 types of outfit flaw detection samples is shown in figure 4.8. The distribution of the gap between the score of outfit flaw detection sample and its base sample is shown in figure 4.9. Two examples of base samples and their associated outfit flaw detection samples are shown in figure 4.7. Notice that the colors are not changed in *edge_image*-wise samples, and the edge_images are not changed in *colors*-wise samples.

4.5.3 Results

Figure 4.7 shows two base samples and the associated outfit flaw detection testing samples. For item-wise testing samples, both *edge_image* and *colors* of an item are replaced. For *edge_image*- and *colors*-wise testing samples, only *edge_image* or *colors* is replaced, respectively. In the table, the replaced feature(s) is enclosed with a red border, and the IFIVs indicate how each item negatively influence the fashionability of the outfit in range of [0,1].

Overall performance As shown in the table 4.4, our method can detection the flaw in outfits perfectly for *item*- and *edge_image*-wise testing samples as the overall

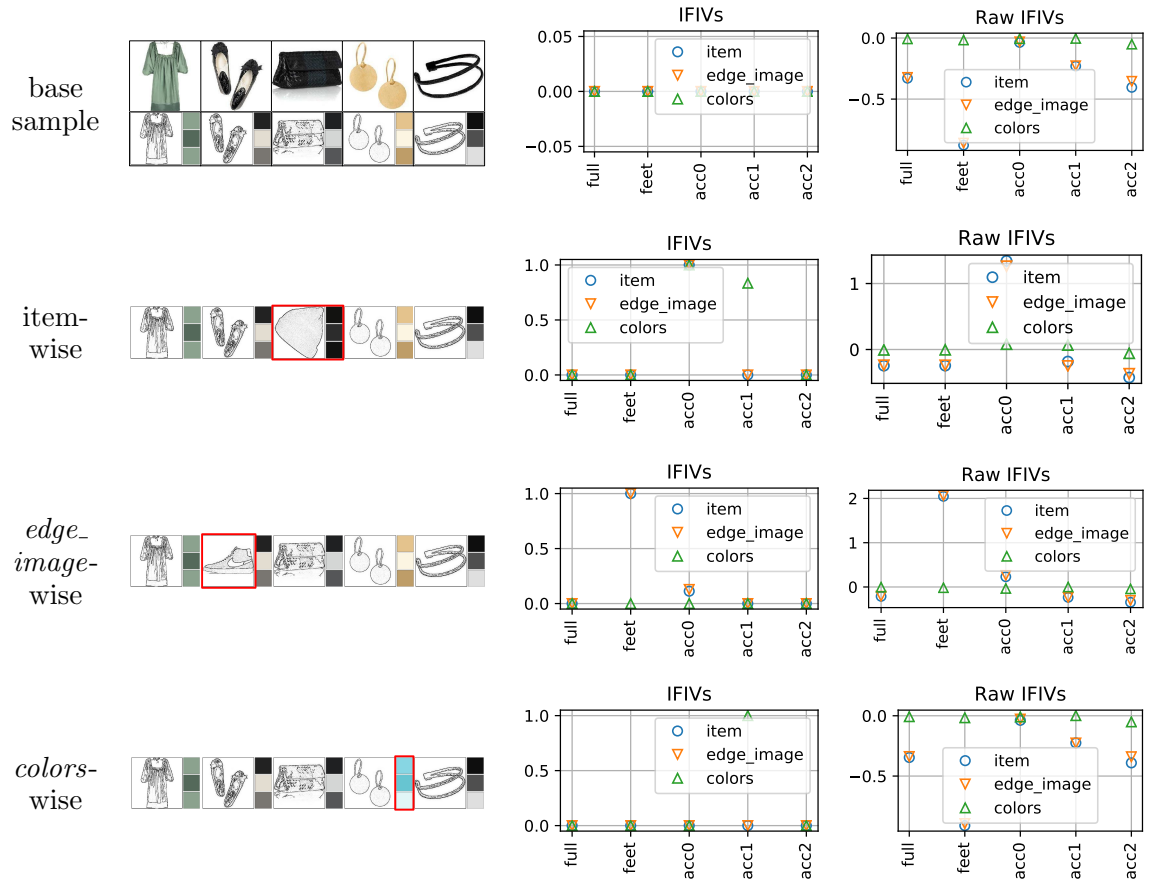


Figure 4.7: Examples of IFIVs computed by the proposed method. The red boxes indicate the replaced entities from the original high-quality outfits, which makes the new outfits have low outfit scores. IFIV scores mean negative IFIV values. Raw IFIV scores means the IFIVs before getting ReLU'd and scaled to [0,1].

accuracy is 99.52 and 99.49 percent respectively. While the 85.37 percent prediction accuracy for *colors*-wise samples is lower than the other two, it is still impressive compared to accuracy by chance which is at 17.86 percent.

