MINING GANS FOR KNOWLEDGE TRANSFER TO SMALL DOMAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the attractive characteristics of deep neural networks is their ability to transfer knowledge obtained in one domain to other related domains. As a result, high-quality networks can be trained in domains with relatively little training data. This property has been extensively studied for discriminative networks but has received significantly less attention for generative models. Therefore, we investigate various scenarios of knowledge transfer for generative models and propose methods to mine the knowledge that is most beneficial to a specific target domain from a single or multiple pretrained GANs. This is done using a miner network that identifies which part of the generative distribution of the pretrained GAN outputs samples closest to the target domain. We perform experiments on several complex datasets using various GAN architectures (BigGAN, Progressive GAN) and show that the proposed method, called MineGAN, effectively transfers knowledge to small domains, outperforming existing methods. In addition, MineGAN can successfully transfer knowledge from multiple pretrained GANs.

1 INTRODUCTION

Generative adversarial networks (GANs) can learn the complex underlying distribution of image collections (Goodfellow et al., 2014). They have been shown to generate high-quality realistic images (Karras et al., 2017; 2019a; Brock et al., 2019) and are used in many applications including image manipulation (Isola et al., 2017; Zhu et al., 2017), style transferring (Gatys et al., 2016), compression (Tschannen et al., 2018), and colorization (Zhang et al., 2016). Since their successful introduction, they have received much research attention. Specifically, a significant number of studies have focused on improving the architectures (Denton et al., 2015; Karras et al., 2017; Radford et al., 2015), and the stability of training (Arjovsky et al., 2017; Gulrajani et al., 2017; Miyato et al., 2018; Zhang et al., 2018).

It is known that high-quality GANs require a significant amount of training data and time. For example, progressive GANs (Karras et al., 2017) are trained on 30K images and are reported to require a month of training on one NVIDIA Tesla V100. Being able to exploit these high-quality pretrained models, not just to generate the distribution on which they are trained, but also to combine them with other models and adjust them to a target distribution is a desirable objective. For example, it might be desirable to only generate women using a GAN trained to generate men and woman alike. Alternatively, one may want to generating smiling people from two pretrained generative models, one for men and one for women. The focus of this paper is on performing these operations using only a small target set of images, and without access to the large datasets used to pretrain the models.

Transferring knowledge to small domains has been extensively studied for discriminative models (Hinton et al., 2014; Oquab et al., 2014; Romero et al., 2015) and, as a result, high-quality networks can be trained in domains with relatively little training data. However, knowledge transfer for generative models has received significantly less attention. Only recently, Wang et al. (2018) studied finetuning from pretrained generative models and showed that it is beneficial for small domains. Noguchi & Harada (2019) proposed instead to reduce the number of trainable parameters, and only finetune the learnable parameters for the batch normalization (scale and shift) of the generator.

In this paper, we address knowledge transfer by adapting a trained generative model for targeted image generation given a small sample of the target distribution. We introduce the process of *mining*

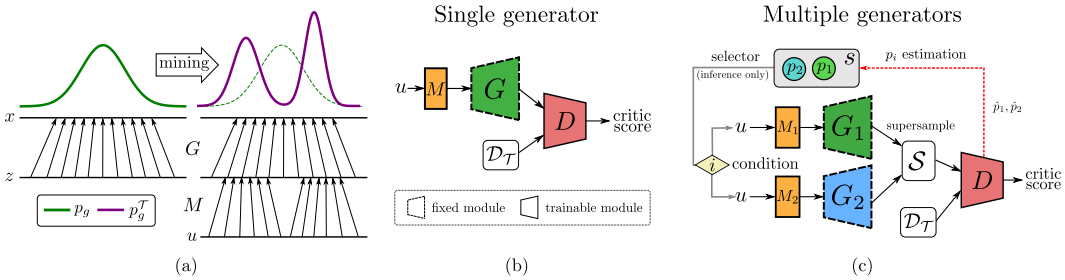


Figure 1: (a) Intuition behind our approach for a simple case. The *mining* operation shifts the prior input distribution towards the most promising regions with respect to given target data $\mathcal{D}_{\mathcal{T}}$. (b) Architecture implementing the proposed *mining* operation on a single GAN. Miner M identifies the relevant regions of the prior distribution so that generated samples are close to the target data $\mathcal{D}_{\mathcal{T}}$. Note that when training the miner the generator remains fixed. (c) Multiple generators. Miners M_1 and M_2 identify subregions of both pretrained generators while selector $\mathcal{S}$ learns which pretrained model is preferable for the inference stage.

of GANs. This is performed by a *mining network* that transfers a multivariate normal distribution into a distribution on the input space of the pretrained GAN in such a way that the generated images resemble those of the target domain. We also extend our mining approach to multiple pretrained GANs, which allows us to aggregate information from multiple sources simultaneously to generate samples akin to the target domain. We show that these networks can be trained by applying a selective backpropagation procedure. To the best of our knowledge, we are the first to investigate transferring knowledge from multiple GANs to a single generative model.

We conduct experiments on multiple datasets, including some high-resolution datasets with high complexity such as LSUN (Yu et al., 2015b), CelebA (Liu et al., 2015) and ImageNet (Krizhevsky et al., 2012).We demonstrate the effects of mining on a single GAN, providing quantitative results as well as insights based on the generated images. We evaluate mining from multiple GANs under several scenarios including different amounts of available training data. The results show that mining outperforms alternative approaches for knowledge transfer for generative models, and that it can successfully transfer knowledge from multiple GANs.

2 GAN FORMULATION

Let $p_{data}(x)$ be a probability distribution over real data x determined by a set of real images $\mathcal{D}$, and let $p_z(z)$ be a prior distribution over an input noise variable z. The generator G is trained to synthesize images given $z \sim p_z(z)$ as input, inducing a generative distribution $p_g(x)$ that should approximate the real data distribution $p_{data}(x)$. This is achieved through an adversarial game (Goodfellow et al., 2014), in which a discriminator D aims to distinguish between real images and images generated by G, while the generator tries to generate images that fool D. In this paper, we follow the WGAN-GP (Gulrajani et al., 2017) approach, which provides better convergence properties by using the Wasserstein loss (Arjovsky et al., 2017) and a gradient penalty term (omitted from our formulation for simplicity). The discriminator (or *critic*) and generator losses are defined as follows:

$$\mathcal{L}_D = \mathbb{E}_{z\sim p_z(z)}[D(G(z))] - \mathbb{E}_{x\sim p_{data}(x)}[D(x)], \tag{1}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z(z)}[D(G(z))]. \tag{2}$$

We also consider families of pretrained generators $\{G_i\}$. Each G_i has the ability to synthesize images given input noise $z \sim p_z^i(z)$. For simplicity and without loss of generality, we assume the prior distributions are Gaussian, i.e. $p_z^i(z) = \mathcal{N}(z|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$. Each generator $G_i(z)$ induces a learned generative distribution $p_g^i(x)$, which approximates the corresponding real data distribution $p_{data}^i(x)$ over real data x given by the set of source domain images $\mathcal{D}_i$.

