

# Creating Capsule Wardrobes from Fashion Images

Wei-Lin Hsiao  
UT-Austin

kimhsiao@cs.utexas.edu

Kristen Grauman  
UT-Austin

grauman@cs.utexas.edu

## Abstract

We propose to automatically create capsule wardrobes. Given an inventory of candidate garments and accessories, the algorithm must assemble a minimal set of items that provides maximal mix-and-match outfits. We pose the task as a subset selection problem. To permit efficient subset selection over the space of all outfit combinations, we develop submodular objective functions capturing the key ingredients of visual compatibility, versatility, and user-specific preference. Since adding garments to a capsule only expands its possible outfits, we devise an iterative approach to allow near-optimal submodular function maximization. Finally, we present an unsupervised approach to learn visual compatibility from “in the wild” full body outfit photos; the compatibility metric translates well to cleaner catalog photos and improves over existing methods. Our results on thousands of pieces from popular fashion websites show that automatic capsule creation has potential to mimic skilled fashionistas in assembling flexible wardrobes, while being significantly more scalable.

## 1. Introduction

The fashion domain is a magnet for computer vision. New vision problems are emerging in step with the fashion industry’s rapid evolution towards an online, social, and personalized business. Style models [22, 38, 34, 17, 27], trend forecasting [1], interactive search [44, 24], and recommendation [41, 18, 31] all require visual understanding with rich detail and subtlety. Research in this area is poised to have great influence on what people buy, how they shop, and how the fashion industry analyzes its enterprise.

A *capsule wardrobe* is a set of garments that can be assembled into many visually compatible outfits (see Fig. 1). Capsules are currently the purview of fashion experts, magazine editors, and bloggers. A capsule creator manually analyzes an inventory to puzzle together a relatively small set of clothing items and accessories that mix and match well. A curated capsule can help consumers get the best value for their dollars (“do more with less”), help vendors pro-



Figure 1: A *capsule wardrobe* is a minimal set of garments that can mix and match to compose many visually compatible outfits.

pose appealing wardrobes from their catalogs, and help a subscription service (e.g., StitchFix, LeTote) ship a targeted box that amplifies a customer’s wardrobe.

We propose to automate capsule wardrobe generation. There are two key technical challenges. First, capsules hinge on having an accurate model of *visual compatibility*. Whereas visual *similarity* asks “what looks like this?”, and is fairly well understood [11, 32, 42, 30], *compatibility* instead asks “what complements this?” It requires capturing how multiple visual items interact, often according to subtle visual properties. Existing compatibility methods [41, 14, 35, 16, 31, 39] assume supervision via labels or co-purchase data, which limits scope and precision, as we will discuss in Sec. 3. Furthermore, they largely cater only to pairwise compatibility. The second challenge is that capsule generation is a complex combinatorial problem. Of all possible garments, we seek the subset that maximizes versatility and compatibility, and, critically, the addition of any one garment introduces *multiple* new outfit combinations.

We introduce an approach to automatically create capsule wardrobes that addresses both of these issues. We first cast capsule creation as a subset selection problem. We de-

fine an objective function characterizing a capsule based on its pieces’ mutual compatibility, the resulting outfits’ versatility, and (optionally) its faithfulness to a user’s preferred style. Then, we develop an efficient algorithm that maps a large inventory of candidate garments into the best capsule of the desired size. We design objectives that are submodular for the addition of new *outfits*, ensuring the “diminishing returns” property that facilitates near-optimal set selection [36, 28]. Then, since each *garment* added to a capsule *expands* the possible outfits, we further develop an iterative approach that exploits outfit submodularity to alternate between fixing and selecting each layer of clothing.

As a second main contribution, we introduce an unsupervised approach to learn visual compatibility from full-body images “in the wild”. We learn a generative model for outfit compositions from unlabeled images that can score  $k$ -way compatibility. Because it is built on predicted attributes, our model can translate compatibility learned from the “in the wild” photos to cleaner catalog photos of individual items, where users need most guidance on mixing and matching.

We evaluate our approach on thousands of garments from Polyvore, popular social commerce websites for fashion. We compare our algorithm’s capsule creations to those manually defined by fashionistas, as well as subjective user studies. Furthermore, we show our underlying compatibility model offers advantages over some state of the art methods. Finally, we demonstrate the practical value of our algorithm, which in seconds finds near-optimal capsules for problem scales that are otherwise intractable.

## 2. Related Work

**Attributes for fashion** Attributes offer a natural representation for clothing, since they can describe relevant patterns (*checked, paisley*), colors (*rose, teal*), fit (*loose*), and cut (*V-neck, flowing*) [4, 2, 8, 43, 6, 23, 33]. Topic models on attributes are indicative of styles [20, 40, 17]. Inspired by [17], we employ topic models. However, whereas [17] seeks a style-coherent image embedding, we use correlated topic models to score novel combinations of garments for their compatibility. Domain adaptation [6, 19] and multi-task curriculum learning [9] are valuable to overcome the gap between street and shop photos. We devise a simple curriculum learning approach to train attributes effectively in our setting. None of the above methods explore visual compatibility or capsule wardrobes.

**Style and fashionability** Beyond recognition tasks, fashion also demands answering: How do we represent *style*? What makes an outfit *fashionable*? The style of an outfit is typically learned in a supervised manner. Leveraging style-labeled data like HipsterWars [22] or Deep-Fashion [33], classifiers built on body keypoints [22], weak meta-data [38], or contextual embeddings [27] show promise. Fashionability refers specifically to a style’s pop-

ularity. It can also be learned from supervised data, e.g., online data for user “likes” [29, 37]. Unsupervised style discovery methods instead mine unlabeled photos to detect common themes in people’s outfits, with topic models [17], non-negative matrix factorization [1], or clustering [34]. We also leverage unlabeled images to discover “what people wear”; however, our goal is to infer visual compatibility for unseen garments, rather than trend analysis [1, 34] or image retrieval [17] on a fixed corpus.

**Compatibility and recommendation** Substantial prior work explores ways to link images containing the *same* or very similar garment [11, 32, 42, 21, 30]. In contrast, compatibility requires judging how well-coordinated or *complementary* a given set of garments is. Compatibility can be posed as a metric learning problem [41, 35, 16], addressable with Siamese embeddings [41] or link prediction [35]. Text data can aid compatibility [29, 39, 14]. As an alternative to metric learning, a recurrent neural network models outfit composition as a sequential process that adds one garment at a time, implicitly learning compatibility via the transition function [14]. Compatibility has applications in recommendation [31, 18], but prior work recommends a garment at a time, as opposed to constructing a wardrobe.

To our knowledge, all prior work requires labeled data to learn compatibility, whether from human annotators curating matches [14, 18], co-purchase data [41, 35, 16], or implicit crowd labels [29]. In contrast, we propose an unsupervised approach, which has the advantages of scalability, privacy, and continually refreshable models as fashion evolves, and also avoids awkwardly generating “negative” training pairs (see Sec. 3). Most importantly, our work is the first to develop an algorithm for generating capsule wardrobes. Capsules require going beyond pairwise compatibility to represent  $k$ -way interactions and versatility, and they present a challenging combinatorial problem.

**Subset selection** We pose capsule wardrobe generation as a subset selection problem. Probabilistic determinantal point processes (DPP) can identify the subset of items that maximize individual item “quality” while also maximizing total “diversity” of the set [25], and have been applied for document and video summarization [25, 12]. Alternatively, submodular function maximization exploits “diminishing returns” to select an optimal subset subject to a budget [36]. For submodular objectives, an efficient greedy selection criterion is near optimal [36], e.g., as exploited for sensor placement [13] and outbreak detection [28]. We show how to adapt such solutions to permit accurate and efficient selection for capsule wardrobes; furthermore, we develop an iterative EM-like algorithm to enable non-submodular objectives for mix-and-match outfits.

### 3. Approach

We first formally define the capsule wardrobe problem and introduce our approach (Sec. 3.1). Then in Sec. 3.2 we present our unsupervised approach to learn *compatibility* and *personalized styles*, two key ingredients in capsule wardrobes. Finally, in Sec. 3.3, we overview our training procedure for cross-domain attribute recognition.

#### 3.1. Subset selection for capsule wardrobes

A capsule wardrobe is a minimal set of garments that combine in versatile ways to create many compatible outfits (see Fig. 1). We cast capsule creation as the problem of selecting a subset from a large set of candidates that maximizes quality (compatibility) and diversity (versatility).

