Fashion++: Minimal Edits for Outfit Improvement

Wei-Lin Hsiao[1,4] Isay Katsman[*2,4] Chao-Yuan Wu[*1,4] Devi Parikh[3,4] Kristen Grauman[1,4]
[1]UT Austin [2]Cornell Tech [3]Georgia Tech [4] Facebook AI Research

Abstract

Given an outfit, what small changes would most improve its fashionability? This question presents an intriguing new vision challenge. We introduce Fashion++*, an approach that proposes minimal adjustments to a full-body clothing outfit that will have maximal impact on its fashionability. Our model consists of a deep image generation neural network that learns to synthesize clothing conditioned on learned per-garment encodings. The latent encodings are explicitly factorized according to shape and texture, thereby allowing direct edits for both fit/presentation and color/patterns/material, respectively. We show how to bootstrap Web photos to automatically train a fashionability model, and develop an activation maximization-style approach to transform the input image into its more fashionable self. The edits suggested range from swapping in a new garment to tweaking its color, how it is worn (e.g., rolling up sleeves), or its fit (e.g., making pants baggier). Experiments demonstrate that Fashion++ provides successful edits, both according to automated metrics and human opinion.*

1. Introduction

"Before you leave the house, look in the mirror and take one thing off." – Coco Chanel

The elegant Coco Chanel's famous words advocate for making small changes with large impact on fashionability. Whether removing an accessory, selecting a blouse with a higher neckline, tucking in a shirt, or swapping to pants a shade darker, often small adjustments can make an existing outfit noticeably more stylish. This strategy has practical value for consumers and designers alike. For everyday consumers, recommendations for how to edit an outfit would allow them to tweak their look to be more polished, rather than start from scratch or buy an entirely new wardrobe. For fashion designers, envisioning novel enhancements to familiar looks could inspire new garment creations.

Motivated by these observations, we introduce a new computer vision challenge: *minimal edits for outfit improvement*. To minimally edit an outfit, an algorithm must propose alterations to the garments/accessories that are slight, yet visibly improve the overall fashionability. A "minimal" edit need not strictly minimize the amount of change; rather, it *incrementally adjusts* an outfit as opposed to starting from scratch. It can be recommendations on which garment to put on, take off, or swap out, or even how to wear the same garment in a better way. See Figure 1.

Figure 1: Minimal outfit edits suggest minor changes to an existing outfit in order to improve its fashionability. For example, changes might entail (left) removing an accessory; (middle) changing to a blouse with higher neckline; (right) tucking in a shirt.

This goal presents several technical challenges. First, there is the question of training. A natural supervised approach might curate pairs of images showing better and worse versions of each outfit to teach the system the difference; however, such data is not only very costly to procure, it also becomes out of date as trends evolve. Secondly, even with such ideal pairs of images, the model needs to distinguish very subtle differences between positives and negatives (sometimes just a small fraction of pixels as in Fig. 1), which is difficult for an image-based model. It must reason about the parts (garments, accessories) within the original outfit and how their synergy changes with any candidate tweak. Finally, the notion of *minimal* edits implies that adjustments may be sub-garment level, and the inherent properties of the person wearing the clothes—*e.g.*, their pose, body shape—should not be altered.

Limited prior work explores how to recommend a garment for an unfinished outfit [9, 13, 31, 44] (*e.g.*, the fill-in-the-blank task). Not only is their goal different from ours, but they focus on clean per-garment catalog photos, and their recommendations are restricted to *retrieved garments* from a dataset. However, we observe that in the fash-

[*] Authors contributed equally.

ion domain, the problem demands going beyond seeking an existing garment to add—to also inferring which garments are detrimental and should be taken off, and how to adjust the presentation and details of each garment (*e.g.*, cuff the jeans above the ankle) within a complete outfit to improve its style.

We introduce a novel image generation approach called Fashion++ to address the above challenges. The main idea is an activation maximization [33] method that operates on localized encodings from a deep image generation network. Given an original outfit, we map its composing pieces (*e.g.*, bag, blouse, boots) to their respective codes. Then we use a discriminative fashionability model as an editing module to gradually update the encoding(s) in the direction that maximizes the outfit's score, thereby improving its style. The update trajectory offers a spectrum of edits, starting from the least changed and moving towards the most fashionable, from which users can choose a preferred end point. We show how to bootstrap Web photos of fashionable outfits, together with automatically created "negative" alterations, to train the fashionability model. [1] To account for both the pattern/colors and shape/fit of the garments, we factorize each garment's encoding to texture and shape components, allowing the editing module to control where and what to change (*e.g.*, tweaking a shirt's color while keeping its cut vs. changing the neckline or tucking it in).

After optimizing the edit, our approach provides its output in two formats: 1) retrieved garment(s) from an inventory that would best achieve its recommendations and 2) a rendering of the same person in the newly adjusted look, generated from the edited outfit's encodings. Both outputs aim to provide *actionable* advice for small but high-impact changes for an existing outfit.

We validate our approach using the Chictopia dataset [24] and, through both automated metrics and user studies, demonstrate that it can successfully generate minimal outfit edits, better than several baselines. Fashion++ offers a unique new tool for data-driven fashion advice and design—a novel image generation pipeline relevant for a real-world application.

2. Related Work

Recognition for fashion. Most prior fashion work addresses recognition problems, like matching street-to-shop [18, 20, 26, 46], searching for products interactively [8, 22, 56], and recognizing garments [27].

Fashion image synthesis. Synthesis methods explore ways to map specified garments to new poses or people. This includes generating a clothed person conditioned on a product image [10, 47, 52] (and vice versa [53]), or conditioned on textual descriptions (*e.g.*, "a woman dressed in sleeveless white clothes") [37, 61], as well as methods for swapping clothes between people [36, 55] or synthesizing a clothed person in unseen poses [2, 3, 23, 28, 35, 40, 57]. Whereas these problems render people in a target garment or body pose, we use image synthesis as a communication tool to make suggestions to minimally edit outfits.

