深度学习

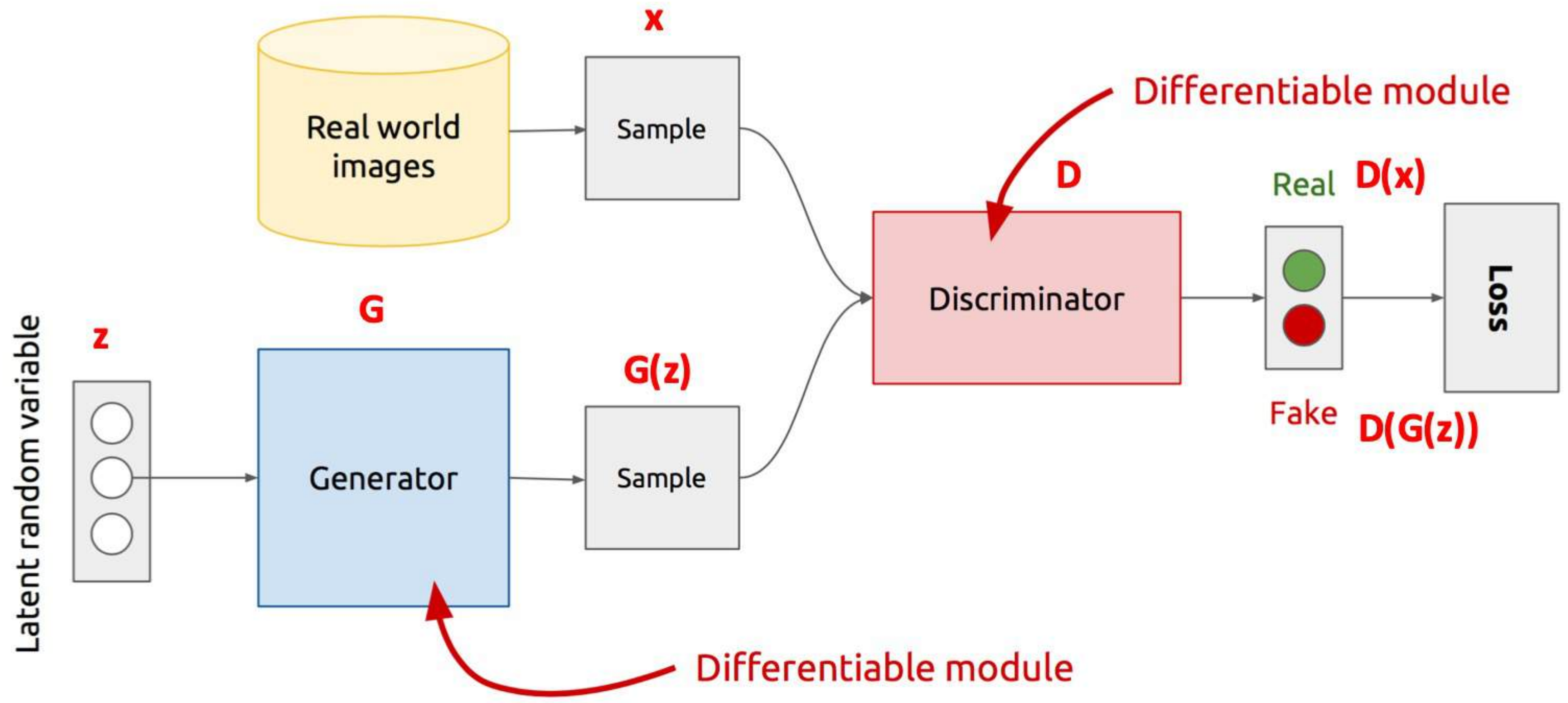
Lecture 13 GAN

Pang Tongyao, YMSC

Generative Adversarial Networks (GANs)