##### 3.1.1 Problem formulation and objective

We formulate the subset selection problem as follows. Let  $i = 0, \dots, (m - 1)$  index the  $m$  layers of clothing (e.g., outerwear, upper body, lower body, hosiery). Let  $A_i = \{s_i^0, s_i^1, \dots, s_i^{N_i-1}\}$  denote the set of candidate garments/pieces in layer  $i$ , where  $s_i^j, j = 0, \dots, (N_i - 1)$  is the  $j$ -th piece in layer  $i$ , and  $N_i$  is the number of candidate pieces for that layer. For example, the candidates could be the inventory of a given catalog. If an outfit is composed of *one and only one* piece from each layer, the candidate pieces in total could generate a set  $\mathcal{Y}$  of  $\prod_i N_i$  possible outfits.

**Objective** To form a capsule wardrobe, we must select only  $T$  pieces,  $A_{iT} = \{s_i^{j_1}, \dots, s_i^{j_T}\} \subseteq A_i$  from each layer  $i$ . The set of outfits  $\mathbf{y}$  generated by these pieces consists of  $A_{0T} \times A_{1T} \times \dots \times A_{(m-1)T}$ . Our goal is to select the pieces  $A_{iT}^*$ ,  $\forall i$  such that their composed set of outfits  $\mathbf{y}^*$  is maximally *compatible* and *versatile*. Fig. 2 visualizes this problem.

To this end, we define our objective as:

$$\begin{aligned} \mathbf{y}^* &= \operatorname{argmax}_{\mathbf{y} \subseteq \mathcal{Y}} C(\mathbf{y}) + V(\mathbf{y}), \\ \text{s.t. } \mathbf{y} &= A_{0T} \times A_{1T} \times \dots \times A_{(m-1)T} \end{aligned} \quad (1)$$

where  $C(\mathbf{y})$  and  $V(\mathbf{y})$  denote the compatibility and versatility scores, respectively.

A naïve approach to find the optimal solution  $\mathbf{y}^*$  requires computation on  $T^m$  outfits in a subset, multiplying by  $\binom{N}{T}^m$  to search through all possible subsets. Since our candidate pool may consist of all merchandise in a shopping site,  $N$  may be on the order of hundreds or thousands, so optimal solutions become intractable. Fortunately, our key insight is that as wardrobes expand, subsequent outfits add diminishing amounts of new styles/looks. This permits a *submodular* objective that allows us to obtain a near-optimal solution efficiently. In particular, greedily growing a set for subset selection is near-optimal if the objective function is submodular; the greedy algorithm is guaranteed to reach a

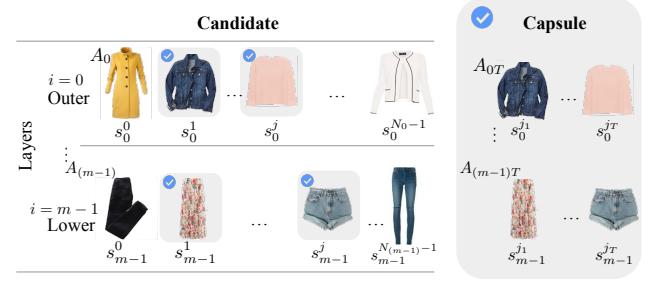


Figure 2: Selecting a subset of pieces from the candidates to form a capsule wardrobe. Left shows candidates from all layers. The selected pieces are checked and compose the subset on the right.

solution achieving at least a constant fraction  $1 - \frac{1}{e}$ , or about 63%, of the optimal score [36].

**Definition 3.1** *Submodularity.* A set function  $F$  is submodular if,  $\forall D \subseteq B \subseteq V, \forall s \in V \setminus B, F(D \cup \{s\}) - F(D) \geq F(B \cup \{s\}) - F(B)$ .

Submodularity satisfies *diminishing returns*. Since it is closed under nonnegative linear combinations, if we design  $C(\mathbf{y})$  and  $V(\mathbf{y})$  to be submodular, our final objective will be submodular as well, as we show next.

We stress that *items* in a subset  $\mathbf{y}$  are *outfits*—not garments/pieces. The capsule is the Cartesian product of all garments selected per layer. Therefore, when greedily growing a set at time step  $t$ , an incremental addition of one garment  $s_i^{j_t}$  from layer  $i$  entails adding  $s_i^{j_t} \times \prod_{i' \neq i} A_{i'(t-1)}$  new outfits. While ultimately the algorithm must add garments to the capsule, for the sake of optimization, it needs to reason about the set *in terms of those garments' combinatorial outfits*. We address this challenge below.

**Compatibility** Suppose we have an algorithm that returns the compatibility estimate  $c(o_j)$  of outfit  $o_j$  (to be defined in Sec. 3.2). We define a set's compatibility score as:

$$C(\mathbf{y}) := \sum_{o_j \in \mathbf{y}} c(o_j), \quad (2)$$

the sum of compatibility scores for all its outfits.  $C(\mathbf{y})$  is modular, a special case of submodularity, since an additional outfit  $o_j$  to any set  $\mathbf{y}$  will increase  $C(\mathbf{y})$  by the same amount  $c(o_j)$ .

**Versatility** A good capsule wardrobe should offer a variety of looks, or styles, for different uses and occasions. We formalize versatility as a *coverage* function over all styles:

$$V(\mathbf{y}) := \sum_{i=1}^K v_{\mathbf{y}}(z_i) \quad (3)$$

where  $v_{\mathbf{y}}(z_i)$  measures the degree to which outfits in  $\mathbf{y}$  cover the  $i$ -th desired style  $z_i$ , and  $K$  is the total number of distinct styles. Sec. 3.2 will define our model for styles  $z_i$ . To satisfy the diminishing returns property, we define  $v_{\mathbf{y}}(i)$  probabilistically:

$$v_{\mathbf{y}}(z_i) := 1 - \prod_{o_j \in \mathbf{y}} (1 - P(z_i | o_j)), \quad (4)$$

where  $P(z_i|o_j)$  denotes the probability of a style  $z_i$  given an outfit  $o_j$ . We define a generative model for  $P(z_i|o_j)$  below. The idea is that each outfit “tries” to cover a style with probability  $P(z_i|o_j)$ , and the style is covered by a capsule if at least one of the outfits in it successfully covers that style. Thus, as  $\mathbf{y}$  expands, subsequent outfits add diminishing amounts of style coverage. A probabilistic expression for coverage is also used in [10] for blog posts.

We have thus far defined versatility in terms of uniform coverage over *all*  $K$  styles. However, each user has his/her own preferences, and a universally versatile capsule may contain pieces that do not meet one’s taste. Thus, as a personalized variant of our approach, we adjust each style’s proportion in the coverage function by a user’s style preference. This extends our capsules with *personalized versatility*:

$$V'(\mathbf{y}) := \sum_{i=1}^K w_i v_{\mathbf{y}}(z_i), \quad (5)$$

where  $w_i$  denotes a personalized preference for each style  $i$ . Sec. 3.2 explains how the personalization weights are discovered from user data.

### 3.1.2 Optimization

A key challenge of subset selection for capsule wardrobes is that our subsets are on *outfits*, but we must form the subset by selecting *garments*. With each garment addition, the subset of outfits  $\mathbf{y}$  grows superlinearly, since every new garment can combine with all previous garments to form new outfits. Submodularity requires each addition to diminish a set function’s gain, but adding more garments yields *more* outfits, so the gain actually increases. Thus, while our objective is submodular for adding outfits, it is not submodular for adding individual garments. However, we can make the following claim:

**Claim 3.2** *When fixing all other layers (i.e., upper, lower, outer) and selecting a subset of pieces one layer at a time, the probabilistic versatility coverage function in Eqn (3) is submodular, and the compatibility function in Eqn (2) is modular. See Supplementary File for proof.*

Thus, given a single layer, our objective function is submodular for garments. By fixing all selected pieces in other layers, any additional garment will be combined with the same set of garments and form the same amount of new outfits. Thus subsets in a given layer no longer grow superlinearly. So the guarantee of a greedy solution on that layer being near-optimal [36] still holds.

To exploit this, we develop an EM-like iterative approach to approximate a greedy solution over all layers: we iteratively fix the subsets selected in other layers, and focus the current selection in a single layer. After sufficient iterations, our subsets converge to a fixed set. Algorithm 2 gives the complete steps.

Our algorithm is quite efficient. Whereas a naïve search would take more than 1B hours for our data with  $N = 150$ ,

---

**Algorithm 1** Proposed iterative greedy algorithm for submodular maximization, where  $\text{obj}(\mathbf{y}) := C(\mathbf{y}) + V(\mathbf{y})$ .

