Diverse Image Generation via Self-Conditioned GANs

Steven Liu[1] Tongzhou Wang[1] David Bau[1] Jun-Yan Zhu[2] Antonio Torralba[1]
[1]MIT CSAIL [2]Adobe Research

Figure 1: Our proposed self-conditional GAN model learns to perform clustering and image synthesis simultaneously. The model training requires no manual annotation of object classes. Here, we visualize several discovered clusters for both Places365 (top) and ImageNet (bottom). For each cluster, we show both real images and the generated samples conditioned on the cluster index.

Abstract

We introduce a simple but effective unsupervised method for generating realistic and diverse images. We train a class-conditional GAN model without using manually annotated class labels. Instead, our model is conditional on labels automatically derived from clustering in the discriminator's feature space. Our clustering step automatically discovers diverse modes, and explicitly requires the generator to cover them. Experiments on standard mode collapse benchmarks show that our method outperforms several competing methods when addressing mode collapse. Our method also performs well on large-scale datasets such as ImageNet and Places365, improving both image diversity and standard quality metrics, compared to previous methods.

1. Introduction

Despite the remarkable progress of Generative Adversarial Networks (GANs) [13, 5, 20], there remains a significant gap regarding the quality and diversity between *class-conditional* GANs trained on labeled data, and *unconditional* GANs trained without any labels in a fully unsupervised setting [30, 26]. This problem reflects the challenge of *mode collapse*: the tendency for a generator to focus on a subset of modes to the exclusion of other parts of the target distribution [12]. Both empirical and theoretical studies have shown strong evidence that real data has a highly multi-modal distribution [33, 39]. Unconditional GANs trained on such data distributions often completely miss important modes, e.g., not being able to generate one of ten digits for MNIST [37], or omitting object classes such as people and cars within synthesized scenes [3]. Class-conditional GANs alleviate this issue by enforcing labels that require the generator to cover all semantic categories. However, in practice, it is often expensive to obtain labels for large-scale datasets.

In this work, we present a simple but effective training method, self-conditioned GANs, to address mode collapse. We train a class-conditional GAN and automatically obtain image classes by clustering in the discriminator's feature space. Our algorithm alternates between learning a feature representation for our clustering method and learning a better generative model that covers all the clusters. Such partitioning automatically discovers modes the generator is currently missing, and explicitly requires the generator to cover them. Figure 1 shows several discovered clusters and corresponding generated images for each cluster.

Empirical experiments demonstrate that this approach successfully recovers modes on standard mode collapse benchmarks (mixtures of Gaussians, stacked MNIST, CIFAR-10). More importantly, our approach scales well to large-scale image generation, achieving better Fréchet Inception Distance and Inception Score for both ImageNet and Places365, compared to previous unsupervised methods. Our code and models are available on our website.

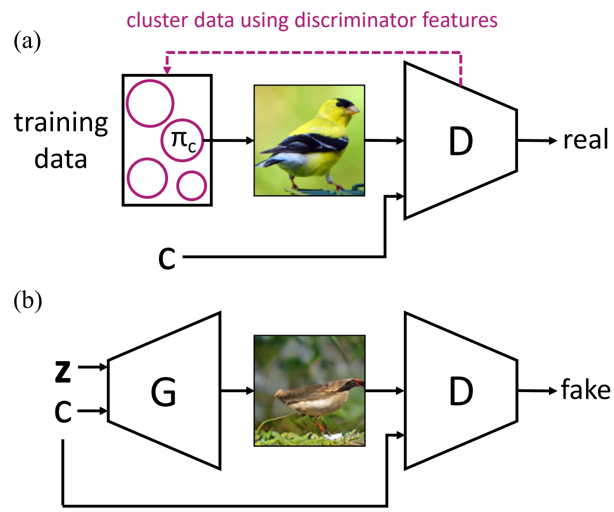


Figure 2: We learn a discriminator D and a generator G that are both conditioned on the automatically discovered cluster c. **(a)** For a specific c, the discriminator D must learn to recognize real images sampled from the cluster π_c of the dataset, and distinguish them from (b) fake images synthesized by the class-conditional generator G. **(b)** The class-conditional generator G synthesizes images from $\mathbf{z}$. By learning to fool D when also conditioned on c, the generator learns to mimic the images in π_c. Our method differs from a conventional conditional GAN as we do not use ground-truth labels to determine the partition $\{\pi_c\}_{c=1}^k$. Instead, our method begins with clustering random discriminator features and periodically reclusters the dataset based on discriminator features.

2. Related Work

Generative Adversarial Networks (GANs). Since the introduction of GANs [13], many variants have been proposed [30, 8, 35, 37, 1, 27, 14, 28], improving both the training stability and image quality. Due to its rapid advance, GANs have been used in a wide range of computer vision and graphics applications [40, 18, 41, 45, 17, 16]. GANs excel at synthesizing photorealistic images for a specific class of images such as faces and cars [19, 20]. However, for more complex datasets such as ImageNet, state-of-the-art models are class-conditional GANs that require ground truth image class labels during training [5]. To reduce the cost of manual annotation, a recent work [26] presents a semi-supervised method based on RotNet [11], a self-supervised image rotation feature learning method. The model is trained with labels provided on only a subset of images. On the contrary, our general-purpose method is not image-specific, and fully unsupervised. Section 4.4 shows that our method outperforms a RotNet-based baseline. A recent method [32] proposes to obtain good clustering using GANs, while we aim to achieve realistic and diverse generation.

