DDA4220 · Fall 2026

Lecture 16

Deep Learning and Applications

Generative adversarial networks

Zhen Liu

CUHK-Shenzhen

Lecture outline

01 Learning from a discriminator

02 Why GAN training is difficult

03 Conditional generation and image translation

04 Style-based generators

05 Adversarial losses in other models

06 Evaluating generators

* denotes optional slides

Generated faces, 2014–2018



2014 GAN



2015 DCGAN



2017 Progressive GAN



2018 StyleGAN

Each face is generated from a random vector. None of these people exist.

From 48×48 grayscale to 1024×1024 color in four years

Source: GAN (2014) · DCGAN (2015) · Progressive GAN (2017) · StyleGAN (2018)

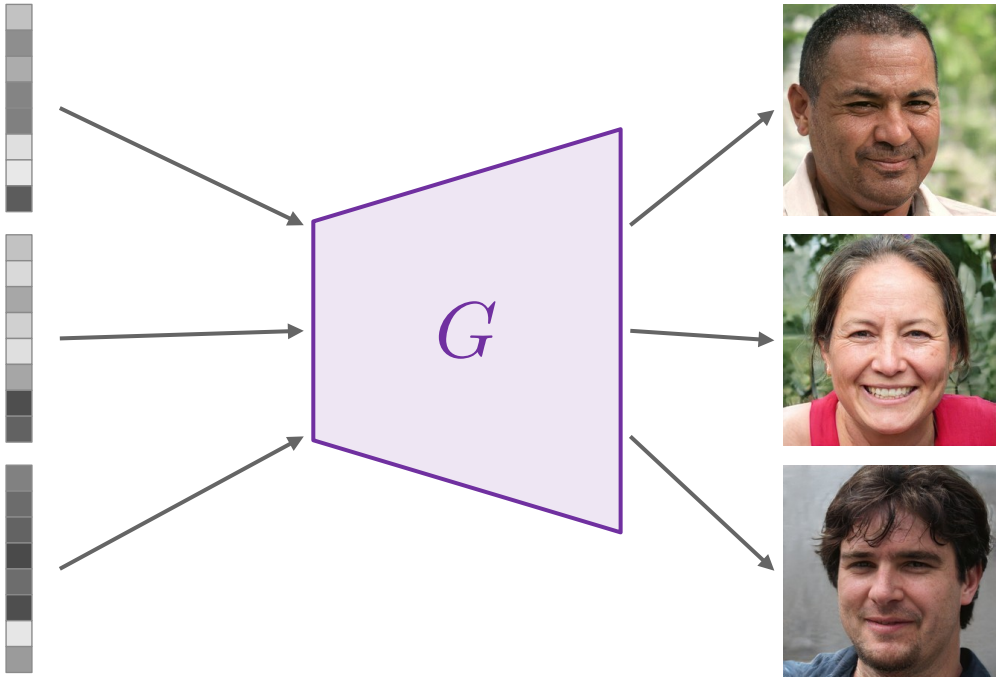
A generator turns random codes into images

random code

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

generated image

$\tilde{x} = G(z)$



G is like the VAE decoder (L15), trained differently

training set: real photographs

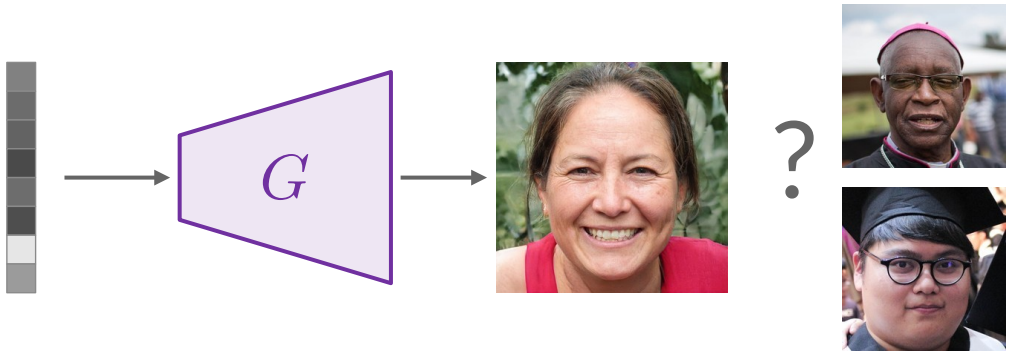


Goal: new samples that look like the training set: realistic and varied

Source: StyleGAN (2018) · FFHQ dataset

Which loss should train the generator?

No target for a random code



Which photo should $G(z)$ match?

Averaging plausible answers blurs

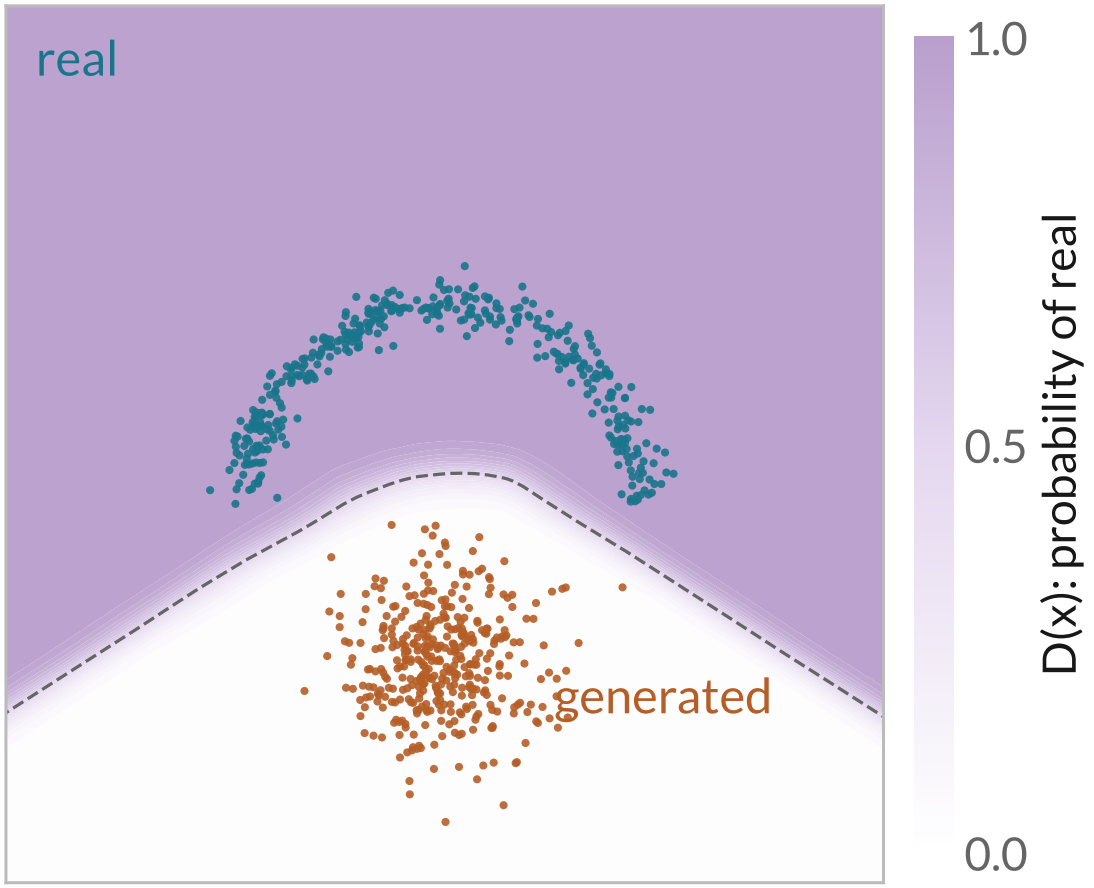


plausible target 1 plausible target 2 pixelwise mean

Squared error prefers the mean (L15)

Need a loss that compares the **collection** of generated images with the **collection** of real images

A classifier can detect a difference



Toy 2-D data:

real crescent, **generated** blob from an untrained G

Train a classifier D; shading $= D(x)$

Same distributions $\rightarrow D(x) = \frac{1}{2}$ everywhere

Here D separates them easily: a detectable difference

Step 1: fix the generator, train the discriminator

The labels are free: we know where each image came from



label 1



label 0

$$L_D = -\frac{1}{m}\sum_{i=1}^{m}\log D(x_i) \; - \; \frac{1}{m}\sum_{i=1}^{m}\log\big(1 - D(G(z_i))\big)$$

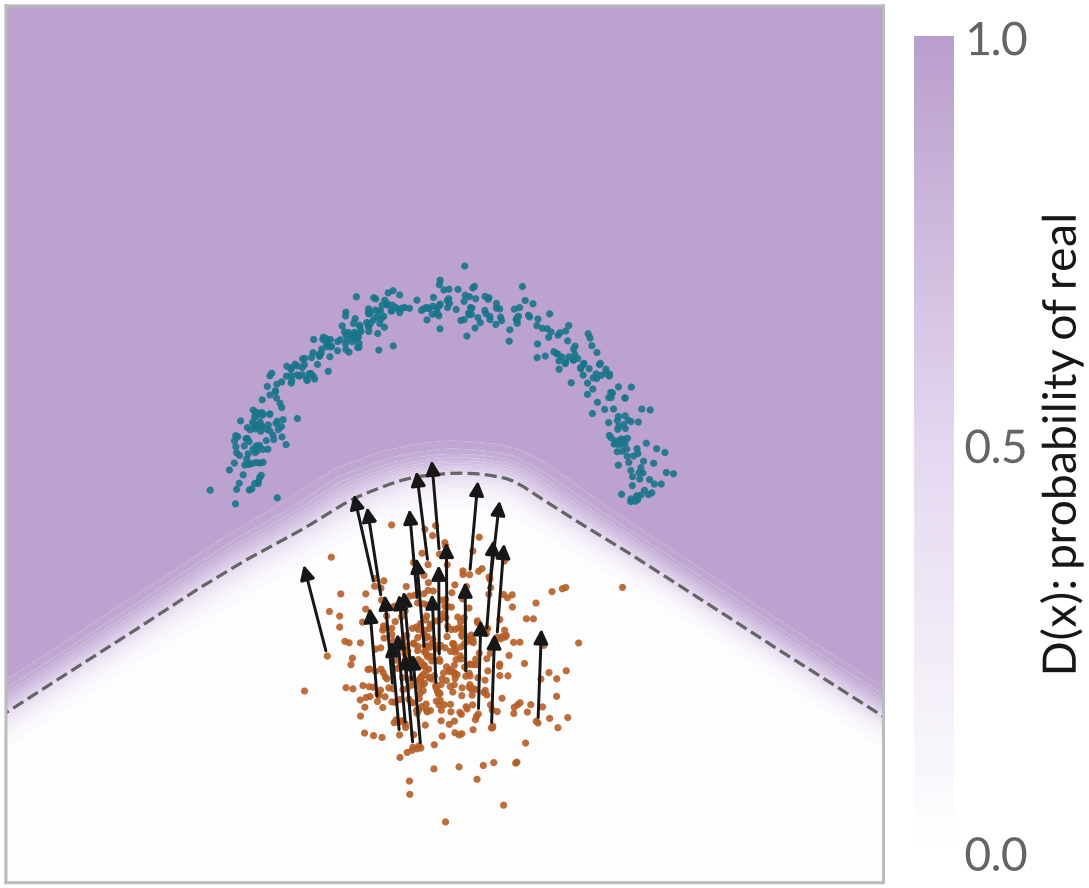
real: $D(x) \to 1$

generated: $D(G(z)) \to 0$

Binary cross-entropy (L02), with the source as the label

Only D's weights change; m: minibatch size

Step 2: move samples toward higher scores



Gradient of D's score with respect to its input:

$$\nabla_x \log D(x)\Big|_{x=\tilde{x}}$$

Arrows point toward a higher "real" score

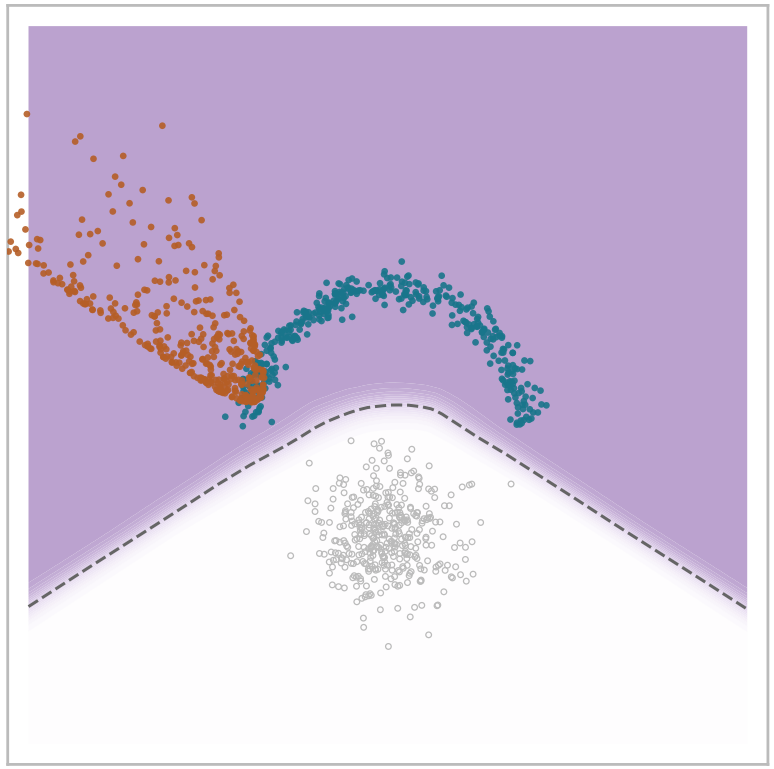
The chain rule carries it into G:

$$\frac{\partial L}{\partial \theta_G} = \frac{\partial L}{\partial \tilde{x}} \frac{\partial \tilde{x}}{\partial \theta_G}$$

Same input gradient as adversarial examples (L07)

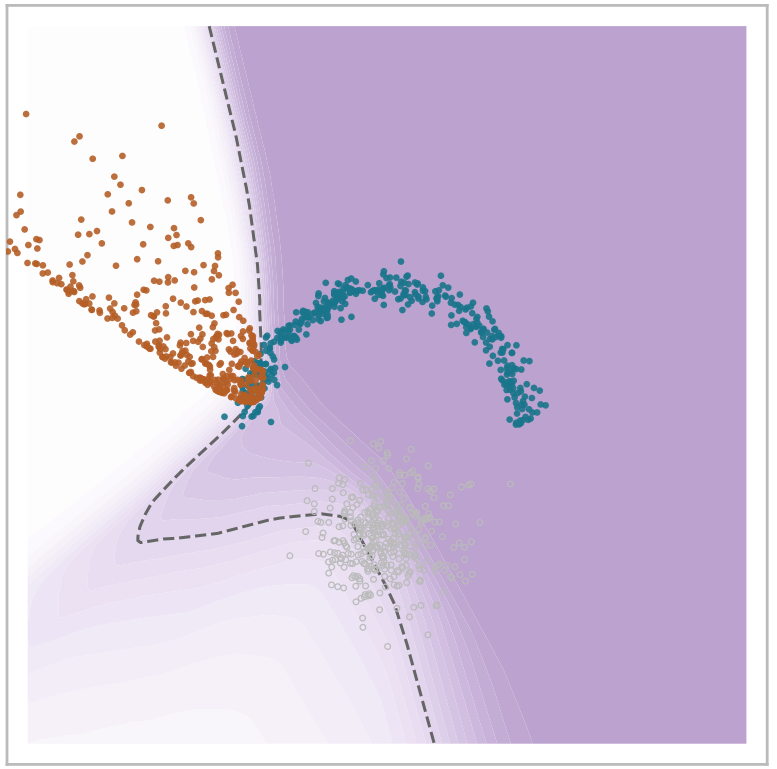
Step 3: the old discriminator becomes outdated

Old D, after G moves its samples



G after 60 steps against a fixed D

D retrained on the new samples

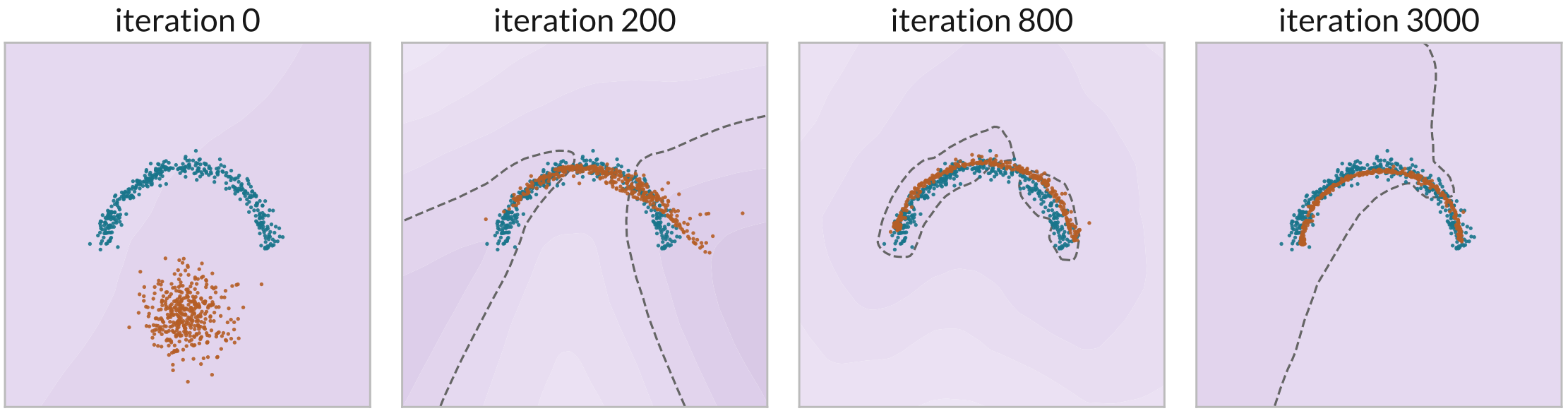


Retrained D: the boundary moves

Neither network can be trained once and frozen

Alternating updates

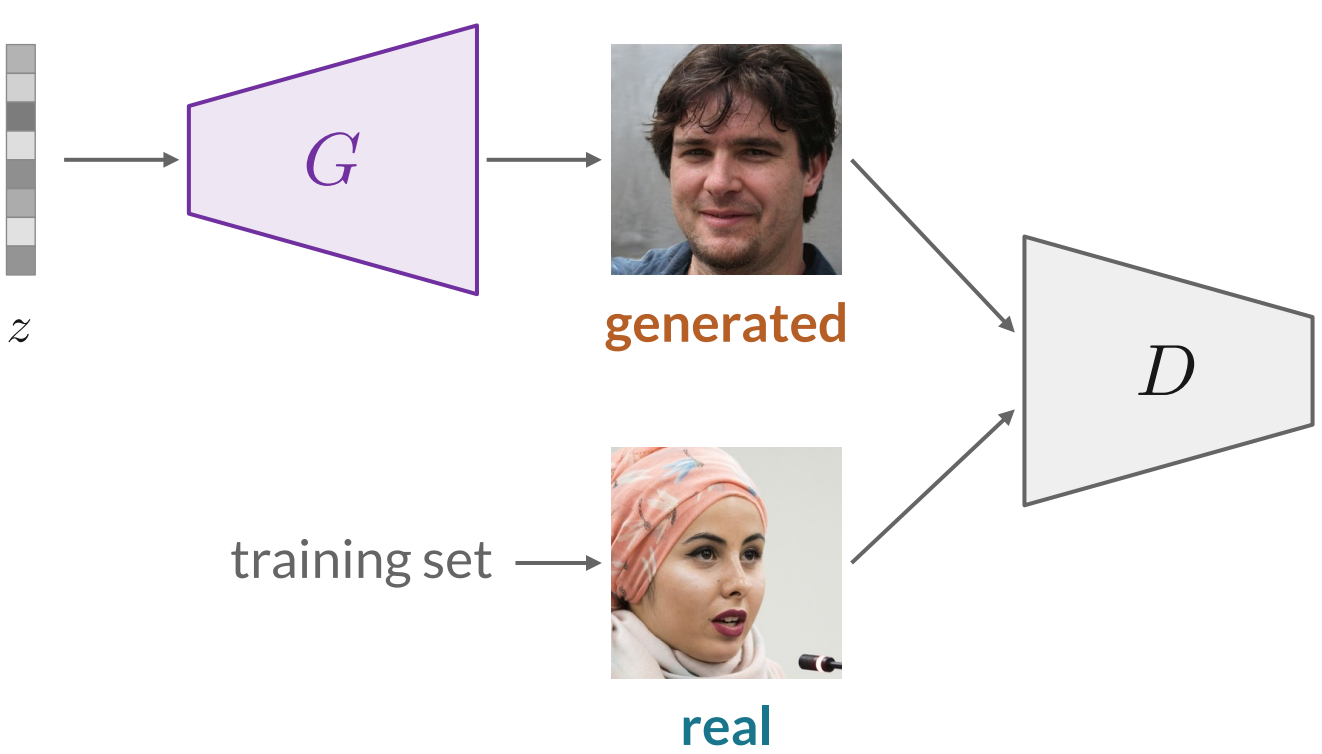
Repeat: one D step, then one G step. Both start untrained.



