

MineGAN: effective knowledge transfer from GANs to target domains with few images

Yaxing Wang¹, Abel Gonzalez-Garcia¹, David Berga¹

Luis Herranz¹, Fahad Shahbaz Khan^{2,3}, Joost van de Weijer¹

¹ Computer Vision Center, Universitat Autònoma de Barcelona, Spain

² Inception Institute of Artificial Intelligence, UAE ³ CVL, Linköping University, Sweden

{yaxing,agonzalez,dberga,lherranz,joost}@cvc.uab.es, fahad.khan@liu.se

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. Given the often enormous effort required to train GANs, both computationally as well as in the dataset collection, the re-use of pretrained GANs is a desirable objective. We propose a novel knowledge transfer method for generative models based on mining the knowledge that is most beneficial to a specific target domain, either from a single or multiple pretrained GANs. This is done using a miner network that identifies which part of the generative distribution of each pretrained GAN outputs samples closest to the target domain. Mining effectively steers GAN sampling towards suitable regions of the latent space, which facilitates the posterior finetuning and avoids pathologies of other methods such as mode collapse and lack of flexibility. 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 domains with few target images, outperforming existing methods. In addition, MineGAN can successfully transfer knowledge from multiple pretrained GANs. Our code is available at: <https://github.com/yaxingwang/MineGAN>.

1. Introduction

Generative adversarial networks (GANs) can learn the complex underlying distribution of image collections [10]. They have been shown to generate high-quality realistic images [14, 15, 4] and are used in many applications including image manipulation [13, 41], style transfer [9], compression [33], and colorization [38]. It is known that high-

quality GANs require a significant amount of training data and time. For example, Progressive GANs [14] 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 instance, it might be desirable to only generate women using a GAN trained to generate men and women alike. Alternatively, one may want to generate 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 domains with limited data has been extensively studied for discriminative models [7, 27, 28, 34], enabling the re-use of high-quality networks. However, knowledge transfer for generative models has received significantly less attention, possibly due to its great difficulty, especially when transferring to target domains with few images. Only recently, Wang et al. [35] studied finetuning from a single pretrained generative model and showed that it is beneficial for domains with scarce data. However, Noguchi and Harada [26] observed that this technique leads to mode collapse. Instead, they proposed to reduce the number of trainable parameters, and only finetune the learnable parameters for the batch normalization (scale and shift) of the generator. Despite being less prone to overfitting, their approach severely limits the flexibility of the knowledge transfer.

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* of GANs. This is performed by a *miner network* that transforms a multivariate normal distribution into a distribution on the input space of the pre-

trained GAN in such a way that the generated images resemble those of the target domain. The miner network has considerably fewer parameters than the pretrained GAN and is therefore less prone to overfitting. The mining step predisposes the pretrained GAN to sample from a narrower region of the latent distribution that is closer to the target domain, which in turn eases the subsequent finetuning step by providing a cleaner training signal with lower variance (in contrast to sampling from the whole source latent space as in [35]). Consequently, our method preserves the adaptation capabilities of finetuning while preventing overfitting.

Importantly, our mining approach enables transferring from 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 a selective backpropagation procedure. Our main contributions are:

- We introduce a novel miner network to steer the sampling of the latent distribution of a pretrained GAN to a target distribution determined by few images.
- We propose the first method to transfer knowledge from multiple GANs to a single generative model.
- We outperform existing competitors on a variety of settings, including transferring knowledge from unconditional, conditional, and multiple GANs.

2. Related work

Generative adversarial networks. GANs consists of two modules: generator and discriminator [10]. The generator aims to generate images to fool the discriminator, while the discriminator aims to distinguish generated from real images. Training GANs was initially difficult, due to mode collapse and training instability. Several methods focus on addressing these problems [11, 31, 21, 3, 22], while another major line of research aims to improve the architectures to generate higher quality images [29, 6, 14, 16, 4]. For example, Progressive GAN [14] generates better images by synthesizing them progressively from low to high-resolution. Finally, BigGAN [4] successfully performs conditional high-realistic generation from ImageNet [5].

Transfer learning for GANs. While knowledge transfer has been widely studied for discriminative models in computer vision [7, 28, 27, 34], only a few works have explored transferring knowledge for generative models [26, 35]. Wang *et al.* [35] investigated finetuning of pretrained GANs, leading to improved performance for target domains with limited samples. This method, however, suffers from mode collapse and overfitting, as it updates all parameters of the generator to adapt to the target domain. Recently, Noguchi and Harada [26] proposed to only update the batch normalization parameters. Although less susceptible to mode collapse, this approach significantly reduces

the adaptation flexibility of the model since changing only the parameters of the batch normalization permits for style changes but is not expected to function when shape needs to be changed. 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 input noise distribution to be truncated during inference to generate realistic samples. The proposed MineGAN does not suffer from this drawback, as it learns how to automatically adapt the input distribution. In addition, we are the first to consider transferring knowledge from multiple GANs to a single target domain.

Iterative image generation. Nguyen et al. [25] have investigated training networks to generate images that maximize the activation of neurons in a pretrained classification network. In a follow-up approach [24] that improves the diversity of the generated images, they use this technique to generate images of a particular class from a pretrained classifier network. In principle, these works do not aim at transferring knowledge to a new domain, and can instead only be applied to generate a distribution that 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 during inference.

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 target domain with few images. For clarity’s sake, we first introduce mining from a single GAN in Section 3.2, but our method is general for an arbitrary number of pretrained GANs, as explained in Section 3.3. Then, we show how the miners can be used to train new GANs (Section 3.4).

