

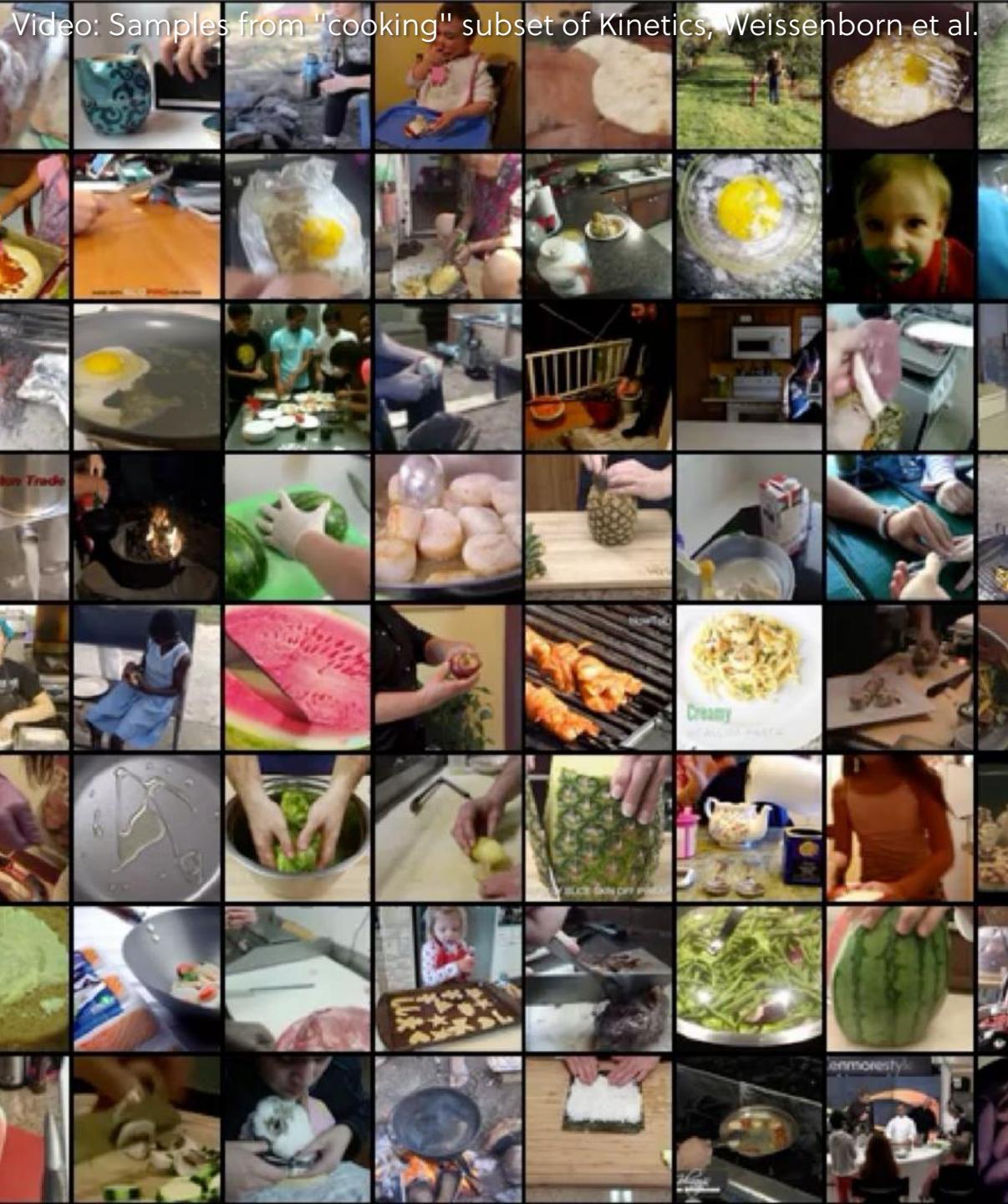
CMP784

DEEP LEARNING

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

Previously on CMP784

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



Lecture overview

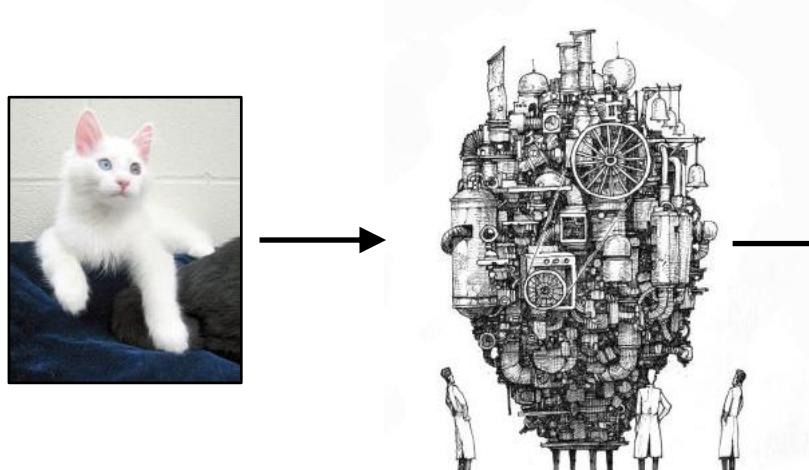
- Generative Adversarial Networks (GANs)
- Normalizing Flow Models

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

$$p(y|x)$$



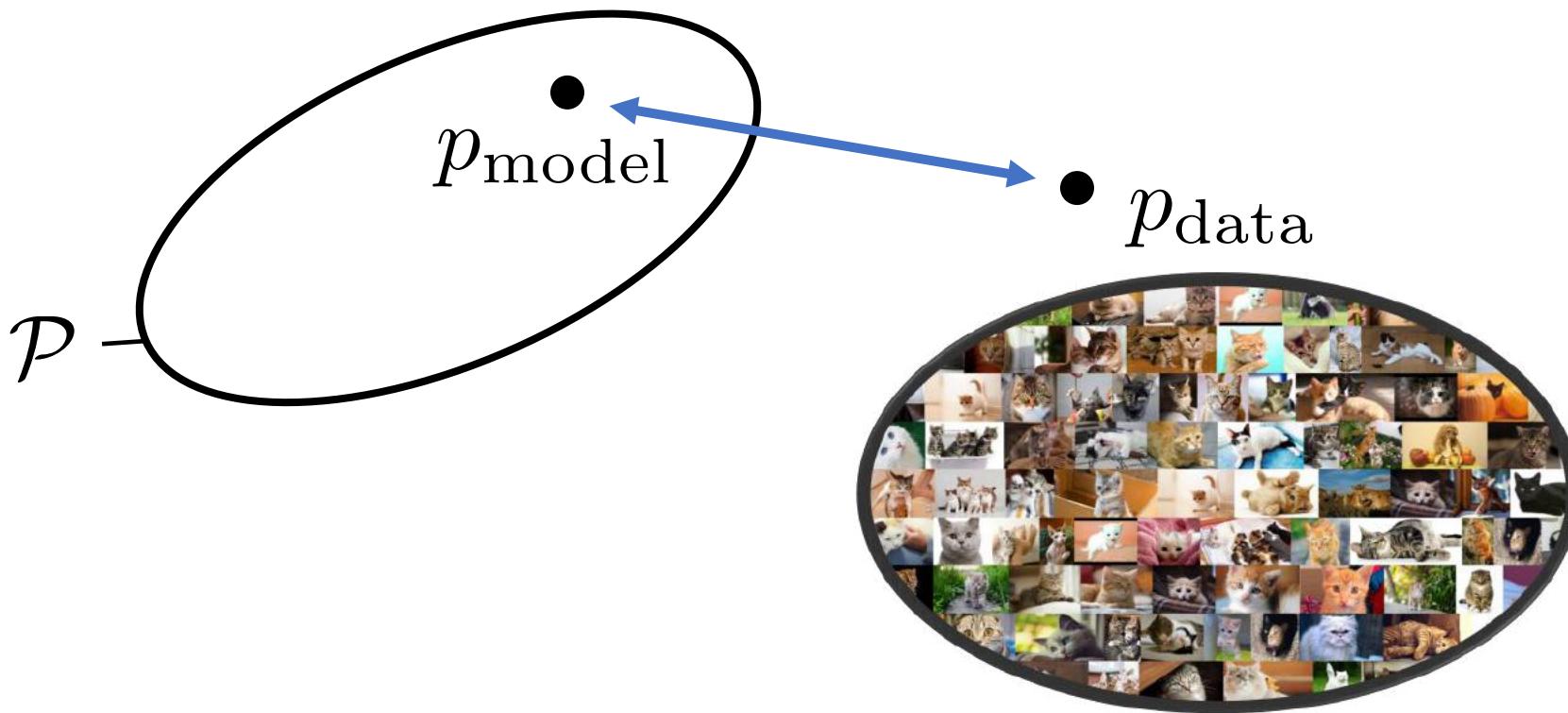
Discriminative models

$$p(x|y)$$

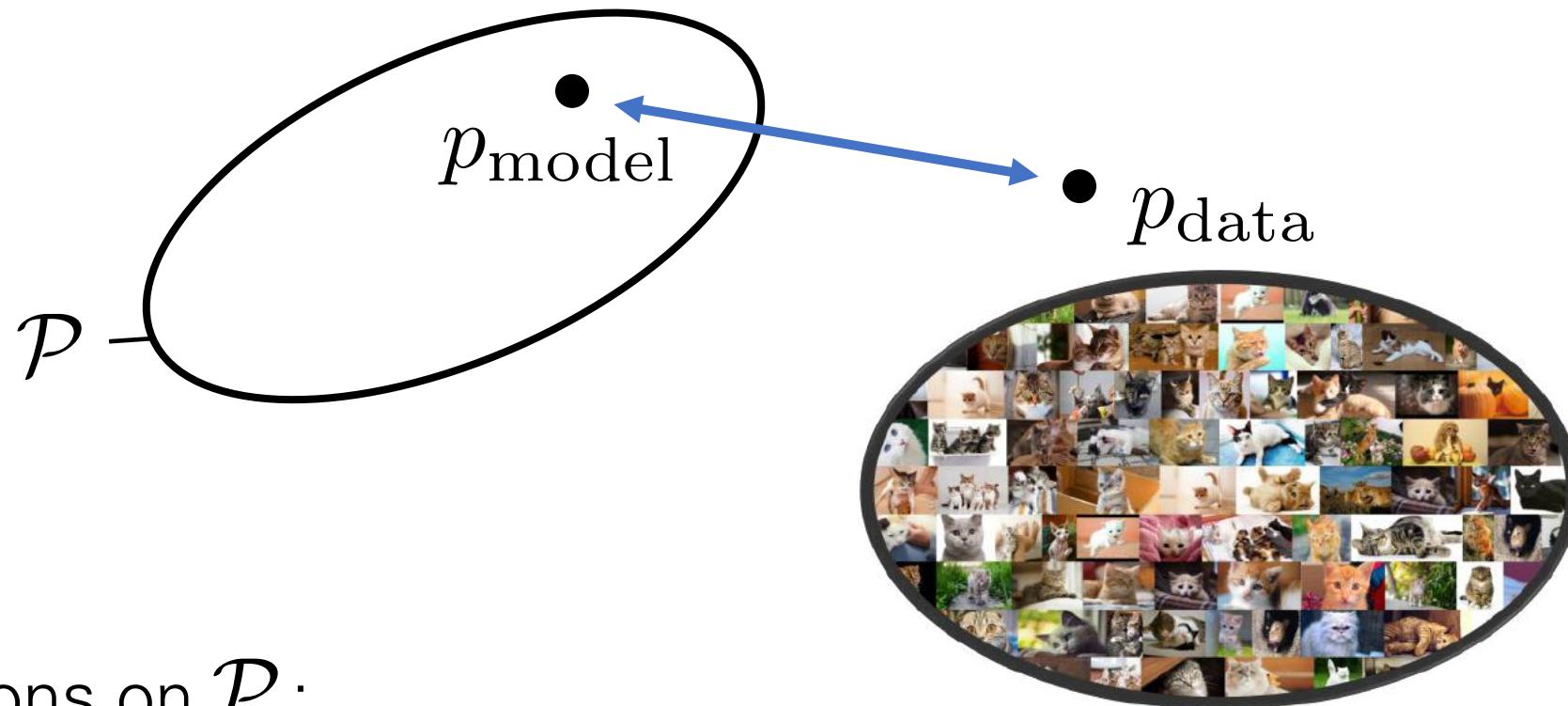


Generative models

Generative Modeling

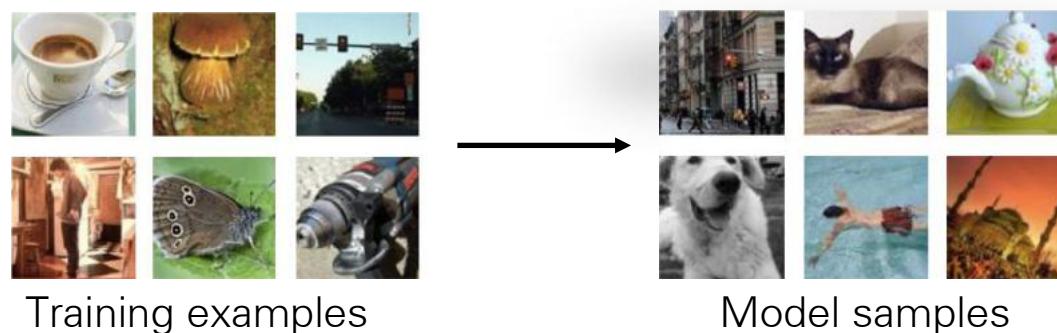


Generative Modeling

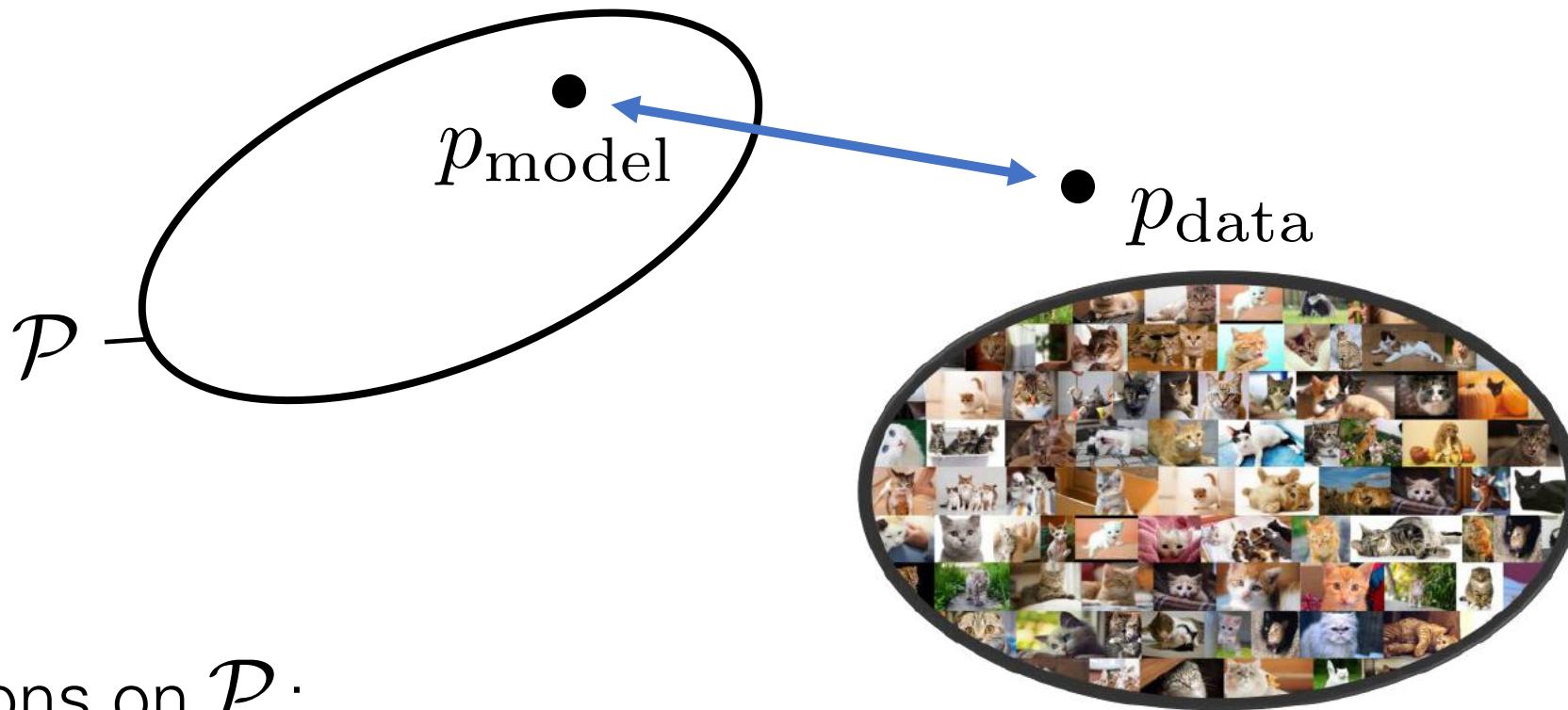


Assumptions on \mathcal{P} :

- tractable sampling

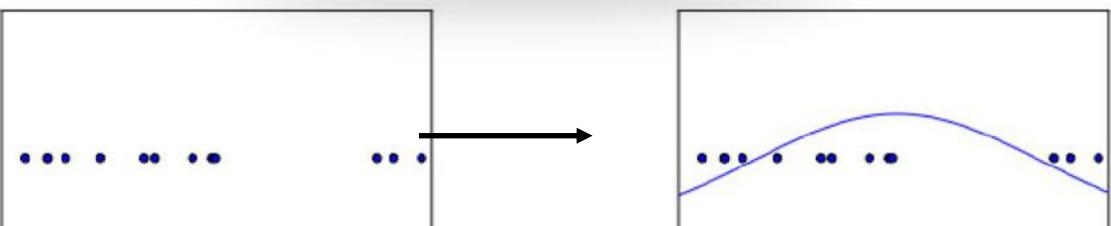


Generative Modeling



Assumptions on \mathcal{P} :

- tractable sampling
- tractable likelihood function



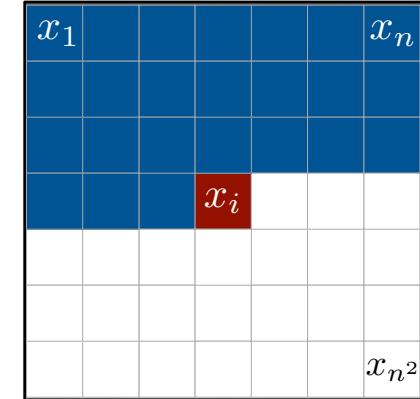
Broad Categories of Generative Models

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

Autoregressive Models

- Explicitly model conditional probabilities:

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



Each conditional can be
a complicated neural net

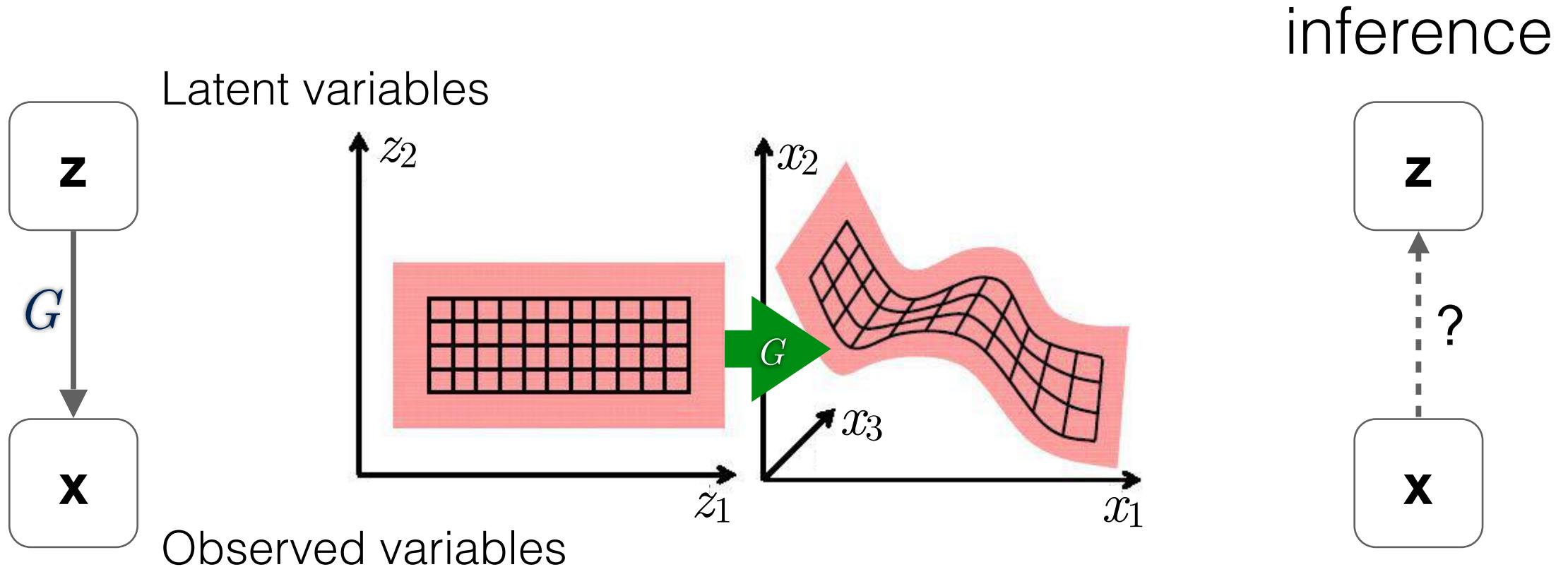
Disadvantages:

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



PixelCNN elephants
(van den Ord et al. 2016)

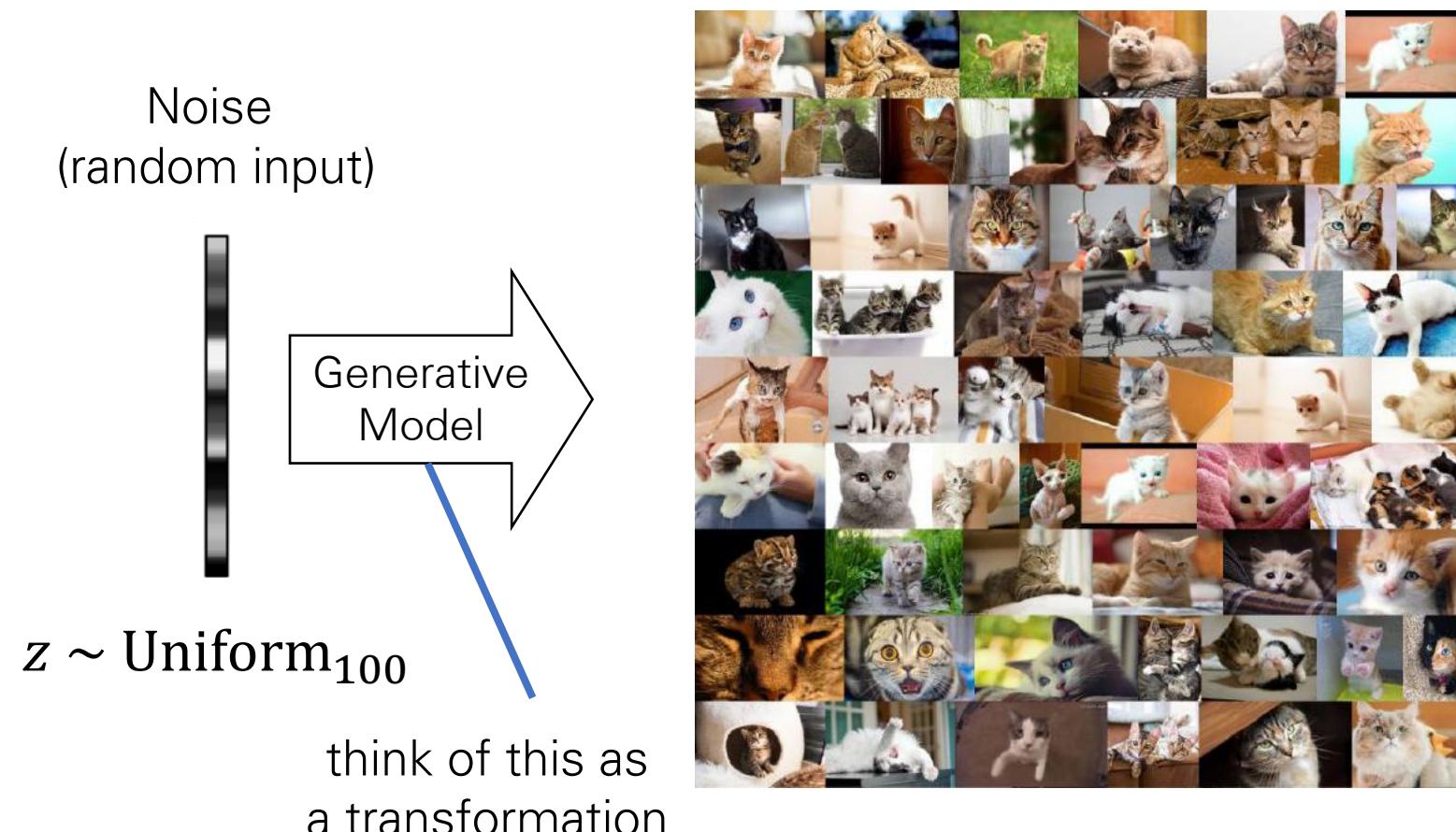
Another way to train a latent variable model



Generative Adversarial Networks

Genetive Adversarial Networks (GANs)

(Goodfellow et al., 2014)



- A game-theoretic likelihood free model

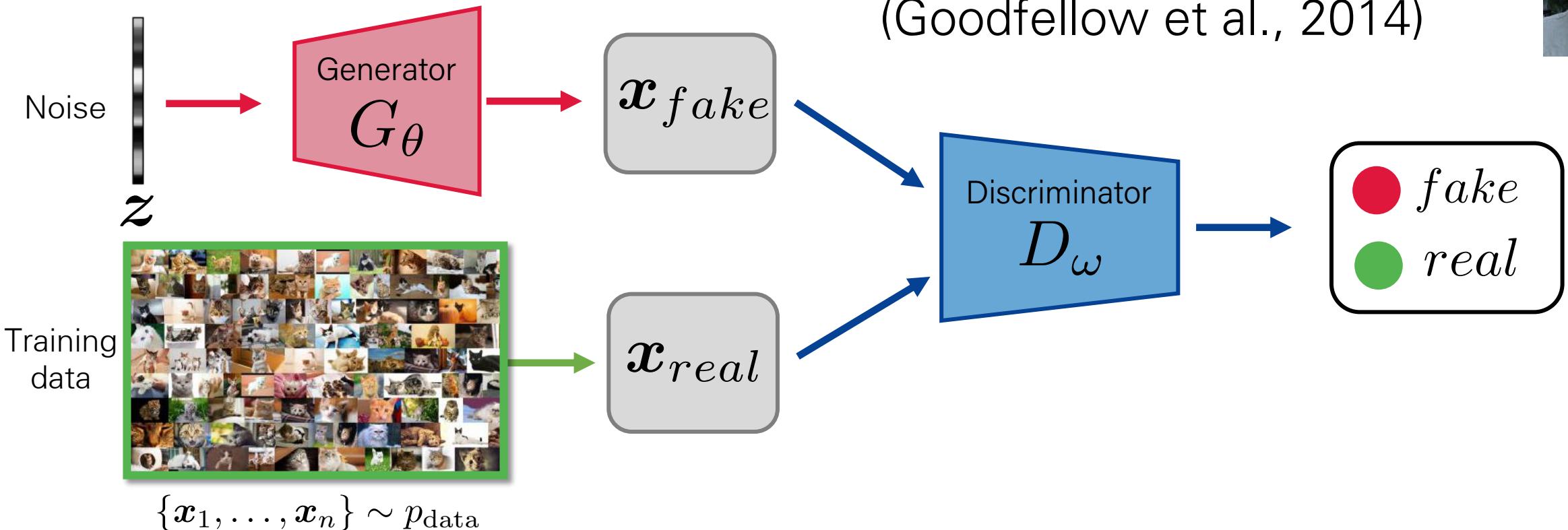
Advantages:

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

Genetive Adversarial Networks (GANs)

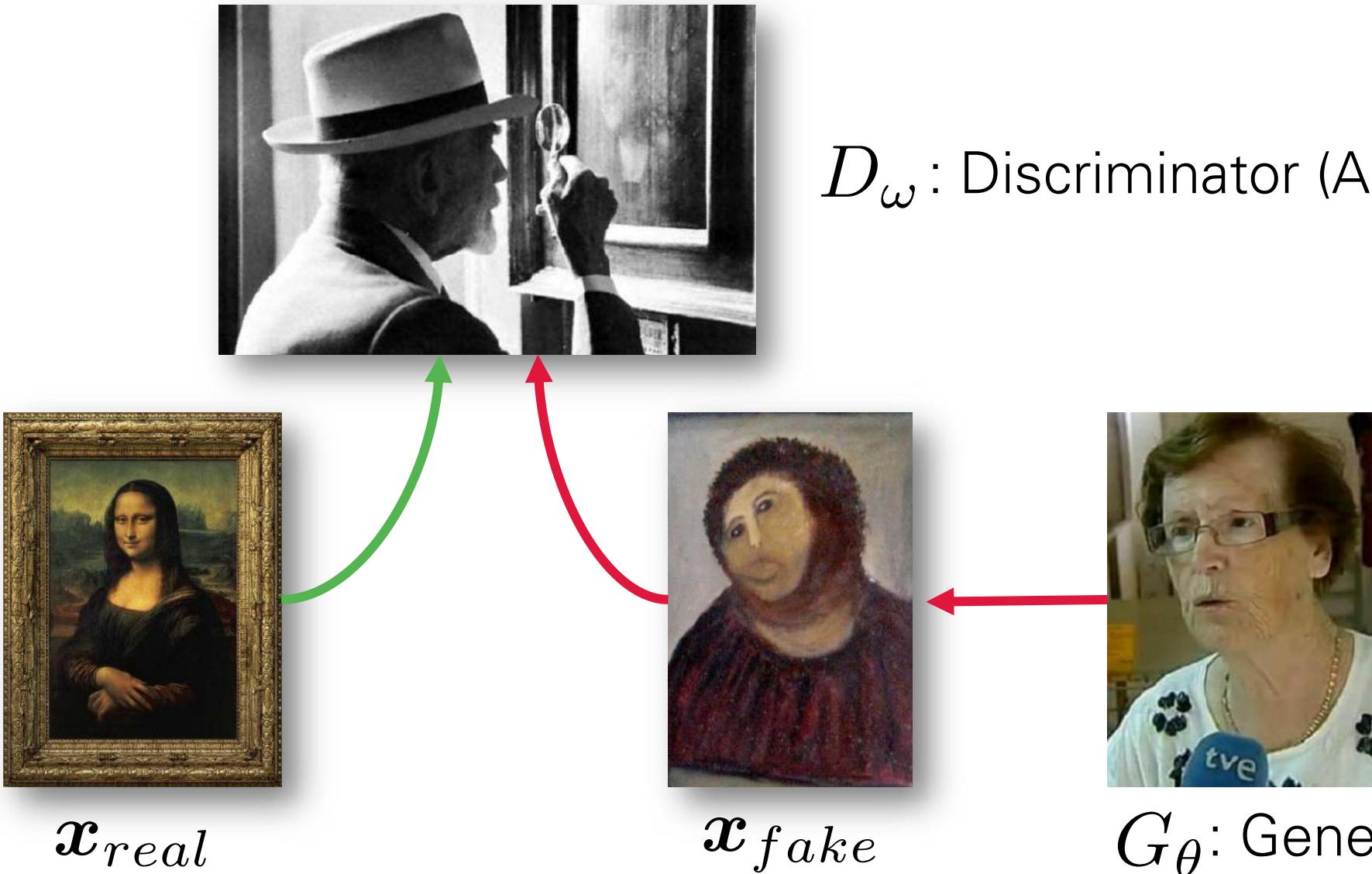


(Goodfellow et al., 2014)



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

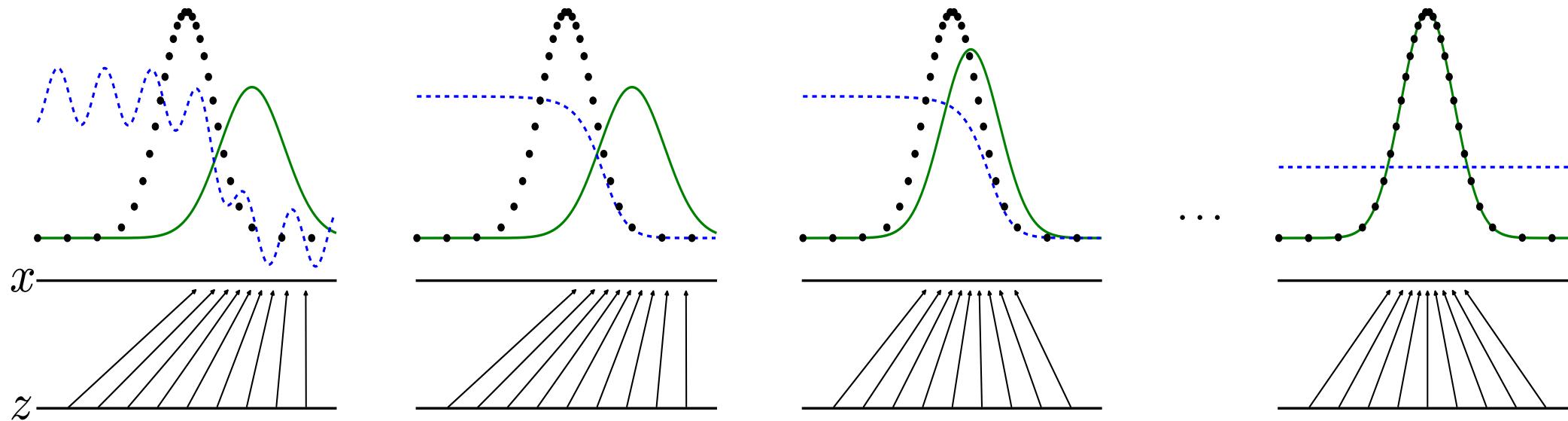
Intuition behind GANs



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)

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



Real data



Noise vector used
to generate data

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

Cross-entropy
loss for binary
classification

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

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)

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



Real data



Noise vector used

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

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_z \log D_{\omega}(G_{\theta}(z))$$

Important question is
“Does this converge??”

of the discriminator being mistaken

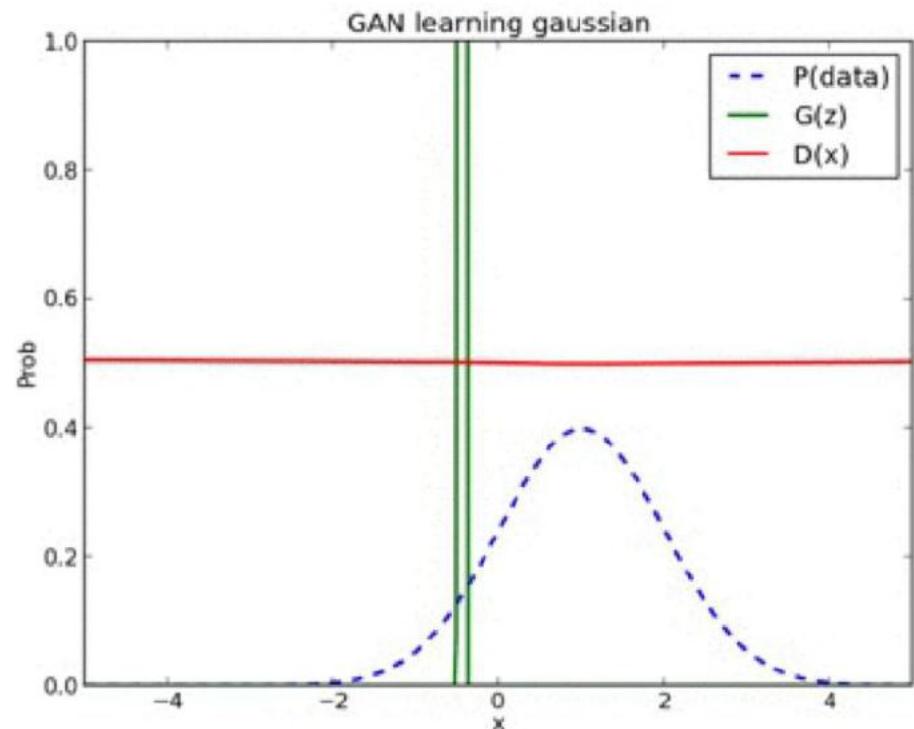
cross-entropy
loss for binary
classification

g-probability

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

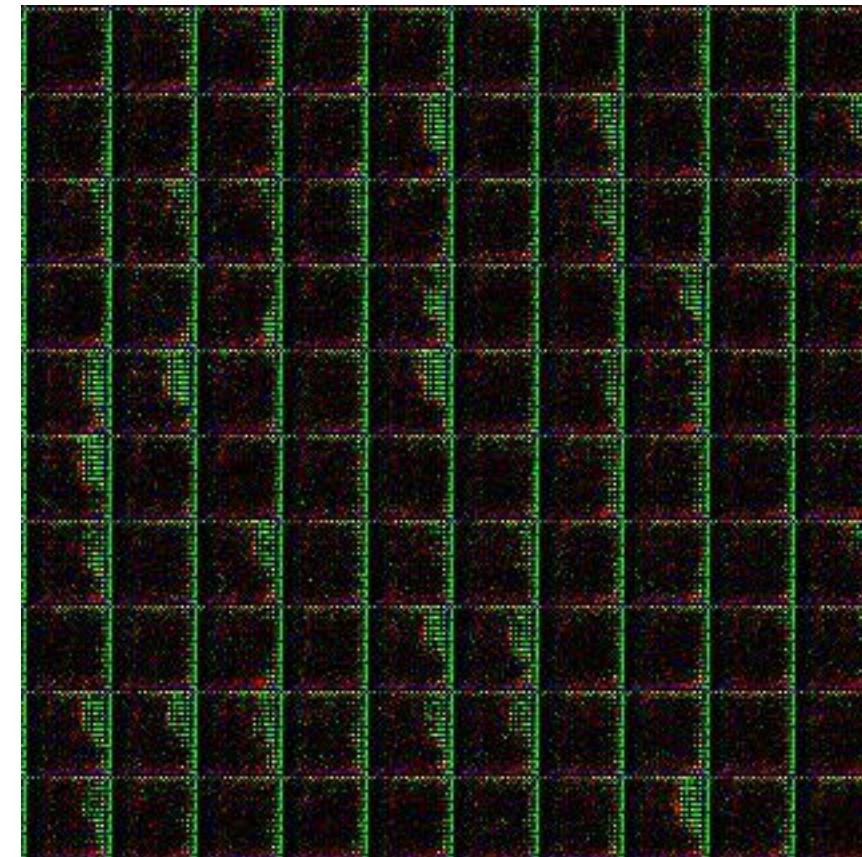
Training Procedure

(Goodfellow et al., 2014)



Source: Alec Radford

Generating 1D points



Source: OpenAI blog

Generating images

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)



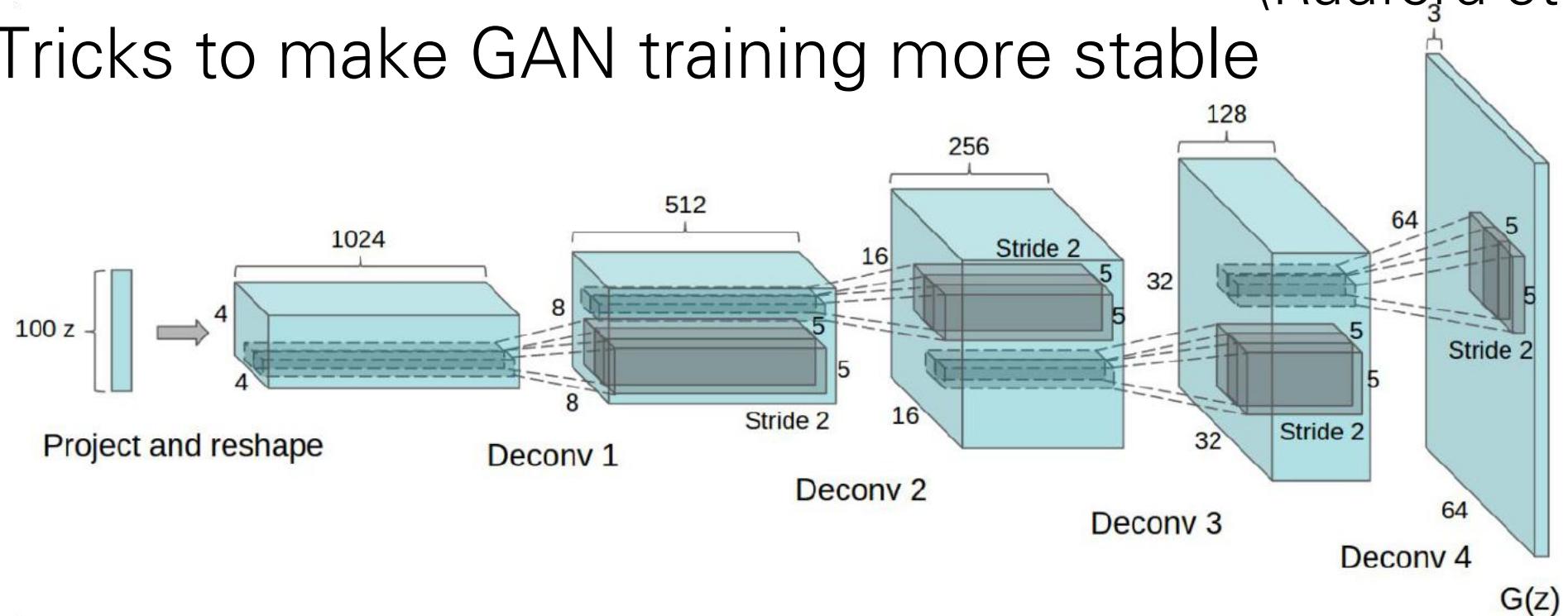
CIFAR10 samples
(convolutional discriminator,
deconvolutional generator)

