Semantically Decomposing the Latent Spaces of Generative Adversarial Networks

Chris Donahue*  
Department of Music  
University of California, San Diego  
La Jolla, CA 92093  
cdonahue@ucsd.edu

Akshay Balsubramani  
Department of Genetics  
Stanford University  
Stanford, CA 94305  
abalsubr@stanford.edu

Julian McAuley  
Department of Computer Science  
University of California, San Diego  
La Jolla, CA 92093  
jmcauley@cs.ucsd.edu

Zachary C. Lipton*  
Department of Computer Science  
University of California, San Diego  
La Jolla, CA 92093  
zlipton@cs.ucsd.edu

Abstract

We propose a new algorithm for training generative adversarial networks to jointly learn latent codes for both identities (e.g. individual humans) and observations (e.g. specific photographs). In practice, this means that by fixing the identity portion of latent codes, we can generate diverse images of the same subject, and by fixing the observation portion we can traverse the manifold of subjects while maintaining contingent aspects such as lighting and pose. Our algorithm features a pairwise training scheme in which each sample from the generator consists of two images with a common identity code. Corresponding samples from the real dataset consist of two distinct photographs of the same subject. In order to fool the discriminator, the generator must produce images that are both photorealistic, distinct, and appear to depict the same person. We augment both the DCGAN and BEGAN approaches with Siamese discriminators to accommodate pairwise training. Experiments with human judges and an off-the-shelf face verification system demonstrate our algorithm’s ability to generate convincing, identity-matched photographs.

1 Introduction

Generative adversarial networks (GANs) learn mappings from latent codes in some low-dimensional space $Z$ to points in the space of natural data $\mathcal{X}$ \cite{2}. They achieve this power through an adversarial training scheme pitting a generative model $G: Z \mapsto \mathcal{X}$ against a discriminative model $D: \mathcal{X} \mapsto [0, 1]$ in a minimax game. The generator learns to map low-dimensional vectors $z \in Z$, sampled i.i.d. according to some prespecified distribution, to plausible counterfeits $G(z) \in \mathcal{X}$. The discriminator learns to differentiate between samples from the resulting generated distribution $P_G$ over $\mathcal{X}$ and the real data distribution $P_R$. The generator’s training objective is to fool the discriminator.

Since their introduction, GANs have attracted widespread attention for their ability to produce high-fidelity images when trained on corpora of natural images. Several papers have since advanced these capabilities through architectural improvements \cite{25} and modifications to the training scheme \cite{3, 2, 41, 4}. Another line of research addresses how to train GANs to produce class-conditional output. Conditional GANS (cGANs), proposed by Mirza and Osindero \cite{23}, require learning a distinct weight vector for each class. This is impractical when there are thousands or millions of

* Corresponding author
classes and few instances of each. Moreover, in such cases we might want to learn a latent space corresponding to the classes themselves.

To make the discussion concrete, let’s consider a dataset consisting of 10M photographs of celebrity faces. This dataset contains 100 photographs (observations) each for 100,000 distinct people (identities). After training a traditional GAN on this dataset, we can synthesize face images at will. By traversing latent space, we might add glasses to a face, transition between a photograph of a woman and one of a man, or shift the pose from profile to center. But what if we want to fix the subject of a photograph and traverse the entire manifold of lighting, pose, expression, etc? Or instead fix these contingent aspects and vary the identity of the subject? The generic GAN framework offers no clear way to do this.

We propose Semantically Decomposed GANs (SD-GANs), which coerce a specified portion of the latent space to correspond to a known source of variation. The technique decomposes $Z$ into one portion $Z_I$ corresponding to identity, and the remaining portion $Z_O$ corresponding to the other contingent aspects of observations. SD-GANs learn through a pairwise training scheme in which each sample from the generator consists of a pair of images with common $z_I \in Z_I$ but differing $z_O \in Z_O$. Each sample from the real dataset consists of two distinct images of the same person (i.e., with the same identity). In order to fool the discriminator, the generator must not only produce diverse and photorealistic images, but also images with the same identity when $z_I$ is fixed. For SD-GANs, we modify the discriminator so that it can determine whether the pair of samples constitute a match.

Our experiments with a dataset of celebrity faces demonstrate that SD-GANs can generate contrasting images of the same subject (Figure 1). The generator learns that certain properties are free to vary across observations but not identity. For example, SD-GANs learn that pose, facial expression, hair styles, black & white vs. color, and lighting can all vary across different photographs of the same individual. On the other hand, the aspects that are more salient for facial verification remain consistent as we vary the observation code $z_O$. We demonstrate that SD-GANs trained on faces generate stylistically contrasting, identity-matched image pairs that human annotators and a state-of-the-art face verification algorithm recognize as depicting the same subject. We also train SD-GANs on a dataset of product images, containing multiple photographs of each product from various perspectives (Figure 4).

2 Generative adversarial networks

GANs leverage the discriminative power of neural networks to learn generative models. The learning process consists of a minimax game between a generative model $G$, parameterized by $\theta_G$, and a discriminative model $D$, parameterized by $\theta_D$. In the original formulation, the discriminative model tries to maximize log likelihood, yielding:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim P_r}[\log D(x)] + \mathbb{E}_{z \sim P_z}[\log 1 - D(G(z))].$$ (1)

Training proceeds as follows: For $k$ iterations, sample one mini-batch from the real distribution $P_R$ and one from the distribution of generated images $P_G$, updating discriminator weights $\theta_D$ to
when the discriminator cannot distinguish between real and generated images, they introduce a new
Additionally, they introduce a method for stabilizing training. Positing that training becomes unstable
with the perceptual quality of generated images.

