Thinking Outside the Pool:
Active Training Image Creation for Relative Attributes

Aron Yu
University of Texas at Austin
aron.yu@utexas.edu

Kristen Grauman
UT Austin and Facebook AI Research
grauman@cs.utexas.edu

Abstract

Current wisdom suggests more labeled image data is always better, and obtaining labels is the bottleneck. Yet curating a pool of sufficiently diverse and informative images is itself a challenge. In particular, training image curation is problematic for fine-grained attributes, where the subtle visual differences of interest may be rare within traditional image sources. We propose an active image generation approach to address this issue. The main idea is to jointly learn the attribute ranking task while also learning to generate novel realistic image samples that will benefit that task. We introduce an end-to-end framework that dynamically “imagines” image pairs that would confuse the current model, presents them to human annotators for labeling, then improves the predictive model with the new examples. On two datasets, we show that by thinking outside the pool of real images, our approach gains generalization accuracy on challenging fine-grained attribute comparisons.

1. Introduction

Visual recognition methods are famously data hungry. Today’s deep convolutional neural networks (CNNs) achieve excellent results on various challenging tasks, and one critical ingredient is access to large manually annotated datasets [6, 8, 18, 59]. A common paradigm has emerged where a supervised learning task is defined, the relevant images/videos are scraped from the Web, and crowdworkers are enlisted to annotate them. A practical downside to this paradigm, of course, is the burden of getting all those annotations, which can be significant.

However, setting efficiency concerns aside, we contend that the standard paradigm also quietly suffers a curation concern. The assumption is that the more data we gather for labeling, the better. And indeed unlabeled photos are virtually unlimited. Yet—particularly when relying on Web photos—the visual variety and information content of the images amassed for labeling eventually reach a limit. The resulting deep network inherits this limit in terms of how well it can generalize. Importantly, while active learning methods can try to prioritize informative instances for labeling, the curation problem remains: existing active selection methods scan the pool of manually curated unlabeled images when choosing which ones to label [12, 54, 55, 65].

We propose to address the curation challenge with active image generation. The main idea is to jointly learn the target visual task while also learning to generate novel realistic image samples that, once manually labeled, will benefit that task. We focus specifically on the fine-grained relative attribute task [33, 42, 49, 50, 58]: given two images, the system must infer which one more evidently exhibits an attribute (e.g., which is more furry, more red, more smiling). The curation problem is particularly pronounced for fine-grained relative attributes. This is because while at test time an image pair may be arbitrarily close in an attribute, it is difficult to locate extensive training samples showing such subtle differences for all attributes [61]. Our idea is for the system to dynamically “imagine” image pairs that would confuse the current learning-to-rank model, then present them to human annotators for labeling. See Figure 1.

To this end, we introduce an end-to-end deep network consisting of an attribute-conditioned image generator, a ranking network, and a control network. The network is seeded with real labeled image pairs of the form “image $X$ exhibits attribute $a$ more than image $Y$”. Then the control network learns to generate latent visual parameters to present to the image generator so as to form image pairs

![Figure 1: Our approach learns to actively create training images for fine-grained attribute comparisons, focusing attention on informative pairs of images unavailable in the real image training pool. Here we show two example pairs it generated to improve learning of open (left) and masculine (right).](image-url)
that would be difficult for the ranker. Thus, rather than simply perturb existing training images (à la “fooling images” [31]), the control module learns a function to create novel instances, potentially improving the exploration of the relevant image space. We train the ranker and controller in an adversarial manner, and we solicit manual labels for the resulting images in batches. As a result, the set of imagined training instances continues to evolve, as does the ranker.

Our main contribution is a new approach to active learning in which samples are newly created so as to best augment a ranker’s training data. On two challenging datasets of Shoes [59, 61] and Faces [18, 42], we demonstrate our approach’s effectiveness. Active generation focuses attention on novel training images that rapidly improve generalization—even after all available real images and their labels are exhausted. It outperforms alternative data augmentation strategies, including prior work that generates synthetic training images in a passive manner [61].

