

MIT Open Access Articles

Diverse Image Generation via Self-Conditioned GANs

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Liu, Steven, Wang, Tongzhou, Bau, David, Zhu, Jun-Yan and Torralba, Antonio. 2020. "Diverse Image Generation via Self-Conditioned GANs." Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

As Published: 10.1109/CVPR42600.2020.01429

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: <https://hdl.handle.net/1721.1/137599>

Version: Original manuscript: author's manuscript prior to formal peer review

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike



Diverse Image Generation via Self-Conditioned GANs

Steven Liu¹ Tongzhou Wang¹ David Bau¹ Jun-Yan Zhu² Antonio Torralba¹
¹MIT CSAIL ²Adobe Research



Figure 1: Our proposed self-conditioned 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) [15, 6, 24], 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 [37, 33]. 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 [14]. Both empirical and theoretical studies have shown strong evidence that real data has a highly multi-modal distribution [40, 46]. 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 [44], or omitting object classes such as people and cars within

synthesized scenes [4]. 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 [?], Fréchet Segmentation Distance [4], and Inception Score [44] for both ImageNet and Places365, compared to previous unsupervised methods.

Our [code](#) and models are available on our [website](#).

2. Related Work

Generative Adversarial Networks (GANs). Since the introduction of GANs [15], many variants have been proposed [37, 10, 42, 44, 1, 34, 16, 35], 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 [47, 22, 48, 54, 20, 19]. GANs excel at synthesizing photorealistic images for a specific class of images such as faces and cars [23, 24]. 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 [6]. To reduce the cost of manual annotation, a recent work [33] presents a semi-supervised method based on RotNet [13], 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 [39] proposes to obtain good clustering using GANs, while we aim to achieve realistic and diverse generation.

Mode collapse. Although GANs are formulated as a min-max 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 [36]. Mode collapse does occur in practice, and it is one of the

fundamental challenges for GANs training [14, 44, 25].

Several solutions to mode collapse have been proposed, including amending the adversarial loss to look ahead several moves (e.g., Unrolled GAN [36]), jointly training an encoder to recover latent distributions (e.g., VEEGAN [45]), adding an auxiliary loss to prevent the catastrophic forgetting [8], packing the discriminator with sets of points instead of singletons [44, 31] and training a mixture of generators with an auxiliary diversity objective [18, 12]. 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 [33, 43] or pre-trained classifiers [43]. In contrast, 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 [11]. GAN-Tree uses clustering to split a GAN into a tree hierarchy of GANs for better mode coverage [29]. 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.

Specifically, to train a GAN to imitate a target distribution p_{real} , we partition the data set into k clusters $\{\pi_1, \dots, \pi_k\}$ 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

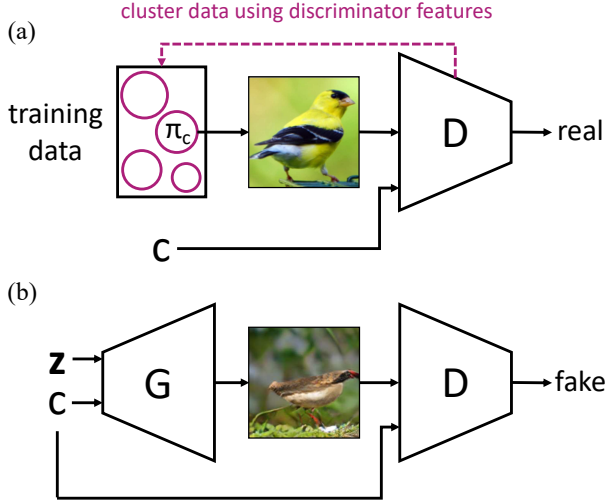


Figure 2: We learn a discriminator D and a generator G that are both conditioned on the automatically discovered cluster c . (a) For a specific c , the discriminator D must learn to recognize real images sampled from the cluster π_c of the dataset, and distinguish them from (b) fake images synthesized 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.

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:

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

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 $G(\mathbf{z}, c)$ that look similar to the real images of cluster c for $\mathbf{z} \sim \mathcal{N}(0, I)$, while $D(\cdot, c)$ tries to distinguish between such

Algorithm 1 Self-Conditioned GAN Training

Initialize generator G and discriminator D
Partition dataset into k sets $\{\pi_1, \dots, \pi_k\}$ 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 $\{1, \dots, m\}$ **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 $\min_G \max_D \mathcal{L}_{\text{GAN}}$ on minibatch $\{(c^{(j)}, x^{(j)}, z^{(j)})\}_j$. \triangleright Eqn. (3)
end for
// Clustering to obtain new partitions
Cluster on D_f outputs of a subset of training set to identify a new partition $\{\pi_c^{\text{new}}\}$ into k sets, using previous centroids as initialization. \triangleright Eqn. (2)
Find the matching $\rho(\cdot)$ between $\{\pi_c^{\text{new}}\}_c$ and $\{\pi_c\}_c$ that minimizes $\mathcal{L}_{\text{match}}$. \triangleright Eqn. (3)
Update all $\pi_c \leftarrow \pi_{\rho(c)}^{\text{new}}$.
end for

generated images and real images of cluster c . They are jointly optimized in the following minimax fashion:

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

When under the condition c , the discriminator is encouraged to give low score for any sample that is not from cluster c because $p_{\text{real}}(\mathbf{x} | c) = 0$ 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 [32] to obtain a new partition into k clusters $\{\pi_c\}_{c=1}^k$ according to the D_f output space, approximately optimizing

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

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 [2]. For subsequent reclustering, we initialize with the means induced by the previous clustering. That is, if $\{\pi_c^{\text{old}}\}_{c=1}^k$ denotes the old cluster

means and $\{\mu_c^{\text{init}}\}_{c=1}^k$ denotes the k -means initialization to compute the new clustering, we take

$$\mu_c^{\text{init}} = \frac{1}{|\pi_c^{\text{old}}|} \sum_{x \in \pi_c^{\text{old}}} D_f^{\text{new}}(x), \quad \forall c, \quad (2)$$

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 $\{\pi_c^{\text{new}}\}_{c=1}^k$ to the old clusters $\{\pi_c^{\text{old}}\}_{c=1}^k$ 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 $\pi_{c'}^{\text{old}}$ is taken as $|\pi_{c'}^{\text{old}} \setminus \pi_c^{\text{new}}|$, the number of samples missing in the new partition. Therefore, we aim to find a permutation $\rho: [k] \rightarrow [k]$ that minimizes the objective:

$$\mathcal{L}_{\text{match}}(\rho) = \sum_c |\pi_c^{\text{old}} \setminus \pi_{\rho(c)}^{\text{new}}|. \quad (3)$$

For a given new partitioning from k -means, we solve this matching using the classic Hungarian min-cost matching algorithm [28], and obtain the new clusters to be used for GAN training in future epochs. Algorithm 1 summarizes the entire training method.

Online clustering variant. We have also experimented with online k -means based on gradient descent [5], where we updated the cluster centers and membership using Equation (1) 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 perform an ablation studies regarding clustering initialization, online vs. batch clustering, and the clustering matching method.

4. Experiments

4.1. Experiment Details

Network architecture. To condition the generator, we learn a unit norm embedding vector for each class, which is fed with the latent input into the first layer of the generator. To condition the discriminator, we let the discriminator output a k -dimensional vector where k is the number of clusters, and mask the output according to the input image’s class label. Across all experiments, we use this method to add conditioning to unconditional backbone networks.

For experiments on synthetic data, our unconditional generator and discriminator adopt the structure proposed in PacGAN [31]. We use a 32-dimensional embedding.

