Semantic Jitter for Learning Binary Visual Attributes

Saket Sadani

Department of Computer Science

University of Texas at Austin

Supervisor

Kristen Grauman

May 25, 2019

Acknowledgements

I would like to sincerely thank Dr. Kristen Grauman and Aron Yu for their advice and mentorship throughout the course of this project, and over the course of my college experience as a whole.

I would also like to thank Dr. Qiang Liu, both for his compassion and the intellectual stimulus he provided.

Abstract

Solving the classification task can allow us to make meaningful predictions about what properties, or attributes, an image exhibits. It can allow us to determine whether a shoe is *fancy* or not, or if a face is *smiling*. With advances in machine learning, our ability to make such predictions has seen significant improvement over the years. However, most successful methods in solving this problem are supervised, meaning that they require labeled data. Depending on the domain, acquiring such data can be difficult. Thus, it is worthwhile to be able to construct more training samples from existing data. Prior work has been done to this effect, often in the form of low-level "jitter," which employs spatial and photometric changes to add variety to the original dataset.

In this work, we train classifiers to learn to predict binary attribute labels by augmenting our training data with new, "semantically" jittered images. To do so, we use an attribute-conditioned generative model to create label-preserving changes to existing data. We show that, in some cases, our proposed techniques result in higher classification accuracy than the baselines on the challenging UT-Zap50k and CelebA datasets. Furthermore, preliminary experiments show that in the case where the original dataset contains a class imbalance, our technique provides more substantial improvements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual attributes are useful for a variety of applications. For faces, work has been done to use them for identification [40, 41] and face verification [20], and for shoes, they could be used for online shopping and auto-tagging purposes. For instance, imagine we have a large selection of shoes we sell, and we want to allow a user to shop based on various factors, say whether they want something *formal*, or *casual*, or *sleek*. For large datasets, it would be either incredibly expensive to have humans exhaustively label all shoes with the all attributes they exhibit, or the labeling would be incomplete, resulting in sub-optimal user experience and profitability. In an analogous example for faces, consider the scenario that an anti-crime organization needs to narrow their suspect pool based on descriptions from witnesses. To be able to do so efficiently, they would likely want to search their faces database based on those descriptions such as *male* or *pointy nose*. This has similar trade-offs to our shoe shopping example - increased cost to label or the risk of missing potential suspects. However, we can avoid these trade offs in both cases if we instead train classifiers to perform this binary attribute prediction needed to tag the shoes or the faces.

This is easier said than done. Imagine that you are shown a shoe and asked if it is *formal* or not. As a human being, you have learned over time what this means. Maybe a parent said to "Put on your formal shoes" for a special event while pointing to some. This requirement for labeled examples extends to a machine learning classifier as well, be it an SVM or a neural network. They generally require many

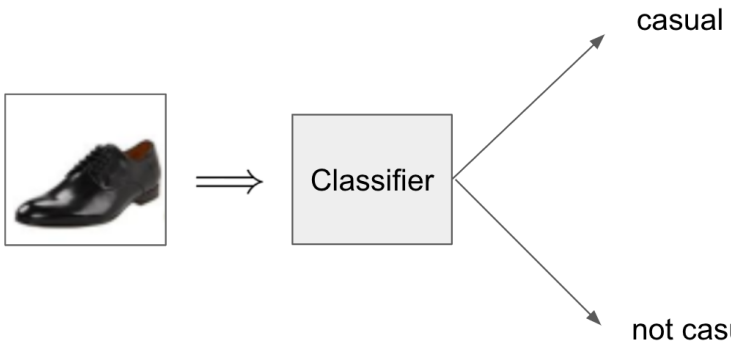
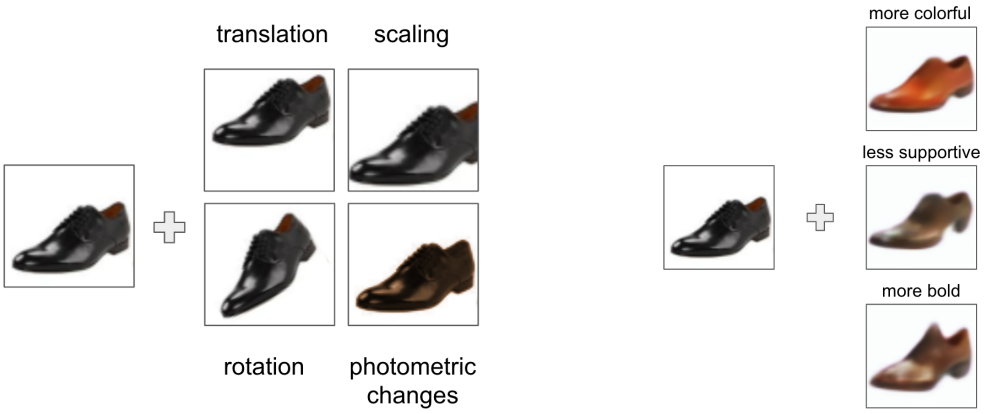


Figure 1.1: An example of a possible binary attribute classifier we may want to train. For training, the classifier sees images along with labels and during test time, simply an image with the goal of outputting the correct label.

more labeled training data because of the large number of parameters that need tuning in these models. Obtaining such amounts of data can be prohibitive, as it usually requires some human intervention to obtain the labels. Generally, current methods to deal with this need for data either construct massive datasets [39], explore a variety of network architectures and pre-training/fine-tuning strategies [21,22,39], or perform some basic data augmentation [24, 27, 42-45].

The main goal of this work is to generate more training images from existing ones, in order to augment the number of labeled images in the training dataset. At their core, images are simply a collection of pixels. However, images can also be described by the presence and strength of their attributes. The key insight here is that we can "semantically jitter" [18] these attributes to produce new, labeled images, after having learned an attribute-conditioned generative model. For instance, say we are training a classifier to determine if a shoe is *casual*, as in Figure 1.1. To generate more examples of comfortable shoes to feed to our classifier, we can attempt to perform some label-preserving changes to this seed image's attributes, say by making it more *colorful* or *bold*, and then using our generative model to produce an image such as in Figure 1.2b.

This proposed approach differs from traditional jittering methods that work at a lower level, performing spatial and photometric changes like rotations, translations, lighting, etc., such as those employed in [24, 27, 42-45] (see Figure 1.2a).



(a) (b)

Figure 1.2: Here we see the general idea behind data augmentation, in the context of "low-level jitter" (a) and "semantic jitter" (b).

These methods directly make pixel level changes to the base image, and have no higher-level knowledge in the form of attributes. Some prior work does utilize attribute-driven data augmentation for classification purposes, such as [32]. However, they use attributes to expand feature descriptors as opposed to generating more labeled images. In this work, we explore different techniques for jittering the attributes in an image, and augmenting the original training data with the newly generated images to train our binary attribute classifiers.

To measure our results, we compare the performance of this classifier against baseline classifiers trained on 1) all real images, and 2) real images augmented with low-level jitter images. We evaluate our methodology on two relevant datasets, the UTZap-50k shoes [7,18], and the CelebA faces [39]. These datasets contain images along with their attribute data. We discuss the details of these datasets in the Results section. As a whole, we see that our method on these datasets adds value in some cases for UT-Zap50k, although the results for CelebA are inconclusive. We do reasonably well on the UT-Zap50k dataset in the case of a class imbalance in the original training data.

Chapter 2

Related Works

In this chapter we discuss related works as they pertain to attributes, the relationship between attributes and image generation, attributes and classification, and data augmentation in general.

2.1 Attributes

Higher level properties of images, known as *attributes*, have shown promise in computer vision across a multitude of applications [3, 4, 13, 8, 14, 15, 9, 10]. They are particularly useful because they describe images in ways that are understandable by humans (e.g., a face "frowning," or a shoe being "sleek") but also by machines. Over the years, they have been used to guide image search [3], discern image "interestingness" [14], create more accurate description of images [4], and to perform transfer learning in the form of zero-shot learning [28]. These attributes come in multiple flavors. For example, *binary attributes* are semantic properties of an image which a human can decide the presence or absence of. Beyond that, the concept of *relative attributes* has also gained traction [5,6], with uses in making fine-grained comparisons between images [7]. Since these relative attributes add a comparative notion to otherwise binary information (e.g., this shoe looks "more sleek" than that one), they allow for improvements in image description [7] and image search [9,10], among other applications.

In addition to using attributes for the above tasks, work as also been done

in better learning the attributes themselves, starting with more traditional, non-neural network methods [11] to more recent CNN driven approaches [12]. However, in this work we focus on improving the quality of the learned attributes through our data augmentation, as opposed to specially crafted models. These two means of better learning attributes are orthogonal, and could likely be combined for better results.