Mode collapse. Although GANs are formulated as a minimax game in which each generator is evaluated against a discriminator, during optimization, the generator faces a slowly-changing discriminator that can guide generation to collapse to the point that maximizes the discriminator [29]. Mode collapse does occur in practice, and it is one of the fundamental challenges for GANs training [12, 37]. Several solutions have been proposed, including amending the adversarial loss to look ahead several moves (e.g., Unrolled GAN [29]), jointly training an encoder to recover latent distributions (e.g., VEEGAN [38]), packing the discriminator with sets of points instead of singletons [37, 25], and training a mixture of generators with an auxiliary diversity objective [15, 10]. Different from the above work, our method partitions the real distribution instead of the generated distribution, and devotes a class-conditioned discriminator to each target partition.

Another related line of research trains class-conditional GANs on unlabelled images by clustering on features obtained via unsupervised feature learning methods [26, 36]. Our method directly clusters on discriminator features that inherently exist in GANs, leading to a simpler method and achieving higher quality generation in our experiments (Section 4.4). Mixture Density GAN proposes to use log-likelihoods of a Gaussian mixture distribution in discriminator feature space as the GAN objective [9]. GAN-Tree uses clustering to split a GAN into a tree hierarchy of GANs for better mode coverage [23]. These methods, while also using clustering or mixture models, are mostly orthogonal with our work. Furthermore, the simplicity of our method allows it to be easily combined with a variety of these techniques.

3. Method

One of the core problems in generating diverse outputs in a high-dimensional space such as images is mode collapse: the support of the generator's output distribution can be much smaller than the support of the real data distribution. One way mode collapse has been empirically lessened is by use of a class-conditional GAN, which explicitly penalizes the generator for not having support on each class.

We propose to exploit this class-conditional architecture, but instead of assuming access to true class labels, we will synthesize labels in an unsupervised way. On a high level, our method dynamically partitions the real data space into different clusters, which are used to train a class-conditional GAN. Because generation conditioned on a cluster index is optimized with respect to the corresponding conditioned discriminator, and each discriminator is responsible for a different subset of the real distribution, our method encourages the generator output distribution to cover all partitions of the real data distribution.

Therefore, to train a GAN to imitate a target distribution p_{real}, we partition the data set into k clusters $\{\pi_1, \ldots, \pi_k\}$ that are determined during training. No ground-truth labels are used; training samples are initially clustered in the randomly initialized discriminator feature space, and the clusters are updated periodically. A class-conditional GAN structure is used to split the discriminator and the generator. Next, we describe two core components of our algorithm:

- Conditional GAN training with respect to cluster labels given by the current partitioning.
- Updating the partition according to the current discriminator features of real data periodically.

Conditional GAN training. The GAN consists of a class-conditional generator $G(\mathbf{z}, c)$ associated with a class-conditional discriminator $D(\mathbf{x}, c)$. We denote the internal discriminator feature layers as D_f and its last layer as D_h so $D = D_h \circ D_f$. The generator and discriminator are trained to optimize the following adversarial objective:

$$\mathcal{L}_{\mathsf{GAN}}(D, G) = \mathop{\mathbb{E}}_{c \sim P_\pi}\Bigg[\mathop{\mathbb{E}}_{\mathbf{x}\sim\pi_c}[\log(D(\mathbf{x}, c)] + \mathop{\mathbb{E}}_{\mathbf{z}\sim\mathcal{N}(0,I)}[\log(1 - D(G(\mathbf{z}, c), c))]\Bigg],$$

where the cluster index c is sampled from the categorical distribution P_π that weights each cluster proportional to its true size in the training set. Here G aims to generate images $G(z, c)$ that look similar to the real images of cluster c for $z \sim \mathcal{N}(0, I)$, while $D(\cdot, c)$ tries to distinguish between such generated images and real images of cluster c. They are jointly optimized in the following minimax fashion:

$$\min_G \max_D \mathcal{L}_{\mathsf{GAN}}(D, G). \quad (1)$$

Algorithm 1 Self-Conditioned GAN Training

Initialize generator G and discriminator D
Partition dataset into k sets $\{\pi_1, ..., \pi_k\}$ using D_f outputs
for number of training epochs **do**
  // Conditional GAN training based on current partition
  for number of training iterations for an epoch **do**
    for j in $\{1, ..., m\}$ **do**
      Sample cluster $c^{(j)} \sim P_\pi$, where c is chosen with probability proportional to $|\pi_c|$.
      Sample image $x^{(j)} \sim \pi_{c^{(j)}}$ from cluster $c^{(j)}$.
      Sample latent $z^{(j)} \sim \mathcal{N}(0, I)$.
    end for
    Update G and D according to $\min_G \max_D \mathcal{L}_{\mathsf{GAN}}$ on minibatch $\{(c^{(j)}, x^{(j)}, z^{(j)})\}_j$. ▷ Eqn. (1)
  end for
  // Clustering to obtain new partitions
  Cluster on D_f outputs of a subset of training set to identify a new partition $\{\pi_c^{\mathsf{new}}\}$ into k sets, using previous cluster centroids as initialization.
  Find the matching $\rho(\cdot)$ between $\{\pi_c^{\mathsf{new}}\}_c$ and $\{\pi_c\}_c$ that minimizes $\mathcal{L}_{\mathsf{match}}$. ▷ Eqn. (4)
  Update all $\pi_c \leftarrow \pi^{\mathsf{new}}_{\rho(c)}$.
end for

When under the condition c, the discriminator is encouraged to give low score for any sample that is not from cluster c because $p_{\mathsf{real}}(\mathbf{x} \mid c) = 0$ for all $\mathbf{x} \notin \pi_c$. So the corresponding conditioned generator is penalized for generating points that are not from cluster c, which ultimately prevents the generator from getting stuck on other clusters. The optimization is shown in Figure 2.

