Generative Adversarial Networks

Presented by Zhanyu Wang

Department of Statistics
Purdue University

February 21, 2019

PURDUE UNIVERSITY

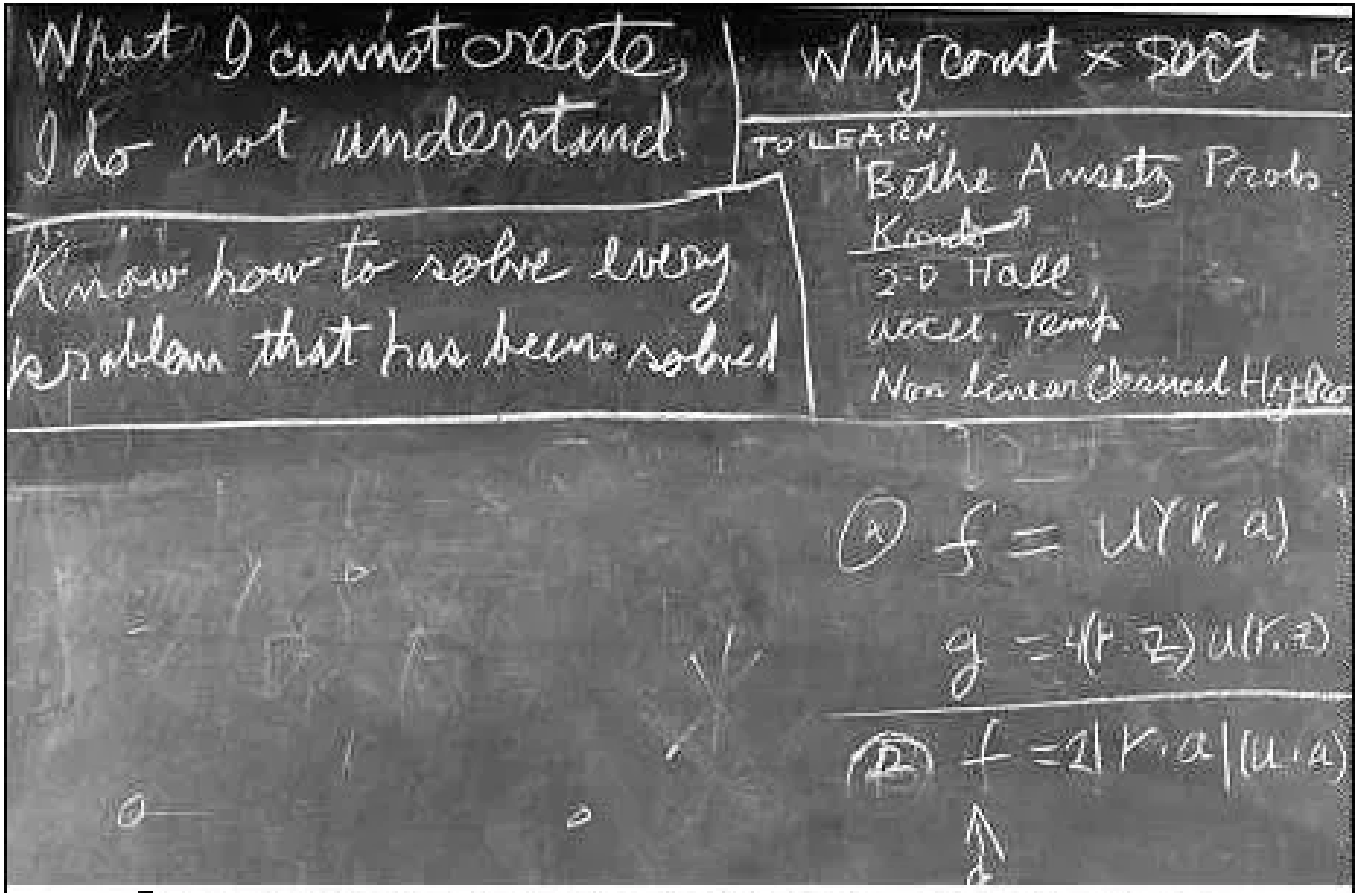
Outline

- Introduction
- Method
- GAN Evolution
- GAN Zoo and Applications
- Takeaways and more

What's the Goal?

Thoughtful quote from Richard Feynman

- What I cannot create, I do not understand.



What's the Goal? (Generative Adversarial Networks)

- Generate meaningful things (Image, Video, Text, Audio, etc.)
- Train a function that generate samples from a distribution:
  - The distribution of human faces:
    Progressive GAN, 2017 [11] StyleGAN, 2018 [12]
  - The distribution of facial expression videos:
    MoCoGAN, 2018 [25]
  - The distribution of 3d objects:
    3D-GAN, 2016 [26]
  - The distribution of music:
    MidiNet, 2018 [27] MuseGAN, 2018 [8]
  - The distribution of drugs:
    ATNC, 2018 [19]
  - The distribution of movie reviews:
    MaskGAN, 2018 [9]

Why study generative modeling?

- Represent high-dimensional structures
- Simulate virtual samples for tasks (e.g. self-driving cars [29])
- Create arts
- Work with multi-modal outputs [31]
- Work with missing data or semi-supervised learning

What's the Method? (Generative Adversarial Networks)

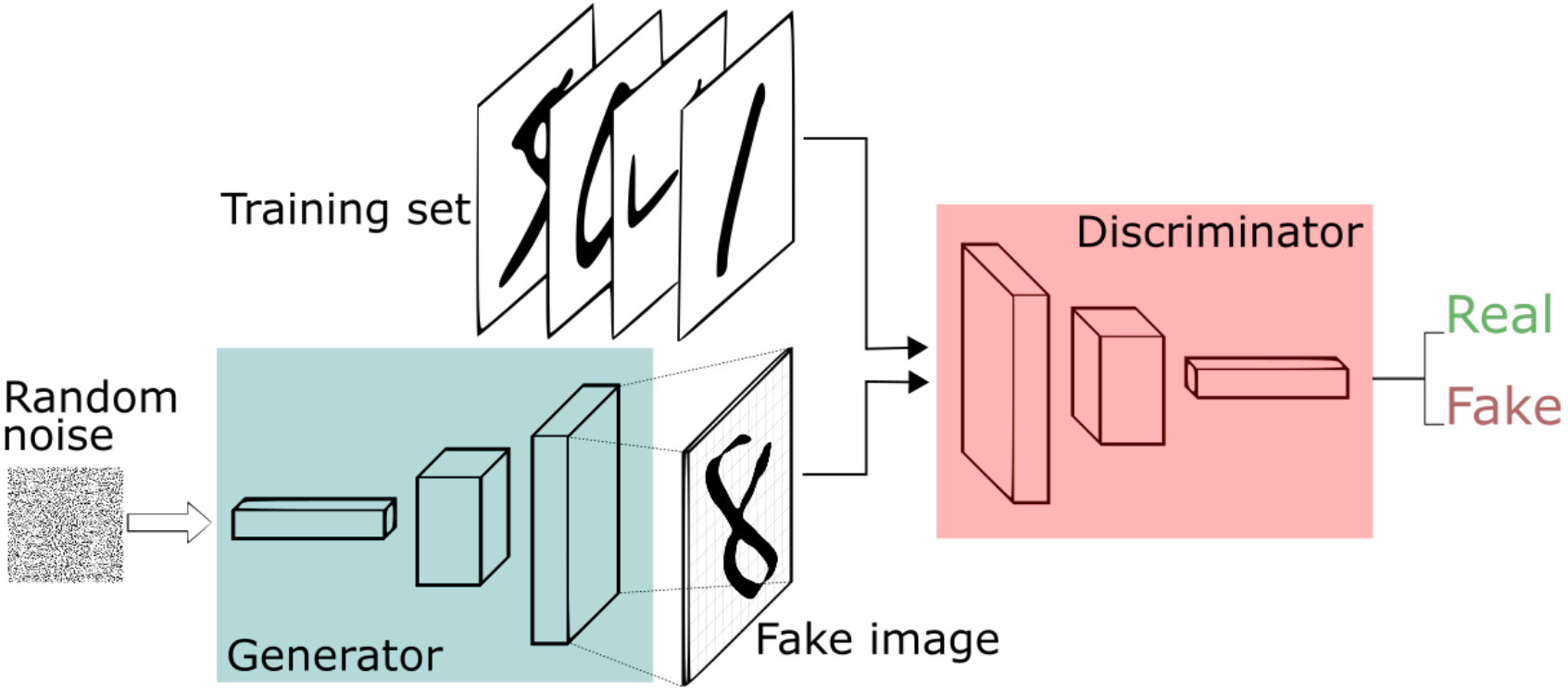


Image credit: Thalles Silva

Another thoughtful quote from Richard Feynman:

- We are trying to prove ourselves wrong as quickly as possible, because only in that way can we find progress.

What's the Method? (Generative Adversarial Networks)

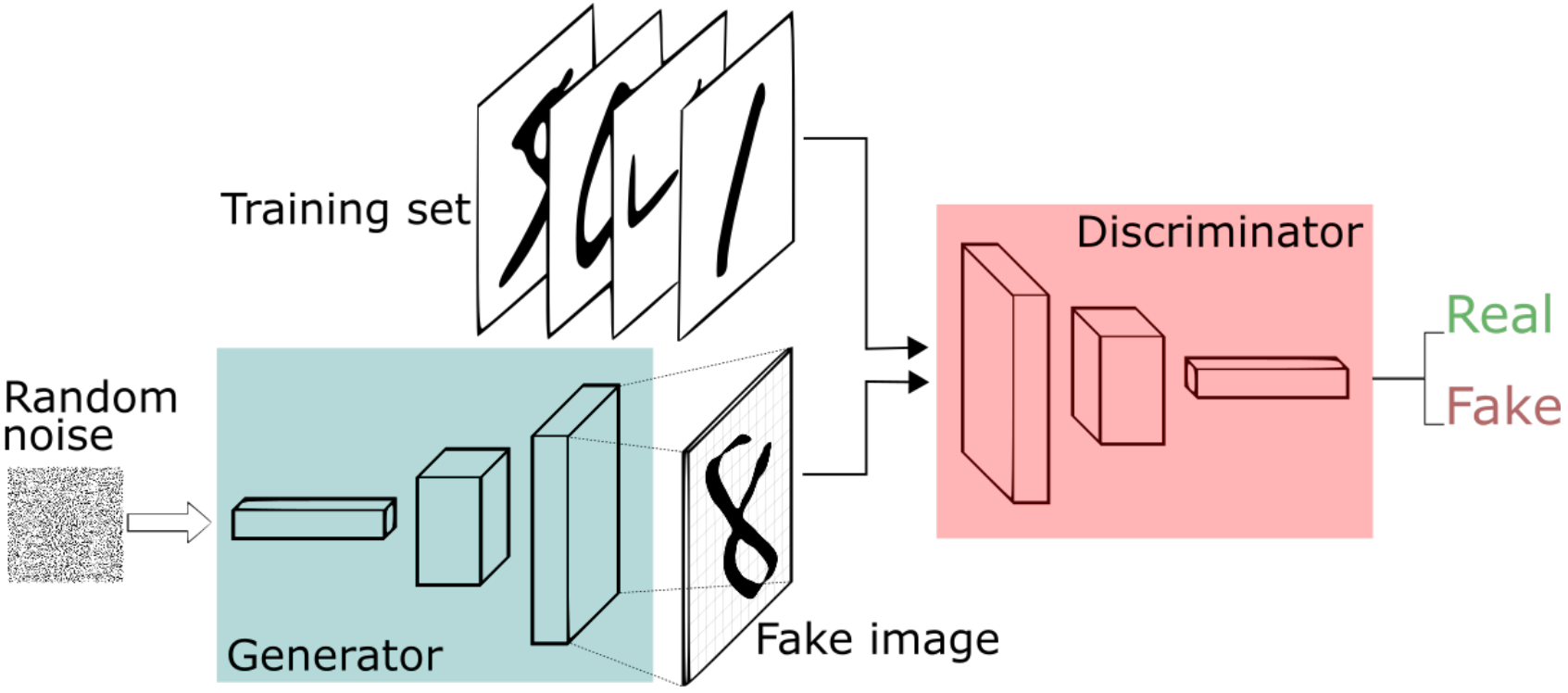
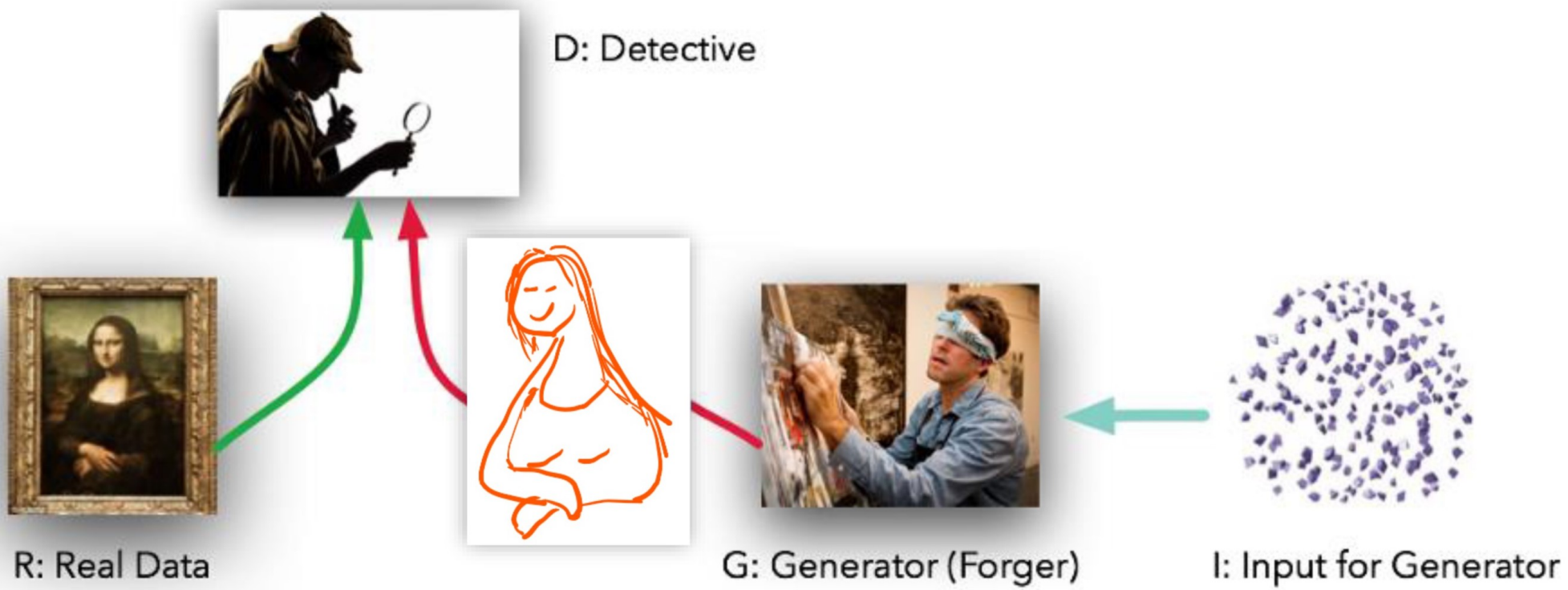