2.2 Attributes and image generation

Prior work has been done on generating synthetic images, especially in recent years with the advent of Generative Adversarial Networks (GANs) [1] and Variational Auto-Encoders (VAEs) [2]. Conditional Variational Auto-Encoders (CVAEs) [16], which are conditioned on additional input data, further improve the generative capabilities of neural networks. Building on the CVAE work, Yan et al. created an attribute-conditioned generative model that produces qualitatively reasonable synthetic images [17]. This is the framework we employ in this work, similar to its usage in [18], which generates synthetic image pairs to "densify" supervision for learning to rank images by attribute strength.

Our work differs from [17] in that we use it to augment our dataset to improve classification accuracy, whereas they focus on the image reconstruction and completion tasks. They do not jitter the attributes, or generate new images, for these tasks. They do show, however, show successful qualitative results in image progression that motivates building on their work to perform dataset augmentation for other tasks. Our work also differs from [18], in both the task we are applying the jitter to, and the jitter technique. We are focused on classifying the presence or absence of an attribute, whereas Yu et al. aim to improve performance in ranking pair-wise attribute strength. For example, given two images of faces, they want to discern which of them is more *smiling*. Their approach is to generate more training data by taking an image from the original data set and generating from it an example that is *more smiling* and one that is *less smiling*. This way, they have another pair of images on which to train. On the other hand, since we are training a binary classifier, we jitter attribute values other than the

one we are classifying on, in order to make our changes label-preserving while still adding variety to the training data.

2.3 Attributes and classification

Attributes play an important role in image classification. For example, in [19], Escorcia et al. show that CNNs trained to do object recognition also implicitly learn information pertinent to attributes. They also reach the conclusion that said attributes play an important role in effectively recognizing objects. For example, in an example of zero-shot learning, Lampert et al. use pre-learned attributes to learn classifiers for previously unseen classes [29]. This differs from our work in that what we are training classifiers to predict the attributes themselves. To that end, there are works such as [21, 22, 39] which attempt classification directly on the attributes contained in the LFW datasets [23], and on the significantly larger CelebA dataset [39]. These works achieve good performance on this task through clever construction of network architectures and by training on the entire corpus these datasets provide (LFW contains ∼10k training data, and CelebA contains ∼160k). In contrast, our proposed method aims to achieve performance improvements by intelligently augmenting a significantly smaller dataset.

The motivation for generating synthetic images is as follows: classification in vision today generally means deep networks. Very successful examples of this are AlexNet [24], VGG [25] and ResNet [46]. However, all these networks require large amounts of labeled data to train. Overcoming this problem has generally been approached in two ways: 1) fine-tuning versions of these networks trained on other datasets [26] and 2) dataset augmentation, often in the form of low-level jitter [24, 27, 42-45] as defined in the introduction. We take advantage of fine-tuning a pre-trained AlexNext, but attempt to improve accuracy by augmenting our dataset with jitter that works at a higher level, namely at the level of attributes instead of spatial transformations such as scaling, rotation and translations.

2.4 Learning with limited data: dataset augmentation

Augmenting datasets, for classification or other tasks, is not a new idea. As mentioned in the previous section, there are simple methods like as low-level jitter, such as that employed by Dosovitisky and others [24, 27, 42-45], which change images via "low-level" changes like scaling, translation, rotation, etc. The authors of the original AlexNet paper [24] also performed dataset augmentation in order to avoid overfitting. They employed PCA to perform intensity transformations, along with image translations and reflections. Other works have applied more complex spatial transformations for label-preserving changes, for example [34]. With the concept of *meta-learning*, Wang et. al [30] teach a "hallucinator" what aspects of an image are important for a low-shot classification task, and use it to generate images in new poses based on that information. However, these methods do not take advantage of attributes.

Other dataset augmentation methods have been used for 3D pose and viewpoint estimation [36, 37]. In [36], Rogez and Schmid synthetically generate realistic images with 3D pose annotations using existing Motion Capture data. In [37], Su et. al. synthesize millions of additional training images by taking an online repository of 3D models and randomly sampling parameters to alter patterns from the seed images. In both cases, these synthetic images are used to train CNNs. Similarly, we create synthetic images from seed iamges to train our classifier, but we do not require 3D models or a graphics engine, so our method is applicable to a larger number of domains. In fact, it would be applicable to any dataset in which we have images and attributes information.

There are some works that do use attributes for the purpose of dataset augmentation. Dixit et al. approach this by using attributes to generate more feature descriptors to augment existing images [32]. However, unlike our approach, their augmentation does not actually add new images to the training set. Lastly, as discussed in detail in Section 2.2, the authors of [17] train a generative model conditioned on attributes, and [18] uses that framework for improving performance on pair-wise attribute ranking. Our work, though it uses the ideas of attribute-

conditioned image generation using a CVAE [17] and "semantic jittering" [18], differs in terms of the task we are tackling (binary attribute classification) and our jittering methodology. Furthermore, the proposed method relies on the data augmentation notion of label-preserving transformations, therefore not requiring any further supervision.

Chapter 3

Approach

Overview

Recall that we are attempting to improve classification of binary visual attribute labels within an image. Specifically, we aim to do so by augmenting our training data with new, labeled images that we construct. We would like to make "label-preserving" changes to our existing data, such that additional supervision is not needed. As such, our methodology takes the follow steps:

1. Train attribute-conditioned generative model (section 3.1)
2. For each attribute we intend to learn, construct a training set that we then jitter by adjusting the strengths of other attributes contained in the image, using the generative model trained (Section 3.2)
3. Train classifiers with the new data combined with the original, and compare with baselines. We also take steps to avoid the domain shift problem (Section 3.3)

Figure 3.1 provides a visual showing the general experimental setup.

3.1 Attribute-conditioned generative model

As discussed in the Related Works section, there are a number of methods used to generate synthetic images. Most of these involve a neural network of some

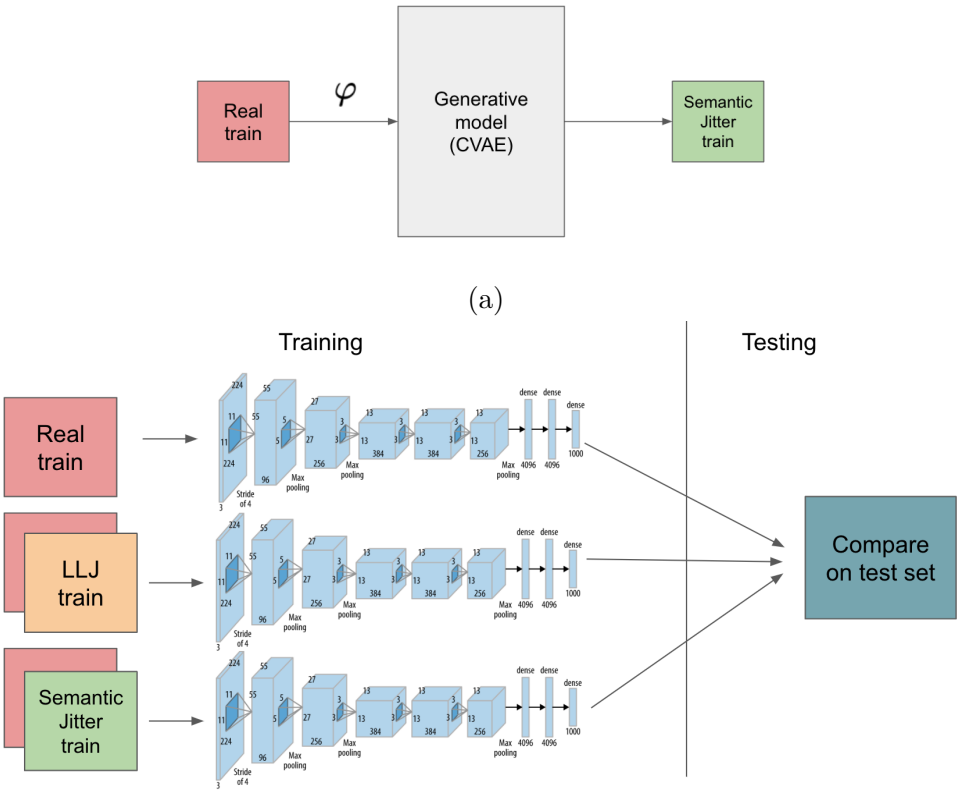
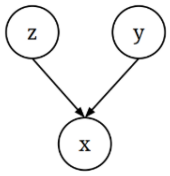


Figure 3.1: This figure summarizes the flow of data: In (a) we use the original training set of real images and their attributes to generate more using our Conditional Variational Autoencoder (CVAE) generative model. Then, in (b) we augment our original data with these synthetic images, and train a classifier to compare against the two baselines. Here, LLJ stands for the extra images produced via the "low-level jitter" mentioned above.



