LOGAN: LATENT OPTIMISATION FOR GENERATIVE ADVERSARIAL NETWORKS

Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, Timothy Lillicrap
DeepMind
London, UK
{yanwu,jeffdonahue,dbalduzzi,simonyan,countzero}@google.com

Authors of BigGAN

Presented by Qing Lyu on 01/29/2020

LOGAN vs BigGAN

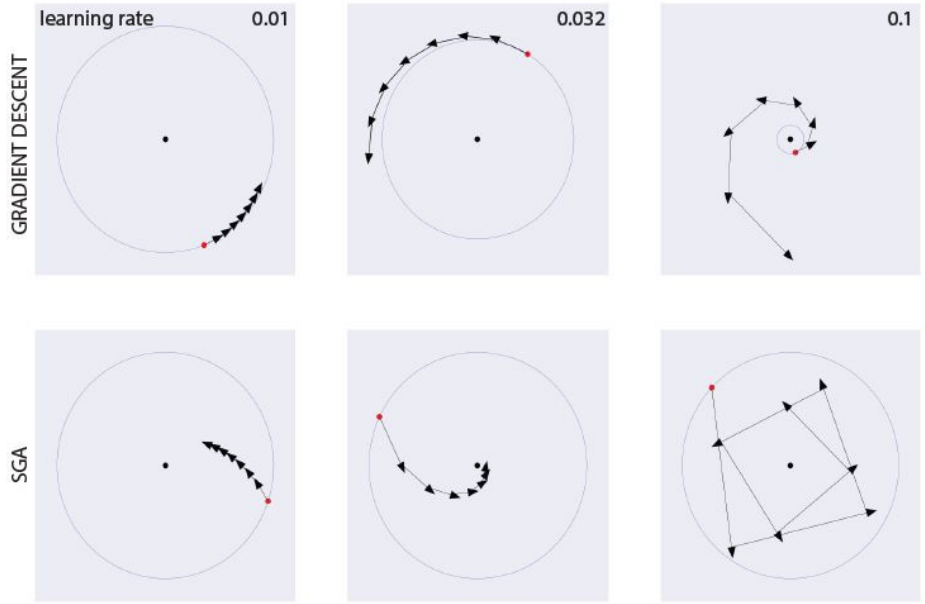
- BigGAN:
  - Large batch: up to 2048
  - More parameters: up to 1.6 billion
  - Truncation trick: truncate prior distribution vector *z*
- LOGAN
  - Focusing on GAN training dynamic
  - Introducing latent vector
  - Strengthen the link between the Generator and Discriminator via updating the latent vector

Contribution

- Authors present a novel analysis of latent optimization in GANs from the perspective of differentiable games and stochastic approximation. They argue that latent optimization can improve the dynamics of adversarial training
- Authors improve latent optimization by taking advantage of efficient second-order updates
- Their algorithm improves the state-of-the-art BigGAN-deep model by a significant margin, without introducing any architectural change or additional parameters

Problems of gradient descent optimization

- the vector-field generated by the losses of the discriminator and generator is not a gradient vector field
  - gradient descent is not guaranteed to find a local optimum and converge to Nash equilibria
  - slow down convergence or lead to phenomena like mode collapses



Dynamics of GAN training

- Training GANs: a minimax game
  - requires carefully balancing updates to *D* and *G*

$$\min_{\theta_D} \max_{\theta_G} \mathbb{E}_{x\sim p(x)} \left[h_D(D\left(x;\theta_D\right))\right] + \mathbb{E}_{z\sim p(z)} \left[h_G(D\left(G\left(z;\theta_G\right);\theta_D\right))\right]$$

$$f(z;\theta_D,\theta_G) = D\left(G\left(z;\theta_G\right);\theta_D\right)$$

$$L_D(z) = h_D(f(z)) \qquad L_G(z) = h_G(f(z))$$

Total loss vector $L(z) = [L_D(z), L_G(z)]^T = [f(z), -f(z)]^T$

Computing the gradients $g = \left[\frac{\partial L_D(z)}{\partial \theta_D}, \frac{\partial L_G(z)}{\partial \theta_G}\right]^T = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T$

Symplectic Gradient Adjustment (SGA)

replace $$g = \left[\frac{\partial L_D(z)}{\partial \theta_D}, \frac{\partial L_G(z)}{\partial \theta_G}\right]^T = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T$$

with $g^* = g + \lambda A^T g$ where λ is a positive constant $A = \frac{1}{2}(H - H^T)$

$$g^* = \left[\frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \partial \theta_D}\right)^T \frac{\partial f(z)}{\partial \theta_G}, \quad -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \partial \theta_G}\right)^T \frac{\partial f(z)}{\partial \theta_D}\right]^T$$

