HGAN: Hybrid Generative Adversarial Network

Seyed Mehdi Iranmanesh
West Virginia University
seiranmanesh@mix.wvu.edu

Nasser M. Nasrabadi
West Virginia University
nasser.nasrabadi@mail.wvu.edu

Abstract

In this paper, we present a simple approach to train Generative Adversarial Networks (GANs) in order to avoid a mode collapse issue. Implicit models such as GANs tend to generate better samples compared to explicit models that are trained on tractable data likelihood. However, GANs overlook the explicit data density characteristics which leads to undesirable quantitative evaluations and mode collapse. To bridge this gap, we propose a hybrid generative adversarial network (HGAN) for which we can enforce data density estimation via an autoregressive model and support both adversarial and likelihood framework in a joint training manner which diversify the estimated density in order to cover different modes. We propose to use an adversarial network to transfer knowledge from an autoregressive model (teacher) to the generator (student) of a GAN model. A novel deep architecture within the GAN formulation is developed to adversarially distill the autoregressive model information in addition to simple GAN training approach. We conduct extensive experiments on real-world datasets (i.e., MNIST, CIFAR-10, STL-10) to demonstrate the effectiveness of the proposed HGAN under qualitative and quantitative evaluations. The experimental results show the superiority and competitiveness of our method compared to the baselines.

1. Introduction

Generative models have extensively grown in recent years. The main goal of a generative model is to approximate the true data distribution which is not known. Generative models are based on finding the model parameters that maximize the likelihood of the training data. This is equivalent to minimizing the Kullback-Leibler (KL) divergence ($D_{KL}(p_{data}||p_{model})$) between the data distribution p_{data} and model distribution p_{model}. Although this objective spans multiple modes of the data, it leads to generating vague and undesirable samples [45]. There are other approaches that minimize $D_{KL}(p_{model}||p_{data})$ which are usually referred to as the reverse KL divergence [34] and this is the main idea behind the generative adversarial networks. Although these models generate sharp images, minimizing the reverse KL divergence causes the model distribution to focus on a single mode of the data and ignore the other modes. This is known as the mode collapse in the generative adversarial models [33]. This happens because the reverse KL divergence measures the dissimilarity between two distributions for the fake samples, and there is no penalty on the fraction of the model distribution that covers the data distribution [2]. To address this problem, the authors in [1] suggested the Wassertein distance which has the weakest convergence among existing GANs. However, they used weight clipping to approximate the Wassertain distance which causes a pathological behavior [33].

In general, the choice for modeling the density function is challenging. There are two ways to estimate the density function namely, implicit methods and explicit methods. Implicit approaches tend to calculate the model parameters without the need for the analytical form of p_{model}. Explicit models have the advantage of explicitly calculating the probability densities. There are two well-known implicit approaches, namely GAN and Variational AutoEncoder (VAE) which try to model the data distribution implicitly. The VAEs try to maximize the data likelihood lower bound, while a GAN performs a minimax game between two players during its optimization in which for an optimal discriminator, the algorithm tries to find a generator that minimizes the Jensen-Shannon divergence (JSD). The JSD minimization has been proven empirically to behave more similar to the reverse KL divergence rather than the KL divergence [21, 33]. This behavior leads to the aforementioned problem of mode collapse in GAN models, which causes the generator to create similar looking images with poor diversity of samples.

In contrast to VAE models which implicitly compute the likelihood of the data space, autoregressive models have the advantage of tractable likelihood and can generate diverse samples. The basic idea of these models is to use the autoregressive connections to model an image pixel by pixel. In fact, autoregressive approaches model the joint distribution of pixels in the image as the product of conditional dis-

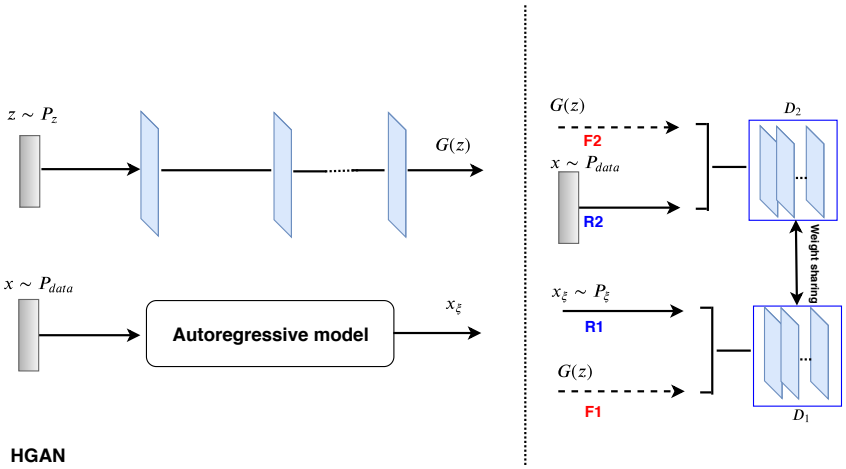


Figure 1. Proposed HGAN framework with an autoregressive model, a generator, and a discriminator is trained by using two types of real data.

tributions [35]. However, these models suffer from a slow synthesis when compared to GANs.