Image manipulation, translation, and style transfer are also popular ways to edit images. There is a large base of literature for generating realistic images conditioned on semantic label maps [16, 48, 58–60], edge maps [38, 51], or 3D models [25, 50], using generative adversarial networks (GANs) [7]. Related ideas are explored in interactive image search, where users specify visual attributes to alter in their query [8, 22, 56]. Style transfer methods [4–6, 14] offer another way to edit images that turn photographs into artwork. Unlike previous work that conditions on segment maps, maps are *generated* in our case; as a result, we enable sub-object shape changes that alter regions' footprints, which generalizes fashion image synthesis. Most importantly, all these works aim to edit images according to *human specified input*, whereas we aim to *automatically* suggest where and how to edit to *improve* the input.

Compatibility and fashionability. Fashionability refers to the popularity or stylishness of clothing items, while compatibility refers to how well-coordinated individual garments are. Prior work recommends garments retrieved from a database that go well together [9, 11–13, 15, 17, 43–45], or even garments generated from GANs [39]. Some also recommend interchangeable items [9, 31, 44] that are equally compatible, or forecast future fashion trends [1]. We address a new and different problem: instead of recommending compatible garments from scratch, our approach tweaks an existing outfit to make it more compatible/fashionable. It can suggest removals, revise a garment, optimize fashionability, and identify *where* to edit—none of which is handled by existing methods. Using online "likes" as a proxy for fashionability, the system in [41] suggests—in words—garments or scenery a user should change to improve fashionability; however, it conditions on meta-data rather than images, and suggests coarse properties specified in words (*e.g.*, Navy and Bags, Black Casual) that often dictate changing to an entirely new outfit.

Activation maximization. Activation maximization [33] is a gradient based approach that optimizes an image to highly activate a target neuron in a neural network. It is widely used for visualizing what a network has learned [29, 34, 42, 49, 54], and recently to synthesize images [19, 32]. In particular, [19] also generates clothing images, but they generate single-garment products rather than full body outfits. In addition, they optimize images to match purchase history, not to improve fashionability.

[1] Fashionability refers to the stylishness of an outfit, the extent to which it agrees with current trends. As we will see in Sec. 3.2, our model defines fashionability by popular clothing choices people wear in Web photos, which can evolve naturally over time with changing trends.

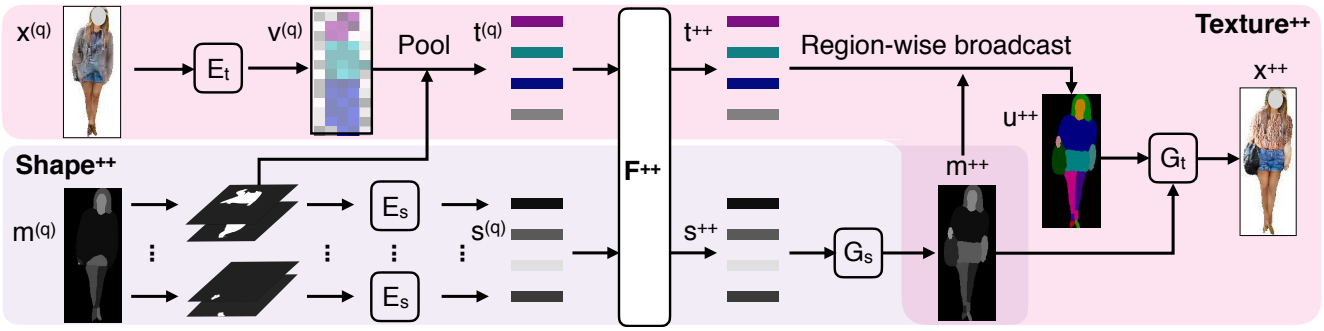


Figure 2: Overview of our Fashion++ framework. We first obtain latent features from texture and shape encoders E_t and E_s. Our editing module F^{++} operates on the latent texture feature $\mathbf{t}$ and shape feature $\mathbf{s}$. After an edit, the shape generator G_s first decodes the updated shape feature $\mathbf{s}^{++}$ back to a 2D segmentation mask $\mathbf{m}^{++}$, and then we use it to region-wise broadcast the updated texture feature $\mathbf{t}^{++}$ into a 2D feature map $\mathbf{u}^{++}$. This feature map and the updated segmentation mask are passed to the texture generator G_t to generate the final updated outfit $\mathbf{x}^{++}$. See Supp. for architecture details.

3. Approach

Minimal editing suggests changes to an existing outfit such that it remains similar but noticeably more fashionable. To address this newly proposed task, there are three desired objectives: (1) training must be scalable in terms of supervision and adaptability to changing trends; (2) the model could capture subtle visual differences and the complex synergy between garments that affects fashionability; and (3) edits should be localized, doing as little as swapping one garment or modifying its properties, while keeping fashion-irrelevant factors unchanged.

In the following, we first present our image generation framework, which decomposes outfit images into their garment regions and factorizes shape/fit and texture, in support of the latter two objectives (Sec. 3.1). Then we present our training data source and discuss how it facilitates the first two objectives (Sec. 3.2). Finally, we introduce our activation maximization-based outfit editing procedure and show how it recommends garments (Sec. 3.3).

3.1. Fashion++ Outfit Generation Framework

The coordination of all composing pieces defines an outfit's look. To control which parts (shirt, skirt, pants) and aspects (neckline, sleeve length, color, pattern) to change—and also keep identity and other fashion-irrelevant factors unchanged—we want to explicitly model their spatial locality. Furthermore, to perform minimal edits, we need to control pieces' *texture* as well as their *shape*. Texture often decides an outfit's theme (style): denim with solid patterns gives more casual looks, while leather with red colors gives more street-style looks. With the same materials, colors, and patterns of garments, how they are worn (*e.g.*, tucked in or pulled out) and the fit (*e.g.*, skinny vs. baggy pants) and cut (*e.g.*, a V-neck vs. turtleneck) of a garment will complement a person's silhouette in different ways. Accounting for all these factors, we devise an image generation framework that both gives control over individual pieces (garments, accessories, body parts) and also factorizes shape (fit and cut) from texture (color, patterns, materials).