Algorithm 1 Semantically Decomposed GAN Training

1: for n in 1:NumberOfIterations do
2:   for m in 1:MinibatchSize do
3:     Sample one identity vector \( z_I \) \( \sim \) Uniform\([-1, 1]^d_I \).
4:     Sample two observation vectors \( z_O^1, z_O^2 \) \( \sim \) Uniform\([-1, 1]^d_O \).
5:     \( z_1 \leftarrow [z_I; z_O^1], z_2 \leftarrow [z_I; z_O^2]. \)
6:     Generate pair of images \( \hat{G}(z_1), \hat{G}(z_2) \), adding them to the minibatch with label 0 (fake).
7:   for m in 1:MinibatchSize do
8:     Sample one identity \( i \in I \) from the real data set.
9:     Sample two distinct images \( x_1, x_2 \sim Pr(x|I = i). \)
10:    Add the pair to the minibatch, assigning label 1 (real).
11:   Update discriminator weights by \( \theta_D \leftarrow \theta_D + \nabla_{\theta_D} V(G, D) \) using its stochastic gradient.
12:   Sample another minibatch of identity-matched latent vectors \( z_1, z_2. \)
13:   Update generator weights by stochastic gradient descent \( \theta_G \leftarrow \theta_G + \nabla_{\theta_G} V(G, D). \)

increase \( V(G, D) \) by stochastic gradient ascent. Then sample a minibatch from \( P_z \), updating the \( \theta_G \) to decrease \( V(G, D) \) by stochastic gradient descent.

Goodfellow et al. [12] show that GAN training corresponds to minimizing the Jensen-Shannon divergence between the generated distribution and the data-generating process. Following this work, several authors have explored minimizing other objectives.

Notably, Arjovsky et al. [3] use the Wasserstein distance, which amounts to maximizing over a more restricted family of discriminator models, with parameters in some set \( \mathcal{W} \) describing Lipschitz-constrained neural networks:

\[
\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim P_z}[f_w(G(z))].
\] (2)

To enforce the Lipschitzness of the discriminator, they perform weight-clipping. They report more stable training than the original GAN and the useful property that discriminator loss tends to correlate with the perceptual quality of generated images.

Zhao et al. [41] propose energy-based GANs (EBGANs), in which the discriminator can be viewed as an energy function. Specifically, they devise a discriminator consisting of an autoencoder:

\[ D(x) = \hat{D}_x(D_x(x)). \]

In the minimax game, the discriminator’s weights are updated to minimize the reconstruction error \( \mathcal{L}(x) = ||x - D(x)|| \) for real data, while maximizing the error \( \mathcal{L}(G(z)) \) for the generator.

More recently, Berthelot et al. [4] extend this work, introducing Boundary Equilibrium GANs (BEGANs), which optimize the Wasserstein distance between autoencoder loss distributions, yielding the formulation:

\[
V(G, D) = \mathcal{L}(x) - \mathcal{L}(G(z)).
\] (3)

Additionally, they introduce a method for stabilizing training. Positing that training becomes unstable when the discriminator cannot distinguish between real and generated images, they introduce a new hyperparameter \( \gamma \), updating the value function on each iteration to maintain a desired ratio between the two reconstruction errors:

\[ \mathbb{E}[\mathcal{L}(G(z))] = \gamma \mathbb{E}[\mathcal{L}(x)]. \]

The BEGAN model produces what appear to us, subjectively, to be the sharpest images of faces yet generated by a GAN. In this work, we adapt both the DCGAN [25] and BEGAN algorithms to the SD-GAN training scheme.

3 Semantically decomposed GANs

To formulate our SD-GAN technique more fully, consider the data’s identity as a random variable \( I \) in a discrete index set \( I \). We seek to learn a latent representation that conveniently decomposes the variation in the real data into two parts: due to \( I \), and due to the other factors of variation in the data, packaged as a random variable \( O \). Ideally, the decomposition of the variation in the data into \( I \) and \( O \)
Figure 2: SD-GAN architectures and vanilla counterparts. Our SD-GAN models incorporate a decomposed latent space and Siamese discriminators. Dashed lines indicate shared weights. Discriminators also observe real samples in addition to those from the generator, not pictured for simplicity.

should correspond exactly to a decomposition of the latent space $Z = Z_I \times Z_O$. This would permit convenient interpolation and other operations on the inferred subspaces $Z_I$ and $Z_O$.

A conventional GAN samples $I, O$ from their joint distribution. Such a GAN’s generative model samples directly from an unstructured prior over the latent space. It does not disentangle the variation in $O$ and $I$, for instance by modeling conditional distributions $P(O \mid I = i)$, but only models their average with respect to the prior on $I$.