2. Related Work

Attribute Comparisons Relative attributes [33] model visual properties in a comparative manner, and a variety of applications have been explored in online shopping [24], fashion [23], biometrics [41], and graphical design [52]. Recent work explores novel learning schemes to learn relative attributes accurately, including new deep network architectures [49, 50, 58], part discovery [42], local learning [59, 60], and multi-task approaches [29]. Our contribution is an approach to actively elicit training samples for ranking, which could facilitate training for many of the above formulations.

Image Generation Photo-realistic image generation has made steady and exciting progress in the recent years, often using generative adversarial networks (GANs) [13, 20, 40, 62, 63, 67] or variational auto-encoders (VAEs) [14, 25, 56]. Conditional models can synthesize an image based on a label map [26, 67] or latent attribute label [27, 52, 56]. We integrate the Attribute2Image [56] network as the image generation module in our full pipeline, though similarly equipped conditional models could also be used. Methods to generate adversarial “fooling images” also use a network to automatically perturb an image in a targeted manner [3, 30, 31]. However, rather than using adversarial generation to understand how features influence a classifier, our goal is to synthesize the very training samples that will strengthen a learned ranker. Unlike any of the above, we create images for active query synthesis.

Learning with Synthetic Images The idea of training on synthetic images but testing on real ones has gained traction in recent years. Several approaches leverage realistic human body rendering engines to learn 3D body pose [45, 46, 53], while others extrapolate from video [7, 22, 34] or leverage 3D scene [64] or object [35, 57] models. Refiner networks that translate between the simulated and real data can help ease the inevitable domain gap [48].

Data augmentation (used widely in practice) can be seen as a hybrid approach, where each real training sample is expanded into many samples by applying low-level label-preserving transformations, or “jitter”, like scaling, rotating, mirroring, etc. [88]. Usually the jitter is manually defined, but some recent work explores how to adversarially generate “hard” jitter for body pose estimation [37], greedily select useful transformations for image classification [35], or actively evolve 3D shapes to learn shape from shading [57]. Such methods share our motivation of generating samples where they are most needed, but for very different tasks. Furthermore, our approach aligns more with active learning than data augmentation: the new synthetic samples can be distant from available real samples, and so they are manually annotated before being used for training.

In the low-shot recognition regime, several recent methods explore creative ways to hallucinate the variability around sparse real examples [9, 15, 16, 26, 33, 61], typically leveraging the observed inter-sample transformations to guide synthesis in feature-space. Among them, most relevant to our work is our “semantic jitter” approach [61], which generates relative attribute training images by altering one latent attribute variable at a time. However, whereas semantic jitter [61] uses manually defined heuristics to sample synthetic training images, we show how to dynamically derive the images most valuable to training, via an adversarial control module learned jointly with the attribute ranker.

Active Learning Active learning has been studied for decades [44]. For visual recognition problems, pool-based methods are the norm: the learner scans a pool of unlabeled samples and iteratively queries for the label on one or more of them based on uncertainty or expected model influence (e.g., [12, 54, 55, 63]). Active ranking models adapt the concepts of pool-based learning to select pairs for comparison [28, 39]. Hard negative mining—often used for training object detectors [11, 47]—also focuses the learner’s attention on useful samples, though in this case from a pool of already-labeled data. Rather than display one query image to an annotator, the approach in [19] selects a sample from the pool then displays a synthesized image spectrum around it in order to elicit feature points likely to be near the true linear decision boundary for image classification. We do not perform pool-based active selection. Unlike any of the above, our approach creates image samples that (once labeled) should best help the learner, and it does so in tight coordination with the ranking algorithm.

In contrast to pool-based active learning, active query synthesis methods request labels on novel samples from a
3. Approach

We propose an end-to-end framework for attribute-based image comparison through active adversarial image generation. We refer to our approach as ATTIC, for Active Training Image Creation.

The goal is to avoid the “streetlight effect” of traditional pool-based active learning, where one looks for more training data where one already has it. Rather than limit training to manually curated real images, as would standard pool-based active learning, ATTIC synthesizes image pairs that will be difficult for the ranker as it has been trained thus far. See Figure 2.

