Learning a Multimodal Prior Distribution for Generative Adversarial Nets

Thomas Goerttler[1⋆] and Marius Kloft[2⋆]

[1] Neural Information Processing Group, Department of Electrical Engineering and Computer Science, Technical University of Berlin, Germany
thomas.goerttler@ni.tu-berlin.de
[2] Department of Computer Science, Technical University of Kaiserslautern, Germany
kloft@cs.uni-kl.de

Abstract. Generative adversarial nets (GANs) have shown their potential in various tasks like image generation, 3D object generation, image super-resolution, and video prediction. Nevertheless, they are still considered as highly unstable to train and are endangered to miss modes. One problem is that real data is usually discontinuous, whereas the prior distribution is continuous. This circumstance can lead to non-convergence of the GAN and makes it hard for the generator to generate fair results. In this paper, we introduce an approach to directly learn modes in the prior distribution - which map to the modes in the real data - by changing the training procedure of GANs. Our empirical results show that this extension stabilizes the training of GANs, and it captures discrete uniform distributions fairer. We use the score of the earth mover's distance as an evaluation metric to underline this effect.

Keywords: Generative Models · Mode Collapse · Learning Latent Distributions

1 Introduction

In 2014 generative adversarial nets (GANs) [8] were proposed as a novel generative model, which does not formulate the distribution of training data explicit but instead allows to sample additional data coming from the distribution. They directly achieved state-of-the-art results on a lot of different tasks from image generation [15], through image super-resolution [17], 3D object generation [18], anomaly detection [5], and video prediction [12].

Despite their success, training GANs is notoriously unstable, and the theoretical knowledge of why GANs work well is still not fully explored [7]. One problem

⋆ The main part of the work was done while Thomas Goerttler and Marius Kloft were with Department of Computer Science, Humboldt University of Berlin, Germany.

is that the distribution of data is usually multimodal and discontinuous, whereas the latent space usually comes from a continuous space e.g., uniform or Gaussian with no mode respectively only one mode. Therefore, the generator function G has to learn a transform from the continuous latent space to the discontinuous multimodal distribution, which can be seen as a mixture of different simple distributions. For example, a human either wears glasses or does not. This transition is discrete and has to be learned by the generator. However, this is quite difficult, and generators tend to learn ambiguous faces.

Additionally, this makes it difficult for the GAN to train and endangers mode collapse when the model only captures a single mode and misses other ones. Gurumurthy et al. [10] propose to define a multimodal prior distribution directly; however, this only works if we know already the real data distribution, which is not the case in practice. If we knew the distribution already, a GAN would not be required anymore.

Therefore, we propose to learn the modes directly in the latent distribution. We achieve this by restricting the prior distribution in the training procedure. Besides, this helps the training procedure to be more stable and finally, helps the GAN not to miss modes, as our results show.

2 Training generative adversarial nets

The idea of GANs, introduced in [8], is to have two adversarial neural nets which play a two-player minimax game. On the one hand, there is a generator function G which learns the distribution p_g over given data $\boldsymbol{x}$, which draws noise $\boldsymbol{z}$ randomly from a prior distribution p_z and generates out of it an implicit distribution p_g. On the other hand, there is a discriminator function D, which tries to distinguish accurately between real data $\boldsymbol{x}$ and the generated data $G(\boldsymbol{z}, \theta_g)$. It returns a single scalar which expresses the probability that a given input comes rather from the data $\boldsymbol{x}$ than from the generator. Both G and D are non-linear mapping differentiable functions and in general, expressed by a neural net. In the training process of GANs, the discriminator D is trained to correctly discriminate between the real data input $\boldsymbol{x}$ and the generated samples $G(\boldsymbol{z}, \theta_g)$. At the same time, the generator G is trained to fool the discriminator as much as possible. Thus, it wants to maximize the function $D(G(\boldsymbol{z}))$, whereas the discriminator wants to minimize it and maximize $D(\boldsymbol{x})$. In [8], the objective function is expressed as followed:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

The objective function is trained via gradient descent step until convergence, which is the case when we have reached a Nash equilibrium. In [15], the authors extended convolution and pooling layers into the architecture. Further extension of the GAN framework are e.g., the Conditional GAN[14] or the unrolled GAN[13].