Computing new partition by clustering. As the training progresses, the shared discriminator layers D_f learn better representations of the data, so we periodically update π by re-partitioning the target dataset over a metric induced by the current discriminator features. We use k-means clustering [2] to obtain a new partition into k clusters $\{\pi_c\}_{c=1}^k$ according to the D_f output space, approximately optimizing

$$\mathcal{L}_{\mathsf{cluster}}(\{\pi_c\}_{c=1}^k) = \mathop{\mathbb{E}}_{c\sim P_\pi}\Bigg[\mathop{\mathbb{E}}_{\mathbf{x}\sim\pi_c}\Big[\|D_f(\mathbf{x}) - \boldsymbol{\mu}_c\|_2^2\Big]\Bigg], \quad (2)$$

where $\boldsymbol{\mu}_c \triangleq \frac{1}{|\pi_c|}\sum_{x\in\pi_c} D_f(\mathbf{x})$ is the mean of each cluster in D_f feature space.

Clustering initialization. For the first clustering, we use the k-means++ initialization, and when reclustering, we initialize the k-means algorithm with the means induced by the previous clustering. That is, if $\{\pi_c^{\mathsf{old}}\}_{c=1}^k$ denotes the old cluster means and $\{\boldsymbol{\mu}_c^0\}_{c=1}^k$ denotes the k-means initialization to compute the new clustering, we take

$$\boldsymbol{\mu}_c^0 = \frac{1}{|\pi_c^{\mathsf{old}}|} \sum_{x \in \pi_c^{\mathsf{old}}} D_f^{\mathsf{new}}(x), \tag{3}$$

where D_f^{new} denotes the current discriminator feature space.

Matching with old clusters. After repartitioning, to avoid retraining the conditional generator and discriminator from scratch, we match the new clusters $\{\pi_c^{\mathsf{new}}\}_{c=1}^k$ to the old clusters $\{\pi_c^{\mathsf{old}}\}_{c=1}^k$ so that the target distribution for each generator does not change drastically. We formulate the task as a min-cost matching problem, where the cost of matching a π_c^{new} to a $\pi_{c'}^{\mathsf{old}}$ is taken as $|\pi_{c'}^{\mathsf{old}} \setminus \pi_c^{\mathsf{new}}|$, the number of samples missing in the new partition. Therefore, we aim to find a permutation $\rho\colon [k] \to [k]$ that minimizes the objective:

$$\mathcal{L}_{\mathsf{match}}(\rho) = \sum_c |\pi_c^{\mathsf{old}} \setminus \pi_{\rho(c)}^{\mathsf{new}}|. \tag{4}$$

For a given new partitioning from k-means++, we solve this matching using the classic Hungarian min-cost matching algorithm [22], and obtain the new clusters to be used for GAN training in future epochs. Algorithm 1 summarizes the entire training method.

Online clustering. We also experimented with online K-means based on gradient descent [4], where we updated the cluster centers and membership using Equation (2) in each iteration. Our online variant achieves comparable results on mode collapse benchmarks (Section 4.2), but performs worse for real image datasets (Section 4.3), potentially due to the training instability caused by frequent clustering updates. Additionally, in Section 4, we also perform an ablation study regarding clustering initialization, online vs. batch clustering, and with or without clustering matching method.

4. Experiments

4.1. Implementation Details

Network architecture. For experiments on synthetic data, our unconditional generator and discriminator adopt the structure proposed in PacGAN[25]. To condition the generator, we add a fully connected layer before the input layer of the generator. To condition the discriminator, we increase the dimension of the output layer to be the number of clusters k we use. We use the outputs of the last hidden layer of the discriminator as features for clustering.

For Stacked-MNIST dataset, we use the DCGAN architecture [35] identical to that used in prior work[25]. For experiments on CIFAR-10, we use the DCGAN architecture identical to that used in SN-GANs [31]. We add conditioning to the GAN similarly to how we add conditioning to the generator of PacGAN. We use the outputs of the last convolutional layer of the discriminatoras features for clustering.

Large-scale GANs are trained on Places365 [43] and ImageNet [7]. In this setting, our GAN adopts the conditional architecture proposed in Mescheder et al. [28], and our unconditional GAN baseline removes all conditioning on the input label. We use the output from the last ResNet block as features for clustering.

Clustering details. To compute cluster centers, for Stacked-MNIST and CIFAR experiments, we cluster a random subset of 25,000 images from the dataset, and for ImageNet and Places365 experiments, we cluster a random subset of 50,000 images from the dataset. For K-means++, we cluster the subset ten times and choose the clustering that obtains the best performance on the clustering objective. We recluster every 25,000 iterations for Stacked-MNIST and CIFAR experiments, and recluster every 75,000 iterations for ImageNet and Places365 experiments. For experiments on synthetic datasets, we cluster a random sample of 10,000 data points and recluster every 10,000 iterations.

Training details. For experiments on synthetic datasets, we use 2-dimensional latents, and train for 400 epochs using Adam [21] with a batch size 100 and learning rate 10^{-3}. The embedding layer used for conditioning the generator has an output dimension of 32.

For CIFAR-10 and Stacked-MNIST experiments, we use 128 latent dimensions, and Adam with a batch size of 64 and a learning rate of 1×10^{-4} with $\beta_1 = 0, \beta_2 = 0.99$. We train for 50 epochs on Stacked-MNIST and 400 epochs on CIFAR-10. The embedding layer used for conditioning the generator outputs a 256-dimensional vector.