We first define the relative attributes problem (Sec. 3.1). Then we describe the key modules in our ATTIC network (Sec. 3.2). Finally we define our training procedure and the active image creation active loop (Sec. 3.3).

3.1. Relative Attributes via Ranking

Relative attributes are valuable for inferring fine-grained differences between images in terms of describable natural language properties. They have wide application in image search, zero-shot learning, and biometrics, as discussed above. At the core of the relative attribute comparison task is a learning-to-rank problem \[33, 49\]. A ranking function \( R_\mathcal{A} : \mathbb{X} \to \mathbb{R} \) for attribute \( \mathcal{A} \) (e.g., comfortable, smiling) maps an image to a real-valued score indicating the strength of the attribute in the image.

Relative attribute rankers are trained using pairs of training images. Let \( \mathcal{P}_\mathcal{A} = \{(x_i, x_j)\} \) be a set of ordered pairs, where humans perceive image \( x_i \) to have “more” of attribute \( \mathcal{A} \) than image \( x_j \). The goal is to learn a ranking function \( R_\mathcal{A} \) that satisfies the relative orderings as much as possible: \( \forall (i,j) \in \mathcal{P}_\mathcal{A} : R_\mathcal{A}(x_i) > R_\mathcal{A}(x_j) \). Having trained such a function \( R_\mathcal{A} \) we can use it to score images for their attribute strength individually, or, similarly, to judge which among a novel pair of images \((x_m, x_n)\) exhibits the attribute more. ATTIC trains a network for each attribute \( \mathcal{A} \); we drop the subscript below when not needed for clarity.

3.2. End-to-End Active Training Image Creation

Let \( \mathcal{P}^R \) be an initial set of real training image pairs used to initialize the ranker \( R \) defined above. Our goal is to improve that ranker by creating synthetic training images \( \mathcal{P}^S \), to form a hybrid training set \( \mathcal{P} = \mathcal{P}^R \cup \mathcal{P}^S \).

To this end, the proposed end-to-end ATTIC framework consists of three distinct components (Fig. 3): the ranker module, the generator module, and the control module. Our model performs end-to-end adversarial learning between the ranker and the control modules. The ranker tries to produce accurate attribute comparisons, while the control module tries to produce control parameters—latent image parameters—that will generate difficult image pairs to confuse the current ranker. By asking human annotators to label those confusing pairs, the ranker is actively improved. We next discuss the individual modules.

3.2.1 Ranking Module

For the ranking module in ATTIC, we employ the state-of-the-art deep DeepSTN approach [49], which integrates a Siamese embedding and a spatial transformer network.
(STN) [21] together with a RankNet ranking loss [5]. RankNet handles pairwise outputs in a single differentiable layer, using a cross-entropy loss. The rank estimates \((y^A, z^A), (y^B, z^B)\). Its architecture is detailed further in Figure 4. The generator module then generates a pair of synthetic images \((\hat{x}^A, \hat{x}^B)\) using these control parameters. The ranker module finally uses the generated synthetic images (once manually labeled) and the real training images to train the ranking model, outputting their corresponding attribute strength \((v^A, v^B)\).

During training, the ranking loss is fed back into the ranker (green dotted line), while the negative ranking loss is fed back into the control module (red dotted line). The decoders within the generator are pre-trained and their parameters are kept frozen throughout training.

\[
p_{ij} = \frac{1}{1 + e^{-(v_i - v_j)}},
\]

(1)

and the ranking loss is:

\[
L_{rank} = -\log(p_{ij}).
\]

(2)

The ranker jointly learns a CNN image encoder \(\phi(x)\) with a spatial transformer that localizes the informative image regions for judging a particular attribute. For example, the STN may learn to clue into the mouth region for the smiling attribute. The ranker combines the features of the full image and this region to compute the attribute strength:

\[
v_i = R_A(x_i) = \text{RankNet}_A(\text{STN}(\phi(x_i))),
\]

(3)

where \(\phi(x)\) denotes application of one stack in the Siamese embedding. See [49] for details.

