



Boulder

Generative Networks; Unconditional GANs



[YouTube Playlist](#)

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu



Boulder



Generative Adversarial Nets

[YouTube Playlist](#)

$p_z(z) \rightarrow$ prior on noise variable z

$G(z; \theta_g) \rightarrow$ generative model (differentiable function)

$p_g(x) \rightarrow$ generator's distribution over data x

$D(x; \theta_d) \rightarrow$ discriminative model

(probability that x came from the data rather than p_g)

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

$\log(1 - D(G(z))) \rightarrow$ saturates early in the learning

(D can reject generated samples with high confidence)

$$D(G(z)) \approx 0 \implies 1 - D(G(z)) \approx 1 \implies \log(1 - D(G(z))) \approx 0$$

$$\max_G \mathbb{E}_{z \sim p_z(z)} \log D(G(z))$$

Proposition: For fixed G , the optimal discriminator D is

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

Proof:

$$V(G, D) = \int_x [p_{\text{data}}(x) \log D(x) + p_g(x) \log(1 - D(x))] dx$$

$$y^* = \frac{a}{a+b} = \arg \max_y [a \log y + b \log(1 - y)]$$

$$C(G) := \max_D V(G, D)$$

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right]$$

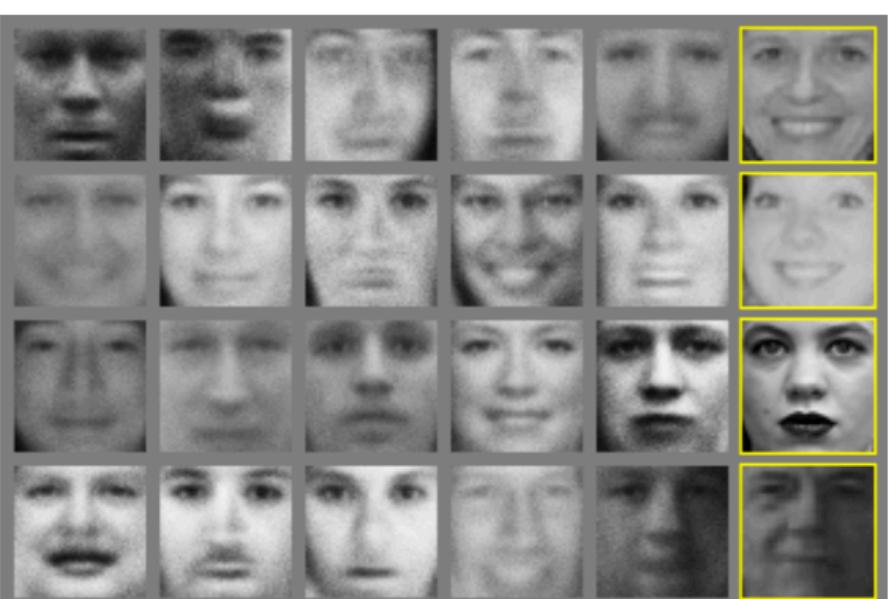
$$C(G) = -\log 4 + KL \left(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2} \right) + KL \left(p_g \parallel \frac{p_{\text{data}} + p_g}{2} \right)$$

$$C(G) = -\log 4 + 2 \underbrace{JSD(p_{\text{data}} \parallel p_g)}$$

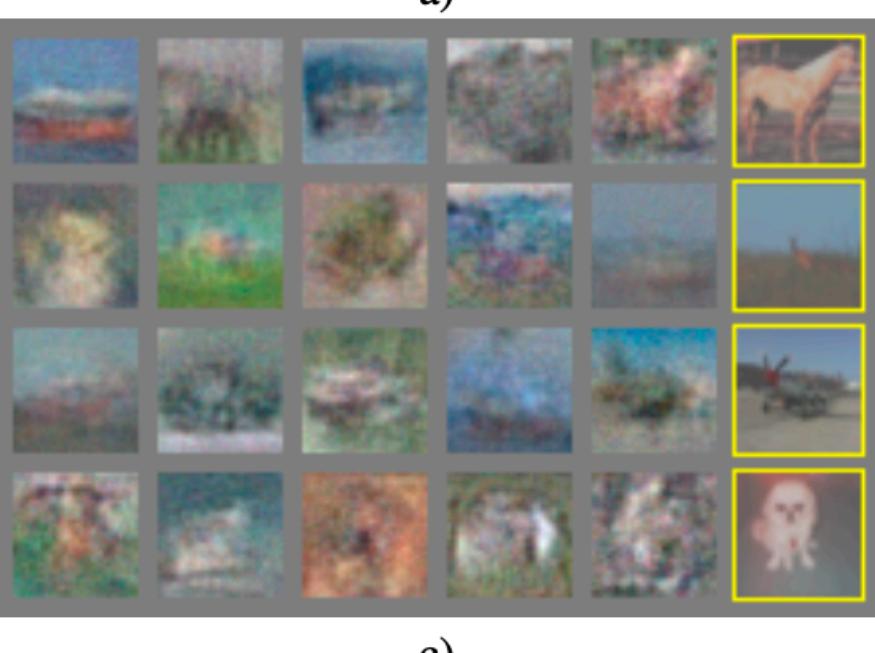
Jensen–Shannon divergence; ≥ 0 ; $= 0$ iff $p_{\text{data}} = p_g$



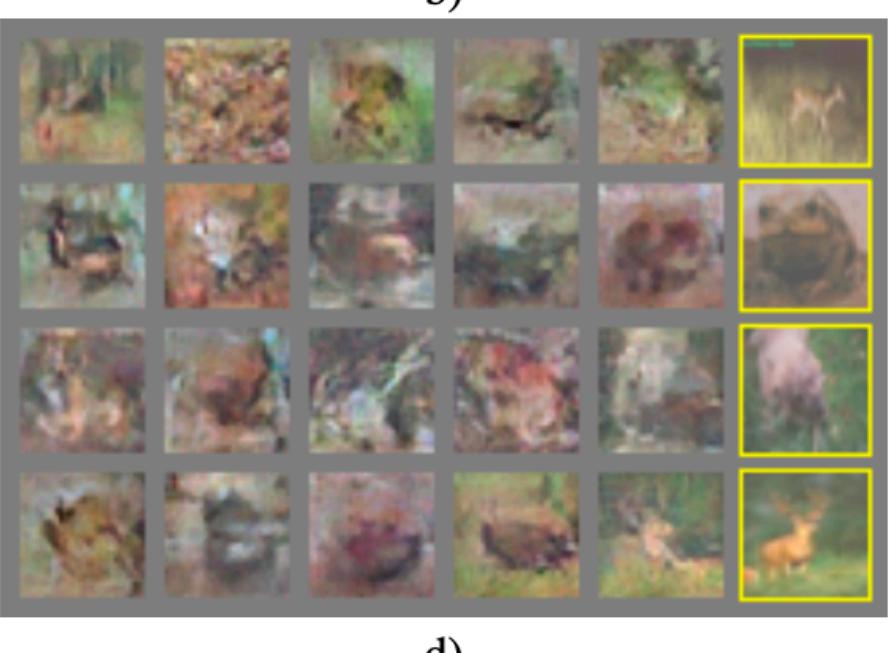
a)



b)



c)



d)

Unsupervised representation learning with deep convolutional generative adversarial networks

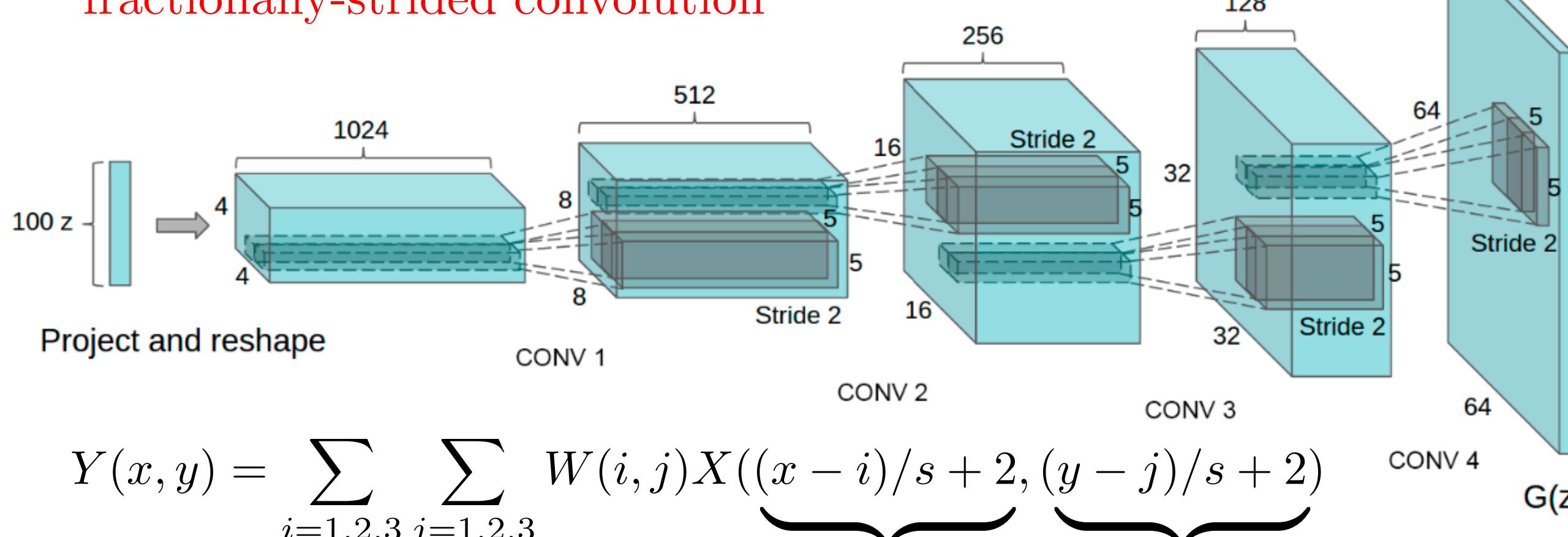

[YouTube Playlist](#)

DCGANs → deep convolutional generative adversarial networks

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

fractionally-strided convolution



Only integer-valued indices participate in the summation!

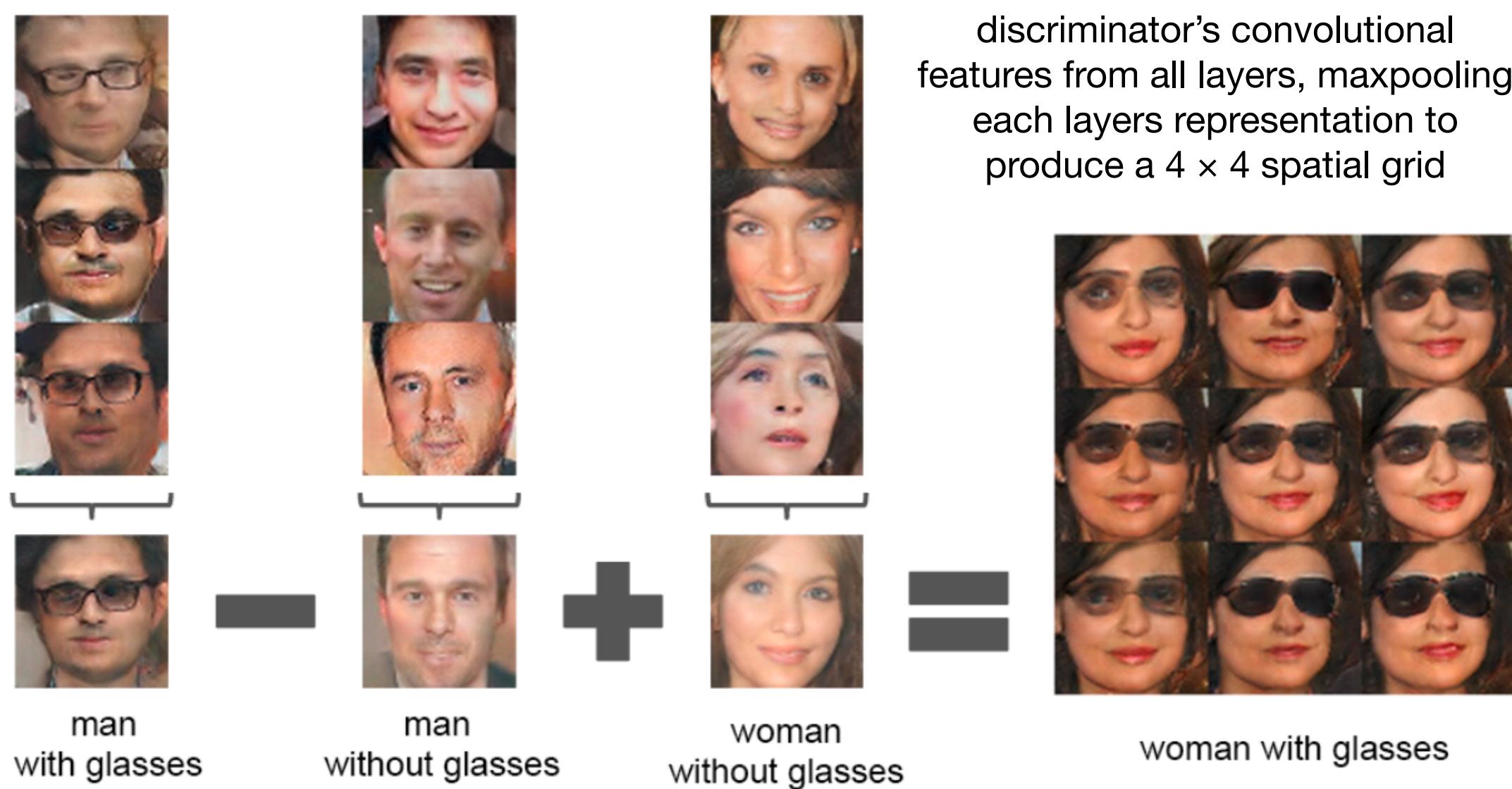
$$Y(x, y) = \sum_{i=1,2,3} \sum_{j=1,2,3} W(i, j) X(\underbrace{s(x - 2) + i}_{=: x'}, \underbrace{s(y - 2) + j}_{=: y'})$$

$$\implies x = (x' - i)/s + 2$$

$$\frac{\partial L}{\partial X(x', y')} = \sum_{i=1,2,3} \sum_{j=1,2,3} W(i, j)^T \frac{\partial L}{\partial Y((x' - i)/s + 2, (y' - j)/s + 2)}$$

DCGAN trained on Imagenet-1k. Features are used to classify CIFAR-10 images.

Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% ($\pm 0.7\%$)	4800
3 Layer K-means Learned RF	82.0%	70.7% ($\pm 0.7\%$)	3200
View Invariant K-means	81.9%	72.6% ($\pm 0.7\%$)	6400
Exemplar CNN	84.3%	77.4% ($\pm 0.2\%$)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% ($\pm 0.4\%$)	512





Boulder



[YouTube Playlist](#)

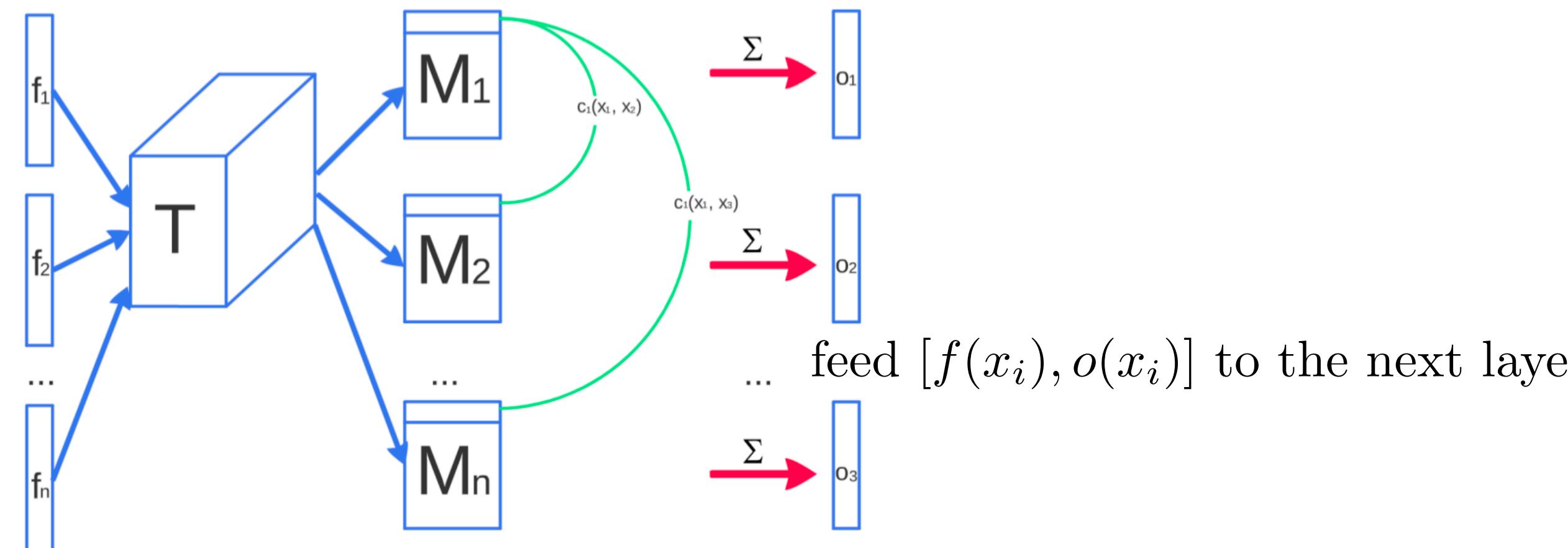
Improved Techniques for Training GANs

Feature Matching

$f(x) \rightarrow$ activations on an intermediate layer of the discriminator

$\|\mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{z \sim p_z(z)} f(G(z))\|_2^2 \rightarrow$ new objective for the generator

Minibatch discrimination (to deal with mode-collapse)



$$M_i = f(x_i)T, f(x_i) \in \mathbb{R}^A, F \in \mathbb{R}^{A \times B \times C}, M_i \in \mathbb{R}^{B \times C}$$

$$c_b(x_i, x_j) = \exp(-\|M_{i,b} - M_{j,b}\|_{L_1}) \in \mathbb{R}, M_{i,b} \in \mathbb{R}^C$$

$$o(x_i)_b = \sum_{j=1}^n c_b(x_i, x_j) \in \mathbb{R} \rightarrow \text{looks at multiple examples}$$

$$o(x_i) = [o(x_i)_1, \dots, o(x_i)_B] \in \mathbb{R}^B \rightarrow \text{output}$$

Historical averaging

$$\|\theta - \frac{1}{t} \sum_{i=1}^t \theta[i]\|_2^2$$

Virtual batch normalization

Each example x is normalized based on the statistics collected on a reference batch of examples and on x itself collected once and fixed at the start of training

Inception Score

$$\exp(\mathbb{E}_x KL(p(y|x) \| p(y)))$$

generated images inception model

$p(y) = \int p(y|x)p_g(x)dx$ should have high entropy (generate varied images)

$p(y|x)$ should have low entropy (generated images should contain meaningful objects)

Semi-supervised learning

$$x \mapsto (l_1, \dots, l_K) \rightarrow \text{logits}$$

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

$$L_{\text{supervised}} = -\mathbb{E}_{\mathbf{x}, y \sim p_{\text{data}}(\mathbf{x}, y)} \log p_{\text{model}}(y|\mathbf{x}, y < K+1)$$

$$L_{\text{unsupervised}} = -\{\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \underbrace{\log [1 - p_{\text{model}}(y = K+1|\mathbf{x})]}_{D(\mathbf{x})} + \mathbb{E}_{\mathbf{x} \sim G} \underbrace{\log [p_{\text{model}}(y = K+1|\mathbf{x})]}_{1 - D(\mathbf{x})}\},$$

$$D(\mathbf{x}) = \frac{Z(\mathbf{x})}{Z(\mathbf{x})+1}, \text{ where } Z(\mathbf{x}) = \sum_{k=1}^K \exp[l_k(\mathbf{x})]. \rightarrow \text{by setting } l_{K+1}(x) = 0, \forall x$$

InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets


[YouTube Playlist](#)

Disentangled Representations

- writing styles from digit shapes on MNIST
- pose from lighting of 3D rendered images
- background digits from the central digit on SVHN
- hair styles, presence/absence of eyeglasses and emotions on CelebA

Regular GAN

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

InfoGAN

$$\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$$

$z \rightarrow$ source of incompressible noise

$c = (c_1, \dots, c_L) \rightarrow$ latent code (semantic features of data)

In information theory, mutual information $I(X; Y)$ between X and Y , measures the “amount of information” learned from knowledge of random variable Y about the other random variable X .

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Computing $I(c; G(z, c))$ requires access to the posterior $P(c|x)$

$Q(c|x) \rightarrow$ auxiliary distribution to approximate $P(c|x)$

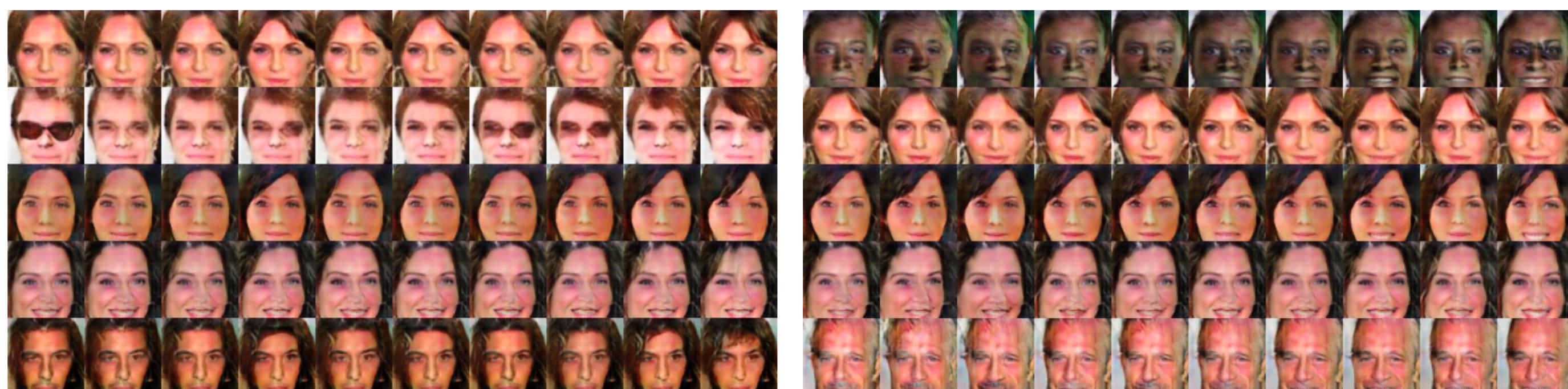
$$\begin{aligned} L_I(G, Q) &= E_{c \sim P(c), x \sim G(z, c)} [\log Q(c|x)] + H(c) \\ &\leq I(c; G(z, c)) \end{aligned}$$

$L_I(G, Q) \rightarrow$ Variational Lower Bound

$$\min_{G, Q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q)$$



(a) Rotation



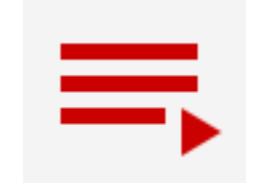
(b) Width

(c) Hair style

(d) Emotion



Boulder



[YouTube Playlist](#)

Least Squares Generative Adversarial Networks

Generative Adversarial Networks

$$\min_G \max_D V_{\text{GAN}}(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

$$C(G) := \max_D V_{\text{GAN}}(D, G)$$

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

$$C(G) = -\log 4 + 2 \underbrace{\text{JSD}(p_{\text{data}} || p_g)}_{\text{Jensen-Shannon divergence; } \geq 0; = 0 \text{ iff } p_{\text{data}} = p_g}$$

Jensen–Shannon divergence; ≥ 0 ; $= 0$ iff $p_{\text{data}} = p_g$

Least Squares Generative Adversarial Networks (LSGANs)

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [(D(\mathbf{x}) - b)^2] \quad a = -1, c = 0, b = 1 \\ \quad a = 0, b = c = 1 \\ \quad + \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})) - a)^2]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})) - c)^2]$$

$a \rightarrow$ label for fake data

$b \rightarrow$ label for real data

$c \rightarrow$ the value that G wants D to believe for fake data

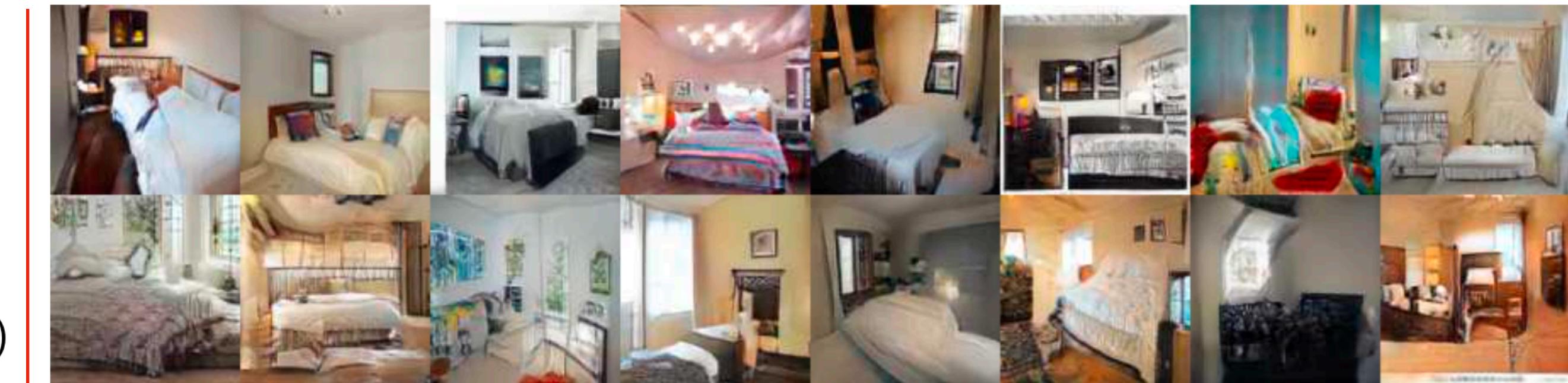
$$D^*(\mathbf{x}) = \frac{bp_{\text{data}}(\mathbf{x}) + ap_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \rightarrow \text{optimal discriminator } D \text{ for a fixed } G$$

$$C(G) := V_{\text{LSGAN}}(G; D^*)$$

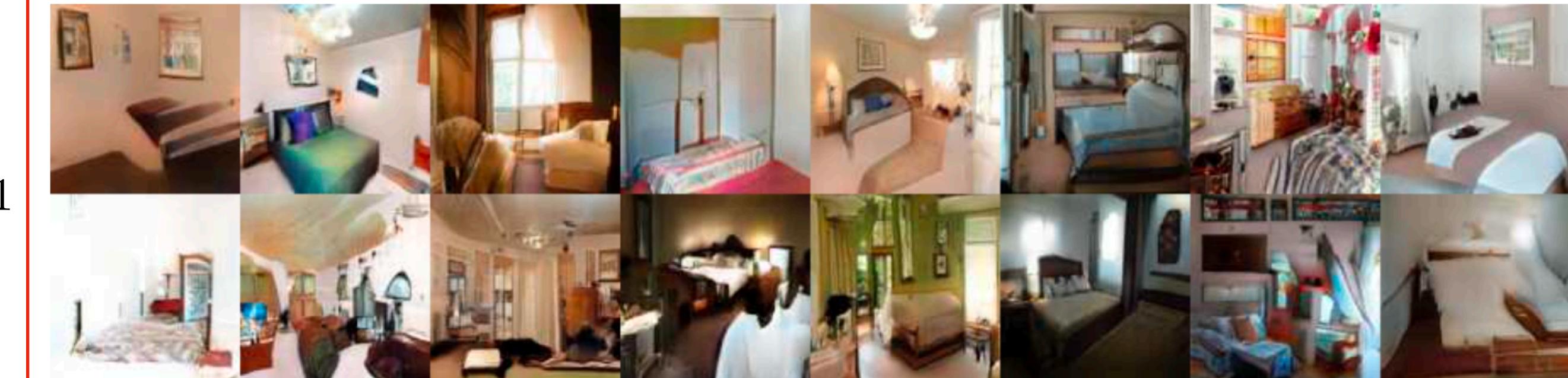
If we set $b - c = 1$ and $b - a = 2$, then

$$2C(G) = \int_{\mathcal{X}} \frac{(2p_g(\mathbf{x}) - (p_d(\mathbf{x}) + p_g(\mathbf{x})))^2}{p_d(\mathbf{x}) + p_g(\mathbf{x})} d\mathbf{x} = \chi_{\text{Pearson}}^2(p_d + p_g \| 2p_g),$$

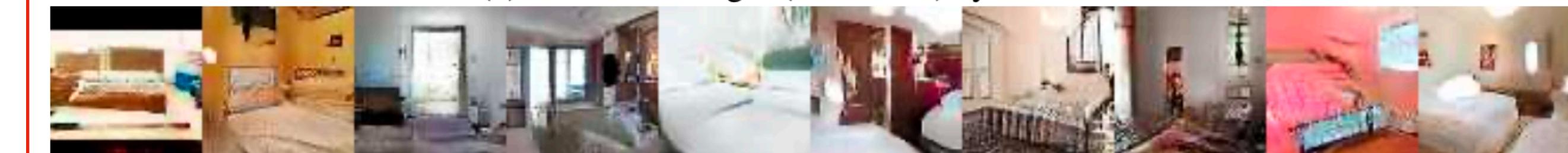
Pearson χ^2 divergence



(a) Generated images (112×112) by LSGANs.



