LOGAN: LATENT OPTIMISATION FOR GENERATIVE ADVERSARIAL NETWORKS

Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, Timothy Lillicrap

Brandon Molyneaux

Rafael Bidese

April 6th, 2020

BigGAN-deep

LOGAN

Introduction

- Most recent advances in large-scale image generation come from network architecture improvements, or regularization of particular parts of the model (BigGAN-deep, cGAN, SN-GAN, etc)
- Inspired by previous work on latent optimization CS-GAN (Wu et al. 2019, cited 9 times in this work)
- This work also analyses the latent optimization from the perspective of differentiable games extensively analyzed by one of the authors in Balduzzi et al. 2018 (cited 8 times in this work).
- The paper proposes a novel training scheme for GANs, which leads to improving the state-of-the-art scores.

Background: GANs

- A GAN consists of a generator that generates image from a latent source, and a discriminator that scores the generated images (Goodfellow et al., 2014).
- Training GANs involves an adversarial game: while the discriminator tries to distinguish generated samples $\hat{x} = G(z)$ from data $x \sim p(x)$, the generator tries to fool the discriminator. This is to optimize the min-max game:

$$\min_{\theta_D} \max_{\theta_G} E_{x \sim p(x)}\left[h_D\left(D(x;\theta_D)\right)\right] + E_{z \sim p(z)}\left[h_G\left(D(G(z;\theta_G);\theta_D)\right)\right] \quad (1)$$

- In this work, the authors focus on the second part of the equation since they are looking to explore interactions between D and G.

Background: GANs

- Loss: $L(z) = [L_D(z), L_G(z)]^T = [f(z), -f(z)]^T$ (Wasserstein Loss), but can be generalized to other losses (Arora et al.,2017)

$$g = \left[\frac{\partial L_D(z)}{\partial \theta_D}, \frac{\partial L_G(z)}{\partial \theta_G}\right]^T = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T \quad (3)$$

- When differentiating $L(z)$ w.r.t. θ_G and θ_D we get g which is not a gradient of a function (vector fields Balduzzi et al., 2018), so it can exhibit cycling and alternating, authors keep simultaneous gradient for the analysis

Background: Latent optimized GANs

- Latent optimization exploits knowledge from D to guide updates of z.
- **Intuitively, the gradient $\nabla_z f(z) = \frac{\partial f(z)}{\partial z}$ points in the direction that satisfies the discriminator D, which implies better samples.**
- Therefore, instead of using the randomly sampled $z \sim p(z)$, Wu et al. (2019) uses the optimized latent

$$\Delta z = \alpha \frac{\partial f(z)}{\partial z}, \; z' = z + \Delta z \tag{4}$$

in the equation 1 for training.

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

Latent Optimized GANs with Automatic Differentiation

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

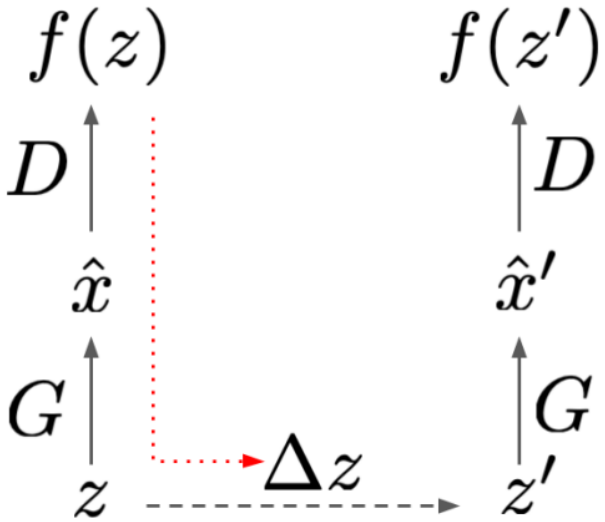
Compute discriminator loss $L_D^{(i)} = D(G(z')) - D(x)$

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



$$\Delta z = \alpha \frac{\partial f(z)}{\partial z} \tag{4}$$

$$z' = z + \Delta z$$

Analysis of the Algorithm (1)

- To understand how latent optimization improves GAN training, we analyze LOGAN as a 3-player differentiable game following Balduzzi et al. (2018); Gemp & Mahadevan (2018);
- This work is strongly inspired on Balduzzi et al. (2018); Gemp & Mahadevan (2018) who proposed Symplectic Gradient Adjustment (SGA) to improve the dynamics of gradient-based methods in adversarial games.
- SGA leverages the interaction between the parameters on the game to update the weights of the players.
- SGA is expensive to scale to BigGAN, because computing all second-order derivatives computationally expensive.
- *SGA uses the adjusted gradient* $g^* = g + \lambda A^T g$,
  - where λ is a positive constant
  - g *is the game gradient*, $g = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T$
  - A is the anti-symmetric component of the Hessian, $A = \frac{1}{2}(H - H^T)$
  - H is the Hessian, $H = \nabla_\theta g$

Analysis of the Algorithm (2)

- Applying SGA to GANs yields the update

$g = \left[\frac{\partial f(z)}{\partial \theta_D}, -\frac{\partial f(z)}{\partial \theta_G}\right]^T$, original GAN's gradient

$$g^* = \left[\frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \partial \theta_D}\right)^T \frac{\partial f(z)}{\partial \theta_G}, \quad -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \partial \theta_G}\right)^T \frac{\partial f(z)}{\partial \theta_D}\right]^T \tag{6}$$

- After simplifications (Appendix B)

$$\left[\frac{dL_D}{d\theta_D}, \frac{dL_G}{d\theta_G}\right]^T = \left[\frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'}, -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'}\right]^T \tag{8}$$

Analysis of the Algorithm (3)

- LOGAN computes these extra terms by **automatic differentiation** when backpropagating through the latent optimization process.
- This **new gradient includes second-order terms**, that didn't existed without the latent optimization.
- The **SGA updates in eq. 6 and the LOGAN updates in eq. 8 are strikingly similar**, suggesting that the latent step used by LOGAN reduces the negative effects of cycling by introducing a symplectic gradient adjustment into the optimization procedure.
- Crucially, latent optimization approximates SGA using only second-order derivatives with respect to the latent z and parameters of the discriminator and generator **separately**.
- In short, latent optimization efficiently **couples the gradients of the discriminator and generator**, as prescribed by SGA, but using the much lower-dimensional latent source z which makes the adjustment scalable

$$g^* = \left[\frac{\partial f(z)}{\partial \theta_D} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_G \partial \theta_D}\right)^T \frac{\partial f(z)}{\partial \theta_G}, \quad -\frac{\partial f(z)}{\partial \theta_G} + \lambda \left(\frac{\partial^2 f(z)}{\partial \theta_D \partial \theta_G}\right)^T \frac{\partial f(z)}{\partial \theta_D}\right]^T \tag{6}$$