GANs

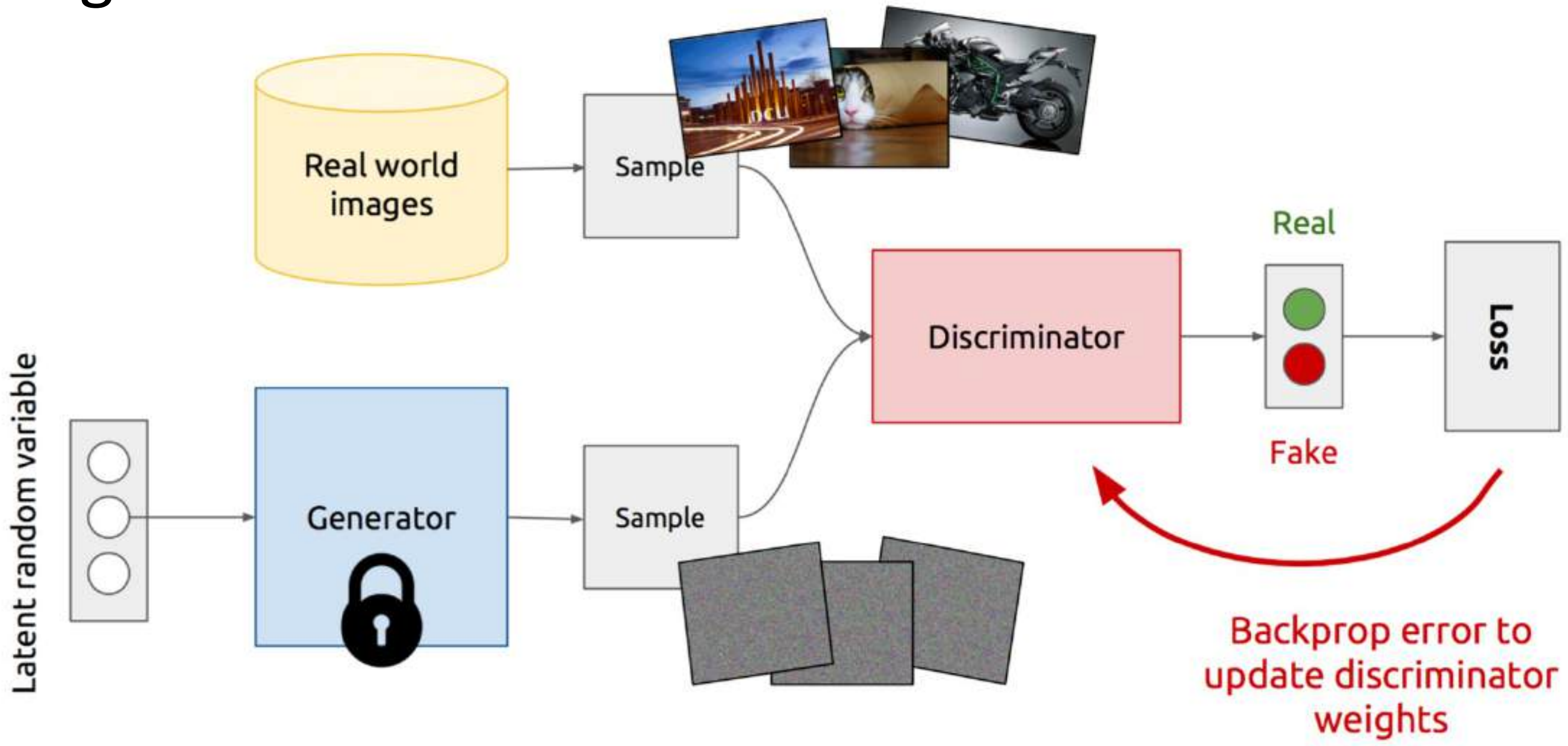
GAN’s Architecture



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GANs

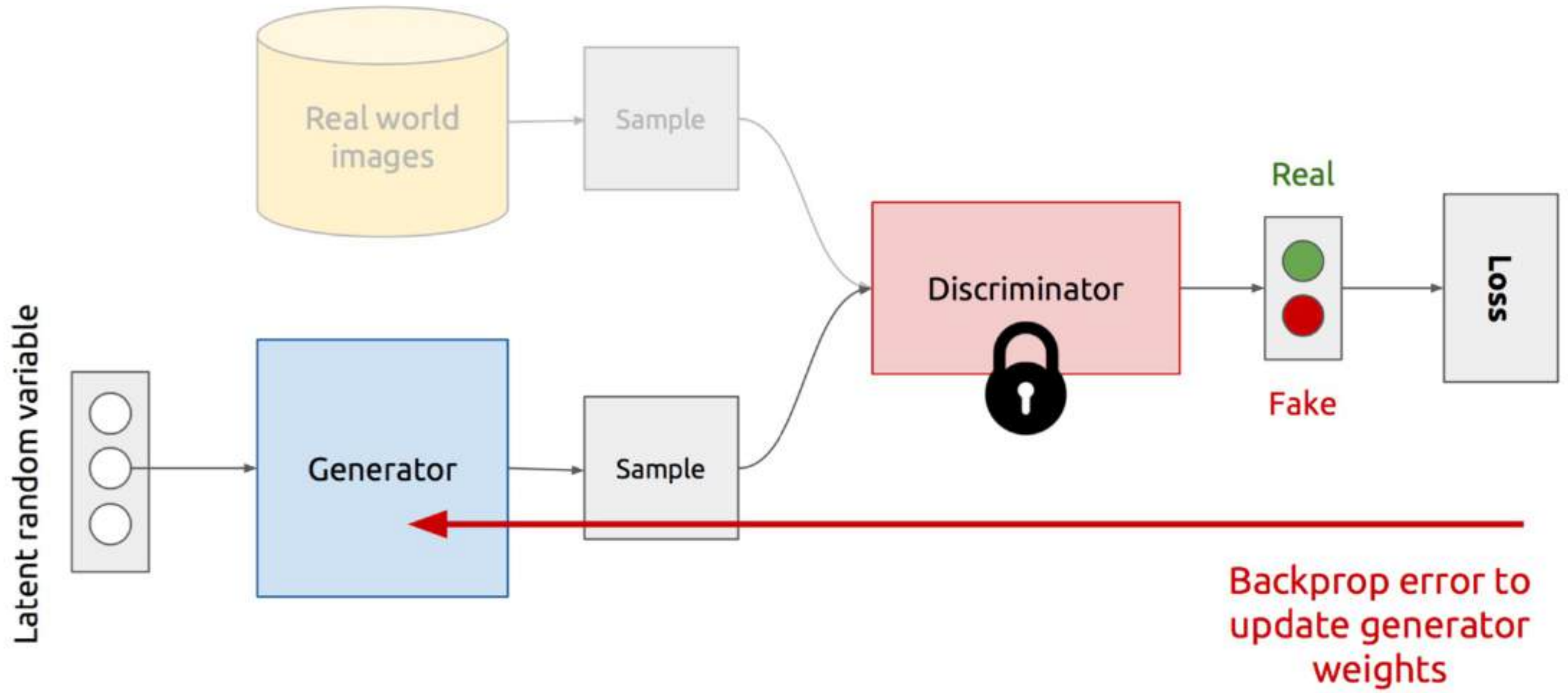
Training Discriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GANs