### 3.2.2 Generator Module

For the generator module, we use an existing attribute-conditioned image generator, Attribute2Image [56]. Let \(y \in \mathbb{R}^N\) denote an \(N\)-dimensional attribute vector containing the real-valued strength of each of the \(N = |A|\) attributes for an image \(x\), and let \(z \in \mathbb{R}^M\) denote an \(M\)-dimensional latent variable accounting for all other factors affecting the image, like lighting, pose, and background. The Attribute2Image network uses a Conditional Variational Auto-Encoder (CVAE) to learn a decoder \(p_\theta(\hat{x}|y, z)\) that can generate realistic synthetic images \(\hat{x}\) conditioned on \((y, z)\). The parameters \(\theta\) are optimized by maximizing the variational lower bound of the log-likelihood \(\log p_\theta(\hat{x}|y)\). The network uses a multilayer perceptron to map the latent factors into their entangled hidden representation, then decodes into image pixels with a coarse-to-fine convolutional decoder. See [56] for more details.

The attribute-conditioned aspect of this generator allows us to iteratively refine the generated images in a semantically smooth manner, as we adversarially update its inputs \((y, z)\) with the control module defined next. We pre-train the generator using \(\{(x_i, y_i)\}\), a disjoint set of training images labeled by their \(N\) attribute strength labels. Subsequently, we take only the decoder part of the model and use it as our generator (see Fig. 3). We freeze all parameters in the generator during active image creation, since the mapping from latent parameters to pixels is independent of the rank and control learning.

### 3.2.3 Control Module

As defined thus far, linking together the ranker and generator would aimlessly feed new image sample pairs to the...
3.3. Training and Active Image Creation

Given the three modules, we connect them in sequence to form our active learning network. The generator and the ranker modules are duplicated for both branches to account for two images in each training pair. The decoders in the generator module are pre-trained and their parameters are kept frozen. During training, we optimize the RankNet loss for the ranker module, while at the same time optimizing the negative RankNet loss for the control module:

$$L_{control} = -L_{rank}. \quad (4)$$

The control module thus learns to produce parameters that generate image pairs that are difficult for the ranker to predict. This instills an adversarial effect where the control module and the ranker module are competing against each other iteratively during training. The learning terminates when the ranker converges. Our adversarial losses promote “hard” examples, and are not to be confused with adversarial “fooling” images [3, 20, 31]: image pairs generated by our method are only fed to the learner if humans can label them confidently.

Figure 5 shows examples of the progression of synthetic images over training iterations. ATTIC captures the interaction between the attributes and traverses the multi-attribute space to places not occupied by real training images.

To generate a batch of synthetic image pairs $P^S = \{\hat{x}^A, \hat{x}^B\}_{i=1}^T$, we sample $T$ vectors $q$ and push them through the control and generator. Then the batch is labeled by annotators, who judge which image shows the attribute more. The resulting pairs accepted by annotators as valid are added to the hybrid training set $P$. We consider with both single-batch, where all synthetic pairs are generated in one go (Sec. 4.2), and multi-batch settings, where we reset the ranker module for each subsequent batch (Sec. 4.3).

The primary novelty of our approach comes from the generation of synthetic image pairs through active query synthesis. From an active learning perspective, instead of selecting more real image pairs to be labeled based on existing pool-based strategies, our approach aims to directly generate the most beneficial synthetic image pairs (Fig. 2). Furthermore, instead of choosing $(y_i, z_i)$ manually when generating the synthetic image pairs [61], ATTIC automates this selection in a data-driven manner.

Here we mean “batch” in the active learning sense: a batch of examples to be manually labeled and then used to augment the training set. This is not to be confused with (mini-)batches for training the neural nets.
4. Experiments

To validate our active generation approach, we explore two fine-grained attribute domains: shoes and faces. To our knowledge, these are the only existing datasets with sufficient instance-level relative attribute labels.

4.1. Experimental Setup

Datasets We use two publicly available datasets, UT-Zappos50K \[^{59}\] and LFW \[^{18}\]. For each, our method uses real images to initialize training and then creates its own synthetic images.

