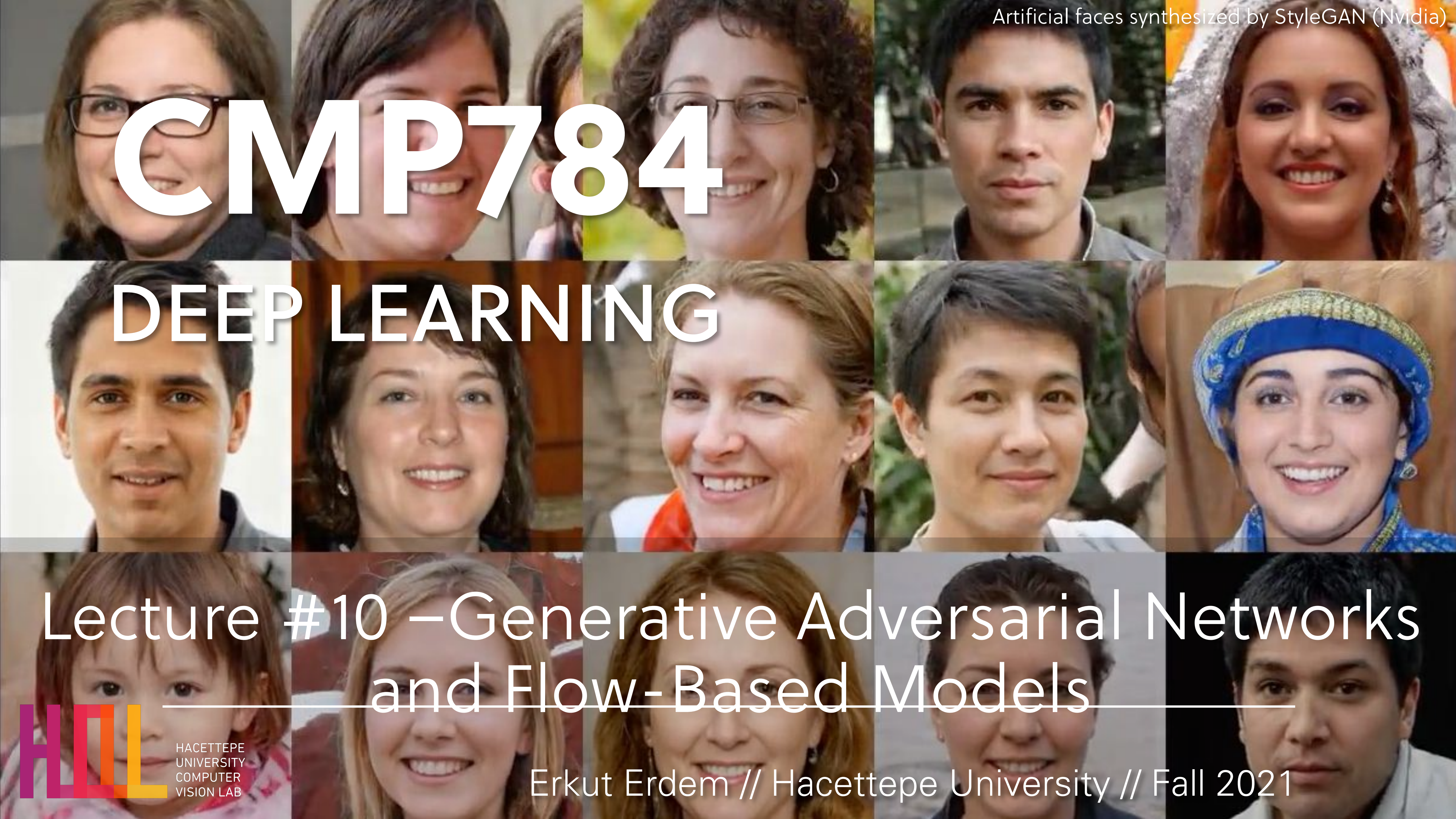


Artificial faces synthesized by StyleGAN (Nvidia)

CMP784

DEEP LEARNING

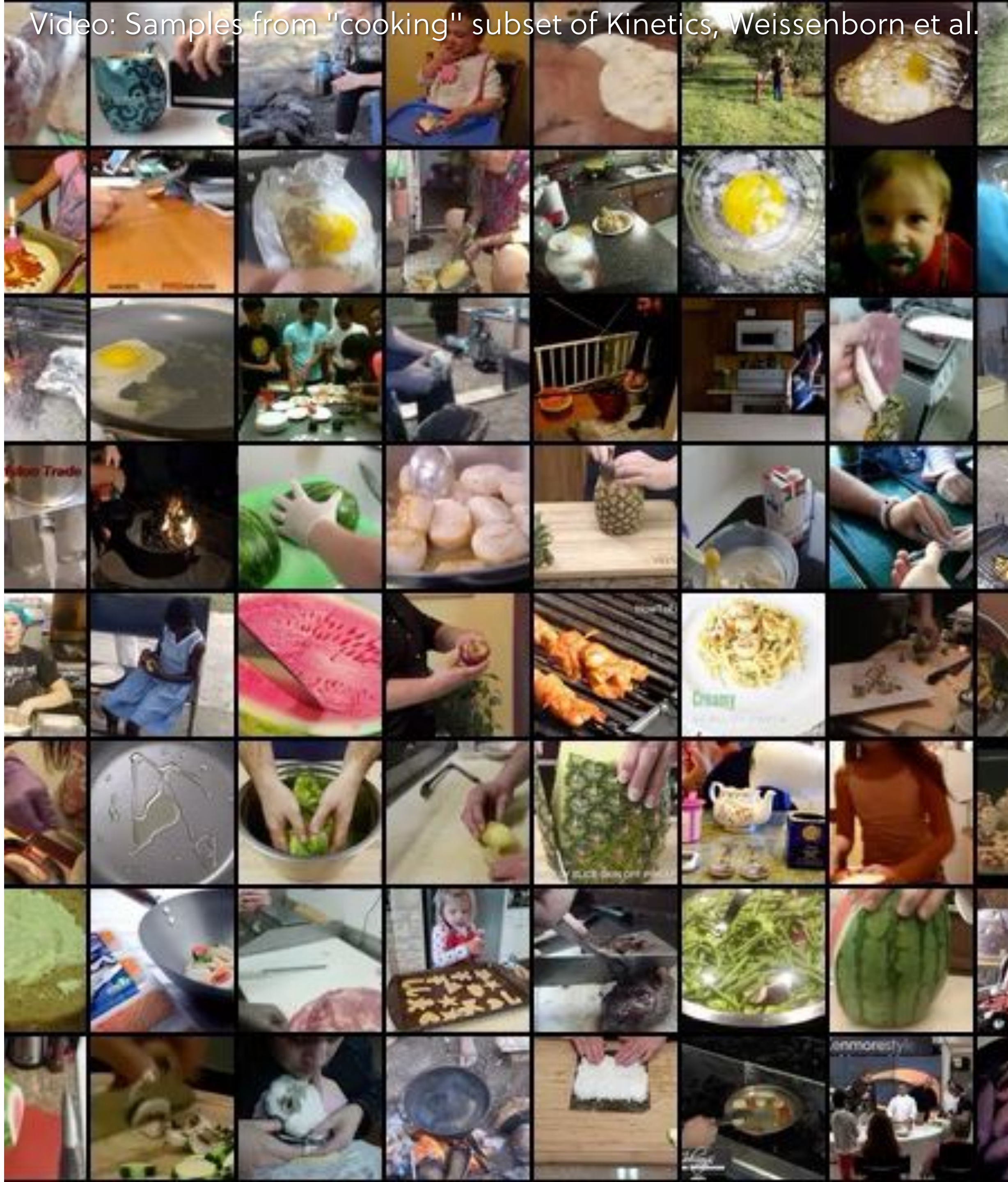
Lecture #10 –Generative Adversarial Networks and Flow-Based Models

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Erkut Erdem // Hacettepe University // Fall 2021

Previously on CMP784

- Supervised vs. Unsupervised Representation Learning
- Sparse Coding
- Autoencoders
- Autoregressive Generative Models



Video: Samples from "cooking" subset of Kinetics, Weissenborn et al.

Lecture overview

- Generative Adversarial Networks (GANs)
- Normalizing Flow Models

Disclaimer: Some of the material and slides for this lecture were borrowed from

—Ian Goodfellow's tutorial on "Generative Adversarial Networks"

—Aaron Courville's IFT6135 class

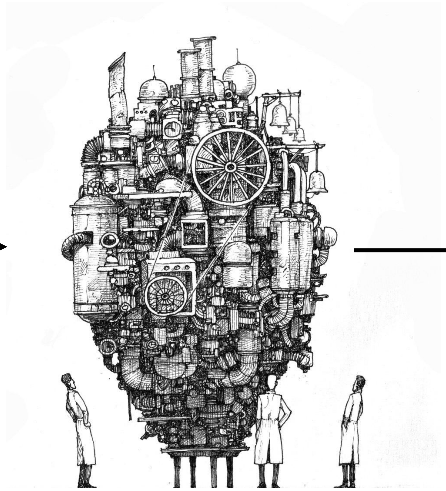
—Bill Freeman, Antonio Torralba and Phillip Isola's MIT 6.869 class

—Chin-Wei Huang slides on Normalizing Flows

Discriminative vs. Generative Models

$$p(y|x)$$

$$p(x|y)$$

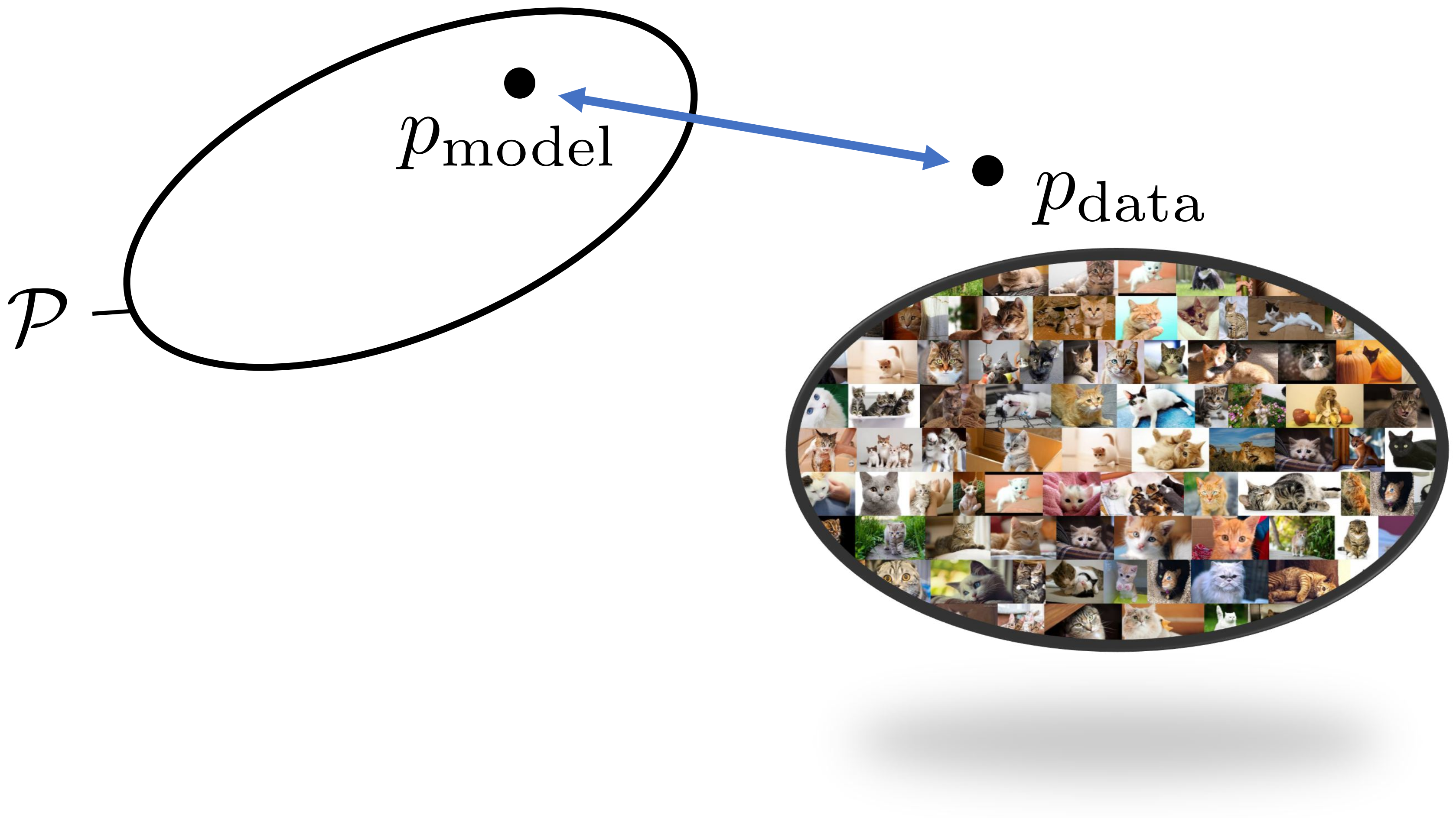


"Cat"

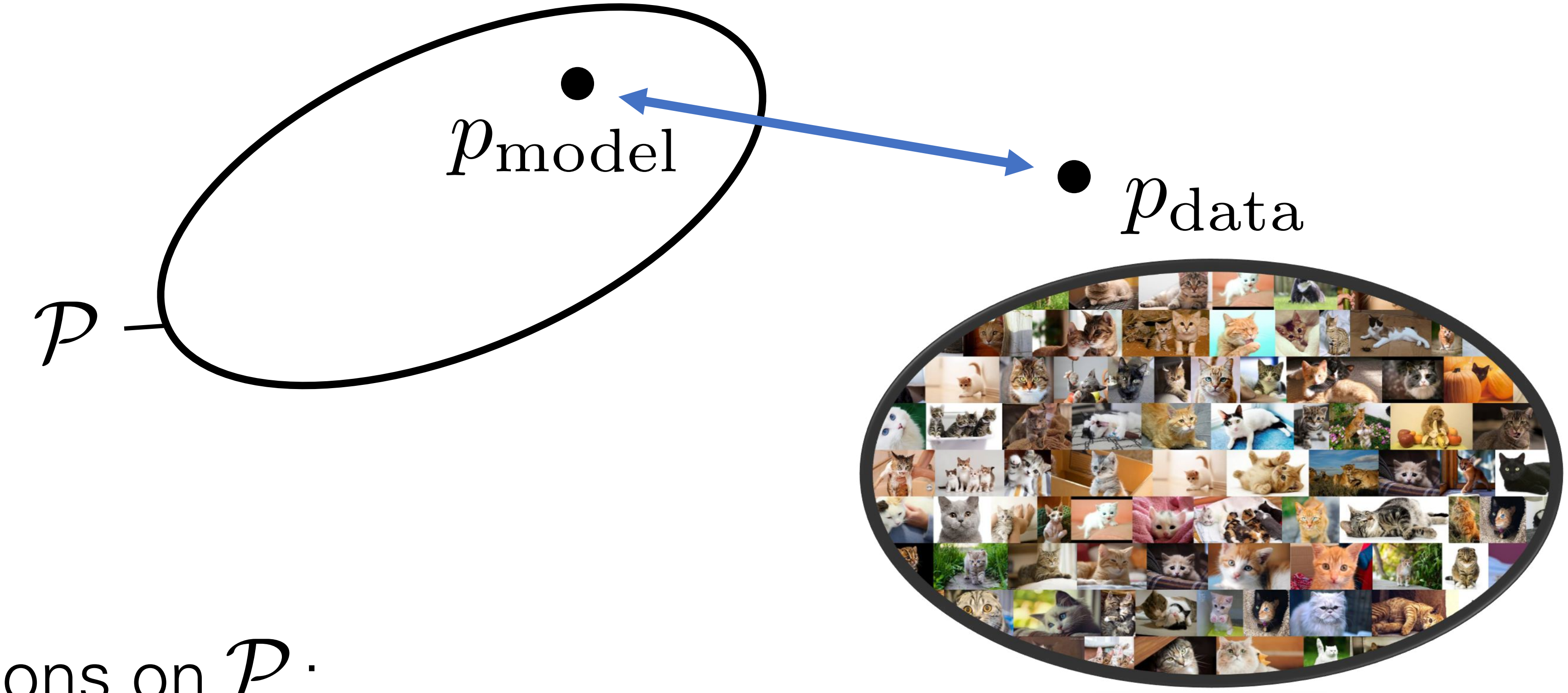
Discriminative models

Generative models

Generative Modeling

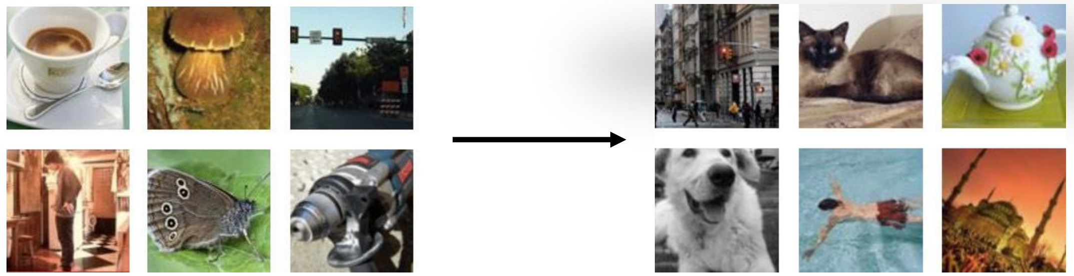


Generative Modeling



Assumptions on $\mathcal{P}$:

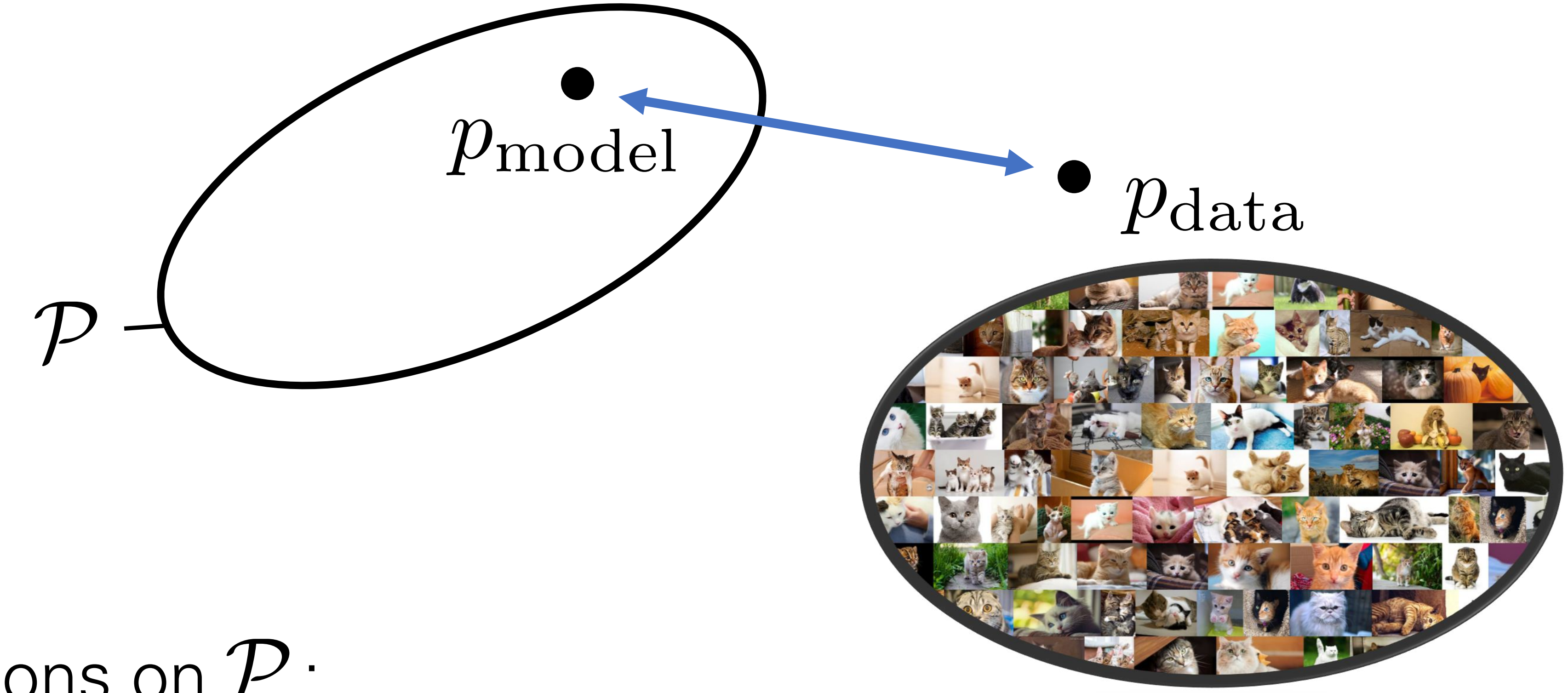
- tractable sampling



Training examples

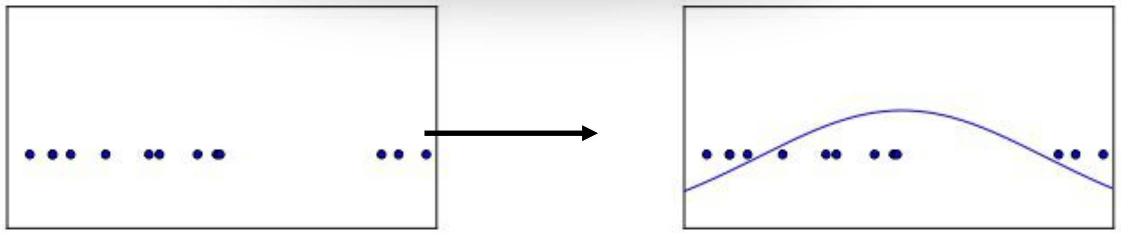
Model samples

Generative Modeling



Assumptions on $\mathcal{P}$:

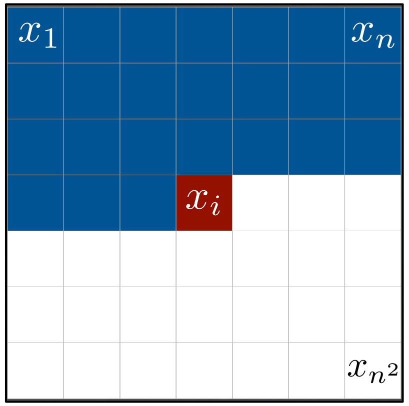
- tractable sampling
- tractable likelihood function



Broad Categories of Generative Models

- Autoregressive Models
- Generative Adversarial Networks (GANs)
- Flow-based Models
- Variational Autoencoders
- Energy-based Models

Autoregressive Models



- Explicitly model conditional probabilities:

$$p_{\text{model}}(\boldsymbol{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})$$

Each conditional can be a complicated neural net

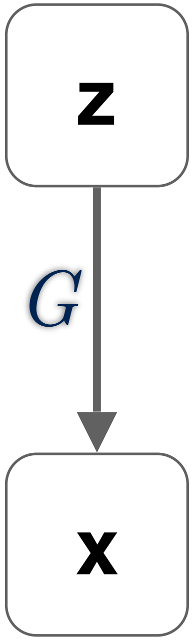
Disadvantages:

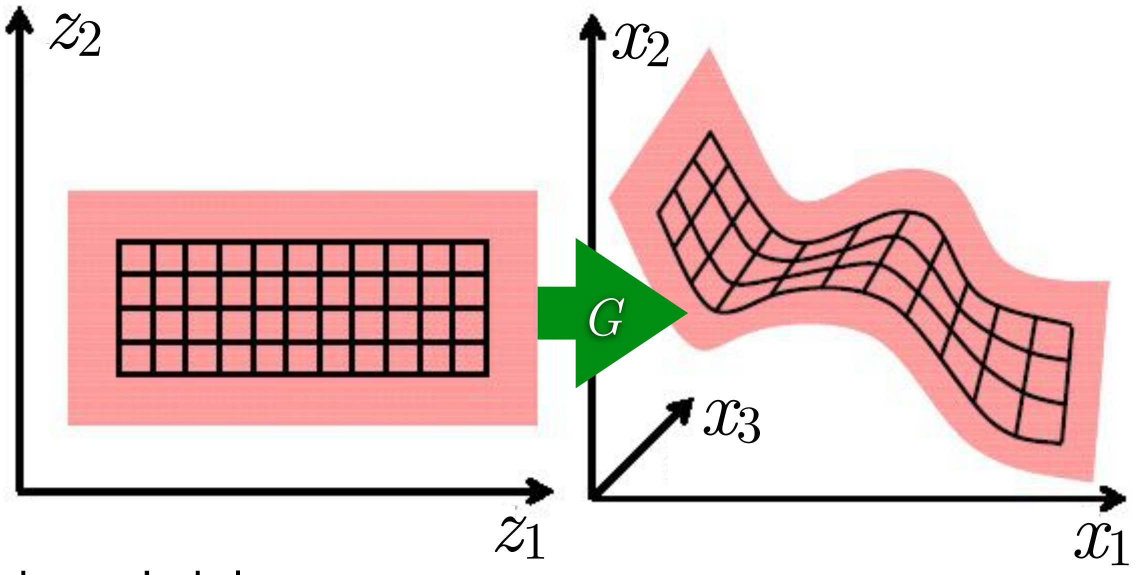
- Generation can be too costly
- Generation can not be controlled by a latent code

PixelCNN elephants
(van den Ord et al. 2016)

Another way to train a latent variable model

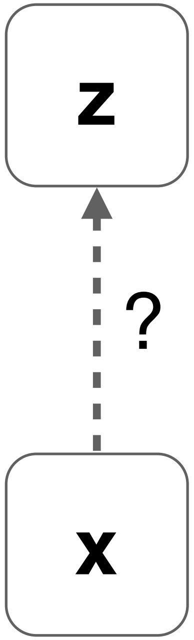
Latent variables





Observed variables

inference

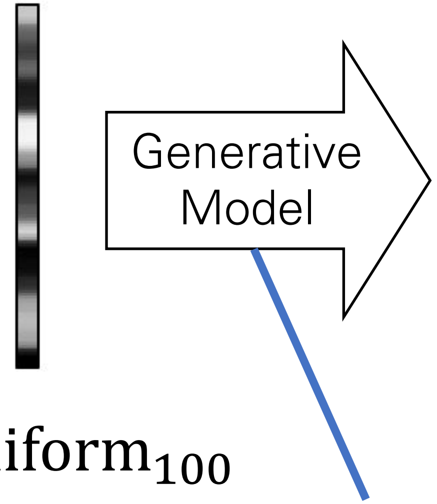


Generative Adversarial Networks

Genetive Adversarial Networks (GANs)

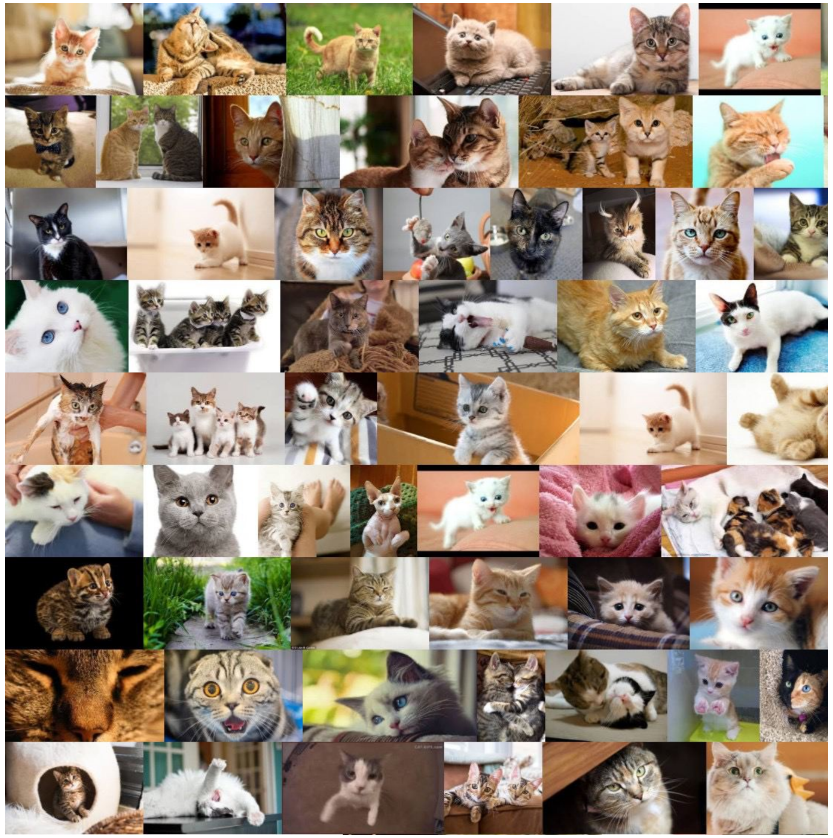
(Goodfellow et al., 2014)

Noise
(random input)



$z \sim \text{Uniform}_{100}$

think of this as
a transformation



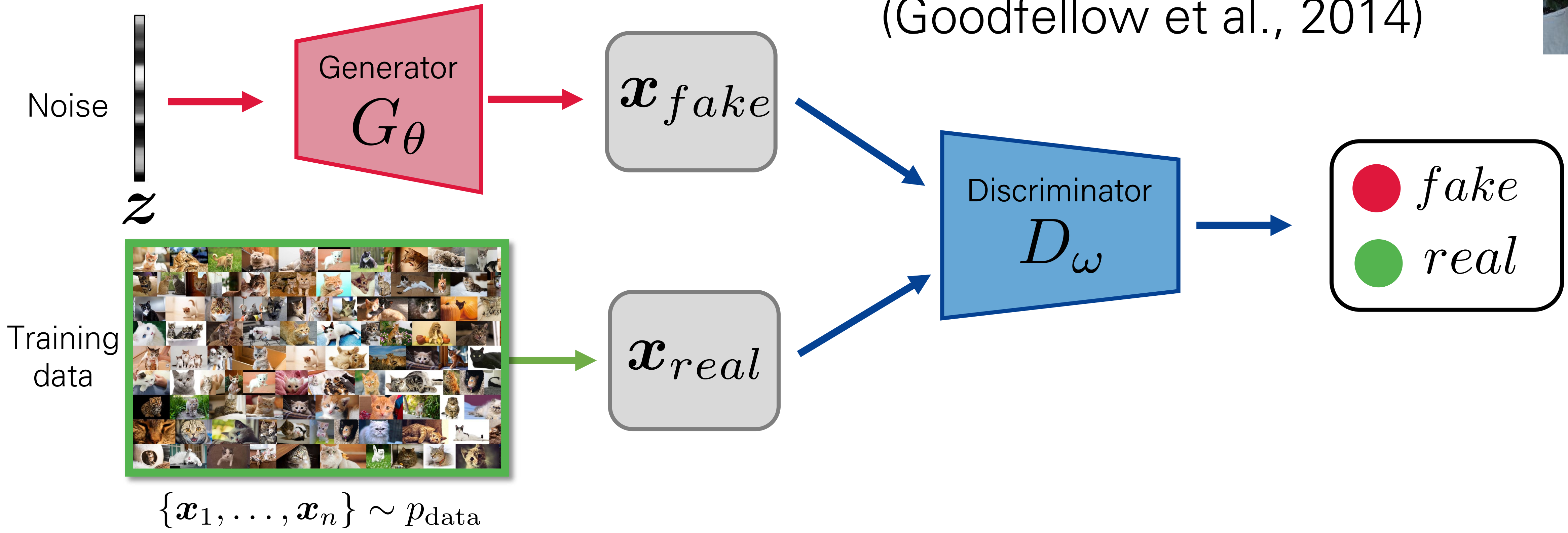
- A game-theoretic likelihood free model

Advantages:

- Uses a latent code
- No Markov chains needed
- Produces the best looking samples

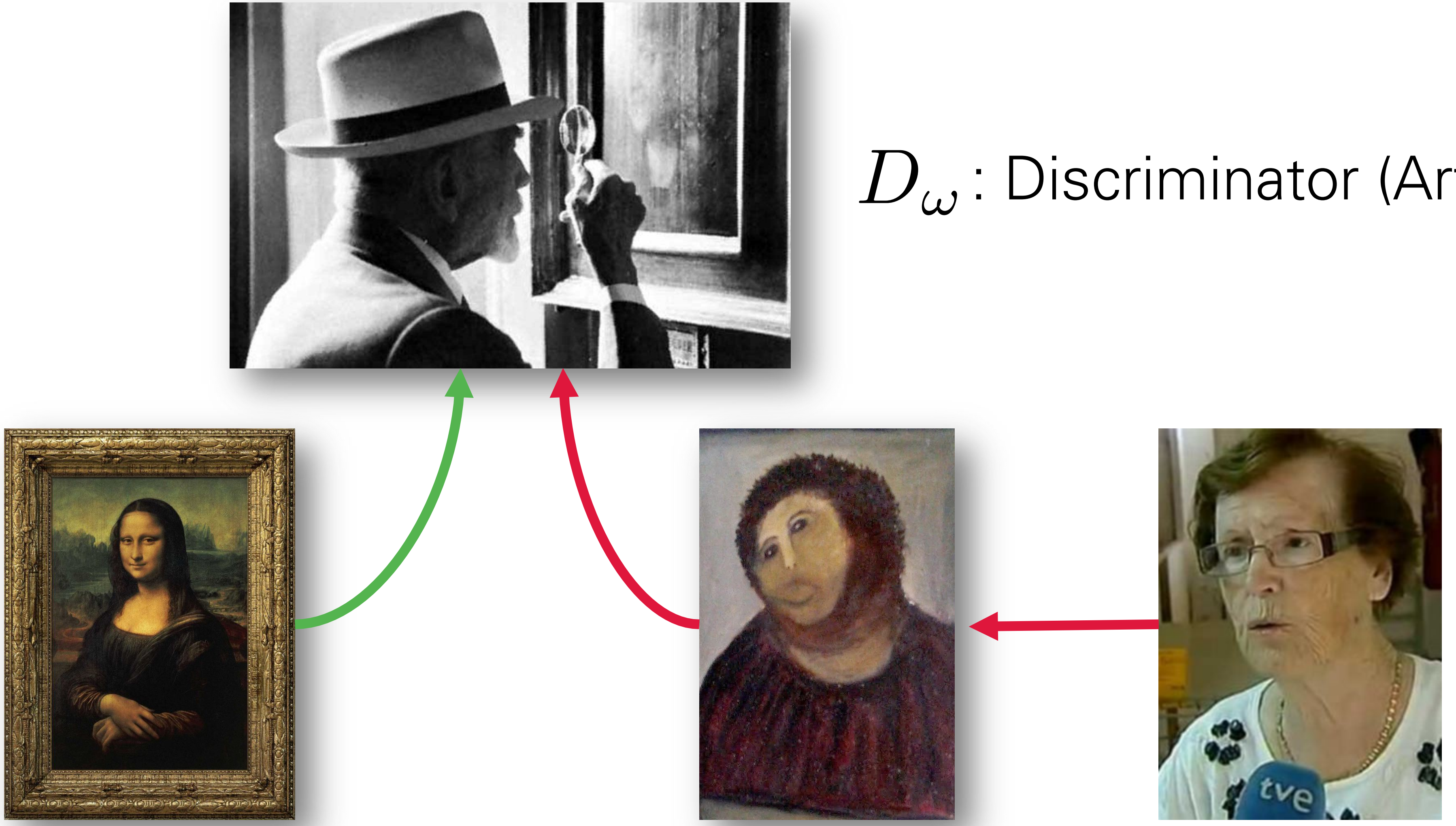
Genetive Adversarial Networks (GANs)

(Goodfellow et al., 2014)



- A game between a generator $G_\theta(\boldsymbol{z})$ and a discriminator $D_\omega(\boldsymbol{x})$
  - Generator tries to fool discriminator (i.e. generate realistic samples)
  - Discriminator tries to distinguish fake from real samples

Intuition behind GANs



D_ω: Discriminator (Art Critic)

$\boldsymbol{x}_{real}$

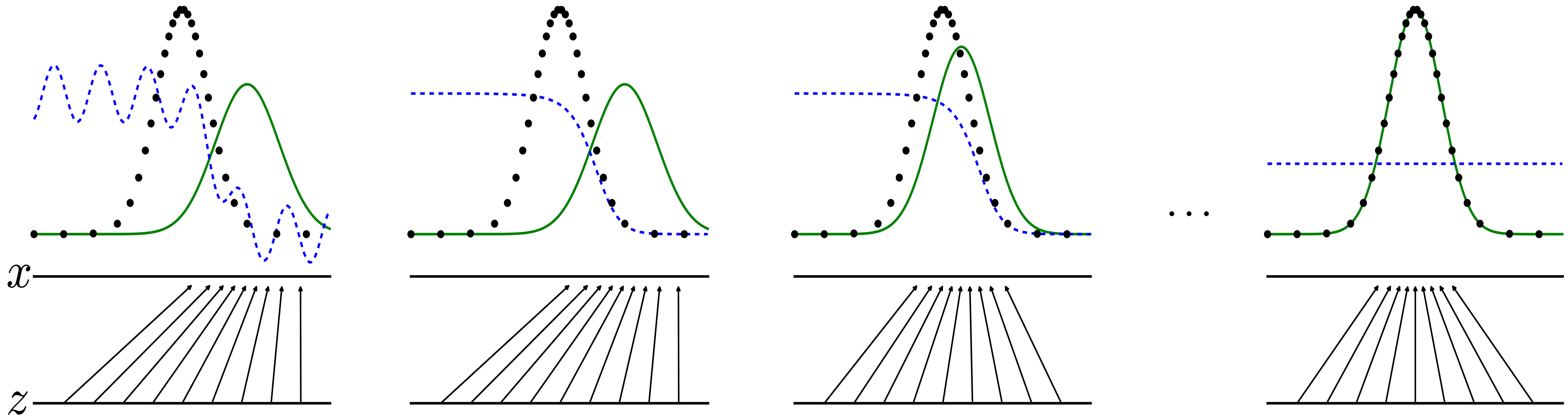
$\boldsymbol{x}_{fake}$

G_θ: Generator (Forger)

Training Procedure

(Goodfellow et al., 2014)

- Use SGD on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples



GAN Training: Minimax Game (Goodfellow et al., 2014)

$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log D_\omega(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \left[\log\left(1 - D_\omega(G_\theta(\boldsymbol{z}))\right)\right]$$

Real data

Noise vector used to generate data

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

Cross-entropy loss for binary classification

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D\left(G(\boldsymbol{z})\right)$$

Generator maximizes the log-probability of the discriminator being mistaken

- Equilibrium of the game
- **Minimizes the Jensen-Shannon divergence between p_{data} and p_x**

GAN Training: Minimax Game (Goodfellow et al., 2014)

$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D_\omega(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [\log (1 - D_\omega(G_\theta(\boldsymbol{z})))]$$

Real data

Noise vector used

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim}$$

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

ross-entropy
ss for binary
assification

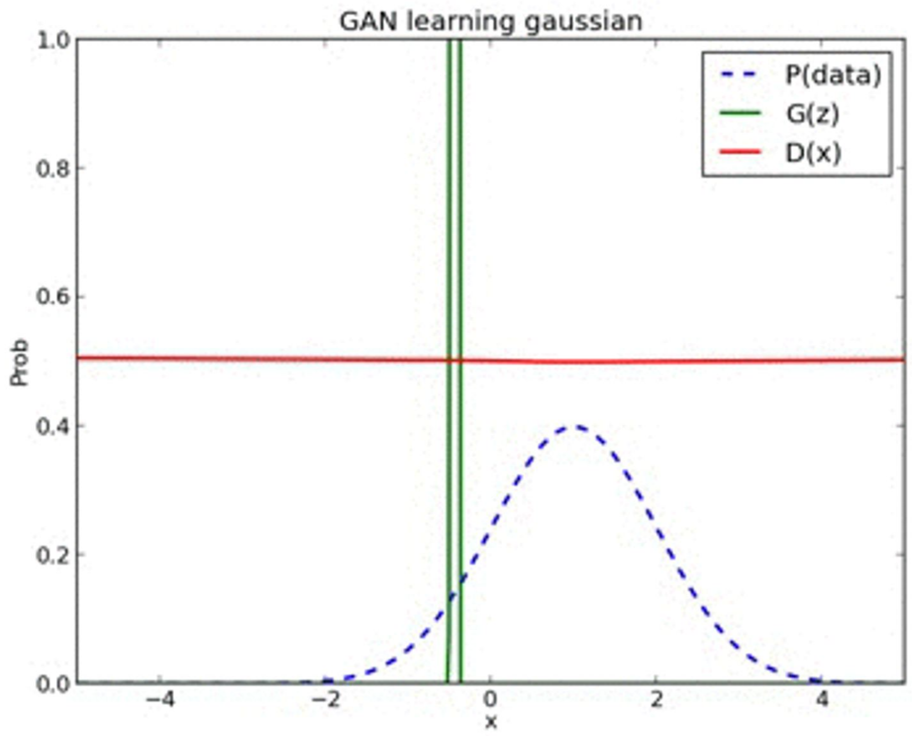
g-probability
of the discriminator being mistaken

Important question is
"Does this converge??"

