CS60010: Deep Learning

Spring 2023

Sudeshna Sarkar

Generative Adversarial Network

24 Mar 2023

VAE vs GAN

- VAEs define intractable density function with latent z

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

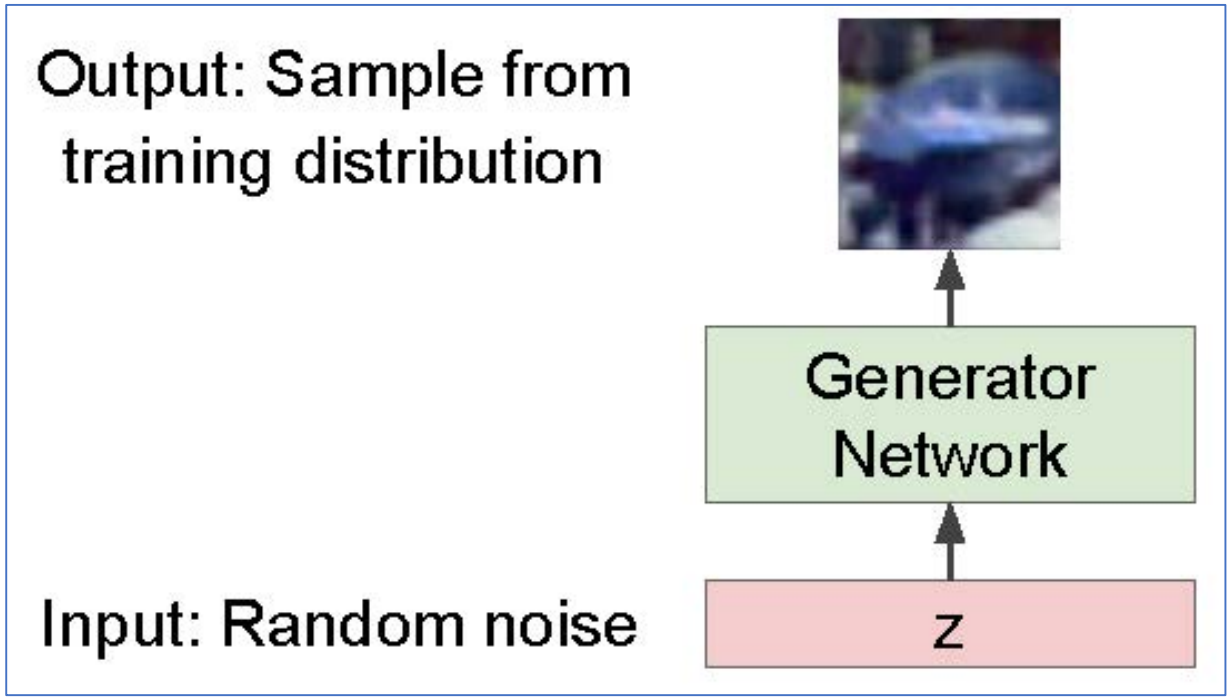
- Optimize lower bound on likelihood.
- Instead of explicitly modeling density, just get the ability to sample.
- GANs : does not model any explicit density function

Only want to sample

Don’t explicitly model density, instead just sample to generate new instances

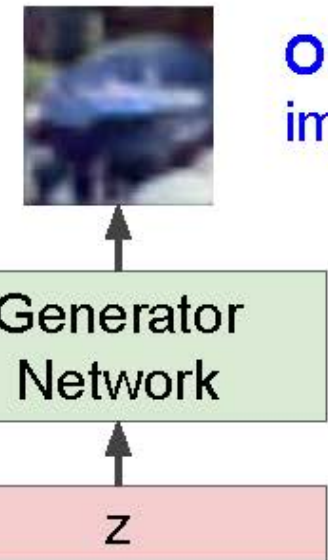
Problem: want to sample from complex distribution – can’t do directly

Solution: sample from random noise – learn a transformation to the data distribution



But we don't know which sample z maps to which training image -> can't learn by reconstructing training images

Output: Sample from training distribution

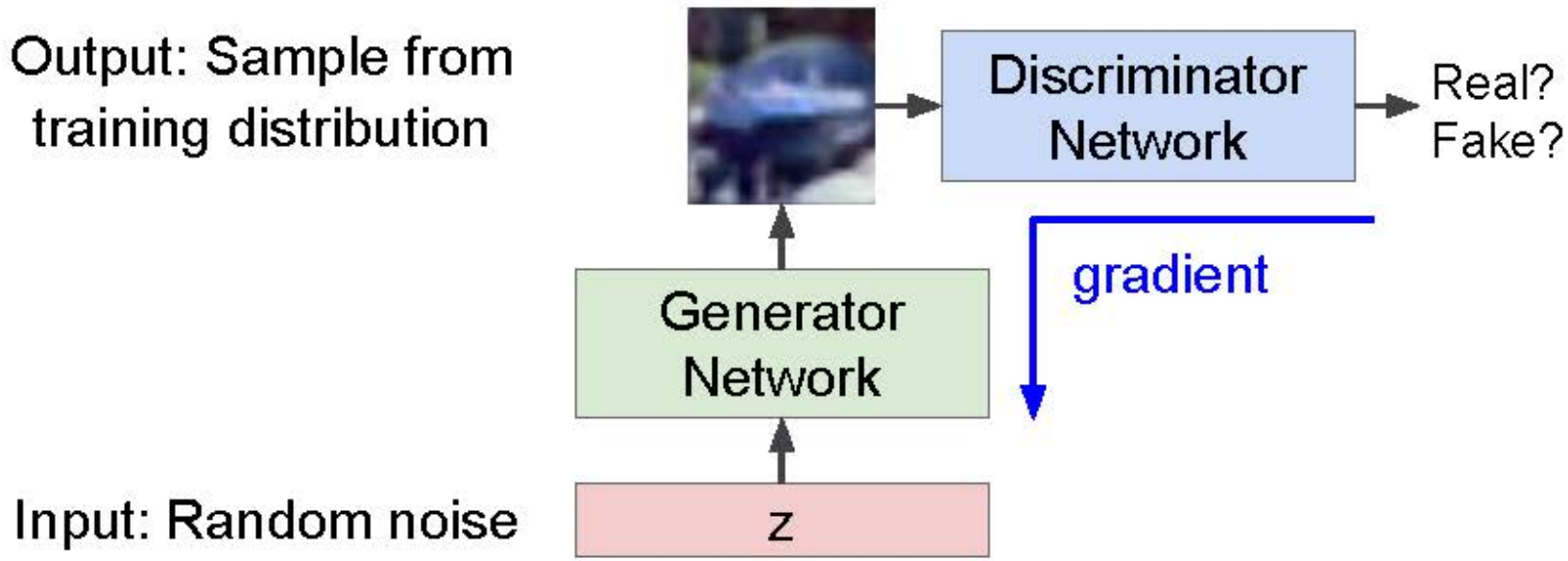


Input: Random noise

Objective: generated images should look "real"