$p_\theta(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z})$

Figure 3.2: This provides a visual of the CVAE model assumption - the generative model for $\boldsymbol{x}$ depends on both $\boldsymbol{z}$ and attributes $\boldsymbol{y}$. This dependence on $\boldsymbol{y}$ allows us to change the resulting $\boldsymbol{x}$ in a controlled way.

form, recently of the GAN or VAE flavor. We use the CVAE generative model proposed by the authors of aptly named Attribute2Image paper [17] for two reasons: its generality and its success on a number of tasks. The authors of [18] used it to improve visual comparisons and the original paper successfully demonstrated attribute progression, meaning that we can intentionally jitter the images to produce variety in our training set.

Let $\boldsymbol{y} \in \mathbb{R}^{N_y}$ be a vector encoding the strength of each attribute $\{A_1, \ldots, A_{N_y}\}$ that the image exhibits. For example, in the case of shoes, it has real values describing how *sleek*, *formal*, *comfortable*, etc. the associated image is. Let $\boldsymbol{z} \in \mathbb{R}^{N_z}$ be the vector of latent variables, which provide a lower dimensional hidden representation of the image, accounting for factors such as pose, lighting, etc. not encoded in the attribute vector $\boldsymbol{y}$. Using the CVAE method proposed in [17], we learn a generative model $p_\theta(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z})$ such that we can produce synthetic images $\boldsymbol{x} \in \mathbb{R}^{N_x}$ following this distribution. This model p_θ is trained by indirectly maximizing the log-likelihood of $p_\theta(\boldsymbol{x}|\boldsymbol{y})$. In practice, this is actually done by maximizing the variational lower bound to make the problem tractable. Maximizing the variational lower bound requires us to introduce an auxiliary distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ to approximate the posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$. The network architecture consists of an CNN encoder, which takes as input an image $\boldsymbol{x}$ and associated attributes $\boldsymbol{y}$ and produces its latent $\boldsymbol{z}$ representation. The decoder, which takes in attributes $\boldsymbol{y}$ and latent variables $\boldsymbol{z}$, and produces an image $\boldsymbol{x}$. The conditioning of the image on the attributes is what allows us to generate new images by tweaking the $\boldsymbol{y}$ vector. See [17] for details.

3.2 Notation and image generation overview

Once the generative model is trained, we can use it to produce new images.

First, to train and measure the performance of our classifiers, we must define the following splits in our data. Note that such splits are analogously constructed for each attribute we want to classify (*comfortable, sleek, casual, smiling*, etc.), but for brevity we describe it in the case of a single attribute.

$$\mathcal{D}_{train}^{Real} = \left\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{z}^{(i)}, l^{(i)})\right\}_{i=1}^{N_{train}}$$
$$\mathcal{D}_{val} = \left\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{z}^{(i)}, l^{(i)})\right\}_{i=1}^{N_{val}}$$
$$\mathcal{D}_{test} = \left\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{z}^{(i)}, l^{(i)})\right\}_{i=1}^{N_{test}}$$

where $l^{(i)} \in \{0, 1\}$ represents the label for the ith image, i.e., the ith example is either *comfortable* or not. From $\mathcal{D}_{train}^{Real}$ we generate the low-level jittered images $\mathcal{D}_{train}^{LLJ}$, and using our CVAE, semantically jittered images $\mathcal{D}_{train}^{S}$.

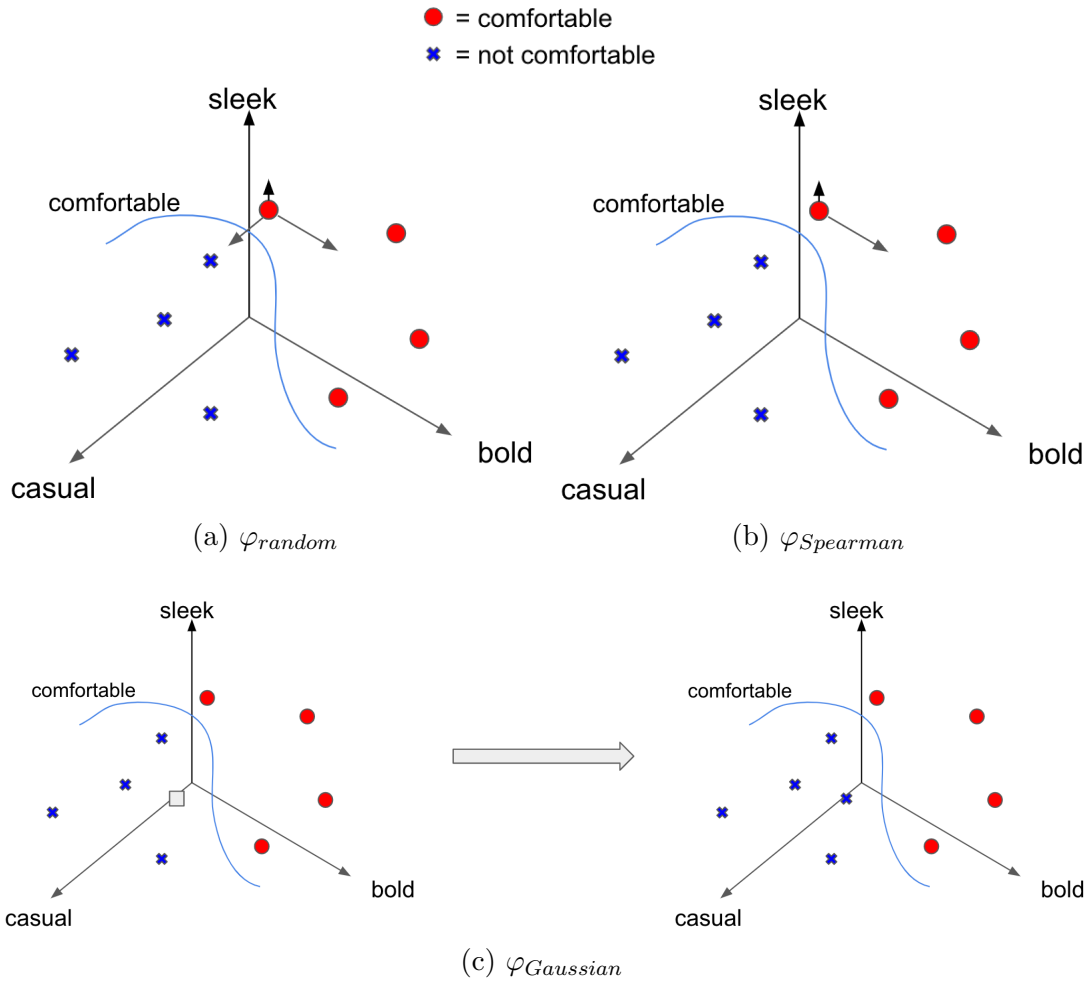
$\mathcal{D}_{train}^{LLJ}$ is generated using the methodology and parameters followed in [27], with $M \in \mathbb{N}$ jittered versions of each original image in $\mathcal{D}_{train}^{Real}$.

To construct $\mathcal{D}_{train}^{S}$, let us first define φ that performs the jittered image generation - it uses some deformed version of $\boldsymbol{y}$ and the latent $\boldsymbol{z}$ associated with a "seed" image to get a new image with, ideally, the same label as the original. The next section discusses the various φ functions we explore and specifics of jittering. For each original image in $\mathcal{D}_{train}^{Real}$, we invoke φ a total of M times. Combining all the resulting images for each original image, we get $\mathcal{D}_{train}^{S}$. This means that $|\mathcal{D}_{train}^{S}| = |\mathcal{D}_{train}^{LLJ}| = MN_{train}$.

3.3 Jitter techniques

In essence, jittering a "seed" image means sending its latent encoding $\boldsymbol{z}$ and attributes $\boldsymbol{y}$ through the decoder of our CVAE, after tweaking $\boldsymbol{y}$. Some important questions are: How many, and which, attributes should be jittered? By how much? How can we keep our changes label-preserving so that we can simply copy over the label of one of the original images?

To begin answering these questions, the following hyperparameters need to



(a) φ_{random} (b) $\varphi_{Spearman}$

(c) $\varphi_{Gaussian}$

Figure 3.3: Here we present the various techniques we use to create new attributes $\boldsymbol{y}$ for image generation. Imagine we are training a classifier to determine if a shoe is *comfortable*. The blue line represents the decision boundary. In (a), we jitter each image's attributes in a random fashion, but risk crossing the decision boundary (in other words, are not "label-preserving"). In (b) we limit which directions we can jitter in based on correlation (say if being *comfortable* is highly correlated with being *casual*). Finally, in (c) we sample a new point, and assign a label based on the label of the "nearest" point from the training set.

be defined: $k_1, k_2 \in \mathbb{Z}$. k_1 controls how much an individual attribute y_j, $j \in \{1, \ldots, N_y\}$ can be jittered, and k_2 controls how many such y_j are jittered for a given $\boldsymbol{y}$. Specifically, when constructing the new, jittered vector $\boldsymbol{y}'$ prior to sending it to the decoder, we apply the following transformation:

Make a copy $\boldsymbol{y}'$ of $\boldsymbol{y}$. For k_2 of the N_y individual attributes within $\boldsymbol{y}$,

$$y'_j \leftarrow y_j + k_1\epsilon_j \quad \text{for } \epsilon_j \sim \mathcal{N}(0, \sigma_j)$$

where all the σ_j are per-attribute value standard deviations computed from $\mathcal{D}^{Real}_{train}$. However, we still need to determine *which* attributes to jitter. Below, we describe the three techniques used, with a visual representation for each in Figure 3.3.