SGA

$$\left[\frac{dL_D}{d\theta_D}, \frac{dL_G}{d\theta_G}\right]^T = \left[\frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'}, -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'}\right]^T \tag{8}$$

LOGAN

LOGAN with Natural Gradient Descent (1)

- NGD is an approximate second-order optimization method (Pascanu & Bengio, 2013; Martens, 2014)
- NGD is expensive in high dimensional parameter spaces, even with approximations (Martens, 2014)
- For a gradient of z, $g = \frac{\partial f(z)}{\partial z}$, NGD computes the update as

$$\Delta z = \alpha F^{-1} g \tag{9}$$

where the Fischer information matrix F is defined as

$$F = E_{p(t|z)}[\nabla \ln p(t|z) \nabla \ln p(t|z)^T] \tag{10}$$

Where, $\ln p(t|z)$ is the log-likelihood function, commonly used for error functions such as cross entropy loss.

Nicolas L Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In Advances in neural information processing systems, pp. 849–856, 2008.

LOGAN with Natural Gradient Descent (2)

- The authors than use the empirical Fisher F' with Tikhonov damping from TONGA (Roux et al., 2007) to reduce the computational cost, since g is already available.

$$F' = g \cdot g^T + \beta I \tag{11}$$

- The damping is necessary to regularize the step size when F' poorly approximates the Hessian.
- Furthermore, using the Sherman-Morrison formula to invert F', we have the update in closed form

$$\left(A + uv^{\mathsf{T}}\right)^{-1} = A^{-1} - \frac{A^{-1}uv^{\mathsf{T}}A^{-1}}{1 + v^{\mathsf{T}}A^{-1}u} \qquad g = \frac{\partial f(z)}{\partial z}$$

$$\Delta z = \alpha \left(\frac{I}{\beta} - \frac{g\, g^T}{\beta^2 + \beta\, g^T\, g} \right) g = \frac{\alpha}{\beta + \|g\|^2}\, g \tag{12}$$

Experiments and Analysis

- The authors followed the standard BigGAN-deep architecture with 3 minor modifications:
  - Increased the size of the latent source
  - Used a uniform distribution
  - Used leakyReLU instead of ReLU

- To apply latent optimization, the authors used a damping factor $\beta = 5.0$ and step size of $\alpha = 0.9$
- No hyperparameters were optimized.

Basic Results

- LOGAN achieved improvements of 17% in IS and 32% in FID
- LOGAN is 2-3 times slower per step compared with BigGAN-deep.
  - This was due to the additional forward and backward pass.
- Optimizing *z* during evaluation did not improve sample scores.

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	$\mathbf{148.2 \pm 3.1}$

Ablation Studies

- Theoretical analysis was verified via ablation studies.
  - Experimented with basic gradient descent to optimizing z
- The authors removed parts of the equation shown below to see how the output is affected.

$$\left[\frac{dL_D}{d\theta_D}, \frac{dL_G}{d\theta_G}\right]^T = \left[\frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'}, -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'}\right]^T \tag{8}$$

- Removal of one (either terms) or both terms made training diverge early, but keeping both terms helped stabilize training.

Truncation and Samples

- Truncation is a technique to illustrate the trade-off between FID and IS in a trained model.
- Curves near upper right corner represent better quality.

