Generative Adversarial Networks – Part II

Du Ang

April 28, 2018

VISION@OUC

Recap

- Supervised Learning vs Unsupervised Learning
- Introduction to generative models
- Popular generative models in detail
  - Variational Autoencoders (VAE)
  - Generative Adversarial Networks (GAN)

Recap:
Supervised Learning vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal:
Learn a function to map x -> y

Examples: Regression, classification, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

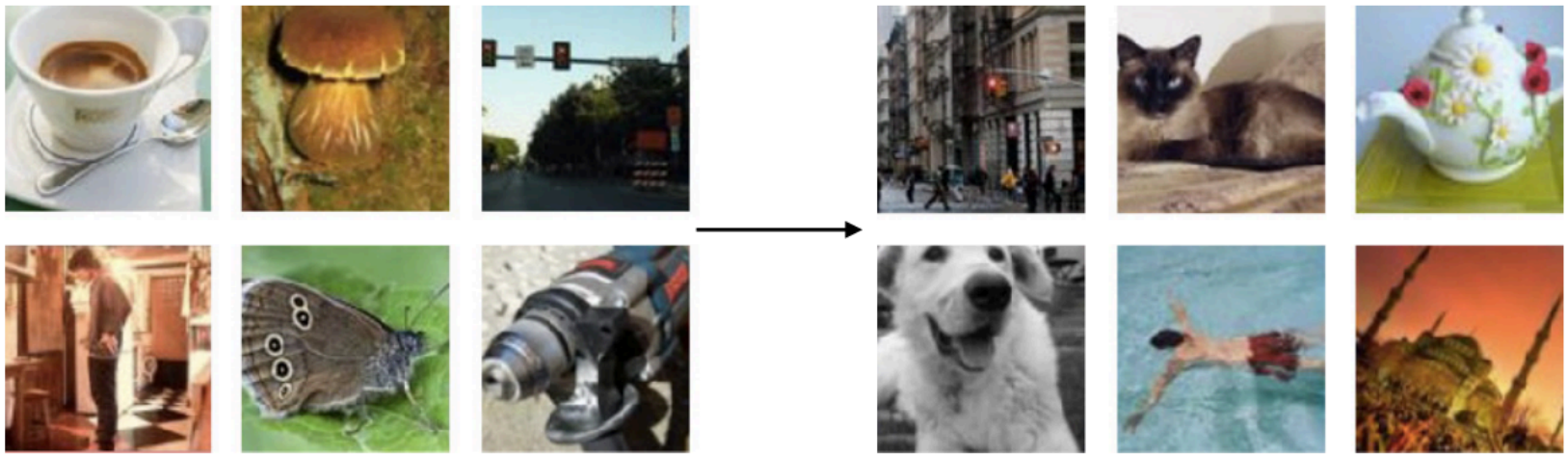
Data: x
Just data, no labels!

Goal:
Learn some underlying hidden *structure* of the data

Examples: Clustering, dimension reduction, feature learning, density estimation, etc.

Recap: Generative Models

Takes a training set, consisting of samples drawn from a distribution p_{data}, and learns to represent an estimate of that distribution with p_{model}.

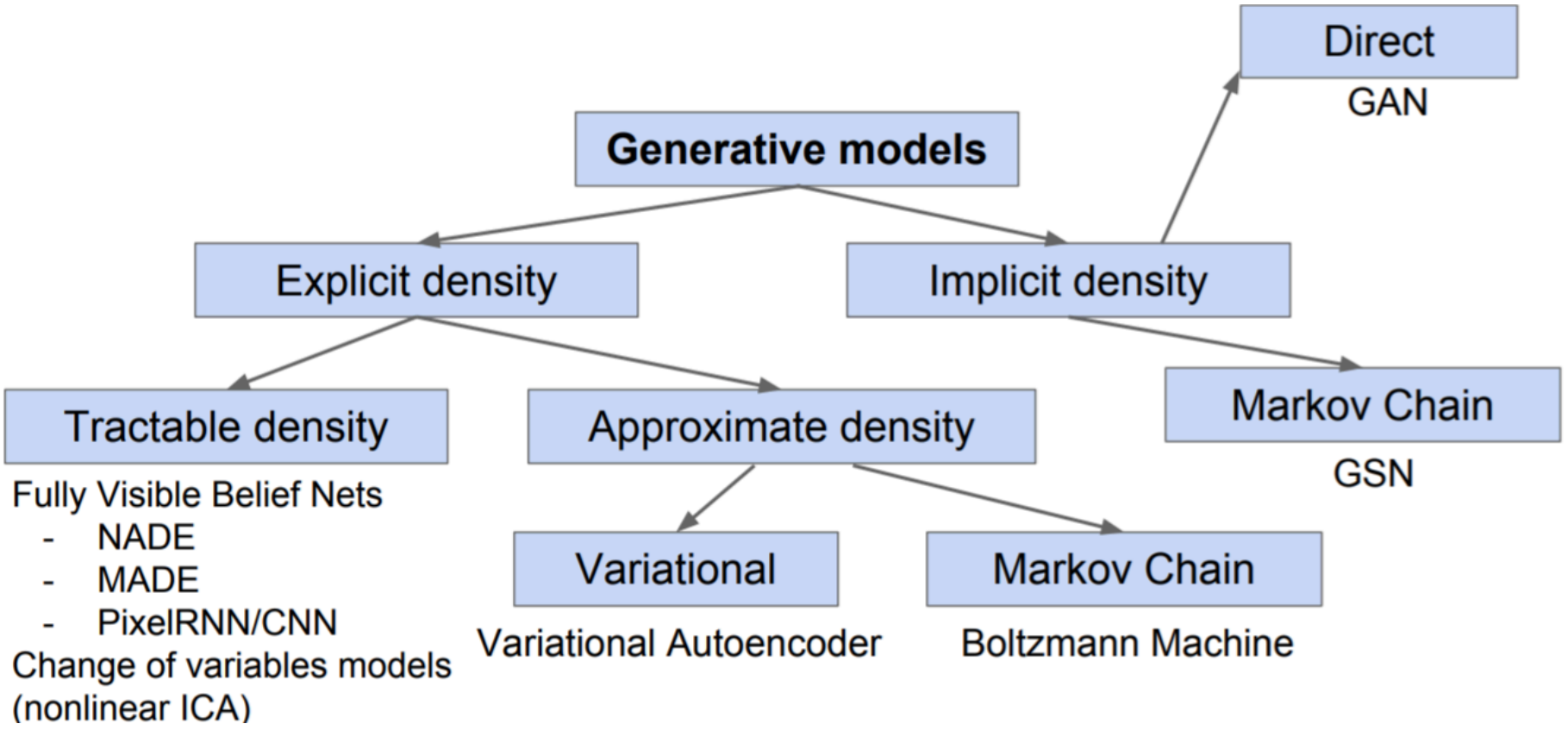


Training data
$p_{data}(x)$

Model samples
$p_{model}(x)$

Images from http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

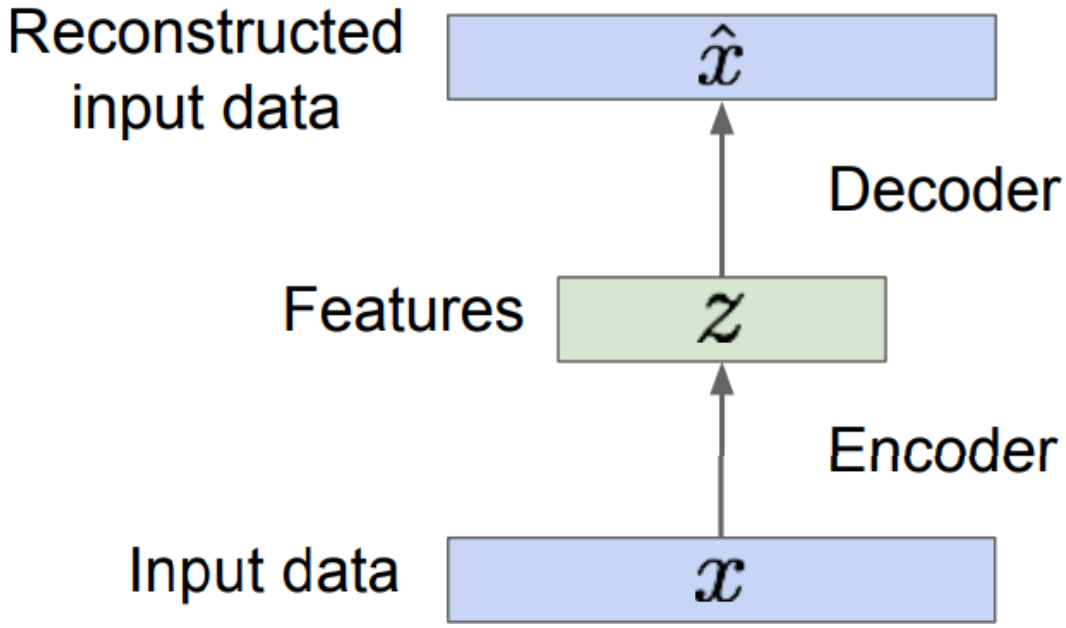
Recap: Taxonomy of Generative Models



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Recap: Autoencoders

- Autoencoders can reconstruct data, and can learn features to initialize a supervised model.
- Features capture factors of variation in training data.



Recap: Variational Autoencoders

Data likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

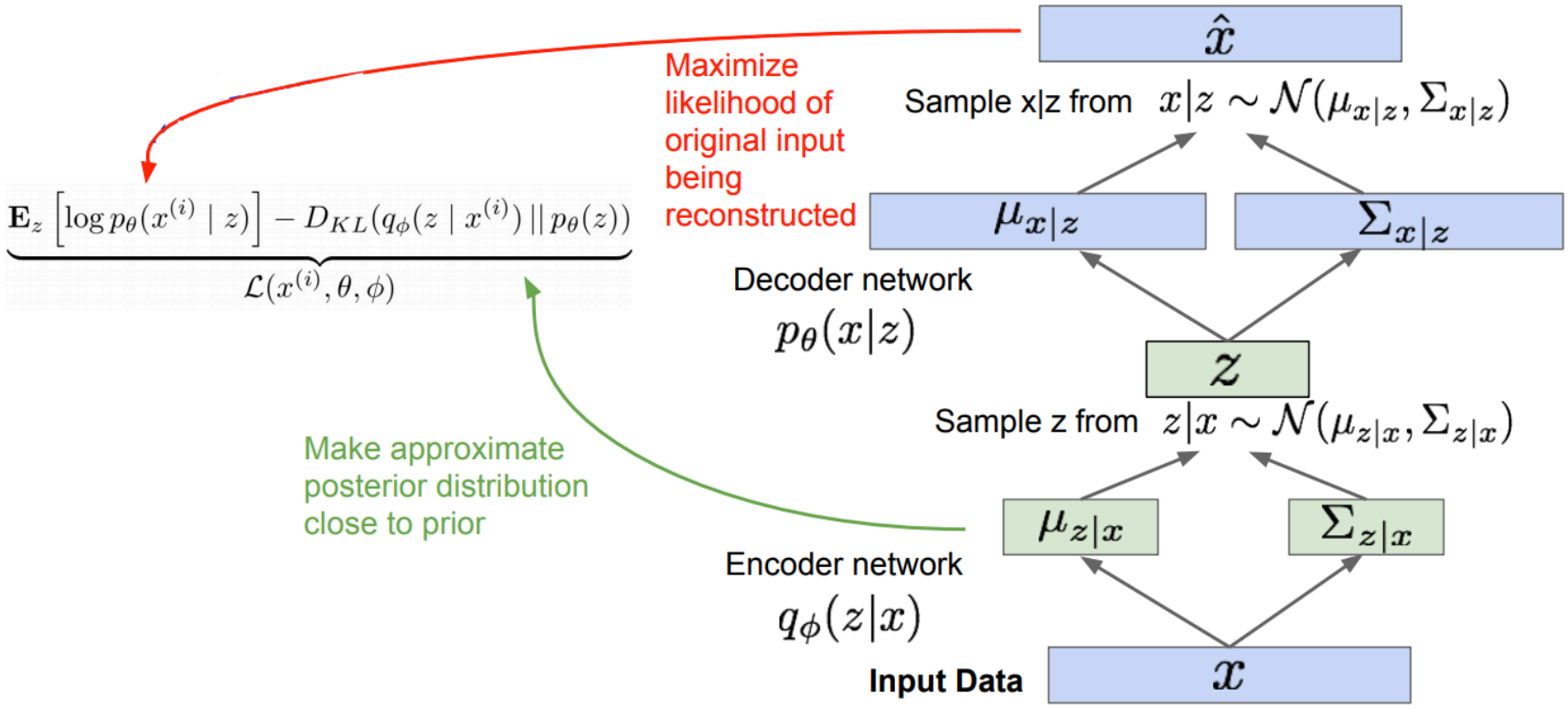
$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

intractable

Kingma and Welling, "Auto-Encoding Variational Bayes" in ICLR, 2014

Recap: Variational Autoencoders



Kingma and Welling, "Auto-Encoding Variational Bayes" in ICLR, 2014

Recap: Variational Autoencoders

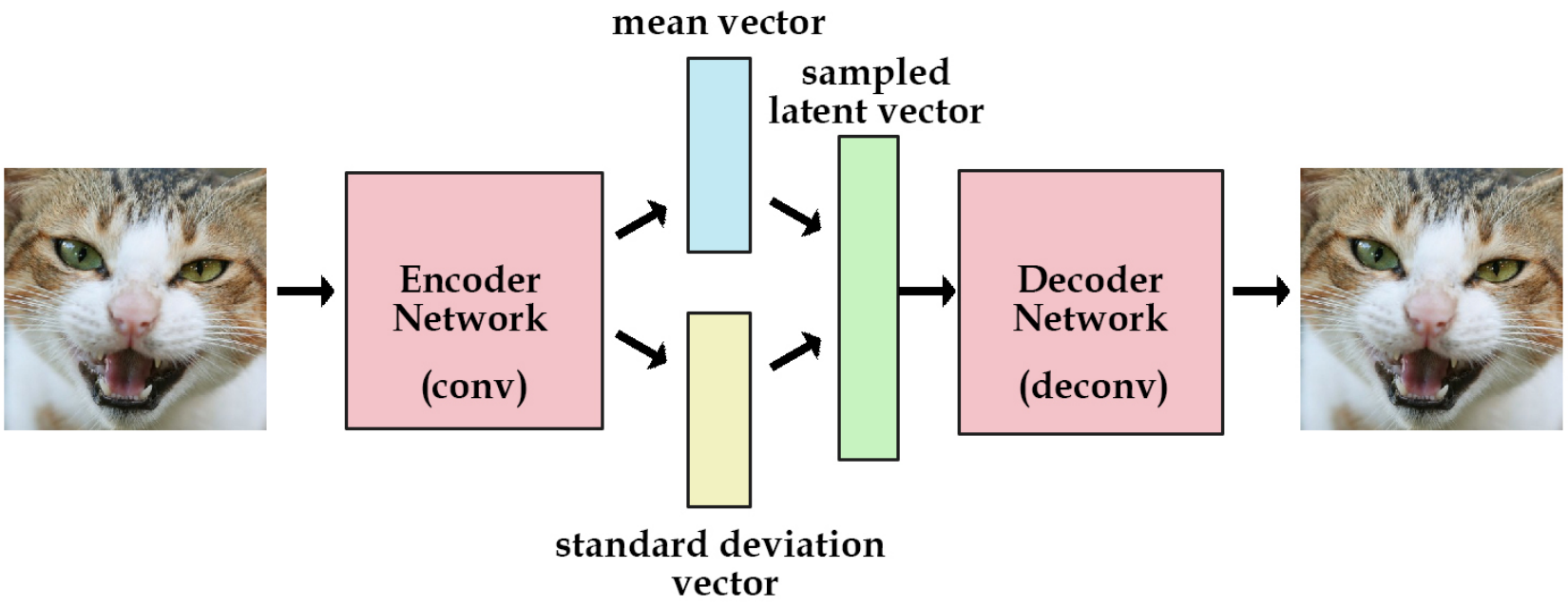


Image from http://kvfrans.com/variational-autoencoders-explained/

Recap: Variational Autoencoders

Defines an intractable density
=> derive and optimize a (vairational) lower bound

Probablisitic spin to traditional autoencoders
=> allows generating data

Recap: Variational Autoencoders

Pros:

- Principled approach to generate models
- Allows inference of $q(z|x)$, can be useful feature representation for other tasks

Cons:

- Maximizes lower bound of likelihood: okay, but not as good evaluation as tractable density
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

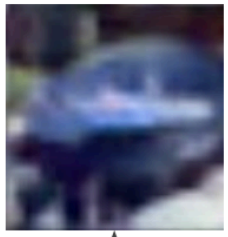
Active areas of research:

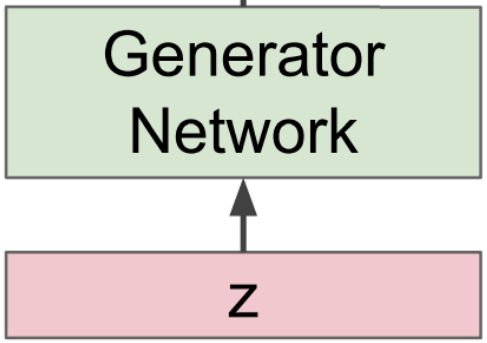
- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
- Incorporating structure in latent variables

Recap: Generative Adversarial Networks

Try a new way

Output: Sample from training distribution





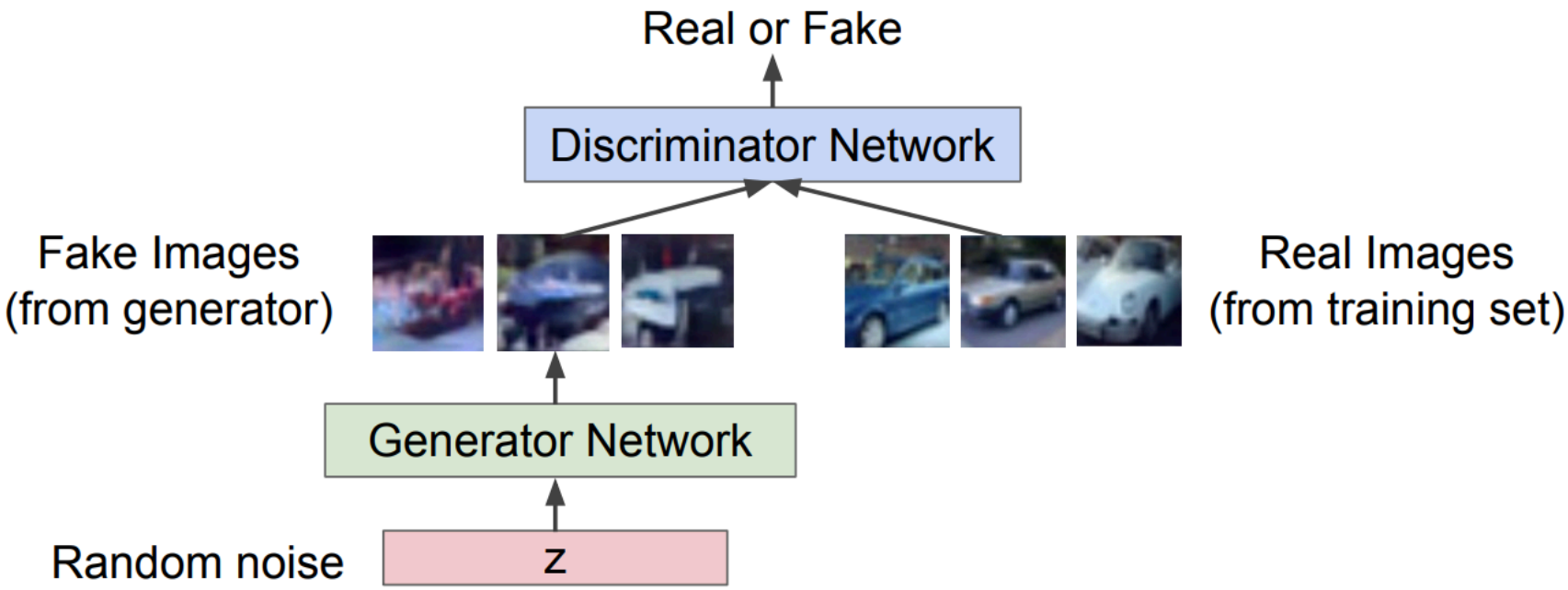
Input: Random noise

Recap: Generative Adversarial Networks

Architecture

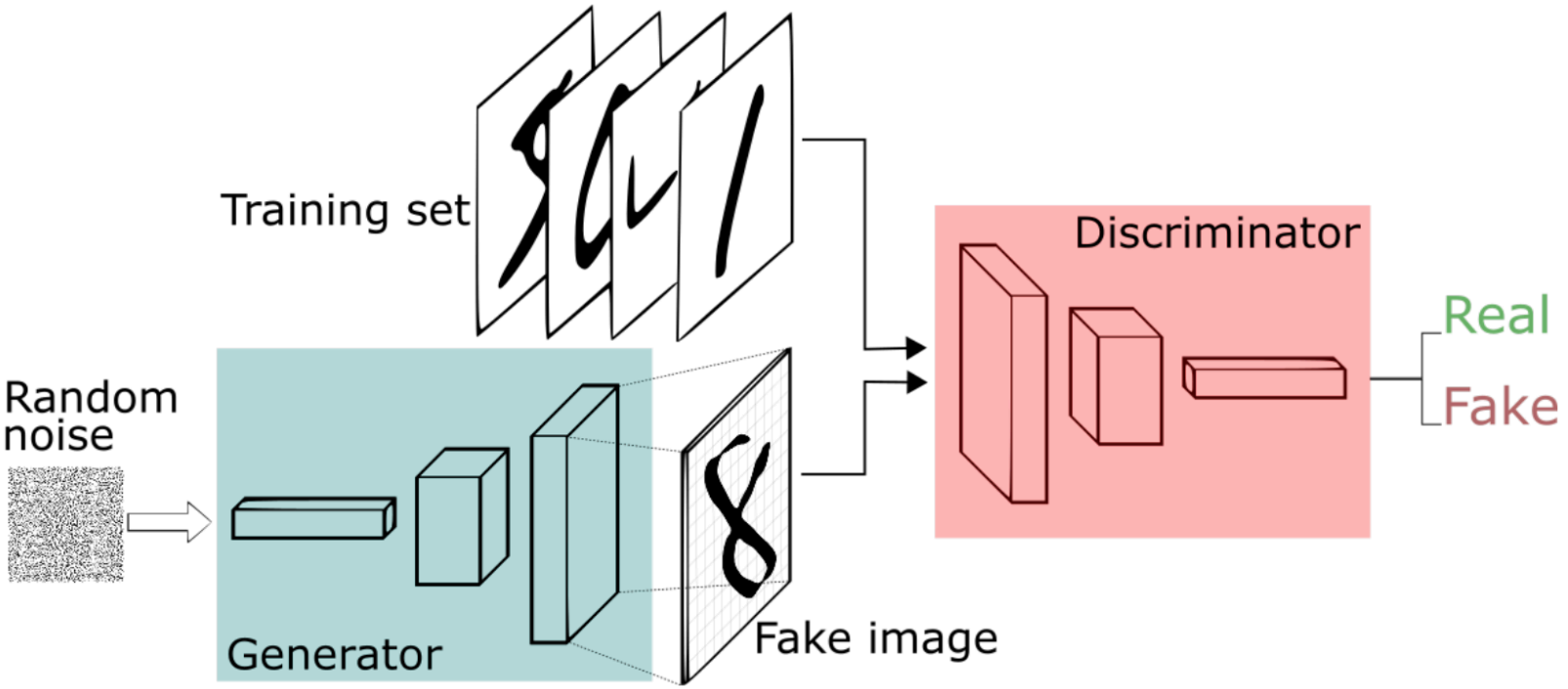
Generator network: try to fool the discriminator by generating real-looking images