Iteration 3,000: samples cover the crescent and $D \approx \frac{1}{2}$ everywhere

Dashed: $D(x) = \frac{1}{2}$. No target point for any code z.

Generative adversarial networks (GANs)



D: classify the source correctly

G: make D call its images real

After training only G is kept: one forward pass per sample

Source: GAN (2014)

One value function, two players

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

$$V(D,G) = \underbrace{\mathbb{E}_{x \sim p_{\text{data}}} \left[\log D(x)\right]}_{\text{real images}} + \underbrace{\mathbb{E}_{z \sim p(z)} \left[\log\left(1 - D(G(z))\right)\right]}_{\text{generated images}}$$

D **maximizes** V

Same as minimizing L_D from step 1

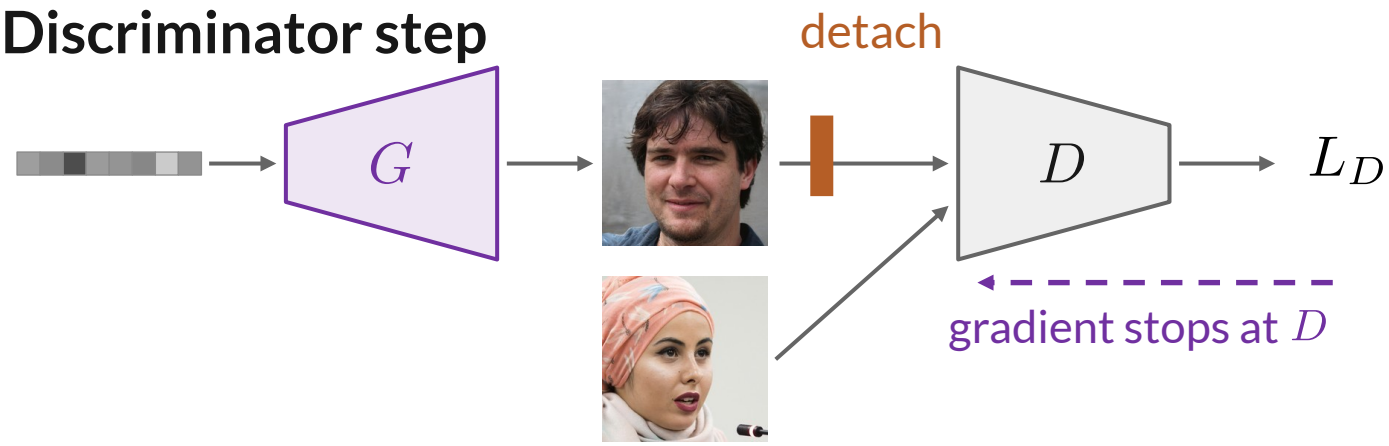
G **minimizes** V

Only the second term depends on G

$\mathbb{E}$: average, estimated with a minibatch

Which parameters each step updates

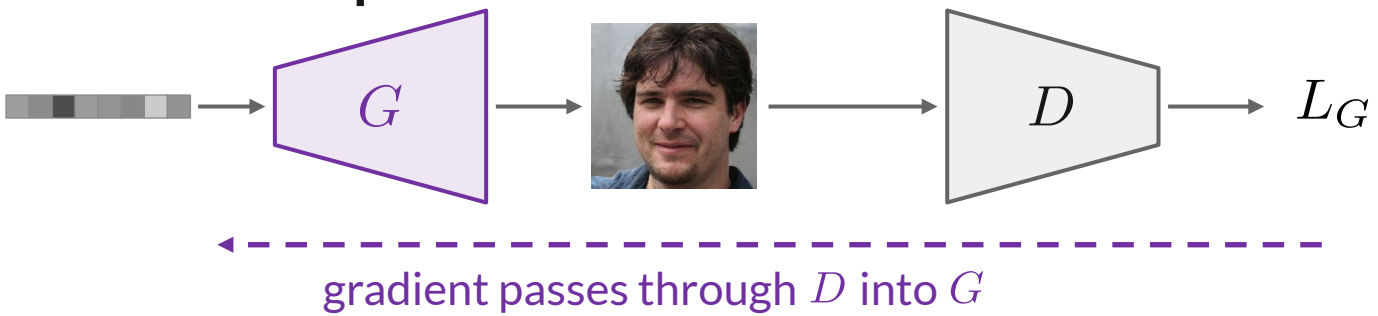
Discriminator step



Update D only

Generated image detached: no gradient reaches G

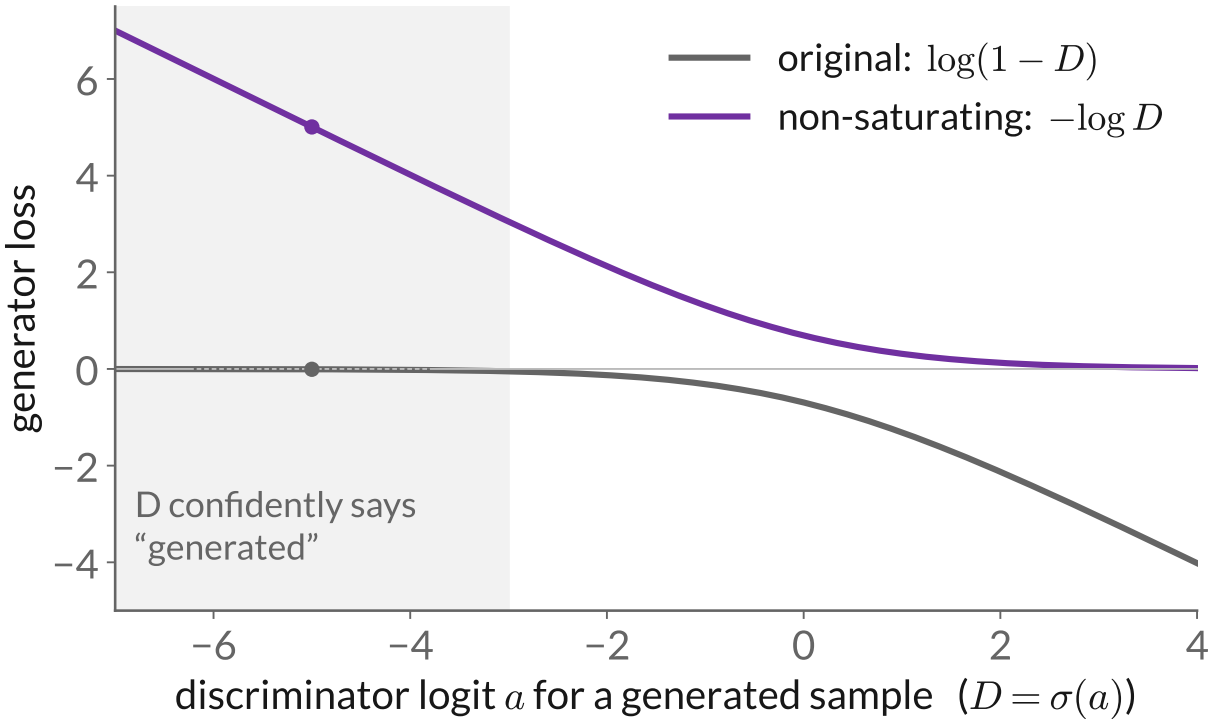
Generator step



Update G only

D's weights fixed, but its computation stays in the graph

Weak generator gradients early in training



Early on D rejects generated images: logit $a \ll 0$, $D = \sigma(a) \approx 0$

Original G loss, slope in the logit:

$$\frac{\partial}{\partial a} \log\big(1 - \sigma(a)\big) = -\sigma(a) \approx 0$$

Flat exactly where G is worst: almost no signal

Dots: slopes at $a = -5$

The non-saturating generator loss

Train G with its samples labeled real:

$$L_G = -\frac{1}{m}\sum_{i=1}^{m} \log D\big(G(z_i)\big)$$

original (minimax)	non-saturating
$\text{minimize } \log\big(1 - D(G(z))\big)$	$\text{minimize } -\log D(G(z))$
$\text{slope in } a: \ -\sigma(a) \approx 0$	$\text{slope in } a: \ \sigma(a) - 1 \approx -1$

Same direction; strong gradient when D rejects the sample

The practical default since the original GAN paper

GAN training algorithm

Algorithm Alternating minibatch training

repeat

1. Sample real images $x_1, \ldots, x_m$ and codes $z_1, \ldots, z_m \sim \mathcal{N}(0, I)$
2. Discriminator step: $$L_D = -\frac{1}{m} \sum_i \left[\log D(x_i) + \log\left(1 - D(G(z_i))\right)\right]$$

   update θ_D only (generated images detached)
3. Generator step: $$L_G = -\frac{1}{m} \sum_i \log D\left(G(z_i)\right)$$

   backpropagate through D; update θ_G only

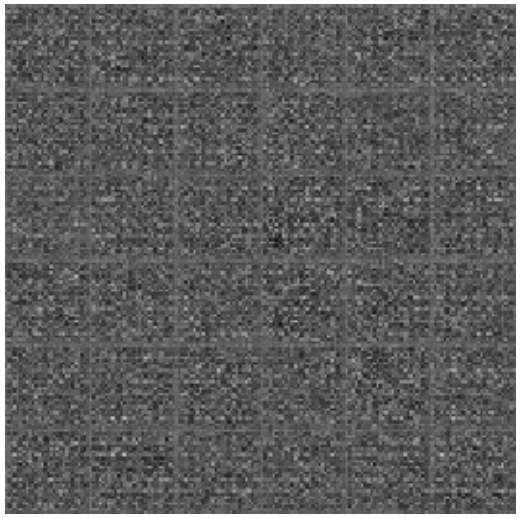
until samples stop improving

DCGAN setting: Adam, learning rate 2×10^{-4}, $\beta_1 = 0.5$

A small GAN trained on MNIST

The same 36 codes z at four points in training

iteration 0



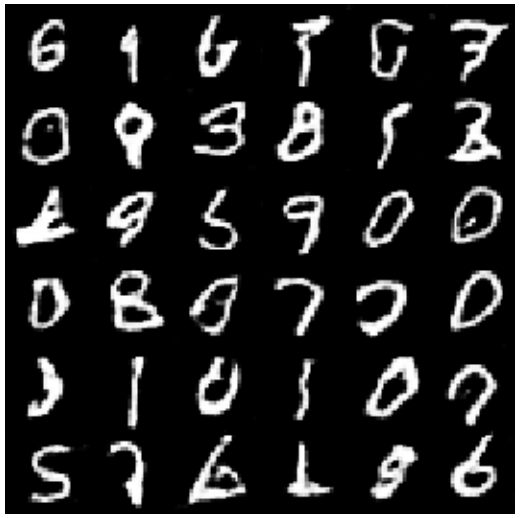
iteration 300



iteration 1,000



iteration 6,000

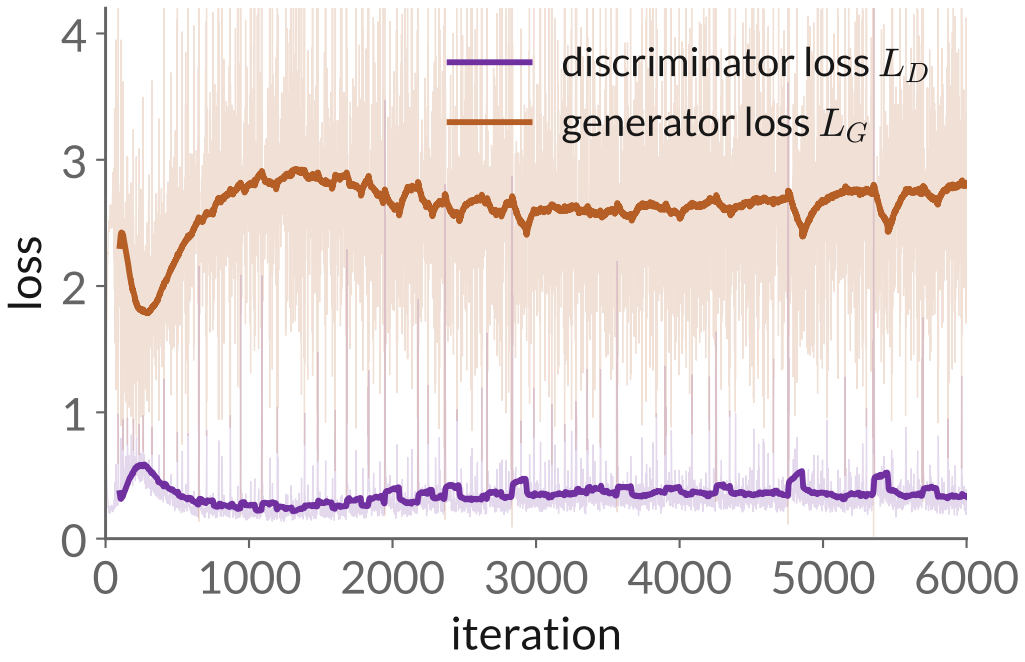


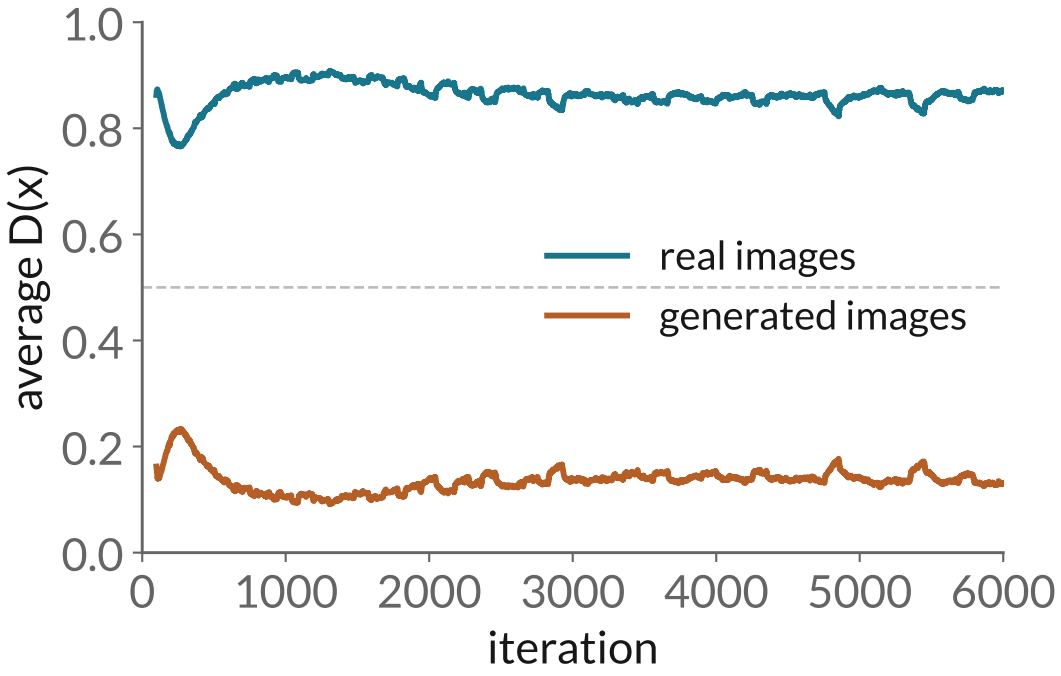
G: linear layer, then two stride-2 transposed convolutions (L08)

D: two stride-2 convolutions, one logit · about 4 minutes on a laptop

GAN losses do not measure progress

Same run; thick lines: averages over 100 iterations

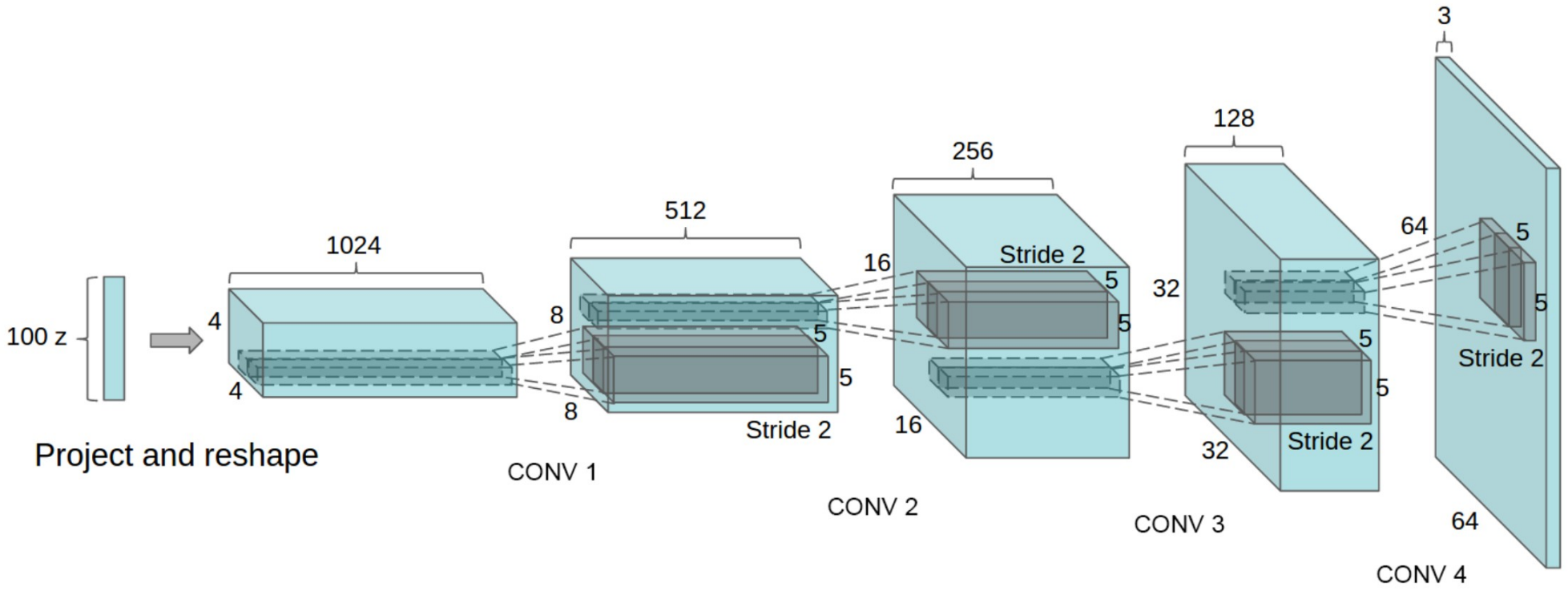




Samples improve from 1,000 to 6,000 iterations; the curves stay flat

Each loss is measured against an improving opponent

DCGAN: a convolutional generator



Project and reshape: $z \in \mathbb{R}^{100}$ to a $4 \times 4 \times 1024$ tensor

Four stride-2 transposed convolutions: $4 \to 8 \to 16 \to 32 \to 64$

tanh output: pixels in $[-1, 1]$

Source: DCGAN (2015)

DCGAN: discriminator and design choices

Generator

- Stride-2 transposed convolutions
- BatchNorm and ReLU
- tanh output

Discriminator

- Stride-2 convolutions, no pooling
- BatchNorm and LeakyReLU
- One logit output

Training: images scaled to $[-1, 1]$; Adam, learning rate 2×10^{-4}, $\beta_1 = 0.5$

A working recipe, not a guarantee of stable training

Source: DCGAN (2015)

DCGAN samples



64×64 bedrooms after 5 epochs on LSUN (3M photos)

One forward pass of G per sample

Source: DCGAN (2015)