Image credit: Thalles Silva

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{y \sim p_{\text{fake}}(y)}[\log(1 - D(y))]$$

$$= \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_Z(z)}[\log(1 - D(G(z)))]$$

What's the Method? (Generative Adversarial Networks)



$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{y\sim p_{\text{fake}}(y)}[\log(1-D(y))]$$

$$= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_Z(z)}[\log(1-D(G(z)))]$$

What's the Method? (Generative Adversarial Networks)

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

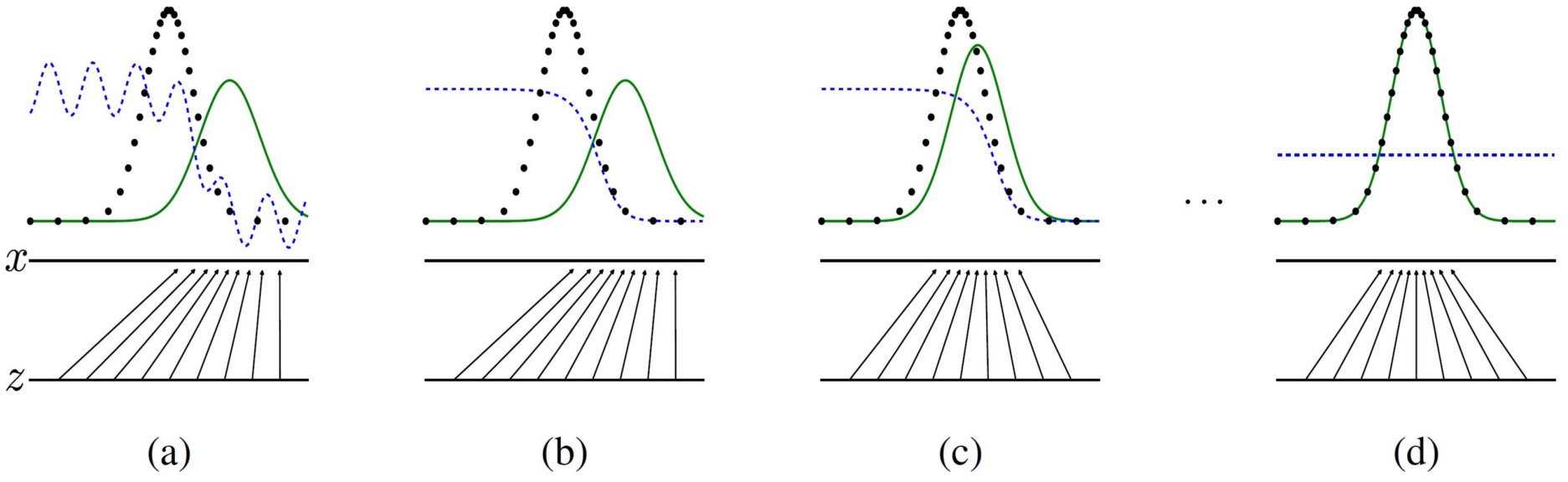
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

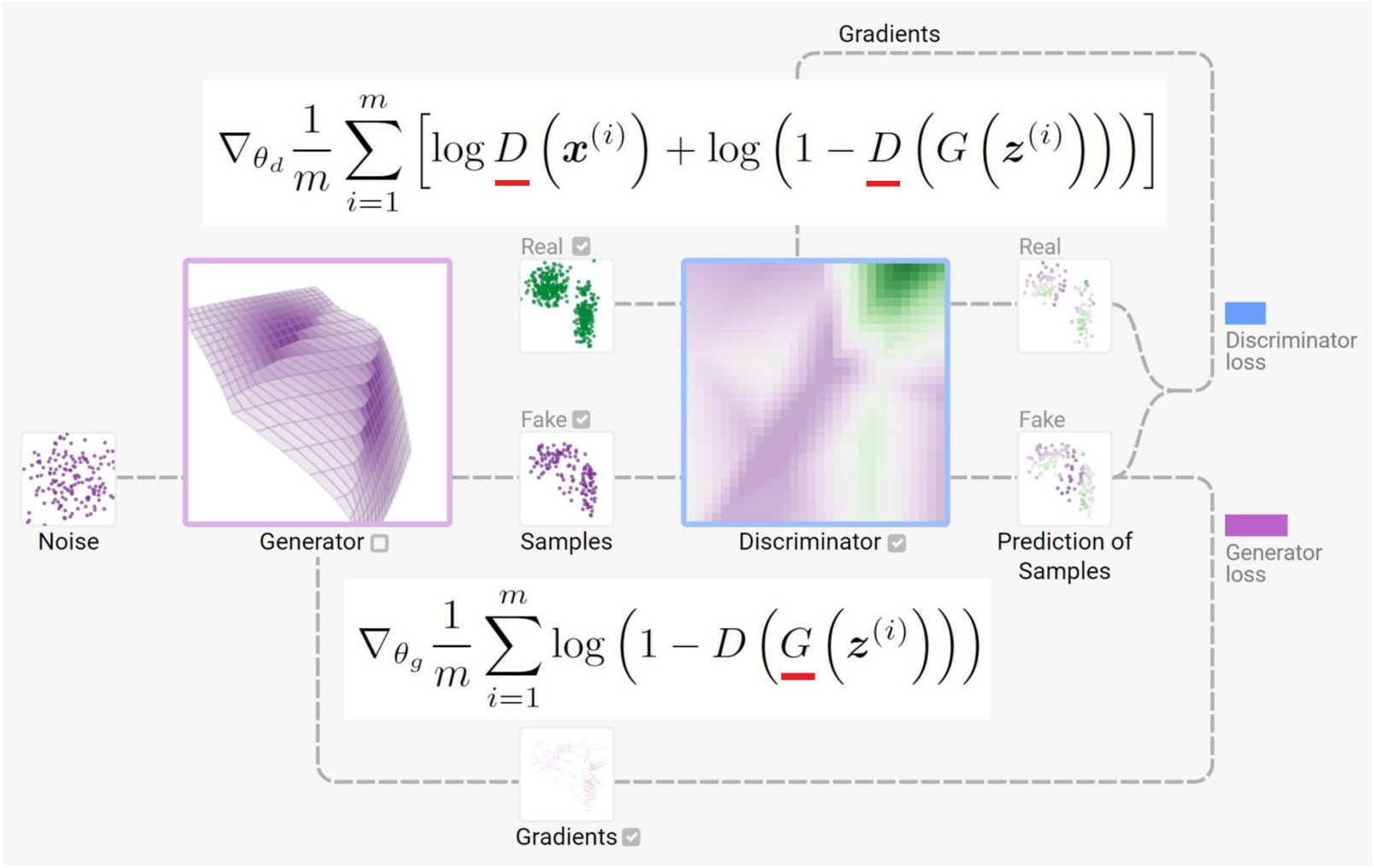
$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_Z(z)}[\log(1 - D(G(z)))]$$

What's the Method? (Generative Adversarial Networks)

- What's going on in training? GAN training visualization



What's the Method? (Generative Adversarial Networks)



Play with GANs in your browser!

Why it works?

$$L_G(D) = \int_x p_{\text{data}}(x) \log(D(x)) dx + \int_x p_{\text{fake(G)}}(x) \log(1 - D(x)) dx$$

$$\frac{\partial L}{\partial D(x)} = \frac{p_{\text{data}}(x)}{D(x)} - \frac{p_{\text{fake(G)}}(x)}{1 - D(x)} = 0$$

$$\boxed{D(x) = \arg\max_D L(G, D) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{fake(G)}}(x)}}$$

$$L_D(G) = \int_x p_{\text{data}}(x) \log(D(x)) dx + \int_x p_{\text{fake(G)}}(x) \log(1 - D(x)) dx$$

$$\begin{aligned} C(G) &= \max_D L_D(G) \\ &= \int p_{\text{data}} \log\left(\frac{p_{\text{data}}}{p_{\text{data}} + p_{\text{fake(G)}}}\right) dx + \int p_{\text{fake(G)}} \log\left(\frac{p_{\text{fake(G)}}}{p_{\text{data}} + p_{\text{fake(G)}}}\right) dx \\ &= -\log(4) + KL\left(p_{\text{data}} \| \frac{p_{\text{data}} + p_{\text{fake(G)}}}{2}\right) + KL\left(p_{\text{fake}} \| \frac{p_{\text{data}} + p_{\text{fake(G)}}}{2}\right) \end{aligned}$$

$$\boxed{G = \arg\max_G C(G) \Rightarrow p_{\text{fake(G)}} = p_{\text{data}}}$$

Does it work better than others?

What are other generative models?