Performance by gap of outfit score between base samples and the associated testing samples The score gaps are the difference between outfit fashionability (judge by the model) of the base samples to the associated outfit flaw detection

Table 4.3: Statistic of base samples and outfit flaw detection samples. Even though we have 3 types of outfit flaw detection samples for *edge_image*-wise, *colors*-wise, and item-wise, their statistic here are identical because the only difference between sample types are the feature(s) that have been modified.

Sample type	Number of samples containing following			
	outfit parts		number of items	
Base sample	Outer	385	3 items	15
	Upper	615	4 items	80
	Lower	664	5 items	364
	Full	379	6 items	375
	Feet	953	7 items	165
	Accessory0	977	8 items	1
	Accessory1	903	**Total**	**1,000**
	Accessory2	722		
Outfit flaw detection sample	Outer	3,850	3 items	450
	Upper	6,150	4 items	3,200
	Lower	6,640	5 items	18,200
	Full	3,790	6 items	22,500
	Feet	9,530	7 items	11,550
	Accessory0	9,770	8 items	80
	Accessory1	9,030	**Total**	**55,980**
	Accessory2	7,220		

samples. If the score gap is low, it means model did not see much difference in the outfit fashionability. As a result, it is also more difficult to identify the flaw since there are not many flaws to be identified in the first place. As shown in figure 4.10 and 4.9, the score gap in the *colors*-wise samples are much lower than the other two sample types, which is consistent with the accuracy values shown in the table 4.4.

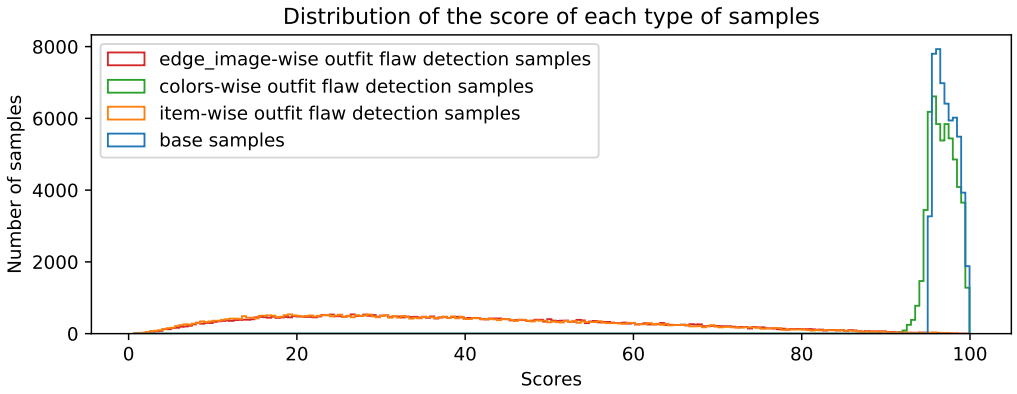
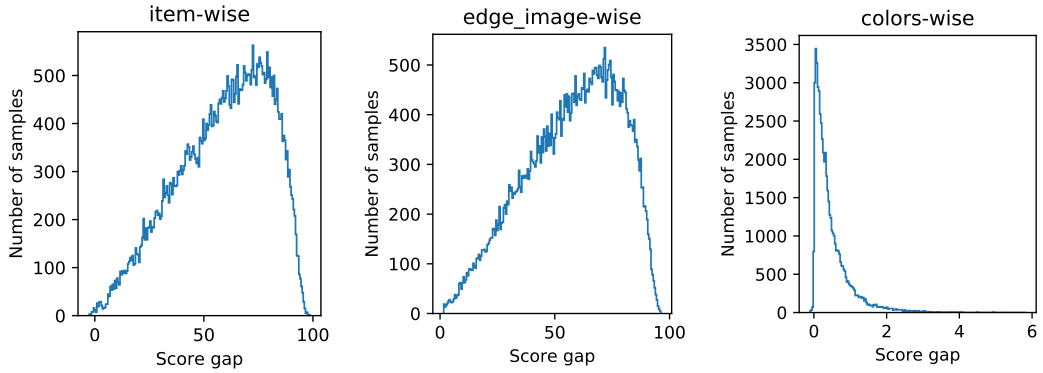


Figure 4.8: The distribution of scores of each type of sample.