(b) Generated images (112×112) by DCGANs.



(b) Generated images (64×64) by DCGANs (reported in [25]).
Generated images on LSUN-bedroom.

Method	Inception Score	LSGANs	Step 0	Step 5k	Step 15k	Step 25k	Step 40k	Target
DCGAN (reported in [10])	6.16							
DCGAN	6.22							
LSGAN (ours)	6.47							

Inception scores on CIFAR-10.



Boulder



Wasserstein GAN

[YouTube Playlist](#)

$(p_\theta)_{\theta \in \mathbb{R}^d} \rightarrow$ parametric family of densities

$\{x^i\}_{i=1}^m \rightarrow$ real data examples

$$\arg \max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^m \log p_\theta(x^i) = \arg \min_{\theta \in \mathbb{R}^d} KL(p_r \| p_\theta)$$

$p_r \rightarrow$ real data density

$Z \sim p(z)$

$g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$

$\rho(p_r, p_\theta) \rightarrow$ distance or divergence

$\mathcal{X} \rightarrow$ compact metric space (e.g., space of images $[0, 1]^d$)

$\sum \rightarrow$ set of all the Borel subsets of \mathcal{X}

$\delta(p_r, p_g) = \sup_{A \in \sum} |p_r(A) - p_g(A)| \rightarrow$ total variation (TV) distance

$KL(p_r \| p_g) = \int \log \left(\frac{p_r(x)}{p_g(x)} \right) p_r(x) d\mu(x) \rightarrow$ Kullback-Leibler (KL) divergence

$JS(p_r, p_g) = KL(p_r \| \frac{p_r + p_g}{2}) + KL(p_g \| \frac{p_r + p_g}{2}) \rightarrow$ Jensen-Shannon (JS) divergence

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)}[\|x - y\|] \rightarrow$$
 Earth-Mover (EM) distance or Wasserstein-1

$\Pi(p_r, p_g) \rightarrow$ set of all joint distributions $\gamma(x, y)$ whose marginals are respectively p_r and p_g

Kantorovich-Rubinstein duality

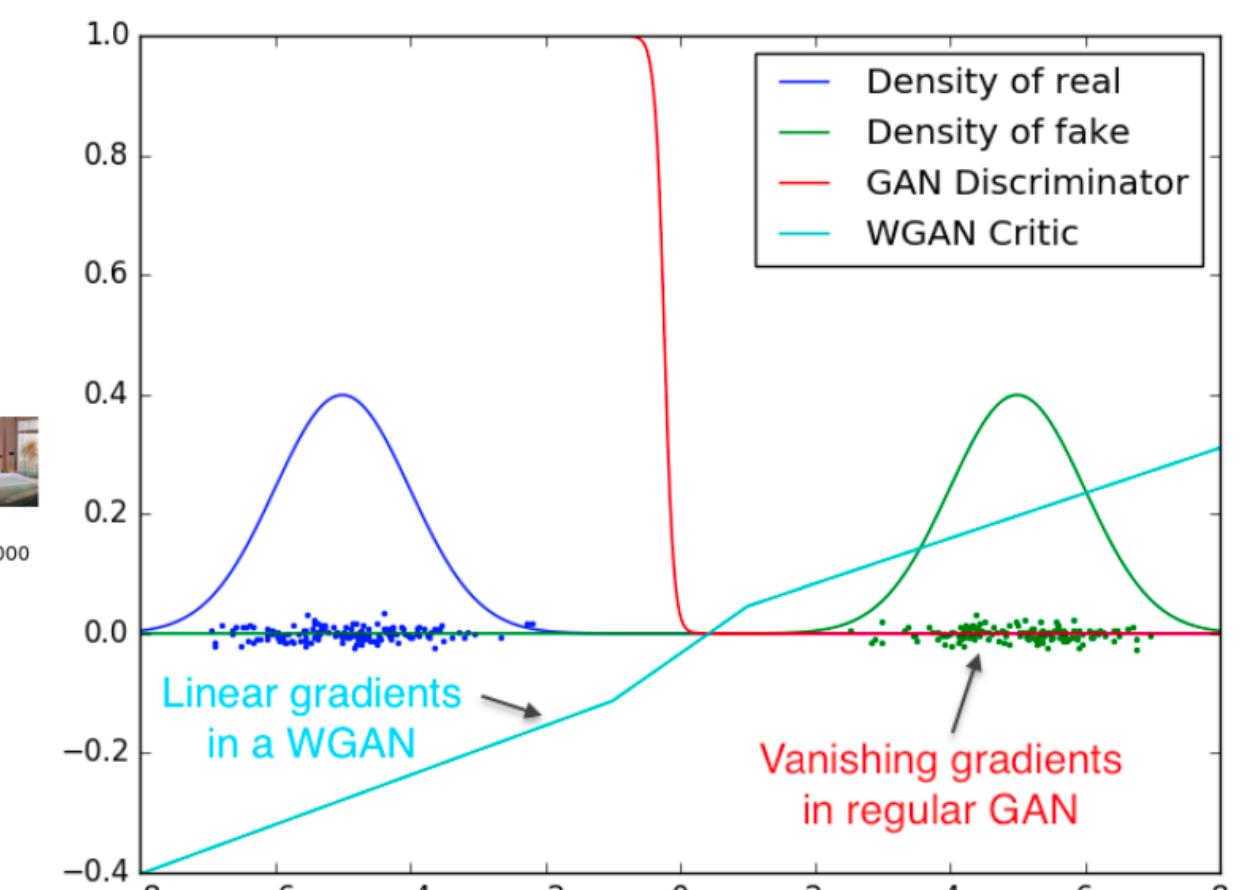
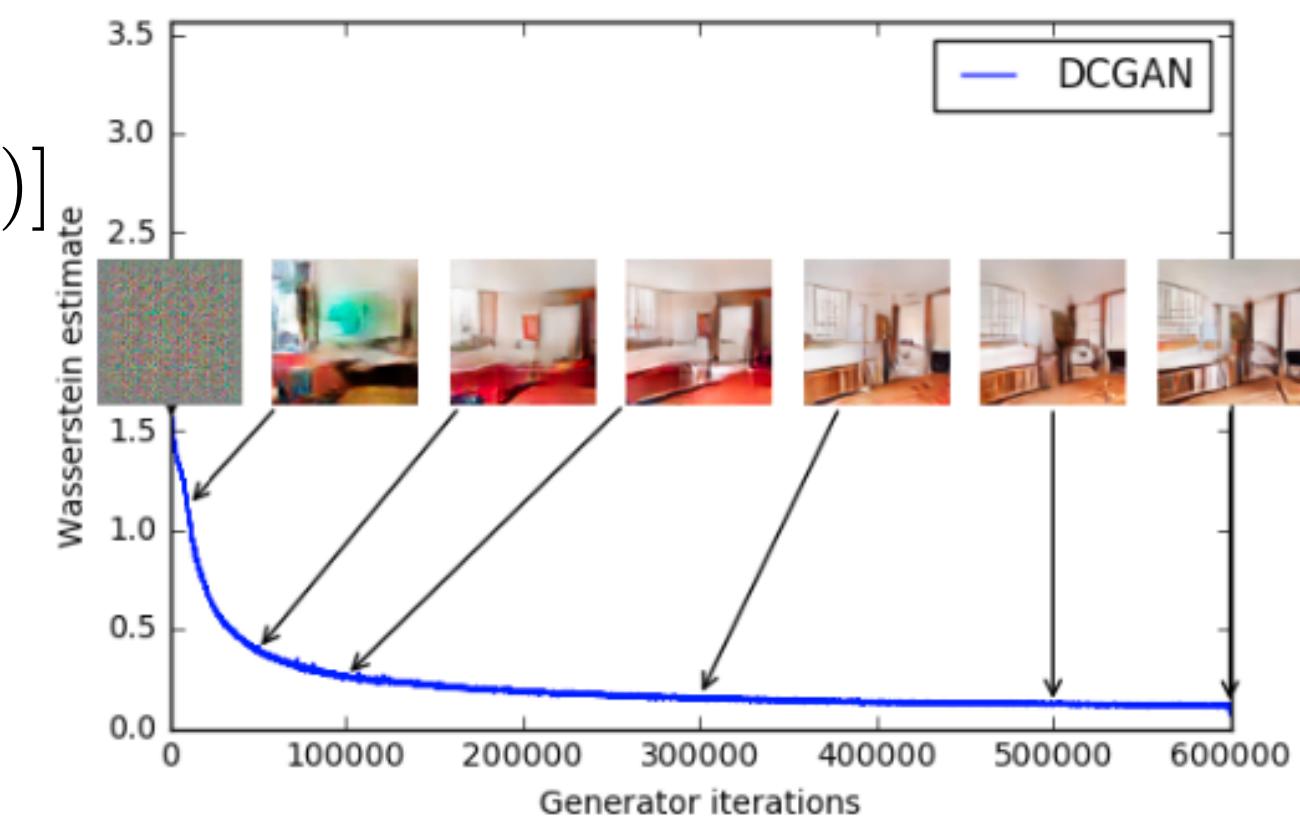
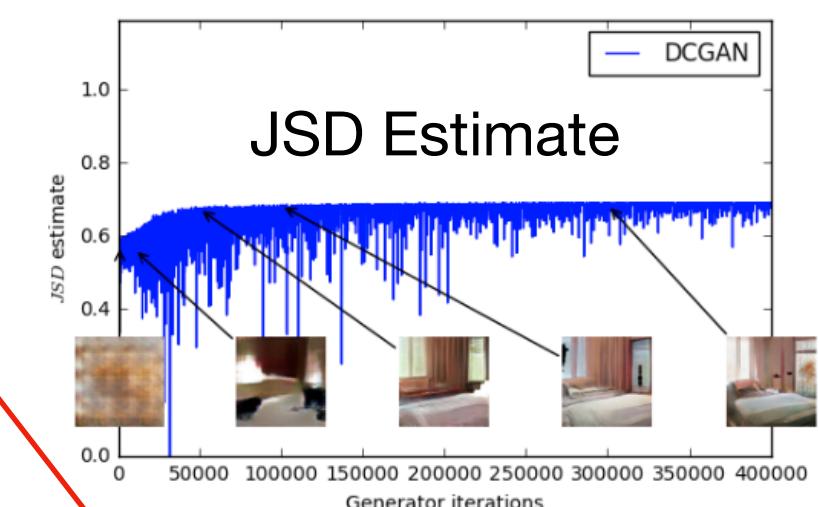
$$W(p_r, p_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)]$$

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p(z)}[f_w(g_\theta(z))]$$

clamp the weights to a fixed box

Integral Probability Metrics (IPMs)

$$d_{\mathcal{F}}(p_r, p_\theta) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_\theta}[f(x)]$$





Boulder



[YouTube Playlist](#)

Improved Training of Wasserstein GANs

GANs

$$\min_G \max_D \mathbb{E}_{x \sim p_r} [\log D(x)] + \mathbb{E}_{\tilde{x} \sim p_g} [\log(1 - D(\tilde{x}))]$$

$$\tilde{x} = G(z), z \sim p(z)$$

Jenson-Shannon Divergence between p_r & p_g

$$\max_G \mathbb{E}_{\tilde{x} \sim p_g} [\log D(\tilde{x})]$$

Wasserstein GANs

Earth-mover (Wasserstein-1) distance $W(q, p)$

minimum cost of transporting mass in order to transform the distribution q into the distribution p

cost is mass times transport distance

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim p_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim p_g} [D(\tilde{x})]$$

critic

$\mathcal{D} \rightarrow$ set of 1-Lipschitz functions

clip the weights of the critic

Gradient penalty

A differentiable function is 1-Lipschitz iff it has gradients with norm at most 1 everywhere

$$L = \underbrace{\mathbb{E}_{\tilde{x} \sim p_g} [D(\tilde{x})] - \mathbb{E}_{x \sim p_r} [D(x)]}_{\text{Original critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{x} \sim p_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{Gradient penalty}}$$

$\hat{x} \rightarrow$ sample uniformly along straight lines between x & \tilde{x}

Modeling discrete data with a continuous generator

$$G : z \mapsto \underbrace{\{v_1, \dots, v_{32}\}}_{\text{1D CNN}}$$

$v_i \in \mathbb{R}^n, n \rightarrow$ vocabulary size
one-hot character vectors

softmax output is passed directly into the critic
(i.e., no sampling step)

$$D : \underbrace{\{v_1, \dots, v_{32}\}}_{\text{1D CNN}} \mapsto d \in \mathbb{R}$$

decoding samples: take the arg max of each output vector

WGAN with gradient penalty (1D CNN)