---

```

1:  $A_{iT} := \emptyset, \forall i$ 
2:  $\Delta_{obj} := \varepsilon + 1$  ▷  $\varepsilon$  is the tolerance degree for convergence
3:  $\text{obj}_{prev}^{m-1} := 0$ 
4: while  $\Delta_{obj}^{m-1} \geq \varepsilon$  do
5:   for each layer  $i = 0, 1, \dots, (m-1)$  do
6:      $A_{iT} = A_{i0} := \emptyset$  ▷ Reset selected pieces in layer  $i$ 
7:      $\text{obj}_{cur}^i := 0$ 
8:     for each time step  $t = 1, 2, \dots, T$  do
9:        $\mathbf{y}_{t-1} = A_{i(t-1)} \times \prod_{i' \neq i} A_{i'T}$ 
10:       $s_i^{j_t} := \text{argmax}_{s \in A_i \setminus A_{i(t-1)}} \delta_s$  ▷ Max increment
11:      where  $\delta_s = \text{obj}(\mathbf{y}_{t-1} \uplus s) - \text{obj}(\mathbf{y}_{t-1})$ 
12:       $A_{it} := s_i^{j_t} \cup A_{i(t-1)}$  ▷ Update layer  $i$ 
13:       $\text{obj}_{cur}^i := \text{obj}_{cur}^i + \delta_{s_i^{j_t}}$ 
14:    end for
15:   end for
16:    $\Delta_{obj}^{m-1} := \text{obj}_{cur}^{m-1} - \text{obj}_{prev}^{m-1}$ 
17:    $\text{obj}_{prev}^{m-1} := \text{obj}_{cur}^{m-1}$ 
18: end while
19: procedure INCREMENTAL ADDITION ( $\mathbf{y}_t := \mathbf{y}_{t-1} \uplus s$ )
20:    $\mathbf{y}_t^+ := s, s \in A_i \setminus A_{i(t-1)}$ 
21:   for  $j \in \{1, \dots, m\}, j \neq i$  do
22:     if  $A_{jT} \neq \emptyset$  then
23:        $\mathbf{y}_t^+ := \mathbf{y}_t^+ \times A_{jT}$ 
24:     end if
25:   end for
26:    $\mathbf{y}_t := \mathbf{y}_{t-1} \cup \mathbf{y}_t^+$ 
27: end procedure

```

---

our algorithm returns an approximate capsule in only 200 seconds. Most computation is devoted to computing the objective function, which requires topic model inference (see below). A naïve greedy approach on garments would require  $O(NT^4)$  time for  $m = 4$  layers, while our iterative approach requires  $O(NT^3)$  time per iteration (details in Supp.) For our datasets, it requires just 5 iterations.

### 3.2 Style topic models for compatibility

Having defined the capsule selection objective and optimization, we now present our approach to model versatility (via  $P(z_i|o_j)$ ) and compatibility  $c(o_j)$  simultaneously.

Prior work on compatibility takes a supervised approach. Given ground truth compatible items (either by manual labels like curated sets of product images on Polyvore [29, 39, 14] or by using Amazon co-purchase data as a proxy [35, 41, 16]), a (usually discriminative) compatibility metric is trained. See Fig. 3. However, the supervised strategy has weaknesses. First, items purchased at the same time can be a weak proxy for visual compatibility. Second, user-created sets often focus on the visualization of the collage, usually contain fewer than two main (non-accessory) pieces, and lack layers like hosiery. Third, sources like Amazon and Polyvore are limited to brands selected by vendors, a fraction of the wide variety of clothing people wear in real life. Fourth, obtaining the *negative* non-compatible examples re-



Figure 3: L to R: Amazon co-purchase example; Polyvore user-curated set; Chictopia full-body outfit (our training source).

quired by supervised discriminative methods is problematic. Previous work [35, 41, 16, 29, 39] generates negative examples by randomly swapping items in positive pairs, but there is no guarantee that the random combinations are true negatives. Not observing a pair of items together does not necessarily mean they do not go well together.

To address these issues, we propose a generative compatibility model that is learned from unlabeled images of people wearing outfits “in the wild” (Fig. 3, right). We explore a topic model—namely Correlated Topic Models (CTM) [26]—from text analysis. CTM is a Bayesian multinomial mixture model that supposes a small number of  $K$  latent *topics* account for the distribution of observed words in any given document. It uses the following generative process for a corpus  $\mathcal{D}$  consisting of  $M$  documents each of length  $L_i$ :

1. Choose  $\eta_i \sim \mathcal{N}(\mu, \Sigma)$ , where  $i \in \{1, \dots, M\}$  and  $\mu, \Sigma$  a  $K$ -dimensional mean and covariance matrix.  
 $\theta_{ik} = \frac{e^{\eta_{ik}}}{\sum_{k=1}^K e^{\eta_{ik}}}$  maps  $\eta_i$  to a simplex.
2. Choose  $\varphi_k \sim \text{Dir}(\beta)$ , where  $k \in \{1, \dots, K\}$  and  $\text{Dir}(\beta)$  is the Dirichlet distribution with parameter  $\beta$ .
3. For each word indexed by  $(i, j)$ , where  $j \in \{1, \dots, L_i\}$ , and  $i \in \{1, \dots, M\}$ .
  - (a) Choose a topic  $z_{i,j} \sim \text{Multinomial}(\theta_i)$ .
  - (b) Choose a word  $x_{i,j} \sim \text{Multinomial}(\varphi_{z_{i,j}})$ .

Only the word occurrences are observed. Following [17], we map textual topic models to visual ones: a “document” is an outfit, a “word” is an inferred visual attribute (e.g., *floral*, *chiffon*), and a “topic” is a style. The model discovers the compositions of visual cues (i.e., attributes) that characterize styles. A topic might capture plaid blue blouses, or tight leather skirts. Prior work [17] models styles with a Dirichlet prior, which treats topics within an image as independent. For compatibility, we find CTM’s logistic normal prior above [26] beneficial to account for style correlations (e.g., a formal blazer is more likely to be combined with a skirt than sporty leggings).

CTM estimates the latent variables by maximizing the posterior distribution given a corpus  $\mathcal{D}$ :

$$p(\theta, z | \mathcal{D}, \mu, \Sigma, \beta) = \prod_{i=1}^M p(\theta_i | \mu, \Sigma) \prod_{j=1}^{L_i} p(z_{ij} | \theta_i) p(x_{ij} | z_{ij}, \beta). \quad (6)$$

First we find the latent variables that fit assembled outfits on full-body images. Next, given an arbitrary combination

of catalog pieces, we predict their attributes, and take the union of attributes on all pieces to form an outfit  $o_j$ . Finally we infer its compatibility by the likelihood:

$$c(o_j) := p(o_j | \mu, \Sigma, \beta). \quad (7)$$

Combinations similar to previously assembled outfits  $\mathcal{D}$  will have higher probability. For this generative model, no negative examples need to be contrived. The training pool should be outfits like those we want the model to emulate; we use full-body photos posted to a fashion website.

Given a database of unlabeled outfit images, we predict their attributes. Then we apply CTM to obtain a set of styles, where each style  $k$  is an attribute distribution  $\varphi_k$ . Given an outfit  $o_j$ , CTM infers its style composition  $\theta_{o_j} = [\theta_{o_j 1}, \dots, \theta_{o_j K}]$ . Then we have:

$$P(z_i | o_j) := P(z_i | \theta_{o_j}) = \theta_{o_j i}, \quad (8)$$

which is used to compute our versatility coverage in Eq (4).

We personalize the styles emphasized for a user in Eq (5) as follows. Given a collection of outfits  $\{o_{j_1}, \dots, o_{j_U}\}$  owned by a user  $p$ , e.g., as shown in that user’s purchase history catalog photos or his/her album on a fashion website, we learn his/her style preference  $\theta_p^{(user)}$  by aggregating all outfits  $\theta_p^{(user)} = \frac{1}{U} \sum_j \theta_{o_j}$ . Hence, a user’s personalized weight  $w_i$  for each style  $i$  is  $\theta_{pi}^{(user)}$ .

Unlike previous work that uses supervision from human created matches or co-purchase information, our model is fully unsupervised. While we train attribute models on a disjoint pool of attribute labeled images, our topic model runs on “inferred” attributes, and annotators do not touch the images from which we learn compatibility.

### 3.3. Cross-domain attribute recognition

Finally, we describe our approach to infer cross-domain attributes on both catalog and outfit images.

**Vocabulary and data collection** Since fine-grained attributes are valuable for style, we build our vocabulary on the 195 attributes enumerated in [17]. Their dataset has attributes labeled only on *outfit* images, so we collect *catalog* images labeled with the same attribute vocabulary. To gather images, we use keyword search with the attribute name on Google, then manually prune those where the attribute is absent. This yields 100 to 300 positive training images per attribute, in total 12K images. (Details in Supp.)

