



Boulder

Generative Networks



[YouTube Playlist](#)

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu



Boulder

Auto-Encoding Variational Bayes



[YouTube Video](#)

$X = \{x^i\}_{i=1}^N \rightarrow$ dataset (N i.i.d samples of some continuous or discrete random variable x)

$z \rightarrow$ unobserved (latent) continuous random variable

$z^i \rightarrow$ generated from $\underbrace{p_{\theta^*}(z)}$
prior distribution

$x^i \rightarrow$ generated from some conditional distribution $\underbrace{p_{\theta^*}(x|z)}$
likelihood (generative model or probabilistic decoder)

$p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz \rightarrow$ marginal likelihood (interactive)

$p_{\theta}(z|x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(x)} \rightarrow$ posterior (interactive)

Image denoising, inpainting, and super-resolution

$\underbrace{q_{\phi}(z|x)}$ \rightarrow an approximation to the posterior distribution
recognition model (probabilistic encoder)

$$\log p_{\theta}(x^1, \dots, x^N) = \sum_{i=1}^N \log p_{\theta}(x^i)$$

$$\log p_{\theta}(x^i) = \underbrace{KL(q_{\phi}(z|x^i)||p_{\theta}(z|x^i))}_{\geq 0} + \underbrace{\mathbb{E}_{q_{\phi}(z|x^i)}[-\log q_{\phi}(z|x^i) + \log p_{\theta}(x^i, z)]}_{\mathcal{L}(\theta, \phi; x^i)}$$



reparameterization trick

$z \sim q_{\phi}(z|x)$ auxiliary noise variable

$z = \underbrace{g_{\phi}(\epsilon, x)}$ with $\overbrace{\epsilon \sim p(\epsilon)}$
differentiable transformation

$$\mathcal{L}(\theta, \phi; x^i) \approx \frac{1}{L} \sum_{\ell=1}^L [\log p_{\theta}(x^i, z^{i,\ell}) - \log q_{\phi}(z^{i,\ell}|x^i)]$$

$$z^{i,\ell} = g_{\phi}(\epsilon^{\ell}, x^i), \epsilon^{\ell} \sim p(\epsilon)$$

Variational Auto-Encoder

$$p_{\theta}(z) = \mathcal{N}(z; 0, I)$$

$$\log q_{\phi}(z|x^i) = \log \mathcal{N}(z; \mu(x^i), \sigma^2(x^i)I)$$

$$g_{\phi}(\epsilon, x) = \mu(x) + \sigma(x)\epsilon$$

$$\epsilon \sim \mathcal{N}(\epsilon; 0, I)$$

$\log p_{\theta}(x|z)$ depends on the type of data
(e.g., Gaussian or Bernoulli)

6	6	/	7	8	1	4	8	2	8
9	6	8	3	9	6	8	3	1	9
5	3	7	1	3	6	9	1	7	9
8	9	0	8	6	9	1	9	6	3
9	2	3	3	3	1	3	8	6	
6	9	9	8	6	1	6	6	6	
9	5	2	6	6	5	1	8	9	9
9	9	9	1	3	1	2	8	2	3
0	4	6	1	2	3	2	0	8	9
9	7	5	4	9	3	4	8	5	1



Boulder

Stochastic Backpropagation and Approximate Inference in Deep Generative Models

Deep Latent Gaussian Models (DLGMs)

$\xi_l \sim \mathcal{N}(\xi_l | \mathbf{0}, \mathbf{I})$, $l = 1, \dots, L$ \rightarrow mutually independent Gaussian variables

$$\mathbf{h}_L = \mathbf{G}_L \xi_L,$$

$$\mathbf{h}_l = T_l(\mathbf{h}_{l+1}) + \mathbf{G}_l \xi_l, \quad l = 1 \dots L-1$$

$$\mathbf{v} \sim \pi(\mathbf{v} | T_0(\mathbf{h}_1)),$$

$\mathbf{G}_l \rightarrow$ matrices

$T_l \rightarrow$ MLPs (multi-layer perceptrons)

$\pi \rightarrow$ any appropriate distribution

$\theta^g \rightarrow$ parameters of the MLPs and the matrices

$p(\theta^g) = \mathcal{N}(\theta | \mathbf{0}, \kappa \mathbf{I}) \rightarrow$ Gaussian prior

Scalable Inference in DLGMs

$\mathbf{V} \rightarrow$ full dataset of size $N \times D$

$\mathbf{v}_n = [v_{n1}, \dots, v_{nD}]^\top \rightarrow$ observations

$$\mathcal{L}(\mathbf{V}) = -\log p(\mathbf{V}) = -\log \int p(\mathbf{V} | \xi, \theta^g) p(\xi, \theta^g) d\xi$$

$$= -\log \int \frac{q(\xi)}{q(\xi)} p(\mathbf{V} | \xi, \theta^g) p(\xi, \theta^g) d\xi$$

$$\leq \mathcal{F}(\mathbf{V}) = D_{KL}[q(\xi) \| p(\xi)] - \mathbb{E}_q [\log p(\mathbf{V} | \xi, \theta^g) p(\theta^g)]$$

$$q(\xi | \mathbf{V}, \theta^r) = \prod_{n=1}^N \prod_{l=1}^L \mathcal{N}(\xi_{n,l} | \mu_l(\mathbf{v}_n), \mathbf{C}_l(\mathbf{v}_n)) \rightarrow \text{recognition model}$$

$$D_{KL}[\mathcal{N}(\mu, \mathbf{C}) \| \mathcal{N}(\mathbf{0}, \mathbf{I})] = \frac{1}{2} [\text{Tr}(\mathbf{C}) - \log |\mathbf{C}| + \mu^\top \mu - D]$$

$$\mathcal{F}(\mathbf{V}) = -\sum_n \mathbb{E}_q [\log p(\mathbf{v}_n | \mathbf{h}(\xi_n))] + \frac{1}{2\kappa} \|\theta^g\|^2$$

$$+ \frac{1}{2} \sum_{n,l} [\|\mu_{n,l}\|^2 + \text{Tr}(\mathbf{C}_{n,l}) - \log |\mathbf{C}_{n,l}| - 1]$$

$$\nabla_{\theta_j^g} \mathcal{F}(\mathbf{V}) = -\mathbb{E}_q [\nabla_{\theta_j^g} \log p(\mathbf{V} | \mathbf{h})] + \frac{1}{\kappa} \theta_j^g$$

$$\nabla_{\theta^r} \mathcal{F}(\mathbf{v}) = ?$$

Stochastic Backpropagation

$$\nabla_{\theta} \mathbb{E}_{q_\theta} [f(\xi)]$$

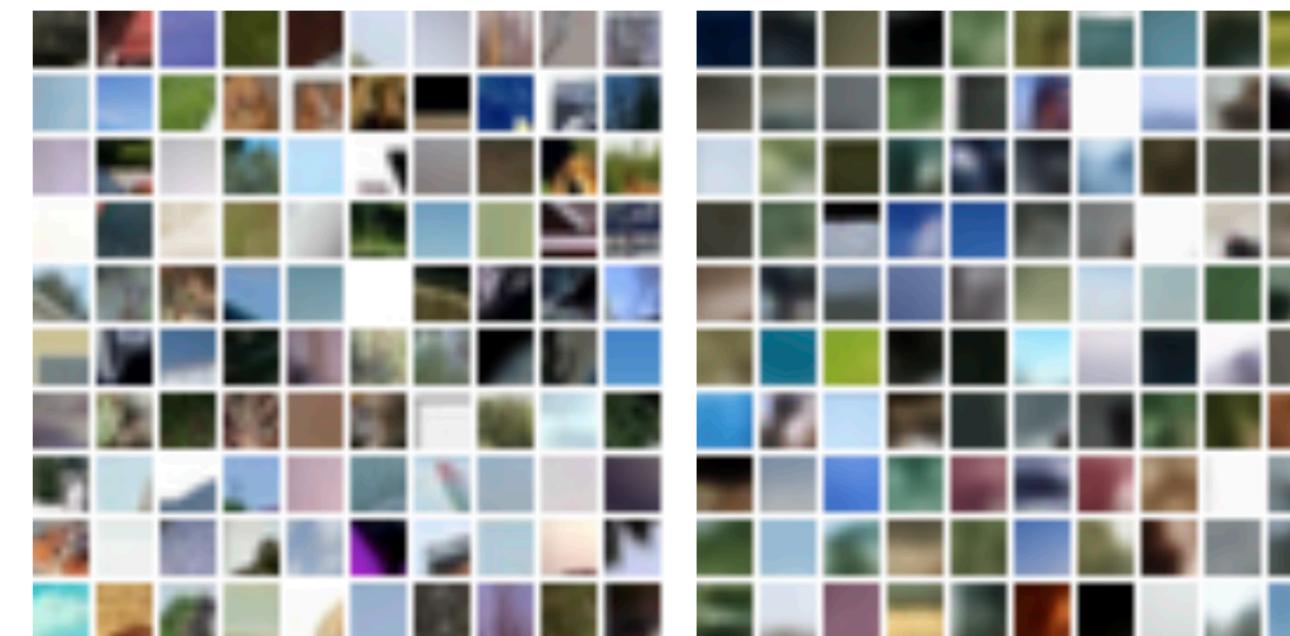
$$f(\xi) = \log p(\mathbf{v} | \mathbf{h}(\xi))$$

$$\nabla_{\mathbf{R}} \mathbb{E}_{\mathcal{N}(\mu, \mathbf{C})} [f(\xi)] = \nabla_{\mathbf{R}} \mathbb{E}_{\mathcal{N}(0, I)} [f(\mu + \mathbf{R}\epsilon)] = \mathbb{E}_{\mathcal{N}(0, I)} [\epsilon \mathbf{g}^\top]$$

$\mathbf{g} \rightarrow$ gradient of the function $f(\xi)$

$$\mathbf{C} = \mathbf{R} \mathbf{R}^\top$$

$$\nabla_{\theta^r} \mathcal{F}(\mathbf{v}) = \nabla_{\mu} \mathcal{F}(\mathbf{v})^\top \frac{\partial \mu}{\partial \theta^r} + \text{Tr} \left(\nabla_{\mathbf{R}} \mathcal{F}(\mathbf{v}) \frac{\partial \mathbf{R}}{\partial \theta^r} \right)$$



(b) CIFAR



(c) Frey



Boulder

Categorical Reparameterization with Gumbel-Softmax

Score Function-based Gradient Estimators (SF)

REINFORCE or likelihood ratio estimator

$$\nabla_{\theta} p_{\theta}(z) = p_{\theta}(z) \nabla_{\theta} \log p_{\theta}(z)$$

$$\nabla_{\theta} \mathbb{E}_z [f(z)] = \mathbb{E}_z [f(z) \nabla_{\theta} \log p_{\theta}(z)]$$

does not require backpropagating through f or the sample z

suffers from high variance

$b(z) \rightarrow$ control variate

$$\mu_b = \mathbb{E}_z [b(z) \nabla_{\theta} \log p_{\theta}(z)] \rightarrow$$
 analytical expectation

$$\nabla_{\theta} \mathbb{E}_z [f(z)] = \mathbb{E}_z [(f(z) - b(z)) \nabla_{\theta} \log p_{\theta}(z)] + \mu_b$$

NVIL, DARN, MuProp, VIMCO

Gumbel-Softmax distribution

$z \rightarrow$ categorical variable with class probabilities

$$\pi = (\pi_1, \dots, \pi_k)$$

k -dimensional one-hot vectors lying on the corners of the $(k - 1)$ -dimensional simplex Δ^{k-1}

$$\mathbb{E}_p[z] = [\pi_1, \dots, \pi_k] \rightarrow$$
 element-wise mean

Gumbel-Max Trick

a simple and efficient way to draw samples z from a categorical distribution with class probabilities π

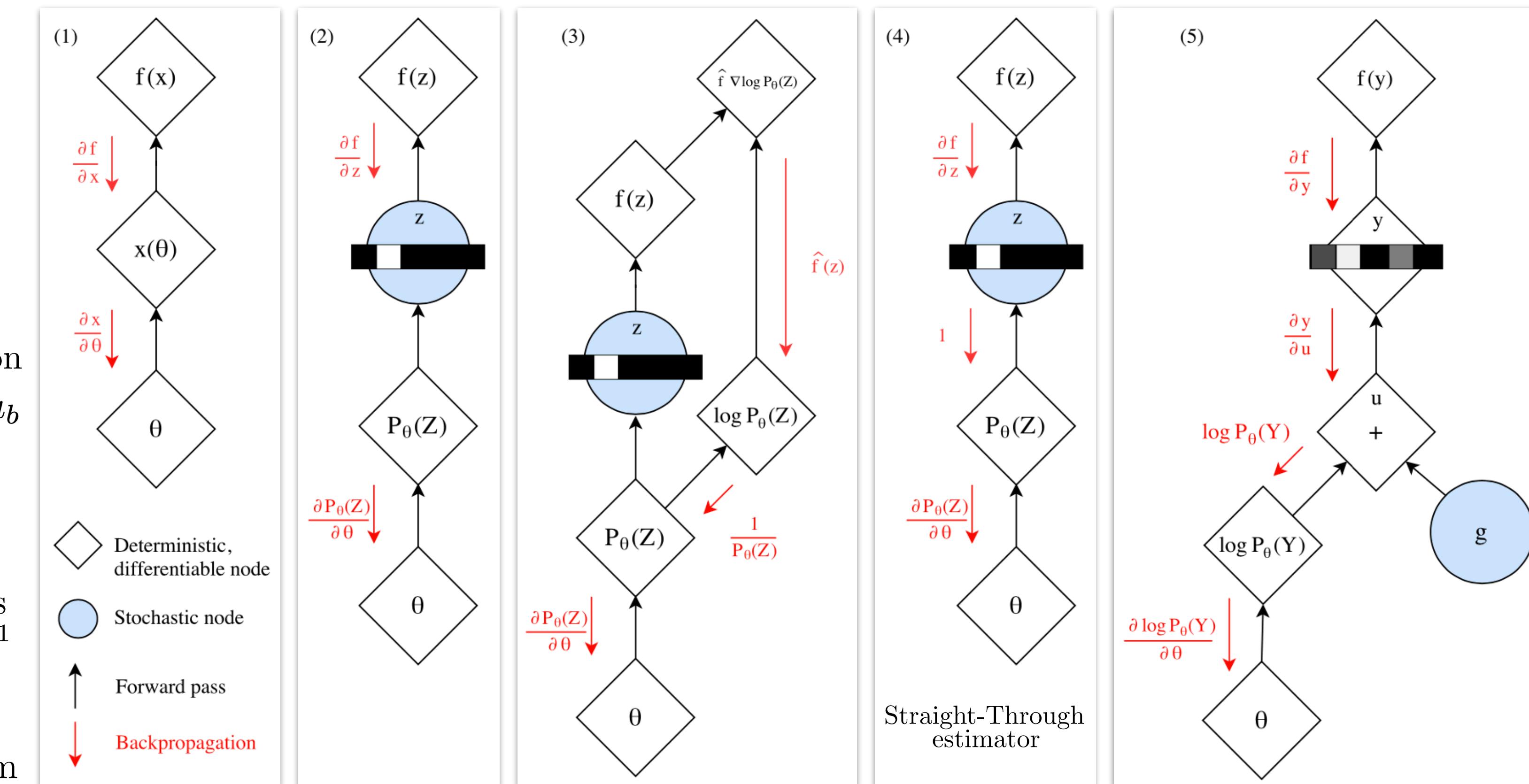
$$z = \text{one_hot} \left(\arg \max_i [g_i + \log \pi_i] \right)$$

$g_1, \dots, g_k \rightarrow$ i.i.d samples drawn from $\text{Gumbel}(0, 1)$

inverse transform sampling

$$u \sim \text{Uniform}(0, 1) \implies g = -\log(-\log(u))$$

softmax: continuous, differentiable approximation to argmax



$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^k \exp((\log(\pi_j) + g_j)/\tau)}$$

for $i = 1, \dots, k$.

$$p_{\pi, \tau}(y_1, \dots, y_k) = \Gamma(k) \tau^{k-1} \left(\sum_{i=1}^k \pi_i / y_i^\tau \right)^{-k} \prod_{i=1}^k (\pi_i / y_i^{\tau+1})$$



Boulder

Generative Adversarial Nets



[YouTube Video](#)

$p_z(z) \rightarrow$ 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:

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

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

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

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

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

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

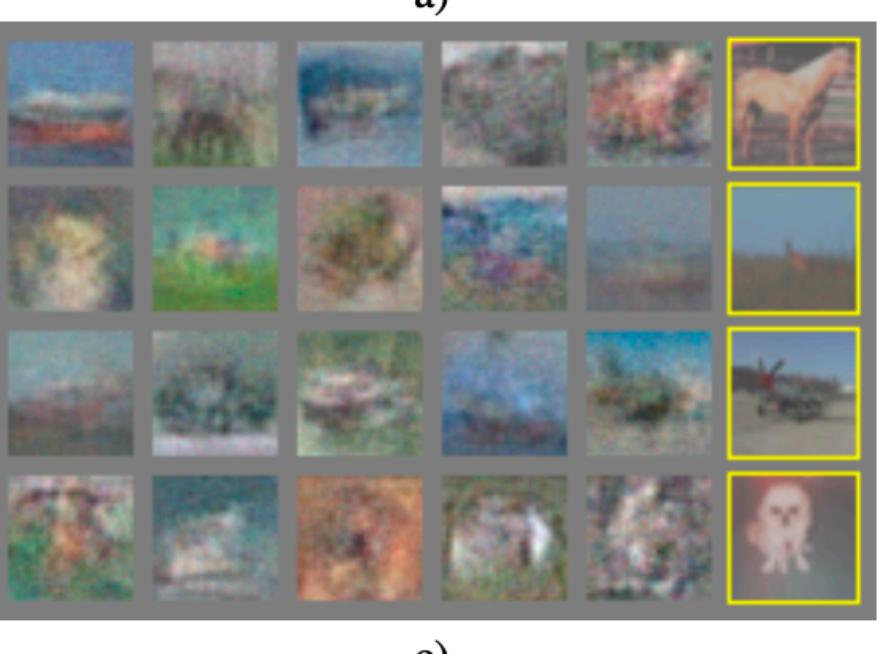
Jensen–Shannon divergence; ≥ 0 ; $= 0$ iff $p_{\text{data}} = p_g$



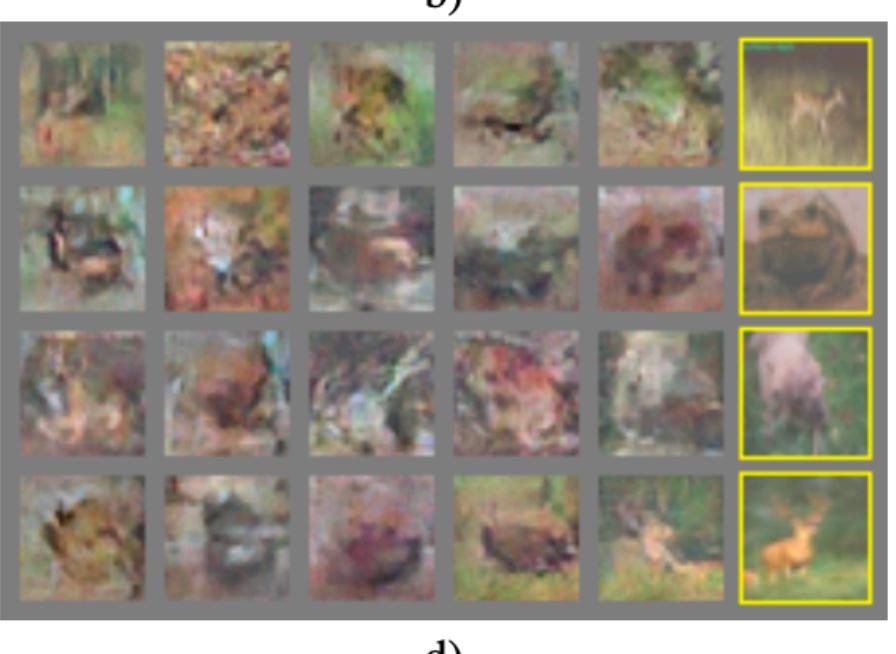
a)



b)



c)



d)