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

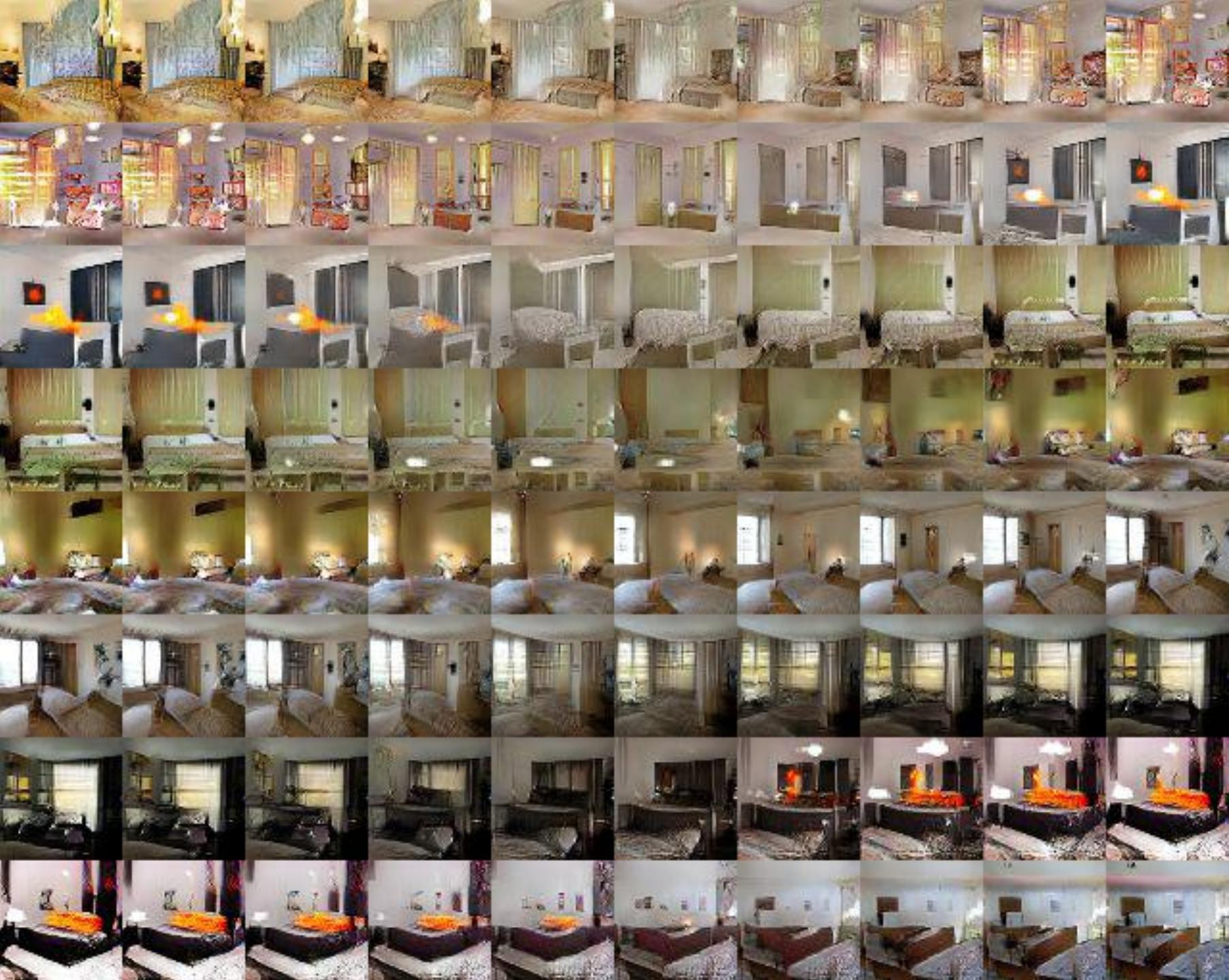
(Radford et al.,
2015)



Walking over the latent space

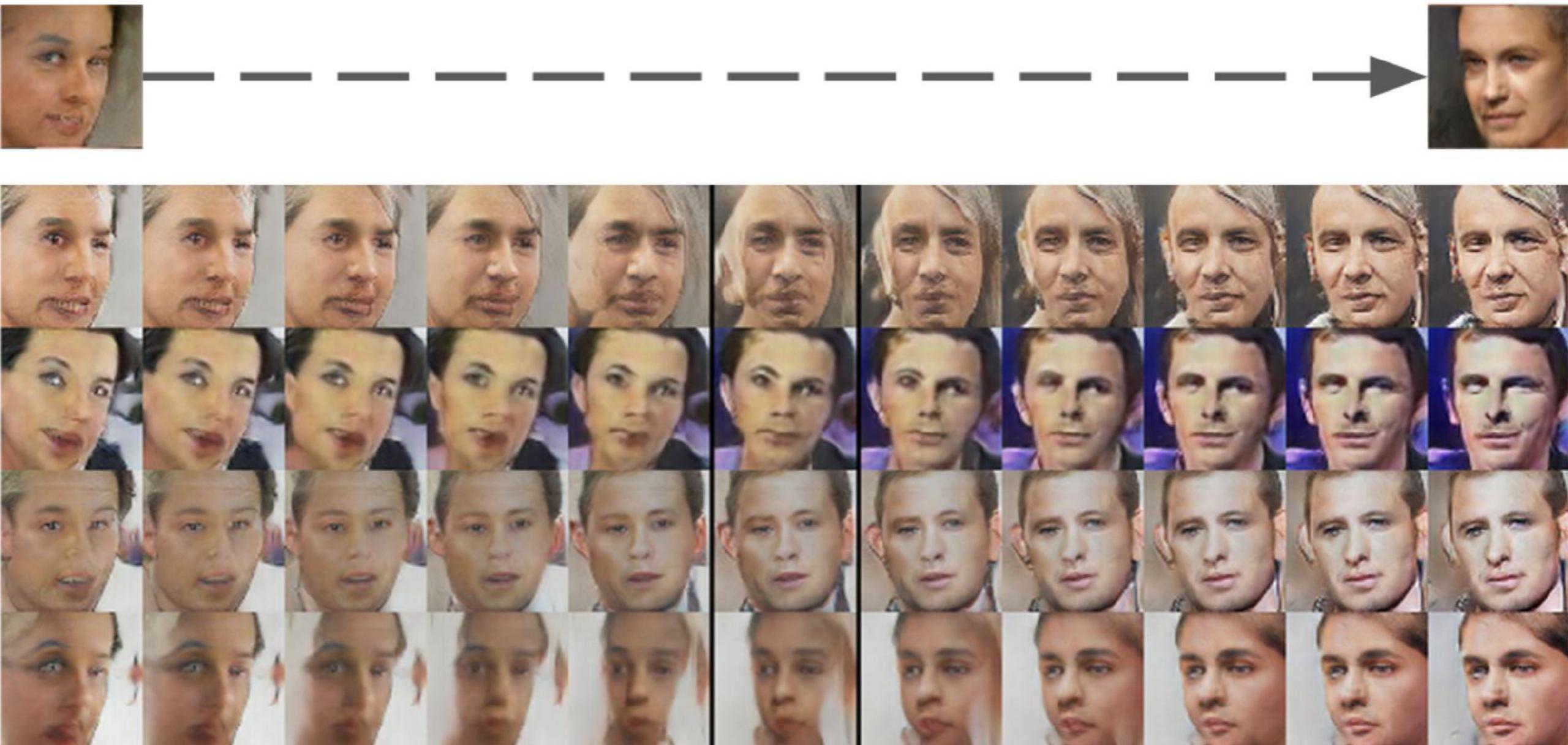
(Radford et al., 2015)

- Interpolation suggests non-overfitting behavior



Walking over the latent space

(Radford et al., 2015)



Vector Space Arithmetic

(Radford et al., 2015)



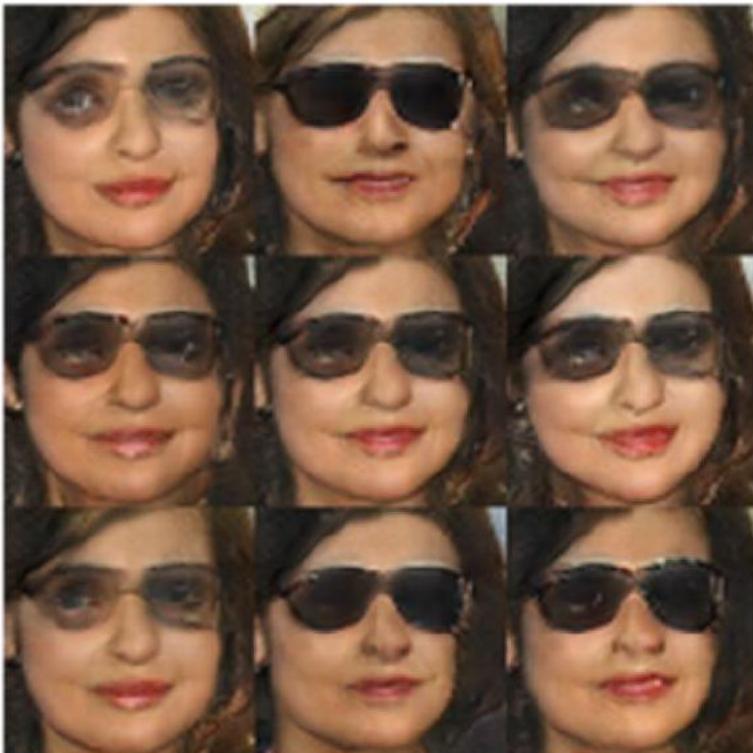
man
with glasses



man
without glasses



woman
without glasses



woman with glasses

Vector Space Arithmetic

(Radford et al., 2015)



smiling
woman



neutral
woman

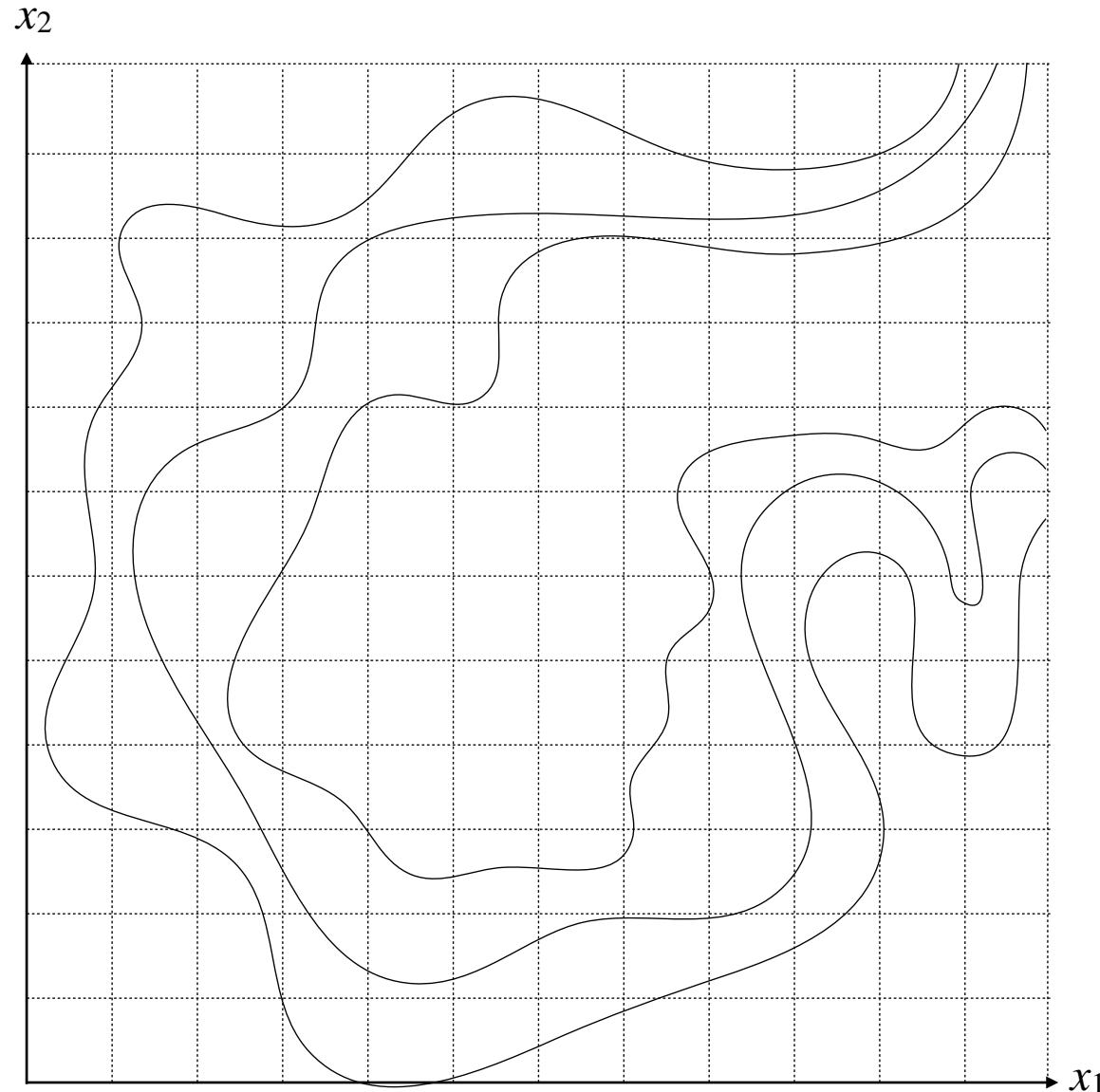


neutral
man

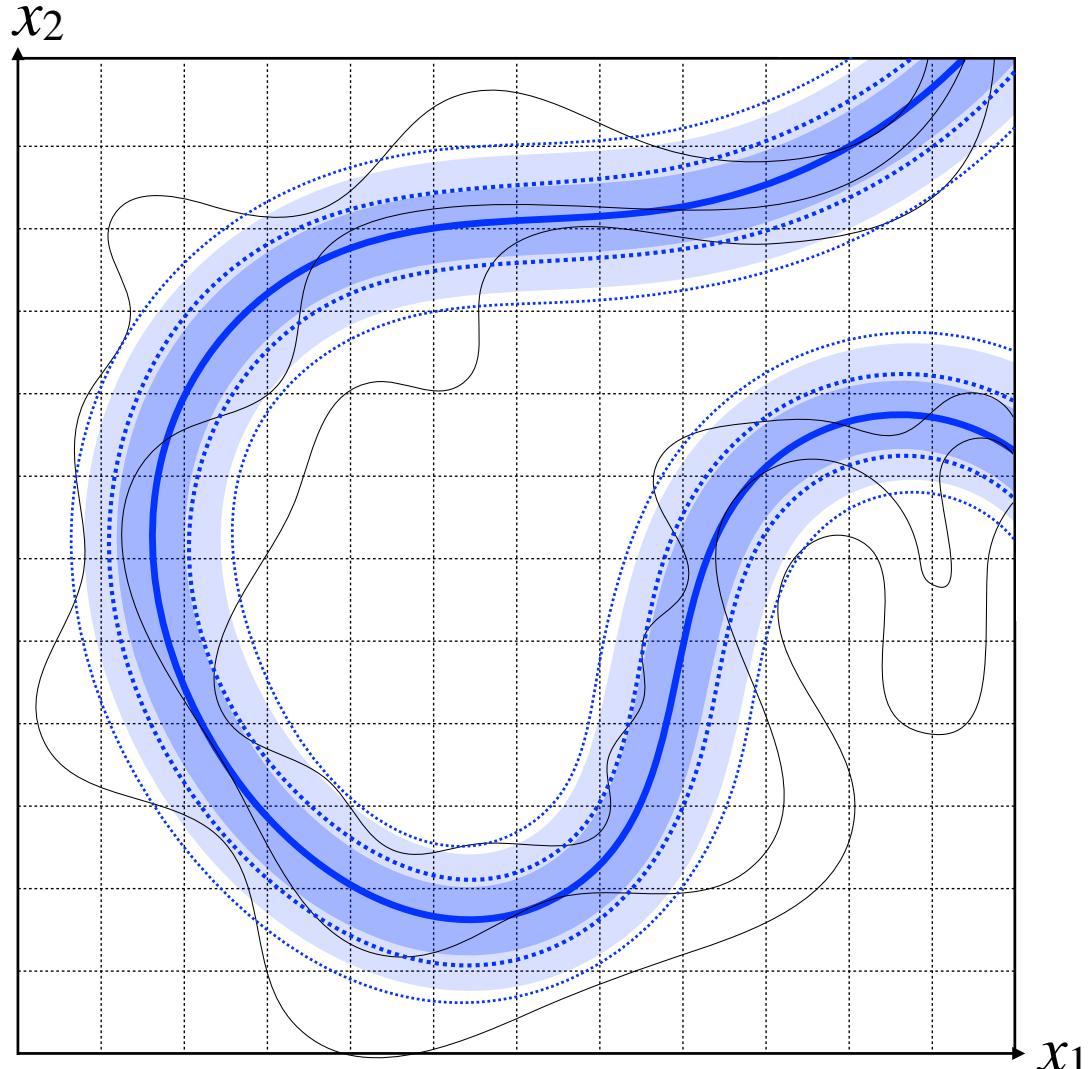


smiling man

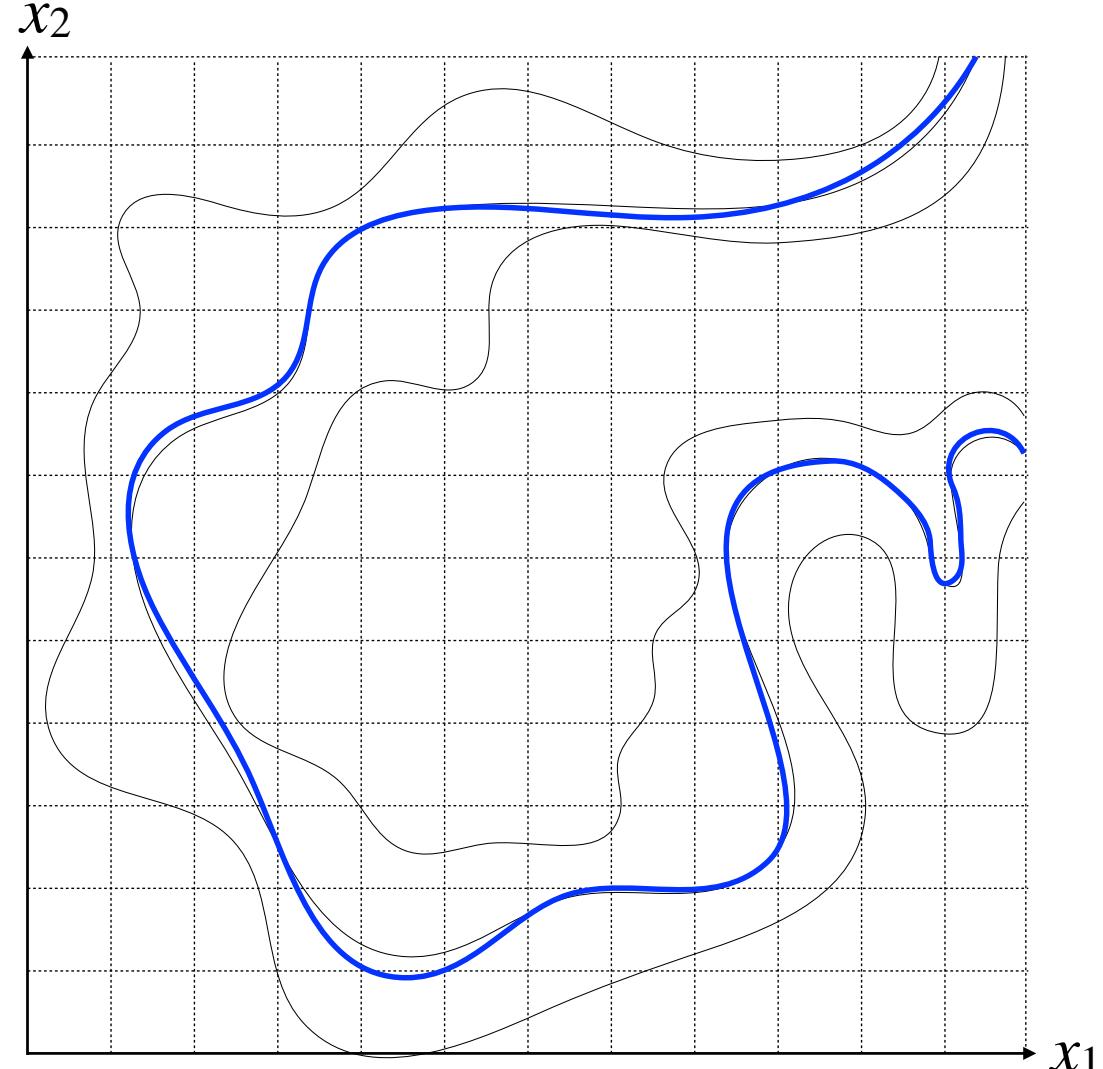
Cartoon of the Image manifold



What makes GANs special?



more traditional max-likelihood approach

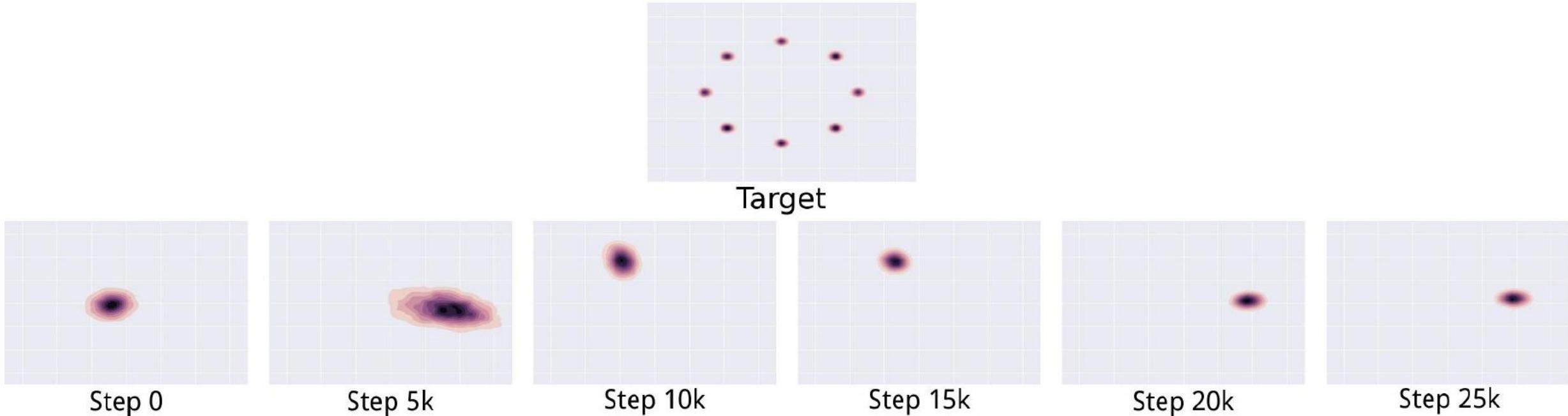


GAN

GAN Failures: Mode Collapse

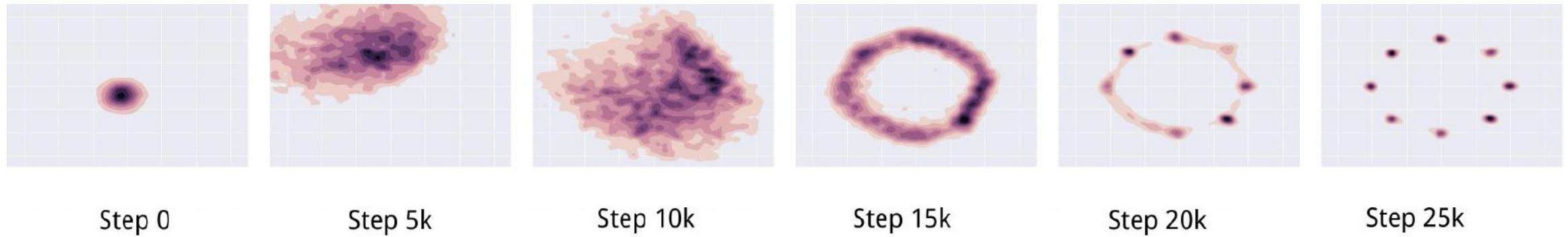
$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$$

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



Mode Collapse: Solutions

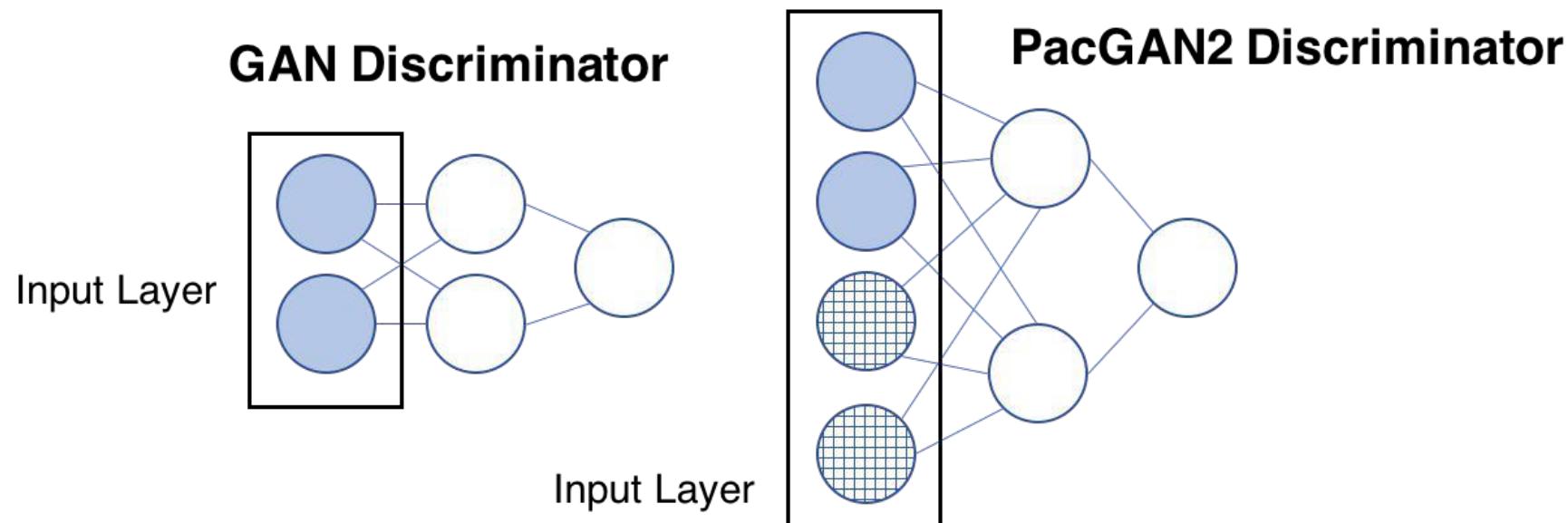
- **Unrolled GANs** (Metz et al 2016): Prevents mode collapse by backproping through a set of (k) updates of the discriminator to update generator parameters



- **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.

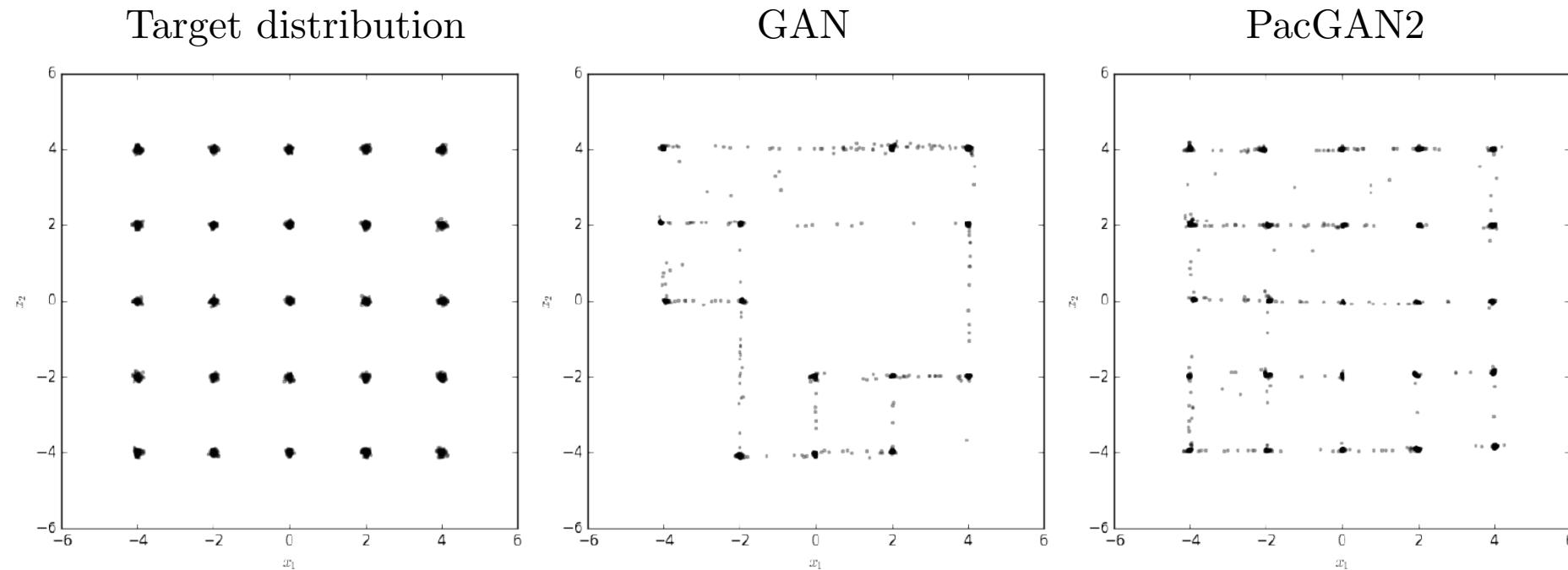


Figure 2: Scatter plot of the 2D samples from the true distribution (left) of 2D-grid and the learned generators using GAN (middle) and PacGAN2 (right). PacGAN2 captures all of the 25 modes.

GAN Evaluation

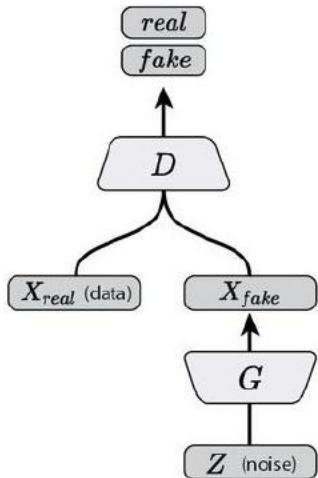
- Quantitatively evaluating GANs is not straightforward:
 - Max Likelihood is a poor indication of sample quality
- Some evaluation metrics
 - **Inception Score (IS):**
 y = labels given gen. image. $p(y|x)$ is from classifier - InceptionNet
$$\text{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x} \sim \mathbb{P}_g} [KL(p_M(y|\mathbf{x}) || p_M(y))]}$$
 - **Fréchet inception distance (FID):** (Currently most popular)
Estimate mean \mathbf{m} and covariance \mathbf{C} from classifier output - InceptionNet
$$d^2((\mathbf{m}, \mathbf{C}), (\mathbf{m}_w, \mathbf{C}_w)) = \|\mathbf{m} - \mathbf{m}_w\|_2^2 + \text{Tr}(\mathbf{C} + \mathbf{C}_w - 2(\mathbf{C}\mathbf{C}_w)^{1/2})$$
 - **Kernel MMD** (Maximum Mean Discrepancy):

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

Subclasses of GANs

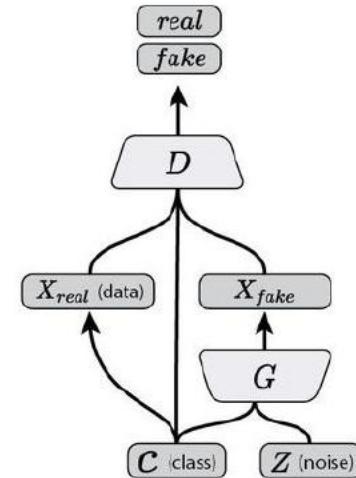
Vanilla GAN

Vanilla GAN
(Goodfellow, et al., 2014)

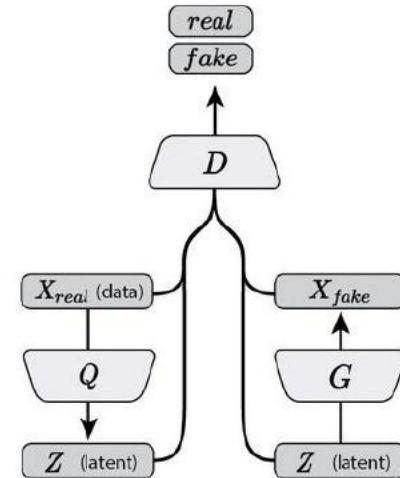


Discriminator Looks at Latent Variables

Conditional GAN
(Mirza & Osindero, 2014)

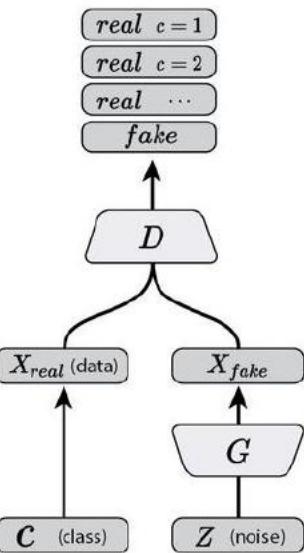


Bidirectional GAN
(Donahue, et al., 2016; Dumoulin, et al., 2016)

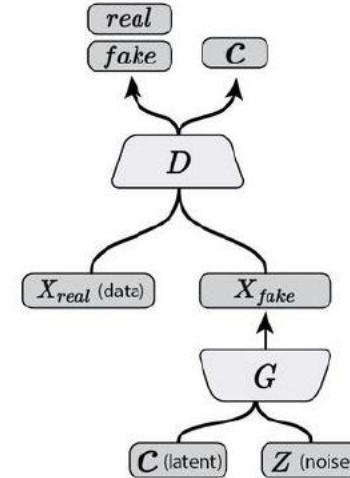


Discriminator Predicts Latent Variables

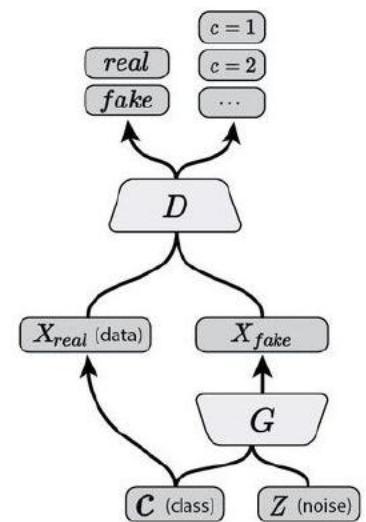
Semi-Supervised GAN
(Odena, 2016; Salimans, et al., 2016)



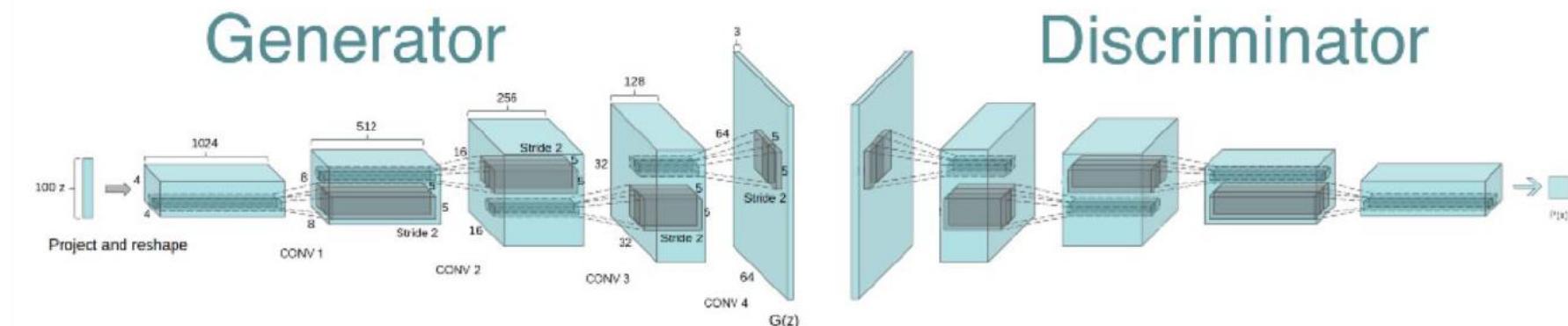
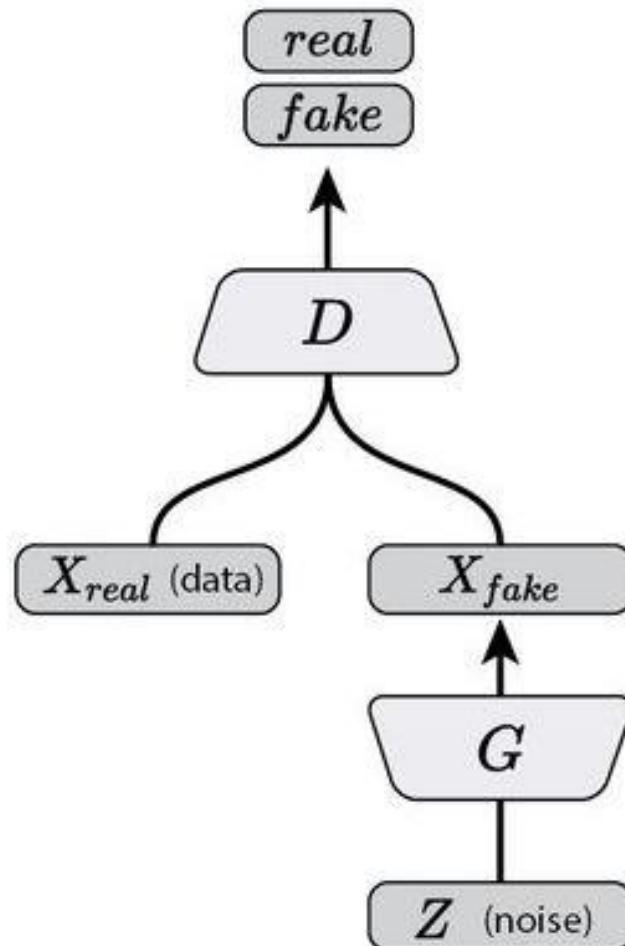
InfoGAN
(Chen, et al., 2016)



Auxiliary Classifier GAN
(Odena, et al., 2016)



Vanilla GAN (Goodfellow et al., 2014)

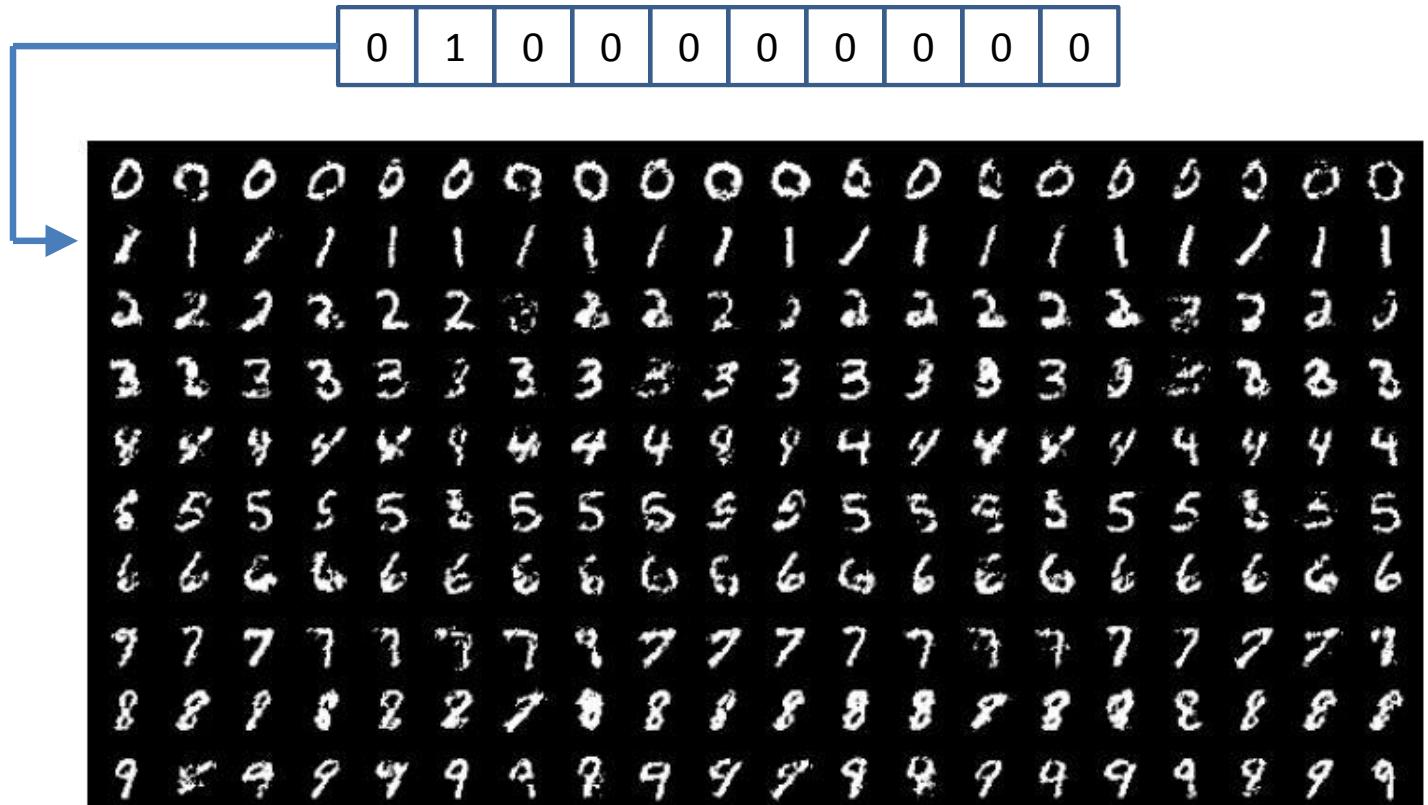
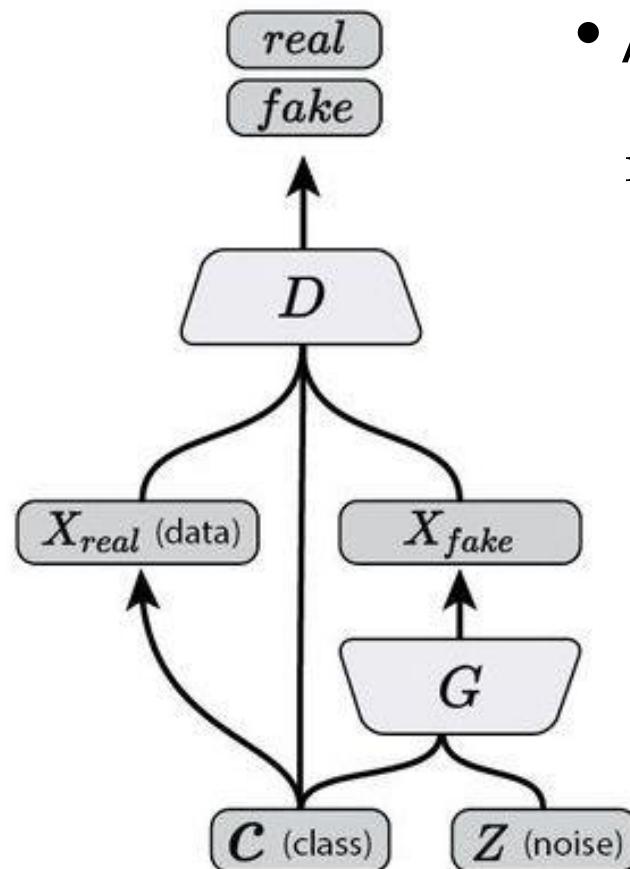