(a) item-wise (b) *edge_image*-wise (c) *colors*-wise

Figure 4.9: Distributions of the score gap between each type of outfit flaw detection samples and their associated base samples. Notice the differences in the value range of each axis between *colors*-wise and others.

Table 4.4: Overall accuracy (%) of outfit flaw detection.

	sample type	prediction accuracy
By chance	all	17.86
Proposed method	item-wise	99.52
	edge_image-wise	99.49
	colors-wise	85.37

Table 4.5: Accuracy (%) of outfit flaw detection by number of items in an outfit.

	sample type	Number of items					
		3	4	5	6	7	8
By chance	all	33.33	25.00	20.00	16.67	14.29	12.50
Proposed method	item-wise	99.78	98.94	99.59	99.67	99.42	75.00
	edge_image-wise	99.33	98.75	99.54	99.66	99.47	75.00
	colors-wise	93.78	90.09	89.34	83.52	81.31	52.50

Table 4.6: Accuracy (%) of outfit flaw detection by outfit part. Note that there are 8 outfit parts in Polyvore409k dataset The *acc* outfit part is *accessory* and there are up to 3 accessories per outfit in this dataset.

	sample type	Outfit part							
		outer	upper	lower	full	feet	acc0	acc1	acc2
By chance	all	15.44	16.97	16.97	18.77	17.36	17.31	16.96	16.50
Proposed method	item-wise	99.25	99.11	99.94	95.57	100.00	99.90	99.99	99.96
	edge_image-wise	99.17	99.32	99.89	95.09	100.00	99.84	99.99	100.00
	colors-wise	78.91	79.93	86.07	85.65	93.05	86.82	85.71	80.11

Performance by number of items in outfits As shown in table 4.5, the accuracy values across the the number of items are quite consistence for *item-* and *edge_image*-wise samples, except at the outfit with 8 items. It cannot be said the same for *colors*-wise samples, though, as the score decreases as the number of items increases. However, as shown in table 4.3, there is only one out of 1,000 base samples that have 8 items, so the performance reported here may be unstable, and could be changed if the number of samples with 8 items increases.

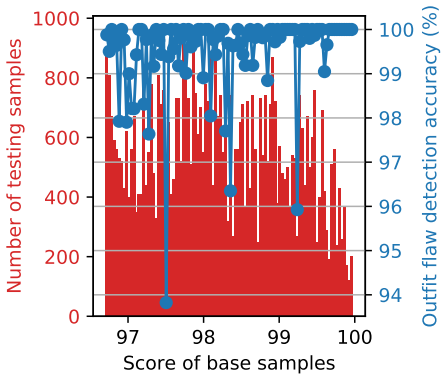
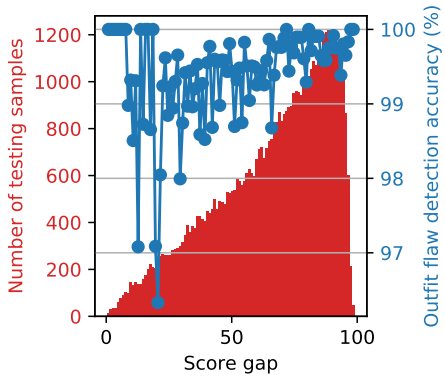
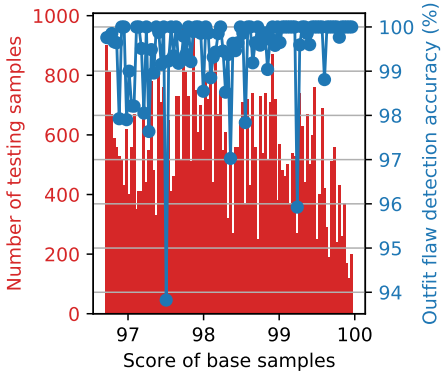
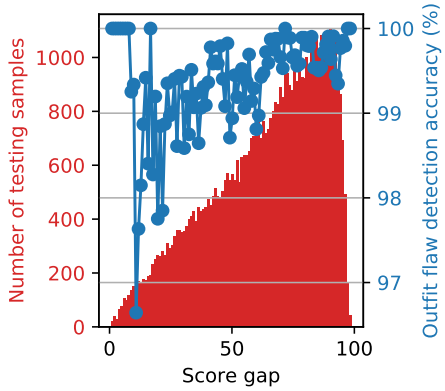
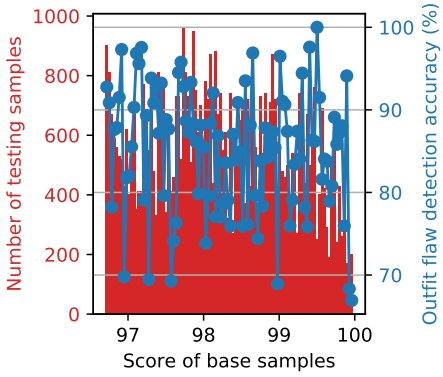
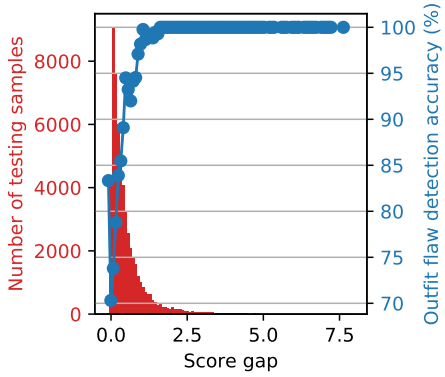
Sample type	Accuracy by/Distribution of testing samples by	
	Score of base samples	Score gap between base and outfit flaw detection samples
item-wise		
edge_image-wise		
colors-wise		