Boulder

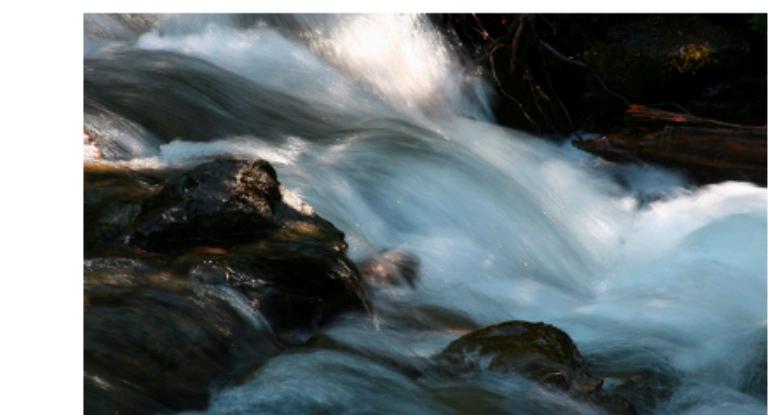
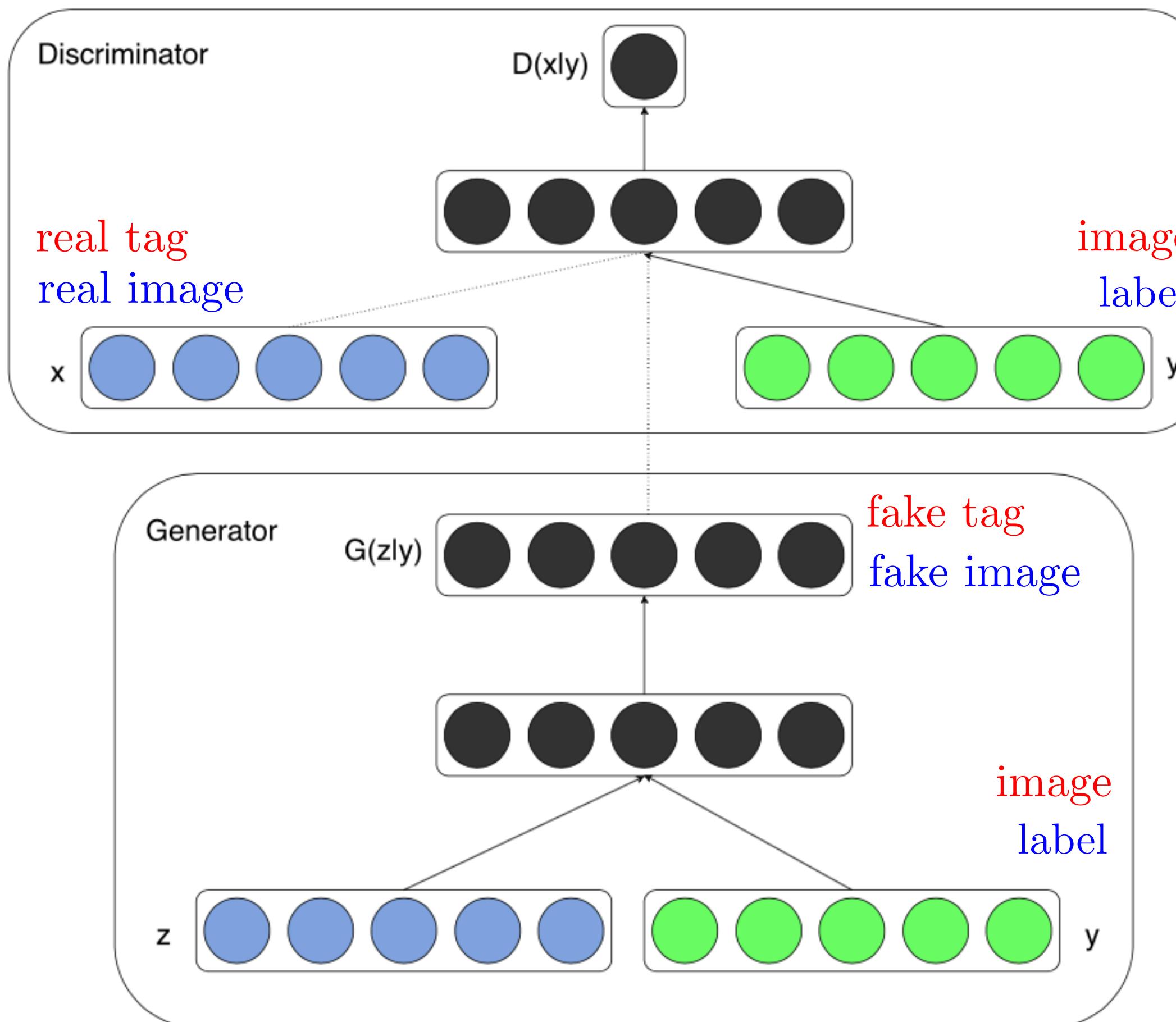
Conditional Generative Adversarial Nets



YouTube Video

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

One-To-Many Mapping



User tags + annotations	Generated tags
montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	<p style="color: red;">cosine similarity</p> <p>taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails</p>
food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life



Unsupervised representation learning with deep convolutional generative adversarial networks



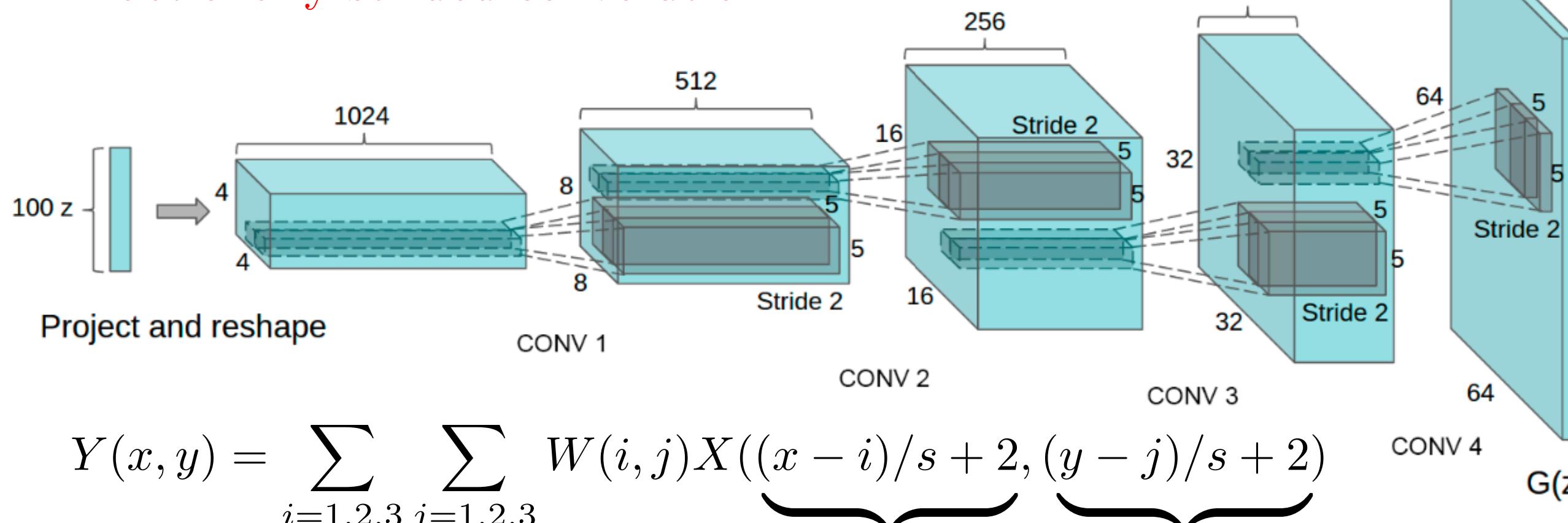
[YouTube Video](#)

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!

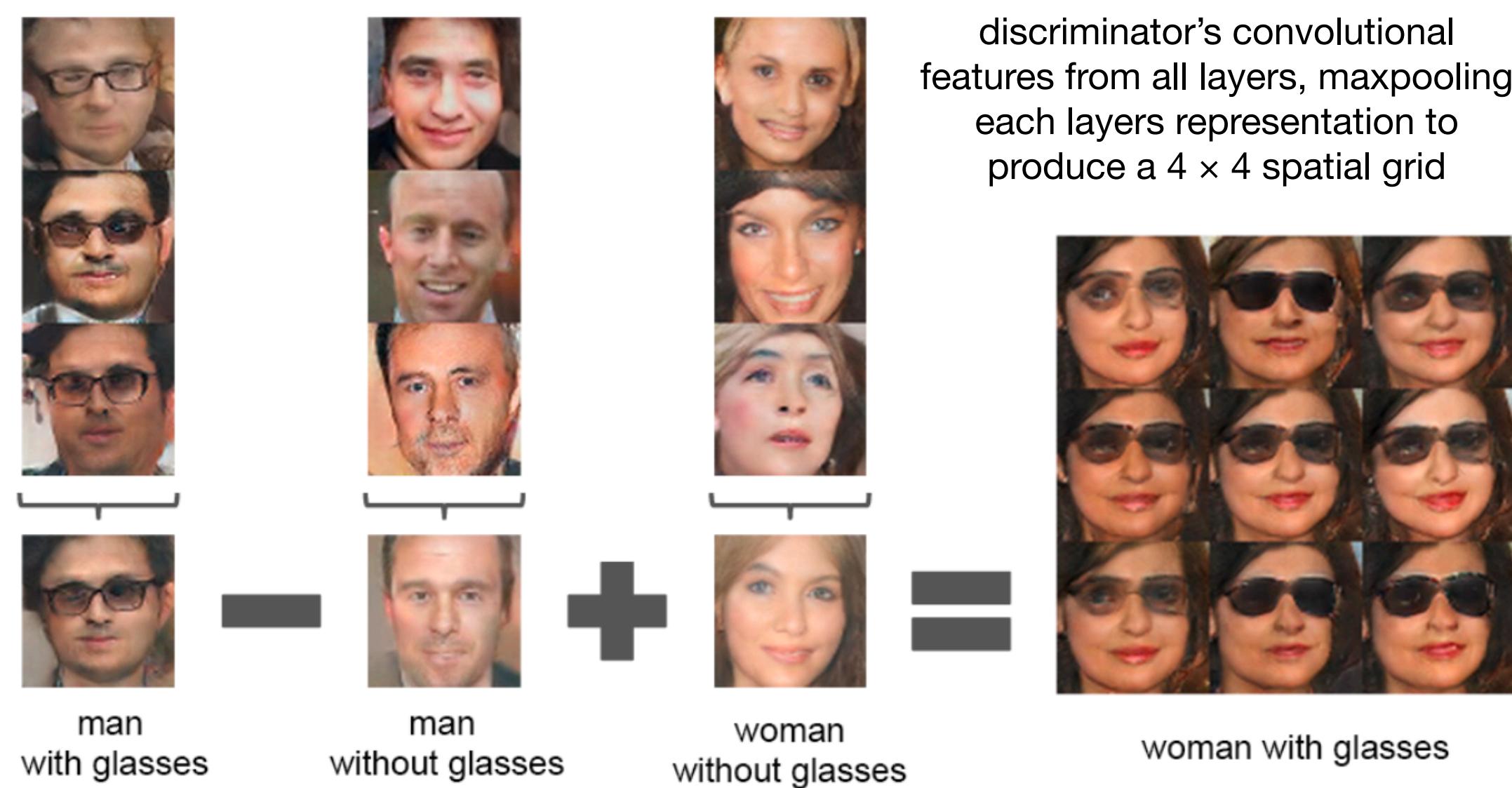
$$Y(x, y) = \sum_{i=1,2,3} \sum_{j=1,2,3} W(i, j) X(\underbrace{s(x - 2) + i}_{=: x'}, \underbrace{s(y - 2) + j}_{=: y'})$$

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

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

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](#)

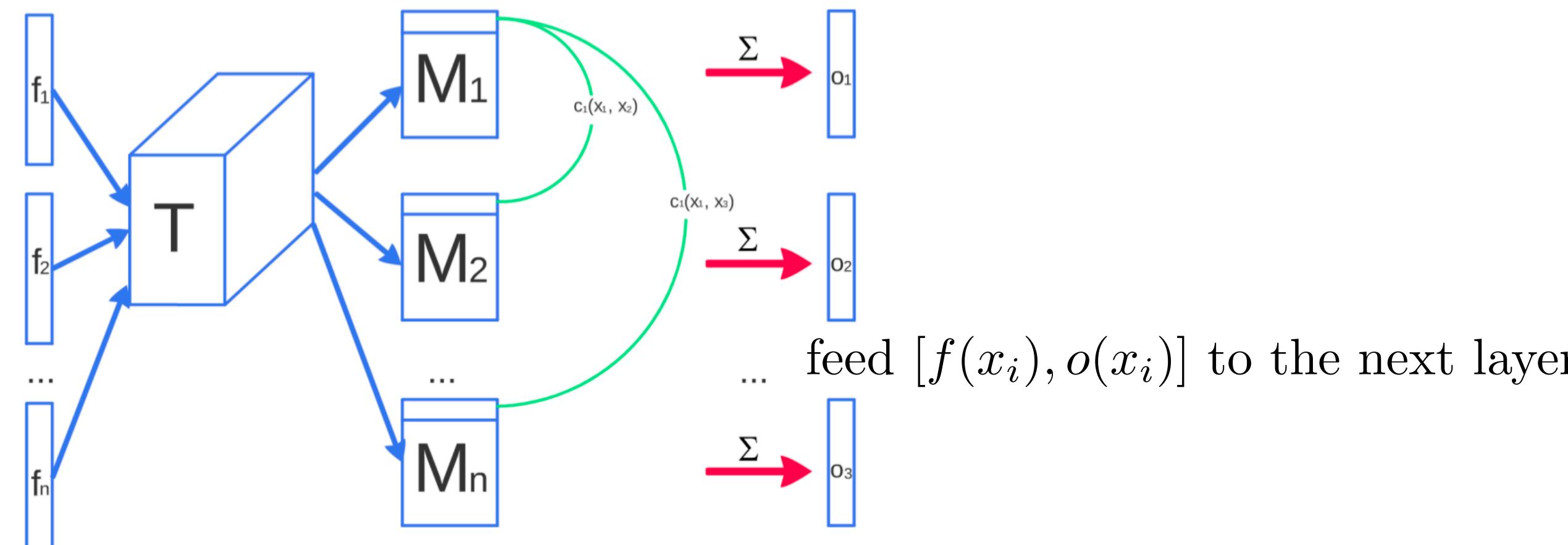
Improved Techniques for Training GANs

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 Video](#)

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 $P(c|x)$

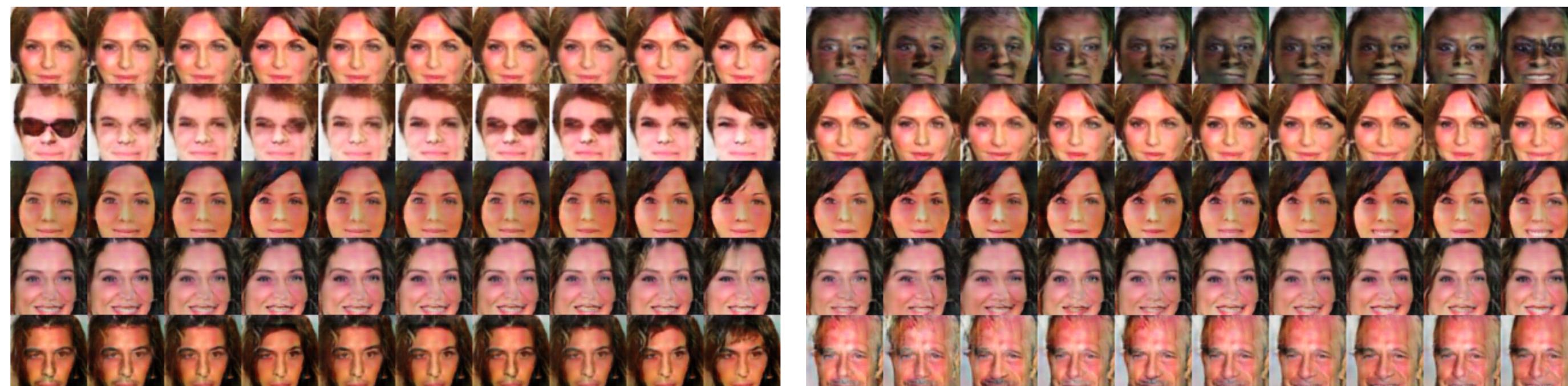
$$\begin{aligned} L_I(G, Q) &= E_{c \sim P(c), x \sim G(z, c)} [\log Q(c|x)] + H(c) \\ &\leq I(c; G(z, c)) \end{aligned}$$

$L_I(G, Q) \rightarrow$ 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 Video](#)

Context Encoders: Feature Learning by Inpainting



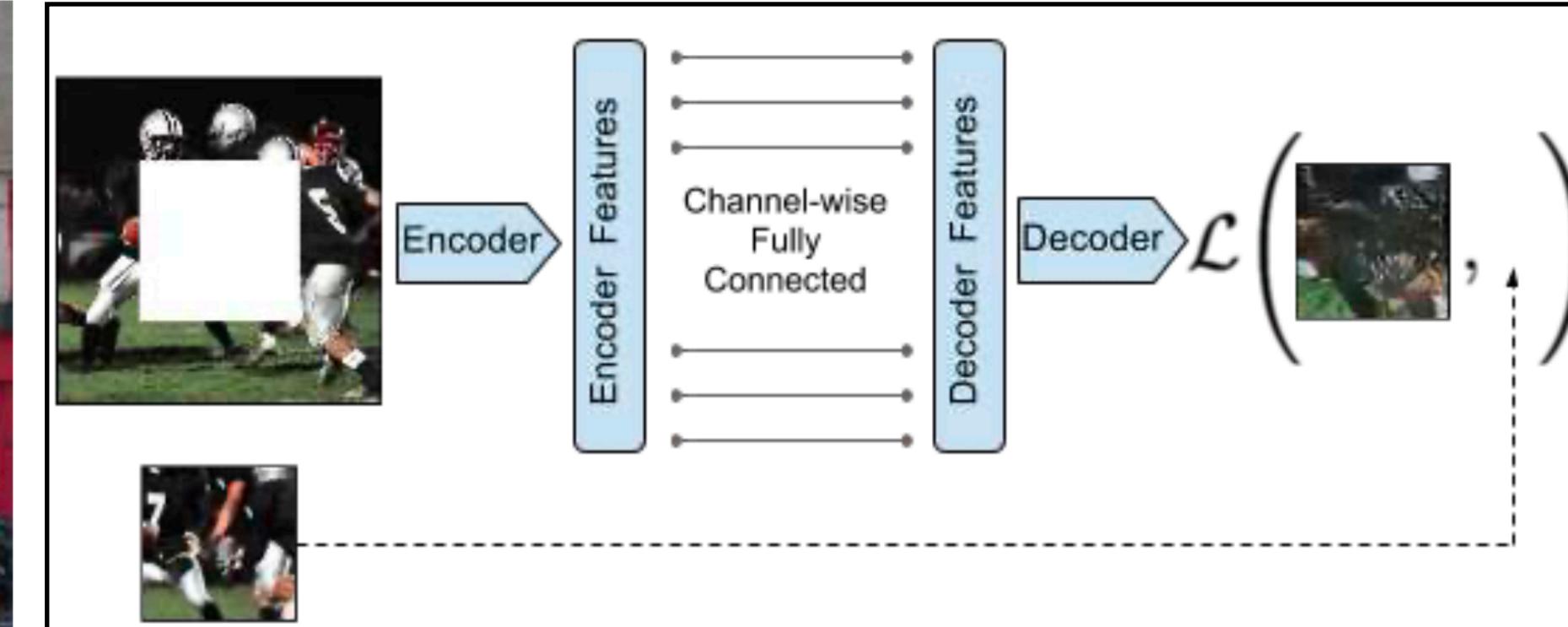
(a) Input context



(c) Context Encoder
(L2 loss)

Loss function
 $x \rightarrow$ ground truth image
 $F \rightarrow$ context encoder

$F(x) \rightarrow$ output
 $\hat{M} \rightarrow$ binary mask (dropped image region)
 1 whenever a pixel is dropped and 0 for input pixels



Encoder
 $\mathbb{R}^{227 \times 227 \times 3} \ni I \triangleright \underbrace{\text{AlexNet(pool5)}}_{\substack{5 \text{ conv layers} \\ 6 \times 6 \times 256 = 9,216 \text{ from scratch}} \in \mathbb{R}^{6 \times 6 \times 256}}$

Channel-wise fully-connected layer

input: m feature maps of size $n \times n$

output: m feature maps of size $n \times n$

$$Y_j = \underbrace{W_j}_{n^2 \times 1} \underbrace{X_j}_{n^2 \times n^2} \in \mathbb{R}^{n^2 \times 1}$$

$mn^4 \rightarrow$ number of parameters (rather than m^2n^4)