Busino game camperate spent odea
In the bankaway of smarling the
SingersMay , who kill that imvic
Keray Pents of the same Reagun D
Manging include a tudancs shat "
His Zuith Dudget , the Denmber
In during the Uitational questio
Divos from The ' noth ronkies of
She like Monday , of macunsuer S

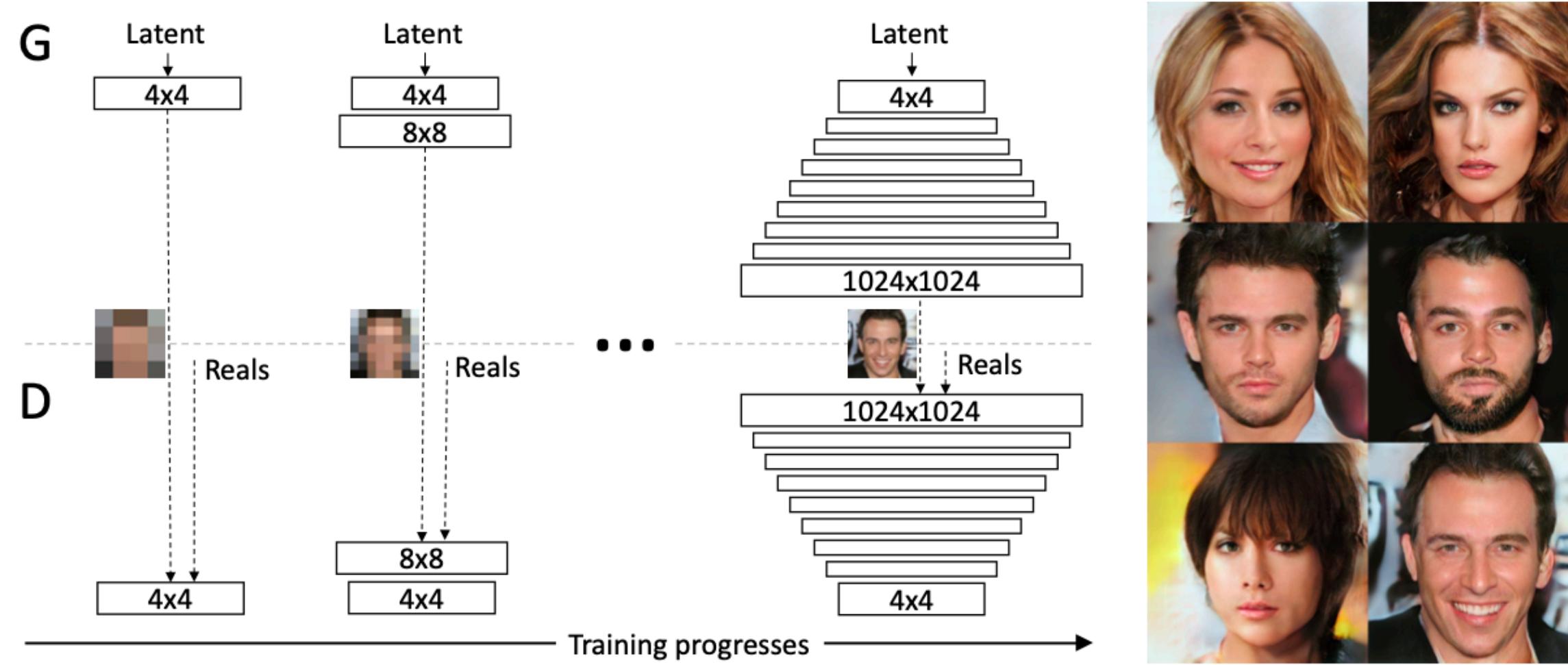
Solice Norkedin pring in since
This record (31.) UBS) and Ch
It was not the annuas were plogr
This will be us , the ect of DAN
These leaded as most-worsd p2 a0
The time I paid0a South Cubry i
Dour Fraps higs it was these del
This year out howneed allowed lo
Kaulna Seto consficates to repor

$$\Delta_n = \{v = (v^1, \dots, v^n) \in \mathbb{R}^n : v^j \geq 0, \sum_j v^j = 1\}$$

$$V_n = \{v = (v^1, \dots, v^n) \in \mathbb{R}^n : v^j \in \{0, 1\}, \sum_j v^j = 1\} \subset \Delta_n$$

$JS(p_r, p_g)$ saturates on $\Delta_n^{32} \supset V_n^{32}$ while $W(p_r, p_g)$ is well-defined

Progressive growing of GANs for improved quality, stability, and variation

[YouTube Playlist](#)


Increasing Variation Using Minibatch Standard Deviation

 $X \in \mathbb{R}^{N \times 512 \times 4 \times 4} \rightarrow \text{feature maps}$

$$M(f, x, y) = \frac{1}{N} \sum_{i=1}^N X(i, f, x, y) \rightarrow \text{mean}$$

$$S(f, x, y) = \sqrt{\frac{1}{N} \sum_{i=1}^N [X(i, f, x, y) - M(f, x, y)]^2} \rightarrow \text{standard deviation}$$

$$z = \frac{1}{512 \times 4 \times 4} \sum_{f,x,y} S(f, x, y) \implies Y(i, f, x, y) := \begin{cases} X(i, f, x, y) & \text{if } f \leq 512; \\ z & \text{if } f = 513. \end{cases}$$

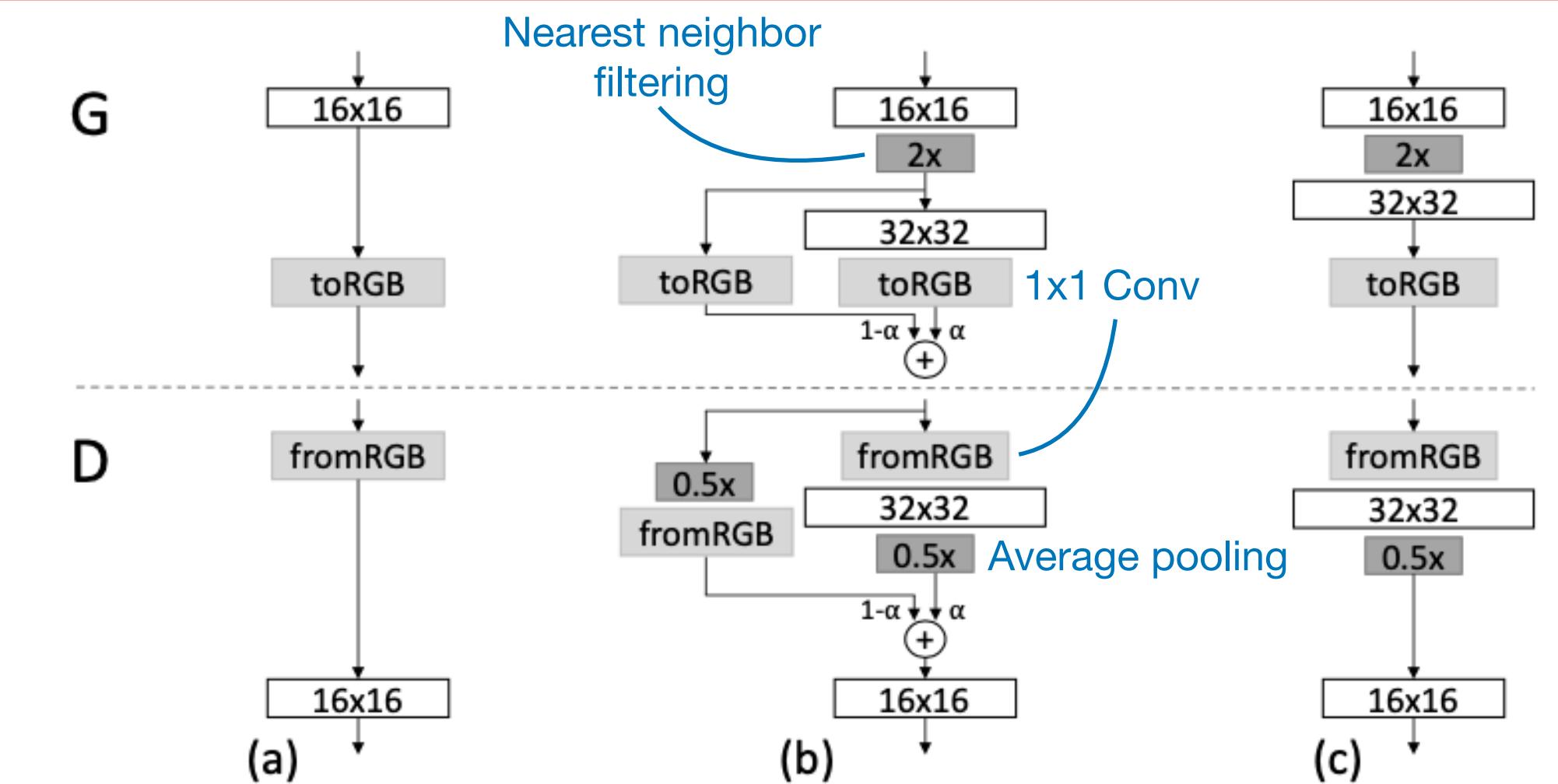
Equalized Learning Rate

$w_i \sim \mathcal{N}(0, 1)$

$\hat{w}_i = w_i / c$

$c = \sqrt{2/n} \rightarrow \text{per layer normalization constant}$

$n = k^2 d, k \rightarrow \text{filter size}, d \rightarrow \text{channels}$



Pixelwise Feature Vector Normalization In Generator

$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (a_{x,y}^j)^2 + \epsilon}, \text{ where } \epsilon = 10^{-8}$

a variant of “local response normalization”



Mao et al. (2016b) (128 × 128)

Gulrajani et al. (2017) (128 × 128)

(256 × 256)



GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium



[YouTube Playlist](#)

TTUR: Two Time-scale Update Rule

$D(., w)$ → discriminator with parameter vector w

$G(., \theta)$ → generator with parameter vector θ

$\tilde{g}(\theta, w)$ → stochastic gradient of the discriminator's loss \mathcal{L}_D

$\tilde{h}(\theta, w)$ → stochastic gradient of the generator's loss \mathcal{L}_G

mini-batches of m real world samples x^i
and m synthetic samples z^i

$$\begin{cases} g(\theta, w) = \nabla_w \mathcal{L}_D \\ h(\theta, w) = \nabla_\theta \mathcal{L}_G \end{cases} \rightarrow \text{true gradients}$$

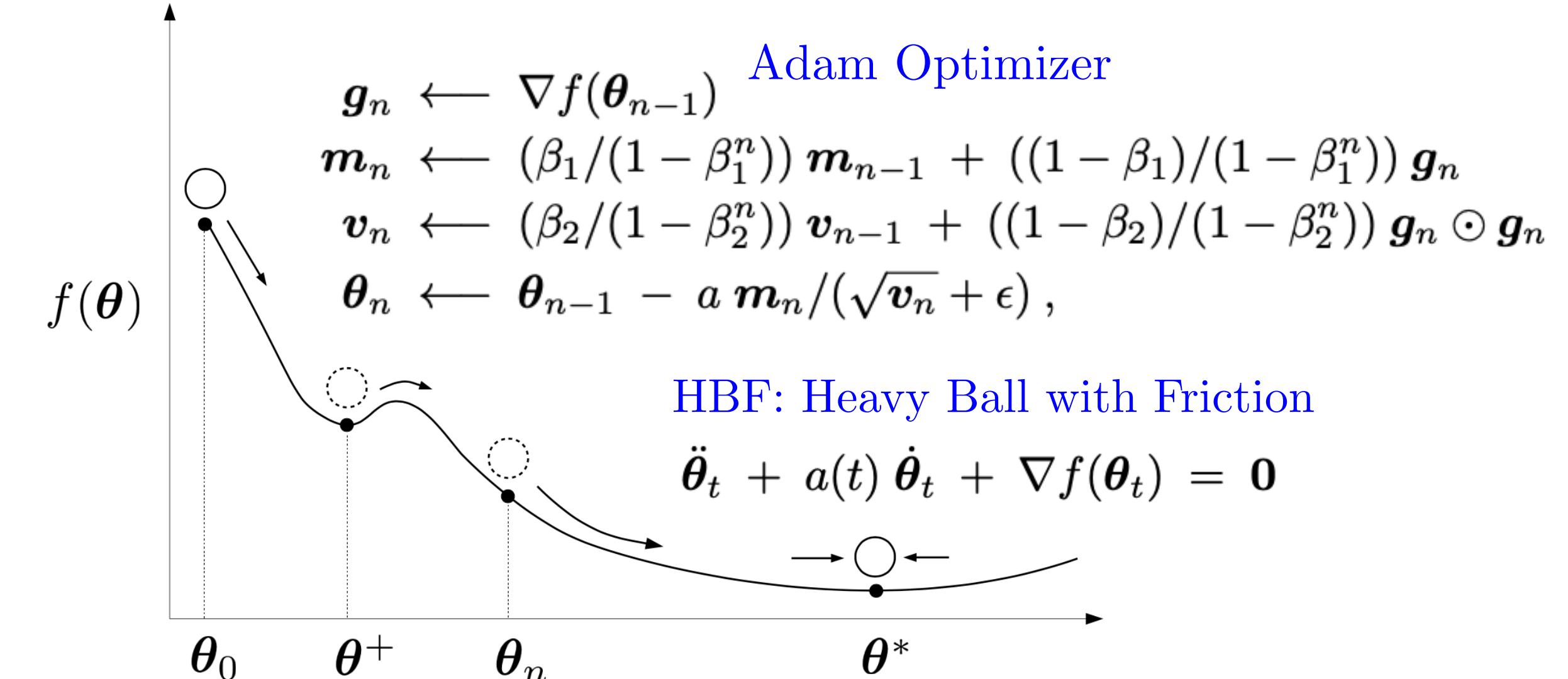
$$\begin{cases} \tilde{g}(\theta, w) = g(\theta, w) + M_w \\ \tilde{h}(\theta, w) = h(\theta, w) + M_\theta \end{cases} \rightarrow \text{random variables } M_w \text{ & } M_\theta$$

$b(n)$ → learning rate for discriminator update (fast)

$a(n)$ → learning rate for generator update (slow)

The proof relies on the fact that there eventually is a time point
when the perturbation of the slow update rule is small enough
to allow the fast update rule to converge