Discriminator network: try to distinguish between real and fake images



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Recap: Generative Adversarial Networks



http://deeplearning.cs.cmu.edu/slides/lec17.GANs.pdf

Recap: Generative Adversarial Networks

Training process: two-player game

Train jointly in minimax game

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data x

Discriminator output for generated fake data G(z)

Discriminator (θ_d) wants to maximize objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)

Generator (θ_g) wants to minimize objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Ian Goodfellow *et al.*, "Generative Adversarial Nets" in NIPS, 2014

Recap: Generative Adversarial Networks

Training procedure

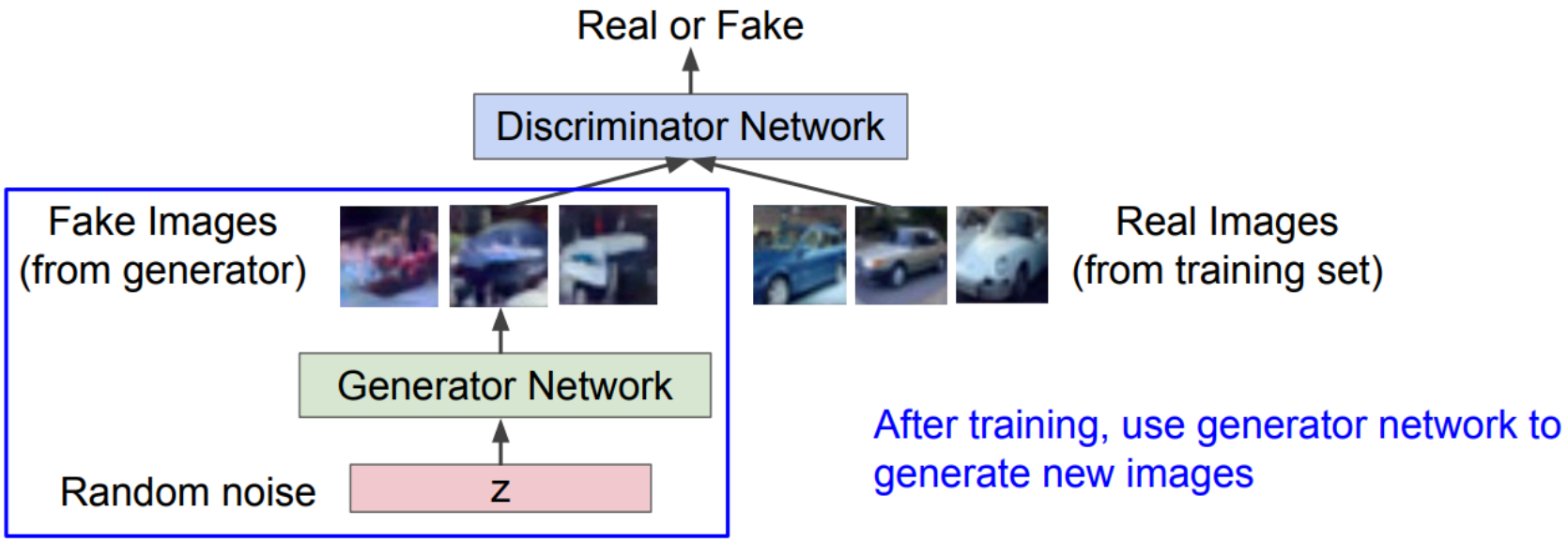
Use SGD-like algorithm of choice (Adam) on two mini-batches simultaneously:

- A mini-batch of training examples
- A mini-batch of generated examples

Optional: run *k* steps of one player for every step of the other player

Recap: Generative Adversarial Networks

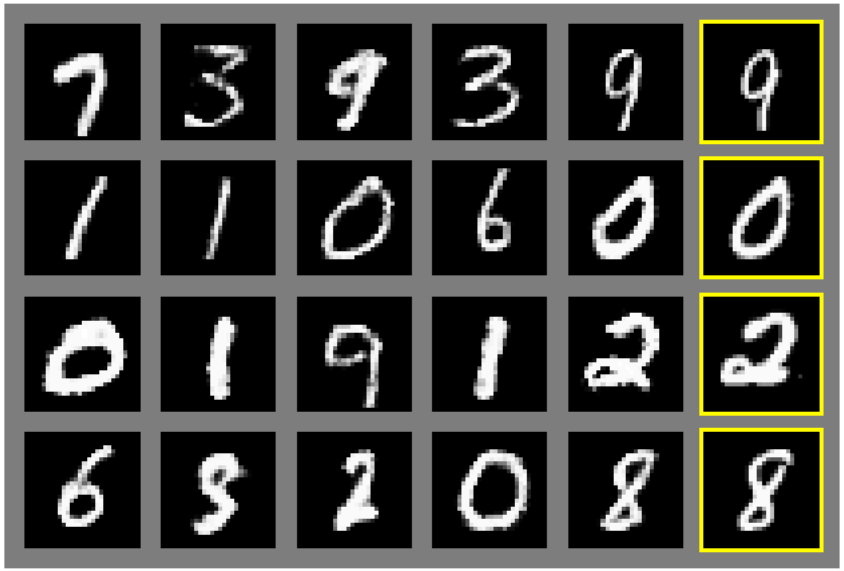
Generate new images



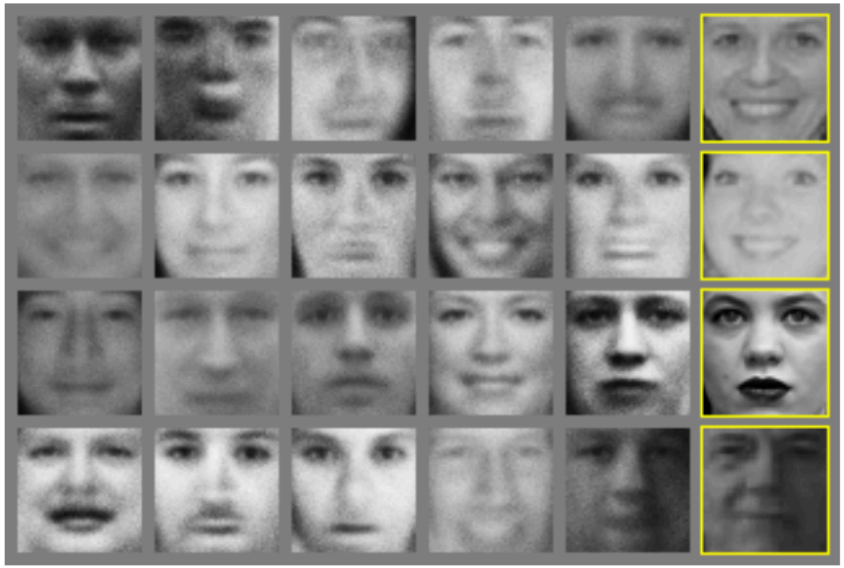
Ian Goodfellow *et al.*, "Generative Adversarial Nets" in NIPS, 2014

Recap: Generative Adversarial Networks

Generated samples



MNIST



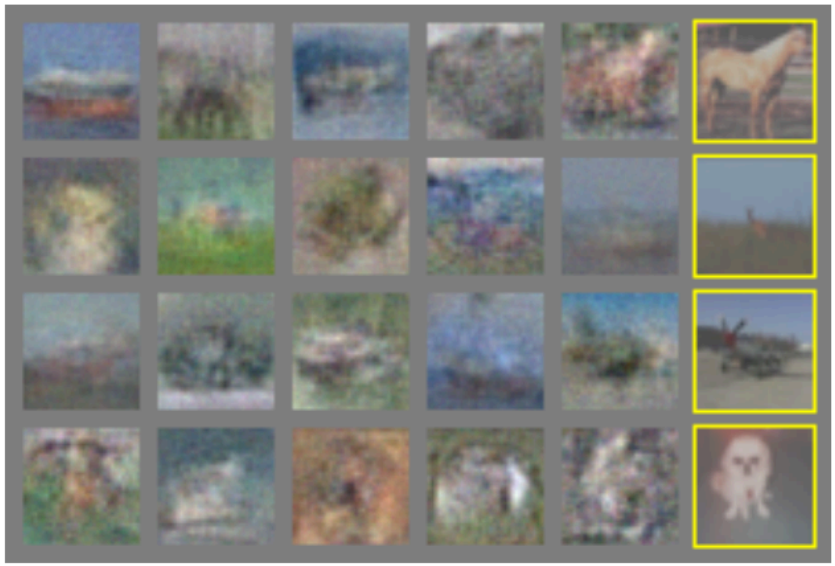
TFD

Nearest neighbor from training set

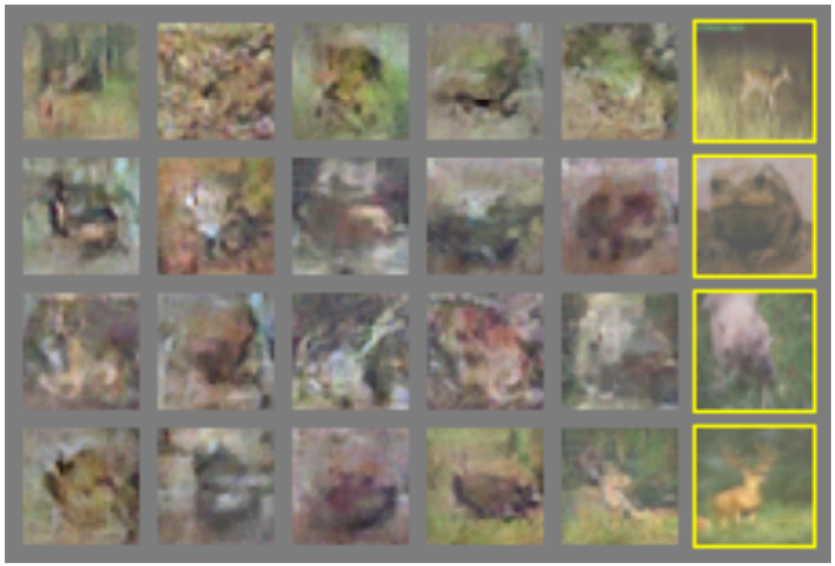
Ian Goodfellow *et al.*, "Generative Adversarial Nets" in NIPS, 2014

Recap:Generative Adversarial Networks

Generated samples



CIFAR-10 (FC)



CIFAR-10 (CNN)

Nearest neighbor from training set

Ian Goodfellow *et al.*, "Generative Adversarial Nets" in NIPS, 2014

Recap: Generative Adversarial Networks

What is the latent space like?

You can interpolate along the hidden space to produce smooth transitions of images.

Recap: Key differences between VAEs and GANs

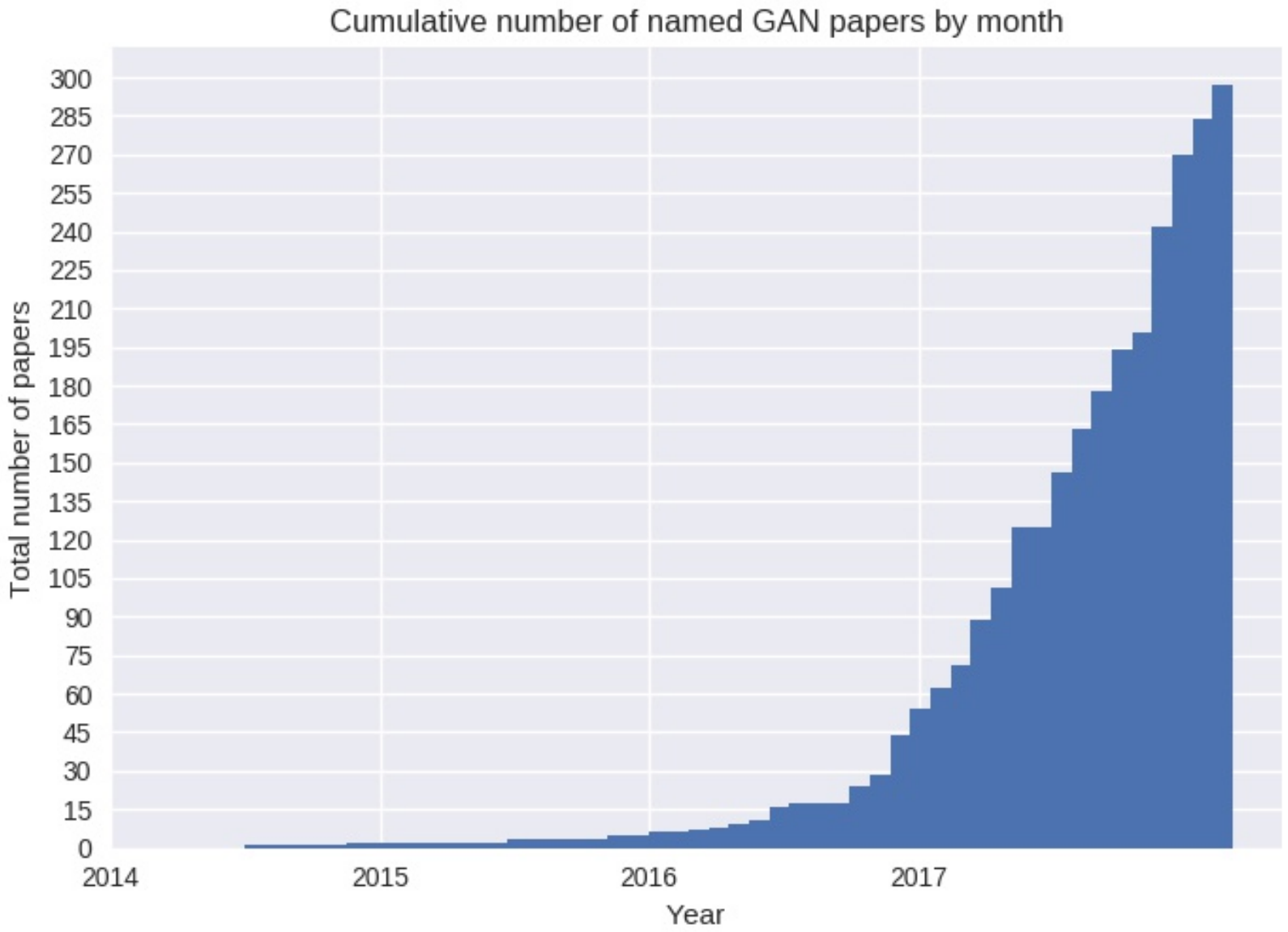
- VAEs are more theoretically grounded than GANs. GANs are more based on what works.
- GANs traditionally only learn the decoder (generator) but there are variations that learn an encoder as well; there are some problems where you want both and some problems where just the decoder will suffice. VAEs learn an encoder/decoder pair.
- GAN decoder sees samples from prior $q(z)$, VAE decoder sees samples from model $p(z \mid x)$.

Recap: Most important difference between VAEs and GANs

"Distance":

- VAE objective for the decoder is some man-made objective function, like L2 distance between images
- GAN objective for the generator is some complicated objective function defined by a neural network

Generative Adversarial Networks



https://github.com/hindupuravinash/the-gan-zoo

2017: Year of GAN

Overview

- Some GANs
- Optimization issues of GANs
- Optimization techniques of GANs
- Applications of GANs
- TensorFlow tutorial

Let’s see some GANs!