Figure 4.10: Accuracy and distribution of testing samples by score of base samples and score gap between testing samples and the associated base samples. Notice the differences in value range of score gaps between *colors*-wise testing samples and others.

Performance by the flawed outfit part in testing samples As shown in table 4.6, for *item*- and *edge_image*-wise testing samples, the outfit flaw detection performance are almost equal across all outfit parts, except the full outfit part, that is at around 95 percent accuracy while the rest is around 99 percent. For *colors*-wise testing samples, however, the accuracies are lower the other two types, and not as equal across the outfit parts.

4.6 Conclusion

In this chapter, we purposed a novel item-feature-wise outfit fashionability explanation technique using the purposed gradient-based method. This method is able to extract and quantify the effect of interpretable features of each item on the fashionability of an outfit both positively and negatively as *Item Feature Influence Value (IFIV)* without any additional item-level attribute annotation. Based on the proposed *IFIV* of each feature of each item in an outfit, we are able to detect the flaws in an outfit in feature level by finding the item-feature has the highest negative *IFIV*. Our outfit flaw detection experiments show that our method can detect the flaw in our testing samples effectively, at 99.52, 99.49, and 85.37 percent item-wise, *edge_image*-wise, and *colors*-wise, respectively. We believe that our work can provide insight into the outfit fashionability prediction, thus increase the trustworthy of the system. Since this work can point out the flaw in an outfit at the feature level, future work includes the outfit recommendation system that can improve the outfit fashionability effectively and be able to design fashion item based on the extracted features.

Chapter 5

Conclusions

Recent years have seen numerous attempt to apply computer vision to real-world applications, such as classification, semantic segmentation, etc. However, there are problems with unique characteristics in the fashion domain that require special care and specific approaches. For example, in semantic segmentation, the areas that look locally similar may have total different semantic or an outfit classification where the number of items can be varied but orderless, unlike both traditional image classification and natural language processing. Therefore, in this dissertation, we presented deep learning-based systems that leverage those special characteristics of data to address fashion specific problems.

In chapter 2, we proposed an extension to FCN architecture to solve the clothing parsing problem. The extension includes the side-path outfit encoder to predict a set of labels in the image, and CRF to produce a consistent label assignment both in terms of clothing semantics and structure within an image. Our model can learn from a pre-trained network and the existing annotated dataset without additional data. In addition, the learned image representation in the outfit encoder is useful

for similar dress-up styles thanks to the internal representation that encompasses combinatorial clothing semantics. This chapter also introduced a refined annotation to Fashionista dataset for better benchmarking of clothing parsing, built with a Web-based tool to create a high-resolution pixel-based annotation. The empirical study using the Fashionista and CFPD dataset shows that our method achieves state-of-the-art parsing performance.