DCGAN (Radford et al., 2015)

Conditional GAN (Mirza and Osindero, 2014)

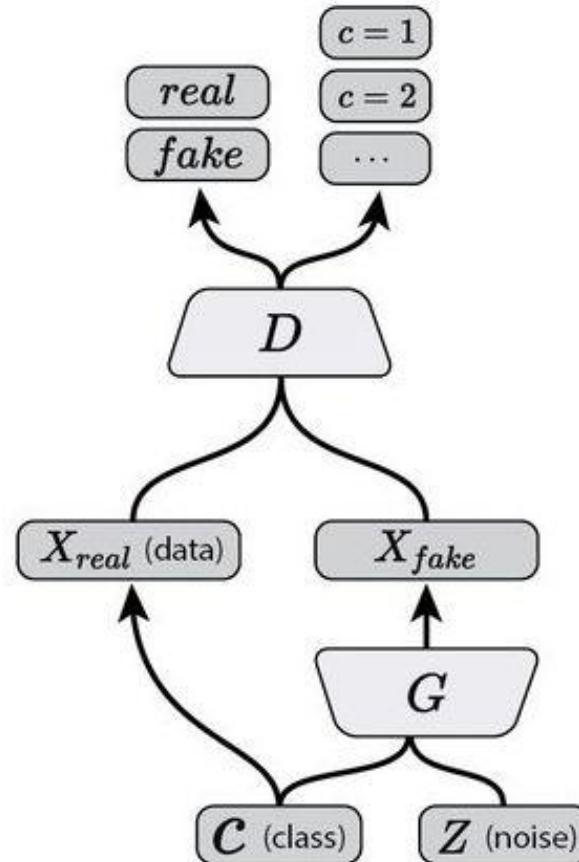
- Add conditional variables \mathbf{y} into G and D

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



Auxiliary Classifier GAN (Odena et al., 2016)

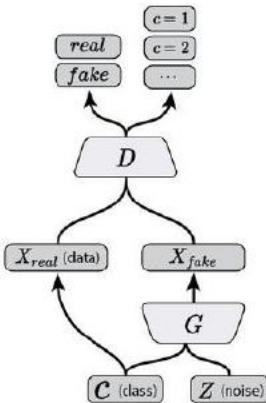
- Every generated sample has a corresponding class label



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

- 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



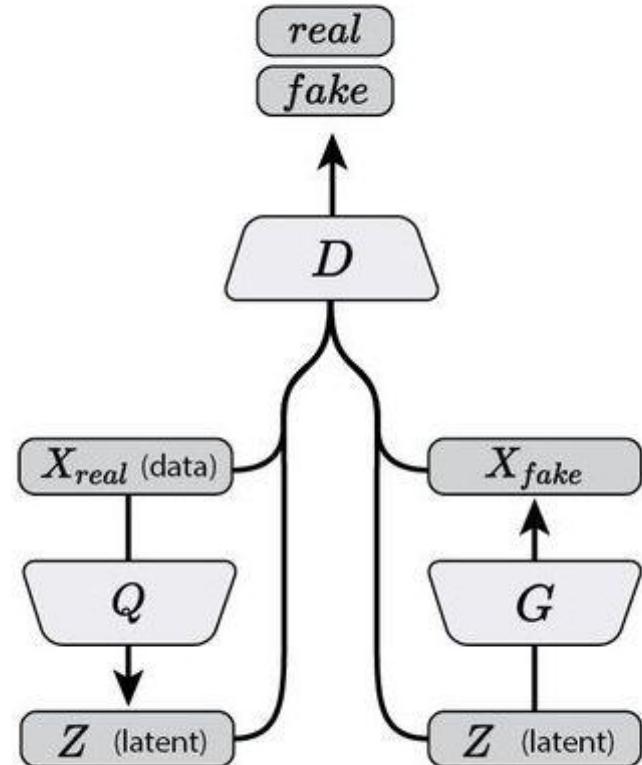
redshank



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.



$$\begin{aligned} \min_G \max_D V(D, G) &= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}, G_z(\mathbf{x})))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G_x(\mathbf{z}), \mathbf{z}))] \\ &= \iint q(\mathbf{x})q(\mathbf{z} \mid \mathbf{x}) \log(D(\mathbf{x}, \mathbf{z})) d\mathbf{x} d\mathbf{z} \\ &\quad + \iint p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}) \log(1 - D(\mathbf{x}, \mathbf{z})) d\mathbf{x} d\mathbf{z}. \end{aligned}$$



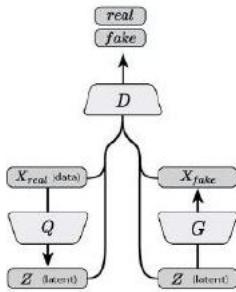
CelebA reconstructions



SVNH reconstructions

Bidirectional GAN

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



LSUN bedrooms



Tiny ImageNet

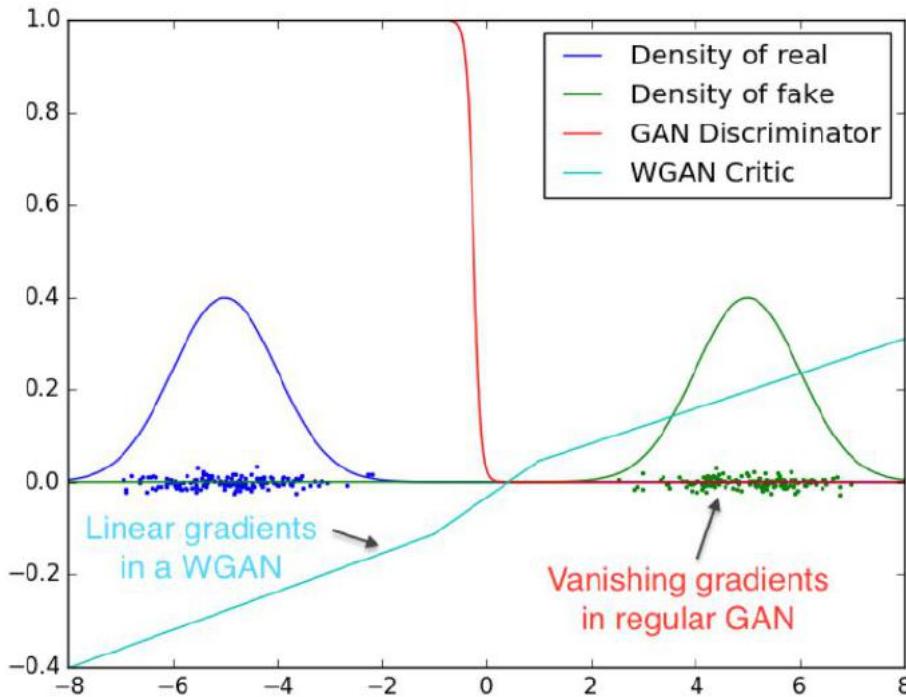


Wasserstein GAN (Arjovsky et al., 2016)

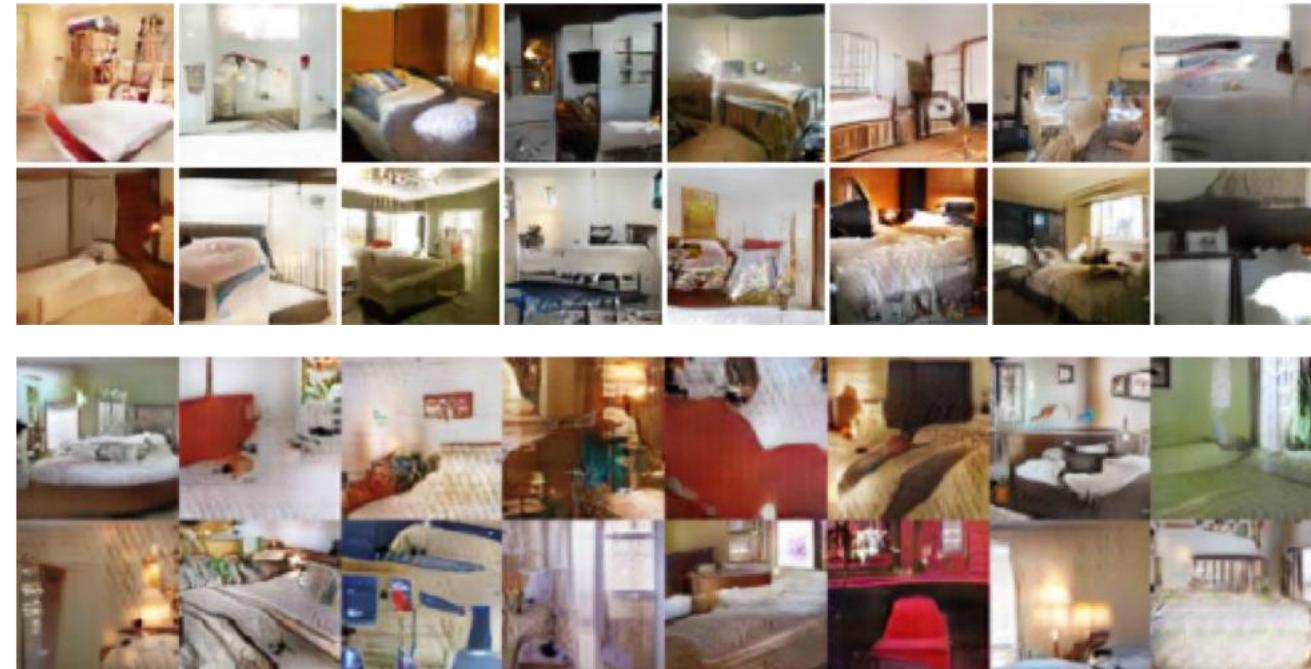
- Objective based on Earth-Mover or Wasserstein distance:

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

- Provides nice gradients over real and fake samples



WGAN

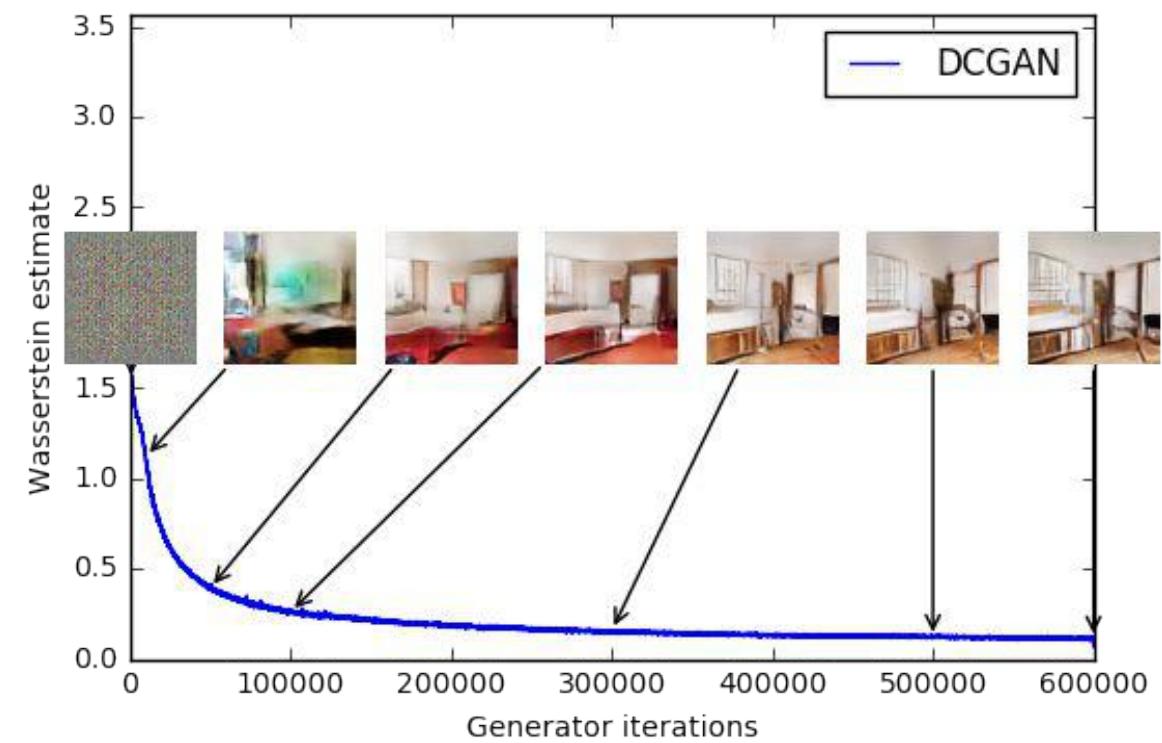
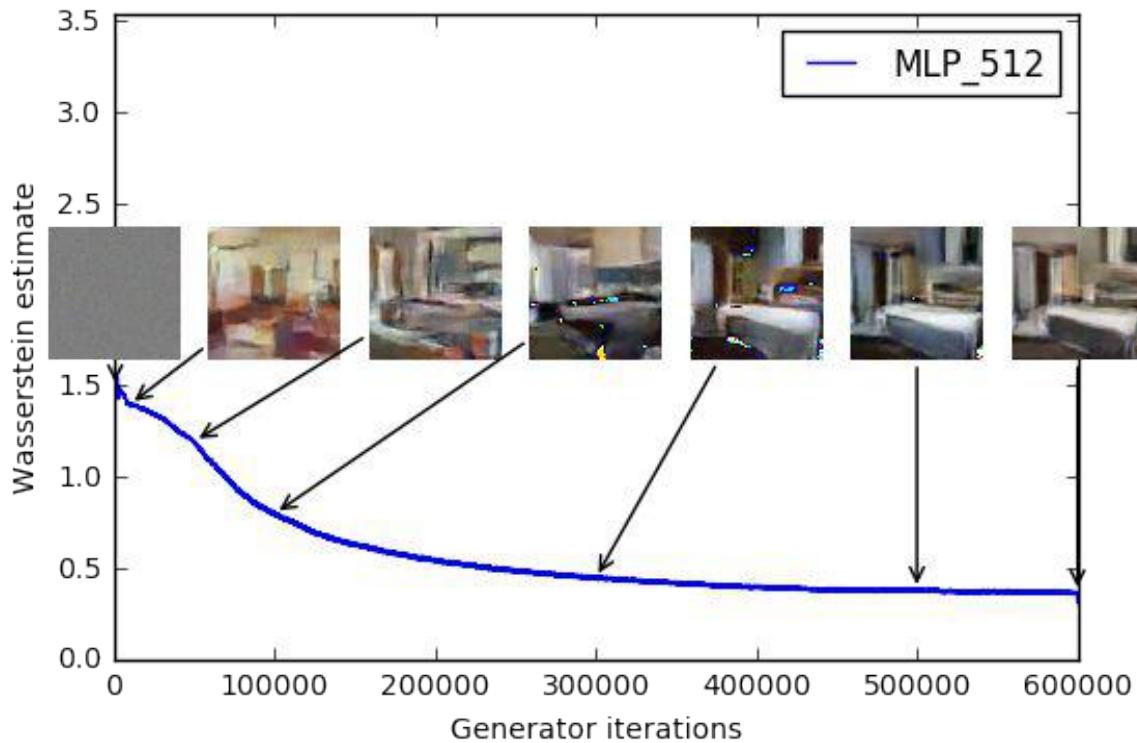


DCGAN



Wasserstein GAN (Arjovsky et al., 2016)

- Wasserstein loss seems to correlate well with image quality.



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

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

- 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

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

WGAN with gradient penalty

Busino game camperate spent odea
In the bankaway of smarling the
SingersMay , who kill that imvic
Keray Pents of the same Reagun D
Manging include a tudancs shat "
His Zuith Dudget , the Denmbern
In during the Uitational questio
Divos from The ' noth ronkies of
She like Monday , of macunsuer S
The investor used ty the present
A papees are cointry congress oo
A few year inom the group that s
He said this syenn said they wan
As a world 1 88 ,for Autouries
Foand , th Word people car , Il
High of the upseader homing pull
The guipe is worly move dogsfor
The 1874 incidested he could be
The allo tooks to security and c

Solice Norkedin pring in since
This record (31.) UBS) and Ch
It was not the annuas were plogr
This will be us , the ect of DAN
These leaded as most-worsd p2 a0
The time I paid0a South Cubry i
Dour Fraps higs it was these del
This year out howneed allowed lo
Kaulna Seto consficates to repor
A can teal , he was schoon news
In th 200. Pesish picriers rega
Konney Panice rimimber the teami
The new centuct cut Denester of
The near , had been one injostie
The incestion to week to shorted
The company the high product of
20 - The time of accomplete , wh
John WVuderenson seqivic spends
A ceetens in indestdredly the Wat

Standard GAN objective

ddddddddd ddd ddd ddd ddd ddd ddd
dd ddd ddd ddd ddd ddd ddd ddd ddd ddd

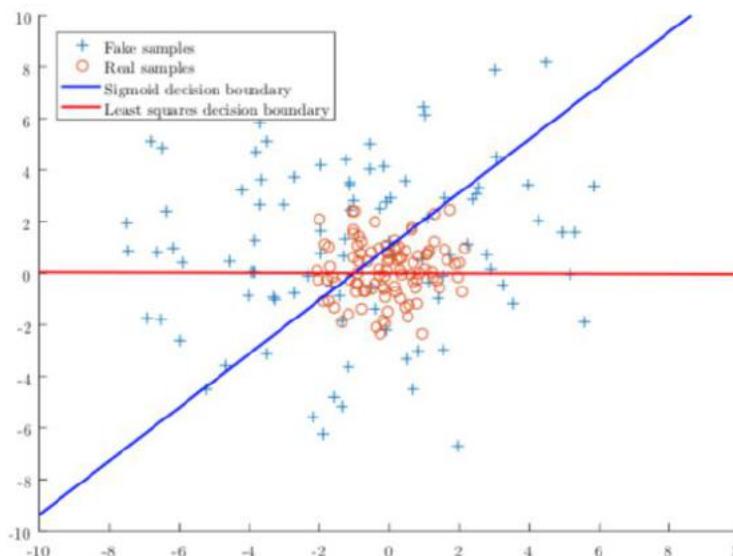
dd ddd ddd ddd ddd ddd ddd ddd ddd ddd
dd ddd ddd ddd ddd ddd ddd ddd ddd ddd

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

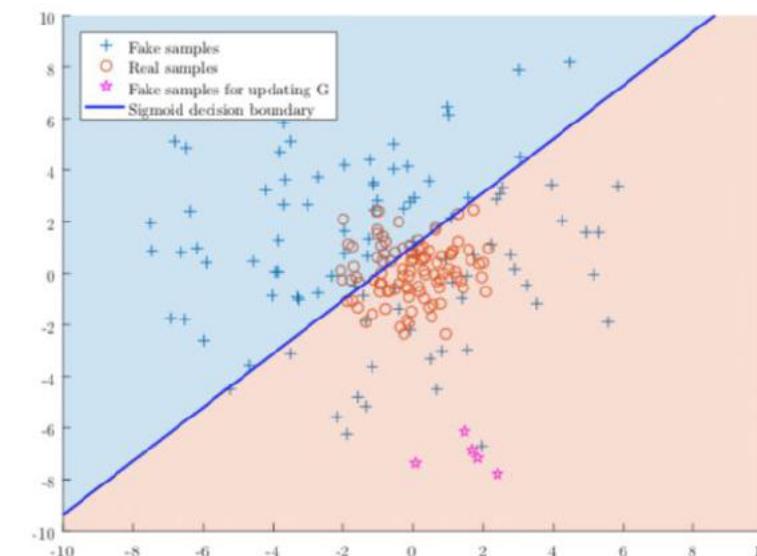
- Use a loss function that provides smooth and non-saturating gradient in discriminator D

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

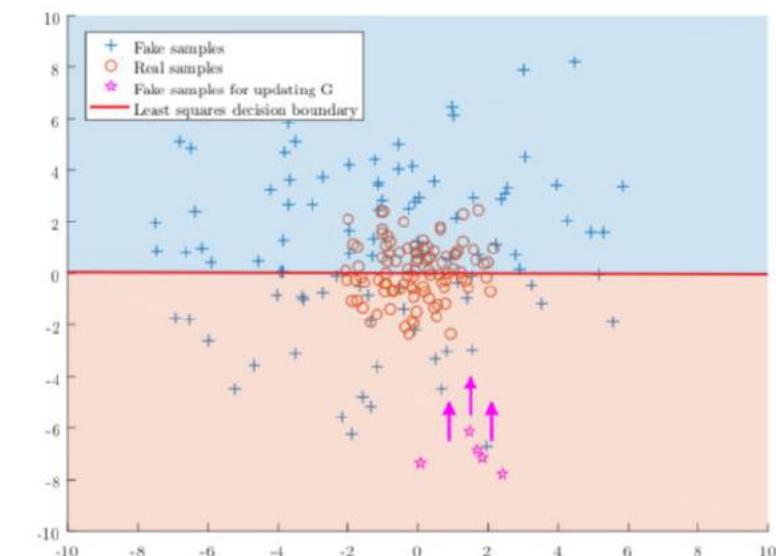
$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})) - c)^2],$$



Decision boundaries of Sigmoid & Least Squares loss functions



Sigmoid decision boundary

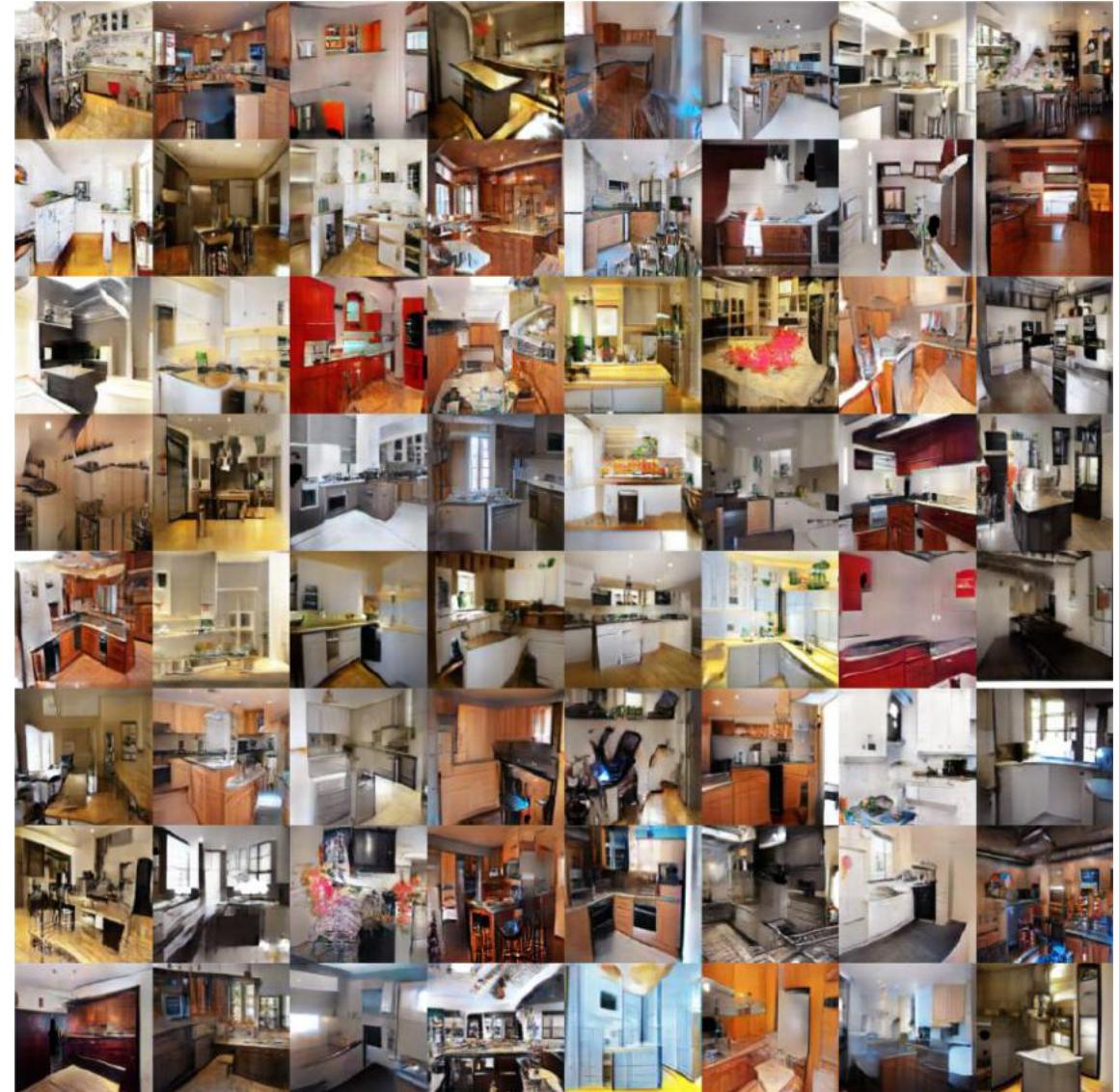


Least Squares decision boundary

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

$$\mathcal{L}(v) = |v - D(v)|^\eta \text{ where } \begin{cases} D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} \\ \eta \in \{1, 2\} \\ v \in \mathbb{R}^{N_x} \end{cases}$$

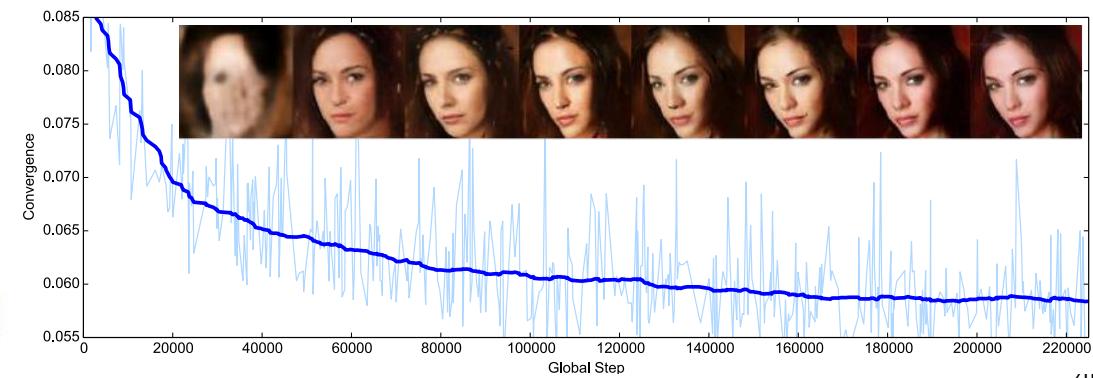
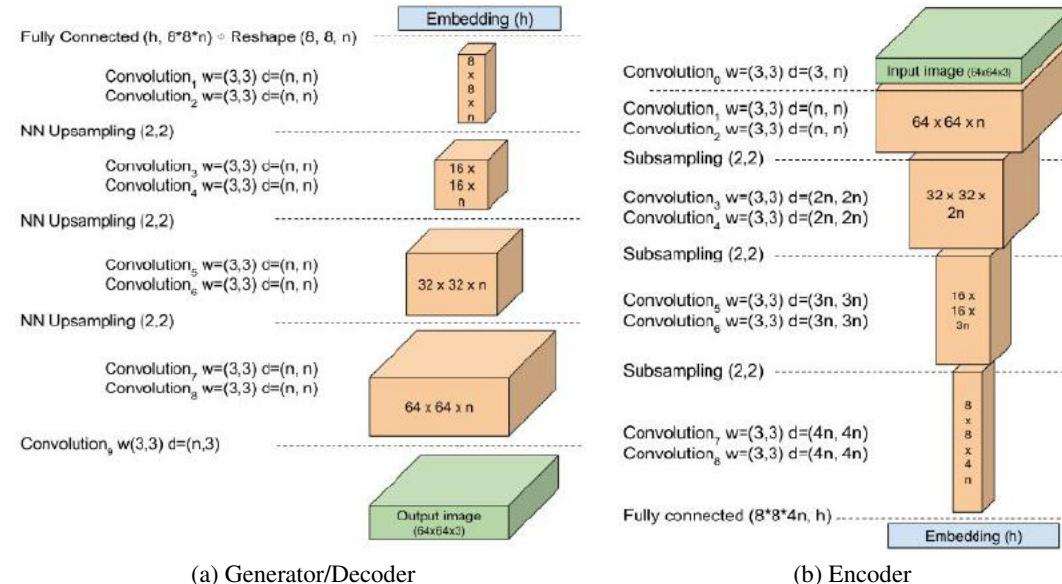
is the autoencoder function.
is the target norm.
is a sample of dimension N_x .

- 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:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t \cdot \mathcal{L}(G(z_D)) \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) \\ k_{t+1} = k_t + \lambda_k (\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))) \end{cases}$$

for θ_D
for θ_G
for each training step t



BEGANs for CelebA

360K celebrity face images
128x128 with 128 filters

(Berthelot et al., 2017)



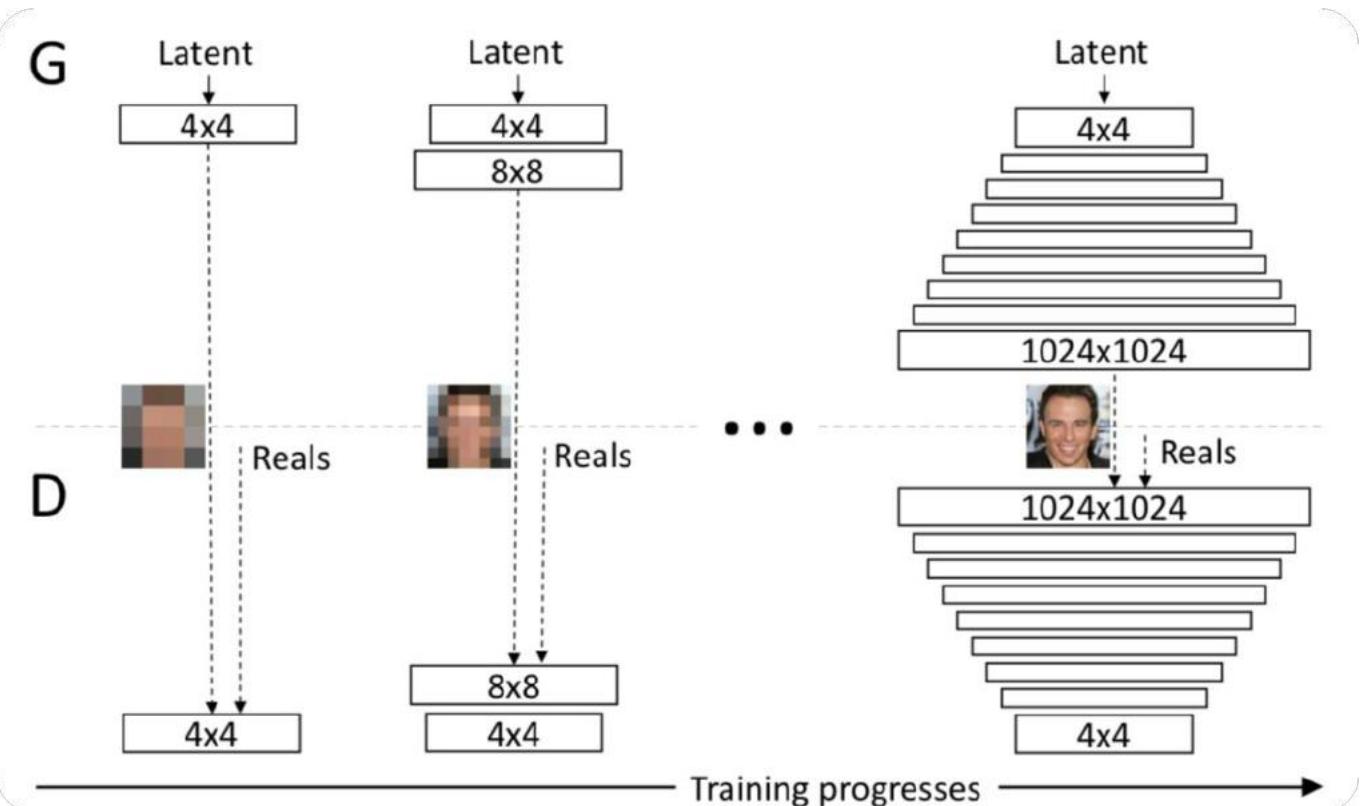
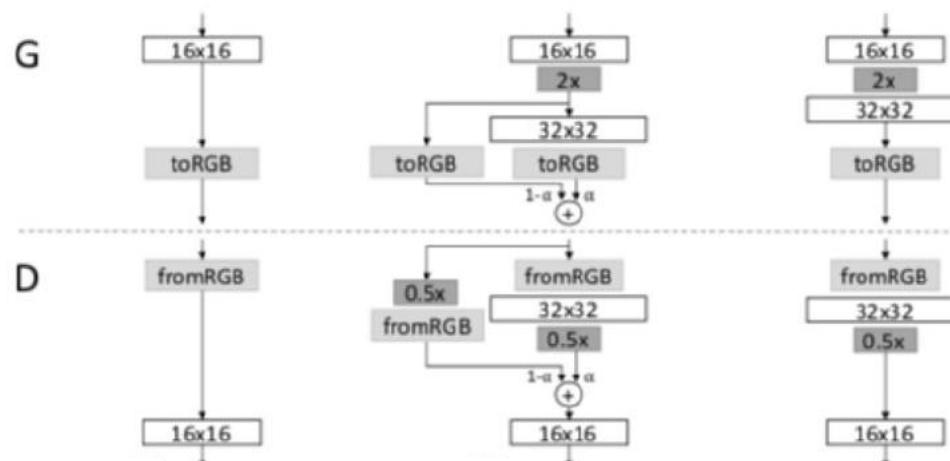
Interpolations in the latent space



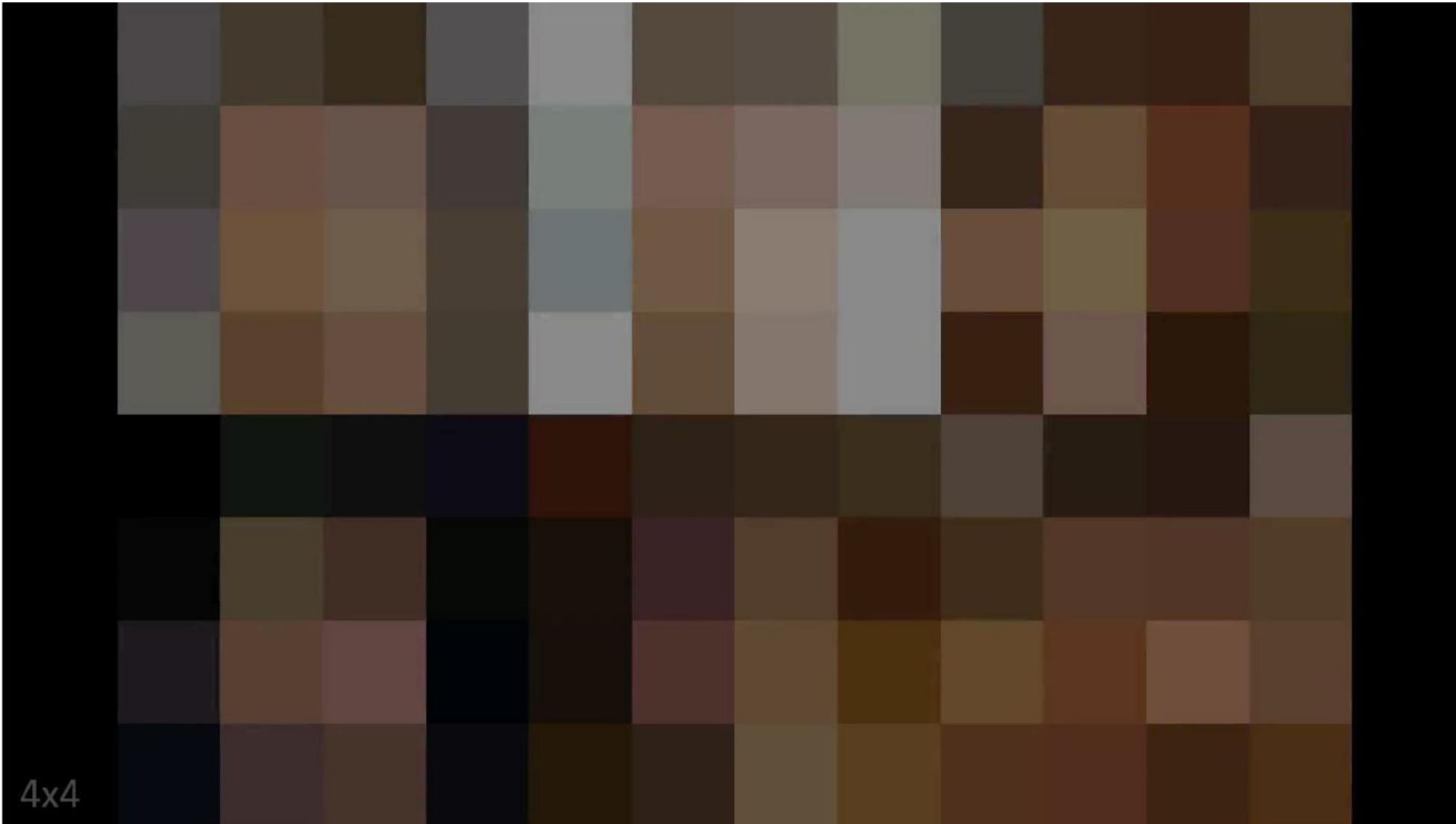
Mirror interpolation example

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)

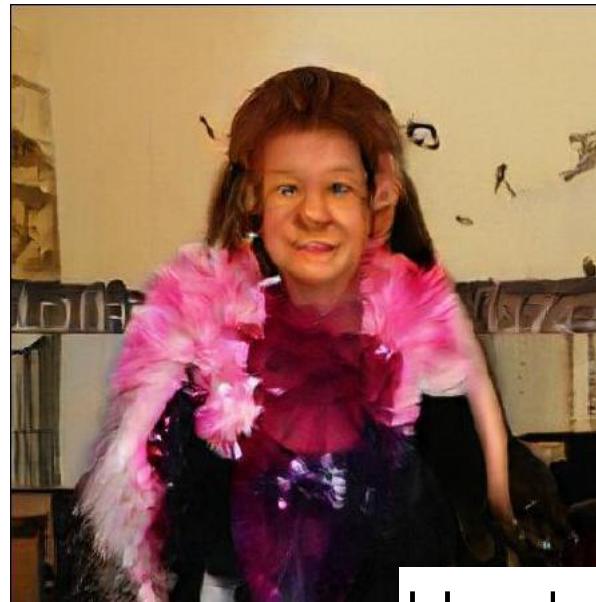
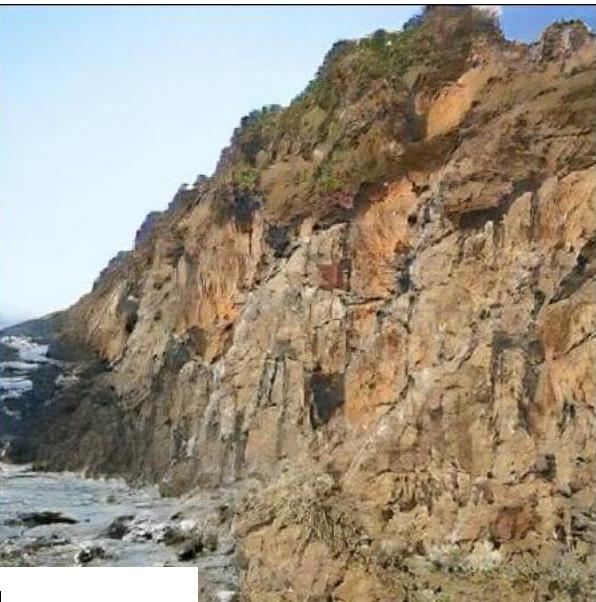
High resolution, class-conditional samples generated by the model



- 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:

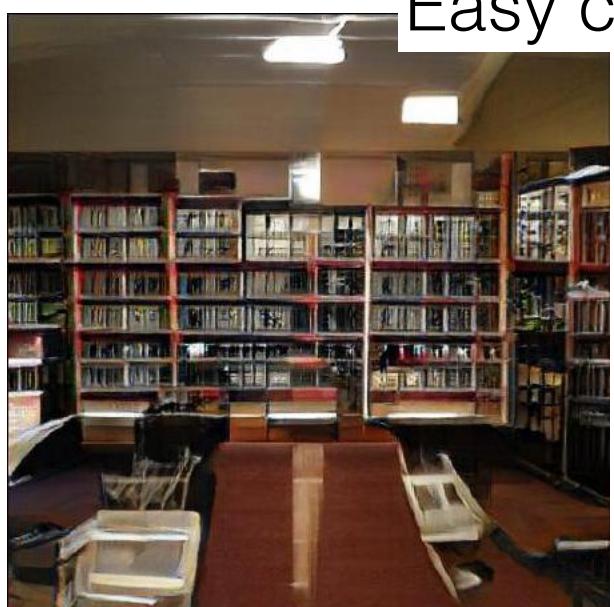
$$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_F^2,$$

BigGANs (Brock et al., 2019)