Some Generative Adversarial Networks

Conditional GAN (CGAN)

CGAN includes a label and learn $p(x|y)$

Generator learns $p(x|z,y)$

Discriminator learns $p(label|x,y)$

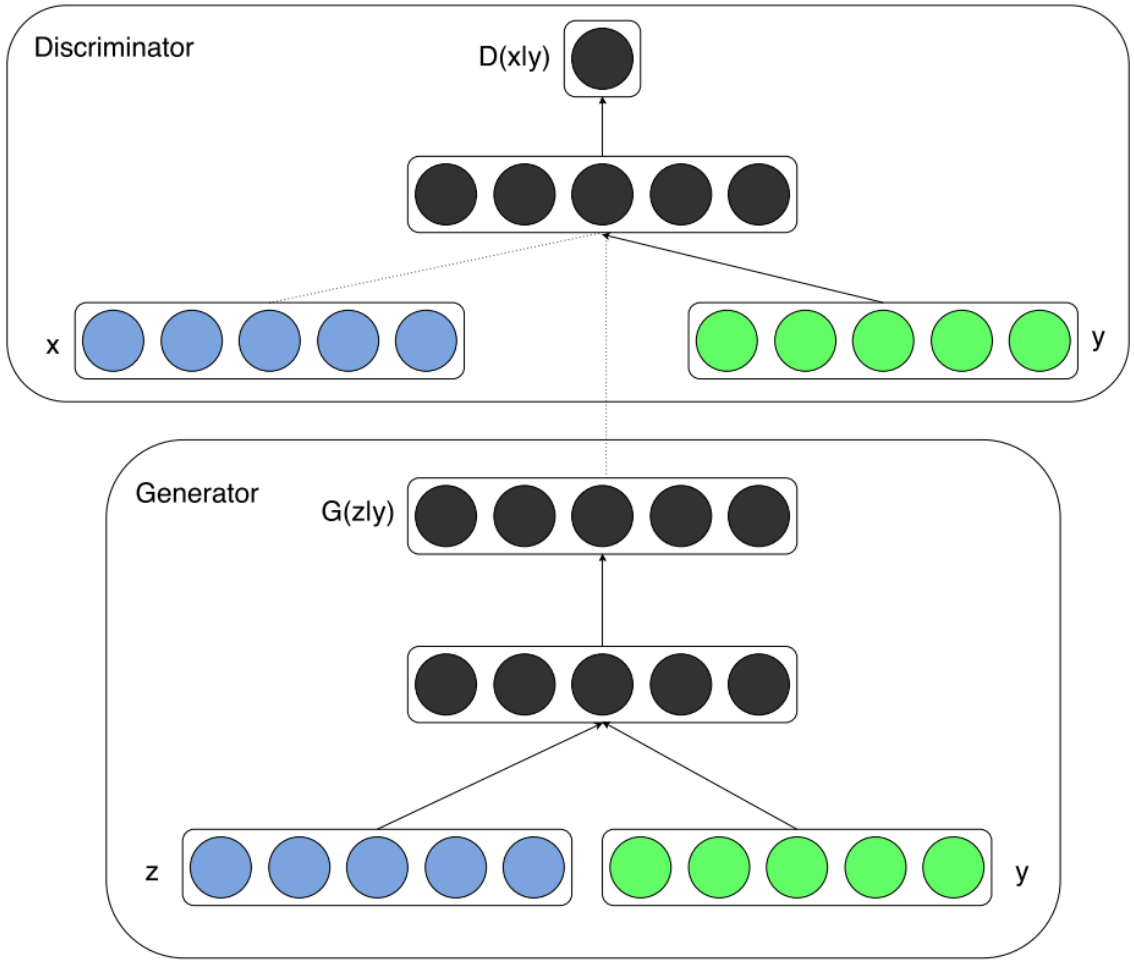
GAN loss: $\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$

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

Mirza and Osindero, “Conditional Generative Adversarial Nets”, Computer Science, 2014

Some Generative Adversarial Networks

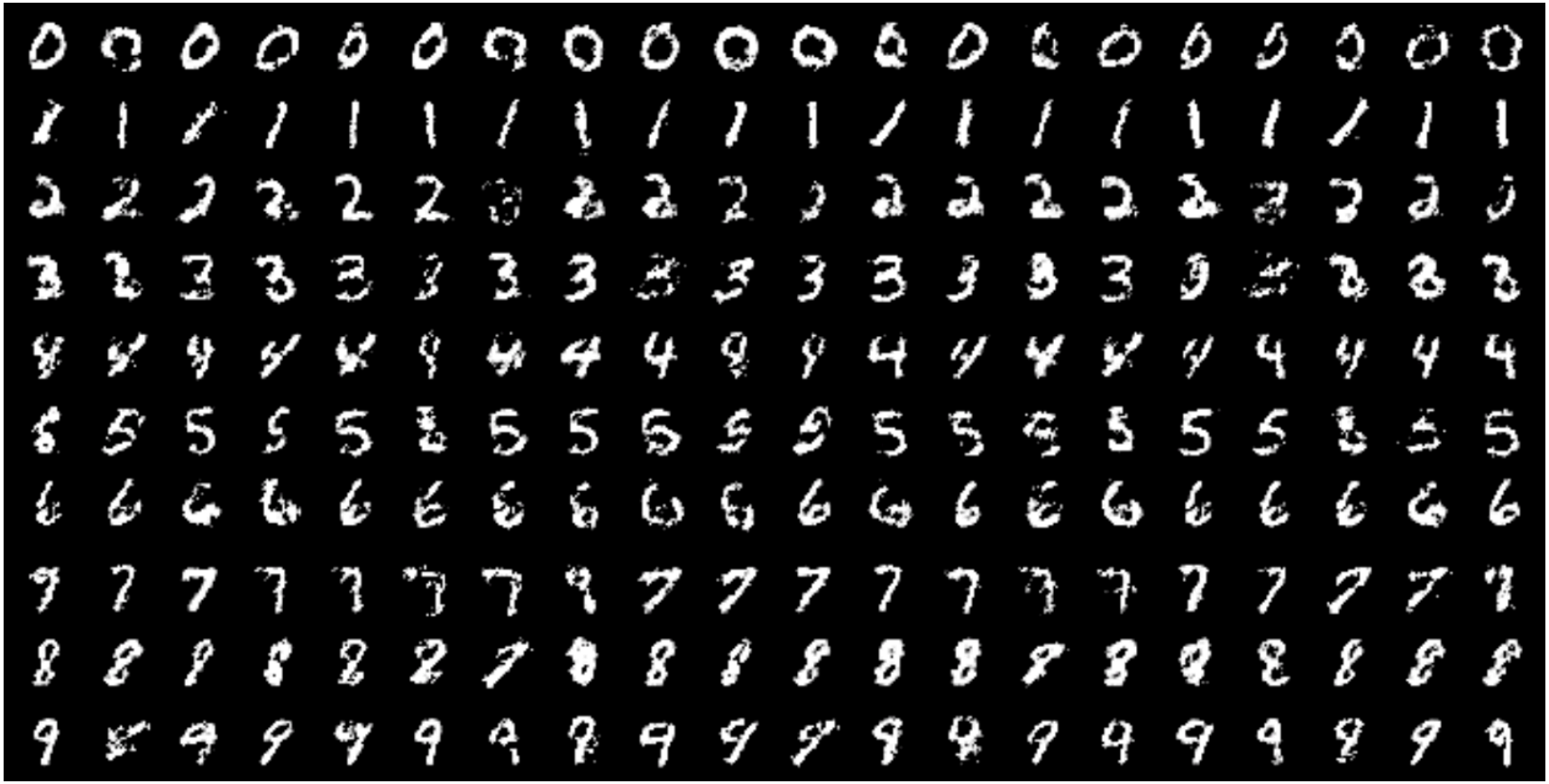
Conditional GAN (CGAN)



Mirza and Osindero, “Conditional Generative Adversarial Nets”, Computer Science, 2014

Some Generative Adversarial Networks

Conditional GAN (CGAN)



Mirza and Osindero, “Conditional Generative Adversarial Nets”, Computer Science, 2014

Some Generative Adversarial Networks

DCGAN

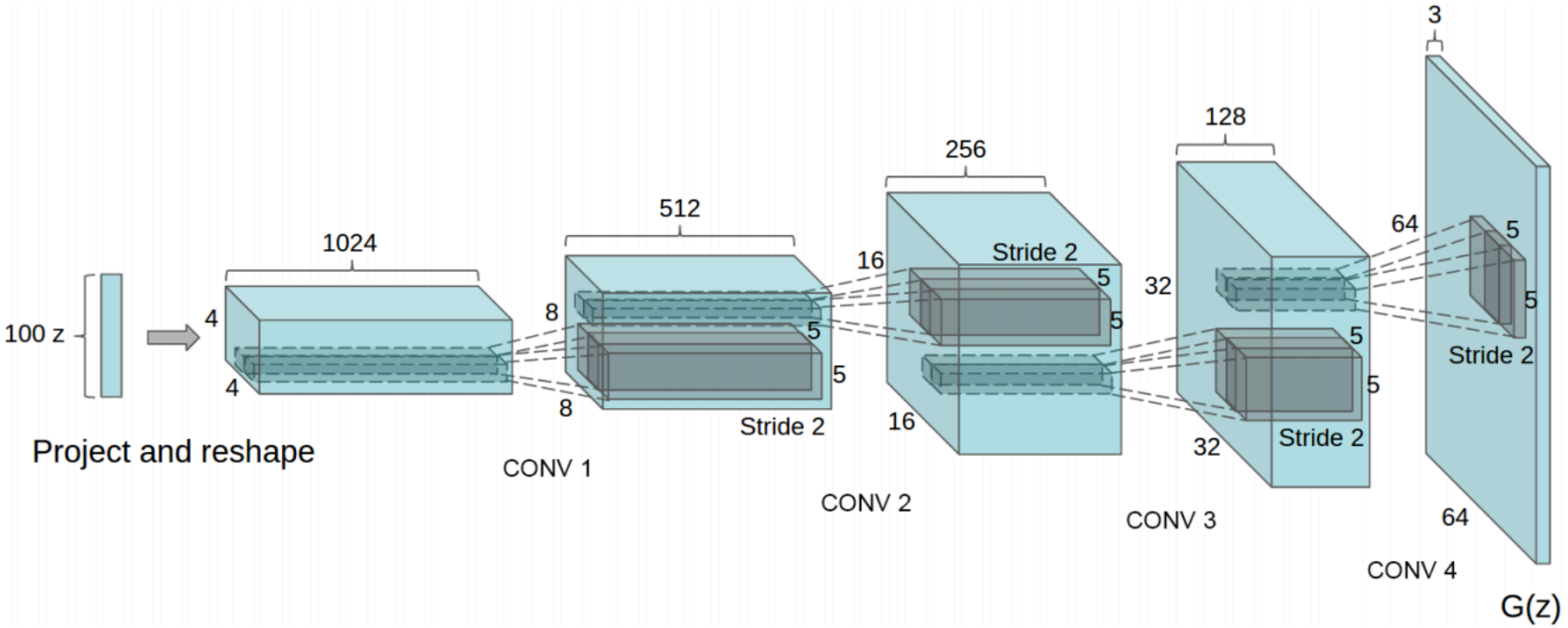
DCGAN made several improvements allowing GANs to be trained on larger/deeper CNNs

- Using Leaky ReLUs in discriminator for all layers
- Use ReLU in generator for all layers except for the output, which uses Tanh
- Using BatchNorm in both the generator and the discriminator
- Replace any pooling layers with strided convolutions
- Remove fully connected hidden layers for deeper architectures

Radford *et al.*, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" in ICLR, 2016

Some Generative Adversarial Networks

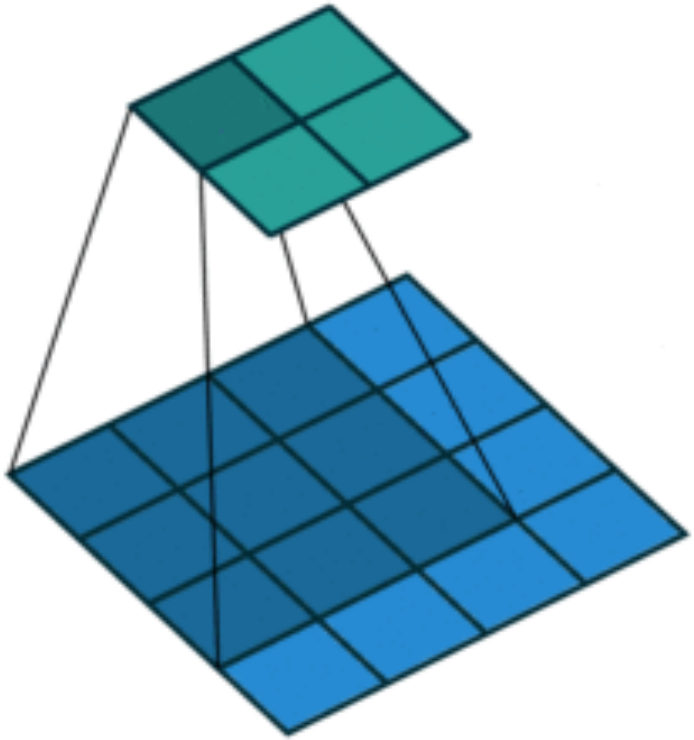
DCGAN generator



Radford *et al.*, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" in ICLR, 2016

Some Generative Adversarial Networks

Convolution



(No padding, unit strides) Convolving a 3 × 3 kernel over a 4 × 4 input using unit strides (i.e., i = 4, k = 3, s = 1 and p = 0).

GIF from https://github.com/vdumoulin/conv_arithmetic

Some Generative Adversarial Networks

Convolution

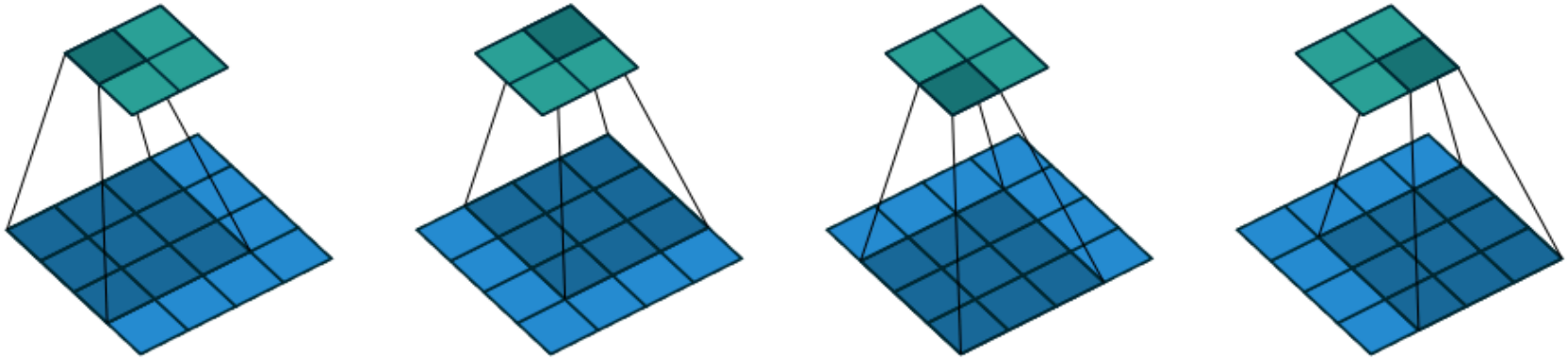


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

Vincent Dumoulin, Francesco Visin, “A guide to convolution arithmetic for deep learning” in arXiv, 2016.

Some Generative Adversarial Networks

Convolution as a matrix operation

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

$\times$

unrolled input vector (16×1)

$=$

unrolled output vector (4×1)

Vincent Dumoulin, Francesco Visin, “A guide to convolution arithmetic for deep learning” in arXiv, 2016.

Some Generative Adversarial Networks

Convolution as a matrix operation

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

C

$\times$

V_{high} unrolled input vector (16×1)

$\|$

V_{low} unrolled output vector (4×1)

Vincent Dumoulin, Francesco Visin, “A guide to convolution arithmetic for deep learning” in arXiv, 2016.

Some Generative Adversarial Networks

Convolution as a matrix operation

$$C \times V_{high} = V_{low}$$

Vincent Dumoulin, Francesco Visin, “A guide to convolution arithmetic for deep learning” in arXiv, 2016.

Some Generative Adversarial Networks

Convolution as a matrix operation

$$C \times V_{high} = V_{low}$$

$$V_{low} \overset{?}{\rightarrow} V_{high}$$

Vincent Dumoulin, Francesco Visin, “A guide to convolution arithmetic for deep learning” in arXiv, 2016.

Some Generative Adversarial Networks

Convolution vs Transposed convolution

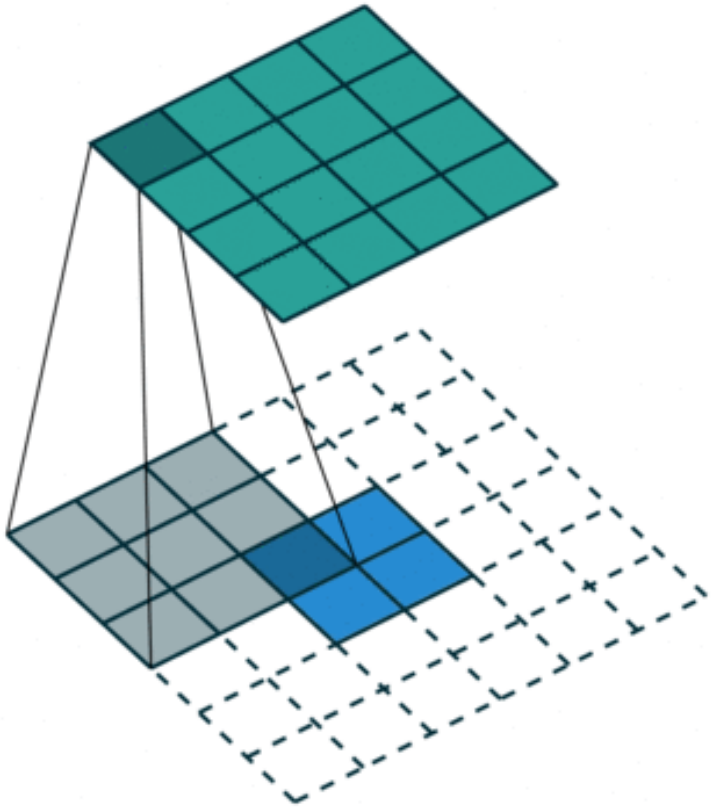
$$C \times V_{high} = V_{low}$$

$$V_{high} = C^T \times V_{low}$$

Vincent Dumoulin, Francesco Visin, “A guide to convolution arithmetic for deep learning” in arXiv, 2016.

Some Generative Adversarial Networks

Transposed convolution



GIF from https://github.com/vdumoulin/conv_arithmetic

Some Generative Adversarial Networks

Transposed convolution

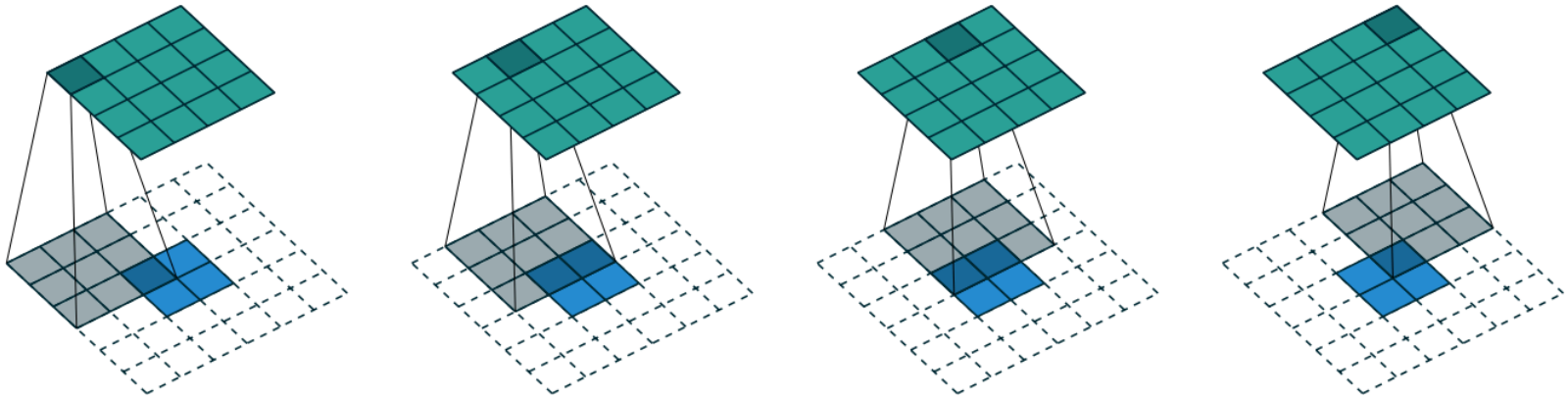


Figure 4.1: The transpose of convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input padded with a 2×2 border of zeros using unit strides (i.e., $i' = 2$, $k' = k$, $s' = 1$ and $p' = 2$).

Vincent Dumoulin, Francesco Visin, "A guide to convolution arithmetic for deep learning" in arXiv, 2016.