The lack of explicit density function in GANs is problematic for two main reasons. Many applications in deep generative models are based on the density estimation. For instance, the count-based exploration methods [36] rely on density estimation have achieved state-of-the-art performance on reinforcement learning environments [15]. The second reason is that the quantitative evaluation of the generalization performance of such models is challenging. Since GANs typically are able to generate sharp samples by memorizing the training data, the evaluation criteria based on ad-hoc sample quality metrics [42] does not capture the mode collapse issue.

Recently some approaches have been trying to solve the mode collapse issue. MGAN [20] trains many generators by using a classifier and a discriminator. Using many generators and also a classifier in addition to the classical GAN make this model computationally complex and prone to over-fit the training dataset. There are some other approaches that attempt to use autoencoders as regularizers or additional losses to penalize the missing modes [48, 49]. In [51] authors used an LSTM-based autoregressive model in their discriminator function and considered the reconstruction loss as the penalty for fake data. However, in their GAN model they trained their discriminator only on the true data as it becomes unbounded for the fake data synthesized by the generator. SNGAN [31] utilizes spectral normalization to stabilize the training of discriminator. It controls the Lipschitz constant of the discriminator to mitigate the exploding gradient problem and the mode collapse issue. SAGAN [52] uses the self attention layer to capture the fine details from distant part of image. The authors combined their model with spectral normalization on both the generator and discriminator. BigGAN [4] is designed for class-conditional image generation. The focus of the BigGAN model is to increase the number of model parameters and batch size, then configure the model and training process. It utilizes the recent techniques introduced by SNGAN [31] and SAGAN [52]. StyleGAN [22], is an approach for training generator models capable of synthesizing very large high-quality images via the incremental expansion of both discriminator and generator models from small to large images during the training process. In addition, it changes the architecture of the generator significantly.

We propose a simple effective GAN architecture and a training strategy with the goal of adversarially distilling the explicit information of the data distribution provided by the autoregressive model in addition to mimicking the real data which leads to generating samples with a distribution very close to the actual data distribution and helps to avoid possible mode collapse. To resolve the issue of sharp good-looking samples but poor likelihood estimation in the case of adversarial learning (and vice versa in the case of maximum likelihood estimation), our proposed hybrid model bridges implicit and explicit learning models by augmenting the adversarial learning with an additional autoregressive model. Our approach combines the implicit and explicit density function estimation into a unified objective function. In our model, the HGAN generator is guided by exploiting the explicit data probability density from the knowledge provided by the autoregressive model while it is also responsible to learn the data distribution via the adversarial learning. HGAN model exploits the complementary statistical properties of data obtained from an autoregressive model by utilizing a GAN to effectively diversify the estimated density function and capturing different modes of the data distribution as well as avoiding possible mode collapse.

In short, our main contributions are: (i) a novel adversarial model to train a generator in a GAN framework in order to stabilize the training process; (ii) the proposed model is able to estimate the data density by mimicking an autoregressive model and simultaneously combining it with the adversarial learning process; (iii) a comprehensive performance evaluation of our proposed method on real-world large-scale datasets of diverse natural scenes as well as mitigating adversarial examples in a defense scenario.

2. Background

2.1. Generative Adversarial Nets:

GAN is a min-max game between a generator G and a discriminator D, both parameterized via neural networks [13]. Training a GAN can be formulated as the following objective function:

$$\min_G \max_D E_{x\sim P_{data(x)}}[logD(x)]+ \\ E_z \sim P_z[log(1-D(G(z)))], \tag{1}$$

where x is from a real data distribution P_{data} and z is a sample from a prior distribution P_z. The generator is a mapping function from z which approximates P_{model}. GAN alternatively optimizes D and G in a minimax game using stochastic gradient-based algorithm. Generator is prone to map every z to a single x that is most probable to be recognized as a true data, and this leads to a mode collapse. Another issue with GAN is that at optimal point of D, minimizing the generator is equal to minimizing the JSD between the true data distribution and model distribution which is empirically shown to cause the mode collapse by generating few modes and ignoring other modes [21].