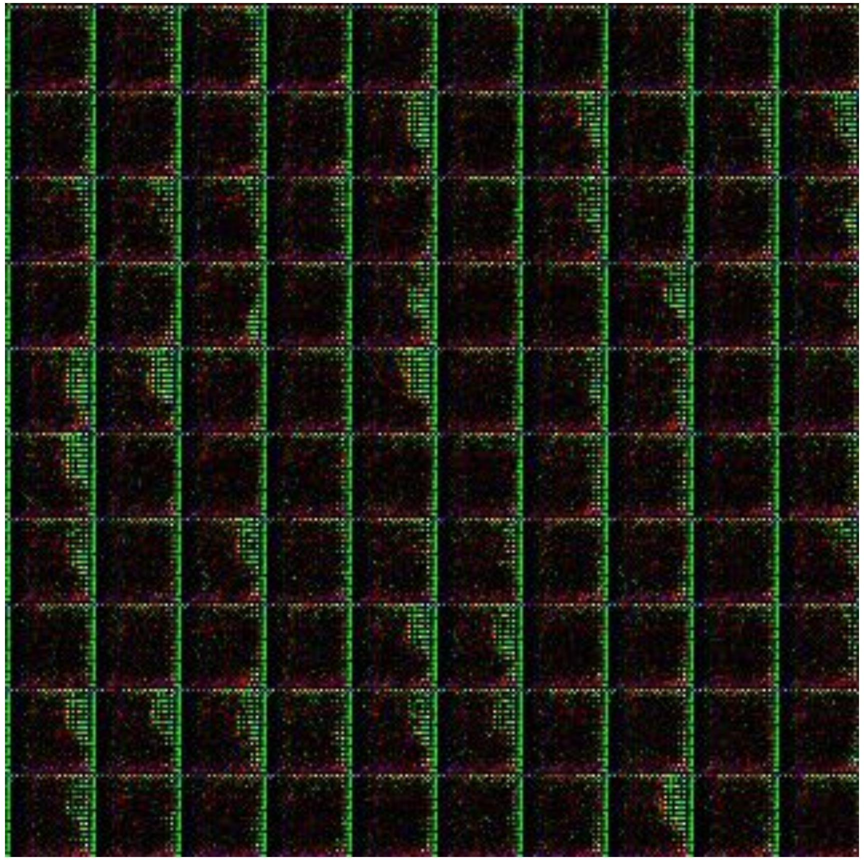
- Equilibrium of the game
- Minimizes the Jensen-Shannon divergence

Training Procedure

(Goodfellow et al., 2014)



Source: Alec Radford



Source: OpenAI blog

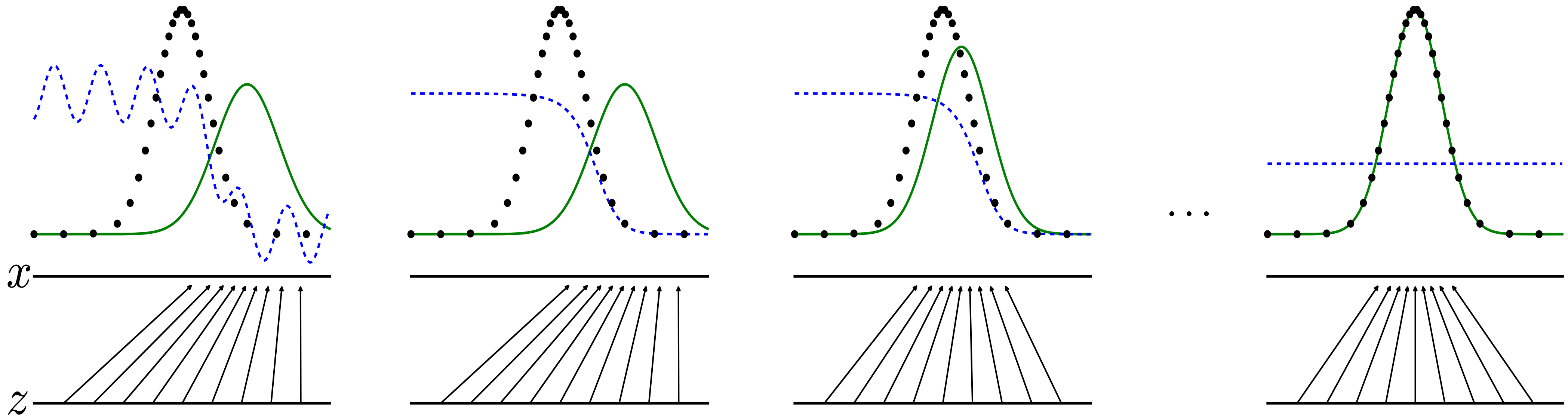
Generating 1D points

Generating images

Training Procedure

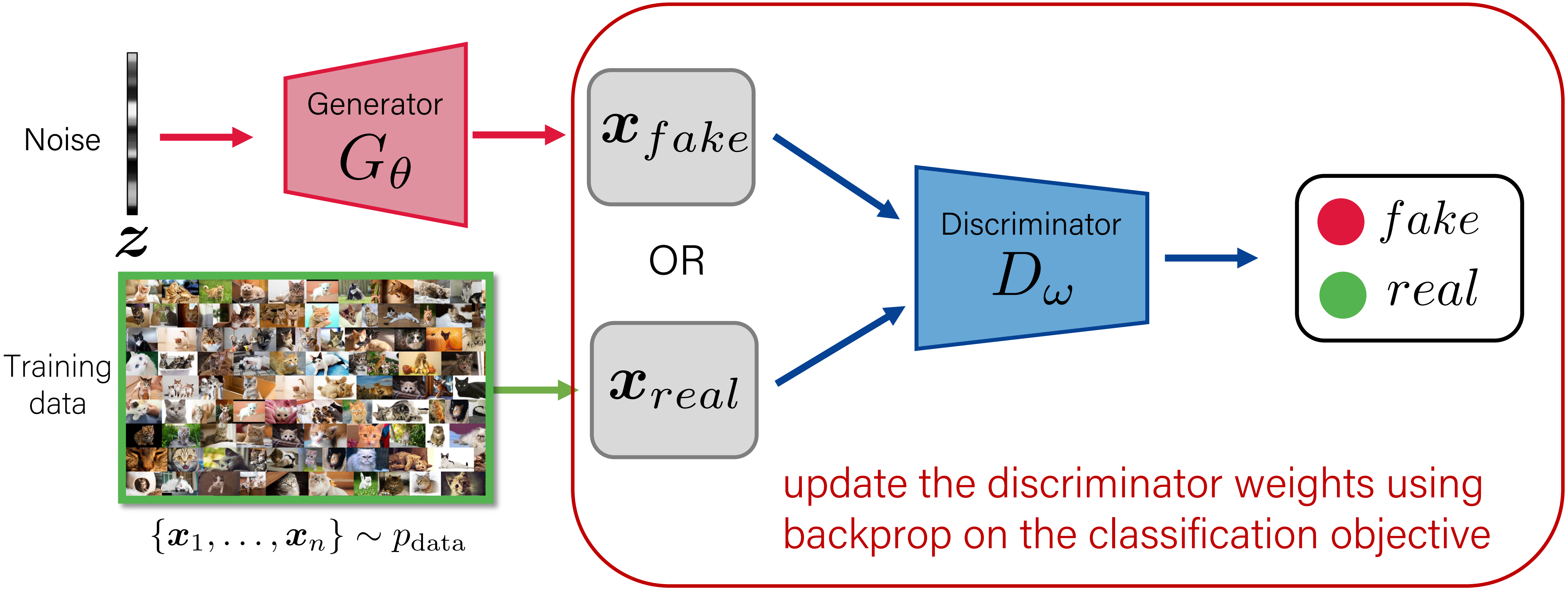
(Goodfellow et al., 2014)

- Use SGD on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples



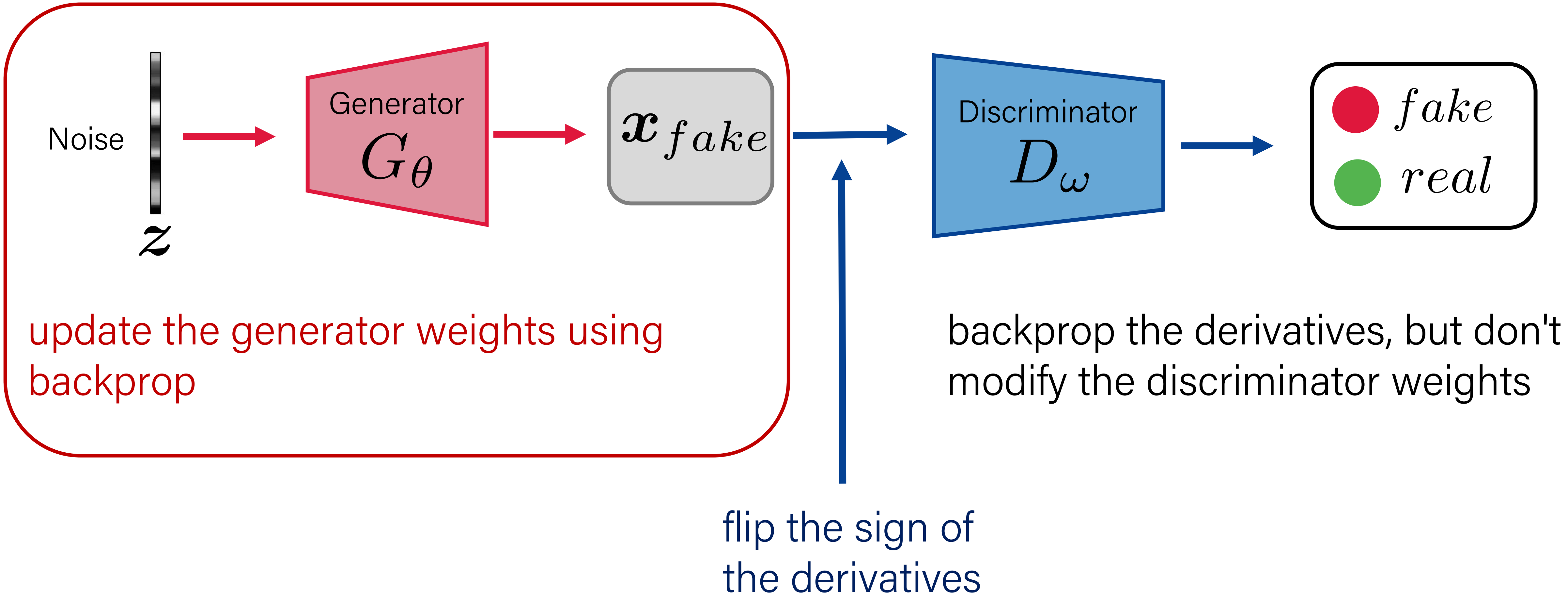
Training Procedure

- Updating the discriminator:



Training Procedure

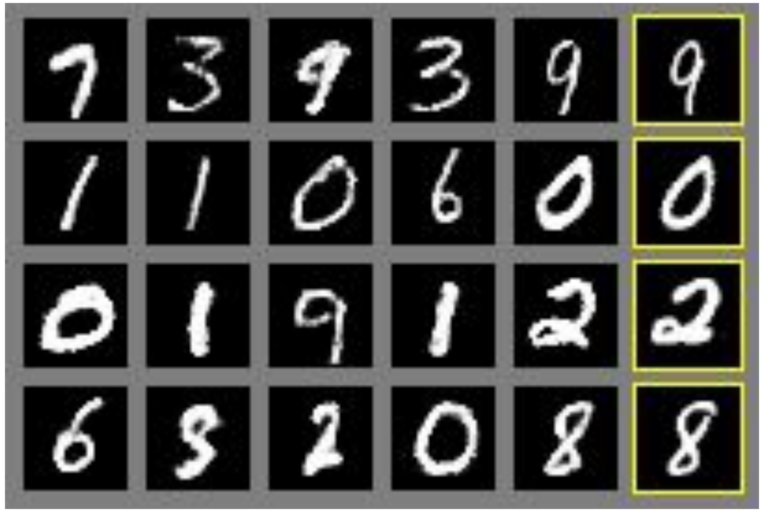
- Updating the generator:



Results

(Goodfellow et al., 2014)

- The generator uses a mixture of rectifier linear activations and/or sigmoid activations
- The discriminator net used maxout activations.



MNIST samples

TFD samples

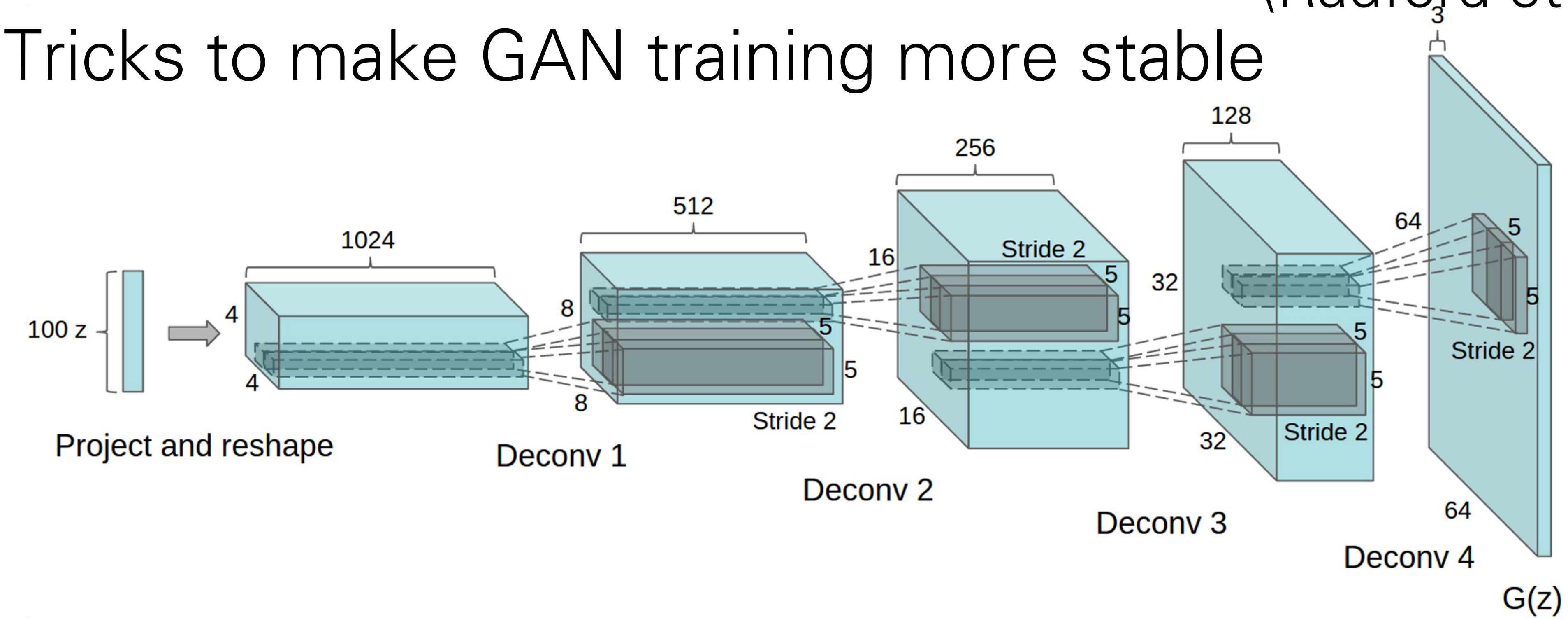
CIFAR10 samples
(fully-connected model)

CIFAR10 samples
(convolutional discriminator, deconvolutional generator)

Deep Convolutional GANs (DCGAN)

(Radford et al., 2015)

- **Idea:** Tricks to make GAN training more stable



- No fully connected layers
- Batch Normalization (Ioffe and Szegedy, 2015)
- Leaky Rectifier in D
- Use Adam (Kingma and Ba, 2015)
- Tweak Adam hyperparameters a bit (lr=0.0002, b1=0.5)

DCGAN for LSUN Bedrooms

64×64 pixels
~3M images

(Radford et al., 2015)

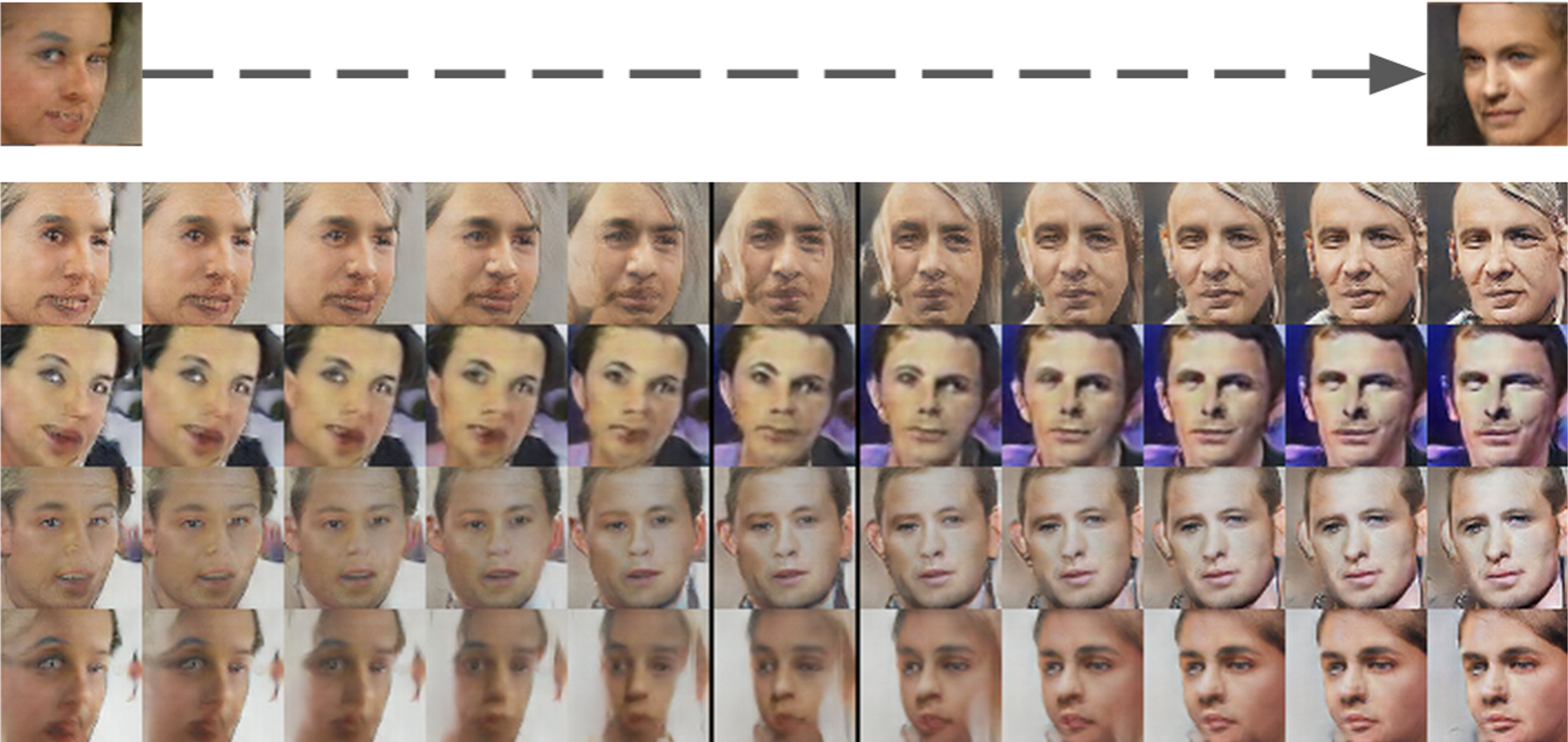
Walking over the latent space

(Radford et al., 2015)

- Interpolation suggests non-overfitting behavior

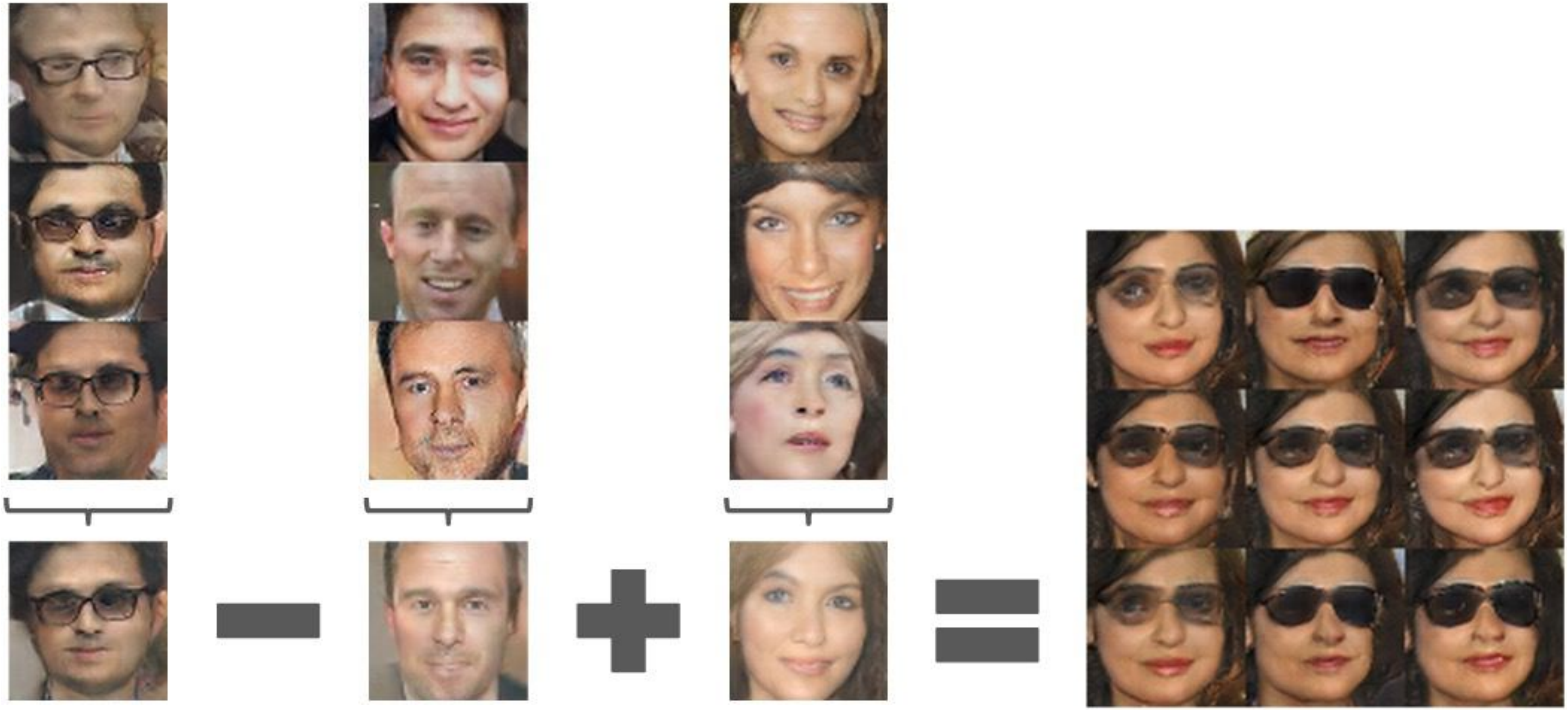
Walking over the latent space

(Radford et al., 2015)



Vector Space Arithmetic

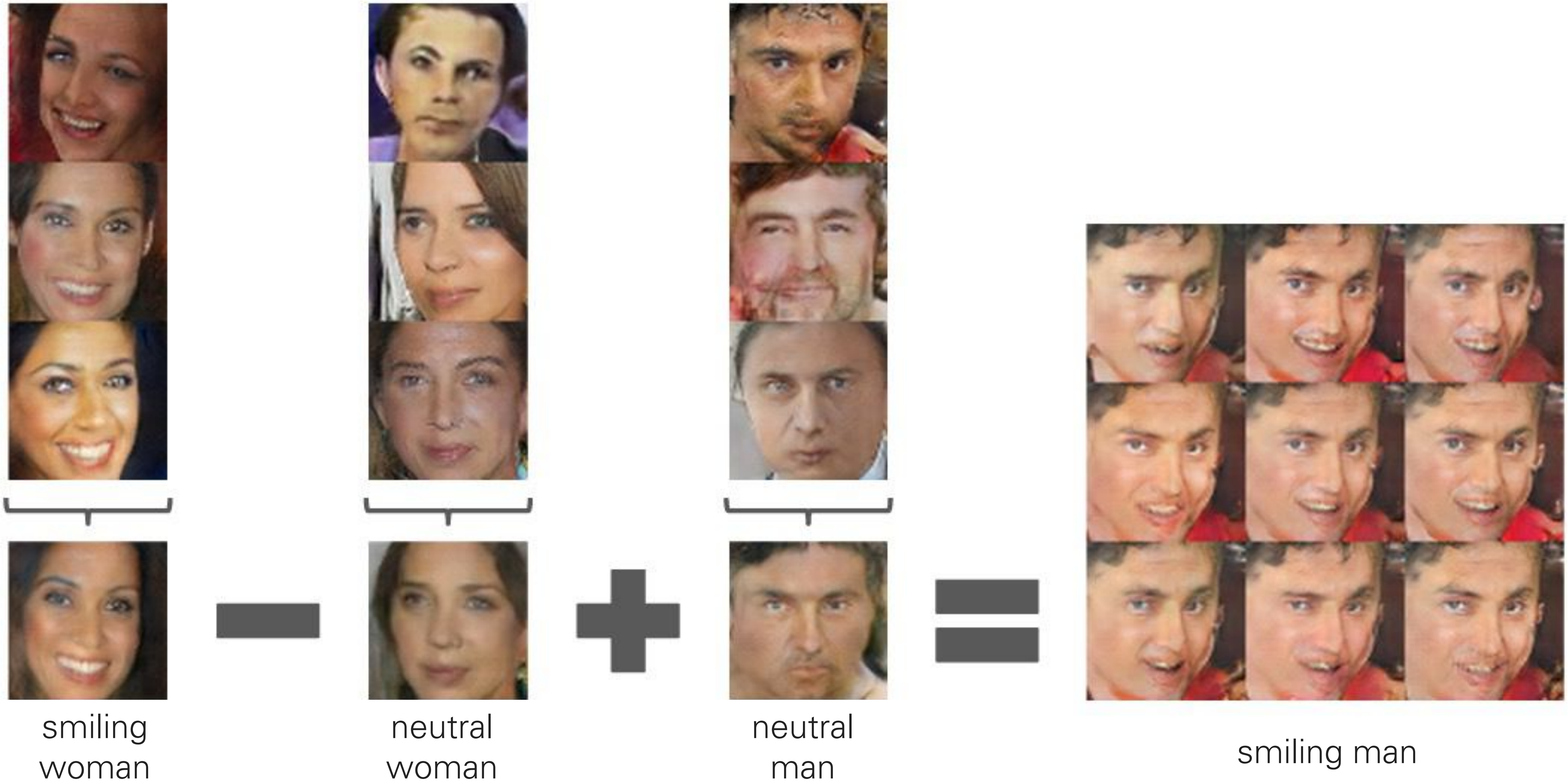
(Radford et al., 2015)



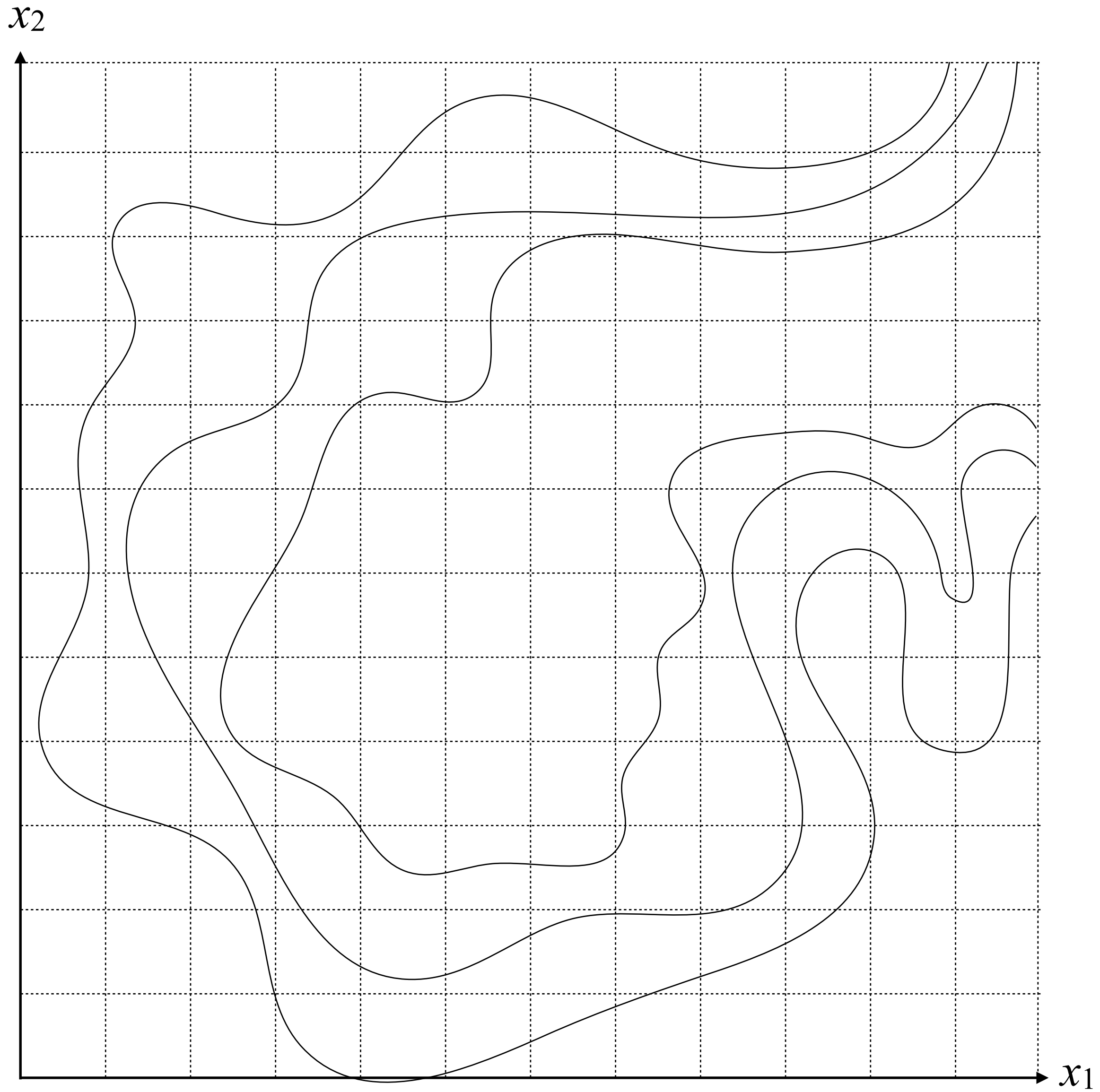
man with glasses − man without glasses + woman without glasses = woman with glasses

Vector Space Arithmetic

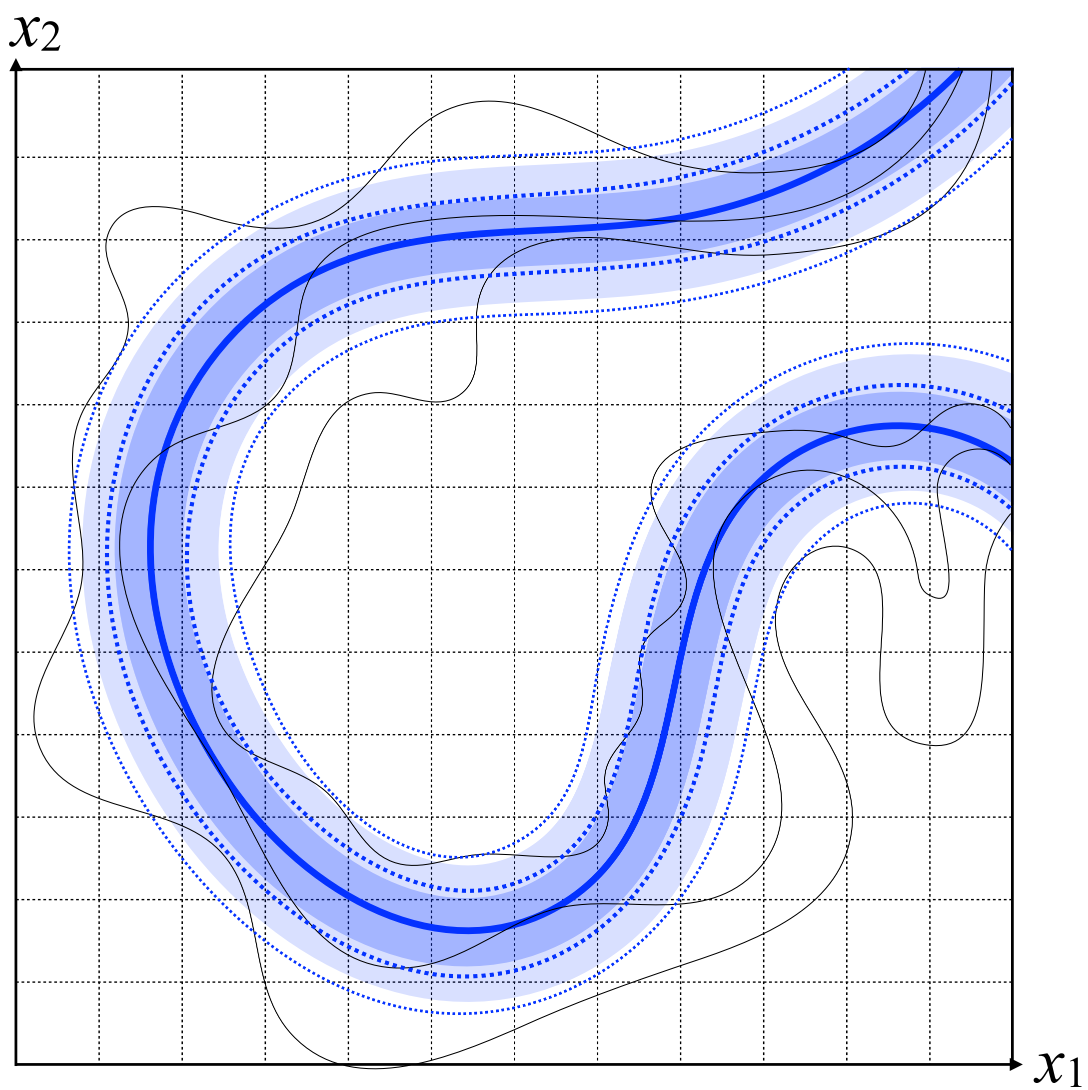
(Radford et al., 2015)



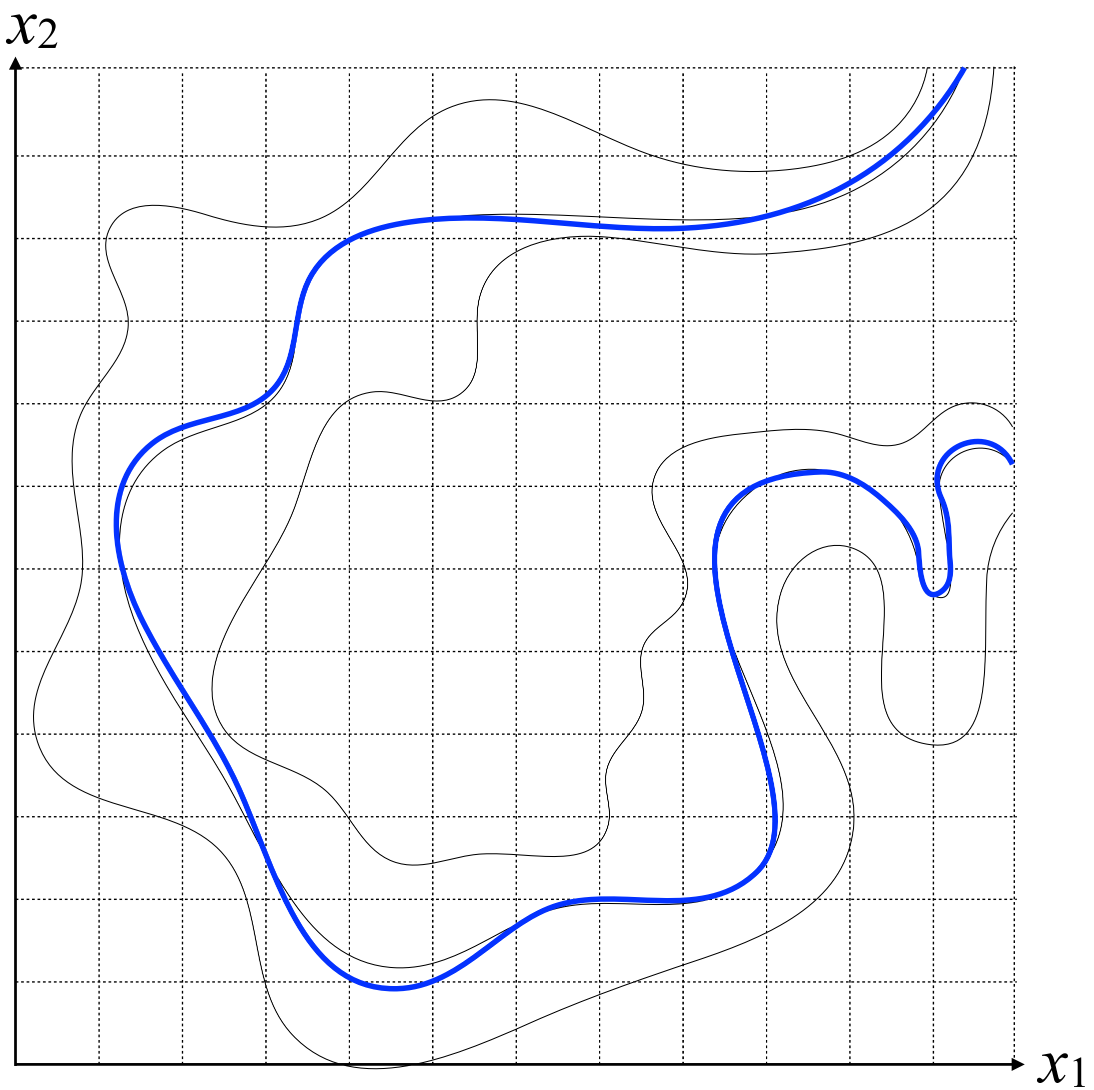
Cartoon of the Image manifold



What makes GANs special?



more traditional max-likelihood approach

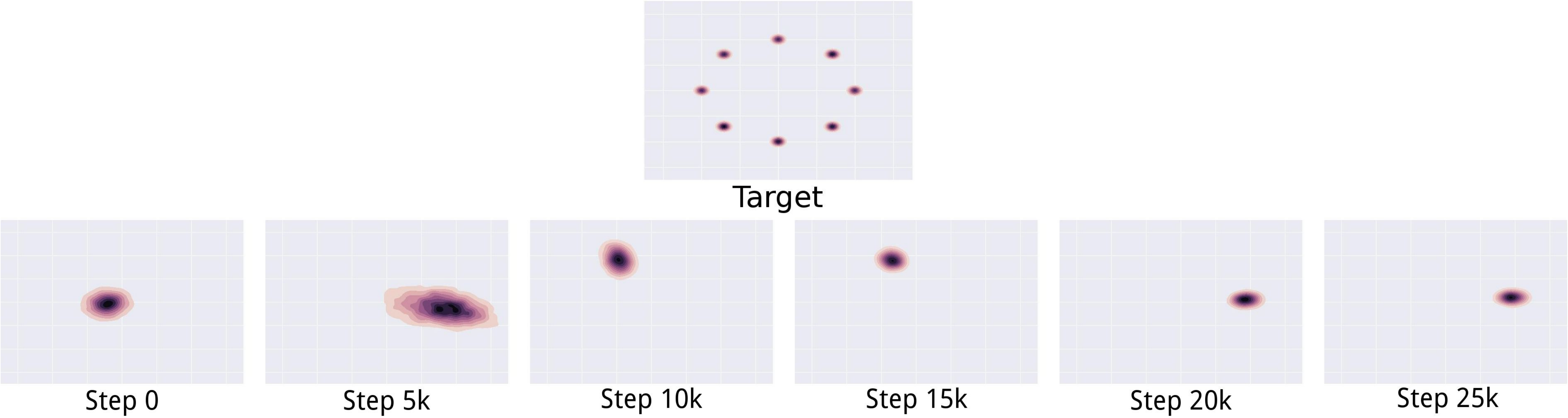


GAN

GAN Failures: Mode Collapse

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

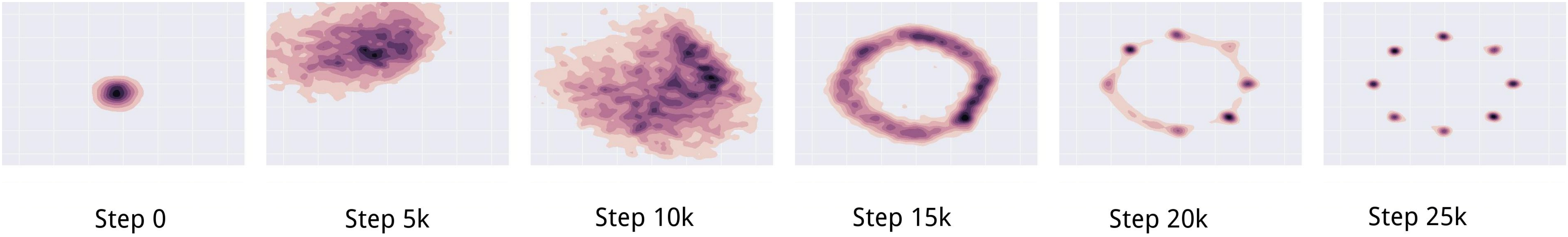
- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point



(Metz et al., 2016)

Mode Collapse: Solutions

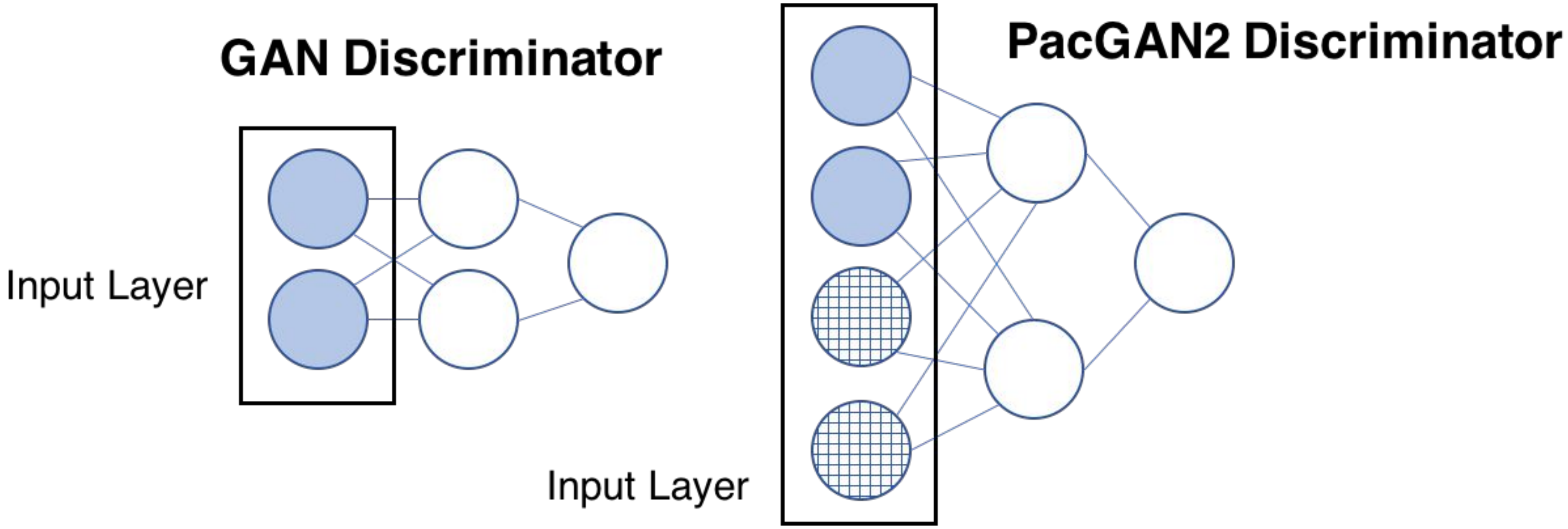
- **Unrolled GANs** (Metz et al 2016): Prevents mode collapse by backproping through a set of (k) updates of the discriminator to update generator parameters



- **VEEGAN** (Srivastava et al 2017): Introduce a reconstructor network which is learned both to map the true data distribution p(x) to a Gaussian and to approximately invert the generator network.

Mode Collapse: Solutions

- **Minibatch Discrimination** (Salimans et al 2016): Add minibatch features that classify each example by comparing it to other members of the minibatch (Salimans et al 2016)
- **PacGAN:** The power of two samples in generative adversarial networks (Lin et al 2017): Also uses multisample discrimination.



Mode Collapse: Solutions

- **PacGAN:** The power of two samples in generative adversarial networks (Lin et al 2017): Also uses multisample discrimination.