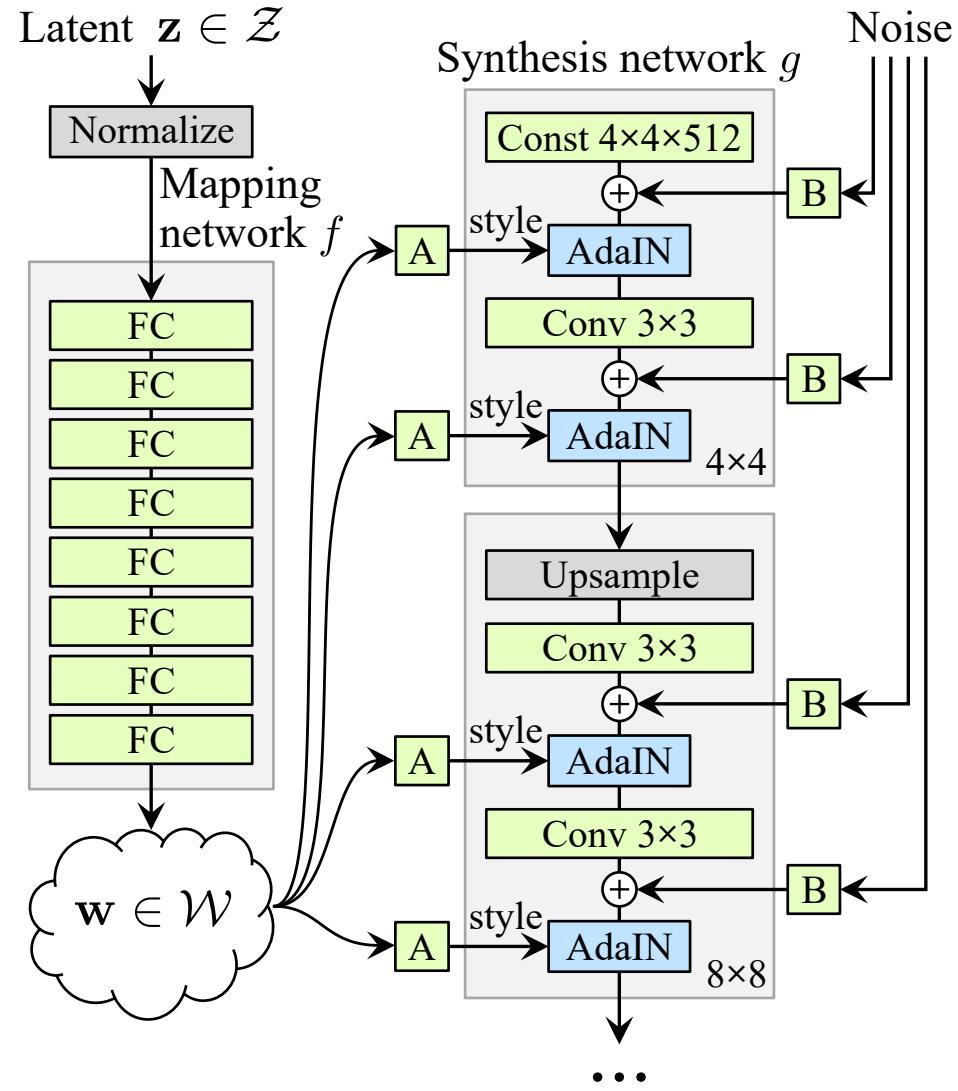
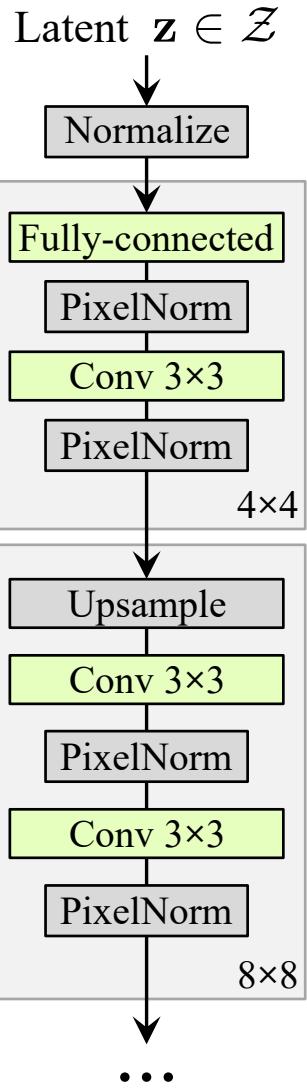
Easy classes

Hard classes



Resolution: 512x512

StyleGAN (Karras et al., 2019)



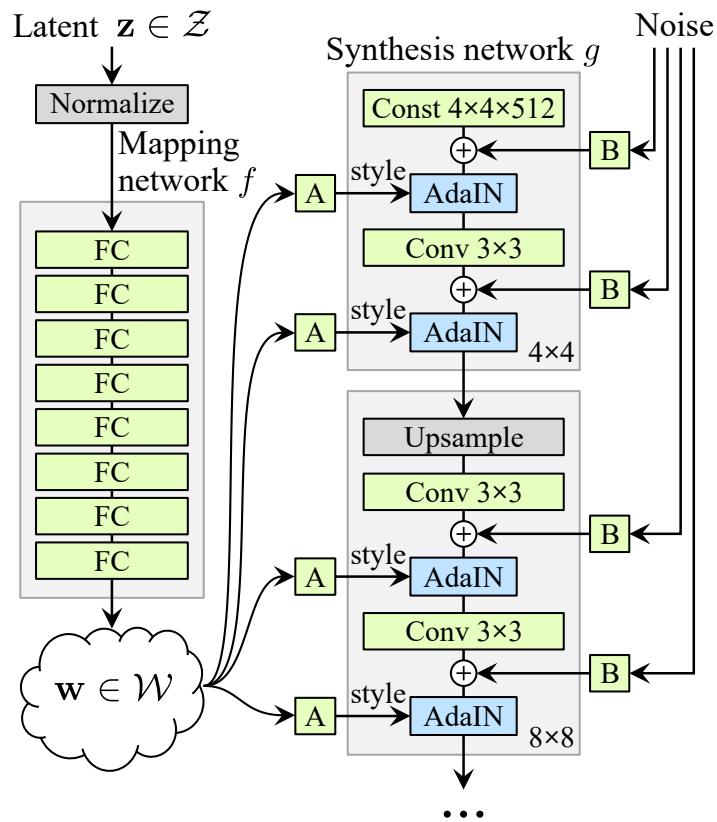
Feature map affine transformation:

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



StyleGAN (Karras et al., 2019)

- Swapping out the destination style for the source style



Some Applications of GANs

Semi-supervised Classification

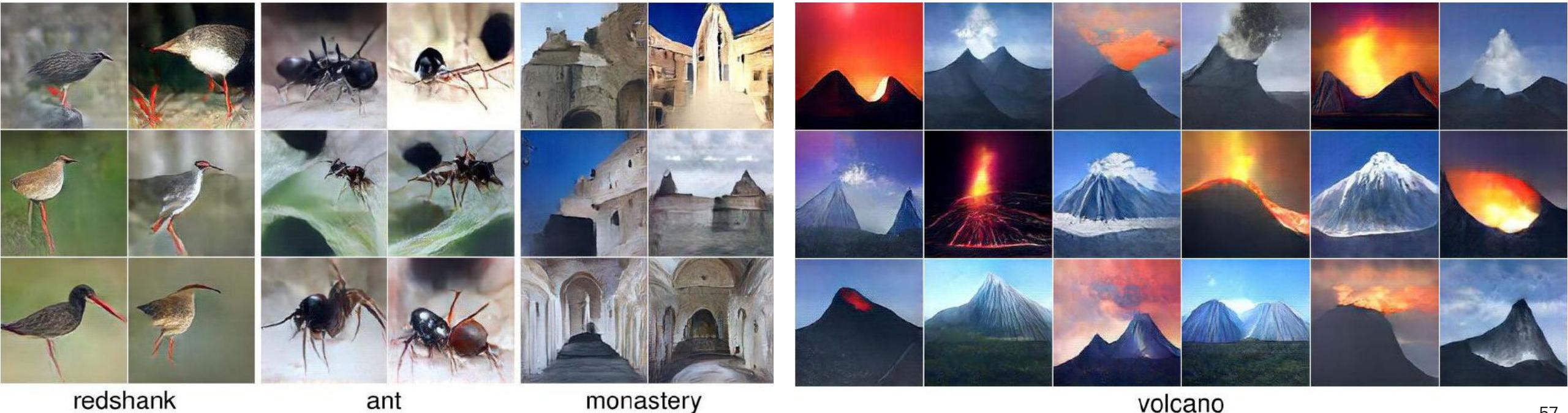
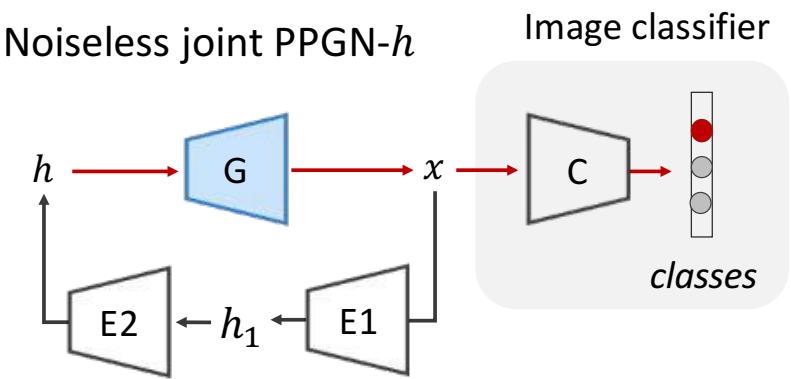
(Salimans et al., 2016;
Dumoulin et al., 2016)

SVNH

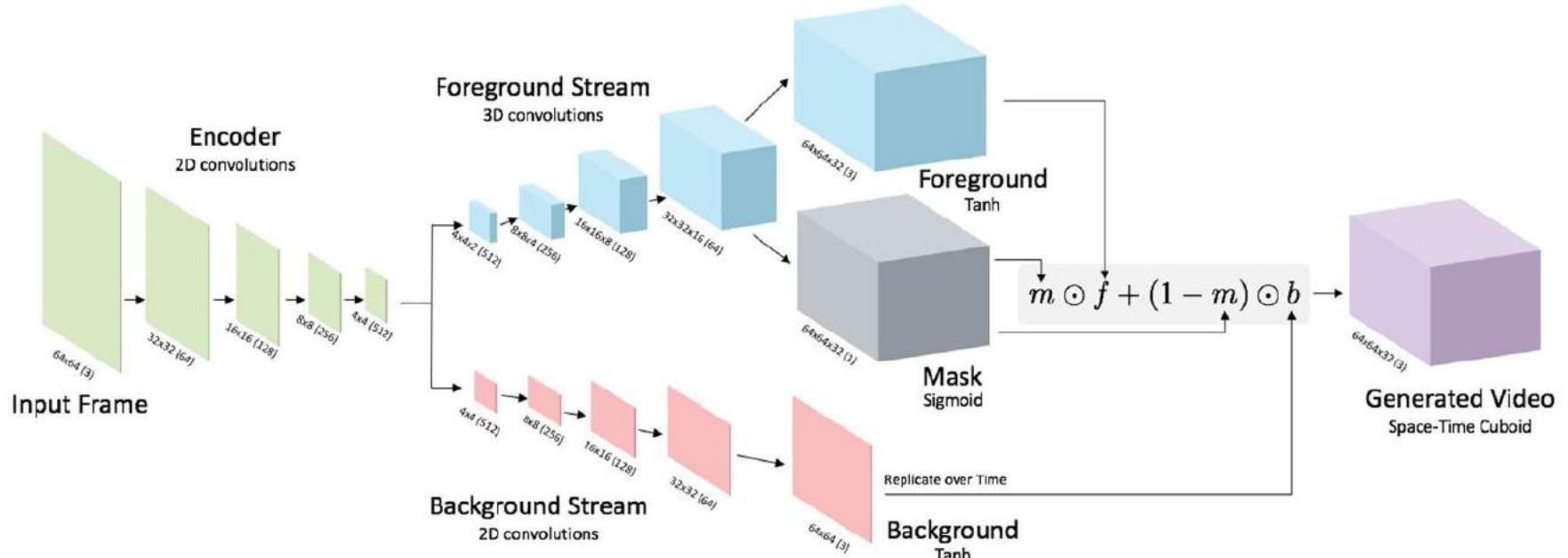
Model	Misclassification rate
VAE (M1 + M2) (Kingma et al., 2014)	36.02
SWWAE with dropout (Zhao et al., 2015)	23.56
DCGAN + L2-SVM (Radford et al., 2015)	22.18
SDGM (Maaløe et al., 2016)	16.61
GAN (feature matching) (Salimans et al., 2016)	8.11 ± 1.3
ALI (ours, L2-SVM)	19.14 ± 0.50
ALI (ours, no feature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



Video Generation (Vondrick et al., 2016)



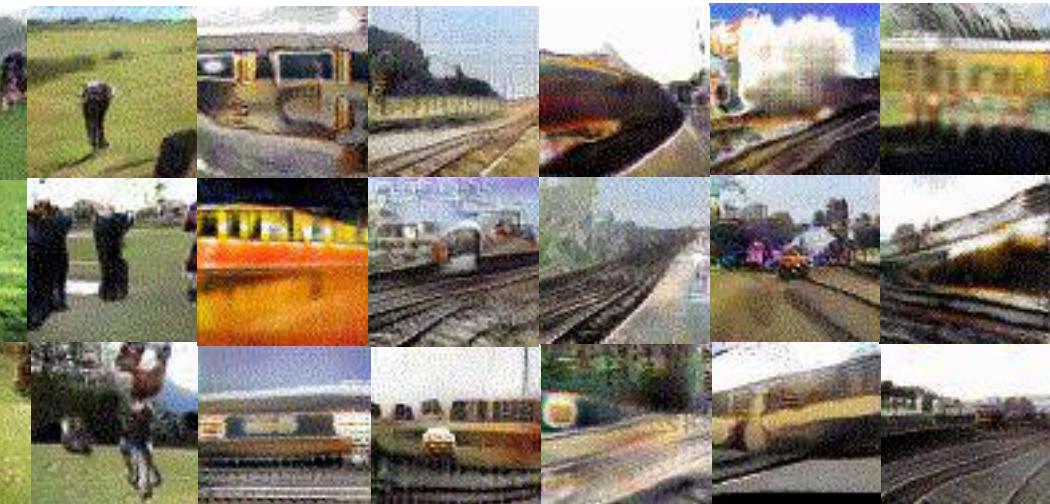
Beach



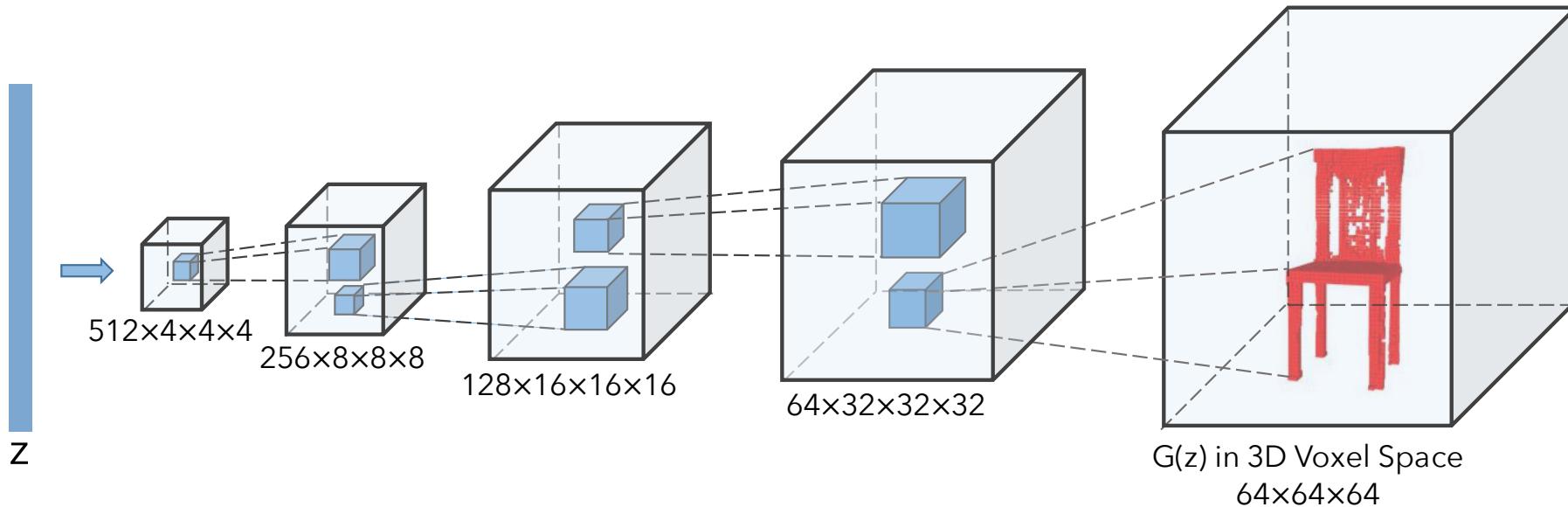
Golf



Train Station



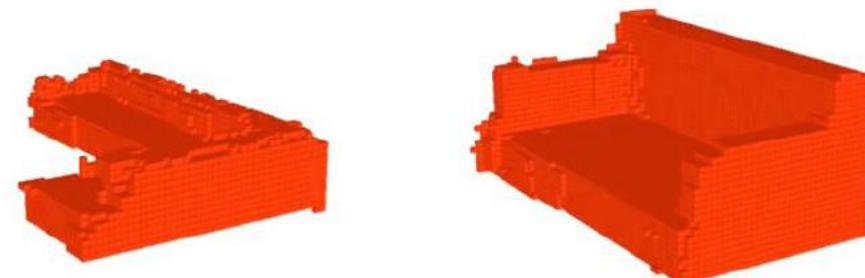
Generative Shape Modeling (Wu et al., 2016)



Chairs



Sofas



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



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

- Combine content loss with adversarial loss

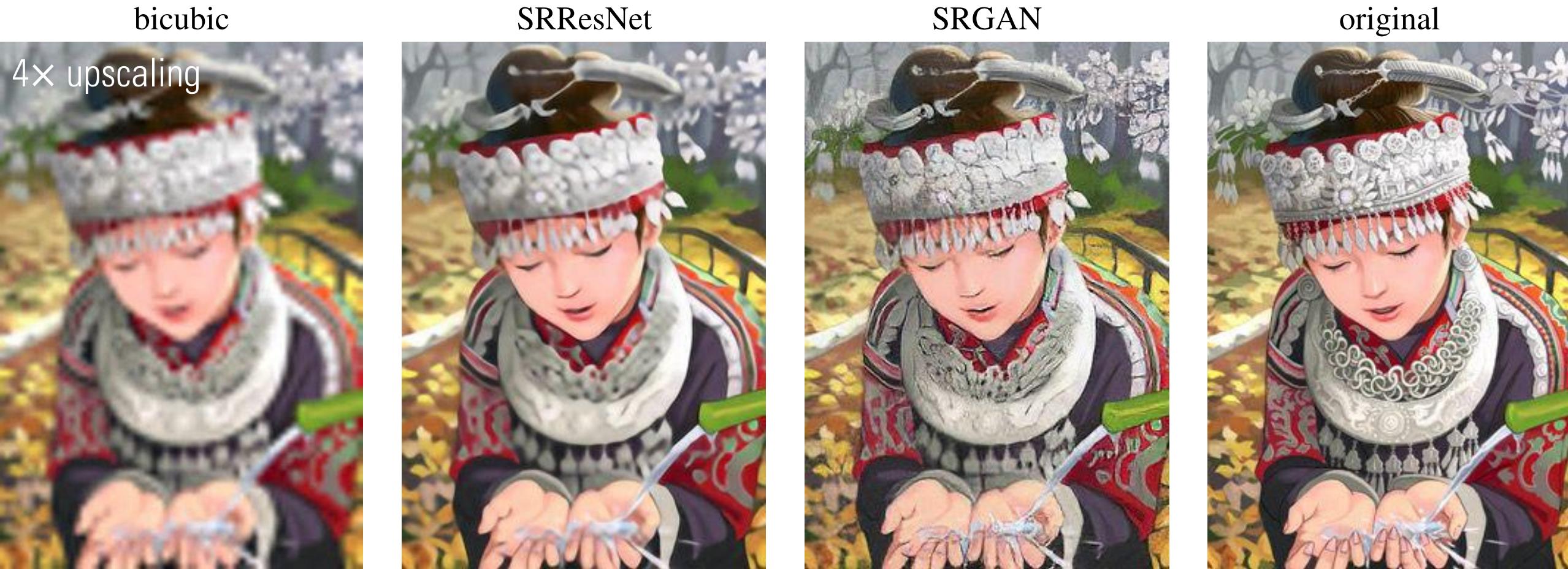
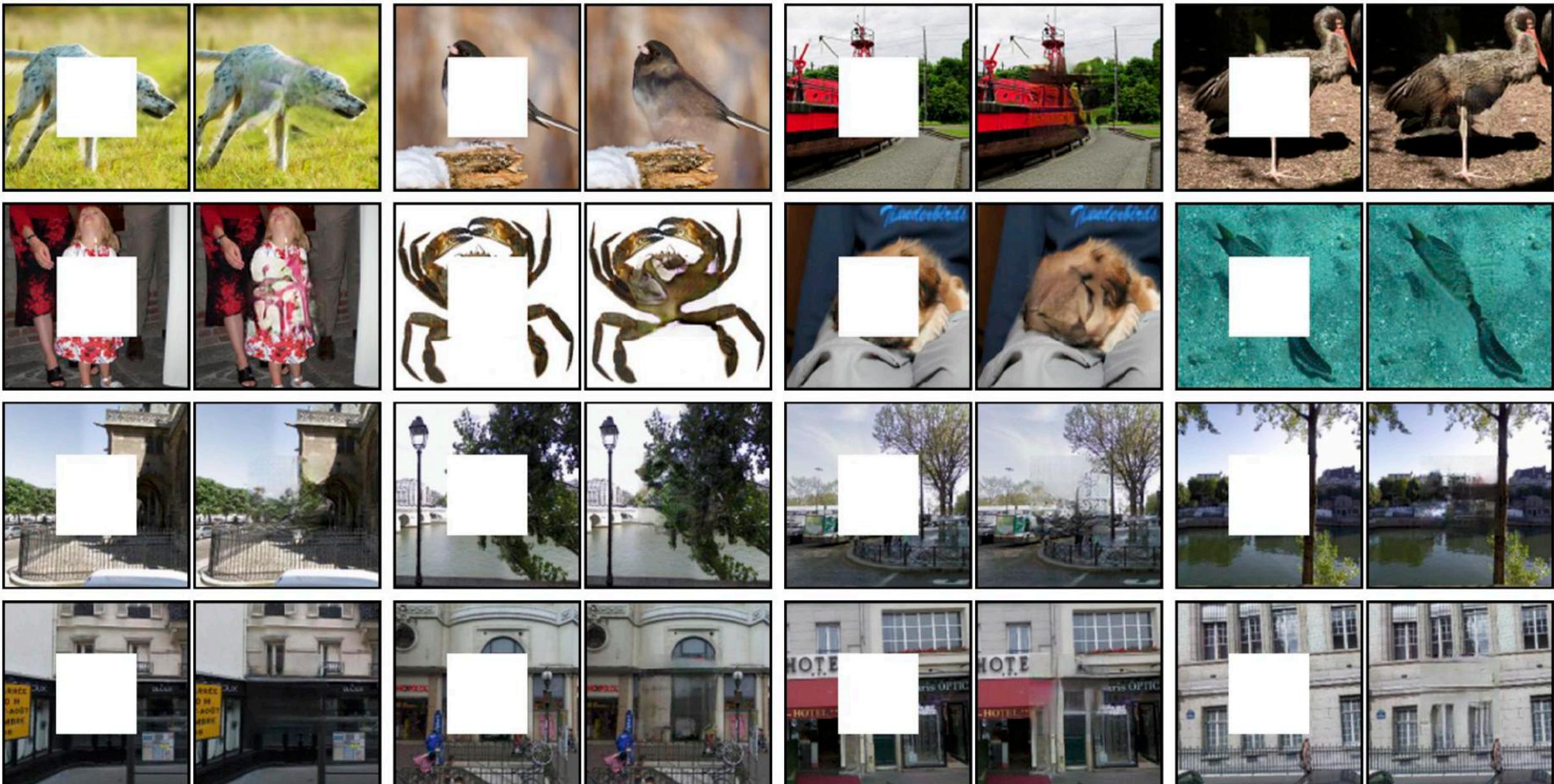


Image Inpainting (Pathak et al., 2016)



Unsupervised Domain Adaptation (Bousmalis et al., 2016)

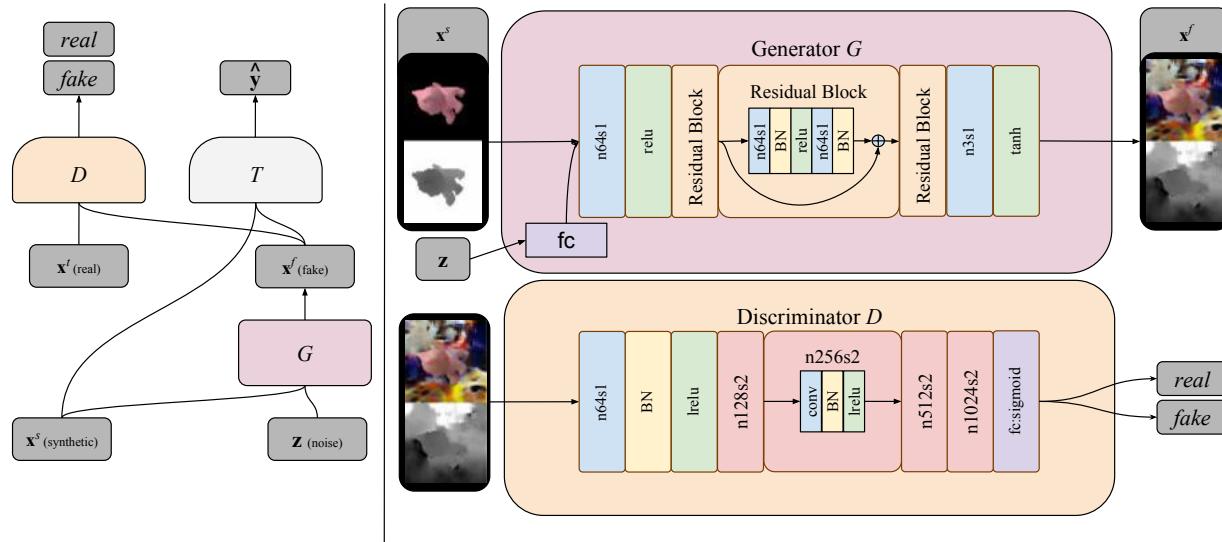


Image examples from the Linemod dataset

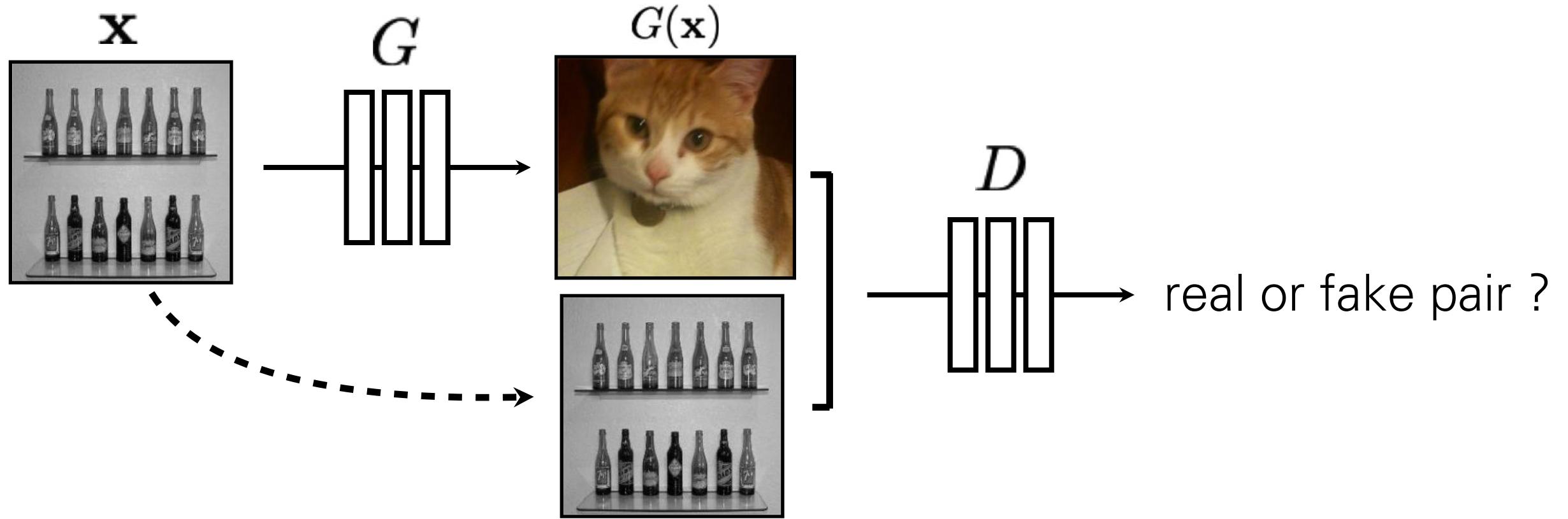


RGDB image samples
(conditioned on a synthetic image)

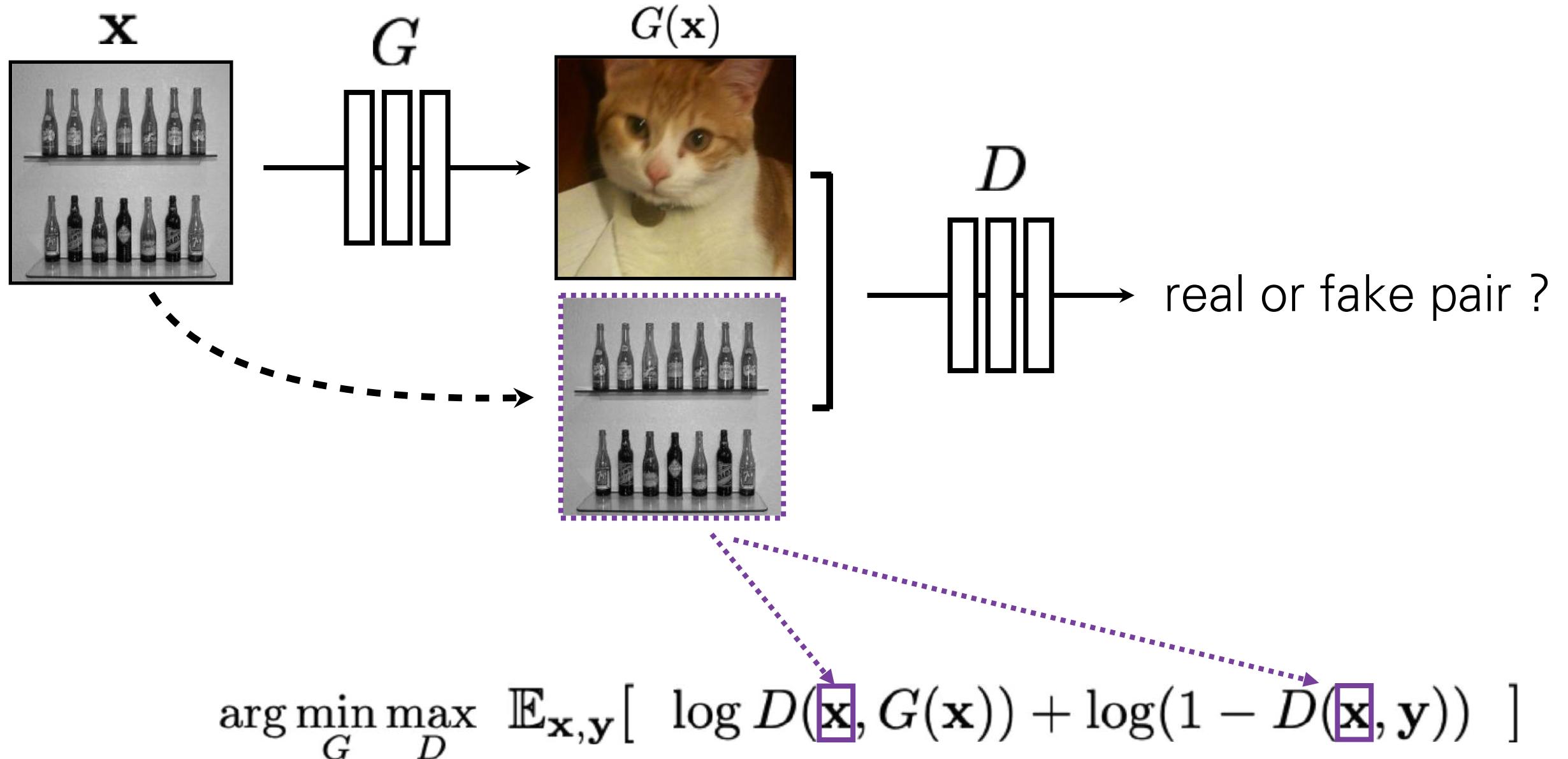
Image to Image Translation (Pix2Pix)

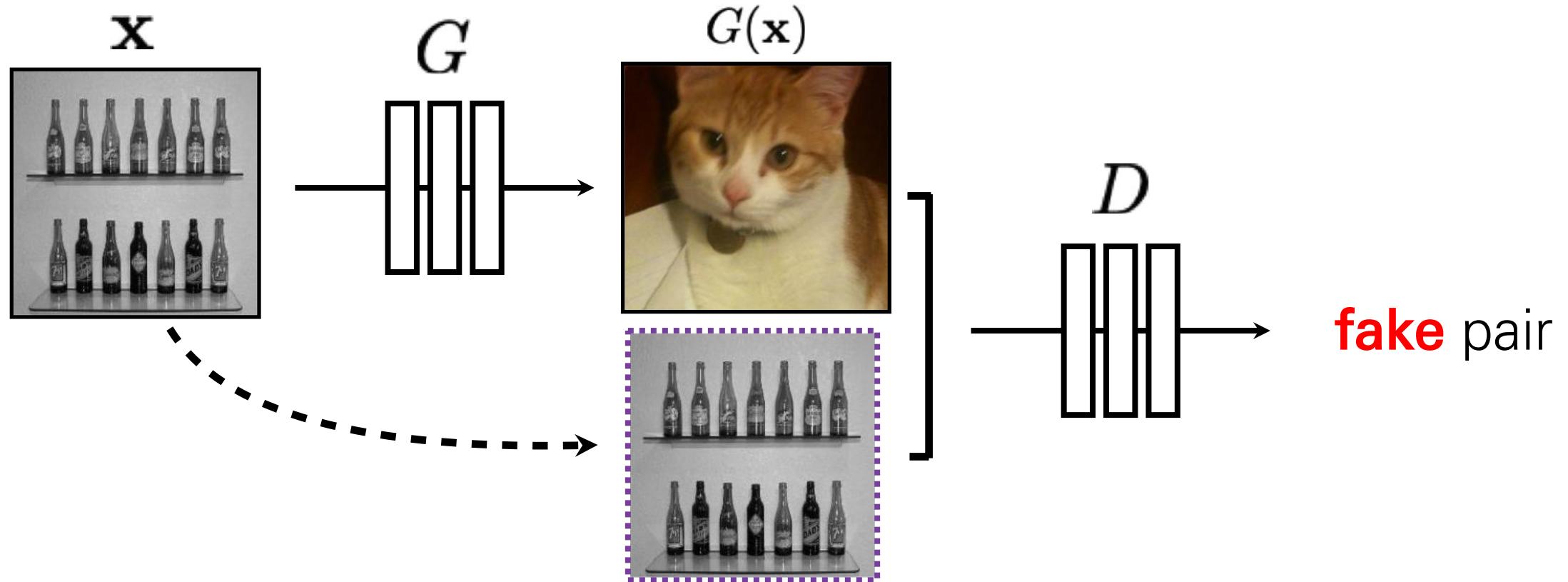


(Isola et al. 2016)

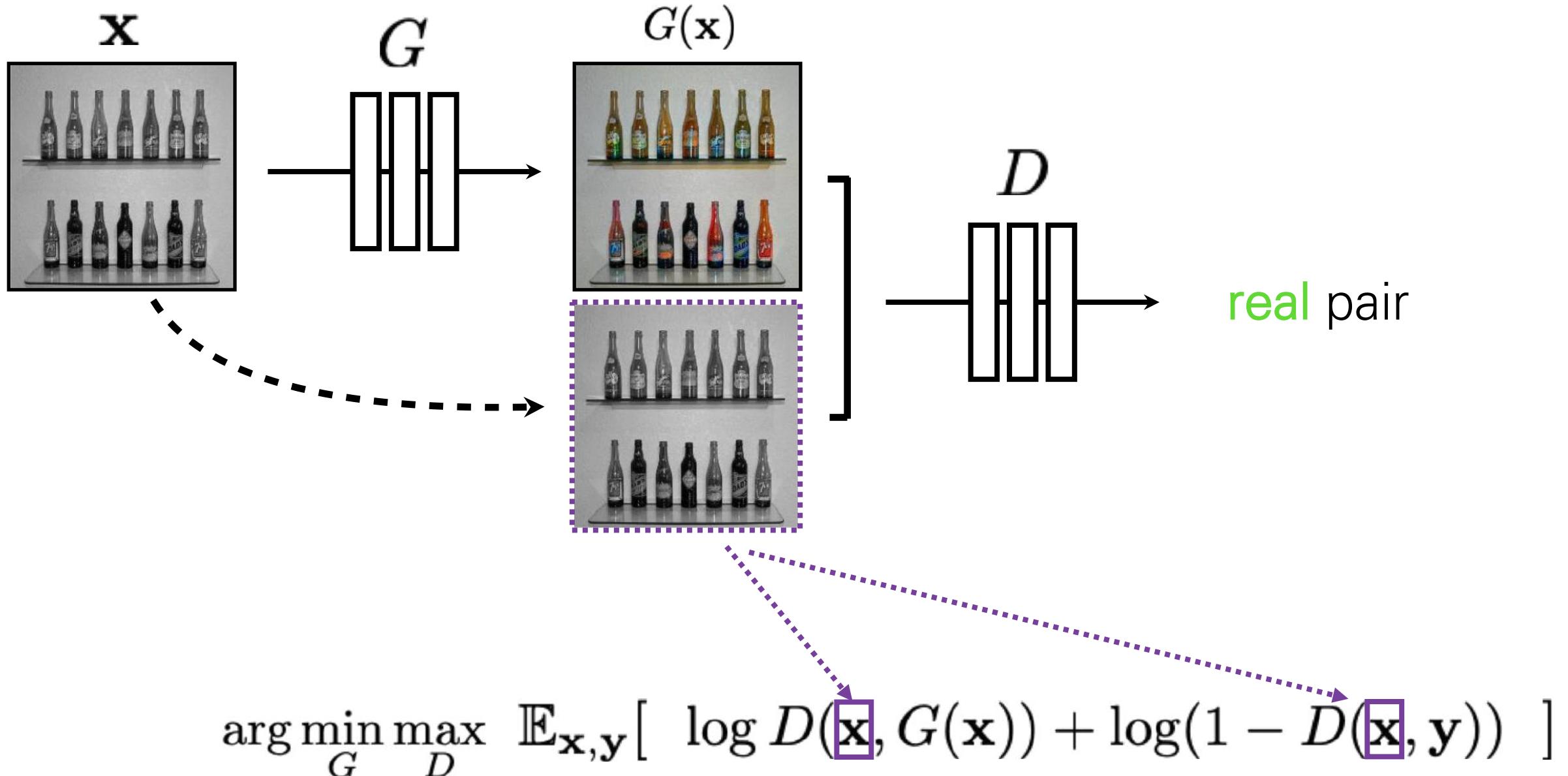


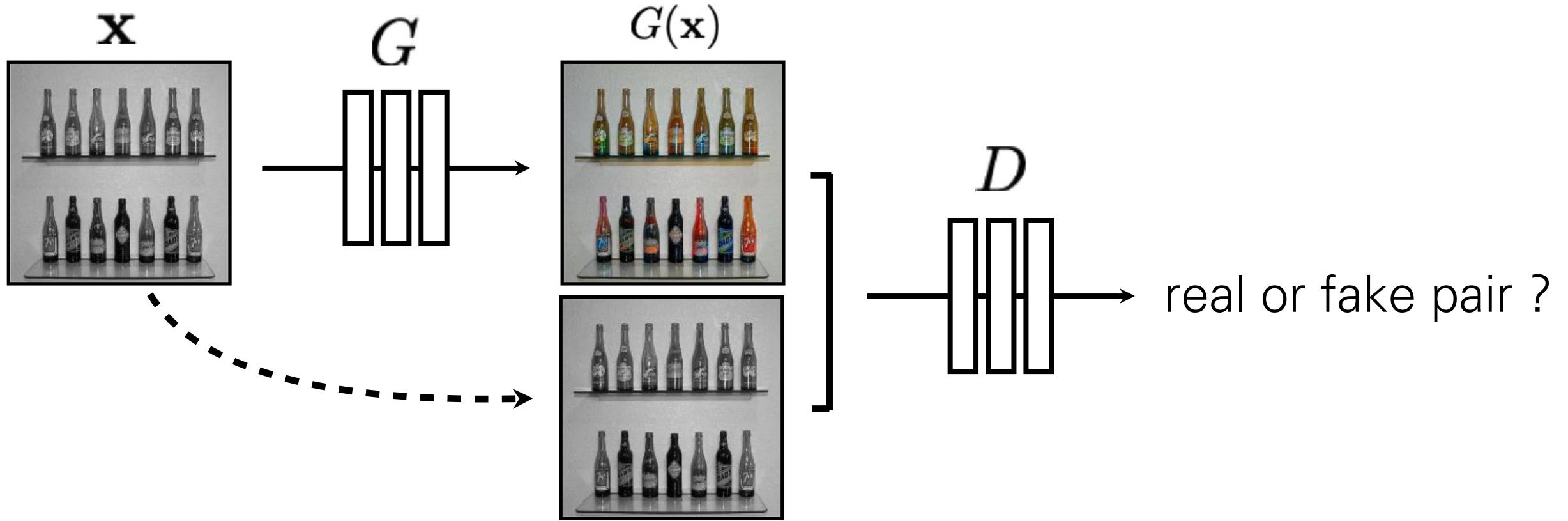
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$