**Curriculum learning from shop to street** Catalog images usually have a clear background, and are free of tricky lighting conditions and deformations from body pose. As a result, attributes are more readily recognizable in catalog images than full-body outfit images. We propose a two stage curriculum learning approach for cross domain attribute recognition. In the first stage, we finetune



Figure 4: Qualitative examples of most (left) and least (right) compatible outfits as scored by our model. Here we show both outfits with 2 pieces (top) and 3 pieces (bottom).

a deep neural network, ResNet-50 [15] pretrained on ImageNet [7], for *catalog attribute* recognition. In the second stage, we first detect and crop images into upper and lower body instances, and then we finetune the network from the first stage for *outfit attribute* recognition. Evaluating on the validation split from the 19K image dataset [17], we see a significant 15% mAP improvement, especially on challenging attributes such as *material* and *neckline*, which are subtle or occupy a small region on outfit images.

## 4. Experiments

We first evaluate our compatibility estimation in isolation (Sec. 4.1). Then, we evaluate our algorithm’s capsule wardrobes, for both quality and efficiency (Sec. 4.2).

### 4.1. Compatibility

**Dataset** Previous work [29, 39, 14] studying compatibility each collects a dataset from [polyvore.com](#). Polyvore is a platform where fashion-conscious users create sets of clothing pieces that go well with each other. While these compatible sets are valuable supervision, as discussed above, collecting “incompatible” outfits is problematic.

While our method uses no “incompatible” examples for training, to facilitate evaluation, we devise a more reliable mechanism to avoid false negatives. We collect 3,759 Polyvore outfits, composed of 7,478 pieces, each with meta-labels such as *season* (*winter*, *spring*, *summer*, *fall*), *occasion* (*work*, *vacation*), and *function* (*date*, *hike*). Tab. 1 summarizes the dataset breakdown. We exploit the meta-labels to generate incompatible outfits. For each compatible outfit, we generate an incompatible one by randomly swapping one piece to another piece in the same layer from an exclusive meta-label. For example, each *winter* (*work*) outfit will swap a piece with a *summer* (*vacation*) outfit. We use outfits that have at least 2 pieces from different layers as positives, and for each positive outfit, we generate 5 negatives. In total, our test set has 2,574 positives and 12,870 negatives. In short, by swapping with the guidance of meta-label, the negatives are more likely to be true negatives. See Supp. for examples.

	fall	winter	spring	summer	vacation	work	date	hike	<b>total</b>
total sets	307	302	307	308	731	505	791	508	3759
> 1 itm.	242	275	227	206	421	454	514	413	2752
> 2 itm.	101	130	71	60	66	177	96	146	847

Table 1: Breakdown of our Polyvore dataset.

**Baselines** We compare with two recent methods: i) MONOMER [16], an embedding trained using Amazon product co-purchase info as a proxy label for compatibility. Since Monomer predicts only pairwise scores between pieces, we average all pairwise scores to get an outfit’s total compatibility, following [14]. ii) BiLSTM [14], a sequential model trained on user-created sets from Polyvore to predict the held-out piece of a layer given pieces from other layers. The probability of the whole sequence is its compatibility metric. For both baselines, we use the authors’ provided code and their same training data sources.

**Implementation** We collect 3,957 “in the wild” outfit images from [chictopia.com](#) to learn compatibility. We apply the cross domain curriculum learning (Sec. 3.3) to predict their attributes, then fit the topic model [5] (Sec. 3.2). Given an outfit consisting of Polyvore garment product images, we predict attributes per piece then pool them for the whole outfit, and infer the outfit’s compatibility. Since both Monomer [16] and BiLSTM [14] are designed to run on *per-garment catalog images*, training them on our full-body outfit dataset is not possible.

**Results** Fig. 5 (left) compares our compatibility to alternative topic models. As one baseline, we extend the polylingual LDA (POLYLDA) style discovery [17] to compute compatibility with likelihood, analogous to our CTM-based approach. Among the three topic model variants, LDA performs worst. PolyLDA [17] learns styles across all body parts, and thus an outfit with incompatible pieces will be modeled by multiple topics, lowering its likelihood. However, PolyLDA is still ignorant of correlation between styles, and our model fares better.

Fig. 5 (right) compares our compatibility to Monomer [16] and BiLSTM [14]. Our model outperforms both existing techniques. Monomer assumes pairwise relations, but many outfits consist of more than two pieces; aggregating pairwise relations fails to accurately capture an

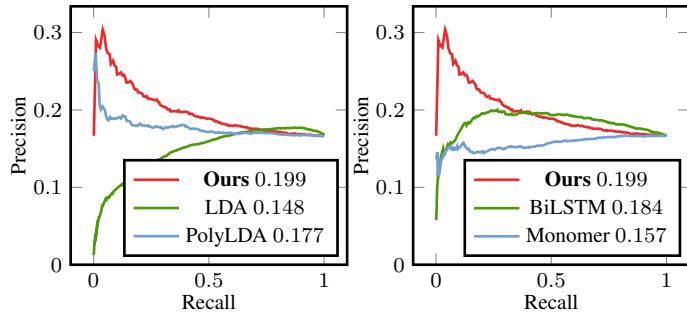


Figure 5: Compatibility accuracy comparisons. Legend shows AP. Left: topic model variants, including PolyLDA styles [17]. Right: state-of-the-art compatibility models [16, 14].

outfit’s compatibility as a whole. Like us, BiLSTM learns compatibility from only positives, yet we perform better. This supports our idea to learn compatibility from “in the wild” full body images.

Fig. 4 shows qualitative results of most and least compatible outfits scored by our model. Its most compatible outfits tend to include basic pieces—so-called *staples*—with neutral colors and low-key patterns or materials. Basic pieces go well with almost anything, making them strong compatibles. Our model picks up on the fact that people’s daily outfits usually consist of at least one such basic piece. Outfits we infer to be incompatible tend to have incongruous color palettes or mismatched fit/cut.

## 4.2. Capsule wardrobe creation

Having validated our compatibility module, we next evaluate our selection algorithm for capsule wardrobes. We consider two scenarios for likely use cases: i) *Adding*—given a seed outfit, optimize the pieces to add, augmenting the starter wardrobe; and ii) *Personalizing*—given a set of outfits the user likes/has worn/owns, optimize a new capsule from scratch to meet his/her taste.

**Dataset** To construct the pool of candidate pieces, we select  $N = 150$  pieces each for the *outer*, *upper*, and *lower* layers and  $N = 50$  for the *one piece* layer from the 7,478 pieces in the Polyvore data. The pool size represents the scale of a typical online clothing vendor. As seed outfits, we use those that have pieces in all *outer*, *upper*, *lower* layers, resulting in 759 seed outfits. We report results averaged over all 759 seed initializations. We consider capsules with  $T = 4$  pieces in each of the  $m = 3$  layers. This gives 12 pieces and 64 outfits per capsule.

**Baselines** The baselines try to find staples or prototypical pieces, while at the same time avoiding near duplicates. Specifically: i) MMR [3]: widely used function in information retrieval that strikes a balance between “relevance” and “diversity”, scoring items by  $\lambda Rel + (1 - \lambda) Div$ . We use our model’s  $p(s|\mu, \Sigma, \beta)$  of a single piece  $s$  to measure MMR relevance, and the visual dissimilarity between selected pieces as diversity. ii) CLUSTER CENTERS: clus-

	Compatibility ( $\downarrow$ )	Versatility ( $\uparrow$ )
Cluster Center	1.16	0.55
MMR- $\lambda=0.3$	3.05	<b>3.09</b>
MMR- $\lambda=0.5$	2.95	2.85
MMR- $\lambda=0.7$	2.12	2.08
naïve greedy	0.88	0.84
Iterative	<b>0.83</b>	0.78

Table 2: Capsules scored by human-created gold standard.