Some Generative Adversarial Networks

DCGAN results

Radford *et al.*, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” in ICLR, 2016

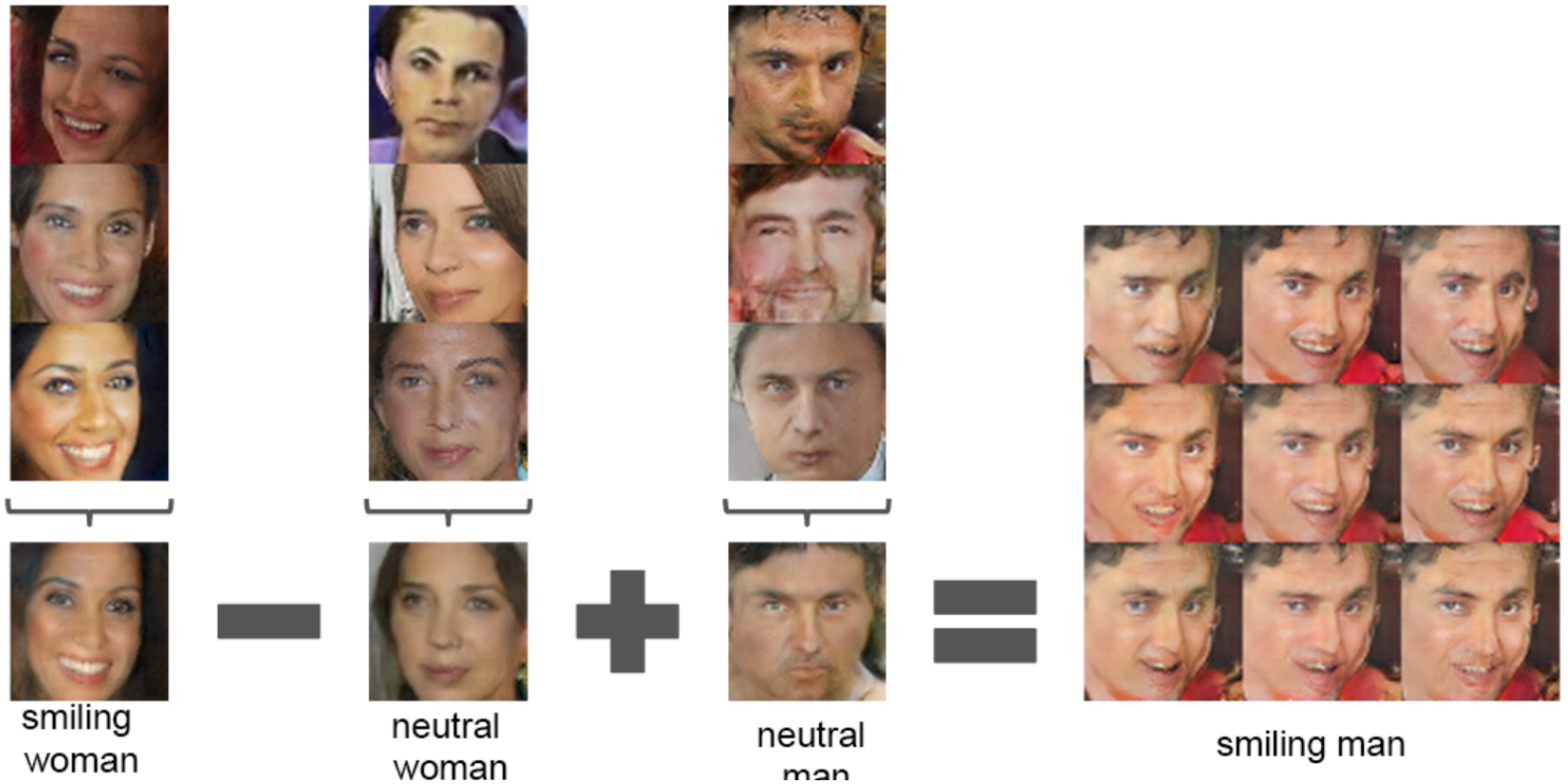
Some Generative Adversarial Networks

DCGAN results

Radford *et al.*, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” in ICLR, 2016

Some Generative Adversarial Networks

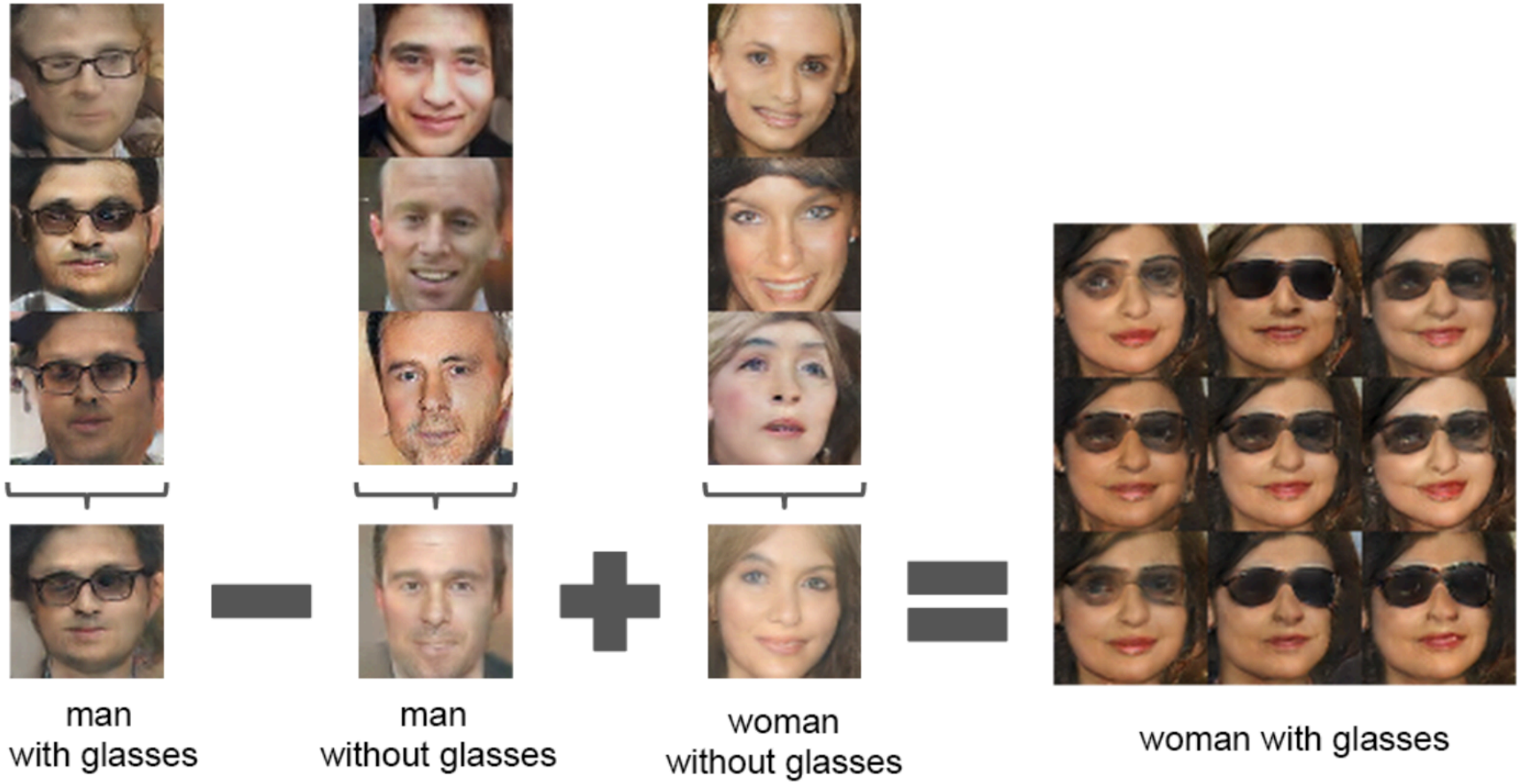
DCGAN vector math



Radford *et al.*, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" in ICLR, 2016

Some Generative Adversarial Networks

DCGAN



Radford *et al.*, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" in ICLR, 2016

Some Generative Adversarial Networks

DCGAN vector math

Results of doing the same arithmetic in pixel space

Radford *et al.*, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" in ICLR, 2016

Some Generative Adversarial Networks

DCGAN vector manipulation

Radford *et al.*, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” in ICLR, 2016

Some Generative Adversarial Networks

DCGAN online demos

Neural Face: https://carpedm20.github.io/faces/

DCGAN face generator: http://mattya.github.io/chainer-DCGAN/

MakeGirlsMoe: https://make.girls.moe/#/

Some Generative Adversarial Networks

InfoGAN

Highly entangled noise

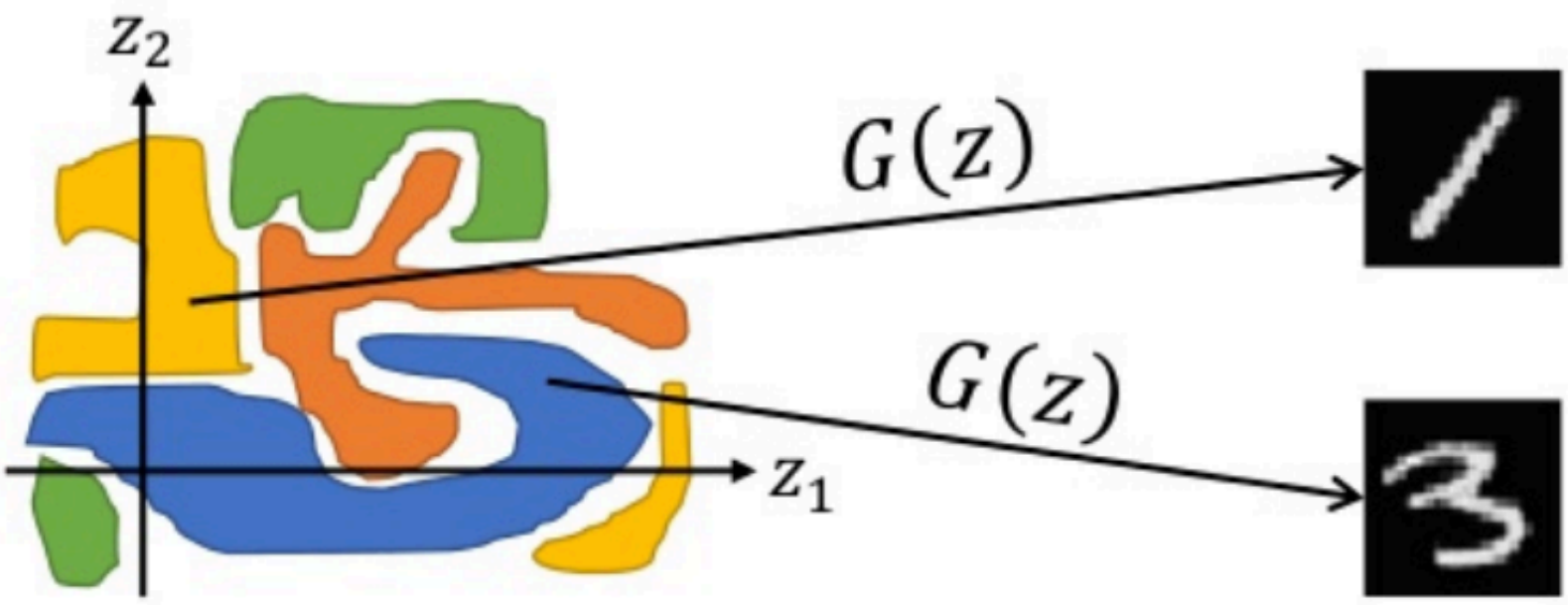


Image from https://www.slideshare.net/ShuheiYoshida2/

Some Generative Adversarial Networks

InfoGAN

Disentangled noise

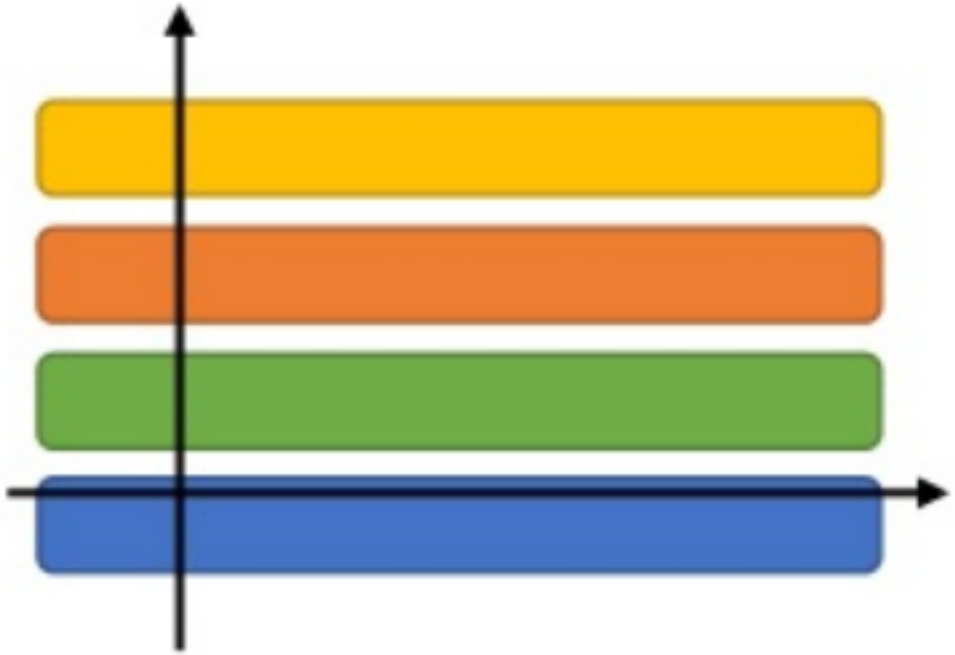


Image from https://www.slideshare.net/ShuheiYoshida2/

Some Generative Adversarial Networks

InfoGAN

Information theory

Mutual information:

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

Objective function:

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

Chen *et al*. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets" in NIPS, 2016.

Some Generative Adversarial Networks

InfoGAN architecture

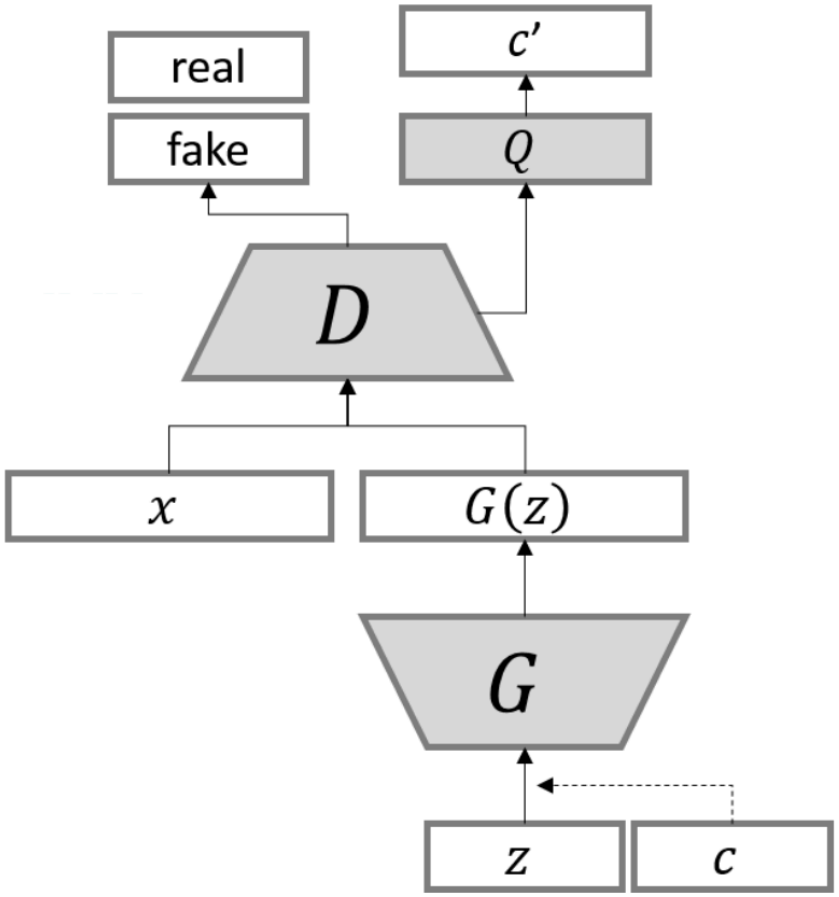


Image from https://github.com/hwalsuklee/tensorflow-generative-model-collections

Some Generative Adversarial Networks

InfoGAN results

(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

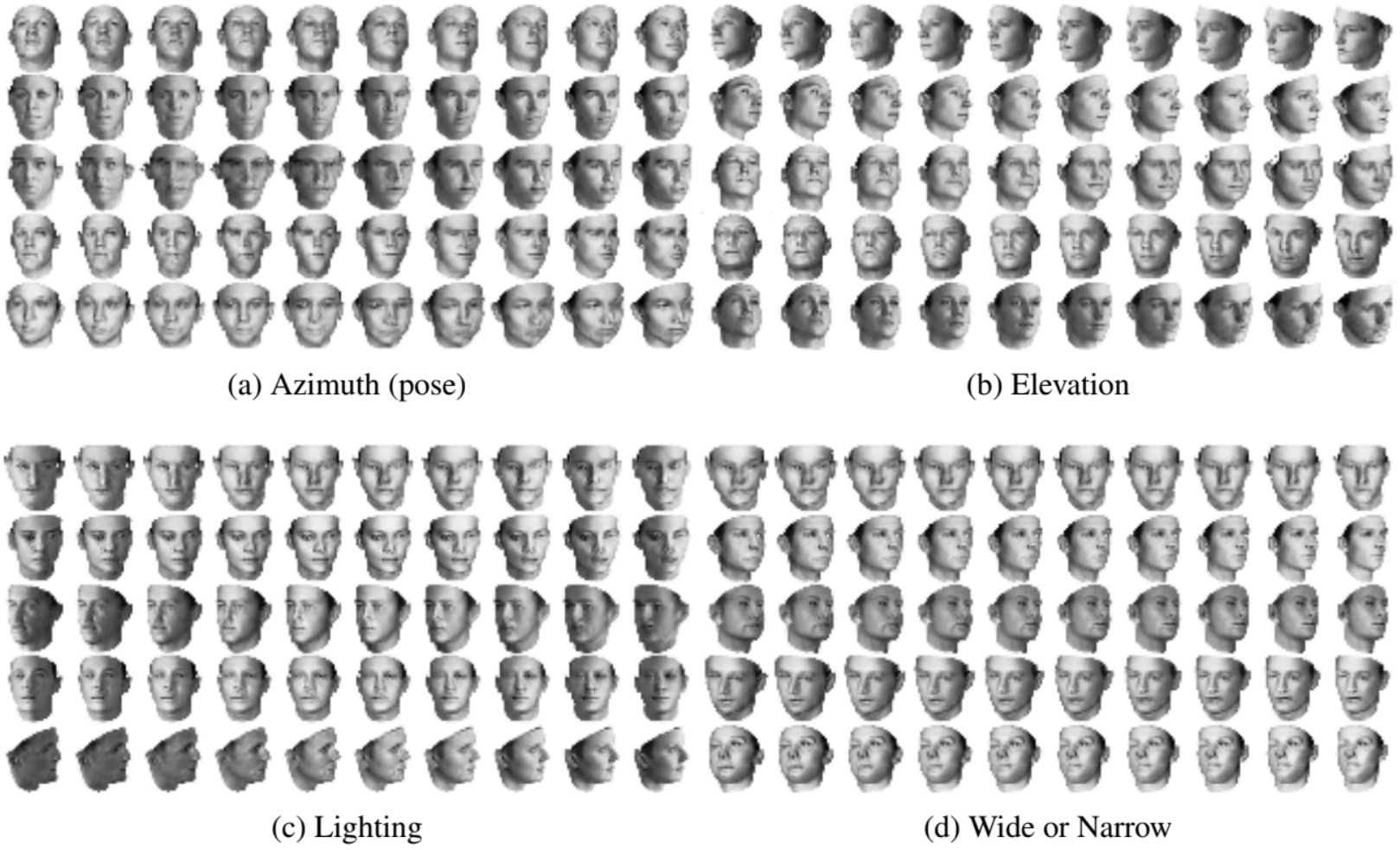
(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Chen *et al*. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets" in NIPS, 2016.

Some Generative Adversarial Networks

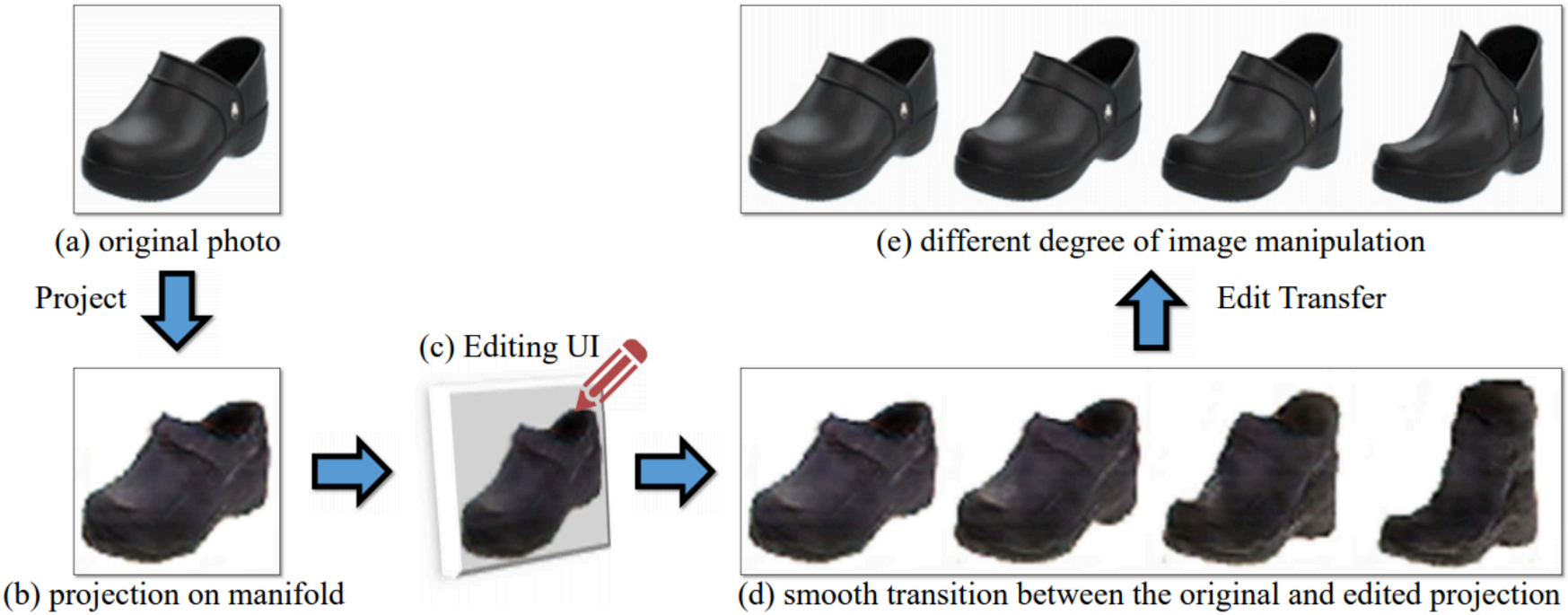
InfoGAN results



Chen *et al*. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets" in NIPS, 2016.