For experiments on Stacked-MNIST, we use the DCGAN architecture [42], following prior work [31]. For experiments

on CIFAR-10 [27], we use the DCGAN architecture, following SN-GANs [38]. For experiments on Places365 [51] and ImageNet [9], we adopt the conditional architecture proposed by Mescheder *et al.* [35], and our unconditional GAN baseline removes all conditioning on the input label. For these four datasets, we use a 256-dimensional embedding.

Clustering details. By default, we use $k = 100$ clusters. We recluster every 10,000 iterations for experiments on synthetic data, every 25,000 iterations for Stacked-MNIST and CIFAR-10 experiments, and every 75,000 iterations for ImageNet and Places365 experiments. The details of the online clustering variant are described in Appendix A.

Training details. For experiments on synthetic datasets, CIFAR-10, and Stacked-MNIST, we train on the standard GAN loss proposed by Goodfellow *et al.* [15]. For experiments on Places365 and ImageNet, our loss function is the vanilla loss function proposed by Goodfellow *et al.* [15] with the R_1 regularization as proposed by Mescheder *et al.* [35]. We find that our method works better with a small regularization weight $\gamma = 0.1$ instead of the default $\gamma = 10$. We explore this choice of hyperparameter in Appendix C.5.

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 $i \in \{0, \dots, 7\}$. The 2D-grid dataset is a mixture of 25 2D-Gaussians, each with means $(2i - 4, 2j - 4)$ and variance 0.0025, for $i, j \in \{0, \dots, 4\}$.

We follow the metrics used in prior work [45, 31]. A generated point is deemed high-quality if it is within three standard deviations from some mean [45]. The number of modes covered by a generator is the number of means that have at least one corresponding high-quality point. To 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 [31].

Our results are displayed in Table 1. We observe that our method generates both higher quality and more diverse samples than competing unsupervised methods. 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, where we update the cluster means and memberships in an online fashion, is also able to perform well, achieving almost perfect performance.

4.3. Stacked-MNIST and CIFAR-10 Experiments

The Stacked-MNIST dataset [30, 45, 31] is produced by stacking three randomly sampled MNIST digits into an RGB image, one per channel, generating 10^3 modes with high

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) \uparrow	% \uparrow	Reverse KL \downarrow	Modes (Max 25) \uparrow	% \uparrow	Reverse KL \downarrow
GAN [15]	6.3 \pm 0.5	98.2 \pm 0.2	0.45 \pm 0.09	17.3 \pm 0.8	94.8 \pm 0.7	0.70 \pm 0.07
PacGAN2 [31]	7.9 \pm 0.1	95.6 \pm 2.0	0.07 \pm 0.03	23.8 \pm 0.7	91.3 \pm 0.8	0.13 \pm 0.04
PacGAN3 [31]	7.8 \pm 0.1	97.7 \pm 0.3	0.10 \pm 0.02	24.6 \pm 0.4	94.2 \pm 0.4	0.06 \pm 0.02
PacGAN4 [31]	7.8 \pm 0.1	95.9 \pm 1.4	0.07 \pm 0.02	24.8 \pm 0.2	93.6 \pm 0.6	0.04 \pm 0.01
Random Labels ($k = 50$)	7.9 \pm 0.1	96.3 \pm 1.1	0.07 \pm 0.02	16.0 \pm 1.0	90.6 \pm 1.6	0.57 \pm 0.07
MGAN ($k = 50$) [18]	8.0 \pm 0.0	71.7 \pm 1.3	0.0073 \pm 0.0014	25.0 \pm 0.0	40.7 \pm 1.2	0.0473 \pm 0.0047
Ours + Online Clustering ($k = 50$)	8.0 \pm 0.0	99.7 \pm 0.1	0.0014 \pm 0.0001	25.0 \pm 0.0	99.7 \pm 0.1	0.0057 \pm 0.0004
Ours ($k = 50$)	8.0 \pm 0.0	99.5 \pm 0.3	0.0014 \pm 0.0002	25.0 \pm 0.0	99.5 \pm 0.1	0.0063 \pm 0.0007

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 [31]. For CIFAR-10, all methods recover all 10 modes.

	Stacked MNIST		CIFAR-10		
	Modes (Max 1000) \uparrow	Reverse KL \downarrow	FID \downarrow	IS \uparrow	Reverse KL \downarrow
GAN [15]	133.4 \pm 17.70	2.97 \pm 0.216	28.08 \pm 0.47	6.98 \pm 0.062	0.0150 \pm 0.0026
PacGAN2 [31]	1000.0 \pm 0.00	0.06 \pm 0.003	27.97 \pm 0.63	7.12 \pm 0.062	0.0126 \pm 0.0012
PacGAN3 [31]	1000.0 \pm 0.00	0.06 \pm 0.003	32.55 \pm 0.92	6.77 \pm 0.064	0.0109 \pm 0.0011
PacGAN4 [31]	1000.0 \pm 0.00	0.07 \pm 0.005	34.16 \pm 0.94	6.77 \pm 0.079	0.0150 \pm 0.0005
Logo-GAN-AE [43]	1000.0 \pm 0.00	0.09 \pm 0.005	32.49 \pm 1.37	7.05 \pm 0.073	0.0106 \pm 0.0005
Random Labels	240.0 \pm 12.02	2.90 \pm 0.192	29.04 \pm 0.76	6.97 \pm 0.062	0.0100 \pm 0.0010
Ours + Online Clustering	995.8 \pm 0.86	0.17 \pm 0.027	31.56 \pm 0.48	6.82 \pm 0.112	0.0178 \pm 0.0029
Ours	1000.0 \pm 0.00	0.08 \pm 0.009	18.03 \pm 0.55	7.72 \pm 0.034	0.0015 \pm 0.0004
Logo-GAN-RC [43]	1000.0 \pm 0.00	0.08 \pm 0.006	28.83 \pm 0.43	7.12 \pm 0.047	0.0091 \pm 0.0001
Class-conditional GAN [37]	1000.0 \pm 0.00	0.08 \pm 0.003	23.56 \pm 2.24	7.44 \pm 0.080	0.0019 \pm 0.0001

Table 3: Fréchet Inception Distance (FID) and Inception Score (IS) metrics for varying k on CIFAR-10. Our method performs well for a wide range of k , but does worse for very small k or very large k . Results are averaged over five trials, with standard error reported.

	CIFAR-10	
	FID \downarrow	IS \uparrow
GAN [15]	28.08 \pm 0.47	6.98 \pm 0.062
$k = 10$	32.55 \pm 2.58	7.03 \pm 0.158
$k = 25$	19.21 \pm 0.73	7.79 \pm 0.040
$k = 100$ (default)	18.03 \pm 0.55	7.72 \pm 0.034
$k = 250$	18.49 \pm 0.53	7.74 \pm 0.053
$k = 1000$	20.76 \pm 0.26	7.48 \pm 0.028
Class Conditional GAN [37]	23.56 \pm 2.24	7.44 \pm 0.080

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 with our online clustering variant.

Our results are shown in Table 2. On both datasets, we achieve large gains in diversity, significantly improving Fréchet Inception Distance (FID) and Inception Score (IS) over even supervised class-conditional GANs.

We also conduct experiments to test how sensitive our

method is to the choice of the number of clusters k . The results with varying k values are shown in Table 3. We observe that our method is robust to the choice of k , as long as it is not too low or too high. Choosing a k too low reduces to unconditional GAN behavior, while choosing a k too high potentially makes clustering unstable, hence hurting GAN training.

4.4. Large-Scale Image Datasets Experiments

Lastly, we measure the quality and diversity of images for GANs trained on large-scale datasets such as ImageNet and Places365. We train 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 $k = 100$.