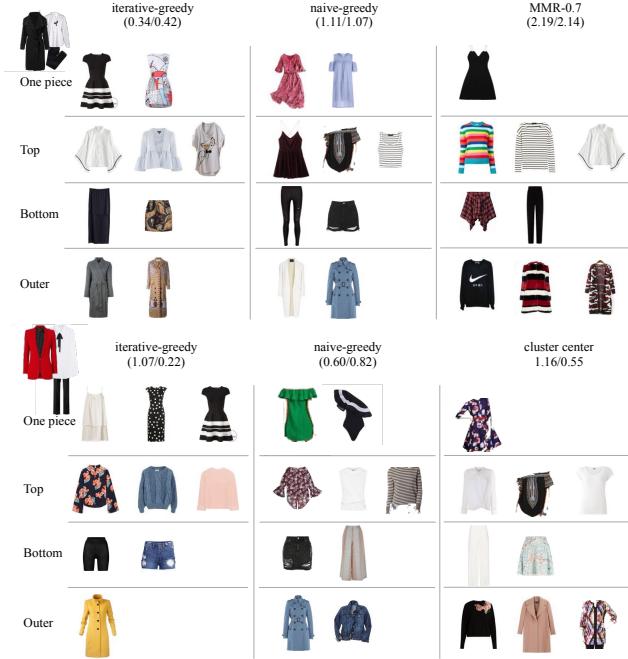


Figure 6: Two example capsules created by different methods for the same seed outfit (shown offset to left). Titles show method and compatibility/versatility scores.

ters the pieces in each layer to  $T$  clusters and then selects a representative piece from each cluster and layer. We cluster with  $k$ -medoids in the 2048-D feature space from the last layer of our catalog attribute CNN.

**Capsule creation quality** First, we compare all methods by measuring how much their capsules differ from human-curated outfits. The gold standard outfits are the Polyvore sets. We measure the visual distance of each outfit to its nearest neighbor in the gold standard; the smaller the distance, the more compatible. A capsule’s compatibility is the summed distances of its outfits. We score capsule versatility piece-wise: we compute the piece-wise visual distance per layer, and sum all distances. All distances are computed on the 2048-D CNN features and normalized by the  $\sigma$  of all distances. We stress that both metrics are independent of our model’s learned compatibility and styles; success is achieved by matching the human-curated gold standard, *not* merely by optimizing our objectives  $C(y)$  and  $V(y)$ .

Tab. 2 shows the results. Our iterative-greedy is nearest to the gold standard for compatibility, and MMR fares worst. Tuning  $\lambda$  in MMR as high as 0.7 to emphasize its rel-

	Objective	Obj./Optimal Obj. (%)	Time
Optimal	40.8	100	131.1 sec
naïve greedy	30.8	76	34.3 sec
Iterative	35.5	87	57.9 sec

Table 3: Quality vs. run-time per capsule for naïve and iterative greedy maximization, compared to the true optimal solution. Here we run at toy scale ( $N = 10$ ) so that optimal is tractable. Our iterative algorithm better approximates the optimal solution, yet is much faster. Run at a realistic scale ( $N = 150$ ), optimal is intractable ( $\sim 1$ B hours), while our algorithm takes only 200 sec.

evance term helps increase its accuracy, as it includes more staples. However, an undesirable effect of MMR (hidden by these numbers) is that for a given  $\lambda$  value, the capsules are almost always the same, *independent of the initial seed outfit*. This is due to MMR’s diversity term being vulnerable to outliers. Hence, while MMR outperforms our method on versatility, it fails to create capsules coherent with the seed wardrobe (see Fig. 6, bottom right). Clustering has acceptable compatibility, but low versatility (see Fig. 6, top right).

**Personalized capsules** Next we demonstrate our approach to tailor a capsule for a user’s taste. As a proof of concept, we select two users from [chictopia.com](http://chictopia.com), and use 200 photos in their albums to learn the users’ style preference (see end of Sec. 3.2). All 7,478 pieces are treated as candidates. Fig. 10 shows the personalized capsules generated by our algorithm. User 1’s album suggests she prefers lady-like looks, and accordingly our algorithm creates a capsule with pastel colors and chiffon material. In contrast, user 2 prefers street, punk looks, and our algorithm creates a capsule with denim, leather material, and dark colors. See Supp. for a comparison to nearest neighbor image retrieval.

**Iterative submodular vs. naïve greedy algorithm** Next, we compare our iterative greedy algorithm—which properly accounts for the superlinear growth of capsules as garments are added—to a baseline greedy algorithm that naïvely employs submodular function maximization, ignoring the combinatorics of introducing each new garment. To verify that our iterative approach better approximates the optimal solution in practice, we create a toy experiment with  $N = 10$  candidates and  $T = 3$  selections per layer. We stress the scale of this experiment is limited only by the need to compute the true optimal solution. All algorithms create capsules from scratch. We run all methods on the same single Intel Xeon 2.66Ghz machine.

Tab. 3 shows the results. Our iterative algorithm achieves 87% of the optimal objective function value, a clear margin better than naïve at 76%. On the toy dataset, solving capsule wardrobes by brute force takes  $\sim 2 \times$  our run-time. Run at the realistic scale of our experiments above ( $N = 150$ ) the brute force solution is intractable,  $\sim 1$ B hours per capsule, but our solution takes only 200 sec.

**Comparative human subject study** How do humans perceive the gap between iterative and naïve greedy’s re-

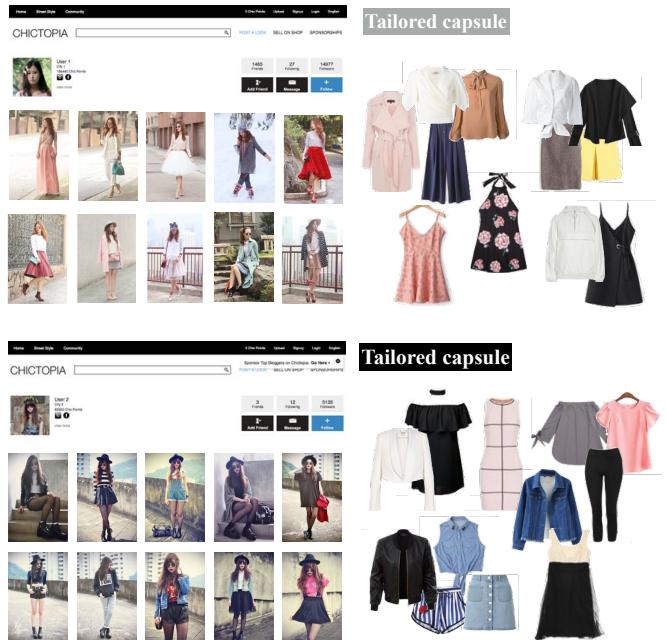


Figure 7: Personalized capsules tailored for user preference.

sults? To analyze this, we perform a human perception study with 14 subjects. Since the data contains women’s clothing, all subjects are female, and they range in age from 20’s to 60’s. Using a Web form, we present 50 randomly sampled pairs of capsules (iterative vs. naïve greedy), displayed in random order per question. Then for each pair, the subjects must select which is better. See Supp. for interface. We use majority vote and weight by their confidence scores. Despite the fact that naïve greedy leverages the same compatibility and versatility functions, 59% of the time the subjects prefer our iterative algorithm’s capsules. This shows the impact of the proposed optimization.

## 5. Conclusion

Computer vision can play an increasing role in fashion applications, which are, after all, inherently visual. Our work explores capsule wardrobe generation. The proposed approach offers new insights in terms of both efficient optimization for combinatorial mix-and-match outfit selection, as well as generative learning of visual compatibility. Furthermore, we demonstrate that compatibility learned in the wild can successfully translate to clean product images via attributes and a simple curriculum learning method. Future work will explore ways to optimize attribute vocabularies for capsules.

**Acknowledgements:** We thank Yu-Chuan Su and Chao-Yuan Wu for helpful discussions. We also thank our human subjects: Karen, Carol, Julie, Cindy, Thaoni, Maggie, Ara, Jennifer, Ann, Yen, Chelsea, Mongchi, Sara, Tiffany. This research is supported in part by an Amazon Research Award and NSF IIS-1514118. We thank Texas Advanced Computing Center for their generous support.

## Appendix Overview

This document consists of:

- Proof of Claim 3.2 in Section 3.1.2 of the main paper
- Implementation details for iterative-greedy algorithm.
- Computation complexity of naive and iterative greedy algorithm in Section 3.1.2 of the main paper
- Vocabulary for predicted attributes in Section 3.3 of the main paper
- Examples of false negative generated by randomly swapping pieces in Section 4.1 of the main paper
- Qualitative example images for most and least compatible outfits scored by baseline methods in Section 4.2 of the main paper
- Qualitative example images for personalized capsules obtained by nearest-neighbor baseline.
- Interface for human subject study in Section 4.2 of the main paper

## 6. Submodularity for objective function

*Claim 3.2 When fixing all other layers (i.e., upper, lower, outer) and selecting a subset of pieces one layer at a time, the probabilistic versatility coverage function in Eqn (3) is submodular, and the compatibility function in Eqn (2) is modular.*

*Proof.* Let  $A_j$  be candidate pieces from each layer  $j$ , where  $j = \{0, 1, \dots, m-1\}$ , and  $D, B$  be any set such that  $D \subseteq B, D = \prod_{j=0}^{m-1} A_j^D, B = \prod_{j=0}^{m-1} A_j^B$ , where  $A_j^D \subseteq A_j, A_j^B \subseteq A_j, \forall j$ . Since  $D \subseteq B, A_j^D \subseteq A_j^B, \forall j$ .