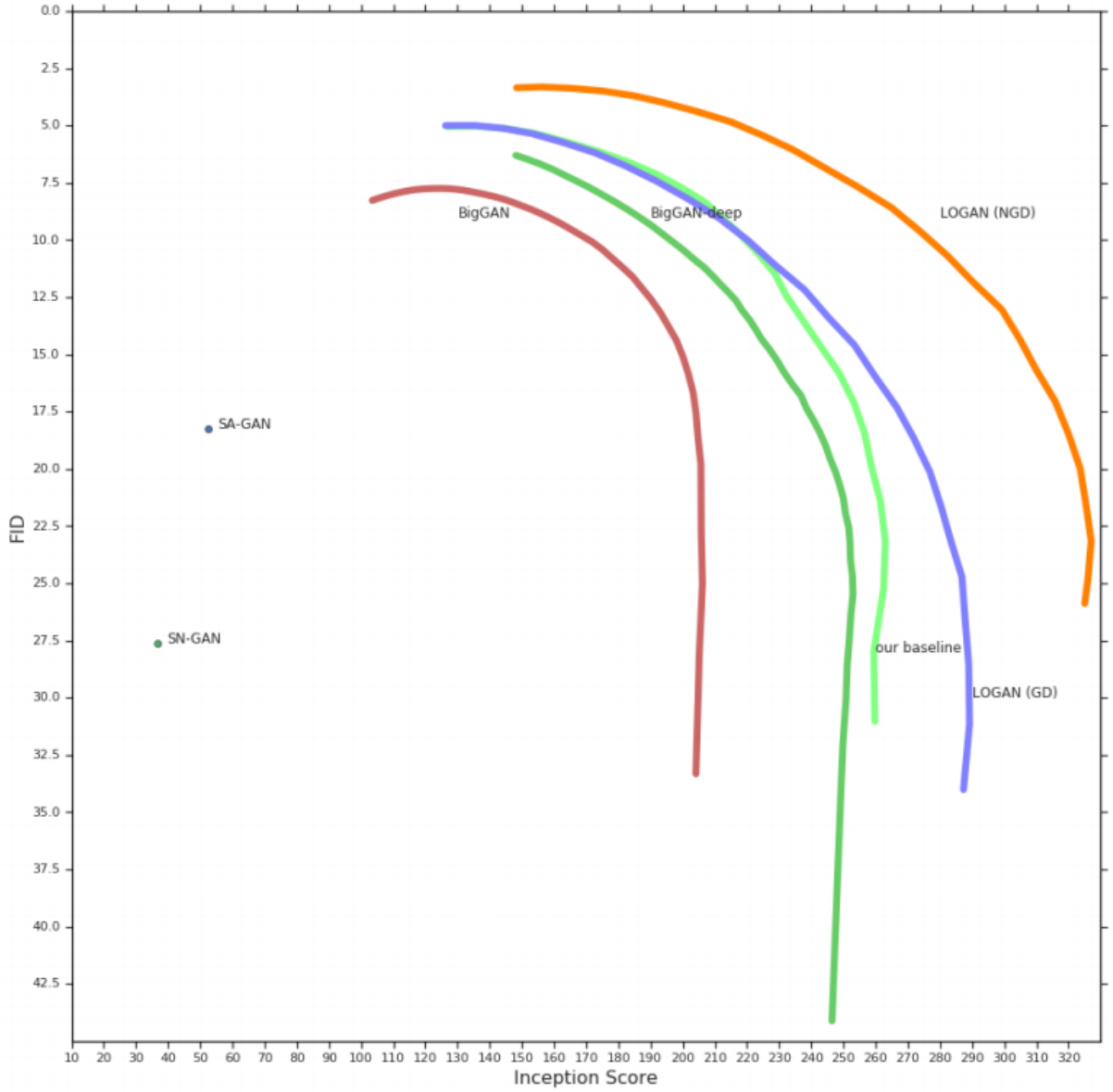
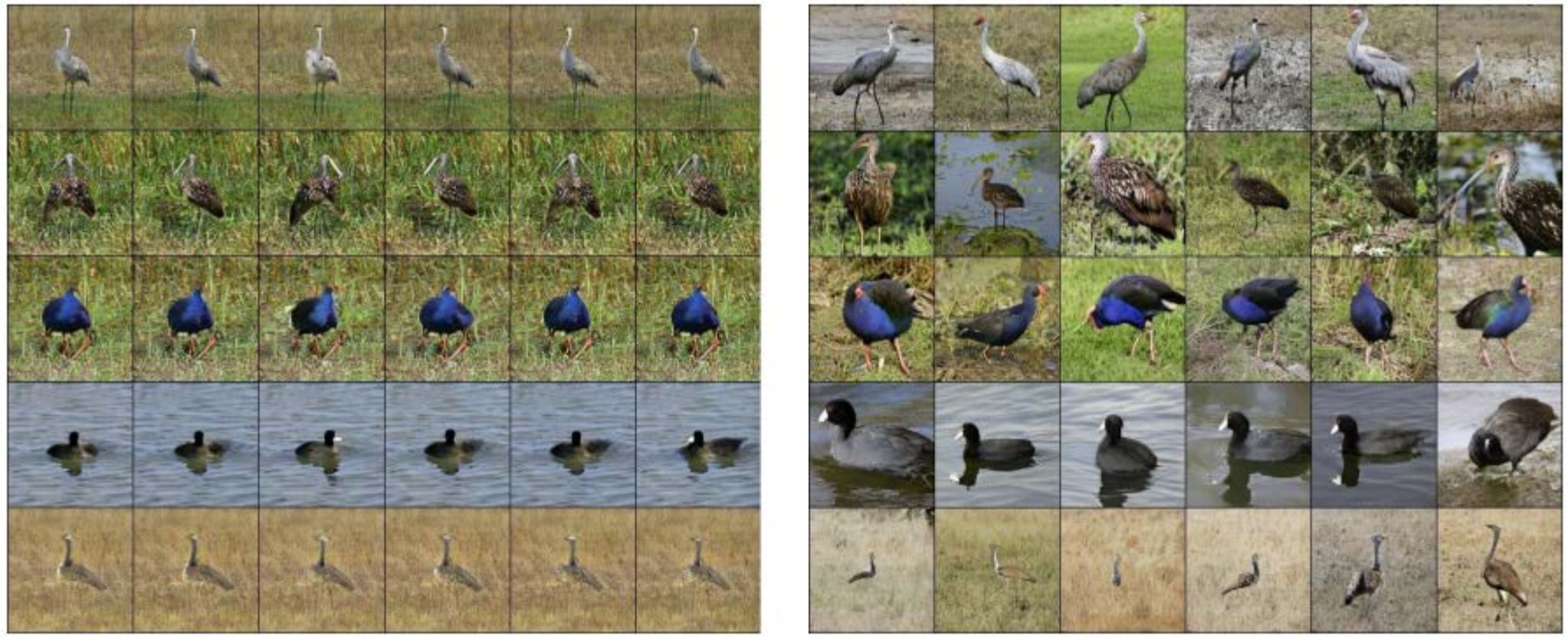
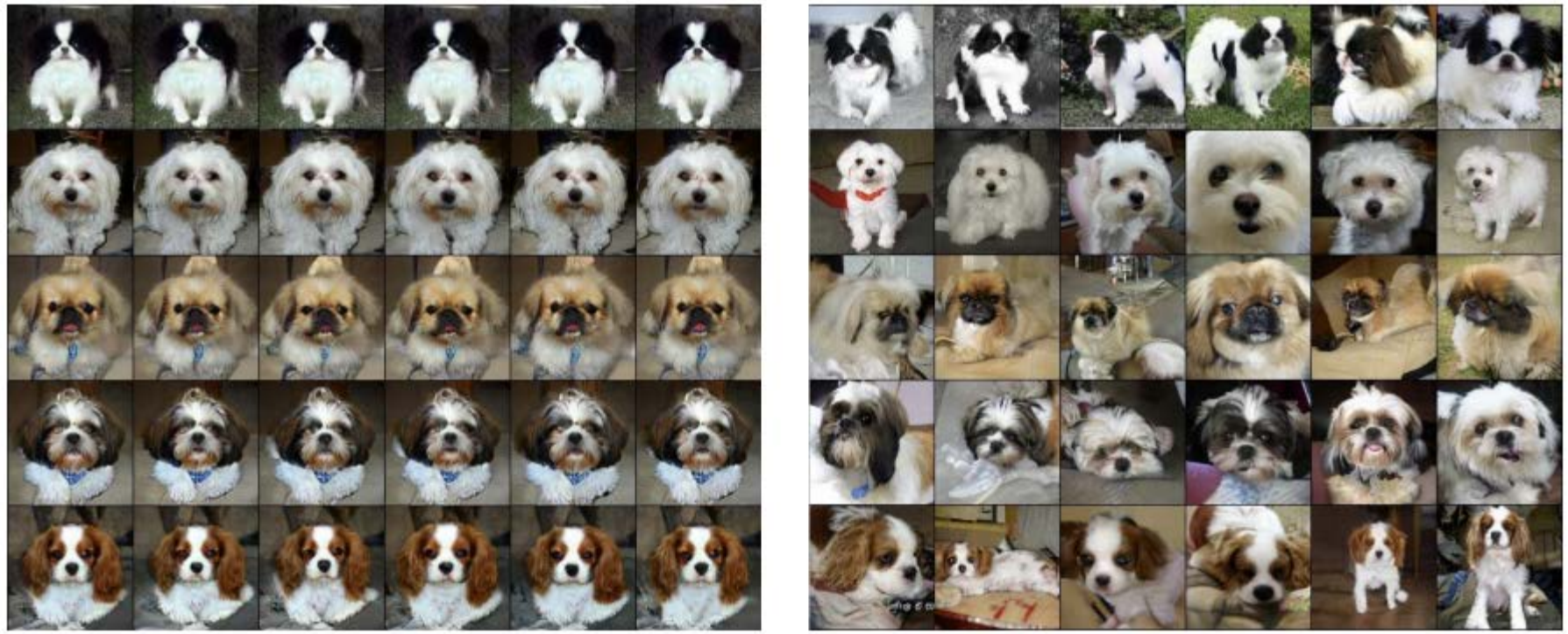