For experiments on Places365 and ImageNet, our loss function is the vanilla loss function proposed by [13] and use R_1 regularization as proposed by [28]. We find that regularization parameter 0.1 works well with our method. We train our networks from scratch using Adam with a batch size of 128 and a learning rate of 10^{-4}, for 200,000 iterations. We choose a 256-dimensional latent space. All metrics are reported using 50,000 samples from the fully trained models, with the Fréchet Inception Distance (FID) computed using the training set. The generator embedding layer outputs a 256-dimensional vector.

4.2. Synthetic Data Experiments

The 2D-ring dataset is a mixture of 8 2D-Gaussians, with means $(\cos\frac{2\pi i}{8}, \sin\frac{2\pi i}{8})$ and variance 10^{-4}, for $i \in \{0, \ldots, 7\}$. The 2D-grid dataset is a mixture of 25 2D-Gaussians, each with means $(2i-4, 2j-4)$ and variance 0.0025, for $i, j \in \{0, \ldots, 4\}$.

We follow the metrics used in prior work [38, 25]. A generated point is deemed high-quality if it is within three standard deviations from some mean [38]. The number of modes covered by a generator is the number of means that have at least one corresponding high-quality point. To

Table 1: Number of modes recovered, percent high quality samples, and reverse KL divergence metrics for 2D-Ring and 2D-Grid experiments. Results are averaged over ten trials, with standard error reported.

	2D-Ring			2D-Grid		
	Modes (Max 8) ↑	% ↑	Reverse KL ↓	Modes (Max 25) ↑	% ↑	Reverse KL ↓
GAN [13]	6.3 ± 0.5	98.2 ± 0.2	0.45 ± 0.09	17.3 ± 0.8	94.8 ± 0.7	0.70 ± 0.07
PacGAN2 [25]	7.9 ± 0.1	95.6 ± 2.0	0.07 ± 0.03	23.8 ± 0.7	91.3 ± 0.8	0.13 ± 0.04
PacGAN3 [25]	7.8 ± 0.1	97.7 ± 0.3	0.10 ± 0.02	24.6 ± 0.4	94.2 ± 0.4	0.06 ± 0.02
PacGAN4 [25]	7.8 ± 0.1	95.9 ± 1.4	0.07 ± 0.02	24.8 ± 0.2	93.6 ± 0.6	0.04 ± 0.01
MGAN ($k = 50$) [15]	7.6 ± 0.2	80.3 ± 1.7	0.32 ± 0.06	22.5 ± 0.6	51.1 ± 1.6	0.36 ± 0.06
Random Labels ($k = 50$)	7.9 ± 0.1	96.3 ± 1.1	0.07 ± 0.02	16.0 ± 1.0	90.6 ± 1.6	0.57 ± 0.07
Our Method ($k = 50$)	$\mathbf{8.0} \pm 0.0$	$\mathbf{99.5} \pm 0.3$	$\mathbf{0.0014} \pm 0.0002$	$\mathbf{25.0} \pm 0.0$	$\mathbf{99.5} \pm 0.1$	$\mathbf{0.0063} \pm 0.0007$

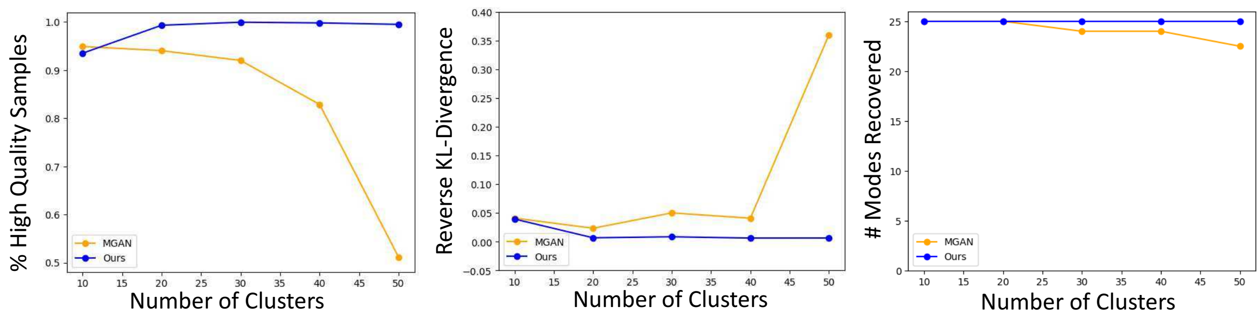


Figure 3: The dependence of our method and MGAN on the choice of number of clusters k for 2D-grid (mixture of 25 Gaussians) experiment. MGAN is sensitive to the value k and performance degrades if k is too large. Our method is generally more stable and scales well with k.

compare the reverse KL divergence between the generated distribution and the real distribution, we assign each point to its closest mean in Euclidean distance, and compare the empirical distributions [25].

Our results are displayed in Table 1. We compare our method to PacGAN [25] for various packing factors, on PacGAN's mother architecture. We observe that our method generates both higher quality and more diverse samples than competing unsupervised methods. The difference in quality can be observed in Table 1. We also see that the success of our method is not solely due to the addition of class conditioning, as the conditional architecture with random labels still fails to capture diversity and quality.

Our online clustering variant, with $k = 50$ on the 2D-grid dataset is able to cover 25 modes with 99.1% high quality samples and a reverse KL of 0.00122 and on the 2D-ring dataset, is able to cover 8 modes with 99.9% high quality samples and a reverse KL of 0.00349.

We also compare our method's sensitivity regarding the number of clusters k to existing clustering-based mode collapse methods [15]. For k chosen to be larger than the ground truth number of modes, we observe that our method levels off and generates both diverse and high-quality samples. For k chosen to be smaller than the ground truth number of modes, our results worsen. Figure 3 plots the sensitivity of k on the 2D-grid dataset, and the same trend holds for the 2D-ring dataset. Therefore, we lean towards using a larger number of clusters to ensure coverage of all modes.