Walking between latent codes

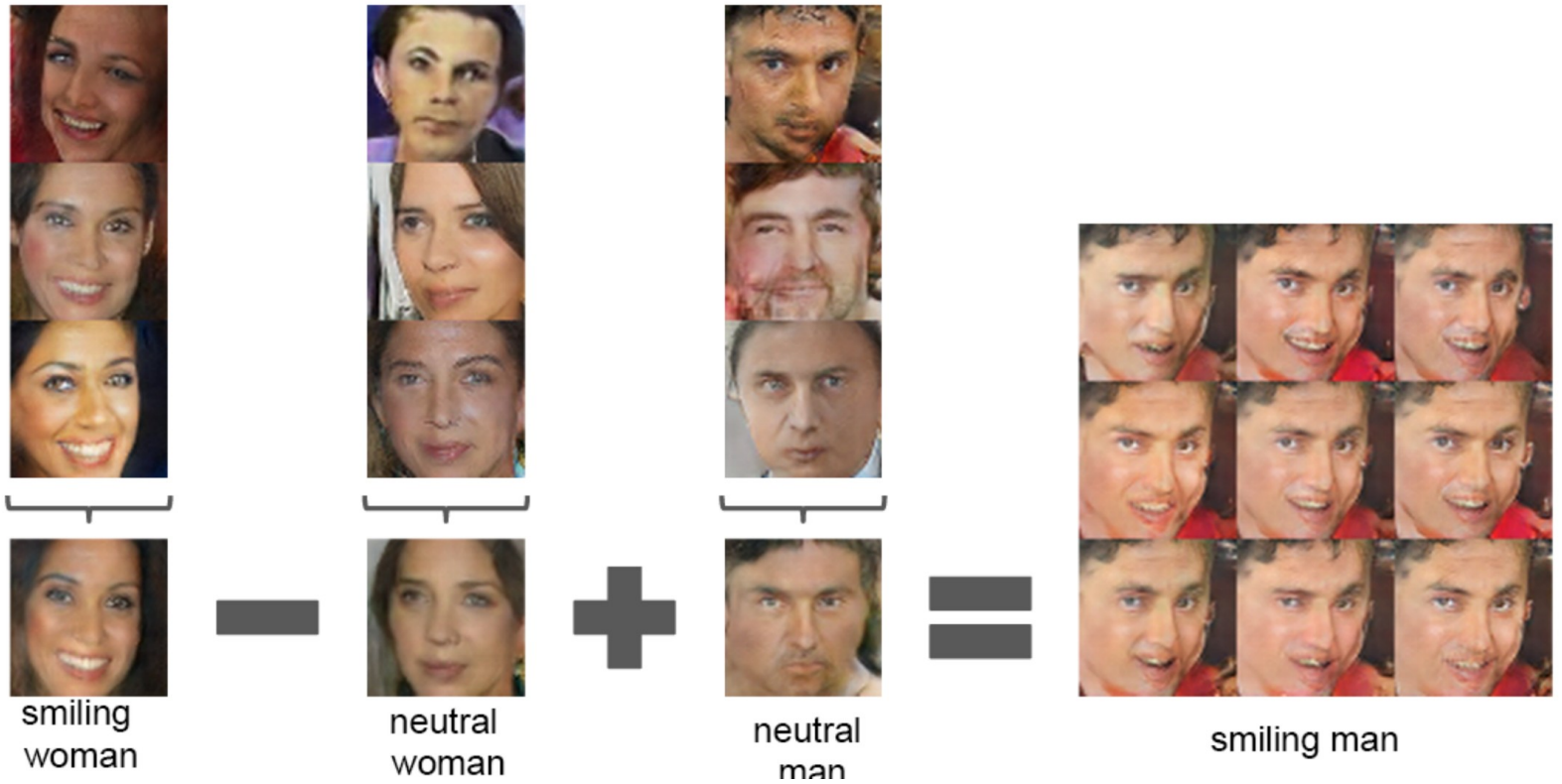


$$z(t) = (1-t)\, z_1 + t\, z_2, \qquad 0 \le t \le 1$$

Each row: $G\big(z(t)\big)$ for t from 0 to 1. Images change gradually.

Source: DCGAN (2015)

Arithmetic on latent codes



Average z over three images per concept

smiling woman – neutral woman + neutral man

$\rightarrow$ **smiling man**

Directions in z can encode attributes

Source: DCGAN (2015)

Lecture outline

01 Learning from a discriminator

02 Why GAN training is difficult

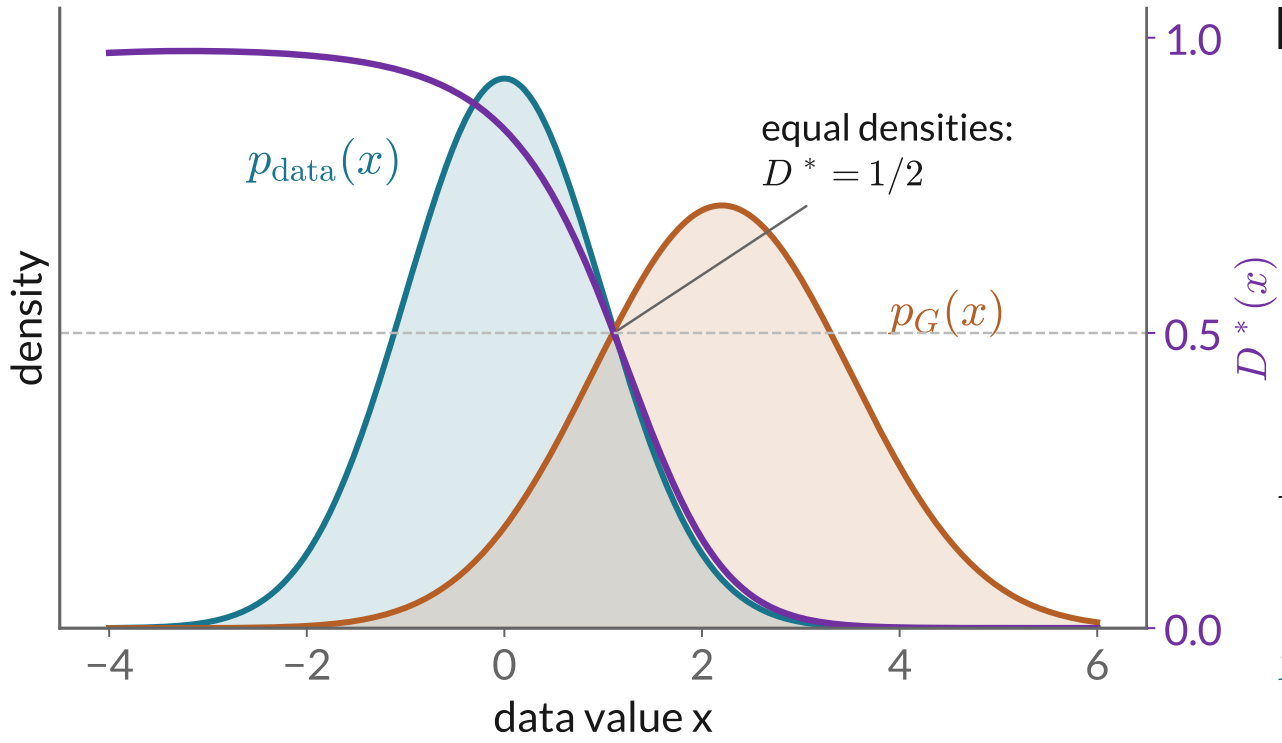
03 Conditional generation and image translation

04 Style-based generators

05 Adversarial losses in other models

06 Evaluating generators

The best discriminator for a fixed generator



For a fixed G, the best D is

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

$D^* = \frac{1}{2}$ **where the densities are equal**

p_{data} , p_G : densities of real and generated data

What the ideal game minimizes

Substitute D^* into V:

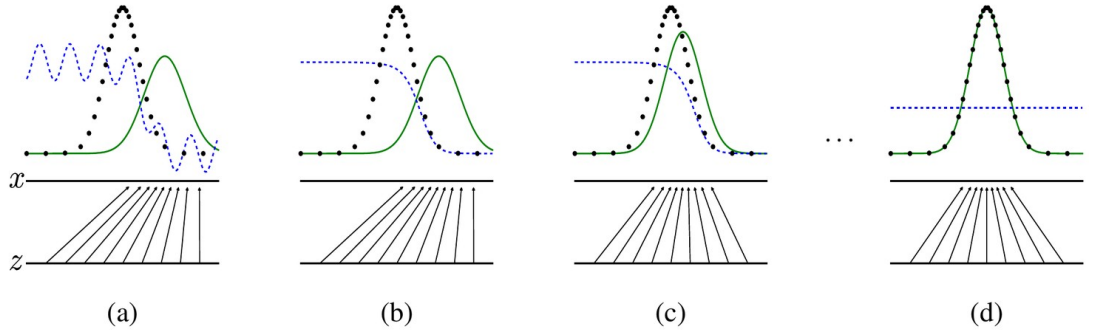
$$V(D^*, G) = -\log 4 + 2\,\mathrm{JS}\big(p_{\text{data}} \,\|\, p_G\big)$$

$$\mathrm{JS}(P\|Q) = \tfrac{1}{2}\,\mathrm{KL}(P\|M) + \tfrac{1}{2}\,\mathrm{KL}(Q\|M), \quad M = \tfrac{1}{2}(P+Q)$$

Jensen–Shannon divergence: ≥ 0, and $= 0$ only if $p_G = p_{\text{data}}$

Assumes an optimal D after every step

Real training: a few gradient steps on finite networks

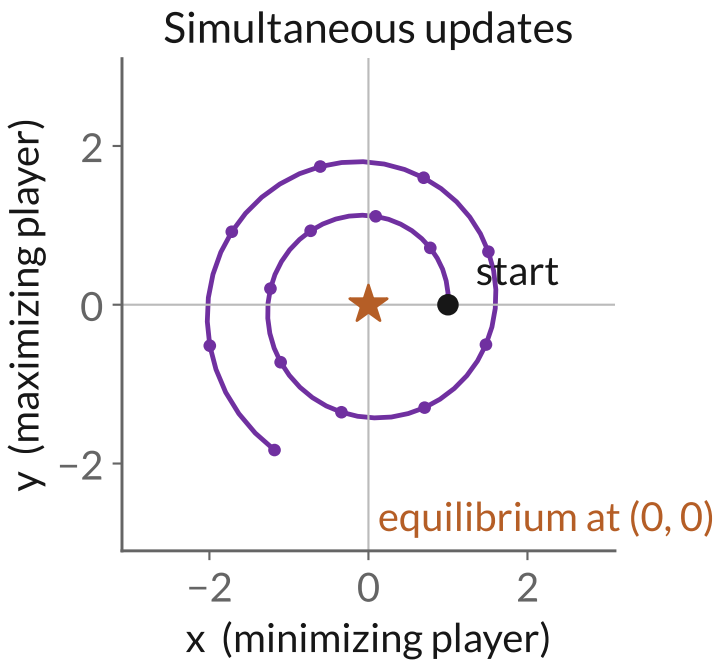


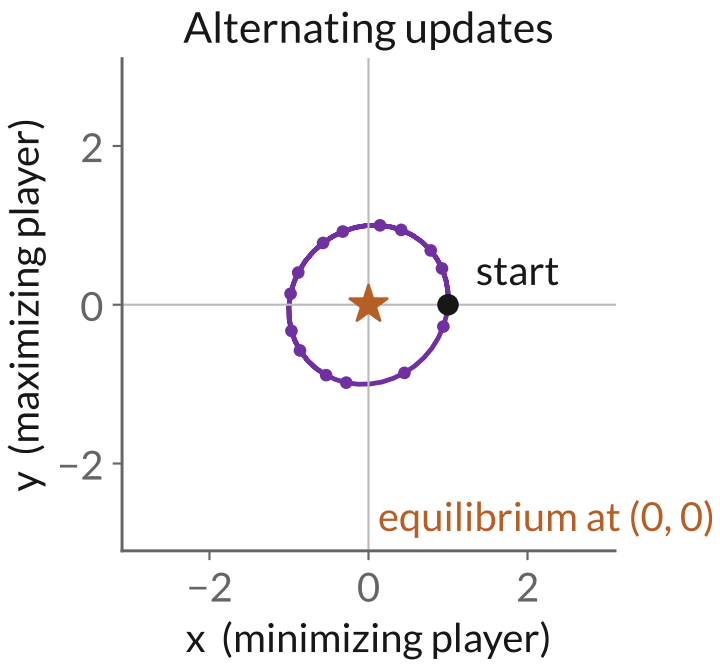
Idealized: D retrained, p_G moves to the data until $D = \frac{1}{2}$

Source: GAN (2014)

Players that keep changing each other's objective

x minimizes, y maximizes $V(x, y) = xy$; equilibrium at $(0, 0)$





$$x \leftarrow x - \eta\, y$$

$$y \leftarrow y + \eta\, x$$

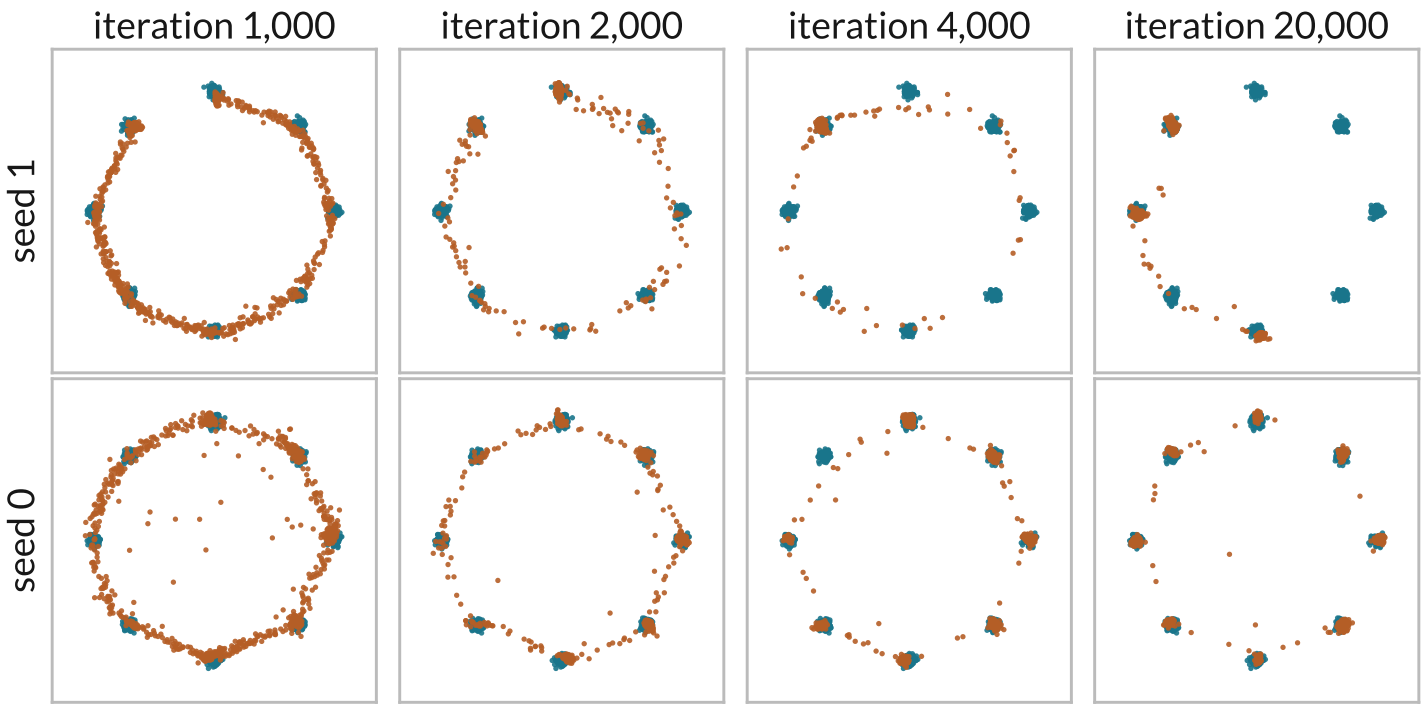
Simultaneous: spirals out

Alternating: circles

Each update moves the other player's target

Mode collapse

Eight Gaussians (teal); same data, networks and optimizer; two random seeds



Seed 1: 8 clusters $\rightarrow$ 1 cluster

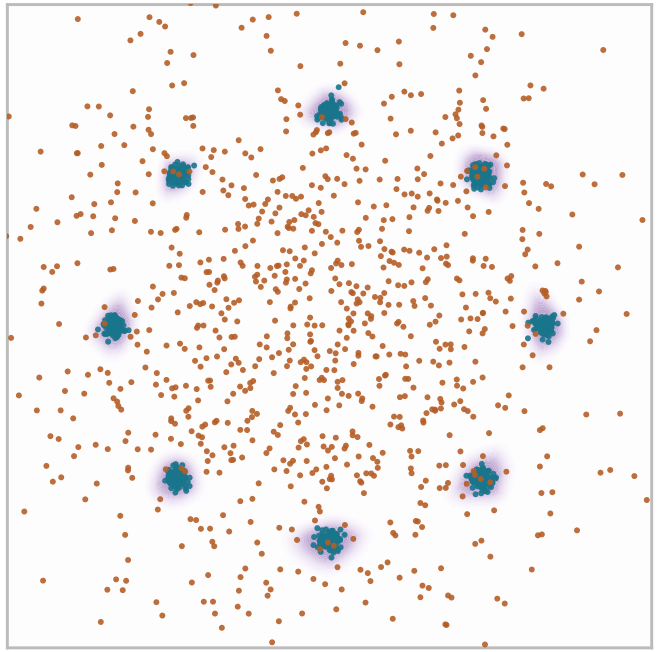
Seed 0: all 8 kept

Mode collapse: samples cover only part of the data

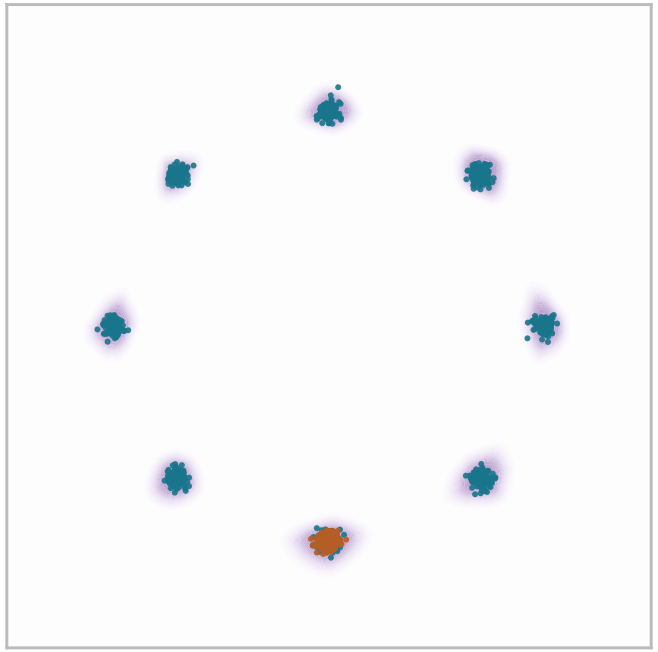
Each sample can still look realistic

Why the generator can collapse

D trained, then frozen



G after 3,000 steps against frozen D



Train D, freeze it, then train only G

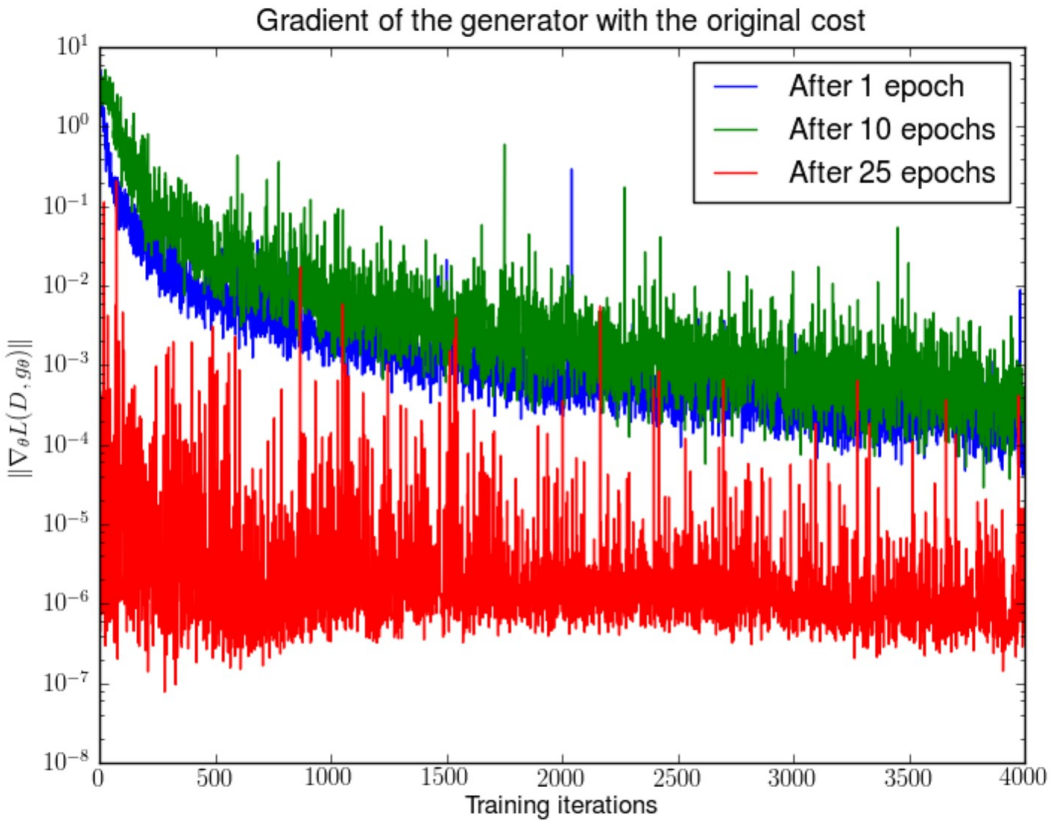
All samples move to the point D scores highest

teal real, **orange** generated, shading: D

G's loss scores each sample separately: nothing rewards variety

As D keeps updating, G hops from mode to mode

When the discriminator wins too easily



Real and early generated images barely overlap: D separates them almost perfectly