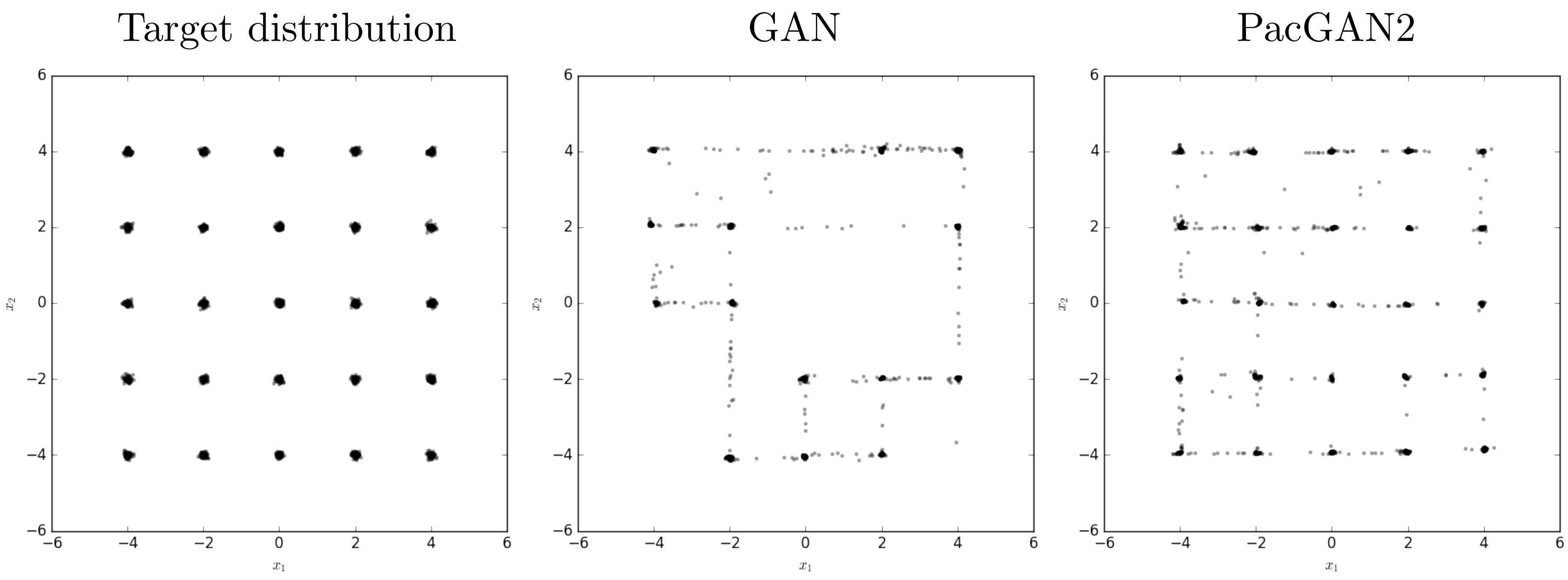


Figure 2: Scatter plot of the 2D samples from the true distribution (left) of 2D-grid and the learned generators using GAN (middle) and PacGAN2 (right). PacGAN2 captures all of the 25 modes.

GAN Evaluation

- Quantitatively evaluating GANs is not straightforward:
  - Max Likelihood is a poor indication of sample quality

- Some evaluation metrics
  - **Inception Score (IS)**:
    y = labels given gen. image. p(y|x) is from classifier - InceptionNet

$$\mathrm{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_g}[KL(p_{\mathcal{M}}(y|\mathbf{x})||p_{\mathcal{M}}(y))]}$$

  - **Fréchet inception distance (FID):** (Currently most popular)
    Estimate mean m and covariance C from classifier output - InceptionNet

$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\big(\boldsymbol{C} + \boldsymbol{C}_w - 2\big(\boldsymbol{C}\boldsymbol{C}_w\big)^{1/2}\big)$$

  - **Kernel MMD** (Maximum Mean Discrepancy):

$$\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left(\mathbb{E}_{\substack{\mathbf{x}_r, \mathbf{x}_r' \sim \mathbb{P}_r, \\ \mathbf{x}_g, \mathbf{x}_g' \sim \mathbb{P}_g}}\left[k(\mathbf{x}_r, \mathbf{x}_r') - 2k(\mathbf{x}_r, \mathbf{x}_g) + k(\mathbf{x}_g, \mathbf{x}_g')\right]\right)^{\frac{1}{2}}$$

Subclasses of GANs

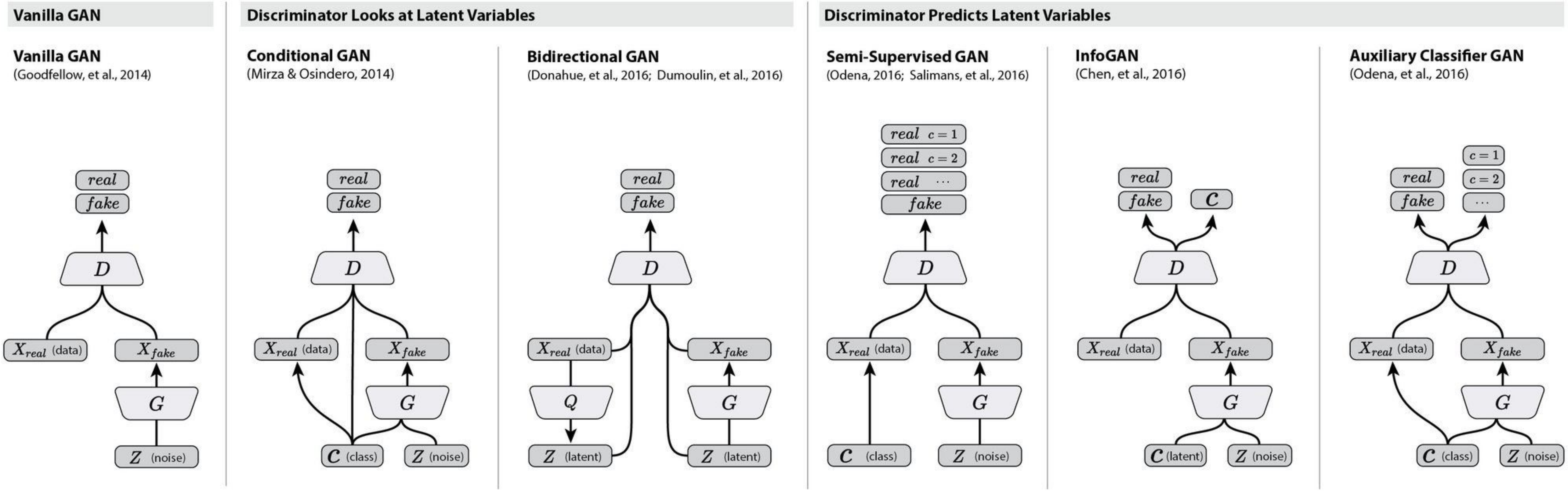
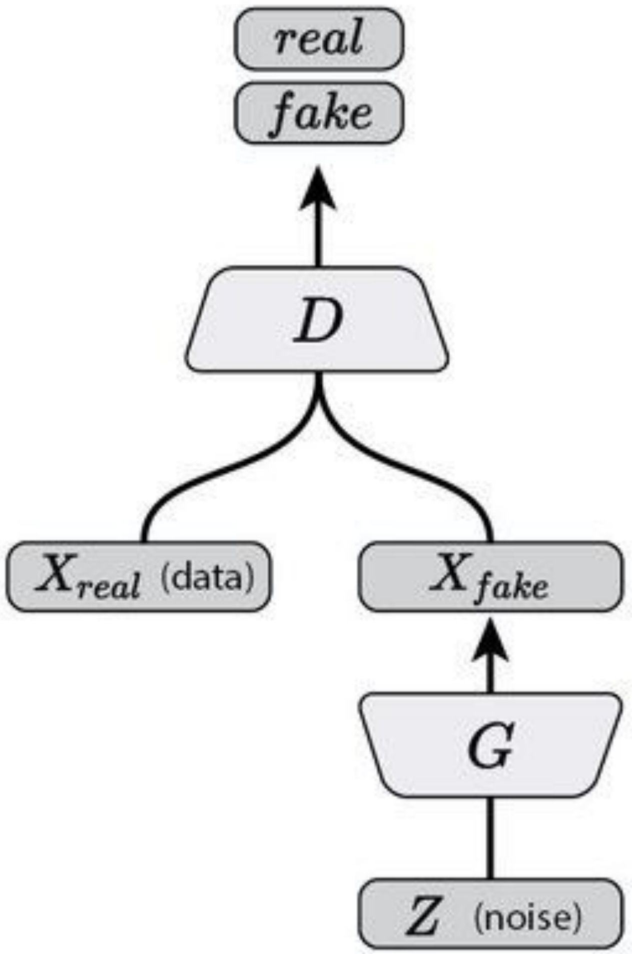
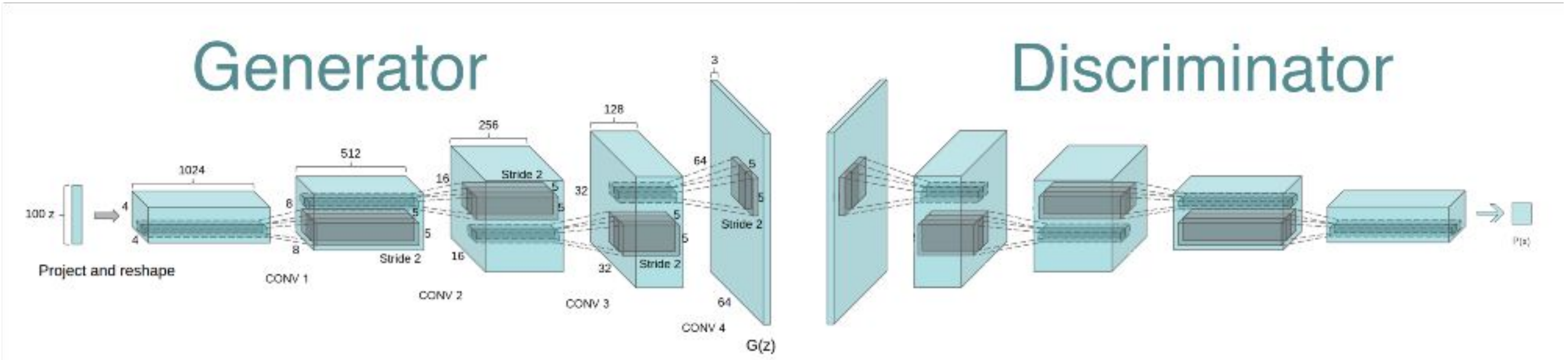


Image: Christopher Olah

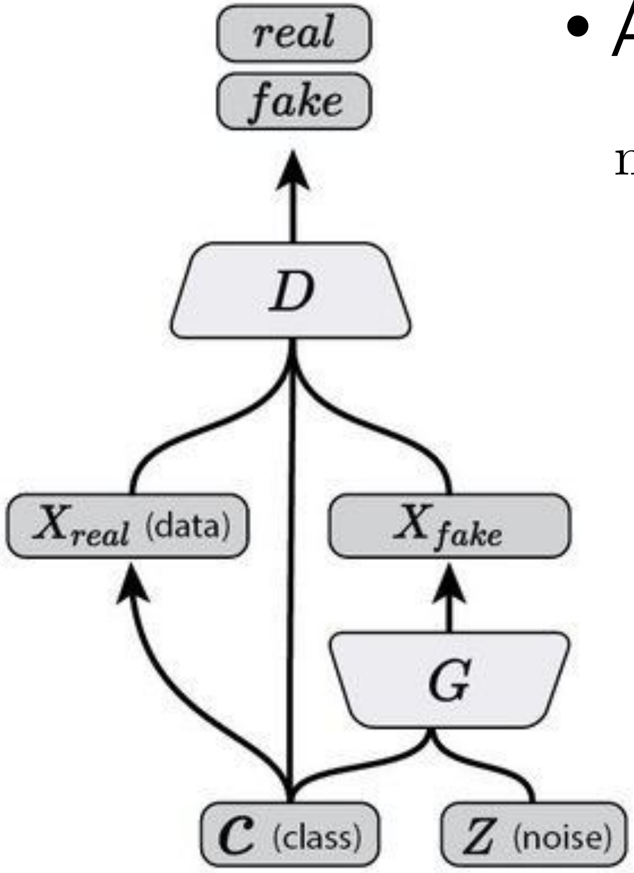
Vanilla GAN (Goodfellow et al., 2014)





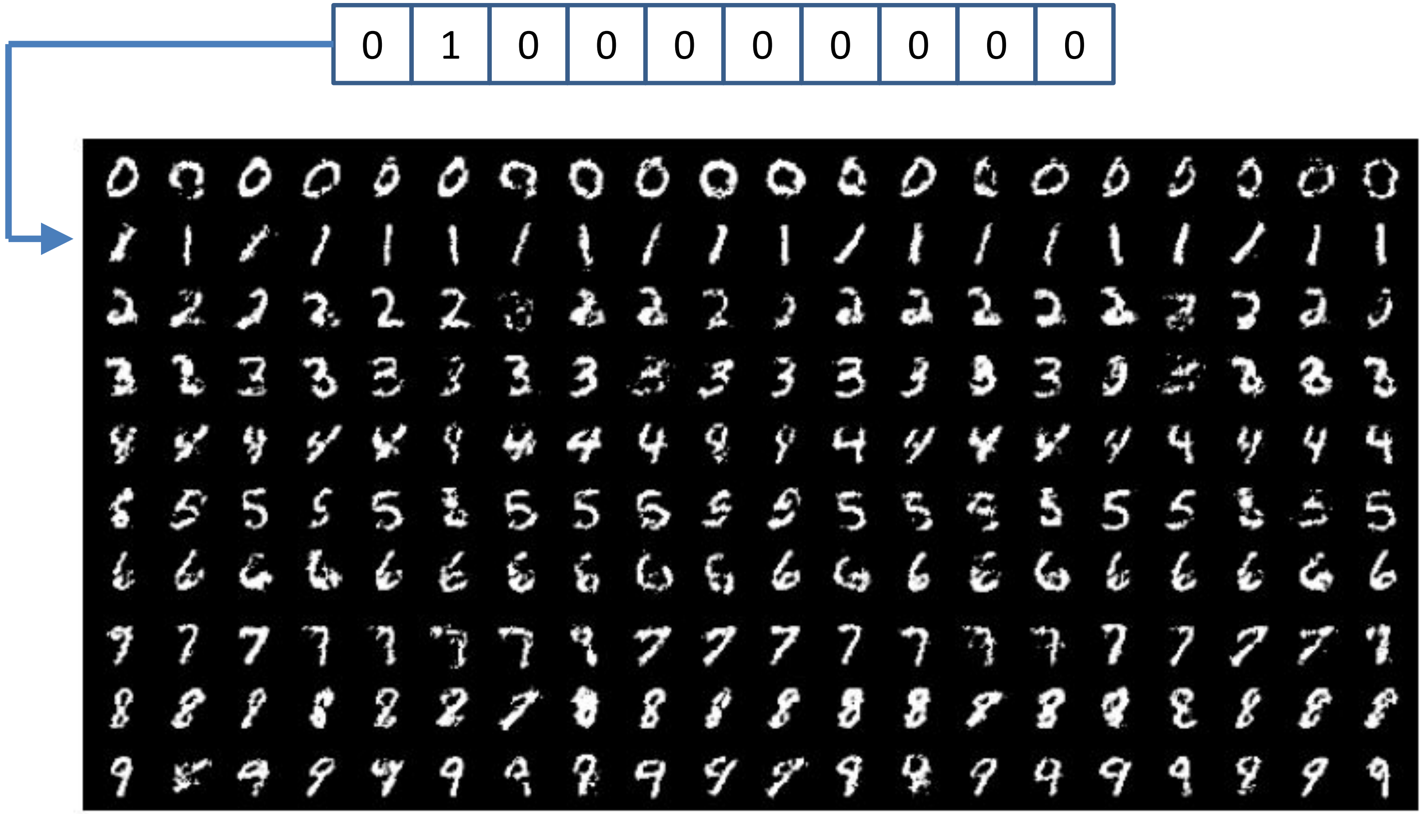
DCGAN (Radford et al., 2015)

Conditional GAN (Mirza and Osindero, 2014)

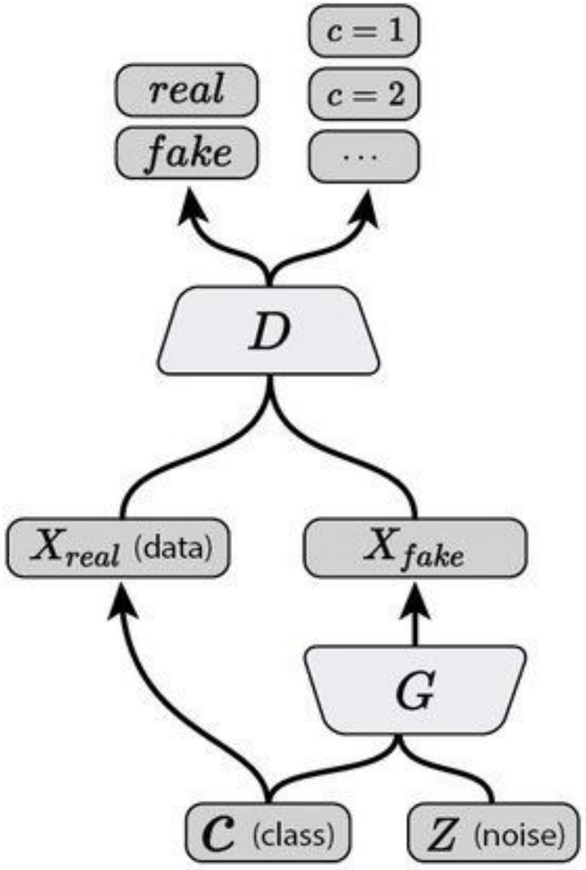


- Add conditional variables $\boldsymbol{y}$ into G and D

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



Auxiliary Classifier GAN (Odena et al., 2016)



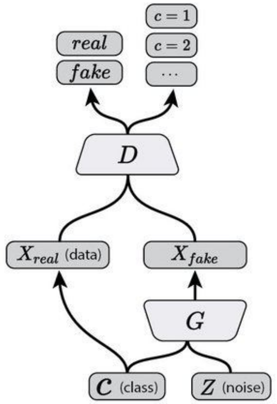
- Every generated sample has a corresponding class label

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$
$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]$$

- D is trained to maximize $L_S + L_C$
- G is trained to maximize $L_C - L_S$

- Learns a representation for z that is independent of class label

Auxiliary Classifier GAN (Odena et al., 2016)

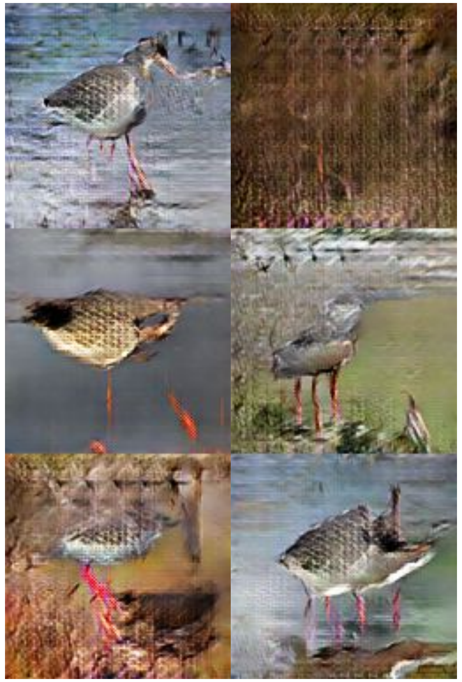


128×128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet

monarch butterfly

goldfinch

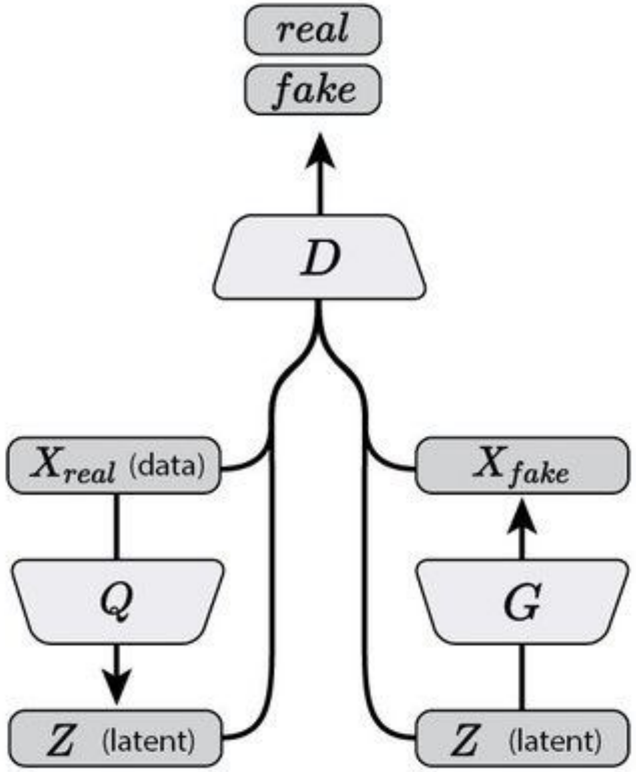
daisy



redshank

grey whale

Bidirectional GAN (Donahue et al., 2016; Dumoulin et al., 2016)



- Jointly learns a generator network and an inference network using an adversarial process.

$$\min_G \max_D V(D,G) = \mathbb{E}_{q(\boldsymbol{x})}[\log(D(\boldsymbol{x}, G_z(\boldsymbol{x})))] + \mathbb{E}_{p(\boldsymbol{z})}[\log(1 - D(G_x(\boldsymbol{z}), \boldsymbol{z}))]$$

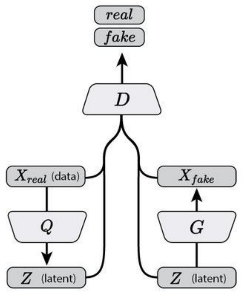
$$= \iint q(\boldsymbol{x})q(\boldsymbol{z} \mid \boldsymbol{x}) \log(D(\boldsymbol{x}, \boldsymbol{z})) d\boldsymbol{x} d\boldsymbol{z}$$

$$+ \iint p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z}) \log(1 - D(\boldsymbol{x}, \boldsymbol{z})) d\boldsymbol{x} d\boldsymbol{z}.$$

CelebA reconstructions

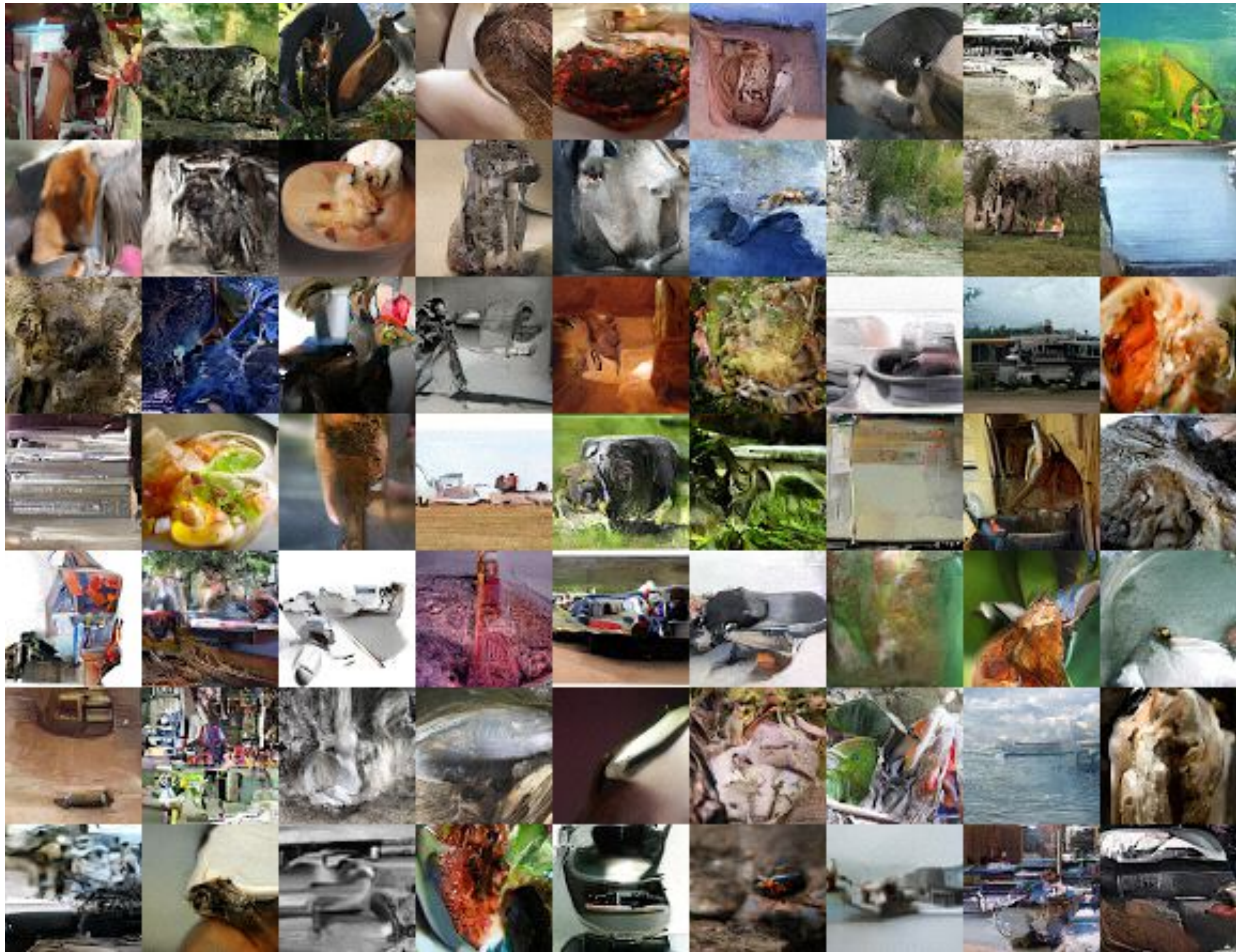
SVNH reconstructions

Bidirectional GAN (Donahue et al., 2016; Dumoulin et al., 2016)



LSUN bedrooms

Tiny ImageNet

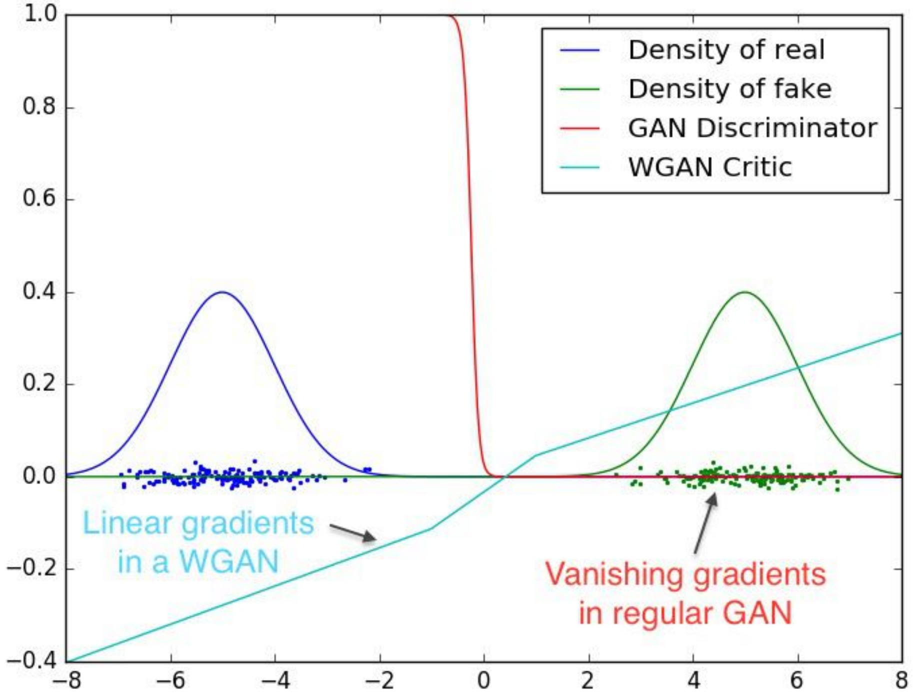


Wasserstein GAN (Arjovsky et al., 2016)

- Objective based on Earth-Mover or Wassertein distance:

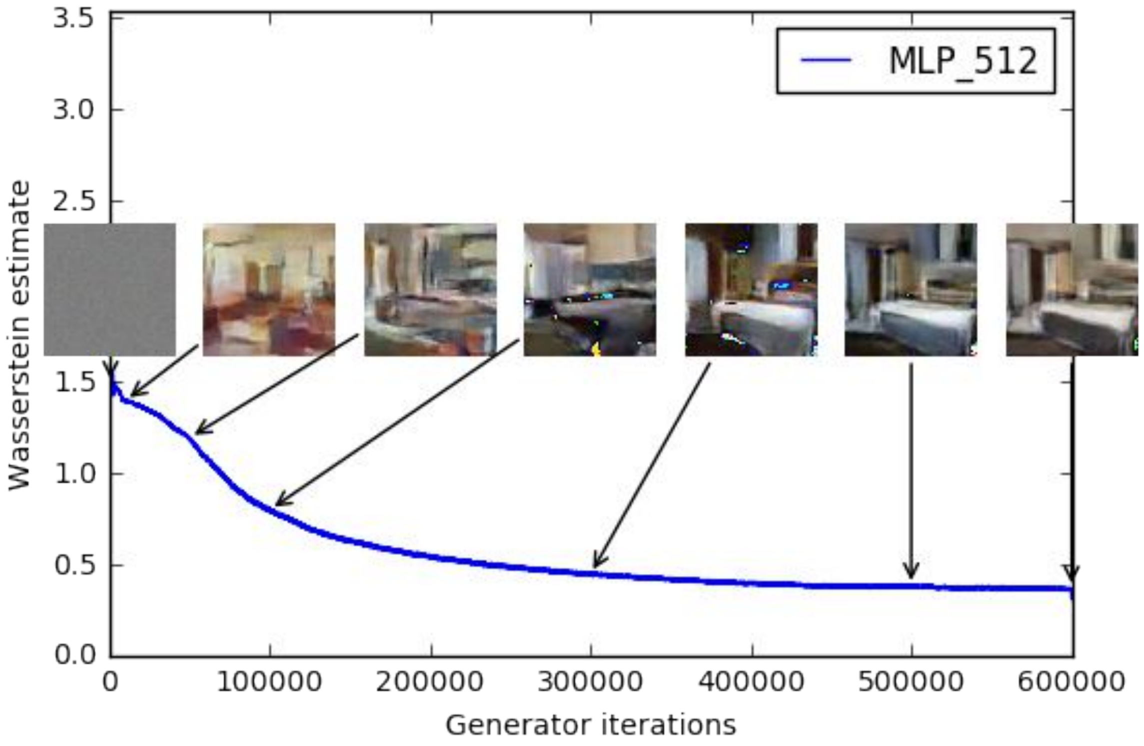
$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [D_\omega(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} [D_\omega(G_\theta(\boldsymbol{z}))]$$

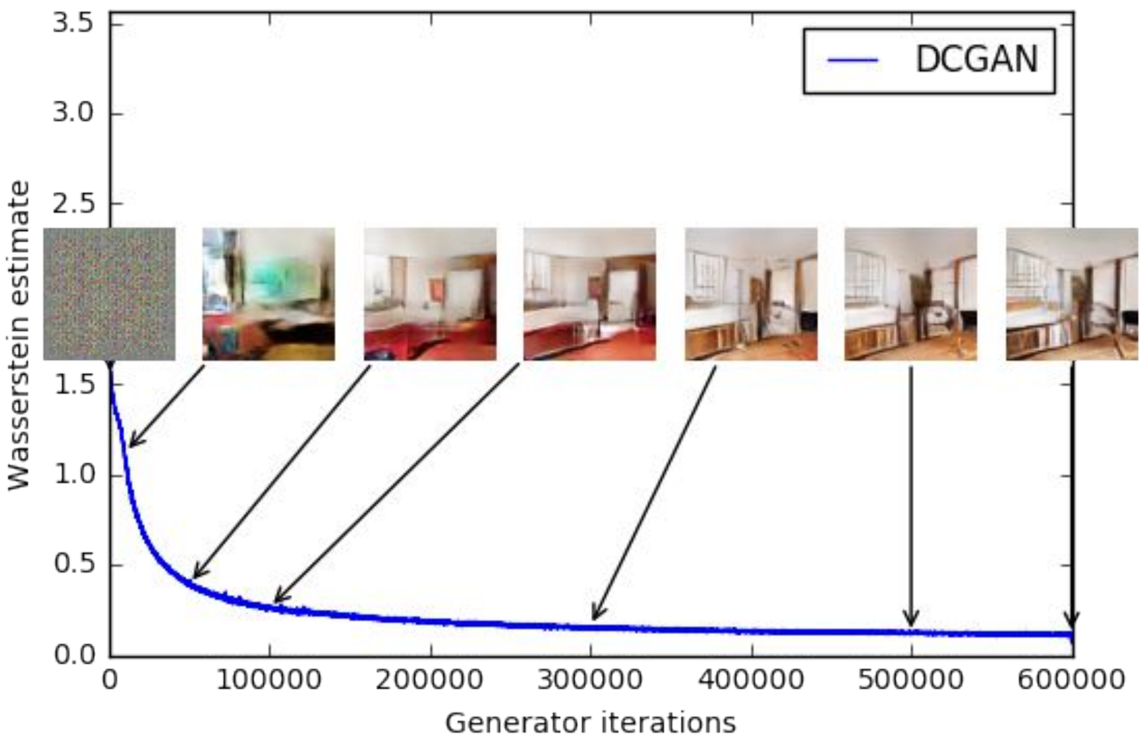
- Provides nice gradients over real and fake samples



Wasserstein GAN (Arjovsky et al., 2016)

- Wasserstein loss seems to correlate well with image quality.





WGAN with gradient penalty (Gulraani et al., 2017)

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} \left[D(\tilde{\boldsymbol{x}}) \right] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} \left[D(\boldsymbol{x}) \right]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\| \nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}) \|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}$$

- Faster convergence and higher-quality samples than WGAN with weight clipping
- Train a wide variety of GAN architectures with almost no hyperparameter tuning, including discrete models

WGAN with gradient penalty	
Busino game camperate spent odea	Solice Norkedin pring in since
In the bankaway of smarling the	ThiS record (31.) UBS) and Ch
SingersMay , who kill that imvic	It was not the annuas were plogr
Keray Pents of the same Reagun D	This will be us , the ect of DAN
Manging include a tudancs shat "	These leaded as most-worsd p2 a0
His Zuith Dudget , the Denmbern	The time I paid0a South Cubry i
In during the Uitational questio	Dour Fraps higs it was these del
Divos from The ' noth ronkies of	This year out howneed allowed lo
She like Monday , of macunsuer S	Kaulna Seto consficutes to repor
The investor used ty the present	A can teal , he was schoon news
A papees are cointry congress oo	In th 200. Pesish picriers rega
A few year inom the group that s	Konney Panice rimimber the teami
He said this syenn said they wan	The new centuct cut Denester of
As a world 1 88 ,for Autouries	The near , had been one injostie
Foand , th Word people car , Il	The incestion to week to shorted
High of the upseader homing pull	The company the high product of
The guipe is worly move dogsfor	20 - The time of accomplete , wh
The 1874 incidested he could be	John WVuderenson seqiivic spends
The allo tooks to security and c	A ceetens in indestredly the Wat
Standard GAN objective	
dddddddddddddddddddddddddddddddd	dddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddd	dddddddddddddddddddddddddddddddd

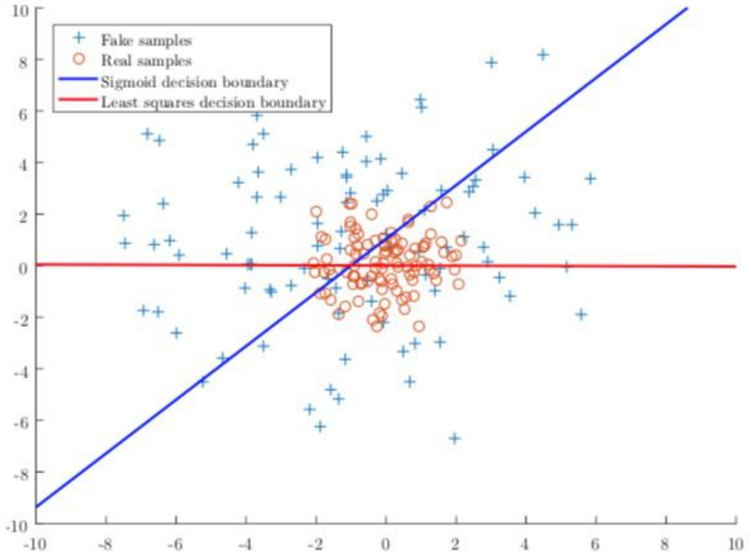
Samples from a character-level GAN language model on Google Billion Word

Least Squares GAN (LSGAN) (Mao et al., 2017)

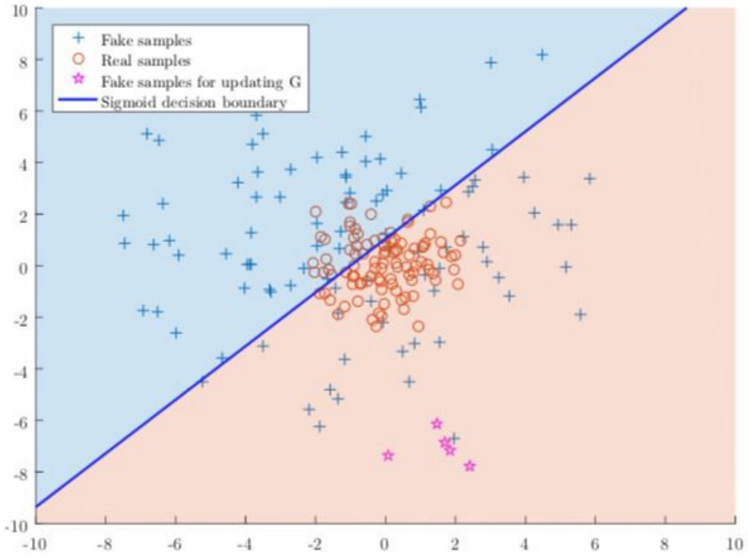
- Use a loss function that provides smooth and non-saturating gradient in discriminator D

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

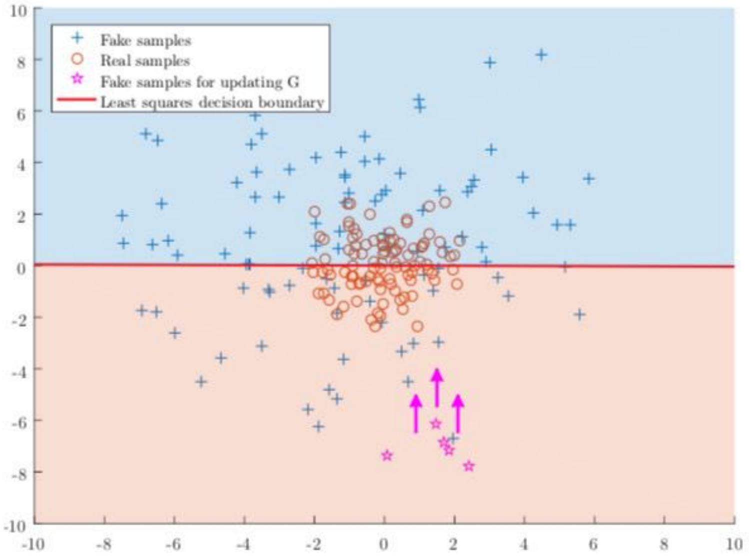
$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$



Decision boundaries of Sigmoid & Least Squares loss functions



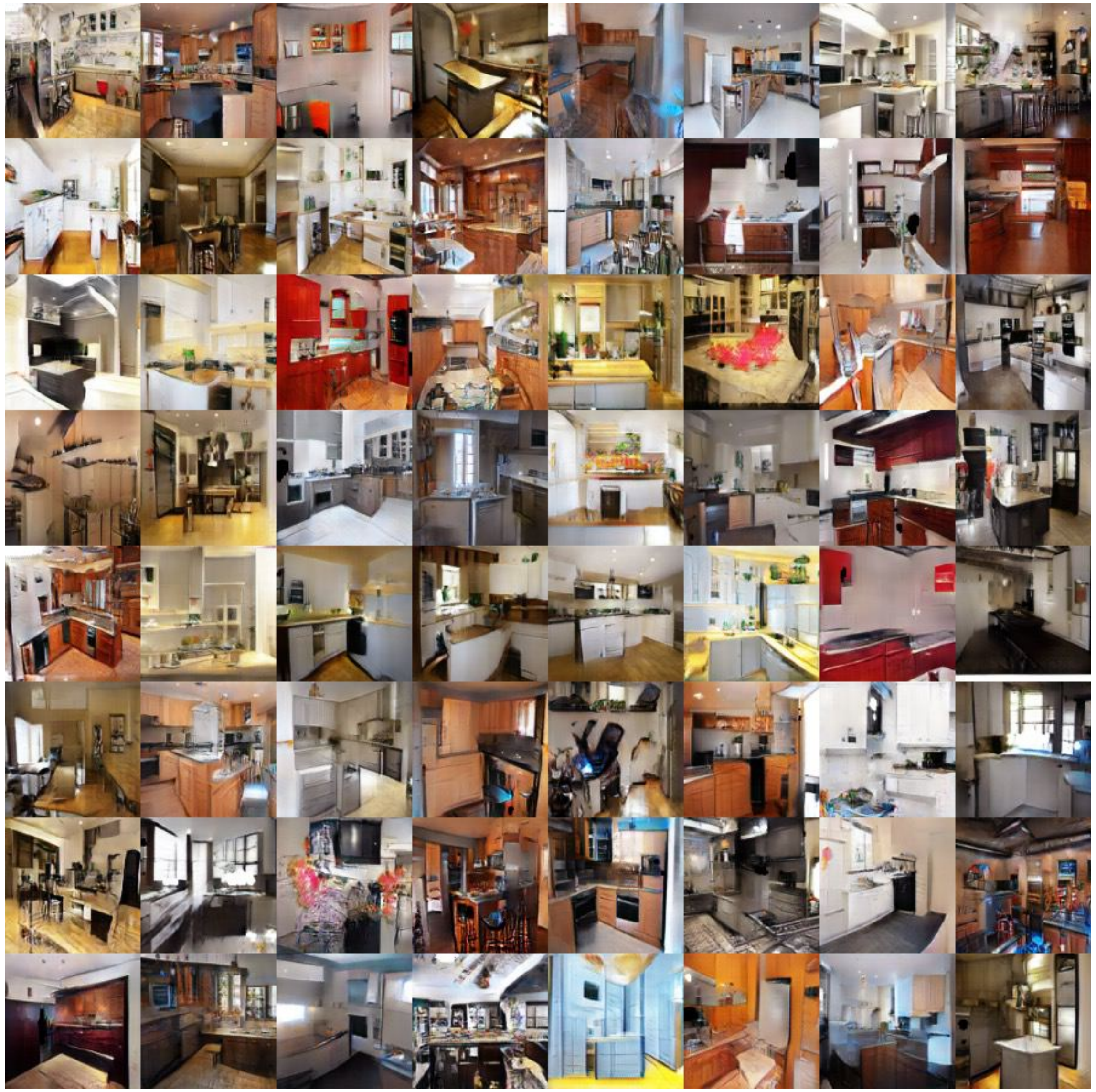
Sigmoid decision boundary



Least Squares decision boundary

Least Squares GAN (LSGAN) (Mao et al., 2017)

