Why do These Match? Explaining the Behavior of Image Similarity Models

Bryan A. Plummer∗
Boston University
bplum@bu.edu

Mariya I. Vasileva∗
University of Illinois at Urbana-Champaign
mvasile2@illinois.edu

Vitali Petsiuk
Boston University
vpetsiuk@bu.edu

Kate Saenko
Boston University
saenko@bu.edu

David Forsyth
University of Illinois at Urbana-Champaign
daf@illinois.edu

Abstract

Explaining a deep learning model can help users understand its behavior and allow researchers to discern its shortcomings. Recent work has primarily focused on explaining models for tasks like image classification or visual question answering. In this paper, we introduce an explanation approach for image similarity models, where a model’s output is a semantic feature representation rather than a classification. In this task, an explanation depends on both of the input images, so standard methods do not apply. We propose an explanation method that pairs a saliency map identifying important image regions with an attribute that best explains the match. We find that our explanations are more human-interpretable than saliency maps alone, and can also improve performance on the classic task of attribute recognition. The ability of our approach to generalize is demonstrated on two datasets from very different domains, Polyvore Outfits and Animals with Attributes 2.

1 Introduction

Many problems in artificial intelligence that reason about complex relationships can be solved by learning some feature embedding to measure similarity between images and/or other modalities such as text. Examples of these tasks include scoring fashion compatibility [8, 11, 28], image retrieval [14, 24, 32], or zero-shot recognition [3, 20, 30]. Reasoning about the behavior of similarity models can aid researchers in identifying potential improvements, or help users understand the model’s predictions which can build trust [21]. However, prior work on producing explanations for neural networks has primarily focused on explaining classification models (e.g., [6, 22, 23, 25, 26, 34]) and does not directly apply to similarity models. Given a single input image, such methods produce a saliency map which identifies pixels that played a significant role towards a particular class prediction (see Figure 1 for an example). On the other hand, a similarity model requires at least two images to produce a score. The interaction between both images defines which features are more important, so replacing just one of the images can result in identifying different salient traits.

For image pairs where similarity is determined by the presence or absence of an object, a saliency map may be sufficient to understand model behavior. However, when we consider the image pair in Figure 1, highlighting the necklace as the region that contributes most to the similarity score is reasonable, but uninformative given that there are no other objects in the image. Instead, what is important is the fact that it shares a similar color with the ring. Whether these image properties or saliency maps are a better fit as an explanation is not determined by the image domain (i.e. attributes

∗Equal Contribution

Preprint. Under review.
for e-commerce imagery vs. saliency for natural imagery), but instead by the images themselves. For example, an image can be matched as formal-wear with an explanation pointing to a shirt’s collar, while two images of animals can match because both have stripes.

Guided by this intuition, we introduce Salient Attributes for Network Explanation (SANE). Our approach generates a saliency map to explain a model’s similarity score, paired with an attribute explanation which identifies important image properties. SANE is a “black box” method, meaning it can explain any network architecture and only needs to measure changes to a similarity score when provided with different inputs. Unlike a standard classifier, which simply predicts the most likely attributes for a given image, our explanation method predicts which attributes are important for the similarity score predicted by a model. Predictions are made for each image in a pair, and allowed to be non-symmetric, e.g., the explanation for why the ring in Figure 1 matches the necklace may be that it contains “black”, even though the explanation for why the necklace matches the ring could be that it is “golden.” A different similarity model may also result in different attributes being deemed important for the same pair of images.

Our SANE model combines two major components: an attribute predictor and a saliency map generator. Given an input image, the attribute predictor outputs a confidence score for each attribute, in addition to an attribute activation map that indicates regions within the image associated with that attribute. We rank attributes as explanations for an image pair by how well their attribute activation map matches the saliency map produced by the generator. Our underlying assumption is that at least one of the attributes present in the image should be able to explain the similarity score assigned to the pair. Although we evaluate only the top-ranked attribute in our experiments, in practice more than one attribute could be used to explain a similarity score. We find that using saliency maps as supervision for the attribute activation maps during training not only improves the attribute-saliency matching, resulting in better attribute explanations, but also boosts attribute recognition performance using standard metrics like average precision.

We evaluate several candidate saliency map generation methods which are primarily adaptations of “black box” approaches that do not rely on a particular model architecture or require access to network parameters to produce a saliency map. These methods generally identify important regions by measuring a change in the output class score resulting from some perturbation of the input image. Similarity models, however, typically rely on a learned embedding space to reason about relationships between images, where proximity between points or the lack thereof indicates some degree of correspondence. An explanation system for embedding models must therefore consider how the distances between embedded points, and thus their similarity, change based on perturbing one or both of the pair of input images. We explore two strategies for adapting these approaches to our task. First, we manipulate just a single image (the one we wish to produce an explanation for) while keeping the other image fixed. Second, we manipulate both images to allow for more complex interactions between the pair. Additional discussion on the ramifications of this choice and details of the saliency methods can be found in Section 3.2.