3.1. 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 [10], 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 WGAN-GP [11], which provides better convergence properties by using the Wasserstein loss [3] and a gradient penalty term (omitted from our formulation for simplicity). The discriminator (or critic) and generator losses are defined as follows:

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

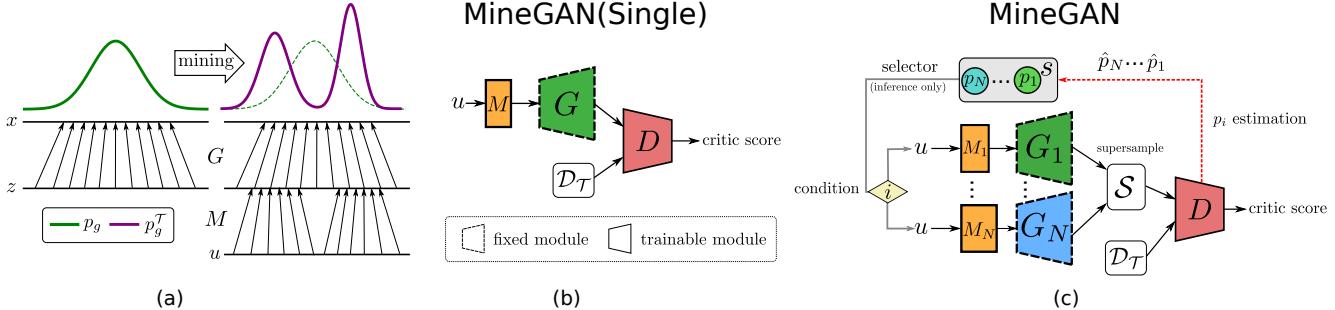


Figure 1: (a) Intuition behind our approach for a simple case. Mining shifts the prior input distribution towards the most promising regions with respect to given target data \mathcal{D}_T . In practice, the input distribution is much more complex. (b) Architecture implementing the proposed mining operation 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}_T . Note that during the first stage, when training the miner, the generator remains fixed. In a second stage, we finetune the miner, generator and discriminator together. (c) Training setup for multiple generators. Miners M_1, \dots, M_N identify subregions of the pretrained generators while selector S learns the sampling frequencies of the various generators.

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

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. $p_z^i(z) = \mathcal{N}(z|\mu_i, \Sigma_i)$. 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 .

3.2. Mining from a single GAN

We want to approximate a target real data distribution $p_{data}^T(x)$ induced by a set of real images \mathcal{D}_T , given a critic D and a generator G , which have been 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 $p_g^T(x)$ by finding those regions in $p_g(x)$ that better approximate the target data distribution $p_{data}^T(x)$ while keeping G fixed. In order to find such regions, mining actually finds a new prior distribution $p_z^T(z)$ such that samples $G(z)$ with $z \sim p_z^T(z)$ are similar to samples from $p_{data}^T(x)$ (see Fig. 1a). For this purpose, we propose a new GAN component called *miner*, implemented by a small 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.

Our full method acts in two stages. The first stage steers the latent space of the fixed generator G to suitable areas for the target distribution. We refer to the first stage as *MineGAN (w/o FT)* and present the proposed mining architecture in Fig. 1b. The second stage updates the weights of the generator via finetuning (G is no longer fixed). *MineGAN* refers to our full method including finetuning.

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 $p_{data}^T(x)$. We implement this with the following modification on the WGAN-GP loss:

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

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

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

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 $p_{data}^T(x)$. For example, $p_{data}(x)$ could be the distribution of human faces (both male and female) while $p_{data}^T(x)$ includes female faces only. On the other hand, in *off-manifold* generation, the overlap between the two distributions is negligible, e.g. $p_{data}^T(x)$ 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. 4). 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 $p_{data}(x)$ [2]. 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} .

3.3. Mining from multiple GANs

In the general case, the mining operation is applied on multiple pretrained generators. Given target data \mathcal{D}_T , the task consists in **mining relevant regions from the induced generative distributions** learned by a family of N generators $\{G_i\}$. In this task, we do not have access to the original data used to train $\{G_i\}$ and can **only use target data** \mathcal{D}_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 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. $\mathcal{S} = \{G_i(z) | z \sim p_z^i(z); i = 1, \dots, N\}$. 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}_T be ‘white kitchens’. The selector should prioritize sampling from G_1 , as the learned generative distribution $p_g^1(x)$ will contain kitchen images and thus will naturally be closer to $p_{data}^T(x)$, the target distribution of white kitchens. Should \mathcal{D}_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, \dots, p_N with $p_i > 0$, $\sum p_i = 1$. 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 $D(G_i(z))$** . Higher critic values indicate that the generated sample from G_i is closer to the real distribution. 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 in the last 1000 minibatches. Note that p_i are learned during training and fixed during inference, where we apply a multinomial sampling function to sample the index.

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

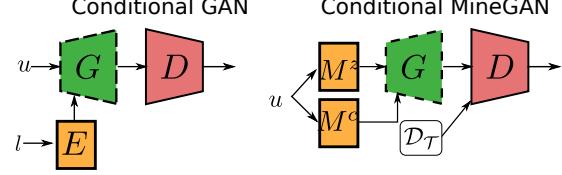


Figure 2: Application of mining in conditional setting (on BigGAN [4]). We apply an additional miner network to estimate the class embedding. \mathcal{D}_T : target data, E : class embedding, l : label.

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 leading to unsatisfactory results empirically. Therefore, the loss for multiple GAN mining is:

$$\mathcal{L}_D^M = \mathbb{E}_{\{u^i \sim p_z^i(u)\}} [\max_i \{D(G_i(M_i(u^i)))\}] - \mathbb{E}_{x \sim p_{data}^T(x)} [D(x)] \quad (5)$$

$$\mathcal{L}_G^M = -\mathbb{E}_{\{u^i \sim p_z^i(u)\}} [\max_i \{D(G_i(M_i(u^i)))\}]. \quad (6)$$

As a result of the max operator we only backpropagate from the generated image that obtained the highest critic score. This allows the critic to assess the frequency of occurrence correctly. Using this strategy, the critic can perform both its tasks: boosting the quality of images as well as driving the miner to closely follow the distribution of the target set. Note that we initialize the single critic D with the pretrained weights from one of the pretrained critics¹.