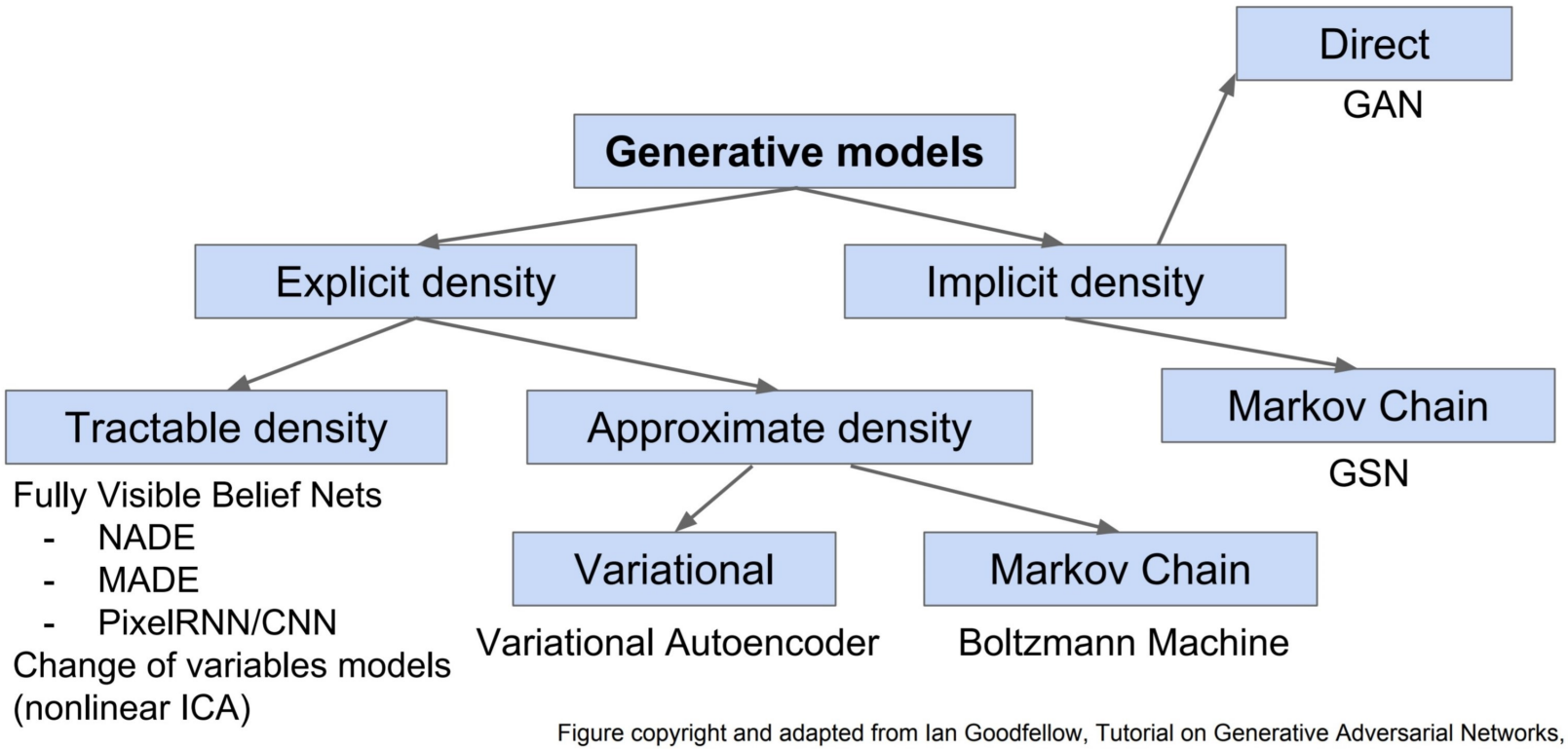


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

- PixelRNN/CNN/CNN++ [17, 22]
- Variational Autoencoder (VAE) [7]

Does it work better than others? (GAN vs. PixelCNN++)

Relation to maximum likelihood:

- PixelCNN++: $P(x) = \prod_{i=1}^{n^2} p(x_i|x_1, \cdots, x_{i-1})$.
- GAN: $D(x) = P(\text{x is real}), 1 - D(x) = P(\text{x is fake})$.

$$\sum_{i=1}^{n} [\log D(x_i) + \log(1 - D(G(z_i)))] = \log P(\forall i, x_i \text{ is real}, G(z_i) \text{ is fake})$$

GAN maximize likelihood with implicit density

- Better sample quality
- Training is faster

PixelCNN++ maximize likelihood using conditional densities.

- Explicit and exact data likelihood
- Training is more stable

Does it work better than others? (GAN vs. PixelCNN++)

model	IS	FID-5K	FID	GAN-train	GAN-test	SWD 16	SWD 32
real images	11.33	9.4	2.1	92.8	-	2.8	2.0
SNGAN	8.43	18.8	11.8	82.2	87.3	3.9	24.4
WGAN-GP (10M)	8.21	21.5	14.1	79.5	85.0	3.8	6.2
WGAN-GP (2.5M)	8.29	22.1	15.0	76.1	80.7	3.4	6.9
DCGAN	6.69	42.5	35.6	65.0	58.2	6.5	24.7
PixelCNN++	5.36	121.3	119.5	34.0	47.1	14.9	56.6

Table 1: CIFAR10 experiments. IS: higher is better. FID and SWD: lower is better. SWD values here are multiplied by 10^3 for better readability. GAN-train and GAN-test are accuracies given as percentage (higher is better).

How to measure sample quality? (What is IS?)

Inception Score (IS, proposed by OpenAI, 2016. [21])

- Use Inception model [24] to get the distribution of labels.
- $IS = \exp(\mathbb{E}_{x \sim p_g} KL(p(y|x) \| p(y)))$, y is the labels of x
- Images containing meaningful objects should have a low entropy $p(y|x)$. $p(y)$ is the marginal distribution by summming all x. It should has high entropy.
- Higher IS is better.
- Focus on classifiable object generation and various kinds of samples. Not focus on the variation within a class.

State of the art

- SAGAN (2018) [28] 52.5 (previous work was 36.8)
- BigGAN (2018) [5] 166.3 (Realworld Data (Imagenet) 233)

Codes for calculating IS

Some criticisms (*A Note on the Inception Score* [3])

How to measure sample quality? (What is FID?)

Frechet Inception Distance (FID, 2017. [10])

- Use Inception-v3 model to get the features of the image.
- Calculating Wasserstein-2 distance W_2 between the features of the real samples and the fake samples. Use normal distribution approximation for W_2:
  $d^2((m, C), (m_w, C_w)) = \|m - m_w\|_2^2 + \mathrm{Tr}(C + C_w - 2(CC_w)^{1/2})$
- Lower FID is better.
- Consistent with Human sensation in authenticity and variation.

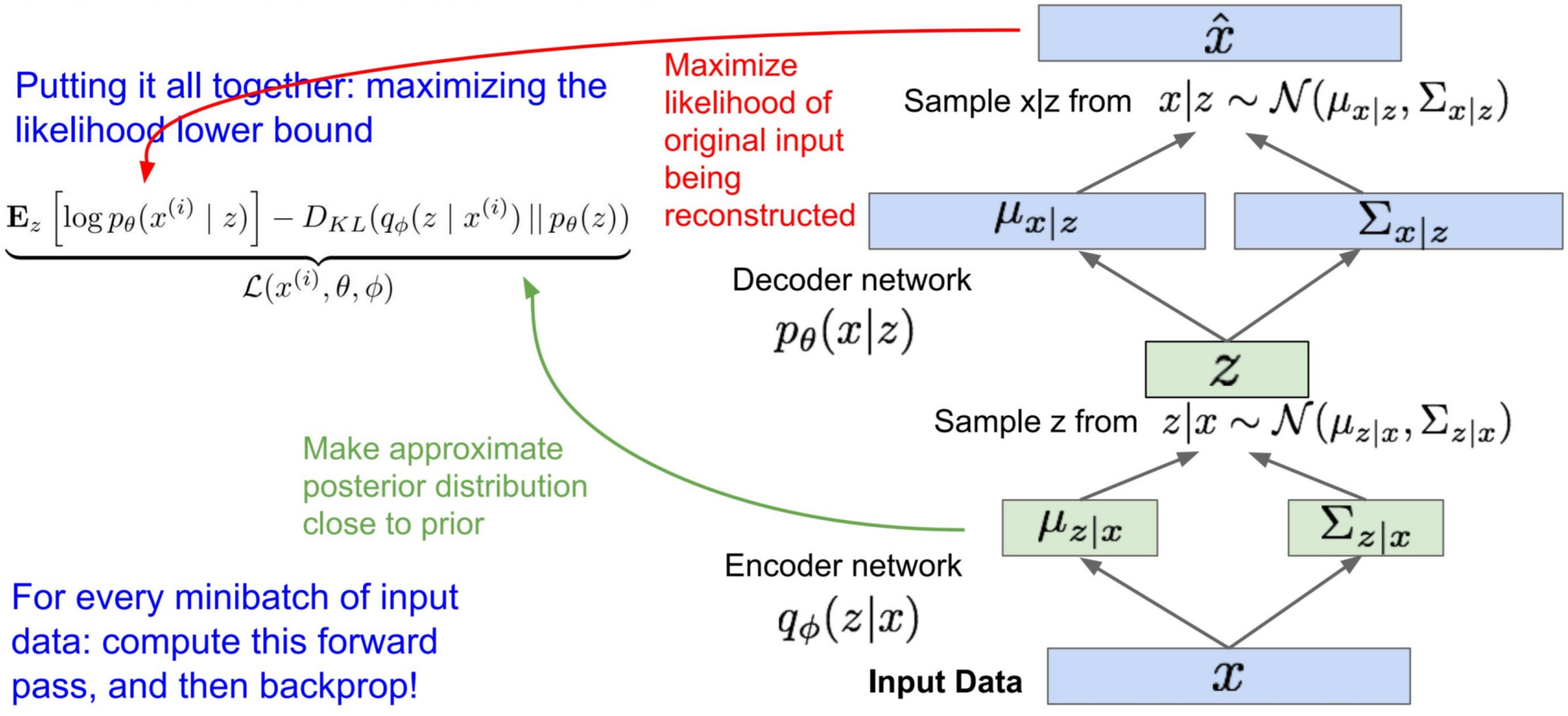
State of the art (50k images)

- SAGAN (2018) [28] 18.65 (previous work was 27.62)
- BigGAN (2018) [5] 9.6

Codes for calculating FID

Does it work better than others? (GAN vs. VAE)

What is Variational Autoencoder?



Does it work better than others? (GAN vs. VAE)

VAE optimizes variational lower bound on likelihood

- Inference of $q(z|x)$: feature representation can be used in other tasks
- Use normal distribution (unimodal distribution) to approximate $p(x|z)$
- Samples blurrier with lower quality

GAN has an adaptive loss function for generating samples

- Samples are good and sharp
- Training is unstable
- GANs as Learned Loss Functions
- GANs as a loss function

Does it work better than others? (GAN vs. VAE)

State of the art of VAE ([6] ICLR 2019 with Rating 9,7,6)

	MNIST	Fashion	CIFAR-10	CelebA	Mean
MM GAN	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2	44.4 ± 2.6
NS GAN	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3	36.7 ± 1.8
LSGAN	7.8 ± 0.6	30.7 ± 2.2	87.1 ± 47.5	53.9 ± 2.8	44.9 ± 13.3
WGAN	$\mathbf{6.7 \pm 0.4}$	21.5 ± 1.6	$\mathbf{55.2} \pm 2.3$	41.3 ± 2.0	31.2 ± 1.6
WGAN GP	20.3 ± 5.0	24.5 ± 2.1	55.8 ± 0.9	30.3 ± 1.0	32.7 ± 2.3
DRAGAN	$7.6 \pm \mathbf{0.4}$	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0	36.9 ± 1.7
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9	36.6 ± 1.1
VAE (cross-entr.)	23.8 ± 0.6	58.7 ± 1.2	155.7 ± 11.6	85.7 ± 3.8	81 ± 4.3
VAE (fixed γ)	51.2 ± 0.8	104.5 ± 1.3	113.0 ± 0.7	119.8 ± 0.9	97.1 ± 0.9
VAE (learned γ)	47.0 ± 0.9	51.5 ± 1.0	$80.1 \pm \mathbf{0.6}$	67.4 ± 2.1	61.5 ± 1.2
2-Stage VAE	13.4 ± 1.3	22.0 ± 0.6	$71.0 \pm \mathbf{0.6}$	45.9 ± 1.4	38.1 ± 1.0
2-Stage VAE*	11.2 ± 0.5	$\mathbf{21.3 \pm 0.4}$	68.0 ± 0.8	$\mathbf{23.8 \pm 0.5}$	$\mathbf{31.1 \pm 0.6}$

Table 1: *FID score comparisons*. For all GAN-based models, the reported values represent the *best* FID obtained across a large-scale hyperparameter search conducted separately for each dataset; default settings are considerably worse (Lucic et al., 2018). Likewise outlier cases (e.g., severe mode collapse) were omitted, which would have otherwise degraded these GAN scores and increased standard deviations still further. In contrast, for the VAE results we used only default training settings across all models and datasets (no tuning), except for the 2-Stage VAE*. Here we simply tested a couple different values for κ and picked the best result for each dataset. Note that specialized architectures and/or random seed optimization can potentially improve the FID score for all models.