Our paper makes the following contributions: 1) we provide the first study of explaining the behavior of image similarity models; 2) we propose a novel explanation approach that combines saliency maps and attributes, which to our knowledge is the first explanation work to use attributes; 3) we validate...
Figure 2: Attribute Model Overview. We use the saliency maps used to explain why two items are considered similar to provide supervision for the attribute heatmaps. During training, each saliency map produced by the generator is encouraged to match at least ground truth attribute’s heatmap.

our method with both automatic metrics and a user study on two diverse datasets, and find that it produces more informative explanations and also improves attribute recognition performance.

2 Related Work

Saliency-based Explanations. Saliency methods can generally be split into “white box” and “black box” approaches. “White box” methods assume access to internal components of a neural network, either in the form of gradients or activations of specific layers (e.g., \[4\] [5] [22] [26] [27] [33] [35] [36]). Most of them produce a saliency map by using some version of backpropagation from class probability to an input image. In contrast, “black box” approaches require no knowledge of the internals (e.g. weights, gradients) of the models. These methods obtain saliency maps by perturbing the input in a predefined way and measuring the effect of it on the model output, such as class score. We adapt and compare three “black box” and one “white box” methods for our saliency map generator in Figure 2. “Black box” approaches include a Sliding Window [34], which masks image regions sequentially, and Randomized Input Sampling for Explanations (RISE) [23], which masks random sets of regions. Both measure the effect removing these regions have on the class score. LIME [25] first obtains a super-pixel representation of an image. Super-pixel regions are randomly deleted, and their importance is estimated using Lasso. “White box” Mask [6] learns a saliency map directly by using different perturbation operators and propagating the error to a low resolution mask. Note that all the methods discussed above (including the four we adapt) do not operate directly on similarity models, which we will discuss further in Section 3.2.

Natural Language-based Explanations. Instead of producing saliency maps, which can sometimes be difficult to interpret, researchers have explored methods of producing text-based explanations. These include methods which justify a model’s answer in the visual question answering task [13] [19], rationalize the behavior of a self-driving vehicle [15], or describe why a category was selected in fine-grained object classification [10]. Recently, Hendricks et al. [2] leveraged attributes to correct mistakes in text-based explanations for fine-grained object classification. However, their goal is to justify a model’s decision by pointing to evidence rather than capturing a model’s behavior. Lad et al. [18] used human-generated attribute explanations describing why two images are similar or dissimilar as guidance for image clustering. Our approach could be used to automatically generate these explanations rather than relying on human feedback.

3 Salient Attributes for Network Explanations (SANE)

We are given a fixed model that predicts the similarity between two images, and must explain why a query image is similar to a reference image. While typical models for predicting similarity are learned from data, using an embedding method and a triplet loss, our approach is agnostic as to how the model being explained is built. Our method consists of two components: the attribute explanation model (Section 3.1), and the saliency map generator (Section 3.2). Although we train a CNN to produce attribute annotations, the image similarity model we wish to produce explanations for is kept fixed. At test time, one recovers a saliency map for the match from the query image in a pair, then uses the attribute explanation model to determine which attribute explains the map (Section 3.3).
3.1 Attribute Explanation Model

Suppose we have access to pairs of images \((I_r, I_q)\). Here, \(I_r\) denotes a reference image and \(I_q\) a query image. We wish to obtain an explanation for the match between \(I_r\) and \(I_q\). Associated with each pair is a saliency map \(m_q\) produced by a saliency map generator as described in Section 3.2. Note that saliency is a relation that is not symmetric, meaning that if we were to produce an analogical saliency map and saliency map, \(I_r, m_r\), will almost surely differ from \(m_q\). Finally, assume we have access to binary attribute annotations \(a_i, i = 1, \ldots, A\), and let \(a_{gt} \in \{0, 1\}^A\) be the set of ground truth attribute annotations for a given query image. If no attribute annotations are provided, an attribute discovery method could be employed (e.g., [7, 29]). We explore using an attribute discovery method in the appendix.