Church



Kitchen

Boundary Equilibrium GAN (BEGAN)

(Berthelot et al., 2017)

- A loss derived from the Wasserstein distance for training auto-encoder based GANs

$$\mathcal{L}(v) = |v - D(v)|^{\eta} \text{ where } \begin{cases} D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} & \text{is the autoencoder function.} \\ \eta \in \{1, 2\} & \text{is the target norm.} \\ v \in \mathbb{R}^{N_x} & \text{is a sample of dimension } N_x. \end{cases}$$

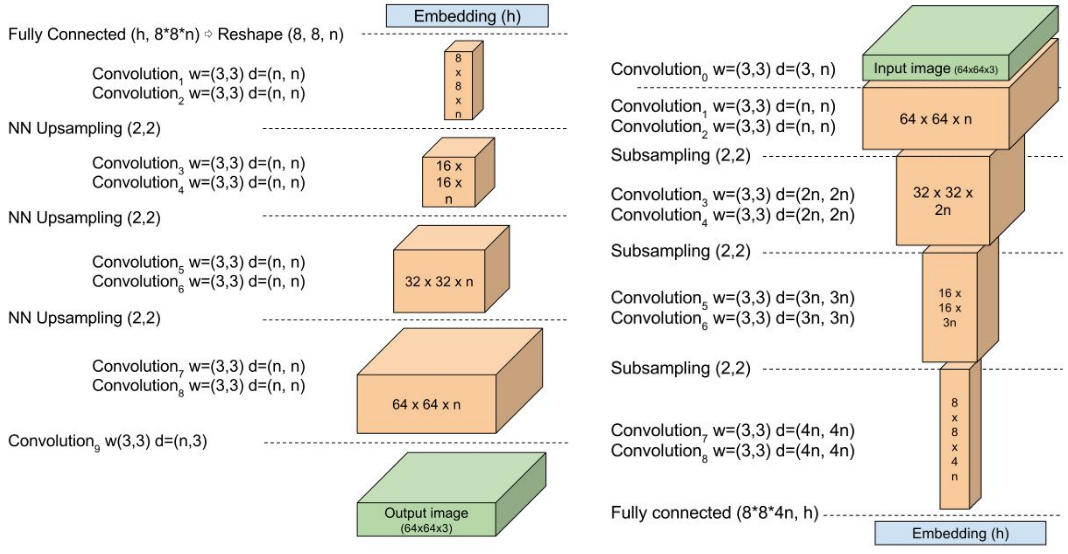
  - Wasserstein distance btw. the reconstruction losses of real and generated data

- **Convergence measure:**

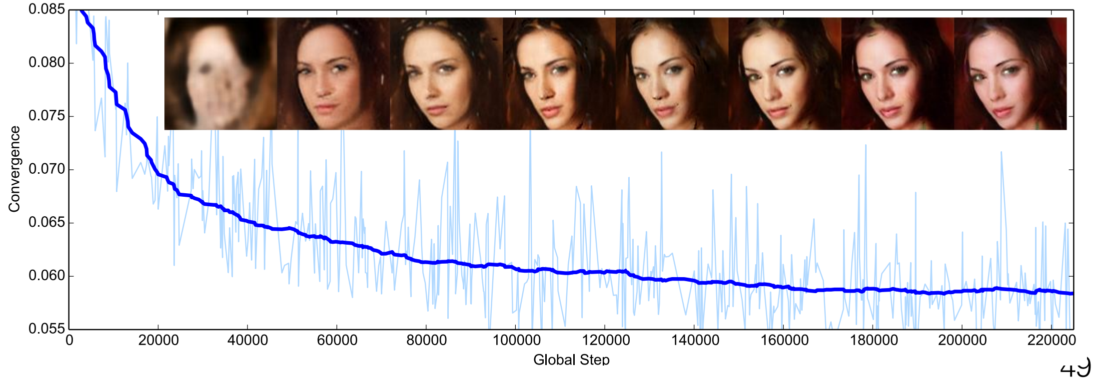
$$\mathcal{M}_{global} = \mathcal{L}(x) + |\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))|$$

- **Objective:**

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t.\mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases}$$



(a) Generator/Decoder (b) Encoder



BEGANs for CelebA

360K celebrity face images
128x128 with 128 filters

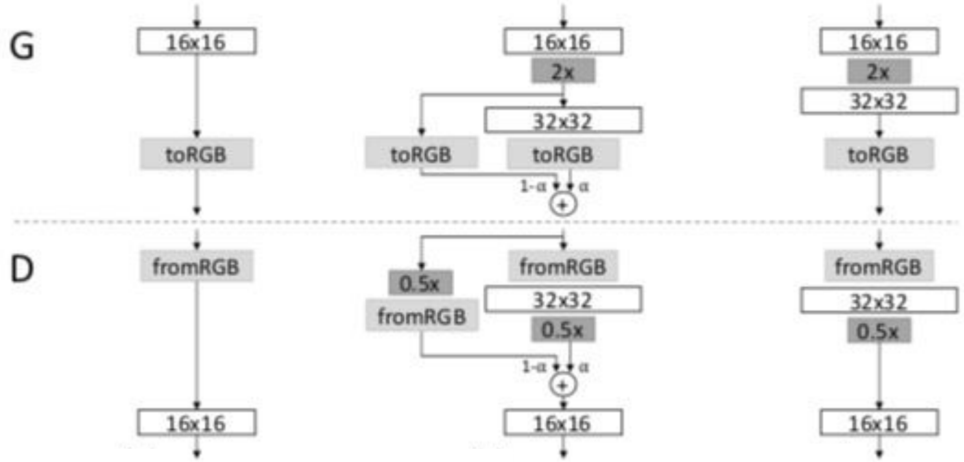
(Berthelot et al., 2017)

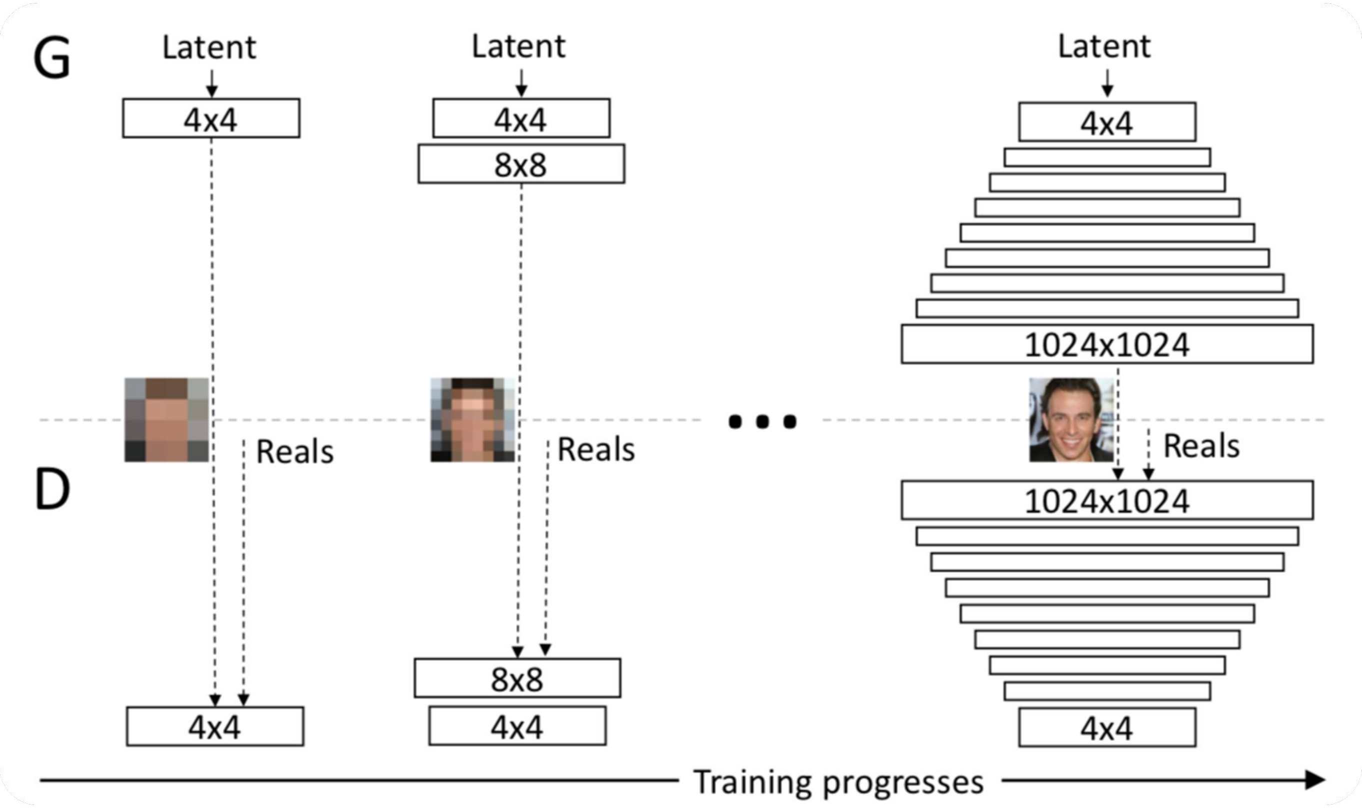
Interpolations in the latent space

Mirror interpolation example

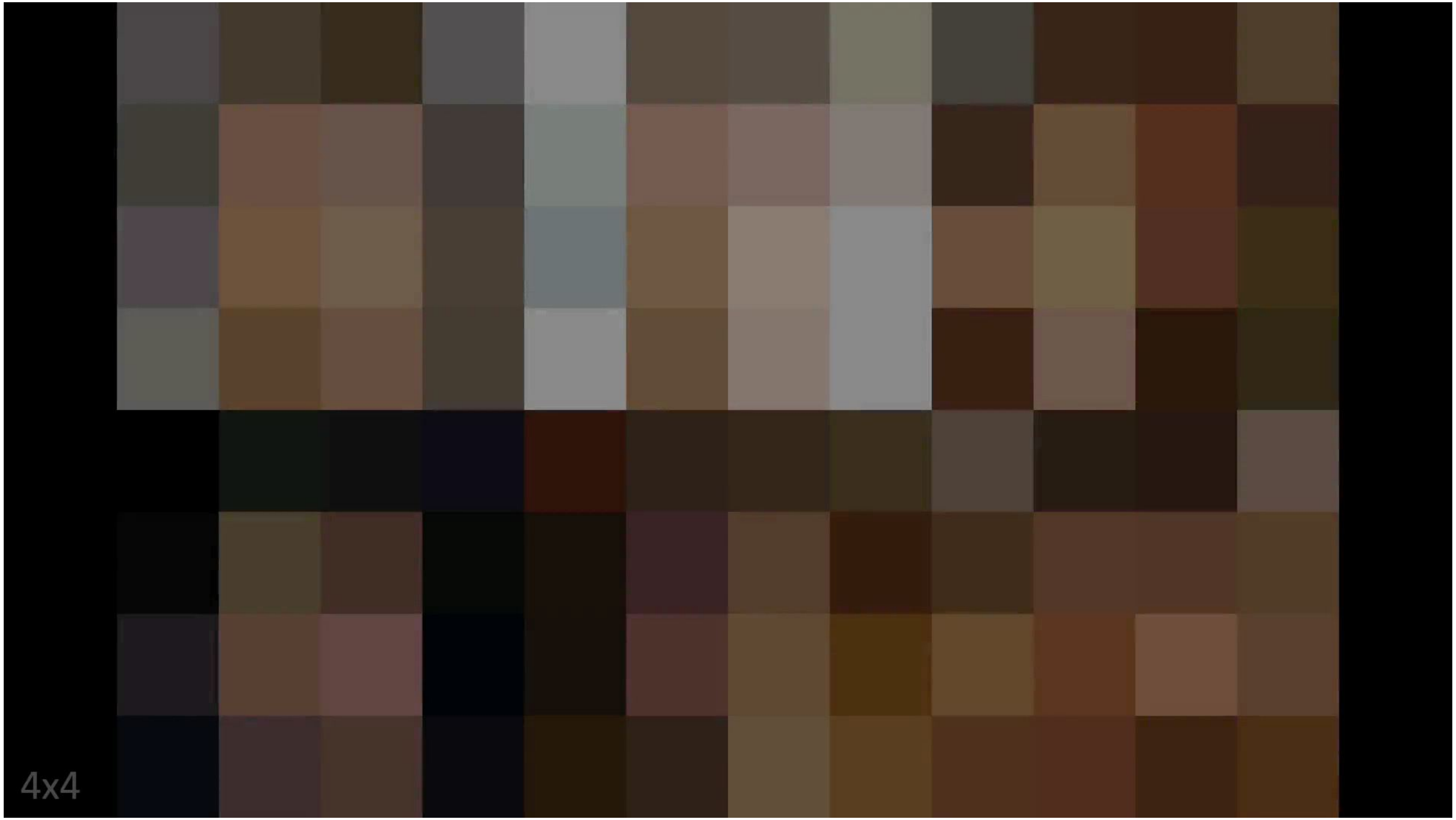
Progressive GANs (Karras et al., 2018)

- Progressively generate high-res images
- Multi-step training from low to high resolutions





Progressive GANs (Karras et al., 2018)



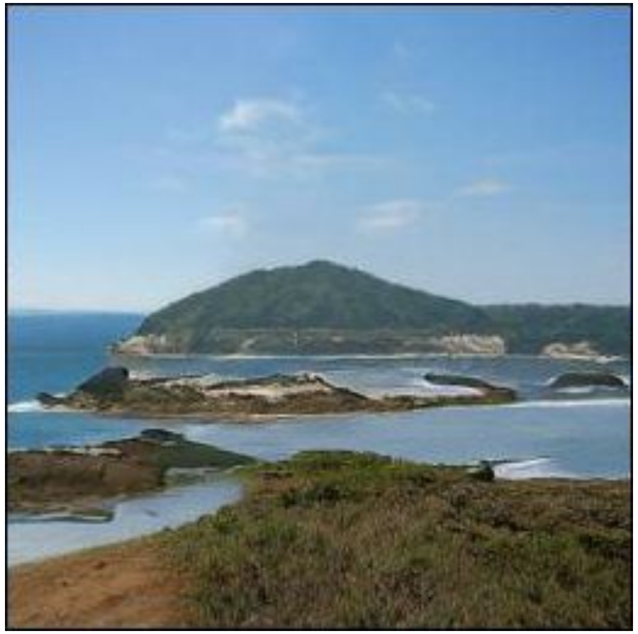
- Training process

Progressive GANs (Karras et al., 2018)

CelebA-HQ
random interpolations

BigGANs (Brock et al., 2019)

High resolution, class-conditional samples generated by the model

- BigGANs trained with 2-4x as many parameters and 8x the batch size compared to prior art.
- Uses Gaussian truncation to sample z (avoid sampling from the tail of the Gaussian distribution)
- Uses multiple other tricks including multiple regularizations including a Gradient penalty regularization and an Orthogonal Regularization:

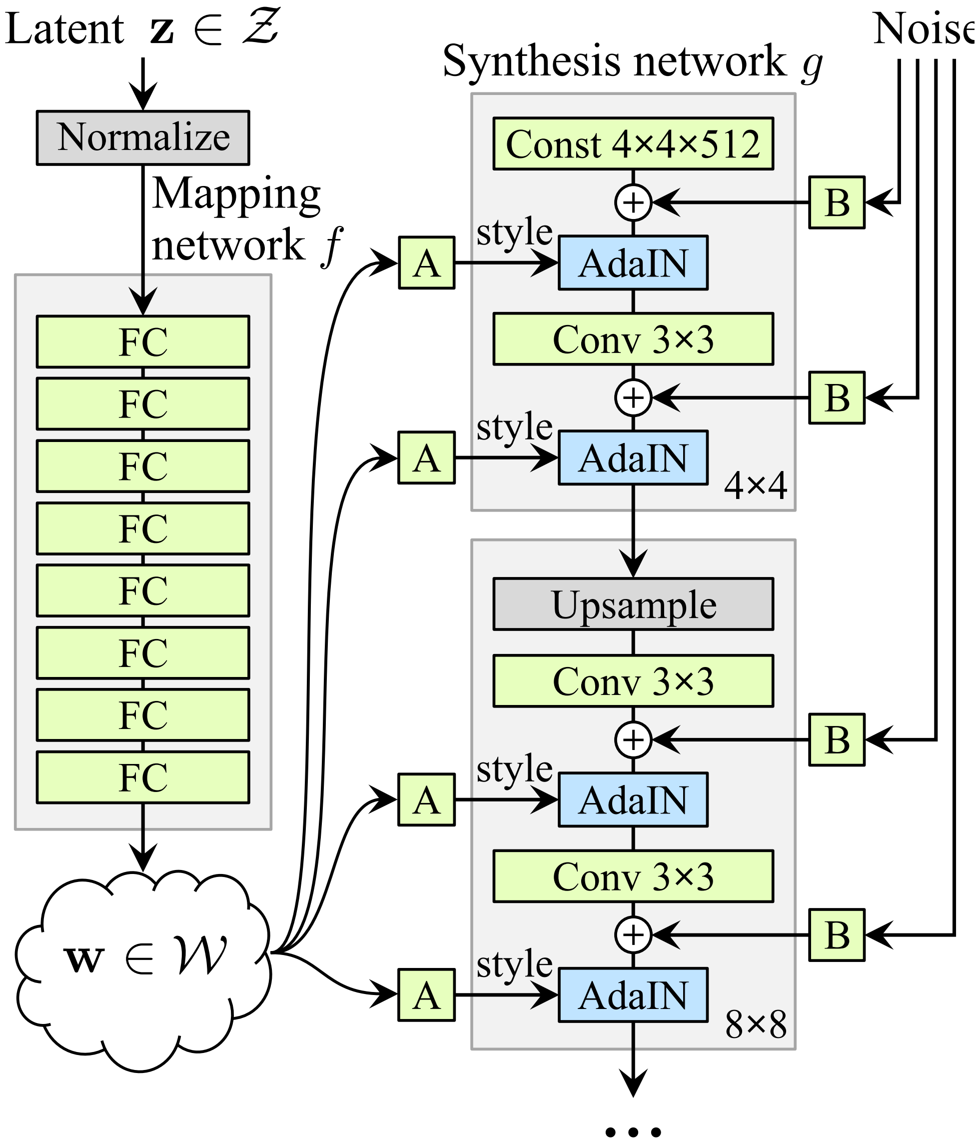
$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_\mathrm{F}^2,$$

BigGANs (Brock et al., 2019)

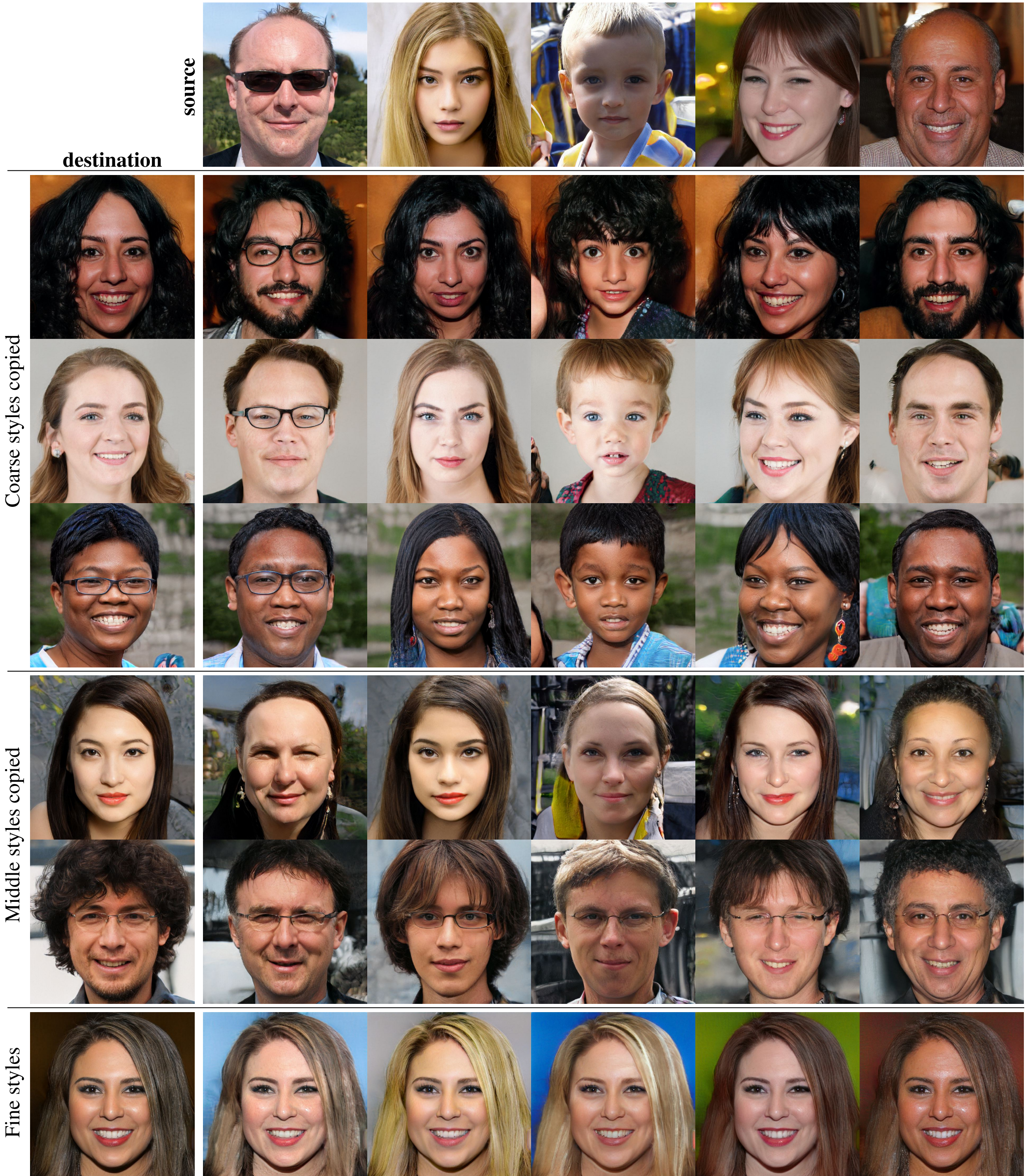
Resolution: 512x512

StyleGANs (Karras et al., 2019)

- A new architecture motivated by the style transfer networks
- allows unsupervised separation of high-level attributes and stochastic variation in the generated images



StyleGANs (Karras et al., 2018)



Some Applications of GANs

Semi-supervised Classification

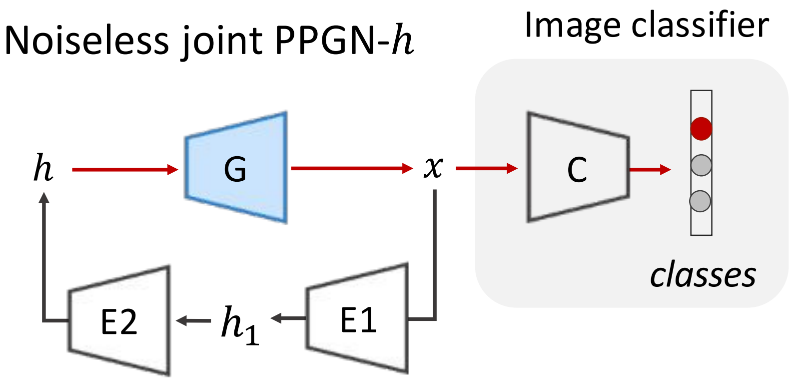
(Salimans et al., 2016; Dumoulin et al., 2016)

SVNH

Model	Misclassification rate
VAE (M1 + M2) (Kingma et al., 2014)	36.02
SWWAE with dropout (Zhao et al., 2015)	23.56
DCGAN + L2-SVM (Radford et al., 2015)	22.18
SDGM (Maaløe et al., 2016)	16.61
GAN (feature matching) (Salimans et al., 2016)	$\mathbf{8.11 \pm 1.3}$
ALI (ours, L2-SVM)	19.14 ± 0.50
ALI (ours, no feature matching)	$\mathbf{7.42 \pm 0.65}$

Class-specific Image Generation (Nguyen et al., 2016)

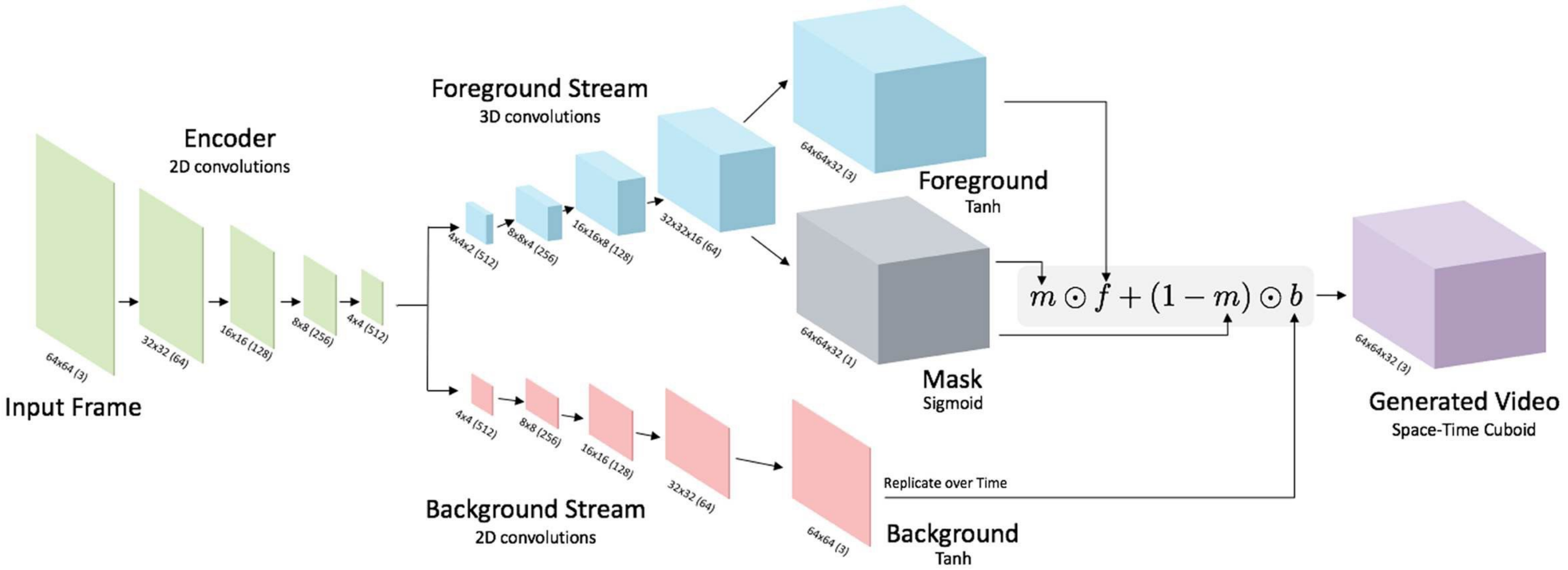
- Generates 227x227 realistic images from all ImageNet classes
- Combines adversarial training, moment matching, denoising autoencoders, and Langevin sampling



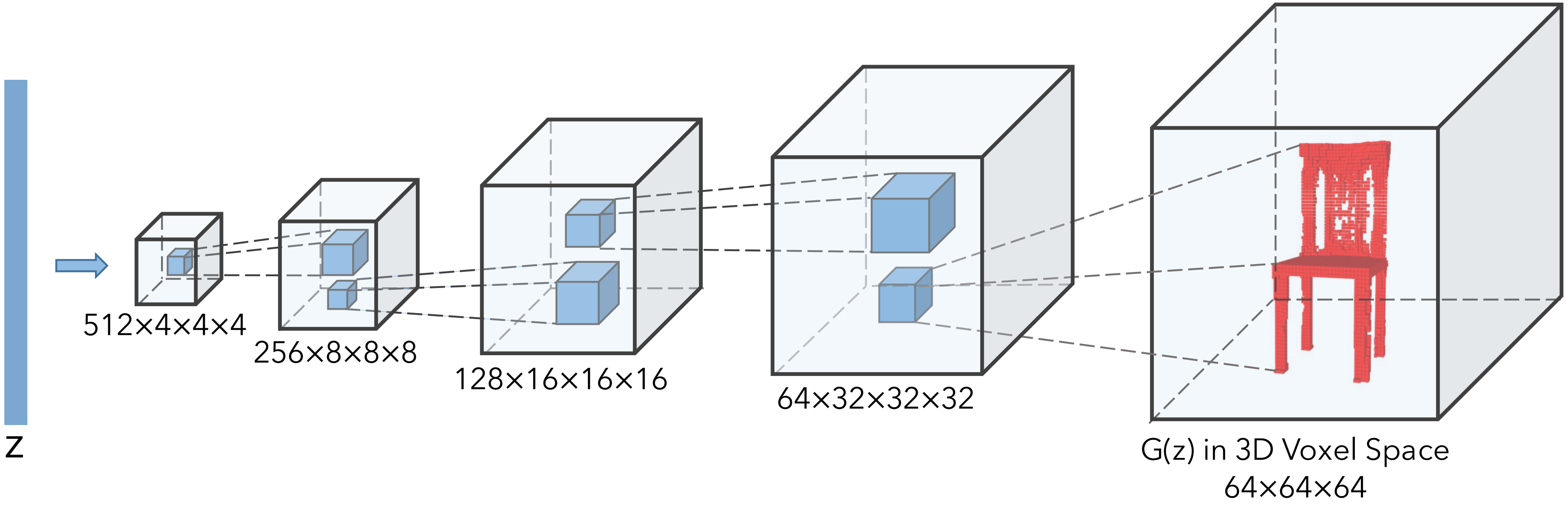
redshank ant monastery

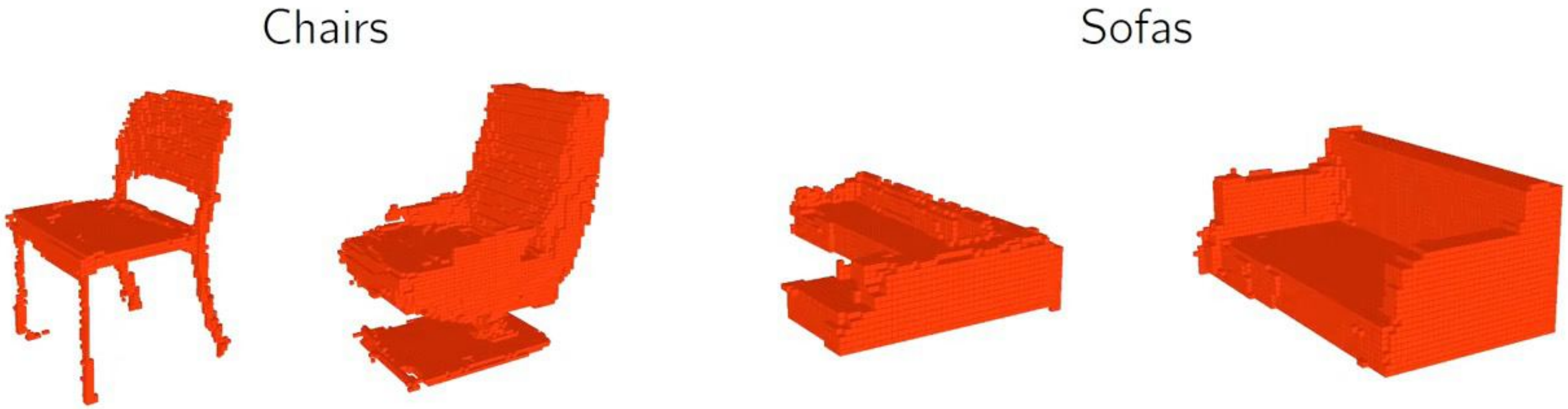
volcano

Video Generation (Vondrick et al., 2016)



Generative Shape Modeling (Wu et al., 2016)





Text-to-Image Synthesis (Zhang et al., 2016)

The small bird has a red head with feathers that fade from red to gray from head to tail

The petals of this flower are white with a large stigma

A unique yellow flower with no visible pistils protruding from the center

This flower is pink and yellow in color, with petals that are oddly shaped

This is a light colored flower with many different petals on a green stem

This flower is yellow and green in color, with petals that are ruffled

The flower have large petals that are pink with yellow on some of the petals

A flower that has white petals with some tones of yellow and green filaments

Text-to-Image Synthesis (Zhu et al., 2019)

This bird has a white throat and a dark yellow bill and grey wings.

This particular bird has a belly that is yellow and brown.

This bird is a lime green with greyish wings and long legs.

This yellow bird has a thin beak and jet black eyes and thin feet.

This bird has wings that are grey and has a white belly.

This bird has wings that are black and has a white belly.

This is a grey bird with a brown wing and a small orange beak.

This bird has a short brown bill, a white eyering, and a medium brown crown.

Single Image Super-Resolution (Ledig et al., 2016)

- Combine content loss with adversarial loss

Image Inpainting (Pathak et al., 2016)

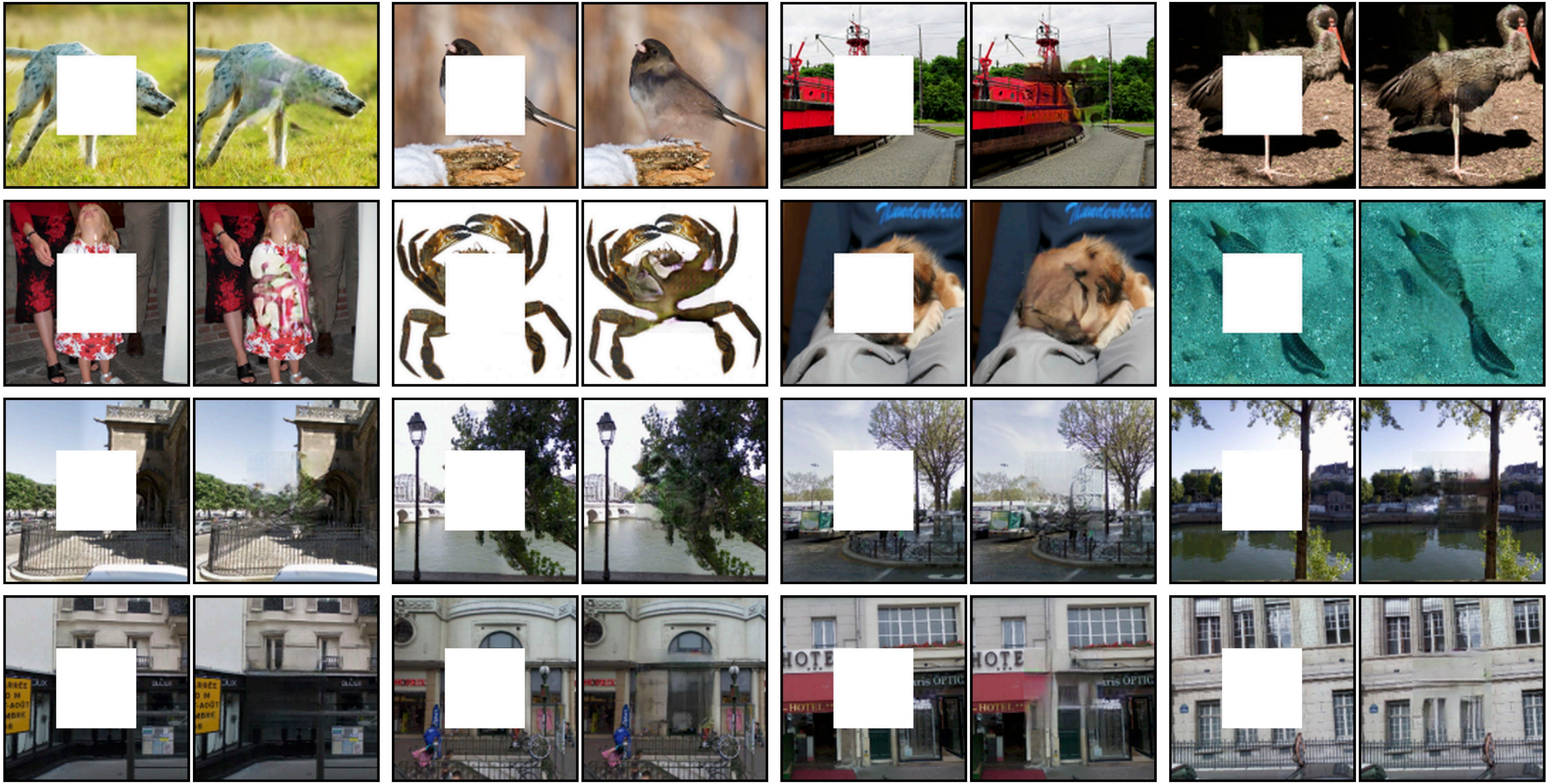
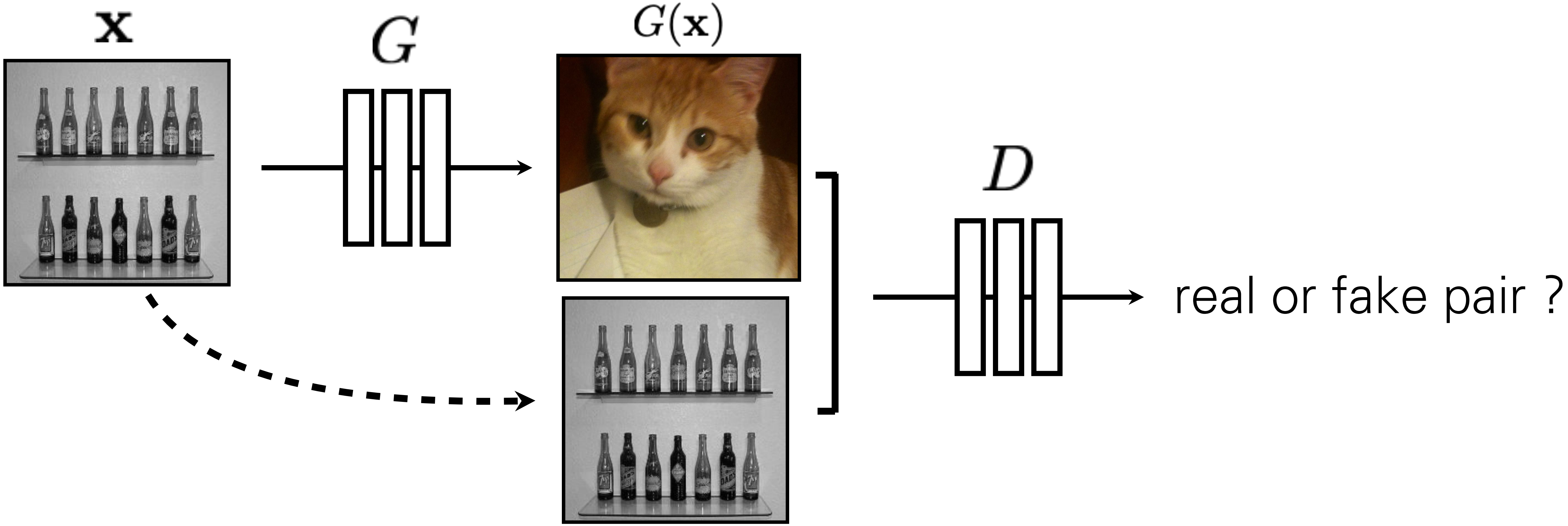
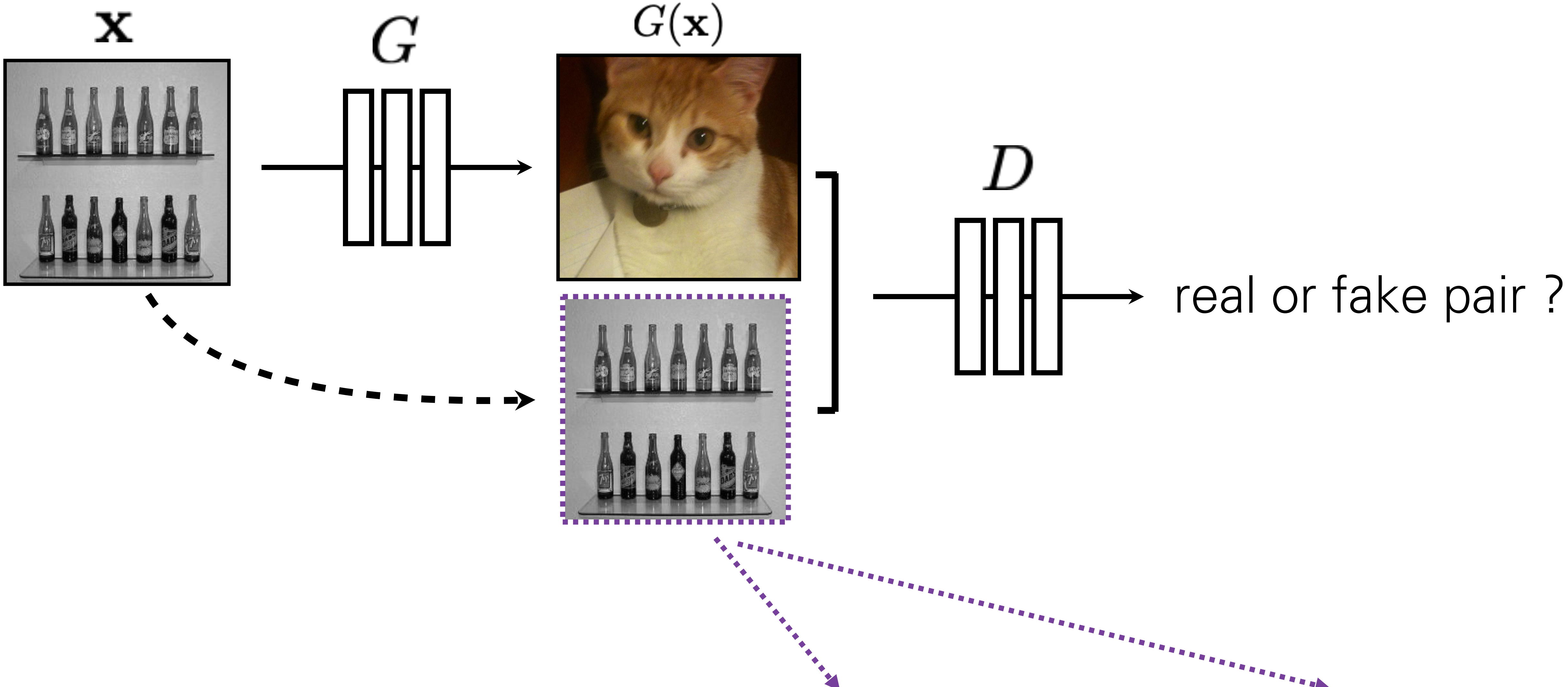


Image to Image Translation (Pix2Pix)