How to get this GAN Framework?

How to get this GAN Framework?

Similar Works before GAN:

- Actor-Critic methods from Reinforcement Learning [4, 13, 18]
- Turing Learning: model versus discriminator [14, 15]
- Predictability Minimization model [23]
  - Proposed by Schmidhuber 1992, who also proposed LSTM.
  - GAN as inverse PM?
  - GAN focuses on generation, while PM focuses on encoding.

GAN Evolution

- Vanilla GAN (2014)
- Deep convolutional GAN (DCGAN, 2015)
- Energy-based GAN (EBGAN, 2016)
- Wasserstein GAN (WGAN, 2017)
- Spectral Normalization for GAN (SNGAN, 2018)

Vanilla GAN (2014)

Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units.

Deep convolutional GAN (DCGAN, 2015) [20]

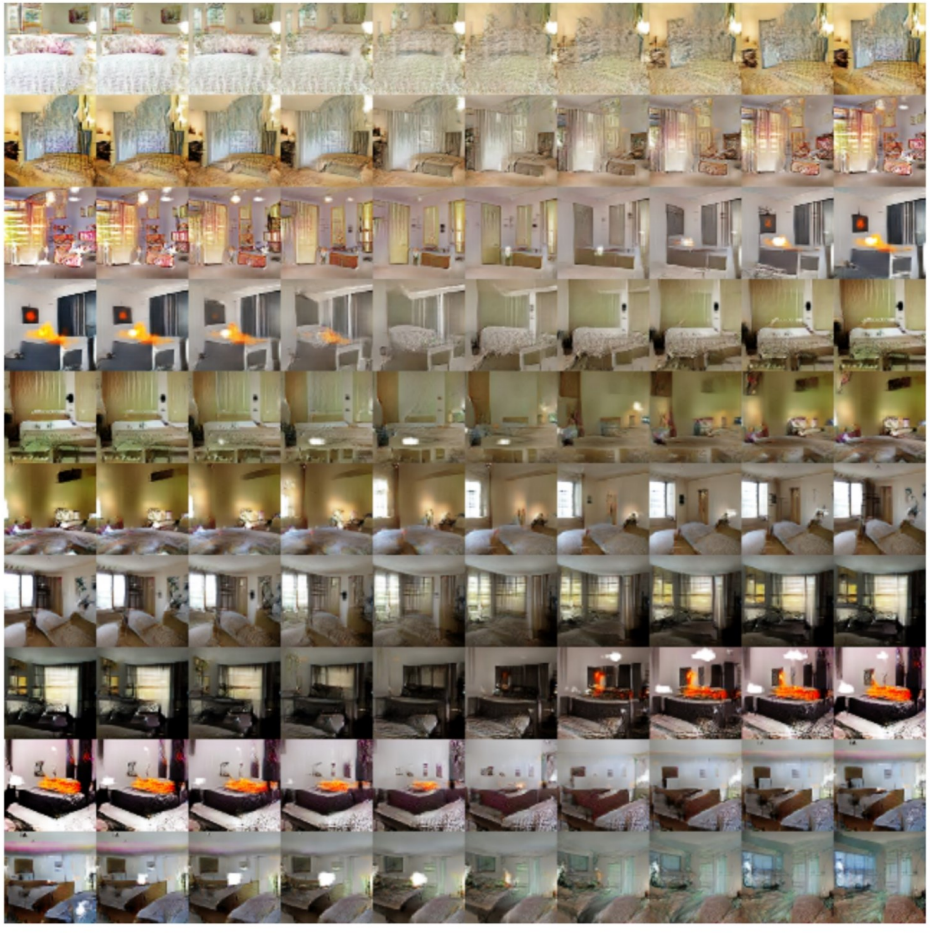
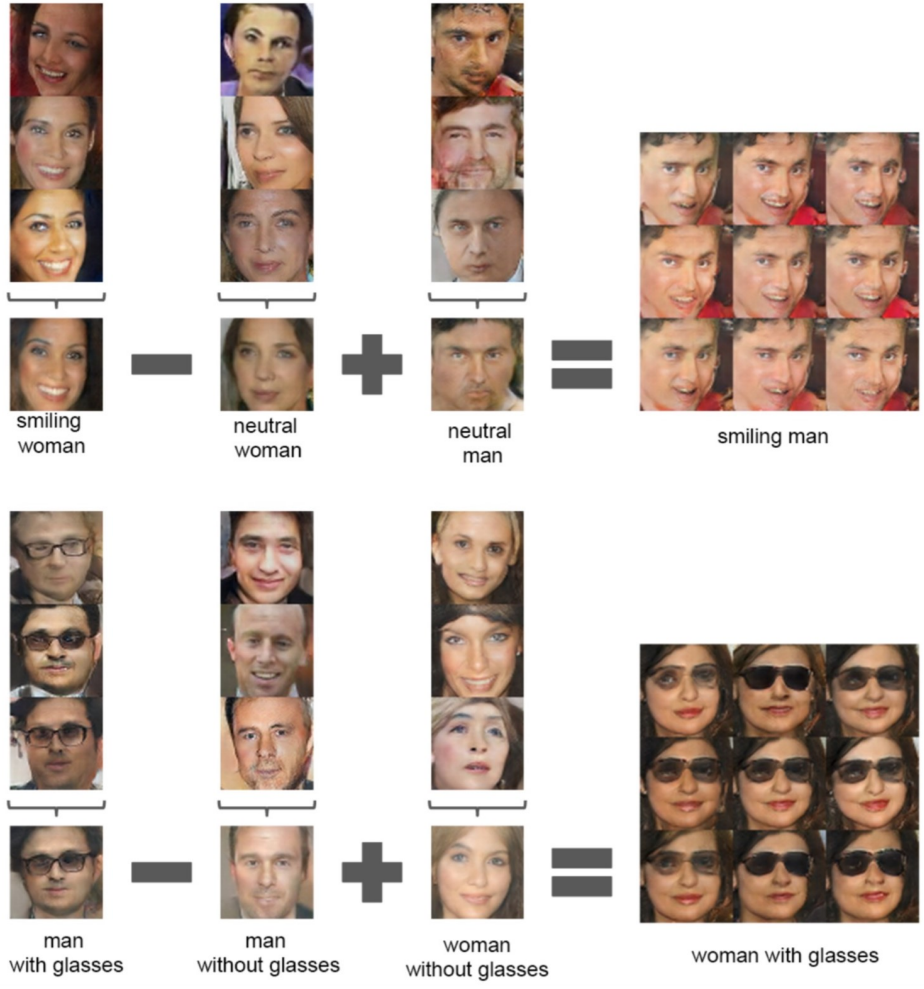


Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.



Deep convolutional GAN (DCGAN, 2015) [20]

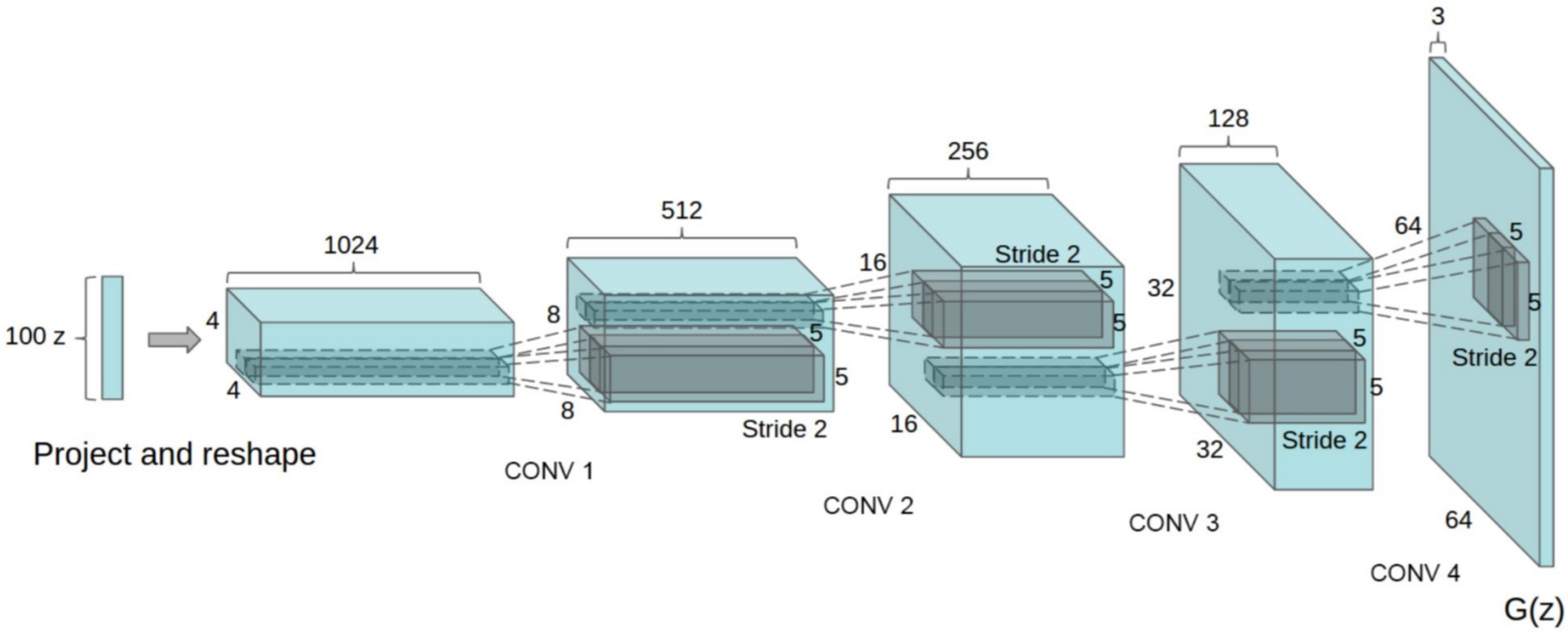


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Transposed convolution animations

DCGAN in PyTorch (official)

Energy-based GAN (EBGAN, 2016) [30]

$$L_D(x,z) = D(x) + [m - D(G(z))]^+, \quad L_G(z) = D(G(z))$$

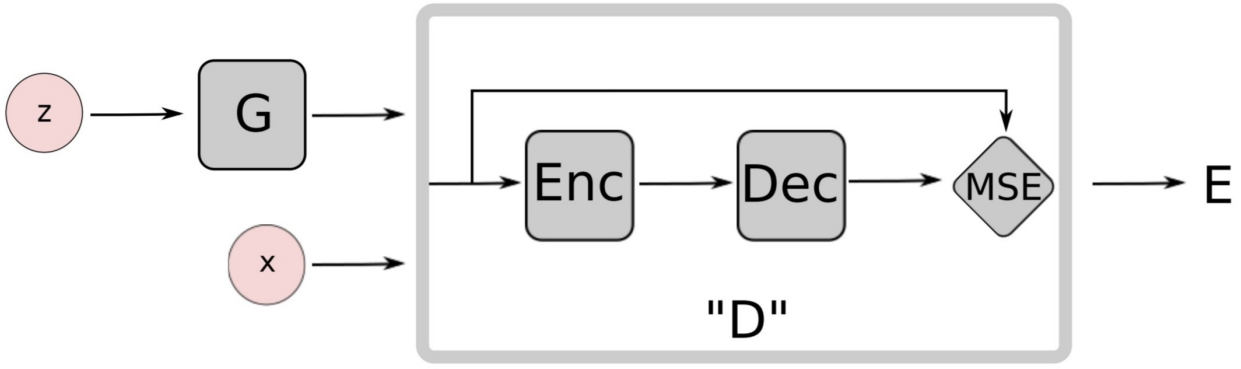
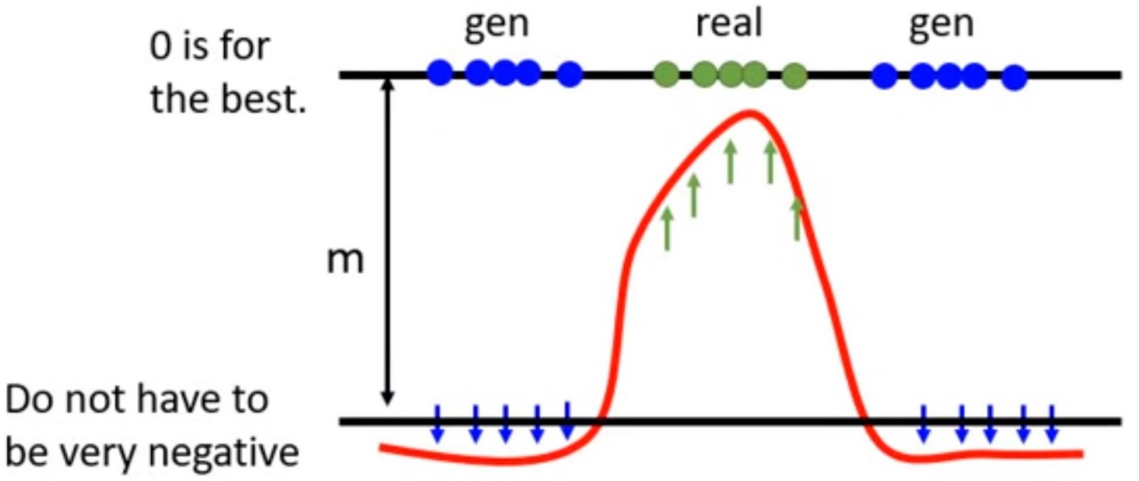


Figure 1: EBGAN architecture with an auto-encoder discriminator.