Conditional GANs. So far, we have only used unconditional GANs. However, conditional GANs (cGANs), which introduce an **additional input variable to condition the generation to the class label**, are used by the most successful approaches [4, 37]. Here we extend our proposed MineGAN to cGANs that condition on the batch normalization layer [8, 4]², more concretely, BigGAN [4] (Fig. 2 (left)). First, a label l is mapped to an embedding vector by means of a class embedding E , and then this vector is mapped to layer-specific batch normalization parameters. The discriminator is further conditioned via label projection [23]. Fig. 2 (right) shows how 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 equal to that of a single GAN and follows Eqs. 3 and 4.

¹We empirically found that starting from any pretrained critic leads to similar results (see Suppl. Mat. (Sec. F))

²See Suppl. Mat. (Sec. D) for results on another type of conditioning.

3.4. Knowledge transfer with MineGAN

The underlying idea of mining is to **predispose the pre-trained model to the target distribution by reducing the divergence between source and target distributions**. The miner network contains relatively few parameters and is therefore less prone to overfitting, which is known to occur when directly finetuning the generator G [26]. We **finalize the knowledge transfer to the new domain by fine-tuning both the miner M and generator G** (by releasing its weights). The risk of overfitting is now diminished as the generative distribution is closer to the target, requiring thus a lower degree of parameter adaptation. Moreover, the training is substantially more efficient than directly finetuning the pretrained GAN [35], where synthesized images are not necessarily similar to the target samples. A mined pre-trained model makes the sampling more effective, **leading to less noisy gradients and a cleaner training signal**.

4. Experiments

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

Evaluation measures. We employ the widely used Fréchet Inception Distance (FID) [12] 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 [12]. However, it suffers from instability on small datasets. For this reason, we also employ Kernel Maximum Mean Discrepancy (KMMMD) with a Gaussian kernel and Mean Variance (MV) for some experiments [26]. Low KMMMD values indicate high quality images, while high values of MV indicate more image diversity.

Baselines. We compare our method with the following baselines. *TransferGAN* [35] directly updates both the generator and the discriminator for the target domain. *VAE* [18] is a variational autoencoder trained following [26], i.e. fully supervised by pairs of latent vectors and training images. *BSA* [26] updates only the batch normalization parameters of the generator instead of all the parameters. *DGN-AM* [25] generates images that maximize the activation of neurons in a pretrained classification network. *PPGN* [24] improves the diversity of DGN-AM by of adding a prior to the latent code via denoising autoencoder. Note that both of DGN-AM and PPGN require the target domain label, and



Figure 3: Results for off-manifold generation of MineGAN(w/o FT). We generate 20 samples of digits ‘5’, ‘8’ or ‘9’.

thus we only include them in the conditional setting.

Architectures. We apply mining to Progressive GAN [14], SNGAN [22], and BigGAN [4]. The miner has two fully connected layers for MNIST and four layers for all other experiments. More training details in Suppl. Mat. (Sec. A).

4.1. Knowledge transfer from unconditional GANs

MNIST dataset. We show results on MNIST to illustrate the functioning of the miner³. We use 1000 images of size 28×28 as target data. We test mining for off-manifold targeted image generation. In off-manifold targeted generation, G is pre-trained to synthesize all MNIST digits except for the target one, e.g. G generates 0-8 but not 9. The results in Fig. 3 are after training only the miner, without an additional finetuning step. Interestingly, the miner manages to steer the generator to output samples that resemble the target digits, by 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. 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’. In conclusion, even though target classes are not in the training set of the pretrained GAN, still similar examples might be found on the manifold of the generator.

Single pretrained model. We start by transferring knowledge from a Progressive GAN trained on *CelebA* [20]. We evaluate the performance on target datasets of varying size, using 1024×1024 images. We consider two target domains: on-manifold, *FFHQ women* [16] and off-manifold, *FFHQ children face* [16]. We refer as *MineGAN* to our full method including finetuning, whereas *MineGAN (w/o FT)* refers to only applying mining (fixed generator). We use training from *Scratch*, and the *TransferGAN* method of [35] as baselines. Figure 5 shows the performance in terms of FID and KMMMD as a function of the number of images in the target domain. MineGAN outperforms all baselines. For the on-manifold experiment, MineGAN (w/o FT) outperforms the baselines, and results are further improved with additional finetuning. Interestingly, for the off-manifold experiment, MineGAN (w/o FT) obtains only slightly worse results than TransferGAN, showing that the miner alone already manages to generate images close to the target domain. Fig. 4 shows images generated when the target data contains 100 training images. Training the model from scratch results in

³We add quantitative results on MNIST in Suppl. Mat. (Sec. D)