3 MINING OPERATIONS ON GANS

Assume we have access to one or more pretrained GANs and wish to use their knowledge to train a new GAN for a (small) target domain. First, we propose a method called *mining GANs*, which learns miner networks that when combined with a pretrained GAN enable the generation of samples that are close to the target domain. Then, we show how the miners can be used to train new GANs.

3.1 MINING FROM A SINGLE GAN

We would like to approximate a target real data distribution $p^{\mathcal{T}}_{data}(x)$ induced by a set of real images $\mathcal{D}_{\mathcal{T}}$, given a critic D and a generator G trained to approximate a source data distribution $p_{data}(x)$ via the generative distribution $p_g(x)$. The *mining* operation learns a new generative distribution $p^{\mathcal{T}}_g(x)$ by finding those regions in $p_g(x)$ that better approximate the target data distribution $p^{\mathcal{T}}_{data}(x)$ while keeping G fixed. In order to find such regions, mining actually finds a new prior distribution $p^{\mathcal{T}}_z(z)$ such that samples $G(z)$ with $z \sim p^{\mathcal{T}}_z(z)$ are similar to samples from $p^{\mathcal{T}}_{data}(x)$ (see Fig. 1a). For this purpose, we propose a new GAN component called *miner*, implemented by a multilayer perceptron M. Its goal is to transform the original input noise variable $u \sim p_z(u)$ to follow a new, more suitable prior that identifies the regions in $p_g(x)$ that most closely align with the target distribution.

Fig. 1b presents the proposed mining architecture, called *MineGAN*. Miner M acts as an interface between the input noise variable and the generator, which remains fixed during training. To generate an image, we first sample $u \sim p_z(u)$, transform it with M and then input the transformed variable to the generator, i.e. $G(M(u))$. We train the model adversarially: the critic D aims to distinguish between fake images output by the generator $G(M(u))$ and real images x from the target data distribution $p^{\mathcal{T}}_{data}(x)$. We implement this with the following modification on the WGAN-GP loss

$$\mathcal{L}^M_D = \mathbb{E}_{u\sim p_z(u)}[D(G(M(u)))] - \mathbb{E}_{x\sim p^{\mathcal{T}}_{data}(x)}[D(x)], \tag{3}$$

$$\mathcal{L}^M_G = -\mathbb{E}_{u\sim p_z(u)}[D(G(M(u)))]. \tag{4}$$

The parameters of G are kept unchanged but the gradients are backgropagated all the way to M to learn its parameters. This training strategy will gear the miner towards the most rewarding regions of the input space, i.e. those that generate images close to $\mathcal{D}_{\mathcal{T}}$. Therefore, M is effectively mining the relevant input regions of prior $p_z(u)$ and giving rise to a targeted prior $p^{\mathcal{T}}_z(z)$, which will focus on these regions while ignoring other ones that lead to samples far off the target distribution $p^{\mathcal{T}}_{data}(x)$.

We distinguish two types of targeted generation: on-manifold and off-manifold. In the *on-manifold* case, there is a significant overlap between the original distribution $p_{data}(x)$ and the target distribution $p^{\mathcal{T}}_{data}(x)$. For example, $p_{data}(x)$ could be the distribution of human faces (both male and female) while $p^{\mathcal{T}}_{data}(x)$ includes female faces only. On the other hand, in *off-manifold* generation, the overlap between the two distributions is negligible, e.g. $p^{\mathcal{T}}_{data}(x)$ contains cat faces. The off-manifold task is evidently more challenging as the miner needs to find samples out of the original distribution (see Fig. 3). Specifically, we can consider the images in $\mathcal{D}$ to lie on a high-dimensional image manifold that contains the support of the real data distribution $p_{data}(x)$ (Arjovsky & Bottou, 2017). For a target distribution farther away from $p_{data}(x)$, its support will be more disjoint from the original distribution's support, and thus its samples might be off the manifold that contains $\mathcal{D}$.

3.2 MINING FROM MULTIPLE GANS

We generalize the mining operation to multiple pretrained generators. Given target data $\mathcal{D}_{\mathcal{T}}$, the task consists in mining relevant regions from the induced generative distributions learned by a family of N generators $\{G_i\}$. We do not have access to the original data used to train $\{G_i\}$ and can only use target data $\mathcal{D}_{\mathcal{T}}$. Fig. 1c presents the architecture of our model, which extends the mining architecture for a single pretrained GAN by including multiple miners and an additional component called the *selector*. In the following, we present this component and describe the training process in detail.

Supersample. In traditional GAN training, a fake minibatch is composed of fake images $G(z)$ generated with different samples $z \sim p_z(z)$. To construct fake minibatches for training a set of miners, we introduce the concept of *supersample*. A supersample $\mathcal{S}$ is a set of samples composed of exactly one sample per generator of the family, i.e. $\mathcal{S} = \{G_i(z)|z \sim p^i_z(z); i = 1, ..., N\}$. Each minibatch contains K supersamples, which amounts to a total of $K \times N$ fake images per minibatch.

Selector. The selector's task is choosing which pretrained model to use for generating samples during inference. For instance, imagine that $\mathcal{D}_1$ is a set of 'kitchen' images and $\mathcal{D}_2$ are 'bedroom' images, and let $\mathcal{D}_{\mathcal{T}}$ be 'white kitchens'. The selector should prioritize sampling from G_1, as the learned generative distribution $p^1_g(x)$ will contain kitchen images and thus will naturally be closer to $p^{\mathcal{T}}_{data}(x)$, the target distribution of white kitchens. Should $\mathcal{D}_{\mathcal{T}}$ comprise both white kitchens and dark bedrooms, sampling should be proportional to the distribution in the data.

