Supplementary materials for Implicit competitive regularization in GANs

Florian Schäfer [*1] **Hongkai Zheng** [*2 3] **Anima Anandkumar** [1]

1. Experimental details

1.1. Euclidean distance on images

In Figure 1 we provide a larger reproduction of Figure 2 from the main paper. We see that also on the larger resolution, the third pair of images is visually indistinguishable, despite having the largest Euclidean distance of all pairs. The textures of this image are very rough, with a rapid alternation of bright and dark pixels. Therefore, a slight warping of the image will lead to dark pixels taking the place of bright ones and vice versa, leading to a large Euclidean distance. A similar effect could be achieved by the wind slightly moving the foliage between, for instance, successive frames of a video. Thus, this phenomenon could be observed in real images.

1.2. ICR as projection

For the experiments in Figure 5 of the main paper, we used two tiny neural networks with 28 parameters and three layers each as generator and discriminator.

The generator $\mathcal{G}$ is composed as follows:

1. Use first four parameters as input, apply arctan nonlinearity.

2. Apply four times four dense layer, followed by arctan.

3. Apply two times four dense layer, followed by the nonlinearity

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} \exp\left(\arctan\left(y\right)/\pi + x\right) \\ \exp\left(\arctan\left(y\right)/\pi - x\right) \end{pmatrix}.$$

The form of the last nonlinearity ensures that the output is restricted to the set

$$\mathcal{S} \coloneqq \left\{ (e^{s+t}, e^{s-t}) \middle| s \in \left[-\frac{1}{2}, \frac{1}{2}\right], t \in \mathbb{R} \right\} \subset \mathbb{R}^2$$

that does not include the target $P_{\text{data}} \coloneqq (2, 2)$. Note that in this simple example the generator does not take any input, but directly maps the weights $w_{\mathcal{G}} \in \mathbb{R}^{28}$ to a pair of real numbers.

The discriminator $\mathcal{D}_\eta$ is composed as follows:

1. Rescale input by the diagonal matrix η, apply four times two dense layer, followed by arctan.

2. Apply four times four dense layer, followed by arctan.

3. Apply one times four dense layer, followed by arctan.

While we did not observe the metastable projection behavior on all runs, we observed it in 13 out of 20 independent runs when using SimGD. When using CGD we observed the projection behavior in 17 out of 20 independent runs (with the same initialization as in the SimGD cases). Furthermore, the number of iterations spent in the projection states was larger when using CGD.

1.3. ICR on MNIST

In our experiments on MNIST, we use the network architectures detailed in Table 1 and Table 2. We train using stochastic SimGD with a learning rate of 0.01. First, we train the GAN for 9,000 iterations with a batch size of 128. We refer to the resulting generator and discriminator as the checkpoint generator and discriminator.

1.3.1. Details about Figure 4 of the main paper

We then create a test set that has a real set X_{real} that has 500 images sampled from MNIST training set, and a fake set X_{fake} that has 500 images generated by the checkpoint generator, as illustrated in Figure 2.

Let D_t, D_c denote, respectively, the discriminator at time step t and the checkpoint discriminator. The Euclidean distance between predictions of D_t and D_c over set X_{real} and X_{fake} in Figure 4 of the main paper is given by

$$\boldsymbol{d}_{\text{set}}(D_t, D_c) = \sqrt{\sum_{x \in X_{\text{set}}} (D_t(x) - D_c(x))^2} \qquad (1)$$

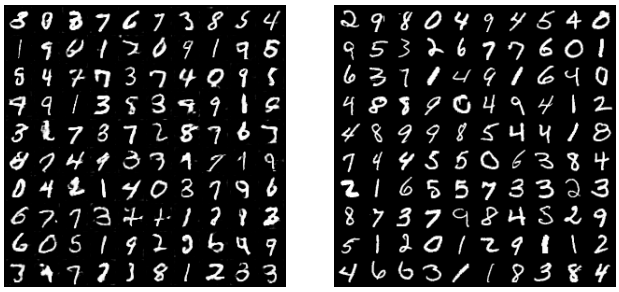
where $\text{set} \in \{\text{real}, \text{fake}\}$.