But we don't know which sample z maps to which training image -> can't learn by reconstructing training images

Solution: Use a discriminator network to tell whether the generate image is within data distribution ("real") or not



Generative Adversarial Networks

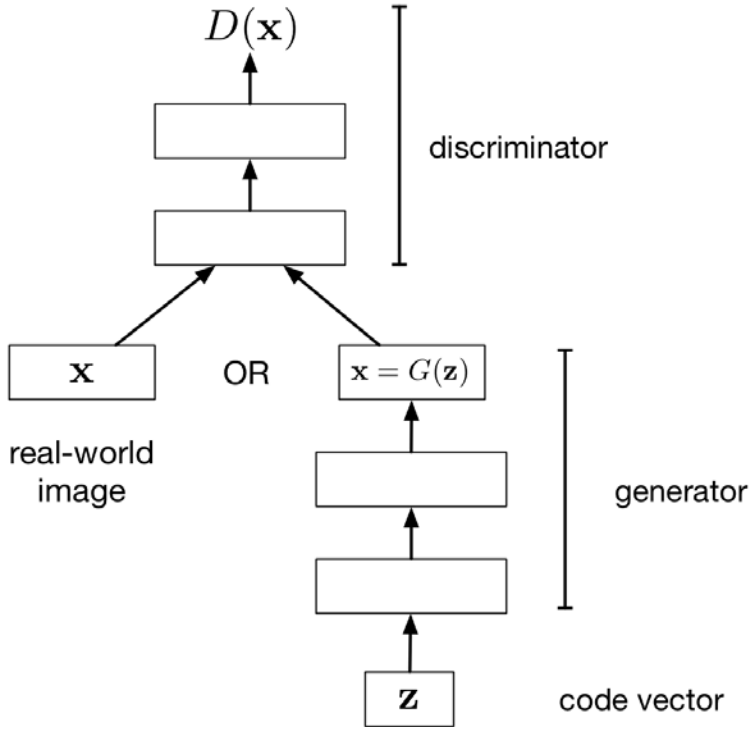
GAN uses a two-network architecture

Discriminator network:

- Try to distinguish between real and fake images
- Compare real and synthetic images as an approximation to the human perceptual difference

Generator network:

- Try to fool the discriminator by generating real looking images



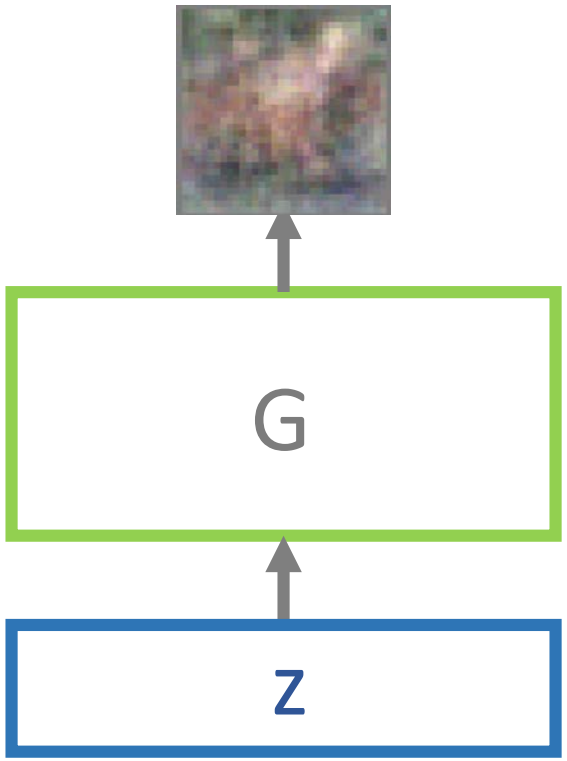
- GANs learn a transformation $p(x|z)$ from a known simple distribution $p(z)$ (e.g. Conditionally Independent Gaussian).

Adversarial Training

- **Any fixed discriminator model will probably be easy to fool**
- But since the discriminator have **learnable parameters**, it can learn to reject new synthetic images
- This is a **two-player minimax game**: generator tries to increase discriminator loss, discriminator tries to decrease it.

Generation Network (G)

Learn a transformation using a Neural network from a simple distribution to the target distribution



Output: Image

Transformation:

Neural-network,

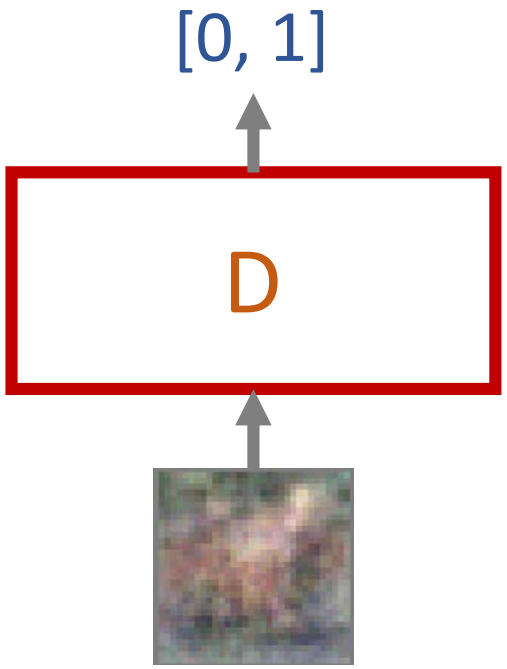
Can have many different architectures

depending on the task

Input: Sample from $N(0, I)$

Discriminator Network (D)

- We want to be able to learn a good transformation so that we can generate samples similar as the samples from the data.
- Measuring the 'goodness' of the samples is hard, but we can use a neural network for it.