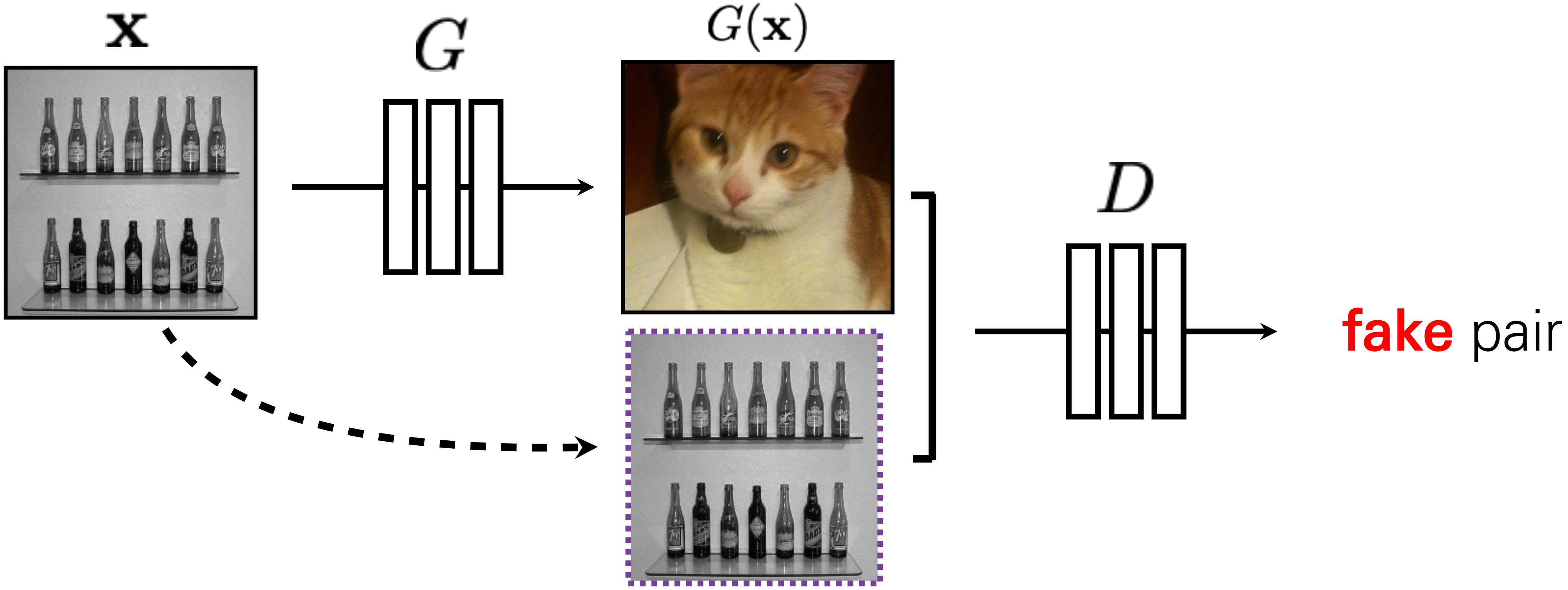
(Isola et al. 2016)



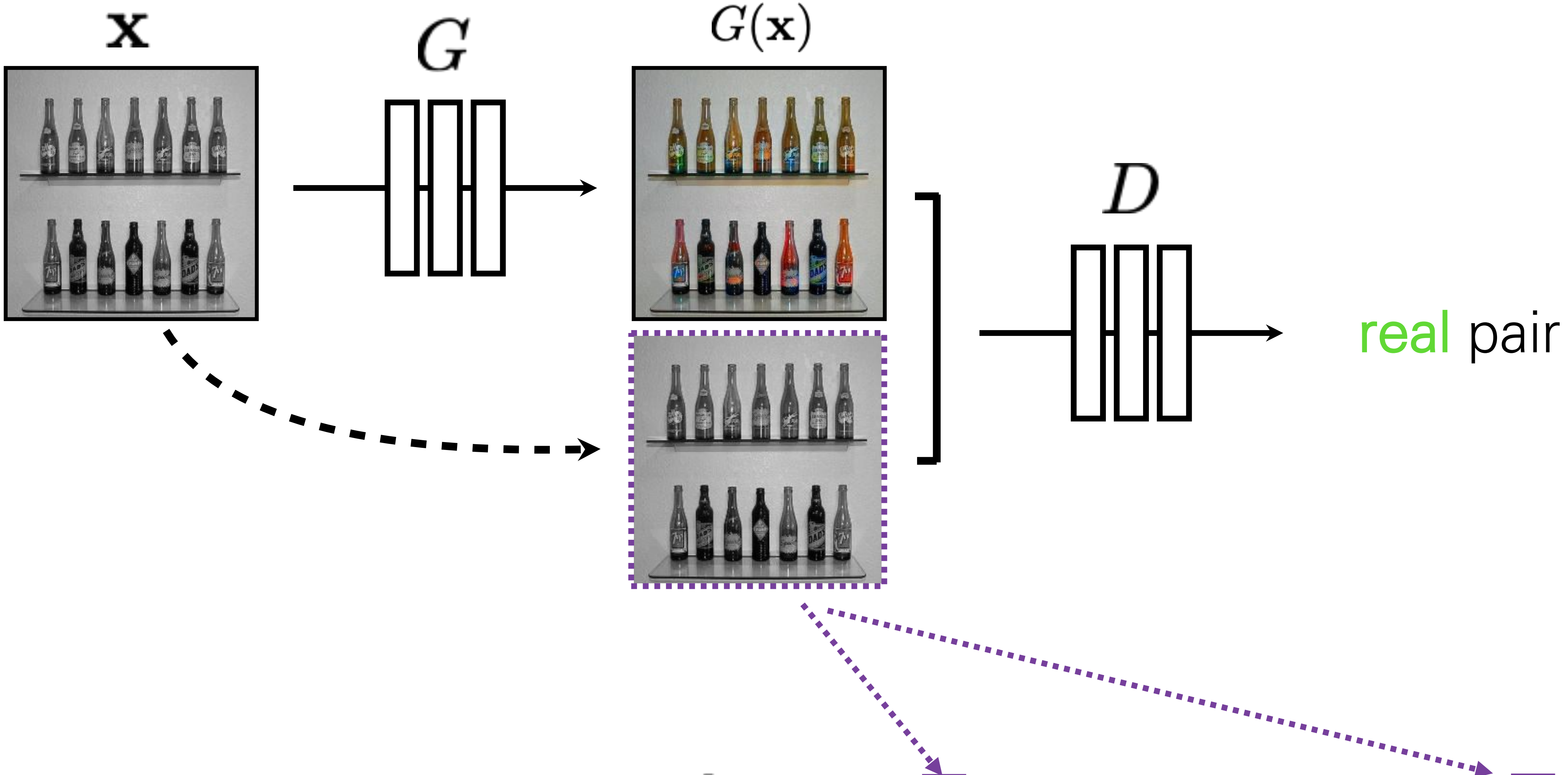
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))\ ]$$



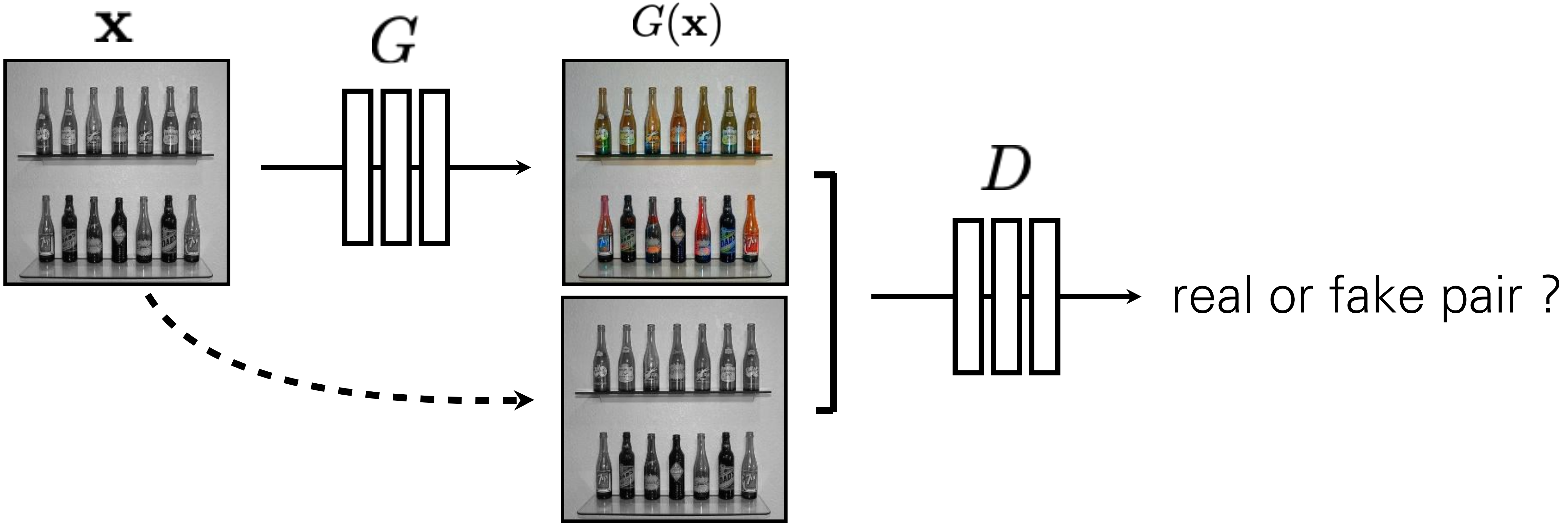
$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\;\log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$



$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



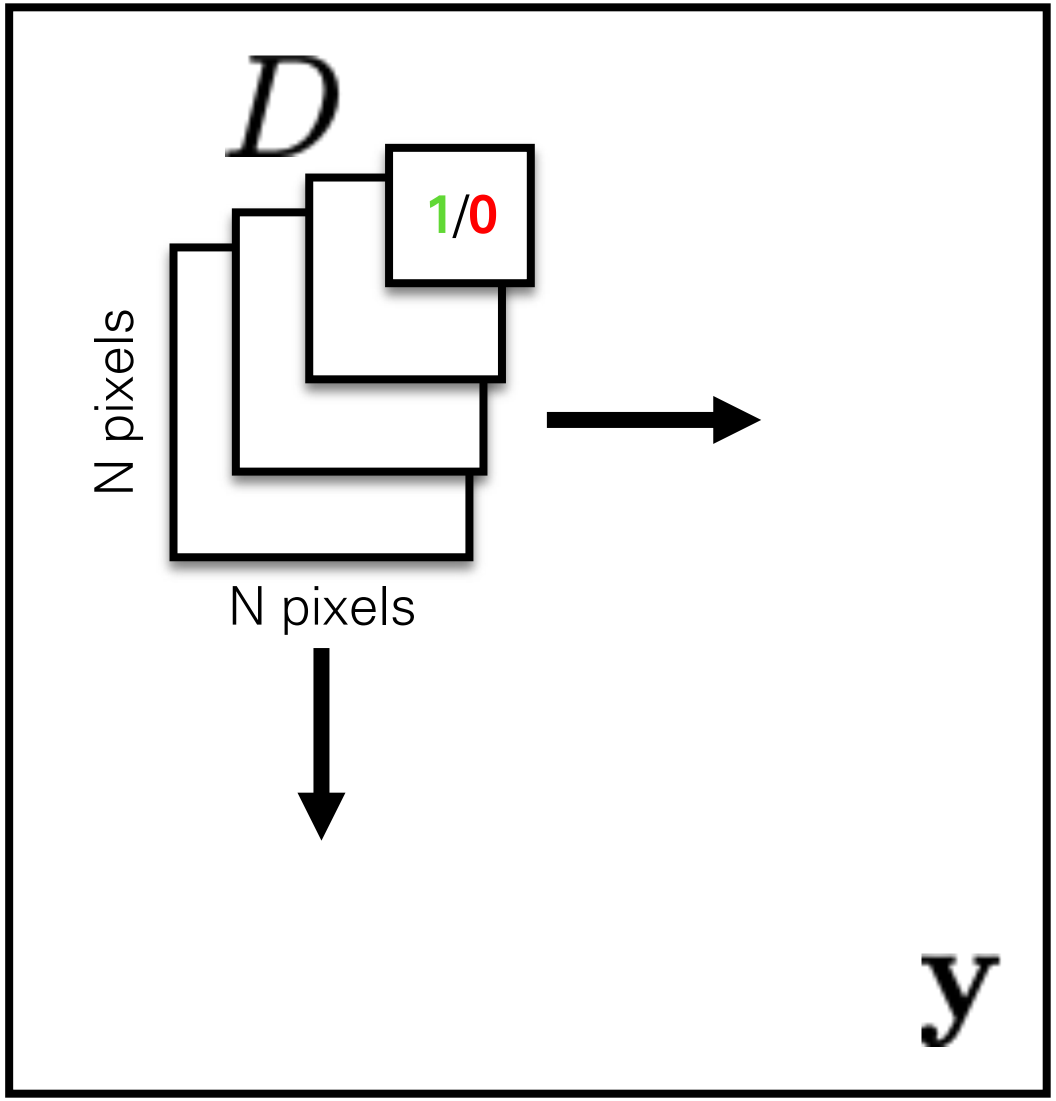
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x},\mathbf{y}))\ ]$$

BW → Color

Input Output Input Output Input Output

Data from [Russakovsky et al. 2015]

Shrinking the capacity: Patch Discriminator



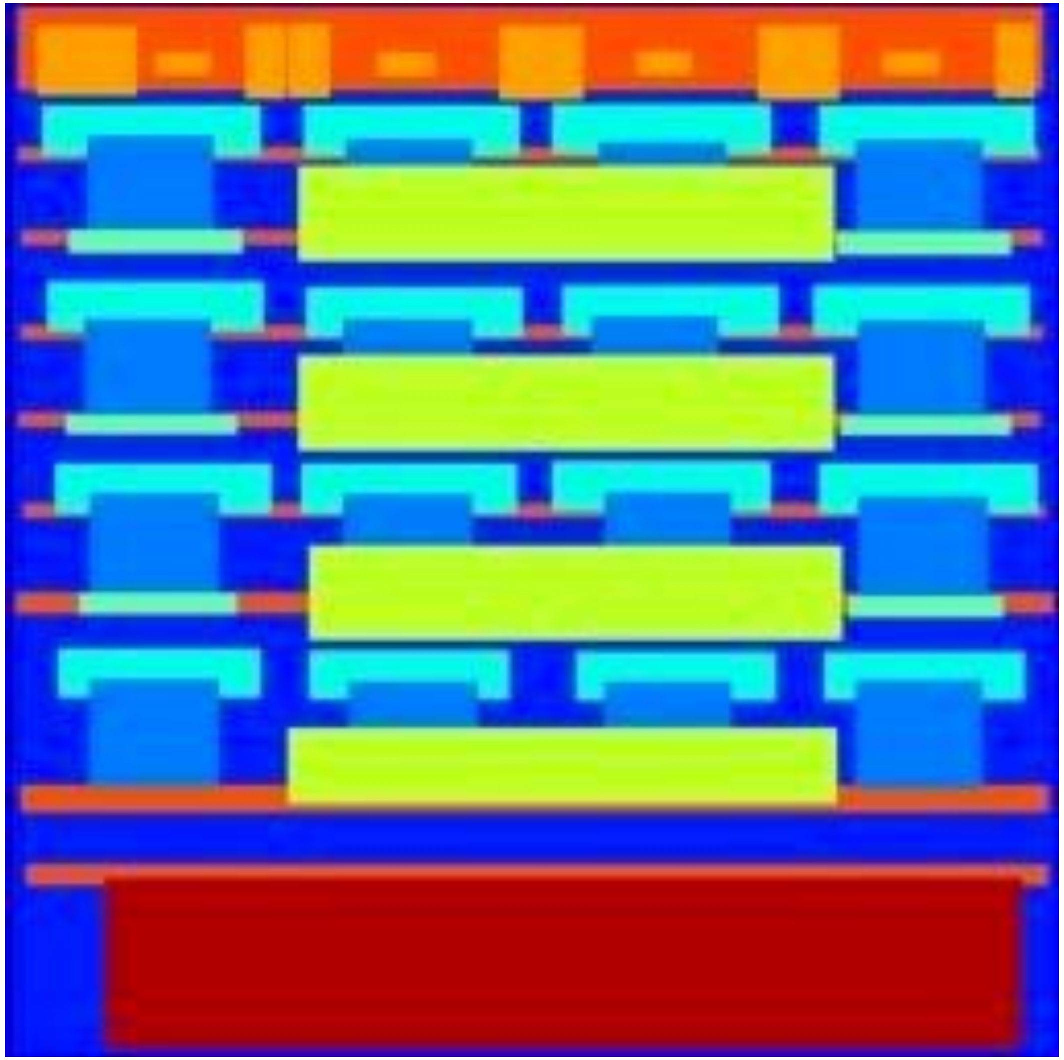
Rather than penalizing if output image looks fake, penalize if each overlapping patch in output looks fake

- Faster, fewer parameters
- More supervised observations
- Applies to arbitrarily large images

[Li & Wand 2016]
[Shrivastava et al. 2017]
[Isola et al. 2017]

Labels → Facades

Input

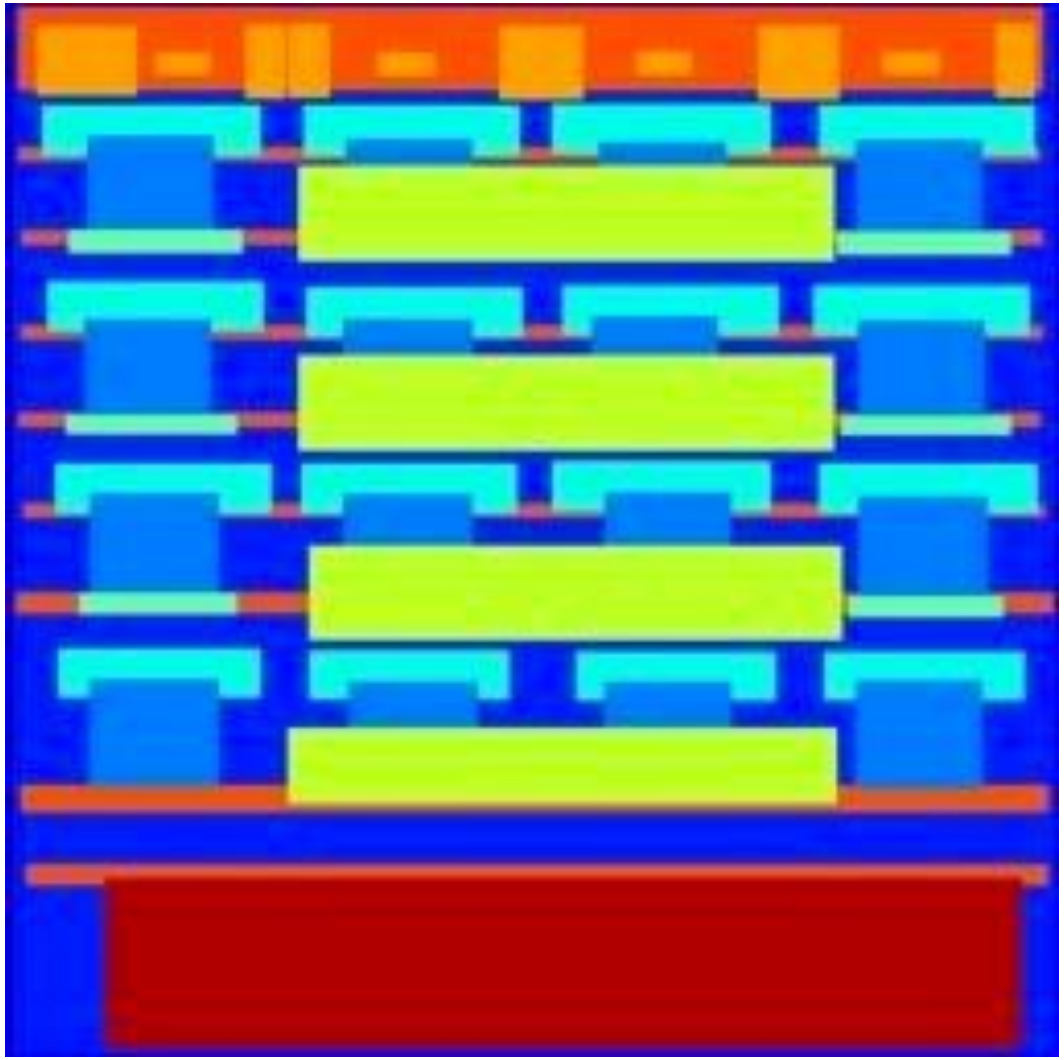


1x1 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

Input

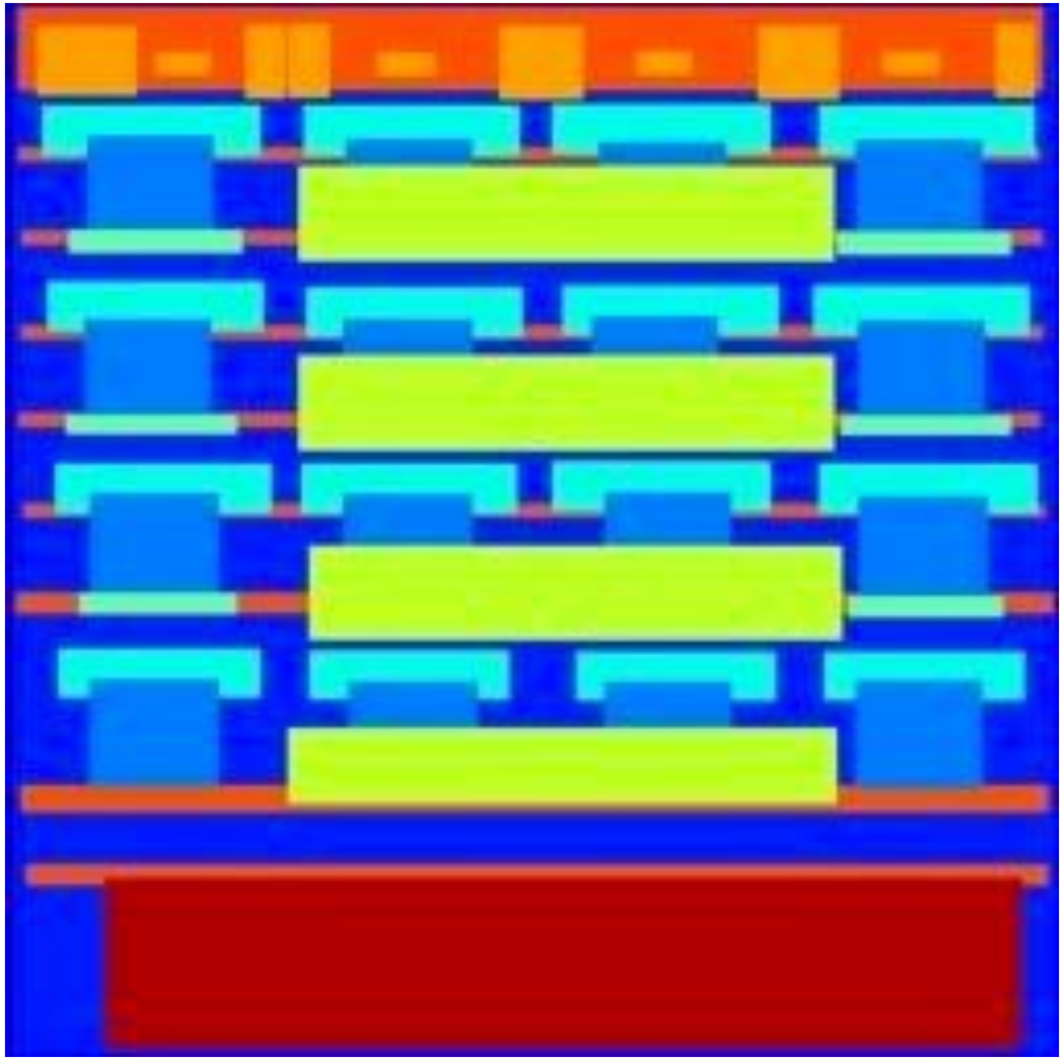


16x16 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

Input

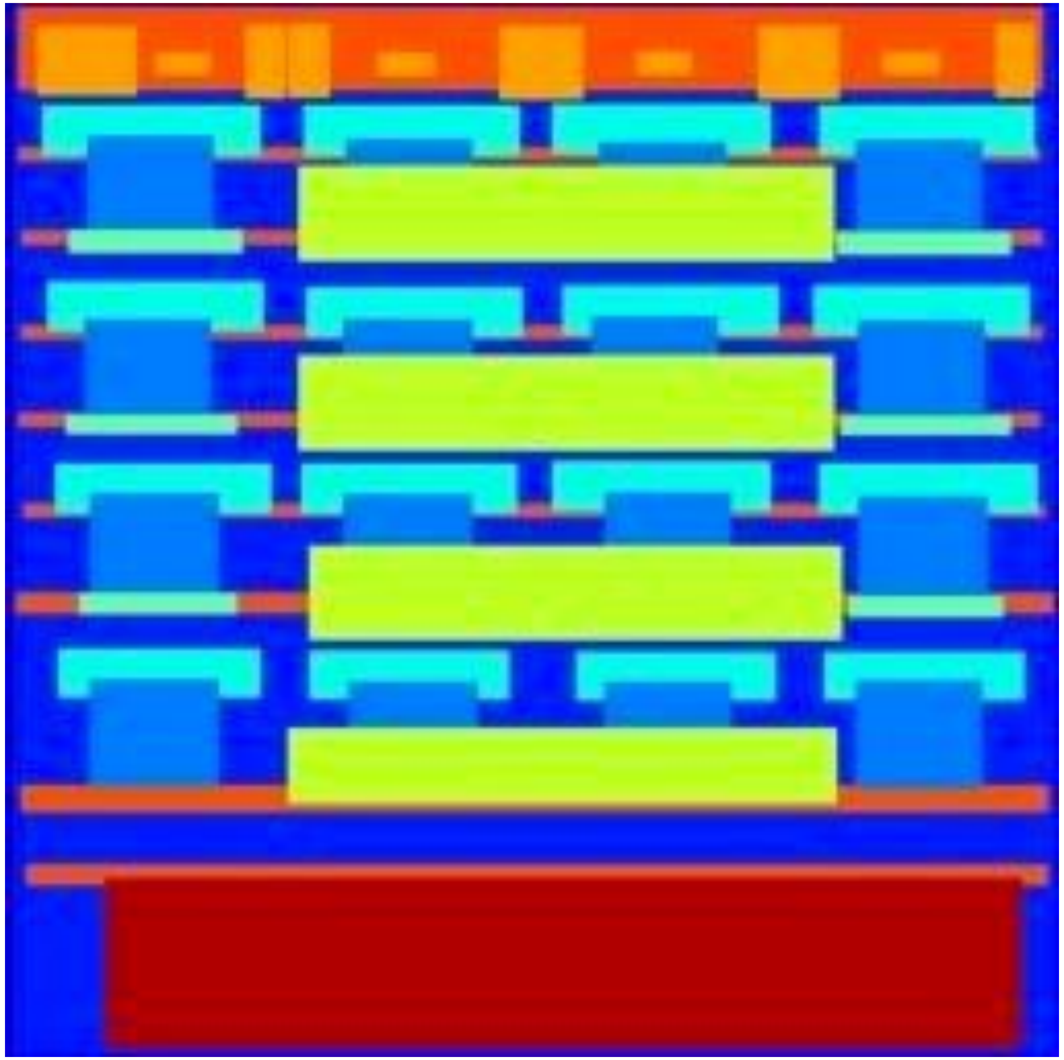


70x70 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

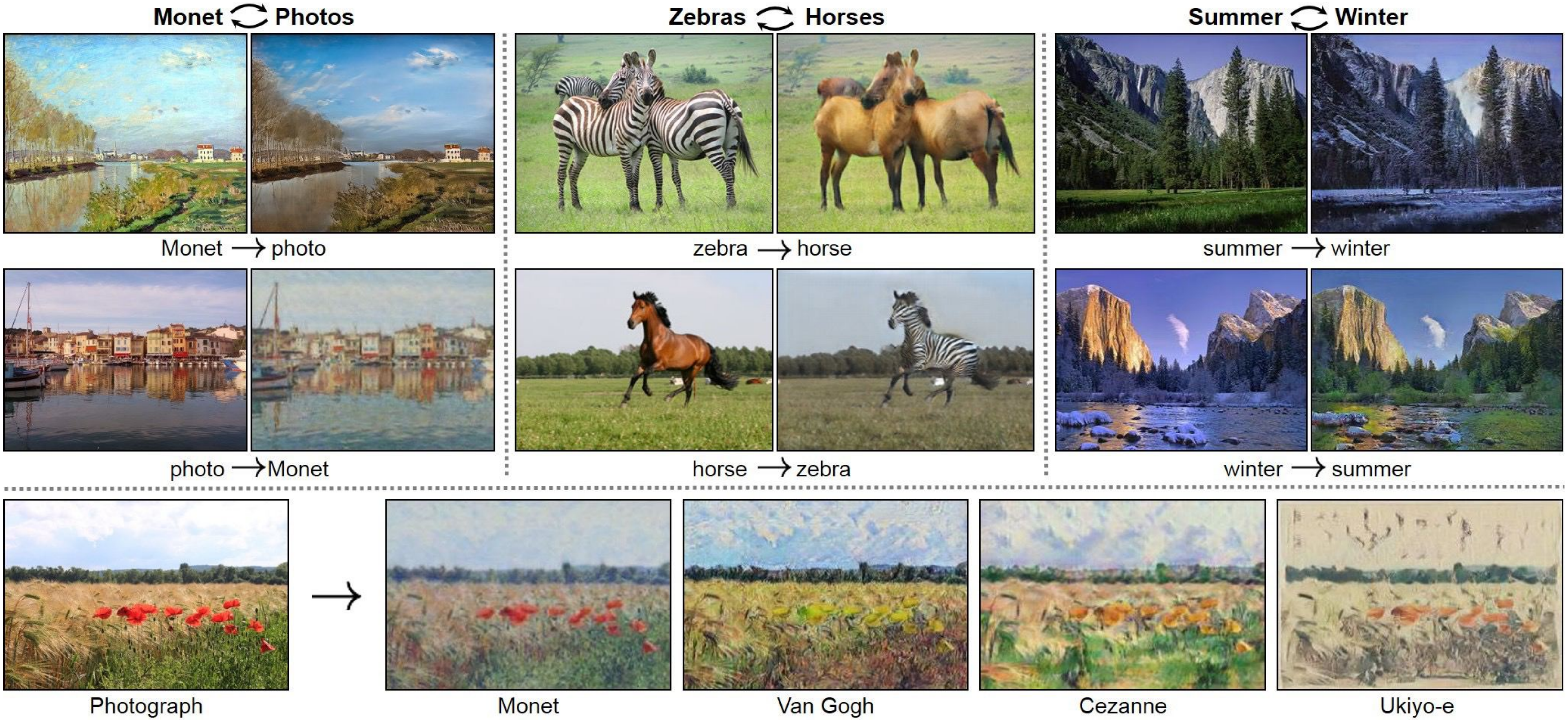
Input



Full image Discriminator

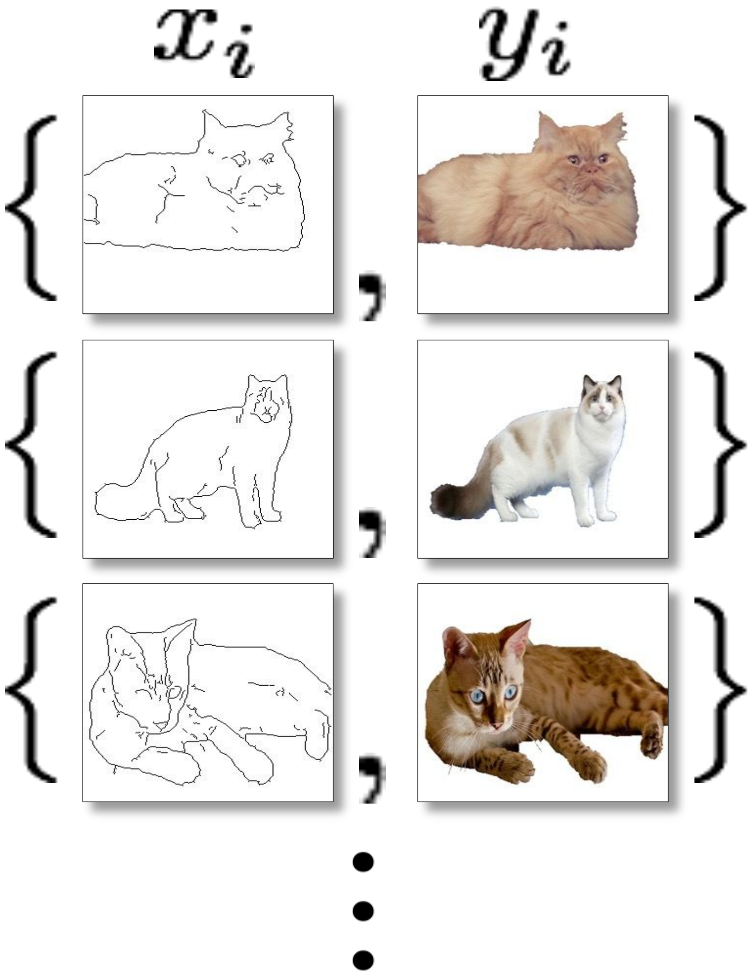
Data from [Tylecek, 2013]

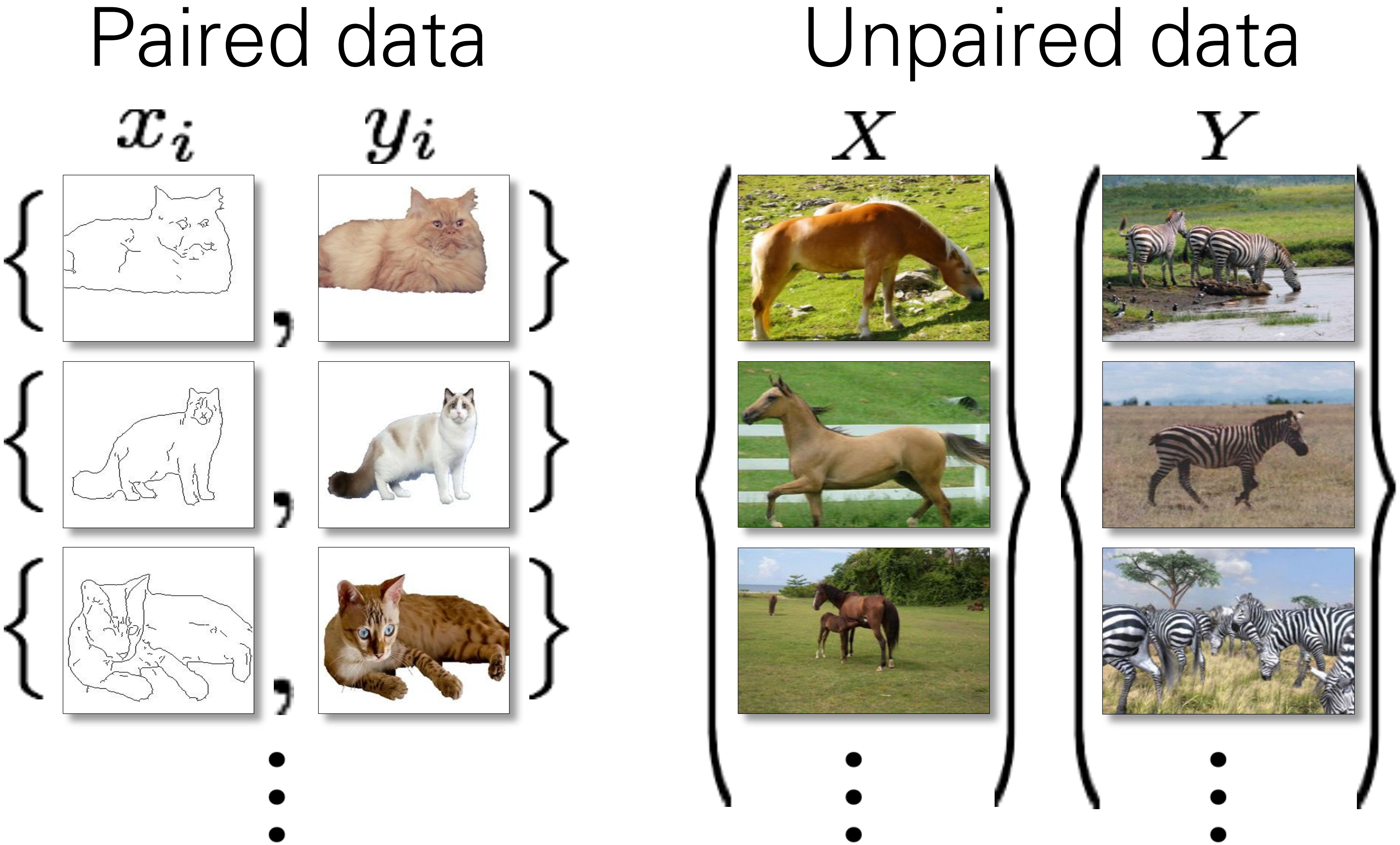
CycleGAN: Pix2Pix w/o input-output pairs

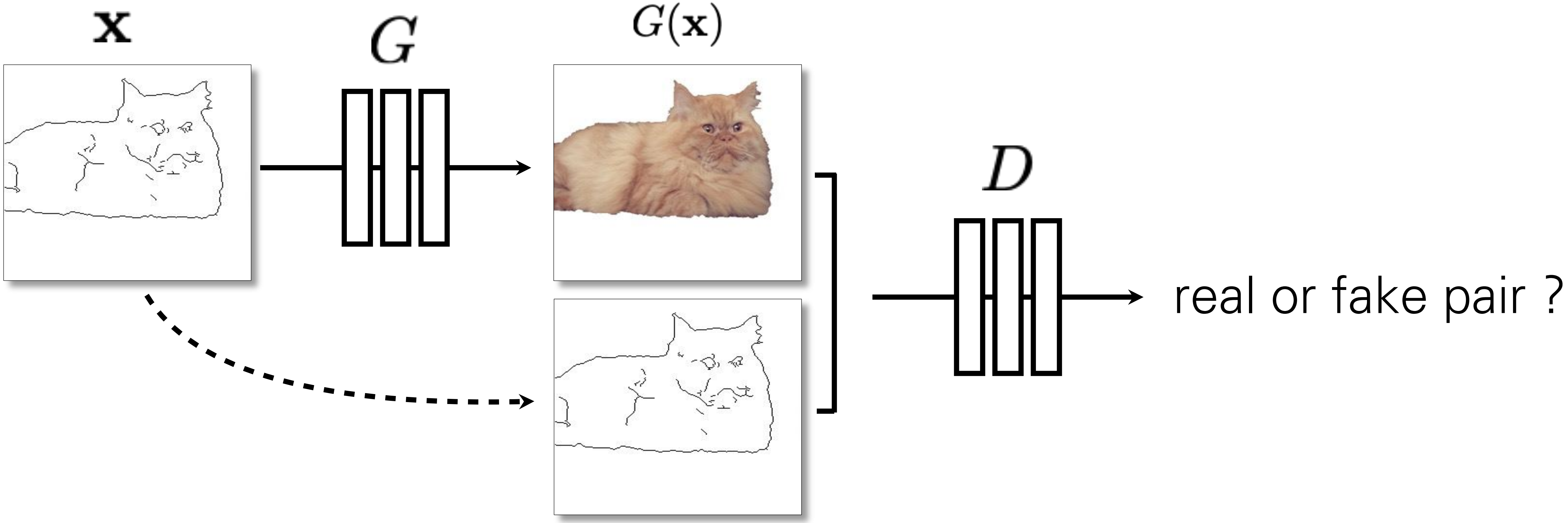


(Zhu et al. 2017)

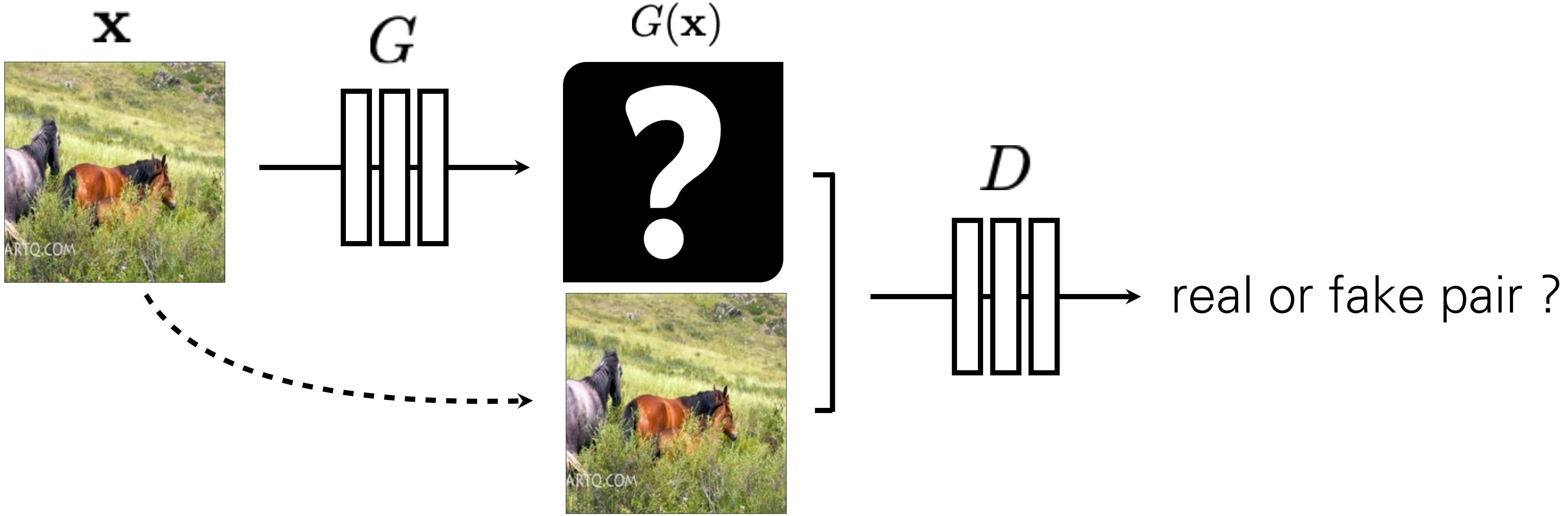
Paired data




Paired data
x_i
y_i
Unpaired data
X
Y

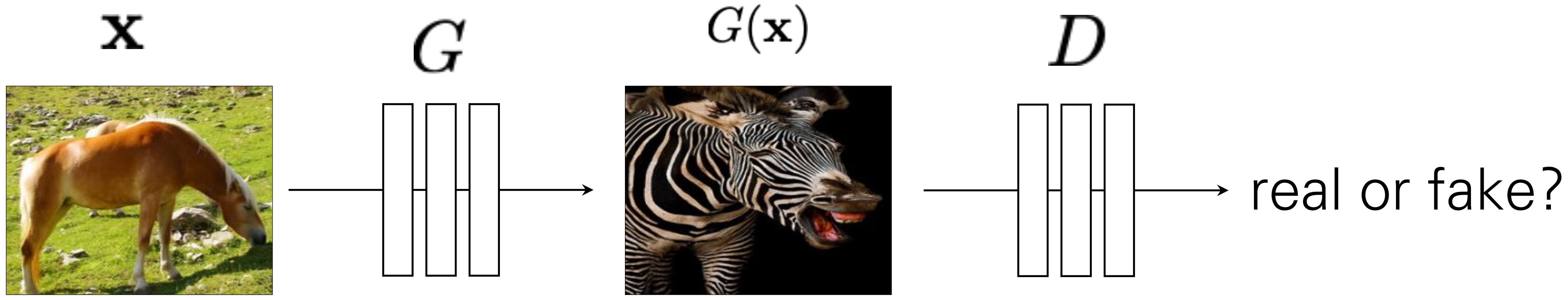


$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x},\mathbf{y}))\ ]$$

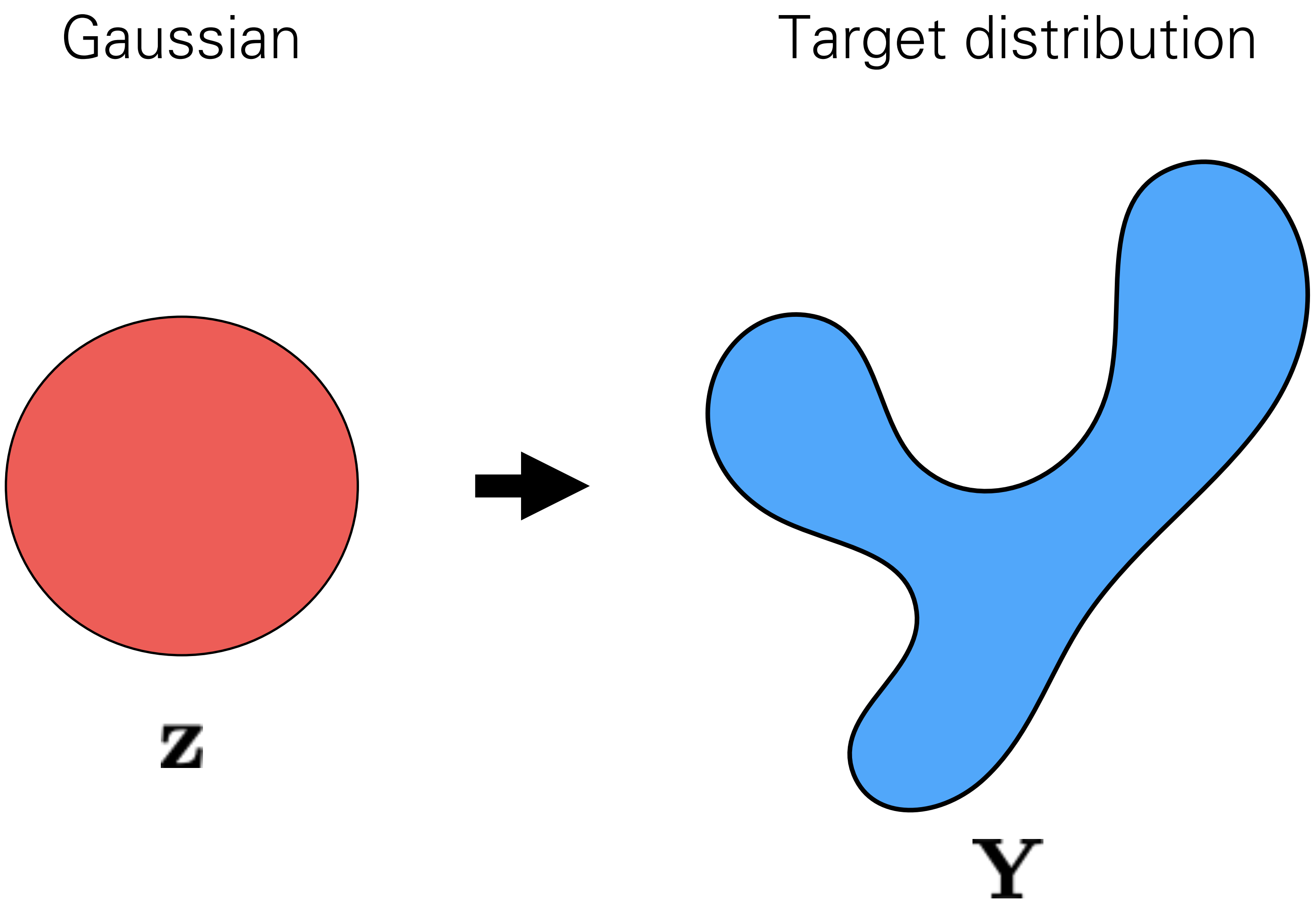
No input-output pairs!