Given a layer  $i$ , let  $s_i \in A_i \setminus A_i^B$  be the piece additionally included. Outfits introduced by including  $s_i$  to  $B$  and  $D$  will be  $O = \{s_i \times \prod_{j \neq i} A_j^B\}$  and  $K = \{s_i \times \prod_{j \neq i} A_j^D\}$ , respectively. Since  $A_j^D \subseteq A_j^B, K \subseteq O$ .

- Given a layer  $i$ , and fixing all other layers, versatility is submodular.

$$\begin{aligned} v_{B \cup O}(z_i) - v_B(z_i) &= 1 - \prod_{o_j \in B \cup O} (1 - P(z_i|o_j)) \\ &\quad - \left(1 - \prod_{o_j \in B} (1 - P(z_i|o_j))\right) \\ &= \prod_{o_j \in B} (1 - P(z_i|o_j)) - \prod_{o_j \in B \cup O} (1 - P(z_i|o_j)) \\ &= \prod_{o_j \in B} (1 - P(z_i|o_j)) \left(1 - \prod_{o_j \in O} (1 - P(z_i|o_j))\right) \end{aligned}$$

Because  $P(z_i|o_j)$  is defined as a probability, it is in the range  $[0, 1]$ , and therefore  $(1 - P(z_i|o_j)) \in [0, 1], \forall j$ . Since  $D \subseteq B$ , we have that  $\prod_{o_j \in B} (1 - P(z_i|o_j)) \leq \prod_{o_j \in D} (1 - P(z_i|o_j))$ . Thus,

$$\begin{aligned} &\prod_{o_j \in B} (1 - P(z_i|o_j)) \left(1 - \prod_{o_j \in O} (1 - P(z_i|o_j))\right) \\ &\leq \prod_{o_j \in D} (1 - P(z_i|o_j)) \left(1 - \prod_{o_j \in O} (1 - P(z_i|o_j))\right) \\ &= \prod_{o_j \in D} (1 - P(z_i|o_j)) \left(1 - \prod_{o_j \in K} (1 - P(z_i|o_j)) \prod_{o_j \in O \setminus K} (1 - P(z_i|o_j))\right) \end{aligned}$$

When  $O \setminus K = \emptyset$ ,

$$\begin{aligned} &v_{B \cup O}(z_i) - v_B(z_i) \\ &\leq \prod_{o_j \in D} (1 - P(z_i|o_j)) \left(1 - \prod_{o_j \in K} (1 - P(z_i|o_j)) \prod_{o_j \in O \setminus K} (1 - P(z_i|o_j))\right) \\ &= \prod_{o_j \in D} (1 - P(z_i|o_j)) \left(1 - \prod_{o_j \in K} (1 - P(z_i|o_j))\right) \\ &= \prod_{o_j \in D} (1 - P(z_i|o_j)) - \prod_{o_j \in D \cup K} (1 - P(z_i|o_j)) \\ &= v_{D \cup K}(z_i) - v_D(z_i) \end{aligned}$$

Since  $K \subseteq O$ , when  $O \setminus K = \emptyset, O = K$ , i.e.  $\{s_i \times \prod_{j \neq i} A_j^B\} = \{s_i \times \prod_{j \neq i} A_j^D\}$ , and thus  $A_j^B = A_j^D, \forall j \neq i$ . Submodularity is closed under nonnegative linear combination, and user's personalized preference for each style  $i$   $w_i \geq 0$ , thus  $V(\mathbf{y})$  and  $V'(\mathbf{y})$  are both submodular when given a layer  $i$  and fixing all other layers.

- Given a layer  $i$ , and fixing all other layers, compatibility is modular.

Since  $s_i \in A_i \setminus A_i^B, \{s_i \times \prod_{j \neq i} A_j^B\} \cap \{A_i^B \times \prod_{j \neq i} A_j^B\} = \emptyset$ , i.e.  $O \cap B = \emptyset$ , and same with  $D \cap K = \emptyset$ .

By  $O \cap B = \emptyset$ , we get

$$\begin{aligned} C(B \cup O) - C(B) &= \sum_{o_j \in B \cup O} c(o_j) - \sum_{o_j \in B} c(o_j) \\ &= \sum_{o_j \in B} c(o_j) + \sum_{o_j \in O} c(o_j) - \sum_{o_j \in B} c(o_j) \\ &= \sum_{o_j \in O} c(o_j) \end{aligned}$$

and  $C(D \cup K) - C(K) = \sum_{o_j \in K} c(o_j)$ .

By  $K \subseteq O$ , we have

$$\sum_{o_j \in O} c(o_j) = \sum_{o_j \in K} c(o_j) + \sum_{o_j \in O \setminus K} c(o_j)$$

When  $O \setminus K = \emptyset$ ,

$$\begin{aligned} C(B \cup O) - C(B) &= \sum_{o_j \in O} c(o_j) \\ &= \sum_{o_j \in K} c(o_j) + \sum_{o_j \in O \setminus K} c(o_j) \\ &= \sum_{o_j \in K} c(o_j) = C(D \cup K) - C(K) \end{aligned}$$

Thus  $C(\mathbf{y})$  is modular when given layer  $i$  and fixing all other layers.

■

## 7. Implementation details for iterative-greedy algorithm

We set our tolerance degree  $\varepsilon = 0.5$ , and find that our iterative-greedy algorithm typically converges after 5 iterations. We use Gibbs sampling [5] for compatibility inference. Due to the sampling process,  $p(\boldsymbol{\theta}, \mathbf{z} | o_j, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \beta)$  fluctuates slightly at different run times. To increase robustness, we further apply a step function on our compatibility score  $c(o_j)$ , so that  $c(o_j) \geq \varepsilon$  is mapped to 1, and otherwise to 0. We fix  $\epsilon = -4.69$ , as validated in the compatibility experiment to give the best precision-recall trade-off.

## 8. Computation complexity for naieve and greedy algorithms

Both the naive greedy and iterative greedy algorithms have a computation bottleneck when computing the objective  $\text{obj}(\mathbf{y}_t)$ . Its complexity is decided by  $N_i$  times  $|\mathbf{y}_t|$ , the size of the incrementally growing subset  $\mathbf{y}_t$  at iteration  $t$ . In the following we show the algorithms of naive and iterative greedy, and analyze their  $|\mathbf{y}_t|$  respectively. Without loss of generality, we assume our algorithms are provided with an initial piece for each layer  $i$ .

---

Naive greedy algorithm for submodular maximization, where  $\text{obj}(\mathbf{y}) := C(\mathbf{y}) + V(\mathbf{y})$ .

---

```

for each time step  $t = 1, 2, \dots, T$  do
     $\mathbf{y}_{t-1} = \prod_{i=0}^{m-1} A_{i(t-1)}$ 
    for each layer  $i = 0, 1, \dots, m-1$  do
         $s_i^t := \operatorname{argmax}_{s \in A_i \setminus A_{i(t-1)}} \delta_s$ 
        where  $\delta_s = \text{obj}(\mathbf{y}_t) - \text{obj}(\mathbf{y}_{t-1})$ 
        where  $\mathbf{y}_t = \mathbf{y}_{t-1} \cup \left\{ s \times \prod_{j \neq i} A_{j(t-1)} \right\}$ 
    end for
end for

```

---

We need to compute  $\text{obj}(\mathbf{y}_t)$  for all  $s \in A_i \setminus A_{i(t-1)}$ . At time step  $t$  and layer  $i$ ,  $|A_i \setminus A_{i(t-1)}| = N_i - (t+1)$ .

Since  $N_i \gg (t+1)$  and is around the same scale for all  $i$ , we shorthand the term  $N_i - (t+1)$  to  $N$ . Considering every candidate  $s$  and the set of outfits  $\{s \times \prod_{j \neq i} A_{j(t-1)}\}$  introduced, computation for all sets introduced by all  $s$  becomes  $N(t+1)^{(i-1)}$  times. Summing over all  $i$  and all  $t$ , the total computation is  $\sum_{t=1}^T \sum_{i=0}^{m-1} N(t+1)^{(i-1)}$  times.

$$\begin{aligned} \sum_{t=1}^T \sum_{i=0}^{m-1} N(t+1)^{(i-1)} &= N \sum_{t=1}^T \frac{1 - (t+1)^m}{1 - (t+1)} \\ &= N \sum_{t=1}^T O(t^{m-1}) \end{aligned}$$

$\sum_{t=1}^T O(t^{m-1})$  is an  $(m-1)$ -th power series for the first  $T$  natural numbers. A closed formula at  $m = 4$  equals  $[\frac{T(T+1)}{2}]^2$ , so the final complexity will be  $O(NT^4)$  for naive greedy.