SGA is computational expensive because computing the second-order derivatives with respect to all parameters is expensive

Latent Optimized GANs

Replace original total loss vector $L(z) = [L_D(z), L_G(z)]^T = [f(z), -f(z)]^T$

with $[L_D(z'), L_G(z')]^T = [f(z'), -f(z')]$

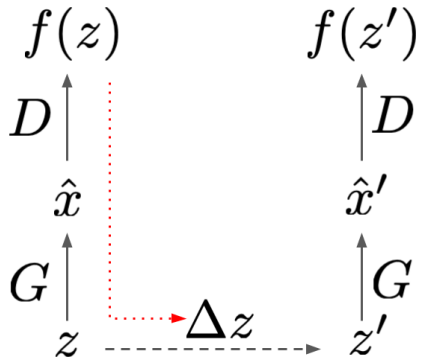
$$\left[\frac{dL_D}{d\theta_D}, \frac{dL_G}{d\theta_G}\right]^T = \left[\frac{\partial f(z')}{\partial \theta_D} + \left(\frac{\partial \Delta z}{\partial \theta_D}\right)^T \frac{\partial f(z')}{\partial \Delta z}, \quad -\frac{\partial f(z')}{\partial \theta_G} - \left(\frac{\partial \Delta z}{\partial \theta_G}\right)^T \frac{\partial f(z')}{\partial \Delta z}\right]^T \tag{7}$$

$$= \left[\frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'}, -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'}\right]^T$$

Comparing with SGA $$g^* = \left[\frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \partial \theta_D}\right)^T \frac{\partial f(z)}{\partial \theta_G}, \quad -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \partial \theta_G}\right)^T \frac{\partial f(z)}{\partial \theta_D}\right]^T$$

Dimension of z is far less than ϑ_G and ϑ_D, indicating higher computational efficiency of LOGAN

Latent Optimized GANs



Algorithm 1 Latent Optimised GANs with Automatic Differentiation

Input: data distribution $p(x)$, latent distribution $p(z)$, $D(\cdot;\theta_D)$, $G(\cdot;\theta_G)$, learning rate α, batch size N

repeat

Initialise discriminator and generator parameters θ_D, θ_G

for $i = 1$ **to** N **do**

Sample $z \sim p(z)$, $x \sim p(x)$

Compute the gradient $\frac{\partial D(G(z))}{\partial z}$ and use it to obtain Δz from eq. 4 (GD) or eq. 12 (NGD)

Optimise the latent $z' \leftarrow [z + \Delta z]$, $[\cdot]$ indicates clipping the value between -1 and 1

Compute generator loss $L_G^{(i)} = -D(G(z'))$

Compute discriminator loss $L_D^{(i)} = D(G(z')) - D(x)$

end for

Compute batch losses $L_G = \frac{1}{N}\sum_{i=1}^{N} L_G^{(i)}$ and $L_D = \frac{1}{N}\sum_{i=1}^{N} L_D^{(i)}$

Update θ_D and θ_G with the gradients $\frac{\partial L_D}{\partial \theta_D}$, $\frac{\partial L_G}{\partial \theta_G}$

until reaches the maximum training steps

LOGAN with natural gradient descent

- NGD is an approximate second-order optimization method
- Using the positive semi-definite Gauss-Newton matrix to approximate the Hessian, NGD often works better than exact second-order methods

LOGAN-GD $\Delta z = \alpha \frac{\partial f(z)}{\partial z}$ $z' = z + \Delta z$

LOGAN-NGD $\Delta z = \alpha F^{-1} g$ $F = \mathbb{E}_{p(t|z)} \left[\nabla \ln p(t|z) \nabla \ln p(t|z)^T \right]$