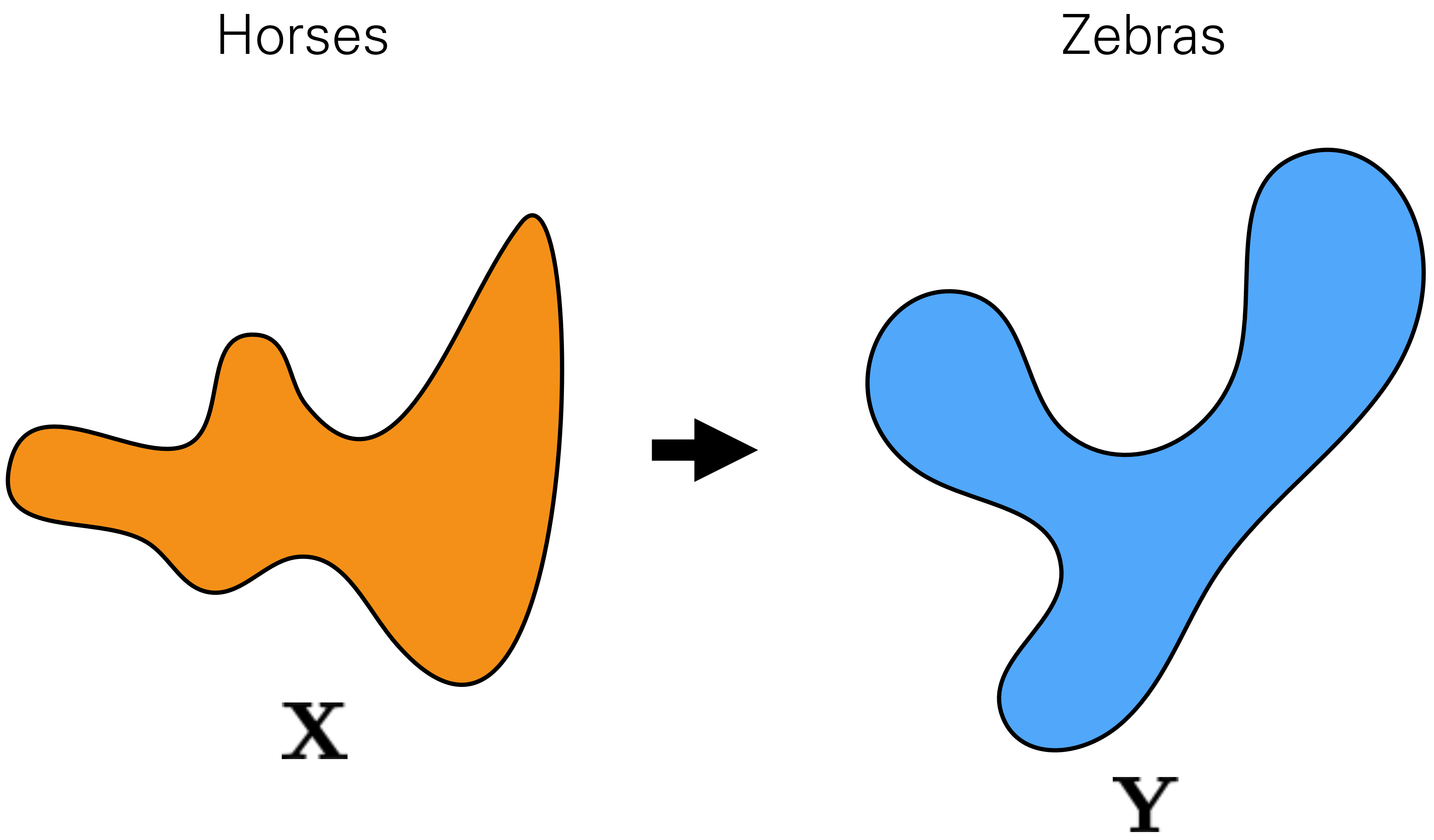


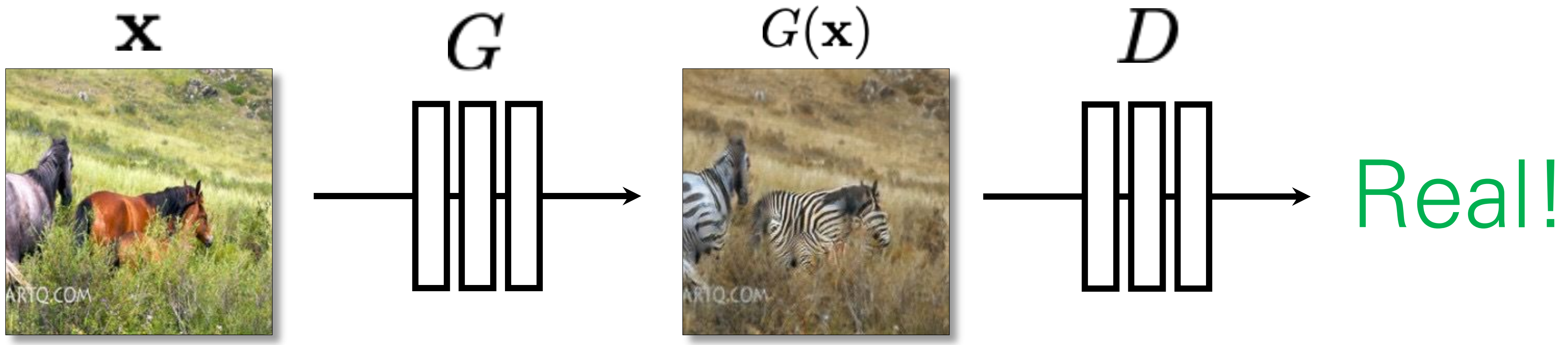
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x}))\ +\ \log(1-D(\mathbf{y}))\ ]$$

Usually loss functions check if output matches a target instance

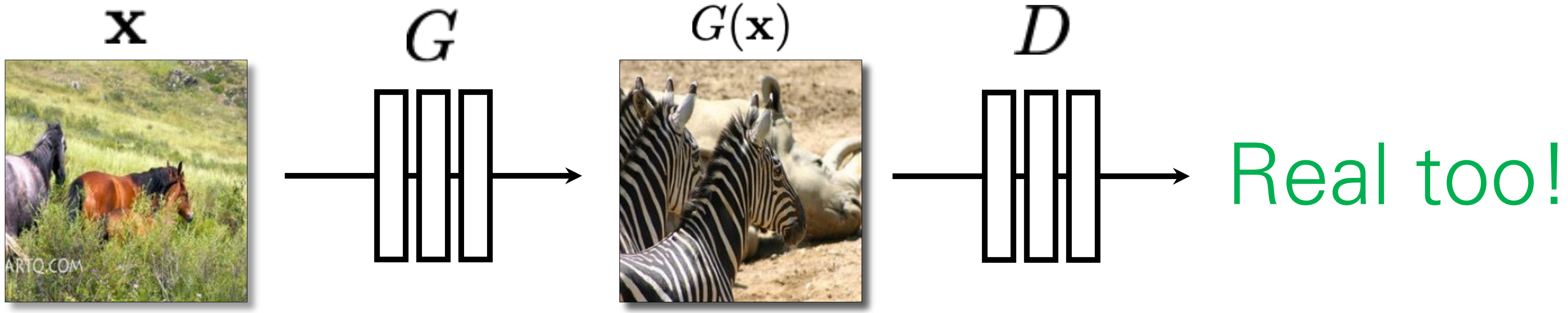
GAN loss checks if output is part of an admissible set


Gaussian
Target distribution
z
Y


Horses
Zebras
X
Y

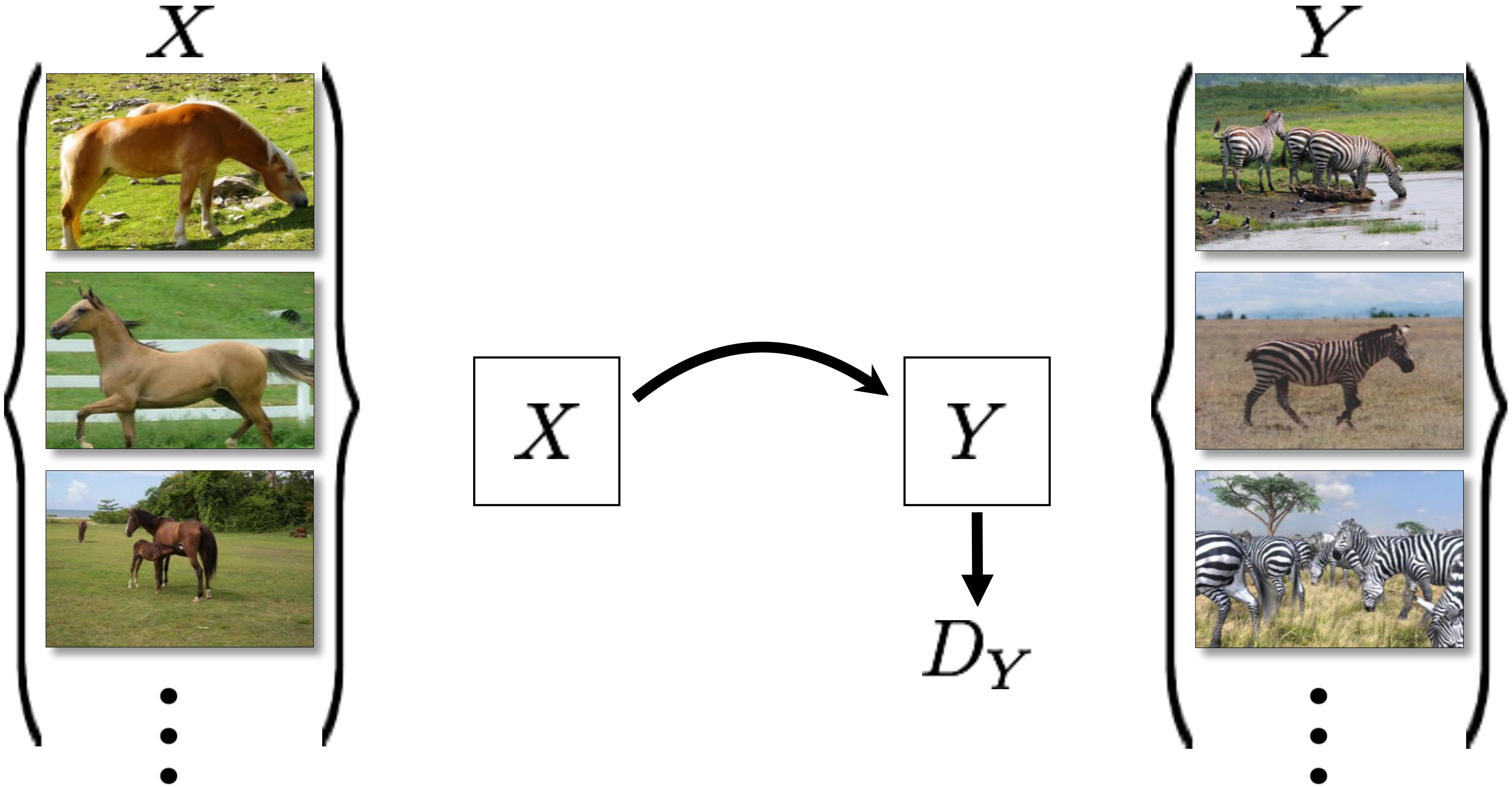


$\mathbf{x}$
G
$G(\mathbf{x})$
D
Real!



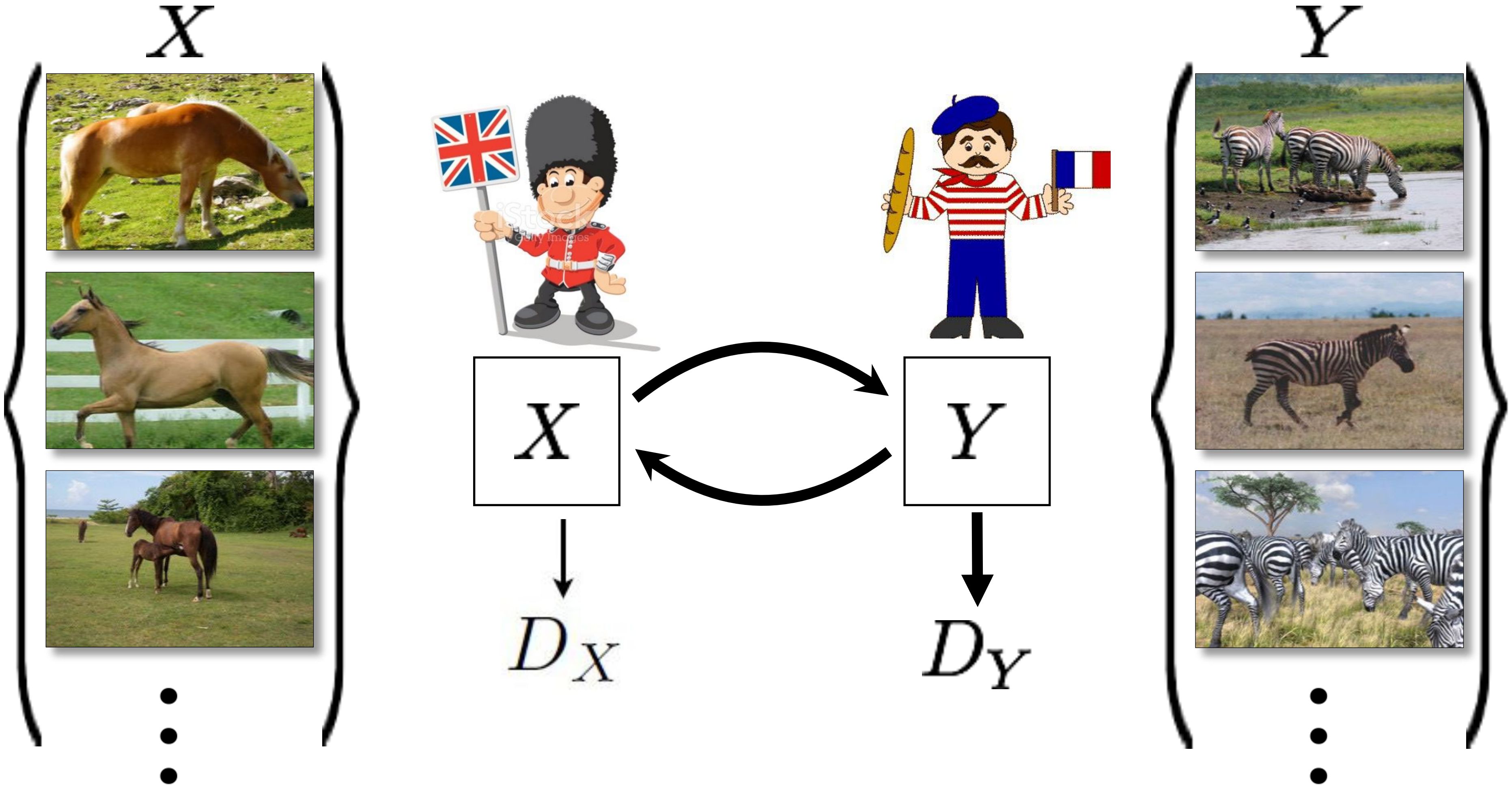
Nothing to force output to correspond to input

Cycle-Consistent Adversarial Networks

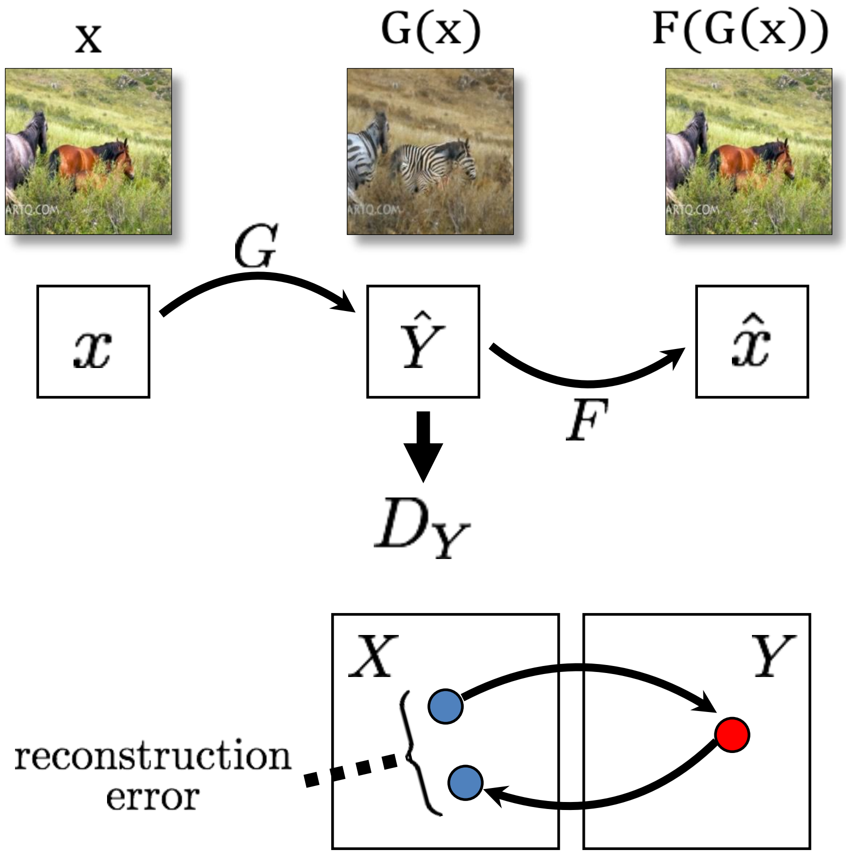


[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

Cycle-Consistent Adversarial Networks

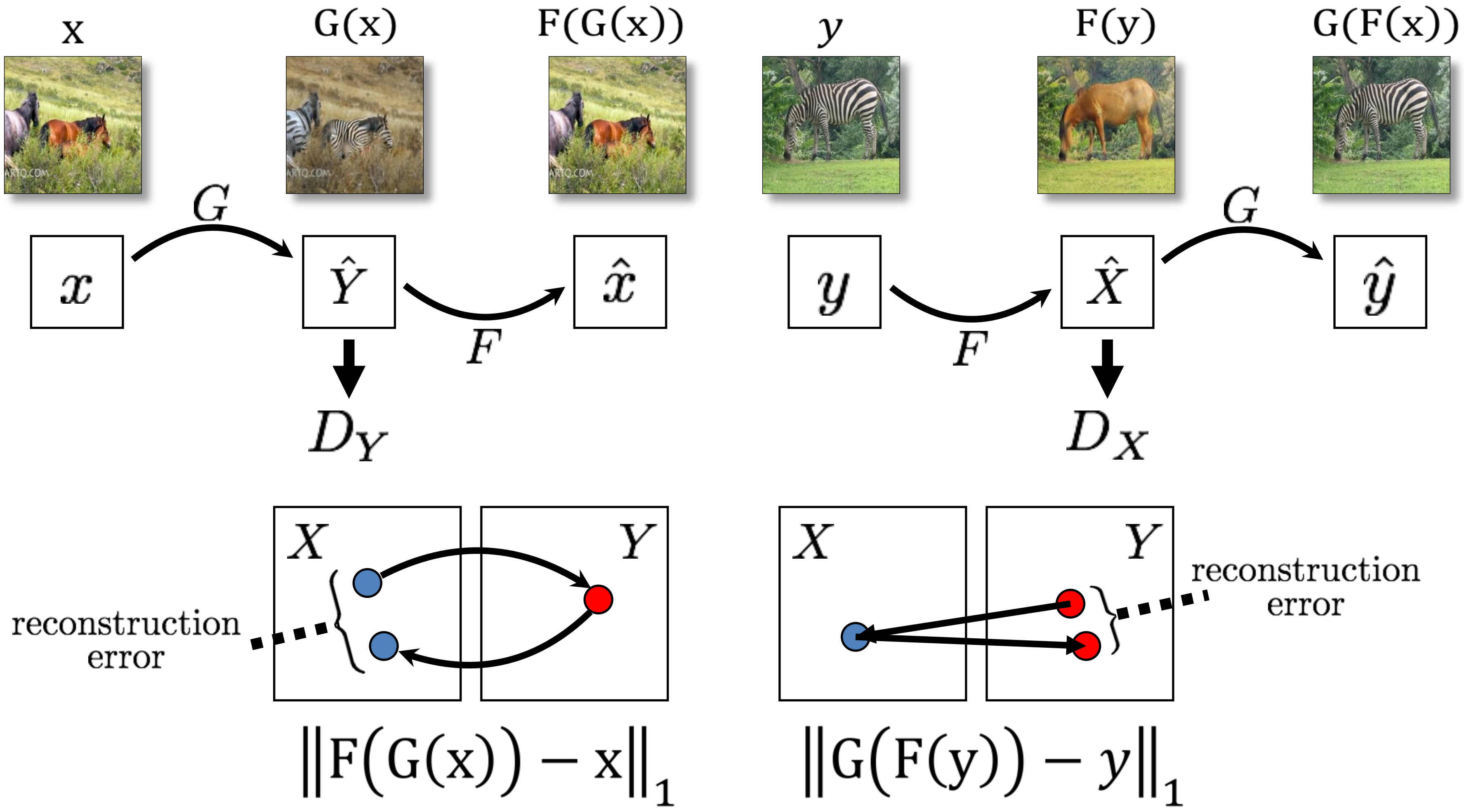


Cycle Consistency Loss



$$\left\| F(G(x)) - x \right\|_1$$

Cycle Consistency Loss




Collection Style Transfer

Photograph
@ Alexei Efros

Monet

Van Gogh

Cezanne

Ukiyo-e

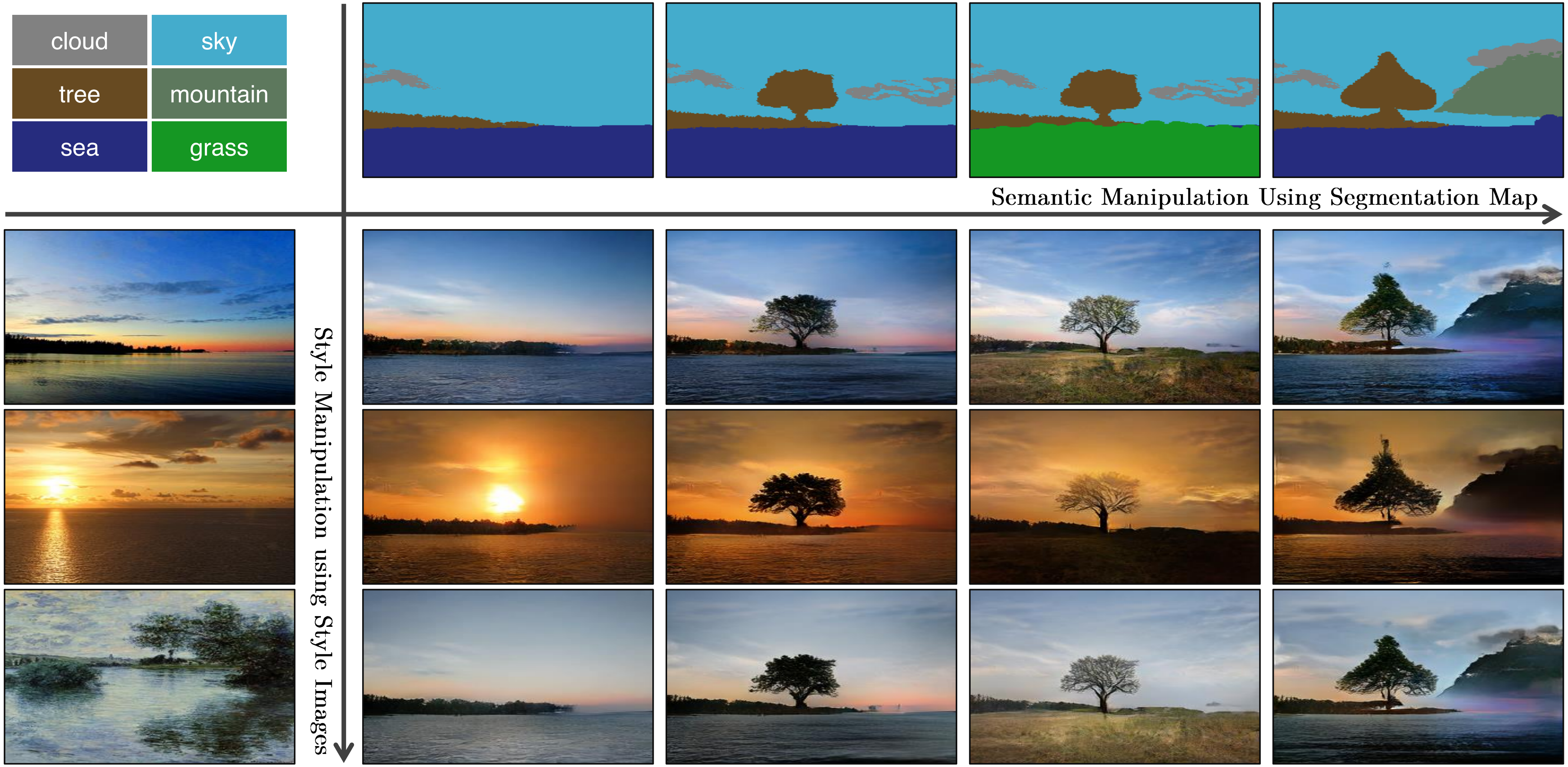

Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Monet's paintings → photos

Monet's paintings → photos

Semantic Image Synthesis (SPADE) (Park et al., 2019)

- Image generation conditioned on semantic layouts



Semantic layout

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

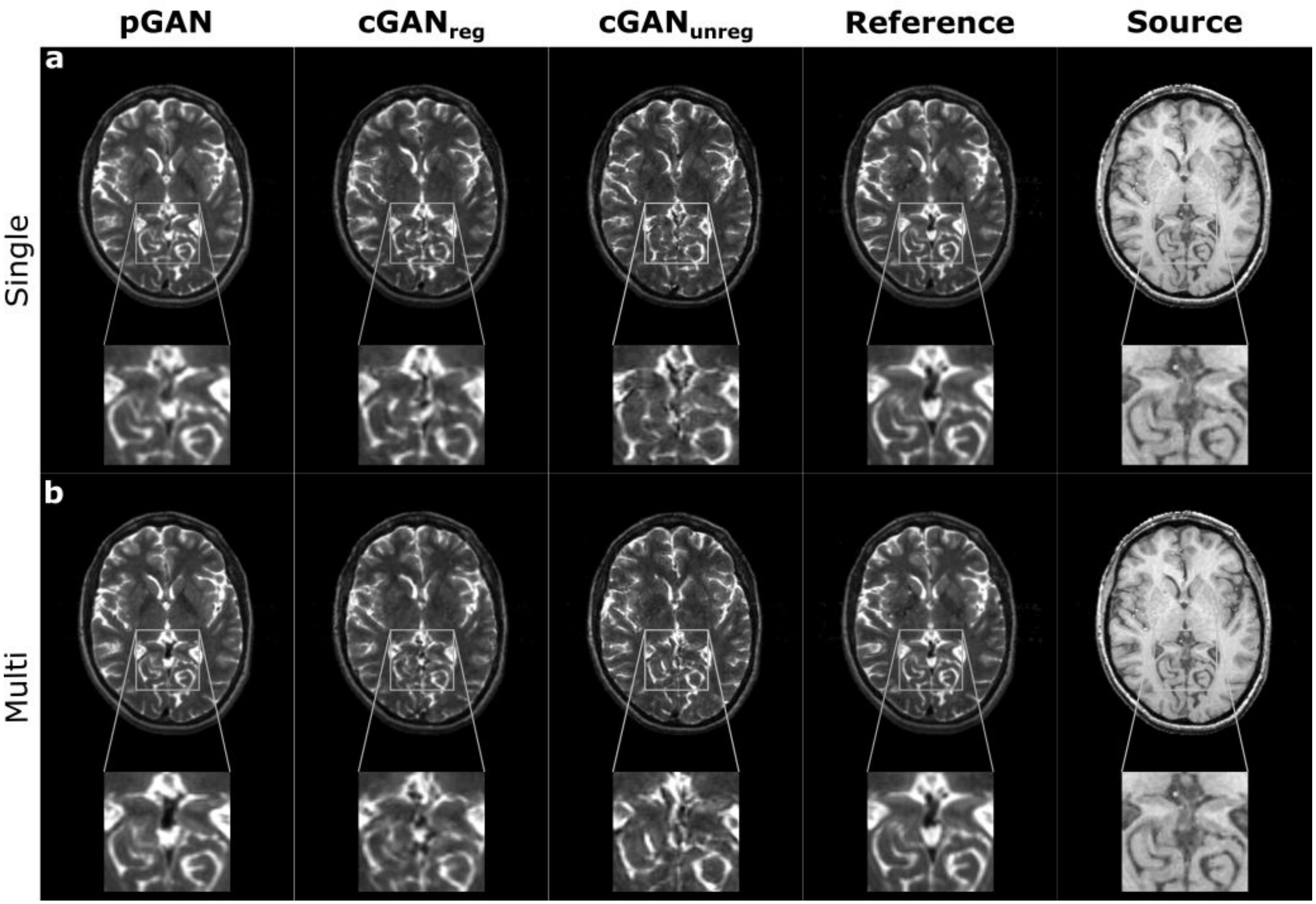
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

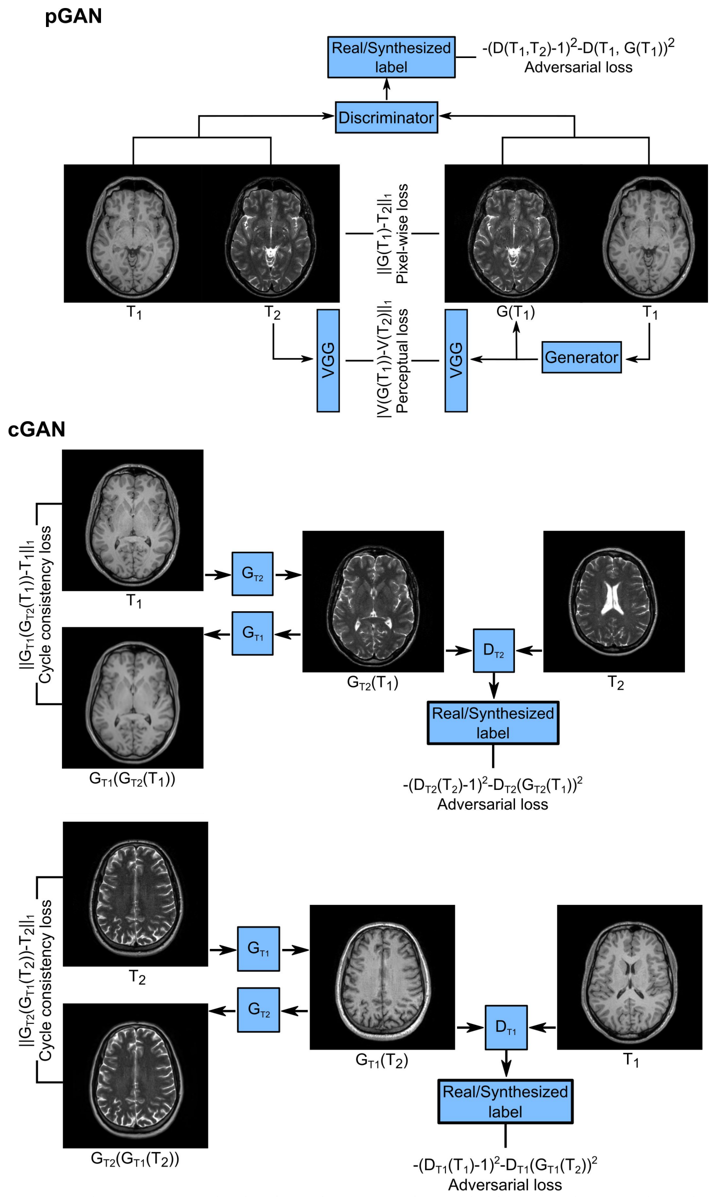
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

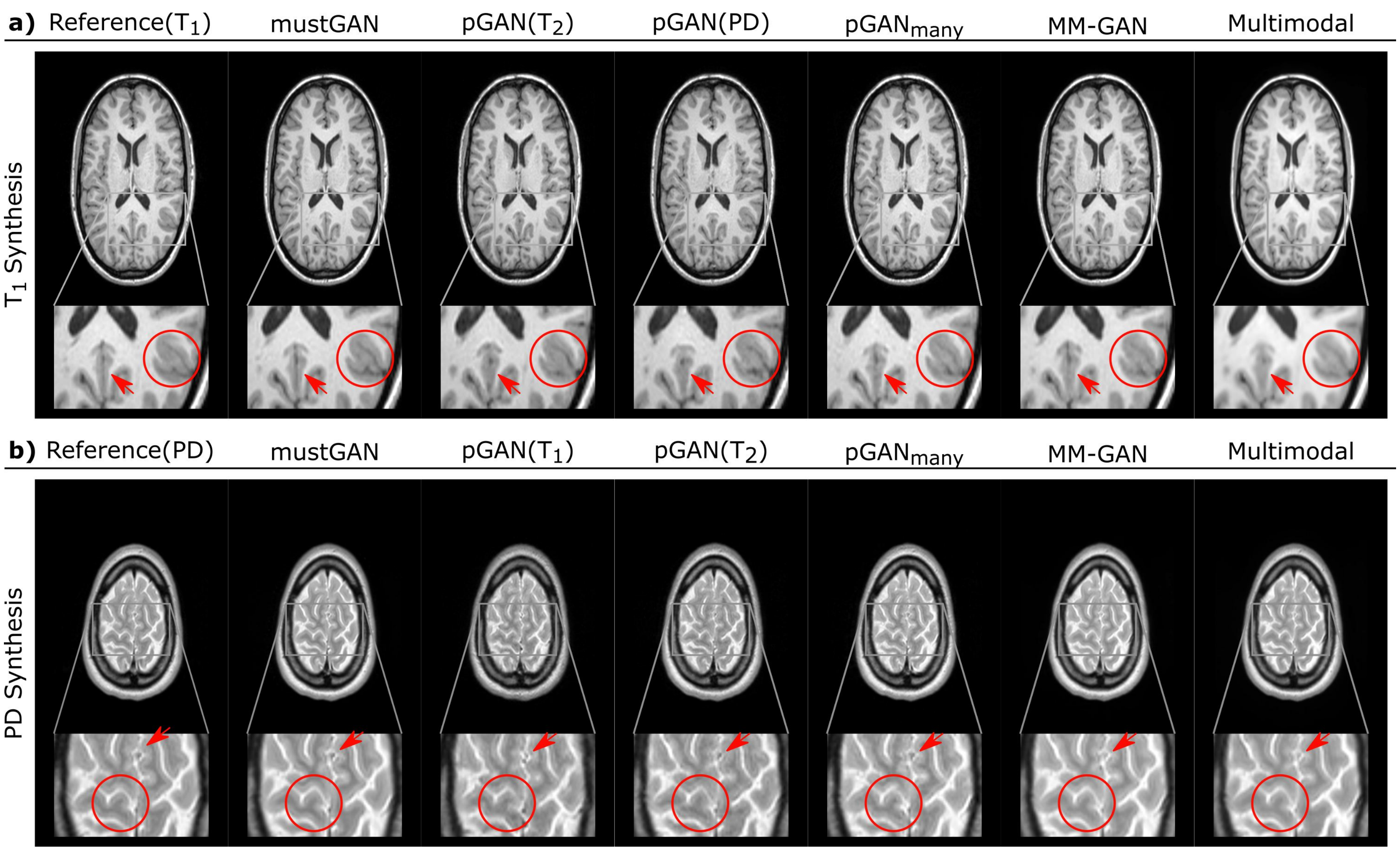
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

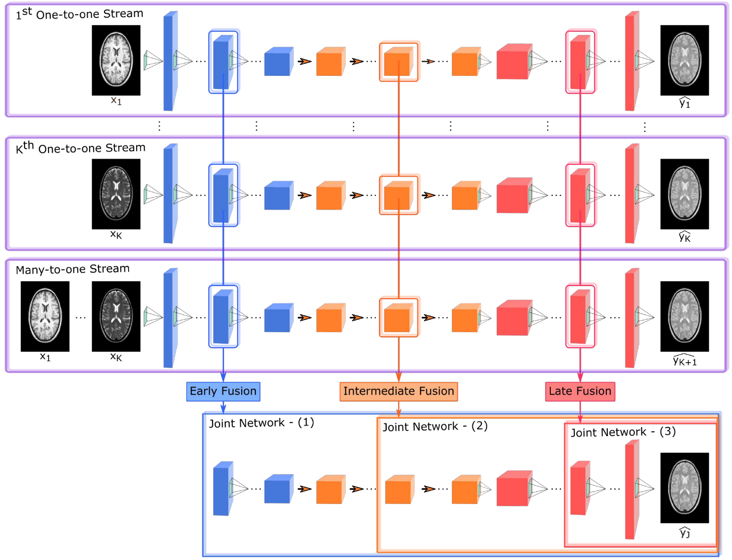
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2018]





- Image Synthesis in Multi-Contrast MRI [Ul Hassan Dar et al. 2019]