Degenerative prior distribution and manifold problem: A huge problem of GANs is that samples from the generator are degenerative when instantiating the GANs. In [2] it is remarked that if the dimension k of the prior distribution p_z is smaller than the dimension n of the data distribution, the output of the generator will always lie in the k-dimensional manifold in the n-dimensional space. Also, the distribution of the real data p_{data} lies often in a o-dimensional manifold with $o << n$. Having two distributions which lie on lower-dimensional manifold, results in the situation that the support of the real data distribution p_{data} and the generated distribution p_g often are non-overlapping. In such cases minimizing divergences is meaningless as they are "maxed out" [2]. Furthermore, the discriminator can be perfect, which leads to instabilities and also to a vanishing gradient problem in the generator [2] [17].

As minimizing divergence is meaningless if the p_g and p_{data} are disjoint, the Wasserstein GAN (WGAN) is aiming at minimizing the Wasserstein distance instead of any type of f-divergence [3]. A theoretical solution for ensuring that the degenerative distributions of the generator and the real data lying on a low-dimensional manifold overlap, is to add noise to the generated and the real data [2] [17].

Mode collapse and mode dropping One common failure of GANs happens when the generator collapses to a set of parameters, and the GAN always outputs the same value. This output fools the discriminator so well that the discriminator cannot discriminate the fake samples from the real data.

Similar to mode collapse is mode dropping. As there is no communication between the points in GANs, it can happen that the loss function is close to the optimum the score of the fake samples $G(\boldsymbol{z})$ are all almost 0.5 which indicates that the algorithm has almost reached the Nash equilibrium, but some modes are not captured and missed out.

Our approach, we introduce in this paper, focuses on manipulating the prior distribution. In [1] also the prior is manipulated, but they use an associative memory in the learning process.

3 Masked and weighted prior distribution for GANs

In this section, we introduce our novel approaches to stabilize the training of GANs by finding modes in the prior distribution. We achieve this by masking and weighting the latent distribution of the GAN during training.

3.1 Using the information of the discriminator

The standard GAN samples a batch $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$[3] of size m from the prior distribution $p_z(\boldsymbol{z})$ and passes them to the generator. The prior distribution has

[3] Note, that we denote the batches as sets although there are not mathematical sets but arrays or vectors of samples. We adapted the set notation from [8]

a dimension of k, and its distribution is defined once and is constant over time. In practice, the uniform distribution $\mathcal{U}_k(-1,1)$ or the standard normal distribution $\mathcal{N}_k(0,1)$ are used. Training a GAN, there are two steps that are iteratively repeated: in the first step the discriminator is updated, and in the second step the generator. In both steps, we sample a batch $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ from the prior distribution $p_z(\boldsymbol{z})$ identically and independently. We propose to use the information the discriminator gives us about every faked sample when we pass the noise through the generator $G(\boldsymbol{z}^{(i)})$. The information we obtain for a noise sample $\boldsymbol{z}^{(i)}$ is a score:

$$s^{(i)} := s(\boldsymbol{z}^{(i)}) := D(G(\boldsymbol{z}^{(i)})) \tag{2}$$

The score $s^{(i)}$ lies in the case of standard GAN in $[0,1]$ and gives us information about how likely it is that the generated samples fool the current parameterization θ_g of the discriminator D. Having this information, we restrict and manipulate the prior distribution before we resample again from the manipulated prior distribution and optimize the generator and the discriminator with a batch of resampled noise values $\{\boldsymbol{z}_r^{(1)},\dots,\boldsymbol{z}_r^{(m)}\}$. We distinguish between two different approaches:

Masking the prior by restricting it to the portion which has a higher probability of fooling the generator. This gives a hard constraint, and only the part of $p_z(\boldsymbol{z})$ is processed, which falls into this region. The portion of the part being restricted is a hyperparameter r which lies in $(0,1]$. Because the rate r determines the portion of the prior distribution we select from, a rate of 1 means that we draw from the normal GAN prior constantly and a rate r close zero means we only optimize for a tiny region. In Definition 1, we denote the density function of the masked prior.

Definition 1. *The probability density function of the masked prior is defined as*

$$p_{z,masked}(z) = \begin{cases} \frac{1}{r}\cdot p_z(z) & \text{if } P(s(p_z(z)) > s(x)) > 1-r \text{ for } x \sim p_z \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Weighting the prior by using the score to define a weight. In this case, we resample from the prior distribution weighted by the scores, respectively, a function of the scores. The new density is given in Definition 2.