2.2. Autoregressive Models:

Autoregressive models can be designed by using recurrent networks (PixelRNNs) or a CNN (PixelCNNs) [47]. These models learn the join distribution of pixels of an image x as a product of conditional distributions $p(x_i|x_1,...,x_{i-1})$, where x_i is a single pixel:

$$p(x) = \prod_{i=1}^{\mathrm{n}^2} p(x_i|x_1,...,x_{i-1}). \tag{2}$$

The ordering of pixel dependencies is row by row and in each row, pixel by pixel. Therefore, every pixel (x_i) depends on all the pixels above and left of it ($x_1,...,x_{i-1}$).

2.3. Knowledge distillation:

Knowledge distillation is mostly used in image classification problem where the output of neural network is a probability distribution over categories. The probability is calculated by applying a softmax function over *logits* which are the output of the last fully connected layer. Hinton et al. [19] used logits to transfer the embedded information in a teacher network to student network. In order to train a student network F to generate student logits $F(x_i)$, a parameter called temperature T is introduced. Afterwards, the generalized softmax layer converts logits vector $t_i = (t_i^1,...,t_i^C)$ to a probability distribution q_i,

$$M_T(t_i) = q_i, \quad where \quad q_i^j = \frac{exp(t_i^j/T)}{\sum_k exp(t_i^k/T)}. \tag{3}$$

where higher temperature T produces softer probability over categories.

Hinton et al. [19] proposed to minimize the KL divergence between teacher and student output as follows:

$$L_{KD}(F,T) = \frac{1}{N}\sum_{i=1}^{N} KL(M_T(t_i)||M_T(F(x_i))). \tag{4}$$

In [50] instead of forcing the student to exactly mimic the teacher by minimizing KL-divergence in Equation (4), the knowledge is transferred from teacher to student via discriminator in a GAN-based approach.

3. Proposed Hybrid GAN:

We now present our novel hybrid approach to tackle the problem of mode collapse in GANs. In general, GANs can generate good-looking samples but have intractable likelihoods. On the other hand, autoregressive models are likelihood-based generative models which can return explicit probability densities. The idea is to utilize a mixture of these two models rather than a single model as in a typical GAN.

In our proposed hybrid model, the generator's first task is to learn the data distribution without any explicit model just like a regular GAN such that $G(z) \simeq x \sim P_{data}$. On the other hand, its second task is to perform sampling where it samples a random vector $z \sim P_z$ and maps it to an autoregressive model P_ξ such that $G(z) \simeq x_\xi \sim P_\xi$. This forces our hybrid model to learn the probability density of the autoregressive model using the adversarial training method. These two tasks together provide a hybrid model which gives more attention to the likelihood of the data for estimating P_{model} in the data space.

A natural question to ask is why one should use adversarial learning when autoregressive model can return a tractable likelihood. The reason is that the synthesis from these autoregressive models are difficult to parallelize and usually inefficient on parallel hardware [24]. Moreover, it is not practical to perform accurate data manipulation since the hidden layers of autoregressive models have unknown

Figure 2. Samples generated by the proposed HGAN compared with the samples generated from DCGAN and AutoGAN on CIFAR-10.

Algorithm 1: HGAN Training procedure using stochastic gradient descent

Input : minibatch images x, number of training batch steps S

1 $\theta_\xi, \theta_G, \phi_D \leftarrow$ initialize network parameters

2 **for** $n = 1$ *to* S **do**

3 $x_\xi \leftarrow p_\xi(x)$ {Forward through auto model }

4 $z \sim \mathcal{N}(0,\ 1)^Z$ {Draw sample of random noise }

5 $\hat{x} \leftarrow G(z)$ {Forward through the generator }

6 $s_{r_1} \leftarrow D(x_\xi)$ {Auto model output}

7 $s_{f_1} \leftarrow D(\hat{x})$ {$G(z)$ output}

8 $\hat{x} \leftarrow G(z)$ {Forward through the generator }

9 $s_{r_2} \leftarrow D(x_\xi)$ {Real data}

10 $s_{f_2} \leftarrow D(x_\xi, \hat{x})$ {$G(z)$ output}

11 $\mathcal{L}_D \leftarrow \log(s_{r_1}) + \log(s_{r_2}) + \log(1 - s_{f_1}) + \log(1 - s_{f_2})$

12 $\phi_D \leftarrow \phi_D - \frac{\partial \mathcal{L}_D}{\partial \phi_D}$ {Update discriminator}

13 $\mathcal{L}_G \leftarrow log(s_{f1}) + log(s_{f2})$

14 $\theta_G \leftarrow \theta_G - \frac{\partial \mathcal{L}_G}{\partial \theta_G}$ {Update generator}

15 $\mathcal{L}_{p_\xi} \leftarrow |x_i - p_\xi(x_i)|$

16 $\theta_\xi \leftarrow \theta_\xi - \frac{\partial \mathcal{L}_{p_\xi}}{\partial \theta_\xi}$ {Update auto model}

17 **end**

marginal distributions [15]. However, GAN models are fast in synthesizing and also can have useful latent space for downstream tasks especially in ones which have an encoder such as AGE or ALI [10, 46]. Fig. 1 illustrates the architecture of our proposed Hybrid GAN (HGAN) model.