Decoder

5 up-conv layers

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2$$

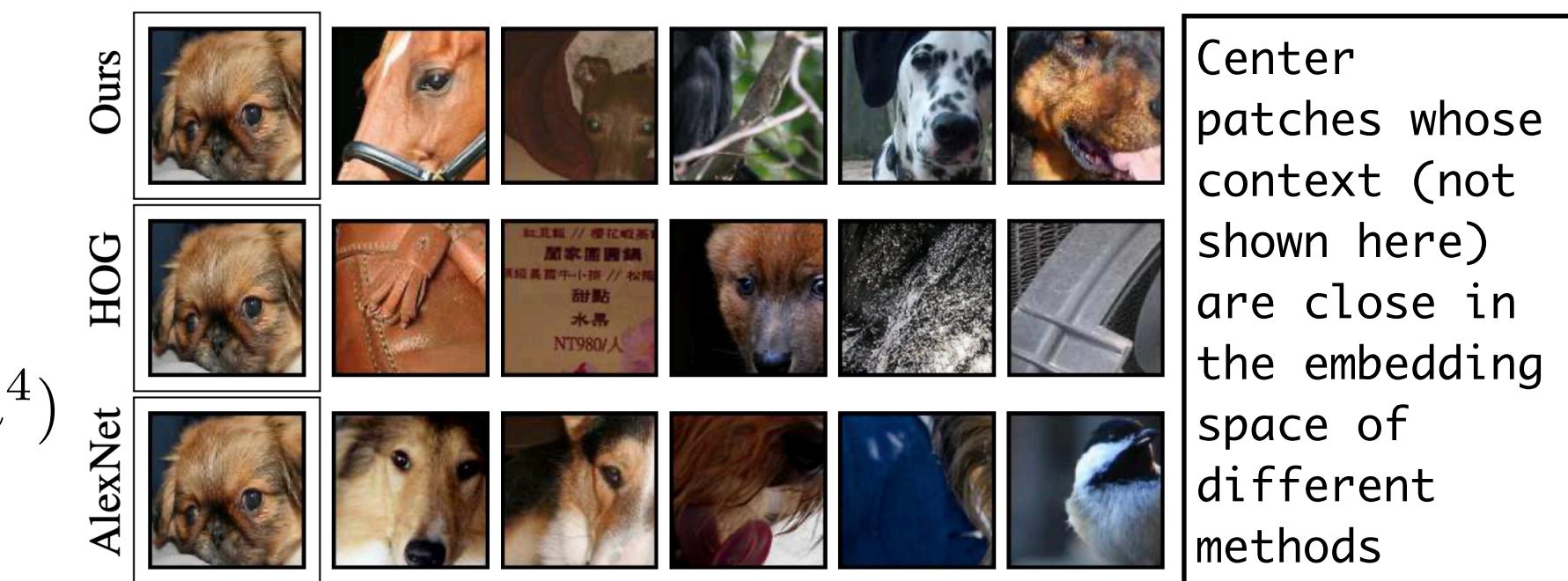
$$\mathcal{L}_{adv} = \max_D \mathbb{E}_{x \in \mathcal{X}} [\log(D(x))]$$

$$+ \log(1 - D(F((1 - \hat{M}) \odot x)))]$$

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}$$



Method	Mean L1 Loss	Mean L2 Loss	PSNR (higher better)
NN-inpainting (HOG features)	19.92%	6.92%	12.79 dB
NN-inpainting (our features)	15.10%	4.30%	14.70 dB
Our Reconstruction (joint)	10.33%	2.35%	17.59 dB



Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	78.2%	56.8%	48.0%
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal <i>et al.</i> [1]	egomotion	10 hours	52.9%	41.8%	-
Doersh <i>et al.</i> [7]	context	4 weeks	55.3%	46.6%	-
Wang <i>et al.</i> [39]	motion	1 week	58.4%	44.0%	-
Ours	context	14 hours	56.5%	44.5%	29.7%



Boulder

Least Squares Generative Adversarial Networks



[YouTube Video](#)

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

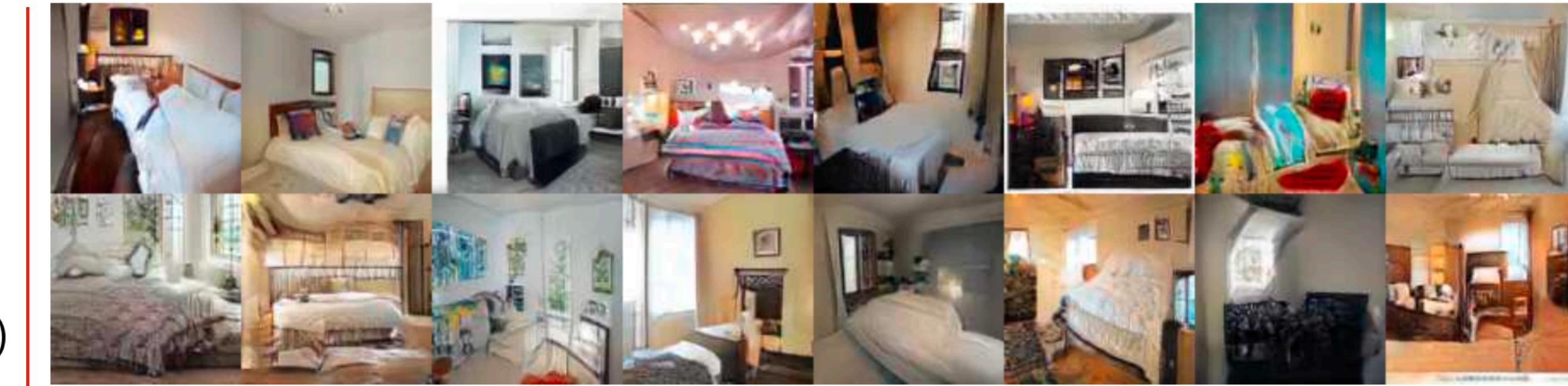
$$D^*(\mathbf{x}) = \frac{bp_{\text{data}}(\mathbf{x}) + ap_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \rightarrow \text{optimal discriminator } D \text{ for a fixed } G$$

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

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

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

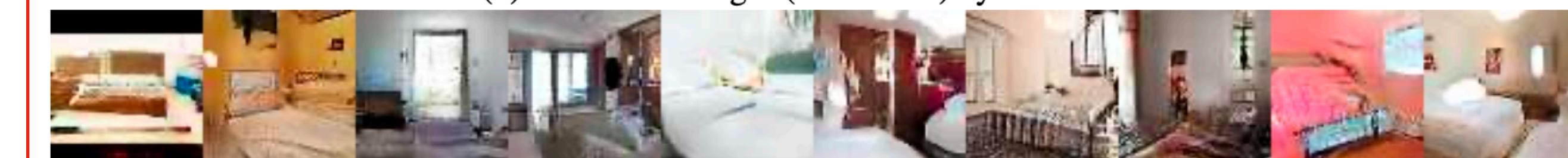
Pearson χ^2 divergence



(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.

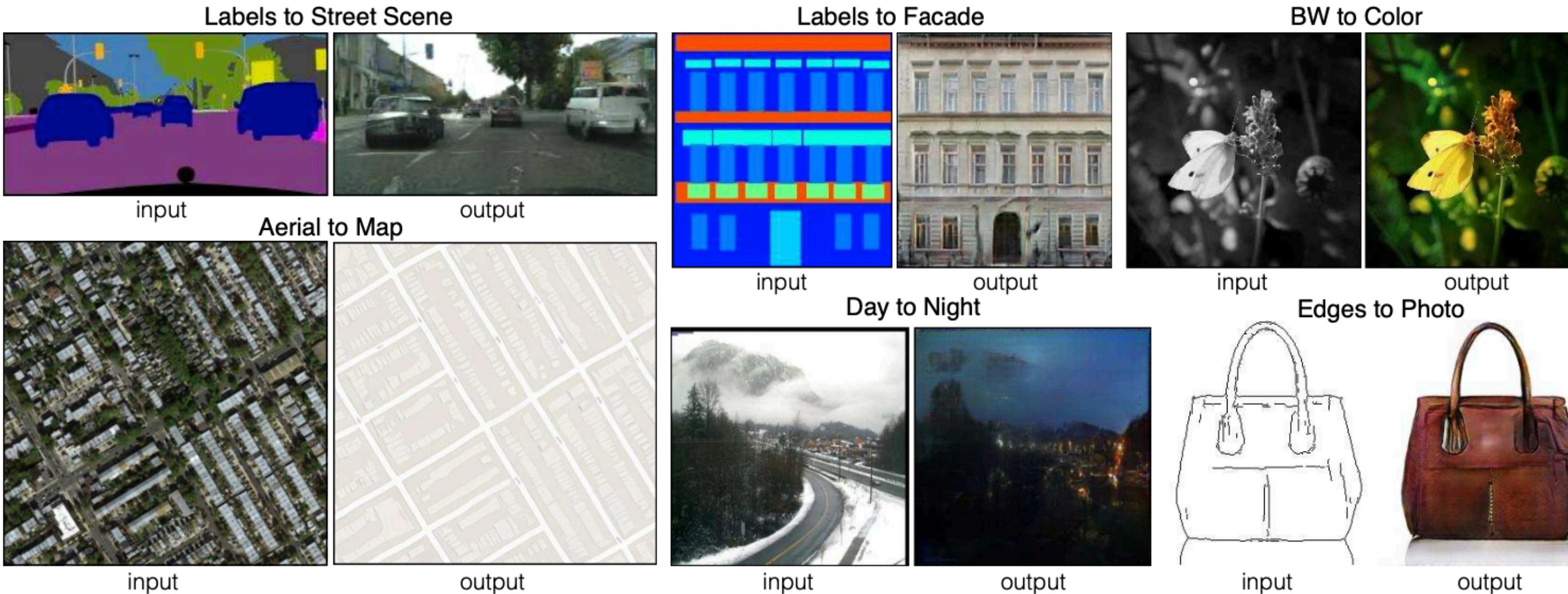


Boulder



[YouTube Video](#)

Image-to-Image Translation with Conditional Adversarial Networks



Background removal



by Kaihu Chen

“Do as I do”



by Brannon Dorsey

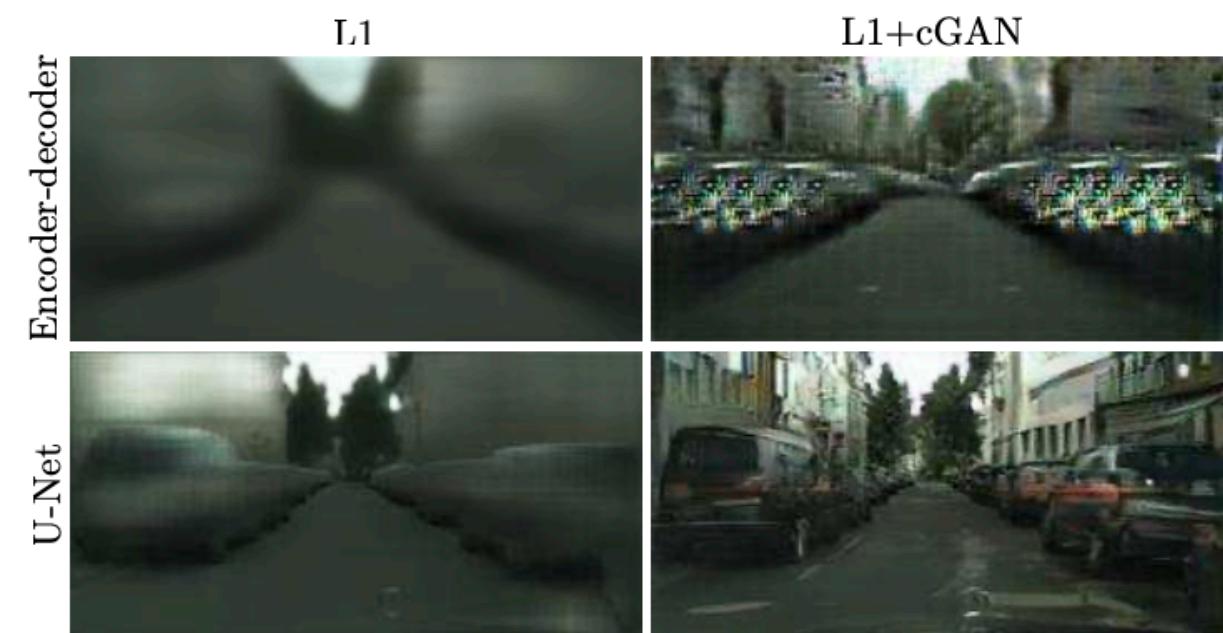
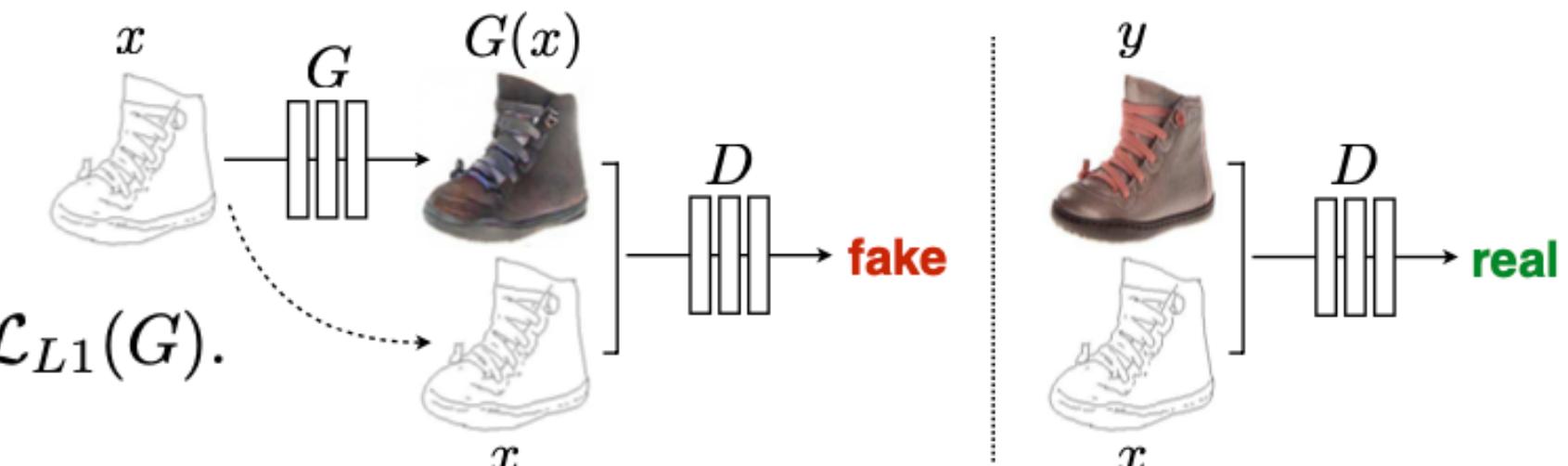
$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \rightarrow \text{objective of a conditional GAN}$$

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D)$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z)))] \rightarrow \text{an unconditional variant}$$



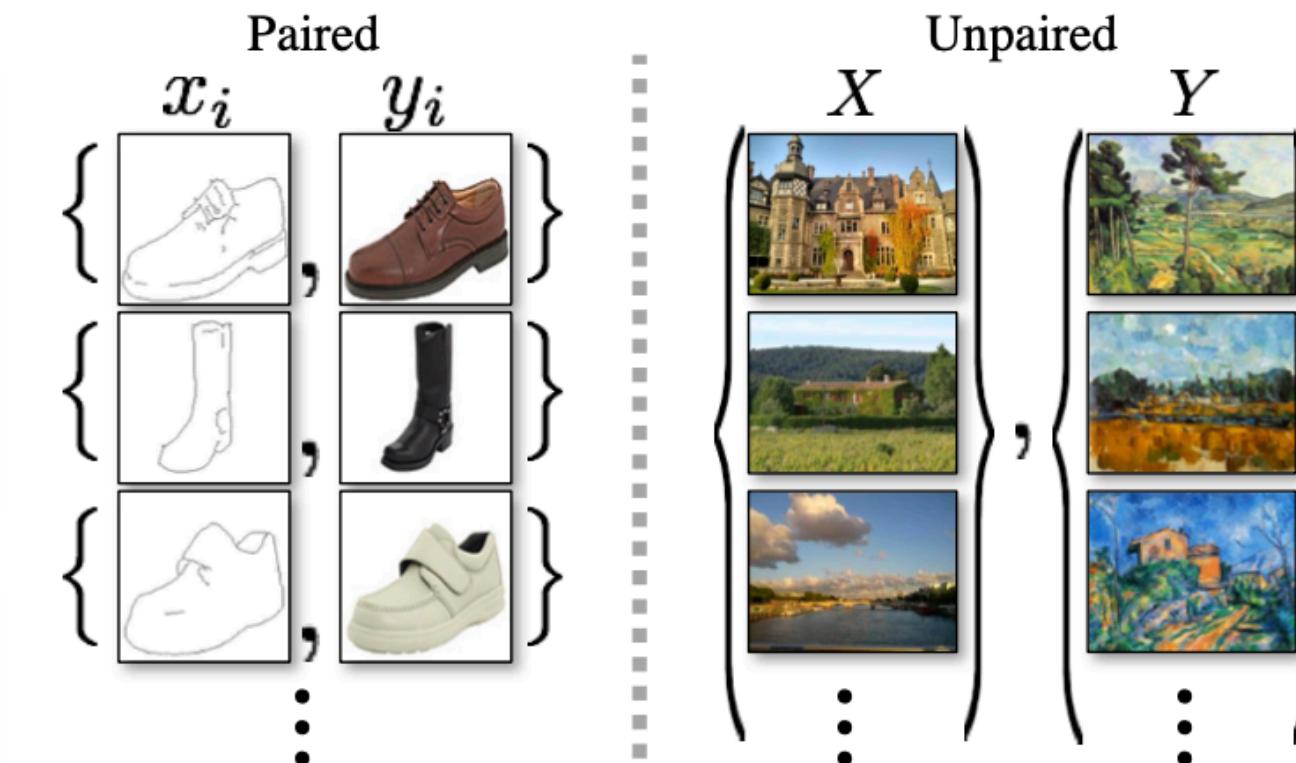
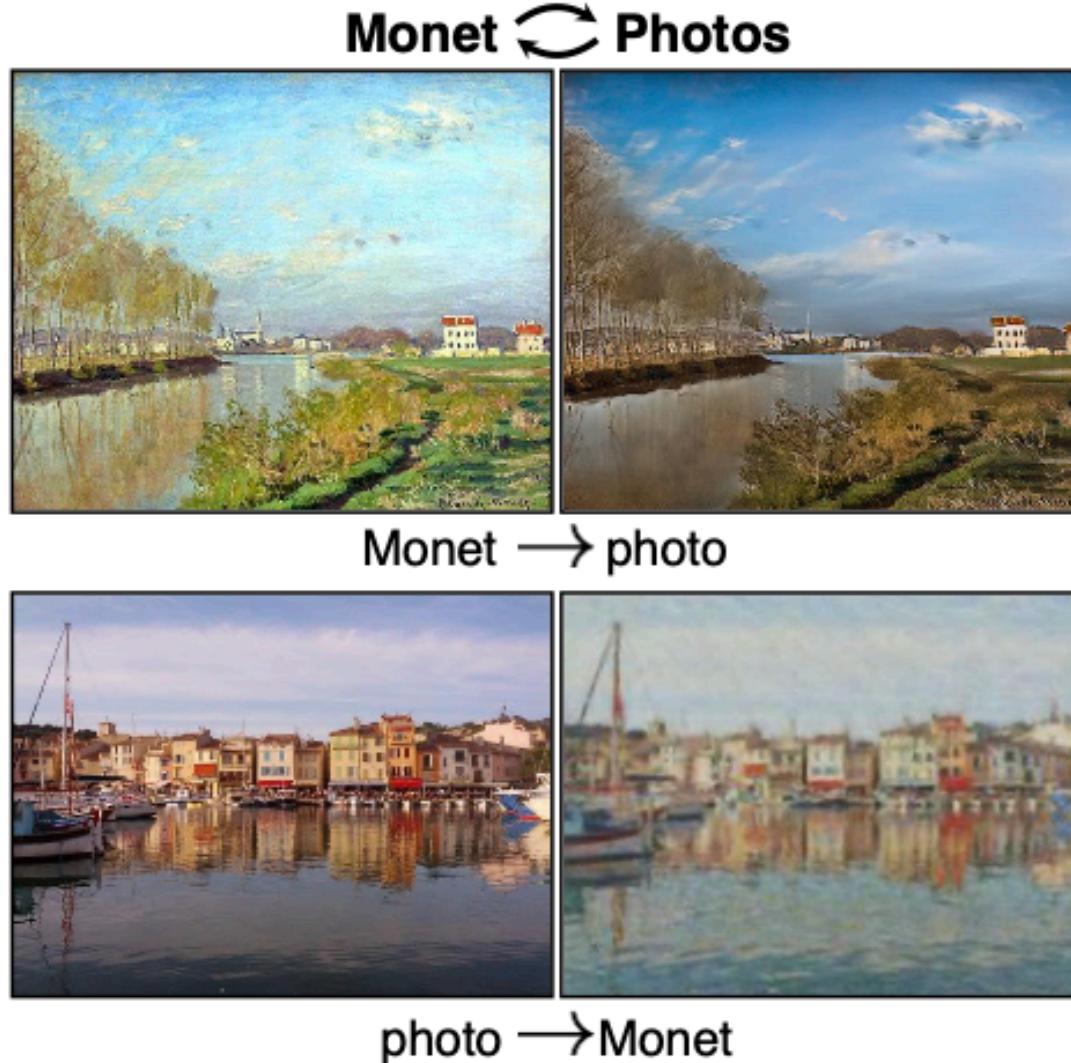