Definition 2. *The probability density function of the weighted prior is defined as*

$$p_{z,weighted}(z) = p_z(z)\cdot w(s(z)) \tag{4}$$

and w is chosen such that it holds

$$\forall s_0, s_1 \in [0,1] : s_0 < s_1 \Rightarrow w(s_0) < w(s_1) \tag{5}$$

and

$$\int_z p_z(z)\cdot w(s(z))dz = 1 \tag{6}$$

Equation 5 ensures that a higher score leads to a higher probability to be drawn and equation 6 guarantees that $p_{\text{z,weighted}}$ is a density as the weights are normalized. We propose to define $w(s)$ such that it additionally holds

$$\forall s_0, s_1 \in (0, 1] : \frac{s_0}{s_1} \Rightarrow \frac{w(s_0)}{w(s_1)} \tag{7}$$

Equation 7 leads to the proportionality between s and $w(s)$.

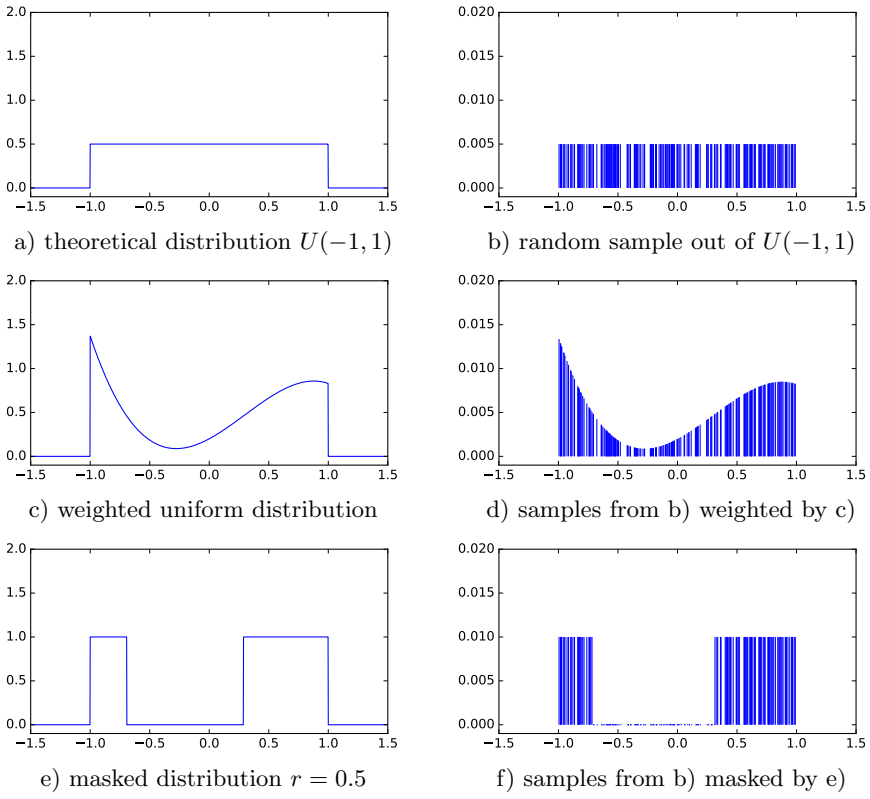


Fig. 1. In a), c) and e) examples of theoretical distributions of constant, weighted, and masked 0.5 can be seen. In the right column, there are the empirical counterparts. Note, that d) and f) are based on restricting b) and are not samples from the restricted density functions c) and e).

In the following, we always call the GAN with a fixed prior distribution traditional GAN or GAN with a constant prior distribution. In Figure 1, we see how the theoretical prior distribution changes by looking at an example in the one-dimensional case. While training a traditional GAN, we draw from a uniform distribution that has a minimum value of -1 and a maximum value of 1. In this

example the scores are determined by the function $f(z) = (z-0.3)^3 + z + 0.0173$ for $x \in [-1, 1]$ (Figure 1 c)). The theoretical distribution of weighting changes due to the priors, if the weighting function is $w(z) = f(z)$. The masked prior of $r = 0.5$ can be seen under e). The lower the masking score is, the higher their density value is because the range we draw from decreases.