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





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

BW → Color

Input



Output



Input



Output



Input

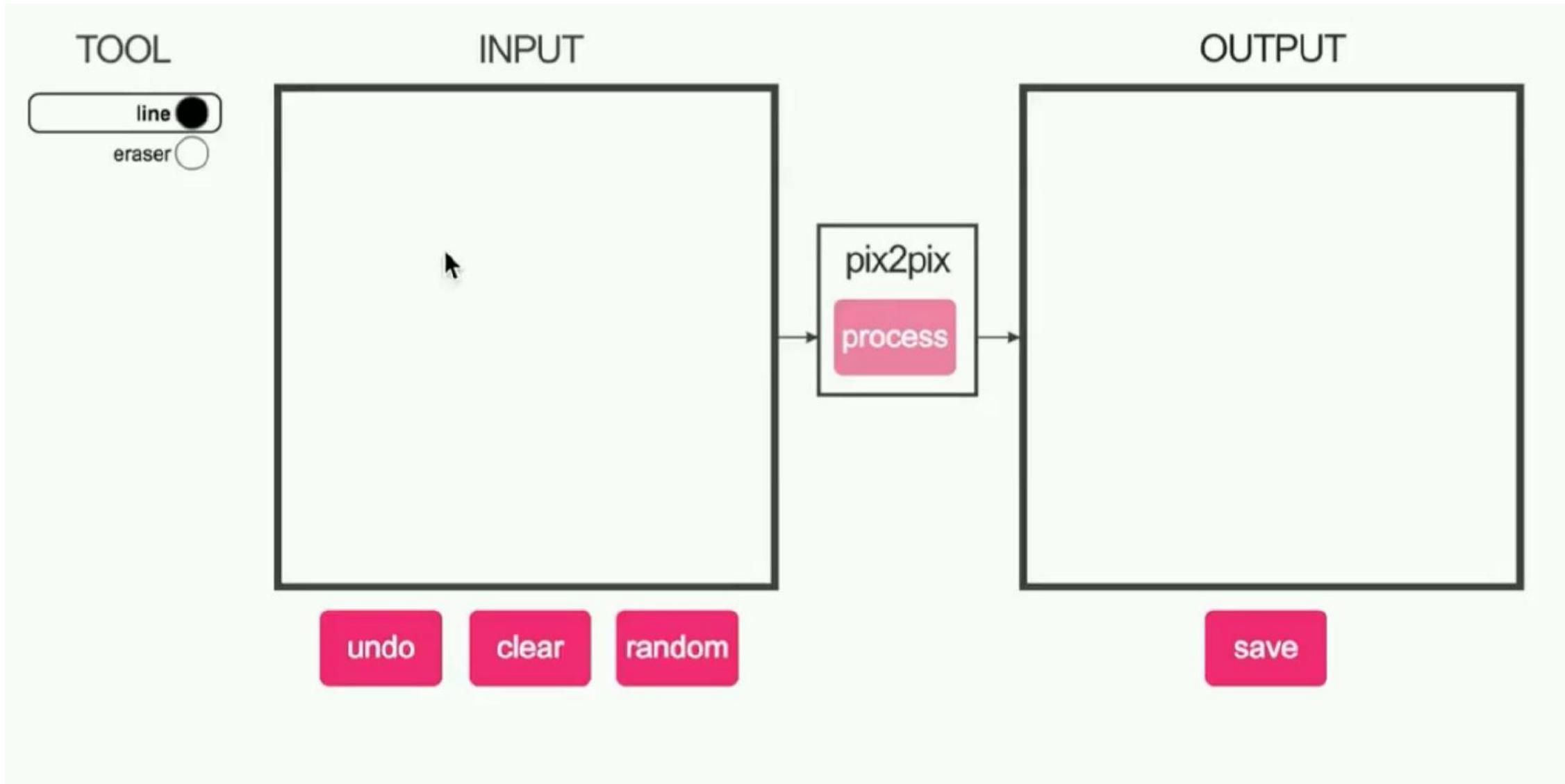


Output

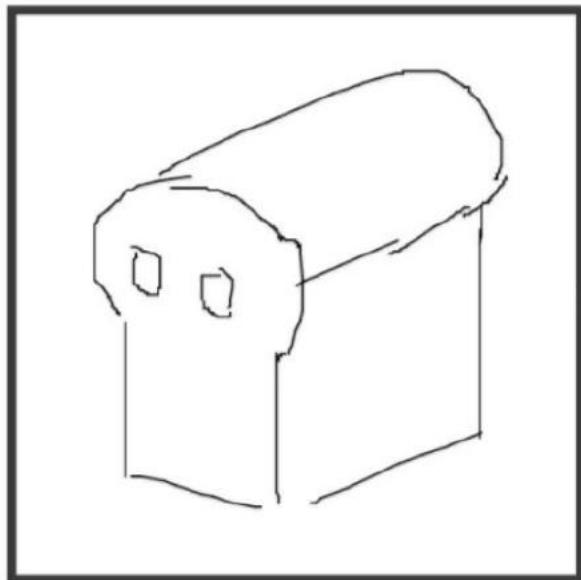


Data from [Russakovsky et al. 2015]

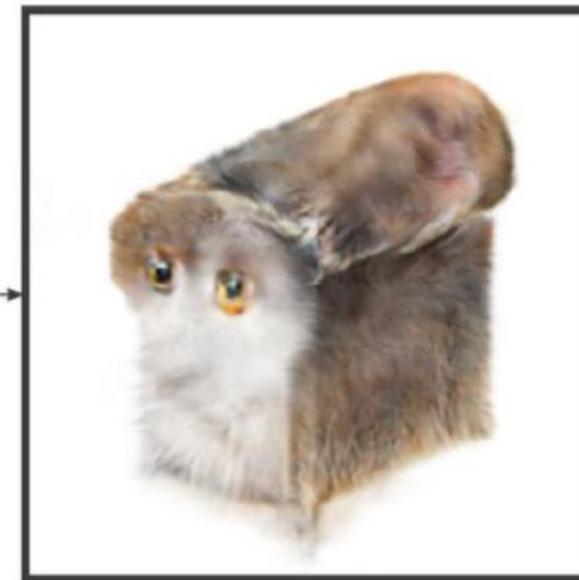
#edges2cats [Chris Hesse]



INPUT



OUTPUT

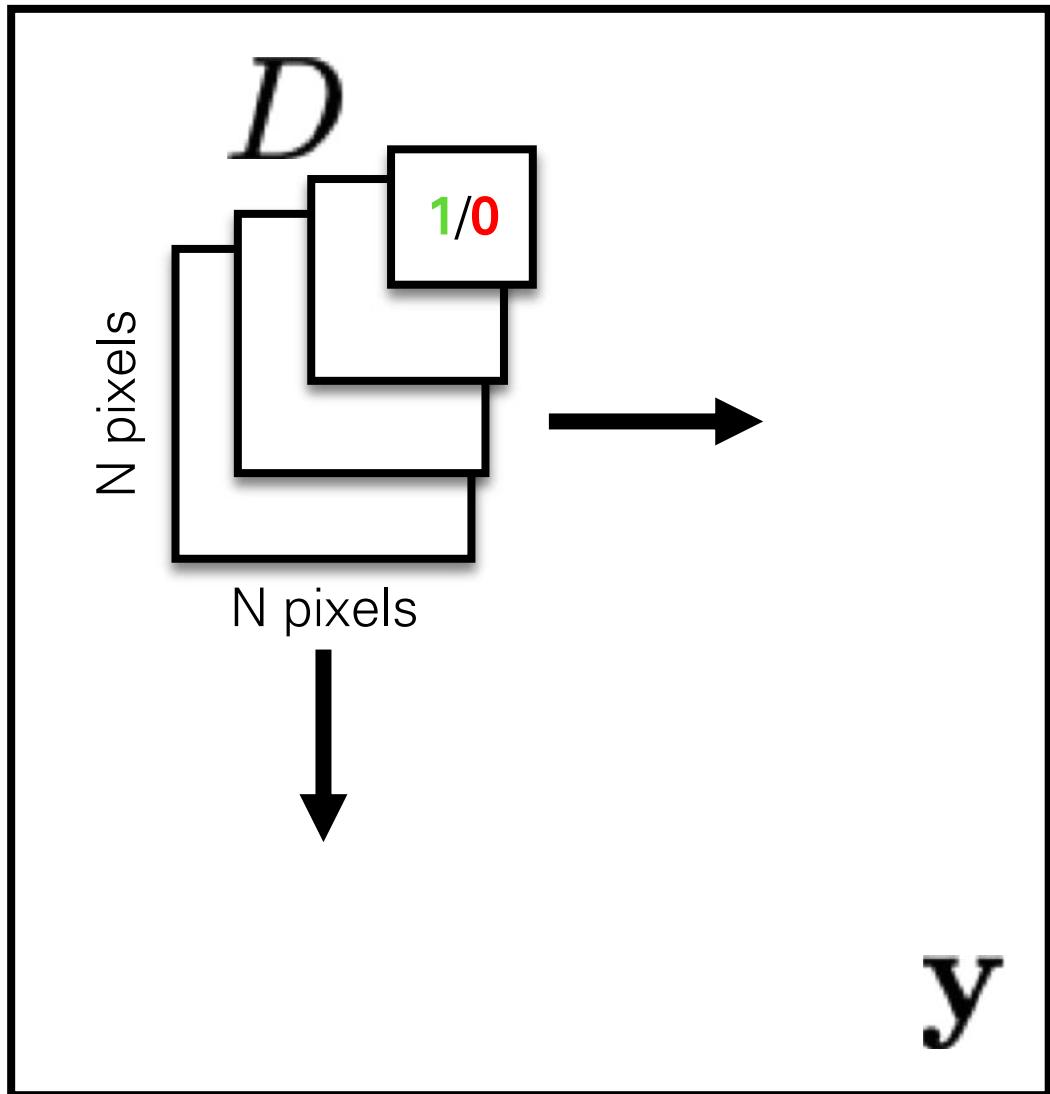


Ivy Tasi @ivymyt



Vitaly Vidmirov @vvid

Shrinking the capacity: Patch Discriminator



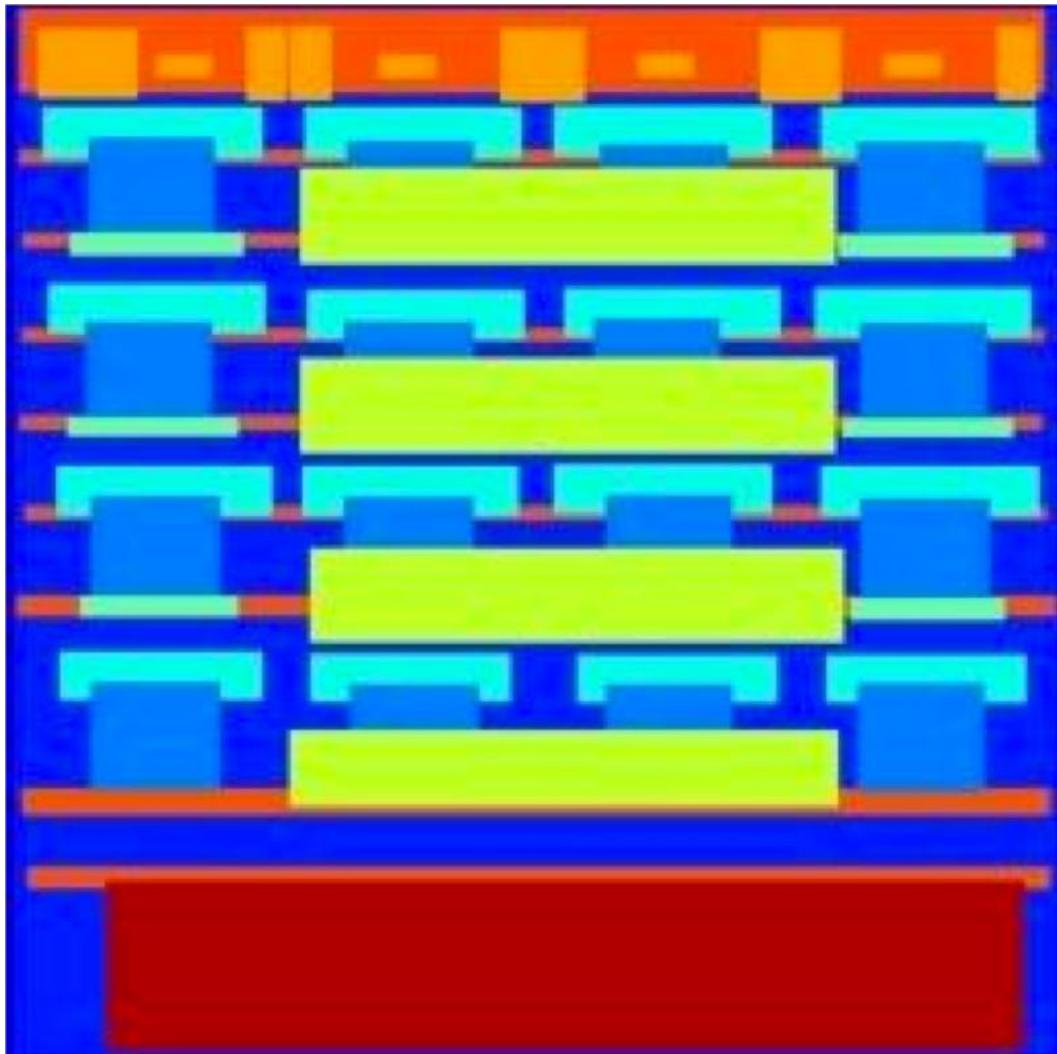
Rather than penalizing if output image looks fake, penalize if each overlapping patch in output looks fake

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

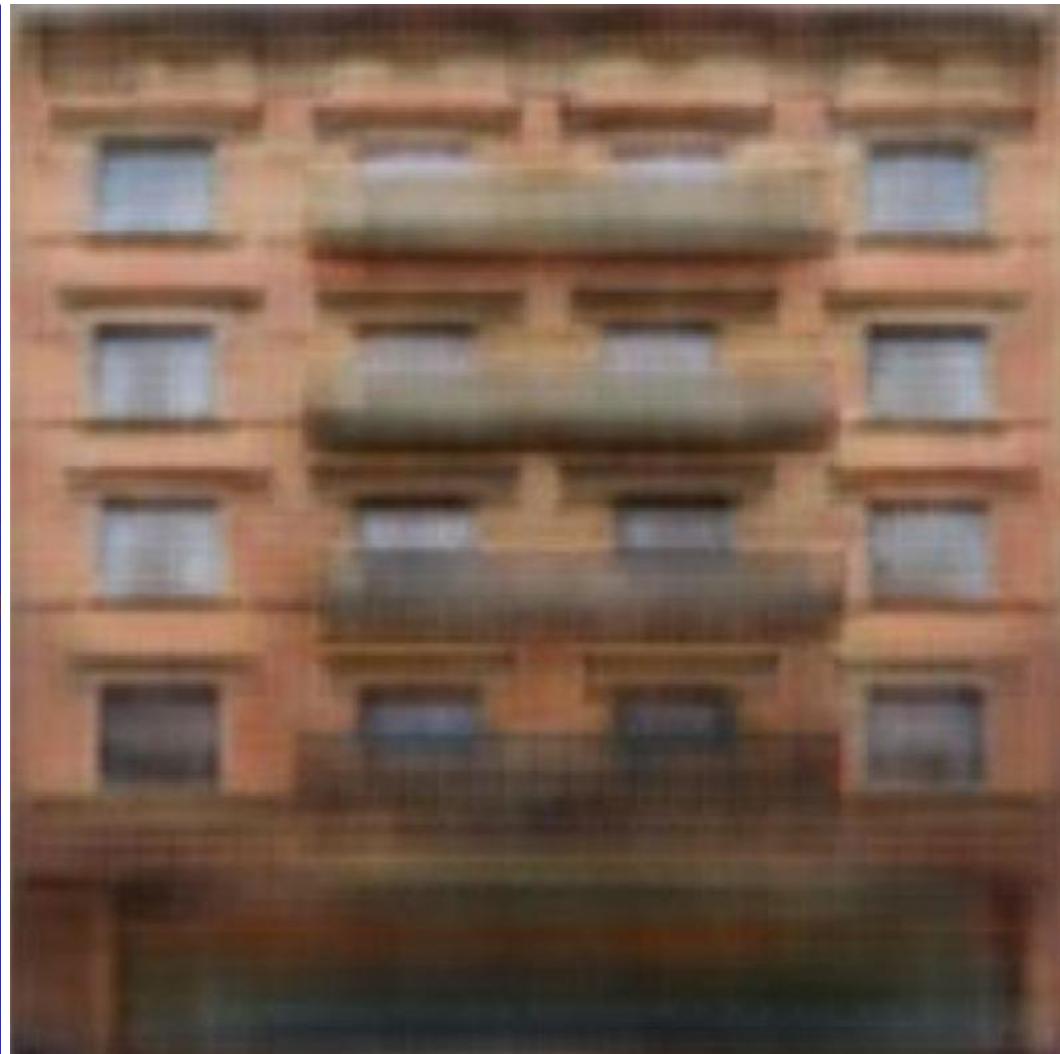
[Li & Wand 2016]
[Shrivastava et al. 2017]
[Isola et al. 2017]

Labels → Facades

Input



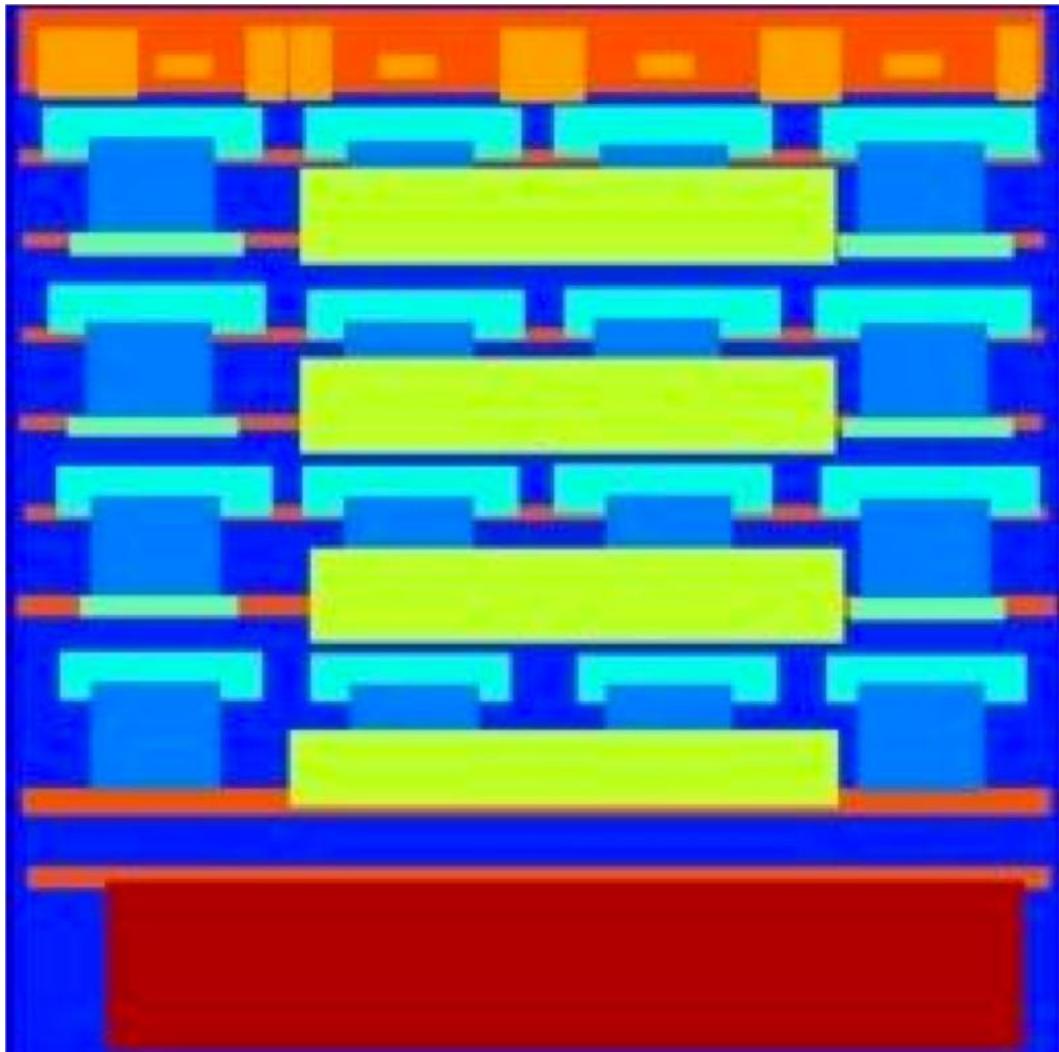
1x1 Discriminator



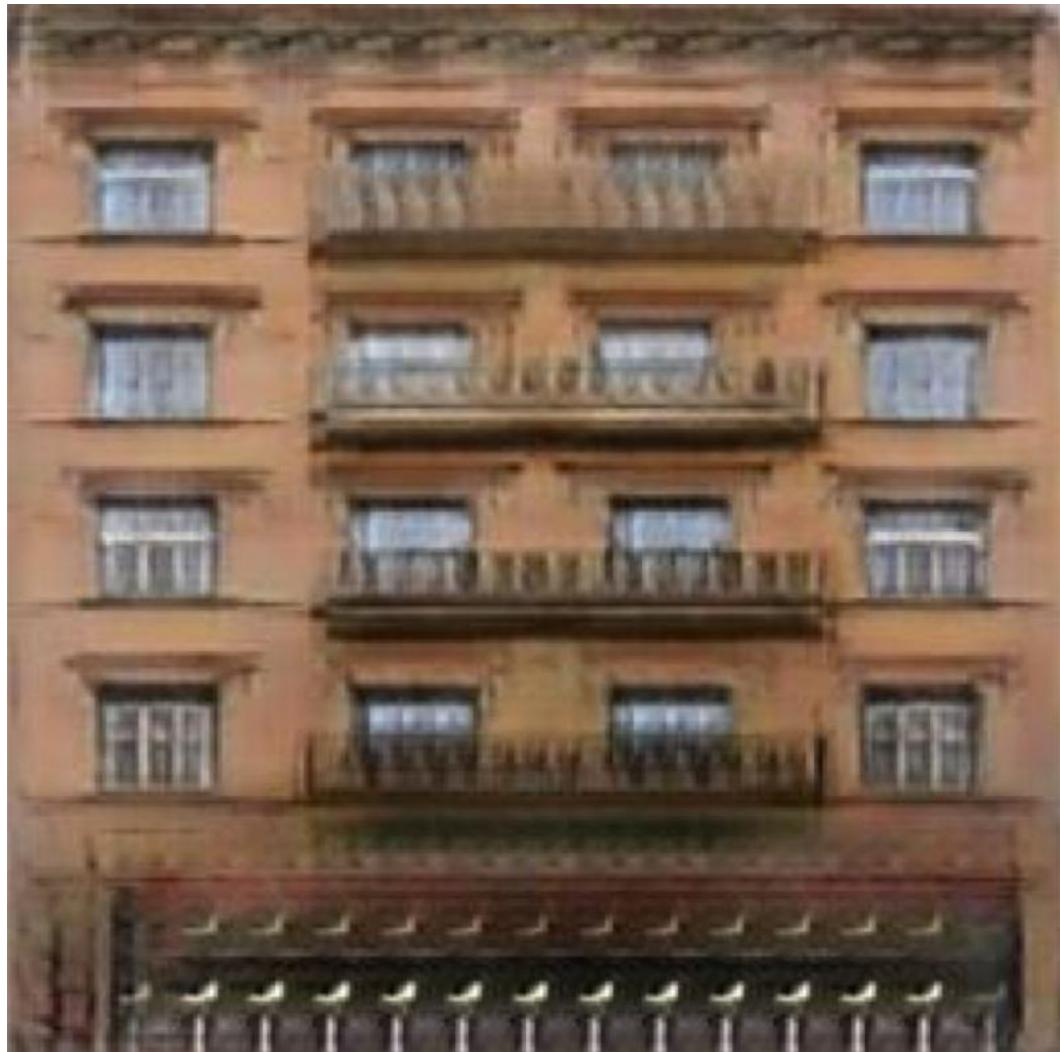
Data from [Tylecek, 2013]

Labels → Facades

Input



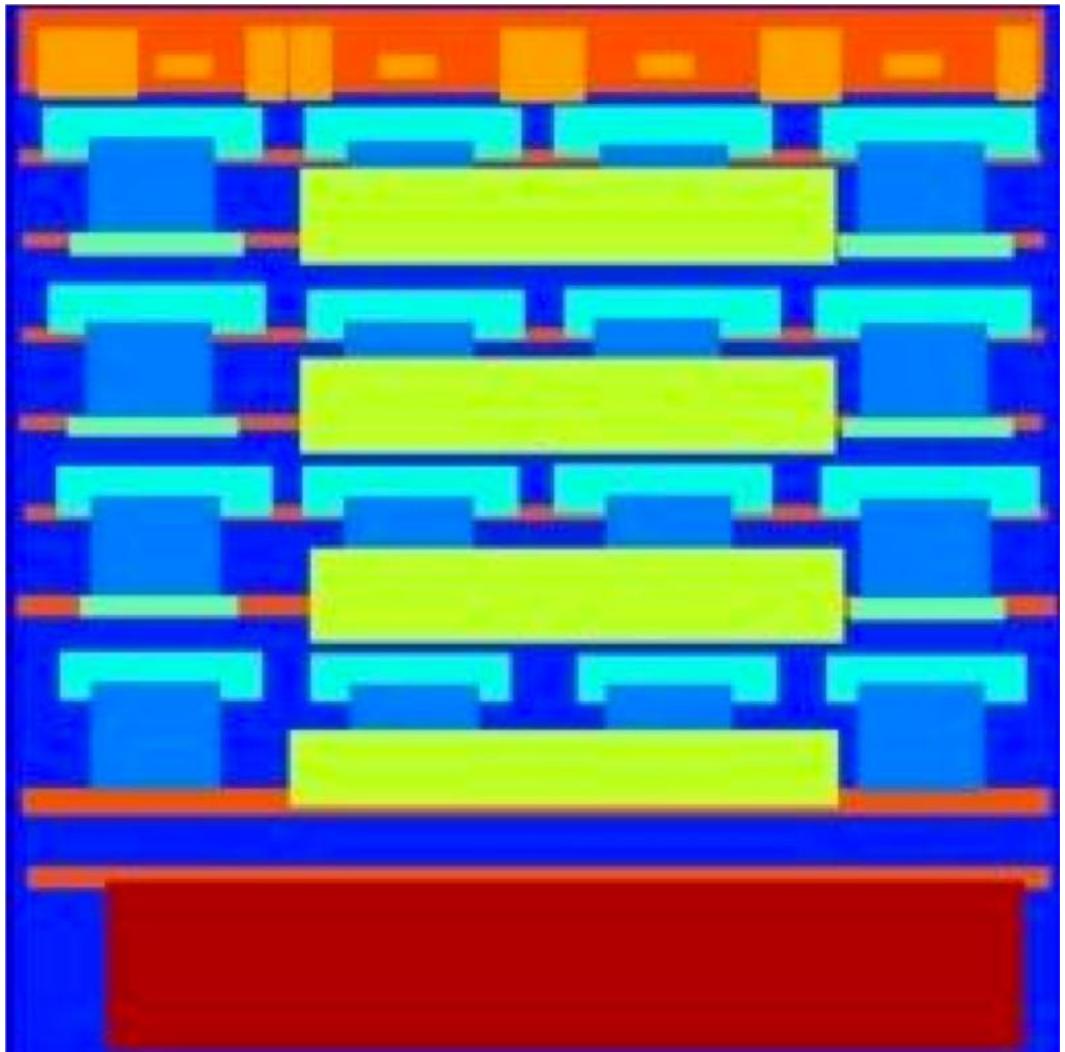
16x16 Discriminator



Data from [Tylecek, 2013]

Labels → Facades

Input



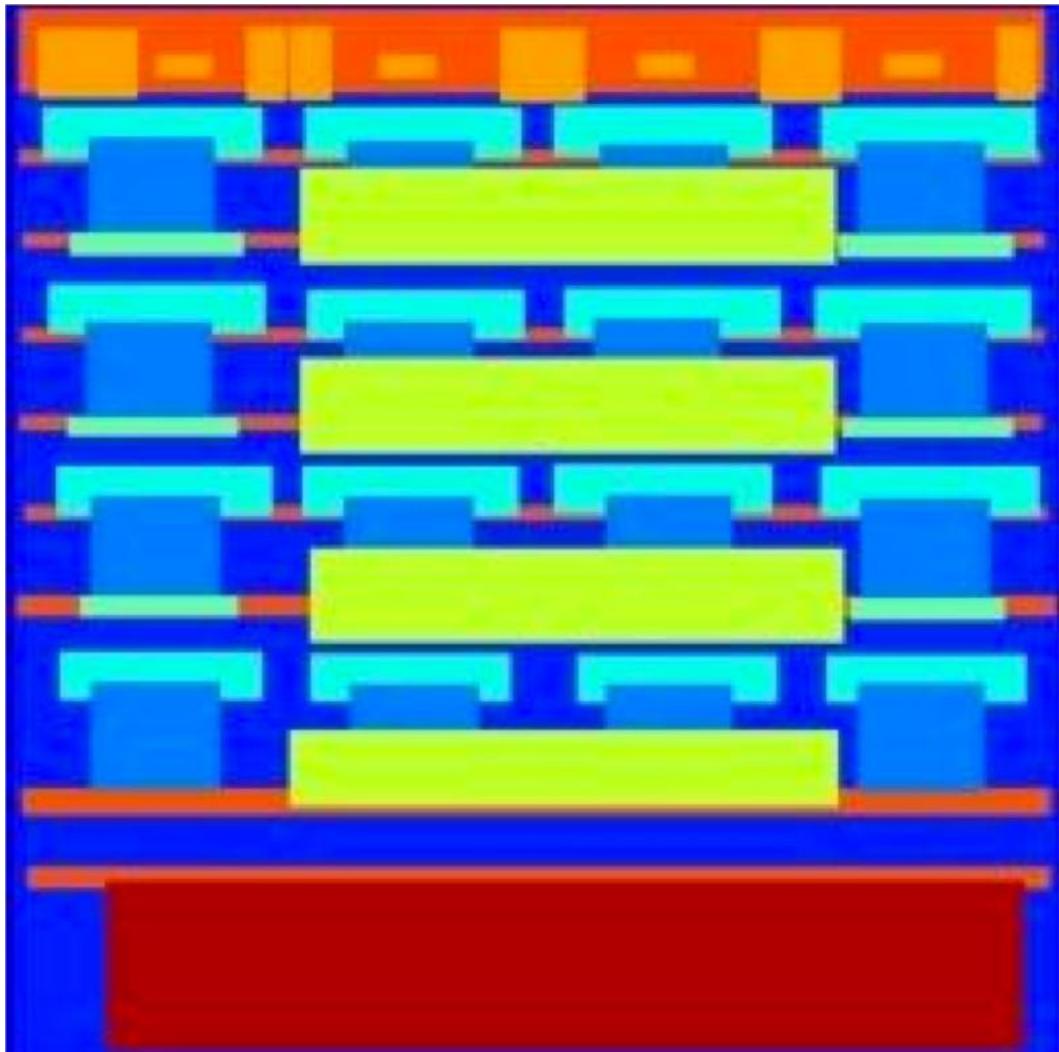
70x70 Discriminator



Data from [Tylecek, 2013]

Labels → Facades

Input

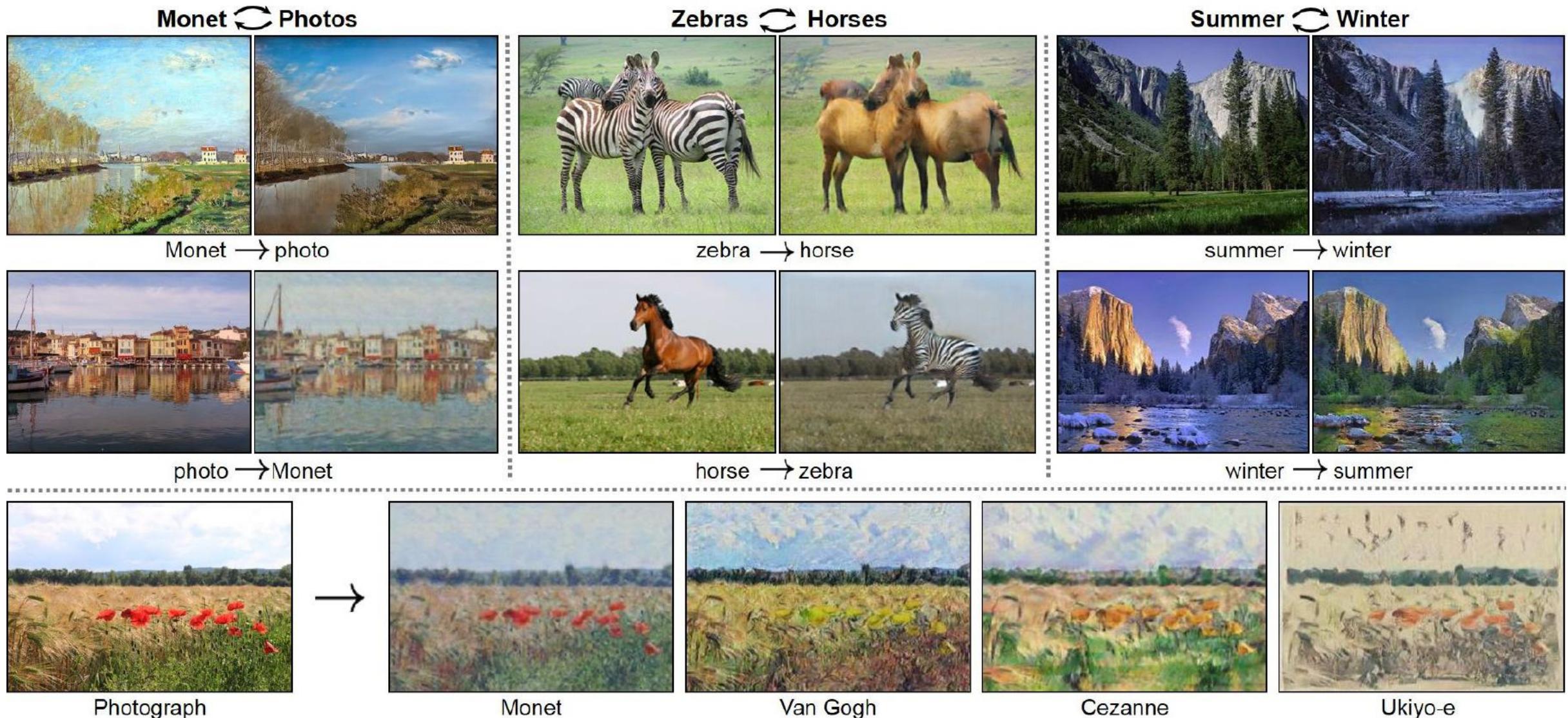


Full image Discriminator



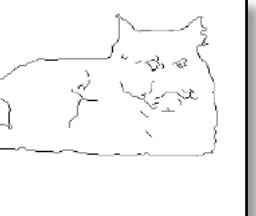
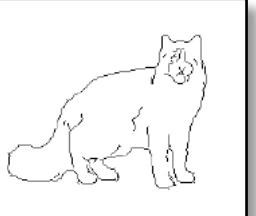
Data from [Tylecek, 2013]

Pix2Pix w/o input-output pairs

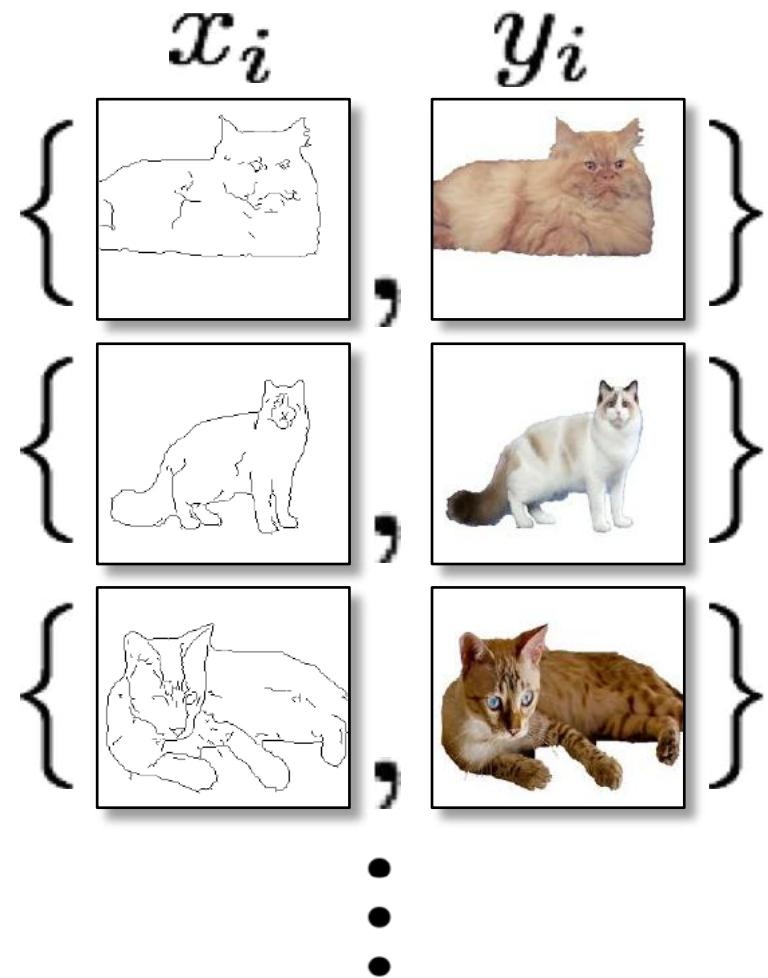


(Zhu et al. 2017)

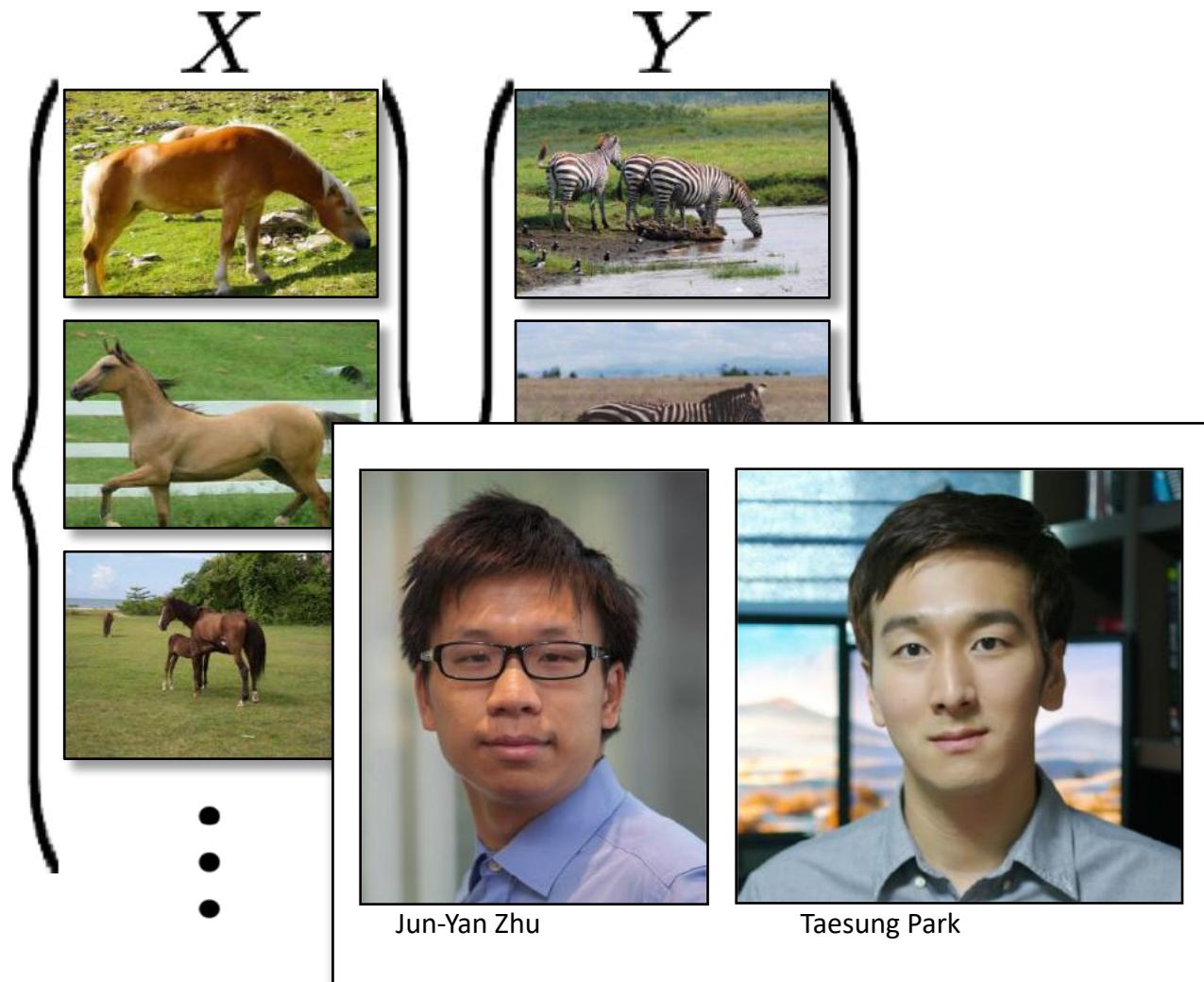
Paired data

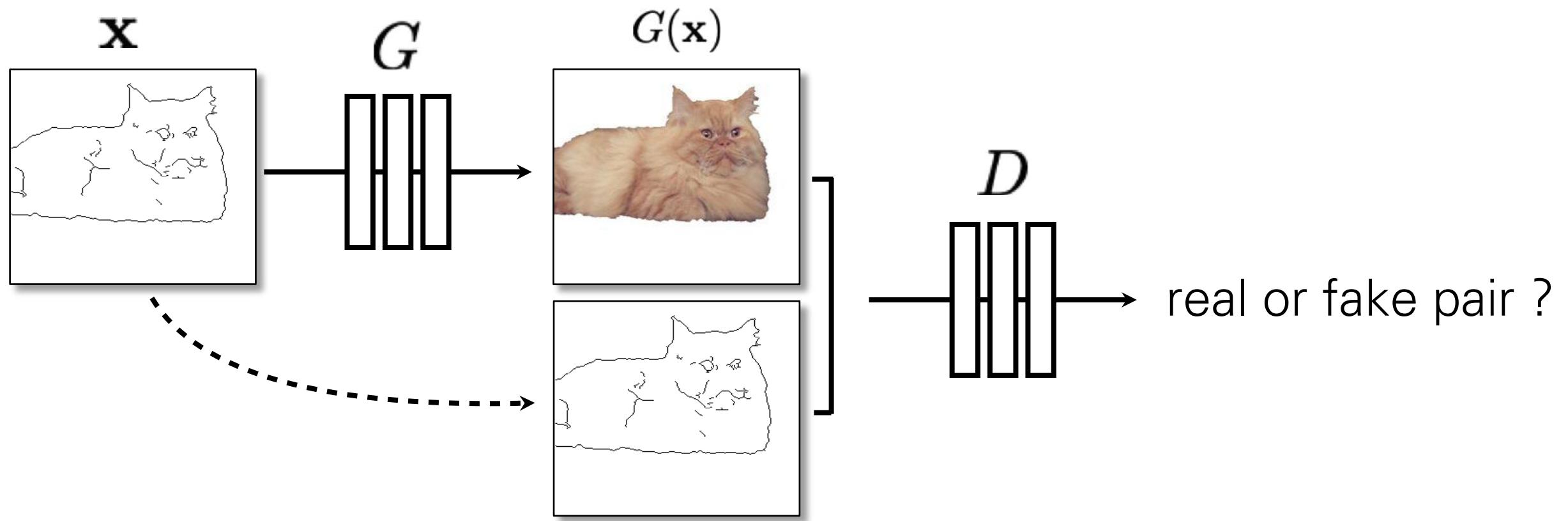
x_i	y_i
{  , }	{  }
{  , }	{  }
{  , }	{  }
⋮	

Paired data

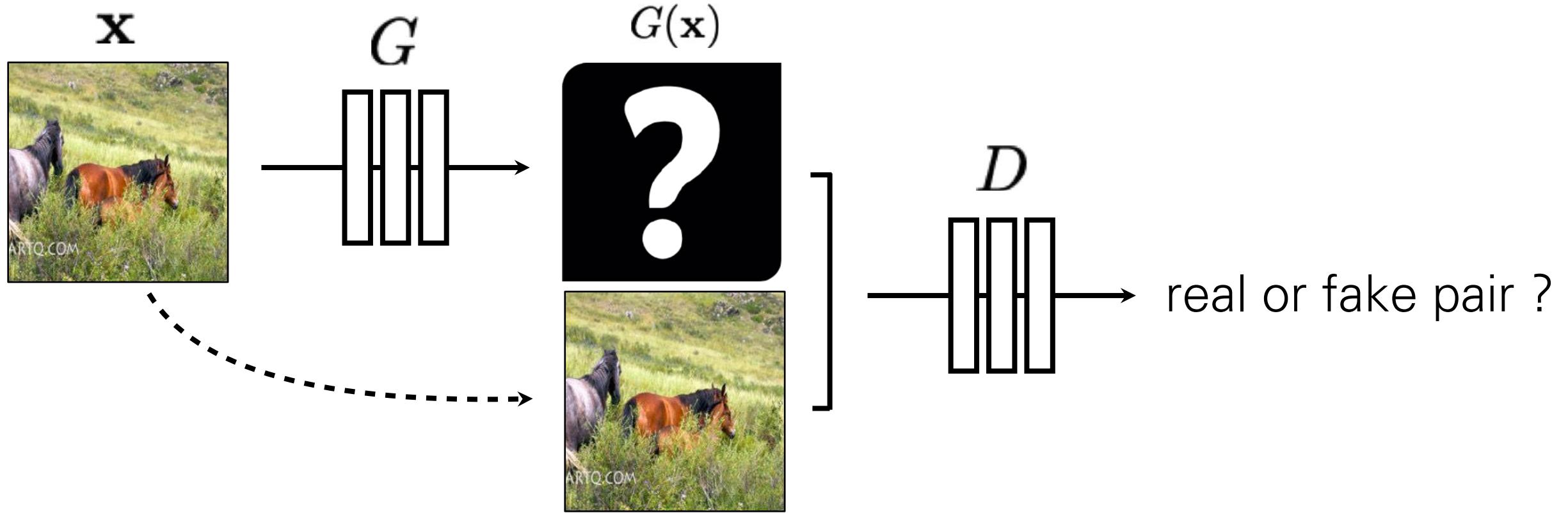


Unpaired data



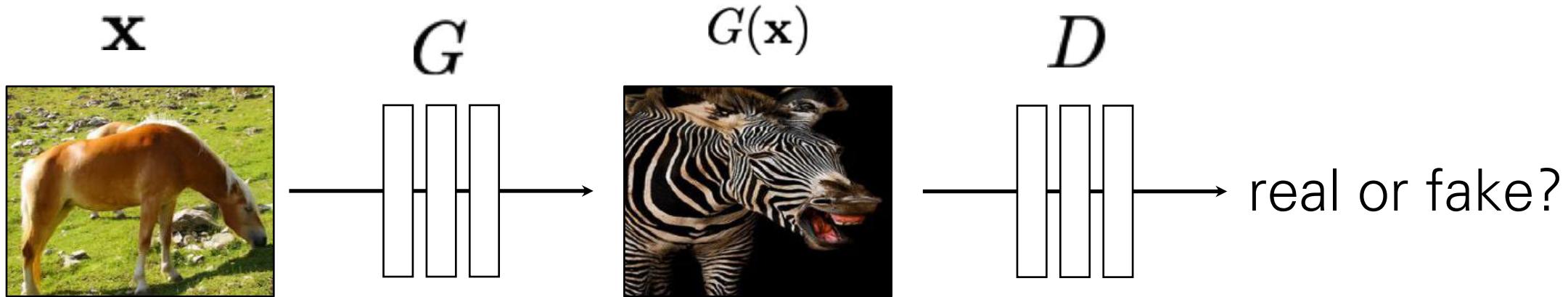


$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))]$$



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

No input-output pairs!