Boulder

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks



[YouTube Video](#)



$$G : X \rightarrow Y$$

$$F : Y \rightarrow X$$

$$F(G(x)) \approx x \text{ and } G(F(y)) \approx y.$$

$D_X \rightarrow$ aims to discriminate between $\{x\}$ and $\{F(y)\}$

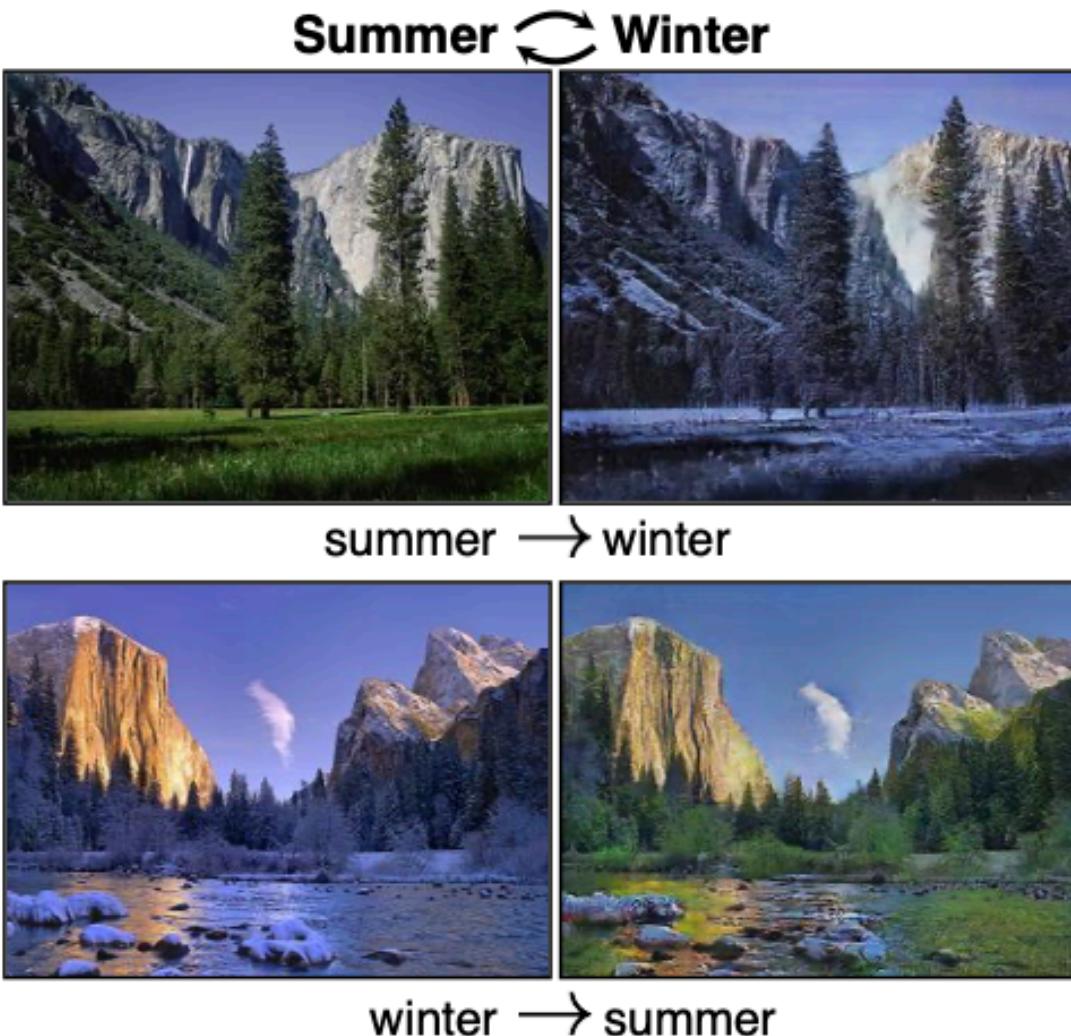
$D_Y \rightarrow$ aims to discriminate between $\{y\}$ and $\{G(x)\}$

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

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

$$\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X).$$

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



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

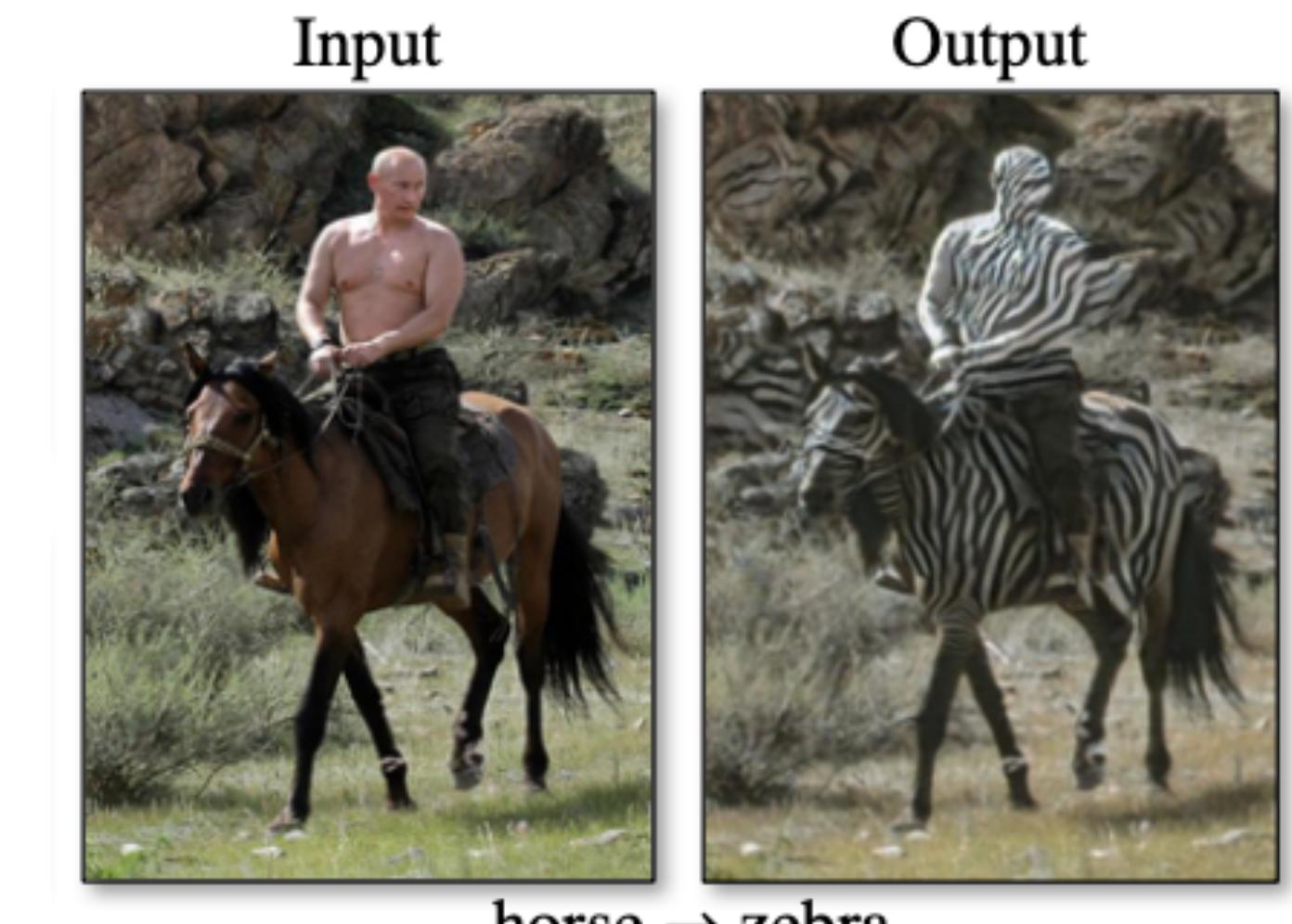
$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

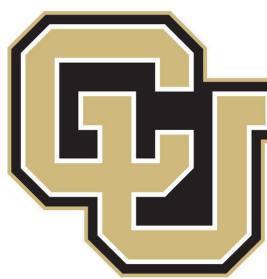
Least-squares Loss

$$\min_G \mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$$

$$\min_D \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$$

update the discriminators using a history of generated images





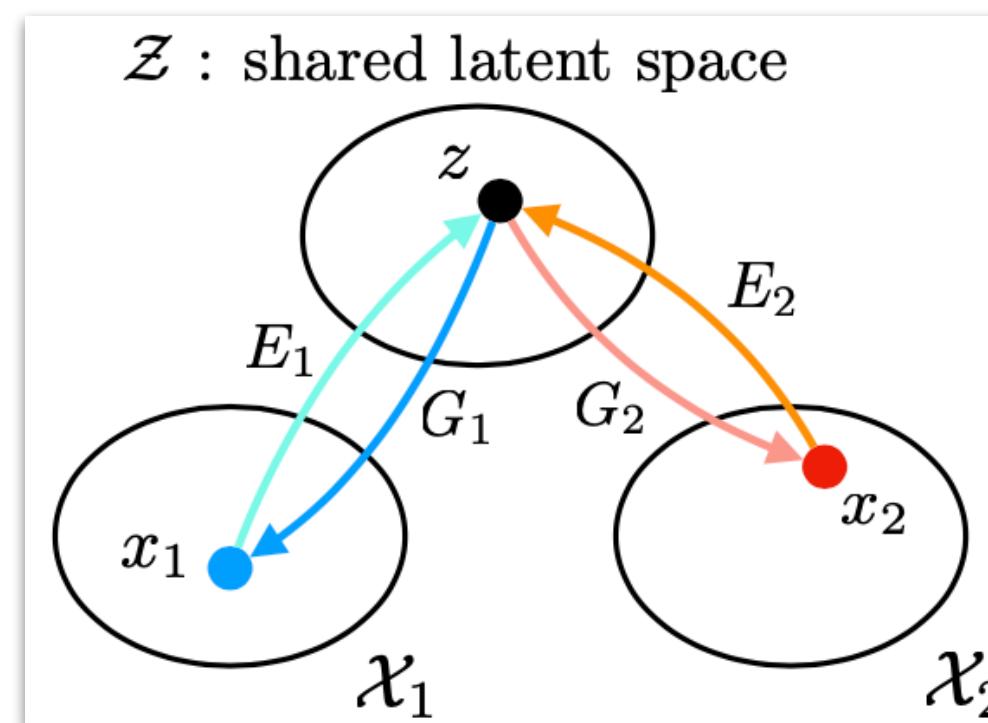
Boulder

Unsupervised Image-to-Image Translation Networks

UNIT: UNsupervised Image-to-image Translation Networks

Goal: To learn a joint distribution of images in different domains by using images from the marginal distributions in individual domains.
Since an infinite set of possible joint distributions can yield the given marginal distributions, we could infer nothing about the joint distribution from the marginal samples without additional assumptions.

Shared Latent Space Assumption



$$z = E_1(x_1) = E_2(x_2)$$

$$x_1 = G_1(z) \quad x_2 = G_2(z)$$

Shared latent space constraint implies cycle-consistency!

$$F^{1 \rightarrow 2} := G_2 \circ E_1 \quad F^{2 \rightarrow 1} := G_1 \circ E_2$$

$$x_1 = F^{2 \rightarrow 1}(F^{1 \rightarrow 2}(x_1))$$

$$x_2 = F^{1 \rightarrow 2}(F^{2 \rightarrow 1}(x_2))$$

$\{E_1, E_2\} \rightarrow$ encoders, $\{G_1, G_2\} \rightarrow$ generators, $\{D_1, D_2\} \rightarrow$ discriminators

Variational Autoencoders (VAEs)

$\{E_1, G_1\} \rightarrow$ VAE for the \mathcal{X}_1 domain (VAE_1)

$$q_1(z_1|x_1) := \mathcal{N}(z_1|E_1(x_1), I)$$

$x_1^{1 \rightarrow 1} = G_1(\underbrace{z_1 \sim q_1(z_1|x_1)}_{\text{re-parametrization trick}}) \rightarrow$ reconstructed image

$$\text{re-parametrization trick } z_1 = E_1(x_1) + \eta, \eta \sim \mathcal{N}(0, I)$$

$$\mathcal{L}_{\text{VAE}_1}(E_1, G_1) = \lambda_1 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)} [\log p_{G_1}(x_1|z_1)]$$

$$p_\eta(z) = \mathcal{N}(z|0, I)$$

$p_{G_1} \rightarrow$ Laplace distribution (equivalent to minimizing the absolute distance)

Weight Sharing

$$E_1 = E_H \circ E_{L,1}, E_2 = E_H \circ E_{L,2}$$

$H, L \rightarrow$ High, Low level features

$$G_1 = G_{L,1} \circ G_H, G_2 = G_{L,2} \circ G_H$$

Note that the weight-sharing constraint alone does not guarantee that corresponding images in two domains will have the same latent code.

Generative Adversarial Networks (GANs)

$$\{G_1, D_1\} \rightarrow \text{GAN}_1$$

G_1 can generate two types of images:

- reconstruction stream (supervised training): $x_1^{1 \rightarrow 1} = G_1(z_1 \sim q_1(z_1|x_1))$
- translation stream (adversarial training): $x_2^{2 \rightarrow 1} = G_1(z_2 \sim q_2(z_2|x_2))$

$$\mathcal{L}_{\text{GAN}_1}(E_1, G_1, D_1) = \lambda_0 \mathbb{E}_{x_1 \sim P_{\mathcal{X}_1}} [\log D_1(x_1)] + \lambda_0 \mathbb{E}_{z_2 \sim q_2(z_2|x_2)} [\log(1 - D_1(G_1(z_2)))]$$

Cycle-consistency (CC)

$$\mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2) = \lambda_3 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) + \lambda_3 \text{KL}(q_2(z_2|x_1^{1 \rightarrow 2}))||p_\eta(z)) - \lambda_4 \mathbb{E}_{z_2 \sim q_2(z_2|x_1^{1 \rightarrow 2})} [\log p_{G_1}(x_1|z_2)]$$

Learning

$$\min_{E_1, E_2, G_1, G_2} \max_{D_1, D_2} \mathcal{L}_{\text{VAE}_1}(E_1, G_1) + \mathcal{L}_{\text{GAN}_1}(E_1, G_1, D_1) + \mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2)$$

$$\mathcal{L}_{\text{VAE}_2}(E_2, G_2) + \mathcal{L}_{\text{GAN}_2}(E_2, G_2, D_2) + \mathcal{L}_{\text{CC}_2}(E_2, G_2, E_1, G_1)$$

Translation

$$F_{1 \rightarrow 2}(x_1) = G_2(z_1 \sim q_1(z_1|x_1))$$

$$F_{2 \rightarrow 1}(x_2) = G_1(z_2 \sim q_2(z_2|x_2))$$





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)

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

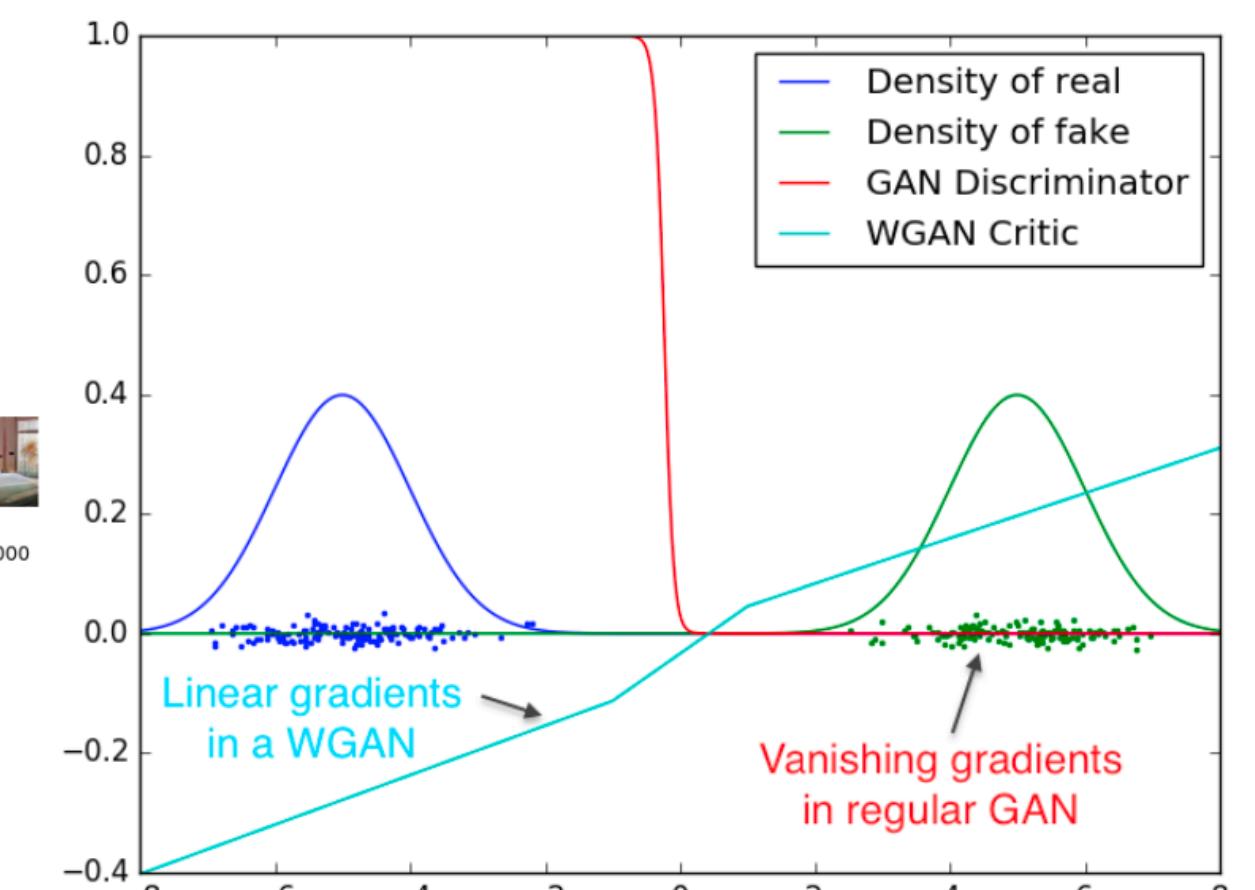
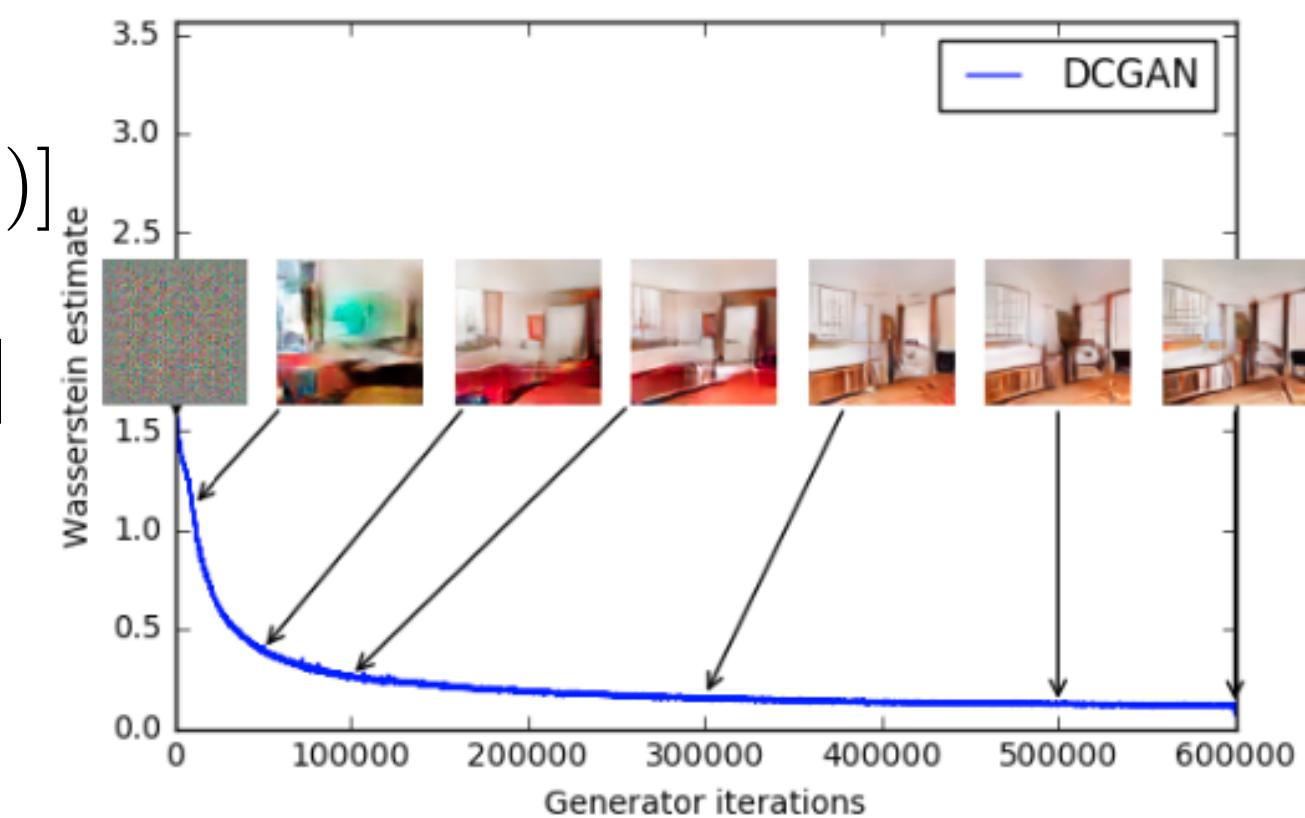
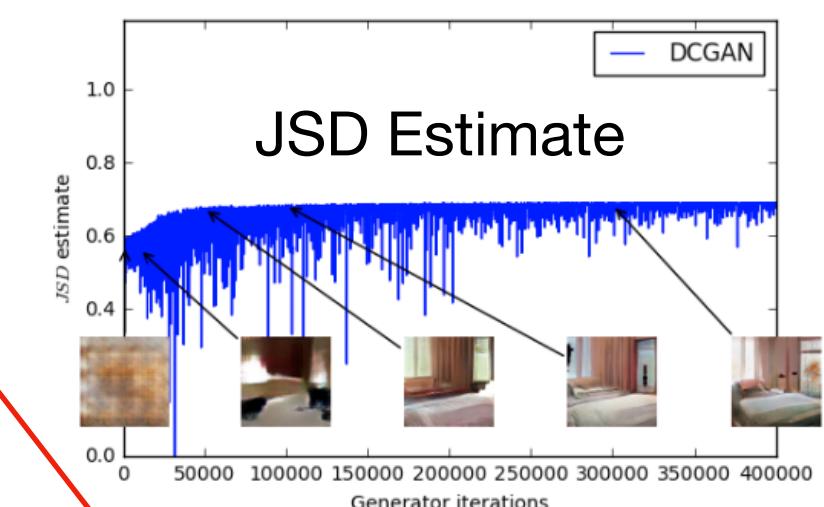




Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network


[YouTube Video](#)


I^{SR} → high-resolution, super-resolved image

I^{LR} → low-resolution input image

I^{HR} → high-resolution counterpart of I^{LR}

$I^{HR} \triangleright$ Gaussian Filter \triangleright downsampling (downsampling_factor = r)

C → number of color channels

$I^{LR} \in \mathbb{R}^{W \times H \times C}$

$I^{SR}, I^{HR} \in \mathbb{R}^{rW \times rH \times C}$

evaluation metrics → $\begin{cases} \text{PSNR: peak signal-to-noise ratio} \\ \text{MOS: mean opinion score} \\ \text{SSIM: structural similarity} \end{cases}$

$G_{\theta_G} : I^{LR} \mapsto I^{HR}$ → generator

$\hat{\theta}^G = \arg \min_{\theta_G} \frac{1}{N} \sum_{n=1}^N \ell^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR})$ → loss function

$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})} [\log D_{\theta_D}(I^{HR})] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})} [\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))]$

$$l^{SR} = \underbrace{l_X^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}$$

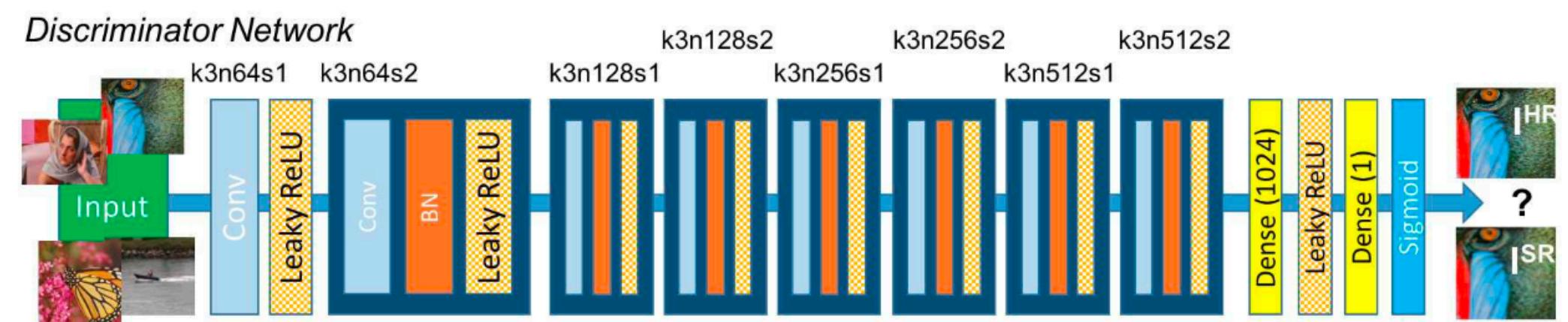
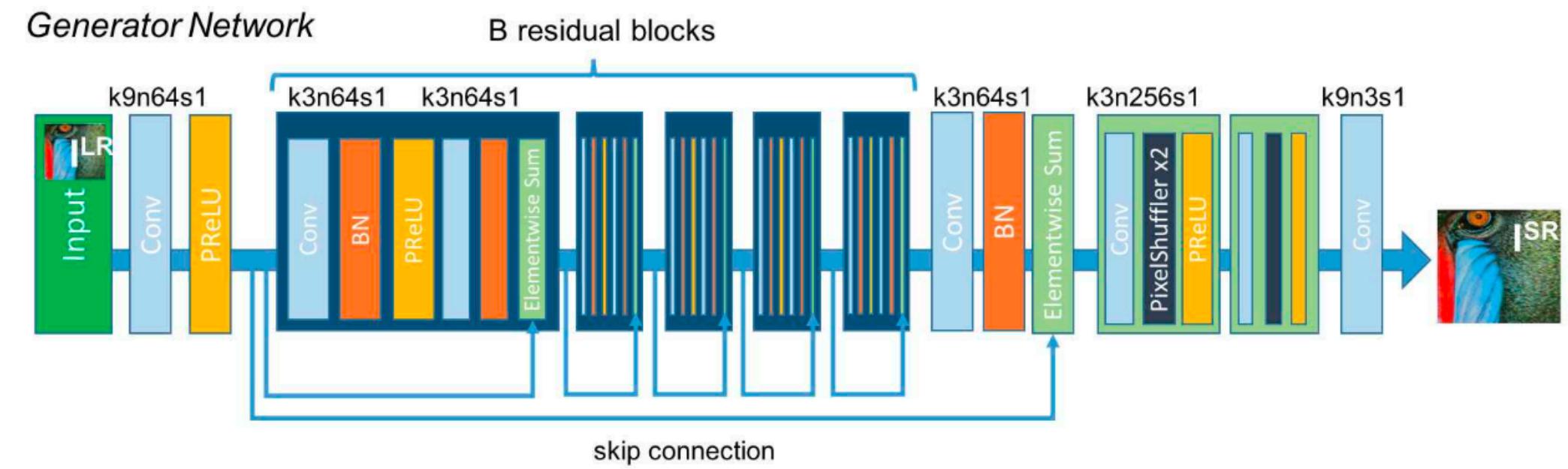
perceptual loss (for VGG based content losses)

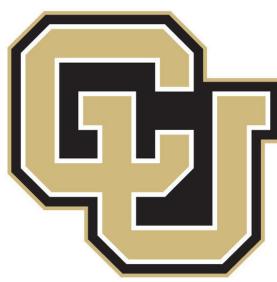
$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

$$l_{MSE}^{SR} = \frac{1}{r^2 W H} \sum_{x=1}^{rW} \sum_{y=1}^{rH} (I_{x,y}^{HR} - G_{\theta_G}(I^{LR})_{x,y})^2$$

$$l_{Gen}^{SR} = \sum_{n=1}^N -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

$\phi_{i,j}$ → feature map obtained by the j -th convolution
(after activation) before the i -th maxpooling layer within the VGG19 network





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

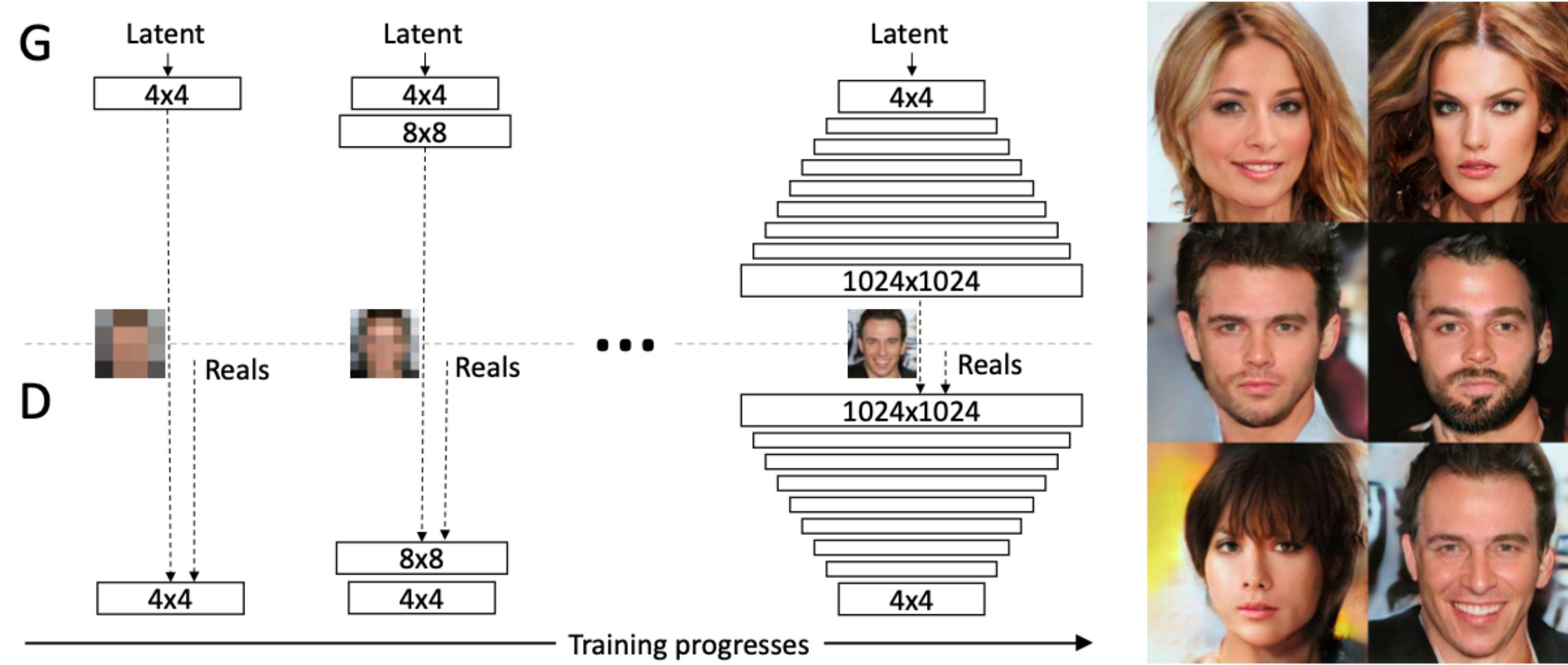
Solice Norkedin pring in since
This record (31.) UBS) and Ch
It was not the annuas were plogr
This will be us , the ect of DAN
These leaded as most-worsd p2 a0
The time I paid0a South Cubry i
Dour Fraps higs it was these del
This year out howneed allowed lo
Kaulna Seto consficates to repor

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

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

$JS(p_r, p_g)$ 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 Video](#)


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

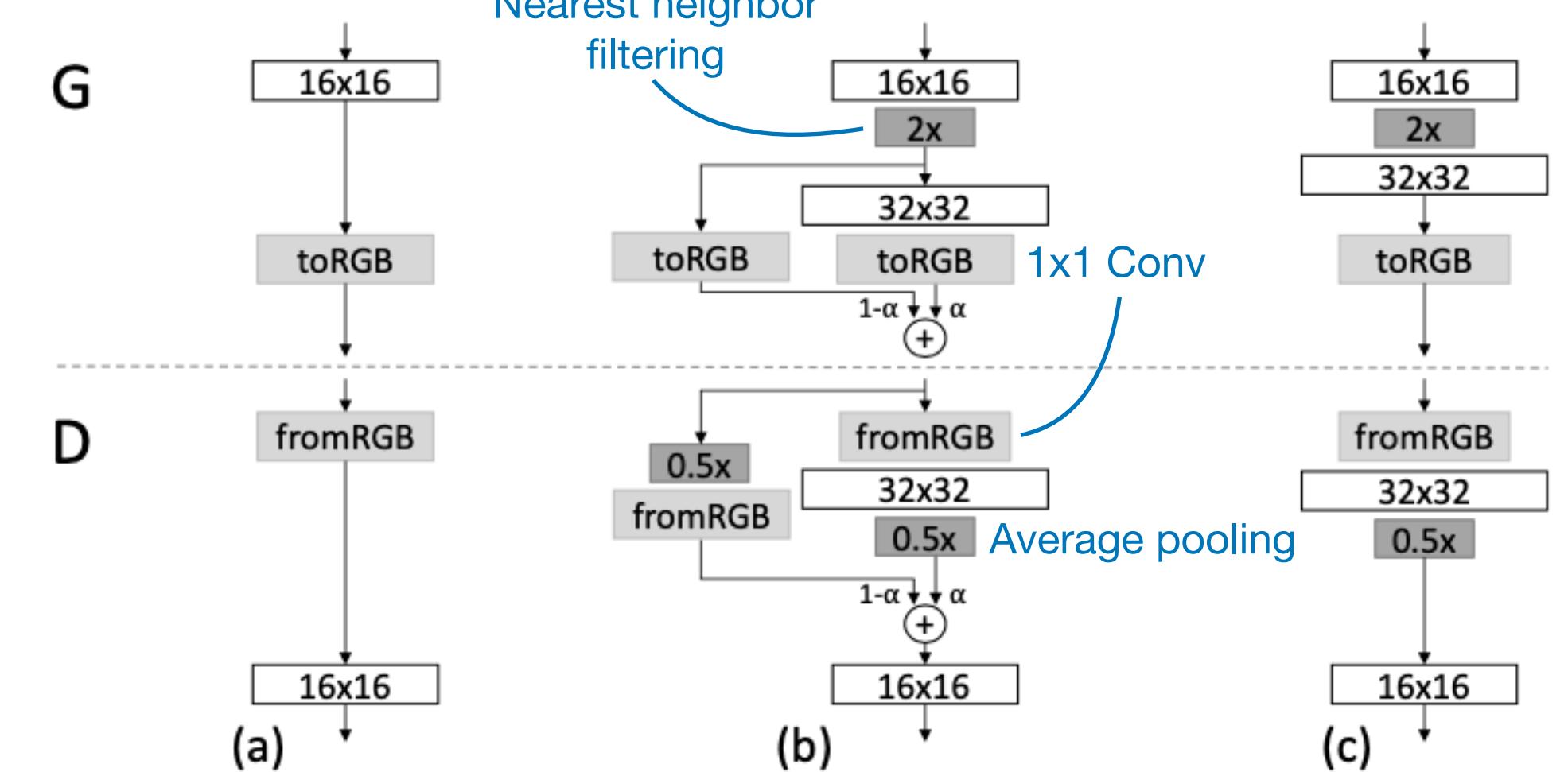
$w_i \sim \mathcal{N}(0, 1)$

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

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

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

Nearest neighbor filtering



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 Video](#)

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

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

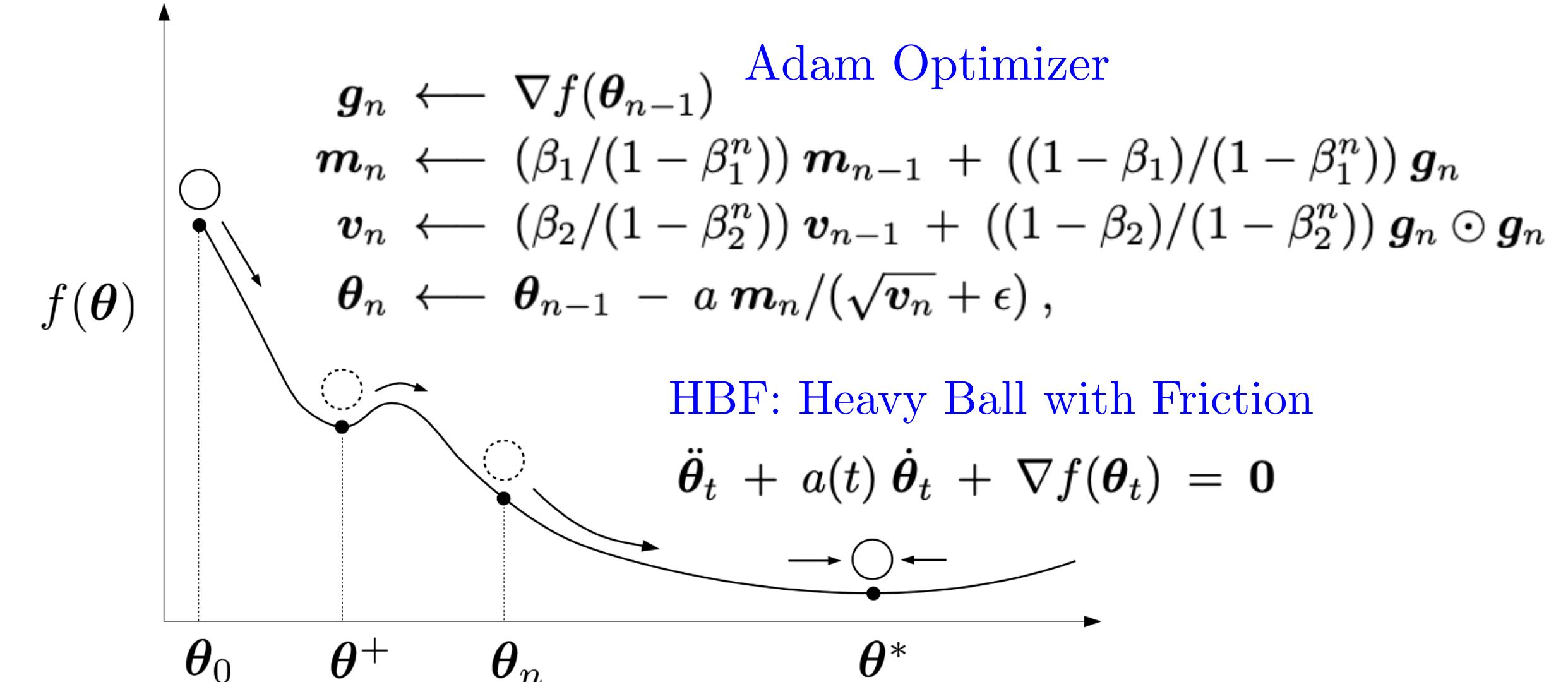
$$\begin{cases} \tilde{g}(\theta, w) = g(\theta, w) + M_w \\ \tilde{h}(\theta, w) = h(\theta, w) + M_\theta \end{cases} \rightarrow \text{random variables } M_w \text{ & } M_\theta$$

$b(n)$ → 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

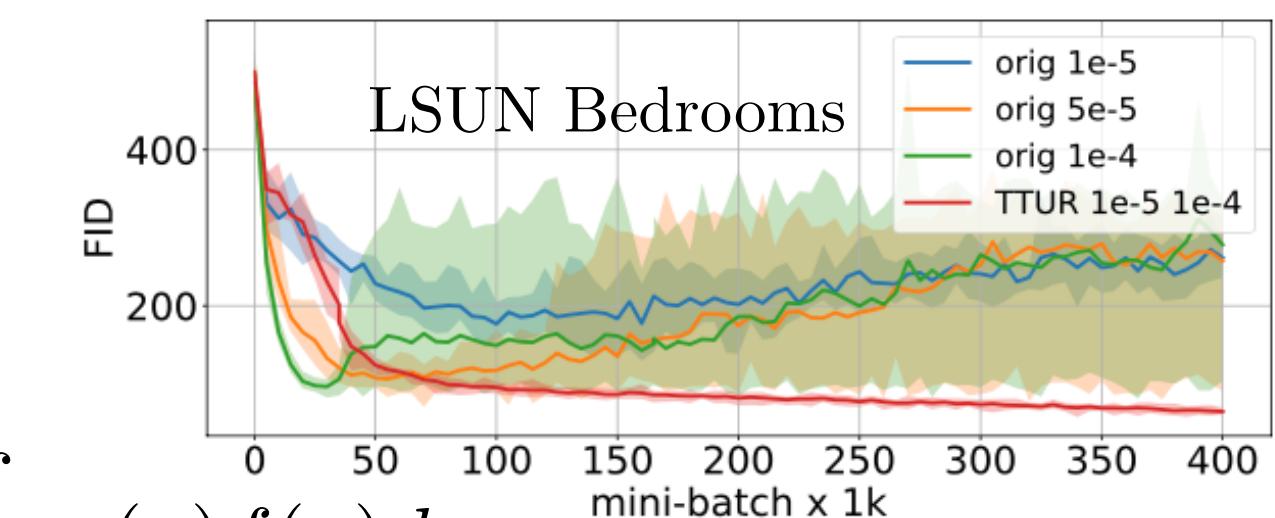
$$p_r(.) = p_g(.) \text{ iff } \int p_r(x) f(x) dx = \int p_g(x) f(x) dx$$

$f(x)$ → 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 Video](#)