MGAN [15] learns a mixture of k generators that can be compared to our k-way conditional generator. In Table 2, we see that MGAN performs worse as k increases. We hypothesize that when k is large, multiple generators must contribute to a single mode, and MGAN's auxiliary classification loss, which encourages each generator's output to be distinguishable, makes it harder for the generators to cover a mode collaboratively. On the other hand, our method scales well with k, because it dynamically updates cluster weights, and does not explicitly require the conditioned generators to output distinct distributions.

We run both our method and MGAN with varying k values on the 2D-grid dataset. The results summarized in Figure 3 confirm our hypothesis that MGAN is sensitive to the choice of k, while our method is more stable and scales well with k. In our arXiv version, we further present results showing that our method is much less sensitive to the variance of each Gaussian than MGAN.

4.3. Stacked-MNIST and CIFAR-10 Experiments

The Stacked-MNIST dataset [24, 38, 25] is produced by stacking three randomly sampled MNIST digits into an RGB image, one per channel, generating 10^3 modes with high probability. To calculate our reverse KL metric, we use pre-trained MNIST and CIFAR-10 classifiers to classify and count the occurrences of each mode. For these experiments, we use $k = 100$. We also compare our method against an online variant, as well as a variant where we start with

Table 2: Number of modes recovered, reverse KL divergence, and Inception Score (IS) metrics for Stacked MNIST and CIFAR-10 experiments. Results are averaged over five trials, with standard error reported. Results of PacGAN on Stacked MNIST are taken from [25]. For CIFAR-10, all methods recover all 10 modes.

	Stacked MNIST		CIFAR-10		
	Modes (Max 1000) ↑	Reverse KL ↓	FID ↓	IS ↑	Reverse KL ↓
GAN [13]	133.4 ± 17.70	2.97 ± 0.216	31.90 ± 1.15	6.93 ± 0.108	0.0132 ± 0.0015
PacGAN2 [25]	$\mathbf{1000.0} \pm 0.00$	$\mathbf{0.06} \pm 0.003$	31.71 ± 1.17	7.03 ± 0.065	0.0124 ± 0.0012
PacGAN3 [25]	$\mathbf{1000.0} \pm 0.00$	$\mathbf{0.06} \pm 0.003$	32.66 ± 1.93	6.96 ± 0.122	0.0123 ± 0.0012
PacGAN4 [25]	$\mathbf{1000.0} \pm 0.00$	0.07 ± 0.005	34.96 ± 1.30	6.99 ± 0.094	0.0149 ± 0.0006
Logo-GAN-AE [36]	$\mathbf{1000.0} \pm 0.00$	0.09 ± 0.005	32.49 ± 1.37	7.05 ± 0.073	0.0106 ± 0.0005
Logo-GAN-RC [36]	$\mathbf{1000.0} \pm 0.00$	0.08 ± 0.006	30.91 ± 1.30	7.04 ± 0.054	0.0103 ± 0.0004
Random Labels	240.0 ± 12.02	2.90 ± 0.192	31.14 ± 1.66	6.64 ± 0.135	0.0157 ± 0.0027
Online Clustering	995.8 ± 0.86	0.17 ± 0.027	32.45 ± 0.44	6.68 ± 0.068	0.0158 ± 0.0016
Ours + Supervised Init.	$\mathbf{1000.0} \pm 0.00$	0.08 ± 0.014	21.14 ± 1.59	7.61 ± 0.077	0.0017 ± 0.0002
Our Method	$\mathbf{1000.0} \pm 0.00$	0.08 ± 0.009	$\mathbf{18.70} \pm 1.28$	$\mathbf{7.79} \pm 0.033$	$\mathbf{0.0015} \pm 0.0002$
Class-conditional GAN [30]	$\mathbf{1000.0} \pm 0.00$	0.08 ± 0.003	30.03 ± 1.58	7.44 ± 0.092	0.0076 ± 0.0006

Table 3: Fréchet Inception Distance (FID), Inception Score (IS), and reverse KL divergence metrics for Places365 and ImageNet experiments. Our method improves in both quality and diversity over previous methods but still fails to reach the quality of fully-supervised class conditional ones.

	Places365		ImageNet	
	FID ↓	IS ↑	FID ↓	IS ↑
GAN [13]	14.21	8.7122	54.17	14.0074
PacGAN2 [25]	18.02	8.5765	57.51	13.5052
PacGAN3 [25]	22.00	8.5637	66.97	12.3420
MGAN [15]	15.78	8.4141	58.88	13.2196
RotNet Feature Clustering	14.88	8.5436	53.75	13.7616
Logo-GAN-AE [36]	14.51	8.1936	50.90	14.4350
Logo-GAN-RC [36]	8.66	10.5474	38.41	18.8583
Random Labels	14.20	8.8154	56.03	14.1666
Our Method	9.76	8.8186	41.76	14.9620
Class Conditional GAN [30]	8.12	10.9738	35.14	24.0382

manual class labels as the first clustering. Details of these variants are described in our arXiv version.

Our results are shown in Table 2. On both datasets, we are able to achieve large gains in diversity, significantly improve Fréchet Inception Distance (FID) and Inception Score (IS) over baselines and even the supervised baselines (e.g., Class-conditional GAN). Interestingly, starting with supervised class labels seems to slightly degrade performance compared to starting with clusters from random discriminator features.

We find that our cluster initialization step is necessary: without it, on CIFAR, we obtain an FID score of 19.85 and Inception Score of 7.41. Furthermore, our cluster rematching step is necessary: without it, on CIFAR, we obtain an FID score of 20.42 and Inception Score of 7.31.