Some Generative Adversarial Networks

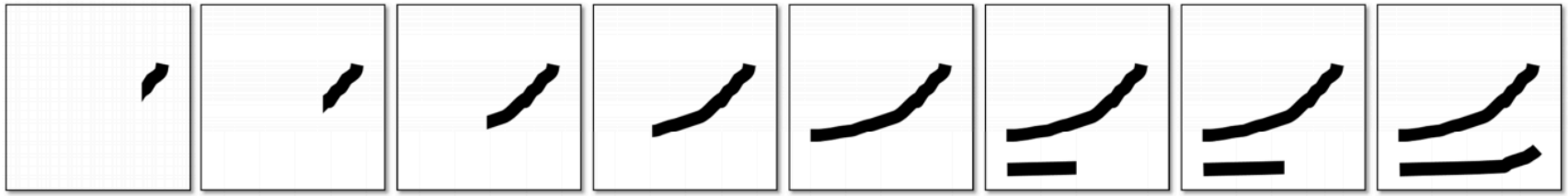
iGAN



Zhu *et al.*, "Generative Visual Manipulation on the Natural Image Manifold" in ECCV 2016.

Some Generative Adversarial Networks

iGAN



(a) User constraints v_g at different update steps

$G(z_0)$ (b) Updated images according to user edits $G(z_1)$

(c) Linear interpolation between $G(z_0)$ and $G(z_1)$

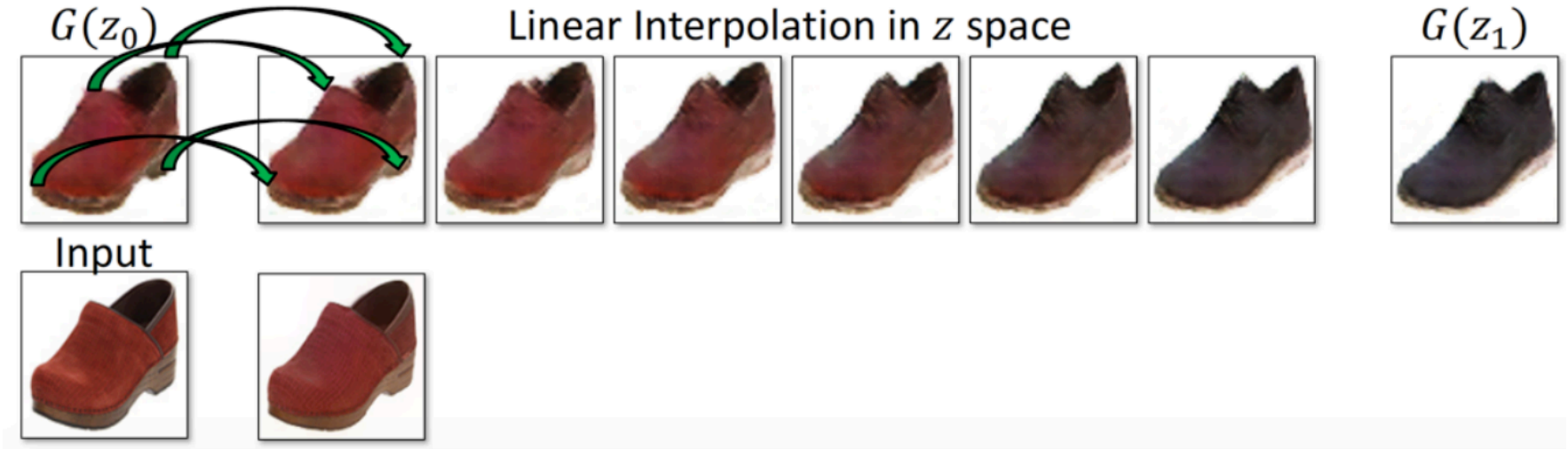
Zhu *et al.*, "Generative Visual Manipulation on the Natural Image Manifold" in ECCV 2016.

Some Generative Adversarial Networks

iGAN

Motion (u, v)+ Color ($A_{3\times4}$): estimate per-pixel geometric and color variation

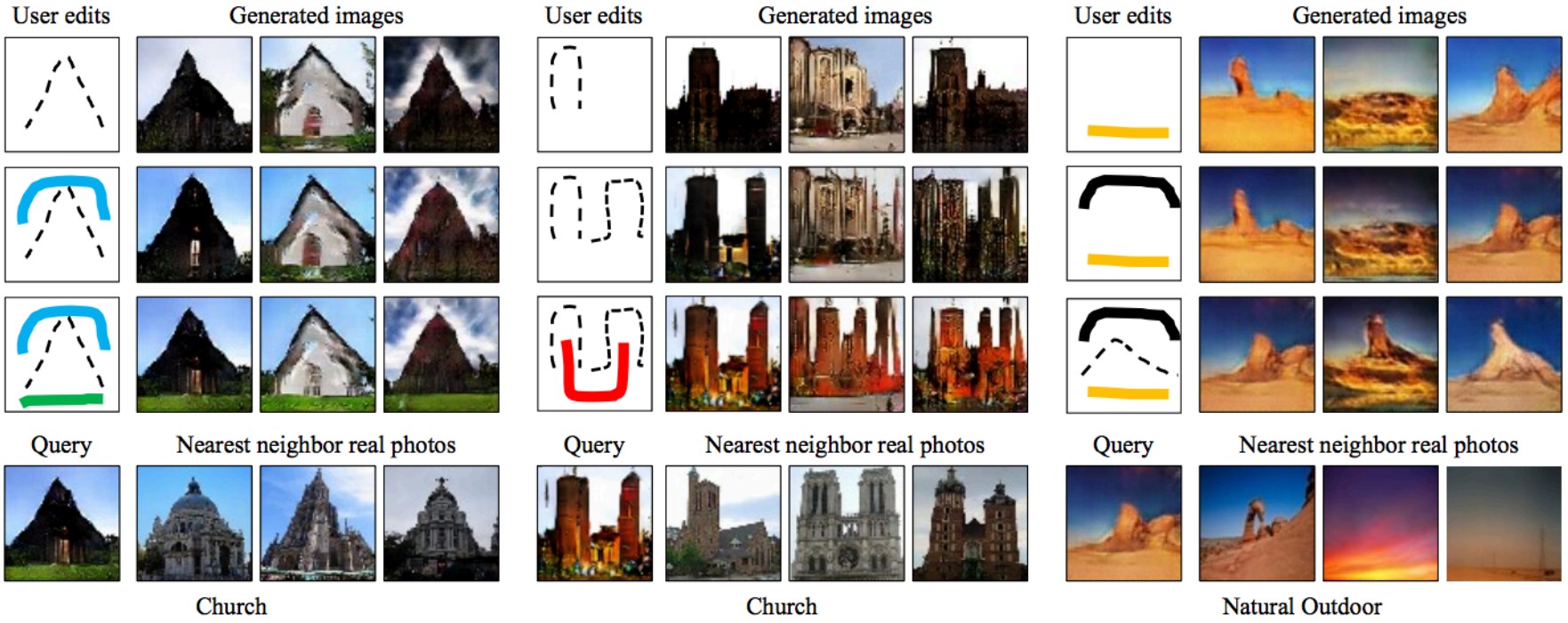
$$\iint \underbrace{\|I(x,y,t) - A \cdot I(x+u, y+v, t+1)\|^2}_{\text{data term}} + \underbrace{\sigma_s(\|\nabla u\|^2 + \|\nabla v\|^2)}_{\text{spatial reg}} + \underbrace{\sigma_c \|\nabla A\|^2}_{\text{color reg}} dxdy$$



Zhu *et al.*, "Generative Visual Manipulation on the Natural Image Manifold" in ECCV 2016.

Some Generative Adversarial Networks

iGAN



Zhu *et al.*, "Generative Visual Manipulation on the Natural Image Manifold" in ECCV 2016.

Some Generative Adversarial Networks

iGAN

Zhu *et al.*, "Generative Visual Manipulation on the Natural Image Manifold" in ECCV 2016.

Too good to be true?

Tricky to train

Original

Generated

李嫣然, “深入浅出 GAN 原理篇”, 2017

The major roadblocks with GANs: various optimization issues

Causes of optimization issues

- Non convergence
- Simultaneous updates require a careful balance between the two players
- There is a stationary point but no guarantee of reaching it
- Mode collapse
- Adversarial optimization is a more general, harder problem than single-player optimization
- Discriminator is highly nonlinear, gradient tends to be noisy or non-informative

Common failures

Mode collapse

Generated points tend to ”herd” to probable regions, causing ”mode collapse”.

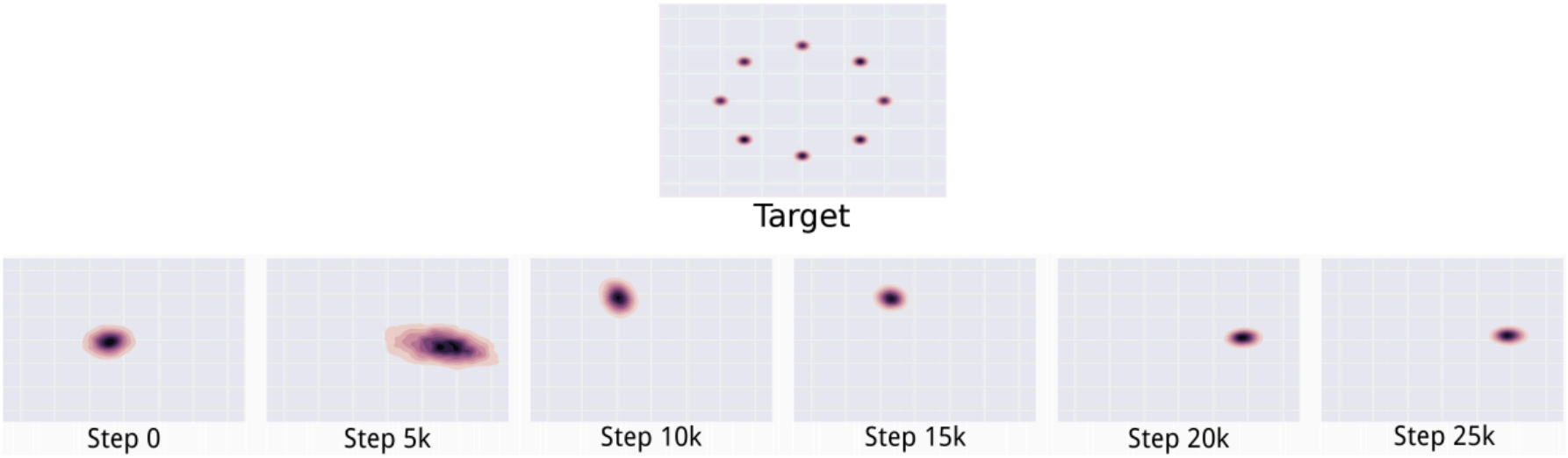
”Mode collapse” GAN generates a subspace really well but doesn’t cover the entire real distribution. For example, train on MNIST and it only generates threes and eights.

https://www.youtube.com/watch?v=ktxhiKhWoEE

Common failures

Looks like “mode collapse”

Sometimes GANs enter into clear cycles. They seem to generate a single digit relatively well, then start generating a different digit, etc. Looks like ”mode collapse” on a rotating set of samples, but it does not differentiate.



Common failures

Hard to describe failures

Sometimes hard to describe failures, but videos like this are relatively typical.

https://www.youtube.com/watch?v=D5akt32hsCQ

Optimization Techniques

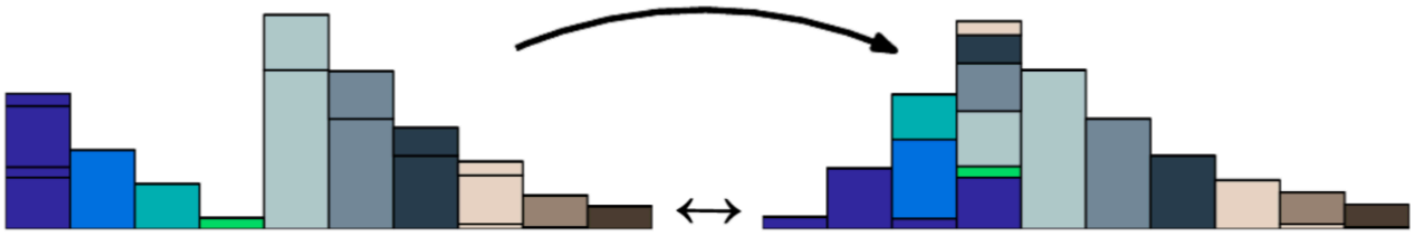
Optimization techniques

- There are many tricks to train them better but some of those tricks do away with what makes GANs so special
- Not every trick works all the time or in combination with other tricks
- Most papers claim to have the golden bullet
- Best current solution is really a combination of techniques

Optimization techniques

- Unrolled GANs, Metz et al. (2016)
- Improved Techniques for Training GANs, Salimans et al. (2016)
- Least-Squares GAN, Mao et al. (2016)
- Amortised MAP Inference, Kaae Sønderby et al. (2016)
- EBGAN, Zhao et al. (2016)
- WGAN, Arjovsky et al. (2017)
- WGAN-GP, Gulrajani et al. (2017)
- GAN optimization is locally stable, Nagarajan and Kolter (2017)
- Numerics of GANs, Mescheder et al. (2017)
- DRAGAN, Kodali et al. (2017)
- Progressive GAN, Karras et al. (2017)
- Lagrangian GAN, Chen et al. (2018)

Wasserstein GAN (WGAN)



P_r P_g

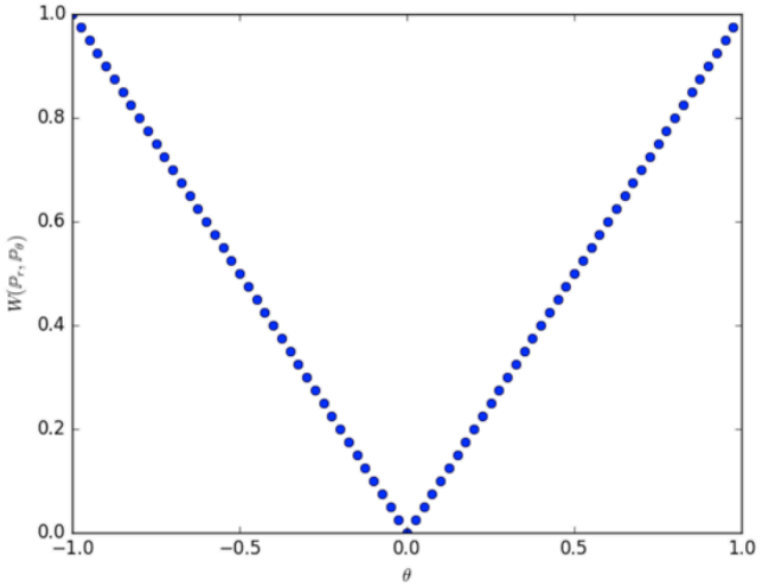
Wasserstein-1 Distance (Earth-Mover Distance):

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\| \right]$$

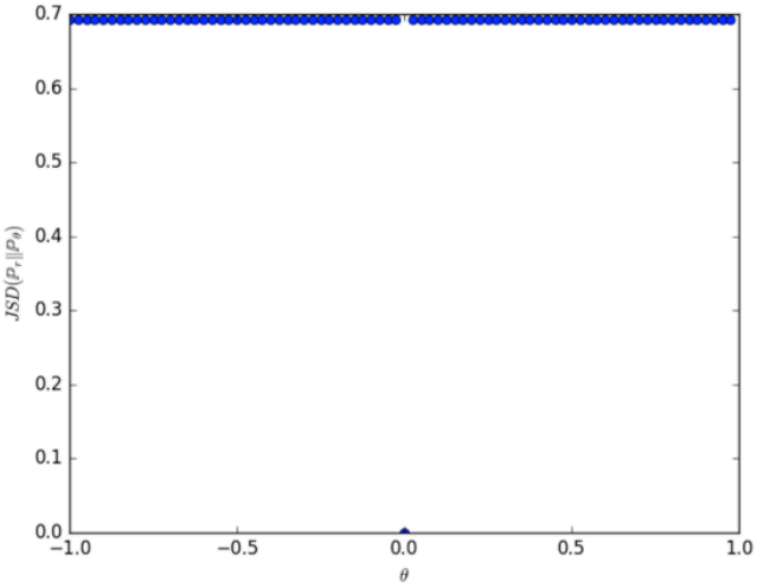
Arjovsky *et al*. "Wasserstein GAN " in arXiv, 2017.

Wasserstein GAN (WGAN)

EM vs JS



Wasserstein Distance



JS Divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m)$$

Arjovsky *et al.* "Wasserstein GAN " in arXiv, 2017.

Wasserstein GAN (WGAN)

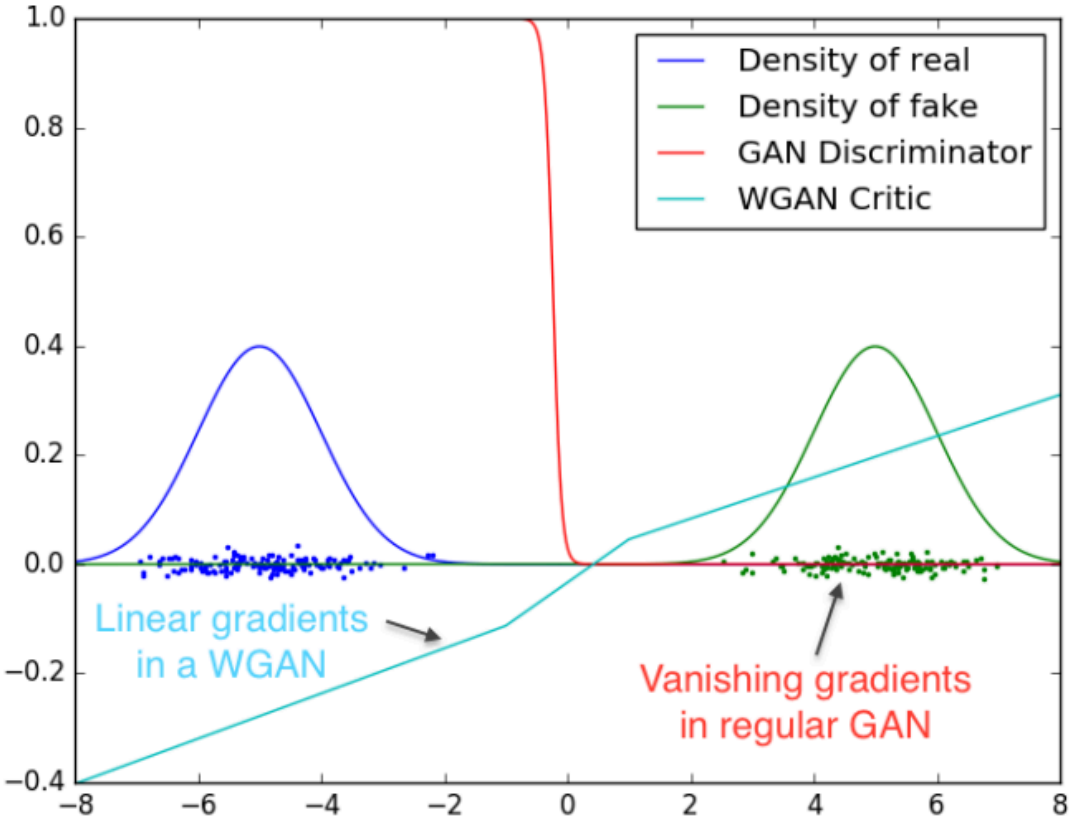


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

Arjovsky *et al*. "Wasserstein GAN " in arXiv, 2017.

Wasserstein GAN (WGAN)

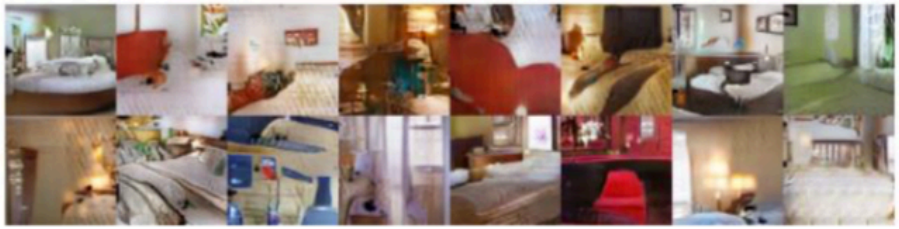
- Claims more stability than traditional GAN
- Loss value is more meaningful and corresponds to how good samples are
- Less tuning required in terms of generator/discriminator balance
- In practice, doesn't actually work so well (see Improved WGAN and WGAN-GP)

Arjovsky *et al*. "Wasserstein GAN " in arXiv, 2017.