Experiment: fix G after 1, 10, 25 epochs; train a new D

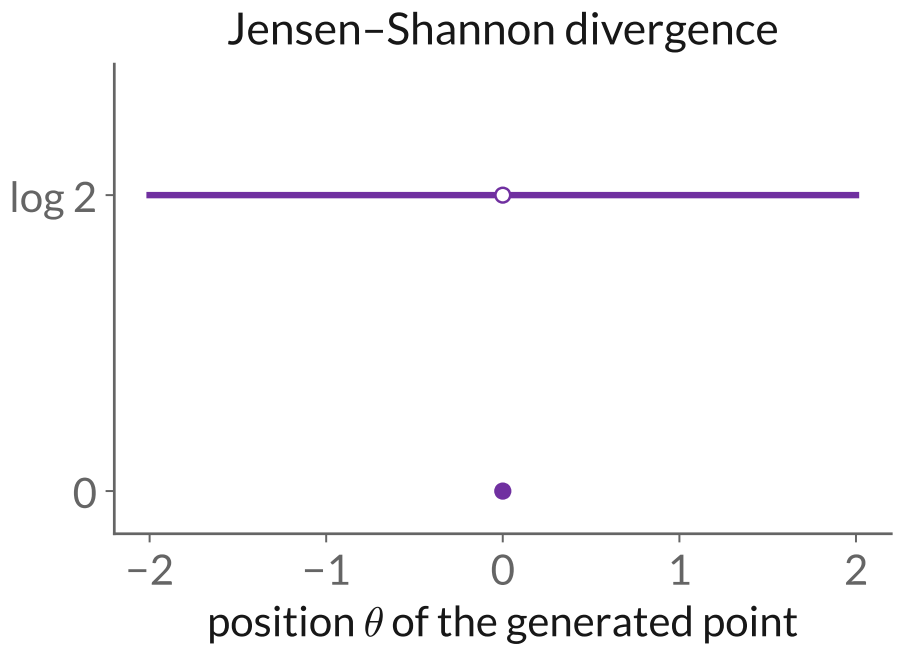
Better D $\rightarrow$ vanishing gradient for G

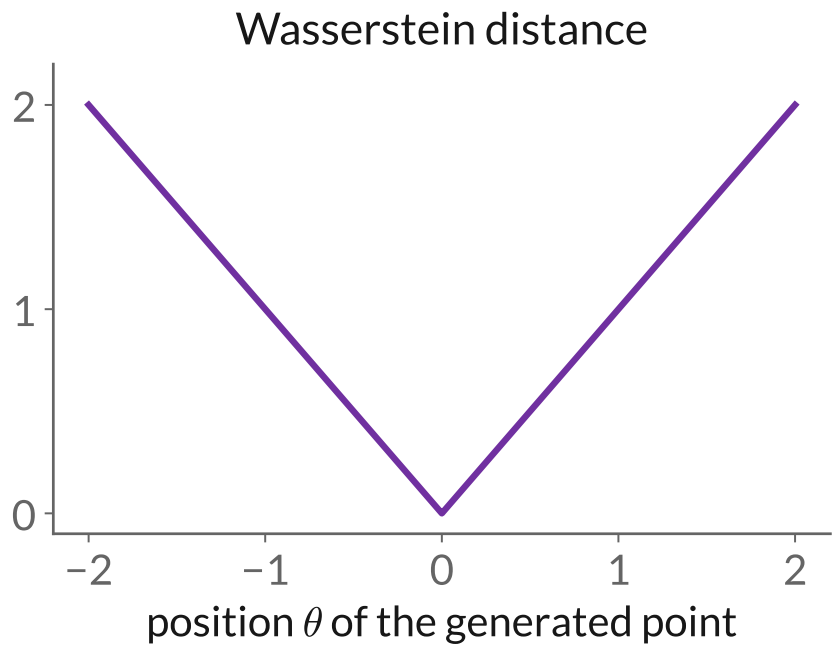
log scale: several orders of magnitude

Source: Towards Principled Methods for Training GANs (2017)

Distributions that do not overlap

Real data: a point at 0. Generated data: a point at θ.



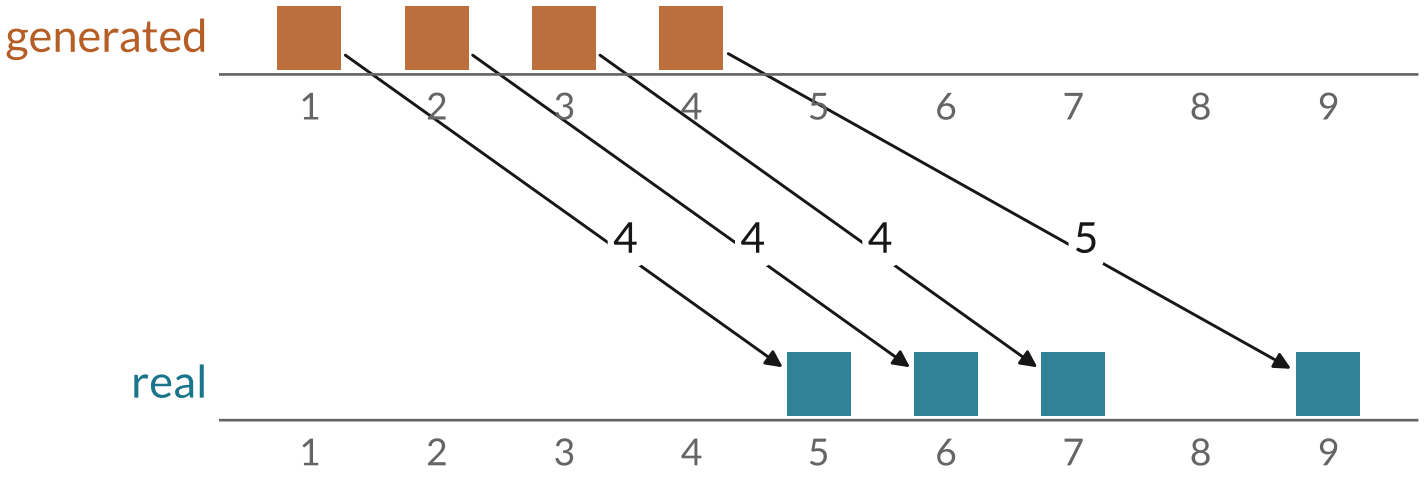


JS $= \log 2$ for every $\theta \neq 0$: zero gradient

$|\theta|$ shrinks as the point approaches the data: a useful direction

Wasserstein distance: the cost of moving mass

The cheapest way to move one pile of probability mass onto the other



Each block: mass $\frac{1}{4}$

$$W = \tfrac{1}{4}(4 + 4 + 4 + 5) = 4.25$$

Two points: $W = |\theta|$

shrinks even without overlap

$$W(p_{\text{data}}, p_G) = \min_{\text{plans}} \sum (\text{mass moved}) \times (\text{distance})$$

Also called the earth mover's distance

Wasserstein GAN: a critic instead of a classifier

Optimal transport duality:

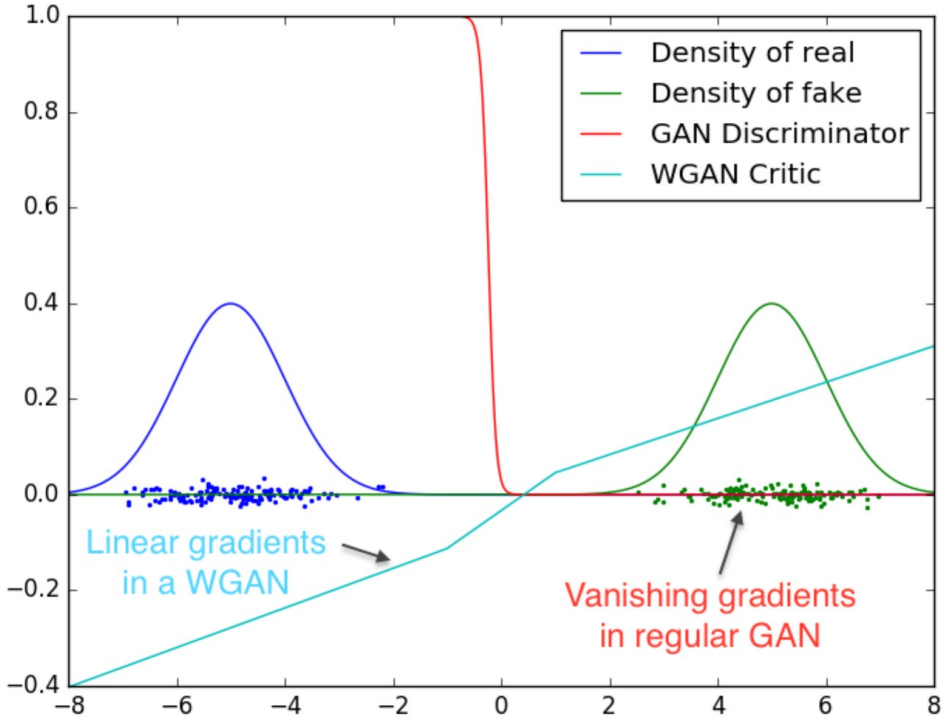
$$W(p_{\text{data}}, p_G) = \max_{f:\ \text{1-Lipschitz}} \mathbb{E}_{x \sim p_{\text{data}}}\left[f(x)\right] - \mathbb{E}_z\left[f(G(z))\right]$$

Critic f: unbounded score, no sigmoid

Critic loss: $\overline{f(G(z))} - \overline{f(x)}$

Generator loss: $-\overline{f(G(z))}$

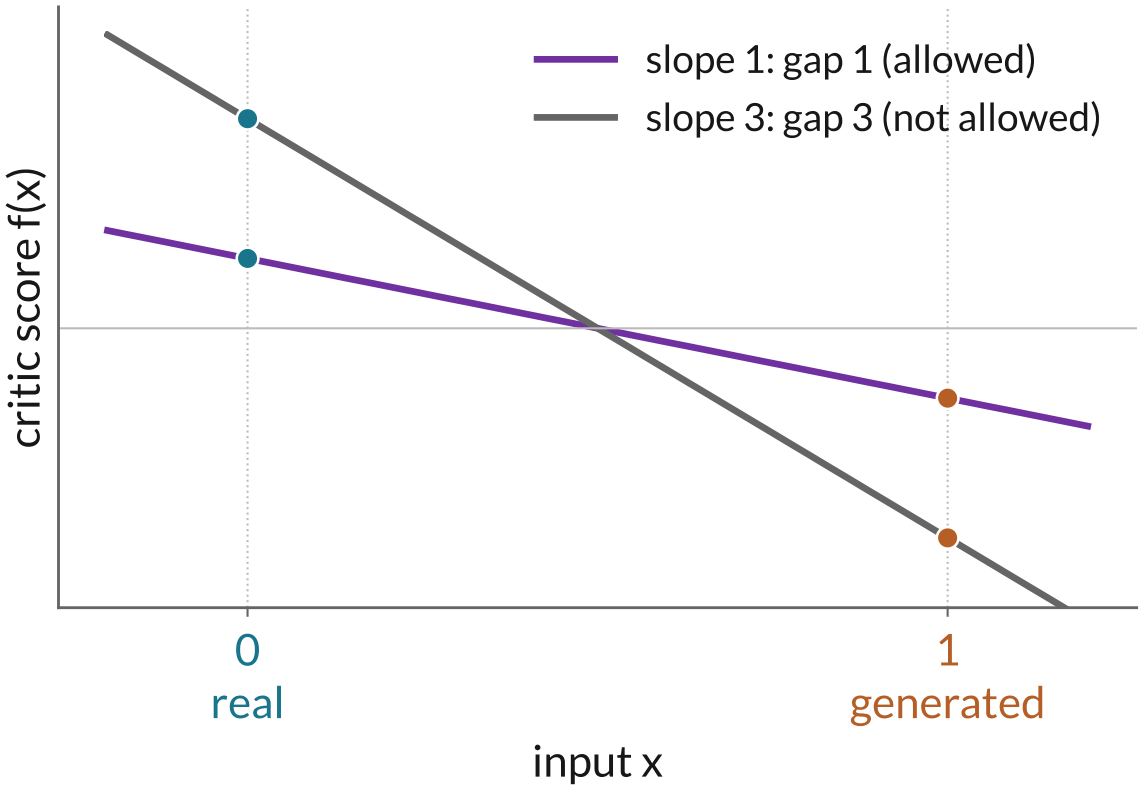
Score gap estimates a distance, not a probability



Right: the discriminator (red) saturates; the critic (cyan) stays linear

Source: Wasserstein GAN (2017)

Why the critic's slope must be limited



Scale f by 3: the gap triples. No limit, no finite maximum.

1-Lipschitz

$$|f(x_1) - f(x_2)| \leq \|x_1 - x_2\|$$

For a differentiable critic:

$$\|\nabla_x f(x)\| \leq 1$$

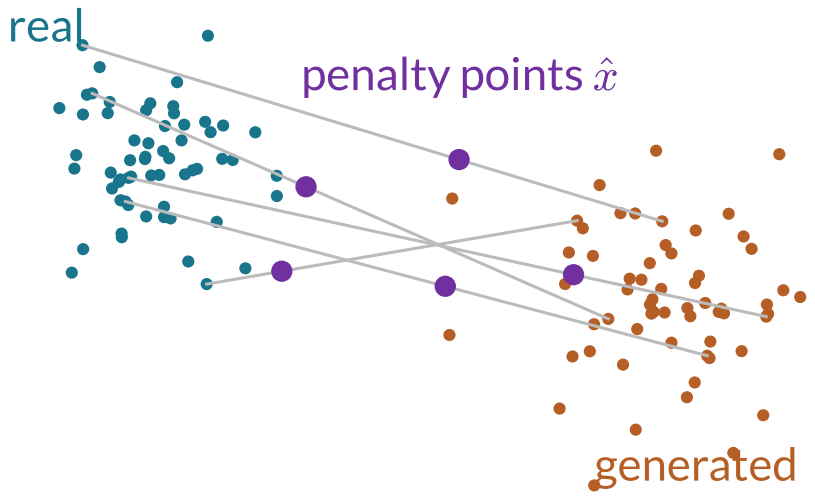
Enforcing the limit: clipping or a gradient penalty

Weight clipping (original WGAN)

Clip every weight to $[-c, c]$

Too simple, or exploding gradients

Where the penalty is checked



Gradient penalty (WGAN-GP)

$$L_f = \underbrace{\overline{f(G(z))} - \overline{f(x)}}_{\text{critic loss}} + \lambda \underbrace{\left(\|\nabla_{\hat{x}} f(\hat{x})\|_2 - 1\right)^2}_{\text{penalty}}$$

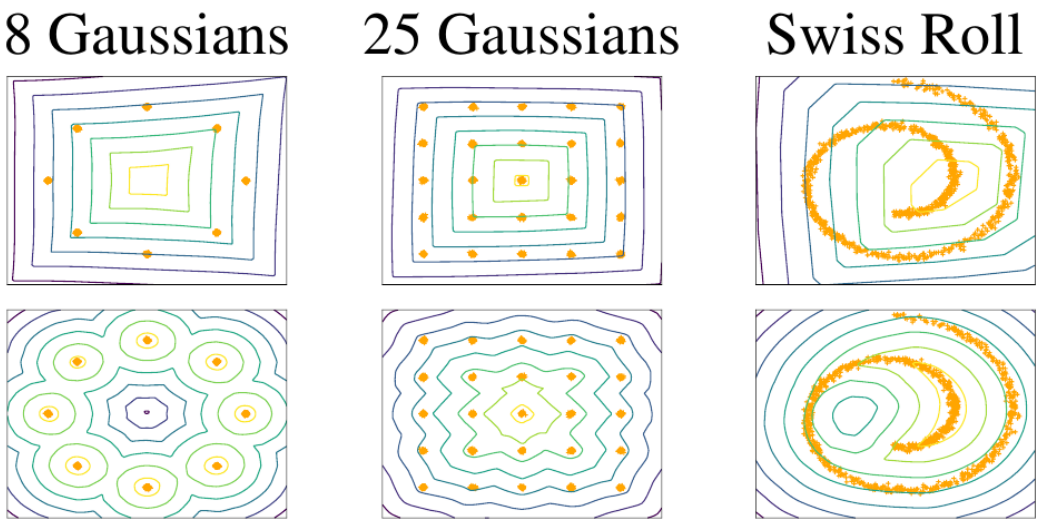
$$\hat{x} = \epsilon\, x + (1 - \epsilon)\, G(z), \qquad \epsilon \sim \mathcal{U}[0, 1]$$

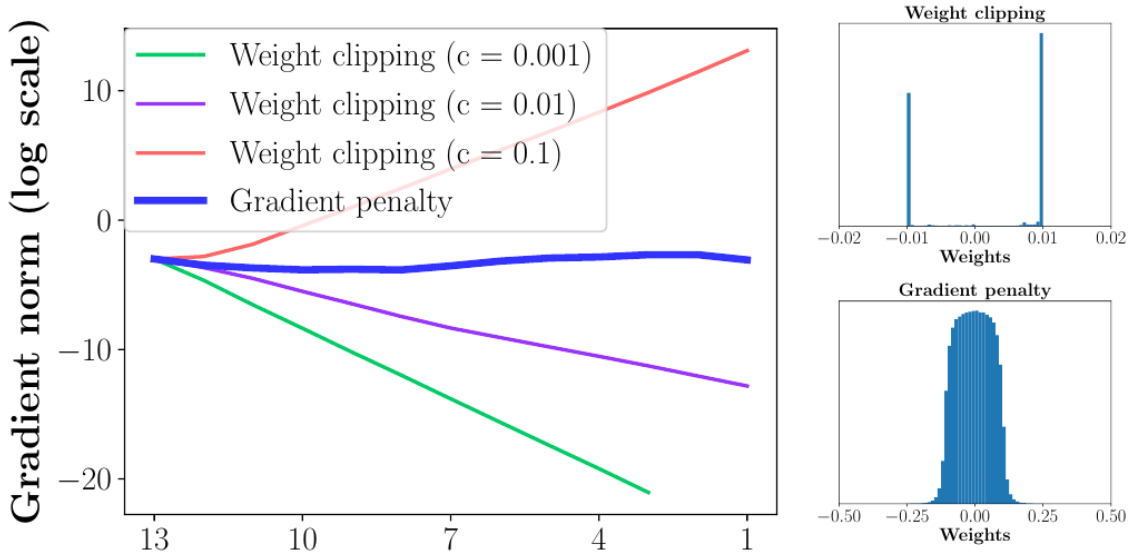
Slope $\neq 1$ between real and generated samples is penalized

$\lambda = 10$; overline: minibatch mean

Source: Wasserstein GAN (2017) · Improved Training of Wasserstein GANs (2017)