4.4. Large-Scale Image Datasets Experiments

Lastly, we measure the quality and diversity of images for GANs trained on large-scale datasets Places365 ImageNet. We trained using all 1.2 million ImageNet challenge images across all 1000 classes and all 1.8 million Places365 images across 365 classes. No class labels were revealed to the model. For both datasets, we choose $k = 100$.

Across datasets, our method improves the quality of generated images in terms of FID, IS, and reverse KL divergence, outperforming all baselines in most cases.

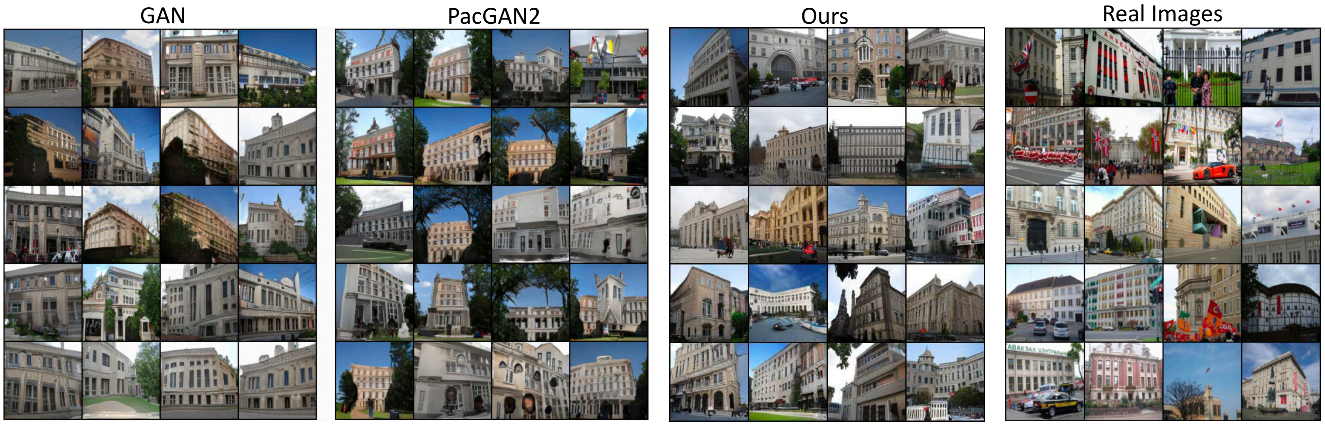
Figure 4 shows samples of the generated output, comparing our model to a vanilla GAN and a PacGAN model and a sample of training images. To highlight differences in diversity between the different GAN outputs, Figure 4a shows a sample of 20 images from each method taken from a single Places365 class. To predict the class label from the generated output, we classify each image using a standard ResNet18 scene classifier [43], and we show the 20 highest-scoring members of the class from a sample of $50{,}000$ generated images. Similarly, Figure 4b shows samples from ImageNet generators filtered to a single ImageNet class, as classified by a standard ResNet50 classifier [34]. We observed that across classes, vanilla GAN and PacGAN tend to synthesize less diverse samples, repeating many similar images. On the other hand, our method improves diversity significantly and does not produce visually similar images.

4.4.1 Reconstruction of Real Images

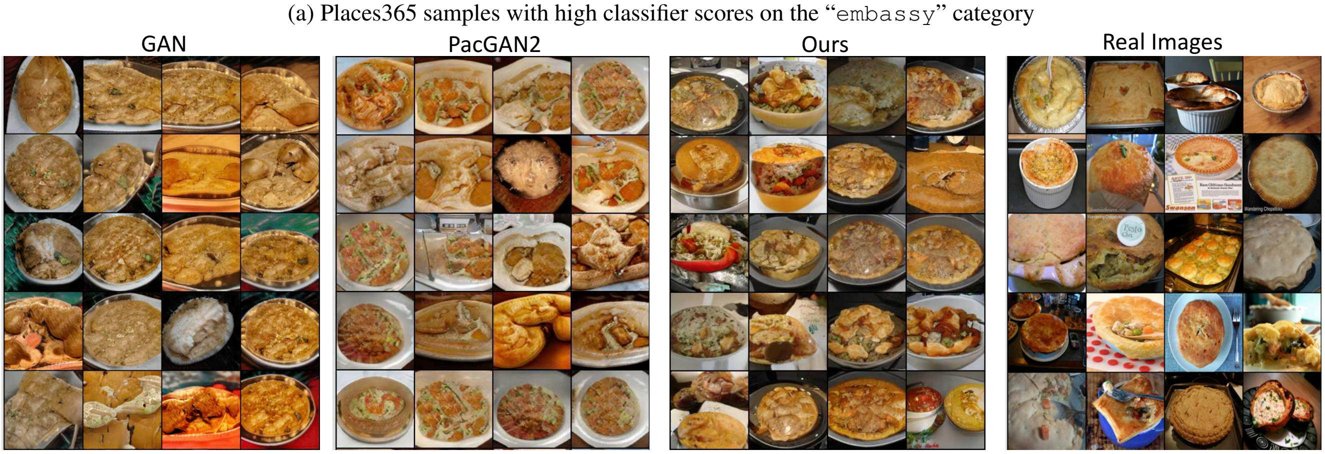
While random samples reveal the positive output of the generator, reconstructions of training set images [44, 6] can be used to visualize the omissions of the generator [3].