Figure 8: Truncation curves with additional baselines. In addition to the truncation curves reported in Figure 3**b**, here we also include the Spectral-Normalised GAN (Miyato et al., 2018), Self-Attention GAN (Zhang et al., 2019), original BigGAN and BigGAN-deep as presented in Brock et al. (2018).



(**a**) (**b**)

Figure 6: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly high inception scores. Samples from the two panels were drawn from truncations correspond to points C, D in figure 3. (FID/IS: (**a**) 27.97/259.4, (**b**) 8.19/259.9)

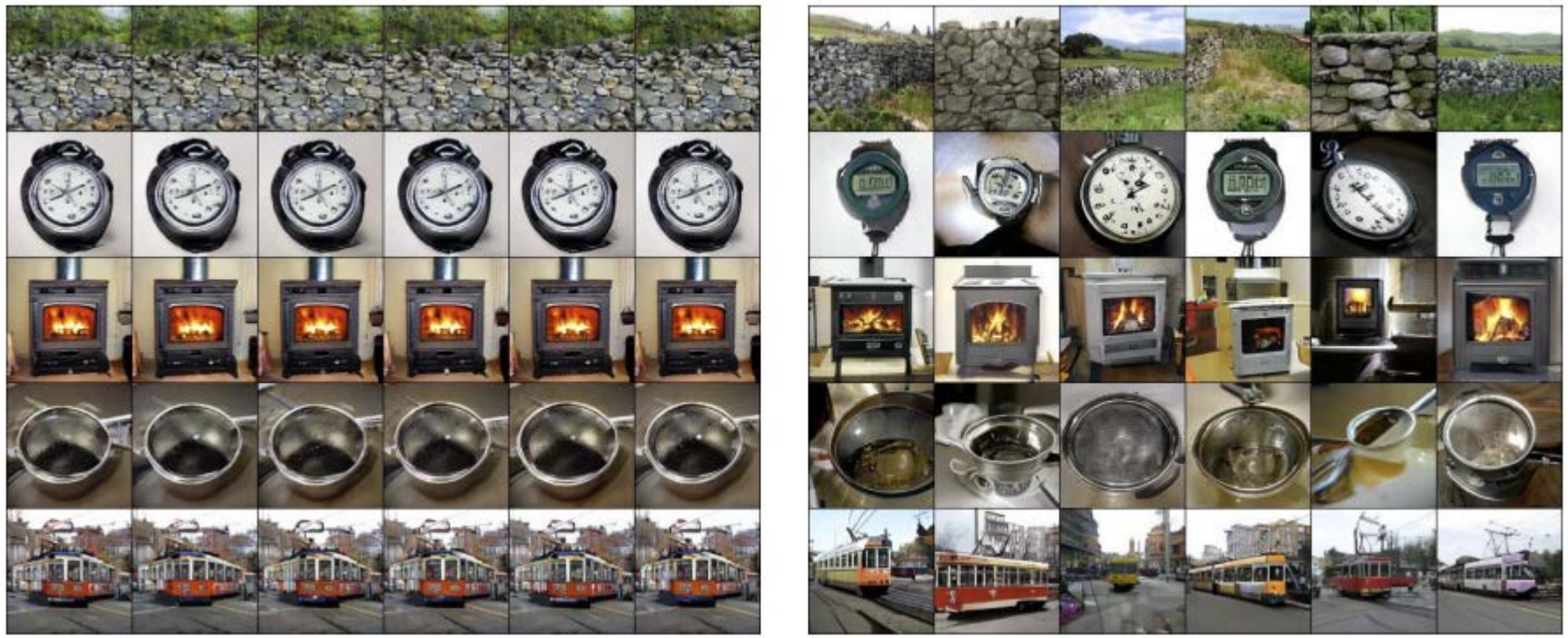


(**a**) (**b**)

Figure 6: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly high inception scores. Samples from the two panels were drawn from truncations correspond to points C, D in figure 3. (FID/IS: (**a**) 27.97/259.4, (**b**) 8.19/259.9)

(**a**) (**b**)

Figure 6: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly high inception scores. Samples from the two panels were drawn from truncations correspond to points C, D in figure 3. (FID/IS: (**a**) 27.97/259.4, (**b**) 8.19/259.9)



(**a**) (**b**)

Figure 6: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly high inception scores. Samples from the two panels were drawn from truncations correspond to points C, D in figure 3. (FID/IS: (**a**) 27.97/259.4, (**b**) 8.19/259.9)