Output: "Realism" of the Image

(Probability of the image being real)

Discriminator:

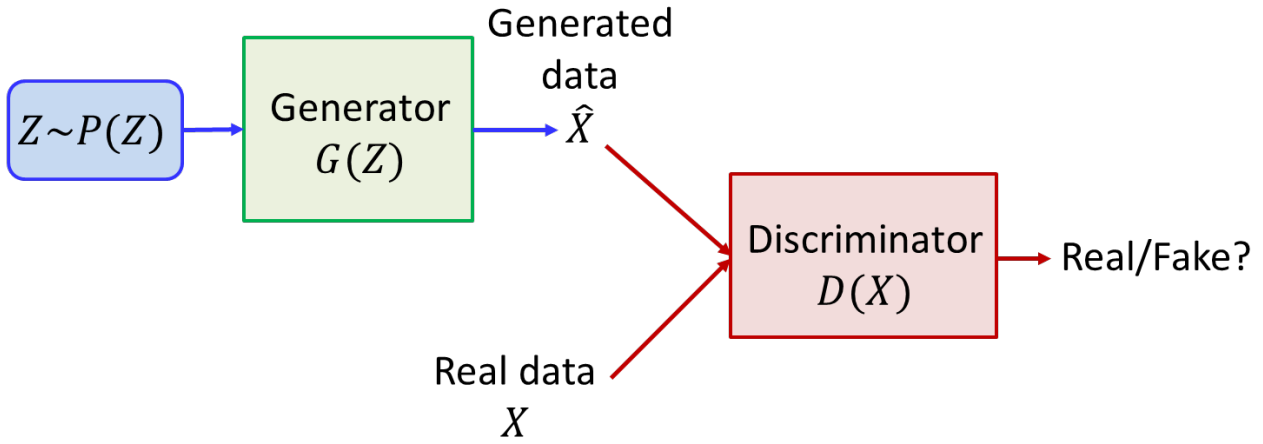
Tries to distinguish between generated

samples or samples from the dataset

Input: Generated/Real Image

- We use 'how distinguishable is the generated samples by the discriminator network' as a proxy to the 'goodness' of examples
- All differentiable, so can be trained end-to-end with the generator

The GAN formulation



- For real data X, the desired output of the discriminator is $D(X) = 1$
  - The log probability that the instance is real, as computed by the discriminator is $\log D(X)$
- For synthetic data $\hat{X}$ the desired output of the discriminator is $D(\hat{X}) = 0$
  - The log probability that the instance is synthetic, as computed by the discriminator, is

$$\log\left(1 - D(\hat{X})\right) = \log\left(1 - D(G(Z))\right)$$

Training GANs : Two-player game

- **Discriminator Network (D):** Tries to distinguish between real images and fake images
- **Generation Network (G):** Tries to fool D by generating realistic images

Train jointly in **minimax game**

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

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

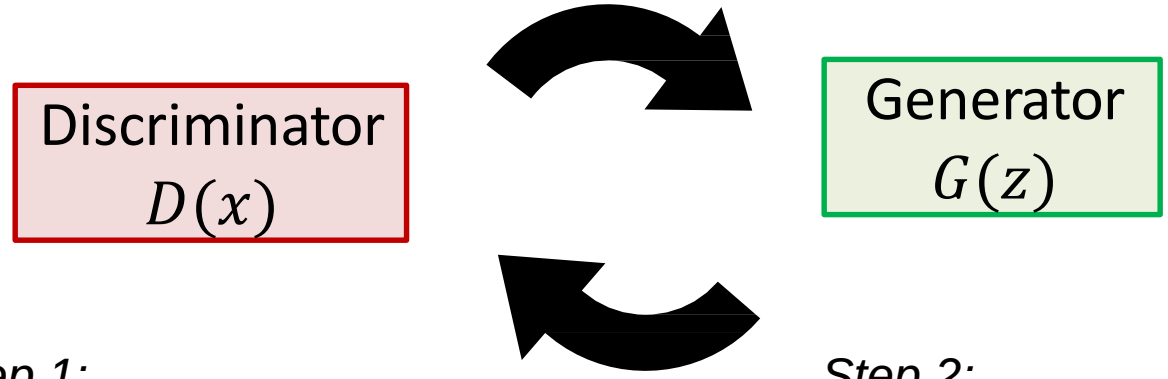
Log Probability of x (real data) considered as 'real'

1 – log Probability of generated data considered as 'real'

- 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)

This also minimizes Jensen-Shannon (JS) Divergence between x and $G(z)$, such that it minimizes the distance between the distributions of real data and generated data

How to Train a GAN?



Step 1:
Train the Discriminator
using the current Generator

Step 2:
Train the Generator
to beat the Discriminator

$$\min_G \max_D E_x \log D(x) + E_z \log\big(1 - D(G(z))\big)$$

The discriminator is not needed after convergence

Training GANs: Two-player game

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

Minimax objective function:

$$\min_G \max_D E_x \log D(x) + E_z \log\big(1 - D(G(z))\big)$$

Alternate between:

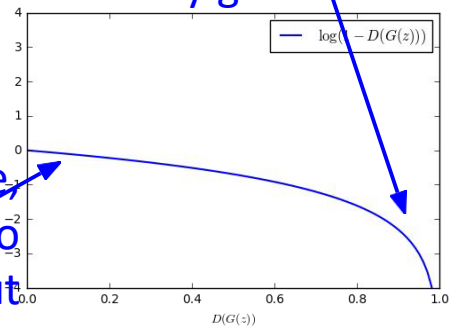
1. **Gradient ascent** on discriminator

$$\max_D E_x \log D(x) + E_z \log\big(1 - D(G(z))\big)$$

2. **Gradient descent** on generator

$$\min_G E_z \log\big(1 - D(G(z))\big)$$

Gradient signal dominated by region where sample is already good



When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

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.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generative Adversarial Networks

Two-player Mini-max Game with Value function V(G, D):

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

- 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.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generative Adversarial Networks

Two-player Mini-max Game with Value function V(G, D):

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

Remember only to train the set of weights (G/D) when optimizing the loss, and freezing the rest of the weights.

- 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.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Training GANs: Two-player game

Minimax objective function:

$$\min_G \max_D E_x \log D(x) + E_z \log\big(1 - D(G(z))\big)$$

Alternate between:

1. **Gradient ascent** on discriminator

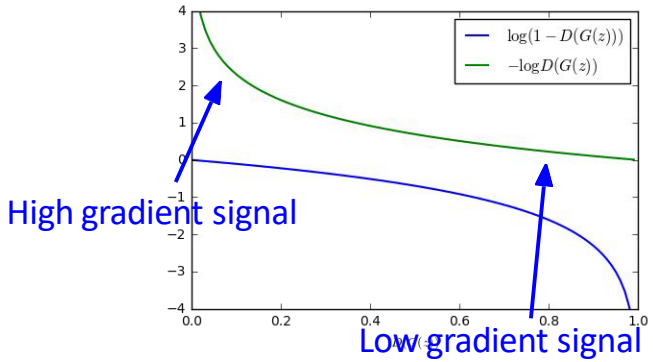
$$\max_D E_x \log D(x) + E_z \log\big(1 - D(G(z))\big)$$

2. ~~**Gradient descent** on generator~~

$$\min_G E_z \log\big(1 - D(G(z))\big)$$

Instead: Gradient ascent on generator, different objective

$$\max_G E_z \log\big(D(G(z))\big)$$



Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better!