So far, we only consider the relationship between item types in an outfit image. In chapter 3, we study outfits as combinations of items by developing outfit graders and outfit recommenders. Given a combination of items as an outfit, our best model can judge if the outfit looks good or not at over 84% accuracy on testing samples, and at 91% matching ratio on evaluations by human annotators. In addition, users can just give a pool of items that user have to our outfit recommender, and it will recommend outfits from the item pool. We also collect a large clothing dataset consisting of over 600,000 clothing items and over 400,000 outfits and use the dataset to learn and evaluate the outfit graders and recommenders.

To provide more transparency and trustworthy of the outfit quality measurement, in chapter 4, we purposed a novel item-feature-wise outfit quality explanation technique using the gradient-based method. This method is able to extract and quantify the effect of interpretable features of each item on the quality of an outfit both positively and negatively as *Item Feature Influence Value (IFIV)* without any additional item-level attribute annotation. Based on the proposed *IFIV* of each feature of each item in an outfit, we are able to detect the flaws in an outfit in feature level by finding the item-feature has the highest negative *IFIV*. Our outfit flaw detection experiments show that our method can detect the flaw in our testing samples effectively, at 99.52,

99.49, and 85.37 percent item-wise, *edge_image*-wise, and *colors*-wise, respectively. We believe that our work can provide insight into the outfit quality prediction, thus increase the trustworthy of the system. Since this work can point out the flaw in an outfit at the feature level, future work includes the outfit recommendation system that can improve the outfit quality effectively and be able to design fashion item based on the extracted features.

Bibliography

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[4] Pongsate Tangseng, Kota Yamaguchi, and Takayuki Okatani. Recommending outfits from personal closet. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, pages 269–277, 2018.

[5] Global fashion industry statistics - international apparel. `https://fashionunited.com/global-fashion-industry-statistics`. Accessed: 2018-12-26.

[6] It's official, first impression lasts forever - times of india. `https://timesofindia.indiatimes.com/home/science/Its-official-first-impression-lasts-forever/articleshow/7331355.cms`. Accessed: 2019-01-08.

[7] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Parsing clothing in fashion photographs. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, 2012.

[8] Kota Yamaguchi, M Hadi Kiapour, and Tamara L Berg. Paper doll parsing: Retrieving similar styles to parse clothing items. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3519–3526, 2013.

[9] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. A high performance crf model for clothes parsing. In *Asian Conference on Computer Vision*, pages 64–81, Cham, 2015.

[10] W. Yang, P. Luo, and L. Lin. Clothing co-parsing by joint image segmentation and labeling. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3182–3189, 2014.

[11] Si Liu, Jiashi Feng, Csaba Domokos, Hui Xu, Junshi Huang, Zhenzhen Hu, and Shuicheng Yan. Fashion parsing with weak color-category labels. *IEEE Transactions on Multimedia*, 16(1):253–265, 2014.

[12] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton 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.

[13] Kota Yamaguchi, Takayuki Okatani, Kyoko Sudo, Kazuhiko Murasaki, and Yukinobu Taniguchi. Mix and match: Joint model for clothing and attribute recognition. In *Proceedings of the British Machine Vision Conference*, pages 51.1–51.12, 2015.

[14] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[15] Jian Dong, Qiang Chen, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Towards unified human parsing and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 843–850, 2014.

[16] Lap-Fai Yu, Sai Kit Yeung, Demetri Terzopoulos, and Tony F Chan. Dressup!: outfit synthesis through automatic optimization. *ACM Transactions on Graphics*, 31(6):134, 2012.

[17] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel 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.

[18] Edgar Simo-Serra and Hiroshi Ishikawa. Fashion style in 128 floats: Joint ranking and classification using weak data for feature extraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 298–307, 2016.

[19] M Hadi Kiapour, Kota Yamaguchi, Alexander C Berg, and Tamara L Berg. Hipster wars: Discovering elements of fashion styles. In *Proceedings of the European Conference on Computer Vision*, pages 472–488, 2014.

[20] Yang Hu, Xi Yi, and Larry S Davis. Collaborative fashion recommendation: a functional tensor factorization approach. In *Proceedings of the 23rd ACM International Conference on Multimedia*, pages 129–138, 2015.

[21] Yuncheng Li, Liangliang Cao, Jiang Zhu, and Jiebo Luo. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Transactions on Multimedia*, 2017.