We model the selector as a random variable s following a categorical distribution parametrized by $p_1, ..., p_N$ with $p_i > 0$ and $\sum p_i = 1$. During training, we estimate the parameters of this distribution as follows. The quality of each sample $G_i(z)$ is evaluated by a single critic D based on its critic value $D(G_i(z))$. Therefore it is expected that samples from generators whose generative distributions are closer to $p^{\mathcal{T}}_{data}(x)$ will on average have higher critic values when training D with $\mathcal{D}_{\mathcal{T}}$ as real data. Based on this, for each supersample $\mathcal{S}$ in the minibatch, we record which generator obtains the maximum critic value, i.e. $\arg\max_i D(G_i(z))$. By accumulating over all K supersamples and normalizing, we obtain an empirical probability value $\hat{p}_i$ that reflects how often generator G_i obtained the maximum critic value among all generators for the current minibatch. We estimate each parameter p_i as the empirical average $\hat{p}_i$ estimated in the last 1000 minibatches. Note that p_i are learned during training and stay fixed during inference, serving as parameters for the categorical distribution of selector s.

Critic and Miner Training. We now define the training behavior of the remaining learnable components, namely the critic D and miners $\{M_i\}$, when minibatches are composed of supersamples. The critic aims to distinguish real images from fake images. This is done by looking for artifacts in the fake images which distinguish them from the real ones. Another, less discussed but equally important task of the critic, is to observe the frequency of occurrence of images: if some (potentially high-quality) image occurs more often among fake images than real ones, the critic will lower its score, and thereby motivate the generator to lower the frequency of occurrence of this image. Training the critic by backpropagating from all images in the supersample prevents it from assessing the frequency of occurrence of the generated images (and we empirically observed this to yield unsatisfactory results). Therefore, we adapt the training loss for multiple GAN mining to:

$$\mathcal{L}^M_D = \mathbb{E}_{\{u^i \sim p^i_z(u)\}}[\max_i\{D(G_i(M_i(u^i)))\}] - \mathbb{E}_{x \sim p^{\mathcal{T}}_{data}(x)}[D(x)] \tag{5}$$

$$\mathcal{L}^M_G = -\mathbb{E}_{\{u^i \sim p^i_z(u)\}}[\max_i\{D(G_i(M_i(u^i)))\}]. \tag{6}$$

As a result of the $\max$ operator we only backpropagate from the generated image that obtained the highest critic score. This is in accordance with the selector, which is trained to select this image anyway. Training with Eq. 6 allows the critic to assess the frequency of occurrence correctly. Using this strategy, the critic can perform both its tasks: boosting the quality of the images as well as driving the miner to follow the distribution of the target set closely. Note that in this case we initialize the single critic D with the pretrained weights from one of the pretrained critics[1].

Conditional GANs. So far, we have only considered unconditional GAN models. However, conditional GANs are highly relevant in the literature, often providing the most successful approaches to realistic image generation (Brock et al., 2019; Zhang et al., 2018). cGANs introduce an additional input variable c, which conditions the generation on the class label. In this section, we extend our proposed MineGAN to cGANs that condition on the batch normalization layer (Dumoulin et al., 2017; Brock et al., 2019). More concretely, we experiment with BigGAN (Brock et al., 2019), which first maps a one-hot conditioning vector to an embedding vector and then maps this vector to layer-specific batch normalization parameters. Therefore, to mine BigGANs, alongside the standard miner M^z we introduce a second miner network M^c, which maps from u to the embedding space, resulting in a generator $G(M^c(u), M^z(u))$. The training is then equal to that of a single GAN and follows Eqs. 3 and 4. See Appendix B for experiments with another type of conditioning.

3.3 KNOWLEDGE TRANSFER WITH MINEGAN

Here we explain how to use mining GANs for knowledge transfer. The underlying idea of mining is to predispose the pretrained model to the target distribution by reducing the divergence between the source and target distributions. The miner network contains relatively few parameters and is therefore less prone to overfitting to the target data, which is known to occur when directly finetuning the generator G (Noguchi & Harada, 2019). We finalize the knowledge transfer to the new domain by finetuning both the miner M and generator G (releasing its weights). The risk of overfitting is now diminished as the generative distribution is closer to the target.

[1]We empirically found that starting from any pretrained critic leads to similar results (see Appedix D).

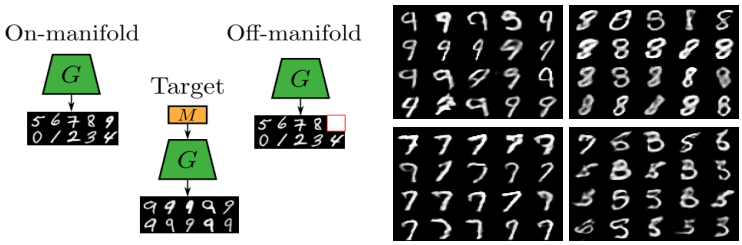


Figure 2: (Left) Illustration of generating target digit '9' from MNIST (unconditional case) for *on-manifold* and *off-manifold* targeted generation. (Right) Results for off-manifold generation. We generate 20 samples of digits '9', '8', '7' or '5'.

4 Related Work

Only few works have explored transferring knowledge for generative models. Wang et al. (2018) investigated finetuning of pretrained GANs, leading to improved performance for target domains with limited samples. This method, however, suffers from mode collapse since it requires all parameters of the generator to be updated to adapt to the target domain. Recently, Noguchi & Harada (2019) proposed to only update the batch normalization parameters. They also replaced the GAN loss with a mean square error loss. As a result, their model only learns the relationship between latent vectors and sparse training samples, requiring the distribution of z to be truncated during inference to generate realistic samples. The proposed MineGAN does not suffer from this drawback. In addition, we are the first to consider transferring knowledge from multiple GANs to a single target domain.

Nguyen et al. (2016) have investigated training networks to generate images that maximize the activation of neurons in a pretrained classification network. In a follow-up paper, which improves the diversity of the generated images, they showed that this technique can be used to generate images of a particular class from a pretrained classifier network (Nguyen et al., 2017). In principle, these works do not aim at knowledge transfer to a new domain, and can instead only be applied to generate a distribution which is exactly described by one of the class labels of the pretrained classifier network. Another major difference is that the generation at inference time of each image is an iterative process of successive backpropagation updates until convergence, whereas our method is feedforward.