EBGAN in PyTorch

Energy-based GAN (EBGAN, 2016) [30]

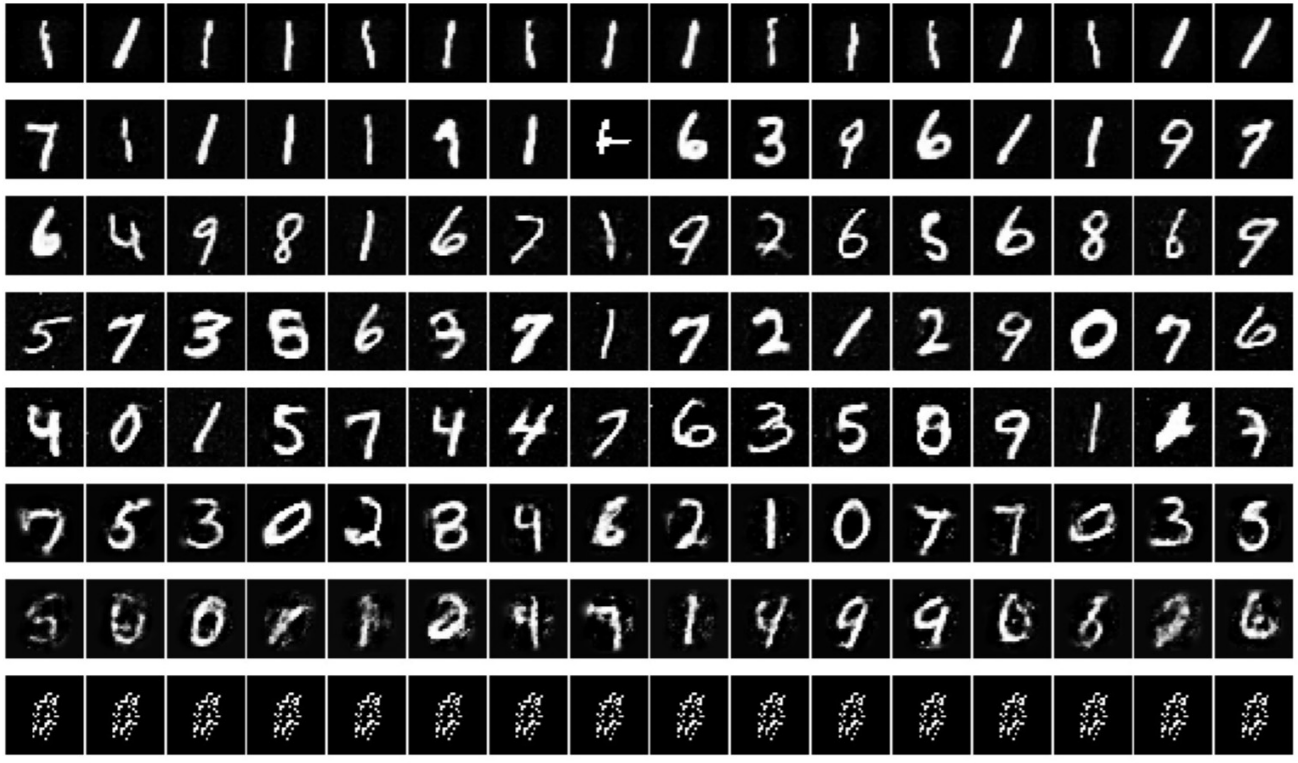


Figure 9: Generation from the EBGAN auto-encoder model trained with different m settings. From top to bottom, m is set to 1, 2, 4, 6, 8, 12, 16, 32 respectively. The rest setting is `nLayerG`=5, `nLayerD`=2, `sizeG`=1600, `sizeD`=1024, `dropoutD`=0, `optimD`=ADAM, `optimG`=ADAM, `lr`=0.001.

Energy-based GAN (EBGAN, 2016) [30]

Figure 6: Generation from the CelebA dataset. Left(a): DCGAN generation. Right(b): EBGAN-PT generation.

Problems in Training GAN: Mode collapse

Figure 1: A DCGAN model is trained with an MLP network with 4 layers, 512 units and ReLU activation function, configured to lack a strong inductive bias for image generation. The results shows a significant degree of mode collapse. [2]

Problems in Training GAN: Vanishing gradient

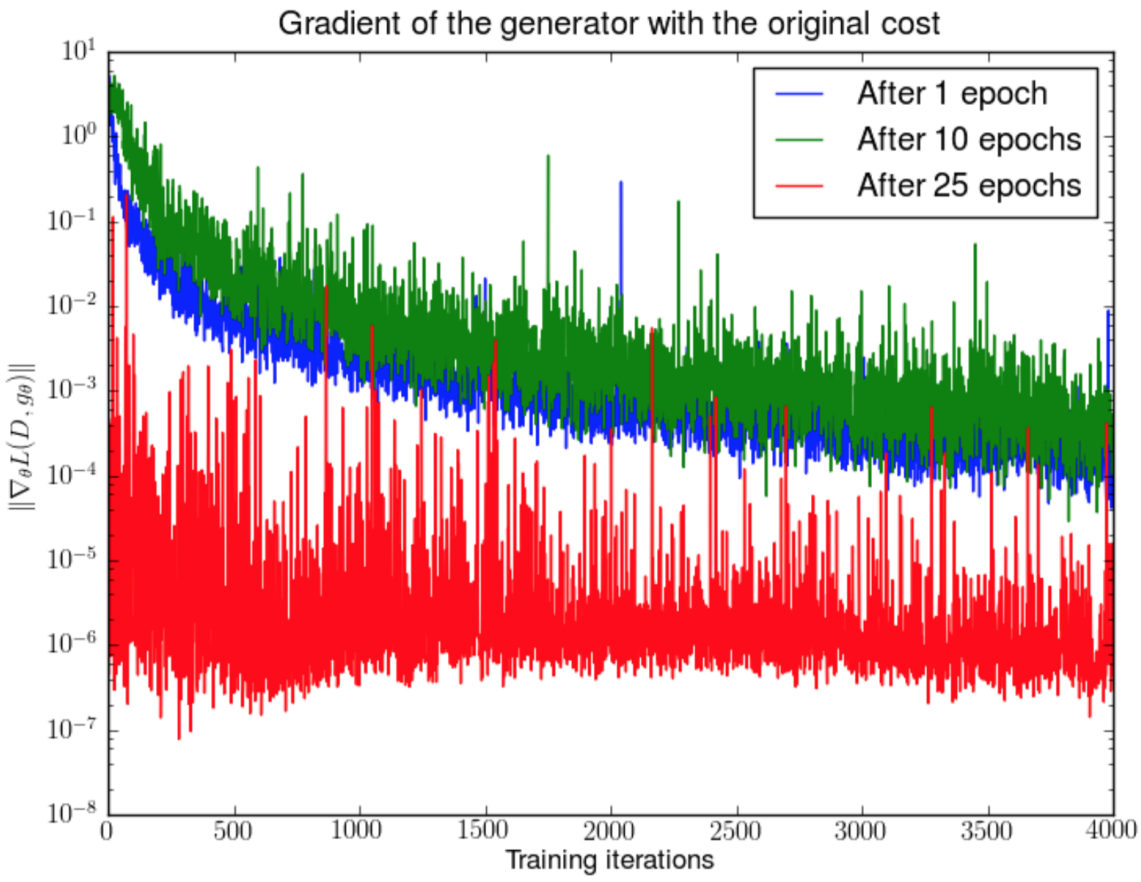


Figure 2: First, a DCGAN is trained for 1, 10 and 25 epochs. Then, with the generator fixed, a discriminator is trained from scratch and measure the gradients with the original cost function. We see the gradient norms decay quickly (in log scale), in the best case 5 orders of magnitude after 4000 discriminator iterations. [1]

Wasserstein GAN (WGAN, 2017) [2]

The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x)$$

The *Jensen-Shannon* (JS) divergence

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

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$.

The *Earth-Mover* (EM) distance or Wasserstein-1

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

$$W(\mathbb{P}_r, \mathbb{P}_g) = \max_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

Example 1 (Learning parallel lines)**.** Let $Z \sim U[0, 1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$

When $\theta_t \to 0$, the sequence $(\mathbb{P}_{\theta_t})_{t\in\mathbb{N}}$ converges to $\mathbb{P}_0$ under the EM distance, but does not converge at all under either the JS, KL, reverse KL, or TV divergences.

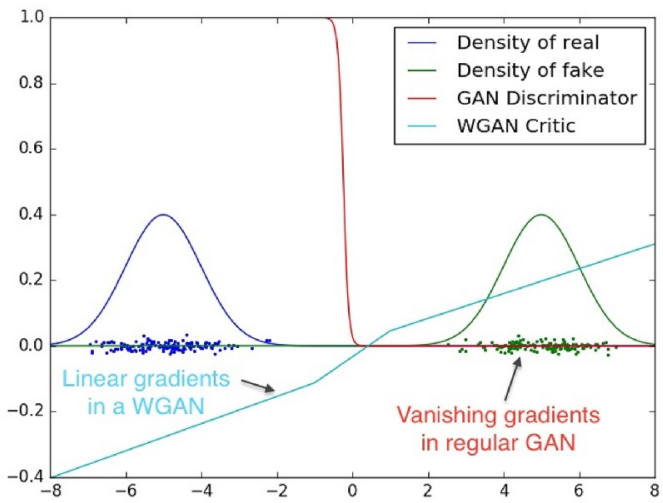
Wasserstein GAN (WGAN, 2017) [2]

1. Remove log in loss functions. Remove sigmoid in D.
2. Weight clipping

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while



Spectral Normalization for GAN (SNGAN, 2018) [16]

$$\sigma(A) := \max_{\boldsymbol{h}:\boldsymbol{h}\neq\mathbf{0}} \frac{\|A\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2\leq 1} \|A\boldsymbol{h}\|_2$$

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^{L}\boldsymbol{h}_{L-1})\|_{\text{Lip}}$$