In naive GAN, the odds that the two distributions p_g and p_{data} share support in high-dimensional space, especially early in training, are very small. If p_g and p_{data} have non-overlapping support the Jensen-Shannon divergence is saturated as is locally constant in θ. Also, there might be a large set of near-optimal discriminators whose logistic regression loss is very close to optimum, but each of these possibly provides very different gradients to the generator. Therefore, training the discriminator might find a different near-optimal solution each time depending on initialization, even for a fixed g_θ and p_{data}. We instead employ autoregressive model to augment the gradient information obtained by ordinary back-propagation. In fact, we are interested in manipulating the feature space of a discriminator, using the autoregressive model as a tool to tell us *how* to perform

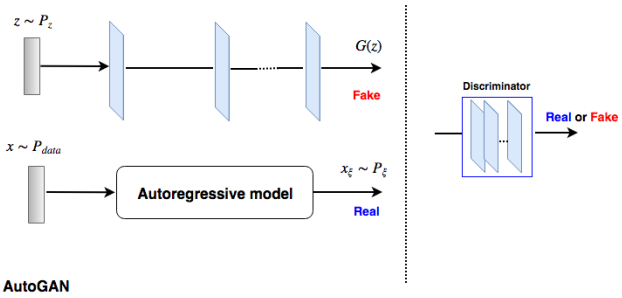


Figure 3. Training $G(z)$ to mimic the autoregressive model's output with an adversarial learning process. In this network which we denote as AutoGAN, the real data is obtained from the autoregressive model's output and fake data is the generated output from $G(z)$.

that manipulation.

In our Hybrid GAN, the discriminator observes two types of real inputs: the real data x and the output of the autoregressive model x_ξ. The fake input, $G(z)$, is mimicking the output of the autoregressive model $x_\xi \sim P_\xi$ in addition to the real data $x \sim P_{data}$. We consider two terms for the discriminator D, namely D_1 and D_2. D_1 is the first discriminator which is related to the first task, where $G(z)$ is fake and x_ξ is real. D_2 is related to the second task, where $G(z)$ is fake and x is real. However, all the parameters are shared between discriminators D_1 and D_2 and in fact there is only one discriminator D.

Algorithm 1 summarizes the training procedure. After getting the input image, output of the autoregressive model, and noise, the proposed model generates a fake image (line 5). s_{r_1} and s_{r_2} indicate the scores for the first (x_ξ) and second real inputs (x). s_{f_1} and s_{f_2} measures the score of fake inputs containing the generator's output ($G(z)$) trying to mimic the first and second real input, respectively. Note that we use $\frac{\partial \mathcal{L}_D}{\partial \phi_D}$ to indicate the gradient of D's objective function with respect to its parameters, and likewise for G and p_ξ.

In the first fake input of discriminator, the generator attempts to generate data that is as close as possible to the

Table 1. Experiment on MNIST dataset containing 10 different modes.

GAN Variants	Chi-square ($\times 10^5$)	KL Div
WGAN	1.32	0.614
MIX+WGAN	1.25	0.567
DFM	1.46	0.623
Improved-GAN	1.13	0.436
ALI	2.34	0.875
BEGAN	1.06	0.944
MAD-GAN	0.24	0.145
GMAN	1.86	1.345
DCGAN	0.90	0.322
MGAN	0.32	0.211
SAGAN	0.29	0.148
SNGAN	0.25	0.146
HGAN	**0.23**	**0.141**

autoregressive model's output. Therefore, the generator's task is to make $G(z) \simeq x_\xi \sim P_\xi$. However, for the second round of fake input, the generator tries to fool the discriminator in a way that its generated data is as close as possible to the real data. Thus, it is responsible to make $G(z) \simeq x \sim P_{data}$. While G acts similar to a typical generator in a regular GAN, our hybrid method tries to maximize the likelihood of a mixture model by adversarially distilling the properties of autoregressive model.

4. Experiments

We show the effectiveness of our proposed approach in different experiments with real-world datasets. For the fair evaluation, we use the same experimental settings that are identical to prior works [33, 12, 20]. Therefore, we use the results from the latest state-of-the-art GAN-based models to compare with ours.