- φ_{random}: Let us say we are training a classifier for *comfortable*. For a seed image's $\boldsymbol{y}$, we need to determine which indices to tweak. The simplest choice for selecting which attributes to jitter is to choose k_2 random indices in $\boldsymbol{y}$, except the index corresponding to *comfortable*. This way, we are more likely to maintain the original label. However, there is no guarantee that attributes describing images in a dataset are independent from each other. Therefore, this method theoretically runs the risk of indirectly jittering the image in a way that is no longer label-preserving, as we may change an attribute that is highly correlated with the attribute we are learning to classify.

- $\varphi_{Spearman}$: This method aims to address the risk in the previous one by making changes that are more likely to be label-preserving. The idea is that we jitter those attributes that are least correlated with the classification attribute. The measure of correlation between attributes we use is Spearman correlation for ranked variables, which indicates how well the values of one attribute can be represented as a monotonic function of another. Specifically, let us say that our classification is being done on the kth attribute. In this case, we compute the Spearman ρ_{kj} correlation coefficient of attribute k with every $j \in \{1, \ldots, N_y\}$. Then, we jitter the k_2 attributes with lowest $|\rho_{ij}|$. By changing the values of unrelated attributes, we are less likely to alter the label from the original image.

- $\varphi_{Gaussian}$: One other method we explore is sampling $\boldsymbol{y}'$ from a multivariate Gaussian distribution. In this case, we are not jittering a seed image, but rather constructing a Gaussian distribution along all the $\boldsymbol{y}$ seen in the

training data, and using that to add more variety than manually tweaking a few attributes. Of course, by doing this, we have lost the ability to copy the label from a seed image, since technically there is none. Thus, we apply a nearest-neighbor heuristic to determine the label of the image that will be generated using this attribute vector, with the metric for determining "distance" being:

$$d(\boldsymbol{y}', \boldsymbol{y}) = \sum_{j=1}^{N_y} |\rho_{kj}|(y'_j - y_j)^2$$

where $\boldsymbol{y}'$ is the sampled vector, ρ_{kj} is the Spearman correlation between attributes k and j, and y_j is the value of the jth attribute in the vector. This distance measure penalizes differences in attributes that are highly correlated with the classification attribute. Consequently, the label l' of our newly generated image is

$$l' = l^{(*)} \quad \text{where} \quad * = \underset{i}{\operatorname{argmin}}\, d(\boldsymbol{y}', \boldsymbol{y}^{(i)}).$$

That is, we assign to our newly generated image the label $l^{(*)}$ of the example in $\mathcal{D}_{train}^{Real}$ whose $\boldsymbol{y}$ minimizes the above distance term. Other metrics could have been used, and may be candidates for exploration in future work. Lastly, though this method is not label-preserving in the same sense as the previous methods, it does not require additional supervision, which is the underlying goal.

For each of methods above, we pass the $\boldsymbol{z}$ of the seed image to the decoder, and in the case of the $\varphi_{Gaussian}$, the latent variables $\boldsymbol{z}^{(*)}$ of the image whose attributes we determine to be the closest match to the sampled $\boldsymbol{y}'$ as determined above. The $\boldsymbol{z}$ for each of the original data are computed in a pre-processing step after training the CVAE, using the general energy minimization for posterior inference proposed by [17].

3.4 Classification overview

Having generated synthetic images in a variety of ways, we would like to compare how classification accuracy improves when using these new data. As mentioned in prior sections, we have $\mathcal{D}_{train}^{Real}$, $\mathcal{D}_{train}^{LLJ}$, and $\mathcal{D}_{train}^{S}$. Note that for a given experiment, if $|\mathcal{D}_{train}^{Real}| = N$, then $|\mathcal{D}_{train}^{S}| = |\mathcal{D}_{train}^{LLJ}| = MN$, for choice of M. To measure the benefit of the dataset augmentation with both forms of jitter, we train three different classifiers for each attribute, with the training data being $\mathcal{D}_{train}^{Real}$, $\mathcal{D}_{train}^{Real} \bigcup \mathcal{D}_{train}^{LLJ}$, and $\mathcal{D}_{train}^{Real} \bigcup \mathcal{D}_{train}^{S}$. See Figure 3.1.

The actual classifier used is the AlexNet CNN [24]. We use one that has been pre-trained on the ImageNet dataset, and replace the fully connected layers and fine-tune those for classifying binary attribute labels. The network is trained using SGD with initial learning rate equal to 0.01, which is reduced over time. Furthermore, early stopping via the validation set $\mathcal{D}_{val}$ is employed to avoid overfitting. The important point is that the same treatment is applied to all sets of training data ($\mathcal{D}_{train}^{Real}$, $\mathcal{D}_{train}^{Real} \bigcup \mathcal{D}_{train}^{LLJ}$, and $\mathcal{D}_{train}^{Real} \bigcup \mathcal{D}_{train}^{S}$), and that we use traditionally accepted methods (AlexNet, fine-tuning, early stopping, dropout, etc.). Since we are exploring the effect of our dataset augmentation methods in a binary classification task, we use binary cross entropy as the loss function. Since the test sets contain a class-imbalance, we measure performance using

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{ precision }^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision } \cdot \text{ recall}}{\text{precision } + \text{ recall}}$$

instead of regular accuracy (to avoid seemingly high performance by simply learning to output the label of the majority class).

3.5 Addressing domain shift

An important point to note is that the images generated by the CVAE do not look the same as the original images (Figure 3.4). This domain shift is natural since the CVAE attempts to model an image based on a lower dimensional representation of latent variables and associated attributes. Therefore, the synthetic images are not expected to look like exactly like the originals. However, the classifier is more

likely to learn well if all the data is in the same domain. Thus, using the same approach, we shift all the original data (training, validation, and test sets) into this "reconstructed" domain. To do so, we simply take each real image $\boldsymbol{x}$, and send its associated $\boldsymbol{y}$ and $\boldsymbol{z}$ thru the decoder to a get the reconstructed image $\boldsymbol{x}'$. Note that these images are not jittered in the sense that $\boldsymbol{y}$ was passed for reconstruction without any modification.

It is reasonable to run our experiment on the reconstructed images because we can also shift the test set to this domain. As as long as the classifier learns to output the correct label (irrespective of the domain of the input images), the numerical results of classification on either domain are comparable. As such, we run our classification experiments in both domains. Initially, we leave the $\mathcal{D}_{train}^{Real}$ dataset untouched and simply augment it with our synthetic images. In an alternative experiment, to alleviate the issues discussed above, we shift $\mathcal{D}_{train}^{Real}$ (and $\mathcal{D}_{train}^{LLJ}$) to the reconstructed domain prior to augmenting with the semantically jittered images.

Note that we could employ domain adaptation techniques such as those discussed in [47, 48] to overcome the domain gap, but we leave this as future work.

(a) (b)

(c) (d)

Figure 3.4: Images with original images from the shoes and faces datasets, along with their reconstructed versions to demonstrate the "domain shift." It would be better, when training a classifier, that the data all be in the same domain.

Chapter 4

Results

In Section 4.1 we discuss the two datasets that we experiment on. Section 4.2 discusses the initials results in the original domain, while Section 4.3 discusses the results after ensuring all data is in the same domain as described in Approach. In Section 4.4 we introduce a scenario in which the data augmentation techniques presented better outperform the baselines for the first dataset. Section 4.5 discusses some qualitative results and provides some commentary on how they relate to the numerical ones.

4.1 Datasets

Below we discuss the details of the two datasets on which we experiment, the UT-Zap50K [7,18] shoes dataset and the CelebA [39] faces dataset. The method was initially developed on the UT-Zap50K dataset.

4.1.1 UT-Zap50K shoes dataset

The first dataset, UT-Zap50k, contains 50,025 images of shoes and attribute values describing the strength of the following attributes exhibited by each image: *Comfortable, Casual, Simple, Sporty, Colorful, Durable, Supportive, Bold, Sleek, Open* [7]. This dataset was initially created from an online shoe catalog and curated for the purpose of improving fine-grained comparisons between relative attributes. Prior work took the labeled pairs in the dataset and learned a ranking

function to get the $\boldsymbol{y}$'s [18]. Thus, the attribute values $\boldsymbol{y}$ essentially represent the degree to which an image does or does not exhibit an attribute. However, by thresholding these relative attribute values, we can get binary attributes. These serve as the labels that we want to learn to predict. The images $\boldsymbol{x}$ across the entire dataset are all centered and placed on a white background for consistency (Figure 4.1).