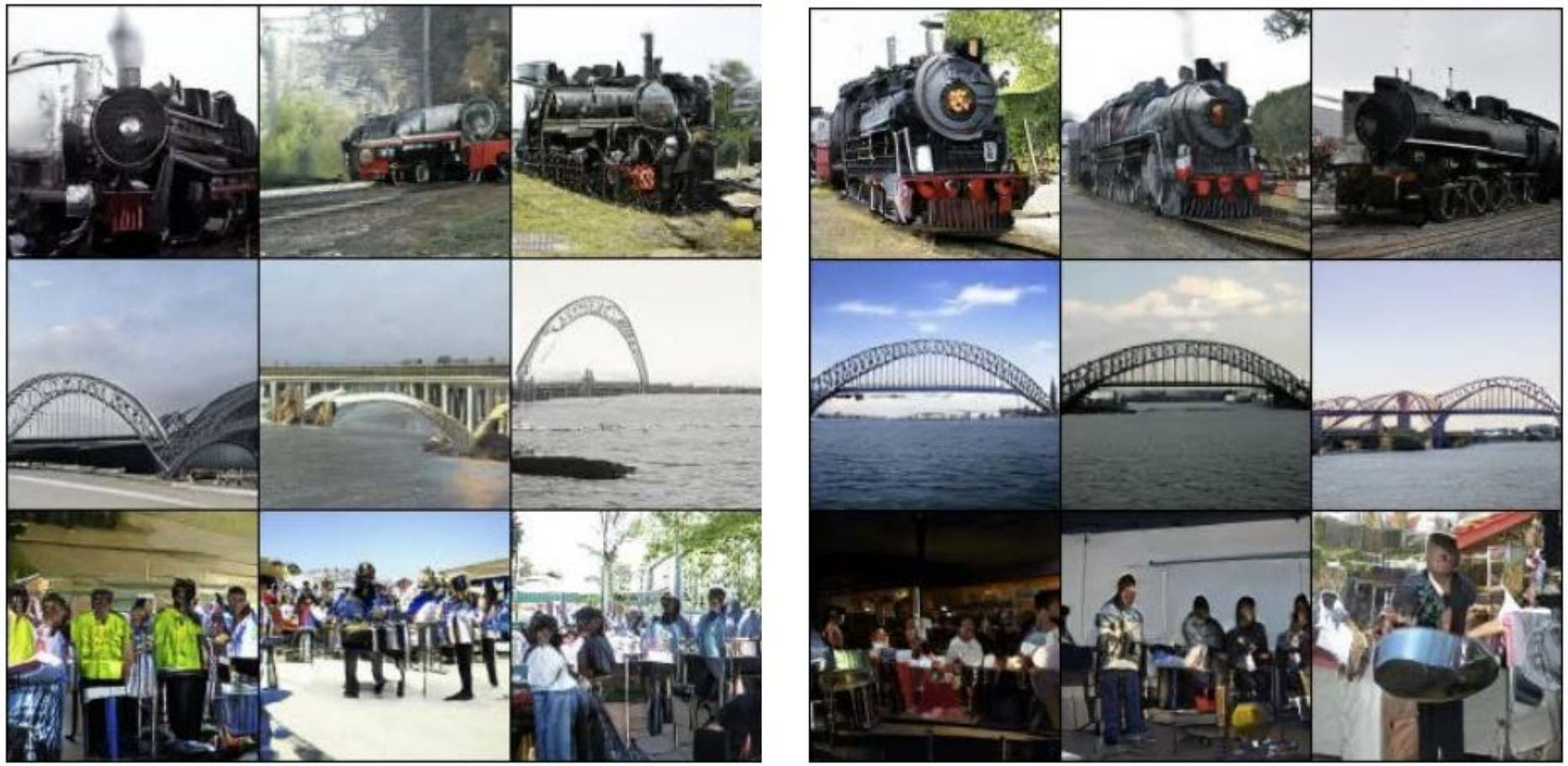


Figure 7: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly low FID. Samples from the two panels were draw from truncations correspond to points A, B in figure 3**b**. (FID/IS: (**a**) 5.04/126.8, (**b**) 5.09/217.0)

Figure 7: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly low FID. Samples from the two panels were draw from truncations correspond to points A, B in figure 3**b**. (FID/IS: (**a**) 5.04/126.8, (**b**) 5.09/217.0)

Figure 7: Samples from BigGAN-deep (**a**) and LOGAN (**b**) with the similarly low FID. Samples from the two panels were draw from truncations correspond to points A, B in figure 3**b**. (FID/IS: (**a**) 5.04/126.8, (**b**) 5.09/217.0)

OpenReview.net

- This paper was submitted to the ICLR 2020 Conference Blind Submission and its reviews are available online
- Two of the reviewers gave a weak accept rating and the chair reviewer rejected the paper.
- Negative points were: the lack of code examples, relationship with SGA is confusing, more examples and comparisons should've been presented
- Positive points were: clear improvement over state-of-the-art, demonstration of efficiently exploiting second order dynamics in GAN's training.

Implementation

*"This results in a scalable and **easy to implement** approach that improves the dynamic interaction between the discriminator and the generator."*

-- Yan Wu et al. 2019

Latent Optimized GANs with Automatic Differentiation

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

Assignment 3 - Q5: CS231n Generative Adversarial Networks

```
def run_a_gan(D, G, D_solver, G_solver, discriminator_loss, generator_loss, show_every=250,
              batch_size=128, noise_size=96, num_epochs=20);
```

```
    [...]
    for epoch in range(num_epochs):
        [...]
            ## latent optimization step
            G_solver.zero_grad(), D_solver.zero_grad()
            z = sample_noise(batch_size, noise_size).type(dtype)
            z_prime = lat_opt_ngd(G,D,z, batch_size) # ngd
            # z_prime = lat_opt_gd(G,D,z, batch_size)  # gd

            ## regular training
            D_solver.zero_grad()
            real_data = x.type(dtype)
            logits_real = D(2* (real_data - 0.5)).type(dtype)

            g_fake_seed = z_prime #sample_noise(batch_size, noise_size).type(dtype)
            fake_images = G(g_fake_seed).detach()
        [...]
```

MNIST

$$\Delta z = \alpha \frac{\partial f(z)}{\partial z} \qquad z' = z + \Delta z \tag{4}$$

```python
def lat_opt_gd(G,D,z, batch_size, alpha=1.2):
    x_hat = G(z)
    f_z = D(x_hat.view(batch_size, 1, 28, 28))

    fz_dz = torch.autograd.grad(outputs=f_z,
                                inputs= z,
                                grad_outputs=torch.ones_like(f_z),
                                retain_graph=True,
                                create_graph= True
                                   )[0]

    delta_z = torch.ones_like(fz_dz)
    delta_z = alpha * fz_dz

    with torch.no_grad():
        z_prime = torch.clamp(z + delta_z, min=-1, max=1)

    return z_prime
```

$$\Delta z = \alpha \left(\frac{I}{\beta} - \frac{g\, g^T}{\beta^2 + \beta\, g^T\, g} \right) g = \frac{\alpha}{\beta + \|g\|^2}\, g \tag{12}$$