In following sections, we present detailed analyses comparing our method with various baselines on realism, diversity, and mode dropping.

4.4.1 Generation Realism and Diversity

Realism metrics. Across datasets, our method improves the quality of generated images in terms of standard quality metrics such as FID and IS. We also evaluate the generated samples with the Fréchet Segmentation Distance (FSD) [4],

Table 4: Fréchet Inception Distance (FID), Fréchet Segmentation Distance (FSD), and Inception Score (IS) 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 ↓	FSD ↑	IS ↑	FID ↓	FSD ↑	IS ↑
GAN [15]	14.21	125.4	8.71	54.17	129.7	14.01
PacGAN2 [31]	18.02	161.1	8.58	57.51	171.9	13.50
PacGAN3 [31]	22.00	221.7	8.56	66.97	190.5	12.34
MGAN [18]	15.78	156.2	8.41	58.88	156.6	13.22
RotNet Feature Clustering	14.88	131.4	8.54	53.75	126.8	13.76
Logo-GAN-AE [43]	14.51	140.2	8.19	50.90	119.7	14.44
Random Labels	14.20	112.4	8.82	56.03	146.3	14.17
Ours	9.56	87.67	8.94	40.30	82.46	15.82
Logo-GAN-RC [43]	8.66	75.51	10.55	38.41	90.03	18.86
Class Conditional GAN [37]	8.12	58.17	10.97	35.14	115.1	24.04

which measures discrepancies between the segmentation statistics of the generated samples and real samples. As shown in Table 4, our method outperforms all the unsupervised baselines tested against on these metrics. Although Logo-GAN-RC outperforms our method, it is based on a supervised pre-trained ImageNet classifier.

Diversity visualizations. 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, for each compared method, we assign 50,000 unconditionally generated images with labels given by standard ResNet50 [17] classifiers trained on respective datasets [51, 41], and visualize images with highest classifier score for certain classes in Figure 4a. We observe 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 produces images with varying color, lighting conditions, and backgrounds.

4.4.2 Quantifying Mode Dropping with Image Reconstruction

While random samples reveal the capacity of the generator, reconstructions of training set images [53, 7] can be used to visualize the omissions (dropped modes) of the generator [4].

Previous GAN inversion methods do not account for class conditioning, despite recent efforts by concurrent work [21]. Here we extend the encoder + optimization hybrid method [53, 4]. 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 both 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

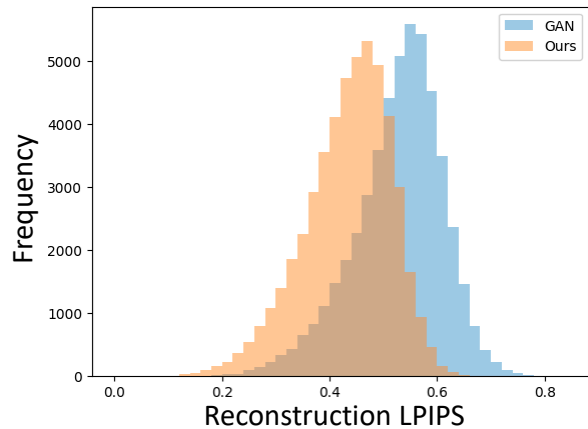


Figure 3: We calculate the reconstruction LPIPS error for 50,000 randomly sampled training images. Overall, our method is able to achieve better reconstruction than a vanilla GAN.

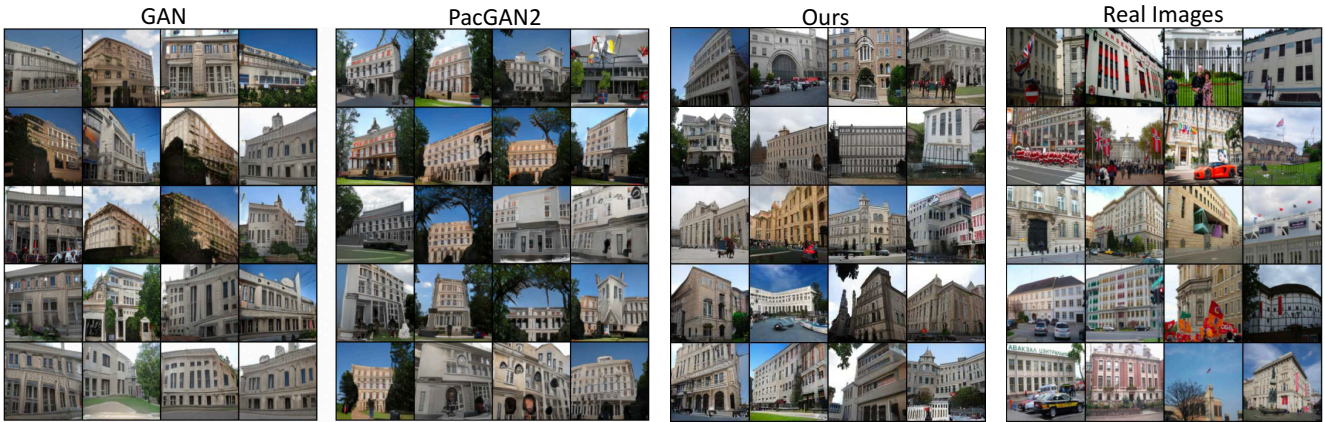
features extracted by F :

$$\mathcal{L}_{\text{rec}}(\mathbf{z}, c) = \|\mathbf{G}(\mathbf{z}, c) - \mathbf{x}\|_1 + \lambda_f \|F(\mathbf{G}(\mathbf{z}, c)) - F(\mathbf{x})\|_2^2. \quad (4)$$

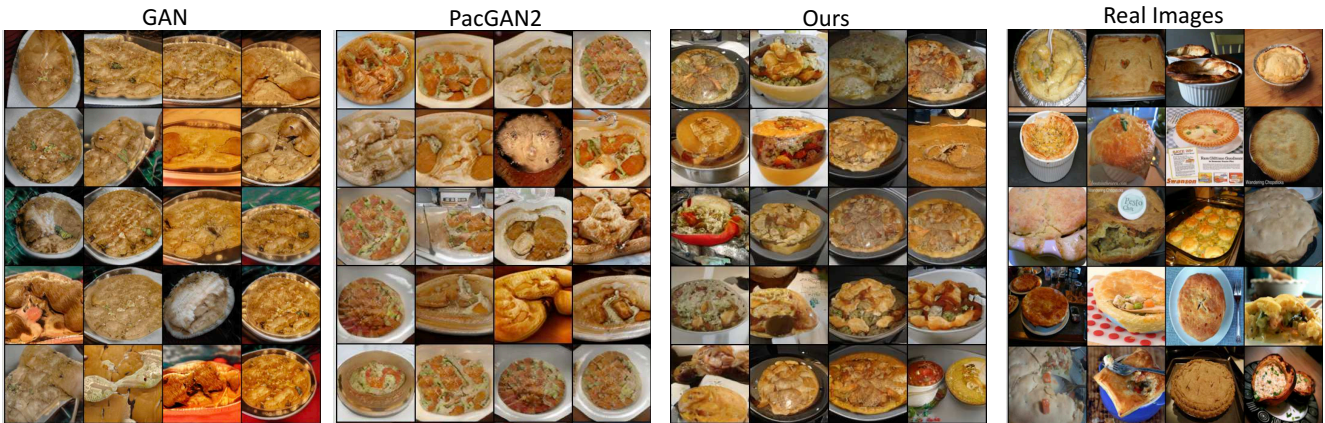
We set $\lambda_f = 5 \times 10^{-4}$. When initialized using F_z and F_c , this optimization faithfully reconstructs images generated by G , and reconstruction errors of real images reveal cases that G omits. More details regarding image reconstruction can be found in Appendix C.3.