Our attribute explanation model produces confidence scores \(\hat{a} \in \mathbb{R}^A\) for \(I_q\). Unlike a standard attribute classifier, however, our goal is not just to predict the most likely attributes in \(I_q\), but rather to identify which attributes contribute the most to the similarity score \(s(I_r, I_q)\) produced by the similarity model we wish to obtain explanations for. To accomplish this, we associate with each attribute \(a_i\) an attribute activation map \(n_i\), representing a downsampled mask of an image that identifies prominent regions in \(I_q\) for that attribute. The attribute activation maps are learned by encouraging the saliency map \(m_q\) to match one of the attribute activation maps corresponding to the ground truth attributes \(a_{gt}\) in \(I_q\) (see Figure 2 for an overview). Our underlying assumption is that at least one of the ground truth attributes of \(I_q\) should be able to explain why \(I_q\) is similar to \(I_r\). Thus, at least one of the attribute activation maps \(n_i\) should closely resemble the saliency map for the match, \(m_q\).

Each attribute confidence score is obtained using a global average pooling layer on its attribute activation map followed by a softmax activation function. The attribute explanation network is trained using a Huber loss [12], sometimes referred to as a smooth \(\ell_1\) loss, which helps encourage sparsity in the predictions. More formally, given a set of confidence scores \(\hat{a}\) and attribute labels \(a_{gt}\), our loss is,

\[
L_{Huber}(\hat{a}, a_{gt}) = \begin{cases} 
\frac{1}{2}(\hat{a}_{gt} - \hat{a})^2 & \text{for } |\hat{a}_{gt} - \hat{a}| \leq 1 \\
\hat{a}_{gt} - \hat{a} & \text{otherwise.}
\end{cases}
\]

(1)

Note that multiple attributes can be present in the image; note also that this loss operates on attributes, not attribute activation maps. Since the confidence scores sum to one (due to the softmax function), we scale a binary label vector by the number of ground truth attributes \(A_{gt}\) (e.g., if there are four attributes for an image, its label would be 0.25 for each ground truth attribute, and zero for all others).

Leveraging saliency maps during training. Rather than simply hoping our attribute activation maps match a saliency map, we explicitly encourage attributes which are useful in explaining the predictions of an image similarity model. Some ground truth attributes may be irrelevant, however, and the rankings of likely attributes for an image may change depending on what it is compared to. We obtain a set of regions that may be important to the decisions of an image similarity model by generating a set of \(K\) saliency maps \(M_q\) to up to \(K\) reference images that are similar. For the image under consideration, we also construct a set of attribute activation maps \(N_{gt}\) corresponding to each ground truth attribute. Then, for each saliency map we find its best match in \(N_{gt}\). We match saliency maps to attributes rather than the other way around since not all annotated attributes are necessarily relevant to the explanation of \(s(I_r, I_q)\). We use an \(\ell_2\) loss between the selected attribute activation map and saliency map, i.e.,

\[
L_{hm} = \frac{1}{K} \sum_{\forall m \in M_q} \min_{\forall n \in N_{gt}} \|m - n\|_2.
\]

(2)

Combined with the attribute classification loss, our model’s complete loss function is:

\[
L_{total} = L_{Huber} + \lambda L_{hm},
\]

(3)

where \(\lambda\) is a scalar parameter. See appendix for implementation details and parameter values.

3.2 Saliency Map Generator

A straightforward approach to producing a saliency map is to manipulate the input image by removing image regions and measuring the effect this has on the similarity score. If a large drop in similarity is
measured, then the region must be of significance to the score. If almost no change was measured, then the model considers the image region irrelevant. The saliency map is generated from this approach by averaging the similarity scores for each pixel location over all instances where it was removed from the input. The challenge then is to determine how to manipulate the input image to discover these important regions. We adapt and compare four saliency methods: Sliding Window [34], RISE [23], LIME [25], and Mask [6]. We now describe how we adapt these models for our task; additional details on each method can be found in the appendix.

Computing similarity scores. Each saliency method we compare was designed to operate on a single image and to measure the effect manipulating the image has on the prediction of a specific object class. However, an image similarity model’s predictions are made for two or more images. Let us consider the case described in Section 3.1 where we are just comparing two images, a query image (i.e. the image we want to produce an explanation for), and a reference image, although our approach extends to consider multiple reference images. Even though we do not have access to a class label, we can measure the effect manipulating an image has on the similarity score between the query and reference images. Two approaches are possible: manipulate both images, or manipulate only the query image.

Manipulating both images would result in NM forward passes through the image similarity model (for N, M the number of query and reference image manipulations, respectively), which is prohibitively expensive unless M << N. But we need only an accurate saliency map for the query image, and so we set M << N in our experiments. There is another danger: for example, consider two images of clothing items that are similar if either they both contain or do not contain a special button. Masking out the button in one image and not the other would cause a drop in similarity score, but masking out the button in both images would result in high image similarity. These conflicting results could make accurately identifying the correct image regions contributing to a decision difficult.