Training Generator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GANs

- Formulation: a minimax game

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\Big(1 - D\big(G(z)\big)\Big)\right]$$

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned}$$

Goodfellow et al, “Generative Adversarial Nets “, 2014

GANs

- When D is optimized,

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

- C(G) is actually the Jensen Shannon divergence between p_{data} and p_g:

$$
C(G) = -\log(4) + \boxed{KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)}
$$

which achieves the global minimum if and only if $p_g = p_{data}$.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator updates

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Non-Convergence

- GANs involve two players:
  - Discriminator is trying to maximize its reward.
  - Generator is trying to minimize Discriminator's reward.
  - SGD can not guarantee converging to a Nash equilibrium
- A simple example: $\min_x \max_y xy$

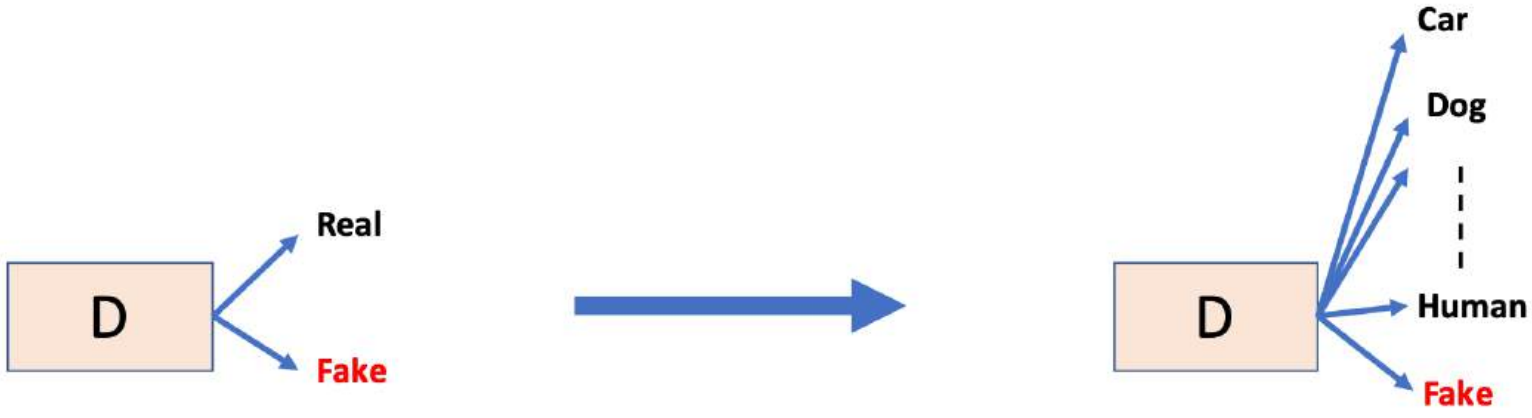
Gradient descent (ascent): $\frac{dx(t)}{dt} = -y, \frac{dy(t)}{dt} = x \Rightarrow \frac{d^2x(t)}{dt^2} = -x(t)$

So the trajectory of (x, y) is a circle:

$$x(t) = x(0)\cos(t) - y(0)\sin(t), y(t) = x(0)\sin(t) + \mathrm{y}(0)\cos(t)$$

NOT CONVERGENT!