5 Experiments

In this section, we first introduce the used evaluation measures and architectures used. Then, we evaluate our method for knowledge transfer from unconditional GANs, considering both a single and multiple pretrained generators. Finally, we assess transfer learning from conditional GANs. Experiments focus on transferring knowledge to small target domains.

Evaluation Measures. We employ the widely used Fréchet Inception Distance (FID) (Heusel et al., 2017) for evaluation. FID measures the similarity between two sets in the embedding space given by the features of a convolutional neural network. More specifically, it computes the differences between the estimated means and covariances assuming a multivariate normal distribution on the features. FID measures both the quality and diversity of the generated images and has been shown to correlate well with human perception (Heusel et al., 2017), but it suffers from instability on small datasets. For this reason, we also employ Kernel Maximum Mean Discrepancy (KMMD) with a Gaussian kernel and Mean Variance (MV) for some experiments (Noguchi & Harada, 2019). Low KMMD values indicate high quality images, while high values of MV indicate more image diversity.

Architectures. We introduce mining to several architectures, including Progressive GAN (Karras et al., 2017), SNGAN (Miyato et al., 2018), and BigGAN (Brock et al., 2019). The training details for all models are included in Appendix A. For the miner, we use four fully connected layers for all experiments except those on MNIST, where we use only two (see Appendix A).

5.1 Transferring knowledge from unconditional GANs

MNIST Dataset. We first evaluate our model on the MNIST (LeCun, 1998) dataset as a proof of concept for generating simple handwritten digits. We test mining for both *on-manifold* and *off-*

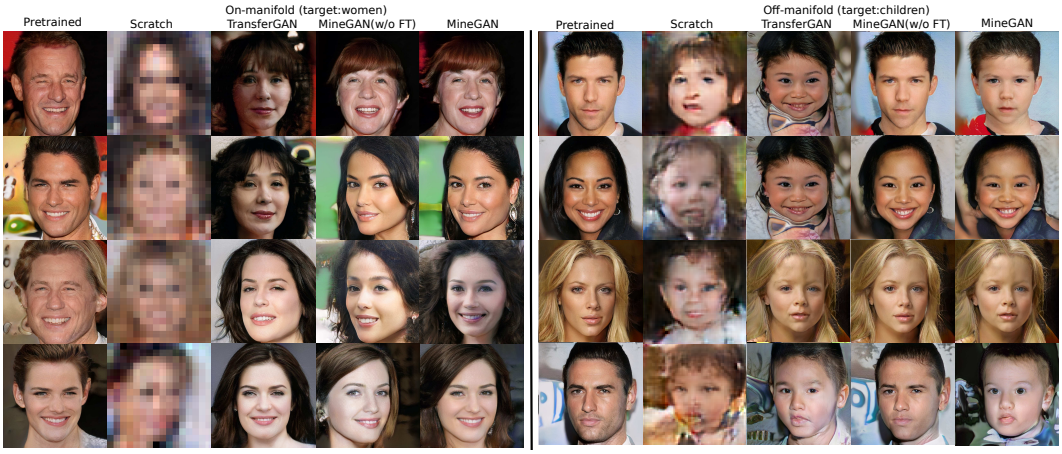


Figure 3: Results: (left) On-manifold (CelebA→FFHQ women), (right) Off-manifold (CelebA→FFHQ children). Based on pretrained *Progressive GAN*. More examples are shown in Appendix C.

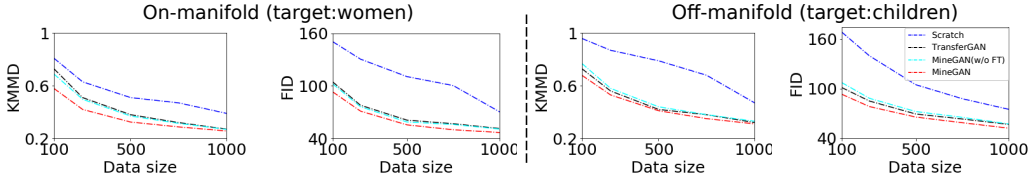
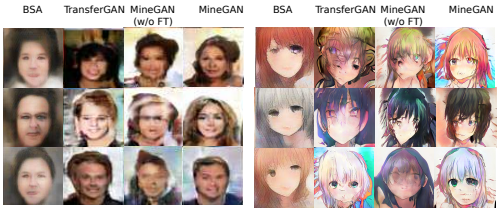


Figure 4: KMMD and FID on CelebA→FFHQ women (left) and CelebA→FFHQ children (right)

manifold targeted image generation (see Fig. 2 left). We use 1000 images of size 28×28 as target data. In the on-manifold case, the trained network G already possesses this exact knowledge, and thus the miner's goal is to correctly identify these regions by restricting the sampling regions in $p(z)$. In off-manifold targeted generation, G is trained to synthesize all MNIST digits except for one, e.g. G generates 0-8 but not 9. This is a more challenging task, as we can observe qualitatively in the right section of Fig. 2 and quantitatively in Appendix B. The miner manages to steer the generator to output samples that resemble the target digits, mostly by using and merging patterns from other digits in the source set. For example, digit '9' frequently resembles a modified 4 while '8' heavily borrows from 0s and 3s. We can also observe that some digits can be more challenging to generate. For example, '5' is generally more distinct from other digits and thus in more cases the resulting sample is confused with other digits such as '3'. On the other hand, '7' can be simply generated by slightly modifying 1s or 9s.

Single Pretrained Model. We start by transferring knowledge from a Progressive GAN trained on *CelebA (Liu et al., 2015)*. We evaluate the performance on target datasets of varying size, and use images of size 1024×1024. We consider two target domains: one on-manifold domain, *FFHQ woman* (Karras et al., 2019b) and one off-manifold domain, *FFHQ children face* (Karras et al., 2019b). We consider two versions of our model: *MineGAN* refers to the mining method combined with finetuning to the target domain, whereas *MineGAN(w/o FT)* only applies mining. We compare our results to training from *scratch*, and the *transferring GANs* method of Wang et al. (2018). In the plots in Fig. 4, we show the performance in terms of FID and KMMD as a function of the number of images in the target domain. The proposed MineGAN framework outperforms all baselines. For the on-manifold experiment, MineGAN already outperforms the other baselines, and results are further improved by additional finetuning. Interestingly, for the off-manifold experiment MineGAN obtains only slightly worse results than Transferring GAN, showing that the miner already manages to generate images close to the target domain. Fig. 3 shows images generated when the target data contains 100 training images. Training the model from scratch generates blurry images, and Transferring GANs results in mode collapse. MineGAN, in contrast, generates high-quality images without mode collapse. The generated images are sharper and have more realistic fine details.