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

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

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

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

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

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

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

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

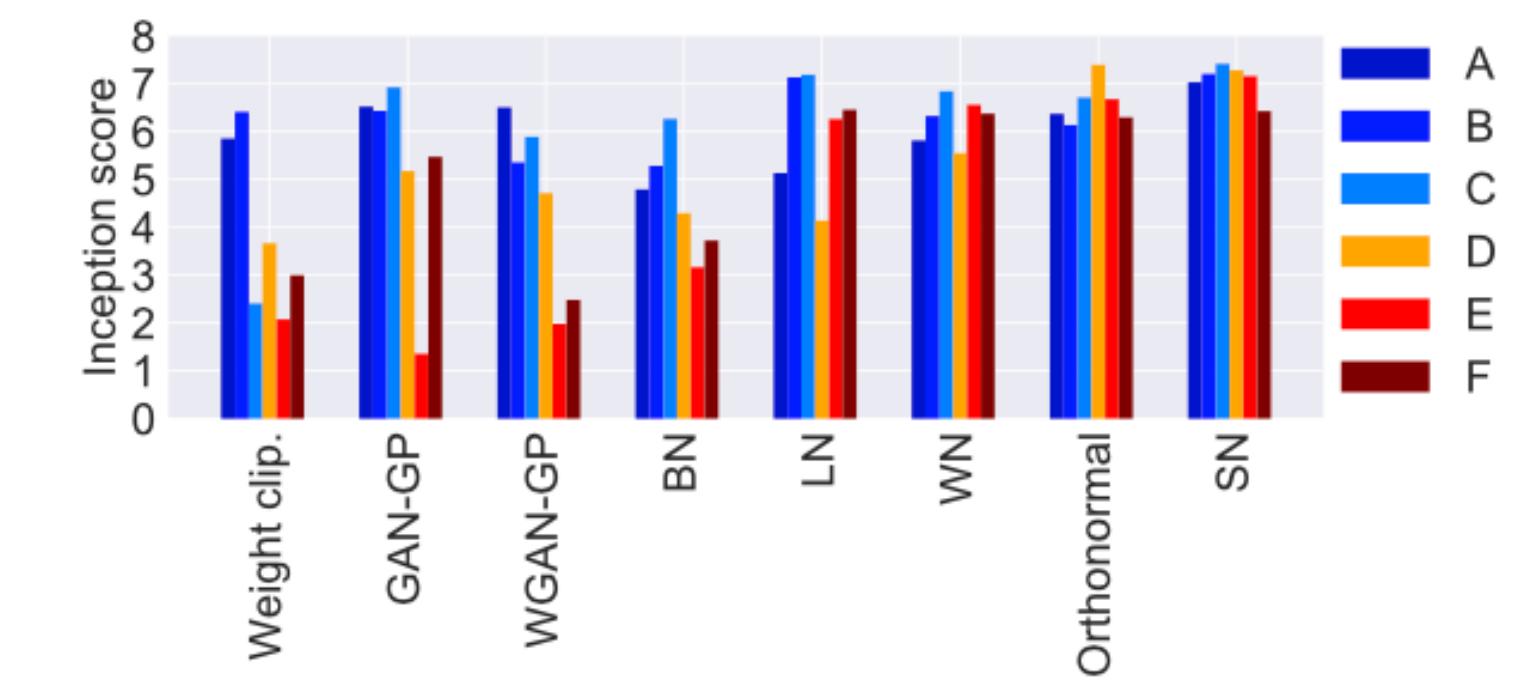
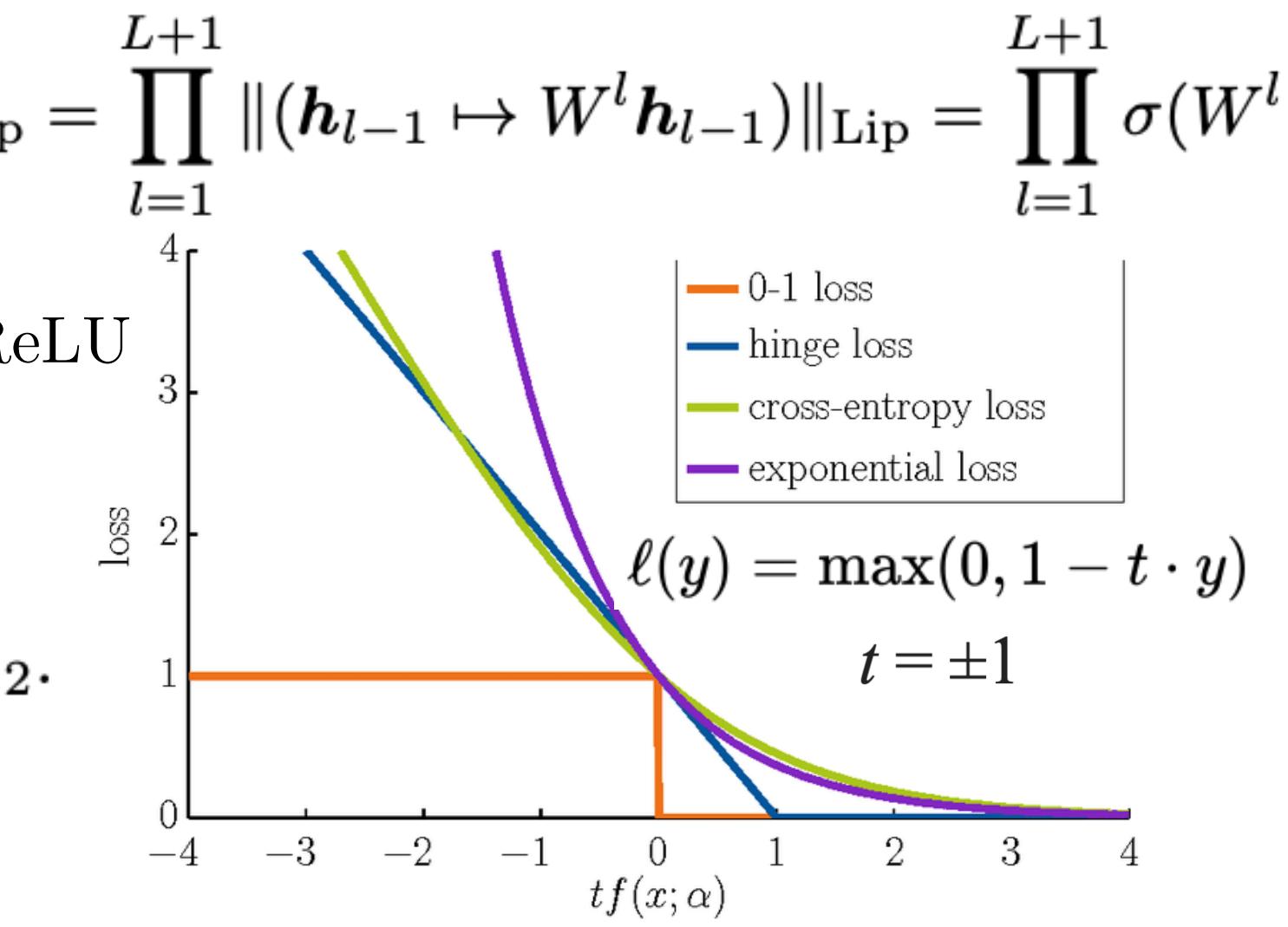
$$\sigma(W) \approx \tilde{\mathbf{u}}^T W \tilde{\mathbf{v}}.$$

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

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs



[YouTube Video](#)

The pix2pix baseline

$G \rightarrow$ generator

$D \rightarrow$ discriminator

$G \rightarrow$ translate semantic label maps to realistic looking images

$D \rightarrow$ distinguish real images from the translated ones

$\{(s_i, x_i)\} \rightarrow$ training data

$s_i \rightarrow$ semantic label map

$x_i \rightarrow$ corresponding natural photo

$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D)$

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{(s, x)}[\log D(s, x)] + \mathbb{E}_s[\log(1 - D(s, G(s)))]$$

U-Net as the generator

patch-based fully convolutional network as the discriminator

256×256

Improving photorealism and resolution 2048×1024

- coarse-to-fine generator
- multi-scale discriminator
- improved adversarial loss

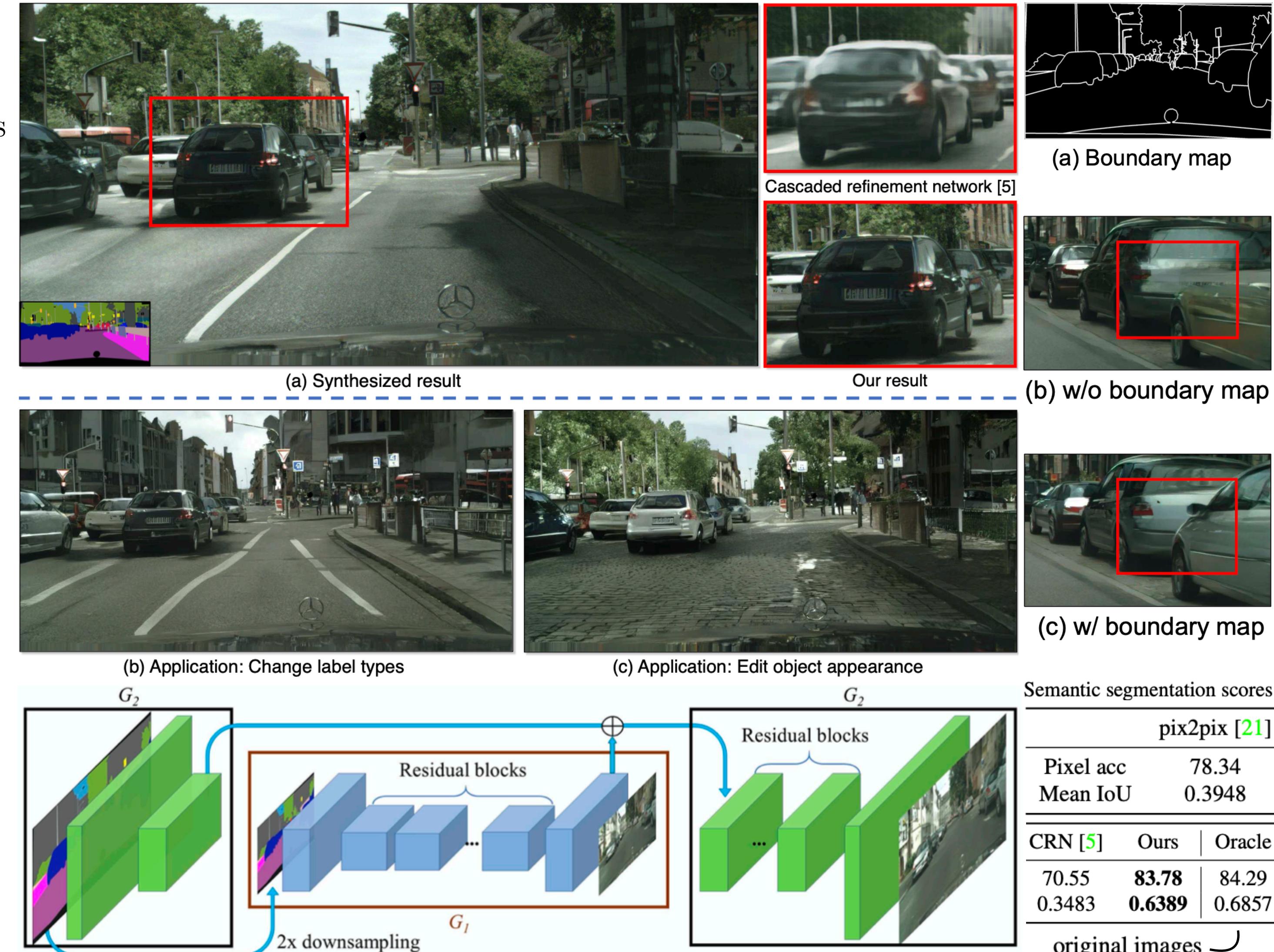
$$\min_G \left(\left(\max_{D_1, D_2, D_3} \sum_{k=1,2,3} \mathcal{L}_{\text{GAN}}(G, D_k) \right) + \lambda \sum_{k=1,2,3} \mathcal{L}_{\text{FM}}(G, D_k) \right)$$

$$\mathcal{L}_{\text{FM}}(G, D_k) = \mathbb{E}_{(\mathbf{s}, \mathbf{x})} \sum_{i=1}^T \frac{1}{N_i} \| | | D_k^{(i)}(\mathbf{s}, \mathbf{x}) - D_k^{(i)}(\mathbf{s}, G(\mathbf{s})) | |_1 \|$$

$D_k^{(i)} \rightarrow$ i -th layer feature extractor of discriminator D_k

$T \rightarrow$ total number of layers

$N_i \rightarrow$ number of elements in each layer



Semantic segmentation scores		
pix2pix [21]		

Pixel acc 78.34

Mean IoU 0.3948

CRN [5] Ours Oracle

70.55 83.78 84.29

0.3483 0.6389 0.6857

original images



Boulder

Large Scale GAN Training for High Fidelity Natural Image Synthesis



[YouTube Video](#)

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

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

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

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



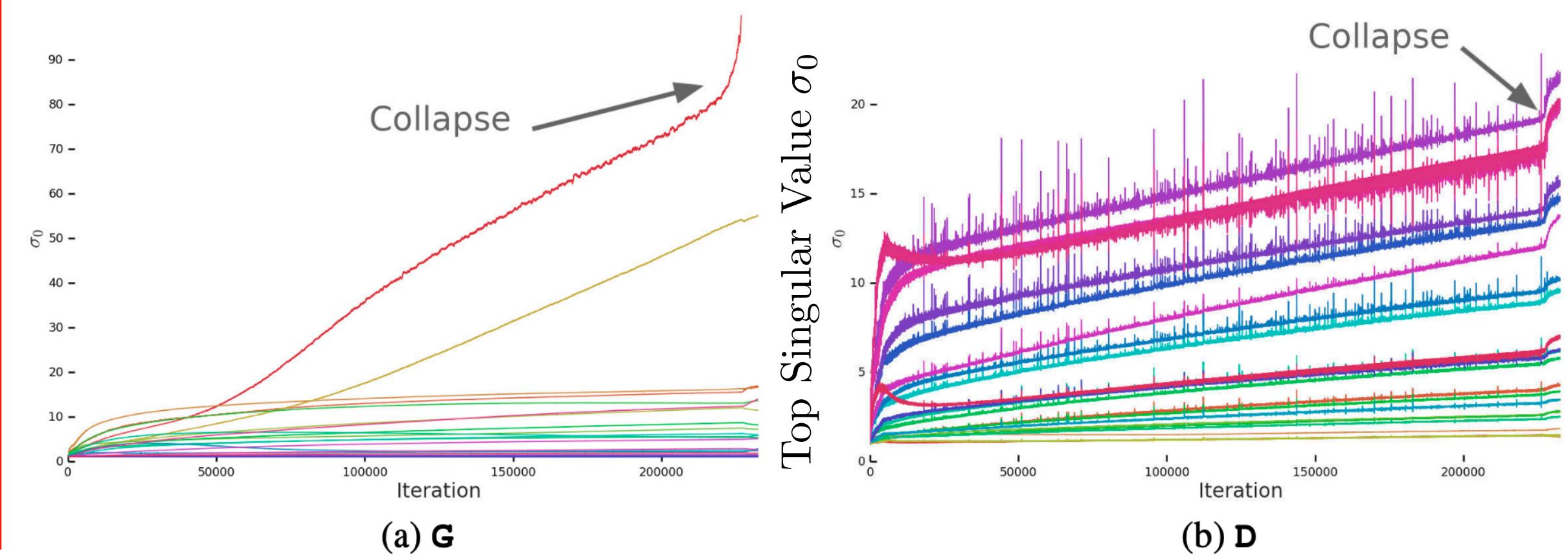
Saturation artifacts from applying truncation to a poorly conditioned model!



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)



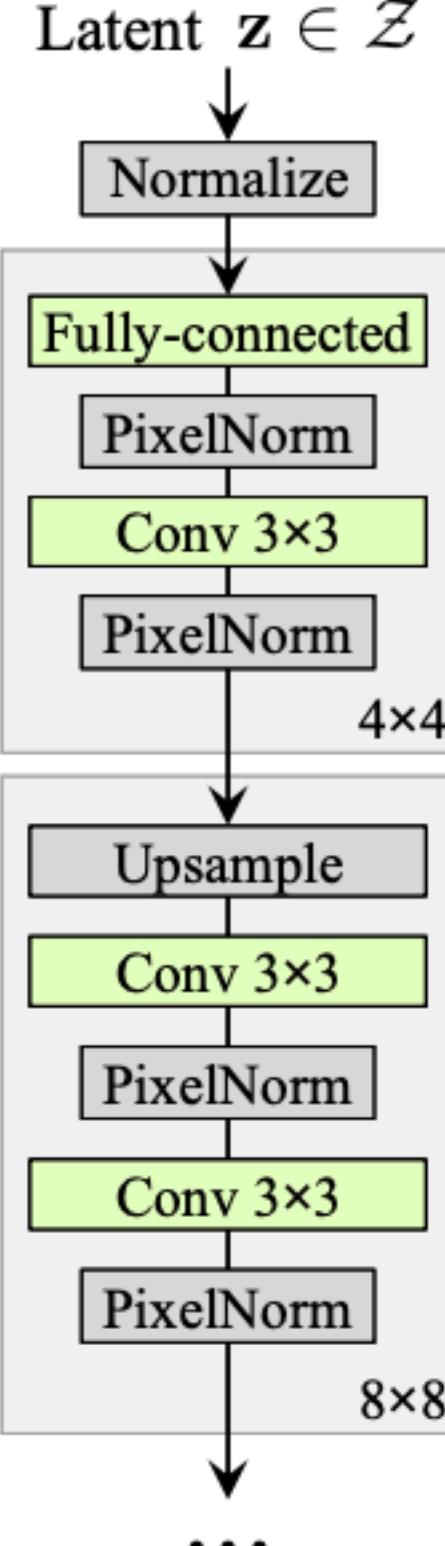


Boulder

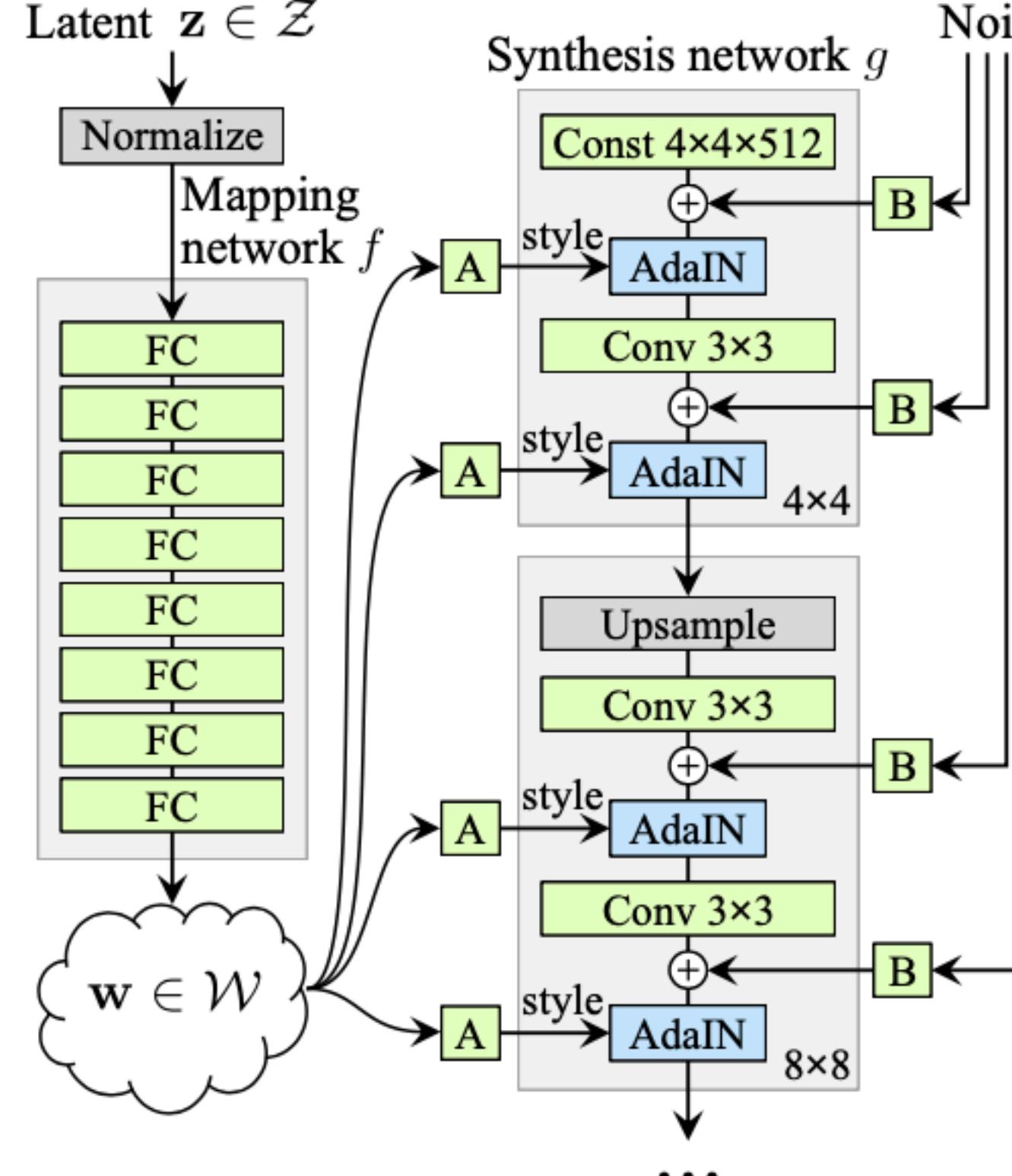
A Style-Based Generator Architecture for Generative Adversarial Networks



[YouTube Video](#)