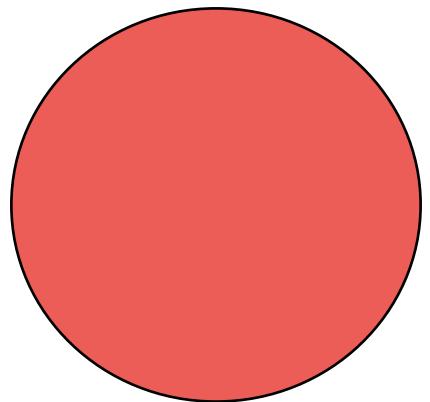


$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$

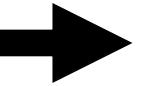
Usually loss functions check if output matches a target instance

GAN loss checks if output is part of an admissible set

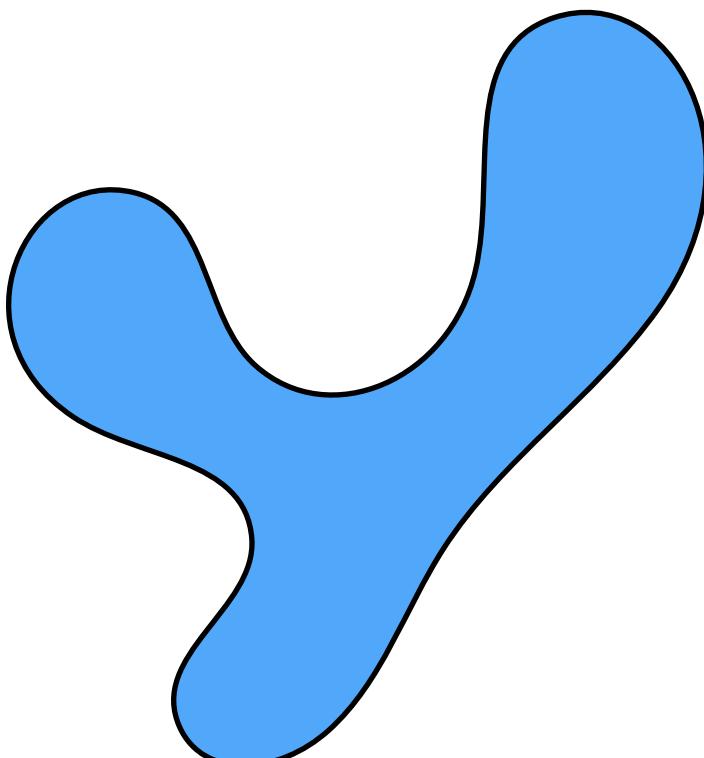
Gaussian



z

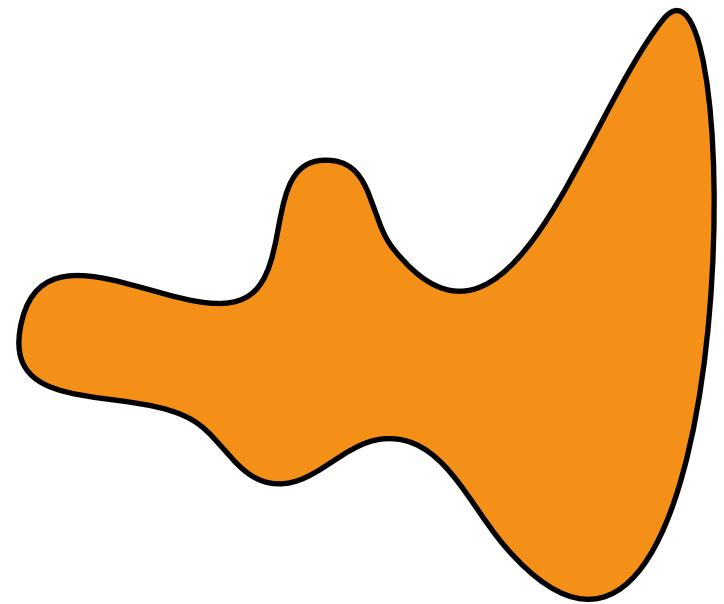


Target distribution



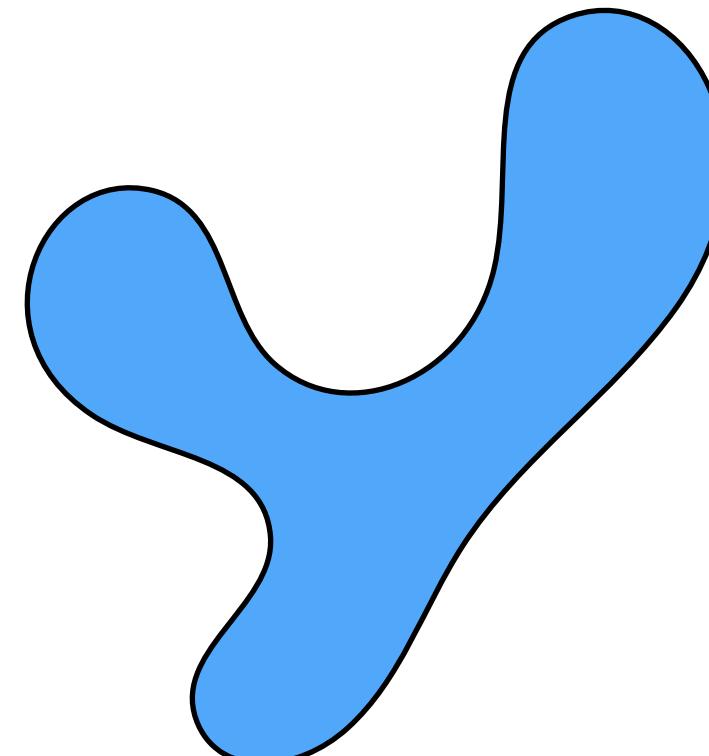
Y

Horses

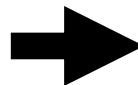


X

Zebras



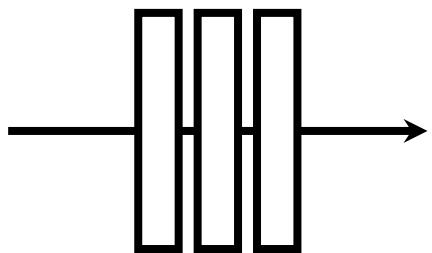
Y



x



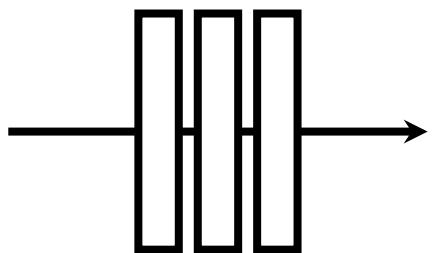
G



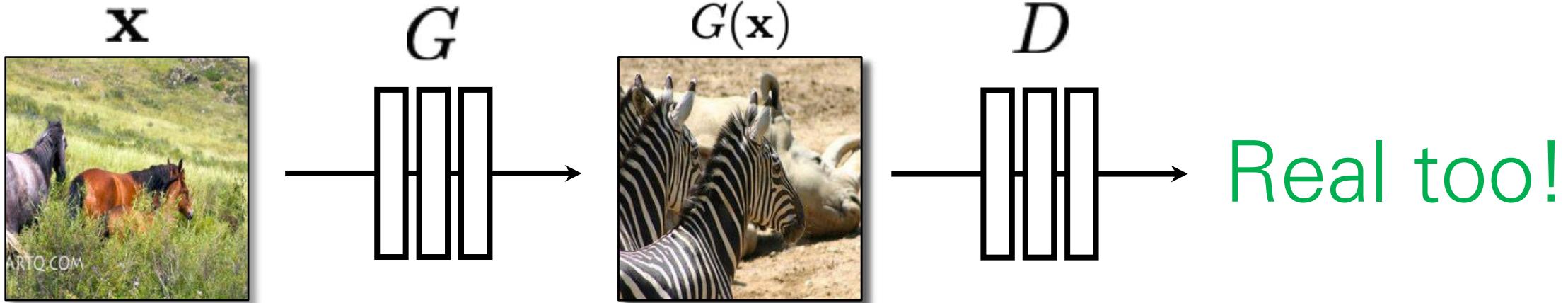
G(x)



D

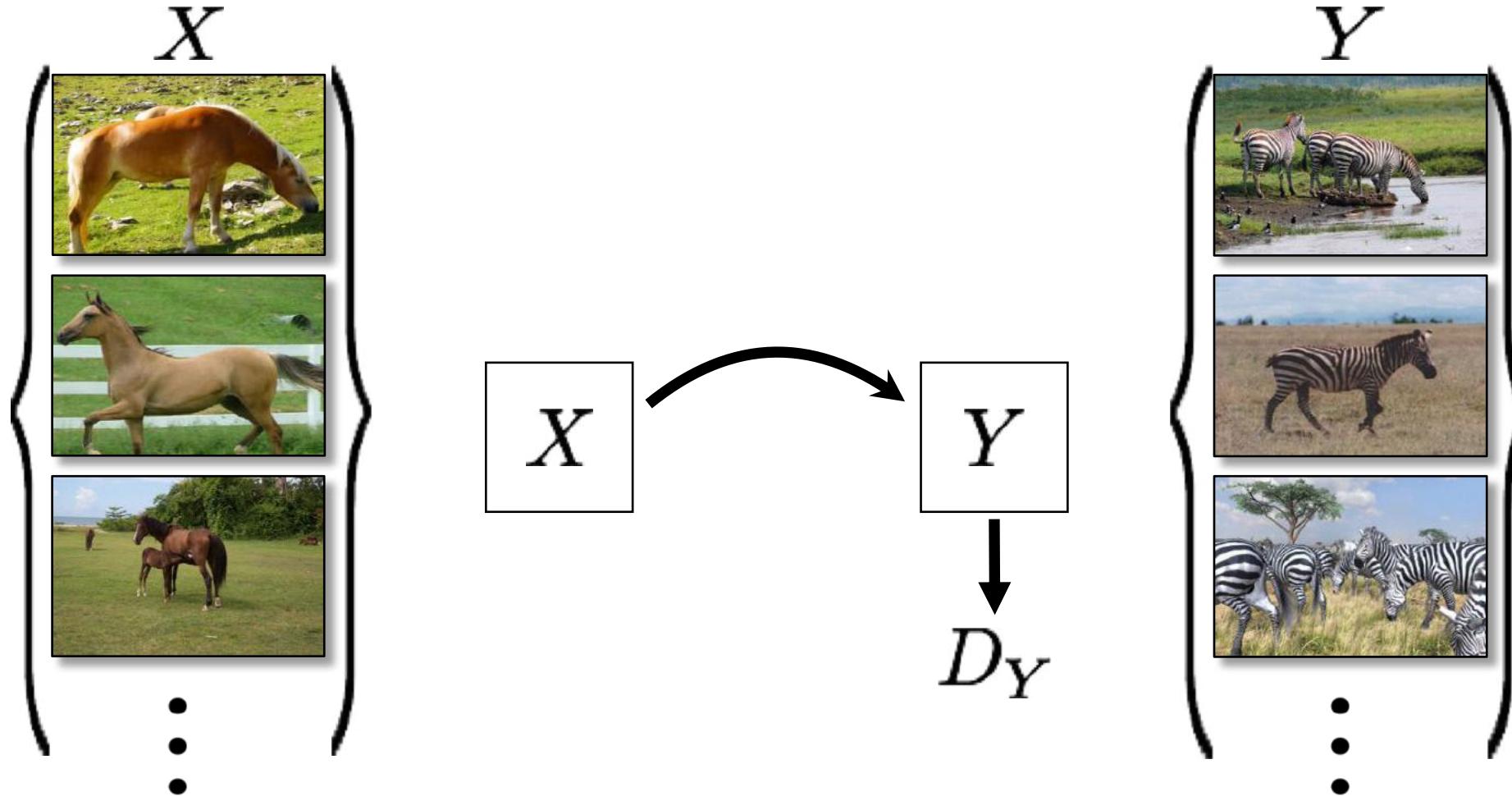


Real!



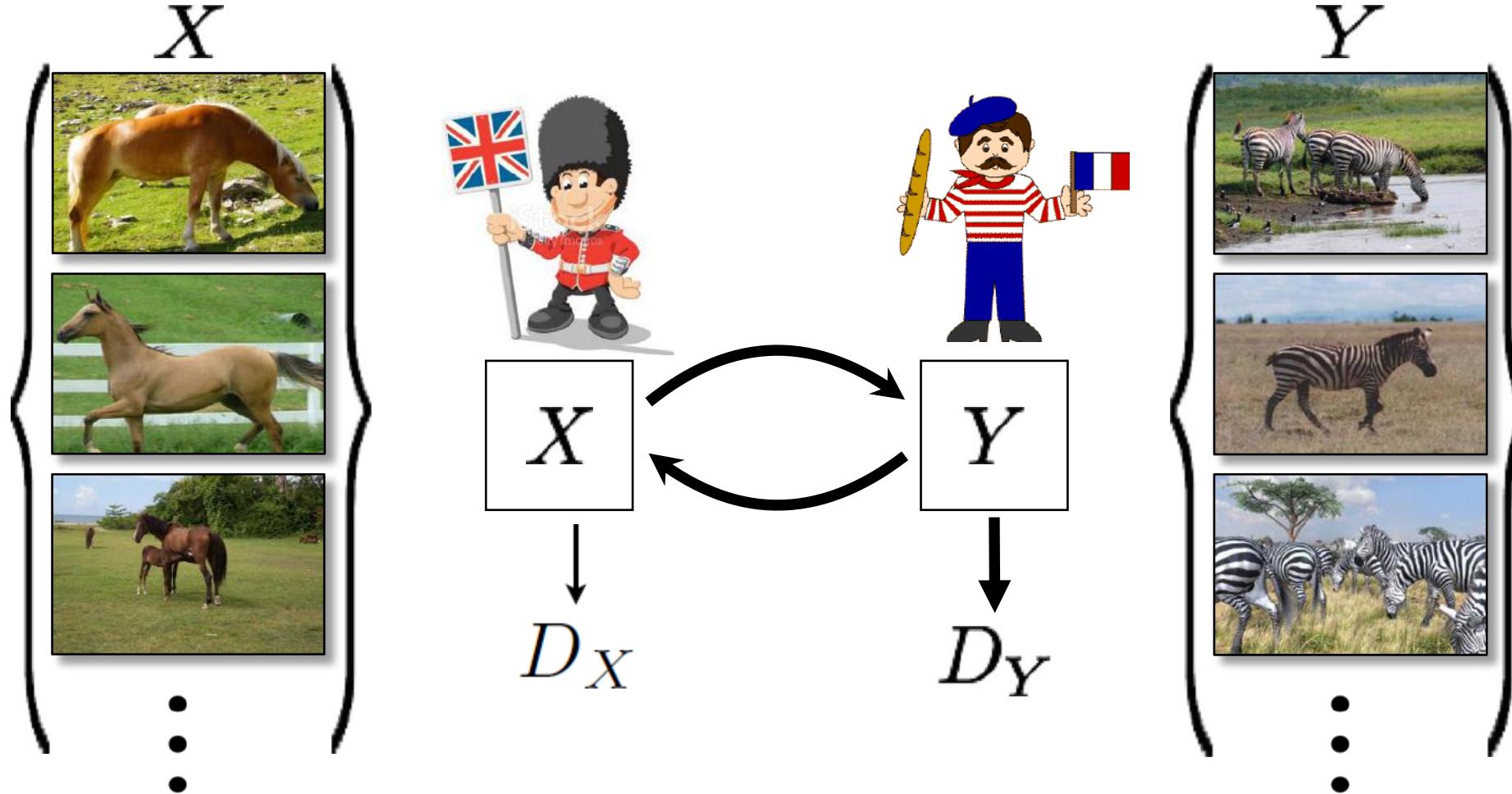
Nothing to force output to correspond to input

Cycle-Consistent Adversarial Networks

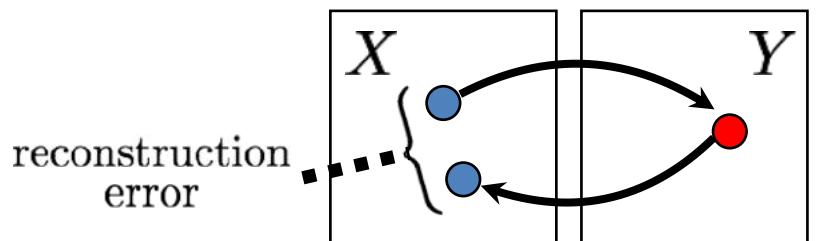
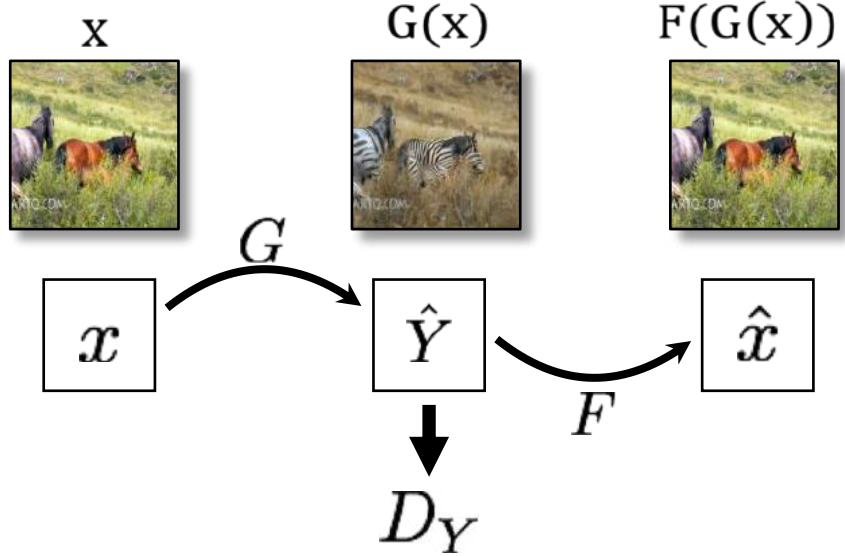


[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

Cycle-Consistent Adversarial Networks

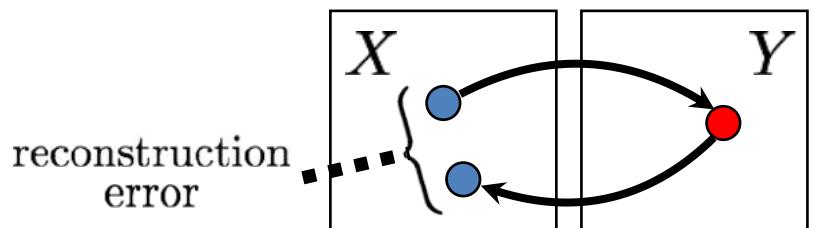
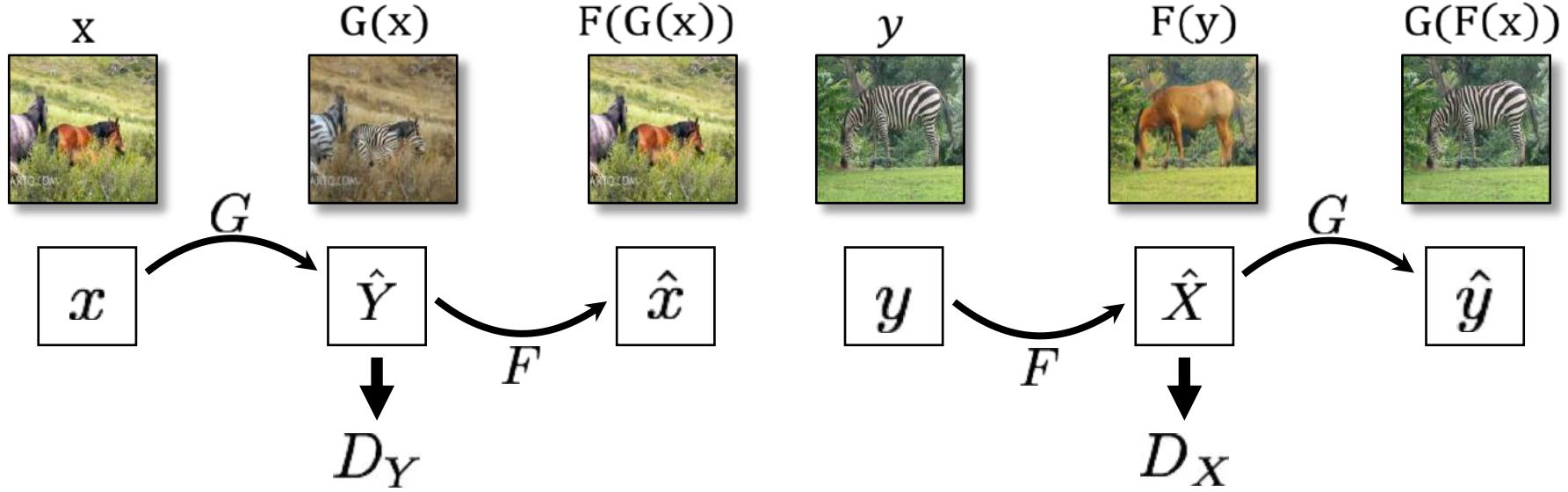


Cycle Consistency Loss

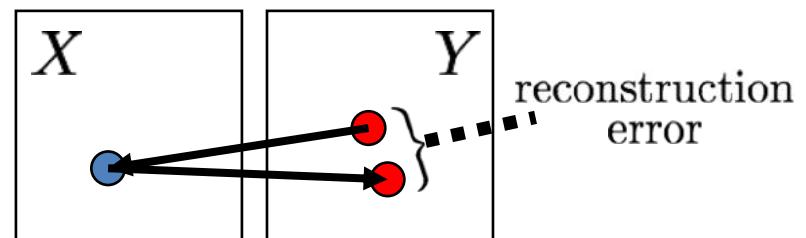


$$\|F(G(x)) - x\|_1$$

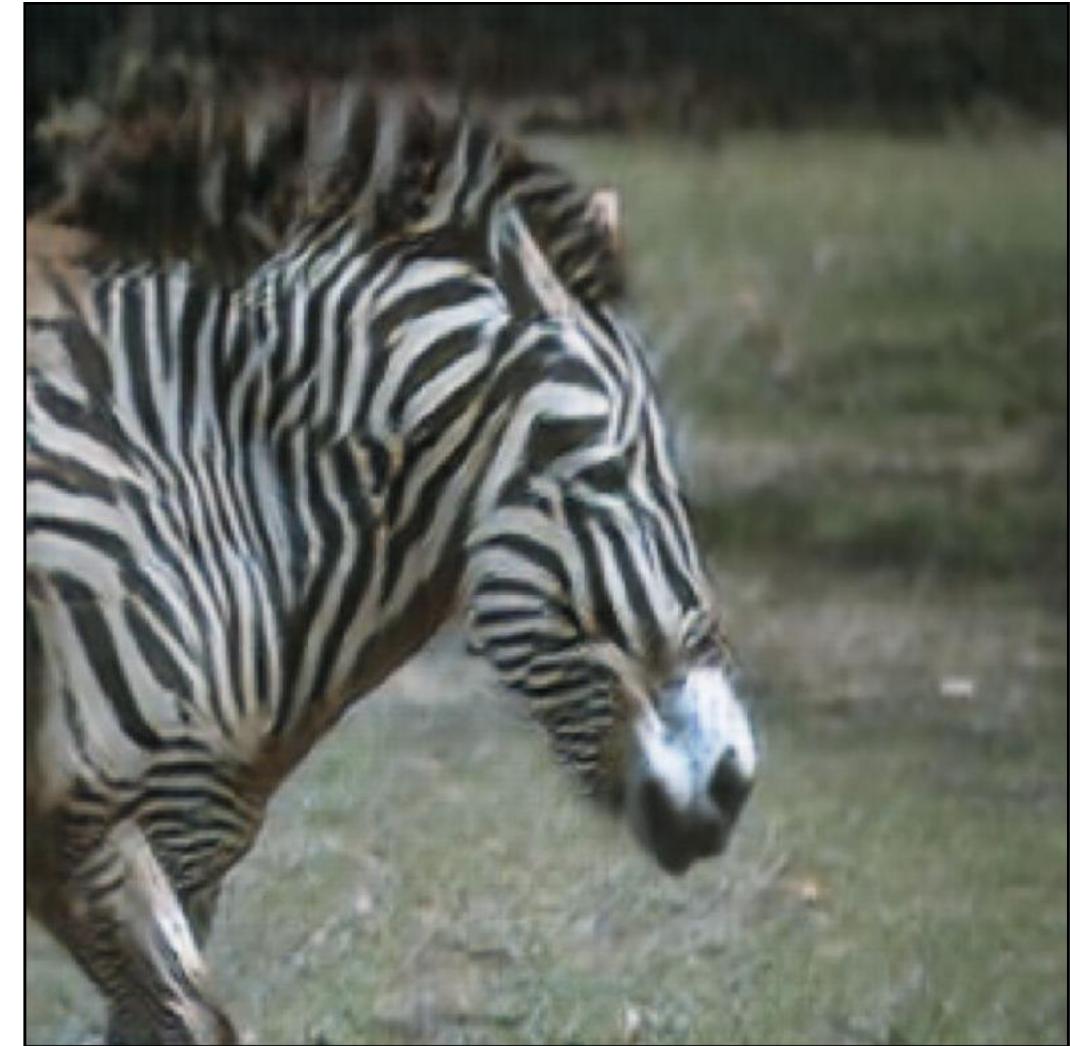
Cycle Consistency Loss



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



$$\|G(F(y)) - y\|_1$$





Collection Style Transfer



Photograph
@ Alexei Efros



Monet



Van Gogh



Cezanne



Ukiyo-e

Input



Monet



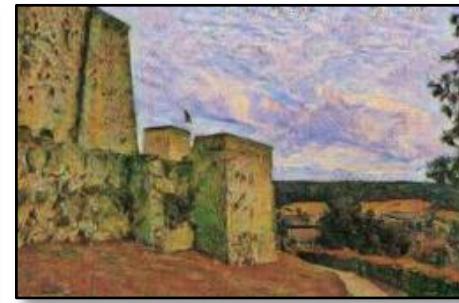
Van Gogh



Cezanne



Ukiyo-e



Monet's paintings → photos



Monet's paintings → photos

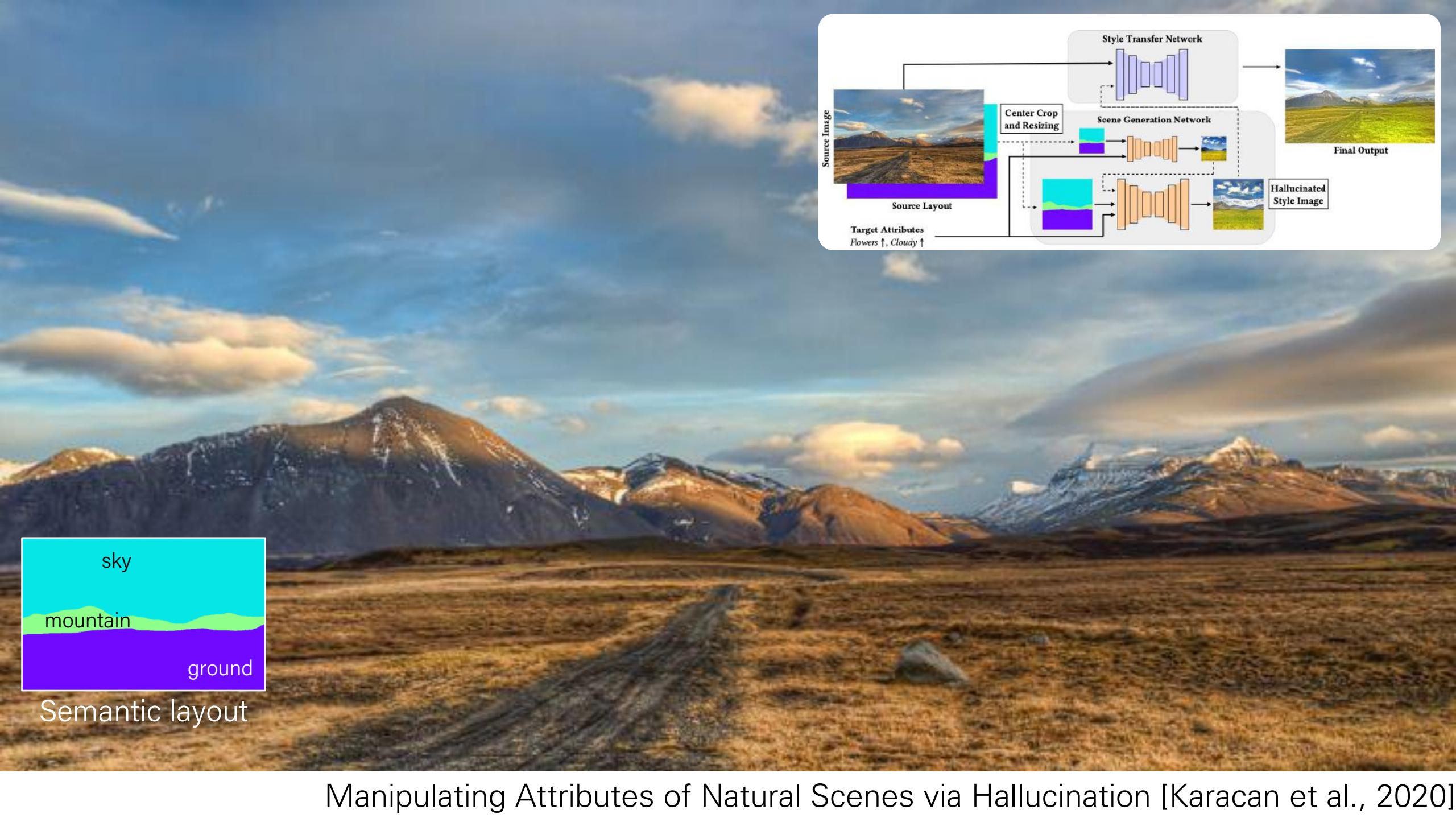


Failure case



Failure case







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



night

prediction



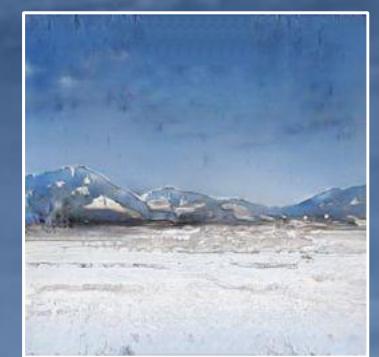
sunset



prediction



snow



prediction



winter



prediction

Spring and clouds



prediction





Moist, rain and fog



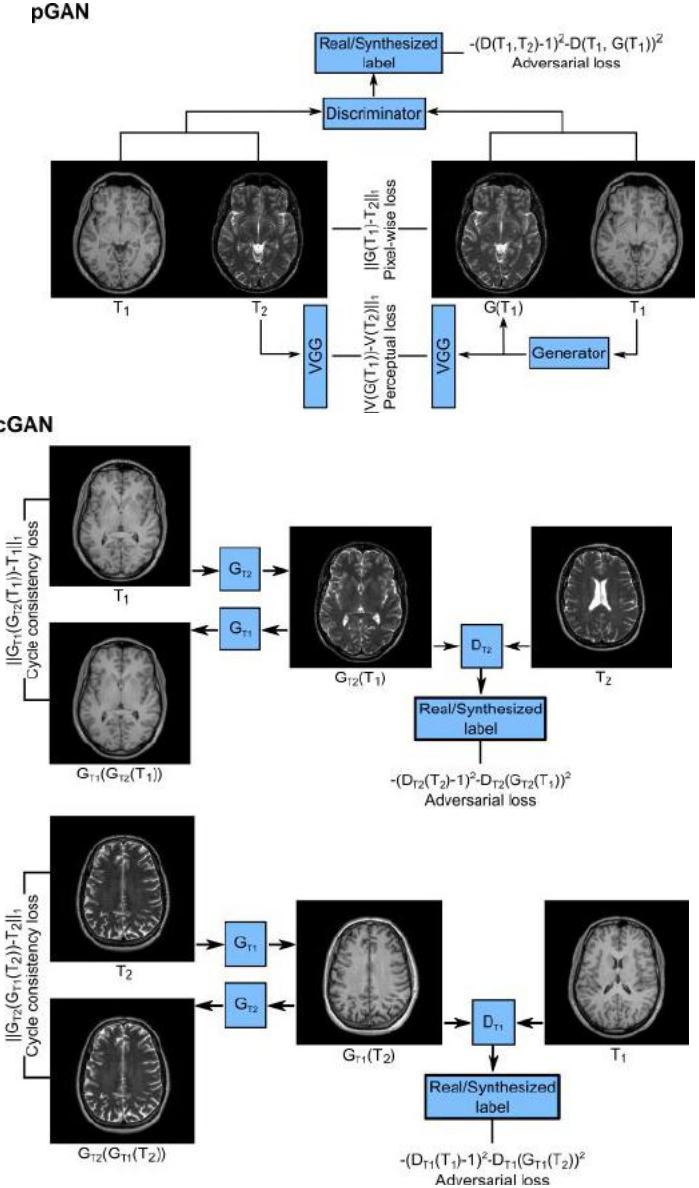
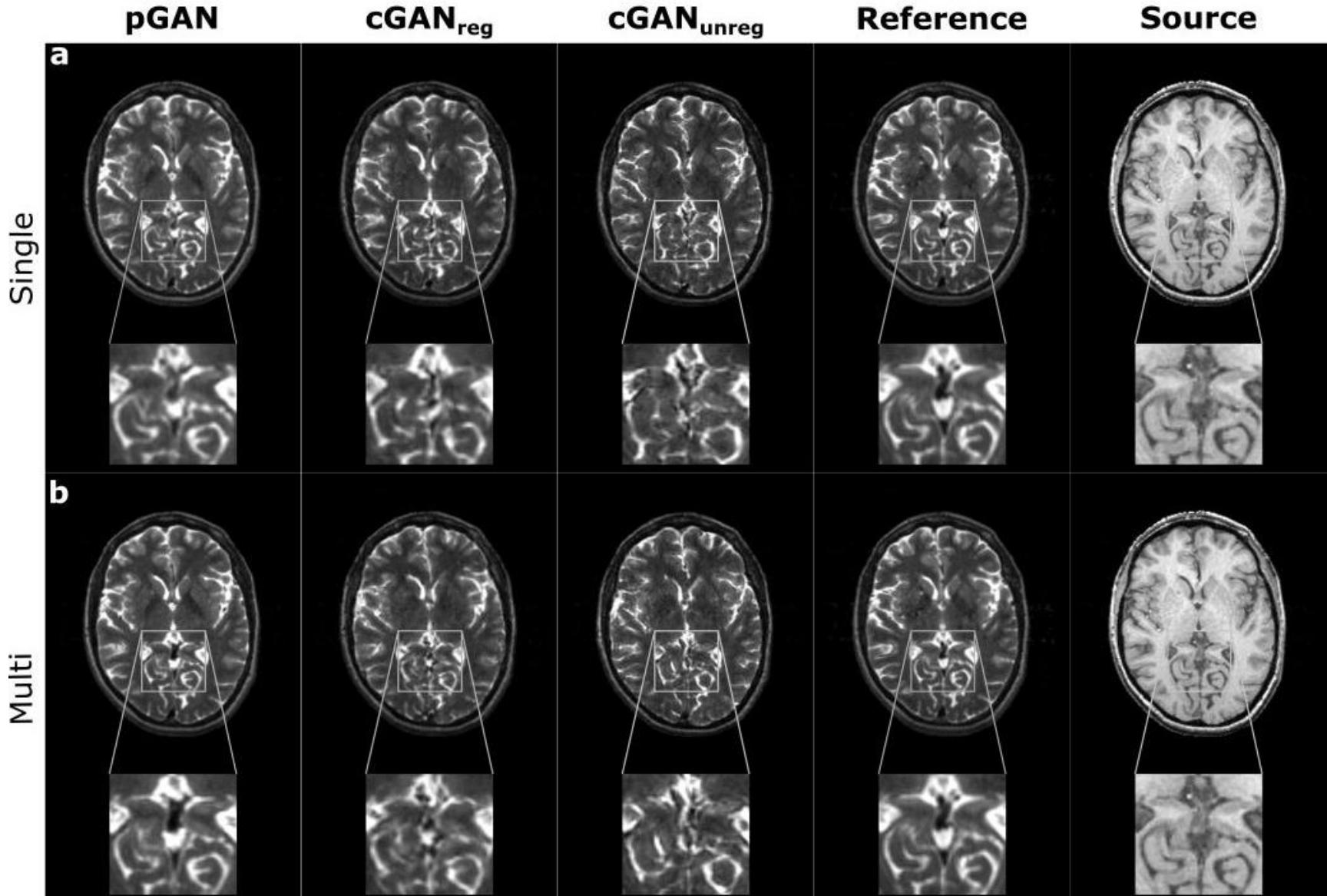
prediction



flowers



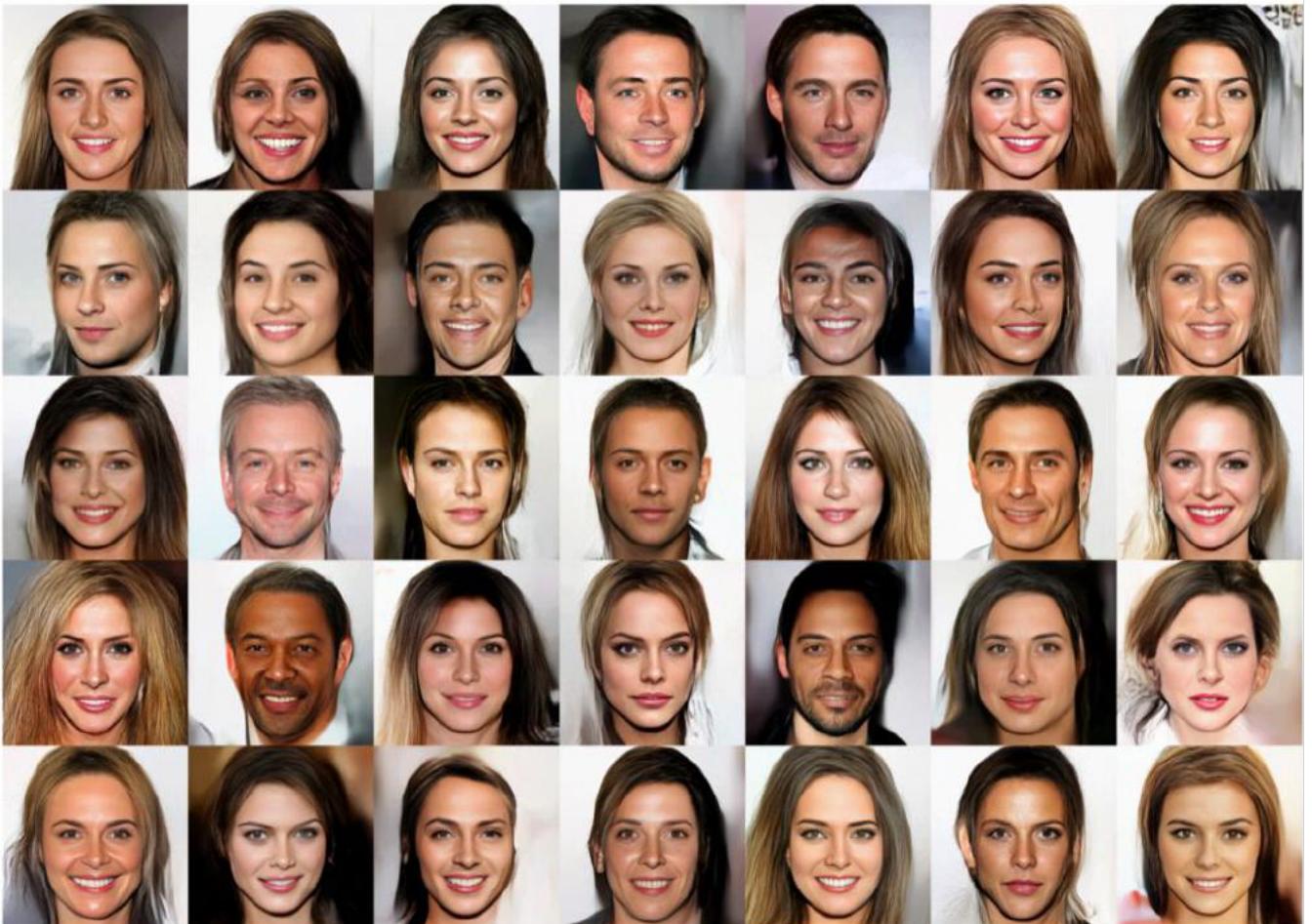
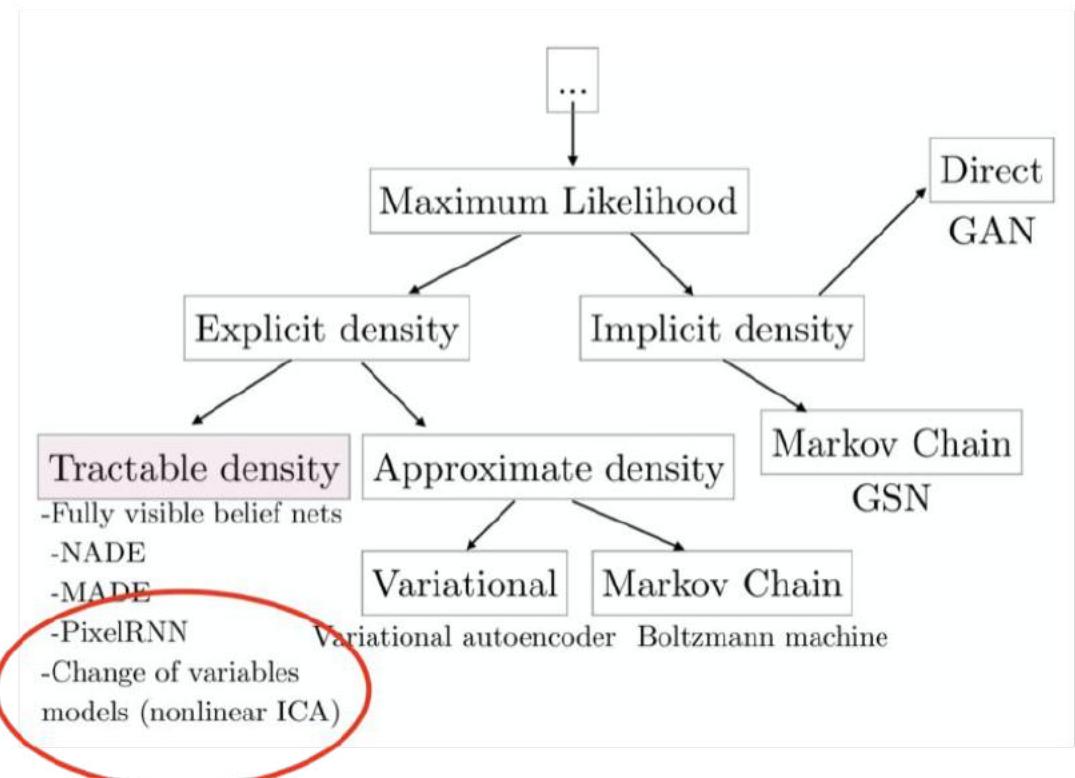
prediction