Previous GAN inversion methods do not account for class conditioning, so we extend the encoder+optimization hybrid method of [44, 3]. We first train an encoder backbone $F : \mathbf{x} \rightarrow \mathbf{r}$ jointly with a classifier $F_c : \mathbf{r} \rightarrow c$ and a reconstruction network $F_z : \mathbf{r} \rightarrow \mathbf{z}$ to recover the class c and the original $\mathbf{z}$ of a generated image. We then optimize $\mathbf{z}$ to match the pixels of the query image $\mathbf{x}$ as well as encoder features extracted by F:

$$\mathcal{L}_{\text{reconst}}(\mathbf{z}, c) = \|G(\mathbf{z}, c) - \mathbf{x}\|_1 + \lambda_f \|F(G(\mathbf{z}, c)) - F(\mathbf{x})\|_2^2.$$



(a) Places365 samples with high classifier scores on the "`embassy`" category



(b) ImageNet samples with high classifier scores on the "`pot pie`" category

Figure 4: Comparing visual diversity between the samples from a vanilla unconditional GAN, PacGAN2, our method, and real images for specific categories. Images are sorted in decreasing order by classifier confidence on the class from top left to bottom right. Our method successfully increases sample diversity compared to vanilla GAN and PacGAN2. More visualizations are available in the arXiv version.

We set $\lambda_f = 5 \times 10^{-4}$. When initialized using F_z and F_c, this optimization faithfully reconstructs images generated by G, and reconstruction errors of real images reveal cases that G omits. More details regarding image reconstruction can be found in our arXiv version.

To evaluate how well our model can reconstruct the data distribution, we compute the average LPIPS perceptual similarity score [42] between $50{,}000$ ground truth images and their reconstructions. Between two images, a low LPIPS score suggests the reconstructed images are similar to target real images. We find that on Places365, our model is able to better reconstruct the real images, with an average LPIPS score of 0.433, as compared to the baseline score of 0.528.

Figure 5 visually compares reconstructions of Places365 training set images created by our model with those of the same architecture trained without self-conditioning and clustering algorithm. Reconstructions done by previous training iterations are also compared to the final models. Visualizations show distinctive features that can be reconstructed by our model that are not present in the baseline model, including improved forms for cars, buildings, and indoor objects. These features correspond to common visual features that appear in the clusters. Self-conditioned classes include classes of images with cars, buildings, indoor rooms, and scenes from specific viewpoints.

We apply a similar procedure to evaluate reconstructions of a particular cluster, obtaining the average LPIPS score between real images in the cluster and their reconstructed images. Figure 6 shows quantitatively that per cluster LPIPS of our method are noticeably better than those of the unconditional GANs baseline, which is consistent with Figure 5.

4.5. Clustering Metrics

We measure the quality of our clustering through Normalized Mutual Information (NMI) and clustering purity across all experiments in Table 4.

NMI is defined as $\mathsf{NMI}(X,Y) = \frac{2I(X;Y)}{H(X)+H(Y)}$, where I is mutual information and H is entropy. NMI lies in $[0,1]$, and higher NMI suggests higher quality of clustering. Purity is defined as $\frac{1}{N}\sum_c \max_y |\pi_c \cap \pi_y^*|$, where $\{\pi_c\}_{c=1}^k$ is the partition of inferred clusters and $\{\pi_y^*\}_{y=1}^k$ is the partition given by the true classes. Higher purity suggests higher clustering quality. Purity is close to 1 when each cluster has a large majority of points from some true class set.