We used Pytorch [38] to implement our framework. The generator and discriminator architecture is adopted from DCGAN [39]. In addition, pixelCNN++ [43] architecture is chosen for the autoregressive model. For training we used Adam optimizer [23] with the first-order momentum of 0.5, the learning rate of 0.0002, and batch size of 64. For the generator the ReLU activation [32], and for the discriminator the Leaky ReLU activation with the slope of 0.2 is considered. Weights are initialized from an isotropic Gaussian: $\mathcal{N}(0,\, 0.01)$ and zero biases. To show the effectiveness of the proposed framework, we perform two types of experiments on MNIST dataset and compare our methods to the other well-know GANs, namely WGAN [1], MIX+WGAN [42], DFM [49], Improved-GAN [42], ALI [10], BEGAN [3], MAD-GAN [12], GMAN [11], DCGAN [39], MGAN [20], SNGAN [31], and SAGAN [52]. It should be noted that our method cannot be compared directly with BigGAN [4] and StyleGAN [22] since the mentioned models are based on larger models and different settings (i.e., BigGAN is using class conditional setting or StyleGAN purpose is for having more control over the latent space for high resolution image generation). Following [12], we reuse the KL-divergence [27] and the number of captured modes [6] as the criteria for the comparison to illustrate the superiority of our method compared to others. Moreover, we perform the quantitative experiments on more complicated real-world datasets namely the CIFAR-10 [25] and STL-10 [8] datasets.

Table 2. Results for the Inception scores on CIFAR-10 dataset.

Objective	Inception Score
DCGAN	6.40
AutoGAN	6.17
HGAN	**7.46**

Table 3. Results for the test MODE scores on the MNIST dataset.

Objective	MODE Score
DCGAN	9.28
AutoGAN	9.32
HGAN	**9.51**

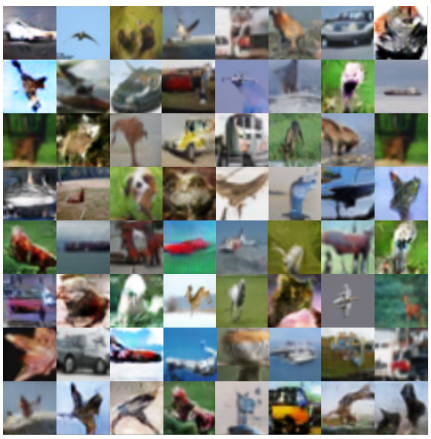
4.1. MNIST

The data distribution of the MNIST dataset can be approximated with ten dominant modes. Here, following [6] we define the term 'mode' as a connected component in the data manifold.

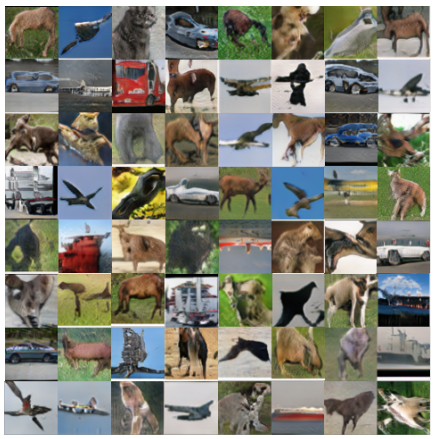
For the sake of evaluation, we train a four-layer CNN classifier on the MNIST digits and then apply it to compute the mode scores in the generated samples from the proposed method. We repeat the procedure and apply the trained classifier to discover the mode scores on different baseline GAN methods. We also have the ground truth by measuring the performance of classifier on the MNIST test set. The number of generated samples from each method is equal to the number of test set which is 10,000. Afterwards, we use Chi-square distance and the KL-divergence to compute distance between the two histograms (ground truth vs. each GAN model). Table 1 shows the performance of our proposed HGAN compared to the other methods. From Table 1, it is evident that our proposed method could outperform the other methods in capturing all the modes in the MNIST dataset.

4.2. Stacked and Compositional MNIST

In this experiment, the goal is to explore the performance of our proposed HGAN in a more challenging scenario. In order to illustrate and compare HGAN with other baselines, we utilized similar setup as in [12]. Authors in [30] created a Stacked MNIST with 25,600 samples where each sample has three channels stacked together with a random digit from MNIST in each of them. Therefore, the Stacked MNIST contains 1,000 distinct modes in the data distribution. In [7], a similar process is applied to MNIST dataset. They created the Compositional MNIST whereby they took



(a) *CIFAR* – 10



(b) *STL* – 10

Figure 4. Images generated by our proposed HGAN trained on natural image datasets.

Table 4. Stacked-MNIST experiment. There are 1,000 modes in the dataset.

GAN Variants	KL Div	# Mode Covered
WGAN	1.02	868
MIX+WGAN	0.98	874
DFM	1.13	843
Improved-GAN	1.45	847
ALI	2.03	802
BEGAN	1.89	819
MAD-GAN	0.91	890
GMAN	2.17	756
DCGAN	2.15	712
MGAN	0.94	**896**
SAGAN	0.97	886
SNGAN	0.91	889
HGAN	**0.88**	891

Table 5. Compositional-MNIST experiment. There are 1,000 modes in the dataset.