We also perform the experiment proposed by Noguchi & Harada (2019) using the same architecture as them, namely a SNGAN. They performed knowledge transfer from a pretrained SNGAN from ImageNet (Krizhevsky et al., 2012) to FFHQ and from ImageNet to Anime Face. Target domains



Method	FFHQ KMMD	FFHQ MV	Anime Face KMMD	Anime Face MV
From scratch	0.890	-	0.753	-
TransferGAN	0.346	0.506	0.347	0.785
VAE	0.744	-	0.790	-
BSA	0.345	0.785	0.342	0.908
MineGAN (w/o FT)	0.349	0.774	0.347	0.891
MineGAN	**0.337**	**0.812**	**0.334**	**0.934**

Figure 5: Results for various knowledge transfer methods. (Left) Generated images. (Right) KMMD and MV.

Figure 6: Results: {car, bus} $\rightarrow$ red vehicles (left) and {Living room, Bridge, Church, Kitchen} $\rightarrow$ Tower (right). Based on pretrained *Progressive GAN*. For TransferGAN we show the pretrained model between parentheses. More examples are shown in Appendix D.

have only 25 images of size 128 $\times$ 128. We added our results to those reported in (Noguchi & Harada, 2019) in Fig. 5 (right). Compared to Batch Statistics Adaptation (BSA), MineGAN (w/o FT) obtains similar KMMD scores, showing that generated images obtain the same quality. MineGAN outperforms BSA both in KMMD score and Mean Variance. The qualitative results (shown in Fig. 5 (left)) clearly show that MineGAN outperforms the baselines. The BSA results show blur artifacts, which are probably caused by the mean square error used to optimize their model.

Multiple Pretrained Models. We now investigate the effect of having more than one pretrained model to mine from. We consider two pretrained Progressive GANs: one on *Cars* and one on *Buses*, both from the LSUN dataset (Yu et al., 2015a). These pretrained networks generate cars and buses of a variety of different colors. We collect a target dataset of 200 images (with resolution 256×256) of *red vehicles*, which contains both red cars and red buses. We consider three target sets with different car-bus ratios (we consider 0.3:0.7, 0.5:0.5, and 0.7:0.3) which allows us to evaluate the estimated probabilities p_i of the selector. To successfully generate *red vehicles* knowledge needs to be transferred from both pre-trained models. Fig. 6 shows the synthesized images. As expected, the limited amount of data makes training from scratch unfeasible. The TransferGAN approach produces only high-quality output samples for one of the two classes; the class which coincides with the pretrained model. TransferGAN cannot extract knowledge from both pretrained GANs. On the other hand, MineGAN synthesizes high-quality images by successfully transferring the knowledge from both source domains simultaneously. Table 1 (left top row) quantitatively validates that our method obtains a significantly lower FID score. Furthermore, the probability distribution predicted by the selector, reported in Table 1 (right top row), matches the class distribution of the target data. Finally, Fig. 6 (left) shows visual examples, showing that only MineGAN generates high-quality images for both classes.

To demonstrate the scalability of MineGAN with multiple pretrained models, we conduct experiments using four different generators, each trained on a different LSUN category including *Livingroom*, *Kitchen*, *Church*, and *Bridge*. We consider two different off-manifold target datasets, one with *Bedroom* images and one with *Tower* images, both containing 200 images. Table 1 (left bottom

Scratch	TransferGAN (car)	TransferGAN (bus)	MineGAN (w/o FT)	MineGAN
190 / 185 / 196	76.9 / 72.4 / 75.6	72.8 / 71.3 / 73.5	67.3 / 65.9 / 65.8	61.2 / 59.4 / 61.5
Scratch	TransferGAN(lroom)	TransferGAN(church)	MineGAN (w/o FT)	MineGAN
176	78.9	73.8	69.2	62.4
Scratch	TransferGAN(lroom)	TransferGAN(church)	MineGAN(w/o FT)	MineGAN
181	65.4	71.5	58.9	54.7

Car		Bus	
0.34 / 0.48 / 0.64		0.66 / 0.52 / 0.36	
livingroom	kitchen	bridge	church
0.07	0.06	0.42	0.45
livingroom	kitchen	bridge	church
0.45	0.40	0.08	0.07

Table 1: Results for {Car, Bus} → Red vehicles with three different target data distributions (ratio cars:buses equal to 0.3:0.7, 0.5:0.5 and 0.7:0.3) (top row) and {Living room, Bridge, Church, Kitchen} → Tower/Bedroom (two bottom rows). We show FID scores between real and generated samples (left) and estimated probabilities p_i for each model (right).

Figure 7: Results for conditional GAN computed with BigGAN. (Left) Off-manifold (ImageNet→Places365). (Right) On-manifold (ImageNet→ImageNet).

rows) again shows that our method obtains significantly better FID scores even when we choose the most relevant pretrained GAN to initialize training for TransferGAN. Table 1 (right bottom rows) shows that the miner identifies the relevant pretrained models, e.g. transferring knowledge from *Bridge* and *Church* for the target domain *Tower*. Finally, Fig. 6 (right) provides visual examples.

5.2 Transferring knowledge from conditional GANs

Here we transfer knowledge from a pretrained conditional BigGAN. We evaluate on two target datasets: on-manifold (ImageNet: *cock*, *tape player*, *broccoli*, *fire engine*, *harvester*) and off-manifold (Places365 (Zhou et al., 2014): *alley*, *arch*, *art gallery*, *auditorium* and *ballroom*). We use 500 images per category. Fig. 7 shows several representative examples of the different methods[2]. If should be noted that for the on-manifold results for TransferGAN we use the label of the target domain class (which is not used by MineGAN). This label cannot be provided for the off-manifold experiment, and in that case TransferGAN fails to obtain satisfactory results. Again, MineGAN manages to produce realistic results, also for the off-manifold case. Further comparison and quantitative results are provided in Appendix E.

6 Conclusions

We presented a new model for knowledge transfer for generative models. It is based on a mining operation that identifies the regions on the learned GAN manifold that are closer to a given target domain. We demonstrated that mining can be applied to single, conditional and multiple pretrained GANs. Experiments with various GAN architectures (BigGAN, Progressive GAN, and SNGAN) on multiple datasets demonstrated the effectiveness of MineGAN. Our results show that we outperform previous approaches, including TranferGAN (Wang et al., 2018) and BSA (Noguchi & Harada, 2019). Finally, we demonstrated that MineGAN can be used to transfer knowledge from multiple domains, which cannot be achieved with previous methods.