Our SD-GAN method learns such a latent space decomposition, partitioning the coordinates of $Z$ into two parts representing the subspaces, so that any $z \in Z$ can be written as the concatenation $[z_I; z_O]$ of its identity representation $z_I \in \mathbb{R}^{d_I} = Z_I$ and its contingent aspect representation $z_O \in \mathbb{R}^{d_O} = Z_O$. SD-GANs achieve this through a pairwise training scheme in which each sample from the real data consists of $x_1, x_2 \sim P_R(x \mid I = i)$, a pair of images with a common identity $i \in I$. Each sample from the generator consists of $G(z_1), G(z_2) \sim P_G(z \mid Z_I = z_I)$, a pair of images generated from a common identity vector $z_I \in Z_I$ but i.i.d. observation vectors $z_1, z_2 \in Z_O$. We assign identity-matched pairs from $P_R$ the label 1 and $z_I$-matched pairs from $P_G$ the label 0. The discriminator can thus learn to reject pairs for either of two primary reasons: 1) not photorealistic or 2) not plausibly depicting the same subject. See Algorithm 1 for pseudocode for SD-GAN training.

3.1 Discriminator architecture

With SD-GANs, we see no reason to alter the architecture of the generator. However, the discriminator must now act upon two images, producing a single output. Moreover, the effects of the two input images $x_1, x_2$ on the output score are not independent. Two images might be otherwise photorealistic but deserve rejection because they clearly depict different identities. To this end, we devise two novel discriminator architectures to adapt DCGAN and BEGAN respectively. In both cases, we first separately encode each image using the same convolutional neural network $D_e$. We choose this Siamese setup [5,7] as our problem is symmetrical in the images, and thus it’s sensible to share weights between the encoders.

To adapt the DCGAN (Figure 2a), we stack the feature maps $D_e(x_1)$ and $D_e(x_2)$ along the channel axis, applying one additional strided convolution. This allows the network to further aggregate information from the two images before flattening and fully connecting to a $[0, 1]$ output neuron. For BEGAN, because the discriminator is an autoencoder, our architecture is more complicated. After
encoding each image, we concatenate the representations \([D_e(x_1); D_e(x_2)] \in \mathbb{R}^{2(d_I+d_O)}\) and apply one fully connected bottleneck layer \(\mathbb{R}^{2(d_I+d_O)} \Rightarrow \mathbb{R}^{d_I+2d_O}\) with linear activation. In alignment with BEGAN, the SD-BEGAN bottleneck has the same dimensionality as the tuple of latent codes \((z_I, z_1^O, z_2^O)\) that generated the pair of images. Following the bottleneck, we apply a second FC layer \(\mathbb{R}^{d_I+2d_O} \Rightarrow \mathbb{R}^{2(d_I+d_O)}\), taking the first \(d_I + d_O\) components of its output to be the input to the first decoder and the second \(d_I + d_O\) components to be the input to the second decoder. The shared intermediate layer gives SD-BEGAN a mechanism to push apart matched and unmatched pairs. We specify our exact architectures in full detail in Appendix B.

4 Experiments

We experimentally validate SD-GANs using two datasets: 1) the MS-Celeb-1M dataset of celebrity face images [13] and 2) a dataset of shoe images scraped from Amazon [22]. Both datasets contain a large number of identities (faces and shoes, respectively) with multiple observations of each. Celebrity faces are a richer domain for our method as both identities and contingent factors are significant sources of variation. Thus, our primary experiments and evaluation focus on this data. In contrast, Amazon’s shoe images tend to vary only with camera perspective for a given product, making this data useful for sanity-checking our approach.

4.1 Faces

From the aligned face images in the MS-Celeb-1M dataset, we select 12,500 celebrities at random and 8 associated images of each, resizing them to 64x64 pixels. We split the celebrities into subsets of 10,000 (training), 1,250 (validation) and 1,250 (test). The dataset has a small number of duplicate images and some label noise (images matched to the wrong celebrity). To detect and remove duplicates, we hash the images. We do not explicitly rid the data of label noise, demonstrating the robustness of our algorithm. Because each training example consists of a pair, the training set contains a total of 280,000 examples \((8^2 \times 10,000)\). We scale the input values to \([-1, 1]\), performing no additional preprocessing or data augmentation on these images.

4.2 Shoes

Product images are another promising application for our method. In this domain, we have access to multiple images of each product in different orientations. Generally, product photographs are captured against white backgrounds and primarily differ in orientation and distance. Accordingly, we expect that SD-GAN training will tie the observation latent space to capture these aspects. We choose to study shoes as a prototypical example of a category of product images. The Amazon dataset contains around 3,000 unique products with the category “Shoe” and multiple product images. We use the same 80\%, 10\%, 10\% split and again hash the images to ensure that subsets are disjoint. There are an average of 6.2 photos of each product.
Table 1: Pairwise face verification results for 10k pairs. 
from MS-Celeb-1M (real) and SD-GANs (generated).
FN: FaceNet, MT: Mechanical Turk, Us: Authors.
*1k pairs, **200 pairs