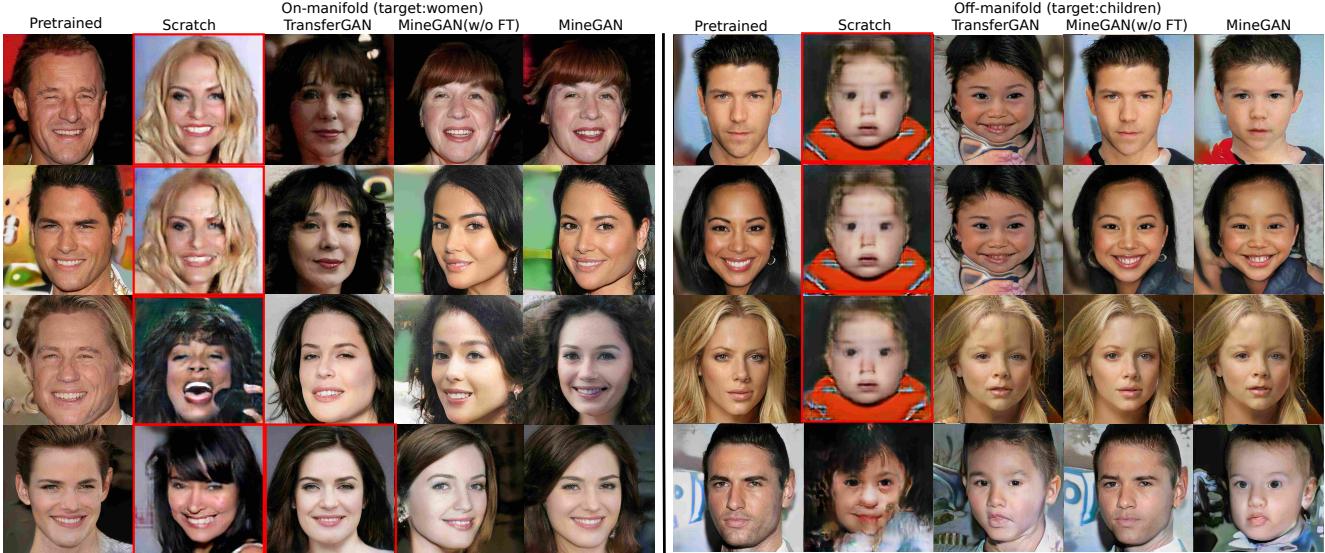


Figure 4: Results: (Left) On-manifold (CelebA→FFHQ women), (Right) Off-manifold (CelebA→FFHQ children). Based on pretrained Progressive GAN. The images in red boxes suffer from overfitting. See Suppl. Mat. (Figs. 11-14 and Sec. E) for more examples.

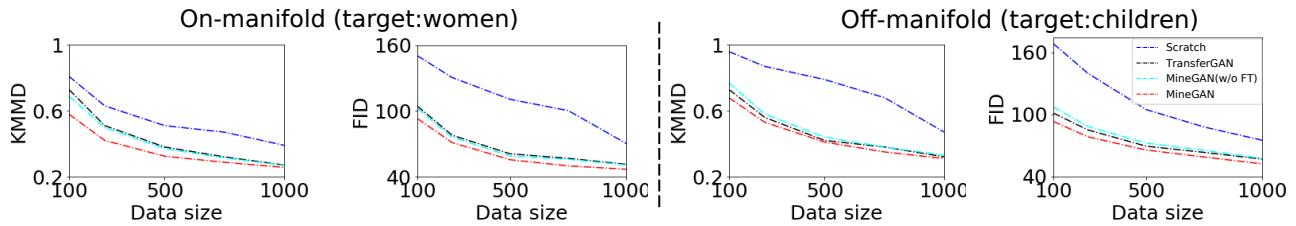


Figure 5: KMMMD and FID on CelebA→FFHQ women (left) and CelebA→FFHQ children (right).

overfitting, a pathology also occasionally suffered by TransferGAN. MineGAN, in contrast, generates high-quality images without overfitting and images are sharper, more diverse, and have more realistic fine details.

We also compare here with Batch Statistics Adaptation (BSA) [26] using the same settings and architecture, namely SNGAN [22]. They performed knowledge transfer from a pretrained SNGAN on ImageNet [19] to FFHQ [16] and to Anime Face [1]. Target domains have only 25 images of size 128×128 . We added our results to those reported in [26] in Fig. 6 (bottom). Compared to BSA, MineGAN (w/o FT) obtains similar KMMMD scores, showing that generated images obtain comparable quality. MineGAN outperforms BSA both in KMMMD score and Mean Variance. The qualitative results (shown in Fig. 6 (top)) clearly show that MineGAN outperforms the baselines. BSA presents blur artifacts, which are probably caused by the mean square error used to optimize their model.

Multiple pretrained models. We now evaluate the general case for MineGAN, where there is more than one pretrained model to mine from. We start with two pretrained Progressive GANs: one on *Cars* and one on *Buses*, both from the LSUN dataset [36]. These pretrained networks

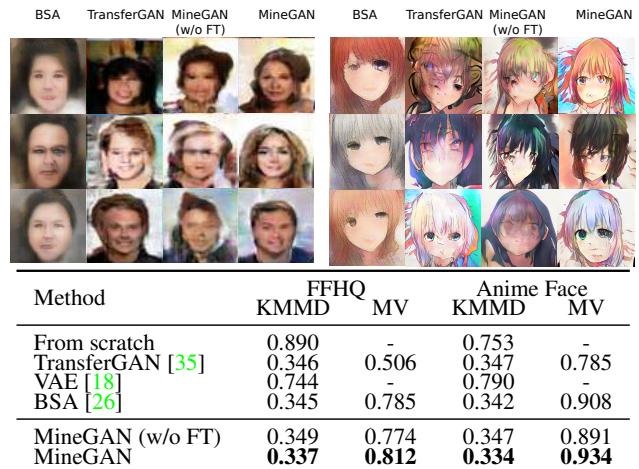


Figure 6: Results for various knowledge transfer methods. (Top) Generated images. (Bottom) KMMMD and MV.

generate cars and buses of a variety of different colors. We collect a target dataset of 200 images (of 256×256 resolution) of *red vehicles*, which contains both red cars and red buses. We consider three target sets with different car-bus ratios (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