Semi-supervision



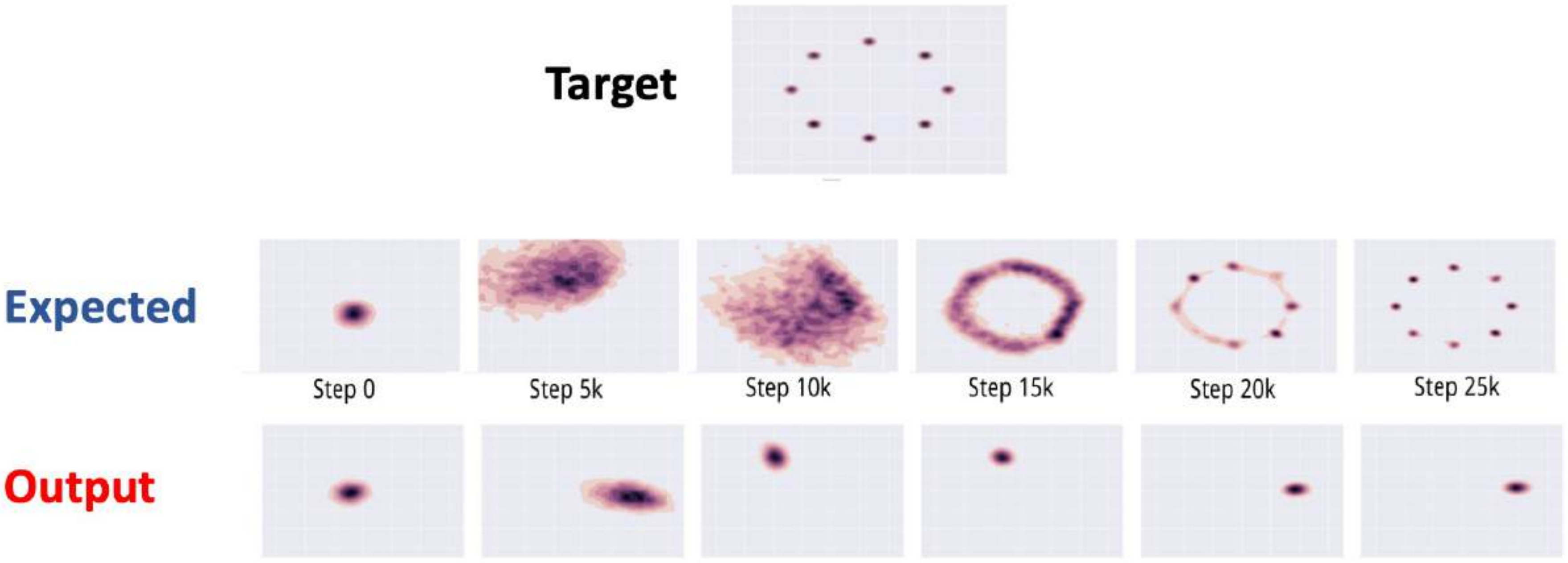
$$
\begin{aligned}
L &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)}[\log p_{\text{model}}(y|\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim G}[\log p_{\text{model}}(y=K+1|\boldsymbol{x})] \\
&= L_{\text{supervised}} + L_{\text{unsupervised}}, \text{ where} \\
L_{\text{supervised}} &= -\mathbb{E}_{\boldsymbol{x},y\sim p_{\text{data}}(\boldsymbol{x},y)} \log p_{\text{model}}(y|\boldsymbol{x}, y<K+1) \\
L_{\text{unsupervised}} &= -\{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} \log[1 - p_{\text{model}}(y=K+1|\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim G} \log[p_{\text{model}}(y=K+1|\boldsymbol{x})]\},
\end{aligned}
$$

- if $p_{model}(y = K + 1|x) = 0$, the supervised loss is the standard loss of training a classifier with K classes.
- if $D(x) = 1 - p_{model}(y = K + 1|x)$, the unsupervised loss is the standard GAN loss.

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Mode-Collapse

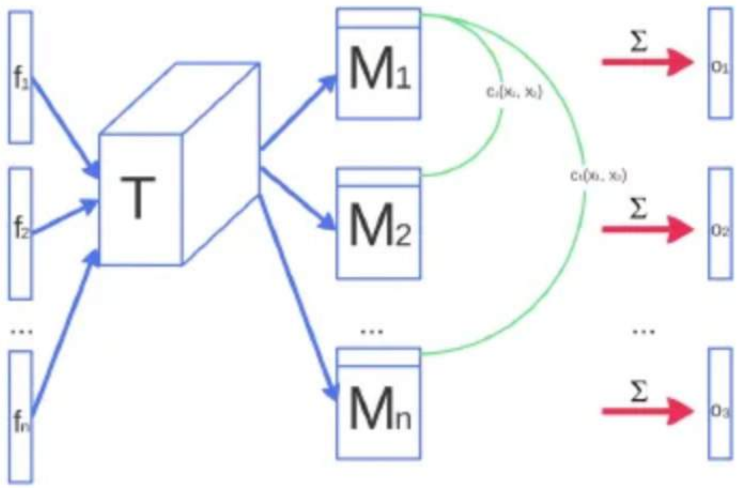
- Generator fails to output diverse samples; places all mass on most likely point.



Metz, Luke, et al. **"Unrolled Generative Adversarial Networks."** arXiv preprint arXiv:1611.02163 (2016).

Heuristic Solutions

- Rewarding sample diversity to avoid mode collapse.
- Mini-batch features: capture diversity between the mini-batch



Minibatch Discrimination

$$c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||M_{i,b} - M_{j,b}||_{L_1})$$

$$o(\boldsymbol{x}_i)_b = \sum_{j=1}^{n} c_b(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}$$

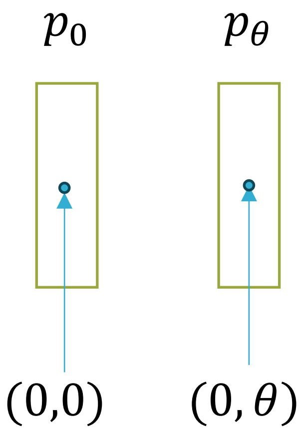
$$o(\boldsymbol{x}_i) = \left[o(\boldsymbol{x}_i)_1, o(\boldsymbol{x}_i)_2, \ldots, o(\boldsymbol{x}_i)_B\right] \in \mathbb{R}^B$$

$$o(\mathbf{X}) \in \mathbb{R}^{n \times B}$$

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Vanishing Gradients

- When the discriminator D is trained very well, GAN loss is to minimize the JS divergence between p_g and p_{data}.
- JS divergence is a constant when the support of two distribution is non-overlapped, which causes vanishing gradients.