Practical Aside:

For G, rather than minimizing:

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

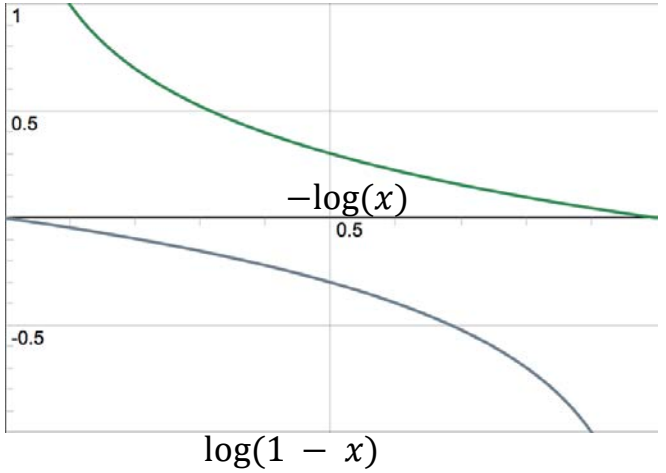
Maximize:

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

Why?

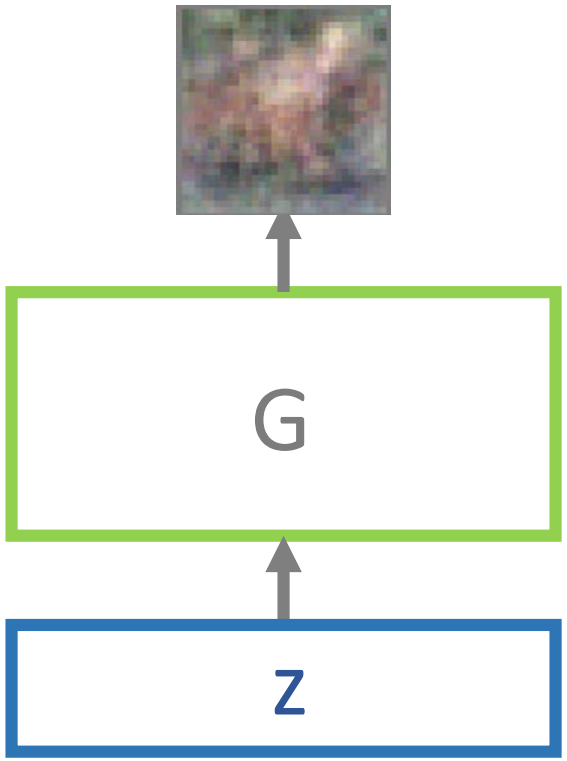
Large Gradients when generated samples are bad

Large Gradients when generated samples are already good



Generation Network (G)

After training, simply take the Generator network to generate new samples.



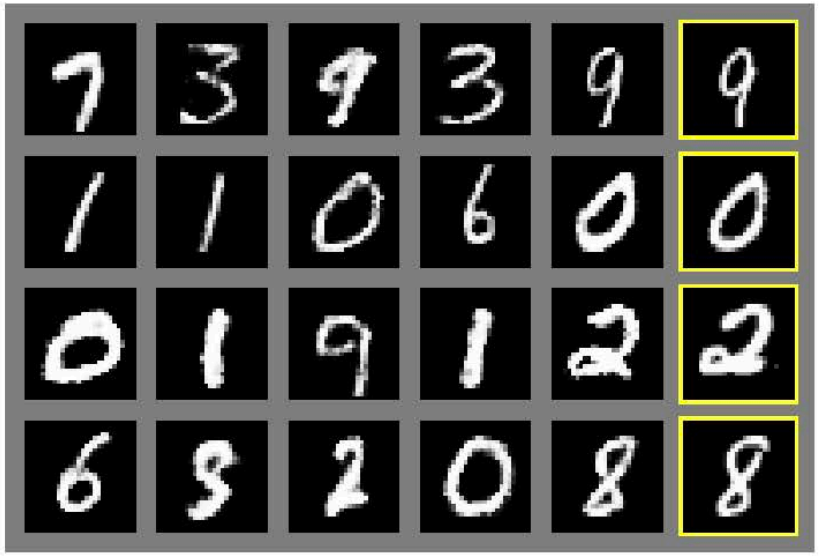
Output: Image

Transformation:
Neural-network,
Can use many different architectures depending on the task