Apply the idea of perturbed ODEs ($\dot{\theta} = \nabla f(\theta)$)



Fréchet Inception Distance

$p_r(.$) → real world data

$p_g(.$) → generating model data

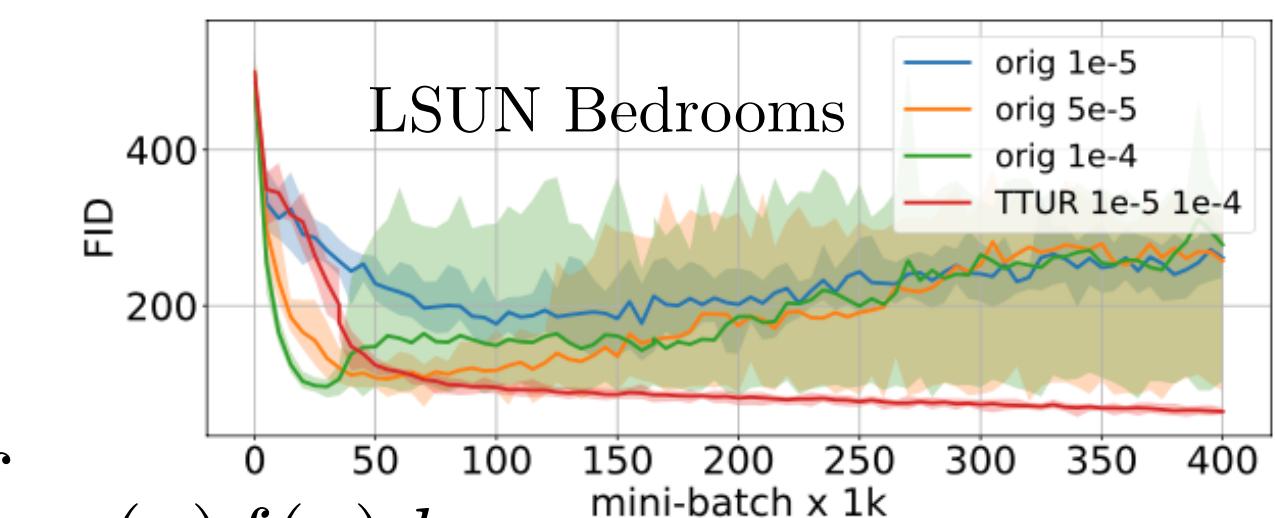
$$p_r(.) = p_g(.) \text{ iff } \int p_r(x) f(x) dx = \int p_g(x) f(x) dx$$

$f(x)$ → basis spanning the function space in which $p_r(.$) & $p_g(.$) live

$f(x)$ → polynomials, first & second moments, Gaussian

x → coding layer of an Inception model

$$d^2((m_g, C_g), (m_r, C_r)) = \|m_g - m_r\|_2^2 + \text{Tr}(C_g + C_r - 2(C_g C_r)^{1/2})$$





Boulder



Spectral Normalization for Generative Adversarial Networks

[YouTube Playlist](#)

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

$$V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{x' \sim p_g} [\log(1 - D(x'))]$$

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

$$D_G^*(x) = \text{sigmoid}(f^*(x))$$

$$f^*(x) = \log p_{\text{data}}(x) - \log p_g(x)$$

$$\nabla_x f^*(x) = \frac{1}{p_{\text{data}}(x)} \nabla_x p_{\text{data}}(x) - \frac{1}{p_g(x)} \nabla_x p_g(x)$$

can be unbounded or even incomputable

$$\arg \max_{\|f\|_{\text{Lip}} \leq K} V(G, D),$$

Spectral Normalization

$g : \mathbf{h}_{in} \mapsto \mathbf{h}_{out}$ → each layer

$$\|g\|_{\text{Lip}} = \sup_h \underbrace{\sigma(\nabla g(h))}_{\text{spectral norm}}$$

$$\sigma(A) := \max_{\mathbf{h}: \mathbf{h} \neq 0} \frac{\|A\mathbf{h}\|_2}{\|\mathbf{h}\|_2} = \max_{\|\mathbf{h}\|_2 \leq 1} \|A\mathbf{h}\|_2,$$

$$f(\mathbf{x}, \theta) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(\dots a_1(W^1 \mathbf{x}) \dots)))),$$

$$\|f\|_{\text{Lip}} \leq \|(\mathbf{h}_L \mapsto W^{L+1} \mathbf{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\mathbf{h}_{L-1} \mapsto W^L \mathbf{h}_{L-1})\|_{\text{Lip}}$$

$$\dots \|a_1\|_{\text{Lip}} \cdot \|(\mathbf{h}_0 \mapsto W^1 \mathbf{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\mathbf{h}_{l-1} \mapsto W^l \mathbf{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l).$$

$$\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$$

$\|a_l\|_{\text{Lip}}$ is equal to 1 → ReLU & Leaky ReLU

$$\bar{W}_{\text{SN}}(W) := W/\sigma(W).$$

Power Iteration Method

$$\tilde{\mathbf{v}} \leftarrow W^T \tilde{\mathbf{u}} / \|W^T \tilde{\mathbf{u}}\|_2, \quad \tilde{\mathbf{u}} \leftarrow W \tilde{\mathbf{v}} / \|W \tilde{\mathbf{v}}\|_2.$$

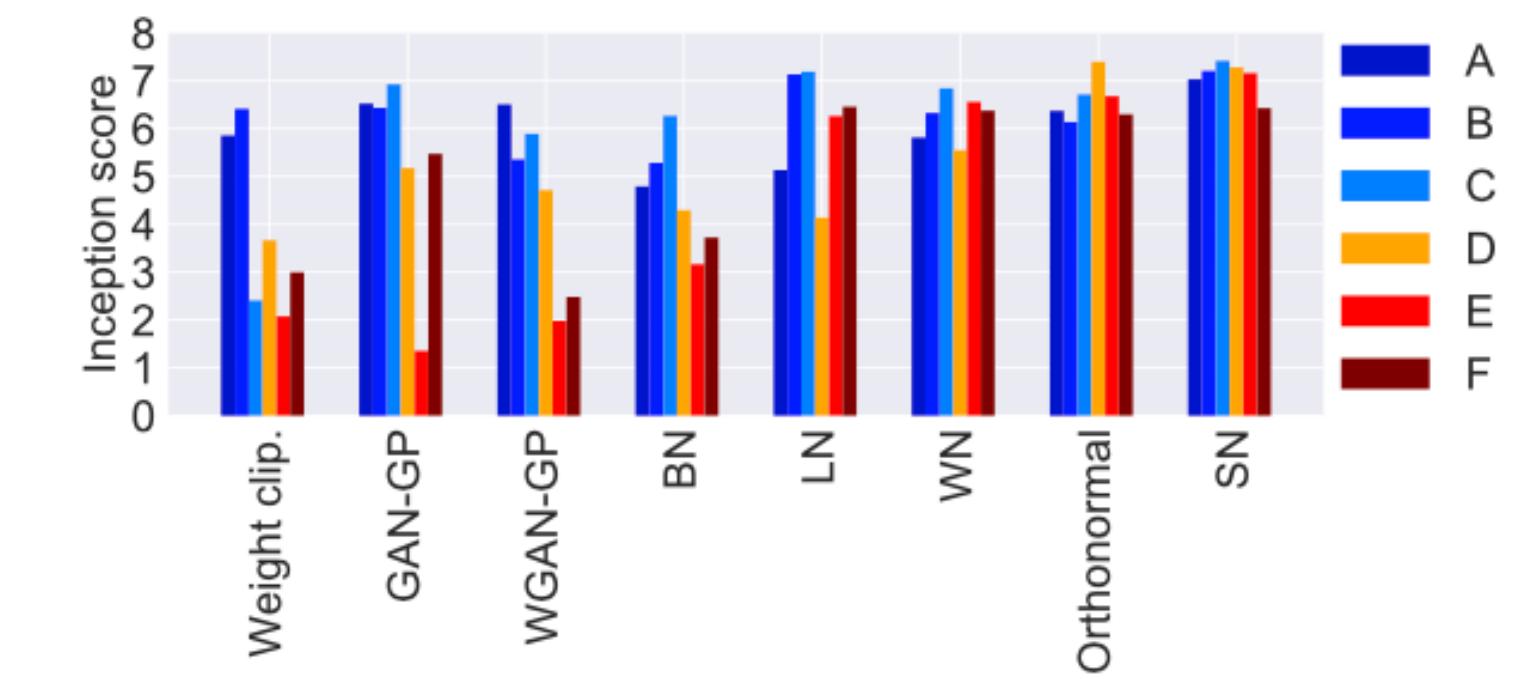
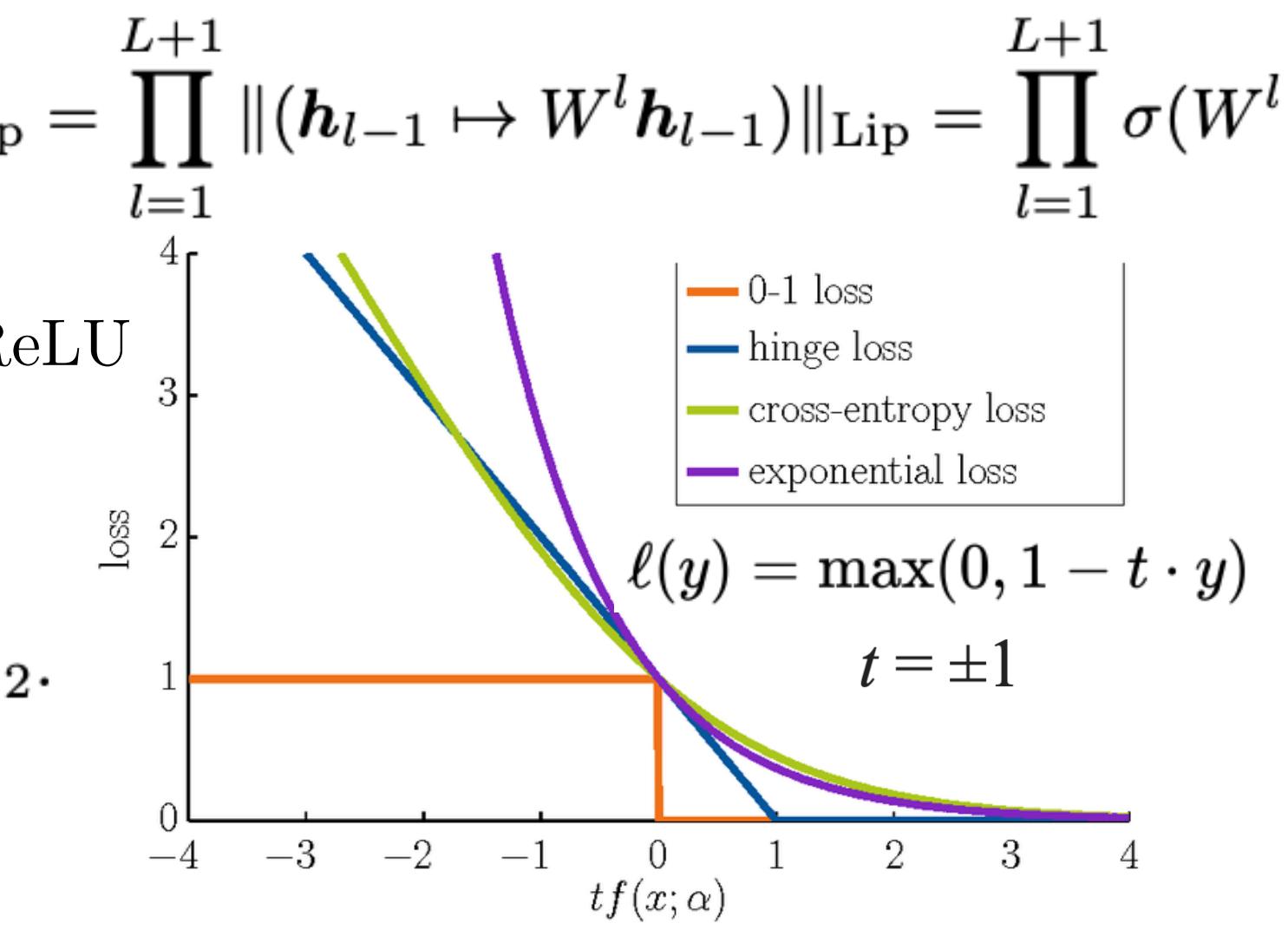
$$\sigma(W) \approx \tilde{\mathbf{u}}^T W \tilde{\mathbf{v}}.$$

Hinge Loss

$$V_D(\hat{G}, D) = \mathbb{E}_{\mathbf{x} \sim q_{\text{data}}(\mathbf{x})} [\min(0, -1 + D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\min(0, -1 - D(\hat{G}(\mathbf{z})))]$$

$$V_G(G, \hat{D}) = - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\hat{D}(G(\mathbf{z}))],$$

Setting	α	β_1	β_2	n_{dis}
A [†]	0.0001	0.5	0.9	5
B [†]	0.0001	0.5	0.999	1
C [*]	0.0002	0.5	0.999	1
D	0.001	0.5	0.9	5
E	0.001	0.5	0.999	5
F	0.001	0.9	0.999	5





Boulder

Large Scale GAN Training for High Fidelity Natural Image Synthesis



[YouTube Playlist](#)

Background

$G \rightarrow$ generator network

$D \rightarrow$ discriminator network

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

$z \in \mathbb{R}^{d_z} \rightarrow$ latent variable

$z \sim p(z) = \mathcal{N}(0, I)$ or $\mathcal{U}[-1, 1]$

BigGANs

- Larger Models
- Larger Batches

1) SA-GAN (Self-Attention Generative Adversarial Networks) architecture

2) Hinge Loss

3) Provide class information to G with class-conditional BatchNorm

4) Provide class information to D with projection

5) Spectral Normalization in G

6) For evaluation, employ moving averages of G's weights

7) Orthogonal Initialization

Truncation Trick

1) Train using $z \sim \mathcal{N}(0, I)$

2) Sample using truncated normal distribution

Improvement in individual sample quality at the cost of reduction in overall sample variety!