---

**Algorithm 2** Proposed iterative greedy algorithm for submodular maximization, where  $\text{obj}(\mathbf{y}) := C(\mathbf{y}) + V(\mathbf{y})$ .

---

```

1:  $A_{iT} := \emptyset, \forall i$ 
2:  $\Delta_{obj} := \varepsilon + 1$  ▷  $\varepsilon$  is the tolerance degree for convergence
3:  $\text{obj}_{prev}^{m-1} := 0$ 
4: while  $\Delta_{obj}^{m-1} \geq \varepsilon$  do
5:   for each layer  $i = 0, 1, \dots, (m-1)$  do
6:      $A_{iT} = A_{i0} := \emptyset$  ▷ Reset selected pieces in layer  $i$ 
7:      $\text{obj}_{cur}^i := 0$ 
8:     for each time step  $t = 1, 2, \dots, T$  do
9:        $\mathbf{y}_{t-1} = A_{i(t-1)} \times \prod_{i' \neq i} A_{i'T}$ 
10:       $s_i^t := \operatorname{argmax}_{s \in A_i \setminus A_{i(t-1)}} \delta_s$  ▷ Max increment
11:      where  $\delta_s = \text{obj}(\mathbf{y}_{t-1} \uplus s) - \text{obj}(\mathbf{y}_{t-1})$ 
12:       $A_{it} := s_i^t \cup A_{i(t-1)}$  ▷ Update layer  $i$ 
13:       $\text{obj}_{cur}^i := \text{obj}_{cur}^i + \delta_{s_i^t}$ 
14:    end for
15:  end for
16:   $\Delta_{obj}^{m-1} := \text{obj}_{cur}^{m-1} - \text{obj}_{prev}^{m-1}$ 
17:   $\text{obj}_{prev}^{m-1} := \text{obj}_{cur}^{m-1}$ 
18: end while
19: procedure INCREMENTAL ADDITION ( $\mathbf{y}_t := \mathbf{y}_{t-1} \uplus s$ )
20:    $\mathbf{y}_t^+ := s, s \in A_i \setminus A_{i(t-1)}$ 
21:   for  $j \in \{1, \dots, m\}, j \neq i$  do
22:     if  $A_{jT} \neq \emptyset$  then
23:        $\mathbf{y}_t^+ := \mathbf{y}_t^+ \times A_{jT}$ 
24:     end if
25:   end for
26:    $\mathbf{y}_t := \mathbf{y}_{t-1} \cup \mathbf{y}_t^+$ 
27: end procedure

```

---

Let  $t_g$  denote at which iteration the while loop is. At iteration  $t_g = 0$ , for layer  $i = 0$ , each candidate piece  $s$  will introduce a set of outfits  $\{s \times \prod_{j \neq i} A_{jT}\}$ . Since each layer has an initial piece,  $|A_{jT}| = 1, \forall j$ , and computation for the objective value of all sets introduced by all  $s$  is  $N$  times. After layer  $i = 0$  selects  $T$  pieces,  $|A_{0T}| = T$ , and thus layer  $i = 1$  computes  $TN$  times objective values. After that, layer

pattern	material	shape	collar	article	color
crochet	translucent	skirt drape pleated	scoop	T-shirt	white
camouflage	leather	skirt drape prairie	vneck	blouse	black
floral	denim	skirt drape flat	square	jacket	red
geo	fur	skirt length long	off-shoulder	blazer	pink
horizontal striped	down	skirt length medium	heart	cardigan	orange
lace		skirt length short	turtle-neck	coat	yellow
leopard		skirt shape tight	shirt collar	vest	green
plaid		skirt shape loose		dress	blue
paisley		skirt shape full		skirt	purple
plain		pants loose		pants	brown
polka dot		pants flared		jeans	gray
tribal		pants peg-leg		leggings	beige
vertical striped		pants skinny		stocking	
zebra		pants short		boots	
		ruffle shirt		shoes	
		ruffle dress		sunglasses	
				hat	
				belt	
				scarf	
				bag	
				socks	
				sweater	

Table 4: Predicted attributes organized by types.

$i = 2$  computes  $T^2N$  times, and so on. So at  $g_t = 0$ , the total computation complexity is  $\sum_{i=0}^{m-1} T^i N$ . For all iterations  $t_g \geq 1$ , we reset selected pieces at each layer  $i$ , and select  $T$  pieces again, so the complexity is  $T^{(m-1)}N, \forall i$ . Summing over all layers, we get  $\sum_{i=0}^{m-1} T^{(m-1)}N = mNT^{(m-1)}$ . At  $m = 4$ , we get computation complexity per  $t_g$  iteration  $O(NT^3)$  for iterative greedy.

## 9. Vocabulary for catalog/outfit attributes

Tab. 4 lists the predicted attributes organized by types: *pattern*, *material*, *shape*, *collar*, *article*, *color*. We pair *pattern*, *material*, and *color* with body parts to get localized attributes. Since modeling correlation between attributes (e.g. *material translucent* co-occurs with *pattern lace*, *neckline scoop* co-occurs with *pattern graphics*) improves each individual attribute accuracy [43, 4], we subsample images from each type and multilabel them for catalog attribute prediction.

## 10. Examples of false negatives

In Section 4.1 we describe our procedure to generate negative (not compatible) outfits for evaluation. Here we give more intuition about why this helps generate safe negatives. In Fig. 8 we show examples of outfits with different meta labels (*season*, *occasion*, *function*), and show negatives generated by swapping pieces from exclusive meta-labels, comparing with negatives randomly generated.



Figure 8: Left: outfits from exclusive meta-labels. Middle: randomly swapping pieces will form actually compatible outfits, i.e. those swapped within the same meta-label. Right: swapping pieces across exclusive meta-labels will be closer to true negatives.

## 11. Qualitative example images for most/least compatible outfits predicted by baselines

Fig. 9 shows most and least compatible outfits predicted by baselines, Monomer [16], BiLSTM [14], along with ours (CTM). Most compatible outfits scored by us are those that consist of staples. Most compatibles scored by BiLSTM are those with a special pattern or material, which are more stylish. Most compatibles scored by Monomer contain mostly white pieces.

## 12. Qualitative example images for personalized capsules obtained by nearest neighbor baseline

We predict attributes on users' outfits and all candidate pieces, and find the visually similar pieces (measured in attribute space) to those worn on each user. The result is shown in Fig. 10, comparing with the result using our method. Forming capsule wardrobes by using nearest neighbor does not take compatibility nor diversity into consideration, thus the results are mainly pieces similar in cut, shape, material and color.

## 13. Human subject interface

Fig. 11 and Fig. 12 show the interface of our human subject study on capsule wardrobes. In the instructions, we first describe the definitions of capsule wardrobes, and show examples of good and bad capsules, following explanations of why they are good and bad. In each question, we show (a) and (b) 2 candidate capsules, and ask subjects to choose which is better, where better is defined in the instructions: better capsules are those that can produce more compatible outfits. We ask subjects to avoid choosing EQUAL. Each question is also followed by confidence rating: from 1 = subtle to 3 = very obvious.

## References

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



Figure 9: Most/least compatible outfits predicted by each method. Each row shows a method: our method tends to score outfits with staples as more compatible; BiLSTM [14] scores outfits more stylish as more compatible; Monomer [16] scores outfits with white pieces as more compatible.



Figure 10: Personalized capsules tailored for user preference.