| Dataset       | Judge | AUC | Acc. | TPR | FPR |
|---------------|-------|-----|------|-----|-----|
| MS-Celeb-1M   | FN    | .913| 86.7 | 78.0| 4.5 |
| SD-DCGAN      | FN    | .823| 74.9 | 69.9| 20.1|
| SD-DCWGAN     | FN    | .770| 68.5 | 51.5| 14.6|
| SD-DCGAN-SC   | FN    | .831| 75.7 | 69.4| 18.0|
| SD-DCGAN k=4 | FN    | .852| 77.6 | 75.0| 22.7|
| SD-DCGAN dI=25| FN    | .835| 76.4 | 75.1| 22.2|
| SD-DCGAN dI=75| FN    | .816| 74.3 | 75.4| 26.8|
| SD-BEGAN      | FN    | .928| 85.7 | 82.4| 11.0|

| Dataset       | Judge | AUC | Acc. | TPR | FPR |
|---------------|-------|-----|------|-----|-----|
| MS-Celeb-1M** | Us    | -   | 85.0 | 82.0| 11.0|
| MS-Celeb-1M*  | MT    | -   | 73.4 | 51.7| 3.7 |
| SD-DCGAN k=4* | MT    | -   | 66.4 | 57.5| 24.4|
| SD-BEGAN*     | MT    | -   | 68.3 | 44.4| 7.8 |

Figure 5: ROC curve for FaceNet on ground truth and generated datasets

4.3 Training details

We train SD-DCGANs on both of our datasets for 500,000 iterations using batches of 16 identity-matched pairs. For MS-Celeb-1M, we compare results using the original GAN loss [12] to those using the Wasserstein distance-based loss proposed by Arjovsky et al. [3] (SD-DCWGAN). To optimize SD-DCGAN, we use the Adam optimizer [18] with hyperparameters $\alpha = 2e-4$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ as recommended by Radford et al. [25]. To optimize SD-DCWGAN, we use RMS-prop [14] with $\alpha = 5e-5$. We also consider a non-Siamese discriminator (SD-DCGAN-SC) that simply stacks the channels of the pair of real or fake images before encoding.

As in [25], we sample latent vectors $z \sim \text{Uniform}([-1, 1]^{100})$. For SD-GANs, we partition the latent codes according to $z_I \in \mathbb{R}^{d_I}$, $z_O \in \mathbb{R}^{100-d_I}$ using values of $d_I = [25, 50, 75]$. Our idea can be trivially applied with $k$-wise training (vs. pairwise). To explore the effects of using $k > 2$, we also implement an SD-DCGAN where we sample $k = 4$ instances each from $P_G(z \mid Z_I = z_I)$ for some $z_I \in \mathcal{Z}_I$ and from $P_G(x \mid I = i)$ for some $i \in \mathcal{I}$. For all experiments, unless otherwise stated, we use $d_I = 50$ and $k = 2$.

We also train an SD-BEGAN on both of our datasets. For MS-Celeb-1M, we tried to train an SD-BEGAN with $k = 4$ but observed early mode collapse (Appendix C). The increased complexity of the SD-BEGAN model significantly increases training time, requiring almost 48 hours to complete 500,000 iterations, thus limiting our ability to perform exhaustive hyperparameter validation. Berthelot et al. [4] sample latent vectors $z \sim \text{Uniform}([-1, 1]^{64})$, however we use 100-dimensional latent vectors for direct comparison to SD-DCGAN. We use the Adam optimizer with the default-hyperparameters from [18] for our SD-BEGAN experiments.

5 Evaluation

The evaluation of generative models is a fraught topic. Quantitative measures of sample quality can be poorly correlated [31] with each other. Accordingly, we design an evaluation to match conceivable uses of our algorithm. Because we hope to produce samples which humans deem to depict the same person, we evaluate our generative model on a face verification task, using both a pretrained face verification model and crowd-sourced human judgments obtained through Amazon’s Mechanical Turk platform.

Recent advancements in face verification using deep convolutional neural networks [27, 24, 35] have yielded accuracy rivaling humans. For our evaluation, we procure FaceNet, a publicly-available face verifier based on the popular Inception-ResNet CNN architecture [28]. The FaceNet model was
pretrained on the CASIA-WebFace dataset \cite{37} and achieves 98.6% accuracy on the benchmark LFW task \cite{15}. \cite{2}

FaceNet ingests normalized, 160x160 color images and produces an embedding $f(x) \in \mathbb{R}^{128}$. FaceNet was trained to minimize the $L_2$ distance between matched pairs of faces and to maximize the distance for mismatched pairs. Accordingly, the embedding space yields a function for measuring the similarity between two faces $x_1$ and $x_2$: $D(x_1, x_2) = ||f(x_1) - f(x_2)||^2_2$. Given two images, $x_1$ and $x_2$, we label them as a match if $D(x_1, x_2) \leq \tau_v$, where $\tau_v$ is the threshold maximizing accuracy on a class-balanced set of pairs from MS-Celeb-1M validation data. We use the same threshold when evaluating both real and synthetic data with FaceNet.

We compare the performance of FaceNet on pairs of images from the MS-Celeb-1M test set against generated samples from our trained SD-DCGAN and SD-BEGAN generative models. To match FaceNet’s training data, we preprocess all images by resizing from 64x64 to 160x160, normalizing each image individually. We prepare 10,000 pairs from MS-Celeb-1M, half identity-matched and half unmatched. From each generative model, we generate 5,000 pairs each with $z^I_1 = z^I_2$ and 5,000 pairs with $z^I_1 \neq z^I_2$. For each sample, we draw observation vectors $z^O$ randomly.