Orthogonal Regularization

$$R_\beta(W) = \beta \|W^T W \odot (1 - I)\|_F^2 \quad \beta = 10^{-4}$$



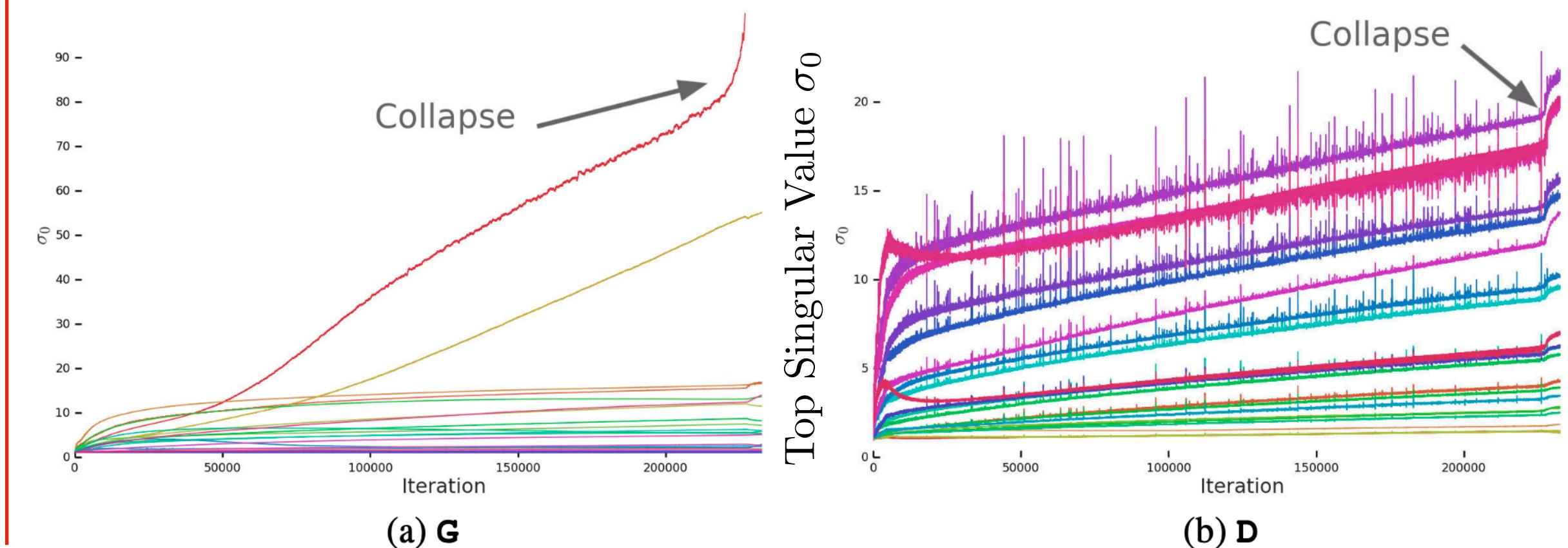
Saturation artifacts from applying truncation to a poorly conditioned model!



The effects of increasing truncation. From left to right, the threshold is set to 2, 1, 0.5, 0.04.

Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5			SA-GAN Baseline	1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77(± 1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03(± 1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85(± 3.83)
2048	96	173.5	✗	✗	✗	295(± 18)	9.54(± 0.62)	92.98(± 4.27)
2048	96	160.6	✓	✗	✗	185(± 11)	9.18(± 0.13)	94.94(± 1.32)
2048	96	158.3	✓	✓	✗	152(± 7)	8.73(± 0.45)	98.76(± 2.84)
2048	96	158.3	✓	✓	✓	165(± 13)	8.51(± 0.32)	99.31(± 2.10)
2048	64	71.3	✓	✓	✓	371(± 7)	10.48(± 0.10)	86.90(± 0.61)

Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better)



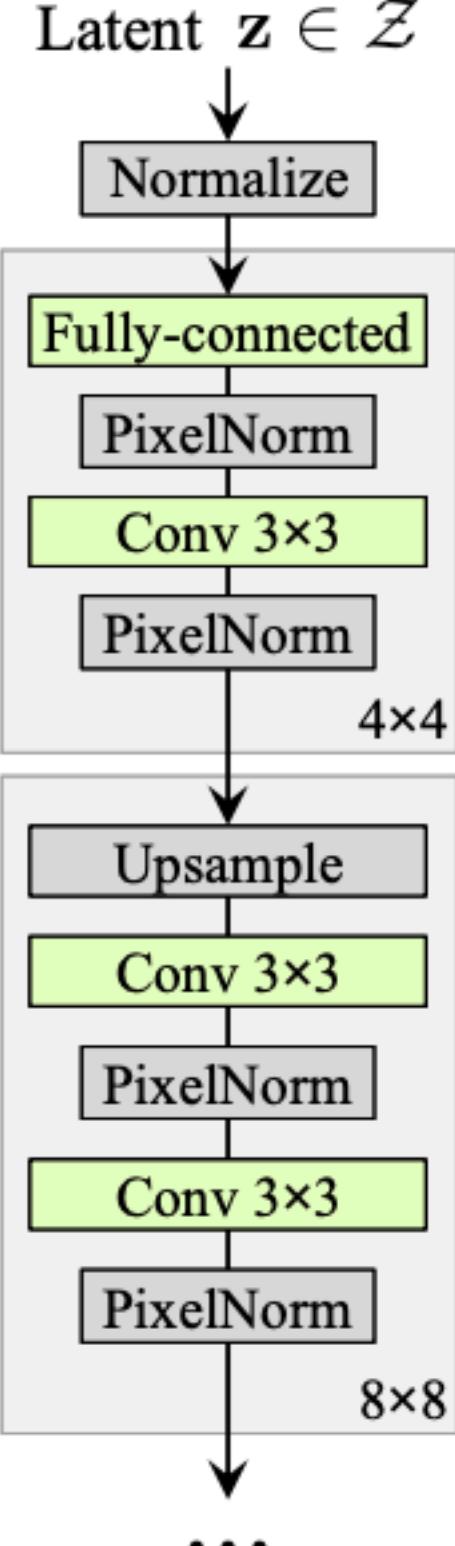


Boulder

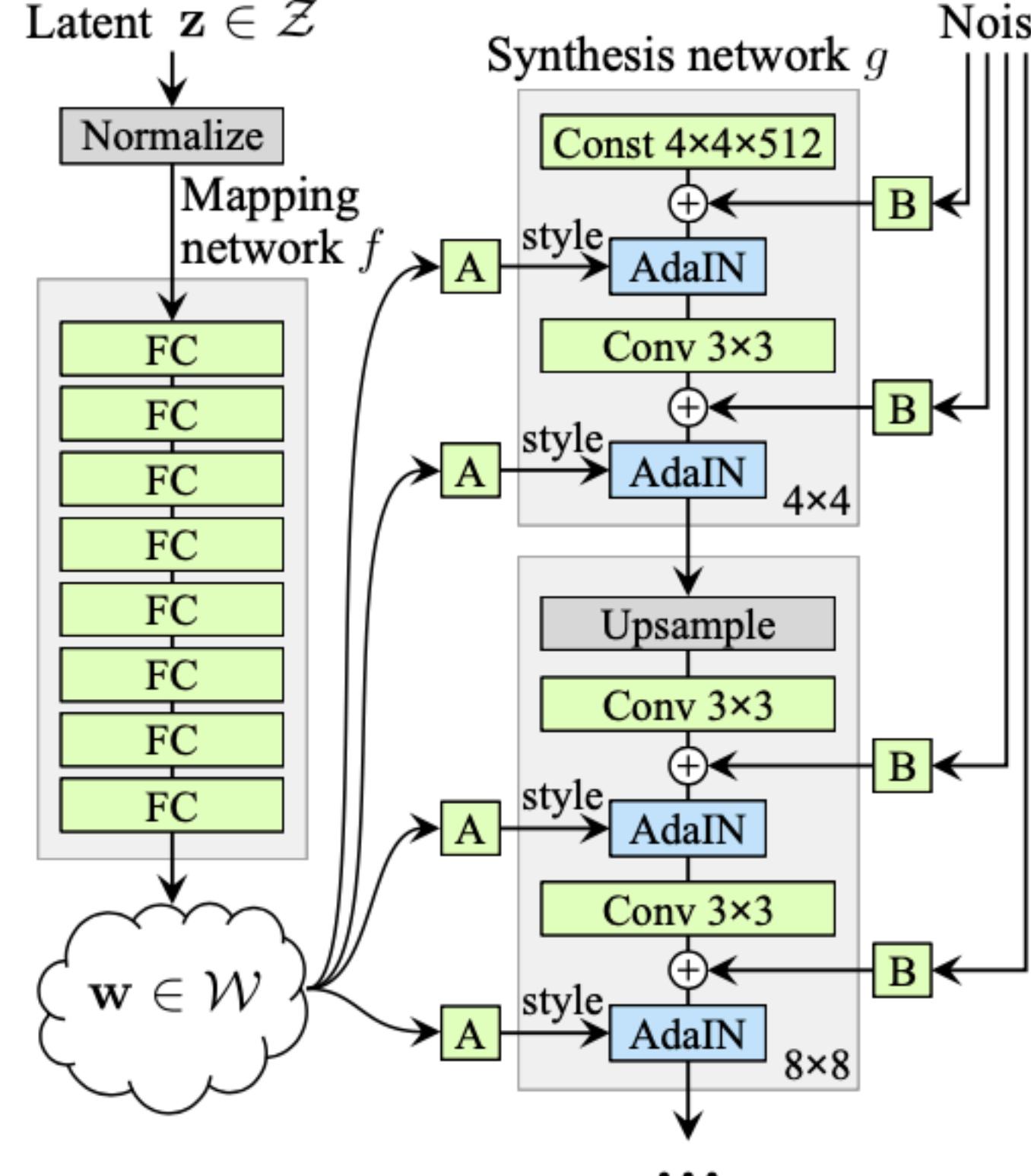
A Style-Based Generator Architecture for Generative Adversarial Networks



[YouTube Playlist](#)



(a) Traditional



(b) Style-based generator

mixing regularization

$$z_1 \mapsto w_1$$

$$z_2 \mapsto w_2$$

at a randomly selected point in the synthesis network switch from w_1 to w_2

Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019.

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

$X \in \mathbb{R}^{C \times H \times W}$ adaptive instance normalization

$$\mathbf{x}_i = X(i) \in \mathbb{R}^{H \times W}$$

$$\mu(\mathbf{x}_i) = \frac{1}{HW} \sum_{h,w} X(i, h, w) \in \mathbb{R}$$

$$\sigma(\mathbf{x}_i) = \sqrt{\frac{1}{HW} \sum_{h,w} (X(i, h, w) - \mu(\mathbf{x}_i))^2} \in \mathbb{R}$$

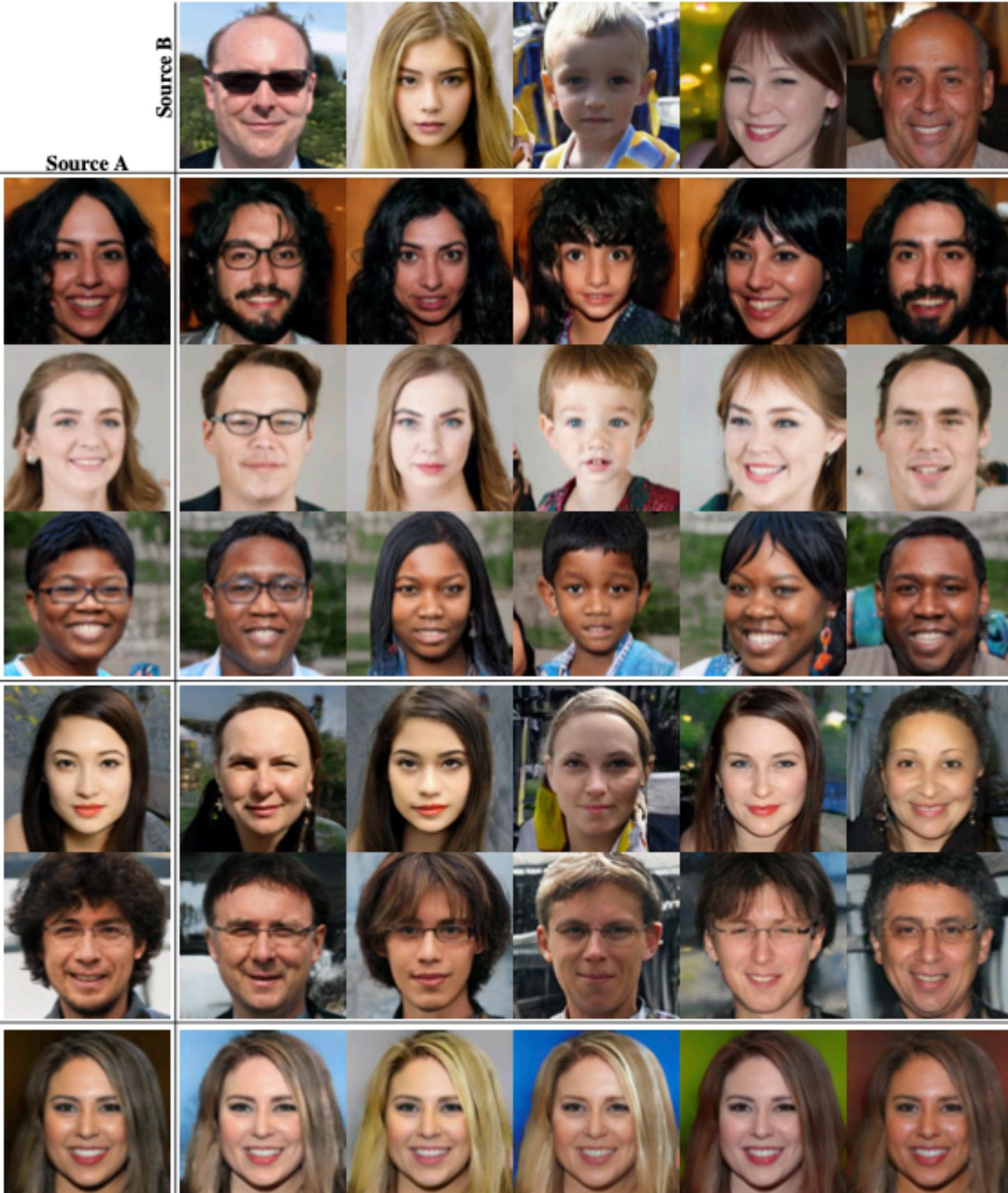
$$\mathbf{w} \xrightarrow[\text{affine}]{} (\mathbf{y}_s, \mathbf{y}_b) \in \mathbb{R}^{2C}$$

$$Z \in \mathbb{R}^{H \times W} \rightarrow \text{Gaussian Noise}$$

$$Y(i, h, w) = b_i Z(h, w)$$

learned per-feature scaling factors

This generator admits a more linear, less entangled representation of different factors of variation