Our system has the following structure at test time: it first maps an outfit image $\mathbf{x}^{(q)}$ and its associated semantic segmentation map $\mathbf{m}^{(q)}$ to a texture feature $\mathbf{t}^{(q)}$ and a shape feature $\mathbf{s}^{(q)}$. Our editing module, F^{++}, then gradually updates $\mathbf{t}^{(q)}$ and $\mathbf{s}^{(q)}$ into $\mathbf{t}^{++}$ and $\mathbf{s}^{++}$ to improve fashionability. Finally, based on $\mathbf{t}^{++}$ and $\mathbf{s}^{++}$, the system generates the output image(s) of the edited outfit $\mathbf{x}^{++}$. Fig. 2 overviews our system. Superscripts (q) and $++$ denote variables before and after editing, respectively. We omit the superscript when clear from context. We next describe how our system maps an outfit into latent features.

Texture feature. An input image $\mathbf{x} \in X \subseteq \mathbb{R}^{H\times W\times C}$ is a real full-body photo of a clothed person. It is accompanied by a region map $\mathbf{m} \in M \subseteq \mathbb{Z}^{H\times W}$ assigning each pixel to a region for a clothing piece or body part. We use $n = 18$ unique region labels defined in Chictopia10k [24]: face, hair, shirt, pants, dress, hats, etc. We first feed $\mathbf{x}$ into a learned texture encoder $E_t : X \rightarrow V$ that outputs a feature map $\mathbf{v} \in V \subseteq \mathbb{R}^{W\times H\times d_t}$. Let r_i be the region associated with label i. We average pool $\mathbf{v}$ in r_i to obtain the texture feature $\mathbf{t}_i = \mathcal{F}^i_{pool}(\mathbf{v}, \mathbf{m}) \in \mathbb{R}^{d_t}, \forall i$. The whole outfit's texture feature is represented as $\mathbf{t} := [\mathbf{t}_0; \ldots; \mathbf{t}_{n-1}] \in \mathbb{R}^{n\cdot d_t}$. See Fig. 2 top left.

Shape feature. We also develop a shape encoding that allows per-region shape control separate from texture control. Specifically, we construct a binary segmentation map $\mathbf{m}_i \in M_B \in \{0, 1\}^{H\times W}$ for each region r_i, and use a shared shape encoder $E_s : M_B \rightarrow S$ to encode each $\mathbf{m}_i$ into a shape feature $\mathbf{s}_i \in S \in \mathbb{R}^{d_s}$. The whole outfit's shape feature is represented as $\mathbf{s} := [\mathbf{s}_0; \ldots; \mathbf{s}_{n-1}] \in \mathbb{R}^{n\cdot d_s}$. See Fig. 2 bottom left.

Image generation. To generate an image, we first use a shape generator G_s that takes in whole-body shape feature $\mathbf{s}$ and generates an image-sized region map $\widehat{\mathbf{m}} \in M$. We then

perform region-wise broadcasting, which broadcasts $\mathbf{t}_i$ to all locations with label i based on $\widehat{\mathbf{m}}$, and obtain the *texture feature map* $\mathbf{u} = \mathcal{F}_{broad}(\mathbf{t}, \widehat{\mathbf{m}}) \in \mathbb{R}^{H \times W \times d_t}$.[2] Finally, we channel-wise concatenate $\mathbf{u}$ and $\widehat{\mathbf{m}}$ to construct the input to a texture generator G_t, which generates the final outfit image. This generation process is summarized in Fig. 2 (right). Hence, the generators G_t and G_s learn to reconstruct outfit images conditioned on garment shapes and textures.

Training. Although jointly training the whole system is possible, we found a decoupled strategy to be effective. Our insight is that if we assume a fixed semantic region map, the generation problem is reduced to an extensively studied image translation problem, and we can benefit from recent advances in this area. In addition, if we separate the shape encoding and generation from the whole system, it reduces to an auto-encoder, which is also easy to train.

Specifically, for the image translation part (Texture++ in Fig. 2), we adapt from conditional generative adversarial networks (cGANs) that take in segmentation label maps and associated feature maps to generate photo-realistic images [48, 60]. We combine the texture encoder E_t and texture generator G_t with a discriminator D to formulate a cGAN. An image $\widehat{x}$ is generated by $G_t(\mathbf{m}, \mathbf{u})$, where $\mathbf{u} = \mathcal{F}(E_t(\mathbf{x}), \mathbf{m})$, and $\mathcal{F}$ is the combined operations of $\mathcal{F}^i_{pool}, \forall i$ and $\mathcal{F}_{broad}$. The discriminator D aims to distinguish real images from generated ones. E_t, G_t and D are learned simultaneously with a minimax adversarial game objective:

$$G_t^*, E_t^* = \underset{G_t, E_t}{\operatorname{argmin}} \max_{D} \mathcal{L}_{\text{GAN}}(G_t, D, E_t) + \mathcal{L}_{\text{FM}}(G_t, E_t, D), \tag{1}$$

where $\mathcal{L}_{\text{GAN}}$ is defined as:

$$\mathbb{E}_{(\mathbf{m},\mathbf{x})}\big(\log D(\mathbf{m}, \mathbf{x}) + \log\big(1 - D\big(\mathbf{m}, G_t\left(\mathbf{m}, \mathbf{u}\right)\big)\big)\big) \tag{2}$$

for all training images $\mathbf{x}$, and $\mathcal{L}_{\text{FM}}$ denotes feature matching loss.

For the shape deformation part of our model (Shape++ in Fig. 2), we formulate a shape encoder and generator with a region-wise Variational Autoencoder (VAE) [21]. The VAE assumes the data is generated by a directed graphical model $p(\mathbf{m}|\mathbf{s})$ and the encoder learns an approximation $q_{E_s}(\mathbf{s}|\mathbf{m})$ to the posterior distribution $p(\mathbf{s}|\mathbf{m})$. The prior over the encoded feature is set to be Gaussian with zero mean and identity covariance, $p(\mathbf{s}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The objective of our VAE is to minimize the Kullback-Leibler (KL) divergence between $q_{E_s}(\mathbf{s}|\mathbf{m})$ and $p(\mathbf{s})$, and the ℓ_1 reconstruction loss:

$$D_{\text{KL}}\big(q_{E_s}(\mathbf{s}|\mathbf{m}) \| p(\mathbf{s})\big) + \mathbb{E}_{\mathbf{m}} \left\| \mathbf{m} - G_s\big(E_s(\mathbf{m})\big) \right\|_1. \tag{3}$$

Note that simply passing in the 2D region label map as the shape encoding $\mathbf{s}$ would be insufficient for image editing. The vast search space of all possible masks is too difficult to model, and, during editing, mask alterations could often yield unrealistic or uninterpretable "fooling" images [34, 42]. In contrast, our VAE design learns the probability distribution of the outfit shapes, and hence can generate unseen shapes corresponding to variants of features from the learned distribution. This facilitates meaningful shape edits.