The alternative is to manipulate the query image alone, i.e. keep a fixed reference image.

3.3 Selecting Informative Attributes

At test time, given a similarity model and a pair of inputs, SANE generates a saliency map and selects an attribute to show to the user. We suspect that not all attributes annotated for a dataset may prove to be useful in explaining the decisions of every image similarity model. We take into account how useful each attribute is at explaining predictions made by a similarity model using held out data. First we count how often an attribute was the best explanation for a pair of images in the validation set. Then, we rank potential attribute explanations using a weighted combination of the attribute confidence score ̂a, how well the attribute activation map n matches the generated saliency map mq, and the prior probability p that each attribute is the best explanation for an image pair. The explanation score is given by,

$$e(m_q, ̂a, n, p) = \phi_1 s + \phi_2 d_{\cos}(m_q, n) + \phi_3 p,$$

where \(d_{\cos}\) denotes cosine similarity, and \(\phi_{1-3}\) are scalar parameters estimated using grid search on held out data.

4 Experiments

Datasets. We evaluate our approach using two datasets from different domains to demonstrate its ability to generalize. The Polyvore Outfits dataset [28] consists of 365,054 fashion product images annotated with 205 attributes and composed into 53,306/10,000/5,000 train/test/validation outfits. The Animals with Attributes 2 (AwA) dataset [31] consists of 37,322 natural images of 50 animal classes annotated with 85 attributes, and is split into 40 animal classes for training, and 10 used at test time. To evaluate our explanations we randomly sample 10,000 ground-truth (query, reference) pairs of similar images for each dataset from the test set.

Image Similarity Models. For the Polyvore Outfits dataset we use the type-aware embedding model released by Vasileva et al. [28]. This model captures item compatibility (i.e. how well two pieces of clothing go together) using a set of learned projections on top of a general embedding, each of which compares a specific pair of item types (i.e. a different projection is used when comparing a top-bottom pair than when comparing a top-shoe pair). For AwA we train a feature representation
Table 1: Comparison of candidate saliency map generator methods described in Section 3.2. We report AUC for the insertion and deletion metrics described in Section 4.1.

| Method          | Fixed Reference? | Polyvore Outfits | Animals with Attributes 2 |
|-----------------|------------------|------------------|---------------------------|
|                 |                  | Insertion (↑)    | Deletion (↓)             | Insertion (↑)    | Deletion (↓)             |
| Sliding Window  | Y                | 60.2             | 53.6                      | 76.9             | 76.8                      |
| RISE            | Y                | 62.0             | 52.0                      | 76.5             | 77.1                      |
| LIME            | Y                | 58.4             | 55.4                      | 77.0             | 71.2                      |
| Mask            | Y                | 59.4             | 53.3                      | 74.5             | 77.3                      |
| Sliding Window  | N                | 59.6             | 54.3                      | 77.6             | 76.3                      |
| RISE            | N                | 58.8             | 55.2                      | 76.0             | 75.6                      |
| Mask            | N                | 58.9             | 54.6                      | 75.8             | 78.4                      |

using a 18-layer ResNet [9] with a triplet loss function that encourages animals of the same type to embed nearby each other. For each dataset/model, cosine similarity is used to compare an image pair's feature representations.

4.1 Saliency Map Evaluation

Metrics. Following Petsiuk et al. [23], we evaluate the generated saliency maps using insertion and deletion metrics which measure the change in performance of the model being explained as pixels are inserted into a blank image, or deleted from the original image. For our task, we generate saliency maps for all query images, and insert or delete pixels in that image only. If a saliency map correctly captures the most important image regions, we should expect a sharp drop in performance as pixels are deleted (or a sharp increase as they are inserted). We report the area under the curve (AUC) created as we insert/delete pixels at a rate of 1% per step for both metrics. We normalize the similarity scores for each image pair across these thresholds so they fall in a [0-1] interval.

Results. Table 1 compares the different saliency map generation methods on the insertion and deletion tasks. We found no consistent winner between the two datasets, with RISE performing best on the Polyvore Outfits dataset and LIME obtaining best performance on the AwA dataset. This is not surprising, since LIME learns which super-pixels contribute to a similarity score. For AwA this means that parts of the animals could be segmented out and deleted or inserted in their entirety before moving onto the next super-pixel. On Polyvore Outfits, however, the important components may be along the boundaries of objects (e.g. the cut of a dress), something not well represented by super-pixel segmentation. Although Mask does not perform as well as other approaches, it tends to produce the most compact regions of salient pixels as it searches for a saliency map with minimal support (see our qualitative comparison of the different methods provided in Figure 3). Notably, we generally obtained better performance when the reference image was kept fixed and only the query image was manipulated. This may be due to the issues from noisy similarity scores as discussed in Section 3.2 and suggests extra care must be taken when manipulating both images.