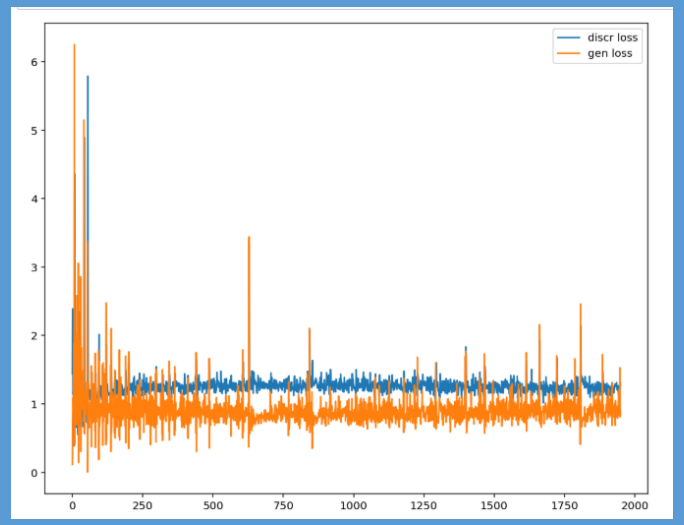
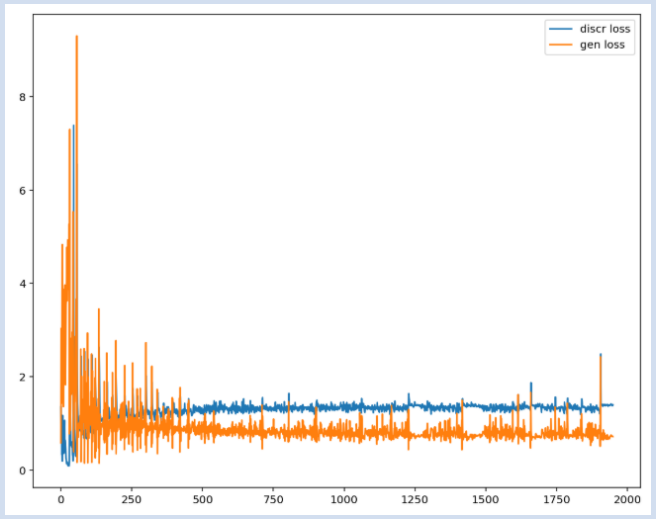
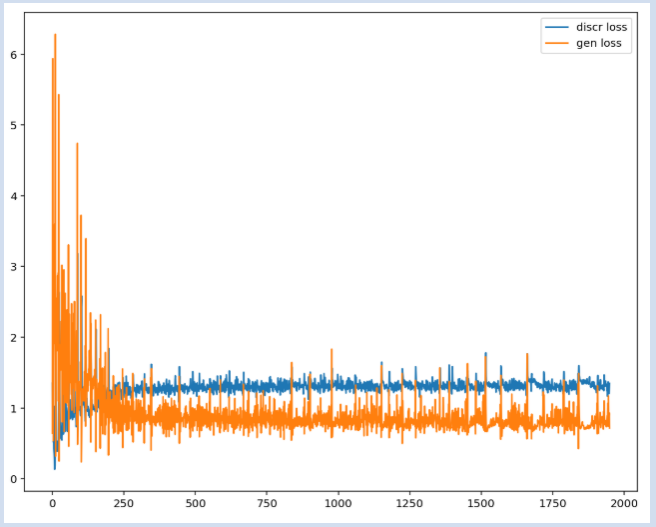
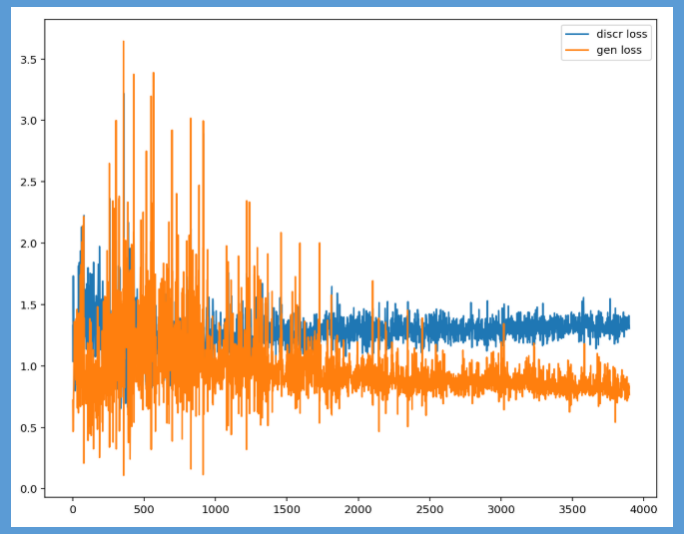
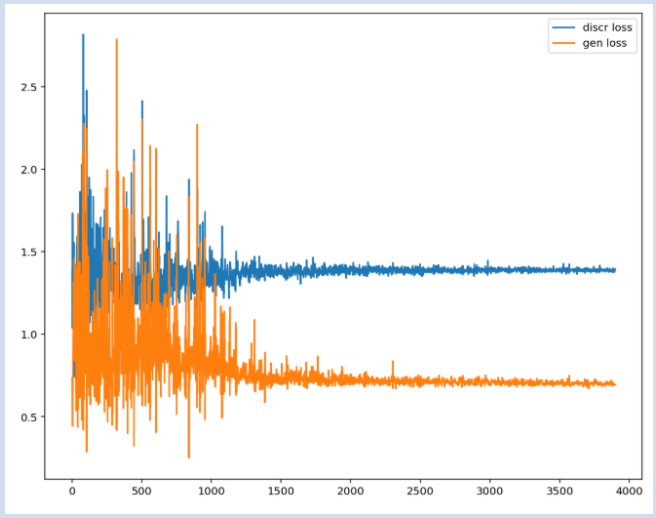
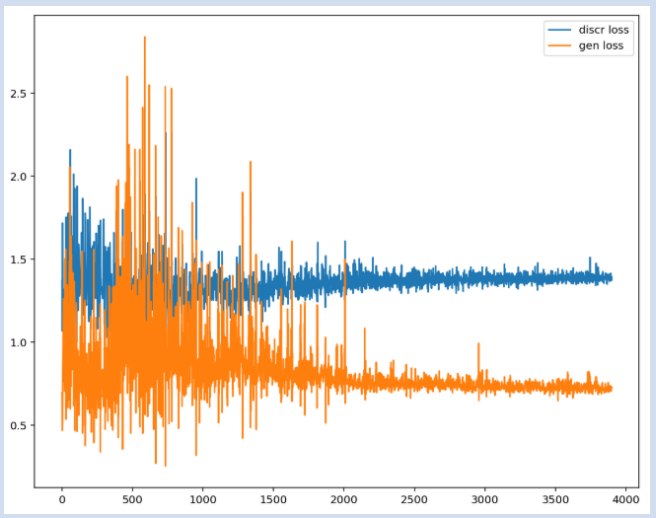
```python
def lat_opt_ngd(G,D,z, batch_size, alpha=0.9, beta=0.1, norm=2):
    x_hat = G(z)
    f_z = D(x_hat.view(batch_size, 1, 28, 28))

    fz_dz = torch.autograd.grad(outputs=f_z,
                                inputs= z,
                                grad_outputs=torch.ones_like(f_z),
                                retain_graph=True,
                                create_graph= True
                                   )[0]

    delta_z = torch.ones_like(fz_dz)
    delta_z = (alpha * fz_dz) / (beta +  torch.norm(delta_z, p=2, dim=0))

    with torch.no_grad():
        z_prime = torch.clamp(z + delta_z * norm, min=-1, max=1)

    return z_prime
```

Vanilla DC-GAN	DC-GAN+LOGAN-GD	DC-GAN+LOGAN-NGD
		
Vanilla GAN	LOGAN-GD	GAN+LOGAN-NGD
		

Thank you!