A subset of ∼13k images were used to train the generative CVAE model. We used small subsets of size N of the the remaining ∼37k to fine-tune a pre-trained AlexNet for classifying on each of the different attributes mentioned above. In order to illustrate the improvement the dataset augmentation provided over the baselines, the N were chosen to be fairly small. For this dataset, we performed two kinds of experiments. The first, discussed in Section 4.2, the training data were made to contain an equal number of positive and negative examples. For instance, if we were training a classifier to determine whether a shoe is *sporty* or not, then there would be $\frac{N}{2}$ sporty shoes and $\frac{N}{2}$ non-sporty shoes in the original dataset used. In the second experiment, we explore the ability of our method to adaptively generate examples of the deficient class. In this case, the data splits were intentionally kept unbalanced (still randomized, however).

4.1.2 CelebA faces dataset

The CelebA dataset is substantially bigger than the shoes dataset, with 202,599 images and 40 binary attribute annotations. These binary attributes serve as the labels when doing classification on said attributes, such as *Arched Eyebrows, Attractive, Bald, Bangs, Black Hair, Blond Hair, Blurry, Brown Hair, Bushy Eyebrows, Chubby, Double Chin, Eyeglasses, Goatee, Gray Hair, High Cheekbones, Male*. The CVAE is trained the same way as the UT-Zap50k dataset, using the methods of [17]. The attribute data that comes with CelebA are binary attribute values ±1 indicating the presence or absence of an attribute. Thus we do not need to threshold for our training/testing labels. However, we do not learn a ranking function for the $\boldsymbol{y}$'s in this case, but treat the binary labels of ±1 as real numbers and jitter those using the methods described in Approach. The images $\boldsymbol{x}$ of the faces are centered and somewhat aligned, though there are still

Figure 4.1: Examples of images from the UT-Zap50k dataset, with positive and negative examples of some of the attributes we aim to predict.

		Casual	Comfort	Simple	Sporty	Colorful	Durable	Supportive	Bold	Sleek	Open	Average
	Real	0.897	0.943	0.763	0.873	0.763	0.902	0.835	**0.757**	0.882	0.852	0.847
	+Low Level Jitter	**0.900**	0.935	**0.766**	**0.876**	**0.768**	0.907	**0.846**	0.752	0.873	0.853	0.848
Ours	+Semantic Jitter - Random	0.899	**0.945**	0.755	0.875	0.760	**0.919**	**0.846**	0.749	**0.894**	0.855	**0.850**
	+Semantic Jitter - Spearman	**0.900**	**0.945**	0.755	0.875	0.754	**0.920**	**0.846**	0.747	0.893	**0.856**	0.849
	+Semantic Jitter - Gaussian	0.893	0.933	0.743	0.861	0.766	0.895	0.832	0.757	0.877	0.844	0.840

Table 4.1: UT-Zap50K original results (N=50): Real, Real + Low Level Jitter, and averages of each of our proposed methods. Numbers displayed are F1 scores, with standard error on the order of 0.001.

some variations in pose, lighting, etc. See Figure 4.2 for examples.

CelebA is an extension upon the CelebFaces dataset [38], to which the authors of [39] added the binary attribute labels. This dataset was then used for the purpose of large scale attribute classification. Liu et al. used the entire training set of ∼160k images to get state of the art performance of this task. The need for such large amounts of training data further motivates generating synthetic images to augment smaller datasets.

Empirically, we found that the training sizes N needed for this dataset are much larger than for the UT-Zap50k classification task, at least on the pre-trained AlexNet that we are fine-tuning. As in the case for UT-Zap50k, for the primary experiment, the randomized training data were made to contain an equal number of positive and negative examples. Currently, we have not explored the class imbalance case for CelebA faces.

4.2 Initial results

The results shown in Tables 4.1 and 4.2 are for classifiers trained on the following datasets (for methodology, refer to Approach). As mentioned above, the same number of positive and negative examples were in the initial dataset $\mathcal{D}_{train}^{Real}$:

1. **Real (Baseline)**: All real images.

2. **+Low Level Jitter (Baseline)**: The real images from **Real**, plus the ones produced via low level jitter.

3. **+Semantic Jitter - Random**: Real images plus images generated by random perturbations of attribute vectors.

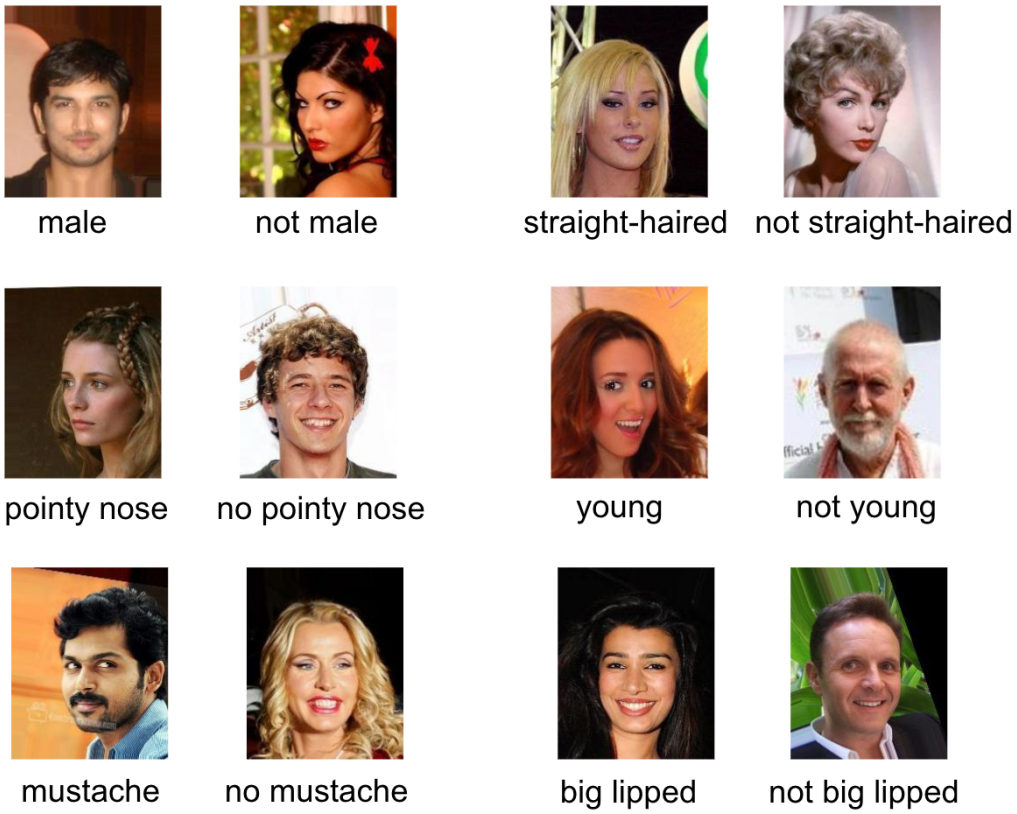


Figure 4.2: Examples of images from the CelebA dataset, with positive and negative examples of some of the attributes we aim to predict.

	5 o Clock Shadow	Arched Eyebrows	Attractive	Bags Under Eyes	Bald	Bangs	Big Lips	Big Nose	Black Hair	Blond Hair	Blurry	Brown Hair	Bushy Eyebrows	Chubby	Double Chin	Eyeglasses	Goatee	Gray Hair	Heavy Makeup	High Cheekbones	Male
Real	43	62	76	52	44	**76**	46	51	**73**	70	**24**	53	48	34	**37**	**72**	36	37	84	78	91
+Low Level Jitter	**46**	**63**	76	**54**	44	73	51	51	71	70	23	53	43	**36**	35	70	39	37	83	**81**	**92**
+Semantic Jitter - Random	45	62	76	52	44	73	**52**	50	70	72	20	53	**49**	30	33	68	39	39	83	**81**	89
+Semantic Jitter - Spearman	**46**	61	76	51	44	73	50	51	71	72	20	53	**49**	31	32	68	38	37	82	80	89
+Semantic Jitter - Gaussian	42	60	76	51	42	64	51	50	68	**74**	18	49	42	33	28	56	**43**	35	84	81	88

	Mouth Slightly Open	Mustache	Narrow Eyes	No Beard	Oval Face	Pale Skin	Pointy Nose	Receding Hairline	Rosy Cheeks	Sideburns	Smiling	Straight Hair	Wavy Hair	Wearing Earrings	Wearing Hat	Wearing Lipstick	Wearing Necklace	Wearing Necktie	Young		Average
Real	86	35	**44**	**94**	**50**	41	**51**	**47**	40	40	88	39	**67**	54	66	91	38	55	82		57
+Low Level Jitter	**87**	35	38	92	44	**46**	**51**	43	**41**	39	88	**40**	**67**	**55**	66	**92**	37	55	83		**58**
+Semantic Jitter - Random	86	34	39	92	46	45	50	44	**41**	**42**	88	**40**	66	54	66	90	37	**61**	**88**		57
+Semantic Jitter - Spearman	86	35	39	90	46	**46**	49	45	40	**41**	87	39	66	51	66	90	38	58	86		56
+Semantic Jitter - Gaussian	79	31	39	89	**50**	42	49	46	40	40	87	35	65	52	65	91	37	50	86		55

Table 4.2: CelebA original results (N=7500): Real, Real + LLJ, and averages of our proposed methods. Note: Numbers displayed are F1 score * 100 (space constraints because displaying results on 40 attributes). Standard error on average was around 1.5, indicating that that may not really be any improvement using these methods.