For simplification, replace F with empirical F' $F' = g \cdot g^T + \beta I$

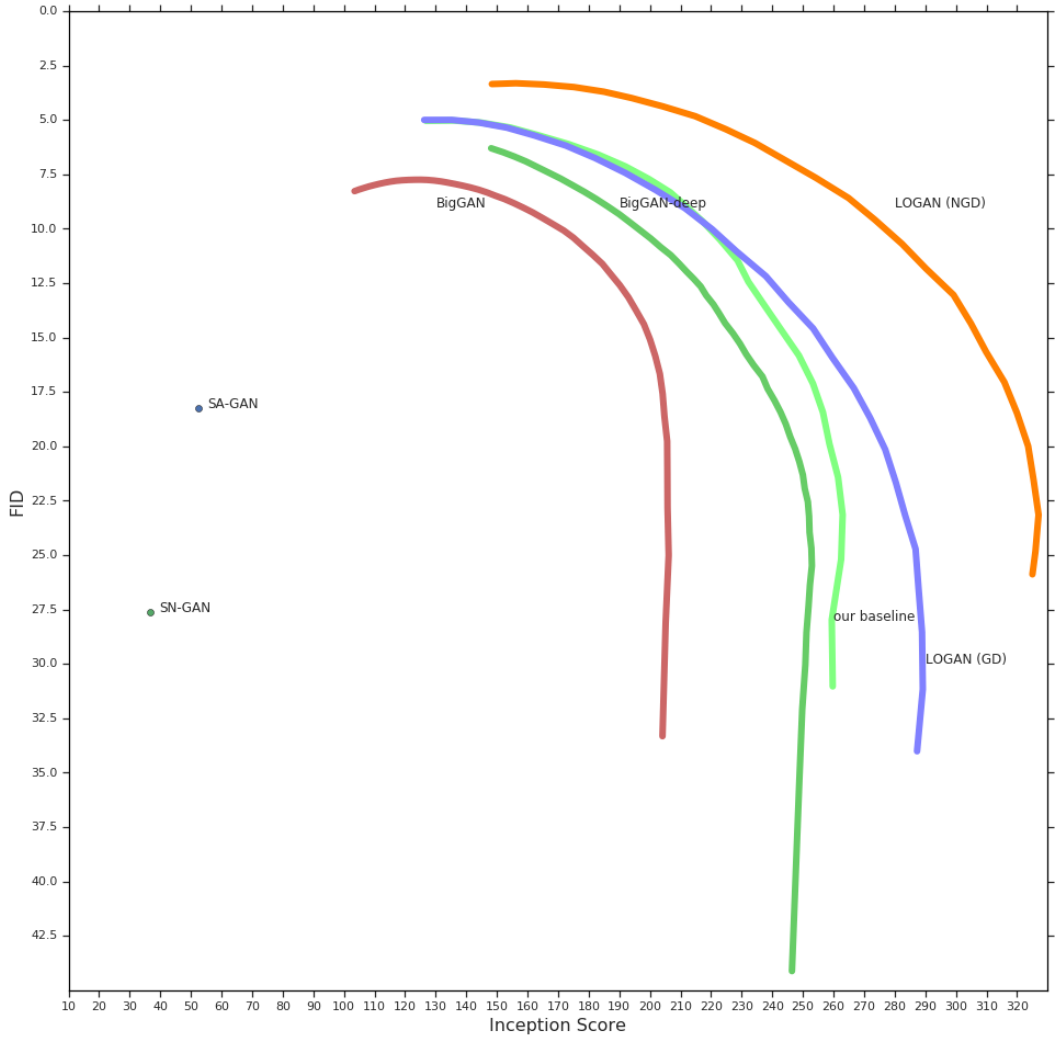
$$\Delta z = \alpha \left(\frac{I}{\beta} - \frac{g\, g^T}{\beta^2 + \beta\, g^T\, g} \right) g = \frac{\alpha}{\beta + \|g\|^2} g$$

Results on ImageNet

Table 1: Comparison of model scores. BigGAN-deep results are reproduced from Brock et al. (2018). "baseline" indicates our reproduced BigGAN-deep with small modifications. The 3rd and 4th columns are from the gradient descent (GD, ablated) and natural gradient descent (NGD) versions of LOGAN respectively. We report the Inception Score (IS, higher is better, Salimans et al. 2016) and Fréchet Inception Distance (FID, lower is better, Heusel et al. 2017).

	BigGAN-Deep	baseline	LOGAN (GD)	LOGAN (NGD)
FID	5.7 ± 0.3	4.92 ± 0.05	4.86 ± 0.09	$\mathbf{3.36 \pm 0.14}$
IS	124.5 ± 2.0	126.6 ± 1.3	127.7 ± 3.5	$\mathbf{148.2 \pm 3.1}$

Results on ImageNet



Results on ImageNet

(a) (b)

Figure 1: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with similarly high IS. Samples from the two panels were drawn from truncation levels corresponding to points C and D in figure 3 **b** respectively. (FID/IS: (**a**) 27.97/259.4, (**b**) 8.19/259.9)

Results on ImageNet

(a) (b)