As $p_z(\boldsymbol{z})$ is continuous, it is impossible to get the score of each point a priori because we have uncountable infinity points. Therefore, we have to find another way to draw appropriately from the theoretical distributions $p_{z,masked}$ and $p_{z,weighted}$. We resample from the batch Z retrieving after masking or weighting it. In the case of masking, this means that we keep the samples with the higher score and resample from them $Z^* \subseteq Z$ as showcased in Figure 1 f). In case of weighting, this means that we assign a weight to every sample and resample again from the batch Z weighted with the batch of weights $W = \{w(s^{(1)}), \dots, w(s^{(n)})\}$(Figure 1 d)). If we have a minibatch size of m, it follows that the pre-sample size n is higher than m to get enough diversity for each training step. This is required because the masked region we resample from only has a size of $r \cdot n$, and we want that value not to be much smaller than m and ideally higher. The distribution - we draw our masked and weighted samples from in the algorithm - is not continuous anymore but is based on the pre-sample batch. Therefore, we do not use densities for masked and weighted prior in the algorithm but probability mass functions. Thus, we slightly adjust Definition 1, leading to Definition 3.

Definition 3. *The probability mass function of the masked prior is defined as*

$$pm_{z,masked}(z) = \begin{cases} \frac{1}{r \cdot n} & \text{if } z \in \{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(n)}\} \text{ and } s(x) > pct_{1-r}(\{s^{(1)}, \dots, s^{(n)}\}) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where pct_{1-r} is the $1-r$ percentile. Note, that we assume that the elements in $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(n)}\}$ are disjunct which is true with a probability of 1 as they are drawn from a continuous distribution. The finite sample version of the weighted prior is defined in Definition 4.

Definition 4. *The probability mass function of the weighted prior is defined as*

$$pm_{z,weighted}(z) = \begin{cases} p(z) \cdot w(s(z)) & \text{if } z \in Z = \{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(n)}\} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and w is chosen such that it holds

$$\forall s_0, s_1 \in [0, 1] : s_0 < s_1 \Rightarrow w(s_0) < w(s_1) \tag{10}$$

and

$$\sum_{i=1}^{n} w(s^{(i)}) = 1 \tag{11}$$

Layer	Layer type	Hyperparameters
input	**z**	500 components sampled from $\mathcal{U}_2(-1, 1)$
1	Fully Connected	128 Neurons, ReLU
2	Fully Connected	128 Neurons, ReLU
output	Fully Connected	2 (3) Neurons, tanh

Layer	Layer type	Hyperparameters
input	**x**	(2, 1) or (3, 1)
1	Fully Connected	128 Neurons, tanh
2	Fully Connected	128 Neurons, tanh
3	Fully Connected	128 Neurons, tanh
output	Fully Connected	1 Neuron, Sigmoid

Table 1. This table shows the parameter setting we use for the eight modes

4 Experiments

In this section, we discuss the results of applying our novel learning algorithm to GANs on different data sets. The data sets are both a synthetic toy data set, and standard deep learning data sets MNIST and CelebA.We train the GAN using a multi-layer perceptron as well as the DCGAN. For the DCGAN we use the implementation of Taehoon Kim[4].

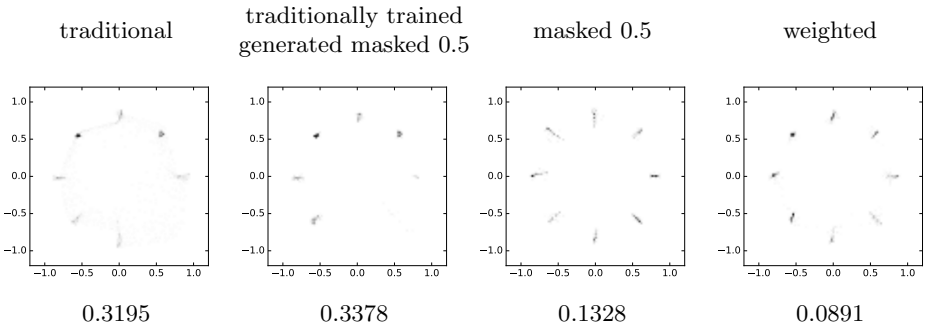


Fig. 2. This figure shows the heatmap of the generated distribution and the EMDs of GAN experiments on eight modes.