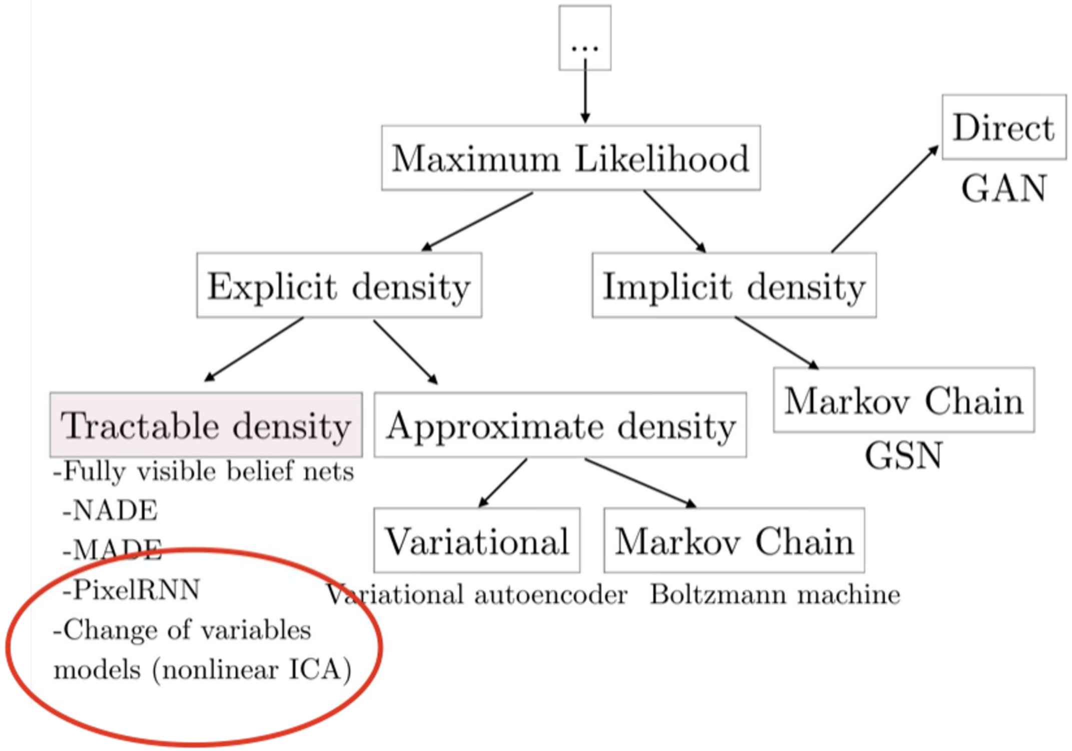




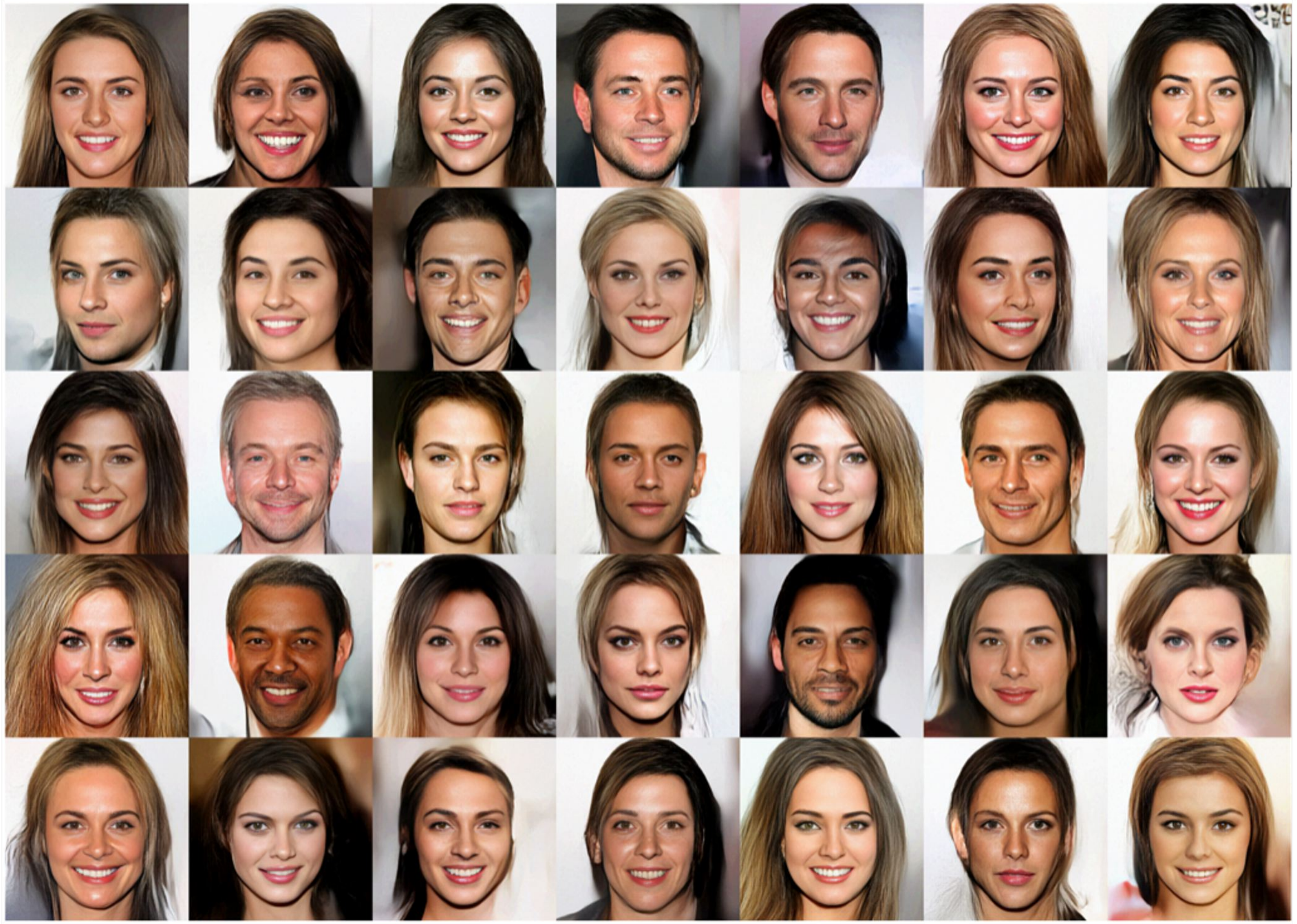
- Image Synthesis in Multi-Contrast MRI [Mahmut Yurt et al. 2021]

Flow-Based Models

Invertible Neural Networks

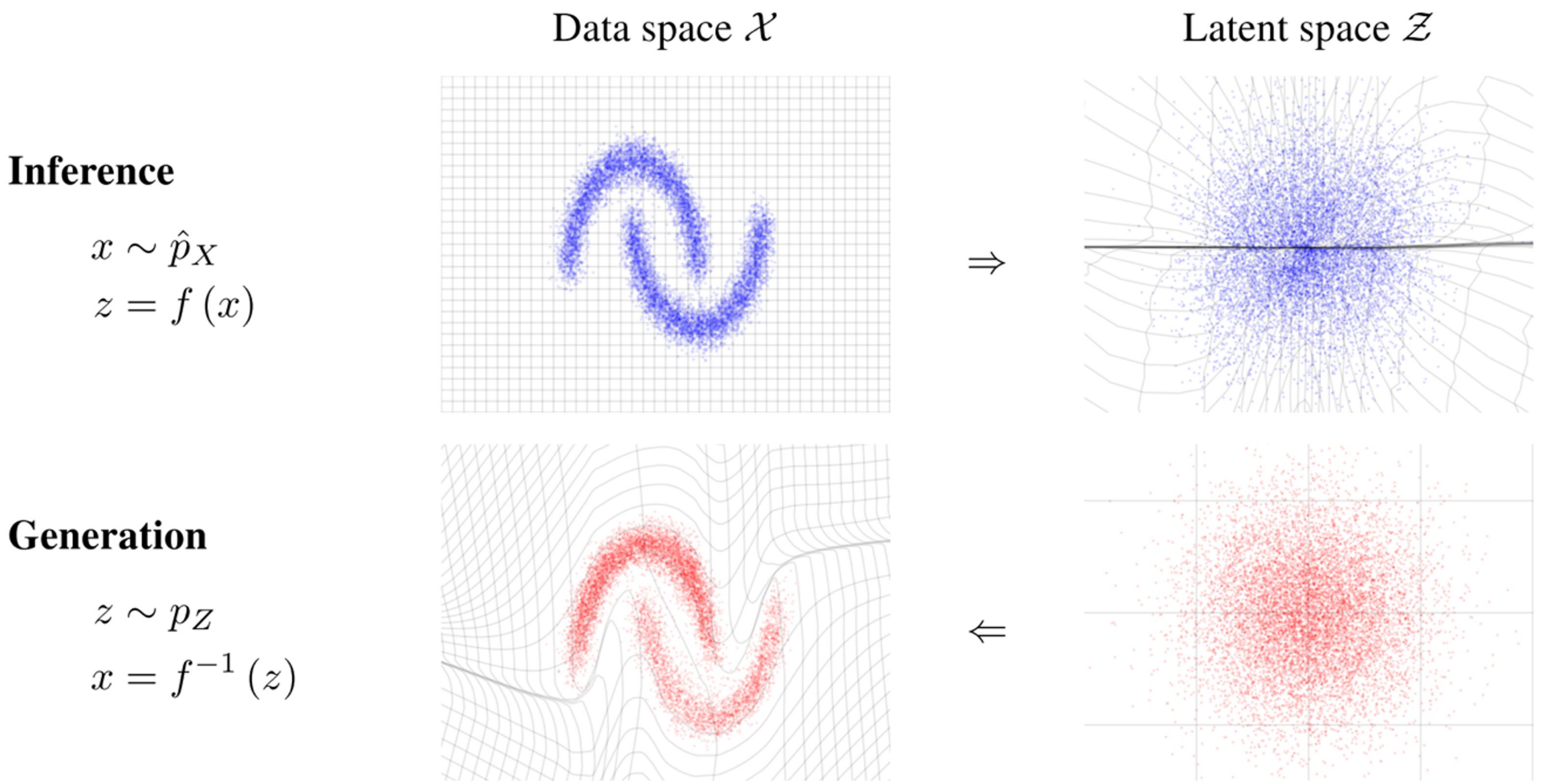


Goodfellow et al. 2016



Kingma and Dhariwal 2018

Normalizing Flows: Translating Probability Distributions



Change of Variable Density Needs to Be Normalized

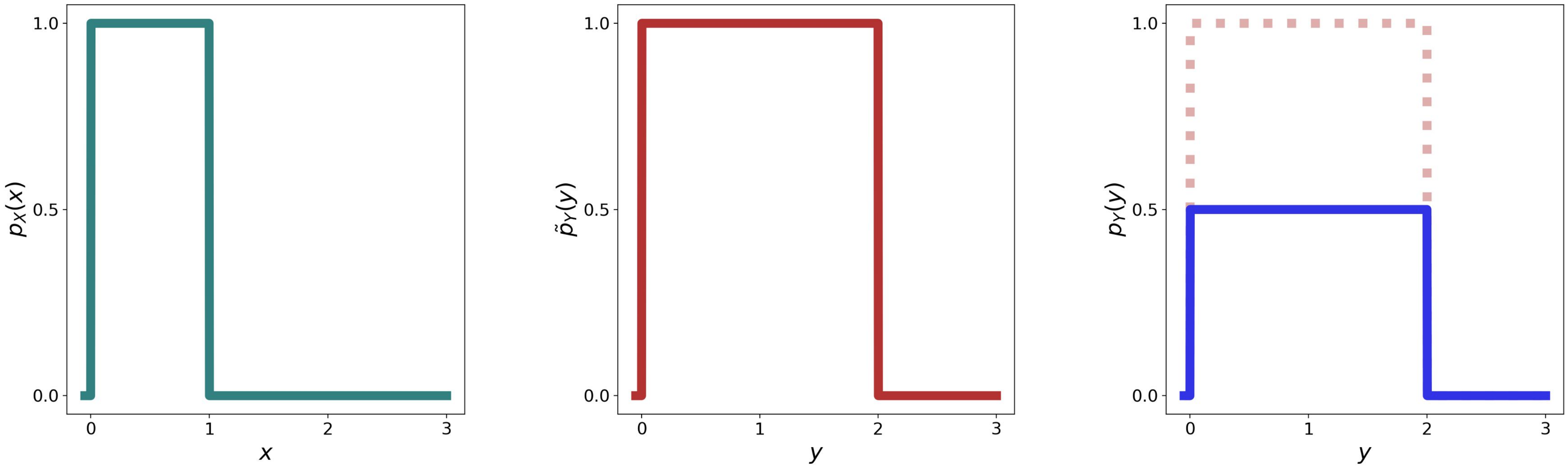
$$X \sim p_X$$

$$Y := 2X$$

$$p_X(x) = \begin{cases} 1 & \text{for } 0 \leq x \leq 1 \\ 0 & \text{else} \end{cases}$$

$$\tilde{p}_Y(y) = p_X(y/2)$$

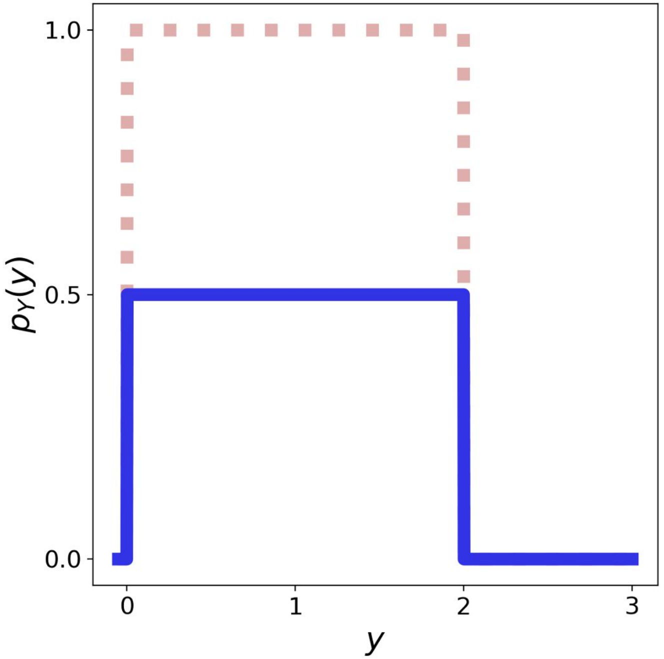
$$p_Y(y) = p_X(y/2)/2$$



Change of Variable Density (m-Dimensional)

For a multivariable invertible mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ $X \sim p_X$ $Y := f(X)$

$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{\partial f^{-1}(y)}{\partial y} \right|$$



$Y := 2X$

$p_Y(y) = p_X(y/2)/2$

Local change of volume

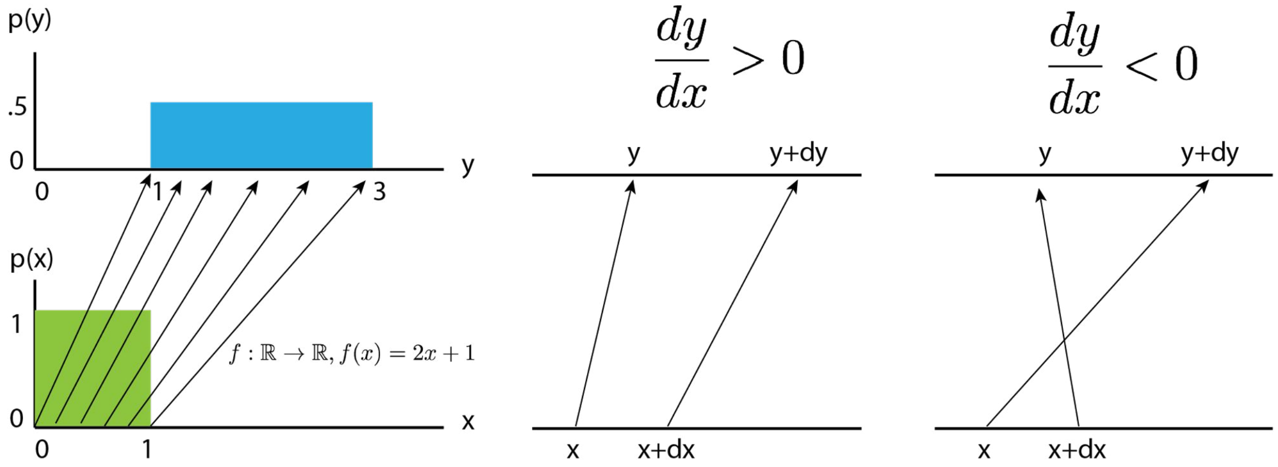
mass = density * volume

Change of Variable Density (m-Dimensional)

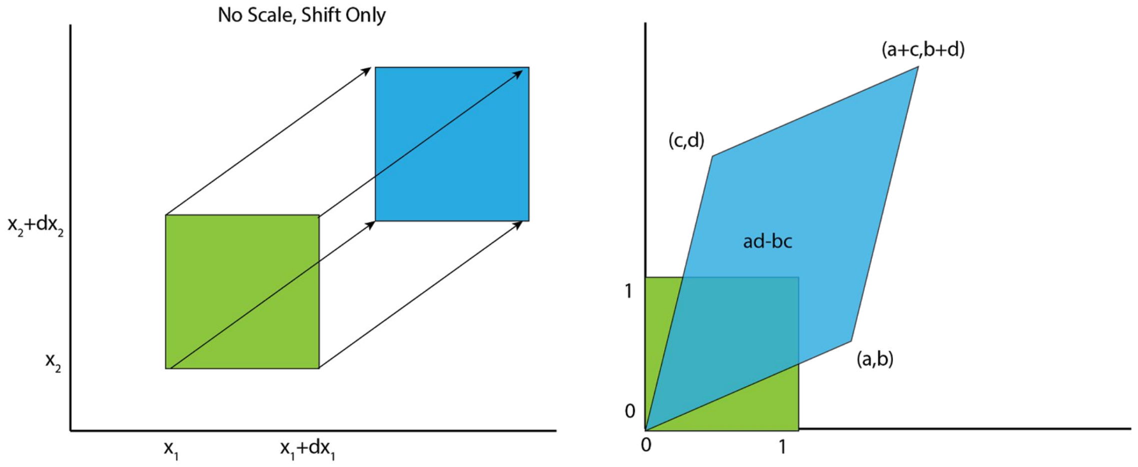
For a multivariable invertible mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ $\quad X \sim p_X \quad Y := f(X)$

$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{\partial f^{-1}(y)}{\partial y} \right|$$

1-D

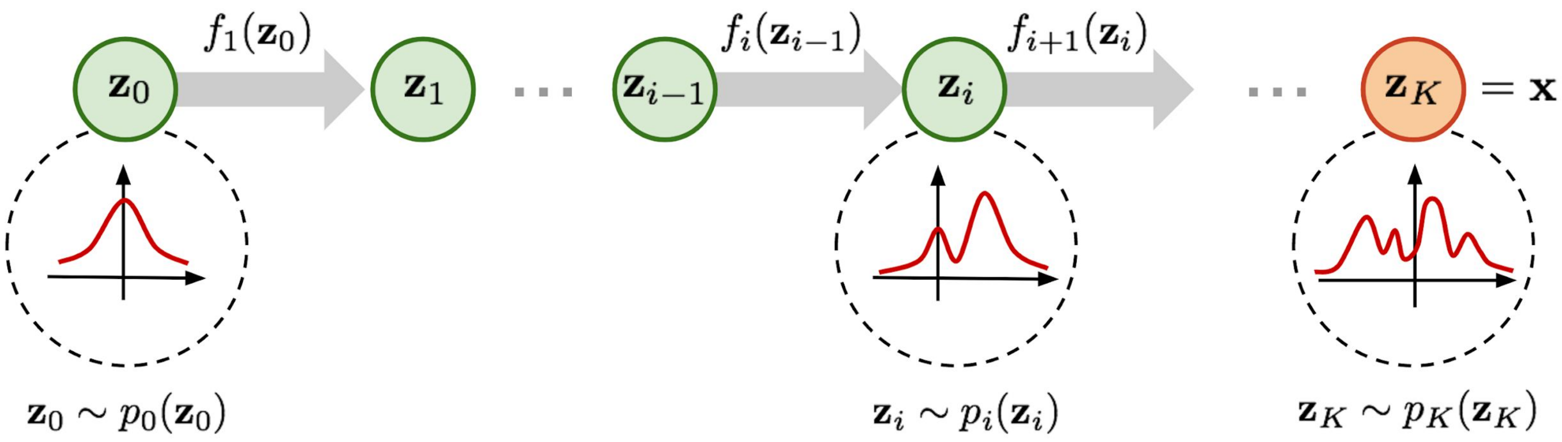


2-D



Chaining Invertible Mappings (Composition)

$$f = f_S \circ \cdots \circ f_2 \circ f_1 \qquad f(x) = f_S(\cdots f_2(f_1(x)))$$



$$\frac{\partial f(x)}{\partial x} = \frac{f_S(x_{S-1})}{\partial x_{S-1}} \cdots \frac{f_2(x_1)}{\partial x_1} \frac{f_1(x_0)}{\partial x_0} \qquad x_s = f_s(x_{s-1}), \quad x_0 = x$$

Chain rule

$$\det\left(\frac{\partial f(x)}{\partial x}\right) = \det\left(\frac{f_S(x_{S-1})}{\partial x_{S-1}}\right) \cdots \det\left(\frac{f_2(x_1)}{\partial x_1}\right) \det\left(\frac{f_1(x_0)}{\partial x_0}\right)$$

Determinant of matrix product

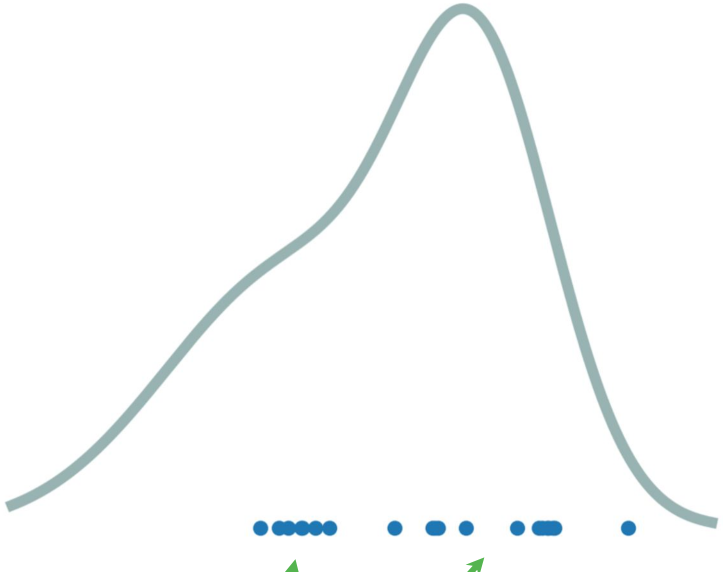
Figure from blog post: Flow-based Deep Generative Models by Lilian Weng, 2018

Training with Maximum Likelihood Principle

$$Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad X = g(Z)$$

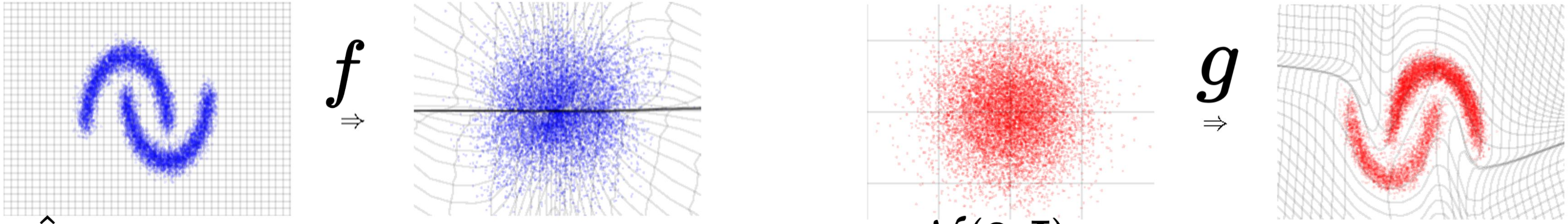
$$g = f^{-1} \text{ bijective}$$

$$\mathbb{E}_x [\log p(x)] = \mathbb{E}_x \left[\log \mathcal{N}(f(x); 0, I) \left| \det \frac{\partial f(x)}{\partial x} \right| \right]$$



Regularizes the entropy

Higher likelihood



$x \sim \hat{p}_X$ Inference $z \sim p_Z = \mathcal{N}(\mathbf{0}, \mathbf{I})$ Generation

Pathways to Designing a Normalizing Flow

1. Require an invertible architecture.
   - Coupling layers, autoregressive, etc.

2. Require efficient computation of a change of variables equation.

$$\log p(x) = \log p(f(x)) + \log \left| \det \frac{df(x)}{dx} \right|$$

Model distribution — Base distribution

$\det \frac{df(x)}{dx}$ → $\mathcal{O}(m^3)$

(or a continuous version) $\log p(x(t_N)) = \log p(x(t_0)) + \int_{t_0}^{t_N} \operatorname{tr}\left(\frac{\partial f(x(t),t)}{\partial x(t)}\right) dt$

Slide by Ricky Chen

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

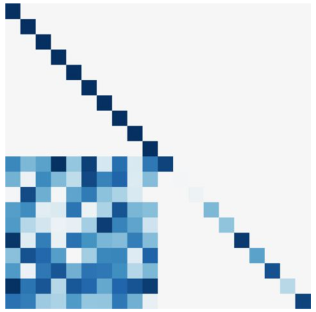
Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

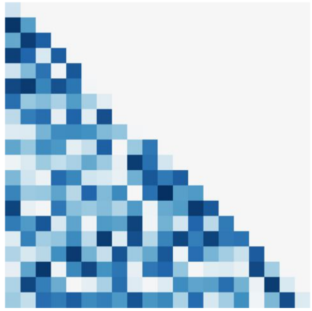
Jacobian



(Lower triangular + structured)

2. Autoregressive

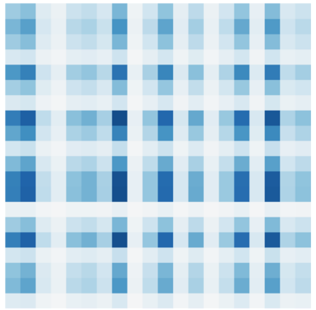
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

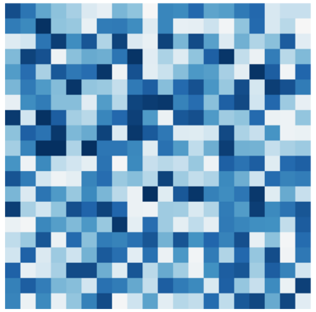
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD



(Arbitrary)

Figures from Ricky Chen

Architectural Taxonomy

Sparse connection

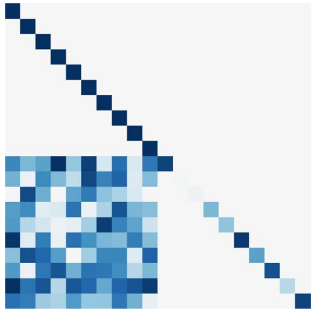
$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

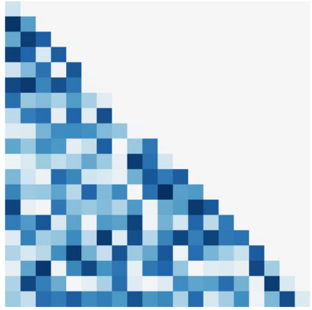
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular + structured)

2. Autoregressive

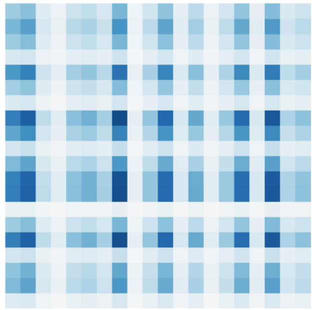
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

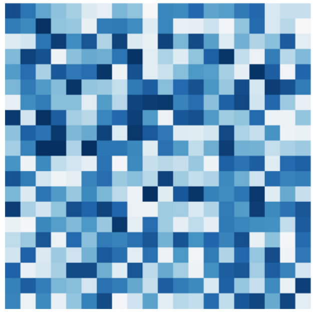
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD



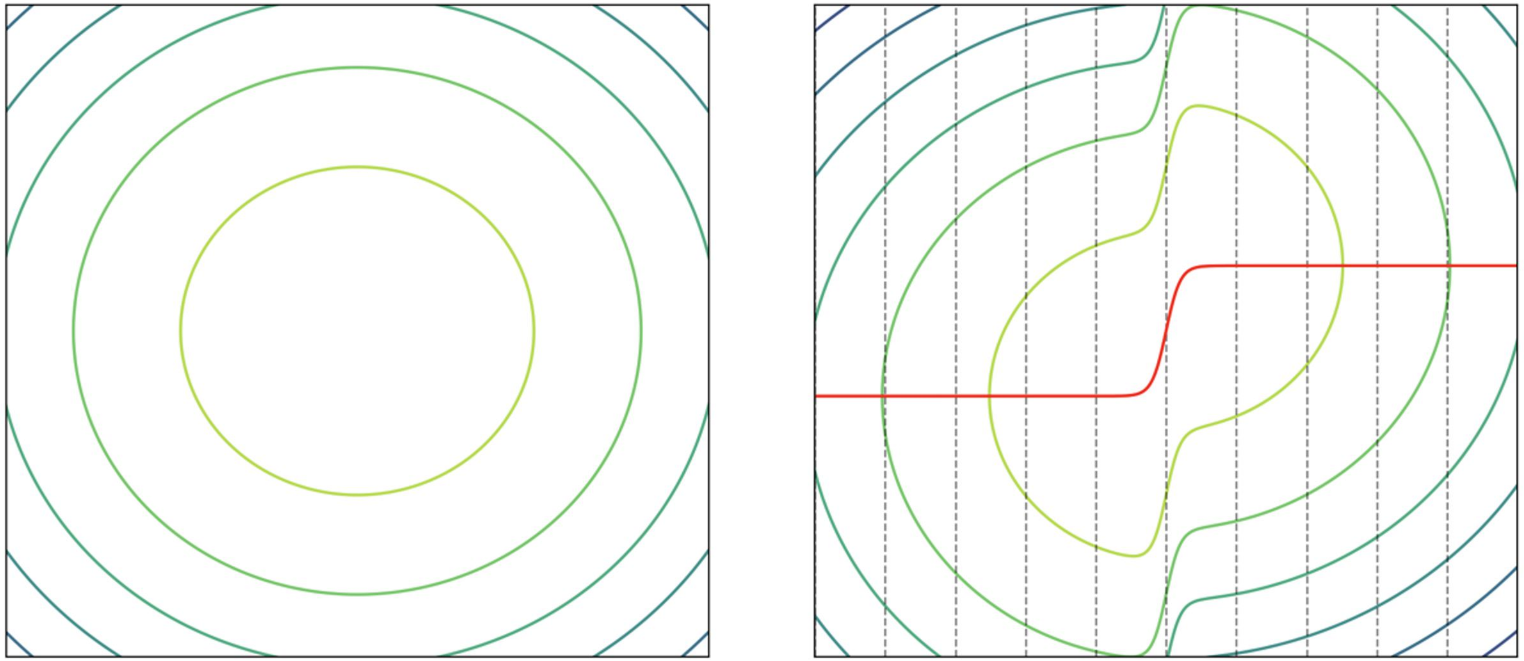
(Arbitrary)

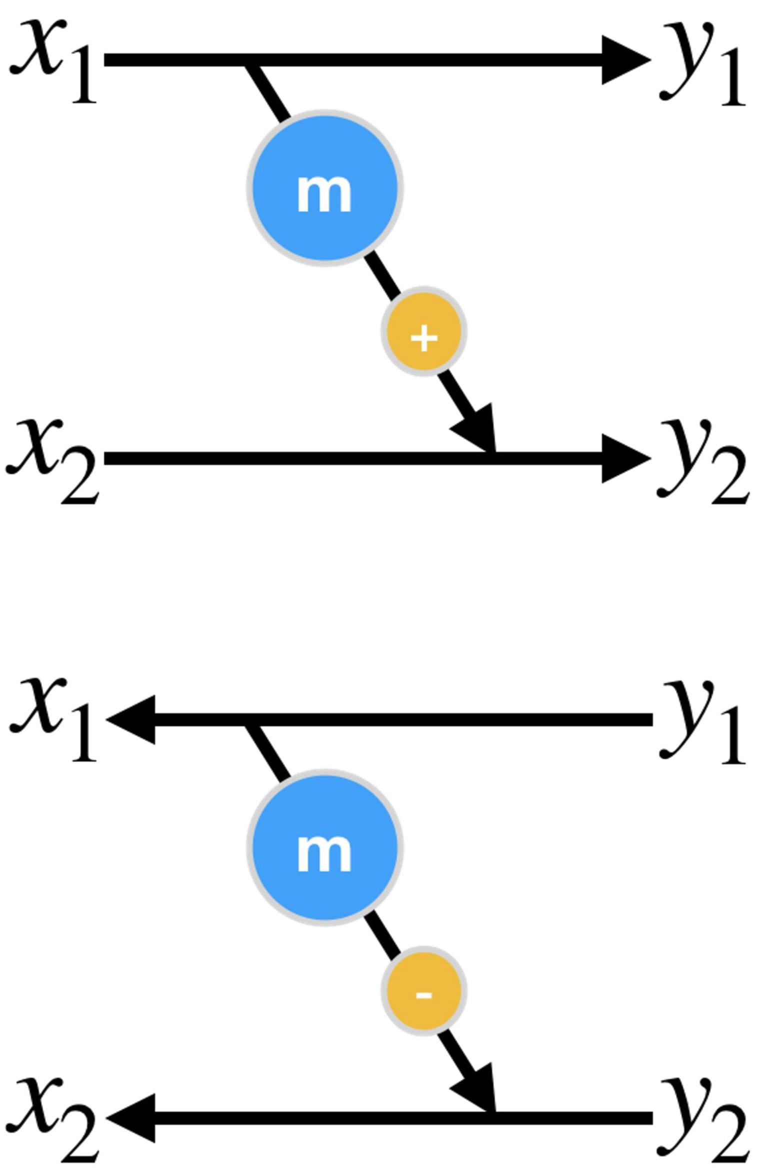
Jacobian

Figures from Ricky Chen

Coupling Law - NICE

- General form $f(\boldsymbol{x})_1 = \boldsymbol{x}_1, \; f(\boldsymbol{x})_2 = \boldsymbol{x}_2 + \mathcal{F}(\boldsymbol{x}_1)$
- Invertibility no constraint
- Jacobian determinant =1 (volume preserving)

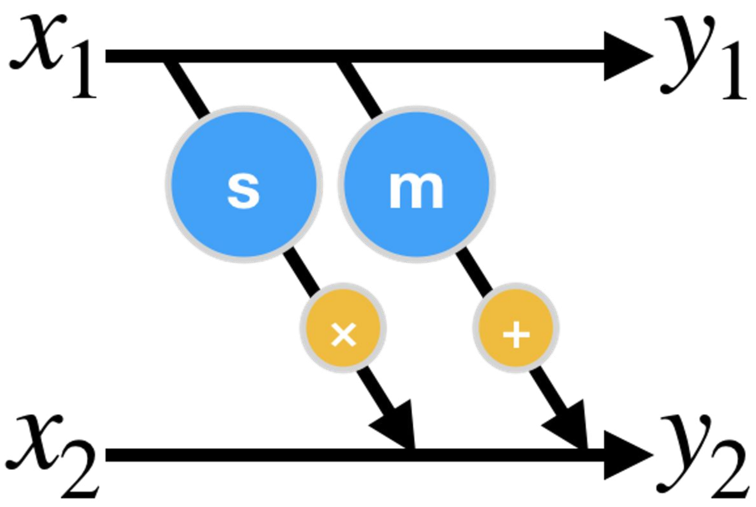




Dinh et al. 2014

Coupling Law - RealNVP

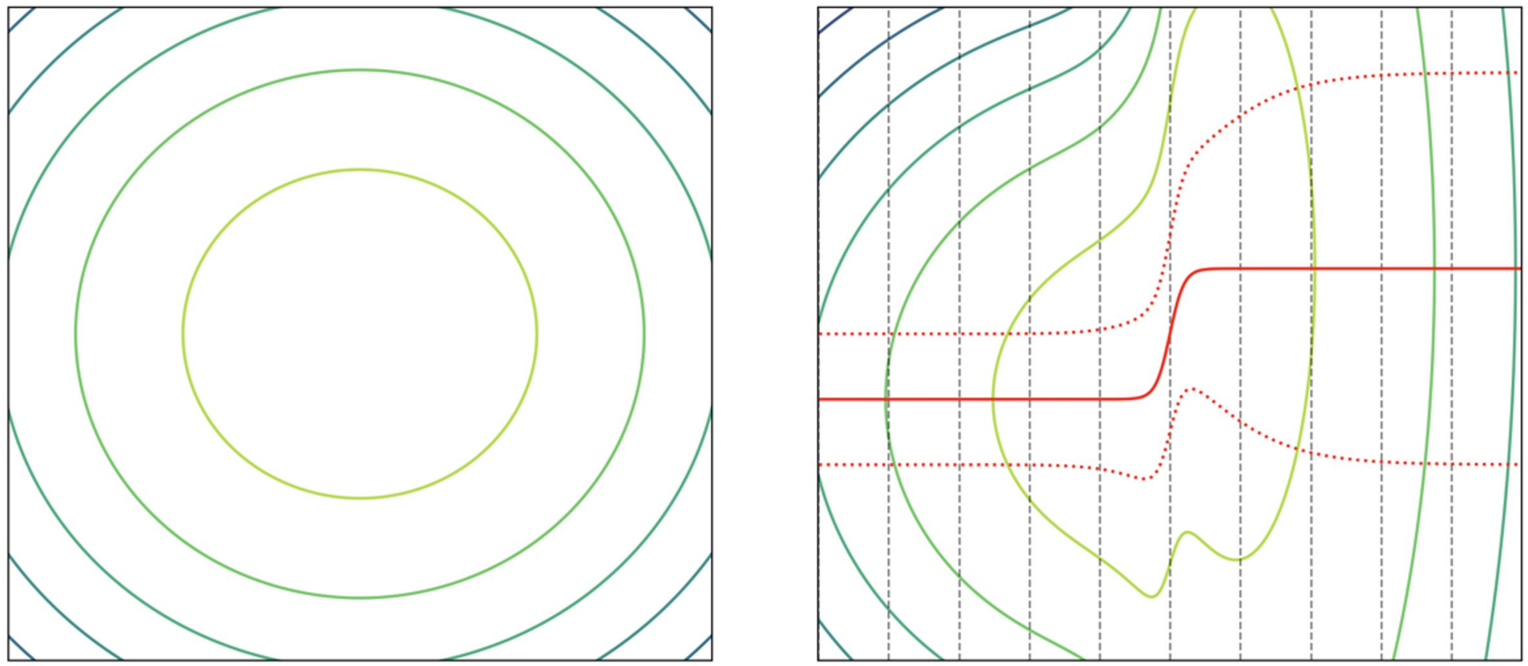
Real-valued
Non-Volume
Preserving

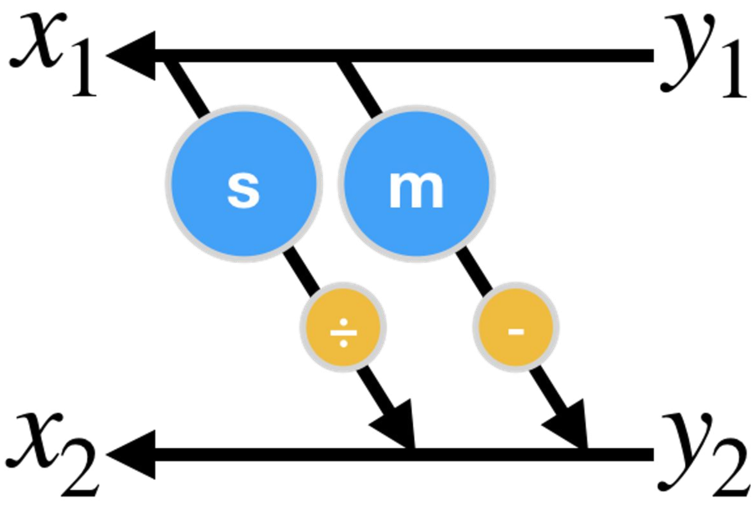


- General form

$$f(\boldsymbol{x})_1 = \boldsymbol{x}_1,$$
$$f(\boldsymbol{x})_2 = s(\boldsymbol{x}_1) \odot \boldsymbol{x}_2 + m(\boldsymbol{x}_1)$$

- Invertibility — s>0 (or simply non-zero)
- Jacobian determinant — product of s





Dinh et al. 2016

Real NVP via Masked Convolution

Partitioning can be implemented using a binary mask b, and using the functional form for y

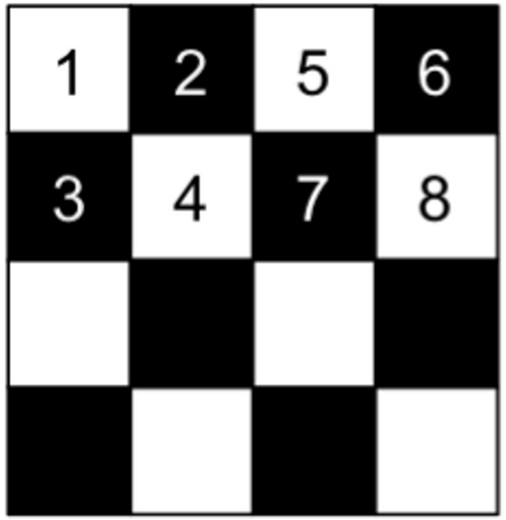
$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s_{-}(b \odot x)) + m(b \odot x))$$

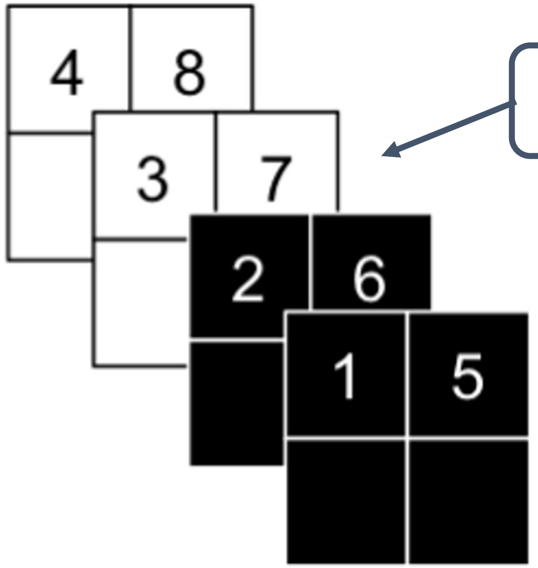
Real NVP via Masked Convolution

Partitioning can be implemented using a binary mask b, and using the functional form for y

$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s\_(b \odot x)) + m(b \odot x))$$

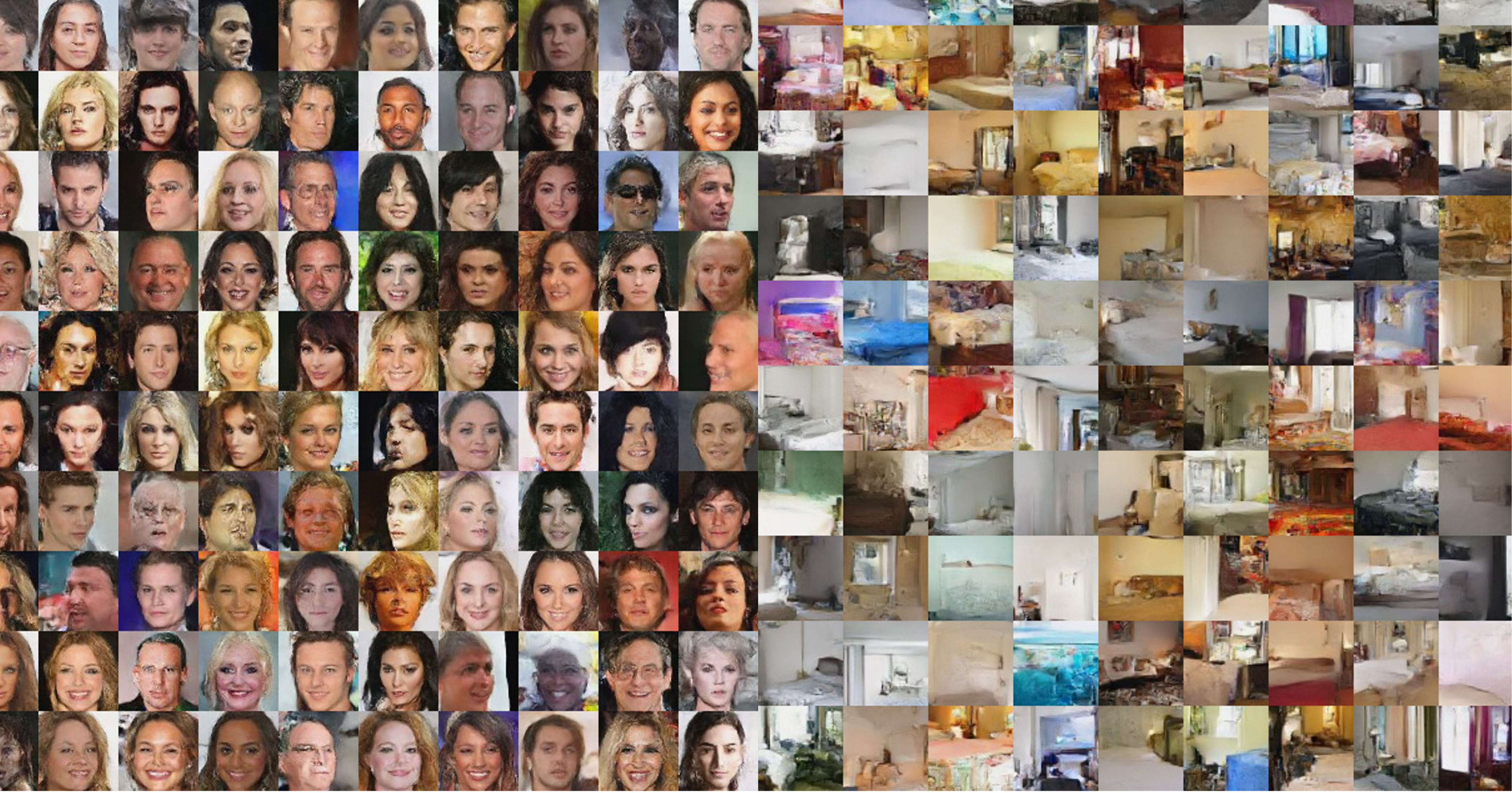
The **spatial checkerboard pattern mask** has value 1 where the sum of spatial coordinates is odd, and 0 otherwise.