[2]Note that $\mathbf{u}$ has uniform features for a region, since it is average-pooled, while $\mathbf{v}$ is not.

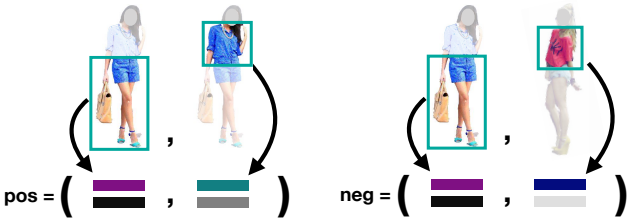


Figure 3: Forming training examples: A fashionable Web photo is the positive (left). We overwrite some garment's features with those from another *distant* outfit to create the negative (right). (Here only two of n garment regions are shown for simplicity.)

Having defined the underlying image generation architecture, we next introduce our editing module for revising an input's features (encodings) to improve fashionability.

3.2. Learning Fashionability from Web Photos

Our editing module (Sec. 3.3) requires a discriminative model of fashionability, which prompts the question: how can we train a fashionability classifier for minimal edits? Perhaps the ideal training set would consist of pairs of images in which each pair shows the same person in slightly different outfits, one of them judged to be more fashionable than the other. However, such a collection is not only impractical to curate at scale, it would also become out of date as soon as styles evolve. An alternative approach is to treat a collection of images from a specific group (*e.g.*, celebrities) as positive exemplars and another group (*e.g.*, everyday pedestrians) as negatives. However, we found such a collection suffers from conflating identity and style, and thus the classifier finds fashion-irrelevant properties discriminative between the two groups.

Instead, we propose to bootstrap less fashionable photos automatically from Web photos of fashionable outfits. The main idea is to create "negative" outfits from fashionista photos. We start with a Chictopia full-body outfit photo (a "positive"), select one of its pieces to alter, and replace it with a piece from a different outfit. To increase the probability that the replacement piece degrades fashionability, we extract it from an outfit that is most dissimilar to the original one, as measured by Euclidean distance on CNN features. We implement the garment swap by overwriting the encoding $\mathbf{z}_i := [\mathbf{t}_i; \mathbf{s}_i]$ for garment i with the target's. See Fig. 3.

We use this data to train a 3-layer multilayer perceptron (MLP) fashionability classifier f. It is trained to map the encoding $\mathbf{z} := [\mathbf{t}; \mathbf{s}]$ for an image $\mathbf{x}$ to its binary fashion-

ability label $y \in \{0, 1\}$.

The benefit of this training strategy is threefold: First, it makes curating data easy, and also refreshes easily as styles evolve—by downloading new positives. Second, by training the fashionability classifier on these decomposed (to garments) and factorized (shape vs. texture) encodings, a simple MLP effectively captures the subtle visual properties and complex garment synergies (see Supp. for ablation study). Finally, we stress that our approach learns from full-body outfit photos being worn by people on the street, as opposed to clean catalog photos of individual garments [9,11,39,43–45]. This has the advantages of allowing us to learn aspects of fit and presentation (*e.g.*, tuck in, roll up) that are absent in catalog data, as well as the chance to capture organic styles based on what outfits people put together in the wild.

3.3. Editing an Outfit

With the encoders E_t, E_s, generators G_t, G_s and editing module F^{++} in hand, we now explain how our approach performs a minimal edit. Given test image $\mathbf{x}^{(q)}$, Fashion++ returns its edited version(s):

$$\mathbf{x}^{++} := G\left(F^{++}\left(E(\mathbf{x}^{(q)})\right)\right), \tag{4}$$

where G and E represent the models for both shape and texture. When an inventory of discrete garments is available, our approach also returns the nearest real garment g_i^{++} for region i that could be used to achieve that change, as we will show in results. Both outputs—the rendered outfit and the nearest real garment—are complementary ways to provide actionable advice to a user.

Computing an edit. The main steps are: calculating the desired edit, and generating the edited image. To calculate an edit, we take an activation maximization approach: we iteratively alter the outfit's feature such that it increases the activation of the fashionable label according to f.

Formally, let $\mathbf{z}^{(0)} := \{\mathbf{t}_0, \mathbf{s}_0, \ldots, \mathbf{t}_{n-1}, \mathbf{s}_{n-1}\}$ be the set of all features in an outfit, and $\tilde{\mathbf{z}}^{(0)} \subseteq \mathbf{z}^{(0)}$ be a subset of features corresponding to the *target regions or aspects* that are being edited (*e.g.*, shirt region, shape of skirt, texture of pants). We update the outfit's representation as:

$$\tilde{\mathbf{z}}^{(k+1)} := \tilde{\mathbf{z}}^{(k)} + \lambda \frac{\partial p_f\left(y = 1 | \mathbf{z}^{(k)}\right)}{\partial \tilde{\mathbf{z}}^{(k)}}, k = 0, \ldots, K-1 \tag{5}$$

where $\tilde{\mathbf{z}}^{(k)}$ denotes the features after k updates, $\mathbf{z}^{(k)}$ denotes substituting only the target features in $\mathbf{z}^{(0)}$ with $\tilde{\mathbf{z}}^{(k)}$ while keeping other features unchanged, $p_f(y = 1|\mathbf{z}^{(k)})$ denotes the probability of fashionability according to classifier f, and λ denotes the update step size. Each gradient step in Eqn (5) yields an incremental adjustment to the input outfit. Fig. 4 shows the process of taking 10 gradient steps with step size 0.1 (see Sec. 4 for details). By presenting this spectrum of edits to the user, one may choose a preferred end point (i.e., his/her preferred tradeoff in the "minimality" of change vs. maximality of fashionability). Finally, as above, $\mathbf{z}^{(K)}$ gives the updated $\mathbf{t}_i^{++}; \mathbf{s}_i^{++}, \forall i$.