To spectate the effect of masking and weighting the prior distribution, we apply our GAN extension on a mixture of eight Gaussians lying in $\mathbb{R}^2$. The eight Gaussian mode data set has been used to show stabilizing effects in [4], [16], and [13].

[4] https://github.com/carpedm20/DCGAN-tensorflow

Besides applying our modified GAN on a mixture of eight Gaussians, we also apply them to MNIST [6] and CelebA [11], two datasets which are commonly used in deep learning and image processing tasks.

4.1 Mixture of Gaussians

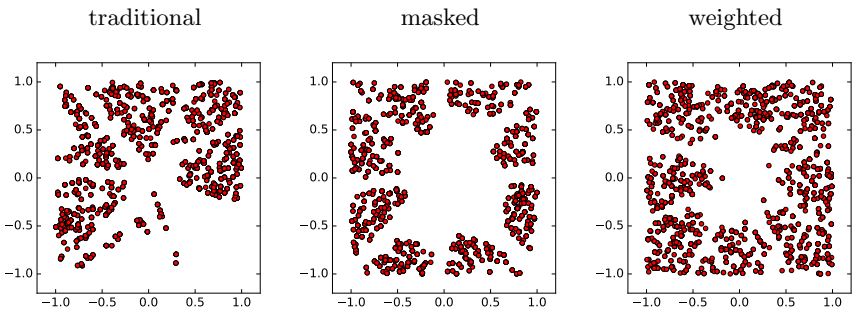
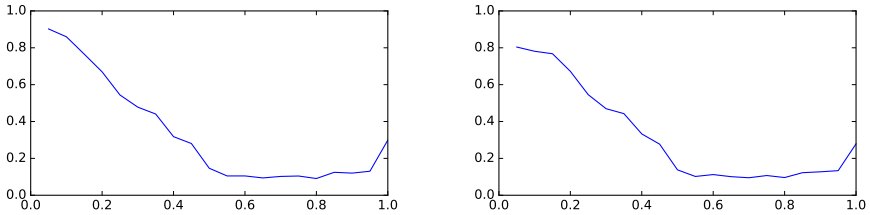


Fig. 3. This figure shows the prior distribution of the GANs after 50 epochs for eight modes. We mask the prior for the traditional GAN also although we do not use this information during training.

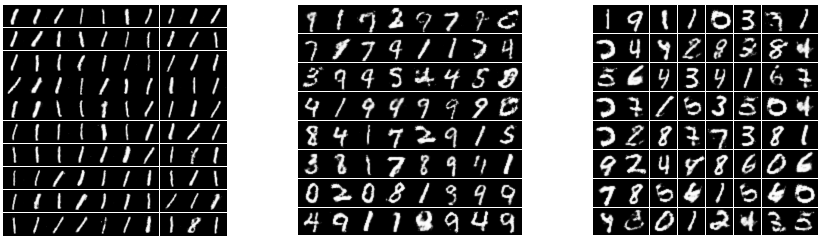
We compare the performance of the different GANs on an example of eight modes. The parameters of the networks are summarized in Table 1. We use Wasserstein-GAN [3] with five discriminator steps per one training step of the generator, a minibatch size of 500, a learning rate of 0.0001, and we train the GAN for 50 epochs. The masking rate is 0.5, and for masking and weighting, we have a pre-sample size of 1000. In Figure 2, we see the result of the GANs. The traditional GAN does not capture all modes. It can be observed that especially two modes have gotten a lot of mass. Also, some outliers between each mode are visible. These are visible anymore if we sampled masked during generation. Having a look at the prior distribution in Figure 3, we see that different areas get a higher score, which leads to the modes. But as we do not use this information during training discriminator and generator. Looking at the results of the masked and the weighted GAN we see a huge improvement. The prior distribution is separated into eight modes which correspond to the eight modes in the resulting distribution. The EMDs of the GAN with a traditional prior distribution is 0.3195. Masking the prior distribution of a traditional GAN only for the generation, the EMD score is even a little bit higher (0.3378). Although the outliers disappear, masking only the prior distribution during the generation does not help in this case. The EMD of the weighted and the masked GAN though are smaller: 0.0891 respectively 0.1328. We repeated this also with the standard GAN with the alternative loss function. The results were similar.