4.2 Attribute Prediction Evaluation

Metrics. To begin, we report the overall performance of our attribute model using mean average precision (mAP) on the standard task of attribute recognition computed over all images in the test set. Two additional metrics are used to evaluate our attribute explanations using the (query, reference) image pairs used in the saliency map experiments. First, we measure the accuracy of the top scoring attribute explanation for each image (i.e. is the returned attribute among the ground truth annotations?) Second, we simulate the effect that removing the attribute from the image would have on the similarity score. After generating the attribute explanation for the query image, we find the most similar image to the query in the test set that does not contain that attribute. For AwA we use the ground truth attribute annotations to identify if an image has an attribute. For Polyvore Outfits, whose attributes are sparsely labeled, we also ensure that the retrieved image has low confidence in the attribute used for an explanation. After retrieving this new image, we compute its similarity with the reference image and return the difference in similarity compared with the original (query, reference) pair. Intuitively, if an attribute was critical for an explanation, then the similarity score should drop more than if a different attribute was selected. Examples of this process can be found in the appendix.
Table 2: Comparison of how attribute recognition (mAP) and attribute explanation (top1 accuracy, attr removal) metrics described in Section 4.2 are affected for different approaches. We use the fixed-reference RISE method as our saliency map generator for both datasets. Higher numbers are better for all metrics.

| Method                      | Polyvore Outfits | Animals with Attributes 2 |
|-----------------------------|------------------|--------------------------|
|                            | Top1 mAP | Attr Accuracy | Removal  | Top1 mAP | Attr Accuracy | Removal |
| Random                      | –        | 1.3          | 0.2      | –        | 38.1          | 0.4     |
| Attribute Classifier        | 24.2     | 49.1         | 0.5      | 66.5     | 73.9          | 0.9     |
| FashionSearchNet [1]        | 24.5     | 49.1         | 0.4      | 66.7     | 75.2          | 1.1     |
| FashionSearchNet + Map Matching | –   | 49.8         | 1.5      | –        | 77.8          | 1.4     |
| SANE                        | 25.7     | 50.0         | 2.2      | 67.1     | 77.1          | 1.8     |
| SANE + Map Matching         | –        | 51.7         | 2.9      | –        | 85.5          | 2.3     |
| SANE + Map Matching + Prior (Full) | –   | 52.2         | 3.5      | –        | 85.1          | 2.7     |

**Compared methods.** We provide three baseline approaches: a random baseline, a sample attribute classifier (i.e., no attribute activation maps), and a modified version of FashionSearchNet [1], an attribute recognition model which also creates a weakly-supervised attribute activation map for comparison. Additional details on these models can be found in the appendix.

**Results.** Table 2 compares the performance of the compared attribute models for our metrics. Our attribute removal metrics demonstrate the effectiveness of our attribute explanations, with our model which matches saliency maps getting the best performance on both datasets. This shows that when we “remove” the attribute predicted by SANE from the image, it has the largest drop in similarity score, compared to baselines. We also see that training our attribute model so it can produce explanations performs best even on the standard attribute recognition task measured with mAP. The top ranked attribute also becomes significantly more accurate when matching it to the saliency map produced for the query image, increasing top1 accuracy by almost 2% for Polyvore Outfits and 8.5% for AwA.
We provide qualitative examples of our explanations in Figure 4. Generally, attributes tend to “shift” as the categories of items changed: for instance, “bulbous” is often seen in examples like the hippopotamus example in the right column, but this became notably less common for categories like chimpanzee or Persian cat. Examples demonstrate that our explanations pass important sanity checks. Firstly, the explanation attribute is well-correlated with the localization of important pixels in the saliency map for each pair. Notice that “striped”, “knitted” and “embroidered” on the left of Figure 4 are sensibly localized, and are also reasonable explanations for the match, while a more abstract explanation like “feminine” is linked to the open toe of the heel, the curve of the sole, and the ankle strap. Secondly, the similarity scores are lower for pairs that are more dissimilar: the second row on the right achieves the lowest similarity score, with the explanation for the model’s decision being that the pig is a ground animal (while the whale is clearly not).