In Table \ref{tab:accuracy}, we report the accuracy, true positive rate and false positive rate of FaceNet using threshold $\tau_v$ for all datasets. We also report the AUROC and present the full ROC curve for all datasets in Figure \ref{fig:roc}. Sample results from the best SD-DCGAN ($k = 4$, $d_I = 50$) and SD-BEGAN ($k = 2$, $d_I = 50$) model as determined by FaceNet are shown in Figures \ref{fig:sd-dcgan} and \ref{fig:sd-began} respectively.

In addition to validating that identity-matched SD-GAN samples are verified by FaceNet, we also demonstrate that humans are similarly convinced through experiments using Mechanical Turk. For these experiments, we use a balanced subset of 1,000 pairs from MS-Celeb-1M and the most promising SD-GANs from our FaceNet evaluation. We ask human annotators to determine if each pair depicts the “same person” or “different people”. Random batches of ten pairs are evaluated by ensembles of three unique annotators and predictions are determined by majority vote. We also manually judge 200 pairs from MS-Celeb-1M to provide a benchmark for assessing the quality of the Mechanical Turk ensembles. Results are summarized in Table \ref{tab:accuracy}.

\section*{6 Discussion}

Our evaluation shows that FaceNet recognizes matched faces in the MS-Celeb-1M with 86.7% accuracy. We (the authors) achieve similar accuracy when manually annotating a small subset. These results indicate some amount of label noise in MS-Celeb-1M. Our most promising SD-DCGAN and SD-BEGAN models produce matched pairs of faces that the pretrained FaceNet model recognizes as matches with an accuracy similar to that achieved on real images. FaceNet verifies pairs of faces from SD-BEGAN with 85.7% accuracy, just shy of the 86.7% achieved on MS-Celeb-1M. At stricter thresholds ($\tau < \tau_v$), true positive rates are lower for SD-GAN pairs than for MS-Celeb-1M but interestingly, for less strict thresholds, the SD-BEGAN achieves a higher true positive rate than MS-Celeb-1M images (Figure \ref{fig:roc}).

\footnote{20170214-092102 pretrained model and code from https://github.com/davidsandberg/facenet}
For all datasets, human annotators on Mechanical Turk answered “same person” less frequently than FaceNet at the accuracy-maximizing threshold $\tau_v$. Even on real data, balanced so that 50% of pairs are identity-matched, human annotators report “same person” only 27.6% of the pairs. Annotators achieve 66.4% and 68.3% accuracy on SD-DCGAN and SD-BEGAN pairs respectively, compared to 73.4% accuracy on real data. Notably, annotators answered “same” for 40.6% of the SD-DCGAN examples, almost twice as often as for the MS-Celeb-1M and SD-BEGAN datasets. This provides some evidence that SD-DCGAN produces less identity diversity.

Face samples from SD-BEGAN appear comparably crisp and globally coherent to those generated by a vanilla BEGAN. Given a SD-GAN trained generator, we can independently interpolate along the identity and observation manifolds (Figure 6). Here, the diagonal represents the entangled interpolation typically shown for ordinary GANS. In Figure 6, we demonstrate that when we vary the observation vector $z_O$, SD-GANs can change the color of clothing, add or remove sunglasses, or change pose. They can also perturb the lighting, color saturation, and contrast of an image, all while keeping the apparent identity fixed. We note, subjectively, that samples from SD-DCGAN tend to appear less photorealistic than those from SD-BEGAN.

On the shoe dataset, we find that the SD-DCGAN model produces convincing results. As desired, manipulating $z_I$ while keeping $z_O$ fixed yields distinct shoes in consistent poses (Figure 4). The identity code $z_I$ appears to capture the broad categories of shoes (sneakers, flip-flops, boots, etc.). We were surprised to find that both the original BEGAN and SD-BEGAN fail to produce diverse pictures of shoes (see Appendix for examples of SD-BEGAN).

7 Related work

Style transfer and novel view synthesis are active research areas. Early attempts to disentangle style and content manifolds used factored tensor representations [30, 34, 10, 29], applying their results to face image synthesis. More recent work focuses on learning hierarchical feature representations using deep convolutional neural networks to separate identity and pose manifolds for faces [43, 26, 44, 36] and products [8]. Gatys et al. [11] use features of a ConvNet, pretrained for image recognition, as a means for discovering content and style vectors.

Since 2014, generative adversarial networks have been used to generate increasingly high-quality images [12, 25, 41, 4]. Conditional GANs, introduced by [23], extend GANS to generate class-conditional data. More recently, conditional GANs have been used to ingest full-resolution images as conditioning information [17, 21, 52], addressing a variety of image-to-image translation and style transfer tasks. Chen et al. [6] devise an information-theoretic extension to GANs in which they maximize the mutual information between a subset of latent variables and the generated data. Their approach appears to disentangle some intuitive factors of variation but provides no method for segmenting the latent code to disentangle known from unknown sources of variance. To our knowledge, we are the first to propose explicitly decomposing the latent space of a GAN to disentangle manifolds of content (identities) and style (observations).