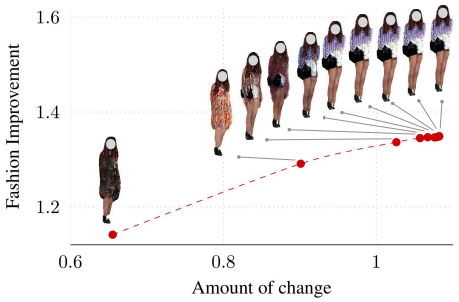


Figure 4: As Fashion++ iteratively edits the outfit, the fashionability improves and eventually saturates as the outfit becomes fashionable enough. (Metrics defined in Sec. 4.1. Dots show average result for *all* test images.)

To further force updates to stay close to the original, one could add a proximity objective, $\|\mathbf{z}^{(k)} - \mathbf{z}^{(0)}\|$, as in other editing work [25, 58]. However, balancing this smoothness term with other terms (users' constraints in their cases, fashionability in ours) is tricky (*e.g.*, [25] reports non-convergence). We found our gradient step approach to be at least as effective to achieve gradual edits.

Optimizing where to edit. A garment for region i is represented as the concatenation of its texture and shape features: $\mathbf{z}_i^{(0)} := [\mathbf{t}_i; \mathbf{s}_i]$. Our approach optimizes the garment that ought to be edited by cycling though all garments to find the one with most impact:

$$i^* = \underset{i=0,\ldots,n-1}{\operatorname{argmax}} \left\| \frac{\partial p_f(y = 1|\mathbf{z}^{(0)})}{\partial \mathbf{z}_i{}^{(0)}} \right\|. \tag{6}$$

By instructing the target $\tilde{\mathbf{z}}^{(0)}$ to be $\mathbf{z}_{i^*}{}^{(0)}$, we can simultaneously optimize *where and how to change* an outfit.

Rendering the edited image. Then we generate the Fashion++ image output by conditioning our image generators G_t, G_s on these edits:

$$\mathbf{x}^{++} = G_t(\mathbf{m}^{++}, \mathbf{u}^{++}), \tag{7}$$

where $\mathbf{u}^{++}$ refers to the broadcasted map of the edited texture components $\mathbf{t}^{++}$, and $\mathbf{m}^{++} = G_s(\mathbf{s}^{++})$ refers to the VAE generated mask for the edited shape components $\mathbf{s}^{++}$. The full edit operation is outlined in Fig. 2.

In this way, our algorithm automatically updates the latent encodings to improve fashionability, then passes its revised code to the image generator to create the appropriate image. An edit could affect as few as one or as many as n garments, and we can control whether edits are permitted for shape or texture or both. This is useful, for example, if we wish to insist that the garments look about the same, but be edited to have different tailoring or presentation (e.g., roll up sleeves)—shape changes only.

Retrieving a real garment matching the edit. Finally, we return the garment(s) g_i^{++} that optimally achieves the edited outfit. Let $\mathcal{I}$ denote an inventory of garments. The best matching garments to retrieve from $\mathcal{I}$ are:

$$g_i{}^{++} := \underset{g_i \in \mathcal{I}}{\operatorname{argmin}} \left\| \mathbf{z}_{g_i} - \mathbf{z}_i{}^{++} \right\|, \tag{8}$$

for $i = 0, \ldots, n-1$, where $\mathbf{z}_{g_i}$ denotes the garment's feature. This is obtained by passing the real inventory garment image for g_i to the texture and shape feature encoders E_t and E_s, and concatenating their respective results.

4. Experiments

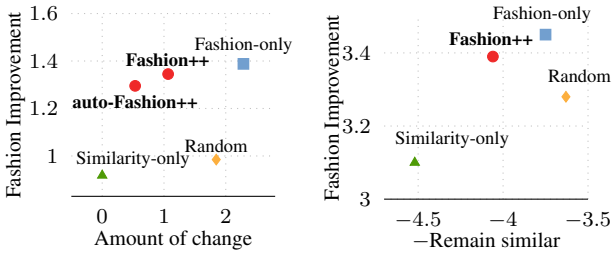
We now validate that Fashion++ (i) makes slight yet noticeable improvements better than baseline methods in both quantitative evaluation (Sec. 4.1) and user studies (Sec. 4.2); (ii) effectively communicates to users through image generation (Sec. 4.2); and (iii) supports all possible edits from swapping, adding, removing garments to adjusting outfit presentations via qualitative examples (Sec. 4.3).

Experiment setup. We use the Chictopia10k [24] dataset for all experiments. We use $15,930$ images to train the generators, and $12,744$ to train the fashionability classifier. We use the procedure described in Sec. 3.2 to prepare positive and negative examples for training the fashionability classifier. We evaluate on $3,240$ such unfashionable examples. We stress that all test examples are from *real world outfits*, bootstrapped by swapping *features* (not pixels) of pieces from different outfits. This allows testing on real data while also having ground truth (see below). We use the region maps provided with Chictopia10k for all methods, though automated semantic segmentation could be used. Model architectures and training details are in Supp.

Baselines. Since our work is the first to consider the minimal edit problem, we develop several baselines for comparison: SIMILARITY-ONLY, which selects the nearest neighbor garment in the database $\mathcal{I}$ (Chictopia10k) to maintain the least amount of change; FASHION-ONLY, which changes to the piece that gives the highest fashionability score as predicted by our classifier, using the database $\mathcal{I}$ as candidates; RANDOM SAMPLING, which changes to a randomly sampled garment. Since all unfashionable outfits are generated by swapping out a garment, we instruct all methods to update that garment. We additionally run results where we automatically determine the garment to change, denoted auto-Fashion++.

4.1. Quantitative comparison

Minimal edits change an outfit by improving its fashionability while not changing it too much. Thus, we evaluate performance simultaneously by *fashionability improvement* and *amount of change*. We evaluate the former by how much the edit gets closer to the ground-truth (GT) outfit. Since each unfashionable outfit is generated by swapping to a garment (we will call it original) from another outfit, and the garment before the swap (we will call it GT) is just one possibility for a fashionable outfit, we form a *set of GT garments* per test image, representing the multiple ways to improve it (see Supp. for details). The *fashion improvement* metric is the ratio of the original piece's distance to the GT versus the edited piece's distance to the GT. Values less than one mean no improvement. The *amount of change* metric scores the edited garment's distance to the original garment, normalized by subtracting SIMILARITY ONLY's number. All distances are Euclidean distance in the generators' encoded space. All methods return the garment in the inventory nearest to their predicted encoding.