Clipping versus penalty on toy data



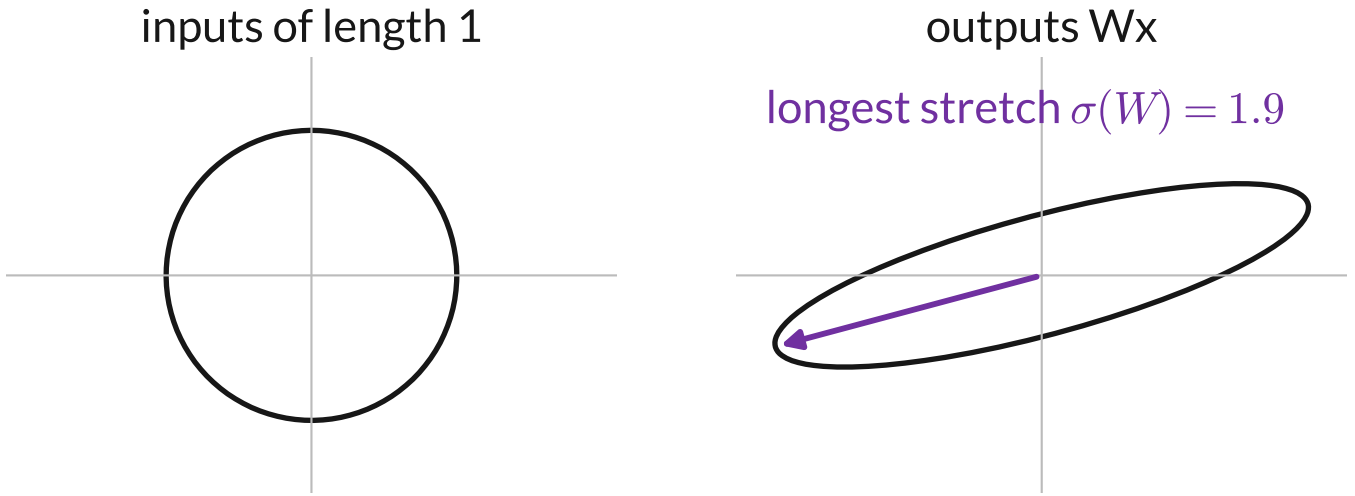


Left: clipping gives overly simple critics; the penalty follows the data

Right: with clipping, gradients vanish or explode with depth

Source: Improved Training of Wasserstein GANs (2017)

Spectral normalization



$\sigma(W)$: largest stretch of W
(largest singular value)

$$\bar{W} = \frac{W}{\sigma(W)}$$

No extra loss term

Every layer stretches by ≤ 1, so with ReLU the network is 1-Lipschitz

Source: Spectral Normalization (2018)

Difficulties and what helps

	cause	what helps
Moving objectives	each update moves the other's objective	small learning rates, Adam with $\beta_1 = 0.5$; no guarantee
Mode collapse	no reward for variety	WGAN-GP, conditioning; measure coverage
Uninformative gradients	D wins too easily	non-saturating loss; limit the slope

Lecture outline

01 Learning from a discriminator

02 Why GAN training is difficult

03 Conditional generation and image translation

04 Style-based generators

05 Adversarial losses in other models

06 Evaluating generators

Class-conditional generation



Give both networks a condition y, e.g. the digit class (one-hot)

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

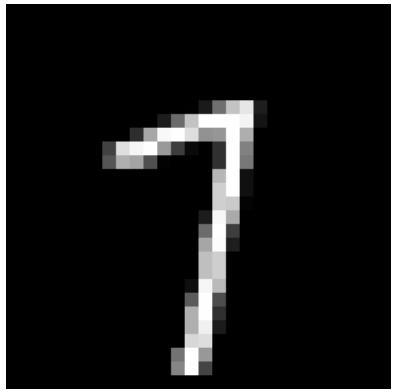
y: which digit

z: everything else (stroke width, slant)

Each row: one label, different codes

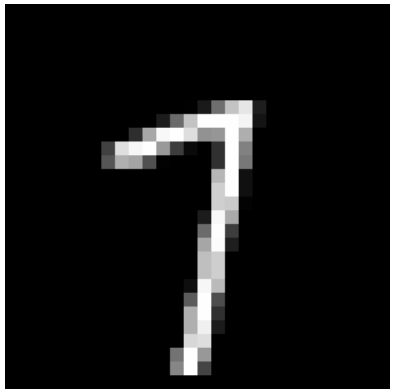
Source: Conditional GAN (2014)

The discriminator must see the condition



requested: 7

realistic and correct



requested: 3

realistic but wrong

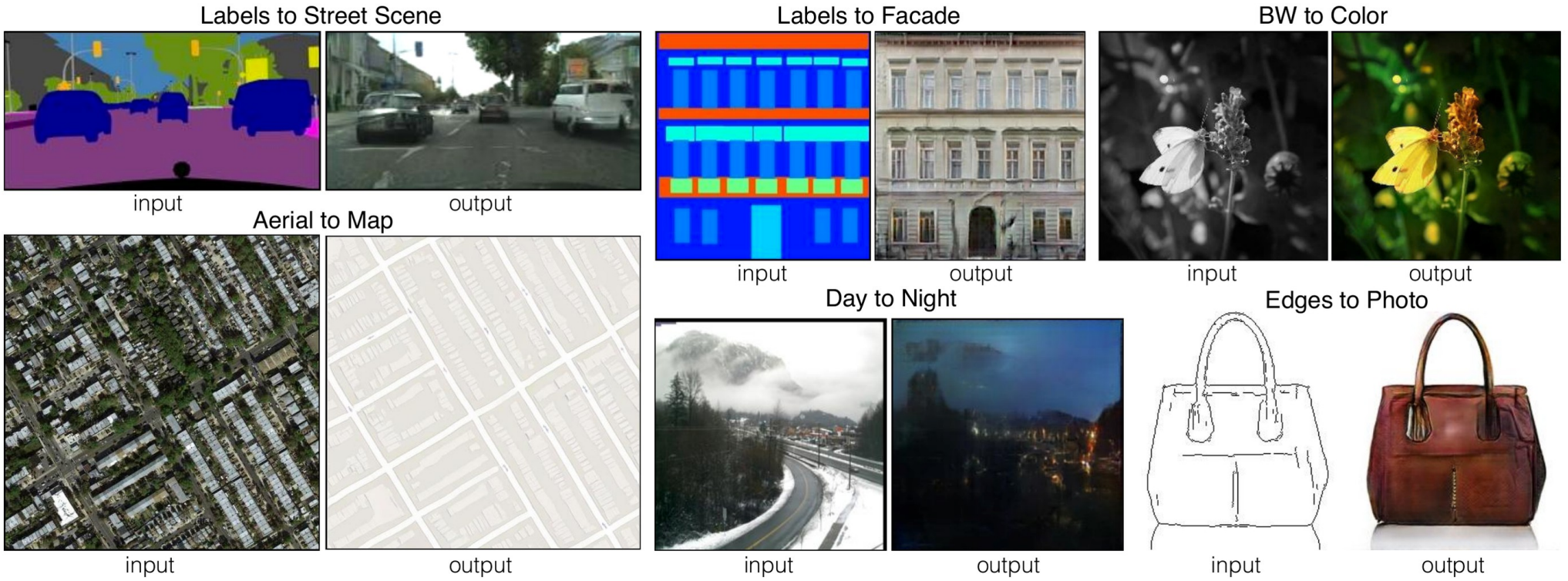
An image-only D accepts the wrong 7

A conditional D judges the pair (x, y)

$$L_D = -\mathbb{E}\big[\log D(x, y)\big] - \mathbb{E}\big[\log\big(1 - D(G(z, y),\ y)\big)\big]$$

Generated pairs use the requested label

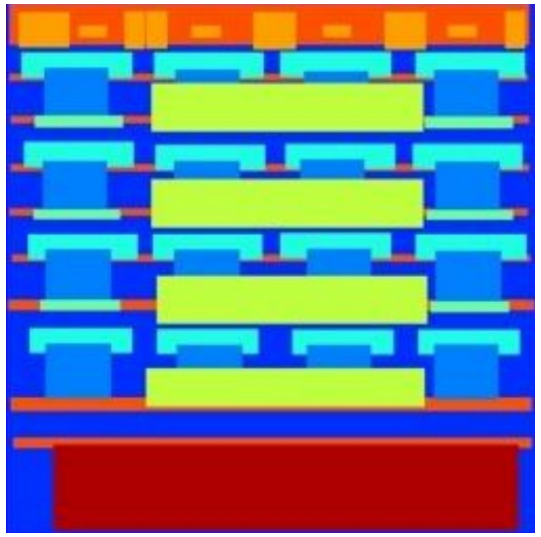
Image-to-image translation



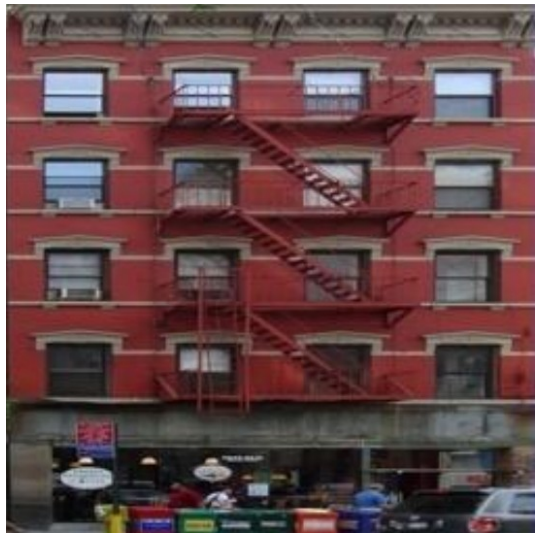
The condition is a whole image. One method for many paired tasks.

Source: pix2pix (2017)

A pixel loss alone blurs



input x: label map



target y: photo



$G(x)$, L1 loss only

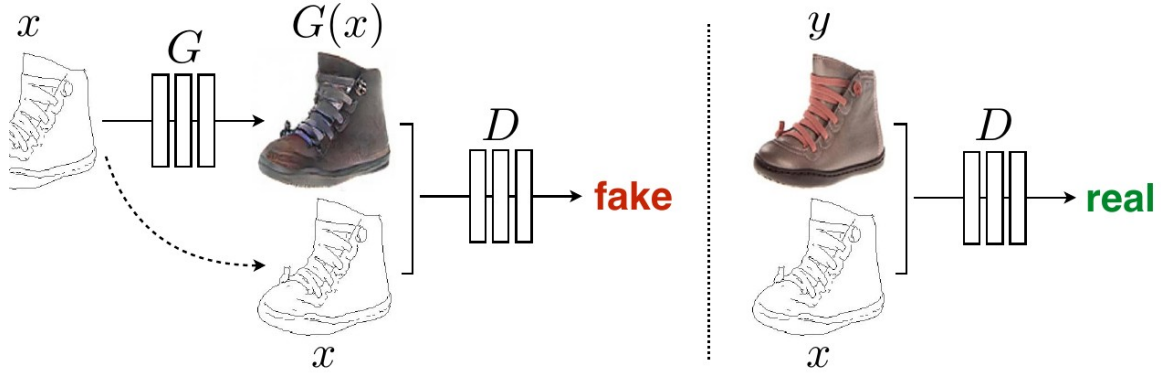
$$L_{L1} = \mathbb{E} \|y - G(x)\|_1$$

Many textures fit one label map; L1 averages them

Layout right, texture blurry: the averaging of L15 again

Source: pix2pix (2017)

pix2pix: a discriminator that judges pairs



D sees the input together with a real or generated photo

Is this a plausible photo of this map?

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

$$\mathcal{L}_{\text{cGAN}} = \mathbb{E}\big[\log D(x, y)\big] + \mathbb{E}\big[\log\big(1 - D(x, G(x))\big)\big]$$

Adversarial term: realistic detail. L1: close to the target ($\lambda = 100$).

Source: pix2pix (2017)

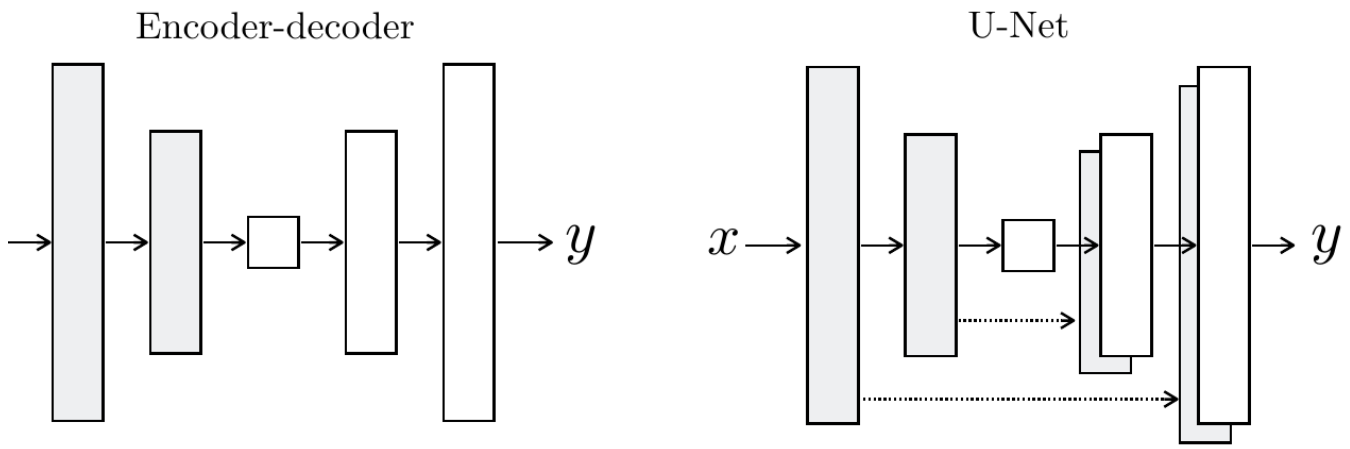
What each loss contributes



L1: blurry · cGAN: sharp but less faithful · both: sharp and faithful

Source: pix2pix (2017)

U-Net generator



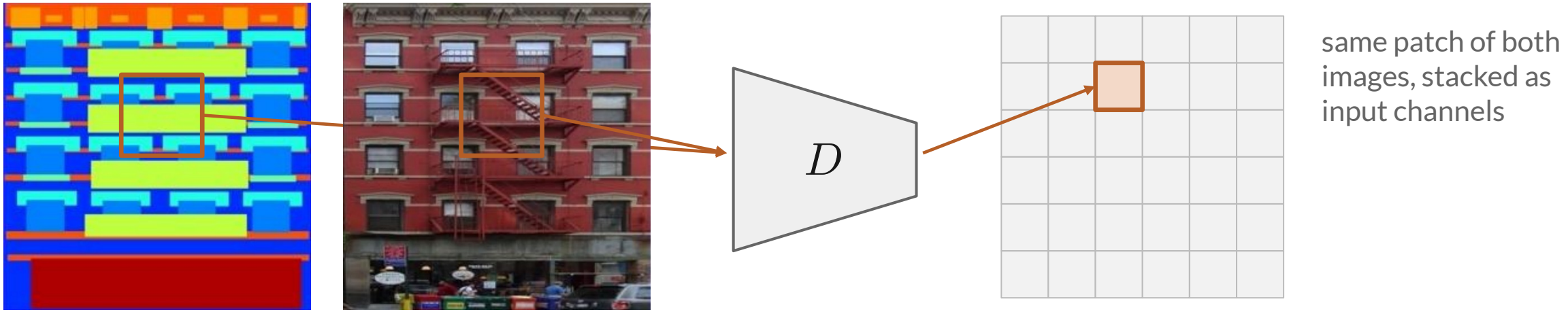
Input and output share their layout

Skip connections carry spatial detail past the bottleneck (L08)

No input noise z: outputs are nearly deterministic

Source: pix2pix (2017)

PatchGAN discriminator

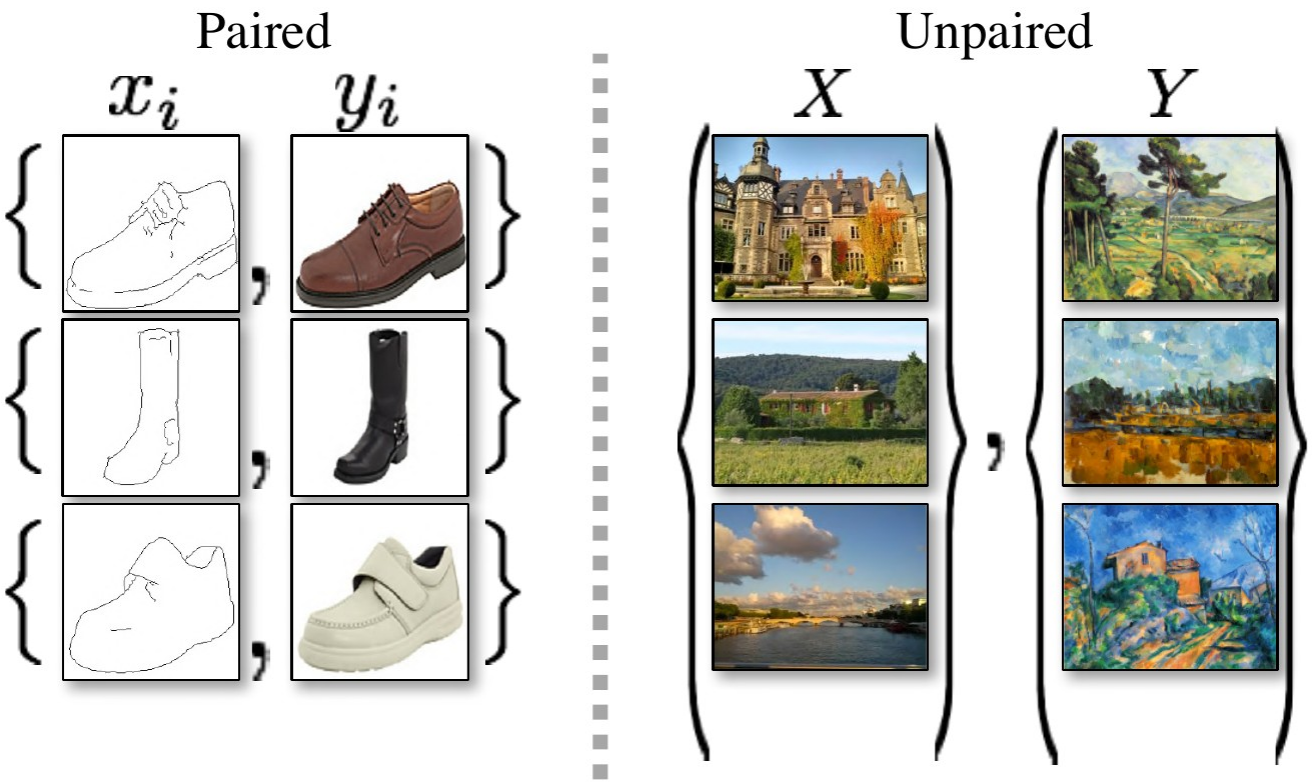


Each score sees a 70×70 patch; the loss averages all scores

D judges local texture; L1 handles the global layout

Source: pix2pix (2017)

Unpaired collections

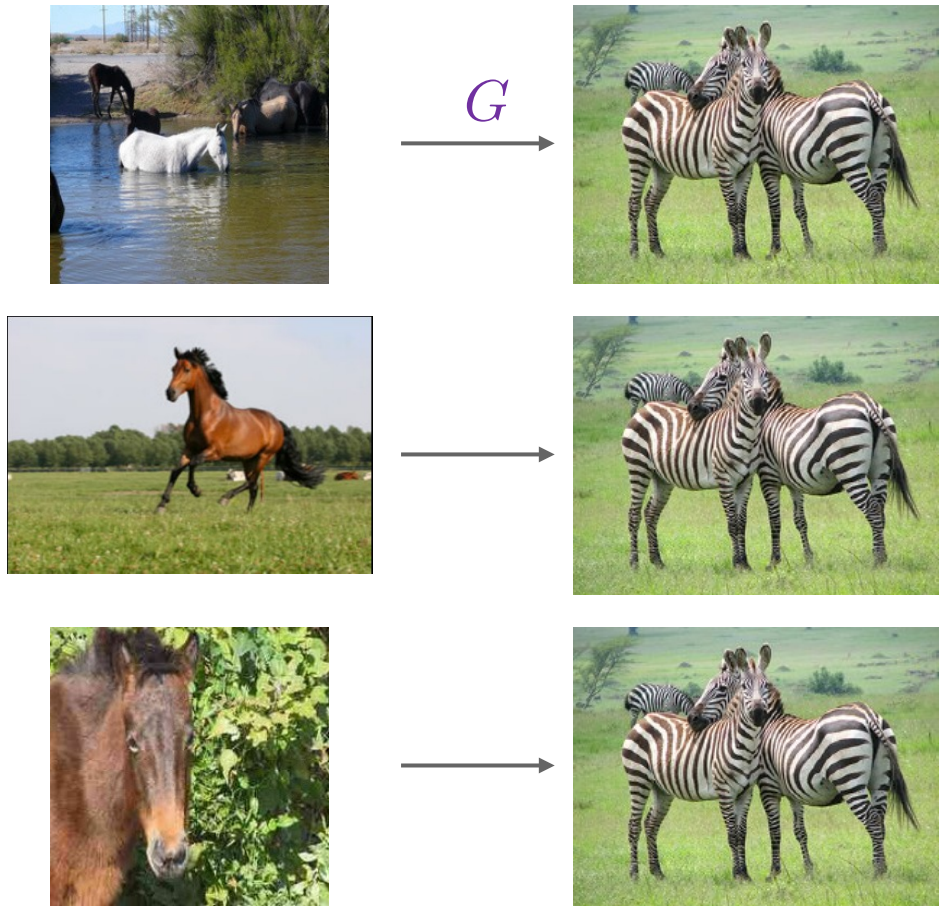


Often no pairs exist: Monet's scenes, a horse and a zebra in the same pose

Only two collections X and Y

Source: CycleGAN (2017)

An adversarial loss alone does not preserve content



input $x \in X$

output $G(x)$

$$\mathcal{L}_{\text{GAN}}(G, D_Y)$$

D_Y only asks: does each output look like a zebra?