[22] Si Liu, Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan. Hi, magic closet, tell me what to wear! In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 619–628, 2012.

[23] Xishan Zhang, Jia Jia, Ke Gao, Yongdong Zhang, Dongming Zhang, Jintao Li, and Qi Tian. Trip outfits advisor: Location-oriented clothing recommendation. *IEEE Transactions on Multimedia*, 19(11):2533–2544, 2017.

[24] Zunlei Feng, Zhenyun Yu, Yezhou Yang, Yongcheng Jing, Junxiao Jiang, and Mingli 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.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[27] Si Liu, Zheng Song, Guangcan Liu, Changsheng Xu, Hanqing Lu, and Shuicheng Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3330–3337, 2012.

[28] George A Cushen and Mark S Nixon. Mobile visual clothing search. In *2013 IEEE International Conference on Multimedia and Expo Workshops*, pages 1–6, 2013.

[29] Wei Di, Catherine Wah, Anurag Bhardwaj, Robinson Piramuthu, and Neel Sundaresan. Style finder: Fine-grained clothing style detection and retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 8–13, 2013.

[30] Yannis Kalantidis, Lyndon Kennedy, and Li-Jia Li. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, pages 105–112, 2013.

[31] Rasmus Rothe, Marko Ristin, Matthias Dantone, and Luc Van Gool. Discriminative learning of apparel features. In *Proceedings of the 14th IAPR International Conference on Machine Vision Applications*, pages 5–9, 2015.

[32] Sirion Vittayakorn, Kota Yamaguchi, Alexander C Berg, and Tamara L Berg.

Runway to realway: Visual analysis of fashion. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 951–958, 2015.

[33] Xianwang Wang and Tong Zhang. Clothes search in consumer photos via color matching and attribute learning. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 1353–1356, 2011.

[34] M Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3343–3351, 2015.

[35] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1096–1104, 2016.

[36] Ziwei Liu, Sijie Yan, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Fashion landmark detection in the wild. In *Proceedings of the European Conference on Computer Vision*, pages 229–245, 2016.

[37] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[38] Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta, and Deva Ramanan. Pixelnet: Towards a general pixel-level architecture. *arXiv preprint arXiv:1609.06694*, 2016.

[39] Xiaodan Liang, Chunyan Xu, Xiaohui Shen, Jianchao Yang, Si Liu, Jinhui Tang, Liang Lin, and Shuicheng Yan. Human parsing with contextualized convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1386–1394, 2015.

[40] Vladimir Nekrasov, Janghoon Ju, and Jaesik Choi. Global deconvolutional networks for semantic segmentation. *arXiv preprint arXiv:1602.03930*, 2016.

[41] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[42] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[43] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[44] George Papandreou, Liang-Chieh Chen, Kevin Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv preprint arXiv:1502.02734*, 2015.

[45] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and

tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[46] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[47] Digvijay Singh Nataraj Jammalamadaka, Ayush Minocha and CV Jawahar. Parsing clothes in unrestricted images. In *Proceedings of the British Machine Vision Conference*, pages 88.1–88.11, 2013.

[48] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Retrieving similar styles to parse clothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(5):1028–1040, 2015.

[49] Jian Dong, Qiang Chen, Wei Xia, Zhongyang Huang, and Shuicheng Yan. A deformable mixture parsing model with parselets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3408–3415, 2013.

[50] Xiaodan Liang, Si Liu, Xiaohui Shen, Jianchao Yang, Luoqi Liu, Jian Dong, Liang Lin, and Shuicheng Yan. Deep human parsing with active template regression. *CoRR*, abs/1503.02391, 2015.

[51] Si Liu, Xiaodan Liang, Luoqi Liu, Ke Lu, Liang Lin, Xiaochun Cao, and Shuicheng Yan. Fashion parsing with video context. *IEEE Transactions on Multimedia*, 17(8):1347–1358, 2015.

[52] Si Liu, Xiaodan Liang, Luoqi Liu, Xiaohui Shen, Jianchao Yang, Changsheng Xu, Liang Lin, Xiaochun Cao, and Shuicheng Yan. Matching-cnn meets knn:

Quasi-parametric human parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1419–1427, 2015.

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

[54] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems*, pages 109–117, 2011.

[55] Ming-Ming Cheng, V A Prisacariu, Shuai Zheng, Philip H. S. Torr, and Carsten Rother. DenseCut: Densely Connected CRFs for Realtime GrabCut. *Computer Graphics Forum*, 34(7), 2015.