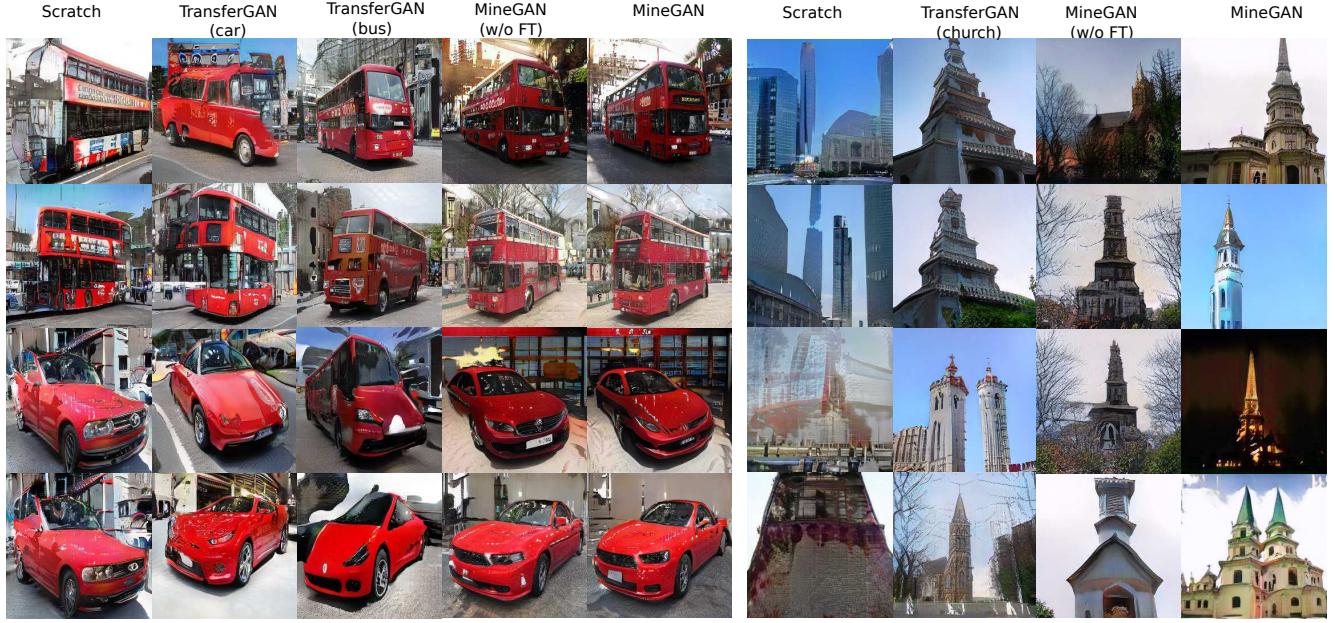


Figure 7: Results: $\{\text{car, bus}\} \rightarrow \text{red vehicles}$ (left) and $\{\text{Living room, Bridge, Church, Kitchen}\} \rightarrow \text{Tower}$ (right). Based on pretrained Progressive GAN. For TransferGAN we show the pretrained model between parentheses. More examples in Suppl, Mat (Sec. F).

Method	\rightarrow Red vehicle	\rightarrow Tower	\rightarrow Bedroom
Scratch	190 / 185 / 196	176	181
TransferGAN (car)	76.9 / 72.4 / 75.6	-	-
TransferGAN (bus)	72.8 / 71.3 / 73.5	-	-
TransferGAN (livingroom)	-	78.9	65.4
TransferGAN (church)	-	73.8	71.5
MineGAN (w/o FT)	67.3 / 65.9 / 65.8	69.2	58.9
MineGAN	61.2 / 59.4 / 61.5	62.4	54.7

Method	\rightarrow Red vehicle	\rightarrow Tower	\rightarrow Bedroom
Estimated p_i			
Car	0.34 / 0.48 / 0.64	-	-
Bus	0.66 / 0.52 / 0.36	-	-
Living room	-	0.07	0.45
Kitchen	-	0.06	0.40
Bridge	-	0.42	0.08
Church	-	0.45	0.07

Table 1: Results for $\{\text{Car, Bus}\} \rightarrow \text{Red vehicles}$ with three different target data distributions (ratios cars:buses are 0.3:0.7, 0.5:0.5 and 0.7:0.3) and $\{\text{Living room, Bridge, Church, Kitchen}\} \rightarrow \text{Tower/Bedroom}$. (Top) FID scores between real and generated samples. (Bottom) Estimated probabilities p_i for each model.

successfully generate *all types* of red vehicle, knowledge needs to be transferred from both pre-trained models.

Fig. 7 shows the synthesized images. As expected, the limited amount of data makes training from scratch result in overfitting. TransferGAN [35] produces only high-quality output samples for one of the two classes (the class that coincides with the pretrained model) and it cannot extract knowledge from both pretrained GANs. On the other hand, MineGAN generates high-quality images by successfully transferring the knowledge from both source domains simultaneously. Table 1 (top rows) quantitatively validates

Method	Label	Off-manifold		On-manifold		Time (ms)
		FID/KMMD	Label	FID/KMMD	Label	
Scratch	No	190 / 0.96	No	187 / 0.93	No	5.1
TransferGAN	No	89.2 / 0.53	Yes	58.4 / 0.39	Yes	5.1
DGN-AM	Yes	214 / 0.98	Yes	180 / 0.95	Yes	3020
PPGN	Yes	139 / 0.56	Yes	127 / 0.47	Yes	3830
MineGAN (w/o FT)	No	82.3 / 0.47	No	61.8 / 0.32	No	5.2
MineGAN	No	78.4 / 0.41	No	52.3 / 0.25	No	5.2

Table 2: 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 milliseconds.

that our method outperforms TransferGAN with a significantly lower FID score. Furthermore, the probability distribution predicted by the selector, reported in Table 1 (bottom rows), matches the class distribution of the target data.

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 (top) 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 (bottom) shows that the miner identifies the relevant pretrained models, e.g. transferring knowledge from *Bridge* and *Church* for the target domain *Tower*. Finally, Fig. 7 (right) provides visual examples.