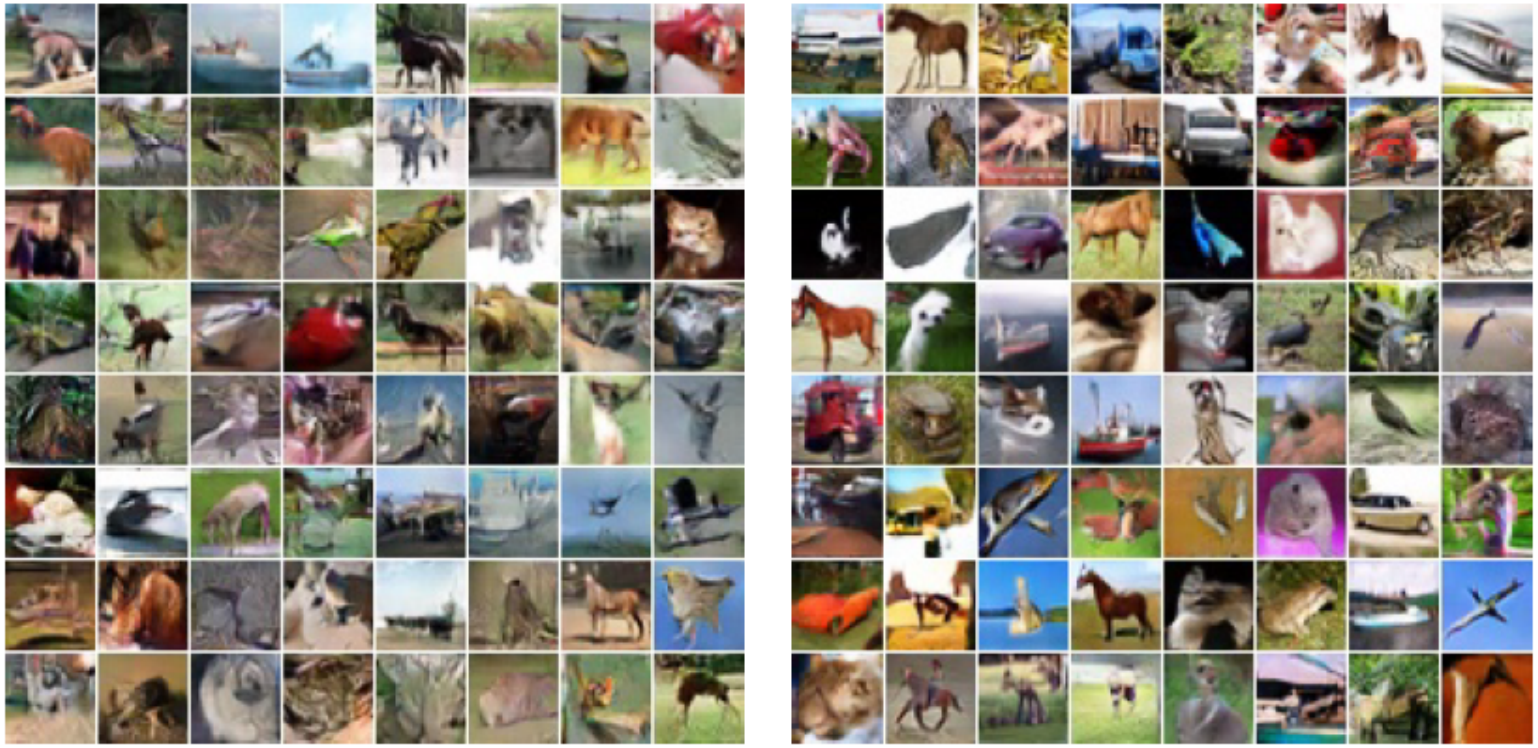
Figure 2: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with similarly low FID. Samples from the two panels were drawn from truncation levels corresponding to points A and B in figure 3 **b** respectively. (FID/IS: (**a**) 5.04/126.8, (**b**) 5.09/217.0)

Results on CIFAR (Appendix)

Table 2: Comparison of Scores. The first and second columns are reproduced from Miyato et al. (2018) and Wu et al. (2019) respectively. We report the Inception Score (IS, higher is better, Salimans et al. 2016) and Fréchet Inception Distance (FID, lower is better, Heusel et al. 2017).

	SN-GAN	CS-GAN	LOGAN (NGD)
FID	29.3	23.1 ± 0.5	$\mathbf{17.7 \pm 0.4}$
IS	7.42 ± 0.08	7.80 ± 0.05	$\mathbf{8.67 \pm 0.05}$

Results on CIFAR (Appendix)



(a) (b)

Figure 9: (**a**) Samples from SN-GAN. (**b**) Samples from LOGAN.