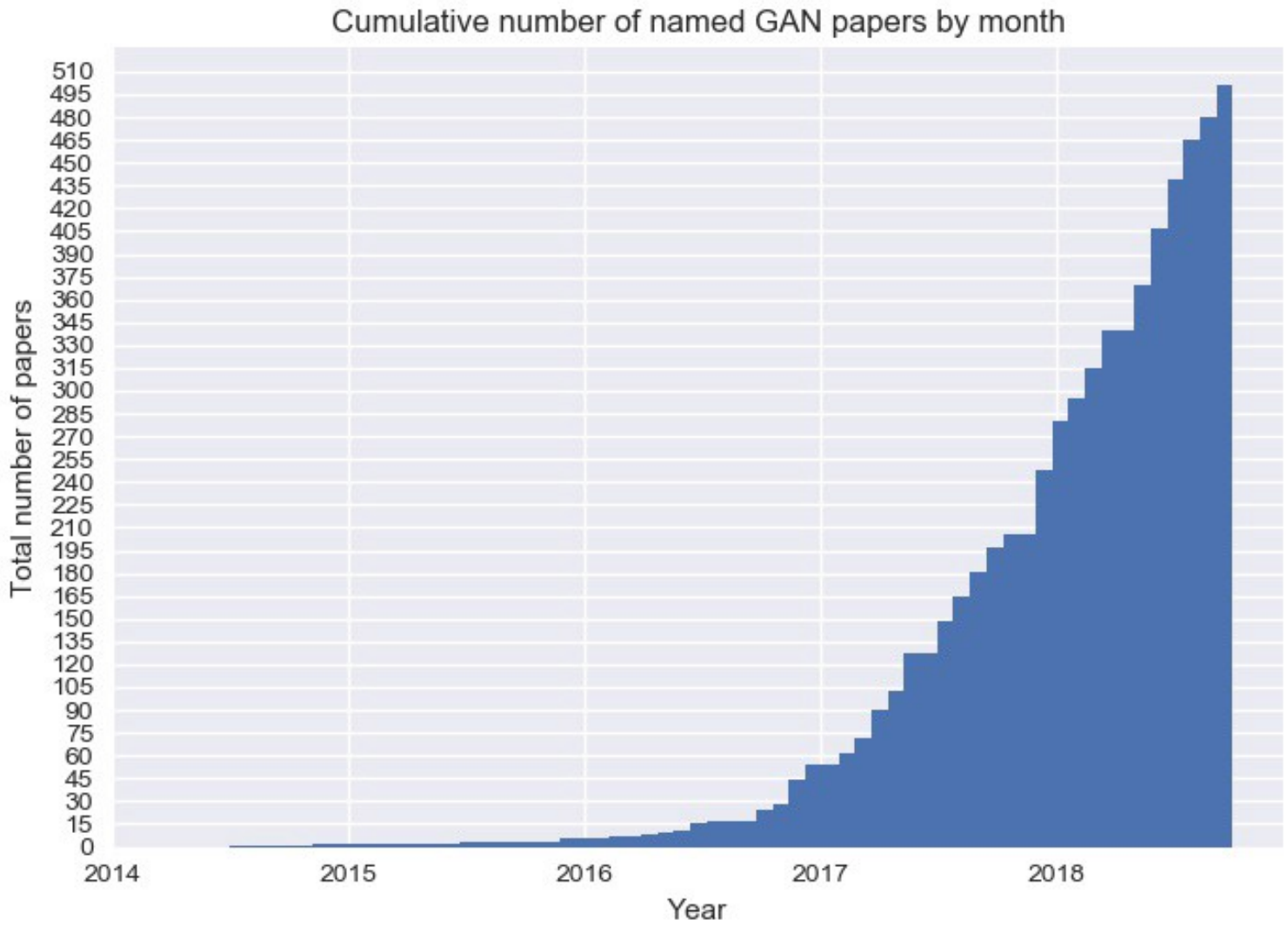
$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^{1}\boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^{l}\boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

$$\bar{W}_{\text{SN}}(W) := W/\sigma(W)$$

GAN Zoo and Applications

- Conditional image generation
- Image to Image translation
- Super-resolution
- Text to Image
- Image Inpainting
- Predicting Next Video Frame
- Text generation

GAN Zoo


Cumulative number of named GAN papers by month
Total number of papers
510
495
480
465
450
435
420
405
390
375
360
345
330
315
300
285
270
255
240
225
210
195
180
165
150
135
120
105
90
75
60
45
30
15
0
2014
2015
2016
2017
2018
Year

CGAN, ACGAN

Conditional Generative Adversarial Nets (2014)

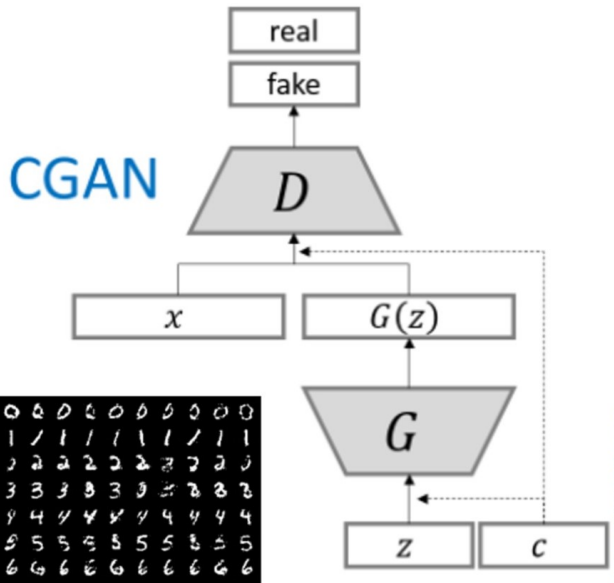
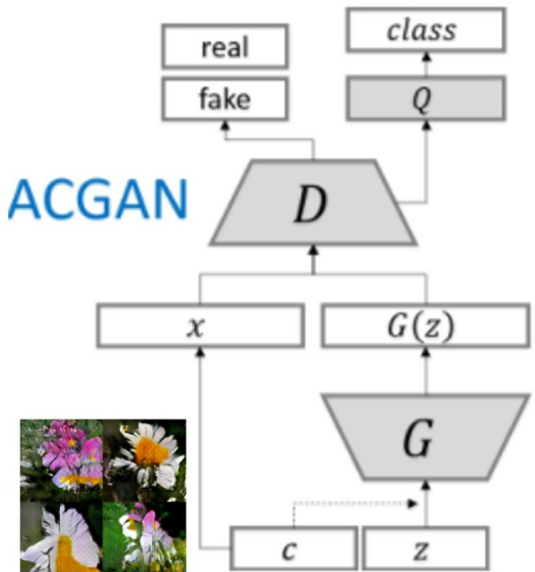


Figure 2: Generated MNIST digits, each row conditioned on one label

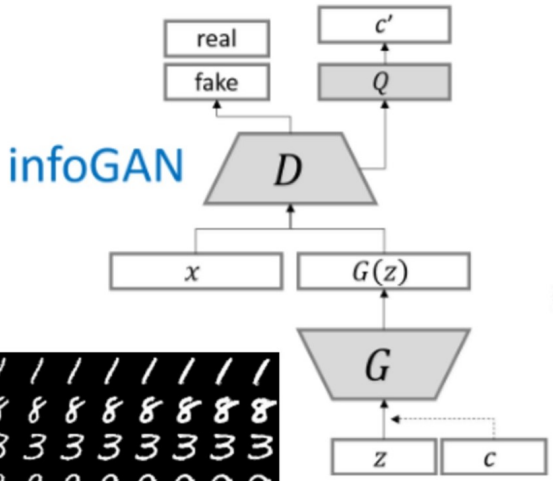
Conditional Image Synthesis With Auxiliary Classifier GANs (2016)

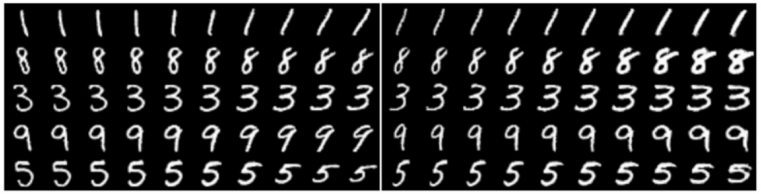


monarch butterfly goldfinch daisy

InfoGAN, SAGAN

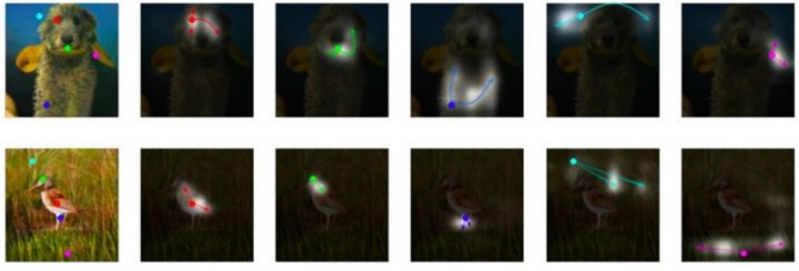
InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets (2016)





(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation) (d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Self-Attention Generative Adversarial Networks (2018)



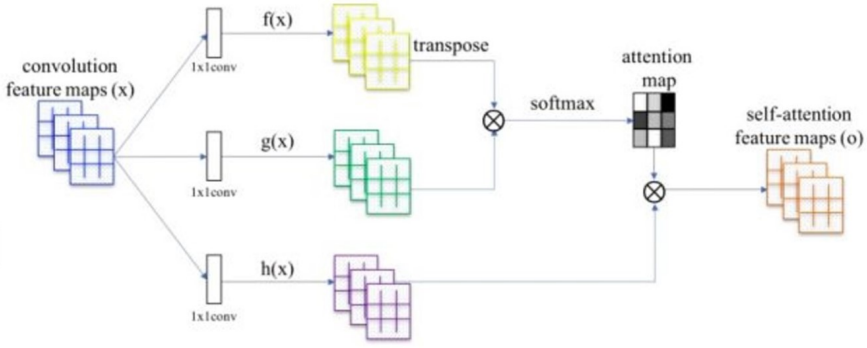


Image-to-image translation with conditional adversarial networks (2016)

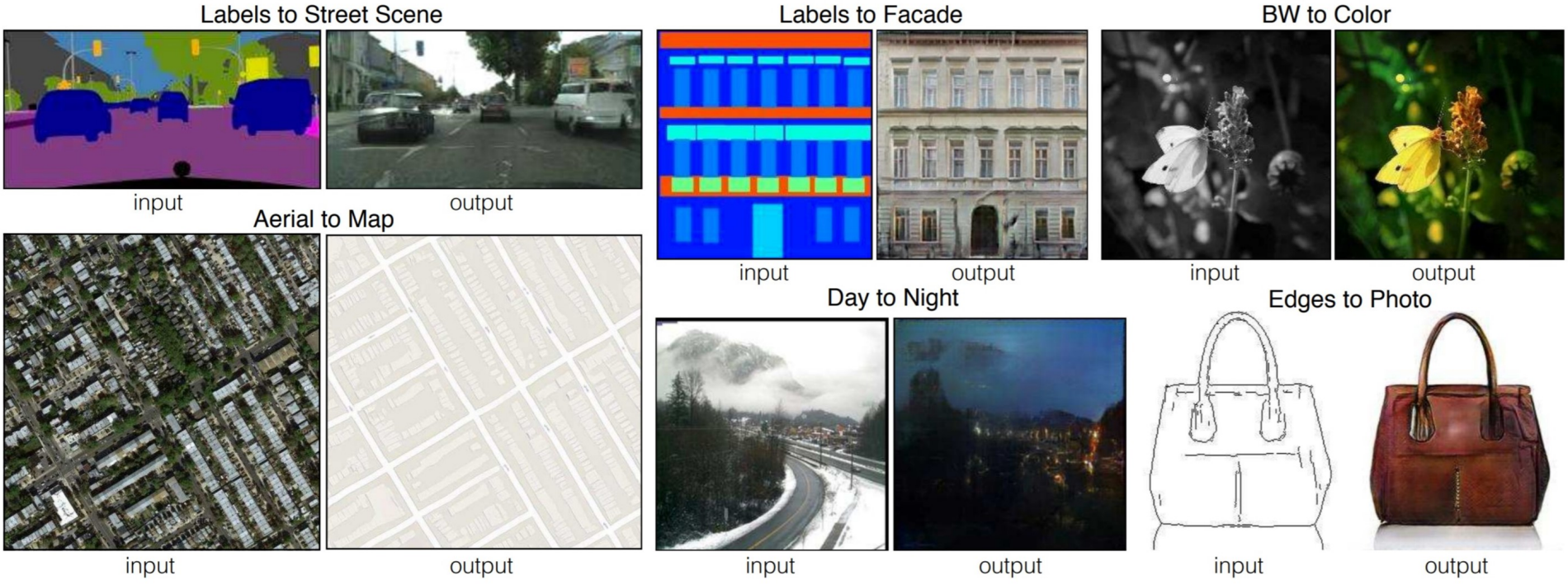


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs (2017)