[2]We were unable to obtain satisfactory results with the BSA method in this setting (images suffered significantly from blur artifacts) and have excluded them from this experiment

REFERENCES

Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *ICLR*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

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

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, pp. 1486–1494, 2015.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pp. 2414–2423, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, pp. 5767–5777, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, pp. 6626–6637, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pp. 1125–1134, 2017.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019a.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pp. 4401–4410, 2019b.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pp. 3730–3738, 2015.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, pp. 3387–3395, 2016.

Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, pp. 4467–4477, 2017.

Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. *arXiv preprint arXiv:1904.01774*, 2019.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pp. 1717–1724. IEEE, 2014.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *ECCV*, pp. 213–229, 2018.

Michael Tschannen, Eirikur Agustsson, and Mario Lucic. Deep generative models for distribution-preserving lossy compression. In *NIPS*, pp. 5929–5940, 2018.

Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *ECCV*, pp. 218–234, 2018.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015a.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015b.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pp. 649–666. Springer, 2016.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *ICLR*, 2014.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pp. 2223–2232, 2017.

A Appendix: Architecture and training details.

MNIST dataset. Our model contains a *miner*, *generator* and *discriminator*. For both unconditional and conditional GANs, we use the same framework (Gulrajani et al., 2017) to design the generator and discriminator. The miner is composed of two fully connected layers with the same dimensionality as the latent space $|z|$. The visual results are computed with $|z| = 16$; we found that the quantitative results improved for larger $|z|$ and choose $|z| = 128$. In MNIST, we consider the case where label c is a one-hot vector. We use the selector to predict the conditioning label. We randomly initialize the weights of the miner following a Gaussian distribution, and optimize the model using Adam (Kingma & Ba, 2014) with batch size of 64. The learning rate of our model is 0.0004, with an exponential decay rates of $(\beta_1, \beta_2) = (0.5, 0.999)$. Note the same configure is also used for the unconditional case.

CelebA Womem, FFHQ Children and LSUN (Tower and Bedroom) Datasets. We design the generator and discriminator based on Progressive GANs (Karras et al., 2017). Both networks use a multi-scale technique to generate high-resolution images. The miner comprises out of four fully

connected layers (8-64-128-256-512), each of which is followed with a *relu* and *pixel normalization* except for last layer. We use a Gaussian distribution to initialize the miner, and optimize the model using Adam (Kingma & Ba, 2014) with batch size of 4. The learning rate of our model is 0.0015, with an exponential decay rates of $(\beta_1, \beta_2) = (0, 0.99)$.

FFHQ Face and Anime Face. We use the same network as (Miyato et al., 2018), namely the SNGAN. The miner consists of three fully connected layers (8-32-64-128). We randomly initialize the weights following a Gaussian distribution. For this additional set of experiments, we use Adam (Kingma & Ba, 2014) with a batch size of 8, following a hyper parameter learning rate of 0.0002 and exponential decay rate of $(\beta_1, \beta_2) = (0, 0.9)$.

Conditional GANs. For conditional GANs, we use the pretrained BigGAN (Brock et al., 2019). We ignore the projection loss in the discriminator, since we do not have access to the label of the target data. The miner consists of two sub-networks: miner M^z and miner M^c. Both M^z and M^c are composed of four fully connected layers (128-128-128-128-120(M^z)/128(M^c)). We use Adam (Kingma & Ba, 2014) with a batch size of 256, learning rate of 0.0001 for miner and generator and 0.0004 for discriminator. The exponential decay rate is $(\beta_1, \beta_2) = (0, 0.999)$. We randomly initialize the weights following a Gaussian distribution.

B APPENDIX: MNIST EXPERIMENT

We expand the MNIST experiments presented in Sec. 5.1 by providing a quantitative evaluation and including results on conditional GANs. As evaluation measures, we use FID (Sec. 5) and classifier error (Shmelkov et al., 2018). To compute classifier error, we first train a CNN classifier on real training data to distinguish between multiple classes (e.g. digit classifier). Then, we classify the generated images that should belong to a particular class and measure the error as the percentage of misclassified images. This gives us an estimation of how realistic and accurate the generated images are in the context of targeted generation.

The conditional architecture in this experiment (Appendix A) conditions by concatenating to the input noize z a one-hot vector c indicating the target class of the image. We extend MineGAN to this type of pretrained conditional models by considering each possible conditioning as an independently trained generator. Given a conditional generator $G(c, z)$, we consider $G(i, z)$ as G_i and apply the presented MineGAN approach on the family $\{G(i, z) | \, i = 1, ..., N\}$. The resulting selector now chooses among the N classes of the model rather than among N pretrained models, but the rest of the MineGAN training remains the same, including the training of N independent miners.

Table 2 presents the results for both unconditional and conditional models, using a noise length of $|z| = 128$. The relatively low error values indicate that the miner manages to identify the correct regions for generating the target digits. The conditional model offers better results than the unconditional one by selecting the target class more often. We can also observe that the off-manifold task is more difficult than the on-manifold task, as indicated by the higher evaluation scores. However, the off-manifold scores are still reasonably low, indicating that the miner manages to find suitable regions from other digits by mining local patterns shared with the target. Overall, these results indicate the effectiveness of mining on MNIST for both types of targeted image generation. In addition, in Fig. 8 we have added a visualization for the off-manifold MNIST classes which were not already shown in Fig. 2.

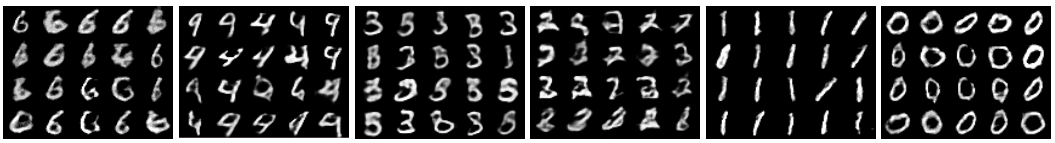


Figure 8: Results for unconditional off-manifold generation of digits ‘6’, ‘4’, ‘3’, ‘2’, ‘1’, ‘0’.