Questions?

Extra Material

Appendix B: Approximate SGA (1)

3−player simultaneous gradient

$$g = \left[\eta\frac{\partial L_G(z')}{\partial \Delta z}, \frac{\partial L_D(z')}{\partial \theta_D}, \frac{\partial L_G(z')}{\partial \theta_G}\right]^T = \left[-\eta\frac{\partial f(z')}{\partial \Delta z}, \frac{\partial f(z')}{\partial \theta_D}, -\frac{\partial f(z')}{\partial \theta_G}\right]^T \qquad H = \begin{bmatrix} -\eta\frac{\partial^2 f(z')}{\partial \Delta z^2} & -\eta\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_D} & -\eta\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_G} \\ \frac{\partial^2 f(z')}{\partial \theta_D \partial \Delta z} & \frac{\partial^2 f(z')}{\partial \theta_D^2} & \frac{\partial^2 f(z')}{\partial \theta_D \partial \theta_G} \\ -\frac{\partial^2 f(z')}{\partial \theta_G \partial \Delta z} & -\frac{\partial^2 f(z')}{\partial \theta_G \partial \theta_D} & -\frac{\partial^2 f(z')}{\partial \theta_G^2} \end{bmatrix}$$

SGA adjusted gradient

$g^* = g + \lambda A^T g$, where λ is a positive constant (5)

$$A = \frac{1}{2}(H - H^T) = \begin{bmatrix} 0 & -\frac{1+\eta}{2}\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_D} & \frac{1-\eta}{2}\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_G} \\ \frac{1+\eta}{2}\frac{\partial^2 f(z')}{\partial \theta_D \partial \Delta z} & 0 & \frac{\partial^2 f(z')}{\partial \theta_D \partial \theta_G} \\ -\frac{1-\eta}{2}\frac{\partial^2 f(z')}{\partial \theta_G \partial \Delta z} & -\frac{\partial^2 f(z')}{\partial \theta_G \partial \theta_D} & 0 \end{bmatrix} \qquad \eta = \frac{1}{N},\ \eta \ll 1 \text{ for large batches}$$

$$\gamma = \frac{1+\eta}{2} \approx \frac{1-\eta}{2}$$

$$g^* = g + \lambda A^T g = \begin{bmatrix} g^*_{\Delta z} \\ g^*_{\mathrm{D}} \\ g^*_{\mathrm{G}} \end{bmatrix} \longrightarrow$$

$g^*_{\Delta z}$ is not used for training and thus is ignored

$$g^*_D = \frac{\partial f(z')}{\partial \theta_D} + \lambda\gamma\left(\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial \Delta z} + \lambda\left(\frac{\partial^2 f(z')}{\partial \theta_G \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial \theta_G} \qquad (17)$$

$$g^*_G = -\frac{\partial f(z')}{\partial \theta_G} - \lambda\gamma\left(\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial \Delta z} + \lambda\left(\frac{\partial f(z')}{\partial \theta_D \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial \theta_D} \qquad (18)$$

Appendix B: Approximate SGA (2)

$$g_D^* = \frac{\partial f(z')}{\partial \theta_D} + \lambda\gamma \left(\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial \Delta z} + \lambda \left(\frac{\partial^2 f(z')}{\partial \theta_G \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial \theta_G} \tag{17}$$

$$g_G^* = -\frac{\partial f(z')}{\partial \theta_G} - \lambda\gamma \left(\frac{\partial^2 f(z')}{\partial \Delta z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial \Delta z} + \lambda \left(\frac{\partial f(z')}{\partial \theta_D \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial \theta_D} \tag{18}$$

Computationally intensive

$$\frac{\partial f(z')}{\partial \Delta z} = \frac{\partial f(z')}{\partial z'}$$

$$g_D^* \approx \frac{\partial f(z')}{\partial \theta_D} + \lambda\gamma \left(\frac{\partial^2 f(z')}{\partial z' \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'} \tag{19}$$

$$g_G^* \approx -\frac{\partial f(z')}{\partial \theta_G} - \lambda\gamma \left(\frac{\partial^2 f(z')}{\partial z' \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'} \tag{20}$$

Still depends on the 3rd player since there are terms dependent on z'

The second derivative term is hard to compute, thus the authors introduce a local approximation

$$\frac{\partial^2 f(z')}{\partial z' \partial \theta_D} \approx \frac{\partial^2 f(z)}{\partial z \partial \theta_D} \qquad \frac{\partial^2 f(z')}{\partial z' \partial \theta_D} \approx \frac{\partial^2 f(z)}{\partial z \partial \theta_D} \tag{21}$$

θ_G

$$g_D^* \approx \frac{\partial^2 f(z')}{\partial \theta_D} + \lambda\gamma \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'}$$

$$g_G^* \approx \frac{\partial^2 f(z')}{\partial \theta_G} + \lambda\gamma \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'}$$

$$\left[\frac{dL_D}{d\theta_D}, \frac{dL_G}{d\theta_G}\right]^T = \left[\frac{\partial f(z')}{\partial \theta_D} + \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_D}\right)^T \frac{\partial f(z')}{\partial z'}, -\frac{\partial f(z')}{\partial \theta_G} - \alpha \left(\frac{\partial^2 f(z)}{\partial z \partial \theta_G}\right)^T \frac{\partial f(z')}{\partial z'}\right]^T \tag{8}$$