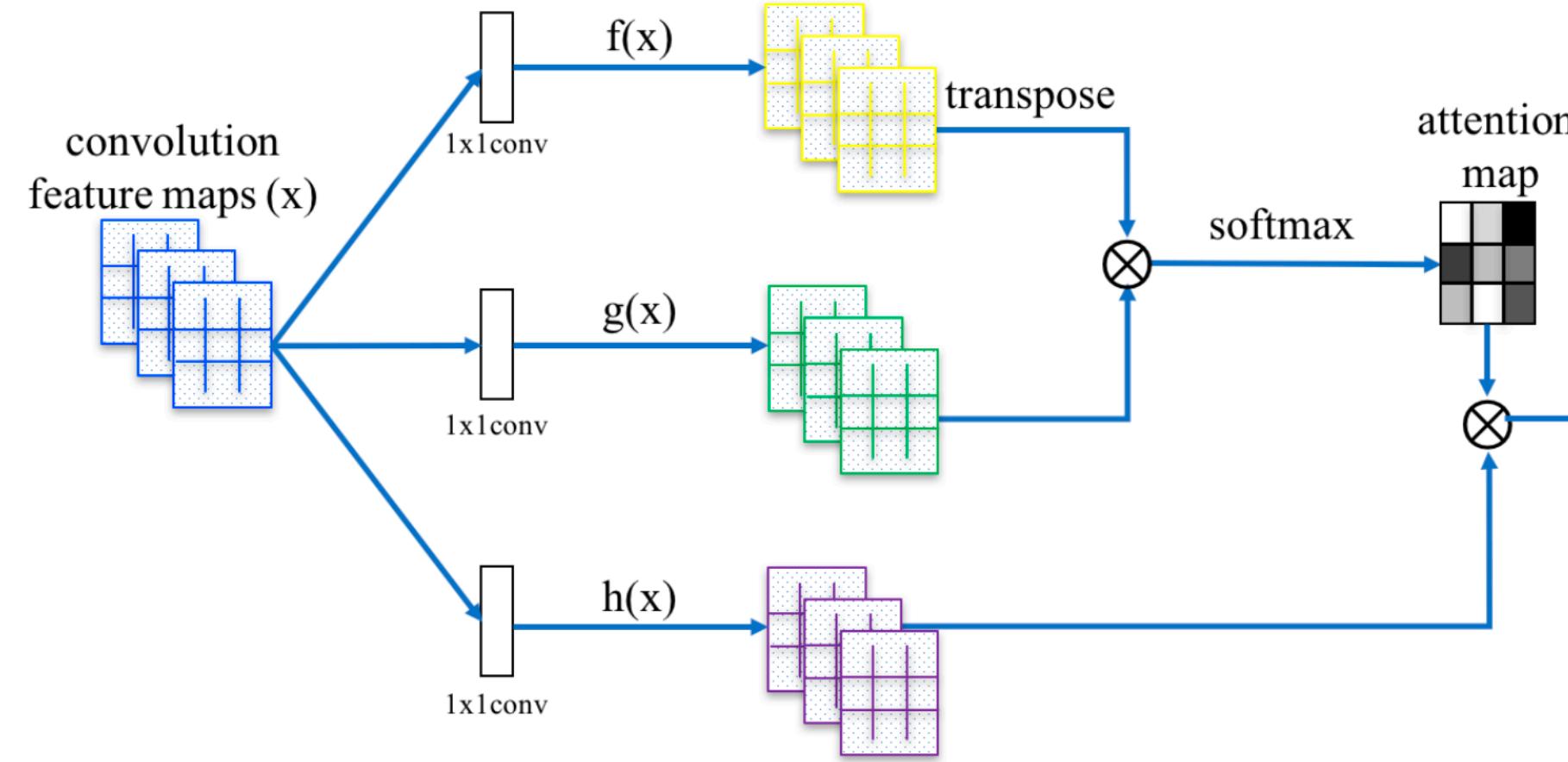


Boulder



Self-Attention Generative Adversarial Networks

[YouTube Playlist](#)



$x \in \mathbb{R}^{C \times N}$, $C \rightarrow \#$ channel, $N \rightarrow \#$ feature locations

$$f = f(x) = W_f x \in \mathbb{R}^{\bar{C}}, W_f \in \mathbb{R}^{\bar{C} \times C}$$

$$\bar{C} = C/8$$

$$g = g(x) = W_g x \in \mathbb{R}^{\bar{C}}, W_g \in \mathbb{R}^{\bar{C} \times C}$$

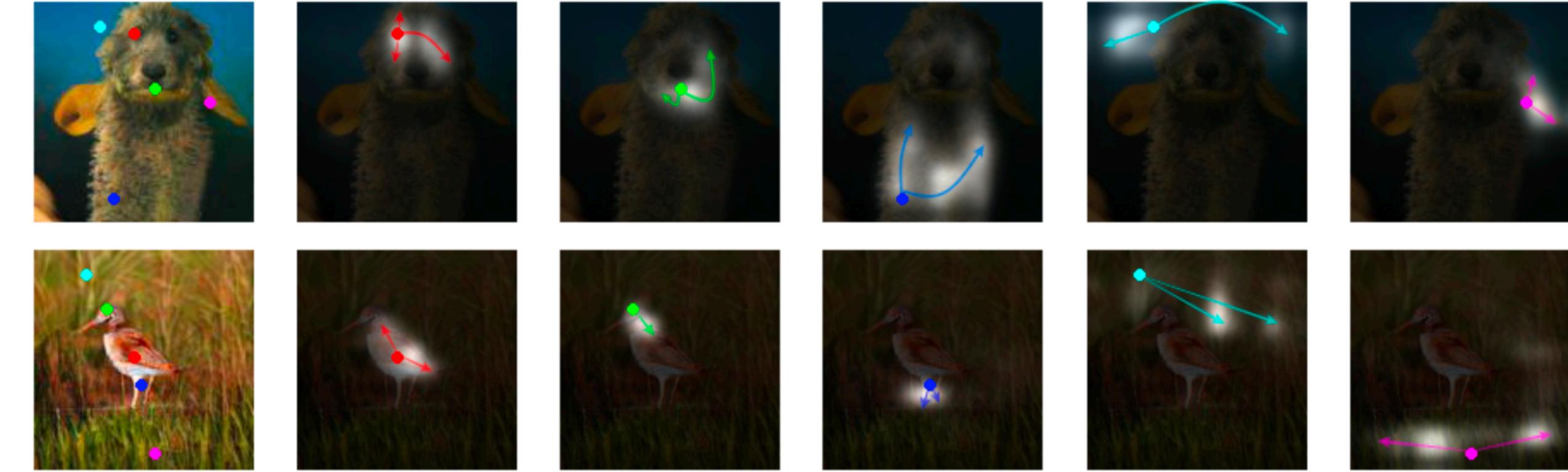
$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^N \exp(s_{ij})} \quad s_{i,j} = f(x_i)^T g(x_j)$$

the extent to which the model attends to the i -th location when synthesizing the j -th region

$$o = (o_1, \dots, o_j, \dots, o_N) \in \mathbb{R}^{C \times N}$$

$$o_j = v \left(\sum_{i=1}^N \beta_{j,i} h(x_i) \right), W_h \in \mathbb{R}^{\bar{C} \times C}, h(x) = W_h x$$

$$v(x) = W_v x, W_v \in \mathbb{R}^{C \times \bar{C}} \quad y_i = \gamma o_i + x_i$$



hinge version of the adversarial loss

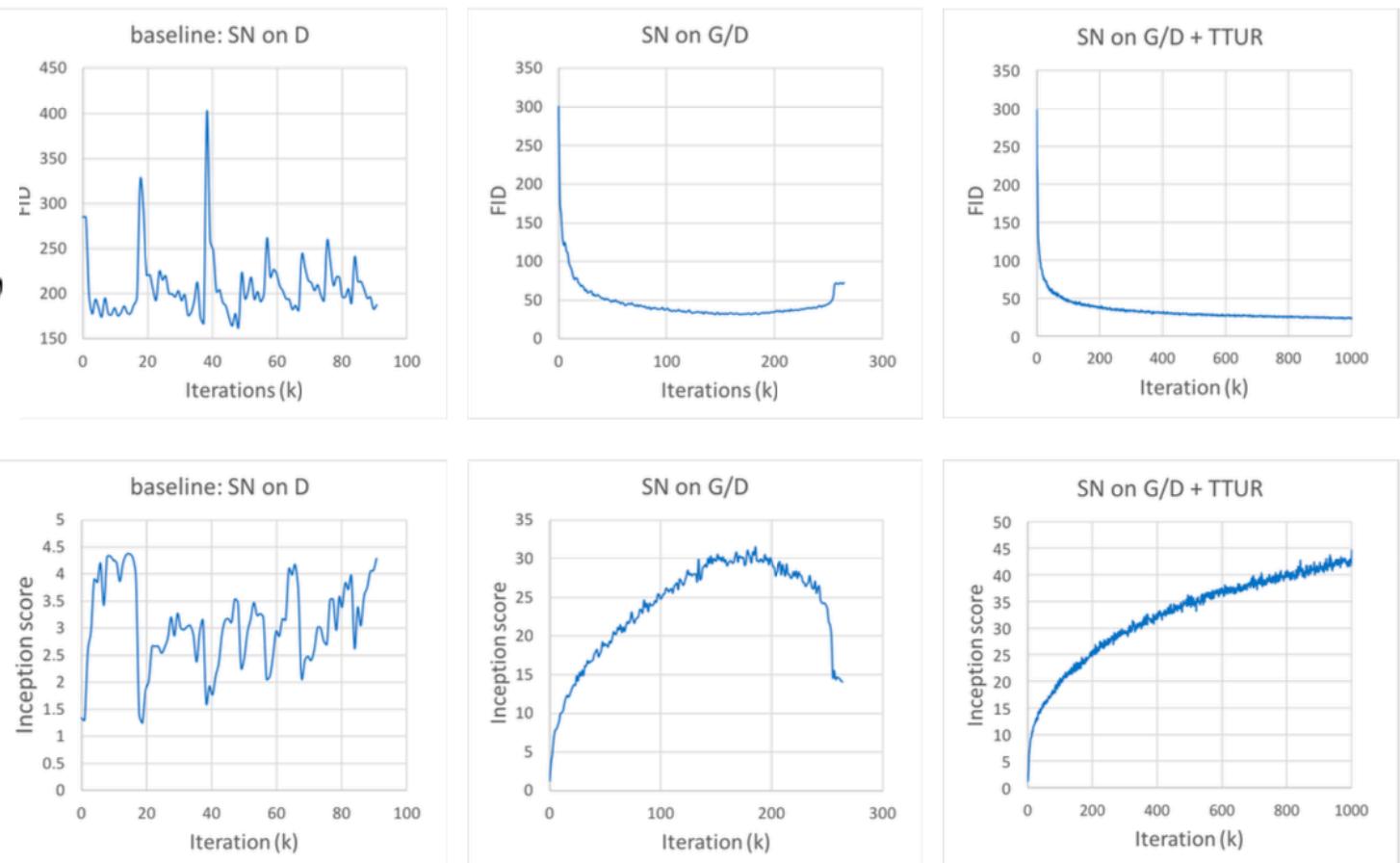
$$\begin{aligned} L_D &= -\mathbb{E}_{(x,y) \sim p_{data}} [\min(0, -1 + D(x, y))] \\ &\quad - \mathbb{E}_{z \sim p_z, y \sim p_{data}} [\min(0, -1 - D(G(z), y))], \\ L_G &= -\mathbb{E}_{z \sim p_z, y \sim p_{data}} D(G(z), y), \end{aligned}$$

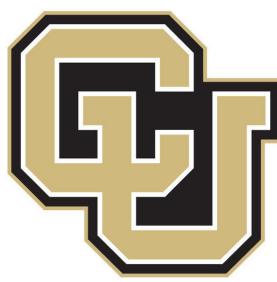
Techniques to stabilize the training of GANs

- 1) Spectral Normalization
- 2) Two Time-scale Update Rule (TTUR)

Evaluation Metrics

- 1) Inception Score
- 2) Fréchet Inception Distance (FID)





Boulder

Analyzing and Improving the Image Quality of StyleGAN



[YouTube Video](#)