Note further that the explanations are non-trivial: they often differ from the most likely attribute in the query image, as predicted by a standard attribute classifier. In other words, our explanation model is indeed utilizing information from each pair of images and the saliency map characterizing the match to produce a sensible interpretable explanation. Lastly, it is a sensible sanity check to ask, does the same query image matched with different reference images result in different explanations? Our qualitative results demonstrate our explanation system indeed has this desirable property: note that in the bottom two rows on the right of Figure 4 the property that makes the hippopotamus similar to the leopard is that it is brown, but the property that makes it similar to the seal is that it is bulbous. We include more examples to support these observations in the appendix.

In Figure 5 we show an example of how directly removing the attribute predicted as the explanation can affect similarity (possible here because the attribute is a color.) Here we see that when we modify the white dress to be a different color, the similarity score drops significantly. The only exception is when we make the dress the same color (black) as the attribute explanation of the pants it is being
Figure 5: Example of the effect replacing the attribute used as an explanation of the model’s behavior has on image similarity score (higher score means items are more compatible).

Table 3: Our user study demonstrates users strongly prefer our computed explanations to random. Numbers show the percentage of tests where subject preferred our explanation, broken down by dataset and gender. Since there are 200 evaluations per dataset, standard errors are small, strongly discouraging explaining the prominent gap in female and male preferences by random effects.

| Dataset                | Total   | Male   | Female  |
|------------------------|---------|--------|---------|
| Polyvore Outfits       | 68.9%   | 63.3%  | 71.5%   |
| Animals with Attributes 2 | 80.0%   | 83.3%  | 78.3%   |

compared to. This demonstrates in a causal way how our predicted attributes can play a significant role in the similarity scores.

**User Study** To evaluate the quality of our explanations, we perform a user study with 20 subjects in the age range 14-50. Using a web form, we present 10 unique questions per subject per dataset of the type “What property of item B better explains why it matches item A?” for randomly selected pairs of similar images, and ask participants to select between the attribute provided by our explanation model, or a random one. We report the percentage of users that favored our explanations vs. random in Table 3. On both datasets, subjects prefer our explanations to random by a significant margin, with a prominent difference between the male and the female user pool on the Polyvore Outfits dataset.

**5 Conclusion**

In this paper we introduced SANE, a method of explaining an image similarity model’s behavior by identifying attributes which were important to the similarity score paired with saliency maps indicating important image regions. We confirmed that humans commonly agree with the attributes selected by SANE to supplement our comparison using machine-generated metrics. In future work we believe closely integrating the saliency generator and attribute explanation model, enabling each component to take advantage of the predictions of the other, would help improve performance.

**References**

[1] K. E. Ak, A. A. Kassim, J. H. Lim, and J. Y. Tham. Learning attribute representations with localization for flexible fashion search. In *CVPR*, 2018.
[2] L. Anne Hendricks, R. Hu, T. Darrell, and Z. Akata. Grounding visual explanations. In *ECCV*, 2018.
[3] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran. Zero-shot object detection. In *ECCV*, 2018.
[4] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. S. Huang. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *ICCV*, 2015.
[5] C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Explaining image classifiers by counterfactual generation. In *ICLR*, 2019.
[6] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.
[7] X. Han, Z. Wu, P. X. Huang, X. Zhang, M. Zhu, Y. Li, Y. Zhao, and L. S. Davis. Automatic spatially-aware fashion concept discovery. In *ICCV*, 2017.
[8] X. Han, Z. Wu, Y.-G. Jiang, and L. S. Davis. Learning fashion compatibility with bidirectional lstms. In *ACM MM*, 2017.
A Candidate Salience Map Generator Descriptions

In this section we provide additional details about each of the candidate saliency map generation methods used in our paper. We split these approaches into two groups: methods which analyze behavior solely through input manipulation (described in Section A.1) and those which use an optimization procedure to learn some parameters in combination with input manipulation (described in Section A.2). Please see Section 3.2 for a description of how these methods are adapted to our task. We also provide a runtime comparison of each approach in Table 4.
A.1 Saliency Maps by Input Manipulation

A straightforward approach to producing a saliency map is to manipulate the input image by removing image regions and measuring the effect this has on the similarity score. If a large drop in similarity is measured, then the region must be important to this decision. If almost no change was measured, then the model considers the image region irrelevant. The saliency map is generated from this approach by averaging the similarity scores for each pixel location over all instances where it was removed from the input. The challenge then is to determine how to manipulate the input image to discover these important regions.

Sliding Window [34]. The first approach to removing regions of an image we shall discuss is a sliding window, where regions are sampled regularly across an image. There is a direct tradeoff, however, with how densely frames are sampled and the computational time it takes to do a forward pass through the network for each manipulated image. If frames are not densely sampled to enable an efficient solution, then it wouldn’t be able to localize important regions accurately. If regions are too densely sampled then removing them might not make enough of a difference in the similarity score to take measurements accurately.