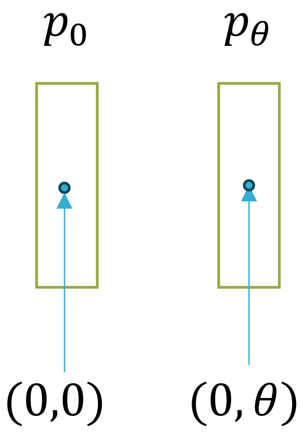
$$JS(p_0, p_\theta) = \begin{cases} 0, & \theta = 0 \\ \log 2, & \theta \neq 0 \end{cases}$$

WGANs

- Wasserstein Distance or Earth-Mover (EM) distance

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \,,$$

Joint distributions whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$



$W(p_0, p_\theta) = |\theta|$ is continuous with respect to θ.

Wasserstein Distance may be a better measure than JS distance for training GAN, that is why Wasserstein GAN is introduced.

WGAN

- Wasserstein distance can be calculated in a dual form

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

f: the discriminator.

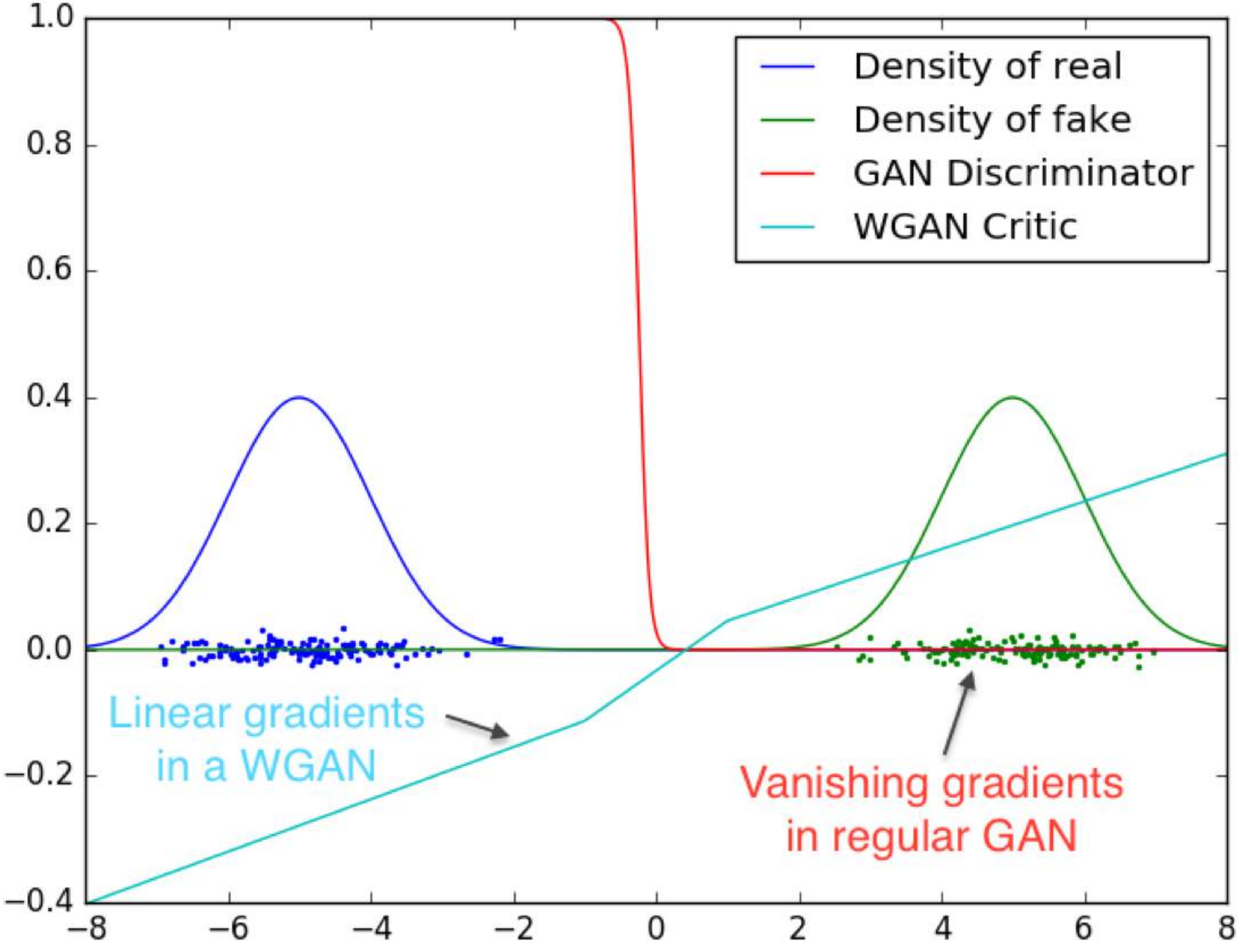
Lipschitz constant

Relaxing the condition $||f||_L \leq 1$ to $||f||_L \leq M$, the canulation is consistent except a multiplicative constant.