(a) **Automatic evaluation.** (b) **Human subject study B.**

Figure 5: For both automatic (a) and human (b) evaluation, Fashion++ best balances improving fashionability while remaining similar. In (b), both axes are the raw Likert scale; we negate the x-axis so that its polarity agrees to the left.

Fig. 5a shows the results.[3] SIMILARITY-ONLY changes the outfit the least, as expected, but it does not improve fashionability. FASHION-ONLY improves fashionability the most, but also changes the outfit significantly. RANDOM neither improves fashionability nor remains similar. Our Fashion++ improves fashionability nearly as well as the FASHION-ONLY baseline, while remaining as similar to the original outfit as SIMILARITY-ONLY. Auto-Fashion++ performs similarly to Fashion++. These results support our claim that Fashion++ makes slight yet noticeable improvements.

Fig. 4 shows that by controlling the amount of change (number of gradient steps) made by Fashion++, one can choose whether to *change less* (while still being more fashionable than SIMILARITY-ONLY) or *improve fashionability more* (while still changing less than FASHION-ONLY).

4.2. Human perceptual study

Next we ask humans to judge the quality of Fashion++'s edits, how it compares with baselines, and whether they know what actions to take to improve outfits based on

[3] We plot ours with $K = 6$ for clarity and since fashionability typically saturates soon after. Results for all K values are in Fig. 4 and Sec. 4.2.

the edits. We perform three human subject test protocols; please see Supp. for all three user interfaces. We randomly sample 100 unfashionable test outfits and post tasks on Mechanical Turk (MTurk). Each sample is answered by 7 people, and in total 282 Turkers answered.

Protocol A. Fashion++ can show users a spectrum of edits (*e.g.*, Fig. 4) from which to choose the desired version. While preference will naturally vary among users, we are interested in knowing to what extent a given degree of change is preferred and why. To this end, we show Turkers an original outfit and edits from $K = 1$ to 10, and ask them to: (i) Select all edits that are more fashionable than the original. (ii) Choose which edit offers the best balance in improving the fashionability without changing too much. (iii) Explain why the option selected in (ii) is best.

For (i), we found that the more we change an outfit (increasing K), the more often human judges think the changed outfit becomes fashionable, with 92% of the changed outfits judged as more fashionable when $K = 10$. Furthermore, when we apply Fashion++ to an already fashionable outfit, 84% of the time the human judges find the changed outfit to be similarly or more fashionable, meaning Fashion++ "does no harm" in most cases (see Supp.). For (ii), no specific K dominates. The top selected $K = 2$ is preferred 18% of the time, and $K = 1$ to 6 are each preferred at least 10% of the time. This suggests that results for $K \leq 6$ are similarly representative, so we use $K = 6$ for remaining user studies. For (iii), a common reason for a preferred edit is being more *attractive*, *catchy*, or *interesting*. See Supp. for detailed results breaking down K for (i) (ii) and more Turkers' verbal explanations for (iii).

Protocol B. Next we ask human judges to compare Fashion++ to the baselines defined above. We give workers a pair of images at once: one is the original outfit and the other is edited by a method (Fashion++ or a baseline). They are asked to express their agreement with two statements on a five point Likert scale: (i) The changed outfit is more fashionable than the original. (ii) The changed outfit remains similar to the original. We do this survey for all methods. We report the median of the 7 responses for each pair.

Fig. 5b shows the result. It aligns very well with our quantitative evaluation in Fig. 5a: FASHION-ONLY is rated as improving fashionability the most, but it also changes outfits as much as RANDOM. SIMILARITY-ONLY is rated as remaining most similar. Fashion++ changes more than SIMILARITY-ONLY but less than all others, while improving fashionability nearly as much as FASHION-ONLY. This strongly reinforces that Fashion++ makes edits that are slight yet improve fashionability.

Protocol C. Finally, it is important that no matter how good the image's exact pixel quality is, humans can get *actionable information* from the suggested edits to improve outfits. We thus ask Turkers how "actionable" our edit is on a five point Likert scale, and to verbally describe the edit. 72% of the time human judges find our images actionable, rating the clarity of the actionable information as $4.16 \pm 0.41/5$. (4 for *agree* and 5 for *strongly agree*). See Supp. for Turkers' verbal descriptions of our edits.

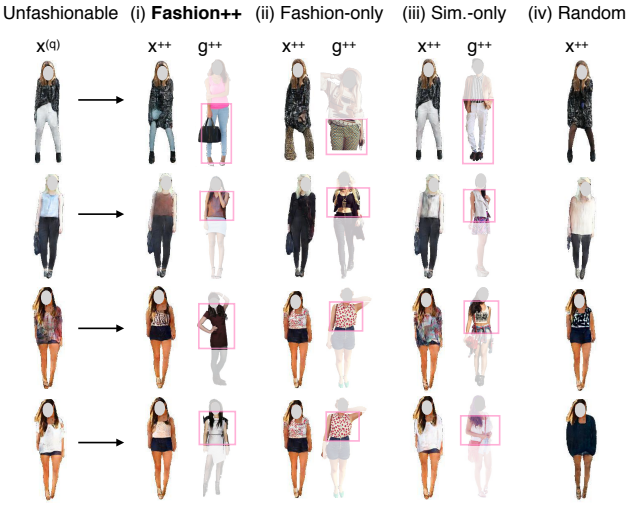


Figure 6: Minimal edit comparisons with baselines. Rows are instances, columns are results for methods: For all but RANDOM (iv), we show both the rendered (left) and retrieved (right) results. Retrieved garments g_i^{++} are in bounding boxes. Best on pdf.

4.3. Minimal edit examples

Now we show example outfit edits. We first compare side-by-side with the baselines, and then show variants of Fashion++ to demonstrate its flexibility. For all examples, we show outfits both before and after editing as reconstructed by our generator.