Wasserstein GAN (WGAN)

GAN & DCGAN

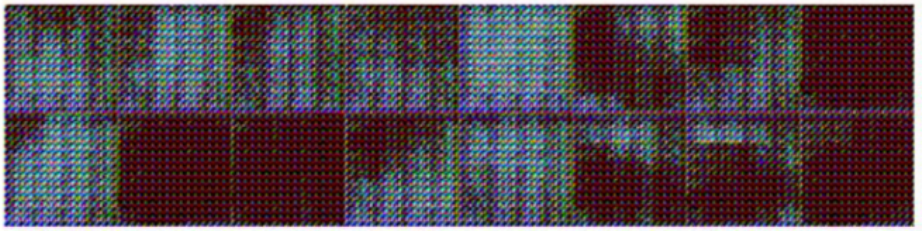
WGAN



DCGAN

Without BN

WGAN



DCGAN

Arjovsky *et al*. “Wasserstein GAN ” in arXiv, 2017.

Progressive GAN

Architecture

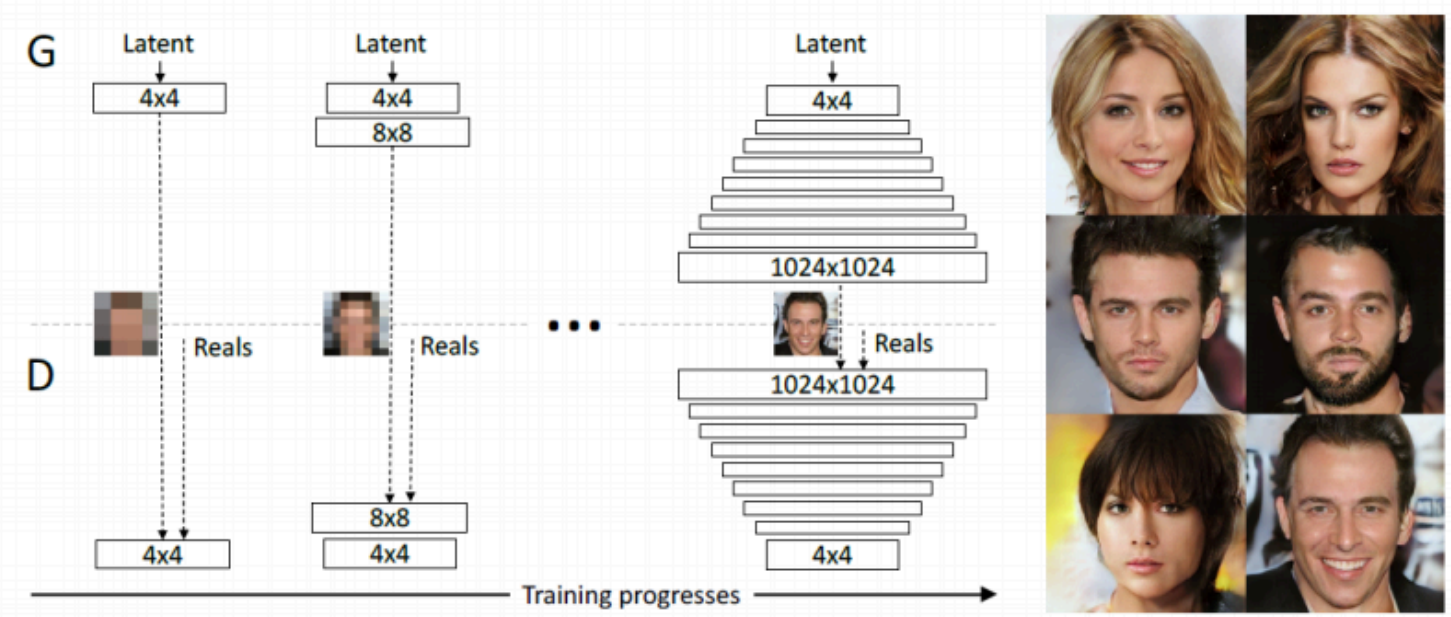


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

Karras *et al*., "Progressive Growing of GANs for Improved Quality, Stability, and Variation" in ICLR, 2018.

Progressive GAN

Video demo: https://www.youtube.com/watch?v=XOxxPcy5Gr4

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

Karras *et al*., "Progressive Growing of GANs for Improved Quality, Stability, and Variation" in ICLR, 2018.

Image-to-Image Translation

Image-to-Image Translation

- pix2pix
- CycleGAN
- UNIT
- MUNIT

pix2pix

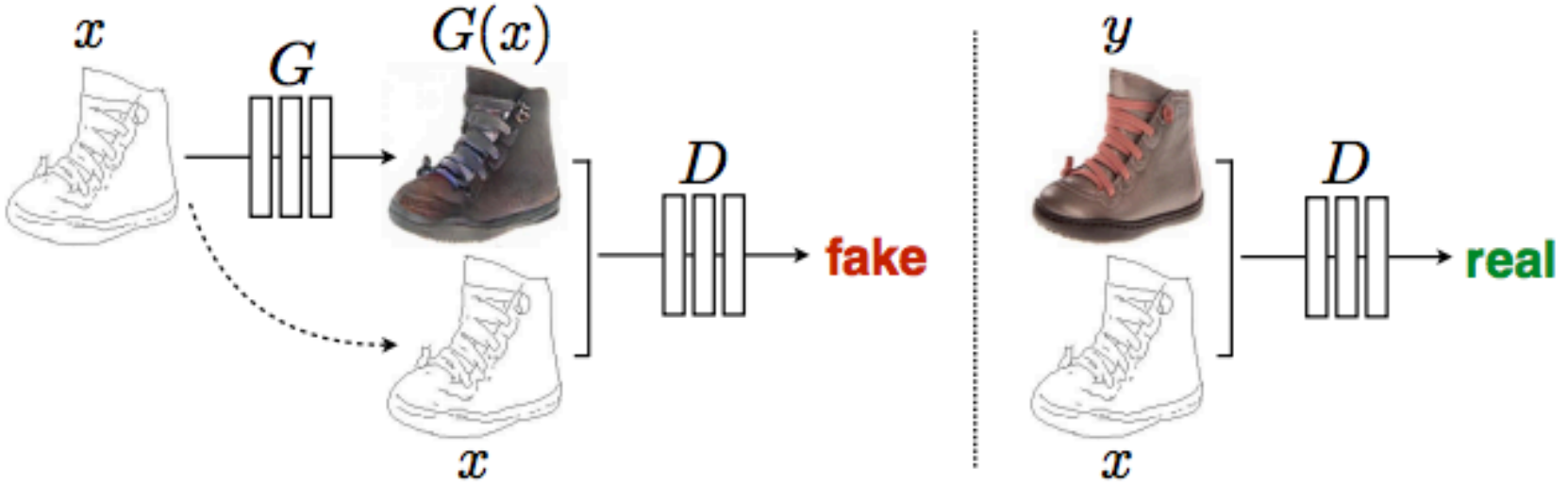


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

Isola *et al.*, "Image-to-Image Translation with Conditional Adversarial Networks" in CVPR, 2017.

pix2pix

Generator

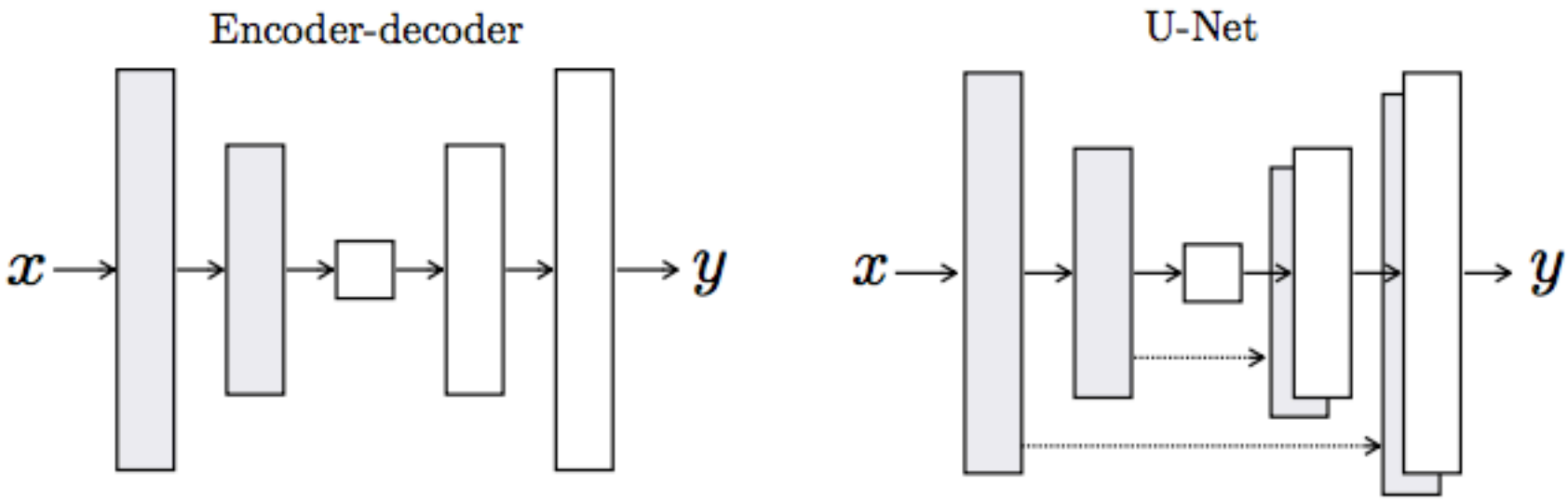


Figure 3: Two choices for the architecture of the generator. The "U-Net" [49] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Isola *et al.*, "Image-to-Image Translation with Conditional Adversarial Networks" in CVPR, 2017.

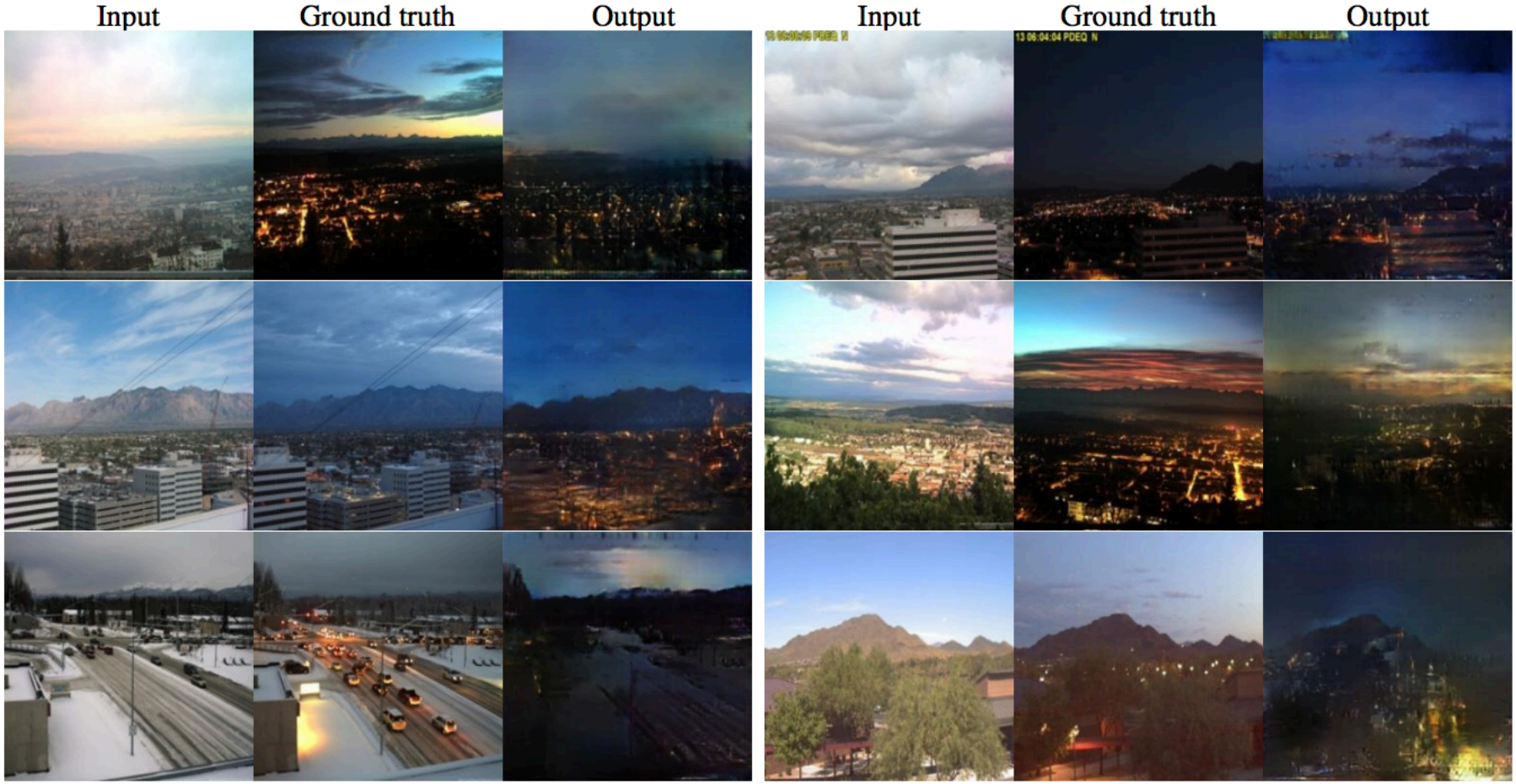
pix2pix

Map ←→ Aerial photo

Isola *et al.*, "Image-to-Image Translation with Conditional Adversarial Networks" in CVPR, 2017.

pix2pix

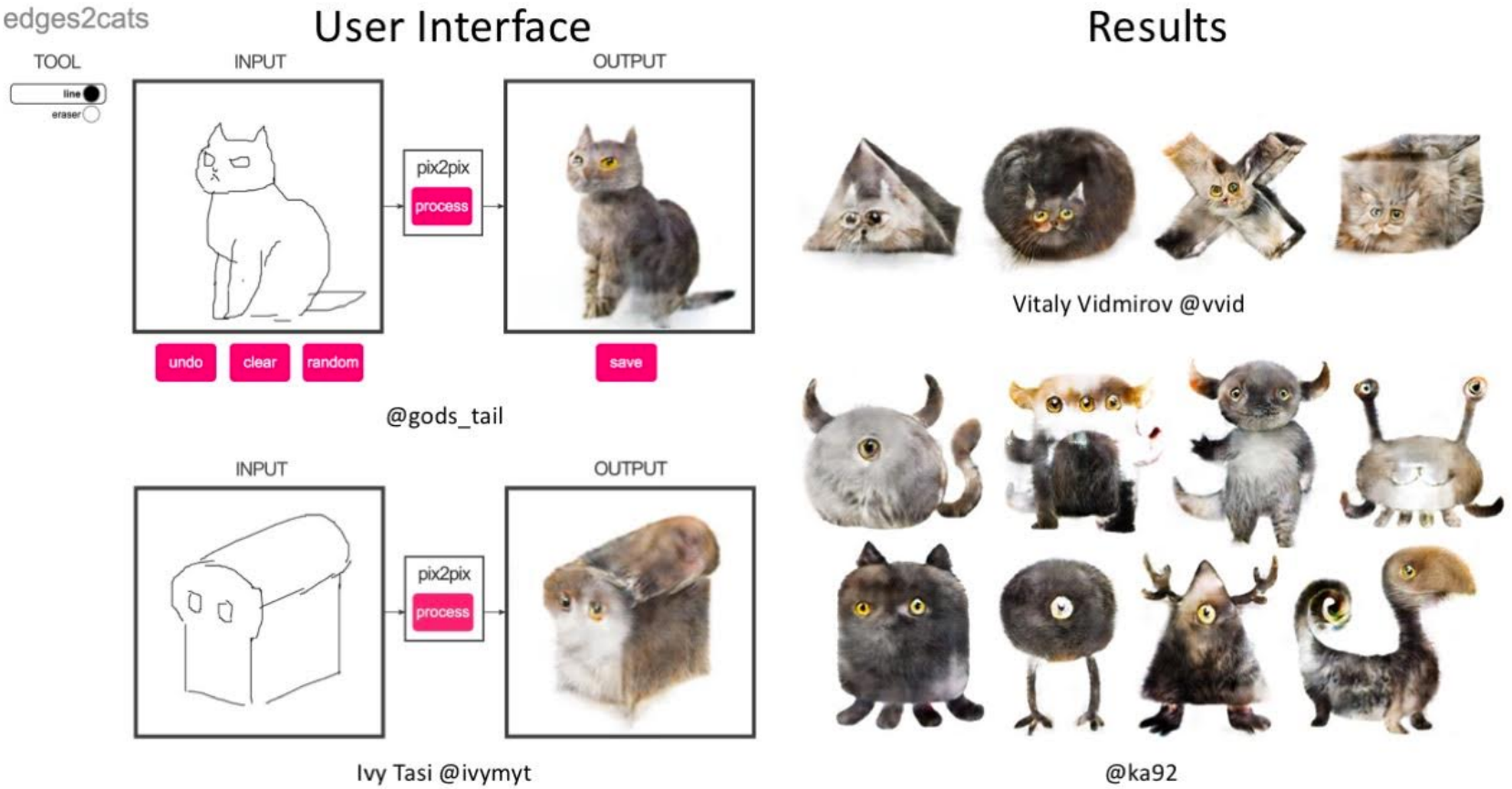
Day ←→ Night



Isola *et al.*, "Image-to-Image Translation with Conditional Adversarial Networks" in CVPR, 2017.

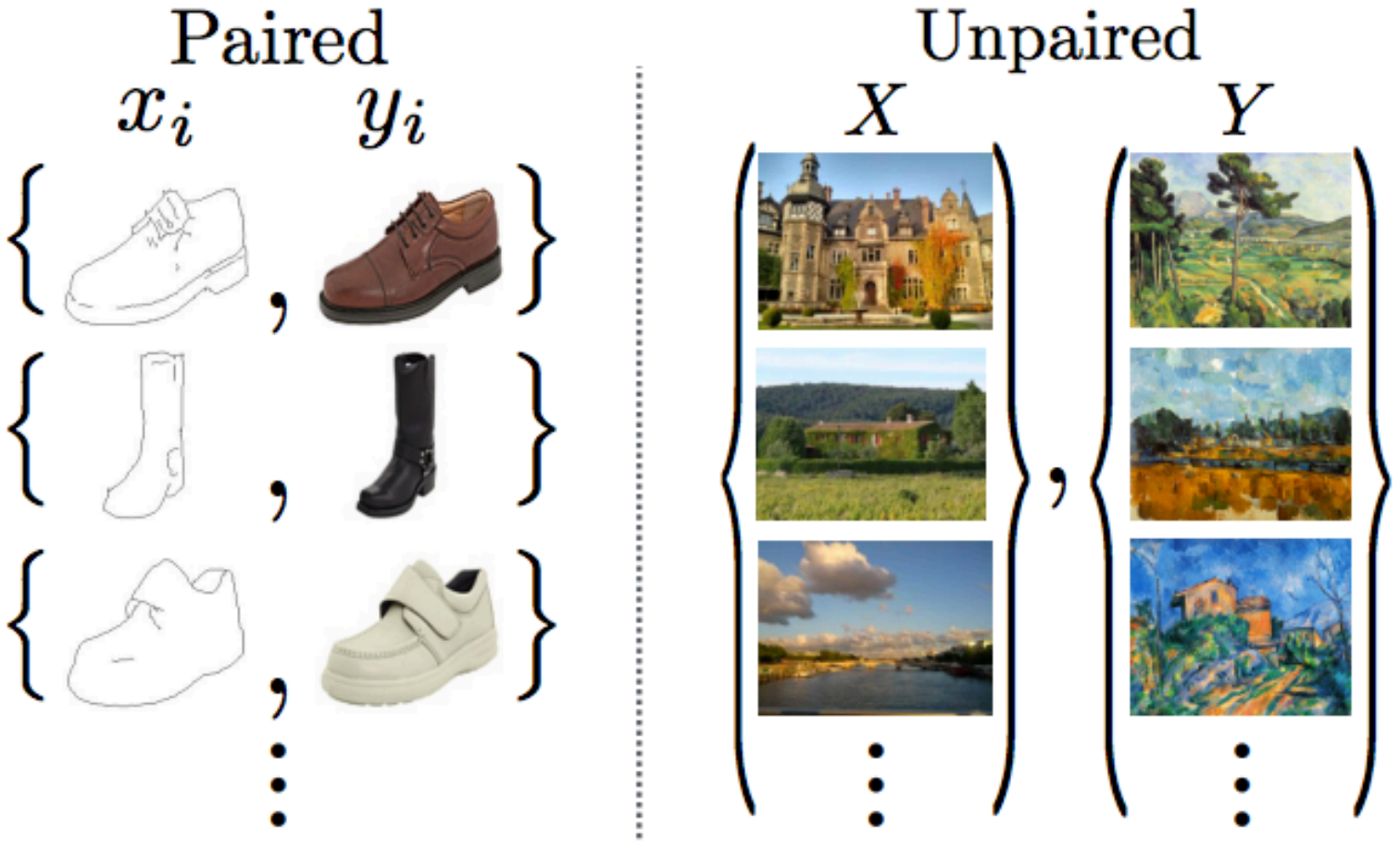
pix2pix

Online demo: https://affinelayer.com/pixsrv/



Isola *et al.*, "Image-to-Image Translation with Conditional Adversarial Networks" in CVPR, 2017.