One might think of our work as offering the generative view of the Siamese networks often favored for learning similarity metrics [5, 7]. Such approaches are used for discriminative tasks like face or signature verification that share the many classes with few examples structure that we study here. In our work, we adopt a Siamese architecture in order to enable the discriminator to differentiate between matched and unmatched pairs. Recent work by Liu and Tuzel [20] propose a GAN architecture with weight sharing across multiple generators and discriminators, but with different problem formulations and objectives from ours.

Several attempts have been made to use GANs for identity-preserving novel view synthesis. Tran et al. [33], Huang et al. [16], Yin et al. [38, 39], Zhao et al. [40] all learn methods to synthesize different body/facial poses conditioned on an input image and a fixed number of pose labels. Antipov et al. [1], Duong et al. [9] both propose using conditional GANs to synthesize artificially aged faces conditioned on both a face embedding and an age vector. These methods investigate image translation. At test time, they must be provided with a source image in order to generate a target image.
8 Conclusions

This paper introduces SD-GANs, a new algorithm for decomposing the latent spaces of GANs to isolate the factors corresponding to a known source of variance. Training them on celebrity faces, we show that SD-GANs convincingly disentangle the factors corresponding to individual people from the more superficial aspects of photographs. Moreover, our thorough evaluation, using both human judgments and an off-the-shelf face verification model, confirms that the identity-matched images from SD-GANs appear to depict the same people. We emphasize that the SD-GAN algorithm’s flexibility makes it useful for diverse datasets, factors of variation, and model architectures.

9 Acknowledgements

The authors would like to thank John Berkowitz and Miller Puckette for their helpful feedback on this work. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575 [32]. The GPUs used for this research were graciously donated by the NVIDIA Corporation.

References

[1] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. arXiv:1702.01983, 2017.
[2] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In ICLR, 2017.
[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv:1701.07875, 2017.
[4] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. arXiv:1703.10717, 2017.
[5] Jane Bromley. Signature verification using a “siamese” time delay neural network. In NIPS, 1994.
[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In NIPS, 2016.
[7] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In CVPR, 2005.
[8] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In CVPR, 2015.
[9] Chi Nhan Duong, Kha Gia Quach, khoa luu, Marios Savvides, et al. Temporal non-volume preserving approach to facial age-progression and age-invariant face recognition. arXiv:1703.08617, 2017.
[10] Ahmed Elgammal and chan-su lee. Separating style and content on a nonlinear manifold. In CVPR, 2004.
[11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In CVPR, 2016.
[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.
[13] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large scale face recognition. In ECCV, 2016.
[14] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini–batch gradient descent.
[15] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In NIPS, 2012.

[16] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. arXiv:1704.04086, 2017.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. arXiv:1611.07004, 2016.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[19] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. ICLR Workshop Track, 2017.

[20] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In NIPS, 2016.

[21] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. arXiv:1703.00848, 2017.

[22] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In SIGIR, 2015.

[23] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv:1411.1784, 2014.

[24] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In BMVC, 2015.

[25] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In ICLR, 2016.

[26] Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In ICML, 2014.

[27] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In CVPR, 2015.

[28] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In AAAI, 2017.

[29] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Tensor analyzers. In ICML, 2013.

[30] Joshua B Tenenbaum and William T Freeman. Separating style and content. In NIPS, 1997.

[31] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. arXiv:1511.01844, 2015.

[32] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, et al. Xsede: accelerating scientific discovery. Computing in Science & Engineering, 2014.

[33] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In CVPR, 2017.

[34] M Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In ECCV, 2002.

[35] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In ECCV, 2016.

[36] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In NIPS, 2015.

[37] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. arXiv:1411.7923, 2014.
[38] Weidong Yin, Yanwei Fu, Leonid Sigal, and Xiangyang Xue. Semi-latent gan: Learning to generate and modify facial images from attributes. *arXiv:1704.02166*, 2017.

[39] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Towards large-pose face frontalization in the wild. *arXiv:1704.06244*, 2017.

[40] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, and Jiashi Feng. Multi-view image generation from a single-view. *arXiv:1704.04886*, 2017.

[41] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *ICLR*, 2017.

[42] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv:1703.10593*, 2017.

[43] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning identity-preserving face space. In *ICCV*, 2013.

[44] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Multi-view perceptron: a deep model for learning face identity and view representations. In *NIPS*, 2014.
A Estimating Latent Codes

We estimate latent vectors for unseen images and demonstrate that the disentangled representation of SD-GANs can be used to pose an estimated identity vector with different observation vectors. In order to find a latent vector \( \hat{z} \) such that \( G(\hat{z}) \) (using pretrained generator \( G \)) is similar to an unseen image \( x \), we can optimize the following with stochastic gradient descent\cite{19}:

\[
\min_{\hat{z}} \| G(\hat{z}) - x \|^2_2
\]

In Figure 7, we depict estimation and linear interpolation across both subspaces for two pairs of images using SD-BEGAN, producing matrices of faces. We also display the corresponding source images being estimated. For both matrices, \( \hat{z}_I \) (identity) is consistent in each row and \( \hat{z}_O \) (observation) is consistent in each column.

In the left matrix, we note that identities appear consistent in all rows while facial position rotates from the left to the right side of the image across the columns. In the right matrix, we see a smiling, color image of a woman transform into a black and white image of a man with a more neutral expression. From top to bottom, we see facial identity morph from that of a man to that of a woman, and from left to right we see consistent changes in smile intensity, chroma and pose.