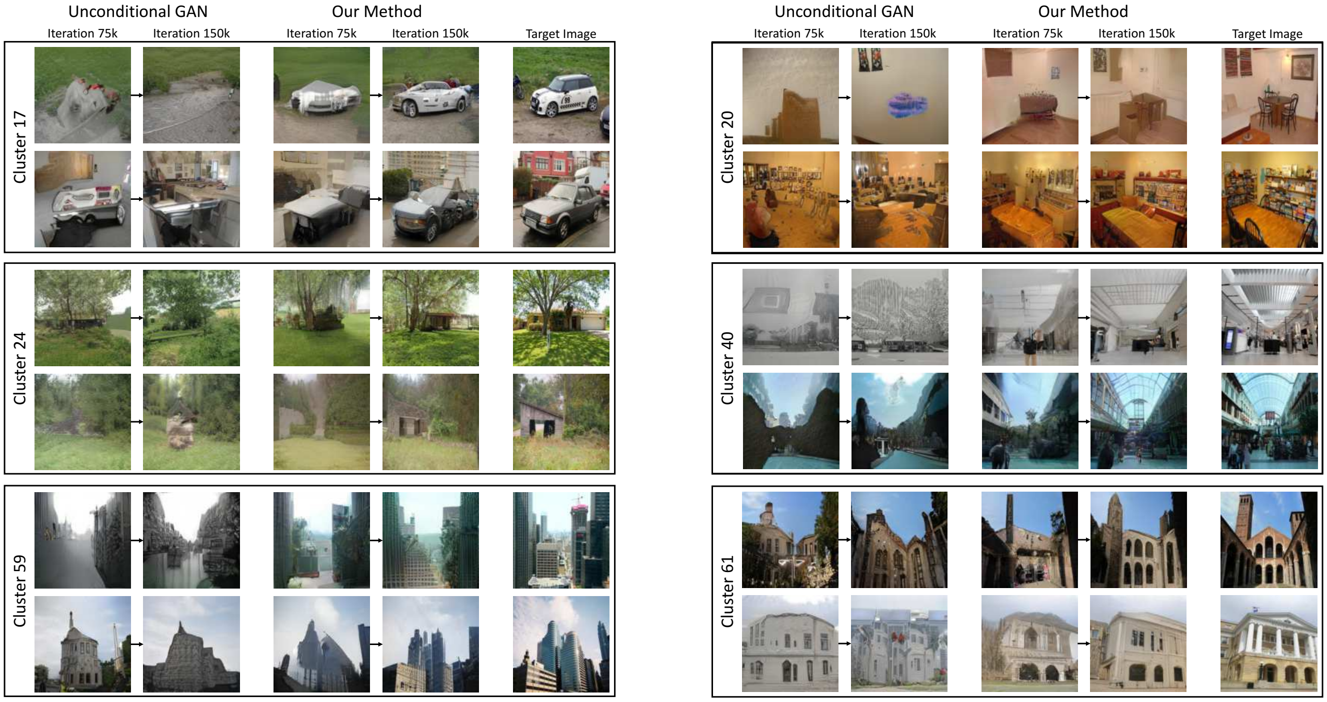


Figure 5: Improvements of GAN reconstructions during training. Each GAN-generated image shown has been optimized to reconstruct a specific training set image from the Places dataset, at right. Selected clusters formed by our algorithm are shown; these clusters include images with cars, skyscrapers, and indoor rooms, as well as scenes sharing specific viewpoints. Reconstructions by generators from an early training iteration of each model are compared with the final trained generators. Conditioning the model results in improved synthesis of clustered features such as wheels, buildings, and indoor objects.

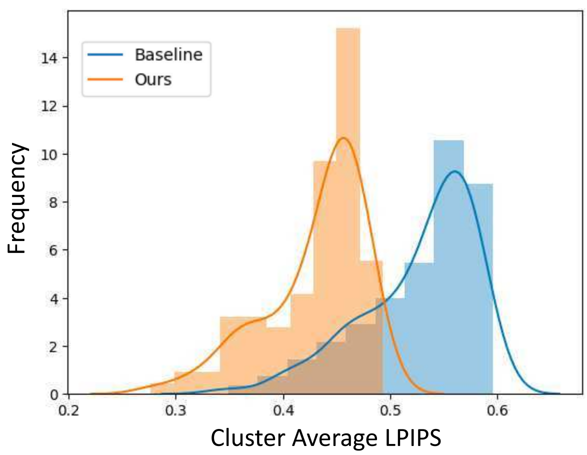


Figure 6: We calculate the average reconstruction LPIPS error for training images in each of the clusters. Overall, our method is able to achieve better reconstruction than a vanilla GAN.

We observe that many of our clusters in large-scale datasets do not correspond directly to true classes, but instead corresponded to object classes. For example, we saw that many clusters corresponded to people and animals, none of which are part of a true class, which is an explanation for low clustering metric scores.

Though our clustering scores are low, they are significantly better than a random clustering. Randomly clustering ImageNet to 100 clusters gives an NMI of 0.0069 and a purity of 0.0176; randomly clustering Places to 100 clusters gives an NMI of 0.0019 and a purity of 0.0137.

Table 4: Normalized Mutual Information (NMI) and purity metrics for the clusters obtained by our method on various datasets.

	2D Ring	2D Grid (25)	Stacked MNIST	CIFAR-10	Places365	ImageNet
NMI	1.0	0.9716	0.3018	0.3326	0.1744	0.1739
Purity	1.0	0.9921	0.1888	0.1173	0.1127	0.1293

5. Conclusion

We have found that when a conditional GAN is trained with clustering labels derived from discriminator features, it is effective at reducing mode collapse, outperforming several previous approaches. We observe that the method continues to perform well when the number of synthesized labels exceeds the number of modes in the data. Furthermore, our method scales well to large-scale datasets, improving Fréchet Inception Distance and Inception Score measures on ImageNet and Places365 generation, and generating images that are qualitatively more diverse than an unconditional GAN.

Acknowledgments. We thank Phillip Isola, Bryan Russell, Richard Zhang, and our anonymous reviewers for helpful comments. We are grateful for the support from the DARPA XAI program FA8750-18-C000, NSF 1524817, NSF BIGDATA 1447476, and a GPU donation from NVIDIA.

References

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 2

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 3

[3] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *ICCV*, 2019. 1, 6

[4] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *NeurIPS*, 1995. 4

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. 1, 2

[6] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017. 6

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 4

[8] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 2015. 2

[9] Hamid Eghbal-zadeh, Werner Zellinger, and Gerhard Widmer. Mixture density generative adversarial networks. In *CVPR*, 2019. 2

[10] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *CVPR*, 2018. 2

[11] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[12] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 1, 2

[13] 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. 1, 2, 4, 5, 6

[14] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017. 2

[15] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *ICLR*, 2018. 2, 5, 6

[16] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 2

[17] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018. 2

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 2

[20] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 2

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[22] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 4

[23] Jogendra Nath Kundu, Maharshi Gor, Dakshit Agrawal, and R Venkatesh Babu. Gan-tree: An incrementally learned hierarchical generative framework for multi-modal data distributions. In *ICCV*, 2019. 2

[24] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 5

[25] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *NeurIPS*, 2018. 2, 4, 5, 6

[26] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In *ICML*, 2019. 1, 2

[27] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *CVPR*, 2017. 2

[28] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018. 2, 4

[29] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *ICLR*, 2017. 2

[30] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1, 2, 6

[31] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 4

[32] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. 2018. 2

[33] Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In *Advances in Neural Information Processing Systems*, pages 1786–1794, 2010. 1

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Pytorch torchvision models. https://pytorch.org/docs/stable/torchvision/models.html, 2019. Accessed 2019-05-23. 6

[35] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2, 4

[36] Alexander Sage, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Logo synthesis and manipulation with clustered generative adversarial networks. In *CVPR*, 2018. 2, 6

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, 2016. 1, 2

[38] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NeurIPS*, 2017. 2, 4, 5

[39] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 1

[40] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 2

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

[42] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[43] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014. 4, 6

[44] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 6

[45] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2