4. **+Semantic Jitter - Spearman**: Real images plus images generated by similar perturbations, except on uncorrelated attributes.

5. **+Semantic Jitter - Gaussian**: Real images plus those generated by multivariate Gaussian sampling of the attribute vectors.

As we can see from Tables 4.1 and 4.2, the results with the dataset augmentation techniques listed above do not offer much improvement in general. For the UT-Zap50K, on average Random Jitter shows a minor increase over the baselines. The numbers shown are averages taken over five different runs, with a standard error on the order of 0.001. With this in mind, the attributes *durable* and *sleek* show substantial improvements over the baselines, potentially because they are less correlated with the other attributes, making changes more label-preserving. However, it is not entirely clear why these particular attributes do well. On average, the improvement is not significant for any of the augmentation techniques.

For the CelebA dataset, on average low-level jitter actually does a little bit better, but there 7/40 attributes for which our experimental classifiers performed

		Casual	Comfort	Simple	Sporty	Colorful	Durable	Supportive	Bold	Sleek	Open	Average
	Real (original)	0.897	0.943	0.763	0.873	0.763	0.902	0.835	0.757	0.882	0.852	0.847
	+Low Level Jitter (original)	0.900	0.935	0.766	0.876	0.768	0.907	0.846	0.752	0.873	0.853	0.848
	Real (reconstructed)	0.889	0.940	0.725	0.870	0.732	0.905	0.851	0.730	0.874	0.883	0.840
	+Low Level Jitter (reconstructed)	0.896	0.944	**0.738**	0.878	0.743	0.913	0.854	0.725	0.888	0.884	0.846
Ours	+Semantic Jitter - Random	**0.897**	0.943	0.732	0.881	**0.752**	**0.930**	0.860	**0.734**	**0.897**	**0.891**	**0.852**
	+Semantic Jitter - Spearman	0.897	**0.945**	0.732	**0.882**	0.748	0.925	**0.865**	0.726	0.894	0.888	0.850
	+Semantic Jitter - Gaussian	0.890	0.925	0.731	0.848	0.754	0.886	0.839	0.738	0.862	0.878	0.835

Table 4.3: UT-Zap50K reconstructed results (N=50): Real, Real + LLJ, and averages of each of our proposed methods. As before, numbers are F1 scores, with standard error once again on the order of 0.001. The bold values show the best results for all classifiers in the *reconstructed space*, indicating that we generally beat the reconstructed baselines. It is, however, also important to compare with the baselines in the original space (above the double lines).

slightly better than the baselines. However, these numbers are the averages across only three trials and the standard error is $\sim$1.5 on average, making the improvements over baselines a little less meaningful. However, the quality of the jitter and the synthetic images may not be the only culprit. It is odd that low-level jitter provides such little improvement over the Real classifier. We speculate that this is due to a sub-optimal experimental design. Using the pre-trained AlexNet as we did in the case of UT-Zap50K may not have been ideal - state-of-the-art classifiers on face datasets [39] use cascading neural networks to first do face detection and then classification, and are pre-trained on faces to begin with. Since faces are more uniform than the thousands of classes in ImageNet, a network pre-trained on faces would give us more conclusive results than the current setup with an AlexNet pre-trained on the ImageNet dataset.

However, there is another likely reason for limited improvements of our method over the baselines. Recall from Approach (Figure 3.4) that the synthetically generated images don't look quite like the real ones. In other words, the real images and the synthetic ones are not quite in the same domain. We hypothesize that this affects the training of the classifiers in a negative manner and in the next section, we present results with all data in the same domain.

4.3 Domain shift avoidance

With all training data in same domain, for the Zap50k dataset, the synthetic images do add some value. We can see this in Table 4.3, as one of the jitter techniques (generally Random Jitter) outperforms both *reconstructed* baselines

	5 o Clock Shadow	Arched Eyebrows	Attractive	Bags Under Eyes	Bald	Bangs	Big Lips	Big Nose	Black Hair	Blond Hair	Blurry	Brown Hair	Bushy Eyebrows	Chubby	Double Chin	Eyeglasses	Goatee	Gray Hair	Heavy Makeup	High Cheekbones	Male
Real	43	64	76	51	**37**	**68**	37	49	**70**	41	11	45	47	46	42	**87**	47	46	**87**	84	90
+Low Level Jitter	**47**	64	76	**54**	35	65	**40**	50	68	41	22	46	48	**48**	**52**	84	57	46	82	**85**	91
+Semantic Jitter - Random	**47**	64	76	50	**37**	66	**40**	50	67	**46**	15	**47**	**49**	**48**	**52**	84	57	46	**87**	**85**	89
+Semantic Jitter - Spearman	43	65	72	50	36	65	36	50	68	42	11	45	48	**48**	37	84	**58**	43	85	74	**92**
+Semantic Jitter - Gaussian	42	**66**	70	48	32	39	31	42	65	45	11	40	40	43	36	73	40	42	76	72	86

	Mouth Slightly Open	Mustache	Narrow Eyes	No Beard	Oval Face	Pale Skin	Pointy Nose	Receding Hairline	Rosy Cheeks	Sideburns	Smiling	Straight Hair	Wavy Hair	Wearing Earrings	Wearing Hat	Wearing Lipstick	Wearing Necklace	Wearing Necktie	Young		Average
Real	**87**	34	**39**	**95**	**50**	39	50	**43**	**59**	**57**	91	27	**64**	**54**	52	**93**	**39**	71	81		57
+Low Level Jitter	**87**	36	34	92	50	41	52	40	55	**57**	91	33	62	**54**	52	91	38	**74**	**88**		**58**
+Semantic Jitter - Random	86	**39**	37	**95**	48	41	**53**	41	55	52	91	**34**	60	53	52	83	38	**74**	**88**		**58**
+Semantic Jitter - Spearman	86	30	24	93	40	41	51	40	57	48	90	28	56	52	50	74	37	69	78		55
+Semantic Jitter - Gaussian	88	29	24	92	45	36	48	31	56	44	92	18	55	40	51	74	35	72	89		51

Table 4.4: CelebA reconstructed results: Real, Real + LLJ, random jitter, spearman jitter , and gaussian sampling. Note: Numbers displayed are rounded F1 score * 100 (space constraints because displaying results on 40 attributes). Standard error on average was around 1.5, indicating once again that that may not really be any improvement using these methods. However, this may be the result of improper experimental design (see Section 4.2).

for 9/10 attributes. Of course, in some cases the margin of improvement is very small (i.e. *comfort*). On average, though, it performs noticeably better than the reconstructed baselines, with double the improvement over low-level jitter. For the attributes on which it did well before, it continues to perform well (*sleek* and *durable*), but with a larger margin.

Note that the Real baseline trained on the original images generally performs better than the Real baseline trained on the reconstructed images. This is to be expected as the reconstructed image is a lossy version of the original. However, this leads to an important point: outperforming the baselines in the reconstructed space is not sufficient - we have to show improvement over the baselines in the original domain, because otherwise there was no benefit to transforming the original, supervised data to the reconstructed space and adding more data in that domain. When comparing our experimental, augmented classifiers against the original domain baselines, the additional value of the augmentation is less apparent. We outperform the original domain baselines on 6/10 attributes, and on average, the improvement is of lesser magnitude.

	Casual	Comfort	Simple	Sporty	Colorful	Durable	Supportive	Bold	Sleek	Open	Average
Real (original)	0.868	0.929	0.773	0.820	0.660	0.857	0.750	0.717	0.791	0.834	0.800
+Low Level Jitter (original)	0.889	0.935	**0.781**	0.829	**0.722**	0.875	**0.833**	**0.721**	0.793	0.844	0.822
Real (reconstructed)	0.882	0.924	0.723	0.813	0.691	0.871	0.796	0.672	0.807	0.848	0.803
+Low Level Jitter (reconstructed)	0.893	0.927	0.736	0.827	0.691	0.886	0.813	0.701	0.813	0.842	0.812
+Semantic Jitter - Random	**0.907**	**0.941**	0.738	**0.851**	0.714	**0.892**	0.819	0.717	**0.824**	**0.867**	**0.827**

Table 4.5: UT-Zap50K results on imbalanced dataset of size N=50, with jitter used to balance out dataset. As before, after repeating experiments five times, std. error is on the order of 0.001.