GAN Variants	KL Div	# Mode Covered
WGAN	0.25	1000
MIX+WGAN	0.21	1000
DFM	0.23	965
Improved-GAN	0.67	934
ALI	1.23	967
BEGAN	0.19	999
MAD-GAN	0.074	1000
GMAN	0.57	929
DCGAN	0.18	980
MGAN	0.12	1000
SAGAN	0.095	1000
SNGAN	0.083	1000
HGAN	**0.078**	**1000**

three random MNIST digits and placed them at three quadrants of a 64×64 dimensional image. This also resulted in a data distribution with 1,000 modes. Distribution of the generated samples was estimated with a pre-trained MNIST classifier which classifies the digits in each channel or quadrants, and consequently decides which of the 1,000 modes is generated by the particular GAN method's generator.

Table 4 and 5 show the performance of the proposed method as well as other GAN methods in terms of the KL divergence and the number of modes recovered for the Stacked and Compositional MNIST datasets. As shown in Table 4, our method outperformed all the other GAN methods in terms of the KL divergence and the number of captured modes. MGAN surpasses ours in only the number of captured modes. It is evident from Table 5 that our proposed HGAN outperforms all the other baselines in terms of the KL divergence and it is the closest to the true data distribution. Also, in terms of the number of captured modes, our method as well as MGAN, MAD-GAN, WGAN, SNGAN and MIX+WGAN capture all the 1,000 modes in the Compositional MNIST experiment.

Table 6. Inception scores on the CIFAR-10 and STL-10 datasets.

Model	CIFAR-10	STL-10
Real data	11.24 ± 0.16	26.08 ± 0.26
WGAN	3.82 ± 0.06	–
MIX+WGAN	4.04 ± 0.07	–
DFM	7.72 ± 0.13	8.51 ± 0.13
Improved-GAN	4.36 ± 0.04	–
ALI	5.34 ± 0.05	–
BEGAN	5.62	–
MAD-GAN	7.34	–
GMAN	6.00 ± 0.19	–
DCGAN	6.40 ± 0.05	7.54
MGAN	$\mathbf{8.33 \pm 0.10}$	$\mathbf{9.22 \pm 0.11}$
SAGAN	7.51 ± 0.15	8.61 ± 0.11
SNGAN	7.58 ± 0.12	8.79 ± 0.14
HGAN	7.46 ± 0.11	8.94 ± 0.13

4.3. Real-world Datasets

In this section, the proposed HGAN framework is applied on more complicated real-world datasets to evaluate its effectiveness on more challenging large-scale image

Table 7. FIDs on CIFAR-10 and STL-10 (lower is better).

Model	DCGAN	DCGAN+TTUR [18]	WGAN-GP [16]	GAN-GP	MGAN	SAGAN	SNGAN	HGAN
CIFAR-10	37.7	36.9	40.2	37.7	26.7	26.3	**25.5**	26.1
STL-10	-	-	55.1	-	-	43.6	43.2	**42.1**

Table 8. Classification accuracies of using Defense-GAN and Defense-HGAN strategies on the MNIST dataset with L = 200 and R = 10.

Attack	No Attack (Defense-GAN)	Defense-GAN	No Attack (Defense-HGAN)	Defense-HGAN
FGSM ($\epsilon = 0.3$)	0.989	0.961	0.991	0.974
PGD	0.989	0.956	0.991	0.969
CW (l_2 norm)	0.989	0.945	0.991	0.965

data.

4.3.1 Datasets.

We use two widely-adopted datasets, namely CIFAR-10 [25] and STL-10 [9]. CIFAR-10 dataset contains 50,000 training images with the resolution of 32×32 for 10 different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. STL-10 dataset subsampled from the ImageNet [41] and is more diverse database compared to CIFAR-10. This dataset composed of 100,000 images with the resolution of 96×96. For the sake of fair comparison with the baselines in [49], we follow the same procedure as in [26] to resize the STL-10 down to 48×48.

4.3.2 Evaluation Protocols.

For quantitative evaluation, we consider the Inception score which was introduced in [42]. This metric computes $exp(\mathbb{E}_x[D_{KL}(p(y|x)||p(y))])$, where $p(y|x)$ is the conditional label distribution for image x estimated by the reference Inception model. The metric rewards good and varied samples and is found to be well-correlated to human judgment. The code provided in [42], is used to compute the Inception score for 10 partitions of 50,000 generated samples. For qualitative evaluation of the quality of images generated by our proposed HGAN framework, we show the samples generated by HGAN which are drawn randomly rather than cherry-picked.