Figure 8: Results for conditional GAN. (Left) Off-manifold (ImageNet→Places365). (Right) On-manifold (ImageNet→ImageNet).

4.2. Knowledge transfer from conditional GANs

Here we transfer knowledge from a pretrained *conditional GAN* (see Section 3.3). We use BigGAN [4], which is trained using ImageNet [30], and evaluate on two target datasets: on-manifold (ImageNet: *cock*, *tape player*, *broccoli*, *fire engine*, *harvester*) and off-manifold (Places365 [39]: *alley*, *arch*, *art gallery*, *auditorium*, *ballroom*). We use 500 images per category. We compare MineGAN with training from scratch, TransferGAN [35], and two iterative methods: DGN-AM [25] and PPGN [24]⁴. It should be noted that both DGN-AM [25] and PPGN [24] are based on a less complex GAN (equivalent to DCGAN [29]). Therefore, we expect these methods to exhibit results of inferior quality, and so the comparison here should be interpreted in the context of GAN quality progress. However, we would like 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 ensuring that the generated images follow a target distribution.

Fig. 8 shows qualitative results for the different methods. As in the unconditional case, MineGAN produces very realistic results, even for the challenging off-manifold case. Table 2 presents quantitative results in terms of FID and KMM. We also indicate whether each method uses the label of the target domain class. Our method obtains the best scores for both metrics, despite not using target label information. PPGN performs significantly worse than our method. TransferGAN has a large performance drop for the off-manifold case, for which it cannot use the target label as it is not in the pretrained GAN (see [35] for details).

⁴We were unable to obtain satisfactory results with BSA [26] in this setting (images suffered from blur artifacts) and have excluded it here.

Another important point regarding DGN-AM and PPGN is that each image generation during inference is an iterative process of successive backpropagation updates until convergence, whereas our method is feedforward. For this reason, we include in Table 2 the inference running time of each method, using the default 200 iterations for DGN-AM and PPGN. All timings have been computed with a CPU Intel Xeon E5-1620 v3 @ 3.50GHz and GPU NVIDIA RTX 2080 Ti. We can clearly observe that the feedforward methods (TransferGAN and ours) are three orders of magnitude faster despite being applied on a more complex GAN [4].

5. Conclusions

We presented a 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. Mining leads to more effective and efficient fine tuning, even with few target domain images. Our method can be applied to single and multiple pretrained GANs. Experiments with various GAN architectures (BigGAN, Progressive GAN, and SNGAN) on multiple datasets demonstrated its effectiveness. Finally, we demonstrated that MineGAN can be used to transfer knowledge from multiple domains.

Acknowledgements. We acknowledge the support from Huawei Kirin Solution, the Spanish projects TIN2016-79717-R and RTI2018-102285-A-I0, the CERCA Program of the Generalitat de Catalunya, and the EU Marie Skłodowska-Curie grant agreement No.6655919.

References

- [1] Anonymous, Danbooru community, Gwern Branwen, and Aaron Gokaslan. Danbooru2018: A large-scale crowd-sourced and tagged anime illustration dataset. <https://www.gwern.net/Danbooru2018>, 2019.
- [2] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *ICLR*, 2017.
- [3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ICLR*, 2017.
- [4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019.
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.
- [6] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, pages 1486–1494, 2015.
- [7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.
- [8] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017.
- [9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016.
- [10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.
- [11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, pages 5767–5777, 2017.
- [12] 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 *NeurIPS*, pages 6626–6637, 2017.
- [13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017.
- [14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2017.
- [15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019.
- [16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.
- [17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.
- [18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.
- [19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012.
- [20] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015.
- [21] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, pages 2794–2802, 2017.
- [22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.
- [23] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *ICLR*, 2018.
- [24] 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*, pages 4467–4477, 2017.
- [25] 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 *NeurIPS*, pages 3387–3395, 2016.
- [26] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. *ICCV*, 2019.
- [27] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724. IEEE, 2014.
- [28] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2010.
- [29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.
- [30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- [31] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, pages 2234–2242, 2016.
- [32] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *ECCV*, pages 213–229, 2018.
- [33] Michael Tschaudt, Eirikur Agustsson, and Mario Lucic. Deep generative models for distribution-preserving lossy compression. In *NeurIPS*, pages 5929–5940, 2018.
- [34] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *CVPR*, pages 4068–4076, 2015.
- [35] 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*, pages 218–234, 2018.
- [36] 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*, 2015.

- [37] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *ICML*, 2018.
- [38] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pages 649–666. Springer, 2016.
- [39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *ICLR*, 2014.
- [40] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, pages 487–495, 2014.
- [41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017.

A. Architecture and training details

MNIST dataset. Our model contains a *miner*, a *generator* and a *discriminator*. For both unconditional and conditional GANs, we use the same framework [11] 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 $|z| = 128$.

We randomly initialize the weights of each miner following a Gaussian distribution centered at 0 with 0.01 standard deviation, and optimize the model using Adam [17] with batch size of 64. The learning rate of our model is 0.0004, with exponential decay rates of $(\beta_1, \beta_2) = (0.5, 0.999)$.

In the conditional MNIST case, label c is a one-hot vector. This differs from the conditioning used for BigGAN [4] explained in Section 3.3.

Here we extend MineGAN to this type of conditional models by considering each possible conditioning as an independently pretrained generator and using the selector to predict the conditioning label. Given a conditional generator $G(c, z)$, we consider $G(i, z)$ as G_i and apply the presented MineGAN approach for multiple pretrained generators on the family $\{G(i, z) | i = 1, \dots, N\}$. 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.