For CelebA, Table 4.4 shows that Random version of our techniques performs roughly as well as low-level jitter on average, but not better. Compared to the 7/40 from Table 4.2, we have that for 9/40 of the attributes in Table 4.4 one of our jitter techniques performs as well or better than the baselines. However, the improvements are once again not very significant and are inconclusive, for reasons discussed in Section 4.2.

4.4 Addressing dataset class imbalance

Recall that for the results shown in section 4.2, $\mathcal{D}_{train}^{Real}$ was balanced in terms of the number of positive and negative examples. It is conceivable, however, that a given dataset is small and not balanced in this sense. In such a scenario, it would be valuable to be able to adaptively generate more examples of the deficient class to boost performance.

To test this, we construct training sets $\mathcal{D}_{train}^{Real}$ that are 80% negative examples and 20% positive. Then, we train the all Real baselines on it as before. For the jitter techniques, though, we create 80% of the synthetic images using the positive images as a seed, and vice versa for the negative class. This effectively yields $\mathcal{D}_{train}^{LLJ}$ and $\mathcal{D}_{train}^{S}$ such that when combined with the initial dataset, there are 50% positive and negative examples.

For the experiments whose results are shown in Table 4.5, we learn from the previous experiments and train our experimental classifiers after shifting the original data to the reconstructed space. As before, the performance gain is only significant if this new classifier performs better than our baselines trained on images in the original domain as well. Once again, it does so for 6/10 of the attributes. We do substantially better than the original domain classifier trained only on the real images. However, on average, we do not beat the original domain

low-level jitter baseline by much, as it also benefits from addressing the class imbalance. In the reconstructed space, we beat both baselines by a substantial margin. Of course, the results of our data augmentation are not better than the results from training a classifier on real images in the original domain from a *balanced* $\mathcal{D}_{train}^{Real}$.

At present, we do not have corresponding results for CelebA, but the results on UT-Zap50k suggest it may be worth exploring. However, to get conclusive, meaningful results, the method should be explored after adopting an experimental setup that is more appropriate for the faces dataset.

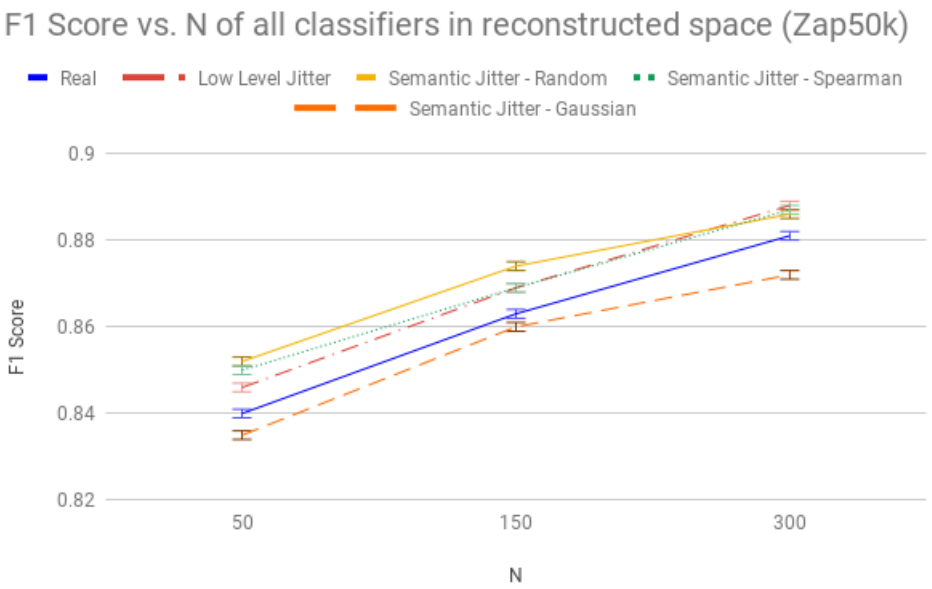
4.5 Effect of increasing N

Recall that N is the amount of data in the original training set $\mathcal{D}_{train}^{Real}$. It is interesting to consider the effects of our data augmentation as this N increases. Of course, increasing N improves all classifiers' performance. The expectation in the scenarios where we outperform baselines is that as N increases, the difference between the baseline classifiers and the experimental ones should decrease, i.e. the marginal benefit of the synthetically generated data decreases for increasing N. Not all results follow this trend, as the synthetic images often add little to no performance improvement. As an example, Figure 4.3 shows this phenomenon on both datasets.

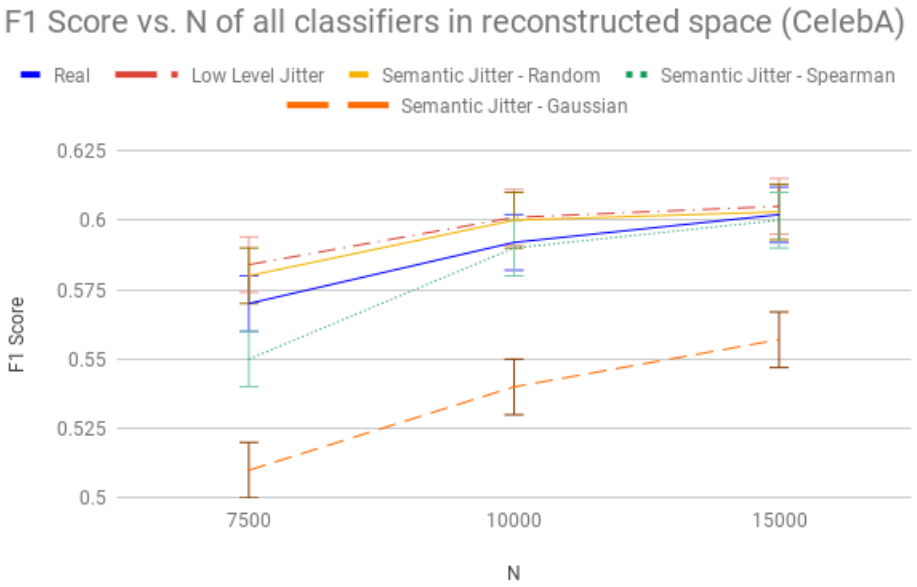
This phenomenon makes intuitive sense for the following reason: As the number of real training images increases, using synthetically generated images, which are by nature imperfect, matters less. The CNN can learn from the original data which is assumed to be labeled correctly. The tricks we play to get labeled synthetic images have room for error in both the image quality and their associated label. Thus, having more real data with human supervision is understandably more valuable.

Once again, the overlapping error bars in Figure 4.3b shows us that we cannot make any conclusions about the results on CelebA as the standard error on our trials is too large. The results on UT-Zap50k are more consistent, and in the reconstructed space (where our classifiers offer more improvement), we see value

added at first, but less so for larger N.



(a) Performance of all classifiers across N on the UT-Zap50k dataset, for average across all attributes.



(b) Performance of all classifiers across N on the CelebA dataset, for average across all attributes.

Figure 4.3: Plots demonstrating trend of increasing N on performance. Data points taken from results run in the reconstructed space. In cases where experimental beats baselines, margin of improvement decreases as N increase.

4.6 Qualitative results

We see from Attribute2Image [17] that we can intentionally change the image visually along a given attribute. Figure 4.4 shows a number of jittered examples that preserve the original label we are classifying on. Alongside our semantically jittered images, we can see the original and a low-level jittered version. Though our version typically looks blurrier than the original image and low-level jitter, it still adds variety along jittered attributes while maintaining the label. In fact, it adds changes that low-level jitter cannot. For instance, in Figure 4.4c, low-level jitter scales up the image, whereas we are able to change some structure of the shoe by making it more "open." Furthermore, by jittering an attribute like "colorful," we can also add changes similar to low-level jitter (Figure 4.4b).

However, this begs the following question: if we can do things that low-level jitter cannot, why do we not beat the original domain low-level jitter baseline by much, on average? Qualitative results provide two primary reasons. First, the generative model does not always produce appropriate images. Figure 4.5 shows that sometimes the images end up disfigured or do not preserve the original label. Figure 4.6 shows that this is especially the case for our Gaussian sampling method. These issues have an effect on the classifier trained, as can be seen from the poor performance of the Gaussian sampling method in the tables above. Second, and perhaps more importantly (as this effects all images produced, not just the disfigured ones), the images produced are not as crisp and they contain less detail. It may be unreasonable to expect this to beat classifiers trained on sharper and more detailed images. Despite the lossy nature of these images, however, we beat the baselines in reconstructed space. This motivates the following claim: since our jitter does seem to add useful variety, if we are able to produce sharper, semantically jittered images, we may be able to consistently outperform low-level jitter in either domain.

As a whole, the qualitative results motivate the validity of our method to add variety to the training data, but also indicate that there is room for improvement if we want to consistently beat all baselines.