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

Flow-Based Models

Invertible Neural Networks



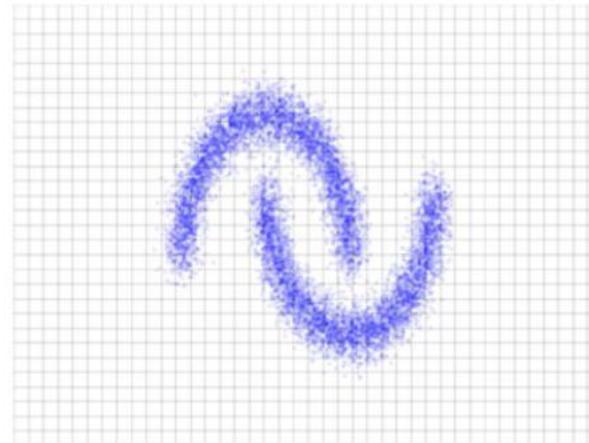
Normalizing Flows: Translating Probability Distributions

Inference

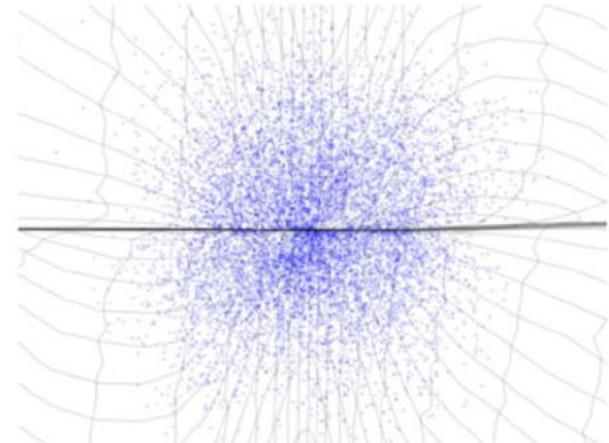
$$x \sim \hat{p}_X$$

$$z = f(x)$$

Data space \mathcal{X}



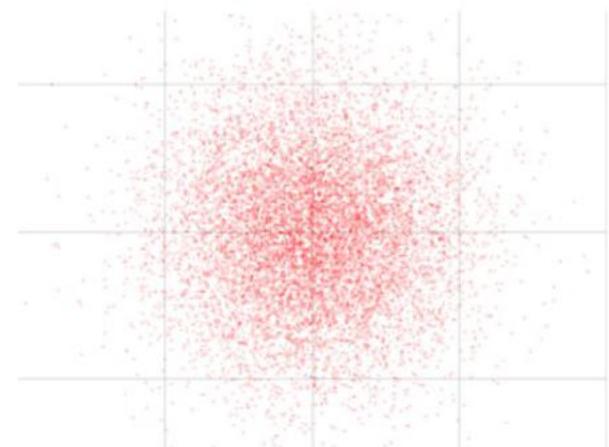
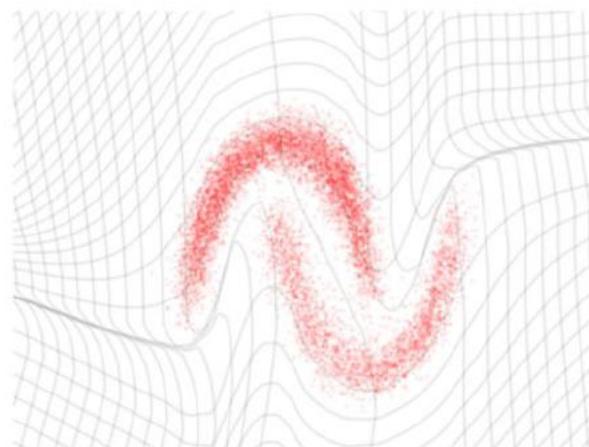
Latent space \mathcal{Z}



Generation

$$z \sim p_Z$$

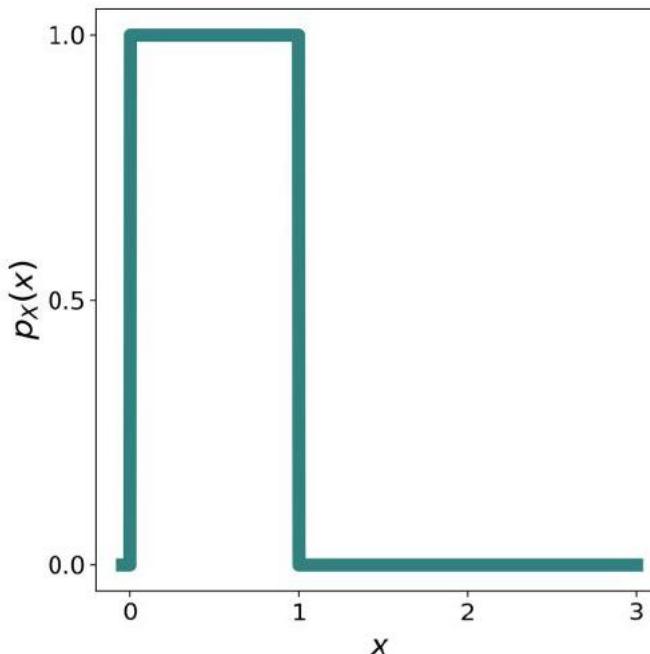
$$x = f^{-1}(z)$$



Change of Variable Density Needs to Be Normalized

$$X \sim p_X$$

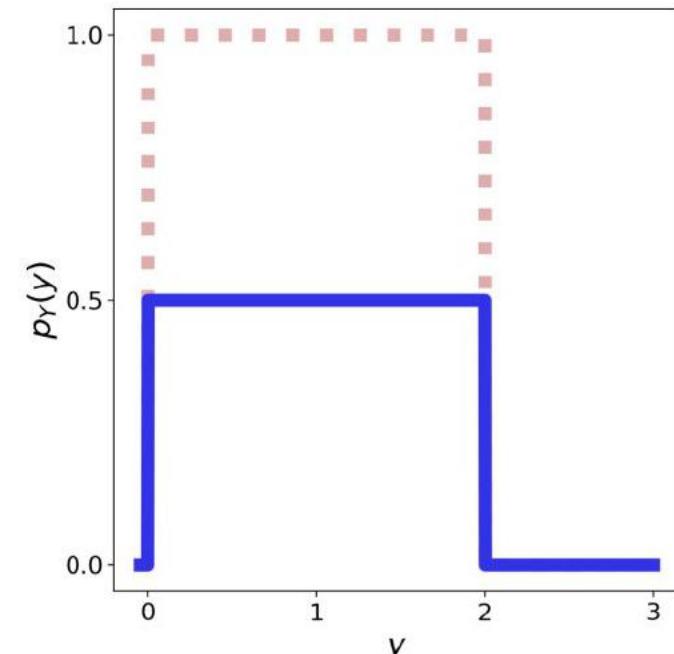
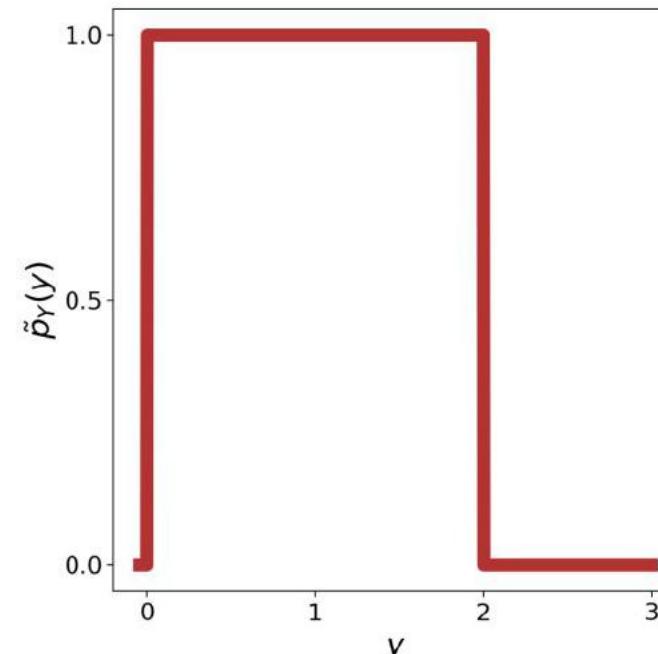
$$p_X(x) = \begin{cases} 1 & \text{for } 0 \leq x \leq 1 \\ 0 & \text{else} \end{cases}$$



$$Y := 2X$$

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

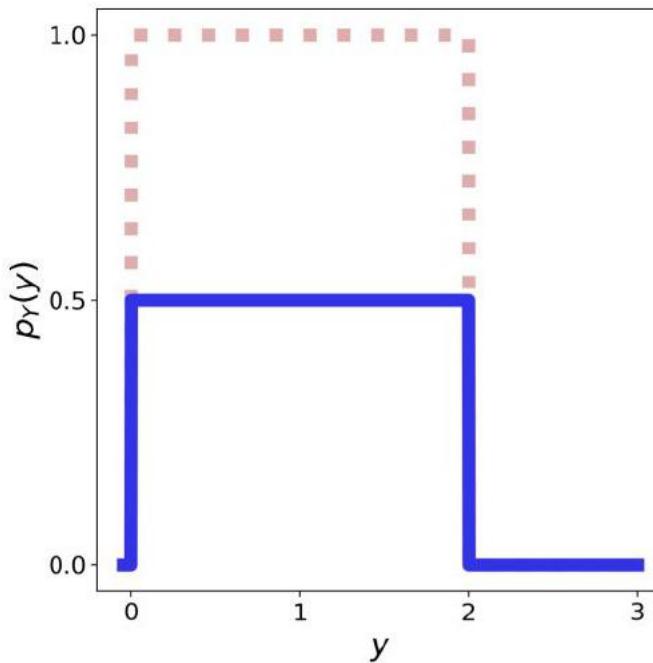
$$p_Y(y) = p_X(y/2)/2$$



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

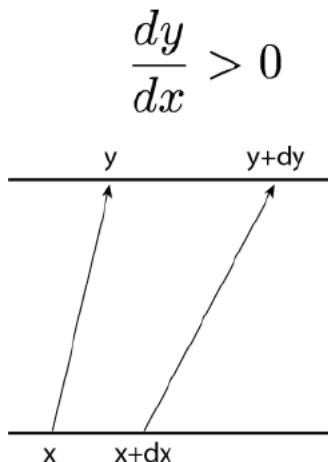
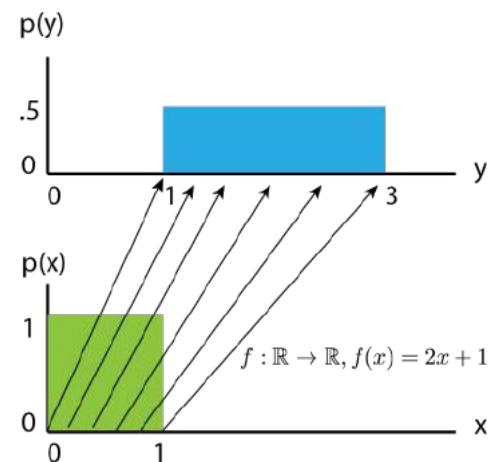
mass = density
* volume

Change of Variable Density (m-Dimensional)

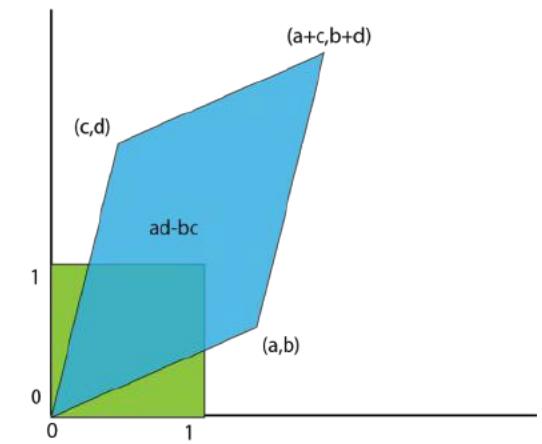
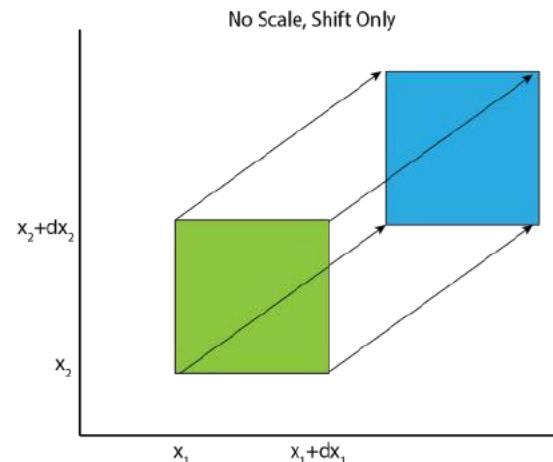
For a multivariable invertible mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ $X \sim p_X$ $Y := f(X)$

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

1-D



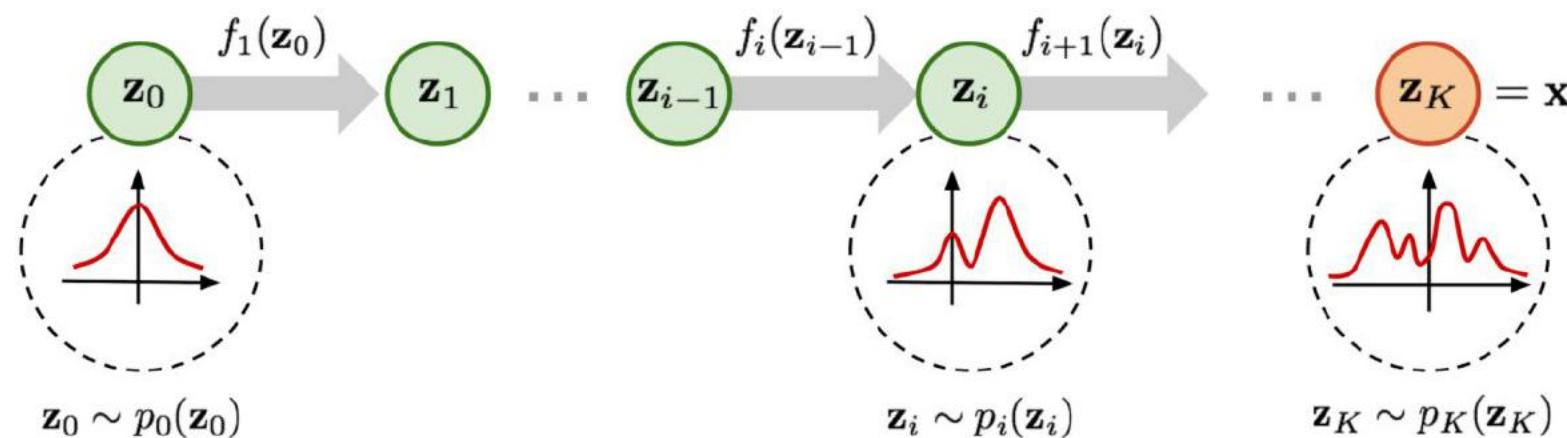
2-D



Chaining Invertible Mappings (Composition)

$$f = f_S \circ \cdots \circ f_2 \circ f_1$$

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



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

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

Training with Maximum Likelihood Principle

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

$g = f^{-1}$ bijective

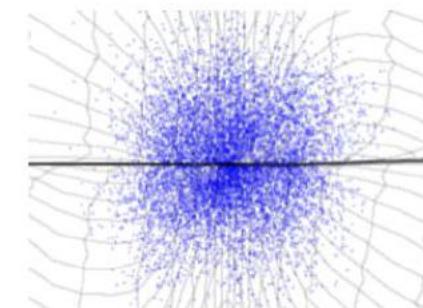
$$\mathbb{E}_x [\log p(x)] = \mathbb{E}_x \left[\log \mathcal{N}(f(x); \mathbf{0}, I) \left| \det \frac{\partial f(x)}{\partial x} \right| \right]$$

Regularizes the entropy

$$x \sim \hat{p}_X$$

Inference

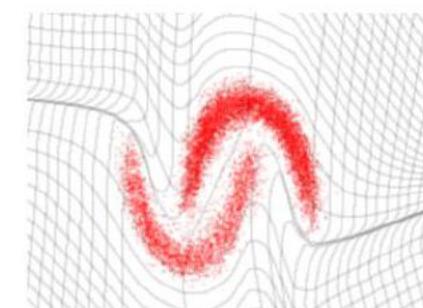
$$f \Rightarrow$$



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

Generation

$$g \Rightarrow$$



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

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

$\mathcal{O}(m^3)$

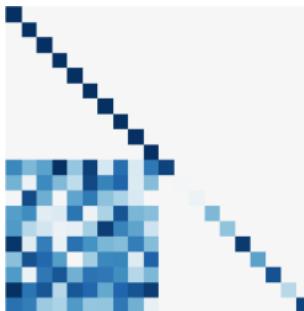
Architectural Taxonomy

Sparse connection

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

1. Block coupling

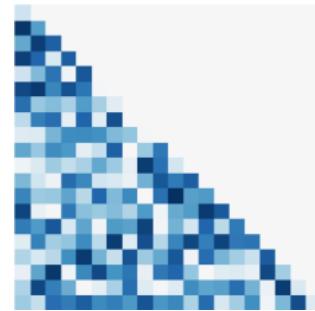
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular +
structured)

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN



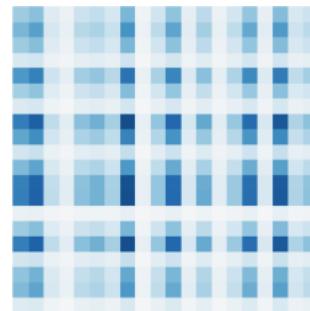
(Lower triangular)

Residual Connection

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

3. Det identity

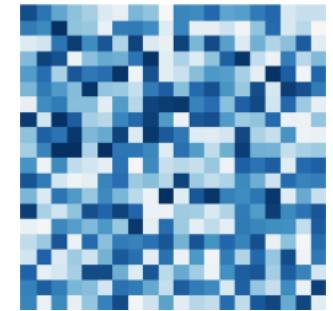
Planar/Sylvester
flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual
Flow
FFJORD



(Arbitrary)

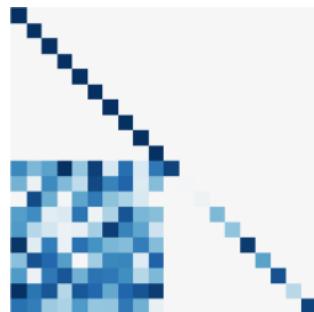
Architectural Taxonomy

Sparse connection

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

1. Block coupling

NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



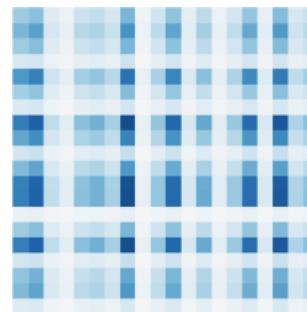
(Lower triangular +
structured)

Residual Connection

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

3. Det identity

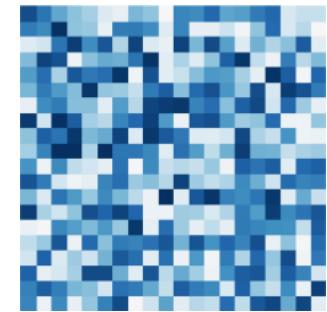
Planar/Sylvester
flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual
Flow
FFJORD



(Arbitrary)

Jacobian

Coupling Law - NICE

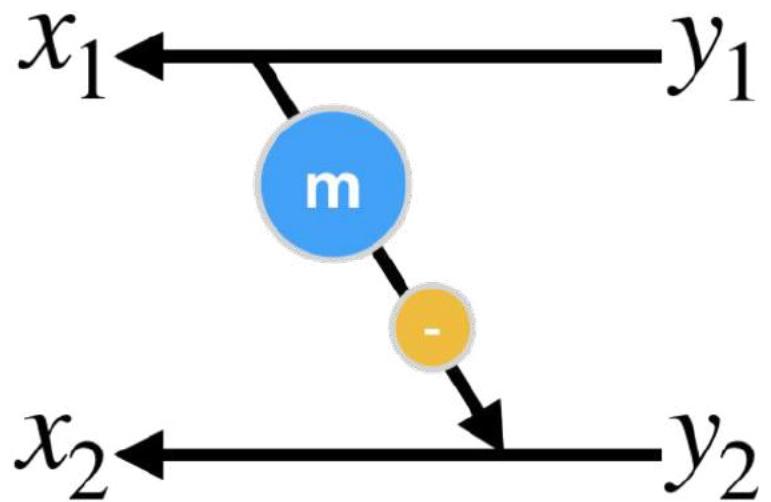
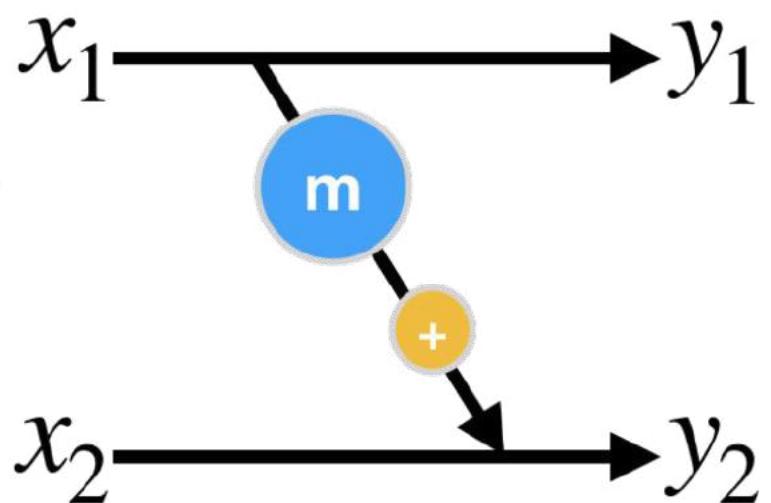
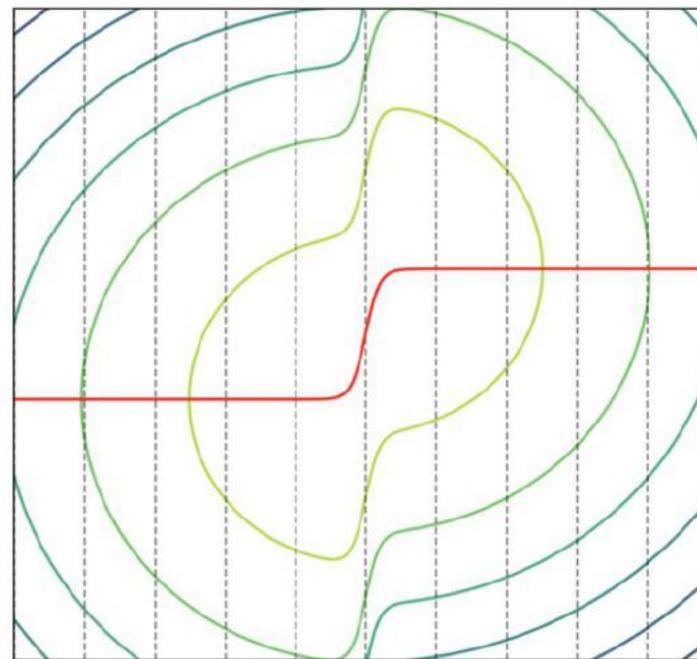
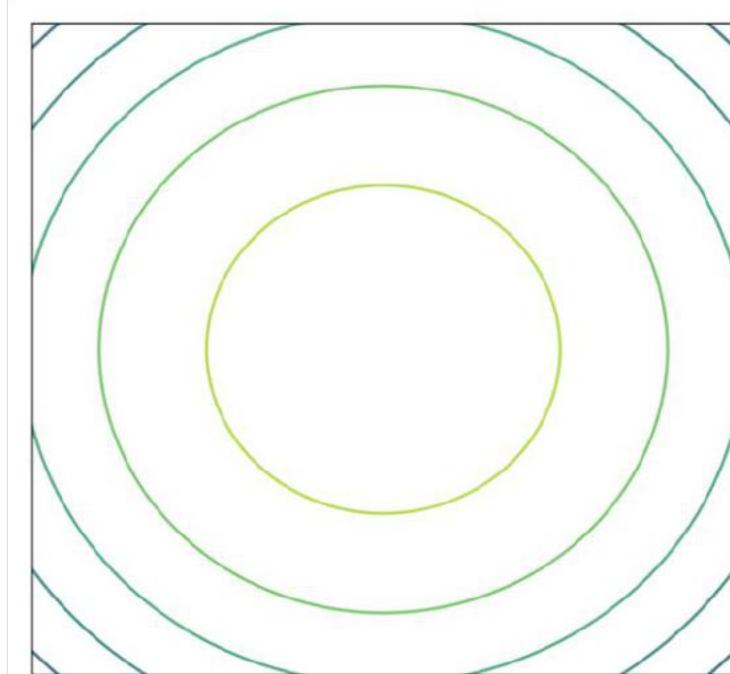
- General form

$$f(\mathbf{x})_1 = x_1, \quad f(\mathbf{x})_2 = x_2 + \mathcal{F}(x_1)$$

- Invertibility

no constraint

- Jacobian determinant = 1 (volume preserving)



Coupling Law - RealNVP

Real-valued
Non-Volume
Preserving

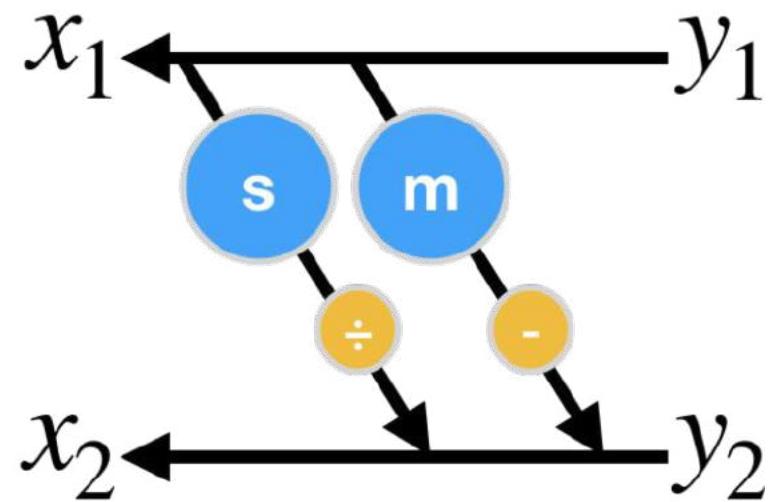
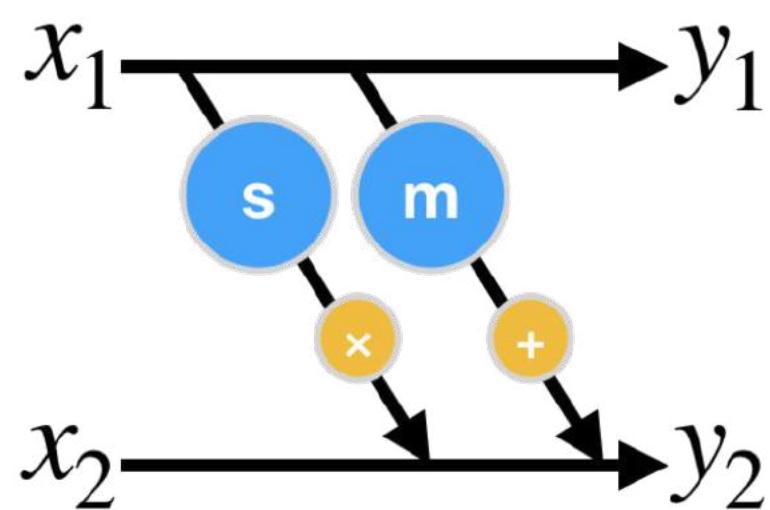
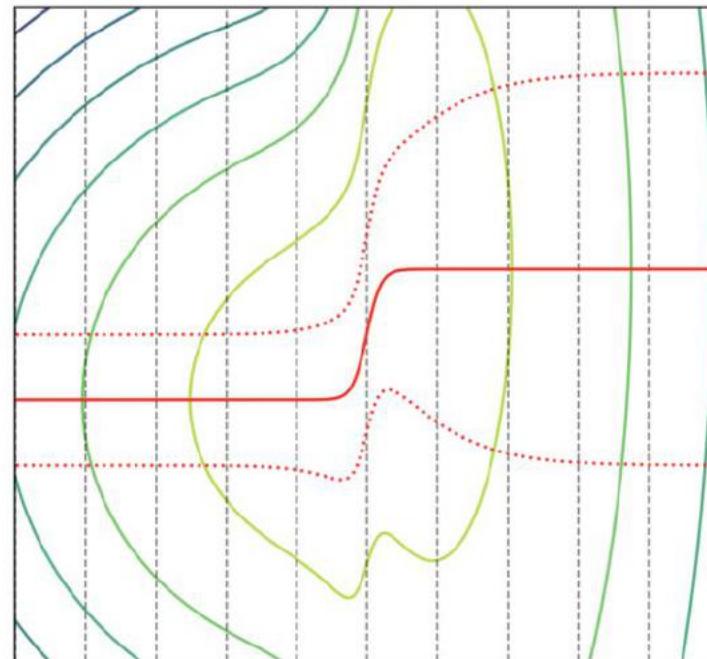
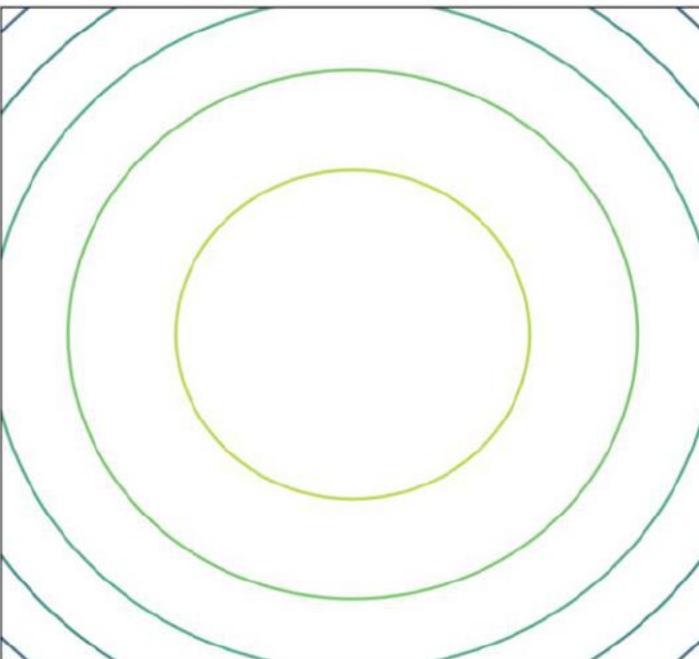
- General form
- Invertibility
- Jacobian determinant

$$f(\mathbf{x})_1 = \mathbf{x}_1,$$

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

$s > 0$ (or simply non-zero)

product of s



Real NVP via Masked Convolution

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

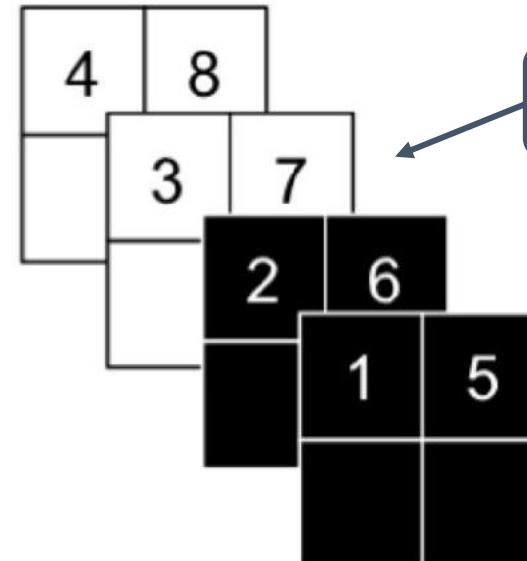
$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s_-(b \odot x))) + m(b \odot x))$$

Real NVP via Masked Convolution

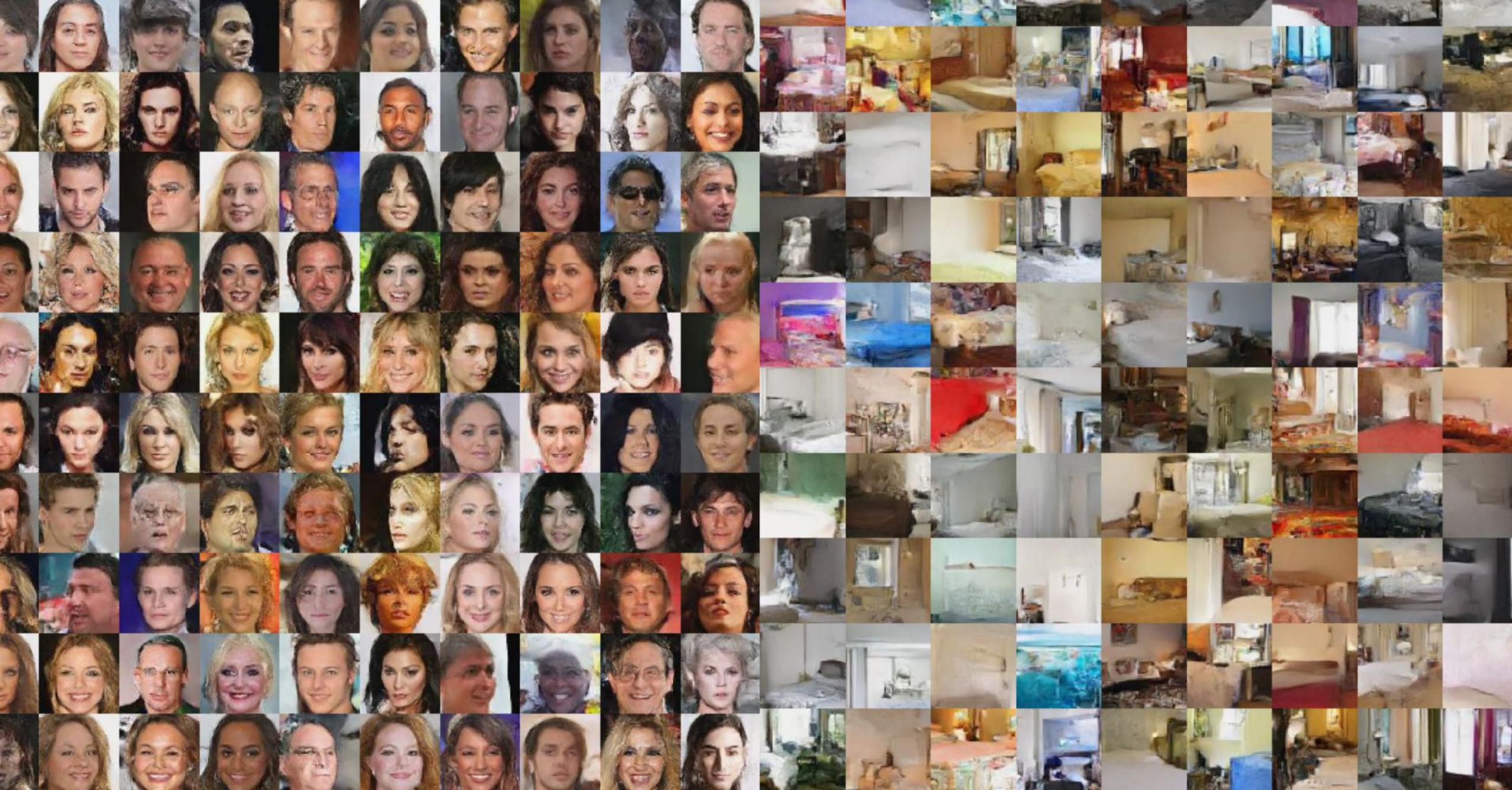
Partitioning can be implemented using a binary mask b , and using the functional form for y

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

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



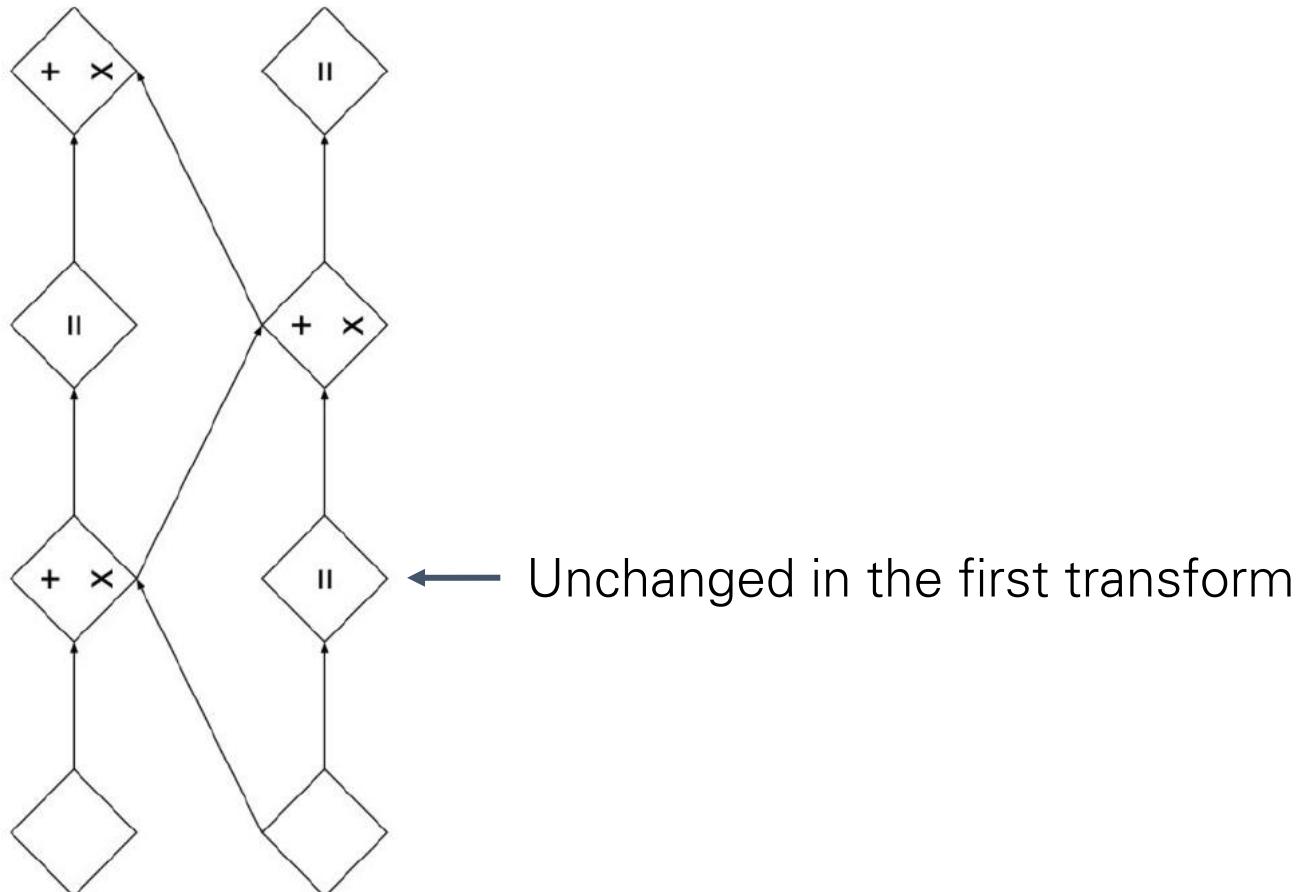
The channel-wise mask b is 1 for the first half of the channel dimensions and 0 for the second half.