Mapping every horse to one zebra satisfies it

Need: each output tied to its own input

Left: an illustration

Source: CycleGAN (2017)

Cycle consistency



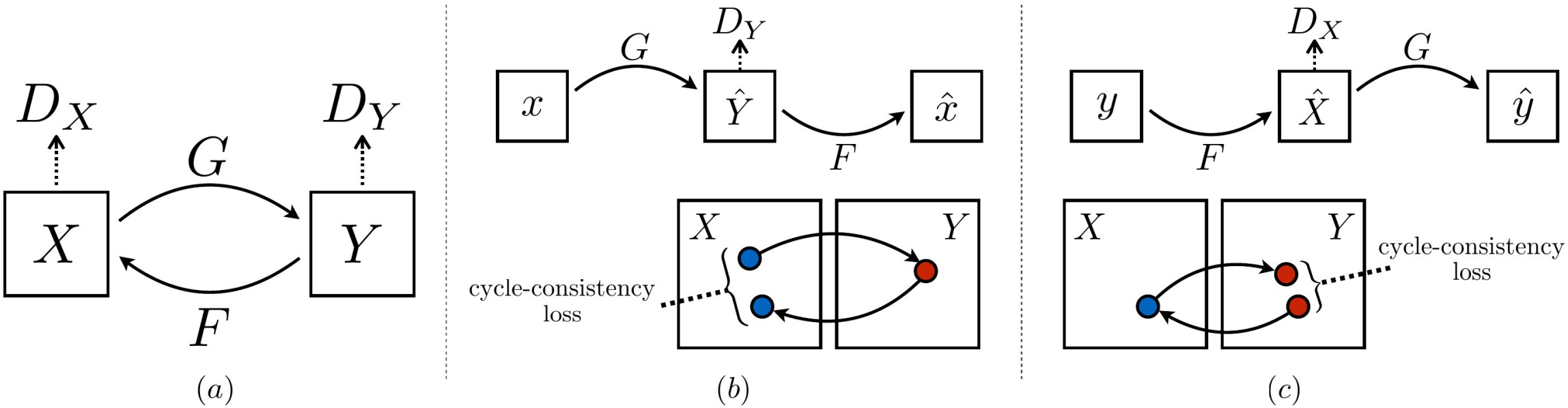
x $G(x)$ $F(G(x))$

Second generator $F : Y \to X$. There and back should return the original.

$$\mathcal{L}_{\text{cyc}} = \mathbb{E}_x \, \|F(G(x)) - x\|_1 + \mathbb{E}_y \, \|G(F(y)) - y\|_1$$

Source: CycleGAN (2017) · reconstructions

The complete CycleGAN objective

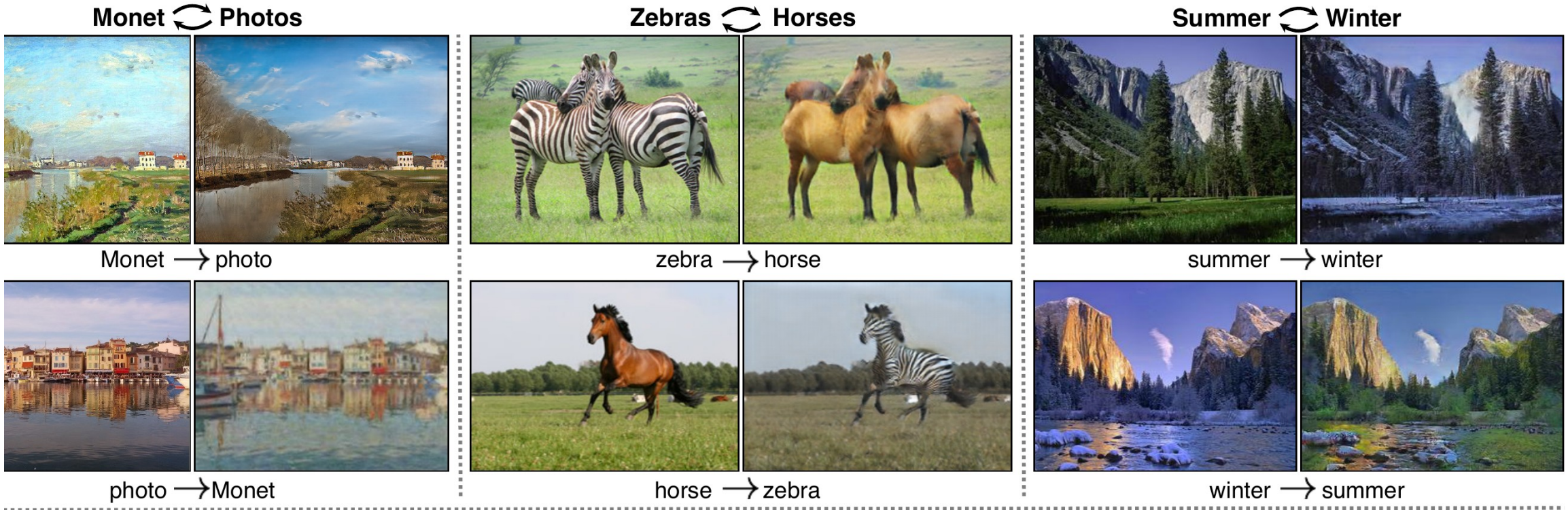


$$\mathcal{L} = \mathcal{L}_{\text{GAN}}(G, D_Y) + \mathcal{L}_{\text{GAN}}(F, D_X) + \lambda \mathcal{L}_{\text{cyc}}(G, F)$$

Two generators, two discriminators, both cycles; $\lambda = 10$

Source: CycleGAN (2017)

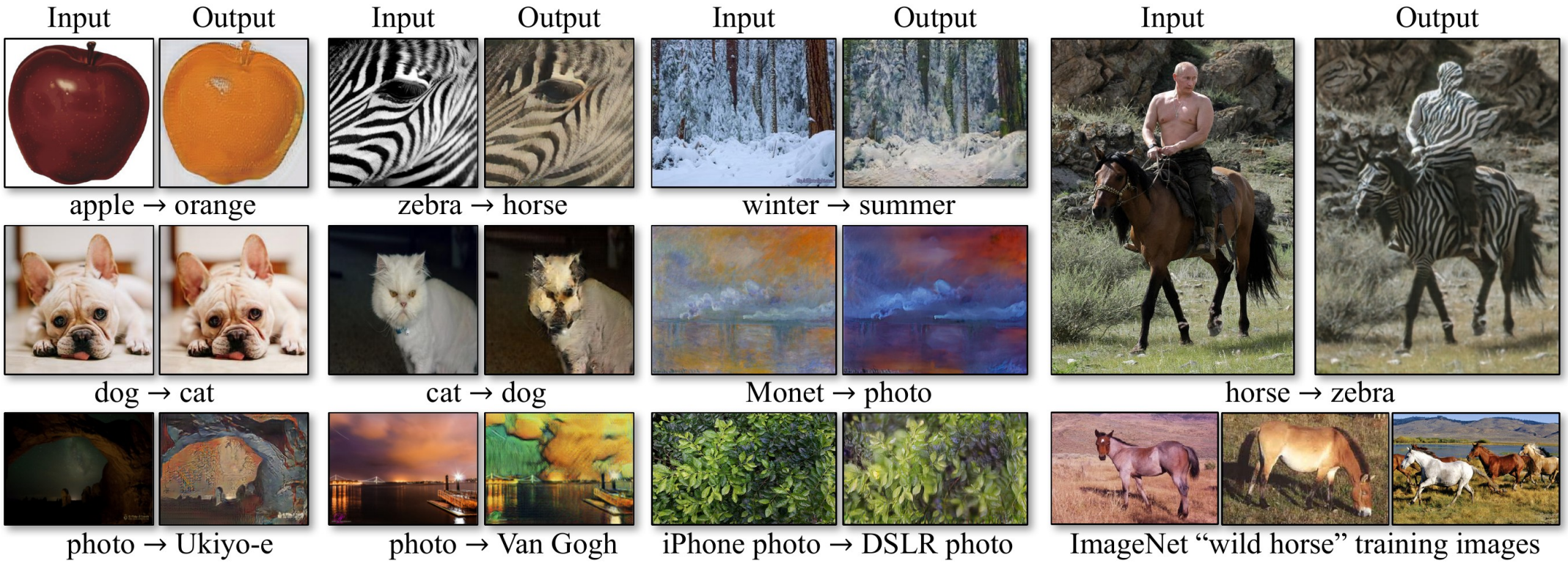
CycleGAN results



Trained on unpaired collections only

Source: CycleGAN (2017)

Limits of cycle consistency



Shape changes fail: dog ↔ cat

The rider gets stripes too

Training horses have no riders

Source: CycleGAN (2017)

Lecture outline

01 Learning from a discriminator

02 Why GAN training is difficult

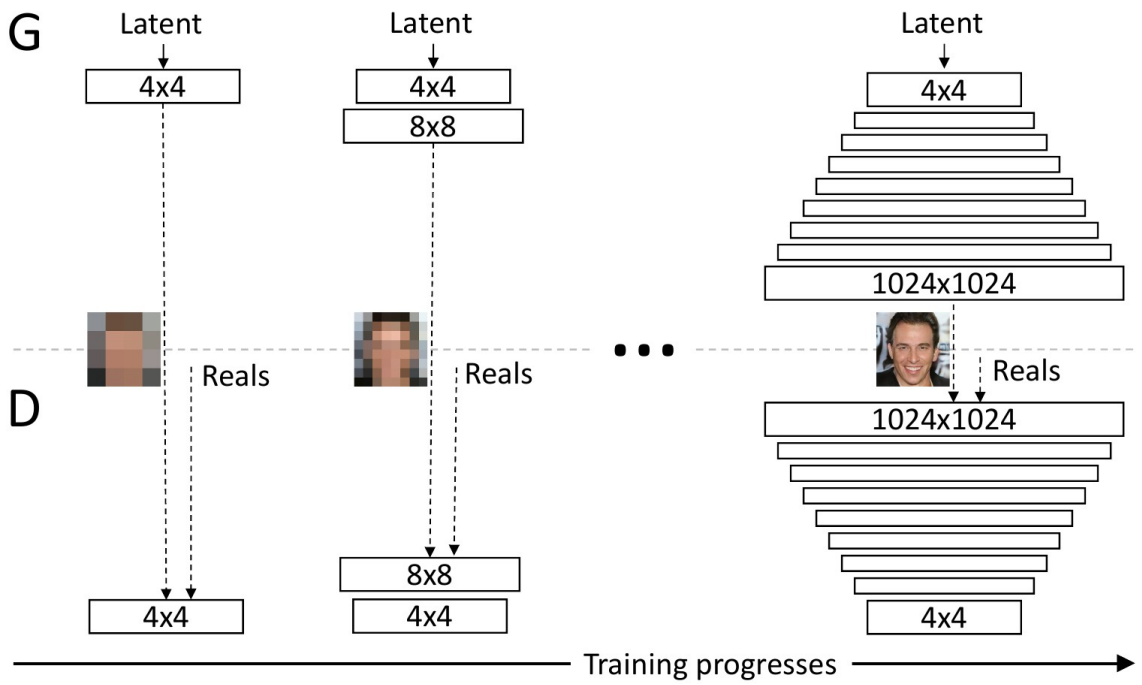
03 Conditional generation and image translation

04 Style-based generators

05 Adversarial losses in other models

06 Evaluating generators

Growing the resolution during training





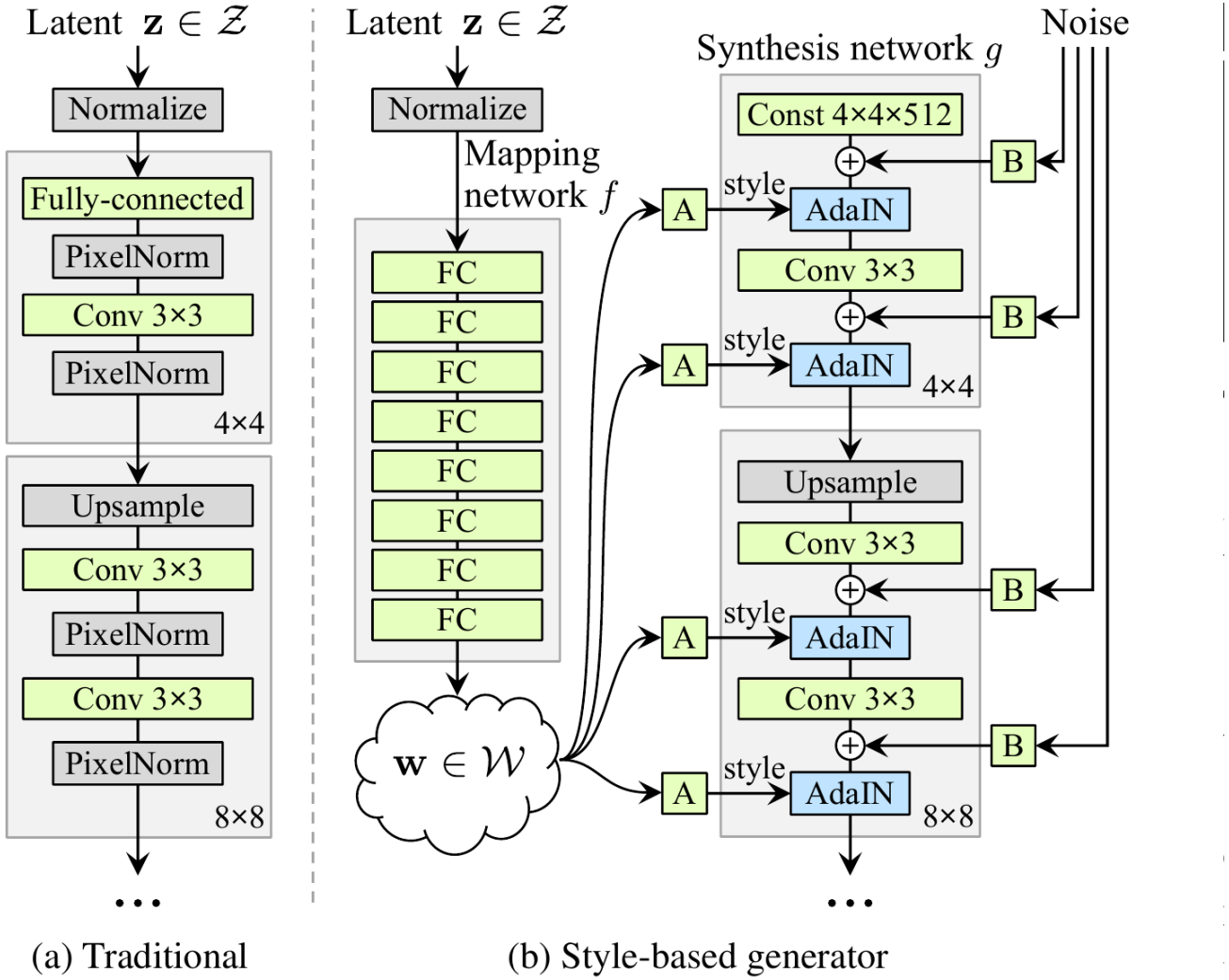
Start at 4×4; add layers to G and D up to 1024×1024

New layers fade in gradually

Right: generated 1024×1024 faces

Source: Progressive GAN (2017)

Where the latent code enters



(a) Traditional (b) Style-based generator

(a) z enters only the first layer

(b) mapping network: $z \rightarrow w$

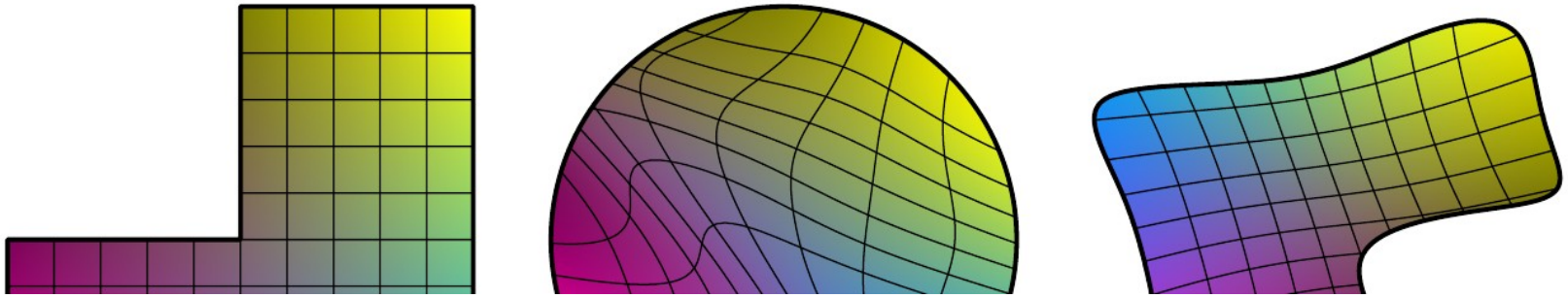
A: $w \rightarrow$ a style (scale, shift) for every layer

B: per-pixel noise

Synthesis starts from a learned constant

Source: StyleGAN (2018)

Why a mapping network?