4.3.3 Inception Results.

Table 6 shows the Inception scores obtained by our proposed HGAN method as well as the baselines. For the fair comparison, only models which are trained completely in an unsupervised manner without the label information are included in Table 6. Also, the reported results on STL-10 for DCGAN and D2GAN are based on the models trained on 32×32 resolution. Table 6 shows the superiority of our proposed HGAN compared to the other methods in the literature for both the STL-10 and CIFAR-10 datasets.

4.3.4 Image Generation.

For the qualitative assessment, we present samples which are randomly selected from the images generated by the proposed HGAN. It can be seen from Fig. 4 that the images generated by HGAN are visually recognizable images of cars, ships, trucks, birds, airplanes, dogs, and horses in the CIFAR-10 database. Moreover, in the case of the STL-10 dataset, HGAN is able to produce images including car, trucks, ships, airplanes, and different kinds of animals including horses, cats, monkeys, deers, and dogs with wider range of background such as sky, cloudy sky, sea, and forest. These visually appealing images confirms the diversity of the generated samples by HGAN.

4.4. Frechet Inception Distance results.

The main disadvantage of the inception score is that it does not compare the statistics of the synthetic samples and the real world ones. Therefore, we evaluate HGAN using the Frechet Inception Distance (FID) proposed in [18]. Table 7 compares the FIDs obtained by HGAN with baselines collected in [20, 31]. It should be noted that some methods in the literature use the Resnet [17] architecture. Here, for the fair comparison we show the results of different methods when using DCGAN architecture.

4.5. Ablation Study

In the previous sections, we examined the mode coverage of the proposed framework compared to the other baselines in three separate experiments. In order to show the effectiveness of our HGAN framework, we perform another experiment with two different datasets, namely the MNIST and CIFAR-10 datasets. In this setup, we consider $G(z)$ within two complete separate training approaches. In the first approach, $G(z)$ is trained as a regular GAN such as a DCGAN, and in the second approach $G(z)$ is trained to mimic the autoregressive model's output with an adversarial training. We denote the first approach as DCGAN and the second approach as AutoGAN. Fig. 3 depicts the framework of AutoGAN. We compare the performance of these two networks with the proposed HGAN in terms of sample quality.

Table 3 and 2 show the highest Inception/MODE scores [42] of DCGAN, AutoGAN, and HGAN monitored

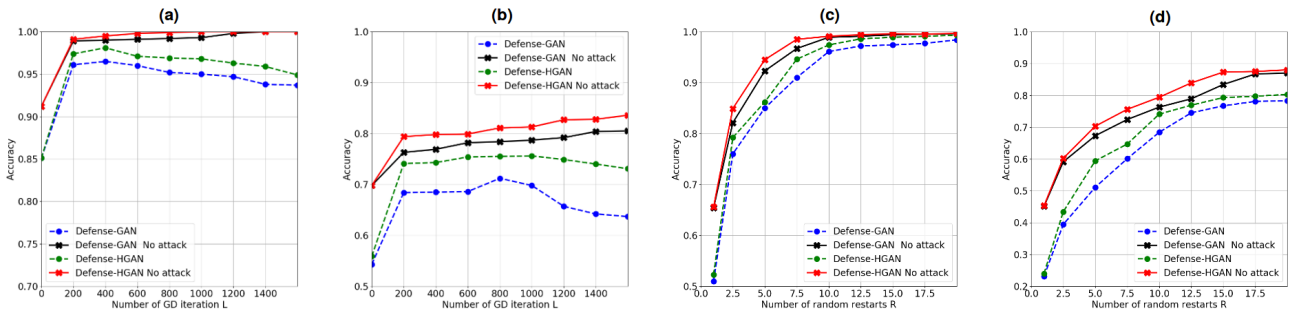


Figure 5. Classification accuracy of Defense-GAN and Defense-HGAN on the MNIST and CIFAR-10 datasets in the case of no attack and also under FGSM white-box attack with $\epsilon = 0.3$. (a) MNIST classification accuracy varying L (with R = 10). (b) CIFAR-10 classification accuracy varying L (with R = 10). (c) MNIST classification accuracy varying R (with L = 100). (d) CIFAR-10 classification accuracy varying R (with L = 100).

Table 9. Classification accuracies of using Defense-GAN and Defense-HGAN strategies on the CIFAR-10 dataset with L = 200 and R = 10.

Attack	No Attack (Defense-GAN)	Defense-GAN	No Attack (Defense-HGAN)	Defense-HGAN
FGSM ($\epsilon = 0.3$)	0.763	0.684	0.794	0.741
PGD	0.763	0.671	0.794	0.738
CW (l_2 norm)	0.763	0.646	0.794	0.731

during the training phase. The samples generated by each of the mentioned methods on the CIFAR-10 dataset is also shown in Fig. 2.