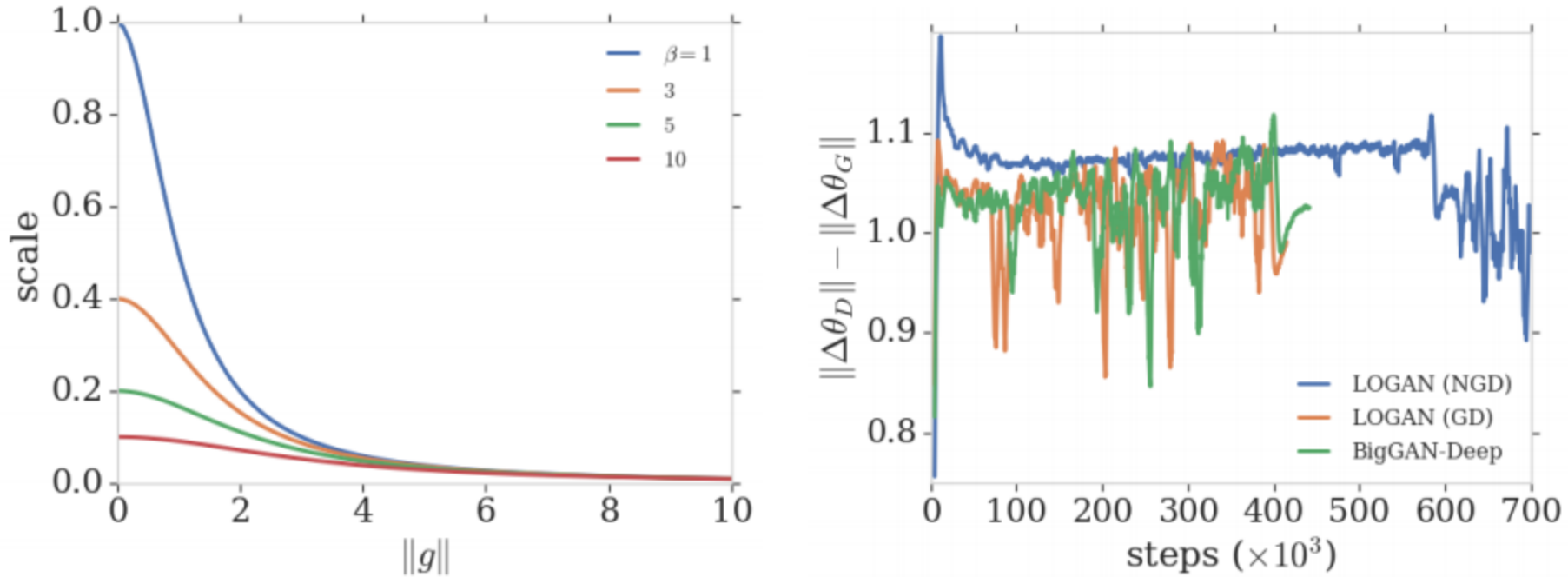


Figure 4: (**a**) Scaling of gradients in natural gradient descent. We use $\beta = 5$ in BigGAN-Deep experiments. (**b**) The update speed of the discriminator relative to the generator shown as the difference $\|\Delta\theta_D\| - \|\Delta\theta_G\|$ after each update step. Lines are smoothed with a moving average using window size 20 (in total, there are 3007, 1659 and 1768 data points for each curve). All curves oscillated strongly after training collapsed.

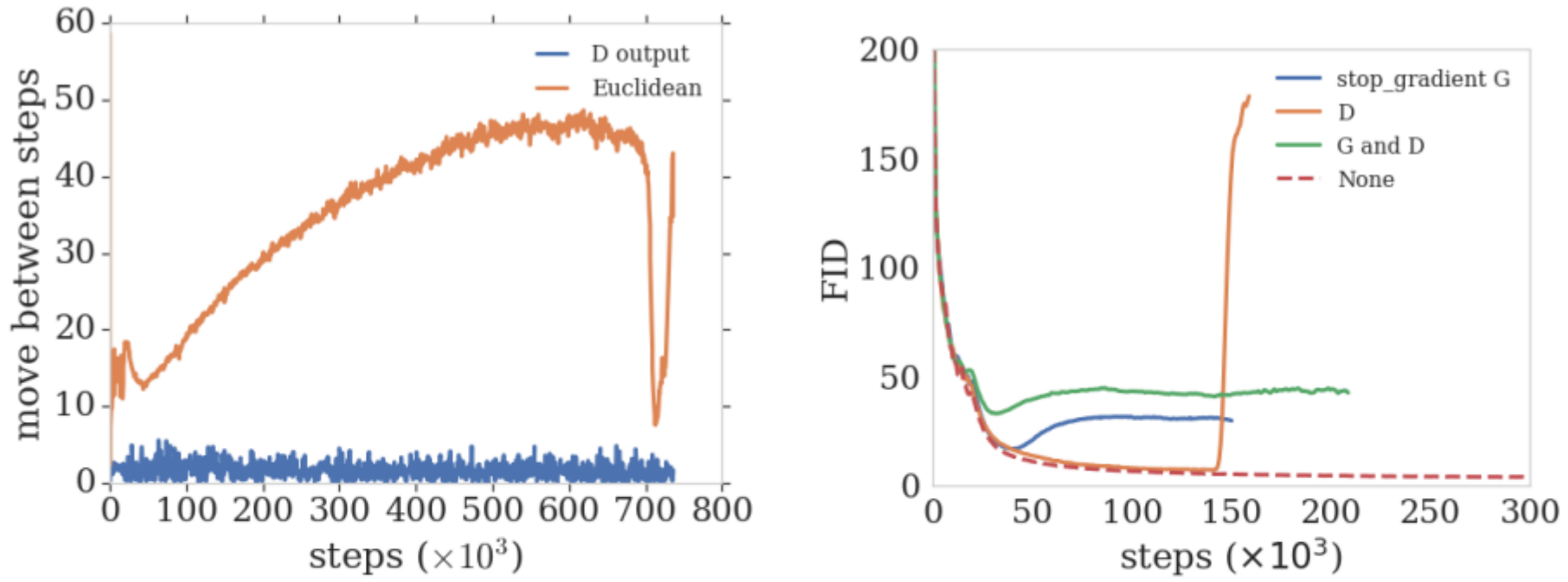


Figure 5: (**a**) The change from Δz across training, in D's output space and z's Euclidean space. The distances are normalised by their standard derivations computed from a moving window of size 20 (1007 data points in total). (**b**) Training curves from models with different "`stop_gradient`" operations. For reference, the training curve from an unablated model is plotted as the dashed line. All instances with `stop_gradient` collapsed (FID went up) early in training.

Background

- Inception Score (IS)
  - Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In NIPS, 2016.
- Fréchet Inception Distance (FID)
  - Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In NIPS, 2017.
- Truncation trick
  - Andrew Brock, Jeff Donahue, Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. 2018