We also want to investigate the influence of the masking rate more detailed. So far, we only used the rate of 0.5, but in general, every value $r \in (0, 1]$ is



a) EMD of WGAN b) EMD of standard GAN

Fig. 4. This figure shows the earth mover's distance depends on the hyperparameter masked rate r for WGAN a) and the standard GAN b).

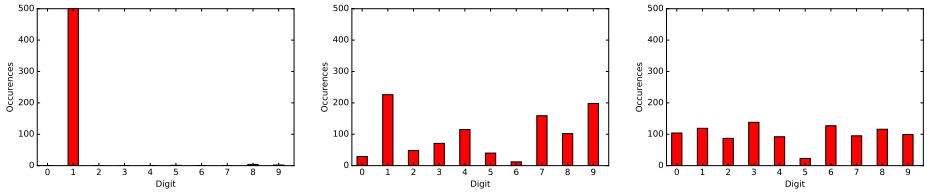


a) traditional b) weighted c) masked 0.05

Fig. 5. This figure shows the plots of GANs we apply on MNIST using only fully connected layers.

possible. We repeated the experiment for several rates from 0 to 1. We used the same parameter setting but increased the pre-sample size to 2000 that we have more points to resample from which especially helps the low masking rates as they mask out most of the points. In Figure 4, we see on the x-axis the rate and on the y-axis the resulting EMD. We plot the average EMD of three different runs of the experiments. We observe that a low masking rate does not work in this case as it restricts too much of the prior distribution. A masking rate of 0.5 to 0.9 improves the GAN in quality as it has a smaller EMD (Figure 4).

MNIST On MNIST we train a GAN using only multilayer perceptron (MLP) networks as well the DCGAN architecture. The hyperparameters of the MLP version are based on the parameters in [8] but we use a slower learning rate of 0.01. The minibatch size is 100 and SGD is used with a momentum of 0.5 at the start, which increases to 0.7 at the epoch of 250. The model is trained 300 epochs and several times with different starting states of the seed. In Figure 5, we see the resulting images of the traditional GAN and the output of the two masked GAN and the weighted GAN. Whereas the quality of the resulting pictures is similar, respectively, one cannot see a clear difference, we see that the traditional GAN lacks to capture different modes and only captures the digit 1. Masking the GAN



a) traditional b) weighted c) masked 0.05

Fig. 6. Barchart of the different distributions of MNIST

and weighting the GAN solves this problem and leads to more stable results. In Figure 6, we observe the bar charts of the resulting distribution drawing 2000 samples from the generator and classifying them. They underline that the modes are captured more fairly for our approach. We also use the DCGAN architecture to learn the MNIST. The architecture of the generator and the discriminator nets are adapted from [15]. For the convolutional and transposed convolutional layers, we use a filter width and filter height of 5 and a stride of 1 respectively 2. The results can be seen in Figure 7. Also, in this setting, the fairness of modes is better when applying masking and weighting.

a) traditional b) masked 0.5 c) weighted

Fig. 7. This figure shows the results we have on MNIST using DCGAN.

CelebA We also apply our new proposed GAN on the CelebA data set. We use the DCGAN architecture as well as the parameters of [15]. We train the model for 10 epochs. Besides, we reduce the depth of the convolution layers for a second experiment. This time we allow it to train for 25 epochs as we want to guarantee that it has time to converge. Reducing the depth of convolution has also been done in [9] to show stabilization effects.

In Figure 8, the results of the three different GANs are shown. We observe that for the first parameter setting traditional GAN, masked GAN, and weighted