(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



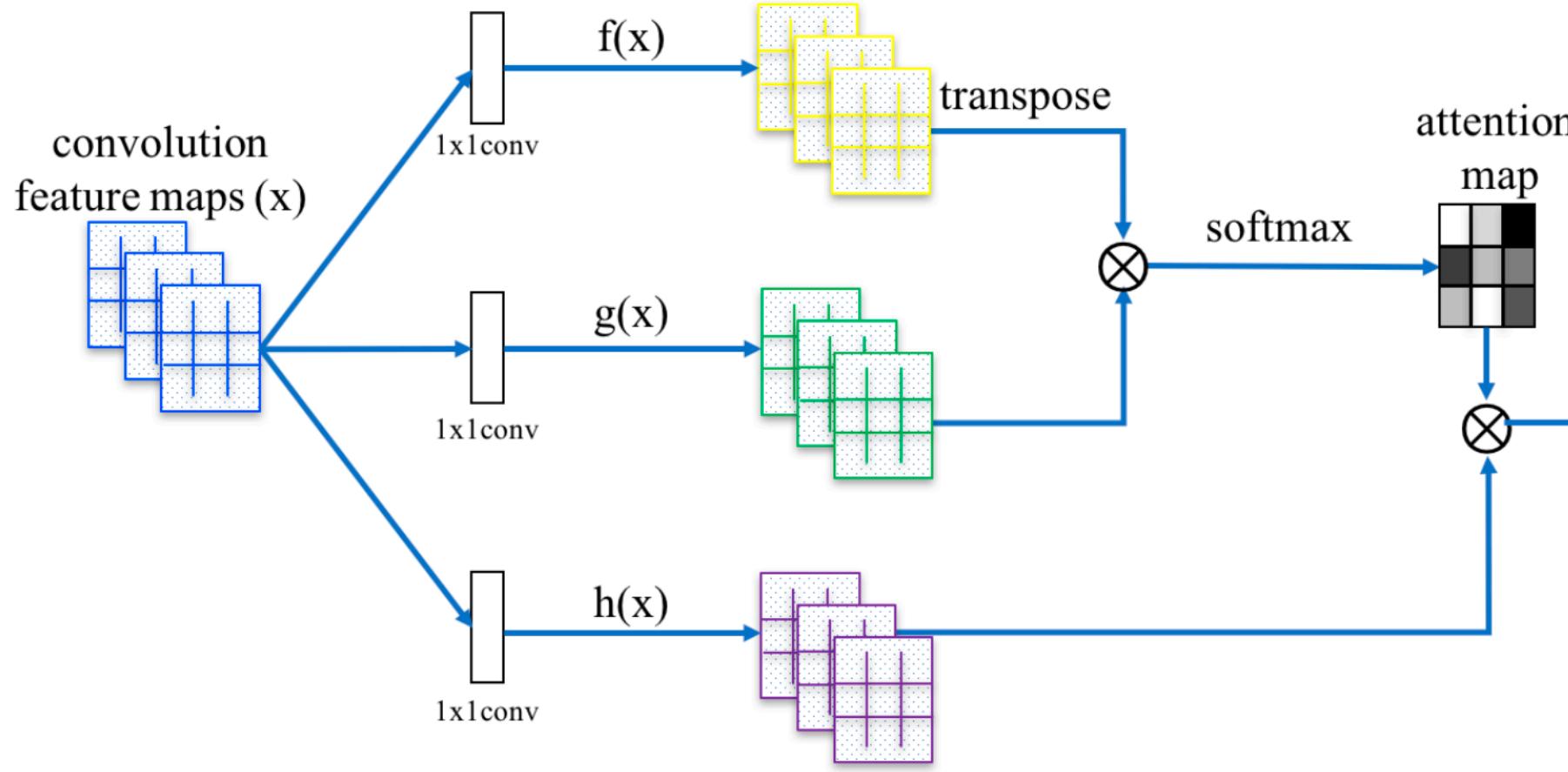


Boulder



[YouTube Video](#)

Self-Attention Generative Adversarial Networks



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

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

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

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

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

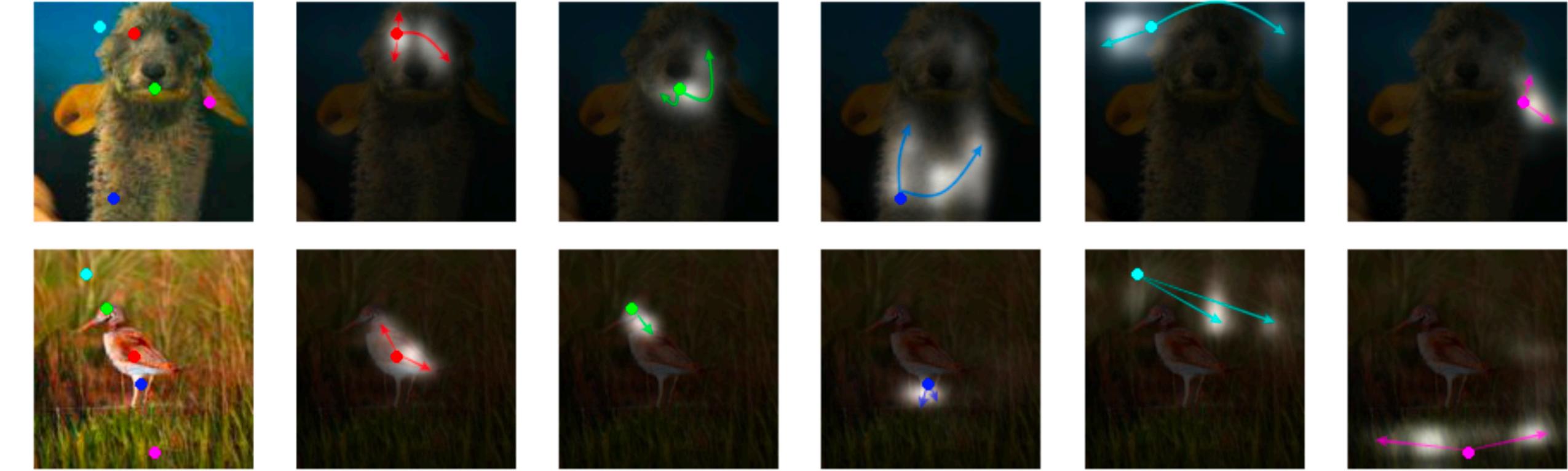
the extent to which the model attends to the i -th location when synthesizing the j -th region

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

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

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

$$y_i = \gamma o_i + x_i$$



hinge version of the adversarial loss

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

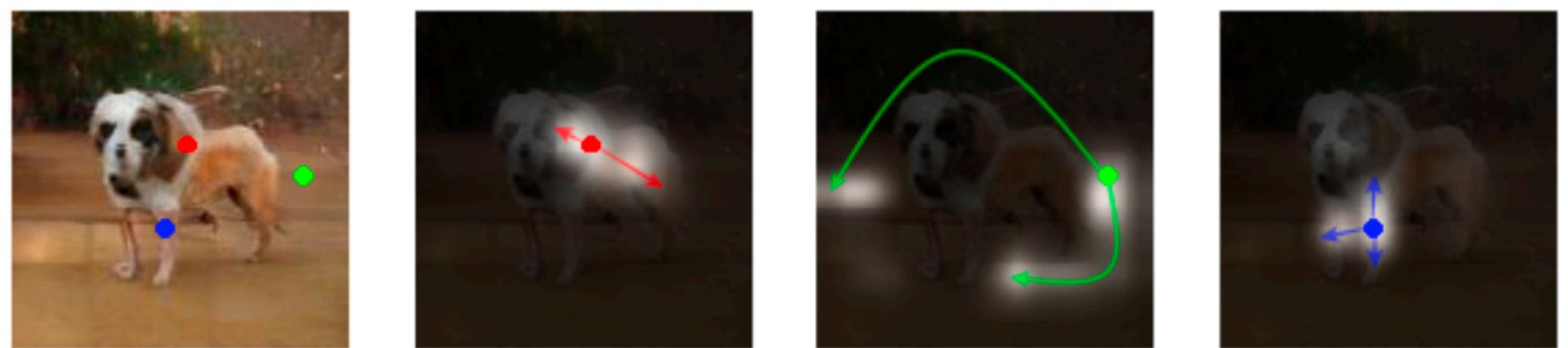
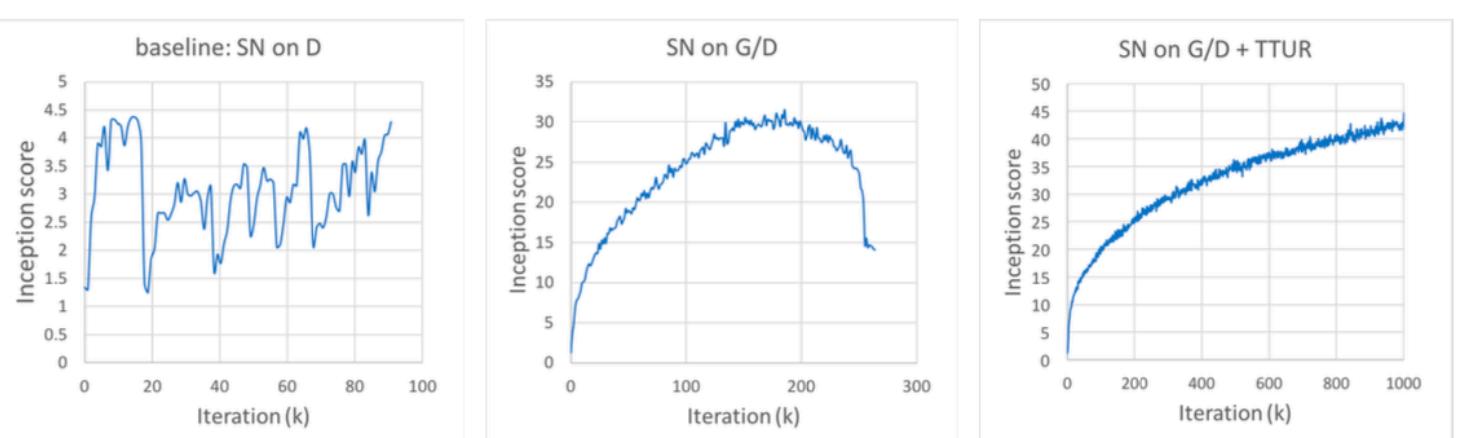
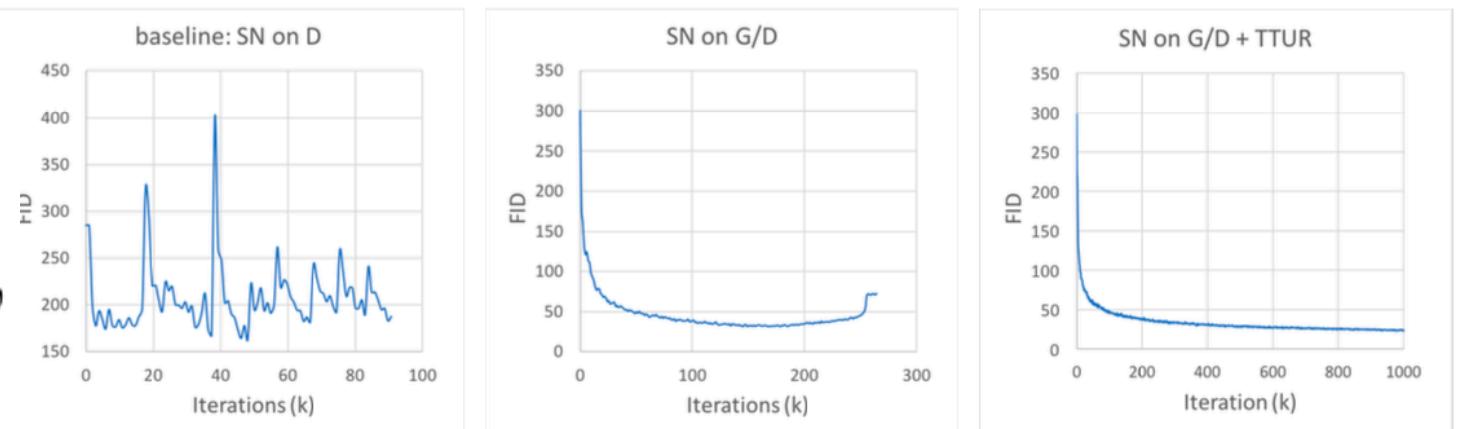
$$L_G = -\mathbb{E}_{z \sim p_z, y \sim p_{data}} D(G(z), y),$$

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)

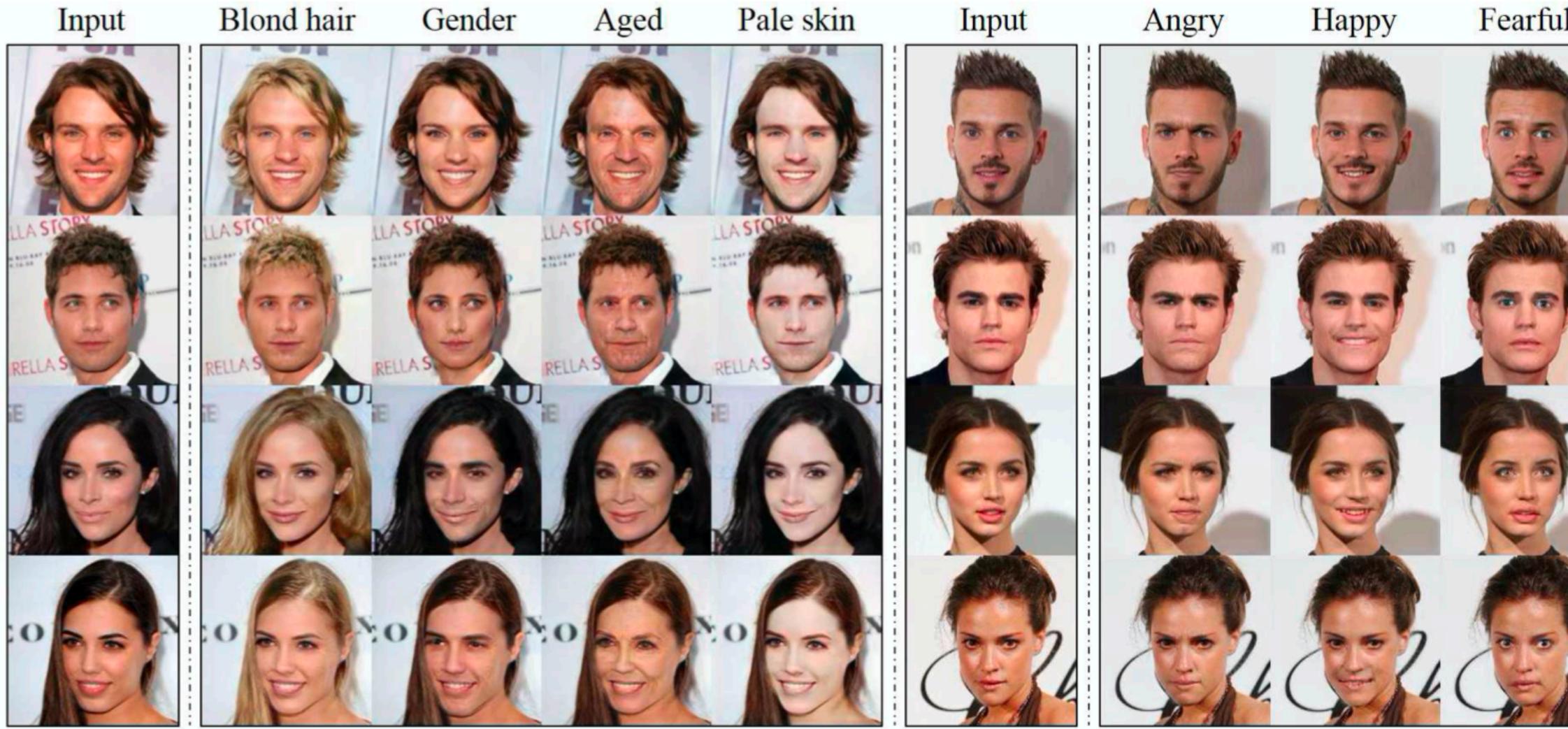




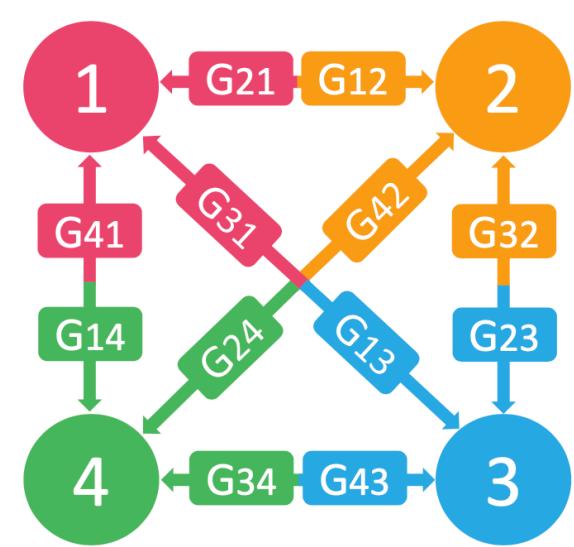
StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation



[YouTube Video](#)



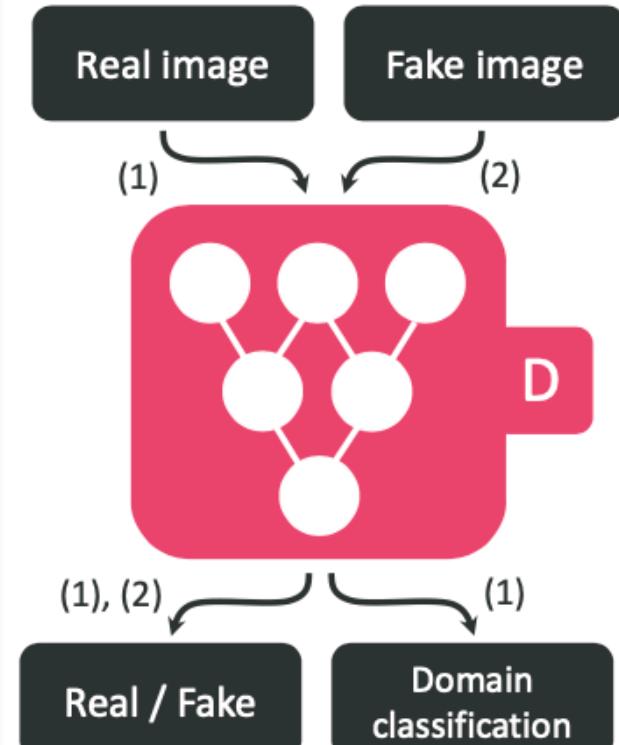
(a) Cross-domain models



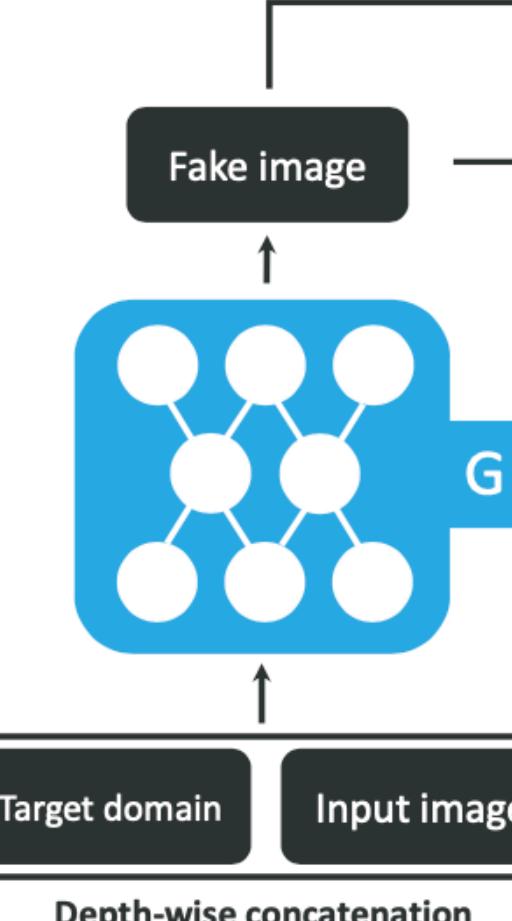
$$\begin{aligned} \mathcal{L}_{adv} &= \mathbb{E}_y[\log D_{src}(y)] + \mathbb{E}_{x,c}[\log(1 - D_{src}(G(x, c)))] \rightarrow \text{adversarial loss} \\ \mathcal{L}_{cls}^r &= \mathbb{E}_{y,c}[-\log D_{cls}(c|y)] \rightarrow \text{domain classification loss of real images} \\ \mathcal{L}_{cls}^f &= \mathbb{E}_{x,c}[-\log D_{cls}(c|G(x, c))] \rightarrow \text{fake images} \\ \mathcal{L}_{rec} &= \mathbb{E}_{x,c,c'}[\|x - G(G(x, c), c')\|_1] \rightarrow \text{reconstruction loss} \\ \lambda_{cls} &= 1 \& \lambda_{rec} = 10 \end{aligned}$$