- In the original paper of WGAN, $||f||_L \leq M$ is guaranteed by clip the network parameters to [-c, c].(Recall that iResNet also requires $||g||_L \leq 1$, g is the residual network.) How they achieve that? (Spectral Normalization)
- The generator is updated via minimizing the Wasserstein distance

$$\min_{g_\theta} W\left(p_{data}, p_g\right) = \min_{g_\theta} \max_{f_w} \mathbb{E}_{x \sim p_{data}} [f_w(x)] - \mathbb{E}_{z \sim q} [f_w(g_\theta(z))]$$

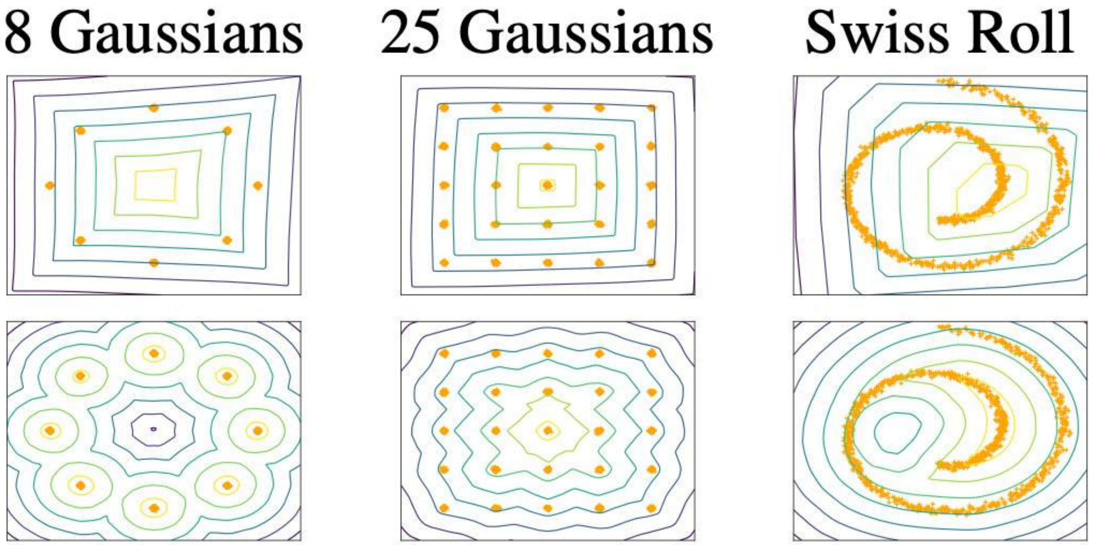
Arjovsky et al. “Wasserstein GAN”, 2017.



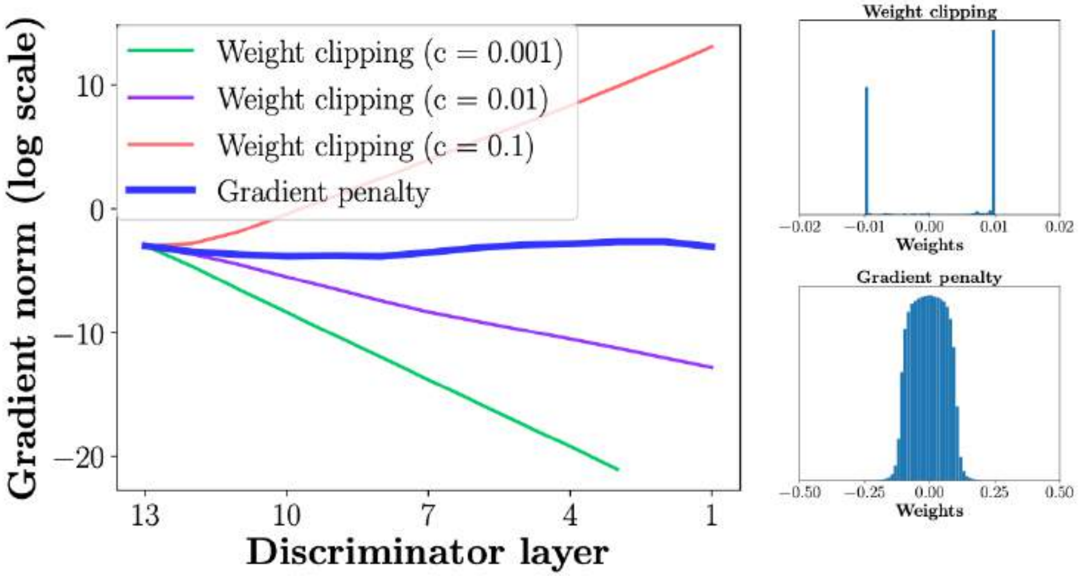
Density of real
Density of fake
GAN Discriminator
WGAN Critic
Linear gradients in a WGAN
Vanishing gradients in regular GAN
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−8
−6
−4
−2
0
2
4
6
8