[56] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, 2014.

[57] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[58] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 1799–1807, 2014.

[59] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and

Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4642–4650, 2015.

[60] Jose Oramas and Tinne Tuytelaars. Modeling visual compatibility through hierarchical mid-level elements. *arXiv preprint arXiv:1604.00036*, 2016.

[61] Wei-Lin Hsiao and Kristen Grauman. Learning the latent "look": Unsupervised discovery of a style-coherent embedding from fashion images. *arXiv preprint arXiv:1707.03376*, 2017.

[62] Junshi Huang, Rogerio S Feris, Qiang Chen, and Shuicheng Yan. Cross-domain image retrieval with a dual attribute-aware ranking network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1062–1070, 2015.

[63] Pongsate Tangseng, Zhipeng Wu, and Kota Yamaguchi. Looking at outfit to parse clothing. *arXiv preprint arXiv:1703.01386*, 2017.

[64] Kota Yamaguchi, Tamara L Berg, and Luis E Ortiz. Chic or social: Visual popularity analysis in online fashion networks. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 773–776, 2014.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[66] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning fashion compatibility with bidirectional lstms. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 1078–1086, 2017.

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

[68] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[69] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[70] Jifeng Dai, Kaiming He, and Jian 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.

[71] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

[72] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei 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.

[73] Mateusz Malinowski, Marcus Rohrbach, and Mario 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.

[74] Mengye Ren, Ryan Kiros, and Richard Zemel. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961, 2015.

[75] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[76] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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.

[77] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[78] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv 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.

[79] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *arXiv preprint arXiv:1704.05796*, 2017.

[80] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[81] Justin Johnson, Andrej Karpathy, and Li 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.

[82] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru 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.

[83] Kevin Matzen, Kavita Bala, and Noah Snavely. StreetStyle: Exploring world-wide clothing styles from millions of photos. *arXiv preprint arXiv:1706.01869*, 2017.

[84] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten de Rijke. Explainable fashion recommendation with joint outfit matching and comment generation. *arXiv preprint arXiv:1806.08977*, 2018.

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

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

[87] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[88] John 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.

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

[90] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *The Statistician*, pages 12–22, 1983.

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

[92] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2901–2907, 2015.

List of Publications

Proceedings of International Conferences

1. Pongsate Tangseng, Kota Yamaguchi, and Takayuki Okatani, "Recommending outfits from personal closet", *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 269–277, 2018

    (a) Pongsate Tangseng, Kota Yamaguchi, and Takayuki Okatani, "Recommending outfits from personal closet", *Proceedings of IEEE International Conference on Computer Vision Workshop (ICCVW)*, pp. 2275–2279, 2017

Non-Peer Reviewed Publication

1. Pongsate Tangseng, Zhipeng Wu, and Kota Yamaguchi, "Looking at outfit to parse clothing", *arXiv preprint arXiv:1703.01386*, 2017.

2. Pongsate Tangseng, and Takayuki Okatani, "Toward explainable fashion recommendation", *arXiv preprint arXiv:1901.04870*, 2019.

Acknowledgments

It would not have been possible to write this dissertation without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

Above all, I would like to express my special appreciation and thanks to my supervisor *Professor Takayuki Okatani* who always gives me valuable advises not only the scientific knowledge but also the fundamental of life for living in Japan while doing scientific research. I feel very honored to have the opportunity to participate as part of his laboratory. Being able to do the research with him is a very precious experience for me.

The good advice, support and friendship of *Associate Professor Kota Yamaguchi* has been invaluable on both an academic and a personal level, for which I am extremely grateful.

I am most grateful to *Assistant Professor Mete Ozay* and *Assistant Professor Masanori Suganuma* for providing me valuable advises and his kindness.

I would also like to thank all of the laboratory members for having precious time in the lab together, helping everything when I asked for help. Thank you very much for your kindness, friendship and support.

For the secretary of the laboratory, *Mrs. Akemi Sakane*, I would like to express my special thank for helping me with every kind of problems since the time I contacted for entering this laboratory.

A special thanks to my family. Words cannot express how grateful I am to my mother and father for all of the sacrifices that you have made on my behalf.

Last, but by no means least, I would like express appreciation to my friends from Japan, Thailand, China and elsewhere for their support and encouragement.

For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely my own.