CycleGAN



Zhu *et al*., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss" in ICCV, 2017.

CycleGAN

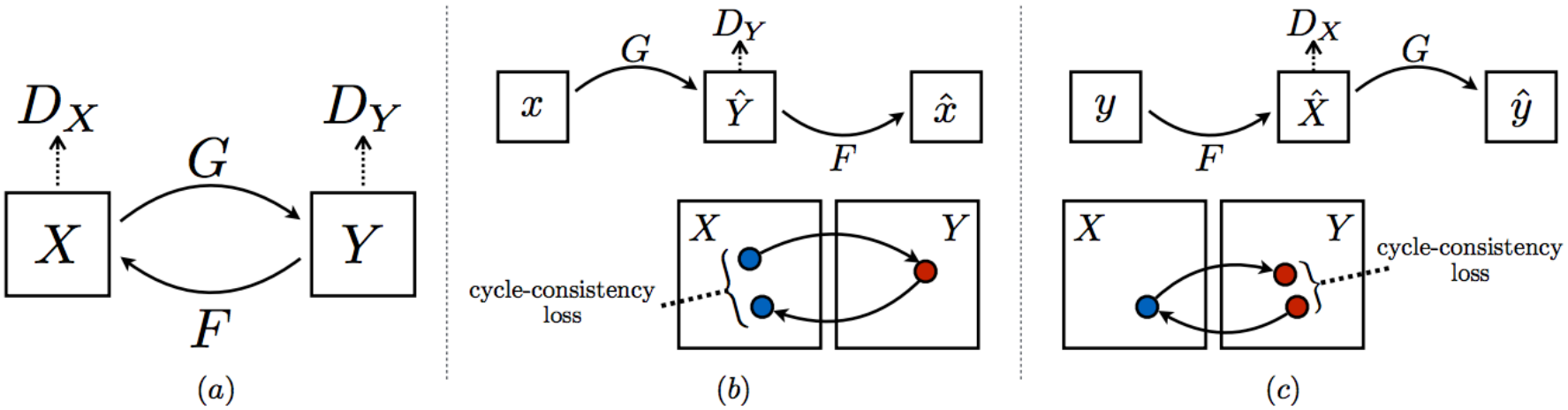


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

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

Zhu *et al.*, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss" in ICCV, 2017.

CycleGAN

horse ←→ zebra

Zhu *et al*., “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss” in ICCV, 2017.

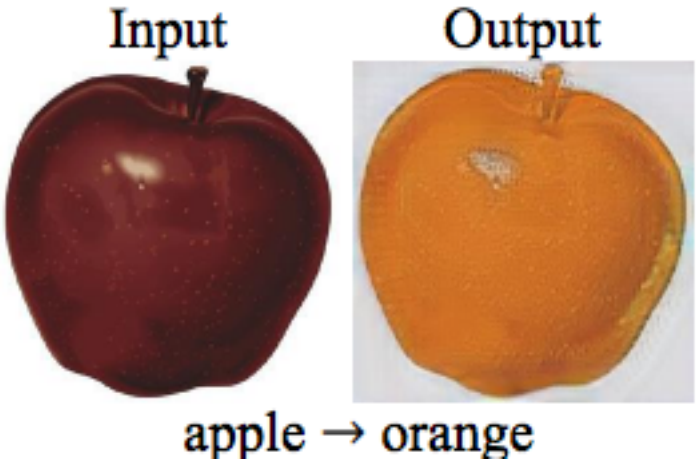
CycleGAN

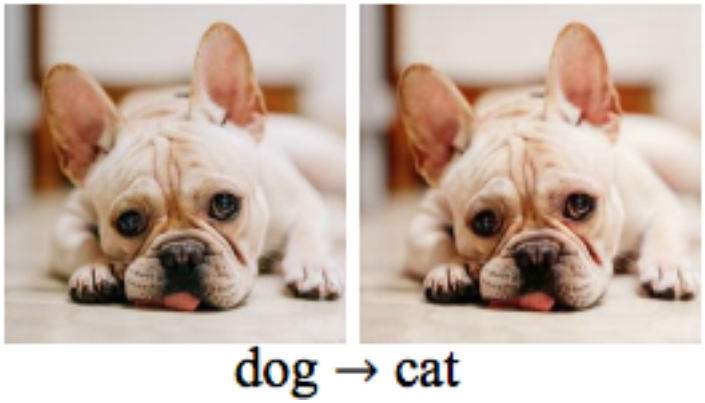
photo ←→ drawing

Zhu *et al*., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss" in ICCV, 2017.

CycleGAN

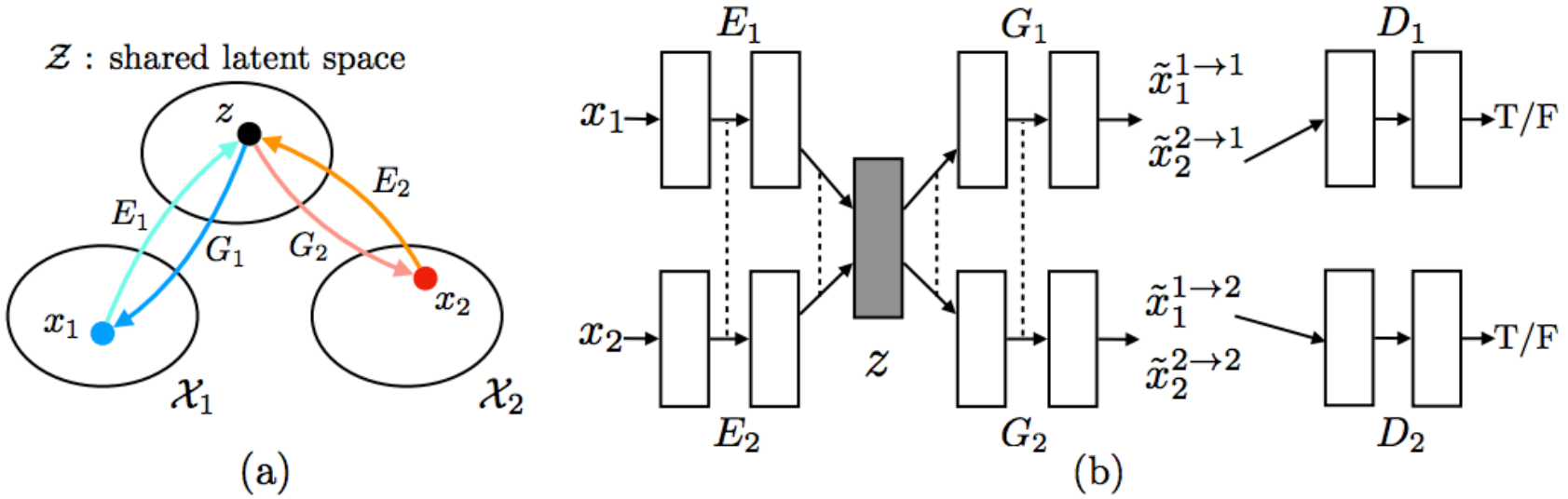
Not perfect





Zhu *et al*., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss" in ICCV, 2017.

UNIT



Liu *et al.*, "Unpaired Image-to-Image Translation" in NIPS, 2017.

UNIT

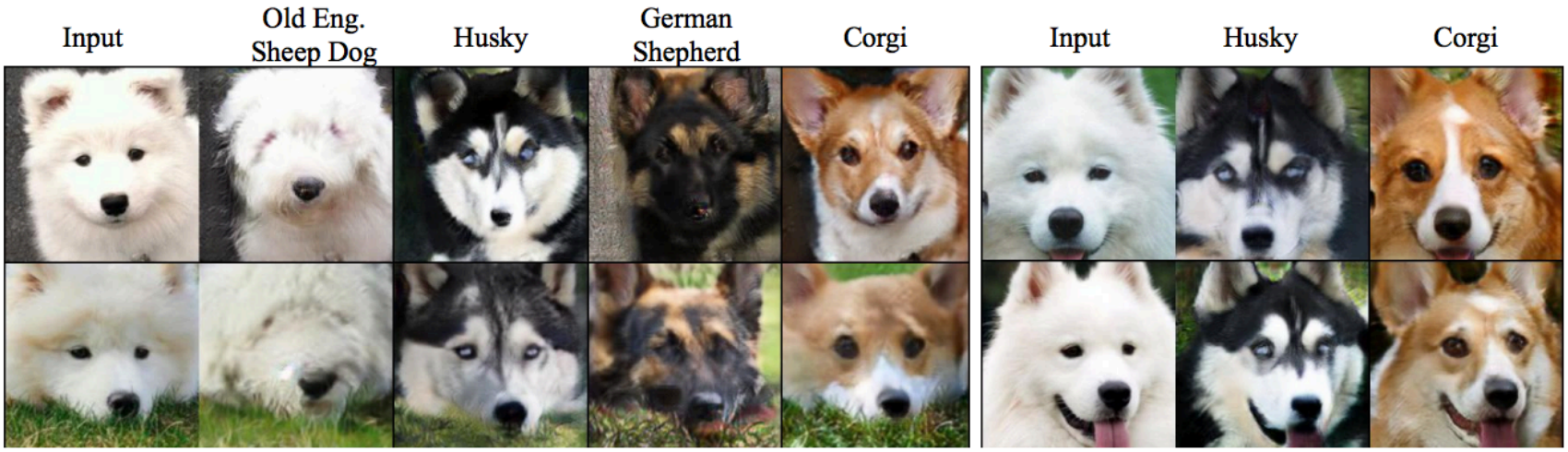


Figure 4: Dog breed translation results.

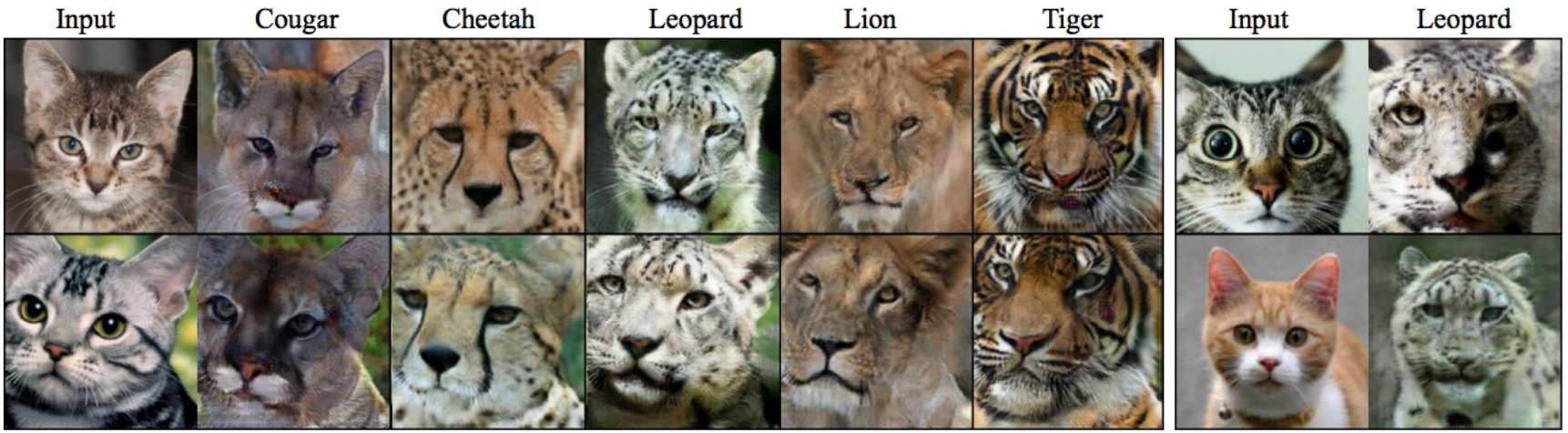
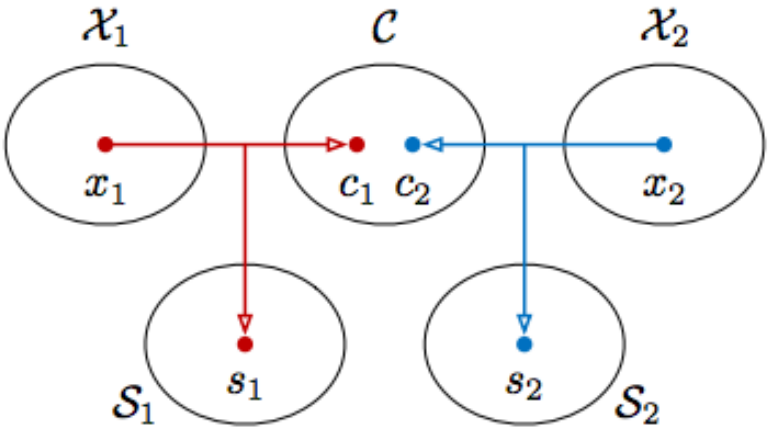


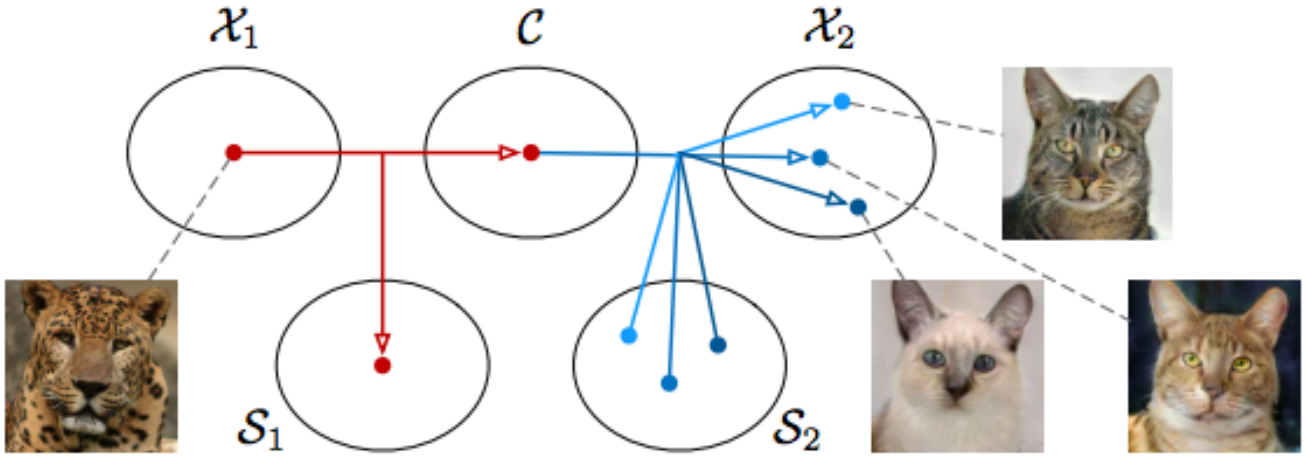
Figure 5: Cat species translation results.

Liu *et al.*, "Unpaired Image-to-Image Translation" in NIPS, 2017.

MUNIT



(a) Auto-encoding

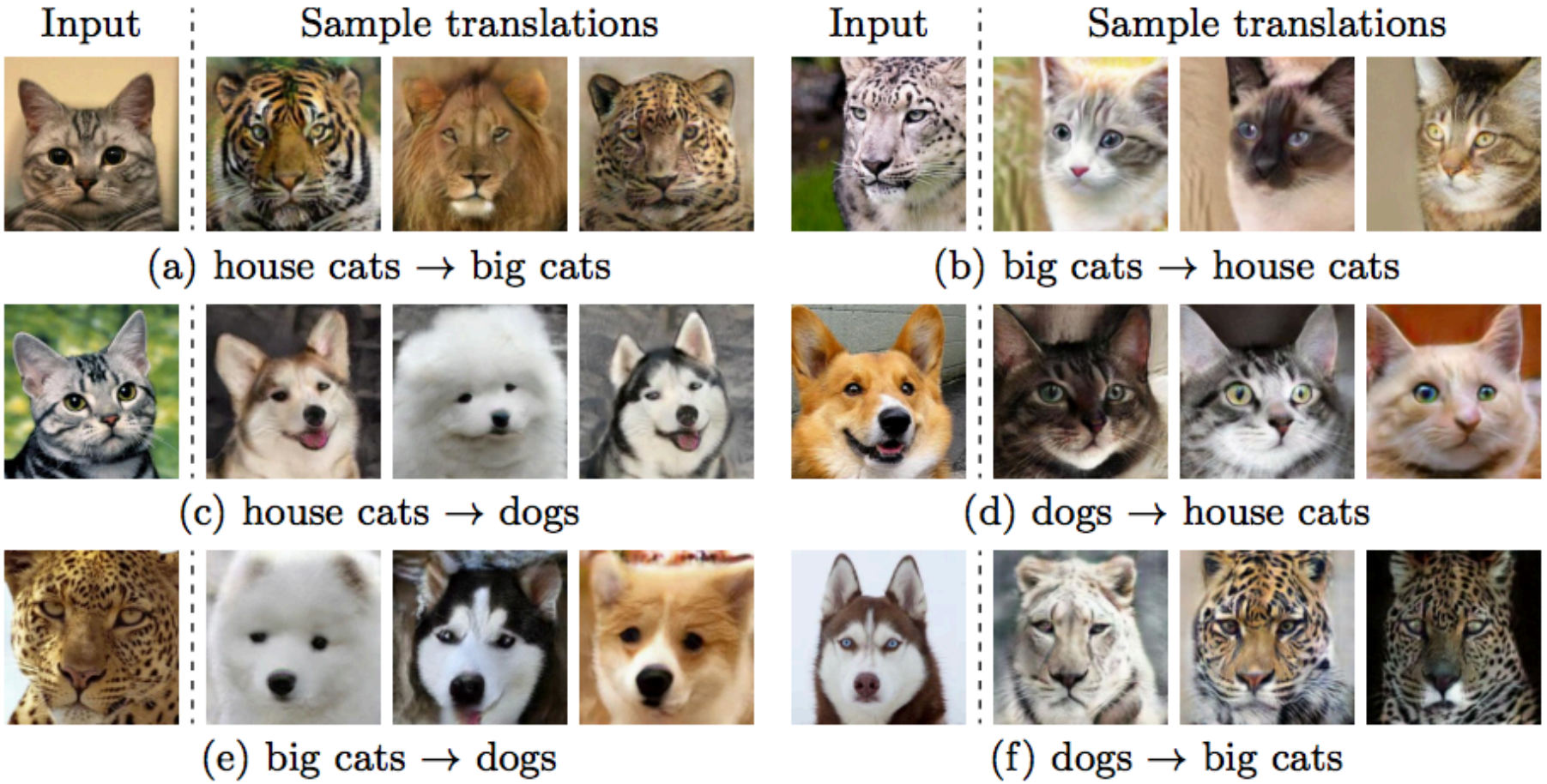


(b) Translation

Huang *et al.*, "Multimodal unpaired Image-to-Image Translation" in arXiv, 2018.

MUNIT

Video demo: https://youtu.be/ab64TWzWn40



Huang *et al.,* "Multimodal unpaired Image-to-Image Translation" in arXiv, 2018.

MUNIT

Fig. 9. Example-guided image translation. Each row has the same content while each column has the same style.

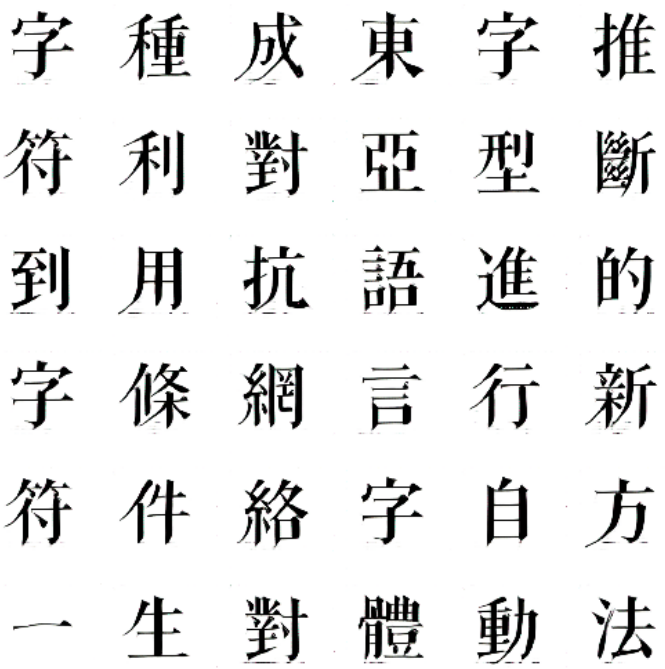
Huang *et al.*, "Multimodal unpaired Image-to-Image Translation" in arXiv, 2018.