Input: Sample from $N(0, I)$

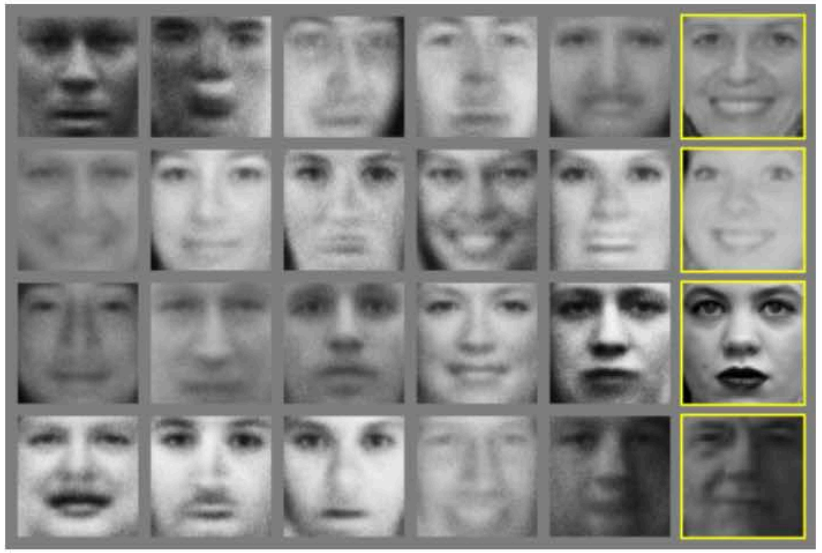
Generation Examples

- Original GAN paper showed good samples for
  - MNIST
  - Toronto Face Database



Generated Images

NN (Nearest-neighbor) In Dataset

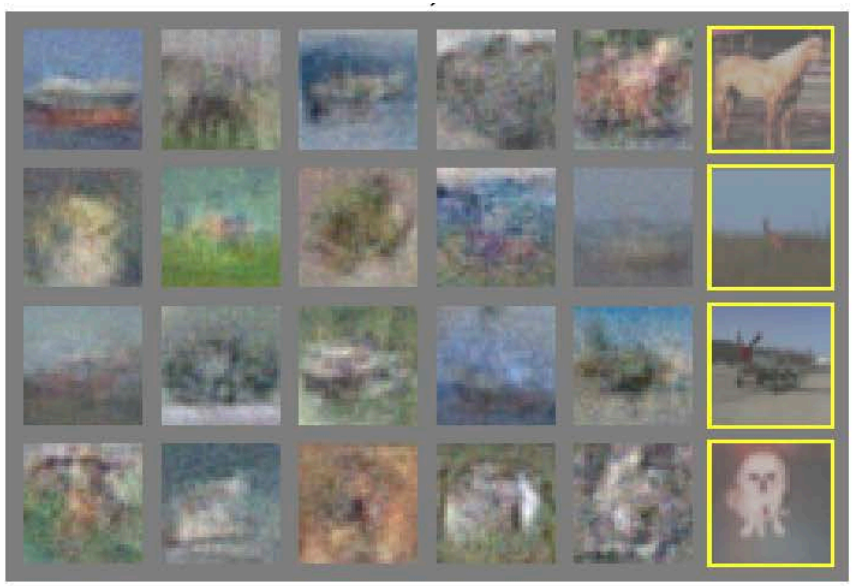


Generated Images

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generation Examples

- Original GAN paper produced perceptually poor image samples for CIFAR:



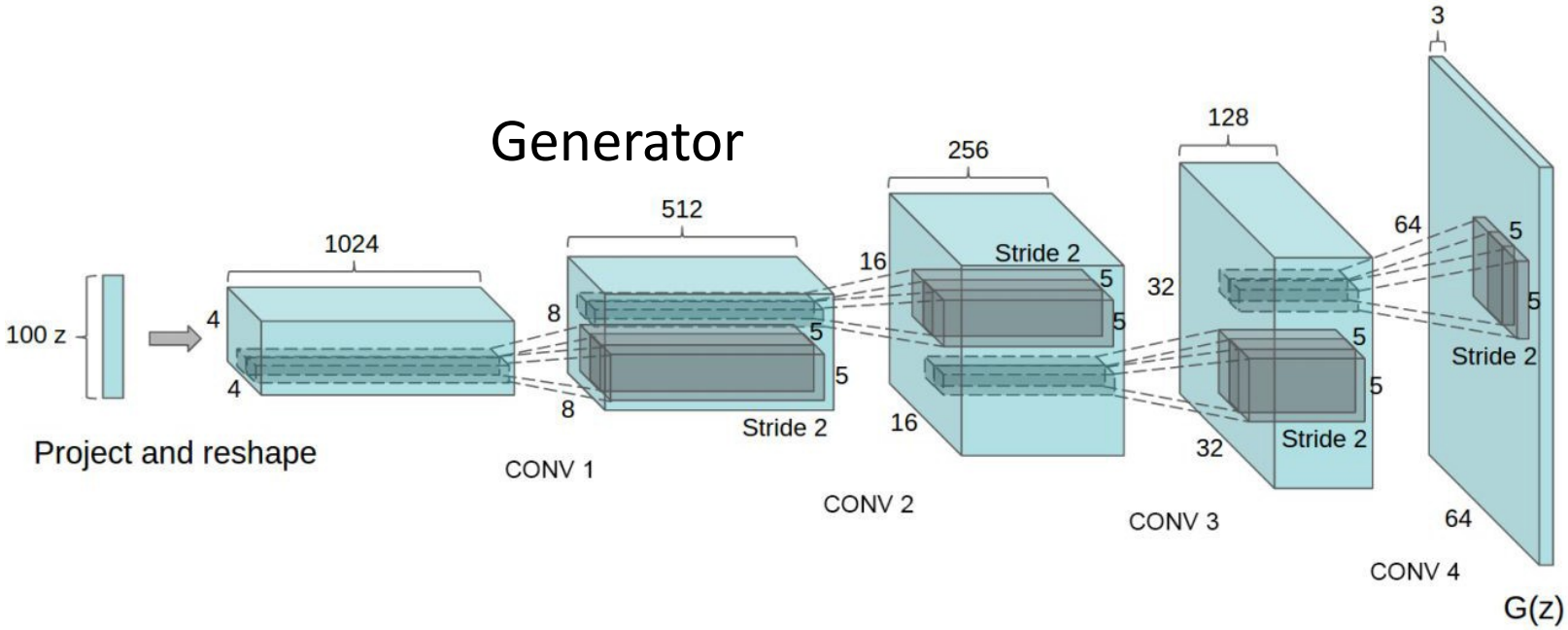
Generated Images **NN In Dataset**

Generated Images **NN In Dataset**

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

GANs + CNNs:
Convolutional Architectures for Generative Adversarial Networks

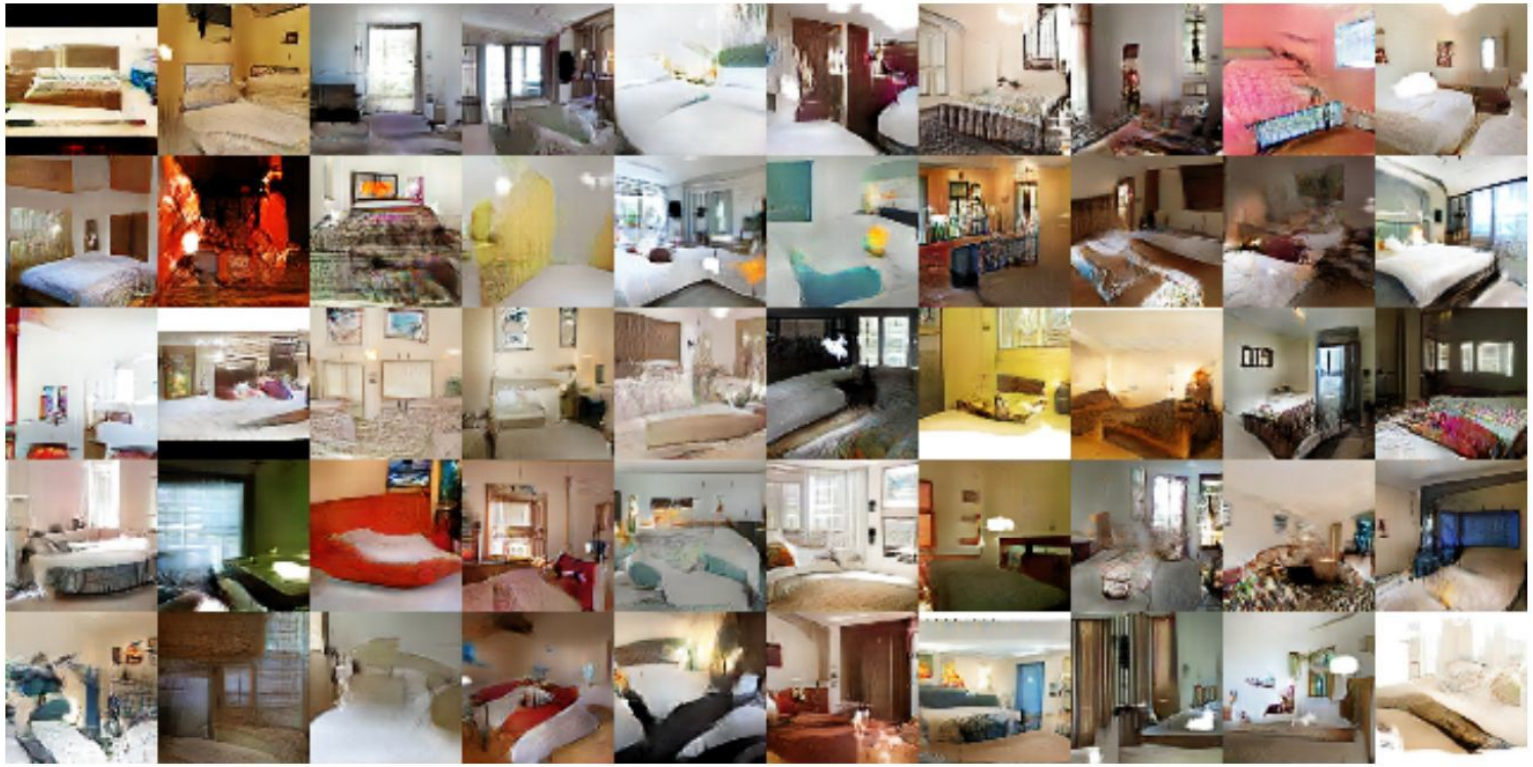
Generative Adversarial Nets: Convolutional Architectures