Table 2: Quantitative results of mining on MNIST, expressed as FID / classifier error.

d	On-manifold		Off-manifold	
	Unconditional	Conditional	Unconditional	Conditional
0	13.4 / 2.5	12.6 / 0.7	21.3 / 2.8	15.6 / 1.1
1	13.1 / 1.7	12.6 / 1.9	15.9 / 2.5	14.8 / 2.1
2	14.6 / 6.3	12.8 / 2.7	23.1 / 5.2	18.2 / 3.6
3	14.1 / 10.1	13.3 / 1.6	22.8 / 7.3	14.2 / 1.5
4	14.7 / 6.4	13.4 / 1.2	23.4 / 6.3	15.3 / 4.2
5	13.1 / 9.3	11.7 / 2.1	21.9 / 10.9	17.2 / 5.7
6	13.4 / 2.8	14.3 / 1.8	24 / 3.1	15.8 / 1.6
7	12.9 / 3.2	14.2 / 1.8	24.8 / 4.9	16.3 / 2.6
8	14.2 / 7.5	14.7 / 5.5	25.7 / 9.8	18.7 / 5.6
9	11.3 / 6.8	11.2 / 2.9	12.5 / 7.4	16.3 / 3.5
Average	13.5 / 5.7	13.1 / 2.2	21.5 / 6.0	16.2 / 3.2

C Appendix: Further results on CelebA

We provide additional results for the on-manifold experiment CelebA→FFHQ women in Fig. 9, and the off-manifold CelebA→FFHQ children in Fig. 10. In addition, we have also performed an on-manifold experiment with CelebA→CelebA women, whose results are provided in Fig. 11.

D Appendix: further results for LSUN

We provide additional results for the experiment ($\{$bus, car$\}$) → Red vehicles in Fig. 12.

We also provide additional results for the experiment $\{$Bedroom, Bridge, Church, Kitchen$\}$ → Tower/Bedroom in Fig. 13.

When applying MineGAN to multiple pretrained GANs, we use one of the domains to initialize the weights of the critic. In Table 1 we used *Church* to initialize the critic in case of the target set *Tower*, and *Kitchen* to initialize the critic for the target set *Bedroom*. We found this choice to be of little influence on the final results. When using *Kitchen* to initialize the critic for target set *Tower* results change from 62.4 to 61.7. When using *Church* to initialize the critic for target set *Bedroom* results change from 54.7 to 54.3.

E Appendix: Transferring knowledge from conditional GANs

We provide quantitative results for mining the BigGAN in Table 3 and qualitative results in Fig. 14. As can be seen, our method obtains the best FID and KMMD scores. DGN-AM (Nguyen et al., 2016) obtains the worst score, since it fails to achieve diversity. PPGN (Nguyen et al., 2017) improves the diversity of the model, but is still worse than our method. It should be noted that both DGN-AM (Nguyen et al., 2016) and PPGN (Nguyen et al., 2017) are based on a less complex GAN (similar to DCGAN). Therefore, the quantitative and qualitative results presented here should be more interpreted to show the general progress of GANs. However, we do want to stress that both DGN-AM and PPGN do not aim to transfer knowledge to new domains. They can only generate samples of a particular class of a pretrained classifier network, and they have no explicit loss which ensures that the generated images follow a target distribution. Another important point is that the generation at inference time of each image is an iterative process of successive backpropagation that updates until convergence, whereas our method is feedforward. In Table 3 we have included the running time of the various algorithms, which clearly shows that the feedforward methods (TransferGAN and MineGAN) are much faster even when applied on a much more complex GAN (BigGAN). We used the default setting of 200 iterations for DGN-AM and PPGN. Timings have been computed with a CPU Intel Xeon E5-1620 v3 @ 3.50GHz, and GPU NVIDIA Quadro K5200.

Figure 9: (CelebA→FFHQ women). Based on pretrained *Progressive GAN*.

Figure 10: (CelebA→ FFHQ children). Based on pretrained *Progressive GAN*.

Figure 11: (CelebA→CelebA women). Based on pretrained *Progressive GAN*.

Figure 12: ({bus, car}) →red vehicles. Based on pretrained *Progressive GAN*.