- [2] L. Bossard, M. Dantone, C. Leistner, C. Wengert, T. Quack, and L. Van Gool. Apparel classification with style. In *ACCV*, 2012.
- [3] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *ACM SIGIR*, 1998.
- [4] H. Chen, A. Gallagher, and B. Girod. Describing clothing by semantic attributes. In *ECCV*, 2012.
- [5] J. Chen, J. Zhu, Z. Wang, X. Zheng, and B. Zhang. Scalable inference for logistic-normal topic models. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

- [6] Q. Chen, J. Huang, R. Feris, L. M. Brown, J. Dong, and S. Yan. Deep domain adaptation for describing people based on fine-grained clothing attributes. In *CVPR*, 2015.
- [7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.
- [8] W. Di, C. Wah, A. Bhardwaj, R. Piramuthu, and N. Sundaresan. Style finder: Fine-grained clothing style detection and retrieval. In *CVPR*, 2013.
- [9] Q. Dong, S. Gong, and X. Zhu. Multi-task curriculum transfer deep learning of clothing attributes. In *WACV*. IEEE, 2017.
- [10] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In *ACM SIGKDD*, 2009.
- [11] J. Fu, J. Wang, Z. Li, M. Xu, and H. Lu. Efficient clothing retrieval with semantic-preserving visual phrases. In *ACCV*, 2012.
- [12] B. Gong, W. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *NIPS*, 2014.
- [13] C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. In *JMLR*, 2005.
- [14] X. Han, Z. Wu, Y.-G. Jiang, and L. S. Davis. Learning fashion compatibility with bidirectional lstms. *ACM MM*, 2017.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [16] R. He, C. Packer, and J. McAuley. Learning compatibility across categories for heterogeneous item recommendation. In *ICDM*, 2016.
- [17] W.-L. Hsiao and K. Grauman. Learning the latent “look”: Unsupervised discovery of a style-coherent embedding from fashion images. In *ICCV*, 2017.
- [18] Y. Hu, X. Yi, and L. Davis. Collaborative fashion recommendation: A functional tensor factorization approach. In *ACM MM*, 2015.

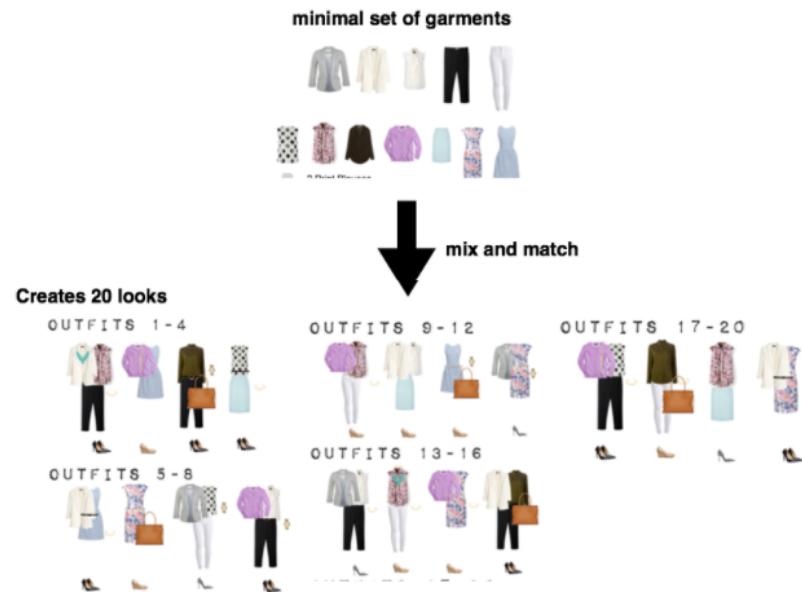
# Capsule Wardrobe comparison

A capsule wardrobe is a minimal set of garments, that mix and match well and create multiple looks. Below we show 1 example of a good capsule and 3 examples of bad capsules.

In each question, we will show you 2 sets of garments, (a) and (b). These are candidate capsules. Please select which capsule you think is better. Better means you can make the most compatible outfits out of it. If after careful examination you cannot choose either (a) or (b) as better, then mark that question as "EQUAL".

Also rate your confidence for each answer: 3 = "very obviously better", 2 = "somewhat better", 1 = "it's subtle."

## Good example



## Three bad examples



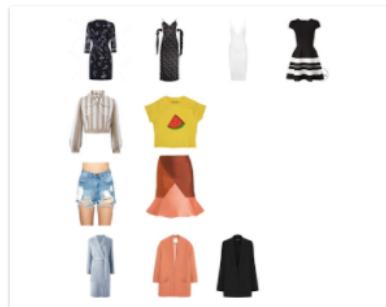
Figure 11: Instructions to guide human subjects: we show textual descriptions of capsule wardrobe definitions, and visual examples of good and bad capsules.

1) Which is better \*



a

EQUAL



b

1) Confidence \*

1

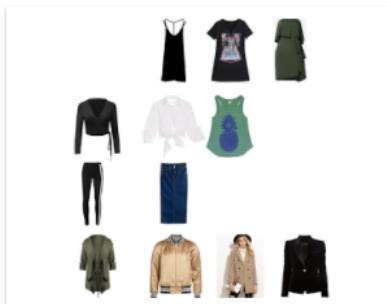
2

3

subtle

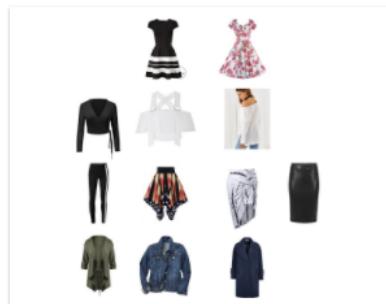
very obvious

2) Which is better \*



a

EQUAL



b

2) Confidence \*

1

2

3

subtle

very obvious

Figure 12: Questions shown to subjects: (a),(b) are sampled pairs of iterative vs. naive greedy capsules. We encourage subjects to avoid selecting EQUAL unless the difference between two capsules is too subtle to tell. Each comparison is followed by a confidence rating for provided answer. Best viewed on pdf.

[19] J. Huang, R. S. Feris, Q. Chen, and S. Yan. Cross-domain image retrieval with a dual attribute-aware ranking network.

In ICCV, 2015.

[20] T. Iwata, S. Watanabe, and H. Sawada. Fashion coordinates

- recommender system using photographs from fashion magazines. In *IJCAI*, 2011.
- [21] 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.
- [22] M. H. Kiapour, K. Yamaguchi, A. Berg, and T. Berg. Hipster wars: Discovering elements of fashion styles. In *ECCV*, 2014.
- [23] T. Kim, S. Kim, S. Na, H. Kim, M. Kim, and B. Jeon. Visual fashion-product search at sk planet. 2016.
- [24] A. Kovashka, D. Parikh, and K. Grauman. WhittleSearch: Image search with relative attribute feedback. In *CVPR*, 2012.
- [25] A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 2012.
- [26] J. D. Lafferty and D. M. Blei. Correlated topic models. In *NIPS*, 2006.
- [27] H. Lee, J. Seol, and S.-g. Lee. Style2vec: Representation learning for fashion items from style sets. *arXiv preprint arXiv:1708.04014*, 2017.
- [28] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *ACM SIGKDD*, 2007.
- [29] Y. Li, L. Cao, J. Zhu, and J. Luo. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Multimedia*, 2017.
- [30] K. Lin, H.-F. Yang, K.-H. Liu, J.-H. Hsiao, and C.-S. Chen. Rapid clothing retrieval via deep learning of binary codes and hierarchical search. In *ACM ICMR*, pages 499–502, 2015.
- [31] S. Liu, J. Feng, Z. Song, T. Zheng, H. Lu, C. Xu, and S. Yan. Hi, magic closet, tell me what to wear! In *ACM MM*, 2012.
- [32] S. Liu, Z. Song, G. Liu, C. Xu, H. Lu, and S. Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *CVPR*, 2012.
- [33] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.
- [34] K. Matzen, K. Bala, and N. Snavely. Streetstyle: Exploring world-wide clothing styles from millions of photos. *arXiv:1706.01869*, 2017.
- [35] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.
- [36] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 1978.
- [37] E. Simo-Serra, S. Fidler, F. Moreno-Noguer, and R. Urtasun. Neuroaesthetics in Fashion: Modeling the Perception of Fashionability. In *CVPR*, 2015.
- [38] E. Simo-Serra and H. Ishikawa. Fashion Style in 128 Floats: Joint Ranking and Classification using Weak Data for Feature Extraction. In *CVPR*, 2016.
- [39] X. Song, F. Feng, J. Liu, and Z. Li. Neurostylist: Neural compatibility modeling for clothing matching. *ACM MM*, 2017.
- [40] K. Vaccaro, S. Shivakumar, Z. Ding, K. Karahalios, and R. Kumar. The elements of fashion style. In *ACM UIST*, 2016.
- [41] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *ICCV*, 2015.
- [42] S. Vittayakorn, K. Yamaguchi, A. C. Berg, and T. L. Berg. Runway to realway: Visual analysis of fashion. In *WACV*, 2015.
- [43] K. Yamaguchi, T. Okatani, K. Sudo, K. Murasaki, and Y. Taniguchi. Mix and match: Joint model for clothing and attribute recognition. In *BMVC*, 2015.
- [44] B. Zhao, J. Feng, X. Wu, and S. Yan. Memory-augmented attribute manipulation networks for interaction fashion search. In *CVPR*, 2017.