$z \in \mathcal{Z} \rightarrow$ input latent code

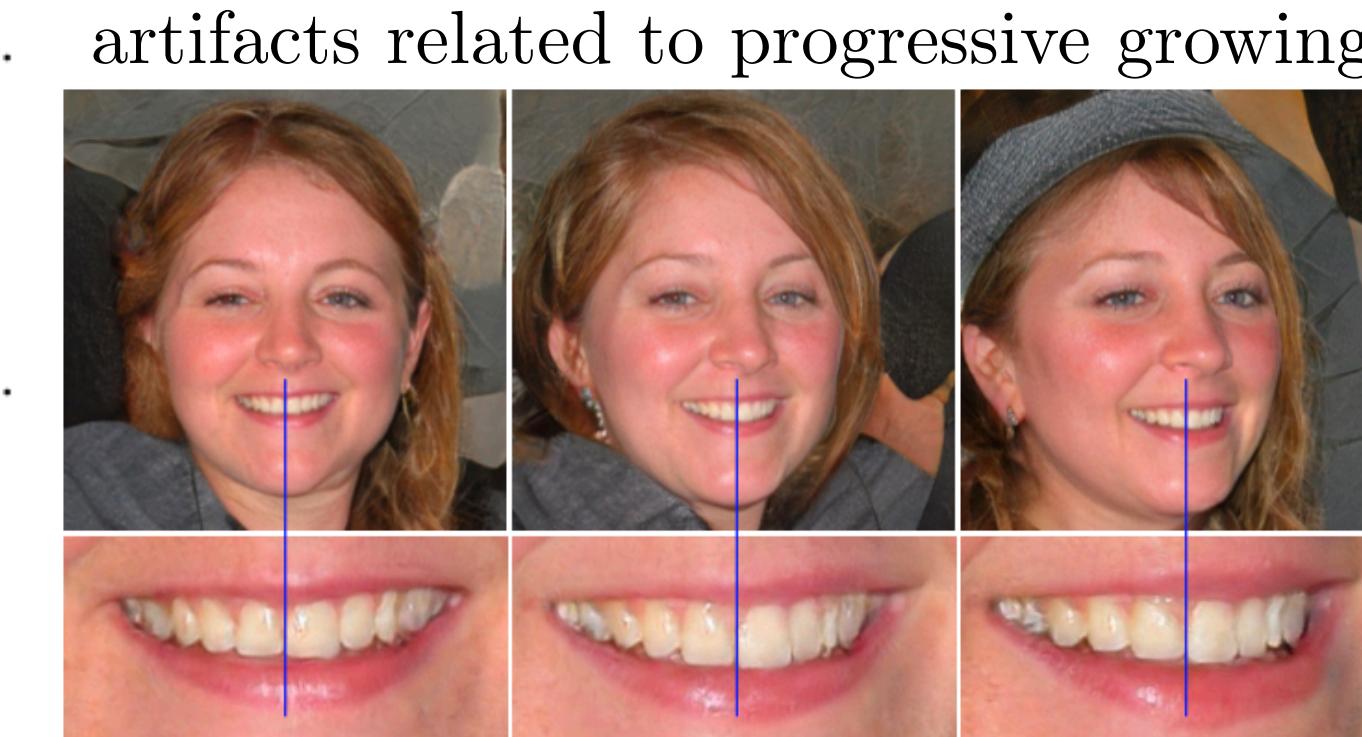
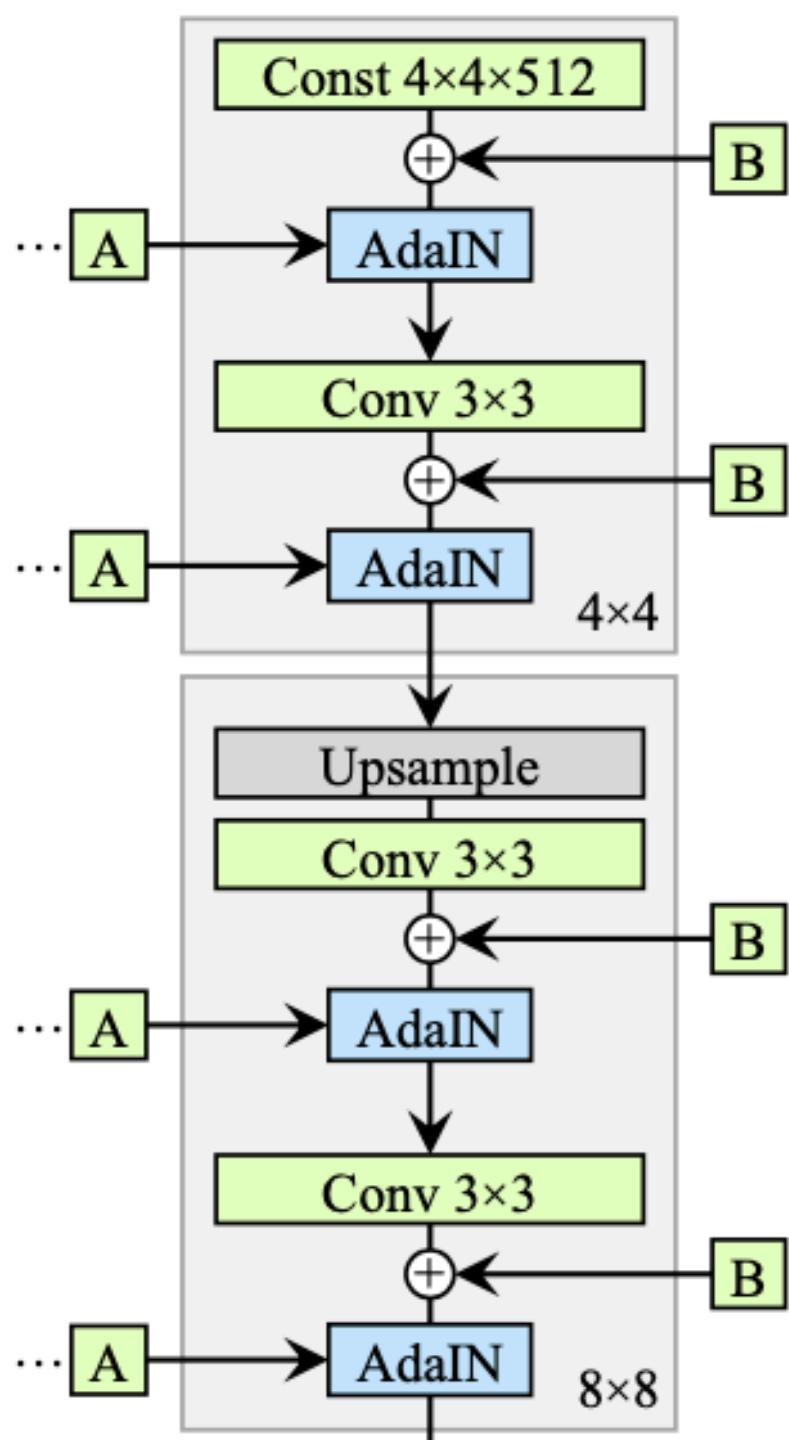
$f \rightarrow$ mapping network

$w = f(z) \in \mathcal{W} \rightarrow$ intermediate latent code

affine transformations \rightarrow styles

$g \rightarrow$ synthesis network

blob-shaped artifacts that resemble water droplets



(a) StyleGAN

- Blob-shaped artifacts due to Adaptive Instance Normalization (AdaIN)
- Artifacts related to progressive growing

$$w'_{ijk} = s_i \cdot w_{ijk} \rightarrow \text{modulation}$$

$$w''_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k} w'_{ijk}^2 + \epsilon} \rightarrow \text{demodulation}$$

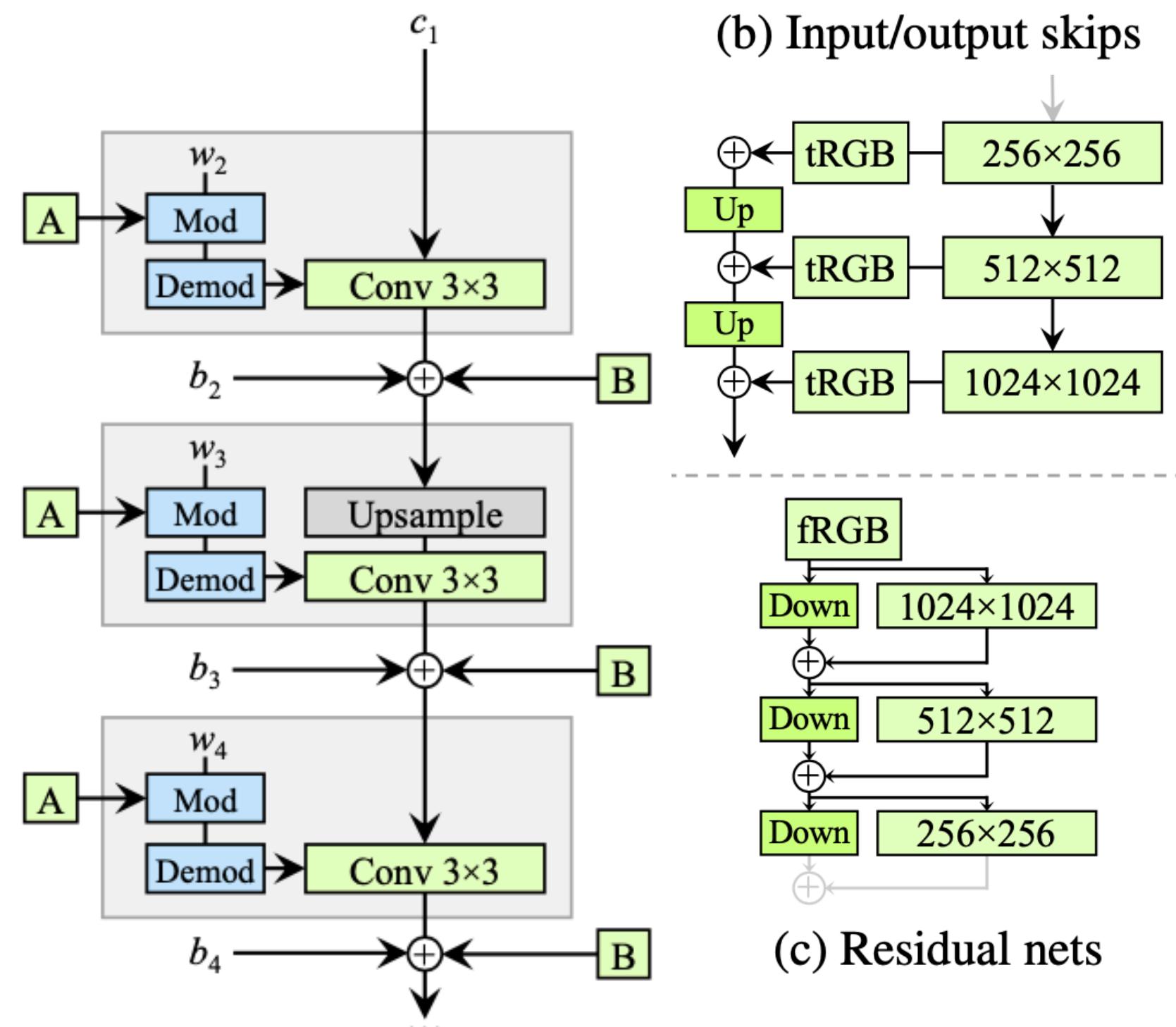
$w, w', w'' \rightarrow$ original, modulated, and demodulated weights

$s_i \rightarrow$ scale corresponding to i -th input feature map

$j \rightarrow$ enumerates the output feature maps

$k \rightarrow$ enumerates the spatial footprint of the convolution

(b) Input/output skips



Path Length Regularization

$$\mathbb{E}_{\mathbf{w}} \mathbb{E}_{\mathbf{y}} (\|\mathbf{J}_{\mathbf{w}}^T \mathbf{y}\|_2 - a)^2$$

$$\mathbf{J}_{\mathbf{w}} = \partial g(\mathbf{w}) / \partial \mathbf{w} \rightarrow \text{Jacobian Matrix}$$

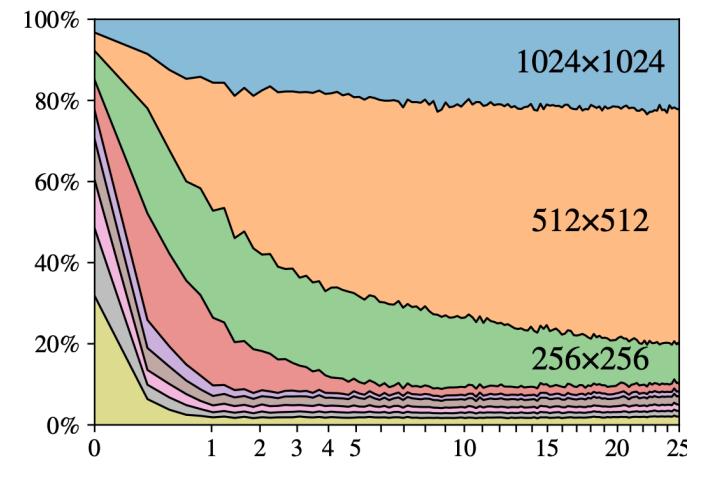
$$\mathbf{y} \sim \mathcal{N}(0, \mathbf{I})$$

$$\mathbf{w} \sim f(\mathbf{z})$$

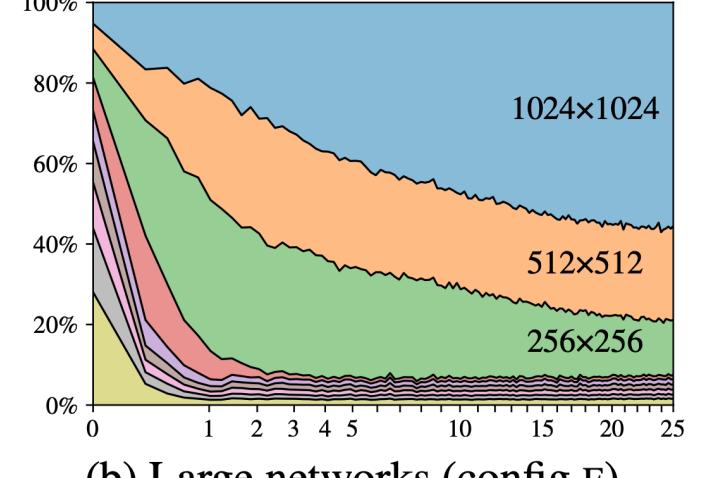
$a \rightarrow$ exponential moving average of $\|\mathbf{J}_{\mathbf{w}}^T \mathbf{y}\|_2$

This is minimized when the Jacobian is orthogonal, preserving length and introducing no squeezing along any dimension

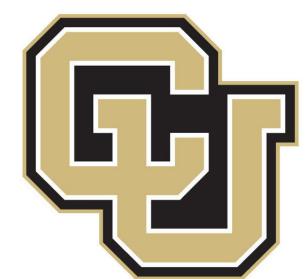
Configuration	FID ↓
A Baseline StyleGAN [21]	4.40
B + Weight demodulation	4.39
C + Lazy regularization	4.38
D + Path length regularization	4.34
E + No growing, new G & D arch.	3.31
F + Large networks (StyleGAN2)	2.84
Config A with large networks	3.98



(a) StyleGAN-sized (config E)



(b) Large networks (config F)



Boulder



Questions?

[YouTube Playlist](#)