General minimal edits comparing with baselines. Fig. 6 shows examples of outfit edits by all methods as well as the retrieved nearest garments. Both FASHION-ONLY (ii) and RANDOM (iv) change the outfit a great deal. While RANDOM makes outfits less fashionable, FASHION-ONLY improves them with more stylish garments. Fashion++ (i) also increases fashionability, and the recommended change bears similarity (in shape and/or texture) to the initial less-fashionable outfit. For example, the bottom two instances in Fig. 6 wear the same shorts with different shirts. FASHION-ONLY recommends changing to the same white blouse with a red floral print for both instances, which looks fashionable but is entirely different from the initial shirts; Fashion++ recommends changing to a striped shirt with a similar color palette for the first one, and changing to a sleeveless shirt with a slight blush for the second. SIMILARITY-ONLY (iii) indeed looks similar to the initial outfit, but stylishness also remains similar.

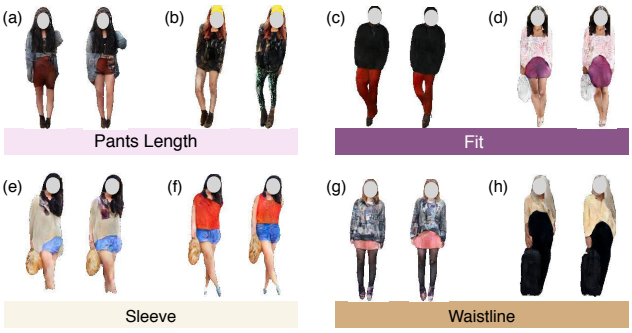


Figure 7: Fashion++ minimal edits on only shape/fit.

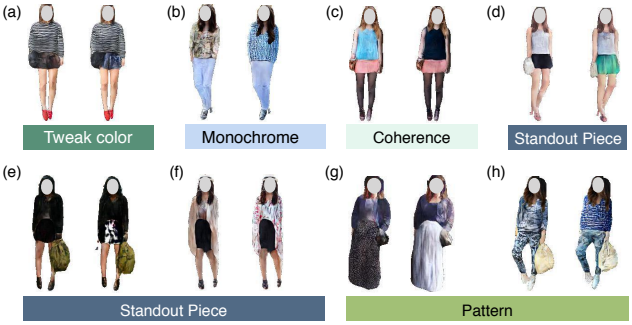


Figure 8: Fashion++ minimal edits on only color/pattern.

Minimal edits changing only shapes. Fig. 7 shows examples when we instruct our model to just change the shape (cf. Sec 3.3). Even with the exact same pieces and person, adjusting the clothing proportions and fit can favorably affect the style. Fig. 7 (a) shows the length of pants changing. Notice how changing where the shorts end on the wearer's legs lengthens them. (b,c) show changes to the fit of pants/skirt: wearing pieces that fit well emphasizes wearers' figures. (d) wears the same jacket in a more open fashion that gives character to the look. (e,f) roll the sleeves up: slight as it is, it makes an outfit more energetic (e) or dressier (f). (g,h) adjusts waistlines: every top and bottom combination looks different when tucked tightly (g) or bloused out a little (h), and properly adjusting this for different ensembles gives better shapes and structures.

Minimal edits changing only textures. Fig. 8 shows examples when we instruct our model to just change the texture. (a) polishes the outfits by changing the bottom a tint lighter. (b) changes the outfit to a monochrome set that lengthens the silhouette. (c) swaps out the incoherent color. (d)-(f) swap to stand-out pieces by adding bright colors or patterns that make a statement for the outfits. (g)-(h) are changing or removing patterns: notice how even with the same color components, changing their proportions can light up outfits in a drastic way.

Figure 9: Fashion++ edits that add/remove clothing pieces.

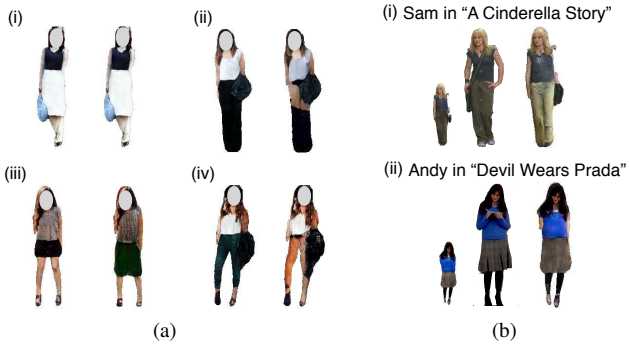


Figure 10: (a): Some failure cases of Fashion++; (b): Fashion++ on notoriously unfashionable characters.

Beyond changing existing pieces. Not only can we tweak pieces that are already on outfits, but we can also take off redundant pieces and even put on new pieces. Fig. 9 shows such examples. In (a), the girl is wearing a stylish dress, but together with somewhat unnecessary pants. (b) suggests to add outerwear to the dress for more layers, while (c) takes off the dark outerwear for a lighter, more energetic look. (d) changes pants to skirt for a better figure of the entire outfit.

Failure cases. A minimal edit requires good outfit generation models, an accurate fashionability classifier, and robust editing operations. Failure in any of these aspects can result in worse outfit changes. Fig. 10a shows some failure examples as judged by Turkers.

Editing celebrities. Fig. 10b shows Fashion++ operating on movie characters known to be unfashionable.

5. Conclusions

We introduced the minimal fashion edit problem. Minimal edits are motivated by consumers' need to tweak existing wardrobes and designers' desire to use familiar clothing as a springboard for inspiration. We introduced a novel image generation framework to optimize and display minimal edits yielding more fashionable outfits, accounting for essential technical issues of locality, scalable supervision, and flexible manipulation control. Our results are quite promising, both in terms of quantitative measures and human judge opinions. In future work, we plan to broaden the composition of the training source, e.g., using wider social media platforms like Instagram [30], bias an edit towards an available inventory, or generate improvements conditioned on an individual's preferred style or occasion.