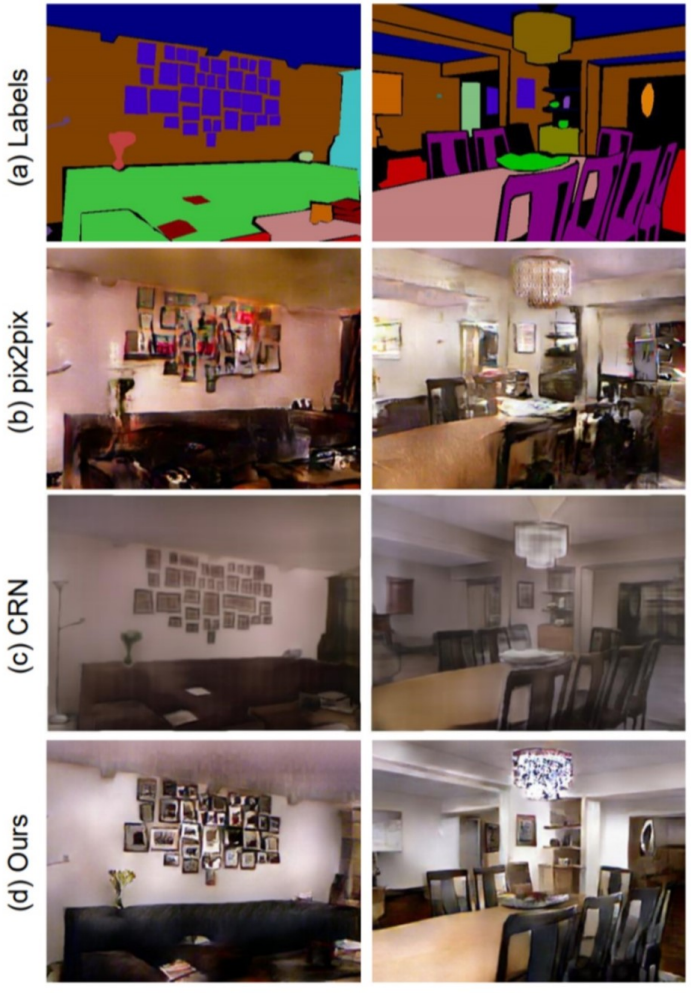


Figure 8: Comparison on the NYU dataset [40]. Our method generates more realistic and colorful images than the other methods.

Figure 10: Results on the ADE20K dataset [63] (label maps shown at lower left corners in (a)). Our method generates images at similar level of realism as the original images.

Figure 11: Diverse results on the Helen Face dataset [49] (label maps shown at lower left corners). With our interface, a user can edit the attributes of individual facial parts in real-time, such as changing the skin color or adding eyebrows and beards. See our video for more details.

StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation (2017)

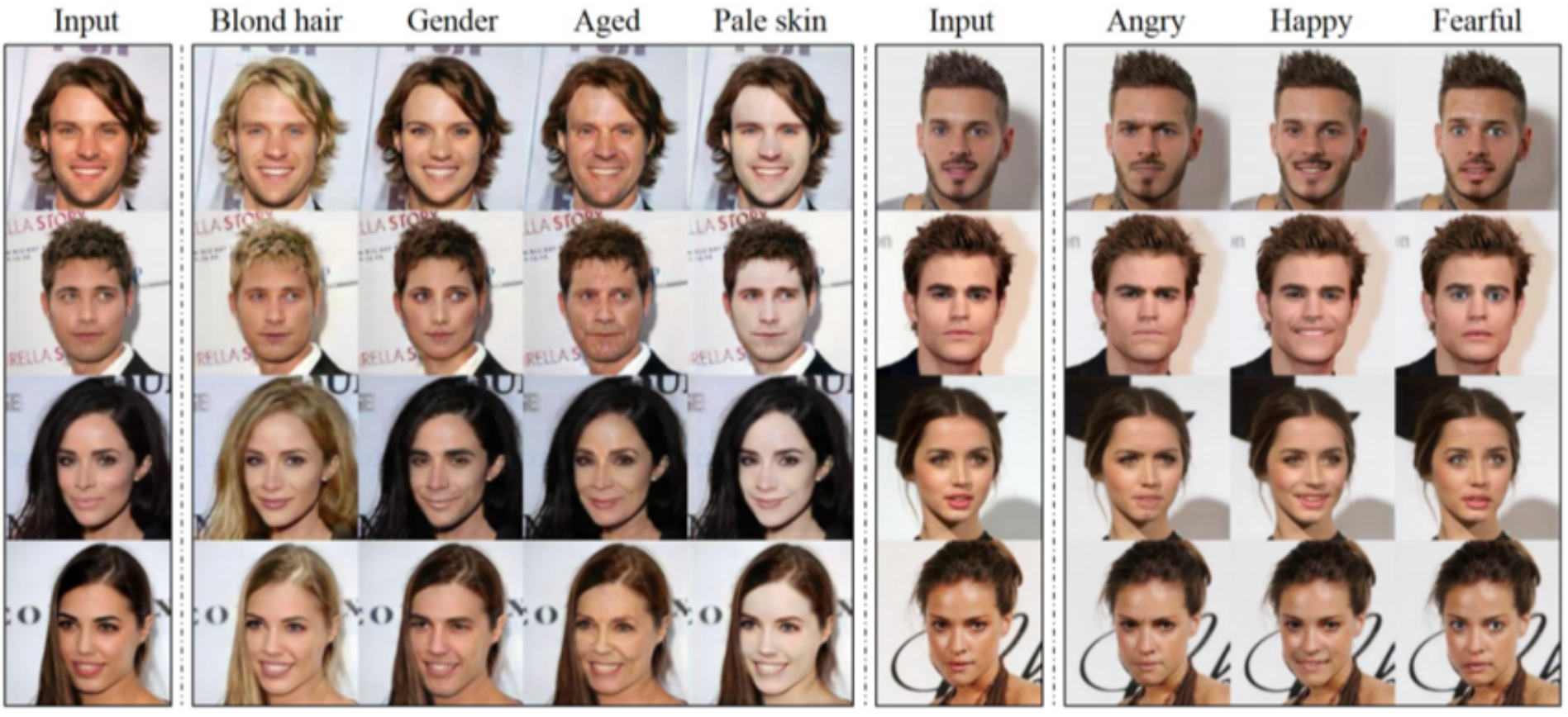


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2018)

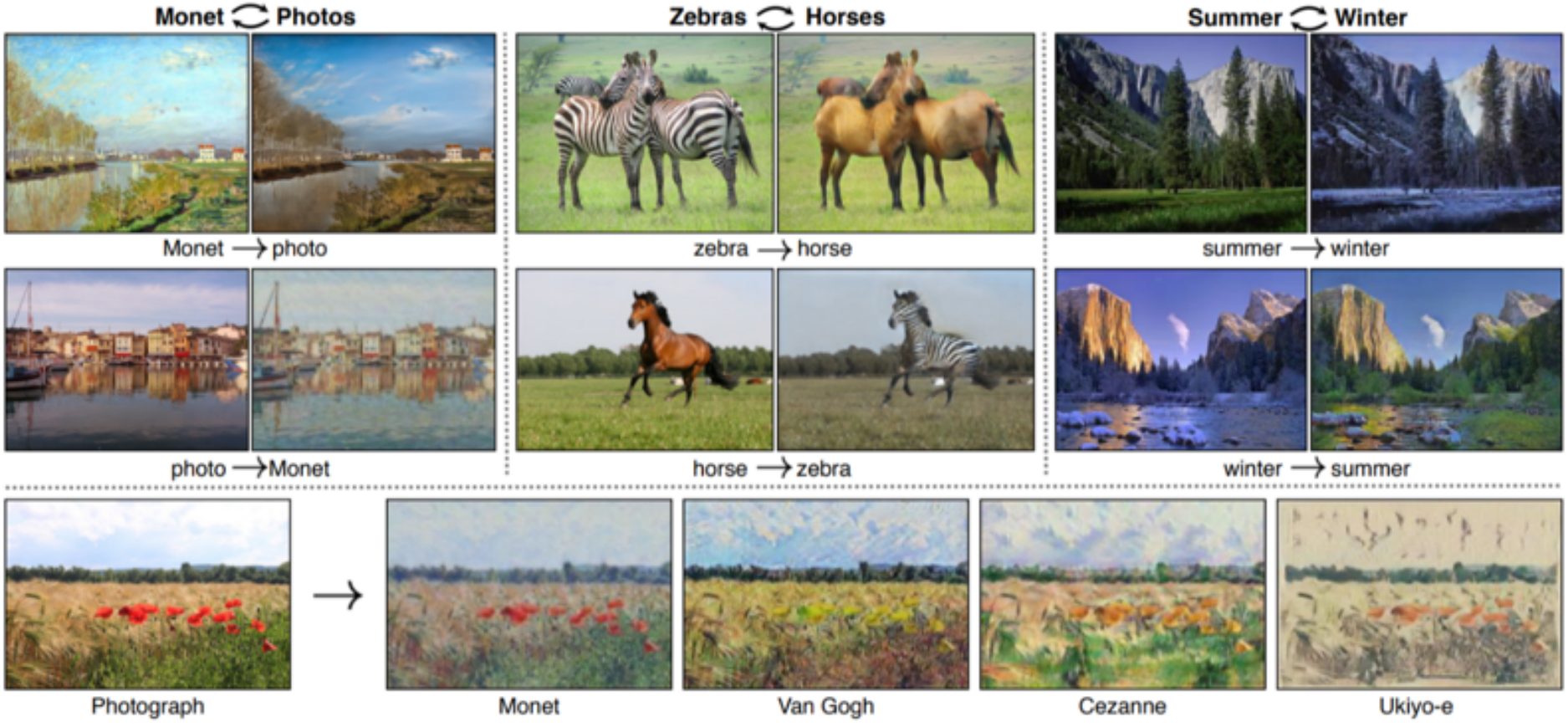


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

A Closed-form Solution to Photorealistic Image Stylization (2018)

(a) Content (b) Style (c) Gatys et al. [6] (d) Luan et al. [21] (e) Ours

Figure 1: Given a content photo (a) and a style photo (b), photorealistic image style transfer algorithms aim at transferring the style of the style photo to the content photo as shown in (c), (d), and (e). Comparing to the existing algorithms [6, 21], the proposed algorithm outputs photos that have more consistent stylizations and much fewer artifacts. Moreover, it runs an order of magnitude faster thanks to its closed-form formulation.