The channel-wise mask b is 1 for the first half of the channel dimensions and 0 for the second half.

Figures from Density Estimation Using Real NVP by Dinh et al., 2017

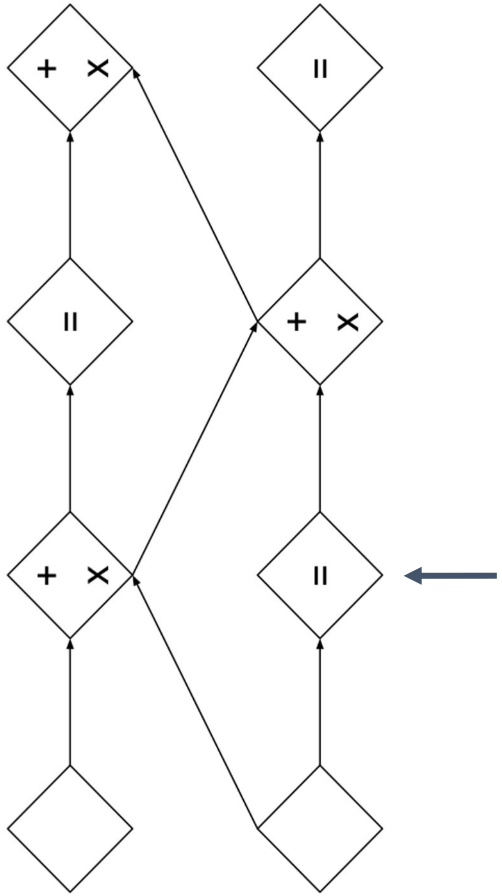


Celeba-64 (left) and LSUN bedroom (right)

Figures from Density Estimation Using Real NVP by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)

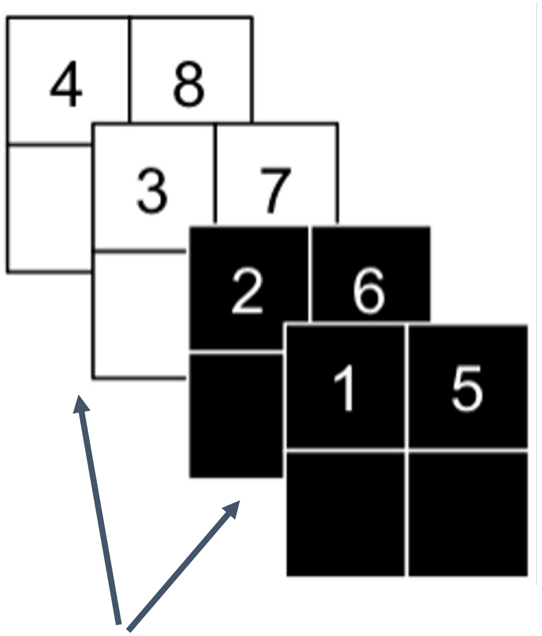


Unchanged in the first transform

Figure from Density Estimation Using Real NVP by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)

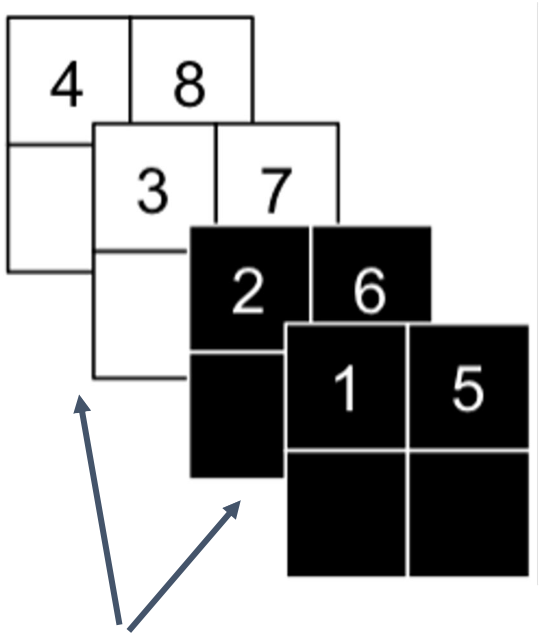


Alternating masks

Figure from Density Estimation Using Real NVP by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)



Alternating masks

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix}$$

Replace with a general invertible matrix W

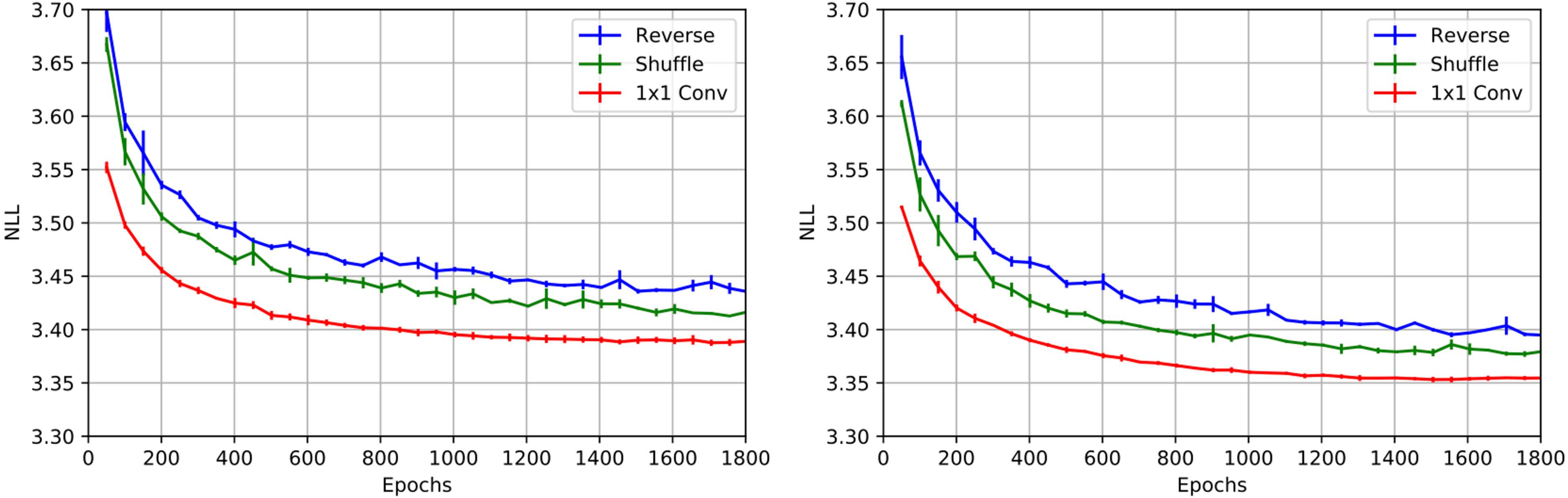
Represent W as a 1x1 convolutional kernel of shape [c, c, 1, 1]; c being # channels

$$\log\left|\det\left(\frac{\partial\,\mathrm{conv2D}(h;W)}{\partial h}\right)\right| = h \cdot w \cdot \log|\det(W)|$$

Figure from Density Estimation Using Real NVP by Dinh et al., 2017

Ablation: Permutation vs 1x1 Convolution

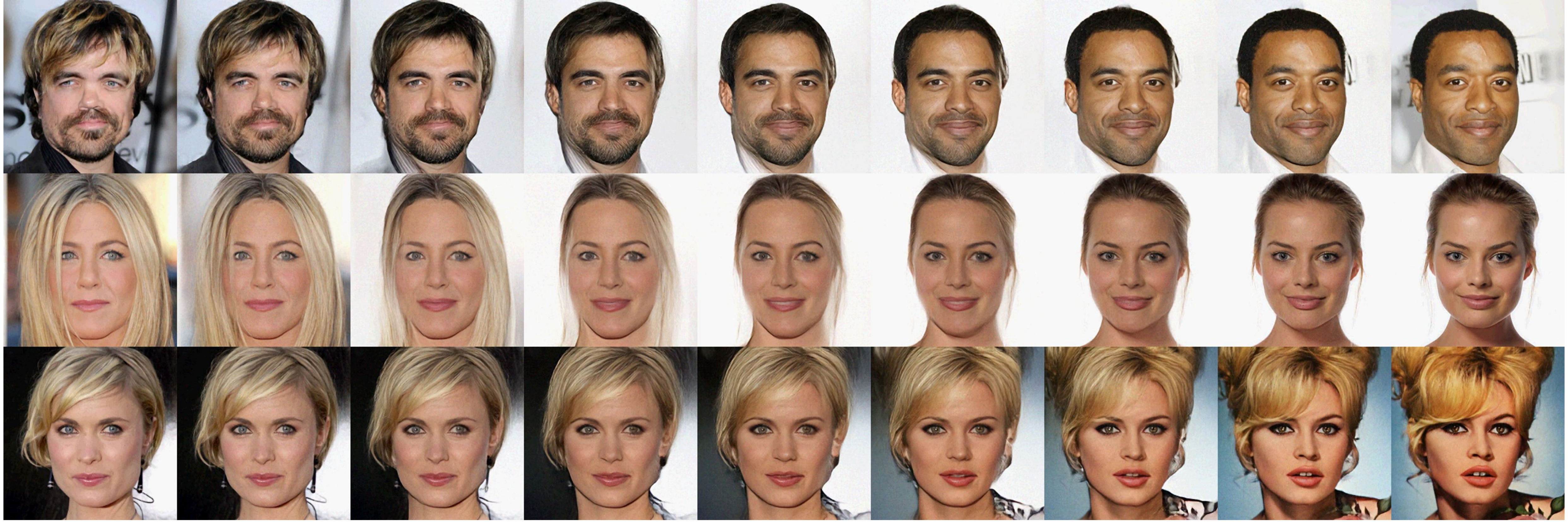
Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**



Bits-per-dim on CIFAR: left: additive, right: affine

Results from Glow: Generative Flow with Invertible 1×1 Convolutions by Kingma and Dhariwal, 2018

Figure from Glow: Generative Flow with Invertible 1×1 Convolutions by Kingma and Dhariwal, 2018



Interpolation with Generative Flows

Figure from *Glow: Generative Flow with Invertible 1×1 Convolutions* by Kingma and Dhariwal, 2018
Video from Durk Kingma's youtube channel

Architectural Taxonomy

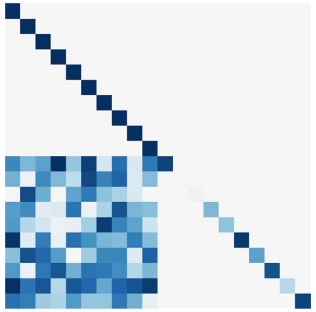
Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

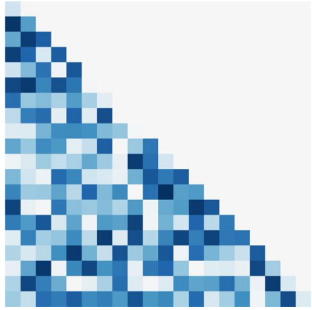
Jacobian



(Lower triangular + structured)

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN



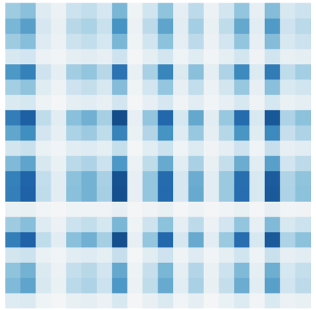
(Lower triangular)

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

3. Det identity

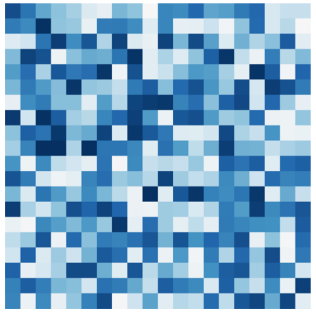
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD

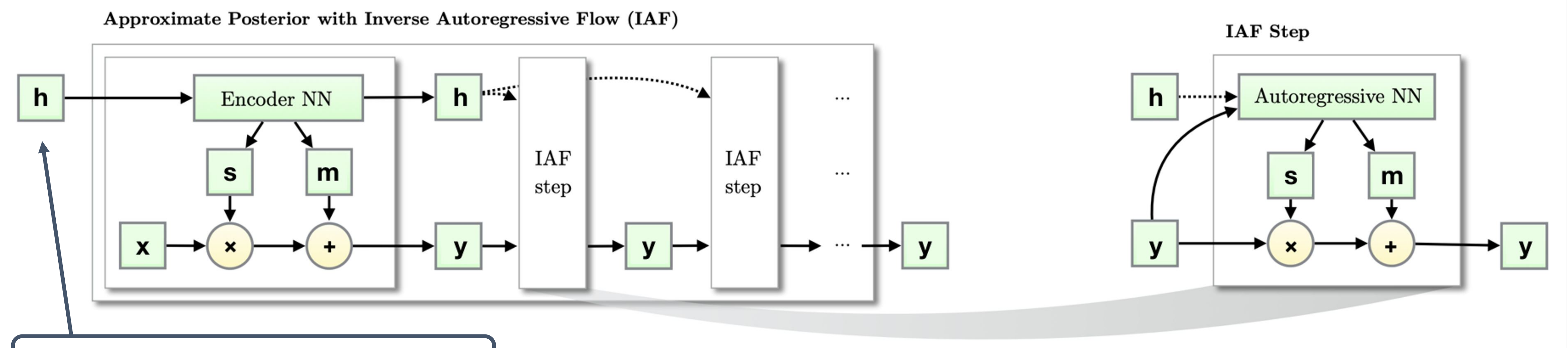


(Arbitrary)

Figures from Ricky Chen

Inverse (Affine) Autoregressive Flows

- General form $f(\boldsymbol{x})_t = s(\boldsymbol{x}_{<t}) \cdot \boldsymbol{x}_t + m(\boldsymbol{x}_{<t})$
- Invertibility s>0 (or simply non-zero)
- Jacobian determinant product of s



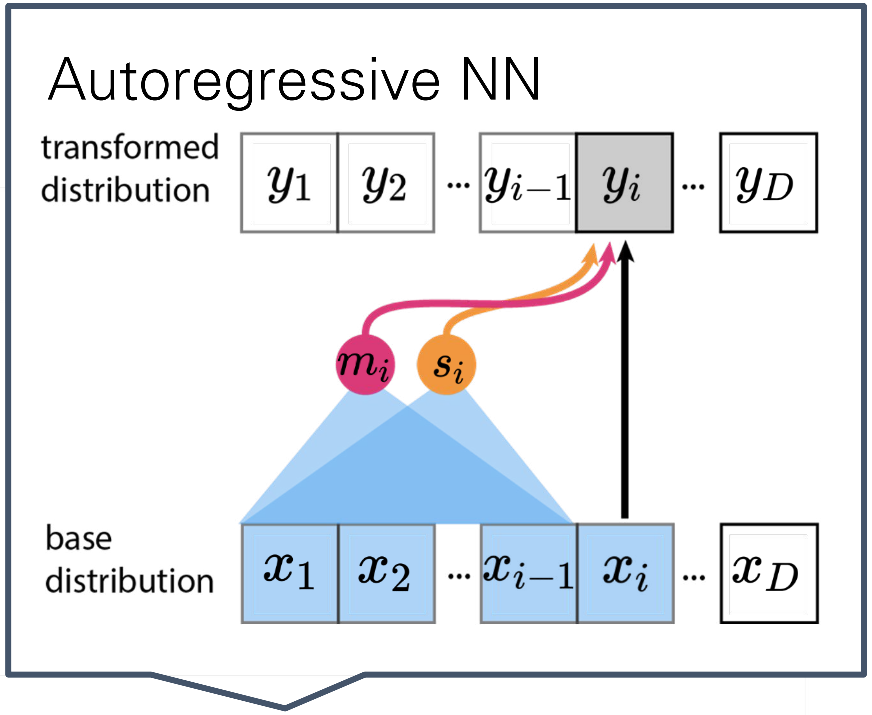
Context vector for conditioning

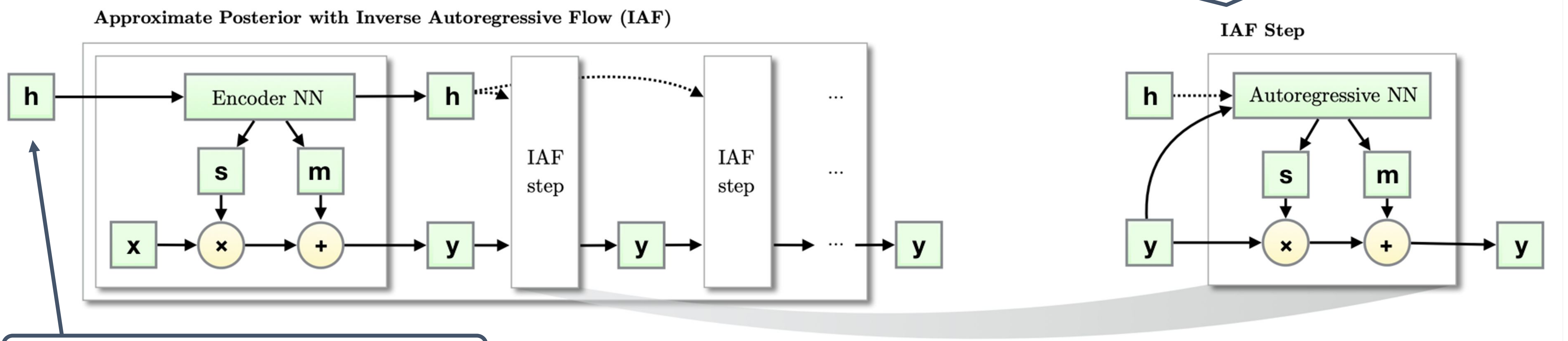
Modified from Kingma et al. 2016

Inverse Autoregressive Flows

Modified from Jang (blog post) 2018

- General form $f(\boldsymbol{x})_t = s(\boldsymbol{x}_{<t}) \cdot \boldsymbol{x}_t + m(\boldsymbol{x}_{<t})$
- Invertibility s>0 (or simply non-zero)
- Jacobian determinant product of s





Context vector for conditioning

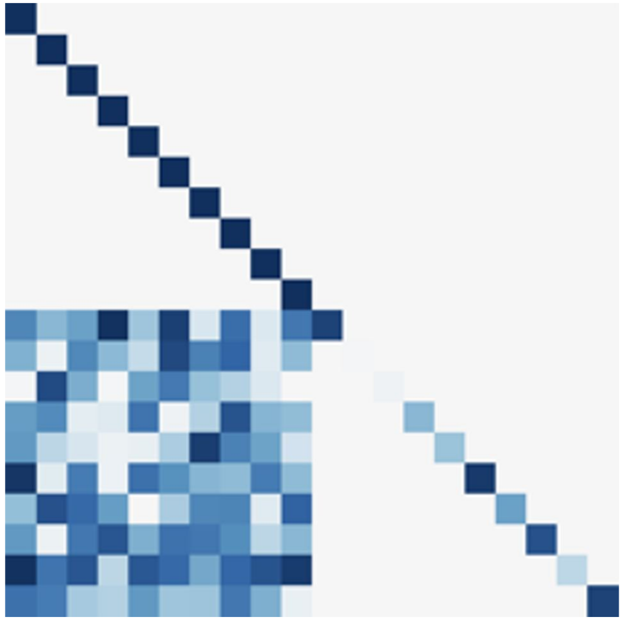
Modified from Kingma et al. 2016

Trade-off between Expressivity and Inversion Cost

Block autoregressive

- Limited capacity
- Inverse takes constant time

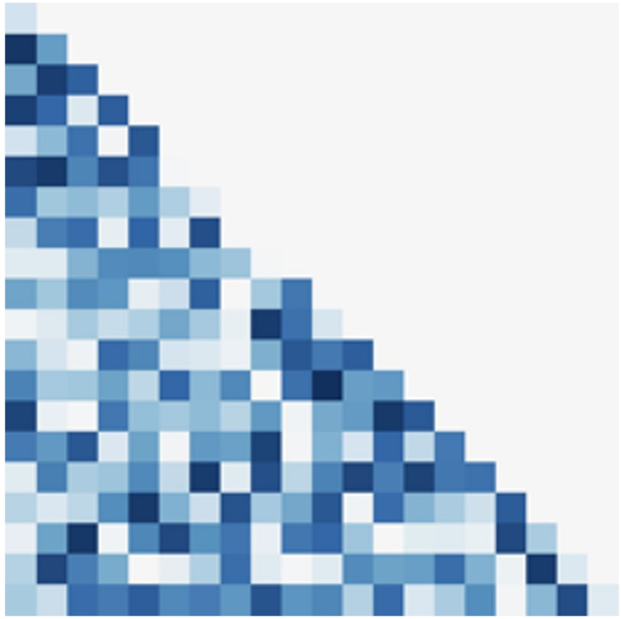
Jacobian



(Block triangular)

Autoregressive

- Higher capacity
- Inverse takes linear time (dimensionality)

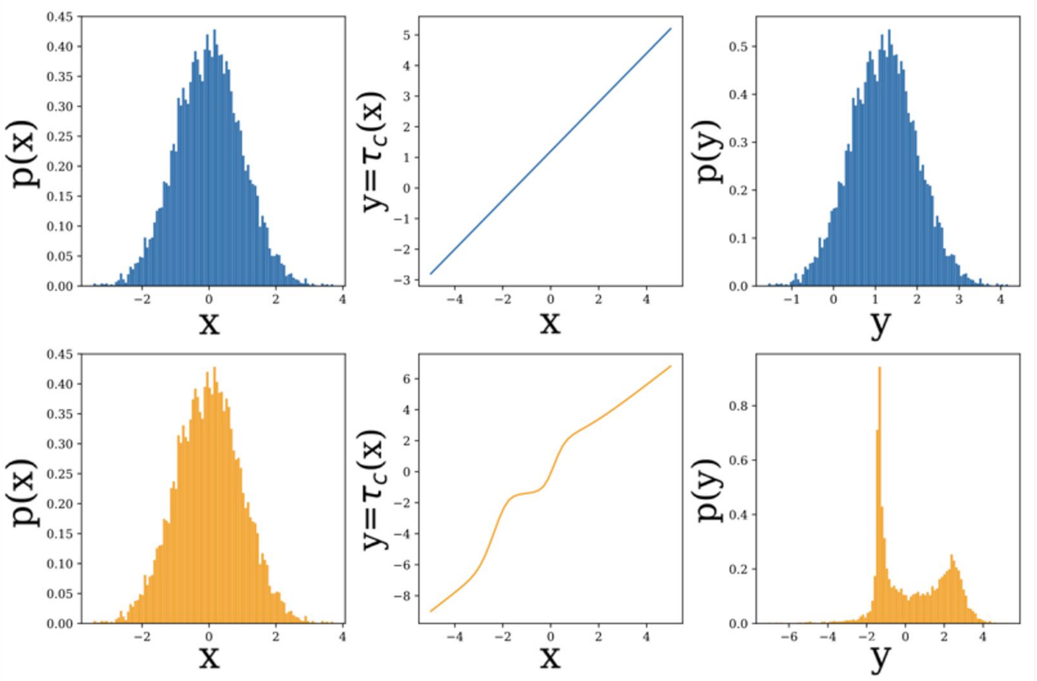


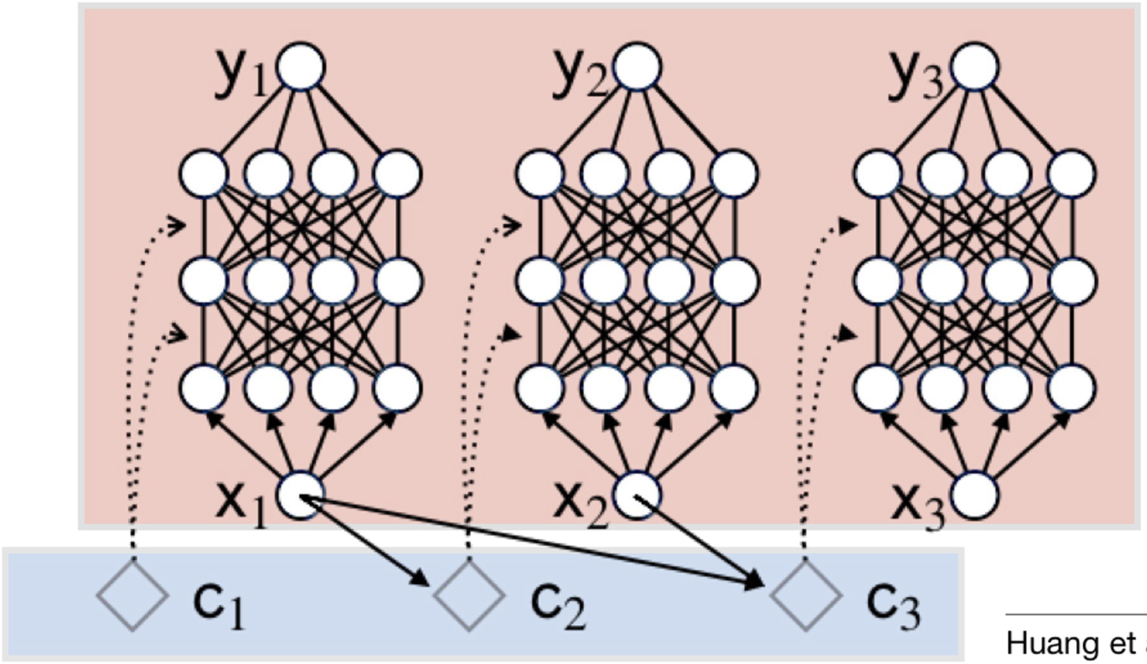
(Triangular)

Figures from Ricky Chen

Neural Autoregressive Flows

- General form $f(\boldsymbol{x})_t = \mathcal{P}(\boldsymbol{x}_t; \mathcal{H}(\boldsymbol{x}_{<t}))$
- Invertibility monotonic activation and positive weight in $\mathcal{P}$
- Jacobian determinant product of derivatives (elementwise)





Huang et al. 2018

Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

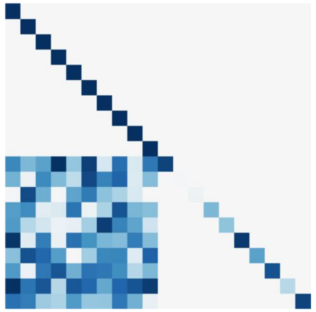
Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

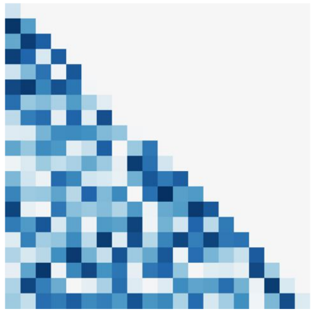
Jacobian



(Lower triangular + structured)

2. Autoregressive

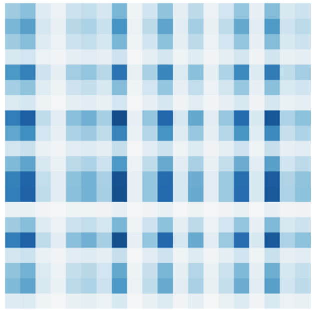
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

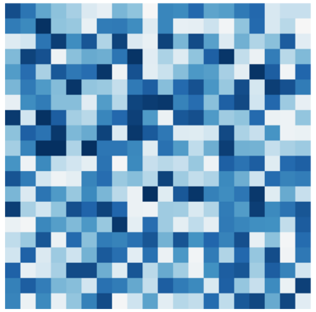
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD

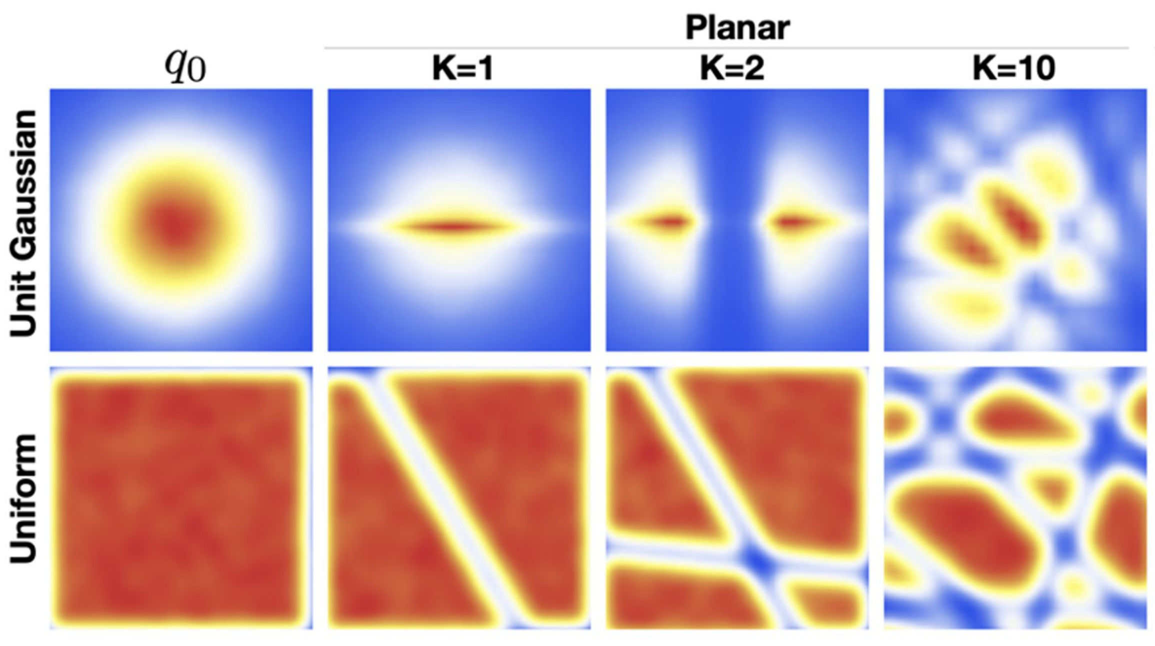


(Arbitrary)

Figures from Ricky Chen

Determinant Identity – Planar Flows

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- Invertibility $\boldsymbol{u}^\top \boldsymbol{w} > -1 \text{ if } h = \tanh$
- Jacobian determinant $\left|\det \frac{\partial f}{\partial \boldsymbol{x}}\right| = \left|\det\left(\boldsymbol{I} + h'(\boldsymbol{w}^\top \boldsymbol{x} + b)\boldsymbol{u}\boldsymbol{w}^\top\right)\right| = \left|1 + h'(\boldsymbol{w}^\top \boldsymbol{x} + b)\boldsymbol{u}^\top \boldsymbol{w}\right|$



VAE on binary MNIST

Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	≤ 89.9
DLGM+NF (k = 10)	≤ 87.5
DLGM+NF (k = 20)	≤ 86.5
DLGM+NF (k = 40)	≤ 85.7
DLGM+NF (k = 80)	≤ 85.1

Rezende et al. 2015

Determinant Identity – Sylvester Flows

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{A}h(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{b})$ $\quad \boldsymbol{A} \in \mathbb{R}^{m \times d}$, $\boldsymbol{B} \in \mathbb{R}^{d \times m}$, $\boldsymbol{b} \in \mathbb{R}^{d}$, and $d \leq m$
- Invertibility Similar to planar flows
- Jacobian determinant Using Sylvester's Thm: $\det(\boldsymbol{I}_m + \boldsymbol{A}\boldsymbol{B}) = \det(\boldsymbol{I}_d + \boldsymbol{B}\boldsymbol{A})$

Model	Freyfaces		Omniglot		Caltech 101	
	-ELBO	NLL	-ELBO	NLL	-ELBO	NLL
VAE	4.53 ± 0.02	4.40 ± 0.03	104.28 ± 0.39	97.25 ± 0.23	110.80 ± 0.46	99.62 ± 0.74
Planar	$\mathbf{4.40 \pm 0.06}$	$\mathbf{4.31 \pm 0.06}$	102.65 ± 0.42	96.04 ± 0.28	109.66 ± 0.42	98.53 ± 0.68
IAF	4.47 ± 0.05	4.38 ± 0.04	102.41 ± 0.04	96.08 ± 0.16	111.58 ± 0.38	99.92 ± 0.30
O-SNF	4.51 ± 0.04	4.39 ± 0.05	99.00 ± 0.29	93.82 ± 0.21	106.08 ± 0.39	94.61 ± 0.83
H-SNF	4.46 ± 0.05	4.35 ± 0.05	$\mathbf{99.00 \pm 0.04}$	$\mathbf{93.77 \pm 0.03}$	$\mathbf{104.62 \pm 0.29}$	$\mathbf{93.82 \pm 0.62}$
T-SNF	4.45 ± 0.04	4.35 ± 0.04	99.33 ± 0.23	93.97 ± 0.13	105.29 ± 0.64	94.92 ± 0.73

Van den Berg et al. 2018

Architectural Taxonomy

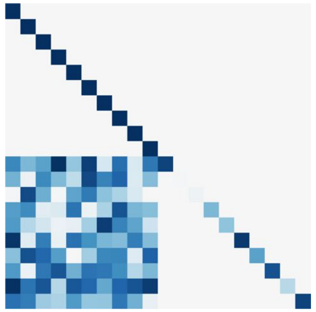
Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow

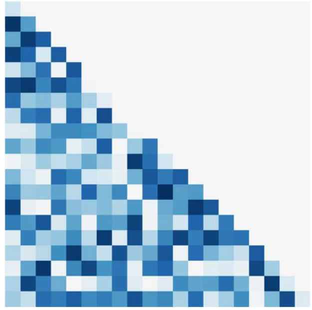
Jacobian



(Lower triangular + structured)

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN



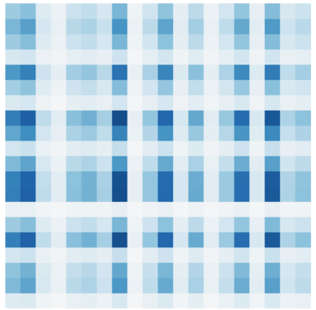
(Lower triangular)

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

3. Det identity

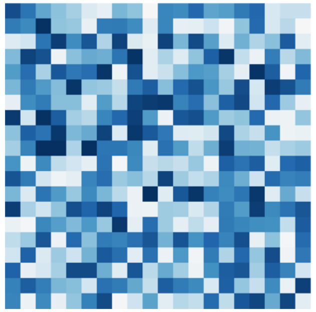
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD



(Arbitrary)

Figures from Ricky Chen

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \mathrm{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \mathrm{tr}\left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\mathrm{tr}\left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \mathrm{tr}\left(\frac{\partial g(x)}{\partial x}^{k} \right)$$

Power series expansion

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr} \left(\frac{\partial g(x)}{\partial x}^k \right)$$

Power series expansion

$$\approx \mathbb{E}_v \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} v^\top \left(\frac{\partial g(x)}{\partial x}^k \right) v \right]$$

Truncation & Hutchinson trace estimator

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log\left|\det \frac{\partial f(x)}{\partial x}\right| = \operatorname{tr}\left(\log \frac{\partial f(x)}{\partial x}\right)$$

Jacobi's formula

$$\operatorname{tr}\left(\log\left(\mathbf{I} + \frac{\partial g(x)}{\partial x}\right)\right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr}\left(\frac{\partial g(x)}{\partial x}^k\right)$$

Power series expansion

$$\approx \mathbb{E}_v\left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} v^\top \left(\frac{\partial g(x)}{\partial x}^k\right) v\right]$$

Truncation &
Hutchinson trace estimator

Bias

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr}\left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr}\left(\log\left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr}\left(\frac{\partial g(x)}{\partial x}^{k} \right)$$

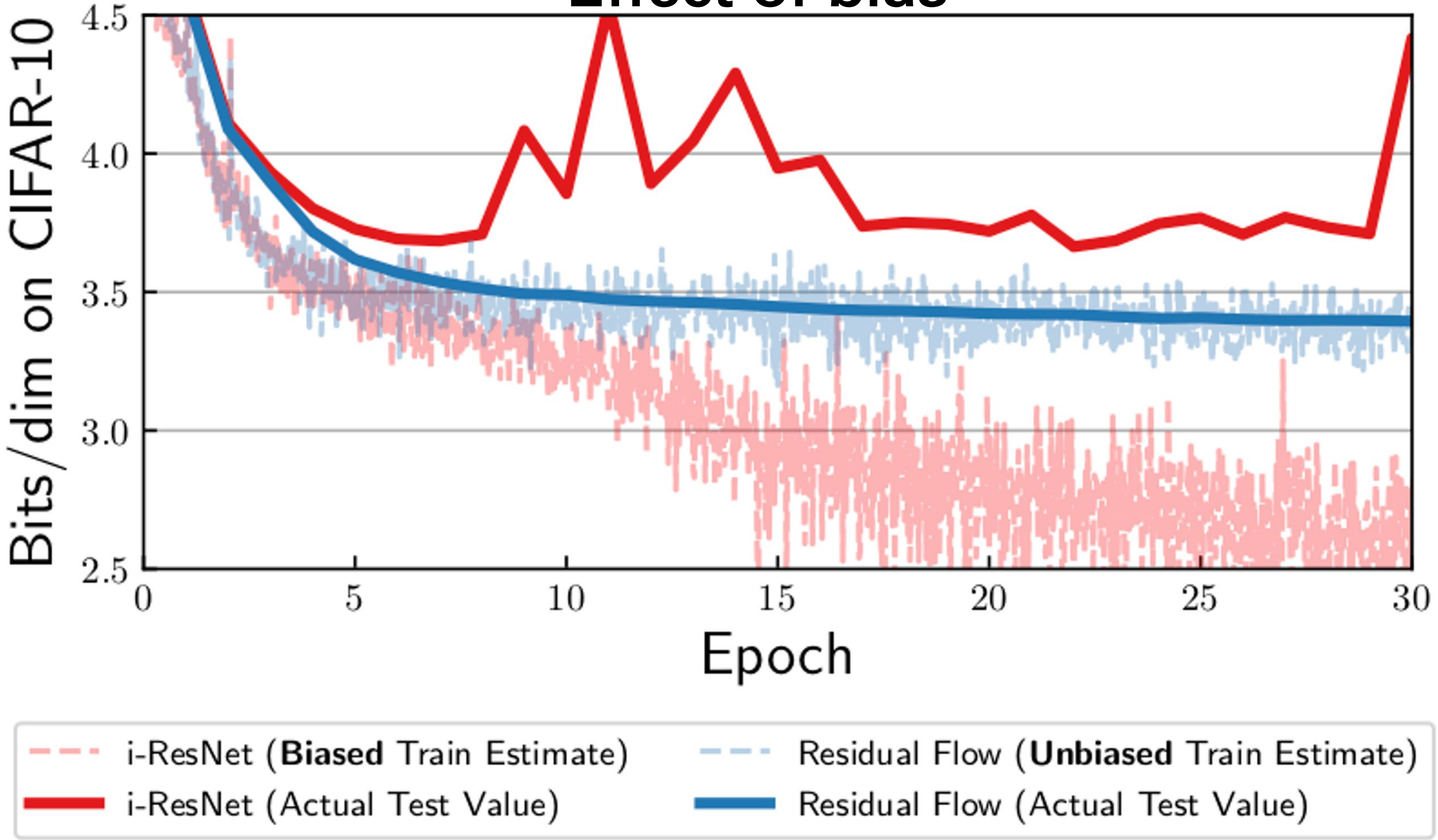
Power series expansion

$$= \mathbb{E}_{v,n}\left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k \cdot \mathbb{P}(N \geq k)} v^{\top} \left(\frac{\partial g(x)}{\partial x}^{k} \right) v \right]$$

Russian roulette estimator & Hutchinson trace estimator

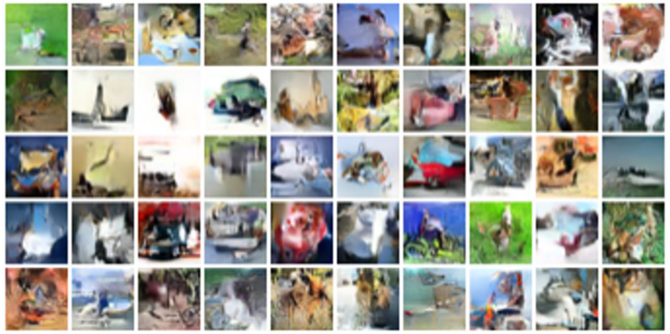
Behrmann et al. 2018

Effect of bias

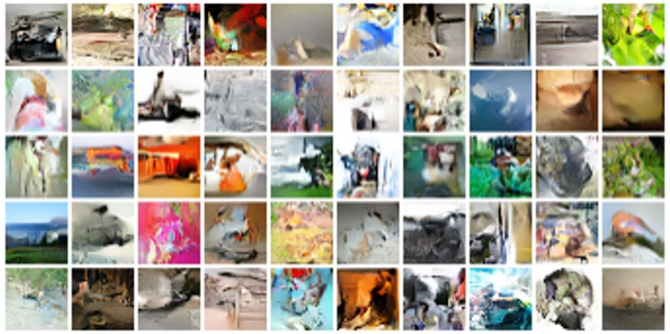


CelebA samples

Cifar10 samples



Imagenet-32 samples



Figures from Residual Flows for Invertible Generative Modeling by Chen et al., 2019

Next lecture:
Variational Autoencoders