To evaluate how well our model can reconstruct the data distribution, we compute the average LPIPS perceptual similarity score [49] 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. A distribution of the LPIPS reconstruction losses across the 50,000 images can be visualized in Figure 3.

Figure 5 visually shows how reconstruction quality evolves as we perform reclustering and training. Visualiza-



(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 Appendix D.

Table 5: Normalized Mutual Information (NMI) and purity metrics for the clusters obtained by our method on various datasets.

	2D Ring	2D Grid	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

tions show our model improves reconstruction during training, and recovers distinctive features, including improved forms for cars, buildings, and indoor objects. On the other hand, the vanilla GAN struggles to reconstruct most of the real images throughout the training.

4.5. Clustering Metrics

We measure the quality of our clustering through Normalized Mutual Information (NMI) and clustering purity across all experiments in Table 5.

NMI is defined as $NMI(X, Y) = \frac{2I(X; Y)}{H(X) + H(Y)}$, 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.

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 see 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.

4.6. Ablation Studies on Cluster Initialization and Matching

We perform ablation studies on the cluster initialization



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 Places365 dataset, at right. Reconstructions by generators from an early training iteration of each model are compared with the final trained generators. Self-conditioning the model results in improved synthesis of clustered features such as wheels, buildings, and indoor objects.

Table 6: Fréchet Inception Distance (FID) and Inception Score (IS) metrics for CIFAR-10 ablations experiments. The full algorithm is not necessary when using R1 regularization, but necessary without regularization. Results are averaged over 5 trials, with standard error reported.

	Without R1		With R1	
	FID ↓	IS ↑	FID ↓	IS ↑
No Matching/Init	25.38 ± 1.41	7.29 ± 0.068	22.70 ± 0.48	7.38 ± 0.062
No Matching	18.91 ± 0.65	7.51 ± 0.073	23.26 ± 0.40	7.27 ± 0.029
No Initialization	19.85 ± 0.65	7.58 ± 0.046	22.02 ± 0.21	7.35 ± 0.024
Full Algorithm	18.03 ± 0.55	7.72 ± 0.034	22.80 ± 0.27	7.32 ± 0.031

and cluster matching used in our full algorithm on CIFAR-10. The results are summarized in Table 6.

We find that these clustering tricks are required for stable clustering and ultimately for GAN quality. The less stable the clustering is, the worse the GAN performs. To quantify cluster stability, we measure the NMI between a given clustering and its previous clustering. We observe that without the two clustering tricks, the clustering is not stable, achieving an NMI of 0.55 compared to the full algorithm’s 0.74.

In contrast, with R1 regularization [35], our method still works well on CIFAR-10 without cluster initialization and matching. We suspect that this is because the GAN regularization loss stabilizes GAN training, preventing cluster instability from hurting GAN training stability. However, our

results on CIFAR-10 worsen when adding R1 regularization. We hypothesize that this is due to discriminator regularization decreasing the quality of the discriminator’s features, which ultimately leads to worse clusters. We investigate this interaction further in Appendix C.5.

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 various standard measures on ImageNet and Places365 generation, and generating images that are qualitatively more diverse than many unconditional GAN methods.

Acknowledgments. We thank Phillip Isola, Bryan Russell, Richard Zhang, and our anonymous reviewers for helpful comments. Tongzhou Wang was supported by the MIT EECS Merrill Lynch Graduate Fellowship. 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] 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] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Zhou Bolei, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *ICLR*, 2019.
- [4] 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.
- [5] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *NeurIPS*, 1995.
- [6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019.
- [7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017.
- [8] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 2019.
- [9] 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.
- [10] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 2015.
- [11] Hamid Eghbal-zadeh, Werner Zellinger, and Gerhard Widmer. Mixture density generative adversarial networks. In *CVPR*, 2019.
- [12] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *CVPR*, 2018.
- [13] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.
- [14] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.
- [15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.
- [16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [18] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. In *ICLR*, 2018.
- [19] 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.
- [20] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018.
- [21] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images to class-conditional generative networks. *arXiv preprint arXiv*, 2020.
- [22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.
- [23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018.
- [24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.
- [25] Mahyar Khayatkhoei, Maneesh K Singh, and Ahmed Elgammal. Disconnected manifold learning for generative adversarial networks. In *NeurIPS*, 2018.
- [26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [28] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [29] 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.
- [30] Yann LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [31] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. In *NeurIPS*, 2018.
- [32] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.
- [33] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In *ICML*, 2019.
- [34] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *CVPR*, 2017.
- [35] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018.
- [36] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. In *ICLR*, 2017.
- [37] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [38] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.
- [39] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *AAAI*, 2019.
- [40] Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In *NeurIPS*, 2010.

- [41] 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.
- [42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.
- [43] Alexander Sage, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Logo synthesis and manipulation with clustered generative adversarial networks. In *CVPR*, 2018.
- [44] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.
- [45] 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.
- [46] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- [47] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016.
- [48] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiao lei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.
- [49] 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.
- [50] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017.
- [51] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014.
- [52] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.
- [53] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.
- [54] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

Appendix A. Additional Experiment details

Compute resource details. For all experiments except ImageNet [9] and Places365 [52] experiments, we used a single Titan Xp GPU. For ImageNet and Places365 experiments, we used four Titan Xp GPUs.

Classifier details. We use a classifier to compute the reverse-KL metric for experiments on synthetic data, Stacked-MNIST, and CIFAR. To classify points for 2D experiments, we classify each point as the label of its nearest mode. To classify samples for Stacked-MNIST experiments, we use the pretrained MNIST classifier on each channel of the generated image. This gives us 3 labels in $0, \dots, 9$, which is used to assign the image a label from $0, \dots, 999$. To classify samples for CIFAR-10, we a pretrained CIFAR-10 classifier. Both the MNIST and CIFAR-10 classifiers are taken from a pretrained model [repository](#).

We also use a classifier to visualize the diversity of the trained models on ImageNet and Places. For ImageNet, we use a pretrained ResNet-50 obtained through the `torchvision` [41] package. For Places, we use a pretrained ResNet-50 released by [50].

Clustering details. To compute cluster centers, for Stacked-MNIST and CIFAR-10 experiments, we cluster a random subset of 25,000 images from the training set, and for ImageNet and Places365 experiments, we cluster a random subset of 50,000 images from the training set. On synthetic datasets, we cluster a random sample of 10,000 data points. While conducting k -means++, we cluster the subset ten times and choose the clustering that obtains the best performance on the clustering objective. We use the outputs of the last hidden layer of the discriminator as features for clustering.

Realism metrics Details. All FIDs, FSDs, and Inception Scores (IS) are reported using 50,000 samples from the fully trained models, with the FID and FSD computed using the training set. No truncation trick is used to sample from the generator.

Reverse KL Details. In all reverse-KL numbers, we report $D_{KL}(p_{\text{fake}} || p_{\text{real}})$, and treat $0 \log 0 = 0$, where we obtain the distributions for a set of samples by using the empirical distribution of classifier predictions.

Online clustering details. As a hyperparameter study, we implement an online variant of our algorithm, where we update our clusters every training iteration. We train a model with a fixed clustering for 25,000 iterations, then apply the online k -means algorithm of [5] using the training mini-batches as the k -means mini-batches. We found that while the online variant was able to do well on easier datasets

such as 2D-grid and Stacked-MNIST, they did not achieve convincing performance on real images.