(a) One combination of factors is missing from the data

(b) A Gaussian z must warp around the gap

(c) A learned w can be laid out evenly

An illustration, not a measured representation

Source: StyleGAN (2018)

Style modulation (AdaIN)

Adaptive instance normalization of channel i of a feature map h:

$$\mathrm{AdaIN}(h_i, w) = \underbrace{s_i(w)}_{\text{style scale}} \underbrace{\frac{h_i - \mu(h_i)}{\sigma(h_i)}}_{\text{normalized channel}} + \underbrace{b_i(w)}_{\text{style shift}}$$

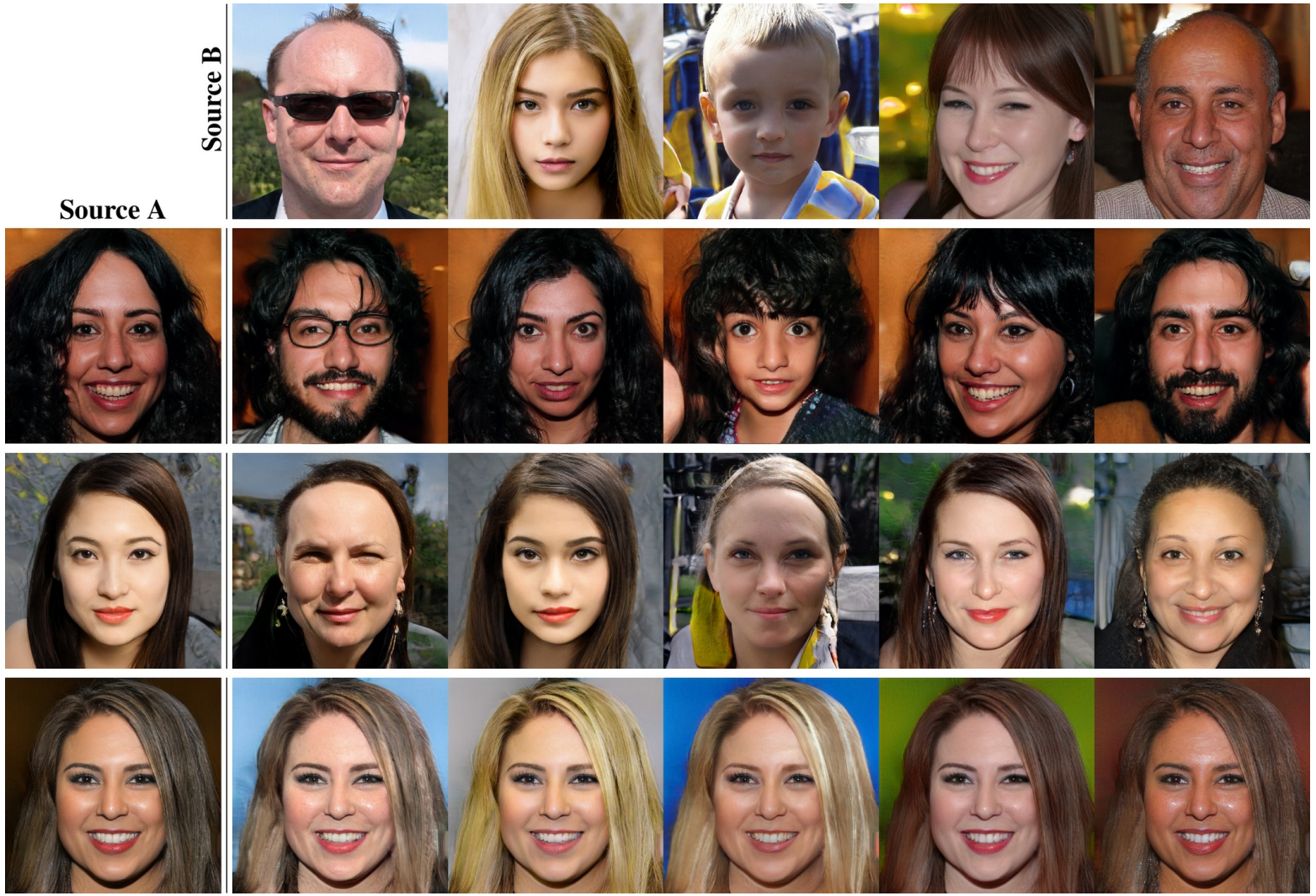
$\mu(h_i), \sigma(h_i)$: mean and standard deviation over positions

$s_i(w), b_i(w)$: from w by the learned layer A

Each layer gets its own style

Source: StyleGAN (2018)

Mixing styles across resolutions



w from A (left) at some layers, from B (top) at the others

Coarse from B: pose, face shape, glasses

Middle from B: features, hairstyle

Fine from B: colors, texture

Source: StyleGAN (2018)

Noise for stochastic detail



Styles fixed, only the noise changes

Changes: hair strands, freckles, background

Fixed: identity, pose, face shape

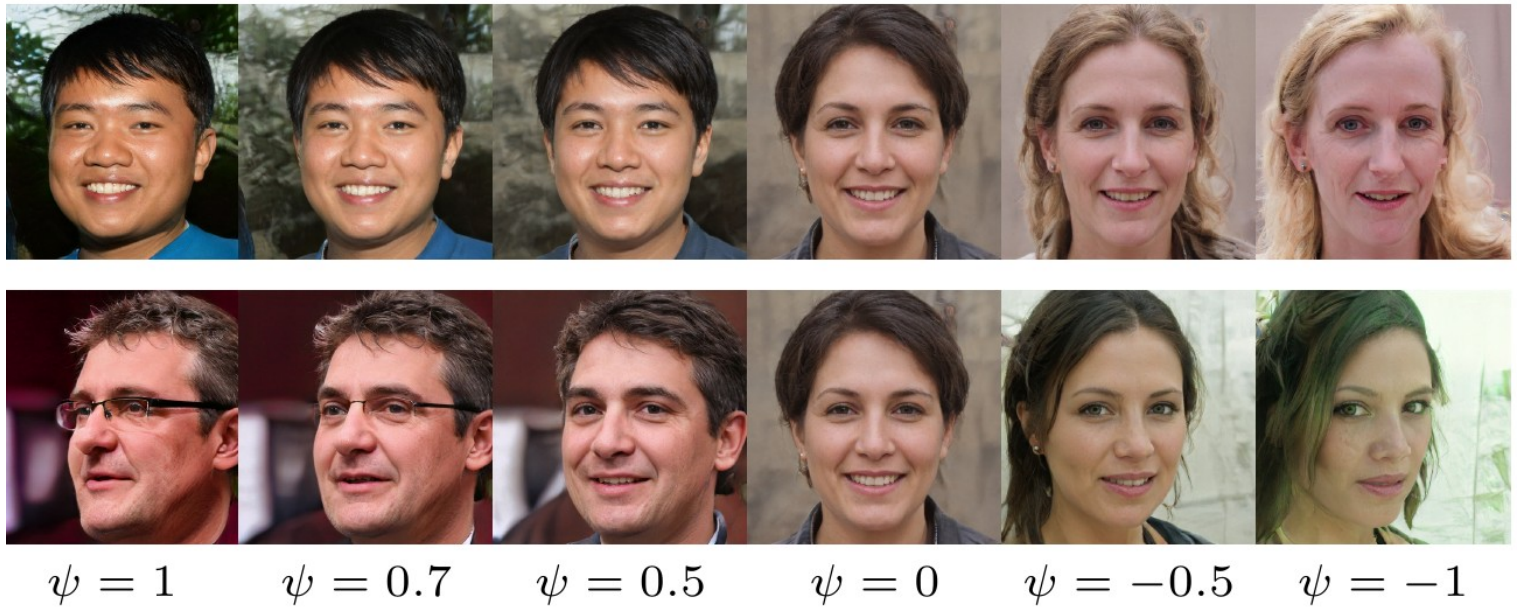
(c): per-pixel standard deviation over 100 noise samples

Source: StyleGAN (2018)

The truncation trick

$$w' = \bar{w} + \psi\,(w - \bar{w})$$

$\bar{w}$: average w; ψ: strength, chosen at sampling time



$\psi < 1$: more typical and cleaner, less varied

$\psi = 0$: the average face

Measured in Section 06

Source: StyleGAN (2018)

StyleGAN2: normalization artifacts



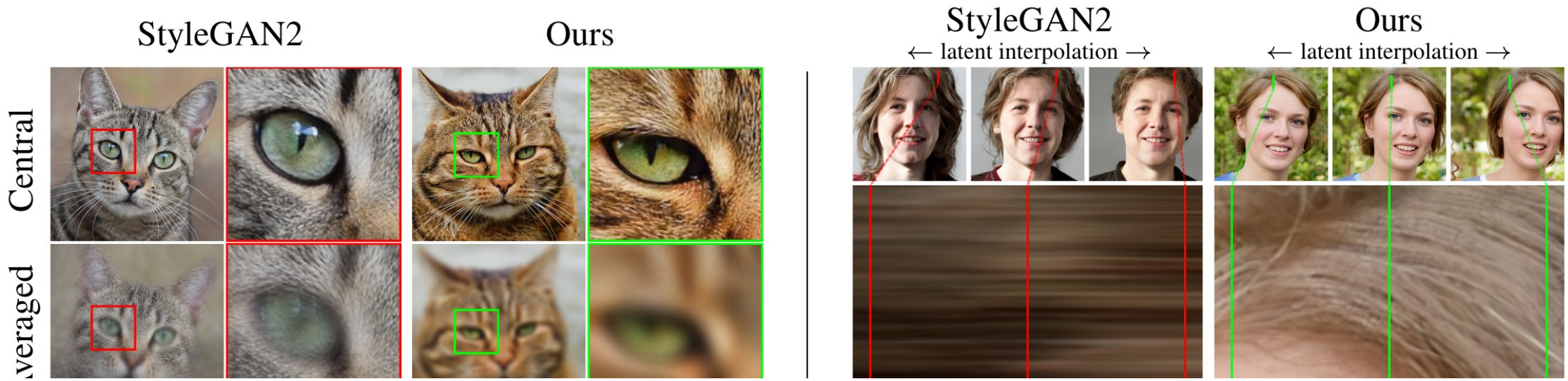
“Droplet” artifacts, clearest in the feature maps (right of each pair)

Cause: per-channel normalization lets one spike control the scaling

Fix: normalize the convolution weights instead (demodulation)

Source: StyleGAN2 (2019)

StyleGAN3: details that move with the object



StyleGAN2: hair and whiskers stick to fixed pixel positions

StyleGAN3: filtered layers, so details move with the face

Source: StyleGAN3 (2021)

Lecture outline

01 Learning from a discriminator

02 Why GAN training is difficult

03 Conditional generation and image translation

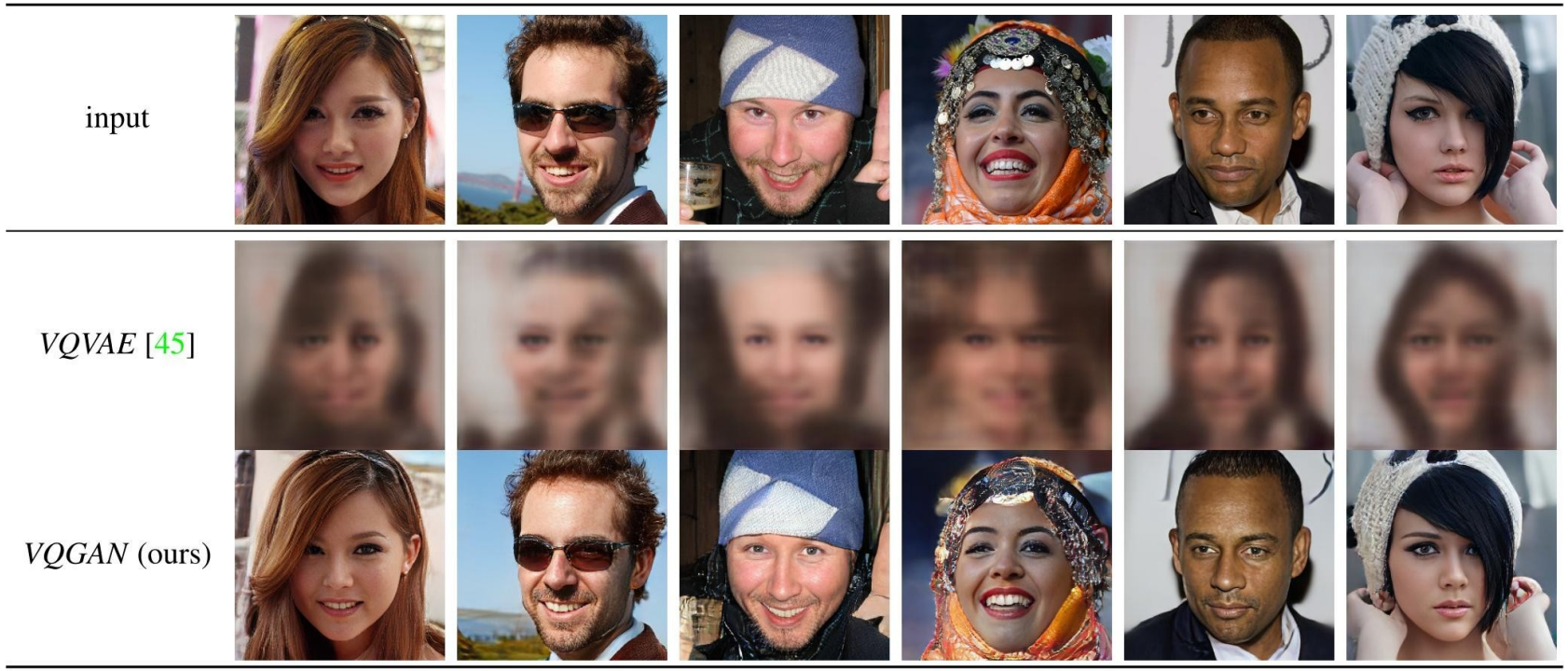
04 Style-based generators

05 Adversarial losses in other models

06 Evaluating generators

An adversarial loss as a realism term

pix2pix: a content loss plus a realism discriminator. The same helps autoencoders.



input

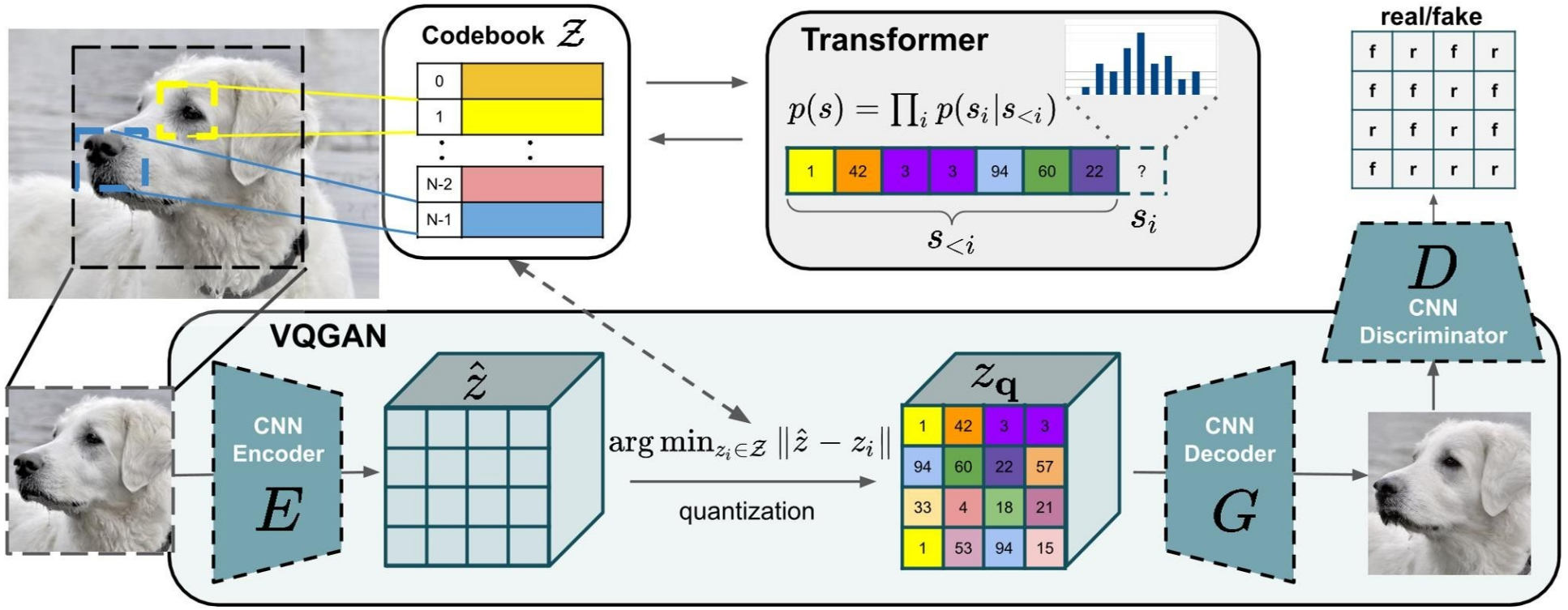
VQ-VAE (L15), pixel loss: blurry

VQGAN, + perceptual and adversarial: sharp

Same code-grid size: $16\times$ smaller than the image in height and width

Source: Taming Transformers (2021)

VQGAN: a VQ autoencoder with a discriminator



E, codebook $\mathcal{Z}$, decoder G: vector quantization as in L15

Patch discriminator D judges the reconstructions

Top: a Transformer over code indices (two slides ahead)

Source: Taming Transformers (2021)

VQGAN training objective

$$\mathcal{L} = \underbrace{\|x - \hat{x}\|_1 + \mathcal{L}_{\text{perc}}(x, \hat{x})}_{\text{reconstruction}} + \underbrace{\|\text{sg}[E(x)] - z_q\|^2 + \beta \|\text{sg}[z_q] - E(x)\|^2}_{\text{codebook and commitment (L15)}} + \underbrace{\lambda \mathcal{L}_{\text{GAN}}(\hat{x})}_{\text{adversarial}}$$

Reconstruction	keeps the content; LPIPS: distance between CNN features (L07)
Codebook	as in VQ-VAE; sg: stop-gradient
Adversarial	realistic local texture; patch D trained in alternation

$$\lambda = \frac{\|\nabla_{G_L} \mathcal{L}_{\text{rec}}\|}{\|\nabla_{G_L} \mathcal{L}_{\text{GAN}}\| + \delta}$$

Balances the two gradients at the decoder's last layer G_L

Source: Taming Transformers (2021) · LPIPS (2018)

Generating images with a learned tokenizer

1 **Train the VQGAN**

images $\leftrightarrow$ grids of code indices (tokens)

2 **Train a prior on the tokens**

VQGAN frozen; a Transformer predicts the next index (L15)

3 **Generate**

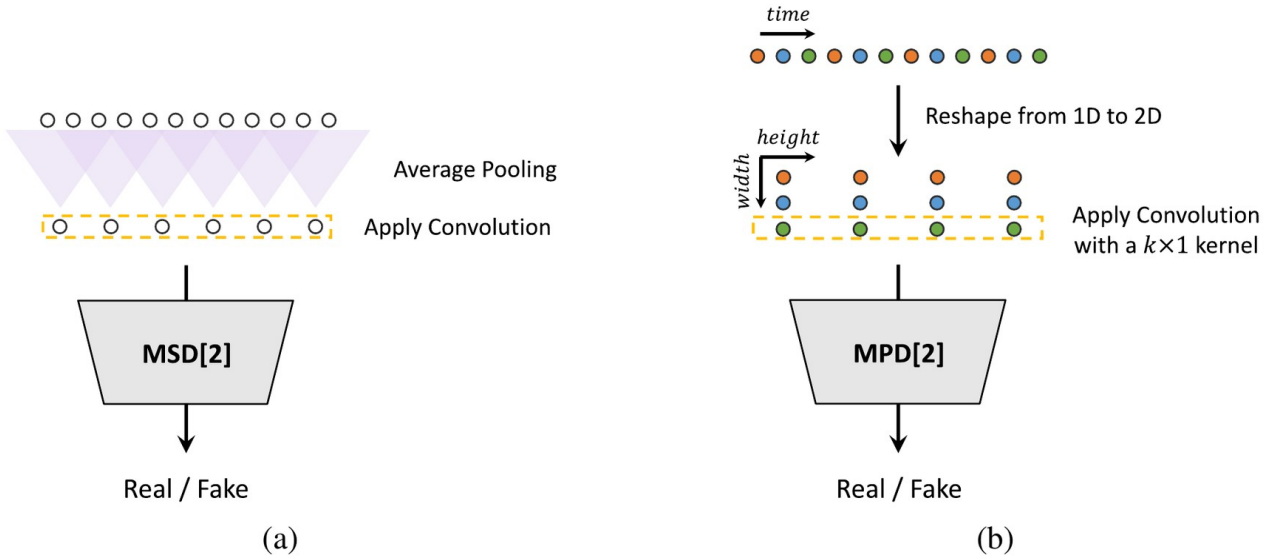
sample tokens, decode with G

Prior: which codes go where. Decoder: the detail.