As it is illustrated in Table 3 and 2, HGAN outperforms both DCGAN and AutoGAN in terms of sample quality. One possible reason behind this is in HGAN, the addition of adversarially distillation of the data information from the autoregressive model (pixelCNN++) in the $G(z)$ objective function can stabilize its optimization, thus avoiding the mode collapse issue. Finally, the hybrid nature of the proposed method leads to a better performance for both datasets.

4.6. Comparison with WGAN in Defense Framework

Despite a very rich research work leading to very interesting GAN algorithms, it is still challenging to assess which algorithm performs better compared to others. In this experiment we evaluate the effectiveness of HGAN compared to WGAN in a defense scenario. We believe this could be another way of assessment for GAN frameworks.

Adversarial examples [40] are neural network inputs which are designed to force misclassification. These inputs often appear normal to humans while cause the neural network to make inaccurate predictions. Various defenses have been proposed to mitigate the effect of adversarial attacks [44, 37, 29]. In this experiment we use our proposed HGAN as a defense mechanism against three different white-box attacks: Fast Gradient-Sign Method (FGSM) [14], Carlini-Wagner (CW) attack (with l_2 norm) [5], and Projected Gradient Descent (PGD) [28]. For the fair comparison, we adopt the same set of experiment as Defense-GAN [44]. Instead of using WGAN, we use our proposed HGAN in Defense-GAN framework which we denote as Defense-HGAN. We also compare Defense-HGAN with Defense-GAN in the case of no attack. Table 8 and 9 show the classification performance of our method compared to Defense-GAN on the MNIST and CIFAR-10 datasets, respectively. It should be noted that the classification accuracy results on the MNIST and CIFAR-10 is 0.994 and 0.886, respectively. We note that Defense-HGAN outperforms Defense-GAN which shows the superiority of our HGAN comparing to WGAN in Defense-GAN framework.

We also compared the effect of different numbers of iterations L and random restarts R for Defense-GAN and Defense-HGAN on the MNIST and CIFAR-10 datasets. Both methods need to look for an appropriate datapoint in the latent space which leads to generating an image closer to the input image. As it is shown in Fig. 5 classification performance of HGAN is better than Defense-GAN which means that HGAN could do a better job in capturing the data distribution compared to WGAN on the MNIST and CIFAR-10 datasets.

5. Conclusion

We have proposed a novel approach to address the mode collapse issue in GANs. Our idea is to design a hybrid model which tries to learn the distribution of data via a mixture of density estimating models utilizing an autoregressive model and an adversarial learning. For this purpose, we introduce a minimax game between a generator, an autore-

gressive model, and a discriminator to optimize the problem of minimizing the JSD between P_{data} and P_{model}. In our proposed HGAN, the generator is responsible to learn the autoregressive model output in addition to modeling the real data just like a regular GAN. Distillation of autoregressive model is beneficial for the HGAN since it also models the distribution of the same data but in an explicit way. It makes the generator to give more attention to the likelihood of the data and stabilize its optimization. This helps the proposed model to capture more data modes which leads to generating a more diversified set of images. Comprehensive study on MNIST and also more challenging real-world datasets show the effectiveness of our HGAN in covering data modes and avoiding mode collapse as well as generating diverse and visually appealing images.

References

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

[2] Duhyeon Bang and Hyunjung Shim. Mggan: Solving mode collapse using manifold guided training. *arXiv preprint arXiv:1804.04391*, 2018.

[3] David Berthelot, Thomas Schumm, and Luke Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[6] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[7] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[8] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[9] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[10] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[11] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

[12] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *CoRR, abs/1704.02906*, 6:7, 2017.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples (2014). *arXiv preprint arXiv:1412.6572*.

[15] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. *arXiv preprint arXiv:1705.08868*, 2017.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] 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 *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[20] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Mgan: Training generative adversarial nets with multiple generators. 2018.

[21] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[25] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

[27] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[28] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[29] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.

[30] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

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

[32] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[33] Tu Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2670–2680, 2017.

[34] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[35] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[36] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

[37] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

[40] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[41] 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. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[43] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[44] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

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

[46] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Adversarial generator-encoder networks. *CoRR, abs/1704.02304*, 2017.

[47] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[48] Ruohan Wang, Antoine Cully, Hyung Jin Chang, and Yiannis Demiris. Magan: Margin adaptation for generative adversarial networks. *arXiv preprint arXiv:1704.03817*, 2017.

[49] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. 2016.

[50] Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. Learning loss for knowledge distillation with conditional adversarial networks. *arXiv preprint arXiv:1709.00513*, 2017.

[51] Yasin Yazici, Kim-Hui Yap, and Stefan Winkler. Autoregressive generative adversarial networks. 2018.

[52] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.