Logo-GAN details. For Logo-GAN experiments, we infer labels by clustering the outputs of the last layer of a pretrained ResNet50 ImageNet classifier. We train another model where labels are from clustering an autoencoder’s latent space. The autoencoder architecture is identical to an unconditional GAN architecture, except that the latent space of the autoencoder is 512. The autoencoders were trained to optimize a mean squared loss using Adam with a learning rate of 10^{-4} with a batch size of 128, for 200,000 iterations. For both these methods, we cluster a batch of 50,000 outputs into $k = 100$ clusters using the k -means++ algorithm.

MGAN details. For MGAN experiments with synthetic Gaussian data, we use $\beta = 0.125$, following the original paper [18] on similar typed data.

Appendix B. Synthetic Data Experiments

B.1. Comparison with MGAN

Choice of k . We compare our method’s sensitivity regarding the number of clusters k to existing clustering-based mode collapse methods [18].

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 6 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 [18] learns a mixture of k generators that can be compared to our k -way conditional generator. In Figure 6, we see that MGAN performs worse as k increases. We hypothesize that when k is large, multiple generators must contribute to a single mode. Therefore, 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 6 confirm our hypothesis that MGAN is sensitive to the choice of k , while our method is more stable and scales well with k .

Variance of data. We compare our method and MGAN on the 2D-grid mixture of Gaussians on the sensitivity to the variance of each Gaussian. In Figure 7, when given $k = 30$ to cover a mixture of 25 Gaussians, MGAN does well in the

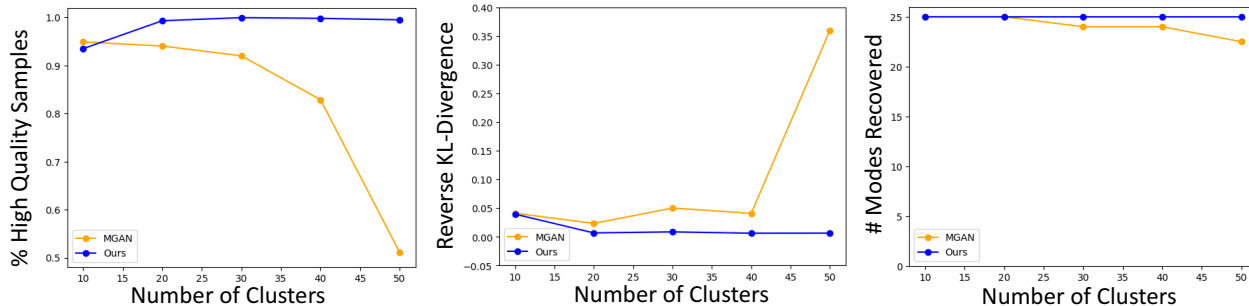


Figure 6: The dependence of our method and MGAN on the choice of number of clusters k for 2D-grid (mixture of 25 Gaussians) experiment. MGAN is sensitive to the value k and performance degrades if k is too large. Our method is generally more stable and scales well with k .

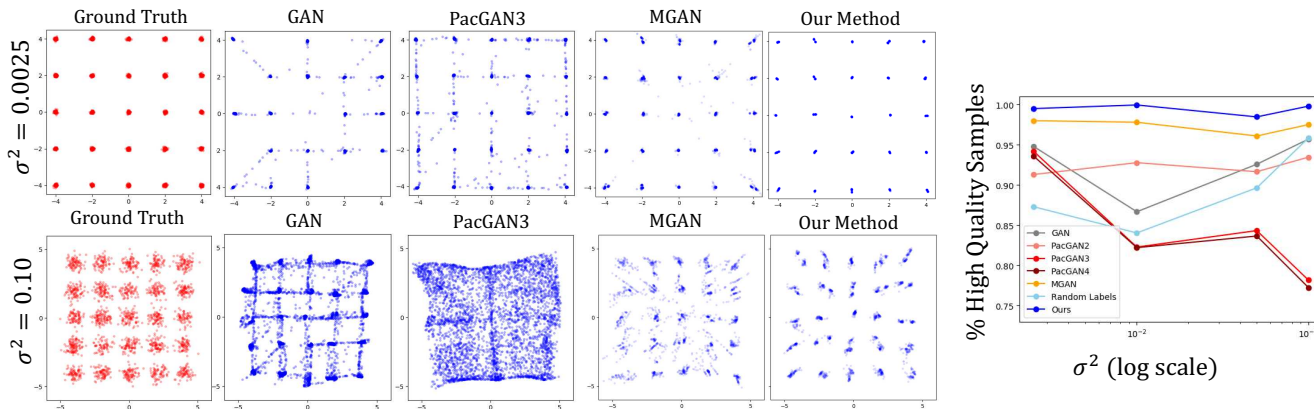


Figure 7: Visual comparison of generated samples on the 2D-grid dataset. Our method successfully covers all modes and generates high quality samples for low variance Gaussians. For high variance Gaussians, PacGAN learns a uniform distribution, while our method generates reasonably high quality samples. In these experiments, we used $k = 30$ for both our methods and MGAN.

large variance $\sigma^2 = 0.1$ experiment, but misses a mode in the small variance $\sigma^2 = 0.0025$ one, as the generators have difficulty working together on a single mode.

Appendix C. Large-Scale Image Datasets

C.1. Hyperparameter Details

For experiments on synthetic datasets, we use 2-dimensional latents, and train for 400 epochs using Adam [26] 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 10^{-4} with $\beta_1 = 0.0$, $\beta_2 = 0.99$. We train for 50 epochs on Stacked-MNIST and 400 epochs on CIFAR-10. The generator embedding is 256-dimensional.

For Places365 and ImageNet experiments, we train our networks from scratch using Adam with a batch size of 128, learning rate of 10^{-4} , $\beta_1 = 0.0$, and $\beta_2 = 0.99$ for 200,000 iterations. The generator embedding is 256-dimensional.

Table 7: Fréchet Inception Distance (FID), Fréchet Segmentation Distance (FSD), and Inception Score (IS) metrics for GANs trained on the combined datasets of Places365 and ImageNet. Our method improves in both quality and diversity over both vanilla unconditional GANs and class-conditional ones.

	Places365 + ImageNet		
	FID ↓	FSD ↓	IS ↑
GAN [15]	36.78	167.1	11.0384
Ours	20.88	68.55	12.0632
Class Conditional GAN [37]	22.05	92.43	14.7734

C.2. Combined ImageNet and Places365 Experiments

We also test our method on the combined ImageNet and Places365 datasets and compare the results to those attained by an unconditional GAN and a class-conditional one. We still use $k = 100$ for our method, but change the regularization on the unconditional GAN to $\gamma = 100$ for convergence. The results are summarized in Table 7. Surprisingly, our method is able to outperform class-conditioned GANs in terms of FID and FSD.

For these experiments, we construct the dataset by treating the classes of ImageNet and Places365 as separate. The combined dataset has roughly 3 million images with 1365 total classes. It is important to note that the classes in this dataset are heavily imbalanced, with the 365 classes from Places365 each containing more images than the 1000 ImageNet classes. Thus, to sample from the class-conditional GAN, we do not sample labels from a uniform distribution, but instead from the distribution of class labels appropriately weighted towards the Places365 classes.

C.3. Reconstruction of Real Images

To train the encoder backbone for the self-conditioned GAN, we follow an analogous procedure to the GAN-seeing work [4], using a similar encoder architecture to GAN-stability’s [35] ResNet2 discriminator architecture. We invert the last seven layers separately in a layerwise fashion, then use these layerwise encoders jointly as initialization for the overall encoder.