Latent diffusion (L18) uses the same kind of autoencoder

Source: Taming Transformers (2021)

Waveforms: HiFi-GAN



Vocoder: mel-spectrogram $\rightarrow$ waveform (22,050 samples per second)

L1 on mel-spectrograms plus adversarial losses

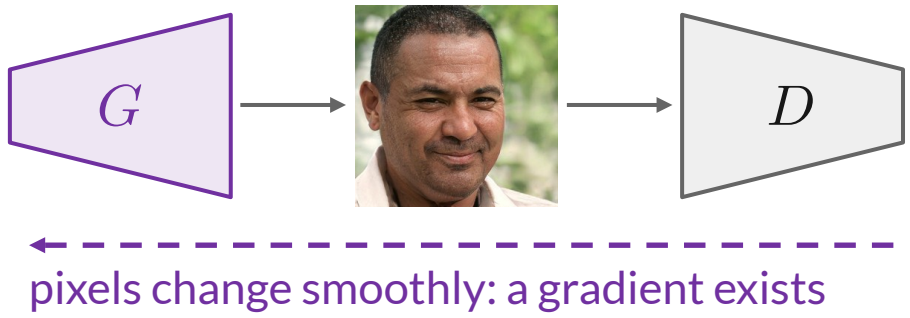
Discriminators at several scales (a) and periods p (b)

One forward pass: much faster than sample-by-sample generation

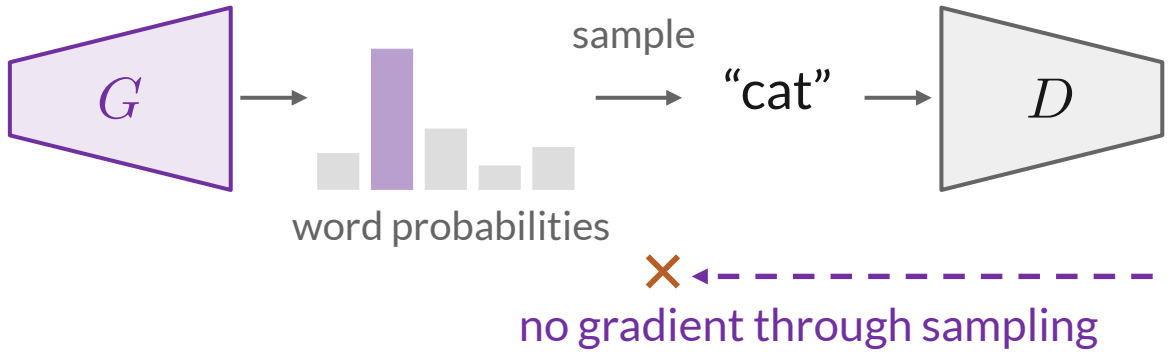
Source: HiFi-GAN (2020)

Why GANs rarely generate text

Image



Text



G learns from the gradient of D's score with respect to the sample

A sampled word cannot change slightly

Text: likelihood-trained autoregressive models

Lecture outline

01 Learning from a discriminator

02 Why GAN training is difficult

03 Conditional generation and image translation

04 Style-based generators

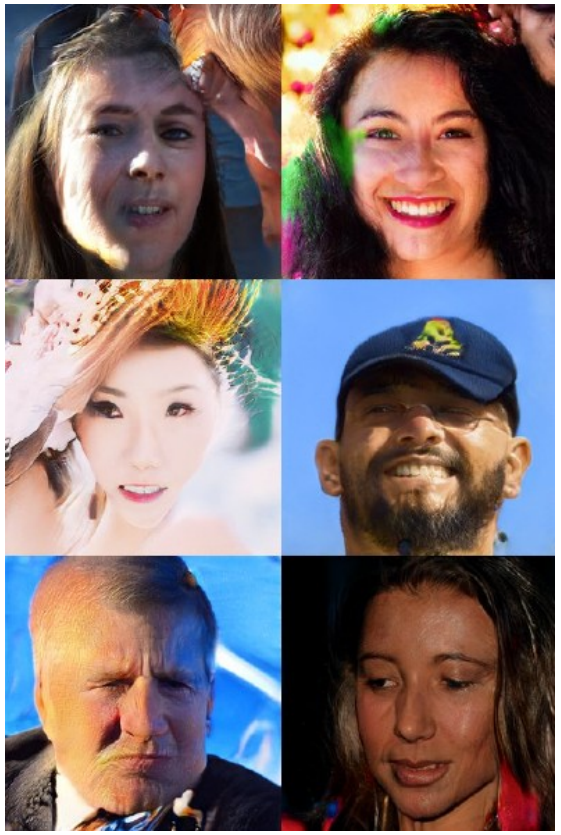
05 Adversarial losses in other models

06 Evaluating generators

Quality and coverage



plausible but similar



varied, some artifacts

Quality: does each sample look real?

Coverage: are all kinds of data generated?

Mode collapse fails coverage

Losses and a few samples cannot tell

Source: Improved Precision and Recall (2019)

Comparing collections in a feature space

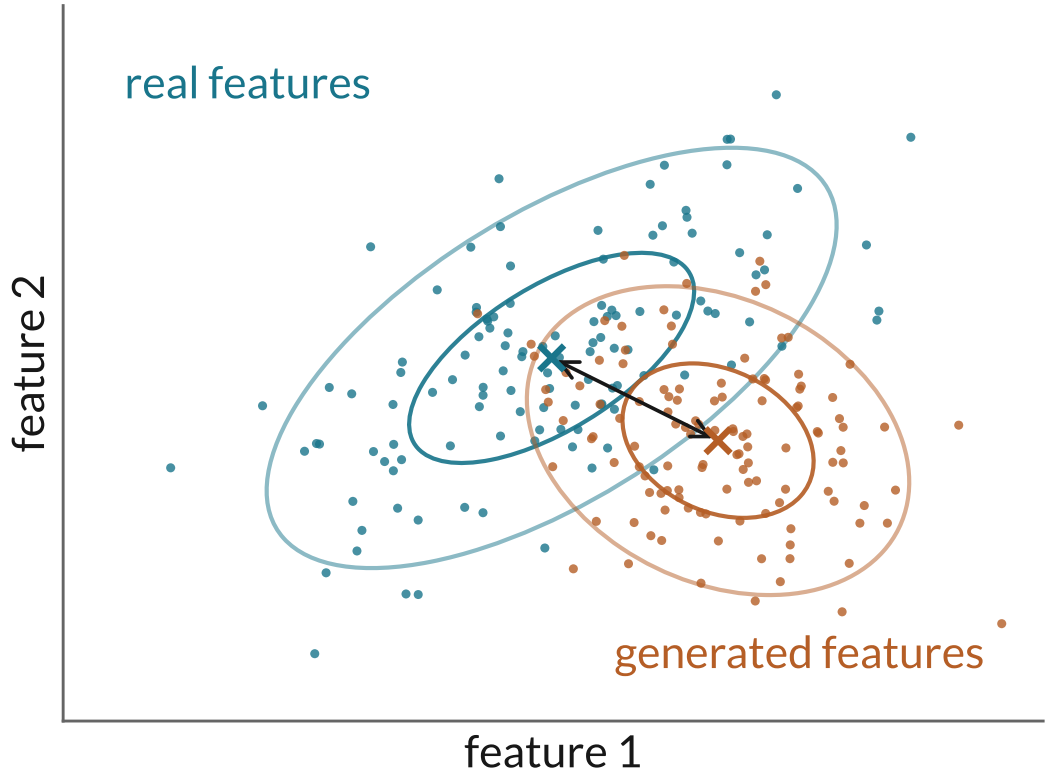


Illustration in two dimensions

Features from a fixed classifier (Inception-v3), 2048-D

Features describe content, not exact pixels (L07)

Summarize each set by its mean μ and covariance Σ

Fréchet Inception Distance (FID)

$$\mathrm{FID} = \underbrace{\|\mu_r - \mu_g\|_2^2}_{\text{different means}} + \underbrace{\mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right)}_{\text{different spreads and correlations}}$$

r: real, g: generated; Tr: sum of the diagonal

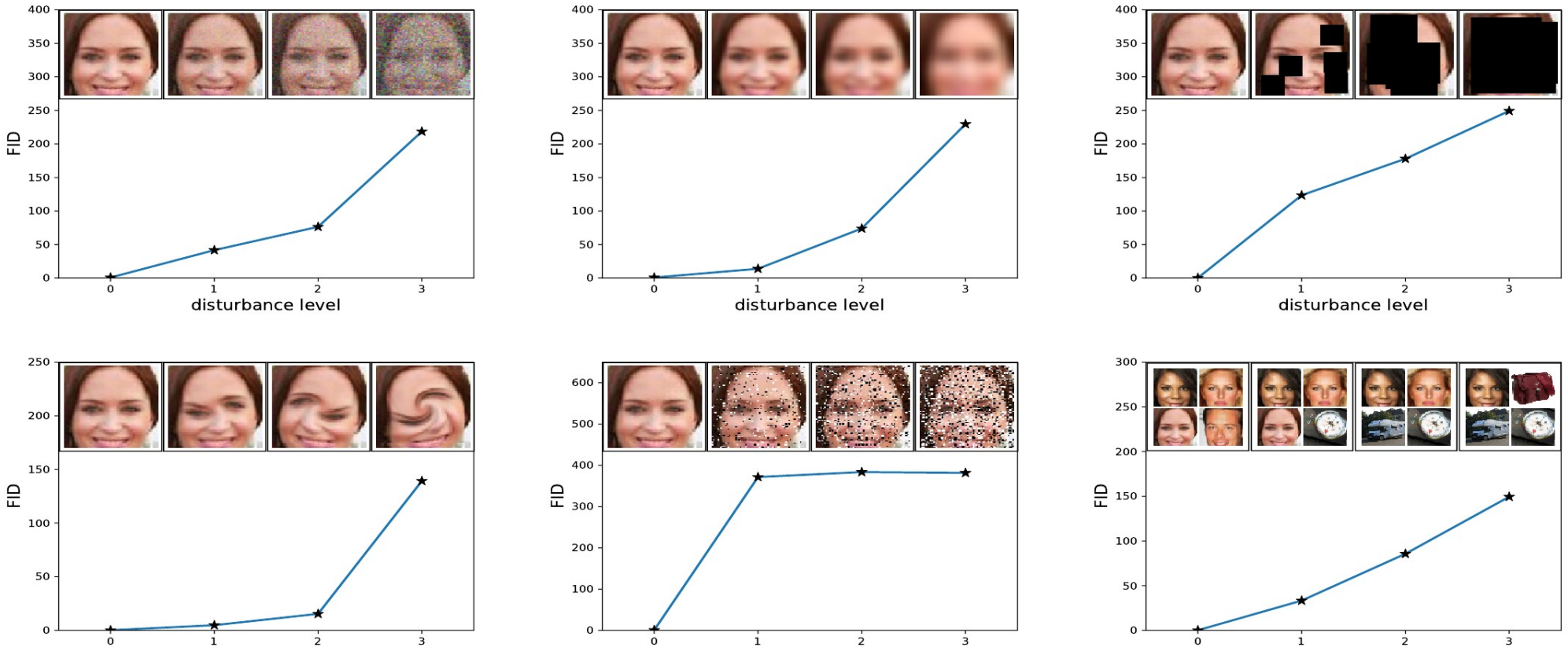
Lower is better; 0 for identical summaries

Detects artifacts (means move) and missing modes (spread shrinks)

Compare only with the same network, sample count and preprocessing

Source: FID (2017)

FID increases with image disturbances

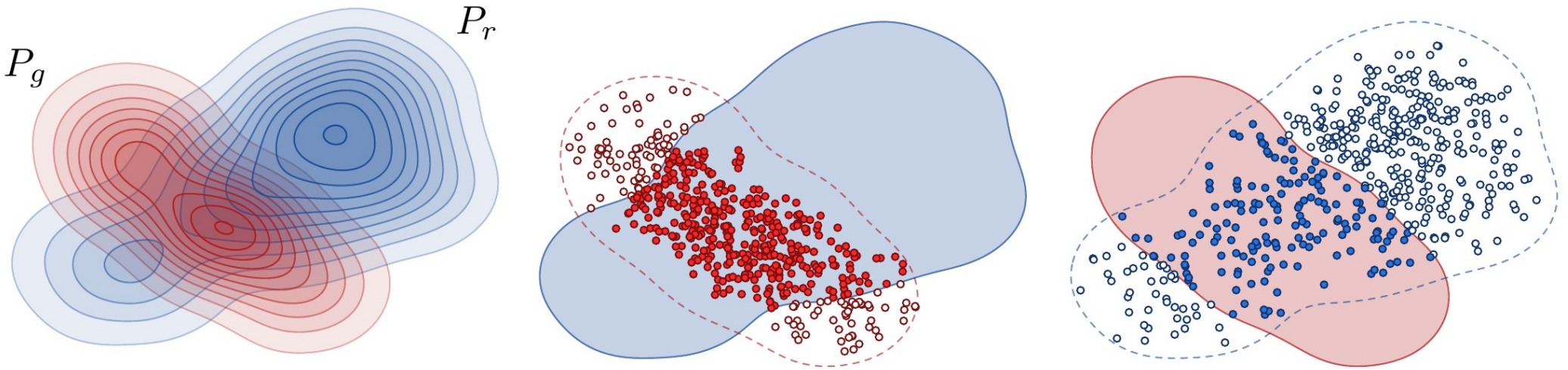


Real faces corrupted more and more strongly: FID rises in every case

noise, blur, black rectangles / swirl, salt-and-pepper, ImageNet mixed in

Source: FID (2017)

Precision and recall for generative models



(a) Example distributions (b) Precision (c) Recall

Precision: generated samples inside the real region (quality)

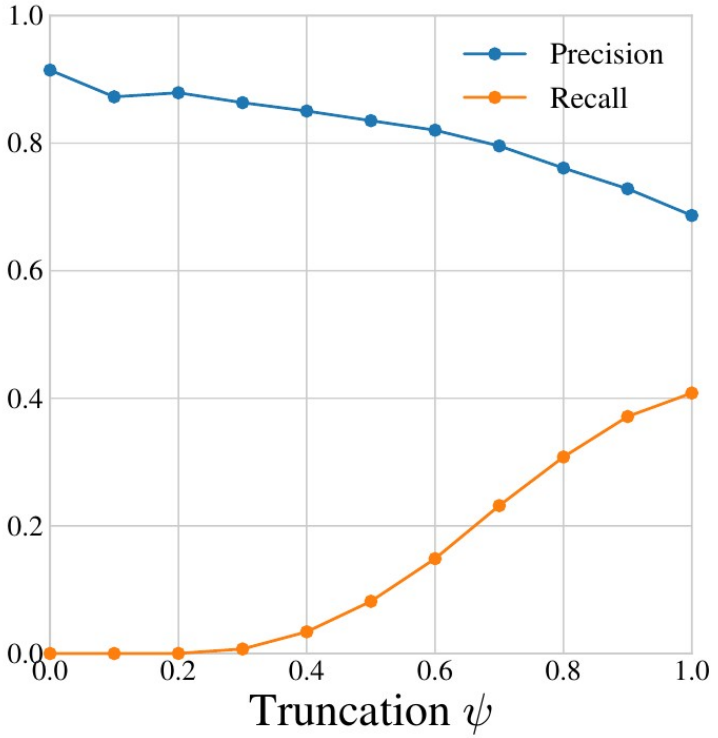
Recall: real samples inside the generated region (coverage)

Regions: union of k-nearest-neighbor balls in feature space

Source: Improved Precision and Recall (2019)

Truncation trades coverage for quality





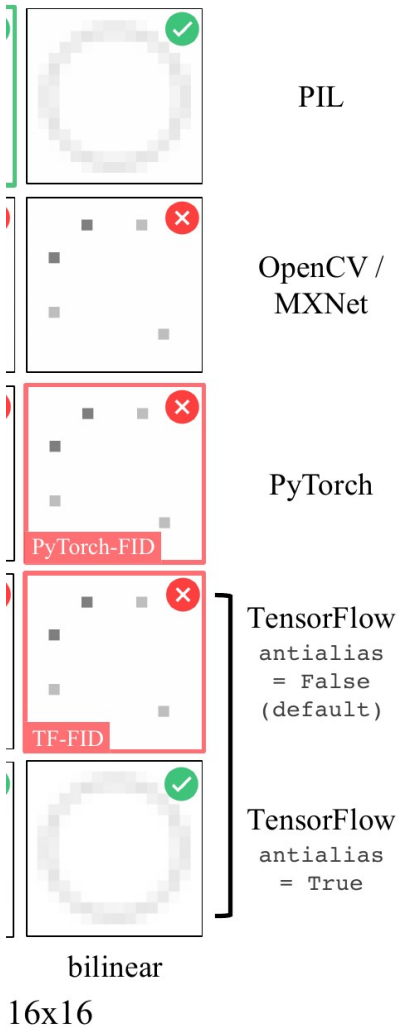
StyleGAN, truncation ψ (Section 04)

Smaller ψ: precision up, recall down

Report quality and coverage separately

Source: Improved Precision and Recall (2019)

The evaluation pipeline matters



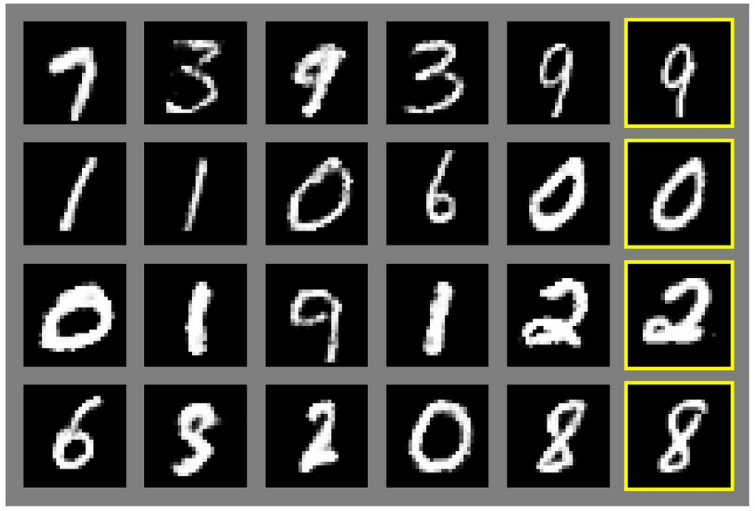
Images are resized before feature extraction

$128 \times 128 \rightarrow 16 \times 16$ in different libraries: clean or aliased

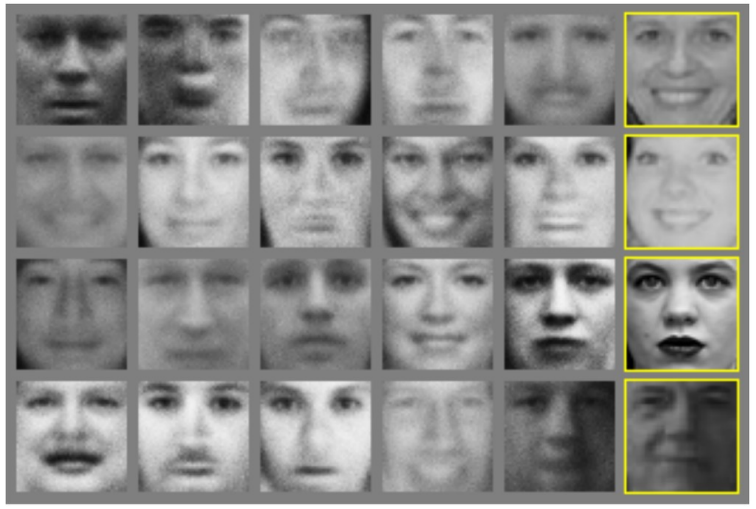
Small pipeline differences shift FID: use one fixed implementation

Source: Clean-FID (2022)

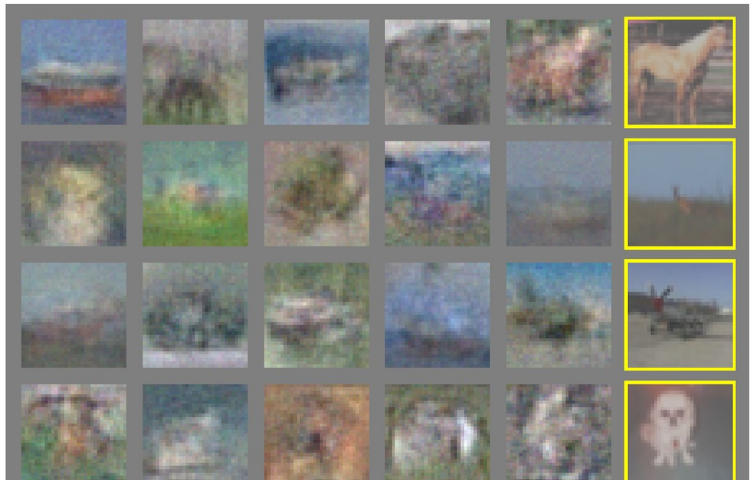
Has the generator memorized its training set?

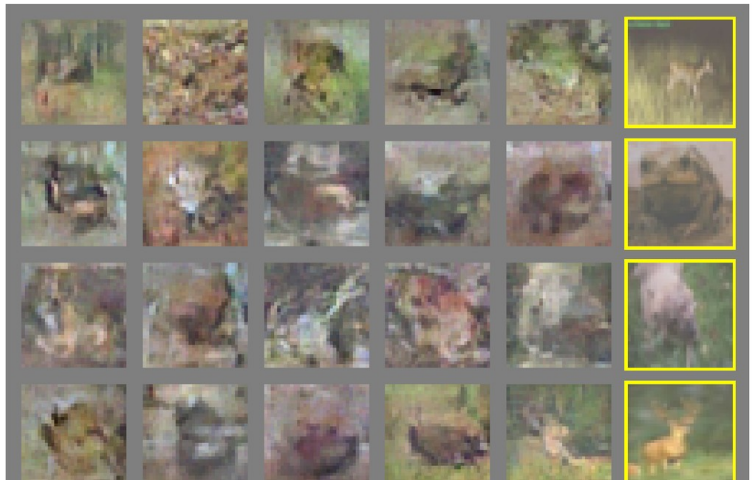


a)



b)





Yellow: nearest training image to each sample

They differ: not copies

A few checks cannot rule out copying rare images

Source: GAN (2014)

Summary

- **Learned comparison:** D's input gradient trains G
- **Training:** alternate updates, non-saturating loss; losses do not show quality
- **Difficulties:** moving objectives, mode collapse, uninformative gradients
- **Control:** conditions (pix2pix, CycleGAN); styles and noise (StyleGAN)
- **Component:** adversarial realism term in VQGAN and vocoders
- **Evaluation:** quality and coverage of collections: FID, precision, recall

Next time: diffusion models

Training

Add noise to real images; learn to remove it. Regression, no game.

Generation

Start from noise; many small denoising steps.

Stable training and good coverage, but many network evaluations per sample