1.4. CIFAR10 experiments

1.4.1. Architecture

For our experiments on CIFAR10, we use the same DCGAN network architecture as in Wasserstein GAN with gradient penalty (Gulrajani et al., 2017), which is reported in Table 5 and Table 4.

Figure 1. A larger reproduction of Figure 2 of the main paper. The first pair is based on an image by Matt Artz, the second pair on an image by Yanny Mishchuk, and the third pair on an image by Tim Mossholder. All images were obtained from https://unsplash.com/.



(a) A part of the fake set (b) A part of the real set

Figure 2. Test set for Figure 4 of the main paper.

Module	Kernel	Stride	Output shape
Gaussian distribution	N/A	N/A	96
Linear, BN, ReLU	N/A	N/A	1024
Linear, BN, ReLU	N/A	N/A	$128 \times 7 \times 7$
ConvT2d, BN, ReLU	4×4	2	$64 \times 14 \times 14$
ConvT2d, Tanh	4×4	2	$1 \times 28 \times 28$

Table 1. Generator architecture for MNIST experiments

Module	Kernel	Stride	Output shape
Input	N/A	N/A	$1 \times 28 \times 28$
Conv2d, LeakyReLU	5×5	1	$32 \times 24 \times 24$
MaxPool	2×2	N/A	$32 \times 12 \times 12$
Conv2d, LeakyReLu	5×5	1	$64 \times 8 \times 8$
MaxPool	2×2	N/A	$64 \times 4 \times 4$
Linear, LeakyReLU	N/A	N/A	1024
Linear	N/A	N/A	1

Table 2. Discriminator architecture for MNIST experiments

1.4.2. HYPERPARAMETERS

We compare the stability and performance of Adam and ACGD by varying the loss functions and regularization methods.

Loss:

1. Original GAN loss (Goodfellow et al., 2014)

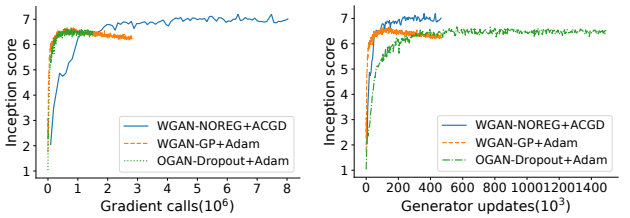
$$\mathcal{L}_o = \mathbb{E}_{x\sim P_{\text{data}}}\left[\log \mathcal{D}(x)\right] + \mathbb{E}_{x\sim P_{\mathcal{G}}}\left[\log\left(1-\mathcal{D}(x)\right)\right].$$

2. Wasserstein GAN loss (Arjovsky et al., 2017)

$$\mathcal{L}_w = \mathbb{E}_{x\sim P_{\text{data}}}\left[\mathcal{D}(x)\right] - \mathbb{E}_{x\sim P_{\mathcal{G}}}\left[\mathcal{D}(x)\right].$$

Regularization:

1. L_2 weight penalty on the discriminator parameters $\lambda \in \{10^{-2}, 10^{-3}, 10^{-4}\}$.



(a) Against gradient call (b) Against the generator iterations

Figure 3. Tensorflow inception scores for important runs

2. Gradient penalty on the discriminator proposed by WGAN-GP paper (Gulrajani et al., 2017).

3. Spectral normalization on the discriminator proposed by SNGAN paper (Miyato et al., 2018).

Each experiment is trained with a batch size of 64. When using Adam and the original GAN loss, we adopt the log-trick as recommended in GAN paper (Goodfellow et al., 2014). When using ACGD, the generator and discriminator share the same loss function. For the training of WGAN-GP, we use the same training strategy and hyperparameters as WGAN-GP[1] (Gulrajani et al., 2017). Hyperparameter setting for each experiment is reported in Table 3.