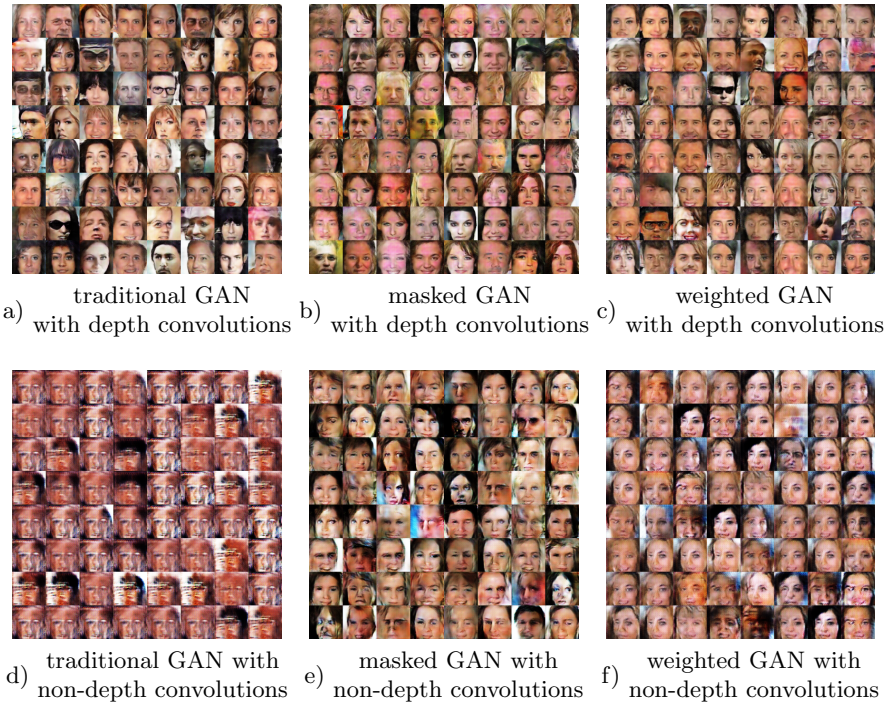


Fig. 8. The upper row figures a), b) and c) show the results of a normal trained DCGAN. Figures d),e) and g) are results where the depth of the convolutional layers is reduced.

GAN produce results of similar quality. If we reduce the depth of the convolutional layers in the generator and the discriminator, the traditional GAN captures only a few modes and is not able to replicate the faces properly. However, also the masked and the weighted GAN images become worse as we see a lot of similar faces and not the full variety like in the training set. Also, the quality is reduced, which is caused by the weaker architecture. Nevertheless, the quality of the results of masked and weighted GAN is better than the traditional GAN.

5 Conclusion and future work

In this paper, we propose a new extension of GANs, which focuses on reducing the prior distribution to particular regions instead of leaving it constant. Our experiments show the potential of the novel idea as it decreases the EMD between the training data and the generated data. In the case of estimating a multimodal distribution, we noticed that masked GAN finds the corresponding island in the latent space. On MNIST, we could observe that the generated distribution is fairer when applying masking and weighting.

In the future, we want to tackle the uncomfortableness that we, in our essential extension, have to use the discriminator to generate new samples. We think that it is worthwhile to eliminate this. We have two different ideas in mind: diminishing the masking and weighting effect by either increase the masking rate or blurring out the weights, and optimizing for the prior distribution, which shows higher gradients.

Acknowledgments We thank Robert Vandermeulen, Lukas Ruff and Grégoire Montavon for fruitful discussions.

References

1. Arici, T., Çelikyilmaz, A.: Associative adversarial networks. CoRR (2016)
2. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. In: ICLR (2017)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: ICML. pp. 214–223 (2017)
4. Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W.: Mode regularized generative adversarial networks. In: ICLR (2017)
5. Deecke, L., Vandermeulen, R.A., Ruff, L., Mandt, S., Kloft, M.: Image anomaly detection with generative adversarial networks. In: ECML PKDD. pp. 3–17 (2018)
6. Deng, L.: The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process. Mag. **29**(6), 141–142 (2012)
7. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive computation and machine learning, MIT Press (2016)
8. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS. pp. 2672–2680 (2014)
9. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: NIPS. pp. 5769–5779 (2017)
10. Gurumurthy, S., Sarvadevabhatla, R.K., Babu, R.V.: Deligan: Generative adversarial networks for diverse and limited data. In: CVPR. pp. 4941–4949 (2017)
11. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: ICCV. pp. 3730–3738 (2015)
12. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016)
13. Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.: Unrolled generative adversarial networks. In: ICLR (2017)
14. Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR (2014)
15. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR (2016)
16. Roth, K., Lucchi, A., Nowozin, S., Hofmann, T.: Stabilizing training of generative adversarial networks through regularization. In: NIPS. pp. 2015–2025 (2017)
17. Sønderby, C.K., Caballero, J., Theis, L., Shi, W., Huszár, F.: Amortised MAP inference for image super-resolution. In: ICLR (2017)
18. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: NIPS. pp. 82–90 (2016)