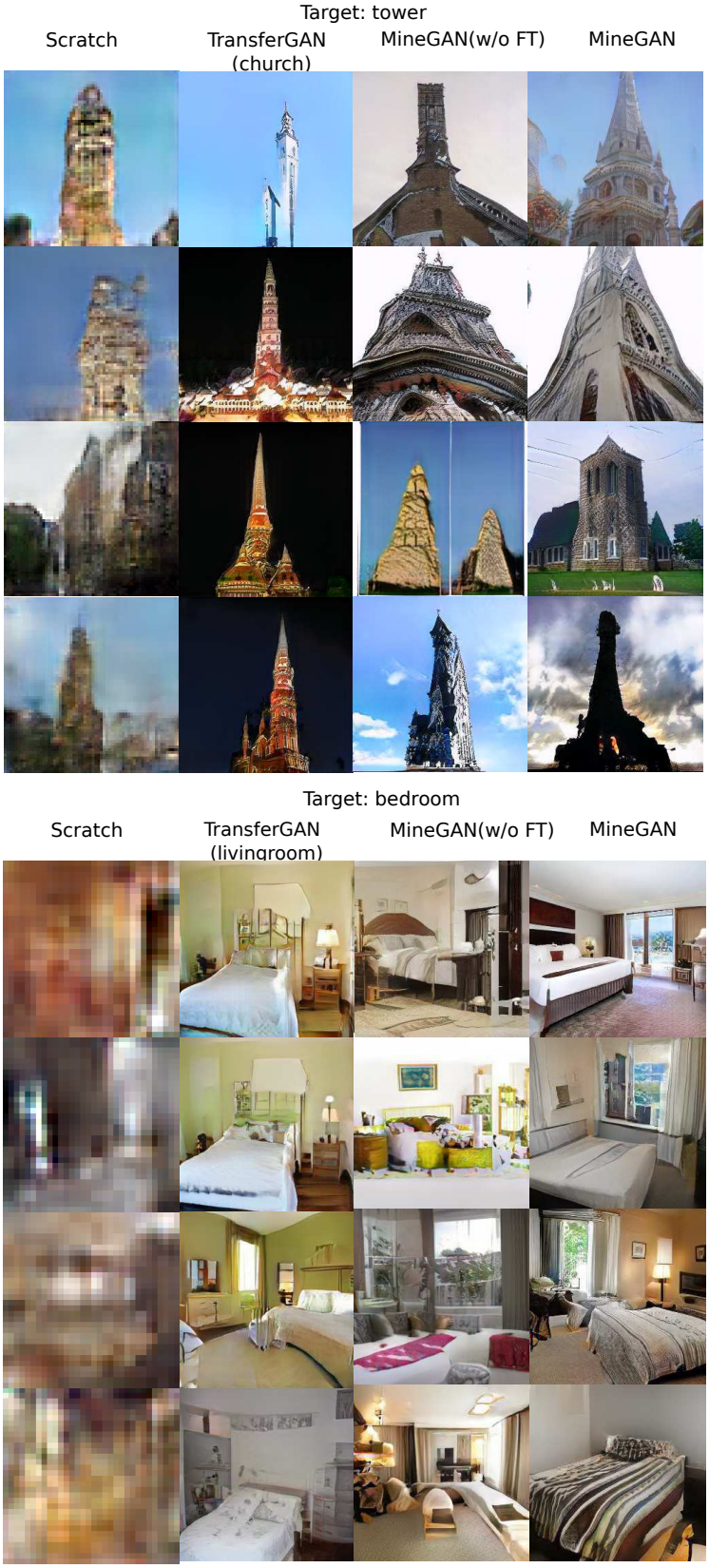


Figure 13: Results for unconditional GAN. (Top) (Livingroom, kitchen, bridge, church)→Tower. (Bottom) (Livingroom, kitchen, bridge, church)→Bedroom. Based on pretrained *Progressive GAN*.

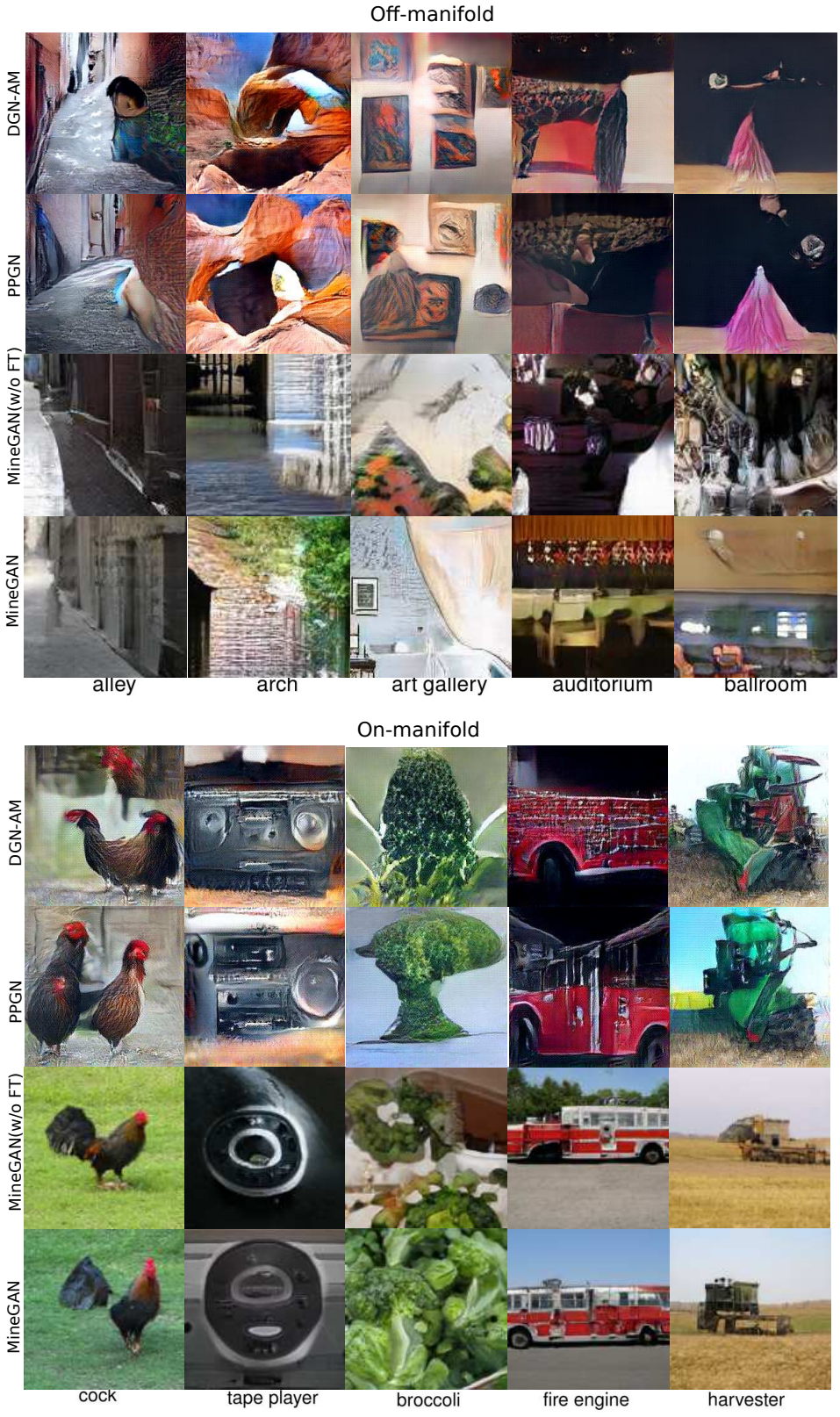


Figure 14: Results for conditional GAN. (Top) Off-manifold (ImageNet→Places365). (Bottom) On-manifold (ImageNet→ImageNet).

	Target label	Off-manifold FID/KMMD	Target label	On-manifold FID/KMMD	Inference time
DGN-AM	Yes	214/0.98	Yes	180/0.95	6.70
PPGN	Yes	139/0.56	Yes	12770.47	8.65
TransferGAN*	No	89.2/0.53	Yes	58.4/0.39	0.0051
Scratch	No	190/0.96	No	187/0.93	0.0051
MineGAN(w/o FT)	No	82.3/0.47	No	61.8/0.32	0.0052
MineGAN	No	58.4/0.41	No	52.3/0.25	0.0052

Table 3: Distance between real data and generated samples as measured by FID score and KMMD value. The off-manifold results correspond to Imagenet $\rightarrow$ Places365, and the on-manifold results correspond to Imagenet $\rightarrow$ Imagenet. We also indicate whether the method requires the target label. Finally, we show the inference time for the various methods in seconds.