References

[1] Z. Al-Halah, R. Stiefelhagen, and K. Grauman. Fashion forward: Forecasting visual style in fashion. In *ICCV*, 2017. 2

[2] Guha Balakrishnan, Amy Zhao, Adrian V Dalca, Fredo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *CVPR*, 2018. 2

[3] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018. 2

[4] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015. 2

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 2

[6] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *CVPR*, 2017. 2

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[8] X. Guo, H. Wu, Y. Cheng, S. Rennie, and R. Feris. Dialog-based interactive image retrieval. In *NIPS*, 2018. 2

[9] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. Learning fashion compatibility with bidirectional lstms. *ACM MM*, 2017. 1, 2, 5

[10] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In *CVPR*, 2018. 2

[11] R. He, C. Packer, and J. McAuley. Learning compatibility across categories for heterogeneous item recommendation. In *ICDM*, 2016. 2, 5

[12] Wei-Lin Hsiao and Kristen Grauman. Learning the latent "look": Unsupervised discovery of a style-coherent embedding from fashion images. In *ICCV*, 2017. 2

[13] Wei-Lin Hsiao and Kristen Grauman. Creating capsule wardrobes from fashion images. In *CVPR*, 2018. 1, 2

[14] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2

[15] C. Huynh, A. Ciptadi, A. Tyagi, and A. Agrawal. Craft: Complementary recommendation by adversarial feature transform. In *ECCV Workshop on Computer Vision For Fashion, Art and Design*, 2018. 2

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 2

[17] T. Iwata, S. Watanabe, and H. Sawada. Fashion coordinates recommender system using photographs from fashion magazines. In *IJCAI*, 2011. 2

[18] Y. Kalantidis, L. Kennedy, and L.-J. Li. Getting the look: Clothing recognition and segmentation for automatic product suggestions in everyday photos. In *ICMR*, 2013. 2

[19] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. Visually-aware fashion recommendation and design with generative image models. In *ICDM*, 2017. 2

[20] M. Hadi Kiapour, X. Han, and S. Lazebnik. Where to buy it: Matching street clothing photos in online shops. In *ICCV*, 2015. 2

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 4

[22] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Image search with relative attribute feedback. In *CVPR*, 2012. 2

[23] Christoph Lassner, Gerard Pons-Moll, and Peter V Gehler. A generative model of people in clothing. In *ICCV*, 2017. 2

[24] Xiaodan Liang, Si Liu, Xiaohui Shen, Jianchao Yang, Luoqi Liu, Jian Dong, Liang Lin, and Shuicheng Yan. Deep human parsing with active template regression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2015. 2, 3, 6

[25] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3d modeling with a generative adversarial network. In *Proceedings of the International Conference on 3D Vision*, 2017. 2, 5

[26] 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 *CVPR*, 2012. 2

[27] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016. 2

[28] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017. 2

[29] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision (IJCV)*, 2016. 2

[30] K. Matzen, K. Bala, and N. Snavely. Streetstyle: Exploring world-wide clothing styles from millions of photos. *arXiv:1706.01869*, 2017. 8

[31] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015. 1, 2

[32] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 2017. 2

[33] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, 2016. 2

[34] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015. 2, 4

[35] Albert Pumarola, Antonio Agudo, Alberto Sanfeliu, and Francesc Moreno-Noguer. Unsupervised person image synthesis in arbitrary poses. In *CVPR*, 2018. 2

[36] Amit Raj, Patsorn Sangkloy, Huiwen Chang, James Hays, Duygu Ceylan, and Jingwan Lu. Swapnet: Image based garment transfer. In *ECCV*, 2018. 2

[37] Negar Rostamzadeh, Seyedarian Hosseini, Thomas Boquet, Wojciech Stokowiec, Ying Zhang, Christian Jauvin, and Chris Pal. Fashion-gen: The generative fashion dataset and challenge. *arXiv preprint arXiv:1806.08317*, 2018. 2

[38] Patsorn Sangkloy, Jingwan Lu, Chen Fang, FIsher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, 2017. 2

[39] Yong-Siang Shih, Kai-Yueh Chang, Hsuan-Tien Lin, and Min Sun. Compatibility family learning for item recommendation and generation. In *Proceedings AAAI*, 2018. 2, 5

[40] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuilière, and Nicu Sebe. Deformable gans for pose-based human image generation. In *CVPR*, 2018. 2

[41] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. Neuroaesthetics in Fashion: Modeling the Perception of Fashionability. In *CVPR*, 2015. 2

[42] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014. 2, 4

[43] Xuemeng Song, Fuli Feng, Jinhuan Liu, and Zekun Li. Neurostylist: Neural compatibility modeling for clothing matching. *ACM MM*, 2017. 2, 5

[44] Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. Learning type-aware embeddings for fashion compatibility. In *ECCV*, 2018. 1, 2, 5

[45] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *ICCV*, 2015. 2, 5

[46] Sirion Vittayakorn, Kota Yamaguchi, Alexander C Berg, and Tamara L Berg. Runway to realway: Visual analysis of fashion. In *WACV*, 2015. 2

[47] Bochao Wang, Huabin Zheng, Xiaodan Liang, Yimin Chen, Liang Lin, and Meng Yang. Toward characteristic-preserving image-based virtual try-on network. In *ECCV*, 2018. 2

[48] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018. 2, 4

[49] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside cnn. *arXiv preprint arXiv:1507.02379*, 2015. 2

[50] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, 2016. 2

[51] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *CVPR*, 2018. 2

[52] Shan Yang, Tanya Ambert, Zherong Pan, Ke Wang, Licheng Yu, Tamara Berg, and Ming C Lin. Detailed garment recovery from a single-view image. *arXiv preprint arXiv:1608.01250*, 2016. 2

[53] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *ECCV*, 2016. 2

[54] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *ICML*, 2015. 2

[55] Mihai Zanfir, Alin-Ionut Popa, Andrei Zanfir, and Cristian Sminchisescu. Human appearance transfer. In *CVPR*, 2018. 2

[56] B. Zhao, J. Feng, X. Wu, and S. Yan. Memory-augmented attribute manipulation networks for interactive fashion search. In *CVPR*, 2017. 2

[57] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, Zequn Jie, and Jiashi Feng. Multi-view image generation from a single-view. In *ACMMM*, 2018. 2

[58] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2, 5

[59] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 2

[60] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017. 2, 4

[61] Shizhan Zhu, Sanja Fidler, Raquel Urtasun, Dahua Lin, and Change Loy Chen. Be your own prada: Fashion synthesis with structural coherence. In *CVPR*, 2017. 2