1.4.3. TENSORFLOW INCEPTION SCORE

We compute the Tensorflow version of the inception scores for important runs of our experiments to show that the relative performance of the different models is largely the same. As reported in Figure 3, our results matches the ones reported in the literature (Figure 3 in (Gulrajani et al., 2017)) with ACGD still outperforming WGAN-GP trained with Adam by around 10%.

1.5. Details on ACGD

In order to make a fair comparison with Adam, we run our experiments with ACGD, a variant of CGD that adaptively adjusts CGD's step size. The algorithm is described in Algorithm 1. ACGD computes individual step sizes for the different parameters. Let $A_{x,t}$ and $A_{y,t}$ denote the diagonal matrices containing the step sizes of x and y at time step t as elements. If $A_{x,t}$ and $A_{y,t}$ are multiples of the identity, the algorithm reduces to CGD with the corresponding step size. The reason we rearrange the terms as shown in Algorithm 1 is that we want the matrix inverse to contain an additive identity (to decrease the condition number) and be symmetric positive definite (so that we can use conjugate gradient (Eisenstat, 1981) for its computation). ACGD adjusts CGD's step size adaptively during training with second moment estimate of the gradients. The update rules are derived from the local game in the same way as for CGD (Schaefer & Anandkumar, 2019):

$$\min_x x^\top \nabla_x f(x_t, y_t) + x^\top [D_{xy} f(x_t, y_t))] y + \frac{1}{2} x^T A_{x,k}^{-1} x,$$
$$\max_y y^\top \nabla_y f(x_t, y_t) + y^\top [D_{yx} f(x_t, y_t))] x - \frac{1}{2} y^T A_{y,k}^{-1} y.$$

Algorithm 1 ACGD, a variant of CGD with RMSProp-type heuristic to adjust learning rates. All operations on vectors are element wise. $D^2_{xy} f$, $D^2_{yx} f$ denote the mixed Hessian matrix $\frac{\partial^2 f}{\partial x \partial y}$ and $\frac{\partial^2 f}{\partial y \partial x}$. β_2^t denotes β_2 to the power t. $\phi(\eta)$ denotes a diagonal matrix with η on the diagonal. Hyperparameter settings for the tested GANs training problems are $\alpha = 10^{-4}, \beta_2 = 0.99$, and $\epsilon = 10^{-5}$

Require: α: Step size
Require: β_2: Exponential decay rates for the second moment estimates
Require: $\max_y \min_x f(x, y)$: zero-sum game objective function with parameters x, y
Require: x_0, y_0 Initial parameter vectors
  $t \leftarrow 0$ Initialize timestep
  $v_{x,0}, v_{y,0} \leftarrow 0$ (Initialize the 2^{nd} moment estimate)
  repeat
    $t \leftarrow t + 1$
    $v_{x,t} \leftarrow \beta_2 \cdot v_{x,t-1} + (1 - \beta_2) \cdot g_{x,t}^2$
    $v_{y,t} \leftarrow \beta_2 \cdot v_{y,t-1} + (1 - \beta_2) \cdot g_{y,t}^2$
    $v_{x,t} \leftarrow v_{x,t}/(1 - \beta_2^t)$
    $v_{y,t} \leftarrow v_{y,t}/(1 - \beta_2^t)$ (Initialization bias correction)
    $\eta_{x,t} \leftarrow \alpha/(\sqrt{v_{x,t}} + \epsilon)$
    $\eta_{y,t} \leftarrow \alpha/(\sqrt{v_{y,t}} + \epsilon)$
    $A_{x,t} = \phi(\eta_{x,t})$
    $A_{y,t} = \phi(\eta_{y,t})$
    $\Delta x_t \leftarrow -A_{x,t}^{\frac{1}{2}}(I + A_{x,t}^{\frac{1}{2}} D^2_{xy} f A_{y,t} D^2_{yx} f A_{x,t}^{\frac{1}{2}})^{-1} A_{x,t}^{\frac{1}{2}}$
    $(\nabla_x f + D^2_{xy} f A_{y,t} \nabla_y f)$
    $\Delta y_t \leftarrow A_{y,t}^{\frac{1}{2}}(I + A_{y,t}^{\frac{1}{2}} D^2_{yx} f A_{x,t} D^2_{xy} f A_{y,t}^{\frac{1}{2}})^{-1} A_{y,t}^{\frac{1}{2}}$
    $(\nabla_y f - D^2_{yx} f A_{x,t} \nabla_x f)$
    $x_t \leftarrow x_{t-1} + \Delta x_t$
    $y_t \leftarrow y_{t-1} + \Delta y_t$
  until x_t, y_t converged