RISE [23]. This method uses Monte Carlo approach to generate saliency maps. A set of \( N \) random binary masks of size \( h \times w \) is sampled where each element is independently set to 1 with probably \( p \), and all other elements are set to 0. Typically these masks are much smaller than the input image, so they are upsampled using bilinear interpolation. This produces small continuous regions within the upsampled mask that can be used to manipulate the input image. To remove the fixed grid structure the masks are upsampled to larger than image size and then cropped randomly. Although this approach does require a significant number of random masks (we found 2,000 to be sufficient in our experiments), we found this approach significantly outperforms using a sliding window that samples a similar number of masks on our task.

A.2 Learned Saliency Maps

We shall now discuss methods which combine input manipulation with an optimization procedure used to directly learn a saliency map. As in Section A.1, we compare generating saliency maps for a single query image at a time using a fixed reference image as well as generating a saliency map by manipulating both the query and reference images.

LIME [25]. Rather than masking regions without any concern over the continuity of a region, this approach to generating saliency maps operates over a superpixel segmentation of an image. Images are manipulated by randomly deleting superpixels in the image. After sampling \( N \) manipulated inputs, the importance of each superpixel is estimated using Lasso. Finally, important regions are selected using submodular optimization.

Mask [6]. In this approach a low resolution saliency map is directly learned using stochastic gradient decent and upsampled to the image size. Instead of manipulating an image by just deleting regions as in other methods, two additional perturbation operators are defined: adding Gaussian noise and image blurring. To help avoid artifacts when learning the mask a total-variation norm is used in addition to an \( L1 \) regularization to promote sparsity. This approach removes the reliance on superpixels and tends to converge in fewer iterations than LIME, although it is considerably slower in practice than other approaches (see Table 4). That said - one advantage it does have over other approaches is the ability to learn the salience map for both the query and reference image jointly (which we take advantage of when we are not using a fixed reference image).

B Additional Experimental or Implementation Details

B.1 Compared Methods

In addition to a random baseline, we provide two for comparison to our model for our attribute experiments in Section 4.2 of the paper. First, we train a simple attribute classifier (i.e. no attribute activation map). Second, we use a modified version of FashionSearchNet [1], which was designed for fashion search using attribute information. This network uses an attribute activation map to identify and extract a region of interest for each attribute. These extracted regions are fed into two branches consisting of three fully connected layers which is trained for both attribute classification and image
Table 4: Runtime comparison of the compared saliency generation methods and how using a fixed reference image, or manipulating both the query and reference images affects performance.

| Method       | Fixed Reference? | Time(s) |
|--------------|------------------|---------|
| Sliding Window | Y                | 0.2     |
| RISE         | Y                | 0.3     |
| LIME         | Y                | 1.2     |
| Mask         | Y                | 4.1     |
| Sliding Window | N               | 2.5     |
| RISE         | N                | 5.8     |
| Mask         | N                | 7.2     |

retrieval. We remove the image retrieval components, and use the same 18-layer ResNet base image encoder used for our other methods (replacing AlexNet [17] which was used for the image encoder in the original paper). This provides a simple baseline and a model with a generic weakly-supervised attribute activation map for comparison.

B.2 Saliency Map Generator Details

**Sliding Window.** When manipulating the inputs of the reference image, we apply 625 occlusion windows each covering a square region of about 12% of image area. When manipulating both images we apply 36 occlusion windows to the reference image.

**RISE.** For both datasets we randomly sample 2,000 random masks upsampled from $8 \times 8$ mask with the probability of preserving a region of 0.5. When manipulating the inputs of the reference image, we generate 30 random masks.

**LIME.** We generate LIME saliency maps using 1000 samples.

**Mask.** We learn a $14 \times 14$ perturbation mask for both datasets. We train the mask for 500 iterations using Adam [16] with a learning rate of 0.1.

B.3 SANE Details

Due to its efficient (see Table 4) and overall good performance (see Table 1 in the paper) we selected the fixed-reference RISE as our saliency map generator. For each training image, we sample up to five similar images using the ground truth annotations of each dataset and generate saliency maps using each sampled image as the reference image. We train our attribute model for 300 epochs using Adam [16] with a learning rate of $5e^{-4}$ and set $\lambda = 5e^{-3}$ in Eq. (2) from the paper. After each epoch, we computed mAP on the validation set and kept the best performing model according to this metric. At test time $\phi_{1-3}$ are set to (0.1, 0.9, 0.05) on Polyvore Outfits, respectively, and (0.4, 0.6, 0.05) for AwA, respectively. Effectively, map matching obtained the largest weight on both datasets, followed by attribute confidence, with the prior only taking a small weight.