Image Super-Resolution

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (Ledig et al. 2016)

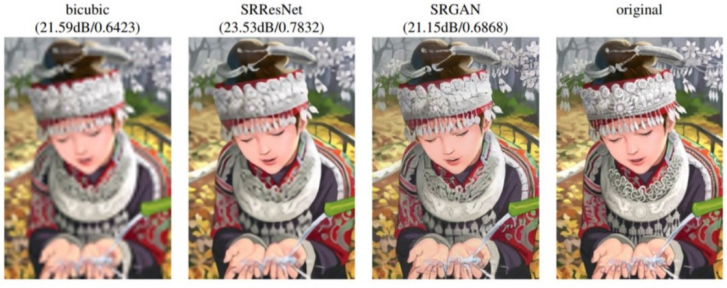


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Recovering Realistic Texture in Image Super-resolution by Deep Spatial Feature Transform (Wang et al. 2018)

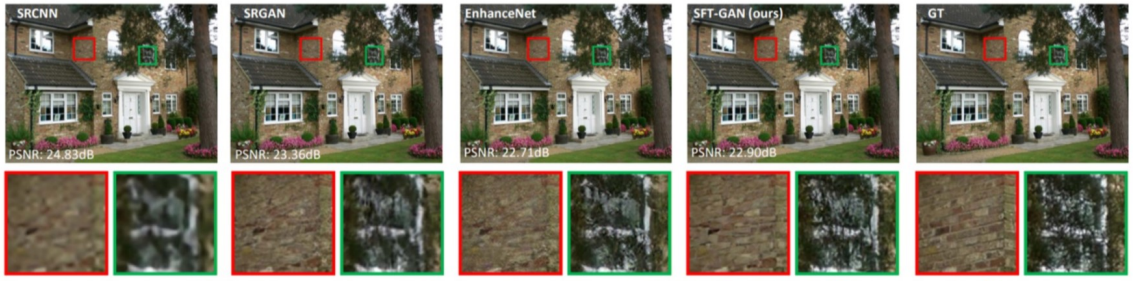
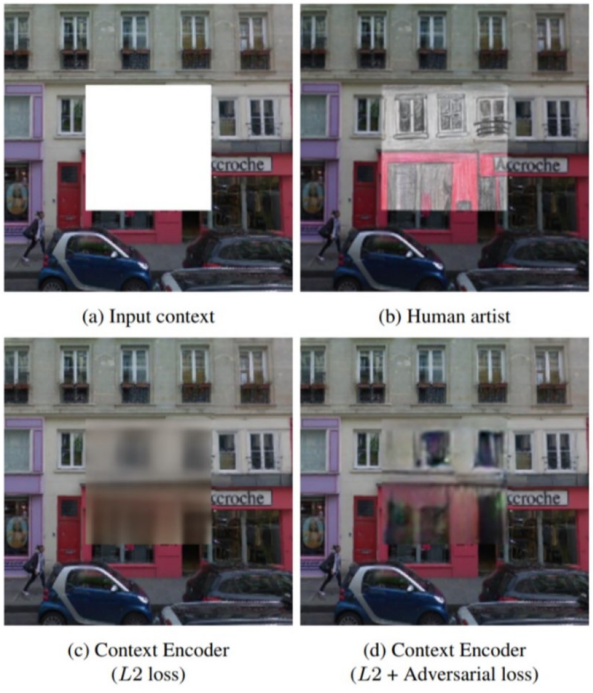


Figure 2. Comparing different SR approaches with downsampling factor ×4: SRCNN [7], SRGAN [27], EnhanceNet [38], our proposed SFT-GAN and the original HR image. SRGAN, EnhanceNet, and SFT-GAN clearly outperform SRCNN in terms of perceptual quality, although they yield lower peak signal-to-noise ratio (PSNR) values. SRGAN and EnhanceNet result in more monotonous textures across different patches while SFT-GAN is capable of generating richer and visually pleasing textures. (**Zoom in for best view**).

Image Inpainting

Context Encoders: Feature Learning by Inpainting (Pathak et al. 2016)



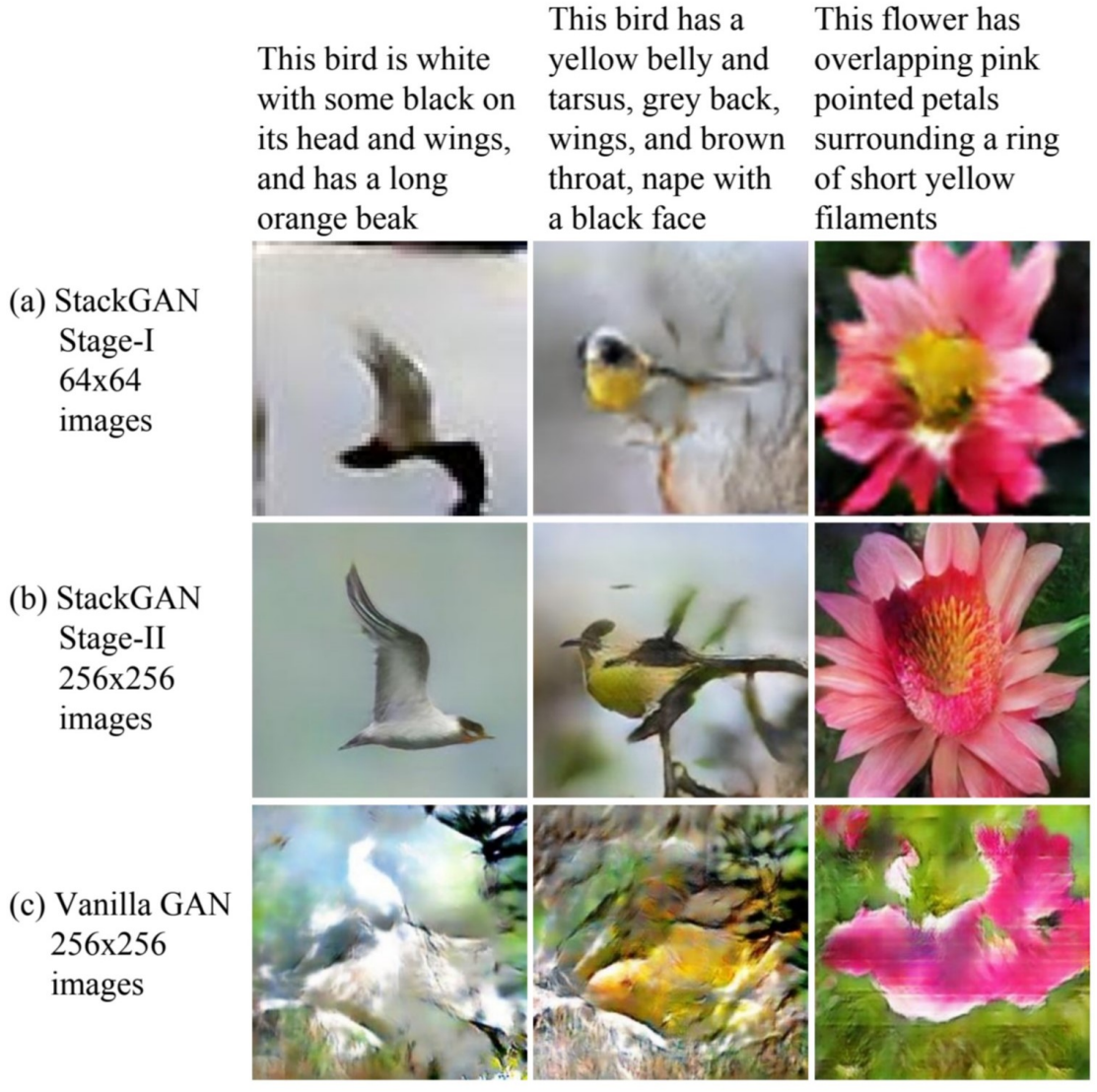
(a) Input context (b) Human artist

(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)

Generative Image Inpainting with Contextual Attention (Yu 2018)

Figure 1: Example inpainting results of our method on images of natural scene, face and texture. Missing regions are shown in white. In each pair, the left is input image and right is the direct output of our trained generative neural networks without any post-processing.

StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks (2017)



Predicting Next Video Frame

Unsupervised Learning of Visual Structure using Predictive Generative Networks (2016)

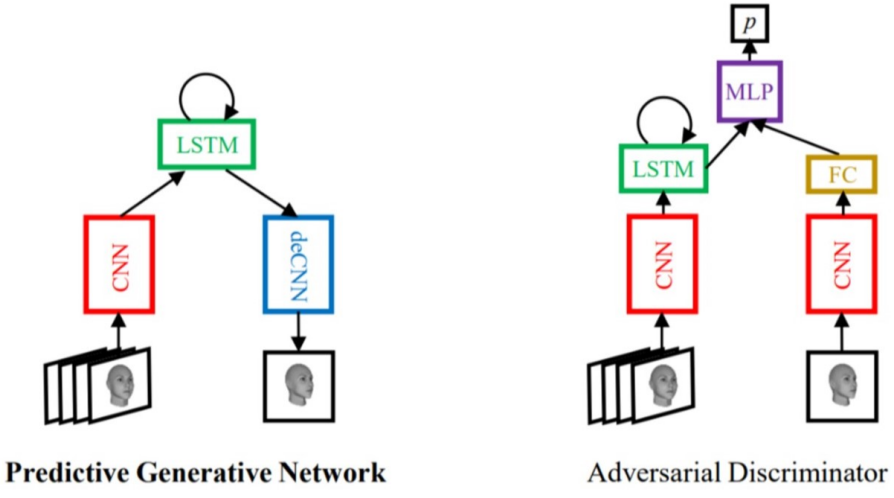
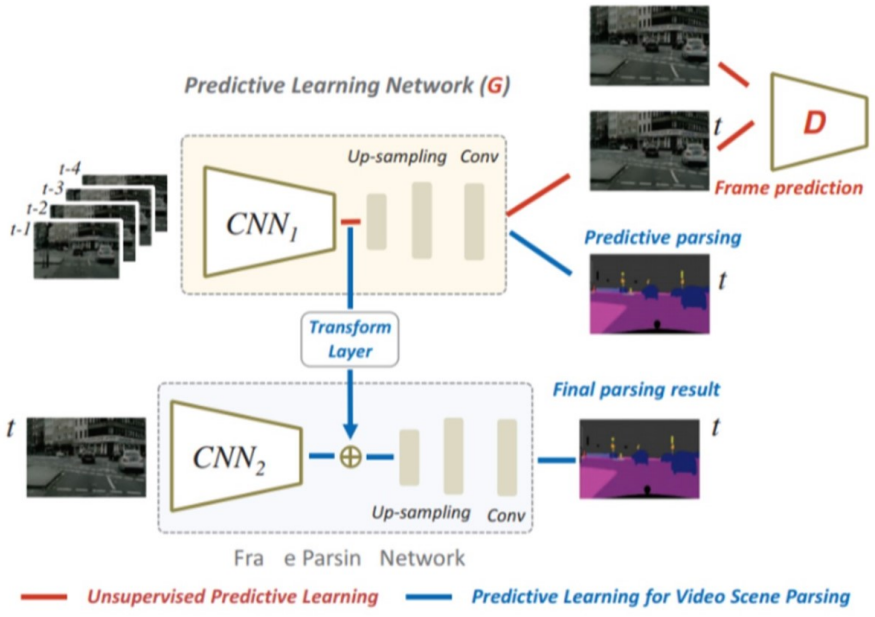


Figure 1: Predictive Generative Network (PGN)

Video Scene Parsing with Predictive Feature Learning (2017)



Text generation

- SeqGAN (SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. Yu et al. 2017)
- MaliGAN (Maximum-Likelihood Augmented Discrete Generative Adversarial Networks. Che et al. 2017)
- RankGAN (Adversarial Ranking for Language Generation. Lin et al. 2017)
- TextGAN (Adversarial Feature Matching for Text Generation. Zhang 2017)
- GSGAN (Gans for sequences of discrete elements with the gumbel-softmax distribution. Kusner 2017.)
- LeakGAN (Long Text Generation via Adversarial Training with Leaked Information. Guo 2017)
- MaskGAN: BETTER TEXT GENERATION VIA FILLING IN THE ___ (Fedus 2018)

Takeaways

- GAN: Generative Adversarial Network
- GAN is good, but training it is hard
- Name is important!

painting with a GAN
this cat does not exist
this person does not exist
this airbnb does not exist

Further readings

- NIPS 2016 tutorial: Generative adversarial networks.
- ICCV 2017 Tutorial on GANs.
- CVPR 2018 Tutorial on GANs.
- really-awesome-gan

References I

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[4] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[6] Bin Dai and David Wipf. Diagnosing and enhancing vae models. 2018.

[7] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[8] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[13] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

References II

[14] Wei Li, Melvin Gauci, and Roderich Groß. A coevolutionary approach to learn animal behavior through controlled interaction. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 223–230. ACM, 2013.

[15] Wei Li, Melvin Gauci, and Roderich Groß. Turing learning: a metric-free approach to inferring behavior and its application to swarms. *Swarm Intelligence*, 10(3):211–243, 2016.

[16] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[17] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[18] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*, 2016.

[19] Evgeny Putin, Arip Asadulaev, Quentin Vanhaelen, Yan Ivanenkov, Anastasia V Aladinskaya, Alex Aliper, and Alex Zhavoronkov. Adversarial threshold neural computer for molecular de novo design. *Molecular pharmaceutics*, 15(10):4386–4397, 2018.

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[21] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[22] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[23] Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

References III

[25] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[26] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[27] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

[28] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[29] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic autonomous driving system testing. *arXiv preprint arXiv:1802.02295*, 2018.

[30] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[31] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.