- Replacing any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Using batchnorm in both the generator and the discriminator.
- Removing fully connected hidden layers for deeper architectures.
- Using ReLU activation in generator for all layers except for the output, which uses tanh.
- Using LeakyReLU activation in the discriminator for all layer.

Generative Adversarial Nets: Convolutional Architectures

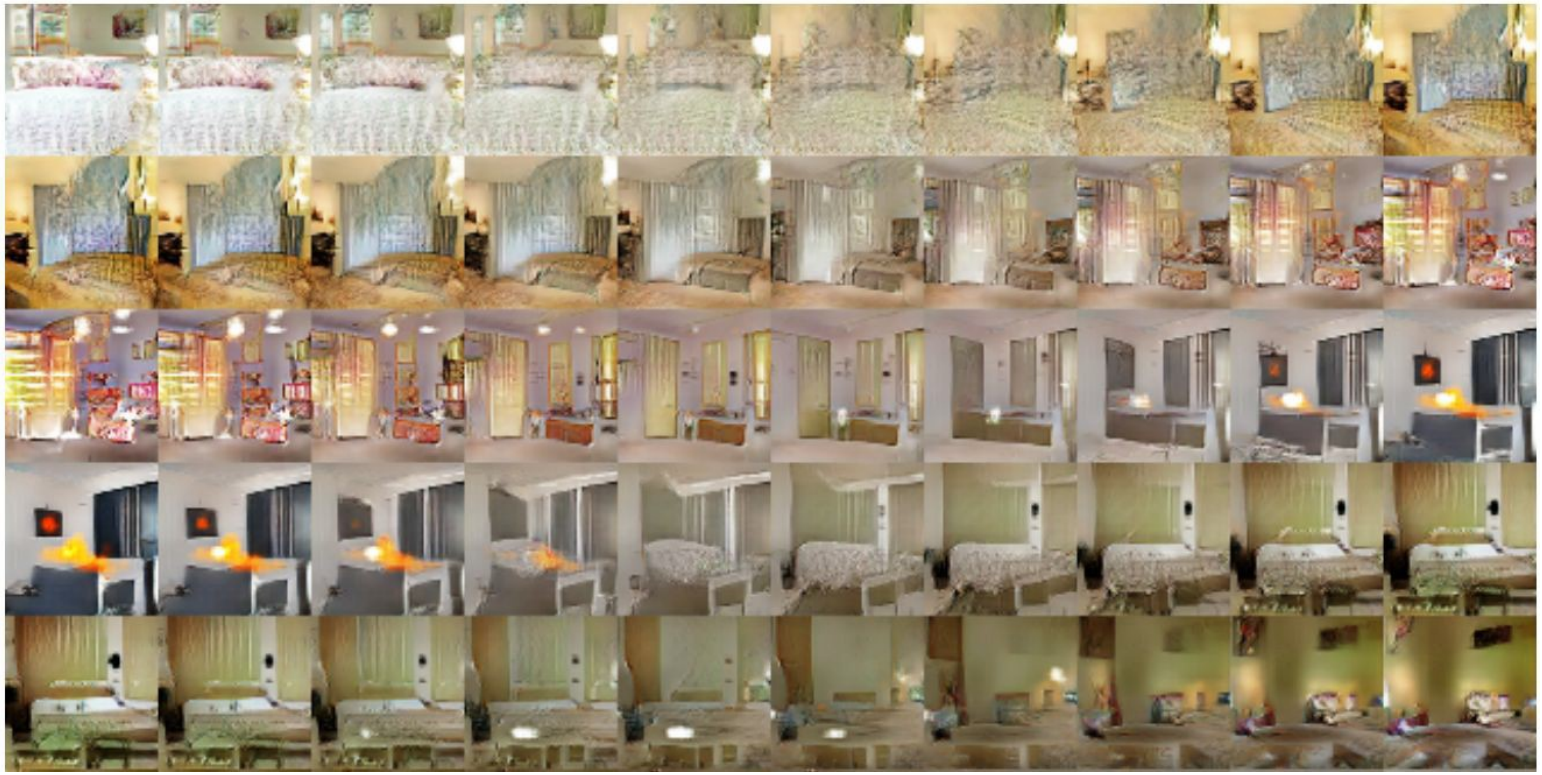
Samples from the model look much better!