CelebA Women, FFHQ Children and LSUN (Tower and Bedroom) datasets. We design the generator and discriminator based on Progressive GANs [14]. Both networks use a multi-scale technique to generate high-resolution images. Note we use a *simple* miner for tasks with dense and narrow source domains. The miner comprises out of four fully connected layers (8-64-128-256-512), each of which is followed by a *ReLU* activation and *pixel normalization* [14] except for last layer. We use a Gaussian distribution centered at 0 with 0.01 standard deviation to initialize the miner, and optimize the model using Adam [17] with batch size of 4. The learning rate of our model is 0.0015, with exponential decay rates of $(\beta_1, \beta_2) = (0, 0.99)$.

FFHQ Face and Anime Face datasets. We use the same network as [22], namely SNGAN. The miner consists of three fully connected layers (8-32-64-128). We randomly initialize the weights following a Gaussian distribution centered at 0 with 0.01 standard deviation. For this additional set of experiments, we use Adam [17] with a batch size of 8, following a hyper parameter learning rate of 0.0002 and exponential decay rate of $(\beta_1, \beta_2) = (0, 0.9)$.

Imagenet and Places365 datasets. We use the pre-trained BigGAN [4]. We ignore the projection loss in the discriminator, since we do not have access to the label of the target data. We employ a more *powerful* miner in order to allocate more capacity to discover the regions related to target domain. 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 of sizes (128, 128)-(128, 128)-(128, 128)-(128, 128)-(128, 120) and (128, 128)-(128, 128)-(128, 128)-(128, 128)-(128, 128), respectively. We use Adam [17] with a batch size of 256, and learning rates of 0.0001 for the miner and the generator and 0.0004 for the discriminator. The exponential decay rates are $(\beta_1, \beta_2) = (0, 0.999)$. We randomly initialize the weights following a Gaussian distribution centered at 0 with 0.01 standard deviation.

The input of BigGAN [4] is a random latent vector and a class label that is mapped to an embedding space. We therefore have two miner networks, the original one that maps to the input latent space (M^z) and a new one that maps to the latent class embedding (M^c). Note that since we have no class label, the miner (M^c) needs to learn what distribution over the embeddings best represents the target data.

B. Evaluation metric details

Similarly to [26], we compute FID between 10,000 randomly generated images and 10,000 real images, if possible. When the number of target image exceeds 10,000, we randomly select a subset containing only 10,000. On the other hand, if the target set contains fewer images, we cap the amount of randomly generated images to this number to compute the FID. Note we also consider KMMMD to evaluate the distance between generated images and real images since FID suffers from instability on small datasets.

C. Ablation study

In this section, we evaluate the effect of each independent contribution to MineGAN and their combinations.

Selection strategies. We ablate the use of the max operation in Eqs. (5) and (6), and replace it with a mean opera-

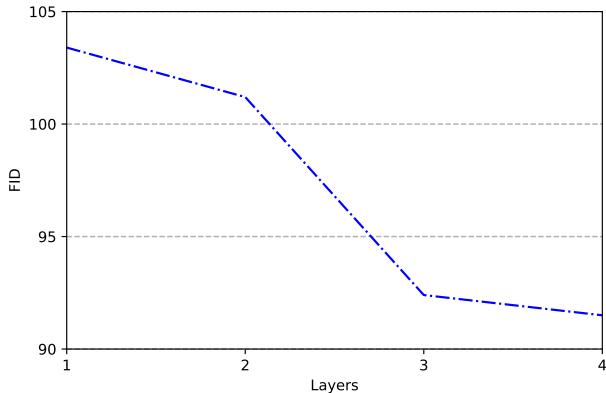


Figure 9: FID values for different number of fully connected layers in the miner. Results are based on the pre-trained *BigGAN* with target class *Arch*.

Method	MineGAN (mean)	MineGAN (max)
Car	0.51	0.34
Bus	0.49	0.66

Table 3: Estimated probabilities p_i for $\{\text{Car}, \text{Bus}\} \rightarrow \text{Red vehicles}$ for MineGAN with mean or max in Eqs. (5) and (6). The actual data distribution is 0.3:0.7 (ratio cars:buses).

tion. We call this setting MineGAN (mean). In this case the backpropagation is not only performed for the image with the highest critic score but for all images. We hypothesized that the max operation is necessary to correctly estimate the probabilities used by the selector. We report the results in Table 3, where we refer to our original model as MineGAN (max). The probability distribution predicted by the selector indicates that MineGAN (mean) equally chooses both pretrained models on Car and Bus, while MineGAN successfully estimates the class distribution of the target data.

Miner Architecture. We performed an ablation study for different variants of the miner based on pretrained BigGAN. The miner always contains only fully connected layers, but we experiment with varying the number of layers. In Fig. 9, we show the results on off-manifold target class *Arch* from Places365 [40]. Using more layers increases the performance of the method. Besides, we find that the results of both 3 and 4 layers are similar, indicating that adding additional layers would only result in slight improvements for our model. Therefore, in this paper, we use miners with 4 fully connected layers.

D. MNIST experiment

We expand the MNIST experiments presented in Section 5.1 by providing a quantitative evaluation and including results on conditional GANs. As evaluation measures, we use FID (Section 5) and classifier error [32]. To compute classifier error, we first train a CNN classifier on real

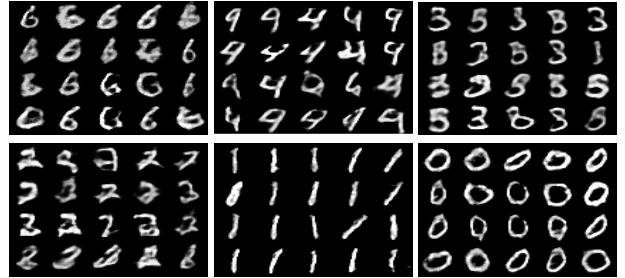


Figure 10: Results for unconditional off-manifold generation of digits ‘6’, ‘4’, ‘3’, ‘2’, ‘1’, ‘0’.