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

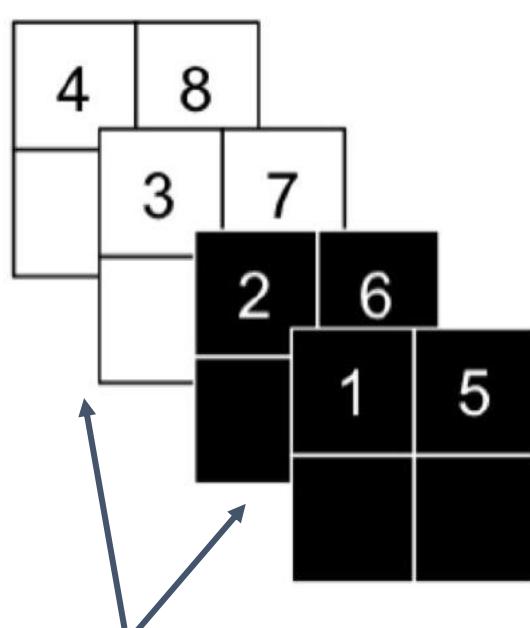
Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)



Glow: Generative Flow with 1x1 Convolutions

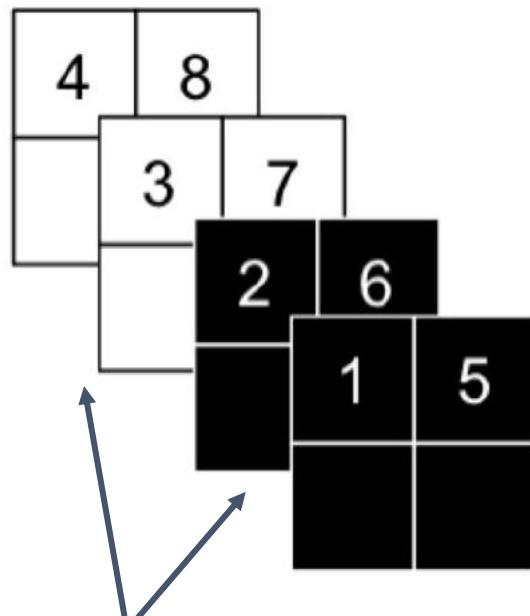
Replacing permutation with 1x1 convolution (soft permutation)



Alternating masks

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)



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

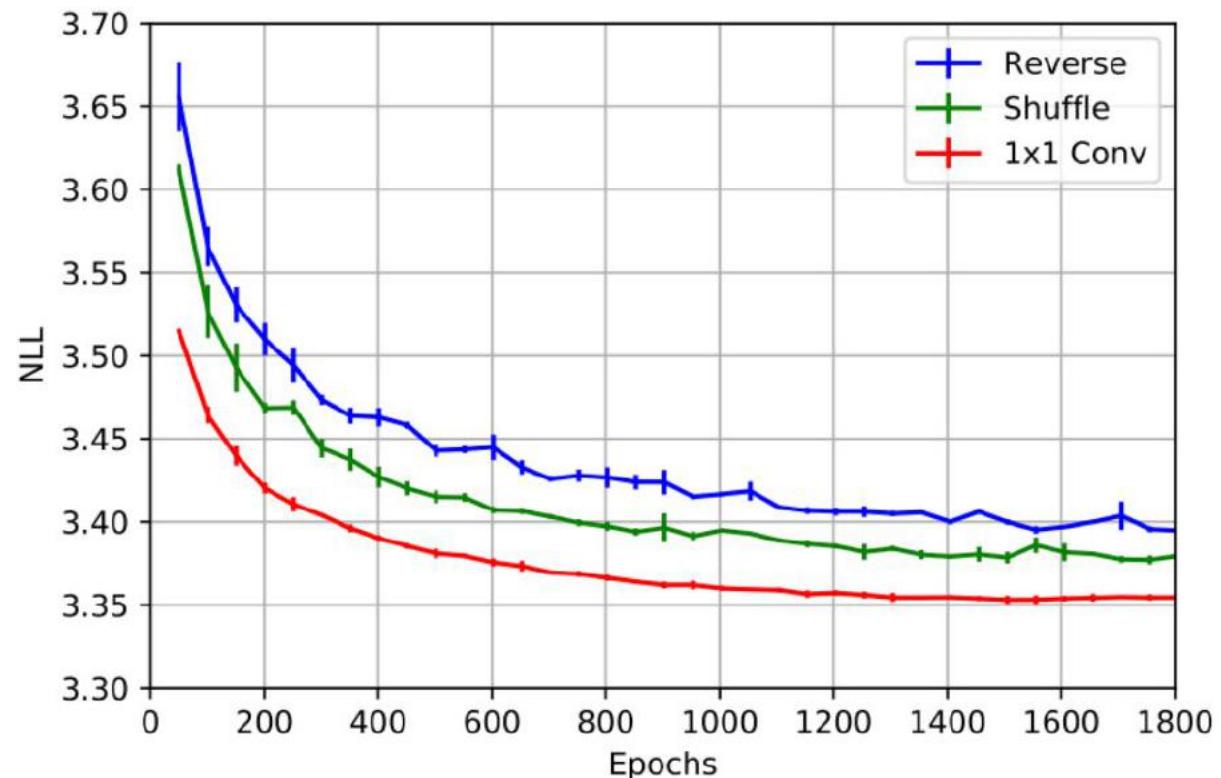
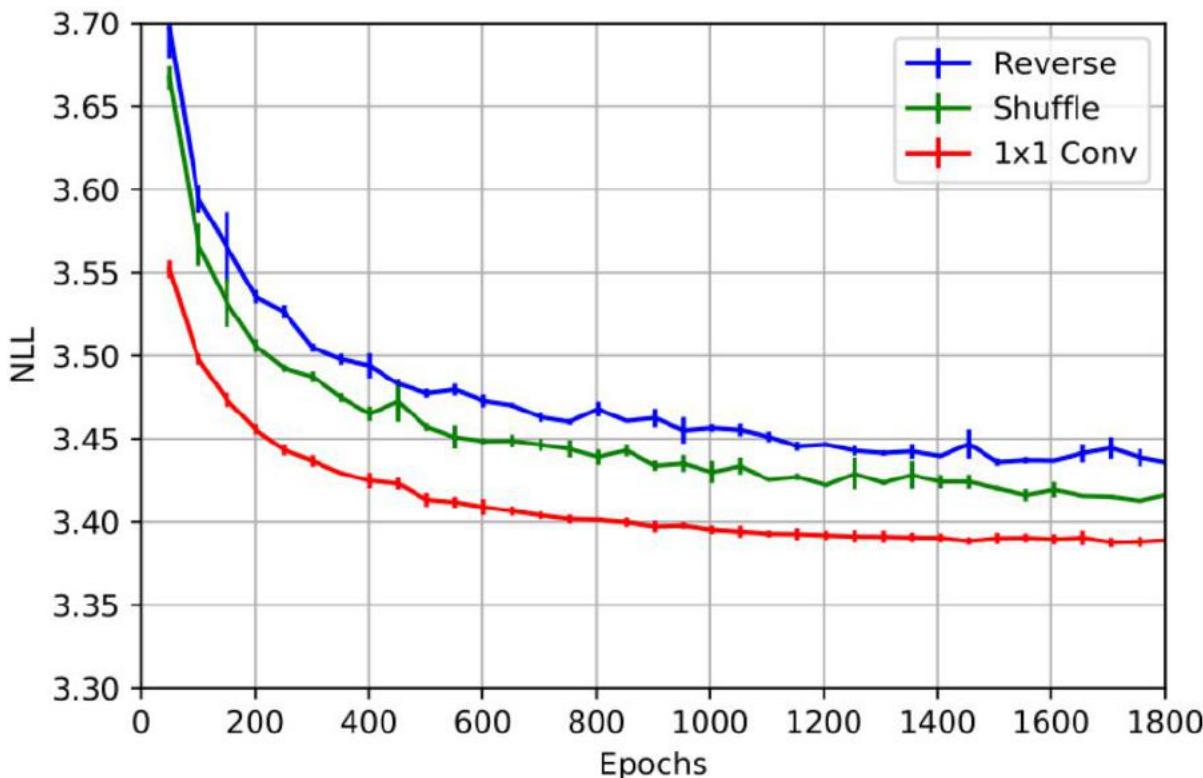
Replace with a general invertible matrix W

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

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

Ablation: Permutation vs 1x1 Convolution

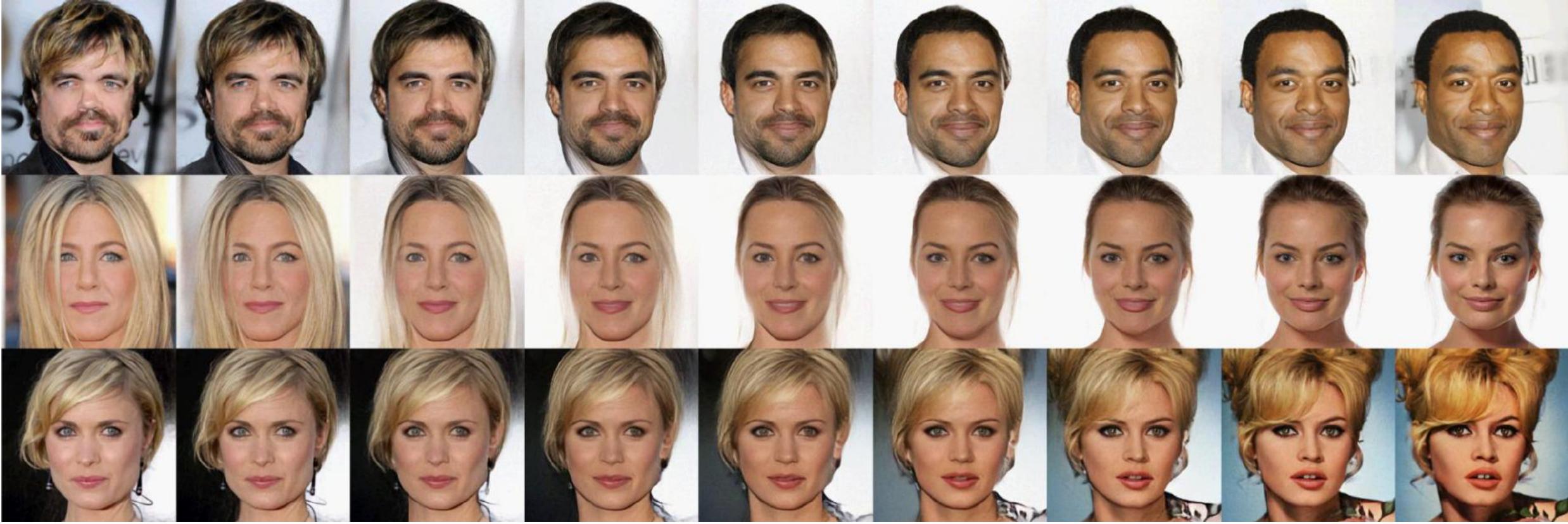
Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	3.35	4.09	3.81	2.38	2.46	2.67



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



Figure from Glow: Generative Flow with Invertible 1x1 Convolutions by Kingma and Dhariwal, 2018



Interpolation with Generative Flows



Figure from *Glow: Generative Flow with Invertible 1x1 Convolutions* by Kingma and Dhariwal, 2018

Video from Durk Kingma's youtube channel

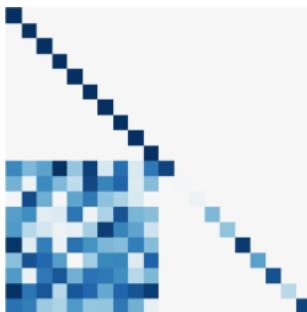
Architectural Taxonomy

Sparse connection

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

1. Block coupling

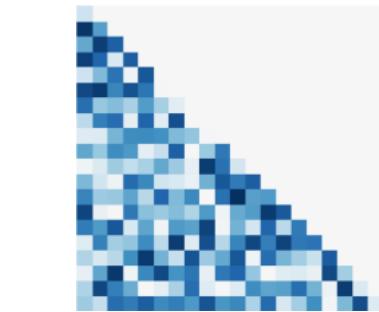
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular +
structured)

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN



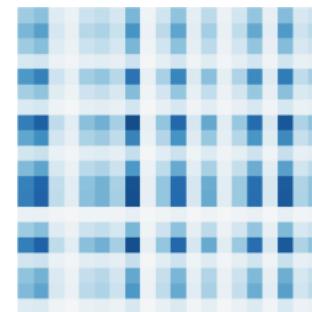
(Lower triangular)

Residual Connection

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

3. Det identity

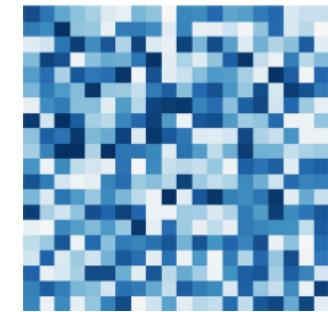
Planar/Sylvester
flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual
Flow
FFJORD



(Arbitrary)

Inverse (Affine) Autoregressive Flows

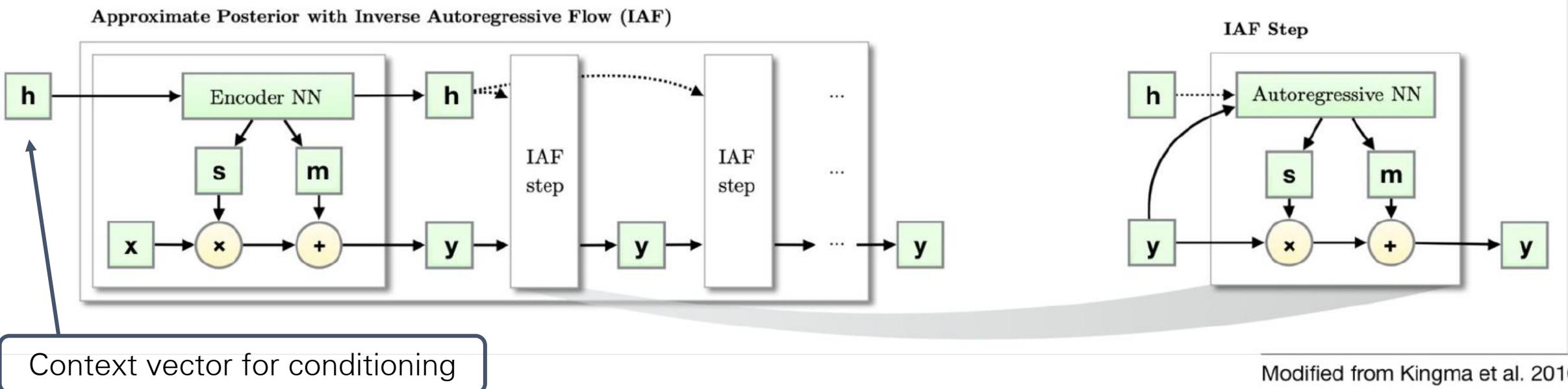
- General form

$$f(\mathbf{x})_t = s(\mathbf{x}_{<t}) \cdot \mathbf{x}_t + m(\mathbf{x}_{<t})$$

- Invertibility

$s > 0$ (or simply non-zero)

- Jacobian determinant product of s

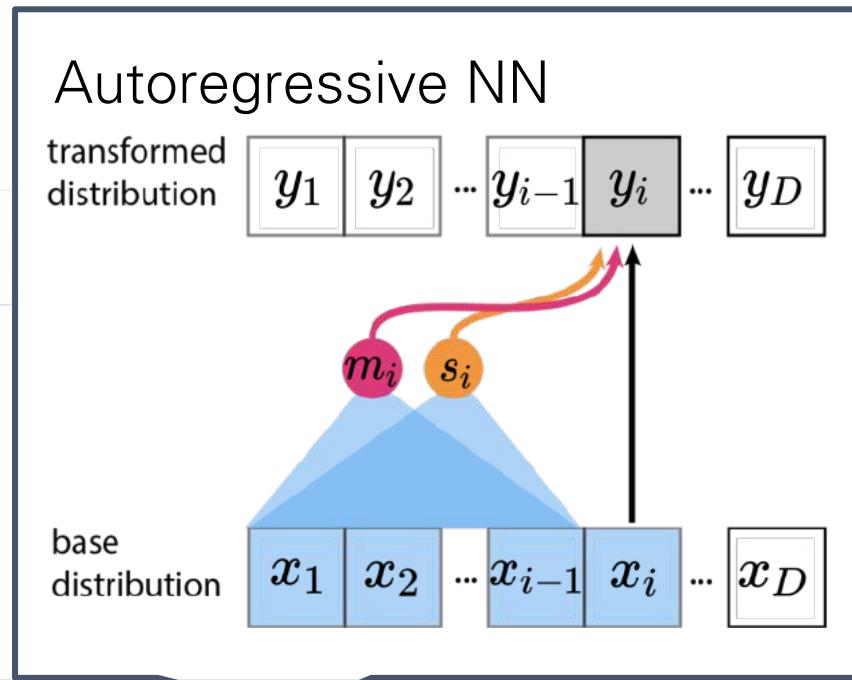


Inverse Autoregressive Flows

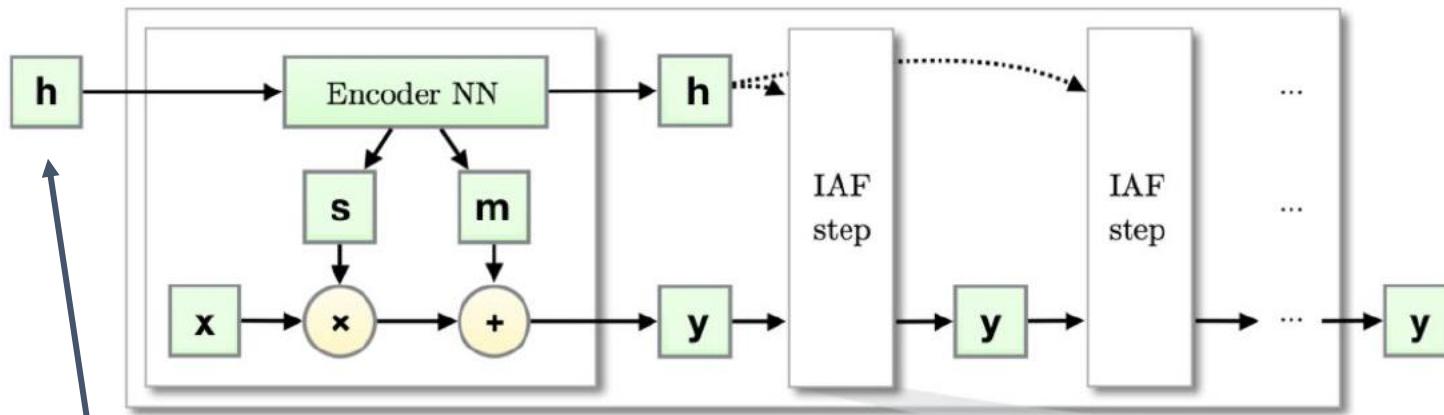
- General form
- Invertibility
- Jacobian determinant product of s

$$f(\mathbf{x})_t = s(\mathbf{x}_{<t}) \cdot \mathbf{x}_t + m(\mathbf{x}_{<t})$$

$s > 0$ (or simply non-zero)



Approximate Posterior with Inverse Autoregressive Flow (IAF)

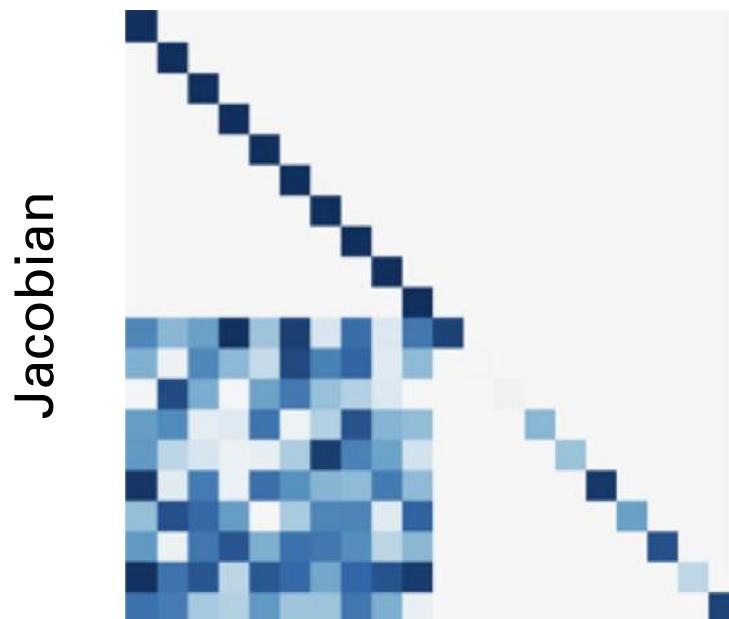


Context vector for conditioning

Trade-off between Expressivity and Inversion Cost

Block autoregressive

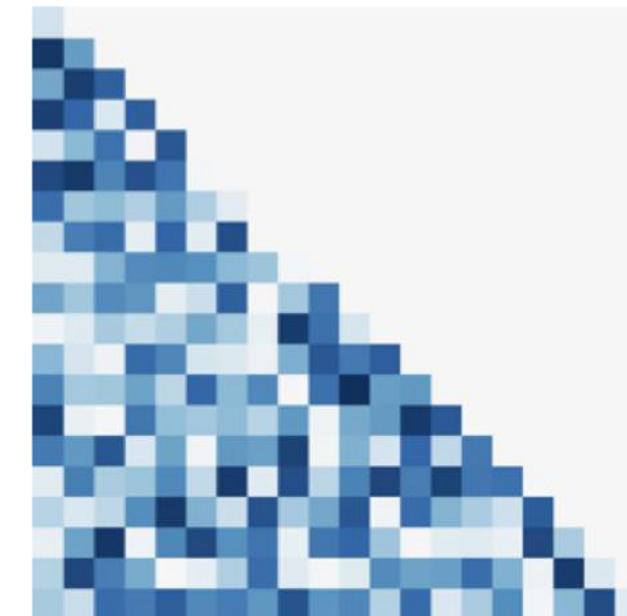
- Limited capacity
- Inverse takes constant time



(Block triangular)

Autoregressive

- Higher capacity
- Inverse takes linear time (dimensionality)



(Triangular)

Neural Autoregressive Flows

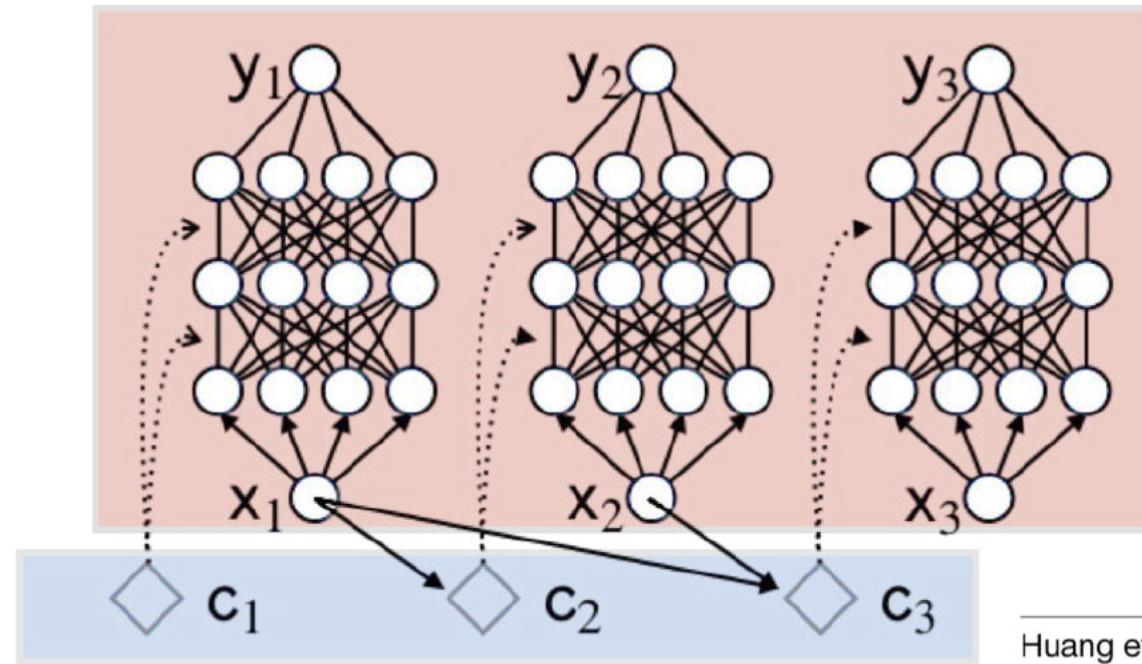
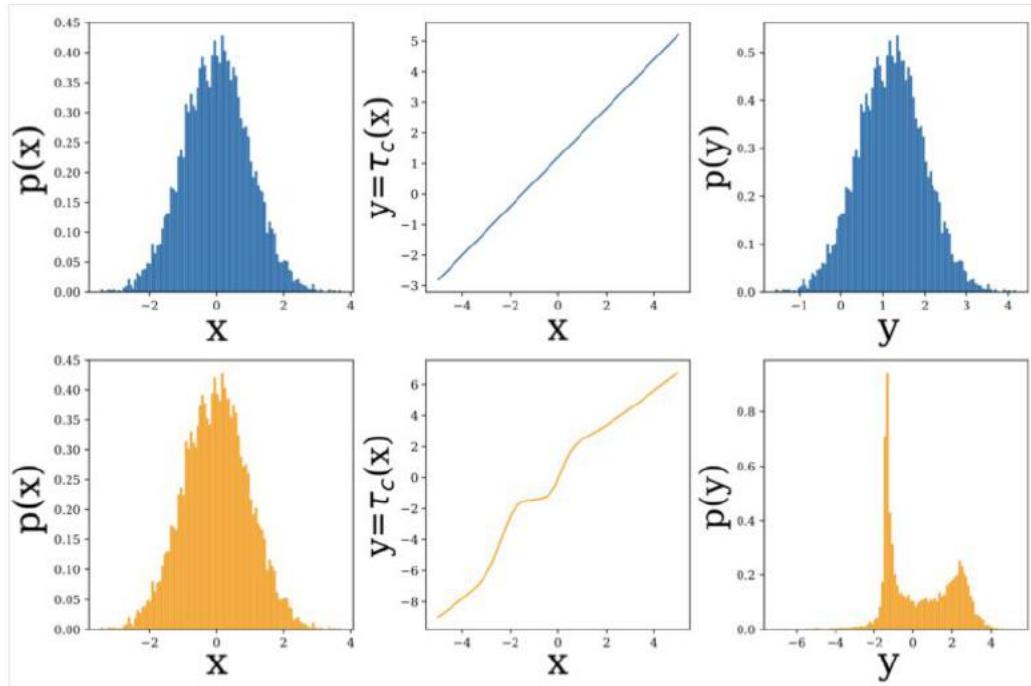
- General form

$$f(\mathbf{x})_t = \mathcal{P}(\mathbf{x}_t; \mathcal{H}(\mathbf{x}_{$$

- Invertibility

monotonic activation and positive weight in \mathcal{P}

- Jacobian determinant product of derivatives (elementwise)



Huang et al. 2018

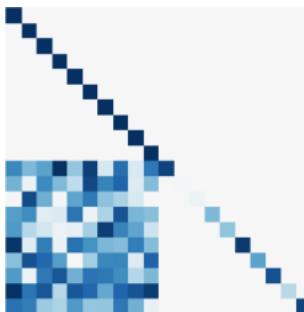
Architectural Taxonomy

Sparse connection

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

1. Block coupling

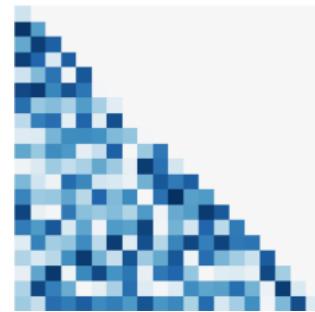
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular +
structured)

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN



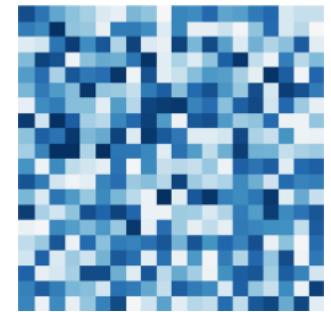
(Lower triangular)

Residual Connection

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

4. Stochastic estimation

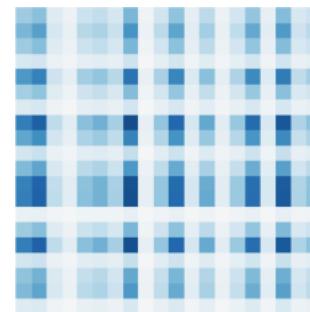
Residual
Flow
FFJORD



(Arbitrary)

3. Det identity

Planar/Sylvester
flows
Radial flow



(Low rank)

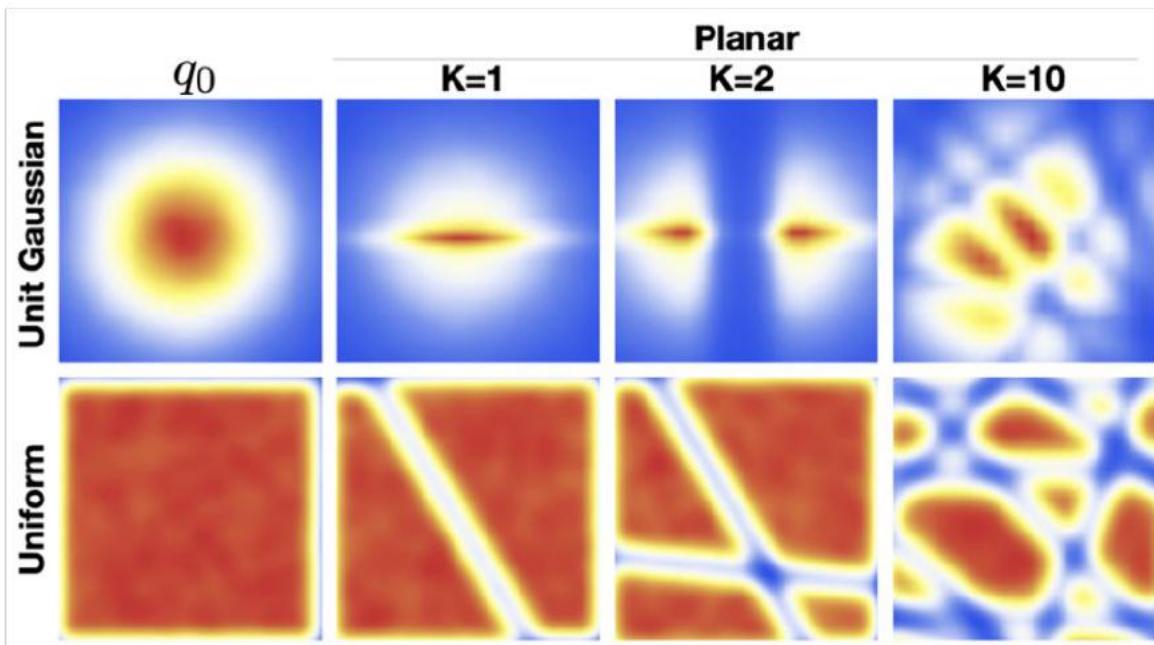
Determinant Identity – Planar Flows

- General form
- Invertibility
- Jacobian determinant

$$f(\mathbf{x}) = \mathbf{x} + \mathbf{u}h(\mathbf{w}^\top \mathbf{x} + b)$$

$$\mathbf{u}^\top \mathbf{w} > -1 \text{ if } h = \tanh$$

$$\left| \det \frac{\partial f}{\partial \mathbf{x}} \right| = \left| \det \left(\mathbf{I} + h'(\mathbf{w}^\top \mathbf{x} + b) \mathbf{u} \mathbf{w}^\top \right) \right| = \left| 1 + h'(\mathbf{w}^\top \mathbf{x} + b) \mathbf{u}^\top \mathbf{w} \right|$$



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

$$f(\mathbf{x}) = \mathbf{x} + \mathbf{A}h(\mathbf{B}\mathbf{x} + \mathbf{b})$$

$\mathbf{A} \in \mathbb{R}^{m \times d}$, $\mathbf{B} \in \mathbb{R}^{d \times m}$, $\mathbf{b} \in \mathbb{R}^d$, and $d \leq m$

- Invertibility

Similar to planar flows

- Jacobian determinant

Using Sylvester's Thm: $\det(\mathbf{I}_m + \mathbf{AB}) = \det(\mathbf{I}_d + \mathbf{BA})$

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	4.40 ± 0.06	4.31 ± 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	99.00 ± 0.04	93.77 ± 0.03	104.62 ± 0.29	93.82 ± 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

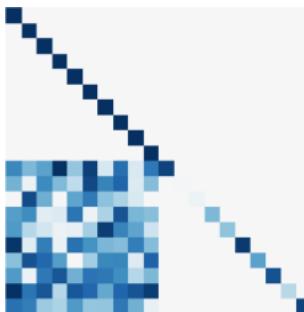
Architectural Taxonomy

Sparse connection

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

1. Block coupling

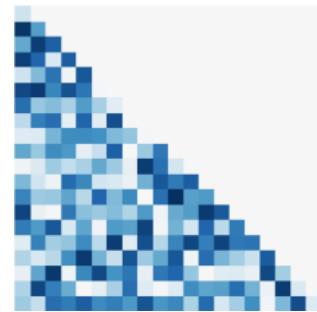
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular +
structured)

2. Autoregressive

IAF/MAF/NAF
SOS polynomial
UMNN



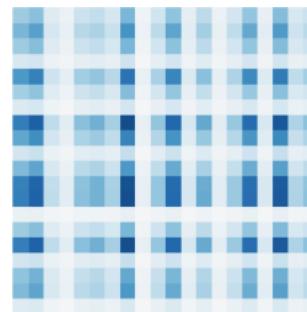
(Lower triangular)

Residual Connection

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

3. Det identity

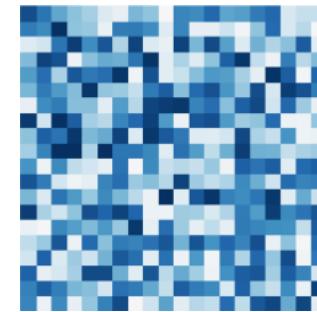
Planar/Sylvester
flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual
Flow
FFJORD



(Arbitrary)

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

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

Jacobi's formula

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

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

Jacobi's formula

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

Power series expansion

Stochastic Estimation for General Residual Form

- General form

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

- Invertibility

$$\left\| \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right\|_2 < 1$$

- Jacobian determinant

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

Jacobi's formula

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

Power series expansion

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

Truncation &
Hutchinson trace estimator

Stochastic Estimation for General Residual Form

- General form

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

- Invertibility

$$\left\| \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right\|_2 < 1$$

- Jacobian determinant

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

Jacobi's formula

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

Power series expansion

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

Truncation &
Hutchinson trace estimator

Bias

Stochastic Estimation for General Residual Form

- General form

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

- Invertibility

$$\left\| \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right\|_2 < 1$$

- Jacobian determinant

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

Jacobi's formula

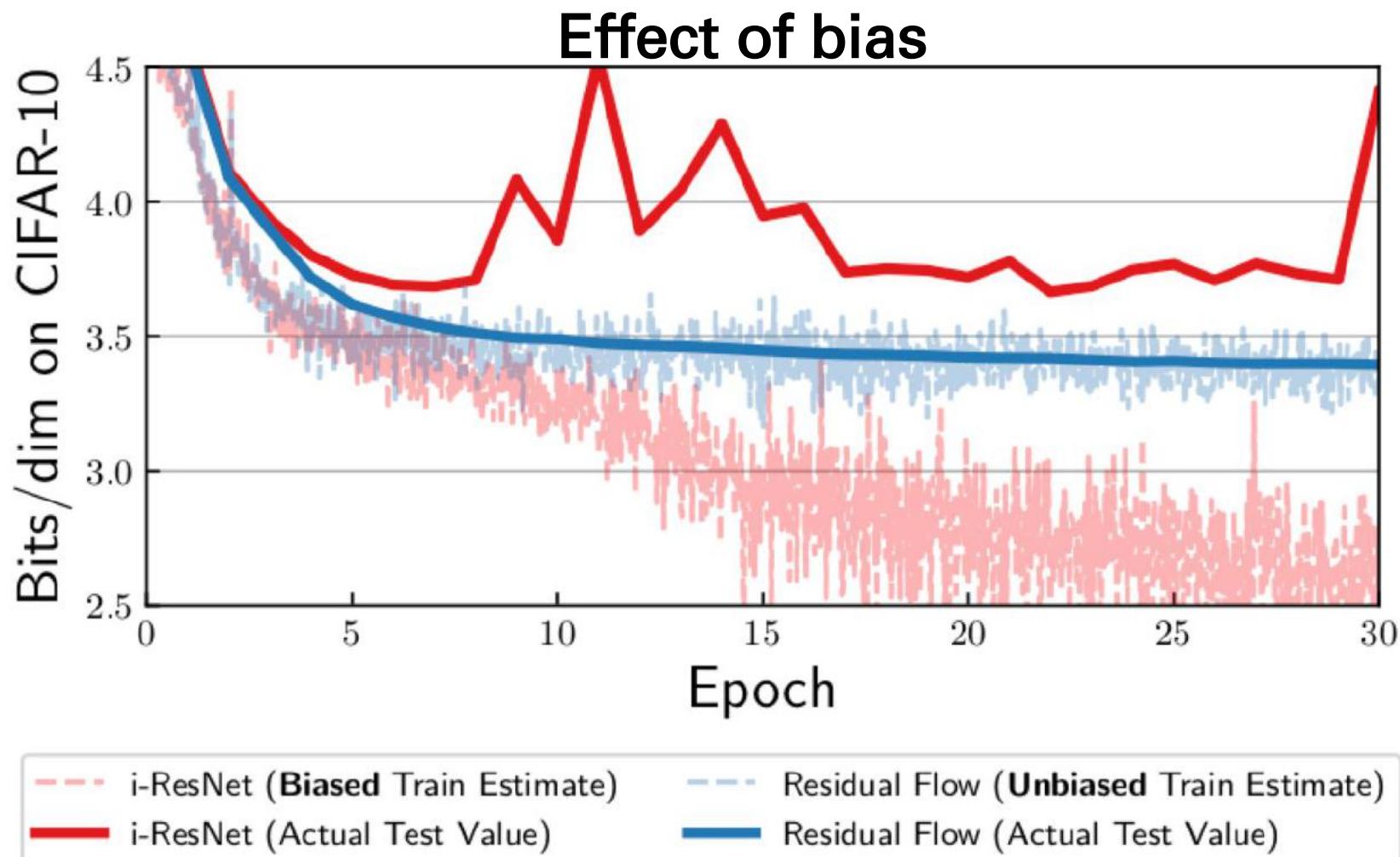
$$\text{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{tr} \left(\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^k \right)$$

Power series expansion

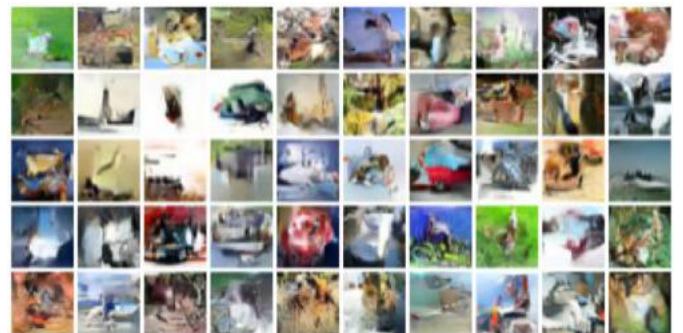
$$= \mathbb{E}_{v,n} \left[\sum_{k=1}^n \frac{(-1)^{k+1}}{k \cdot \mathbb{P}(N \geq k)} v^\top \left(\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}^k \right) v \right]$$

Russian roulette estimator &
Hutchinson trace estimator

CelebA samples



Cifar10 samples



Imagenet-32 samples



Next lecture:
Variational Autoencoders