- **Catalog Shoe Images**: We use the UT-Zappos50K dataset \[^{59}\] with the fine-grained attributes from \[^{61}\]. The dataset consists of over 50,000 shoe images from Zappos.com. There are 10 attributes (comfort, casual, simple, sporty, colorful, durable, supportive, bold, sleek, and open), each with about 4,000 labeled pairs.

- **Human Face Images**: We use the LFW dataset \[^{18}\] and the LFW-10 dataset \[^{42}\]. LFW consists of over 13,000 celebrity face images from the wild along with real-valued labels on 73 attributes. LFW-10 consists of a subset of 2,000 images from LFW along with relative labels on 10 attributes. We use the 8 attributes (bald, dark hair, big eyes, masculine, mouth open, smiling, visible forehead, and young) in the intersection of these two datasets. For the real image pairs, there are about 500 labeled pairs per attribute from LFW-10.

All images for all methods are resized to 64 x 64 pixels to match the output resolution of the image generator. For all experiments, we use high quality (high agreement/high confidence) relative labels, following the protocol of \[^{61}\]. We collect annotations for our method’s automatically generated training pairs using Mechanical Turk; we obtain 5 worker responses per label and take the majority vote. Workers are free to vote for discarding a pair if they find it illegible, which happened for just 17% of the generated pairs. Of the other 83% accepted pairs, at least 4 of 5 annotators agree on the same label 63% of the time, and they rate 48% of their annotations as “very confident” and 95% as at least “somewhat confident”. See Supp File for the data collection interface and instructions to annotators.

Implementation Details During training, we validate all hyperparameters (learning rate, learning rate decay, weight decay) on a validation set. We run all experiments (including individual batches) to convergence or to a max of 250 and 100 epochs for shoes and faces, respectively.

- For the individual modules, implementation details are as follows. Ranker: We pre-train the DeepSTN ranking network without the global image channel using only the real image pairs (see \[^{49}\] for details on the two rounds of training). Generator: We use the code provided by the authors of Attribute2Image \[^{56}\] with all default parameters. We pre-train the image generators using a disjoint set of real images (38,000 and 11,000 images for shoes and faces, respectively), which do not have any associated relative labels. We freeze the decoder in ATTIC throughout end-to-end training for the ranker and control. Control: We initialize the layers using ReLU initialization \[^{17}\].

Baselines We consider the following baselines:

- **SemJitter \[^{61}\]**, an existing approach for data augmentation that uses synthetic images altered by one attribute at a time using a manually defined sampling policy. We use the authors’ provided synthetic shoe image pairs, Zap50K-Synth, for this purpose. It contains 2,000 labeled pairs per attribute. For the synthetic face image pairs, we collect relative labels on 1,000 pairs per attribute.

- **Jitter**, the traditional data augmentation process where the real images are jittered through low-level geometric and photometric transformations. We follow the jitter protocol defined in \[^{10}\], which includes translation, scaling, rotation, contrast, and color. The jittered image pairs retain the corresponding real pairs’ respective labels.

- **Real**, which trains with only real labeled image pairs.

Note that all methods use the same state-of-the-art ranking network for training and predictions, hence any differences in results will be attributable to the training data and augmentation strategy.

4.2. Competing with Real Training Images

First we test our hypothesis that restricting models to only available Web photos for training can be limiting. We compare the data augmentation baselines and ATTIC to the Real baseline, where all methods are given the exact same amount of total manual annotations. The Real and Jitter baselines use all \(n\) available real labeled image pairs. SemJitter and ATTIC use \(\frac{n}{2}\) of the real labeled image pairs (\(\frac{n}{2}\)), which are sampled uniformly at random, and then augment those pairs with \(\frac{n}{2}\) manually labeled synthetic image pairs that they generate.