Given a fake image x generated from latent z and class c with $G = G_f \circ G_i$ where $G_i: z, c \rightarrow r$, the encoder $E = E_f \circ E_i$ outputs a latent vector \hat{z} and a class embedding \hat{e}_c , where $E_i: x \rightarrow r$ and is optimized for the loss function

$$\mathcal{L}_{\text{enc}} = \|\hat{z} - z\|_2 + \|G(\hat{z}, c) - x\|_2 + \|G_i(z, c) - E_i(x)\|_2 + \mathcal{L}_{\text{classification}}(\hat{e}_c, c) \quad (5)$$

This loss function is identical to the one used in the GAN-seeing work, except with an additional classification loss and an altered reconstruction loss in pixel space.

The classification loss, $\mathcal{L}_{\text{classification}}(\hat{e}_c, c)$, arises from the encoder additionally predicting the class embedding vector of a generated image. To obtain logits from an embedding prediction, we multiply it with the normalized true embedding vectors of the generator. The classification loss is the standard cross-entropy loss between these logits and the true cluster index.

To measure reconstruction loss in pixel space, we take the ground truth embedding vector and predicted latent vector as input to the generator, and measure the mean squared error between the reconstructed image and the original image. We found that using the ground truth embedding vector, as opposed to the predicted embedding vector, improves inversion quality.

To optimize the latent for a specific image, we follow Equation 4, where the class is inferred by taking the index of the largest logit in the model’s latent prediction. We use the output of the seventh layer of the encoder as features for the feature space loss.

C.4. Visualizing Generators with GAN Dissection

We study the number of interpretable units in each learned generator. Previous work [3] has shown that a high number

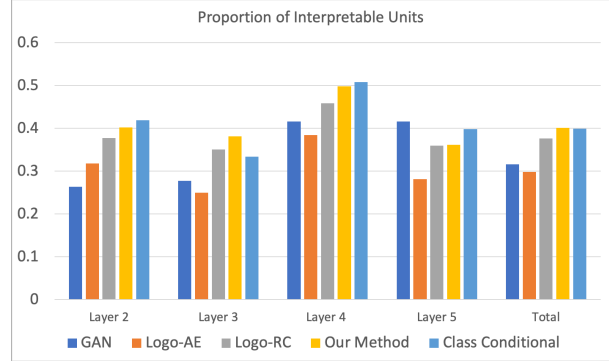


Figure 8: GAN Dissection results. We measure the proportion of interpretable units of generators trained on Places. We dissect the early to mid layers of the generator and count the number of units with IoU greater than 0.04.

of interpretable units is positively correlated with the performance of GANs. Using the GAN Dissection method [3], we analyze each generator and count the number of units in each layer with IoU scores greater than 0.04. As shown in Figure 8, our generator has more interpretable units, on par with class-conditional models.

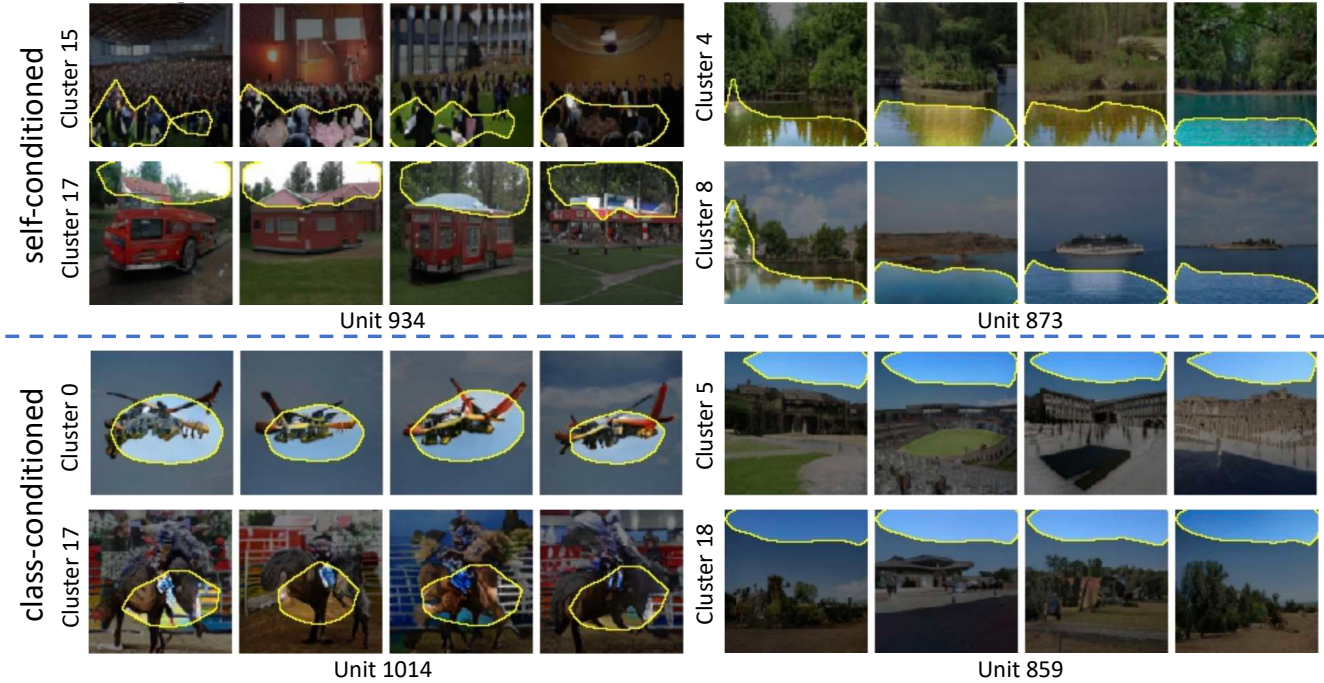
We also visualize the internal mechanism of a general class-conditioned GAN. Using the GAN Dissection method [3], we dissect our generator conditioned on a specific cluster. With this, we are able to visualize each unit’s concept when conditioned on a specific cluster. By monitoring the interpretable units, we can observe how the behavior of each unit changes as we change the conditioned cluster. We again deem a unit interpretable if it has an IoU score of higher than 0.04.

We observe that surprisingly, some units in the generator are responsible for different concepts when conditioning on different clusters. For example, the same units which are responsible for drawing people in one cluster are also responsible for drawing trees in a different cluster. There is also a significant amount of parameter sharing: units are reused to encode the same concept across different conditions. For example, water units and sky units tend to be reused across several conditions. The same phenomena in supervised class-conditional GANs. However, across the 20 conditions tested, we find that the self-conditioned GAN has a higher proportion of units whose concepts change across conditions. For a layer consisting of 1024 units, we found 663 self-conditioned GAN units change meaning across conditions, compared to 562 units changing for a class-conditioned GAN. We can visualize these phenomena for both GANs on Places365 in Figure 9.

C.5. Interaction with discriminator regularization

We find that our method’s performance suffers if regularization is too strong and that our method performs best

Figure 9: Some units of the self-conditioned GAN (top) and a class-conditioned GAN (bottom) trained on Places365 correspond to different concepts when conditioned on different clusters (left), while other units correspond to the same concept across conditions (right).