Choi, Yunjey, et al. "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

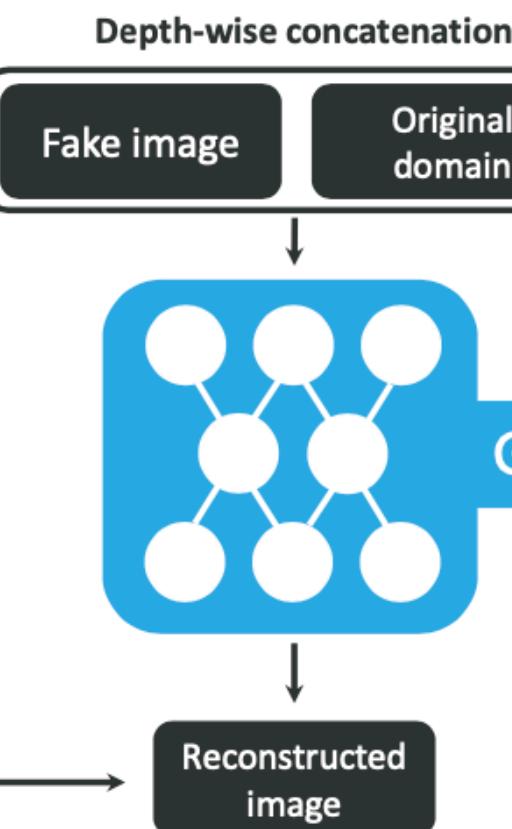
(a) Training the discriminator



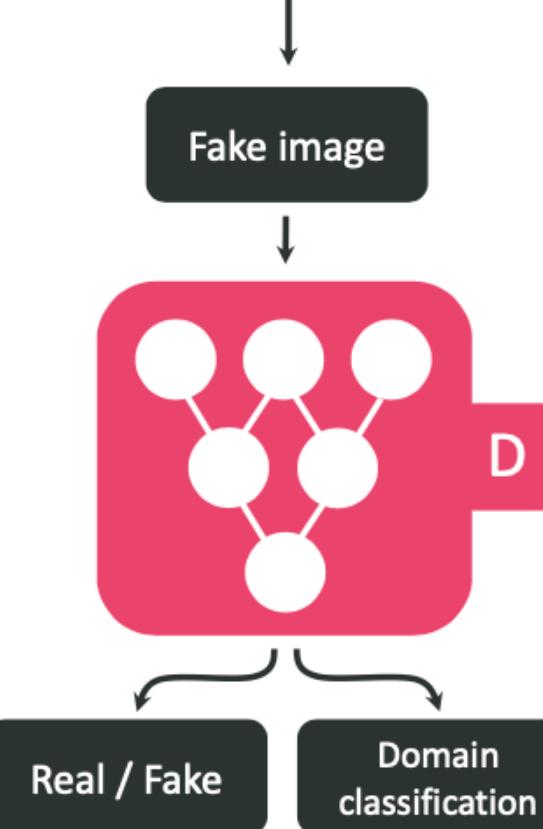
(b) Original-to-target domain



(c) Target-to-original domain



(d) Fooling the discriminator



Training with multiple datasets

$n \rightarrow$ number of datasets (e.g., $n = 2$)

CelebA and RaFD

$m \rightarrow$ mask vector (n -dimensional one-hot vector)

$\tilde{c} := [c_1, \dots, c_n, m] \rightarrow$ concatenate

$c_i \rightarrow$ vector for labels of the i -th dataset
assign zero values to the remaining $n - 1$ unknown labels

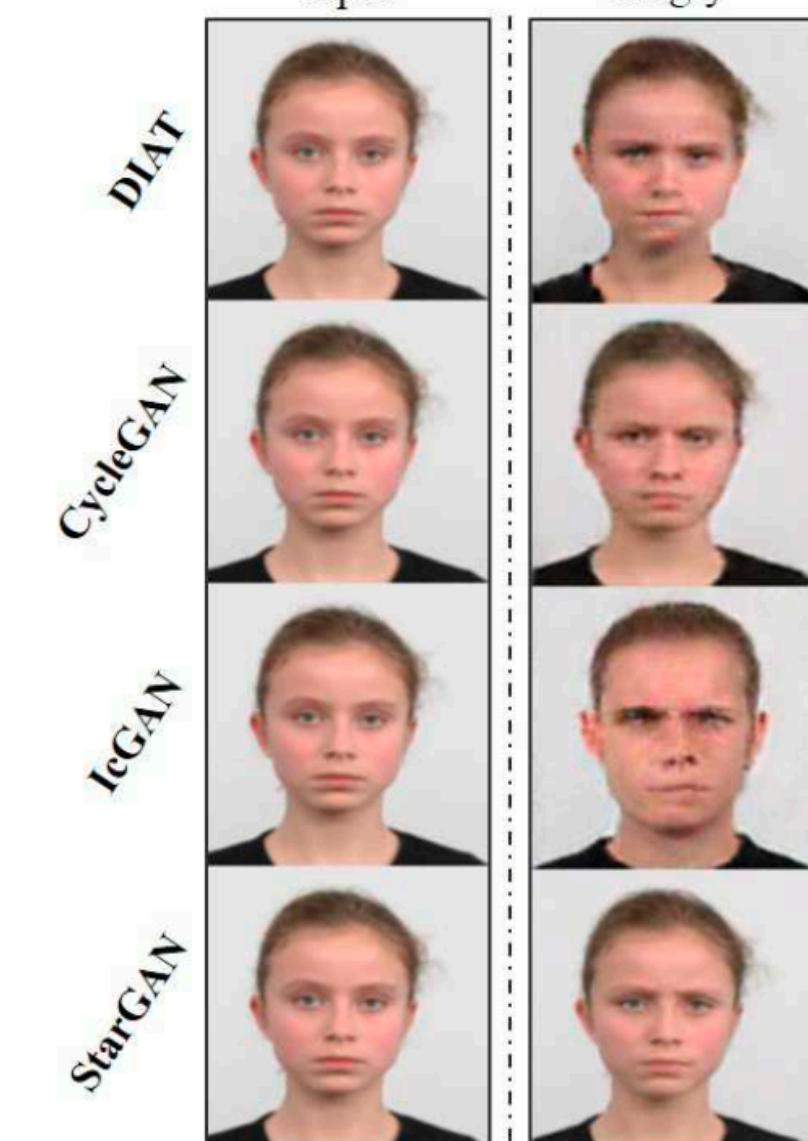
Improved GAN training

$$\mathcal{L}_{adv} = \mathbb{E}_y[D_{src}(y)] - \mathbb{E}_{x,c}[D_{src}(G(x, c))]$$

$$- \lambda_{gp} \mathbb{E}_{\hat{y}}[(\|\nabla_{\hat{y}} D_{src}(\hat{y})\|_2 - 1)^2]$$

$\hat{y} \rightarrow$ sampled uniformly along a straight line between y and $G(x, c)$

$$\lambda_{gp} = 10 \rightarrow \text{gradient penalty}$$





Analyzing and Improving the Image Quality of StyleGAN

Boulder

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

$f \rightarrow$ mapping network

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

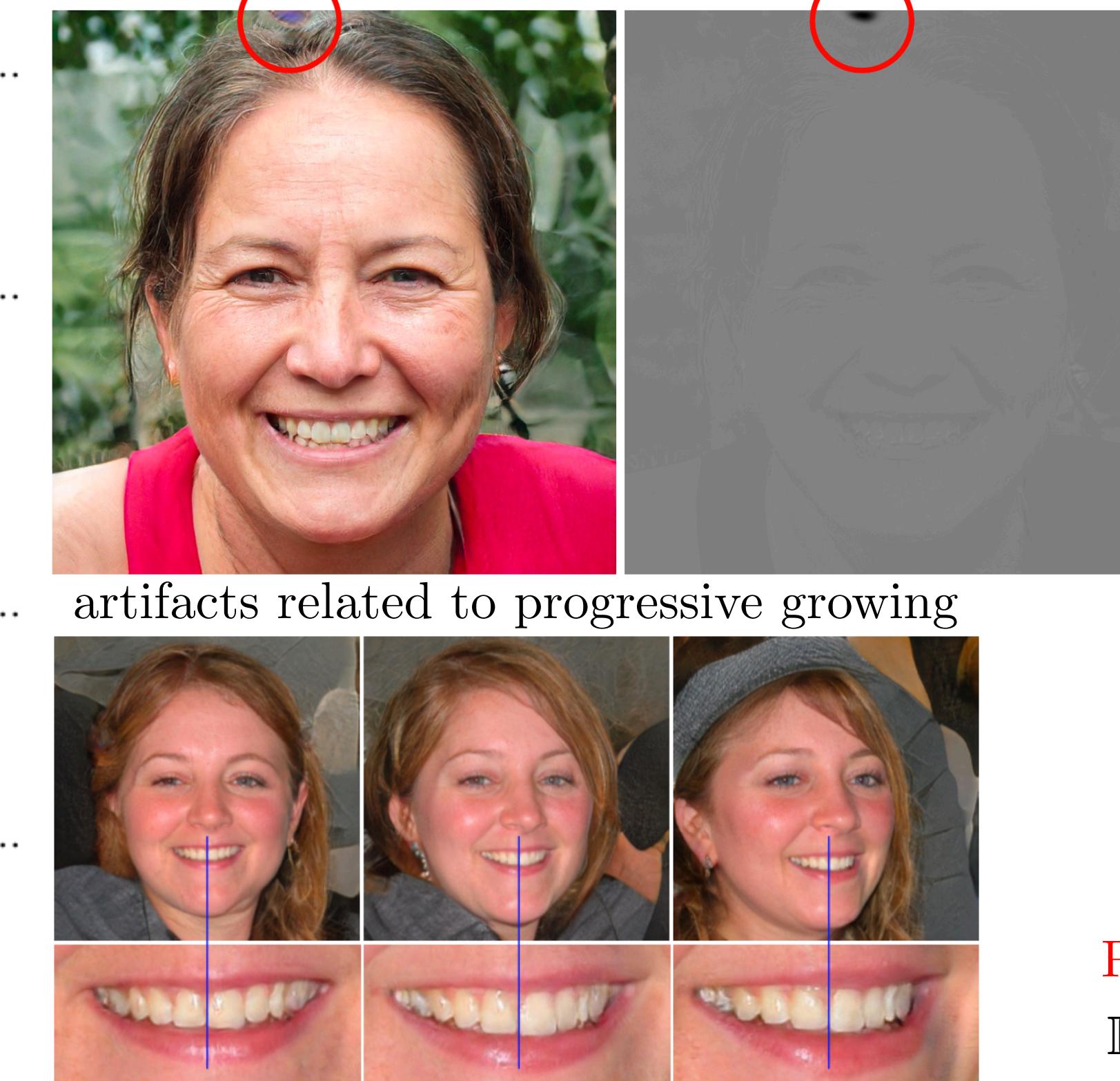
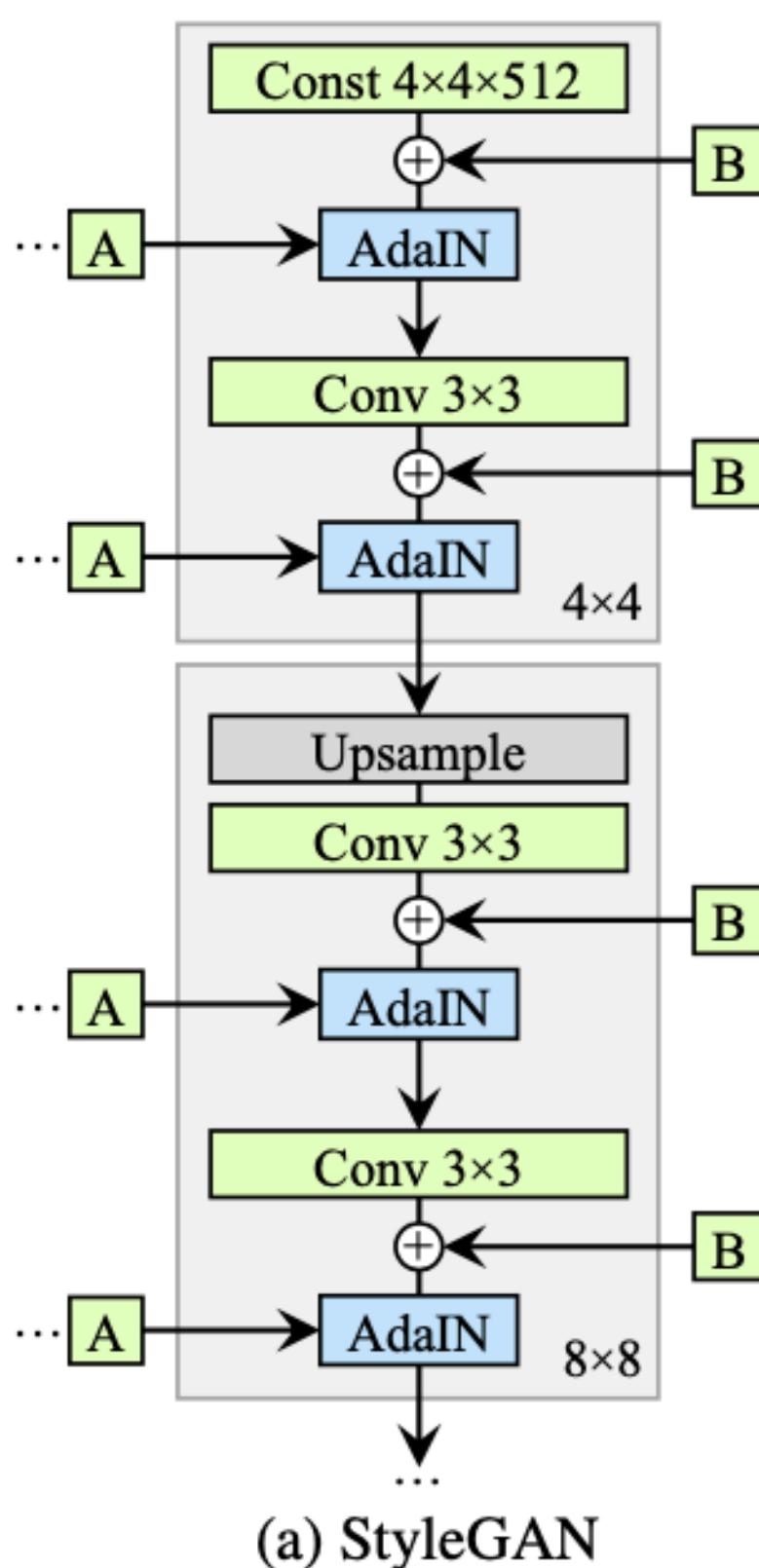
affine transformations \rightarrow styles

$g \rightarrow$ synthesis network

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

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

blob-shaped artifacts that resemble water droplets



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

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

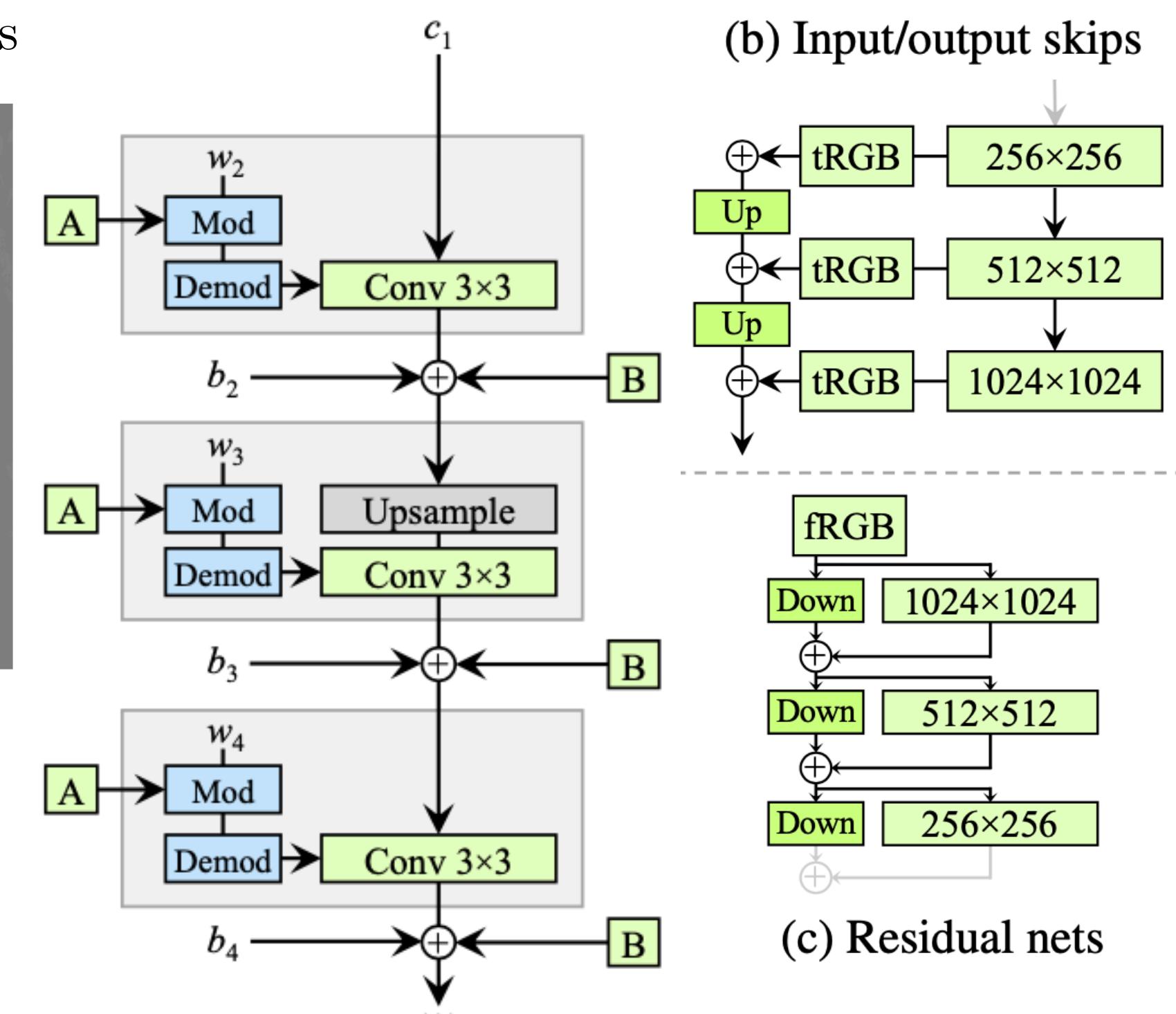
$w, w', w'' \rightarrow$ original, modulated, and demodulated weights

$s_i \rightarrow$ 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

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

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

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

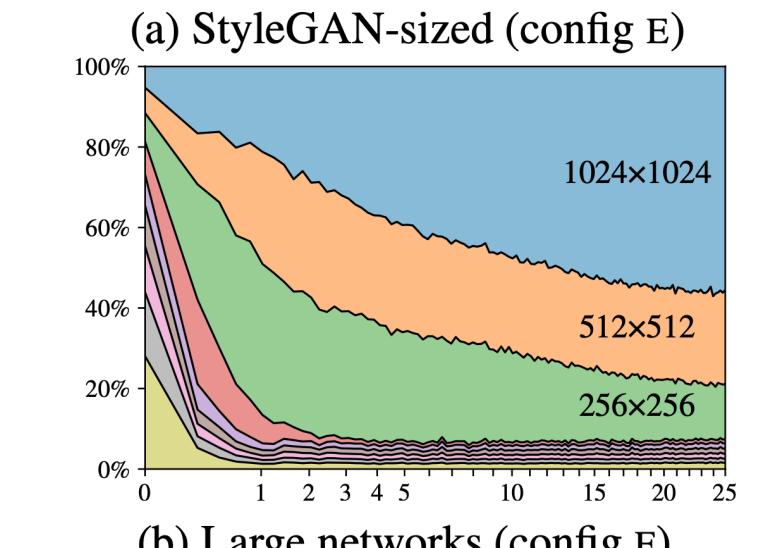
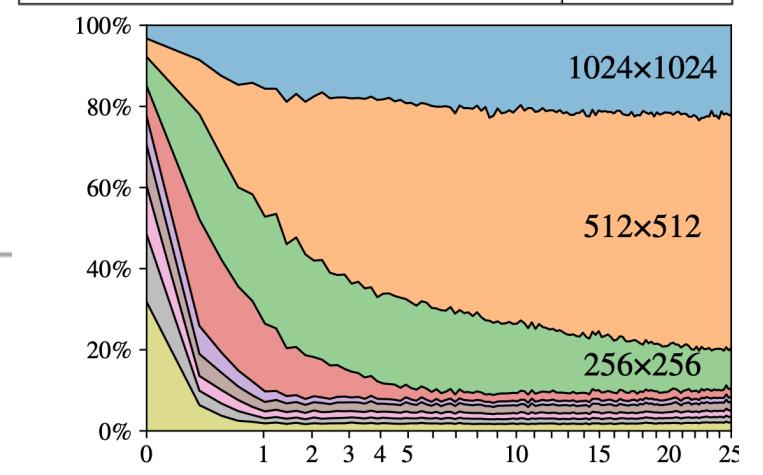
$$\mathbf{w} \sim f(\mathbf{z})$$

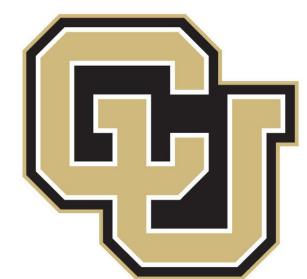
$a \rightarrow$ exponential moving average of $\|\mathbf{J}_{\mathbf{w}}^T \mathbf{y}\|_2$

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

FFHQ, 1024x1024

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





Boulder



Questions?

[YouTube Playlist](#)