Tables 1 and 2 show the results for the Shoes and Faces datasets, respectively. We first look at the standard scenario (middle rows) where the synthetic images generated by SemJitter and our method are labeled by human annotators. Though using exactly the same amount of manual labels as the Real baseline, our method nearly always outperforms it. This shows that simply having more real image pairs labeled is not always enough; our generated sam-
<table>
<thead>
<tr>
<th></th>
<th>Comfort</th>
<th>Casual</th>
<th>Simple</th>
<th>Sporty</th>
<th>Colorful</th>
<th>Durable</th>
<th>Supportive</th>
<th>Bold</th>
<th>Sleek</th>
<th>Open</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real [49]</td>
<td>84.26</td>
<td>88.58</td>
<td>88.99</td>
<td>88.18</td>
<td>94.10</td>
<td>82.83</td>
<td>83.96</td>
<td>88.25</td>
<td>84.35</td>
<td>83.87</td>
<td>86.74</td>
</tr>
<tr>
<td>Jitter</td>
<td>82.87</td>
<td>87.96</td>
<td>86.65</td>
<td>87.58</td>
<td>93.91</td>
<td>80.51</td>
<td>84.63</td>
<td>88.66</td>
<td>83.37</td>
<td>79.84</td>
<td>85.60</td>
</tr>
<tr>
<td>SemJitter [61]</td>
<td>83.10</td>
<td>89.20</td>
<td>88.76</td>
<td>88.49</td>
<td>94.10</td>
<td>82.37</td>
<td>85.08</td>
<td>89.07</td>
<td>86.31</td>
<td>82.26</td>
<td>86.87</td>
</tr>
<tr>
<td>ATTIC (ours)</td>
<td>83.80</td>
<td>89.51</td>
<td>89.23</td>
<td>88.18</td>
<td>94.10</td>
<td>85.85</td>
<td>86.41</td>
<td>88.04</td>
<td>87.78</td>
<td>83.07</td>
<td>87.62</td>
</tr>
<tr>
<td>SemJitter-Auto [61]</td>
<td>84.72</td>
<td>88.89</td>
<td>89.70</td>
<td>89.39</td>
<td>94.29</td>
<td>83.99</td>
<td>86.41</td>
<td>89.07</td>
<td>85.82</td>
<td>83.87</td>
<td>87.59</td>
</tr>
<tr>
<td>ATTIC-Auto (ours)</td>
<td>87.04</td>
<td>89.20</td>
<td>91.57</td>
<td>91.21</td>
<td>94.48</td>
<td>87.94</td>
<td>87.31</td>
<td>89.90</td>
<td>86.31</td>
<td>85.75</td>
<td>89.07</td>
</tr>
</tbody>
</table>

Table 1: Accuracy for the 10 attributes in the Shoes dataset. The standard scenario (middle rows) is where the synthetic images generated by SemJitter and our method are labeled by human annotators. The “Auto” scenario (last two rows) is where an additional \( n \) unlabeled synthetic image pairs are added for all methods except Real; those images adopt their inferred attribute labels. Note that Jitter already uses all \( n \) labeled real images plus \( n \) (not manually labeled) jittered images. In all cases, all methods use exactly \( n \) total labels.

<table>
<thead>
<tr>
<th></th>
<th>Bald</th>
<th>DarkHair</th>
<th>BigEyes</th>
<th>Masculine</th>
<th>MouthOpen</th>
<th>Smiling</th>
<th>Forehead</th>
<th>Young</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real [49]</td>
<td>79.80</td>
<td>86.77</td>
<td>78.18</td>
<td>92.96</td>
<td>87.50</td>
<td>74.44</td>
<td>80.00</td>
<td>78.76</td>
<td>82.30</td>
</tr>
<tr>
<td>Jitter</td>
<td>81.82</td>
<td>86.87</td>
<td>80.00</td>
<td>91.55</td>
<td>84.17</td>
<td>79.70</td>
<td>83.16</td>
<td>81.35</td>
<td>84.81</td>
</tr>
<tr>
<td>SemJitter [61]</td>
<td>81.82</td>
<td>87.50</td>
<td>83.64</td>
<td>92.96</td>
<td>88.33</td>
<td>79.70</td>
<td>83.16</td>
<td>81.35</td>
<td>84.99</td>
</tr>
<tr>
<td>ATTIC (ours)</td>
<td>84.85</td>
<td>88.24</td>
<td>85.46</td>
<td>95.78</td>
<td>81.96</td>
<td>84.21</td>
<td>80.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SemJitter-Auto [61]</td>
<td>82.83</td>
<td>88.24</td>
<td>81.82</td>
<td>94.37</td>
<td>85.00</td>
<td>75.19</td>
<td>83.16</td>
<td>79.28</td>
<td>83.73</td>
</tr>
<tr>
<td>ATTIC-Auto (ours)</td>
<td>85.86</td>
<td>90.44</td>
<td>83.36</td>
<td>94.37</td>
<td>82.50</td>
<td>76.69</td>
<td>84.26</td>
<td>78.24</td>
<td>84.59</td>
</tr>
</tbody>
</table>