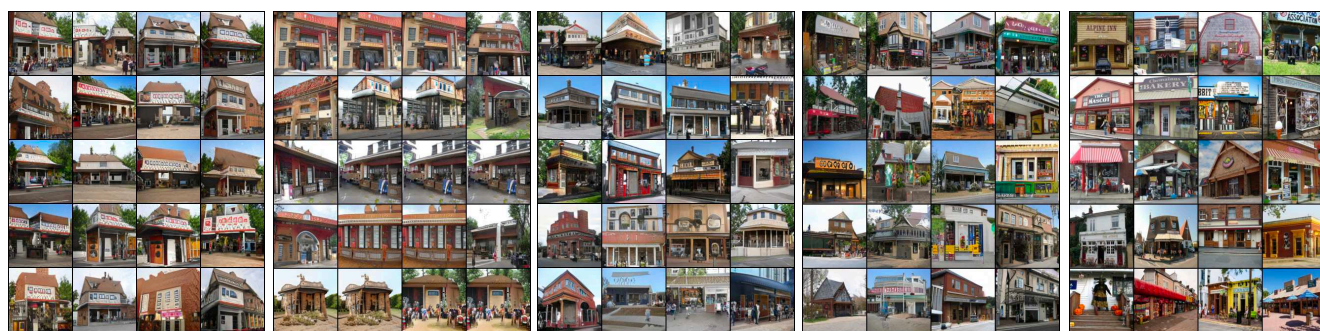
with weak regularization. To test this, we turn off the regularization in the experiments of Section 4.4 and train our method, as well as unconditional and class-conditional baselines, using a smaller architecture (half number of layers compared to the ResNet-based generator we adapt from [35]). Our method is able to match the performance of supervised models on ImageNet and Places365, achieving FID 22.68 on Places365 and FID 55.26 on ImageNet, compared to the class-conditional scores of 29.33 on Places365 and 53.36 on ImageNet and the unconditional scores of 75.09 on Places365 and 87.24 on ImageNet.

We hypothesize that this is due to discriminator regularization leading to worse discriminator features to cluster with. To measure this, we train a linear ImageNet classifier over the last layer features of an R1-regularized ImageNet discriminator and similarly so for an unregularized ImageNet discriminator. Both discriminators in this case are fully class-conditional, meaning that they have seen true class labels throughout the training. We find that the classifier trained on top of the R1-regularized discriminator features severely underperforms the classifier trained on the unregularized discriminator features, achieving 30.7% top-1 accuracy compared to 59.8%. The features chosen were the penultimate layer’s outputs, and the linear classifier was trained using a standard cross-entropy loss with the Adam optimizer. We use a learning rate 10^{-4} with a batch size of 128, $\beta_1 = 0.9$, and $\beta_2 = 0.99$.

Appendix D. Additional Qualitative Results

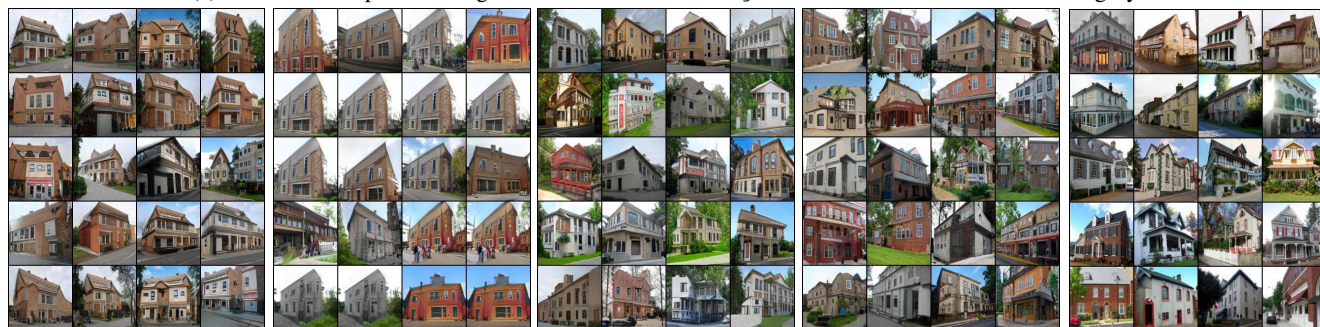
We present more qualitative comparisons regarding the sample diversity of our method compared to various baselines on Places365 and ImageNet. Figure 10 (Places365) and Figure 11 (ImageNet) visualize the sample diversity of various methods, all of which do not use labels.

We also present additional visual results on Places365 and ImageNet. In Figure 12a (Places365) and Figure 12b (ImageNet), we show clusters inferred by our method and corresponding generated samples. More examples can be found on our [website](#).



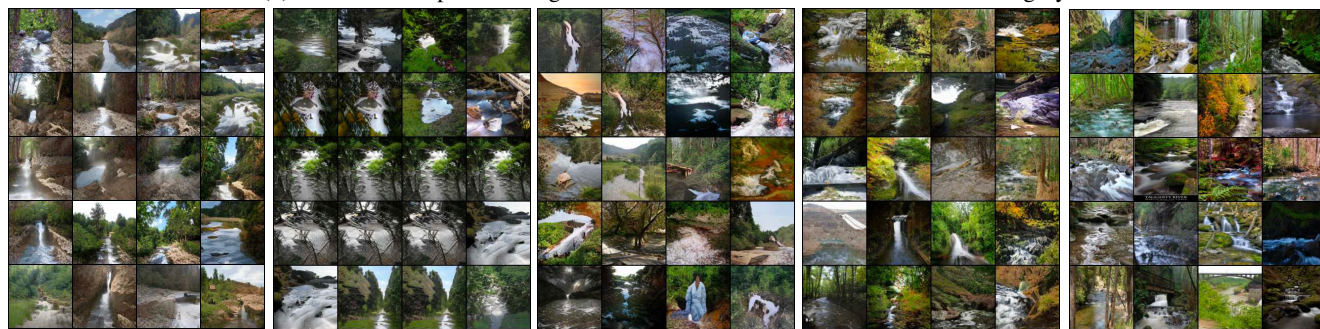
GAN PacGAN2 Logo-AE Ours Real Images

(a) Places365 samples with high classifier scores on the “general store outdoor” category.



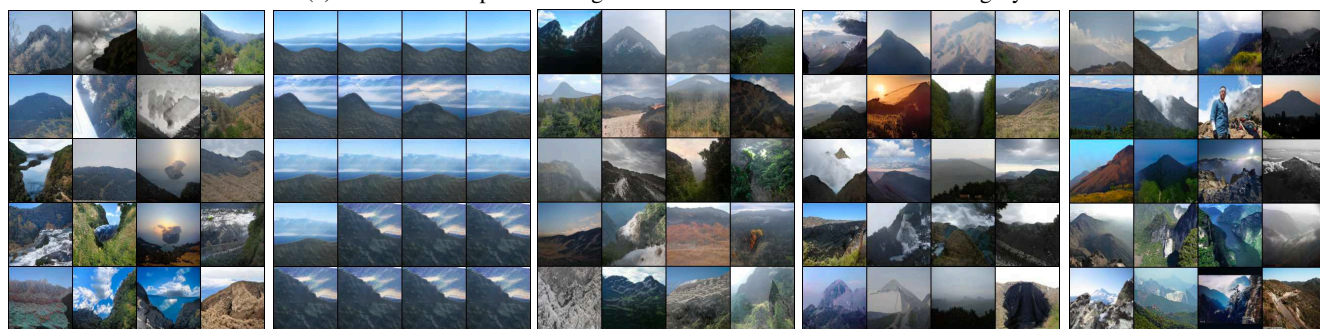
GAN PacGAN2 Logo-AE Ours Real Images

(b) Places365 samples with high classifier scores on the “inn outdoor” category.



GAN PacGAN2 Logo-AE Ours Real Images

(c) Places365 samples with high classifier scores on the “river” category.



GAN PacGAN2 Logo-AE Ours Real Images

(d) Places365 samples with high classifier scores on the “mountain” category.