Radford et al, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

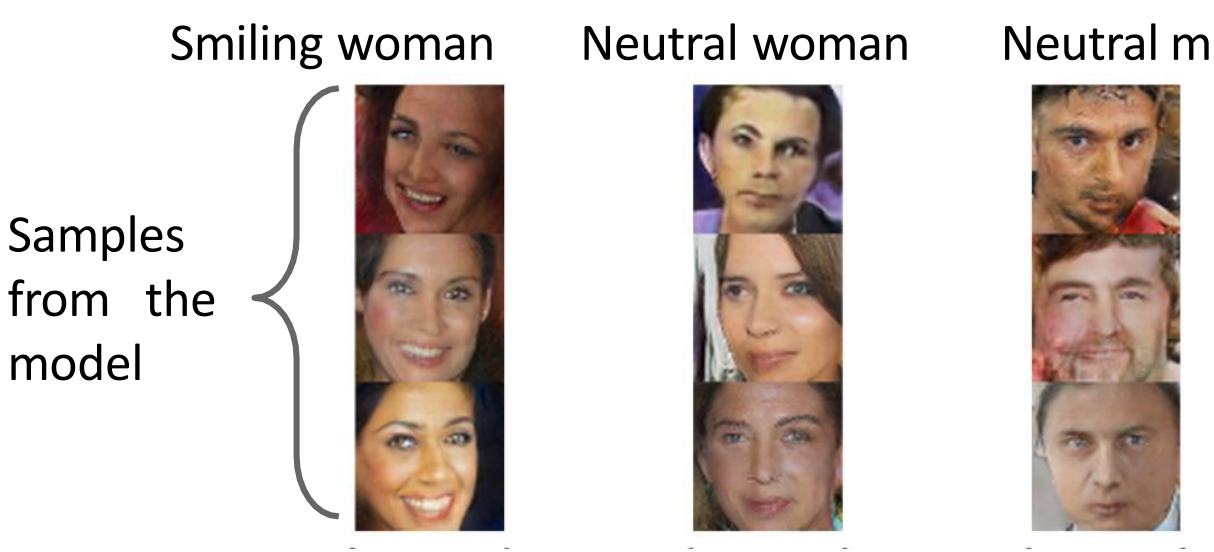
Interpolating between random points in latent space



Radford et al, ICLR 2016

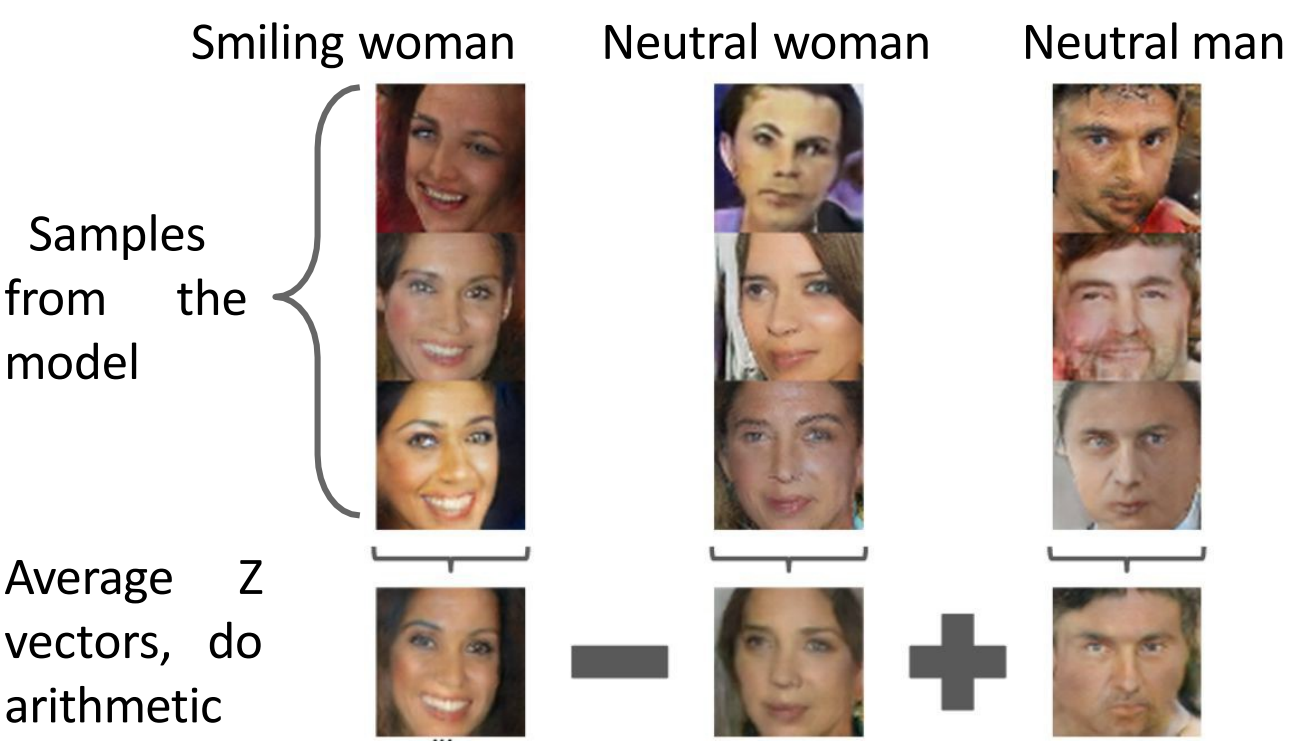
Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