Table 2: Accuracy for the 8 attributes in the Faces dataset. Format is the same as Table 1.

Figure 6: Active learning curves for the Shoes (left) and Faces (right) datasets. We show the average gain over the Real baseline after each batch of additional generated image pairs. Our approach nearly doubles the gain achieved by SemJitter for both domains.

4.3. Active vs. Passive Training Image Generation

Next we examine more closely ATTIC’s active learning behavior. In this scenario, we suppose the methods have exhausted all available real training data (i.e., we use all \( n \) real labeled image pairs to initialize the model), and our goal is to augment this set. We generate the synthetic (labeled) image pairs in batches (again, not to be confused with the mini-batches when training neural networks). After each batch, we have them annotated, update the ranker’s training set, and re-evaluate it on the test set. The weights of the
control module are carried over from batch to batch, while the ranker module restarts at its pre-trained state at the beginning of each batch.

Figure 6 shows the results for both datasets. We plot active learning curves to show the accuracy improvements as a function of annotator effort—steeper curves are better, as they mean the system gets more accurate with less manual labeling. We see the average gains of our approach over the Real baseline increase most sharply compared to the baselines. Our approach achieves a gain of over 3% and 8% for the two domains, respectively, which is almost double that of the SemJitter baseline. Jitter falls short once again, suggesting that traditional low-level jittering has limited impact in these fine-grained ranking tasks. Please see Supp File for the individual performance plots of each attribute. ATTIC improves more than ATTIC-Auto, though has the burden of gathering labels from annotators.

4.4. Qualitative Analysis

The results show the synthetic image pairs generated by our approach outperform those selected by the heuristic and passive selection processes of SemJitter and Jitter in most scenarios. The advantage of our active generation approach is its ability to modify the generated image pairs in a way that is best for the learning of the model.

Figure 5 shows examples of how the synthetic images look between the first and the last epoch of the training. The final pairs selected for labeling also demonstrate subtler visual differences than the initial pairs, suggesting that our model has indeed learned to generate “harder” pairs.

Figure 7 compares these “harder” pairs to those from the real image pairs. Overall we see that the actively generated synthetic pairs tend to have fine-grained differences and/or offer visual diversity from the real training samples. The righthand side of Figure 7 shows examples of generated pairs rejected by annotators as illegible, which occurs 17% of the time. The relative low rate of rejection is an encouraging sign for making active query synthesis viable for image labeling tasks.

5. Conclusion

We introduced an approach for actively generating training image pairs that can benefit a fine-grained attribute ranker. Our approach lets the system think “outside of the pool” in annotation collection, imagining its own training samples and querying human annotators for their labels. On two difficult datasets we showed that the approach offers real payoff in accuracy for distinguishing subtle attribute differences, with consistent improvements over existing data augmentation techniques. In future work we plan to explore joint multi-attribute models for the ranker and consider how human-in-the-loop systems like ours might allow simultaneous refinement of the image generation scheme.

Acknowledgements We thank Qiang Liu, Bo Xiong, and Xinchen Yan for helpful discussions. This research is supported in part by ONR PECASE N00014-15-1-2291, NSF IIS-1514118, and a gift from Qualcomm.
References


[39] Buyue Qian, Xiang Wang, Fei Wang, Hongfei Li, Jieping Ye, and Ian Davidson. Active learning from relative queries. In IJCAI International Joint Conferences on Artificial Intelligence, 2013. 2


[63] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In ICCV, 2017. 2