B Architecture Descriptions

We list here the full architectural details for our SD-DCGAN and SD-BEGAN models. In these descriptions, \( k \) is the number of images that the generator produces and discriminator observes per identity (usually 2), and \( d_I \) is the number of dimensions in the latent space \( Z_I \) (identity). In our experiments, dimensionality of \( Z_O \) is always 100 - \( d_I \). As a concrete example, the bottleneck layer of the SD-BEGAN discriminator autoencoder (“fc2” in the table) with \( k = 2, d_I = 50 \) has output dimensionality 150.

We emphasize that generators are parameterized by \( k \) in the tables only for clarity and symmetry with the discriminators. Implementations do not need to separate \( k \) in the generator abstraction. Instead, \( k \) can be collapsed into the batch size.

For the stacked-channels versions of these discriminators, we simply change the number of input image channels from 3 to 3\( k \) and set \( k = 1 \) wherever \( k \) appears in the table.
Table 2: Input abstraction for both SD-DCGAN and SD-BEGAN generators during training (where \( z_0 \) is always different for every pair or set of \( k \))

| Operation | Input Shape | Kernel Size | Output Shape |
|-----------|-------------|-------------|--------------|
| \([z_i; z_o]\) | \([d_I, k, k, 100-d_I]\) | \([k, k, 100-d_I]\) | \([k, k, 100-d_I]\) |
| dup \(z_i\) | \([d_I, k, k, 100-d_I]\) | \([k, k, 100-d_I]\) | \([k, k, 100-d_I]\) |
| concat | \([k, d_I, k, 100-d_I]\) | \([k, 100]\) | \([k, 100]\) |

Table 3: SD-DCGAN generator architecture

| Operation | Input Shape | Kernel Size | Output Shape |
|-----------|-------------|-------------|--------------|
| \(z\) | \((k, 100)\) | \((k, 100)\) |
| fc1 | \((k, 8192)\) | \((100, 8192)\) | \((k, 8192)\) |
| reshape | \((k, 4, 4, 512)\) | \((k, 4, 4, 512)\) | \((k, 4, 4, 512)\) |
| bnorm | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) |
| relu | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) |
| upconv1 | \((k, 4, 4, 512)\) | \((5, 5, 128, 256)\) | \((k, 16, 16, 128)\) |
| bnorm | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| relu | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| upconv2 | \((k, 8, 8, 256)\) | \((5, 5, 128, 128)\) | \((k, 32, 32, 64)\) |
| bnorm | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| relu | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| upconv3 | \((k, 8, 8, 256)\) | \((5, 5, 128, 128)\) | \((k, 32, 32, 64)\) |
| bnorm | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) |
| relu | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) |
| upconv4 | \((k, 32, 32, 64)\) | \((5, 5, 64, 3)\) | \((k, 64, 64, 3)\) |
| tanh | \((k, 64, 64, 3)\) | \((k, 64, 64, 3)\) | \((k, 64, 64, 3)\) |

Table 4: SD-DCGAN discriminator architecture

| Operation | Input Shape | Kernel Size | Output Shape |
|-----------|-------------|-------------|--------------|
| \(x\) or \(G(z)\) | \((k, 64, 64, 3)\) | \((5, 5, 3, 64)\) | \((k, 64, 64, 3)\) |
| downconv1 | \((k, 64, 64, 3)\) | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) |
| lrelu(a=0.2) | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) | \((k, 32, 32, 64)\) |
| downconv2 | \((k, 32, 32, 64)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| bnorm | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| lrelu(a=0.2) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) | \((k, 16, 16, 128)\) |
| downconv3 | \((k, 16, 16, 128)\) | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) |
| bnorm | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) |
| lrelu(a=0.2) | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) | \((k, 8, 8, 256)\) |
| downconv4 | \((k, 8, 8, 256)\) | \((k, 4, 4, 512)\) | \((k, 4, 4, 512)\) |
| stackchannels | \((k, 4, 4, 512)\) | \((4, 4, 512)\) | \((4, 4, 512)\) |
| downconv5 | \((4, 4, 512)\) | \((3, 3, 512, 512)\) | \((2, 2, 512)\) |
| flatten | \((2, 2, 512)\) | \((2048, 1)\) | \((1)\) |
| fc1 | \((2048, 1)\) | \((1)\) | \((1)\) |
| sigmoid | \((1)\) | \((1)\) | \((1)\) |
Table 5: SD-BEGAN generator architecture