Figure 10: Places365 [52] samples from an unconditional GAN, PacGAN2 [31], Logo-GAN AE [43], our self-conditioned GAN, and real images. We observe that competing methods exhibit severe mode collapse, frequently producing similar images with a few patterns, whereas our method captures a wider range of diversity. Samples are all sorted in rank order of a classifier score for a single category. Please refer to our [website](#) for additional examples.



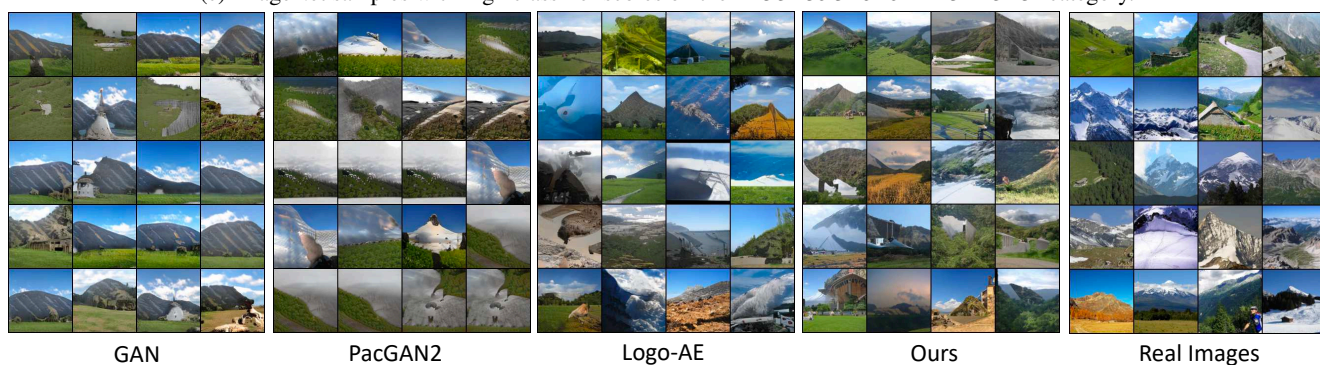
(a) ImageNet samples with high classifier scores on the “library” category.



(b) ImageNet samples with high classifier scores on the “wallet” category.

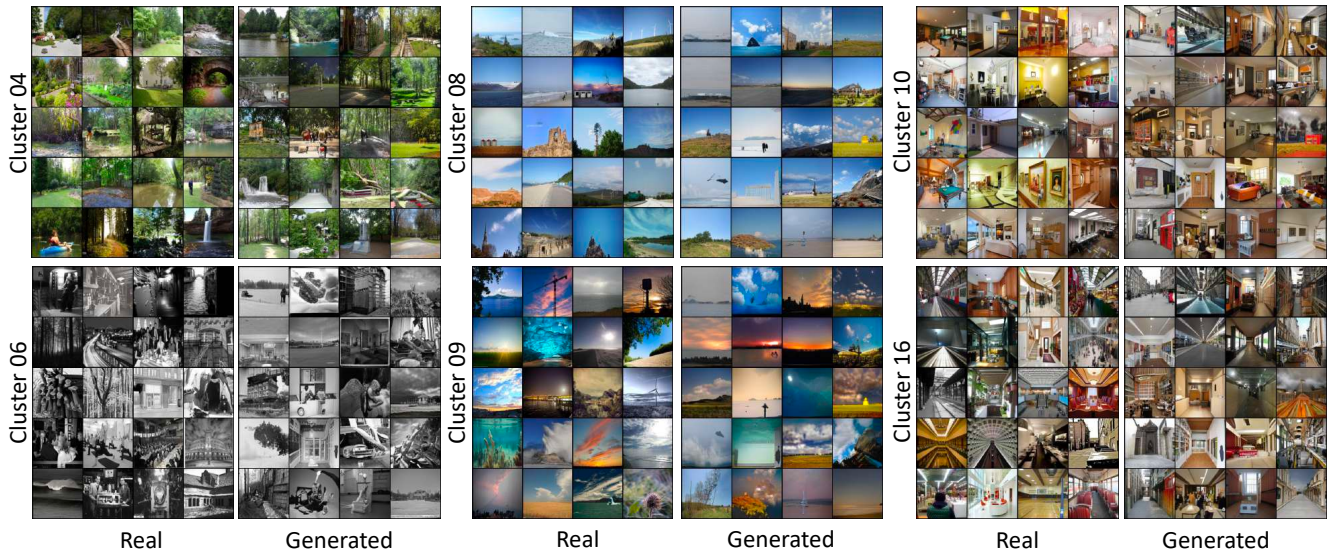


(c) ImageNet samples with high classifier scores on the “recreational vehicle” category.

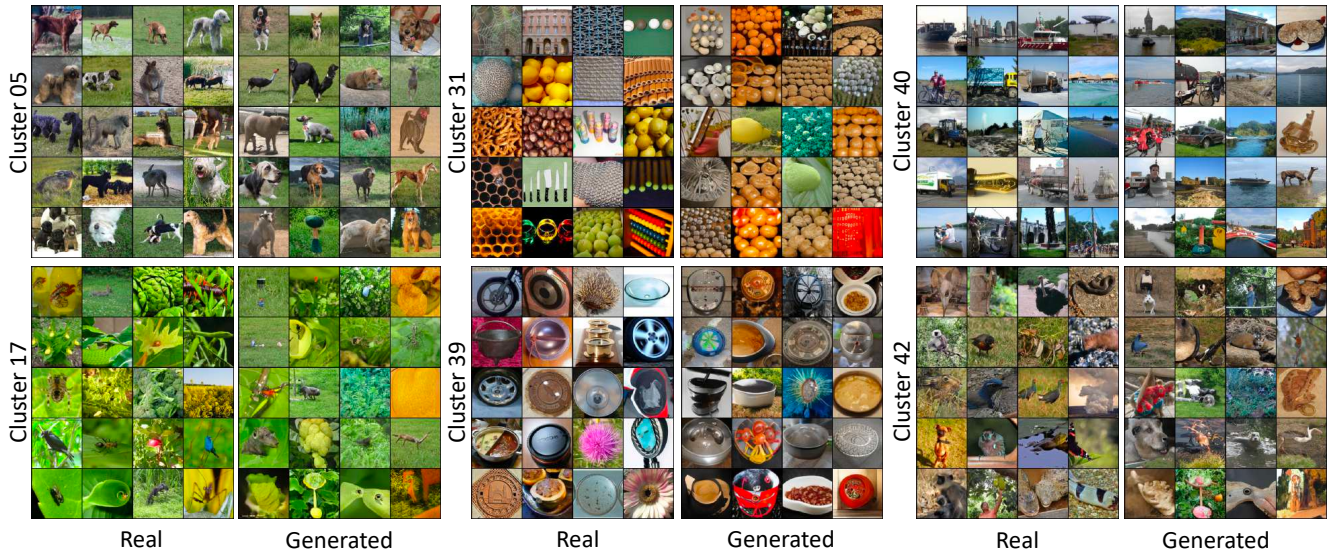


(d) ImageNet samples with high classifier scores on the “alp” category.

Figure 11: ImageNet [9] samples from an unconditional GAN, PacGAN2 [31], Logo-GAN AE [43], our self-conditioned GAN, and real images. We observe that competing methods exhibit severe mode collapse, frequently producing similar images with a few patterns, whereas our method captures a wider range of diversity. Samples are all sorted in rank order of a classifier score for a single category. Please refer to our [website](#) for additional examples.



(a) Inferred clusters and corresponding samples on the Places365 dataset.



(b) Inferred clusters and corresponding samples on the ImageNet dataset.

Figure 12: Additional inferred clusters and generated samples for our method trained on ImageNet and Places365. Please refer to our [website](#) for additional examples.