Tons of other applications

Tons of applications

- Face Synthesis
- Anime Illustration Generation
- Colorize Pictures
- Photo/Video Editing
- Edge to Drawing
- Text to Image
- Image Inpainting
- Image Restoration
- Super Resolution
- Image Deraining/Desnowing
- 2D to 3D
- and so on …

Style Transfer



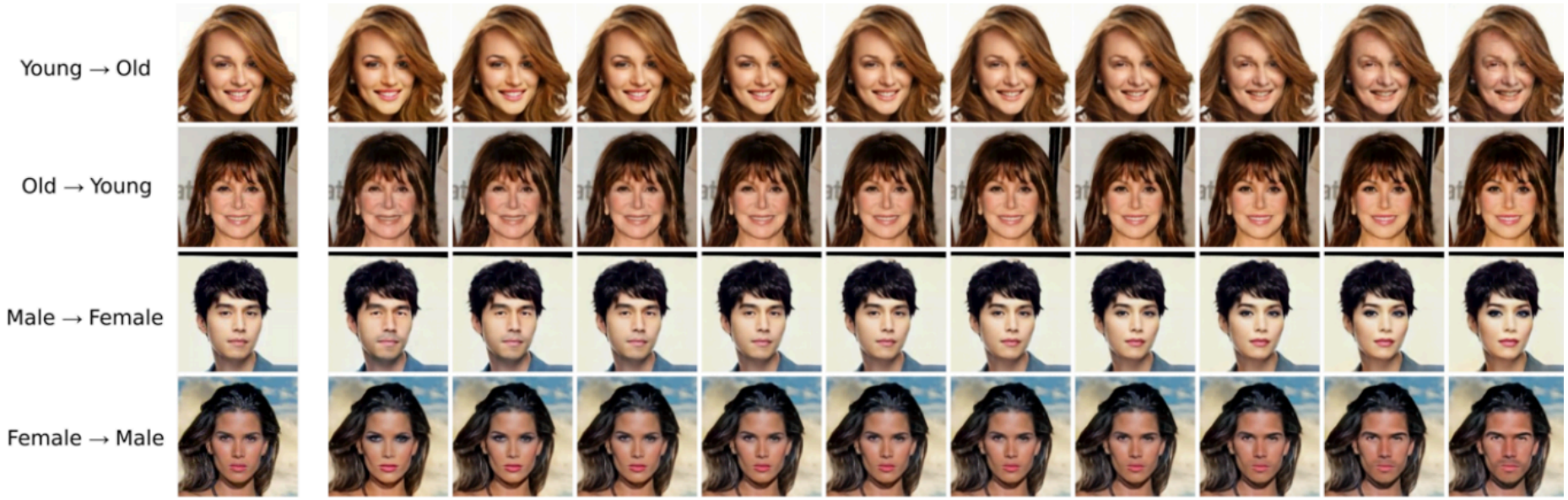
Images from https://github.com/kaonashi-tyc/zi2zi

Chinese Poem Generation

- 曾中书惟静，汝道卖晚寒。池好可河夫，高怀礼呼山。
- 风下带红云，目烟坐井缘，头老起香寥，谢斜影日生。
- 一机不识飞，各有何处人。试知人无处，不复行相人。
- 清暑浮云晓，天水征群近。流流只是入，年事比玩庭。

Zhang *et al.*, “Flexible and Creative Chinese Poetry Generation Using Neural Memory” in arXiv preprint 2017.

Face aging



Lample *et al.*, "Fader Networks: Manipulating Images by Sliding Attributes" in arXiv preprint 2017.

Cross-generation Face Recognition

李嫣然, “深入浅出 GAN 原理篇”, 2017

Finding Abducted Children

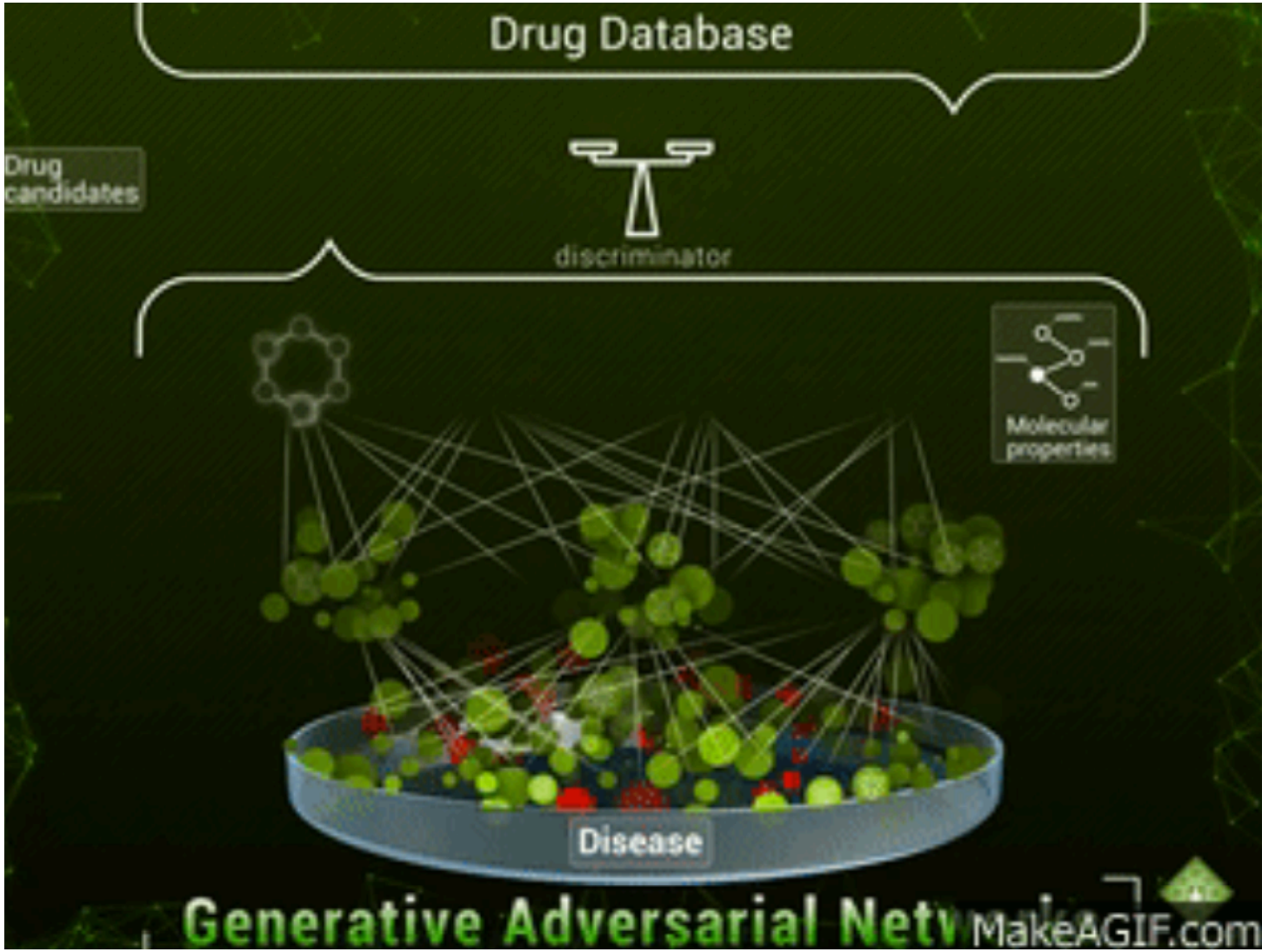
李嫣然, “深入浅出 GAN 原理篇”, 2017

Image Restoration & Face Recognition (maybe GANs are not used)

【意不意外？ 逃犯听张学友的演唱会 被人像识别认出落网】7日晚，张学友南昌演唱会开演后没多久，民警就在演出中心的看台上将一名男子带离现场。据警方介绍，男子姓敖，31岁，江西人，被列为网上追逃。前往现场看演唱会时被智慧安保人像识别功能锁定，让民警在茫茫人海中找到他。 看看新闻Kn...

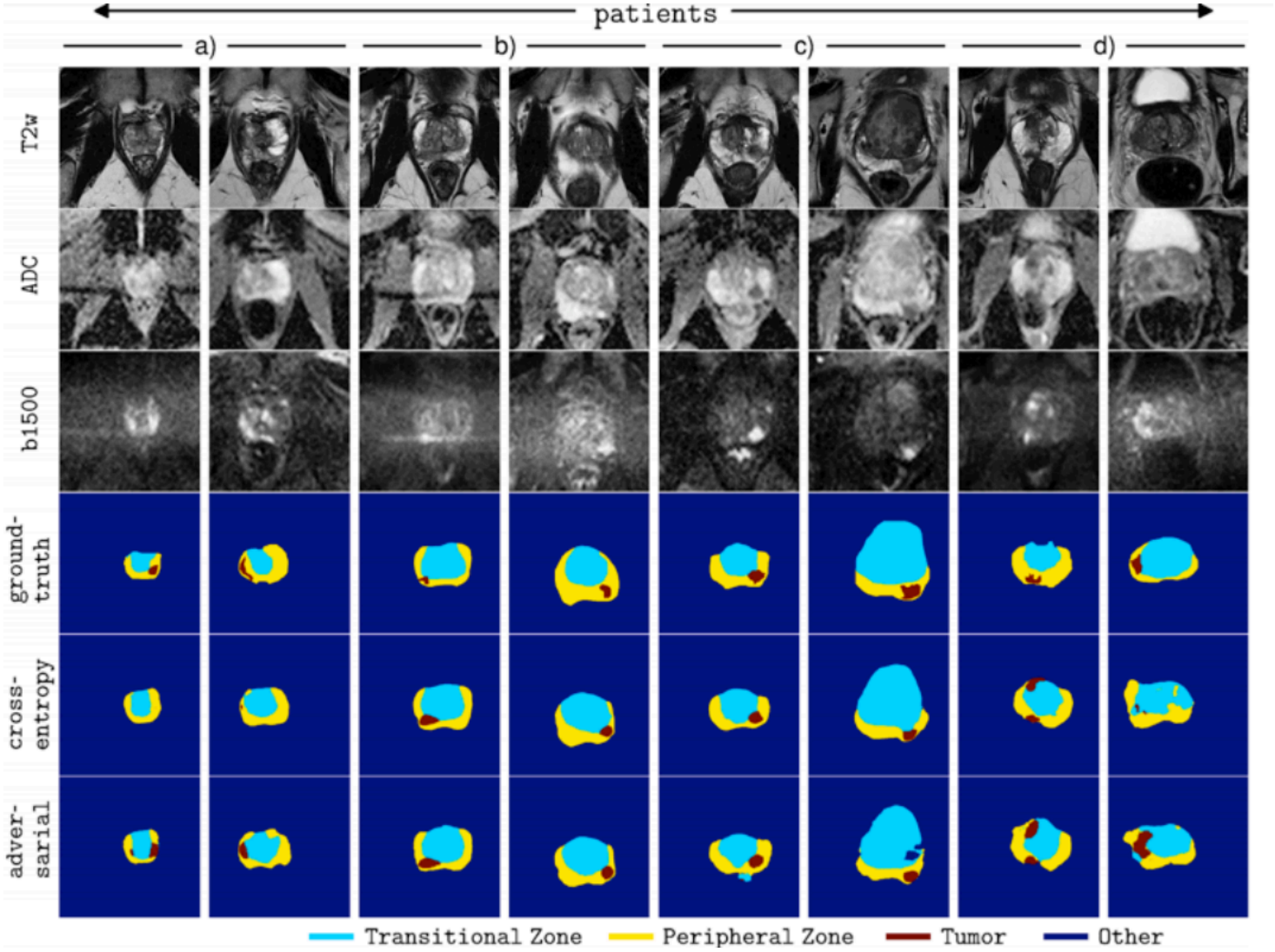
微博@环球时报, 2018.

Drug Discovery



Kadurin *et al*., "The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology". Oncotarget. 2017.

Automatic Cancer Detection



Simon Kohl *et al.,* "Adversarial Networks for the Detection of Aggressive Prostate Cancer". arXiv preprint 2017.

Take away message

GAN have a lot of potential

- Flexible to support a variety of problems
- Potential to represent meaningful metrics

GAN optimization, architecture, etc. is still an active area of research

TensorFlow Tutorial

What is TensorFlow?

"Open source software library for numerical computation using data flow graphs"

Why TensorFlow?

Denny Britz @dennybritz · 25 Dec 2017

I'm going through my newsletters to write up a year-end summary of developments and achievements in AI.

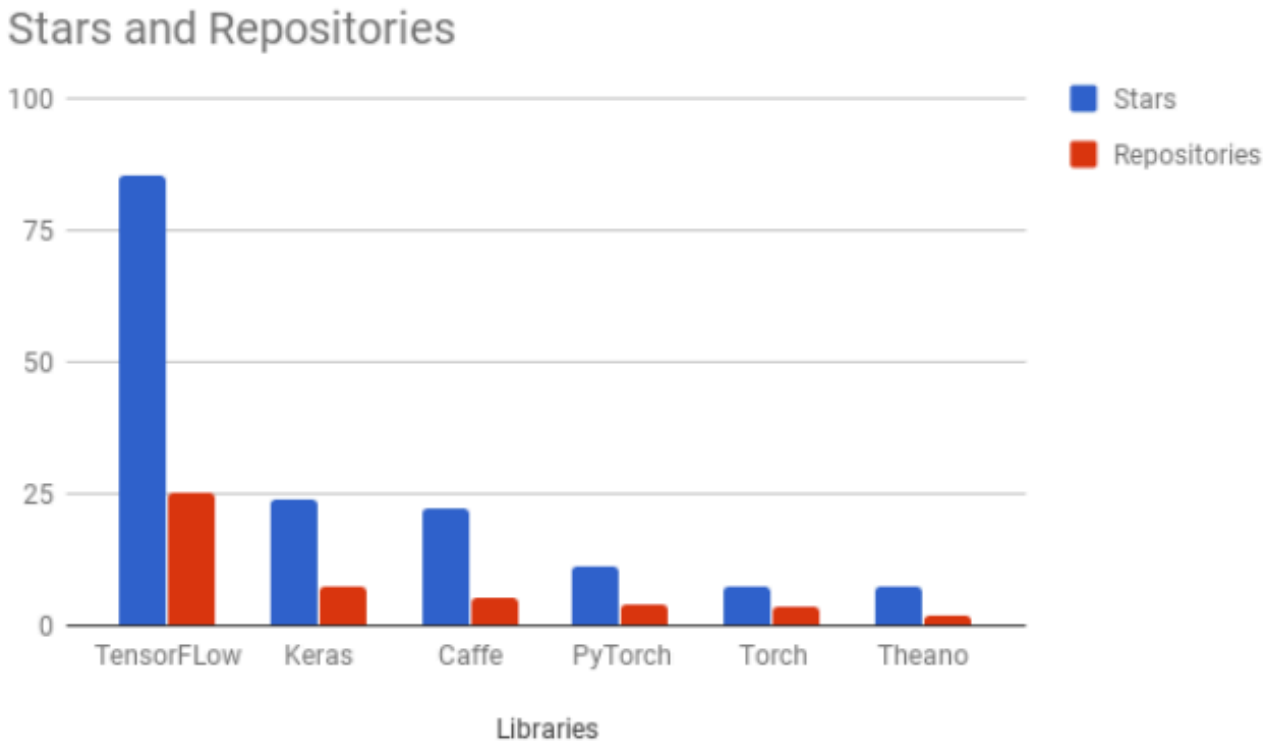
Fun fact: Almost every week, a company released a new generic or task-specific Deep Learning "framework" 😅

- TensorFlow
- Keras
- PyTorch
- MXNet
- Caffe
- Torch
- CNTK
- PaddlePaddle
- ……

Why TensorFlow?

Flexibility + Scalability

Popularity



Companies using TensorFlow

Getting started

```
import tensorflow as tf
```

Graphs and Sessions

Data Flow Graphs

TensorFlow separates definition of computations from their execution

Phase 1: assemble a graph

Phase 2: use a session to execute operations in the graph.

Graphs and Sessions

Data Flow Graphs

TensorFlow separates definition of computations from their execution

Phase 1: assemble a graph

Phase 2: use a session to execute operations in the graph.

Changed in eager mode!!!

What's a tensor?

An n-dimensional array

- 0-d tensor: scalar (number)
- 1-d tensor: vector
- 2-d tensor: matrix
- and so on

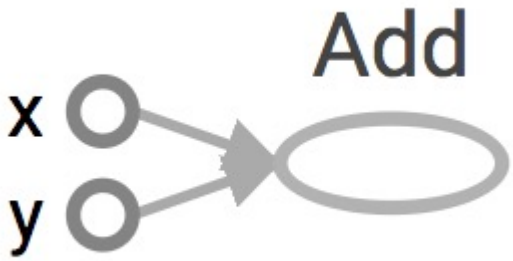
Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

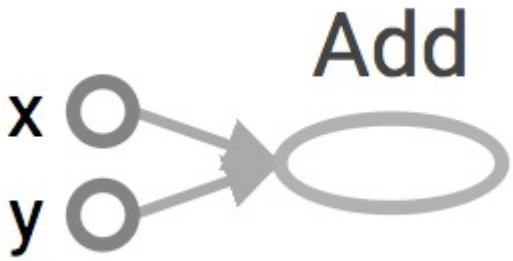
Visualize with tensorboard



Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

Visualize with tensorboard



Why x, y?

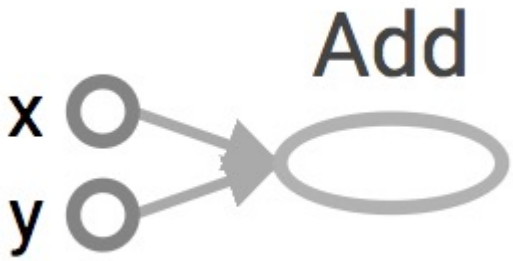
TF automatically names the nodes when you don't explicitly name them.

```
x = 3
y = 5
```

Data Flow Graphs

```
import tensorflow as tf
a = tf.add(3, 5)
```

Visualize with tensorboard



Nodes: operators, variables, and constants

Edges: tensors

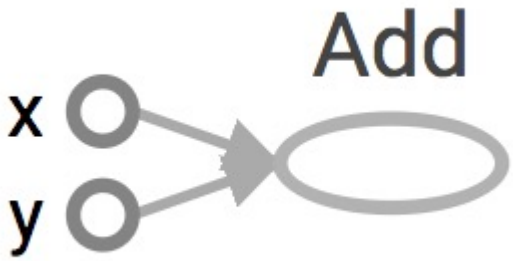
Data Flow Graphs

```
import tensorflow as tf

a = tf.add(3, 5)

print(a)
```

Visualize with tensorboard




```
>> Tensor("Add:0", shape=(), dtype=int32)
(Not 8)
```

How to get the value of *a*?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of *a*

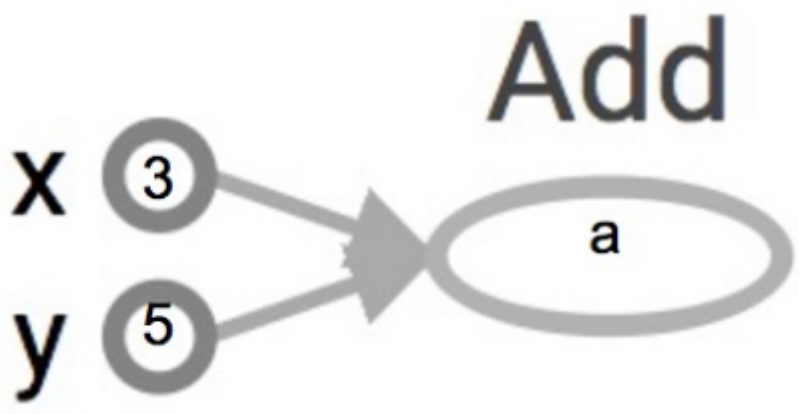
How to get the value of *a*?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of *a*

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print(sess.run(a))
sess.close()
```

```
>> 8
```

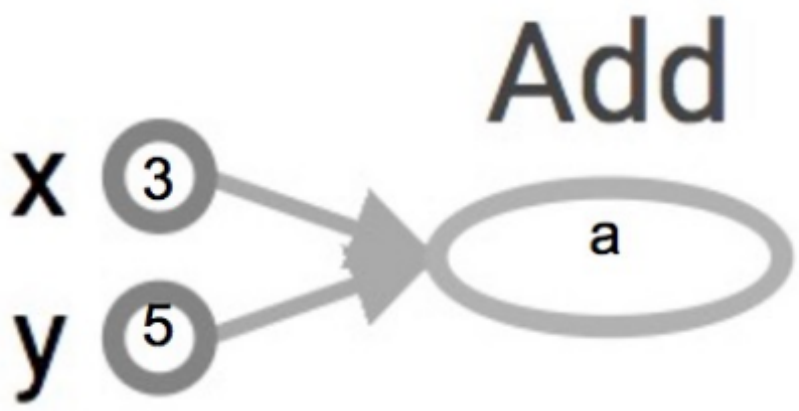


How to get the value of *a*?

Create a **session**, assign it to variable sess so we can call it later

Within the session, evaluate the graph to fetch the value of *a*

```
import tensorflow as tf
a = tf.add(3, 5)
with tf.Session() as sess:
        print(sess.run(a))
```




```
>> 8
```

tf.Session()

A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

Session will also allocate memory to store the current values of variables.

Thank you!