| Operation   | Input Shape | Kernel Size | Output Shape |
|-------------|-------------|-------------|--------------|
| \( x \)     | (k,100)    | (100,8192)  | (k,100)     |
| fc1         | (k,100,)   | (100,8192)  | (k,100,128) |
| reshape     | (k,100,8192) | (k,8,8,128) |              |
| elu         | (k,8,128)  | (3,128,128) | (k,8,8,128) |
| conv2d      | (k,8,128)  | (3,128,128) | (k,8,8,128) |
| elu         | (k,8,128)  | (3,128,128) | (k,8,8,128) |
| upsample2   | (k,8,128)  | (k,16,16,128) | (k,16,16,128) |
| conv2d      | (k,16,16,128) | (k,16,16,128) | (k,16,16,128) |
| elu         | (k,16,16,128) | (k,16,16,128) | (k,16,16,128) |
| conv2d      | (k,16,16,128) | (k,16,16,128) | (k,16,16,128) |
| elu         | (k,16,16,128) | (k,16,16,128) | (k,16,16,128) |
| upsample2   | (k,32,32,128) | (k,32,32,128) | (k,32,32,128) |
| conv2d      | (k,32,32,128) | (k,32,32,128) | (k,32,32,128) |
| elu         | (k,32,32,128) | (k,32,32,128) | (k,32,32,128) |
| conv2d      | (k,32,32,128) | (k,32,32,128) | (k,32,32,128) |
| elu         | (k,32,32,128) | (k,32,32,128) | (k,32,32,128) |
| downconv2d  | (k,64,64,128) | (k,32,32,256) | (k,32,32,256) |
| elu         | (k,32,32,256) | (k,32,32,256) | (k,32,32,256) |
| conv2d      | (k,32,32,256) | (k,32,32,256) | (k,32,32,256) |
| elu         | (k,32,32,256) | (k,32,32,256) | (k,32,32,256) |
| conv2d      | (k,32,32,256) | (k,32,32,256) | (k,32,32,256) |
| elu         | (k,32,32,256) | (k,32,32,256) | (k,32,32,256) |
| downconv2d  | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| elu         | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| conv2d      | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| elu         | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| downconv2d  | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| elu         | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| conv2d      | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| elu         | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| conv2d      | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| elu         | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| flatten     | (k,8,8,512)  | (32768,100)  | (k,100)      |
| fc1         | (k,32768)   | (32768,100)  | (k,100)      |
| concat      | (k,100)     | (100k,100k)  | (100k,100k)  |
| fc2         | (100k,)     | (d1+(100-d1)k,) | (100k,)     |
| fc3         | (d1+(100-d1)k,) | (d1+(100-d1)k,100k,) | (100k,)     |
| split       | (100k,)     | (k,100)      | (k,100)      |
| SD-BEGAN G   | (k,100)     | (k,8,8,128)  | (k,64,64,3)  |

Table 6: SD-BEGAN discriminator autoencoder architecture. The decoder portion is equivalent to, but does not share weights with, the SD-BEGAN generator architecture. We avoid repeating it for brevity.

| Operation   | Input Shape | Kernel Size | Output Shape |
|-------------|-------------|-------------|--------------|
| \( x \)     | (k,64,64,3) | (3,3,3,128) | (k,64,64,3)  |
| conv2d      | (k,64,64,3) | (3,3,128,128) | (k,64,64,128) |
| elu         | (k,64,64,3) | (3,128,128)  | (k,64,64,128) |
| conv2d      | (k,64,64,128) | (k,16,16,128) | (k,16,16,128) |
| elu         | (k,64,64,128) | (k,16,16,128) | (k,16,16,128) |
| conv2d      | (k,64,64,128) | (k,16,16,128) | (k,16,16,128) |
| elu         | (k,64,64,128) | (k,16,16,128) | (k,16,16,128) |
| downconv2d  | (k,64,64,128) | (k,16,16,384) | (k,16,16,384) |
| elu         | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| conv2d      | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| elu         | (k,16,16,384) | (k,16,16,384) | (k,16,16,384) |
| downconv2d  | (k,16,16,384) | (k,8,8,512)  | (k,8,8,512)  |
| elu         | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| conv2d      | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| elu         | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| conv2d      | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| elu         | (k,8,8,512)  | (3,3,3,512)  | (k,8,8,512)  |
| flatten     | (k,8,8,512)  | (32768,100)  | (k,100)      |
| fc1         | (k,32768)   | (32768,100)  | (k,100)      |
| concat      | (k,100)     | (100k,100k)  | (100k,100k)  |
| fc2         | (100k,)     | (d1+(100-d1)k,) | (100k,)     |
| fc3         | (d1+(100-d1)k,) | (d1+(100-d1)k,100k,) | (100k,)     |
| split       | (100k,)     | (k,100)      | (k,100)      |
| SD-BEGAN G   | (k,100)     | (k,64,64,3)  | (k,64,64,3)  |
C  Face Samples

Below we present samples from each model reported in Table [1] for qualitative comparison. In each matrix, \( z_I \) is the same across all images in a row and \( z_O \) is the same across all images in a column. We draw identity and observation vectors randomly for these samples.

Figure 8: Generated samples from SD-DCGAN

Figure 9: Generated samples from SD-DCGAN trained with the Wasserstein GAN loss

Figure 10: Generated samples from SD-DCGAN with stacked-channel discriminator

Figure 11: Generated samples from SD-DCGAN with \( k = 4 \)
Figure 12: Generated samples from SD-DCGAN with $d_I = 25$

Figure 13: Generated samples from SD-DCGAN with $d_I = 75$

Figure 14: Generated samples from SD-BEGAN

Figure 15: Generated samples from SD-BEGAN with $k = 4$, demonstrating mode collapse
D  Shoe Samples

Below we present samples from an SD-DCGAN and SD-BEGAN trained on our shoes dataset.

Figure 16: Generated samples from SD-DCGAN

Figure 17: Generated samples from SD-BEGAN