(a) Original label: **not durable**. Low level jitter: rotation. Semantic jitter: ↑ simple, ↑ colorful.

(b) Original label: **casual**. Low level jitter: color changes. Semantic jitter: ↓ open, ↑ colorful.

(c) Original label: **not sporty**. Low level jitter: scaled up and slight color change. Semantic jitter: ↑ open.

(d) Original label: **not sleek**. Low level jitter: shrinked and slight color change. Semantic jitter: ↓ casual, ↓ supportive.

(e) Original label: **not young**. Low level jitter: scaled up and color changes. Semantic jitter: ↑ black hair, ↑ smiling.

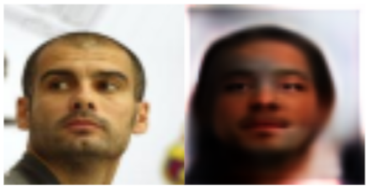
(f) Original label: **no bangs**. Low level jitter: translation. Semantic jitter: ↓ arched eyebrows, ↓ bags under eyes.

(g) Original label: **beard**. Low level jitter: scaled up and slight color change. Semantic jitter: ↑ oval face, attractive.

(h) Original label: **straight hair**. Low level jitter: rotationd and color change. Semantic jitter: ↓ big nose, ↓ male.

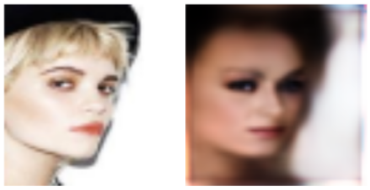
Figure 4.4: Sample triplets from both datasets with {original image, a low-level jittered version, a semantically jittered version (random attribute selection)}. In each case, we see that, visually, the jittered images have the same label as the original image. We cannot claim that our method is strictly better than low-level jitter - the image variety is of a different type.

(a) Jittered image somewhat mangled



(b) Jitter did not preserve label: **balding**

(c) Jittered image somewhat mangled



(d) Jitter did not preserve label: **blond hair**

(e) Jittered image totally mangled

(f) Jitter did not preserve label: **beard**

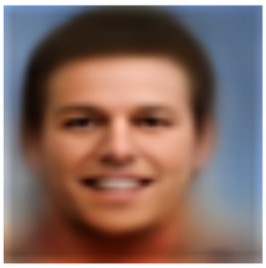
Figure 4.5: Samples from both datasets showing cases where our image generation did not work well. We see cases where the generated image is mangled, or the jitter was not label-preserving.

(a) Correctly labeled as **not casual**

(b) Incorrectly labeled as **casual**

(c) Correctly labeled as **open**

(d) Incorrectly labeled as **not open**



(e) Correctly labeled as **male**

(f) Incorrectly labeled as **not male**

(g) Correctly labeled as **wearing eyeglasses**

(h) Incorrectly labeled as **wearing eyeglasses**

Figure 4.6: Samples from both datasets showing cases where our Gaussian sampling method produces jittered images with correct and incorrect labels. The prevalence of incorrectly labeled images indicates a better labeling scheme needs to be applied.

Chapter 5

Conclusions and Future Work

As a whole, results on the second dataset (CelebA) were inconclusive. This may be in part due to suboptimal experimental setup - AlexNet may not have been the appropriate architecture for the task. An immediate next step would be to provide a more rigorous examination of our data augmentation technique on CelebA using more a more tailored experimental setup, such as using a ResNetpre-trained on faces.

With the series of experiments undertaken, results on Zap50k show some promise. While keeping the real data in the original domain we noticed that the synthetic images only helped in a select few cases. With the domain shift trick, we were able to increase the value of data augmentation using our semantic jitter, and in some cases it showed substantial improvement over all baselines. Lastly, when using this technique to address dataset class imbalance, we had similar positive results.

Of course, our techniques are most useful when there is a limited amount of real data - our synthetic images are not a replacement for real images, but in the absence of enough, we can boost performance by creating some more. As the amount of real data increases, the marginal utility of this data augmentation decreases.

While we have shown instances where our approach has merit, it is not valuable in all cases, and often offers little improvement over low-level jitter. Based on the qualitative results, we hypothesize that our jitter does add meaningful variety, but to beat all baselines we likely need higher quality generative models.

One avenue for future work to achieve this would be to train the CVAE to learn a larger dimensional latent encoding, $\boldsymbol{z}$. We could also try using a conditional GAN instead and perform similar jittering.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[3] N. Kumar, P. Belhumeur, and S. Nayar, "Facetracer: A search engine for large collections of images with faces," in *European conference on computer vision*, pp. 340–353, Springer, 2008.

[4] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1778–1785, IEEE, 2009.

[5] D. Parikh and K. Grauman, "Relative attributes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 503–510, IEEE, 2011.

[6] Y. Souri, E. Noury, and E. Adeli, "Deep relative attributes," in *Asian Conference on Computer Vision*, pp. 118–133, Springer, 2016.

[7] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 192–199, 2014.

[8] A. Sadovnik, A. Gallagher, D. Parikh, and T. Chen, "Spoken attributes: Mixing binary and relative attributes to say the right thing," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2160–2167, 2013.

[9] A. Kovashka, D. Parikh, and K. Grauman, "Whittlesearch: Image search with relative attribute feedback," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2973–2980, IEEE, 2012.

[10] A. Kovashka, D. Parikh, and K. Grauman, "Whittlesearch: Interactive image search with relative attribute feedback," *International Journal of Computer Vision*, vol. 115, no. 2, pp. 185–210, 2015.

[11] V. Ferrari and A. Zisserman, "Learning visual attributes," in *Advances in neural information processing systems*, pp. 433–440, 2008.

[12] C. Huang, C. Change Loy, and X. Tang, "Unsupervised learning of discriminative attributes and visual representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5175–5184, 2016.

[13] M. Mitchell, K. Van Deemter, and E. Reiter, "Generating expressions that refer to visible objects," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics (ACL), 2013.

[14] S. Dhar, V. Ordonez, and T. L. Berg, "High level describable attributes for predicting aesthetics and interestingness," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1657–1664, IEEE, 2011.

[15] B. Siddiquie, R. S. Feris, and L. S. Davis, "Image ranking and retrieval based on multi-attribute queries," 2011.

[16] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, pp. 3483–3491, 2015.

[17] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*, pp. 776–791, Springer, 2016.

[18] A. Yu and K. Grauman, "Semantic jitter: Dense supervision for visual comparisons via synthetic images," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5570–5579, 2017.

[19] V. Escorcia, J. Carlos Niebles, and B. Ghanem, "On the relationship between visual attributes and convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1256–1264, 2015.

[20] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 365–372, IEEE, 2009.

[21] N. Zhuang, Y. Yan, S. Chen, H. Wang, and C. Shen, "Multi-label learning based deep transfer neural network for facial attribute classification," *Pattern Recognition*, vol. 80, pp. 225–240, 2018.

[22] S. Kang, D. Lee, and C. D. Yoo, "Face attribute classification using attribute-aware correlation map and gated convolutional neural networks," in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 4922–4926, IEEE, 2015.

[23] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014.

[27] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1734–1747, 2016.

[28] D. Jayaraman and K. Grauman, "Zero-shot recognition with unreliable attributes," in *Advances in neural information processing systems*, pp. 3464–3472, 2014.

[29] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014.

[30] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7278–7286, 2018.

[31] E. G. Miller, N. E. Matsakis, and P. A. Viola, "Learning from one example through shared densities on transforms," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, vol. 1, pp. 464–471, IEEE, 2000.

[32] M. Dixit, R. Kwitt, M. Niethammer, and N. Vasconcelos, "Aga: Attribute-guided augmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7455–7463, 2017.

[33] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[34] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. Fisher, and L. Hansen, "Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation," in *Artificial Intelligence and Statistics*, pp. 342–350, 2016.

[35] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, pp. 2234–2242, 2016.

[36] G. Rogez and C. Schmid, "Mocap-guided data augmentation for 3d pose estimation in the wild," in *Advances in neural information processing systems*, pp. 3108–3116, 2016.

[37] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2686–2694, 2015.

[38] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, pp. 1988–1996, 2014.

[39] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

[40] O. K. Manyam, N. Kumar, P. Belhumeur, and D. Kriegman, "Two faces are better than one: Face recognition in group photographs," in *2011 International Joint Conference on Biometrics (IJCB)*, pp. 1–8, IEEE, 2011.

[41] F. Song, X. Tan, and S. Chen, "Exploiting relationship between attributes for improved face verification," *Computer Vision and Image Understanding*, vol. 122, pp. 143–154, 2014.

[42] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," *arXiv preprint arXiv:1312.5402*, 2013.

[43] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis.,"

[44] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "High-performance neural networks for visual object classification," *arXiv preprint arXiv:1102.0183*, 2011.

[45] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[47] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[48] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.