We provide an example of the attribute removal process in Figure 6. After identifying an attribute to remove in an image, we search for the most similar image to the input from a database that doesn’t contain the input attribute. We see on the left side of Figure 6 that some attributes like colors are largely retained when the attribute has to do with a non-color based attribute. On the returned AwA images on the right side of Figure 6 we see how some attributes can lead to significant changes in the images or almost none at all depending on the attribute selected to remove.

In Section 3.3 we discuss how we estimate how likely each attribute is a “good” explanation in held-out data. This is used as a prior to bias our attribute selections towards attributes that are known to be good attribute explanations. In Figure 7 we show the prior for the AwA dataset. Note, however, that this prior would change for a different image similarity model. For example, if the image similarity model was more biased towards colors, then we would expect to see the likelihood for “black,” “brown,” and “gray” to increase.

B.4 User Study Examples

Users were tasked with selecting an attribute which best describes why two items are similar. One attribute was selected by our model, and the other was selected at random. An example of the
Figure 6: Examples of the attribute removal process used to evaluate how good an attribute is as an explanation. We measure the similarity of the input image and some reference image as well as between the returned image and the reference image. If a large drop in similarity is measured then the attribute is considered a “good” explanation. If similarity stays about the same or increases, the attribute is considered a “poor” explanation, e.g., trying to remove “active” from the pandas on the right.

Figure 7: The likelihood each attribute in the AwA dataset was identified as the best attribute for an image pair on held-out data. We use this prior in Section 3.3 as a bias in our attribute selection procedure.
Figure 8: Examples of the style of question we asked in the user study.

Table 5: Discovered attribute removal

| Dataset                | Random | Full frame | Patch | Supervised |
|------------------------|--------|------------|-------|------------|
| Polyvore Outfits       | 1.6    | 1.8        | 2.1   | 3.5        |
| Animals with Attributes| 2.0    | 1.4        | 2.2   | 1.8        |

questions presented to users for each dataset is provided in Figure 8. The results are found in Section 4.2.

C Discovering Useful Attributes

For datasets without attribute annotations, or those where the annotated attributes doesn’t cover the extent of the visual attributes present in the dataset (i.e. there are many unannotated attributes) we propose a method of discovering attributes that are useful for providing model explanations. An attribute that is useful for explanations would commonly appear in the high importance regions of saliency maps. When generating saliency maps for a query image, if many reference images attend to the same region of the query image then it is likely they are all matching to it for similar reasons (i.e. there may be some attribute that they share which matches the query). Given this observation, we discover attributes using the following saliency-based procedure:

1. Obtain $K$ similar images for query image $q$ using k-NN.
2. Generate a saliency map over $q$ for each of the similar (reference) images.
3. Keep only those reference images which have their saliency peaks in the most common location (such as a unit square in a $7 \times 7$ grid) and pick top $N$ of them that have the highest similarity.
4. For each reference image, generate its saliency map with $q$ and crop a $30 \times 30$ patch around the peak saliency region in the reference image.
5. Upsample all the generated patches to full image resolution and get their embeddings.
6. Cluster the patches produced for multiple queries $q$. Each cluster represents an attribute. If multiple patches were extracted from an image and they got assigned to different clusters, this image would be labeled with multiple attributes.

Figure 9a illustrates the clustering produced by this procedure for a set of queries from Polyvore Outfits dataset.

To evaluate this approach we compare it to randomly assigning images to clusters and to clustering based on their own embeddings, disregarding the saliency of image regions (Figure 9b). Saliency-based attribute discovery works best among the three unsupervised methods for Polyvore Outfits data, but full-frame clustering outperforms it for the AwA dataset (Table 5). We suspect the full frame clustering works better for AwA since it considers the background more than the patch-based method (Polyvore Outfits image’s typically have white backgrounds). In addition, our discovered attributes would likely be noisier due to the similarity model focusing on the background patches in some images as well. Although our initial results are promising, attempting to discover attributes useful for explanations warrants additional investigation.
Figure 9: Six clusters defining the attributes for two approaches to attribute discovery.
(a) Each image is assigned a list of clusters that have patches from this image. Clustering is performed on salient patches.
(b) Each image is assigned one of the clusters as an attribute. Clustering is performed on full-frame images.
Figure 10: Additional qualitative examples comparing the saliency map generator candidates on the Polyvore Outfits dataset.
Figure 11: Additional qualitative examples of our SANE explanations on the Polyvore Outfits dataset.
Figure 12: Additional qualitative examples of our SANE explanations on the AwA dataset.