WGAN-GP

- Weight clipping still suffers from



failing to capture higher moments of the data distribution



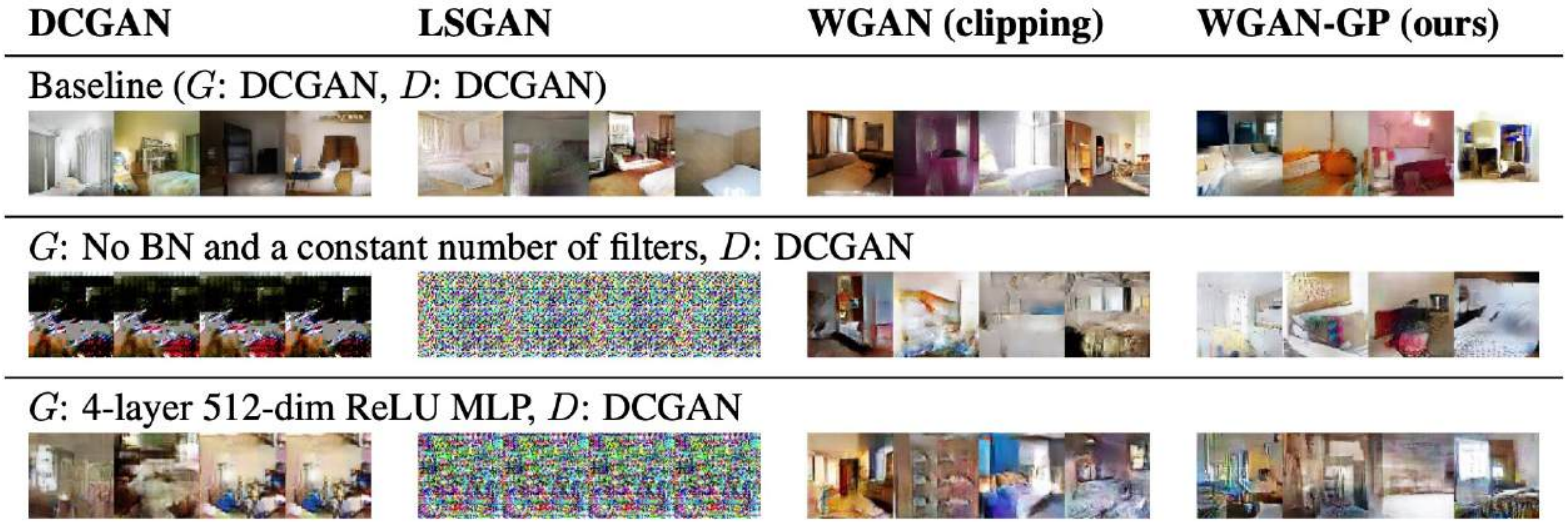
gradient vanishing or exploding

Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." *arXiv preprint arXiv:1704.00028* (2017).

WGAN-GP

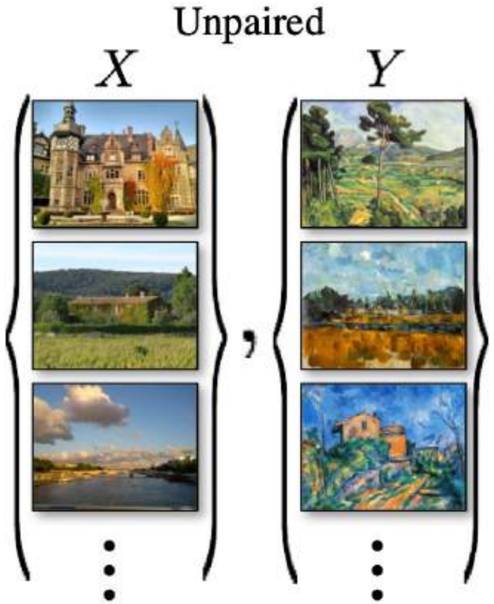
- Alternative way to enforce the Lipschitz constraint

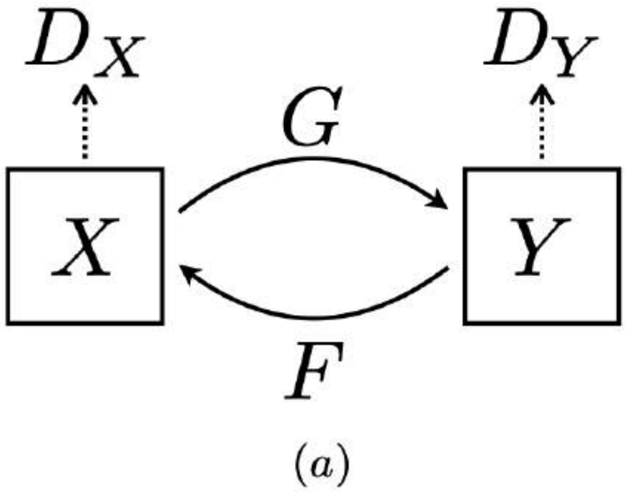
$$L = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \left[D(\tilde{\boldsymbol{x}}) \right] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x}) \right] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right].$$



Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." *arXiv preprint arXiv:1704.00028* (2017).