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

Table 4: Quantitative results of mining on MNIST, expressed as FID / classifier error.

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.

Table 4 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. 10 we have added a visualization for the off-manifold MNIST classes which were not already shown in Fig. 2.

E. Further results on CelebA

We provide additional results for the on-manifold experiment CelebA→FFHQ women in Fig. 11, and the off-manifold CelebA→FFHQ children in Fig. 12. In addition, we have also performed an on-manifold experiment with CelebA→CelebA women, whose results are provided in Fig. 13.

F. Further results for LSUN

We provide additional results for the experiment (<{Bus, Car}) → Red vehicles in Fig. 16 and for the experiment {Bedroom, Bridge, Church, Kitchen} → Tower/Bedroom in Fig. 17. When applying MineGAN to multiple pretrained GANs, we use one of the domains to initialize the weights of the critic. In Fig. 17 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.



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



Figure 12: (CelebA → FFHQ children). Based on pretrained *Progressive GAN*.



Figure 13: (CelebA→CelebA women). Based on pretrained *Progressive GAN*.

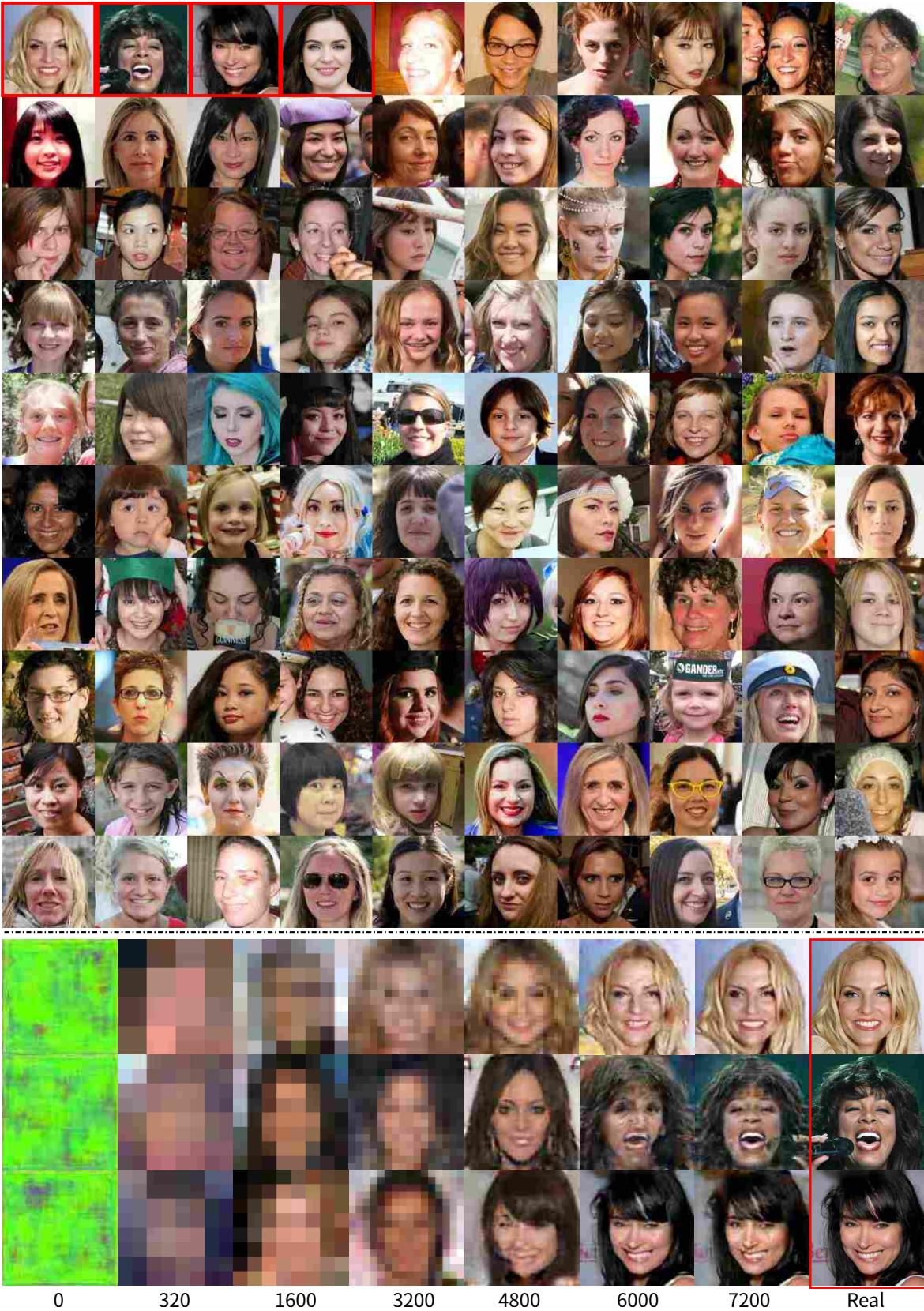


Figure 14: (Top) 100 women faces from HHFQ dataset. (Bottom) training of model from scratch: the images start with low quality and iteratively overfit to a particular training image. Red boxes identify images which are remembered by the model trained from scratch or from TransferGAN (see Fig. 4). Based on pretrained *Progressive GAN*.



Figure 15: 100 children faces from HHFQ dataset. Red boxes identify images which are remembered by the model trained from scratch (see Fig. 4). Based on pretrained *Progressive GAN*.



Figure 16: ($\{\text{bus, car}\}$) \rightarrow red vehicles. Based on pretrained *Progressive GAN*.



Figure 17: Results for unconditional GAN. (Top) (Livingroom, kitchen, bridge, church)→Tower. (Bottom) (Livingroom, kitchen, bridge, church)→Bedroom. Based on pretrained *Progressive GAN*.