[1]Check more details in WGAN-GP official repository: https://github.com/igul222/improved_wgan_training/blob/master/gan_cifar.py

Experiment	Loss	Optimizer	Learning rate	Spectral Normalization	L_2 penalty	GP weight	Critic iterations
OGAN-0.0001L2+Adam	L_o	Adam	10^{-4}	N/A	10^{-4}	N/A	1
OGAN-0.0001L2+ACGD	L_o	ACGD	10^{-4}	N/A	10^{-4}	N/A	1
OGAN-NOREG+Adam	L_o	Adam	10^{-4}	N/A	N/A	N/A	1
OGAN-NOREG+ACGD	L_o	ACGD	10^{-4}	N/A	N/A	N/A	1
WGAN-GP+Adam	L_w	Adam	10^{-4}	N/A	N/A	10	5
WGAN-SN+Adam	L_w	Adam	10^{-4}	Yes	N/A	N/A	1
WGAN-NOREG+ACGD	L_w	ACGD	10^{-4}	N/A	N/A	N/A	1
WGAN-GP+ACGD	L_w	ACGD	10^{-4}	N/A	N/A	10	5
WGAN-0.01L2+Adam	L_w	Adam	10^{-4}	N/A	10^{-2}	N/A	1
WGAN-0.001L2+Adam	L_w	Adam	10^{-4}	N/A	10^{-3}	N/A	1
WGAN-0.001L2+ACGD	L_w	ACGD	10^{-4}	N/A	10^{-3}	N/A	1

Table 3. Settings for all the experiments that occurred in Figure 7 of the main paper.

Module	Kernel	Stride	Output shape
Gaussian distribution	N/A	N/A	128
Linear, BN, ReLU	N/A	N/A	$256 \times 4 \times 4$
ConvT2d, BN, ReLU	4×4	2	$128 \times 8 \times 8$
ConvT2d, BN, ReLU	4×4	2	$64 \times 16 \times 16$
ConvT2d, Tanh	4×4	2	$3 \times 32 \times 32$

Table 4. Generator architecture for CIFAR10 experiments

Module	Kernel	Stride	Output shape
Input	N/A	N/A	$3 \times 32 \times 32$
Conv2d, LeakyReLU	4×4	2	$64 \times 16 \times 16$
Conv2d, LeakyReLU	4×4	2	$128 \times 8 \times 8$
Conv2d, LeakyReLu	4×4	2	$256 \times 4 \times 4$
Linear	N/A	N/A	1

Table 5. Discriminator architecture for CIFAR10 experiments

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In International conference on machine learning, pp. 214–223, 2017.

Eisenstat, S. C. Efficient implementation of a class of preconditioned conjugate gradient methods. SIAM Journal on Scientific and Statistical Computing, 2(1):1–4, 1981.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Advances in neural information processing systems, pp. 2672–2680, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In Advances in neural information processing systems, pp. 5767–5777, 2017.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957, 2018.

Schaefer, F. and Anandkumar, A. Competitive gradient descent. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), Advances in Neural Information Processing Systems 32, pp. 7623–7633. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8979-competitive-gradient-descent.pdf.