Cycle-GAN

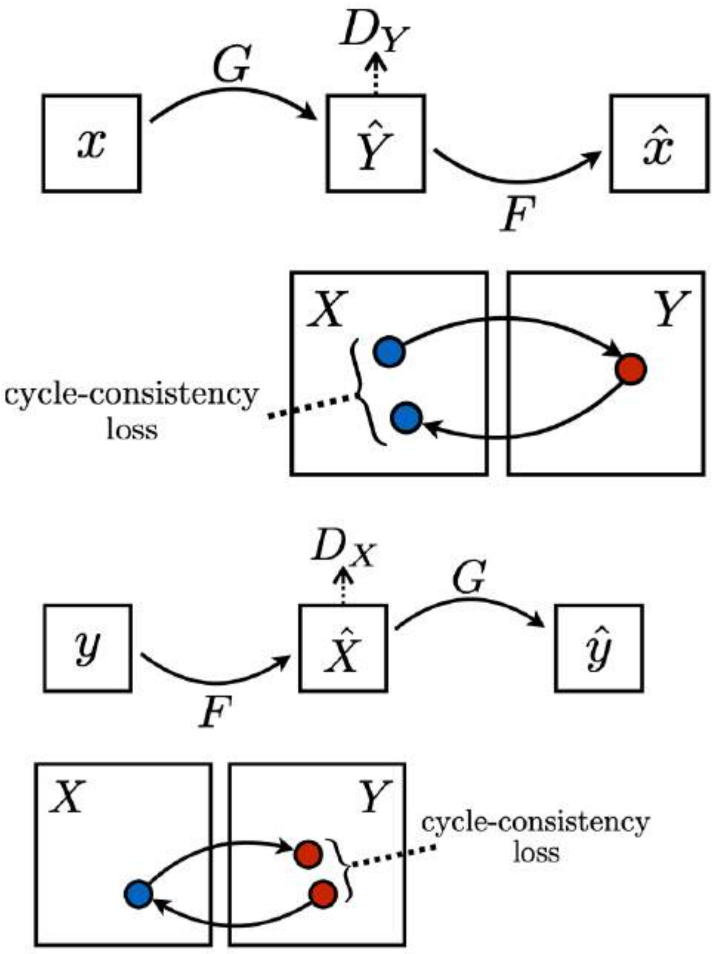




- Unpaired training dataset $\{(X, Y)\}$
- G: translate X to Y
- F: translate Y to X
- D_X: distinguish X and F(Y)
- D_Y: distinguish Y and G(X)

Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”, 2020

Cycle-GAN



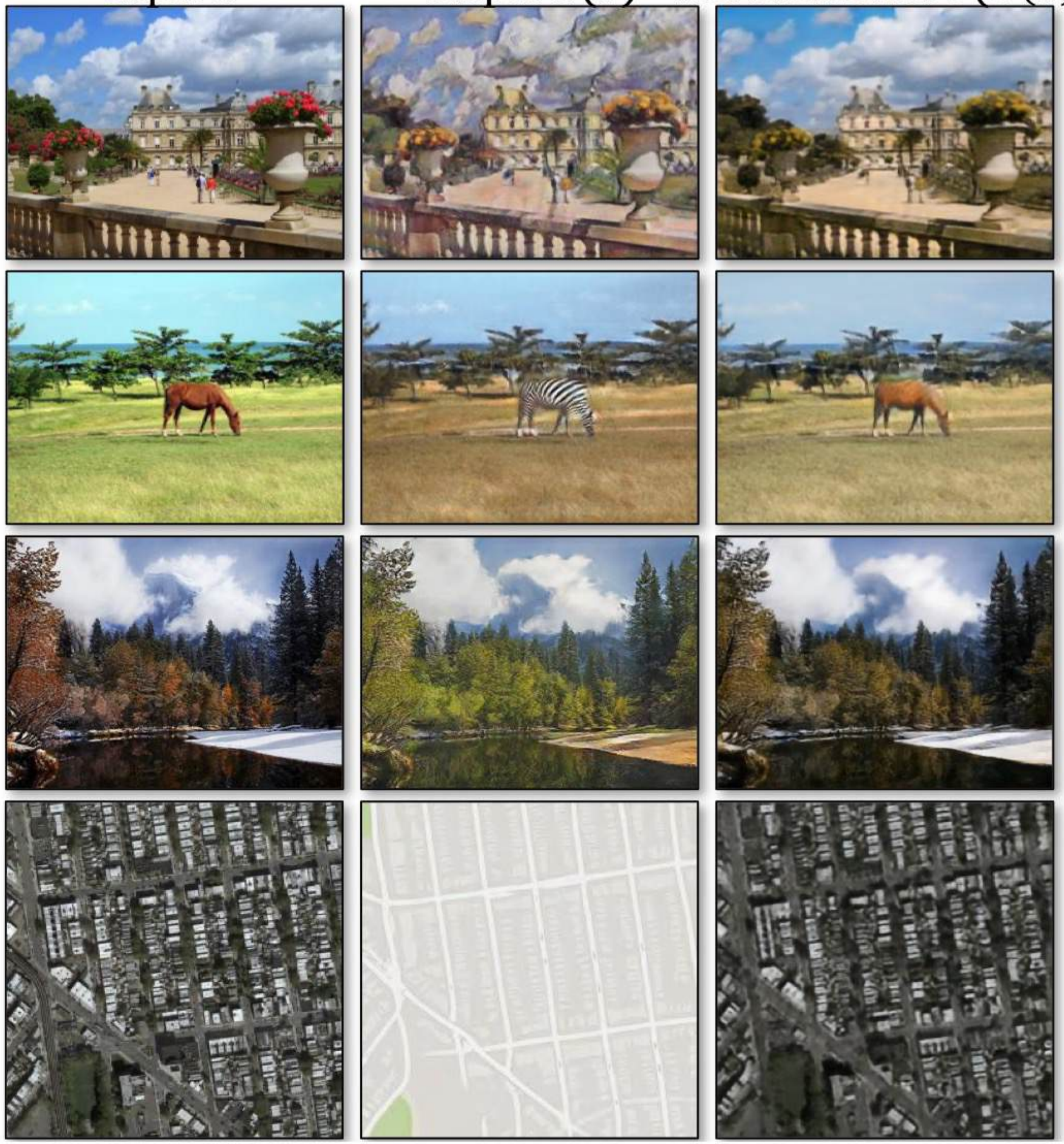
$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = &\ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = &\ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],\end{aligned}$$

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”, 2020

Input x Output $G(x)$ Reconstruction $F(G(x))$



BigGAN

Large batch size, Large model, and many techniques to stabilize the training of GAN…

Figure 6: Samples generated by our BigGAN model at 512×512 resolution.

Brock et al. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”, 2019