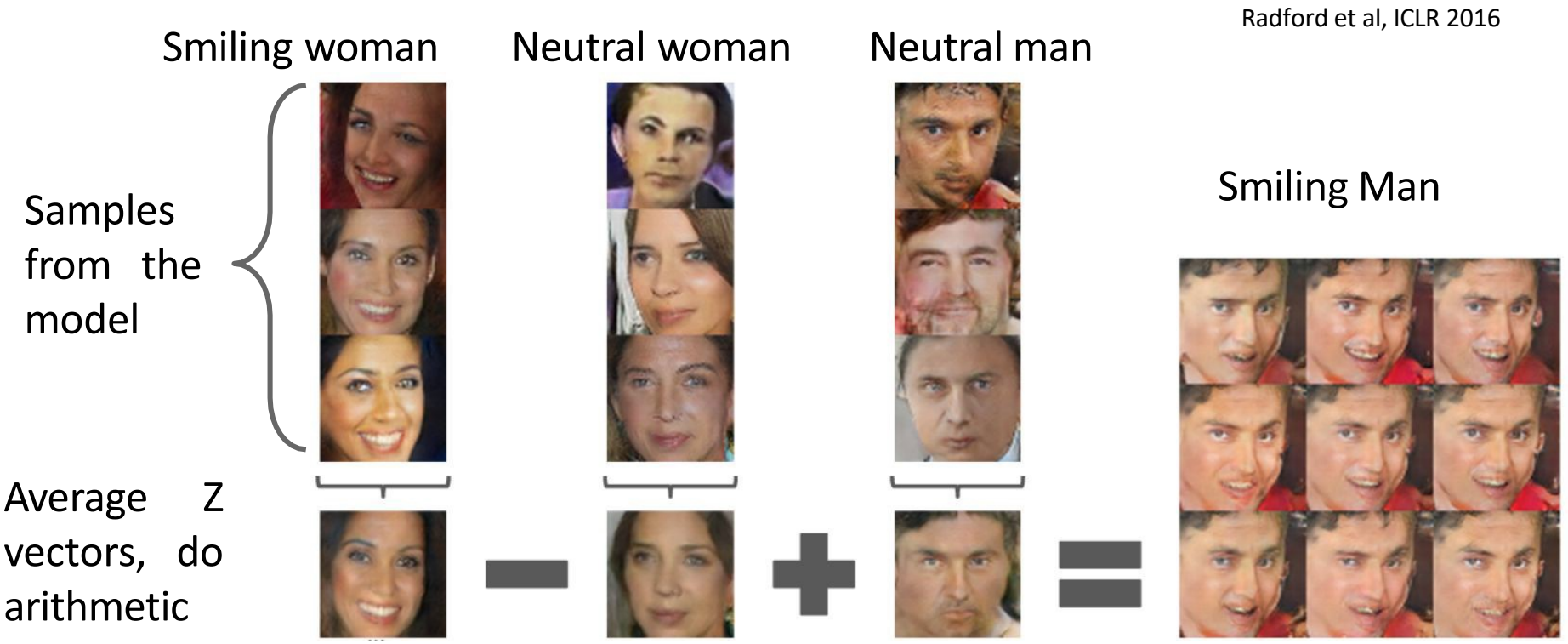
Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016

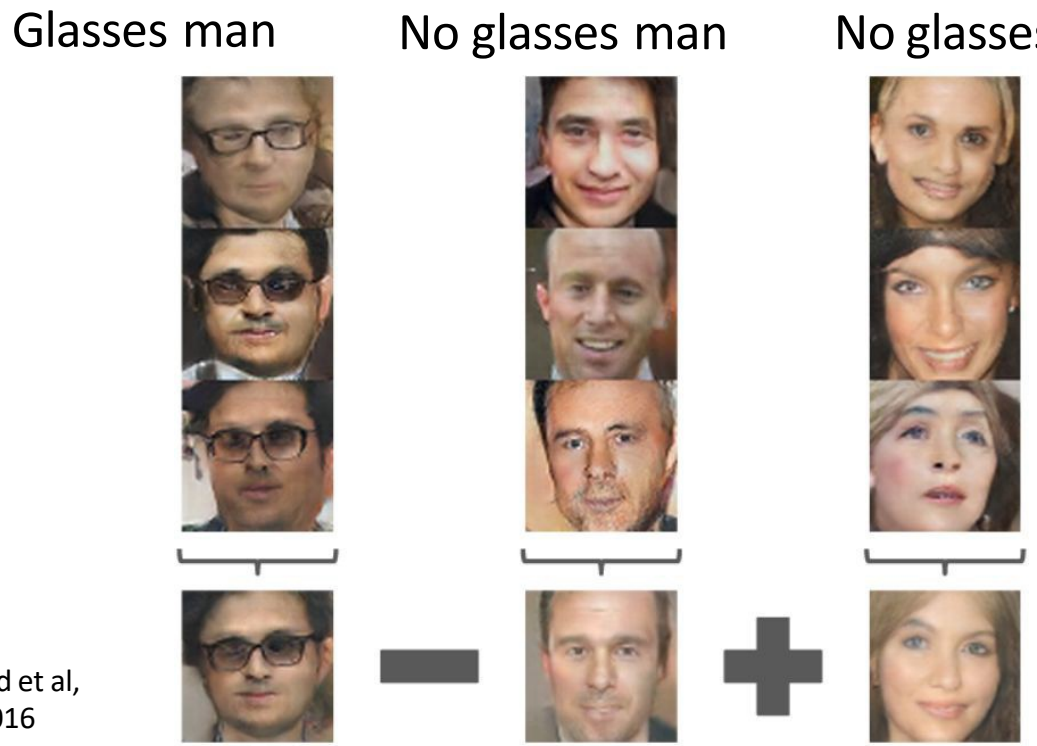


Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



Generative Adversarial Nets: Interpretable Vector Math



Radford et al,
ICLR 2016

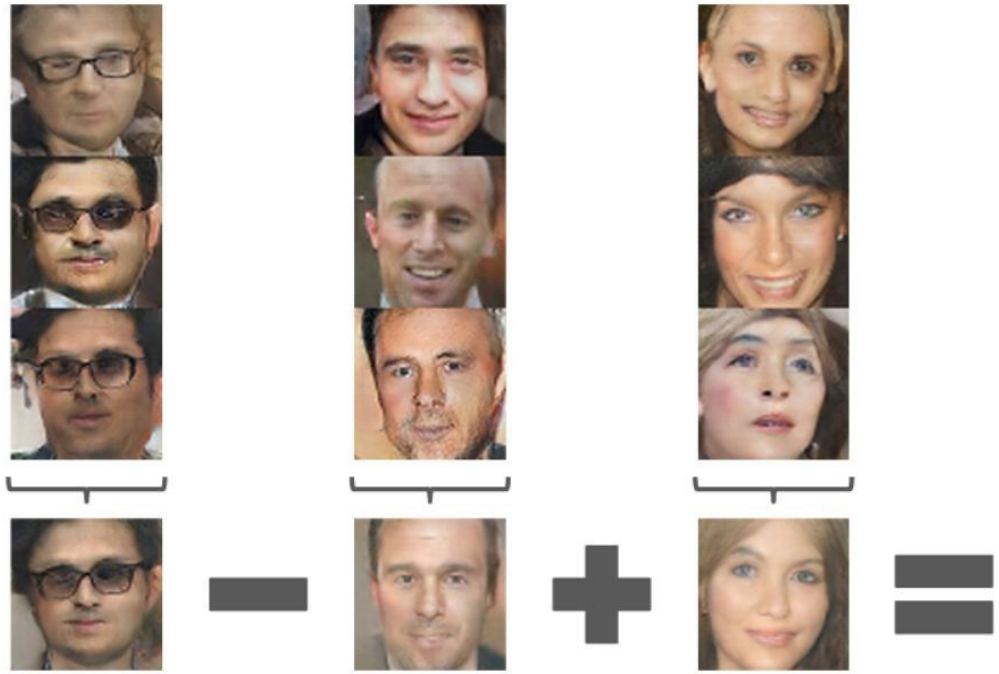
Generative Adversarial Nets: Interpretable Vector Math

Glasses man

No glasses man

No glasses woman

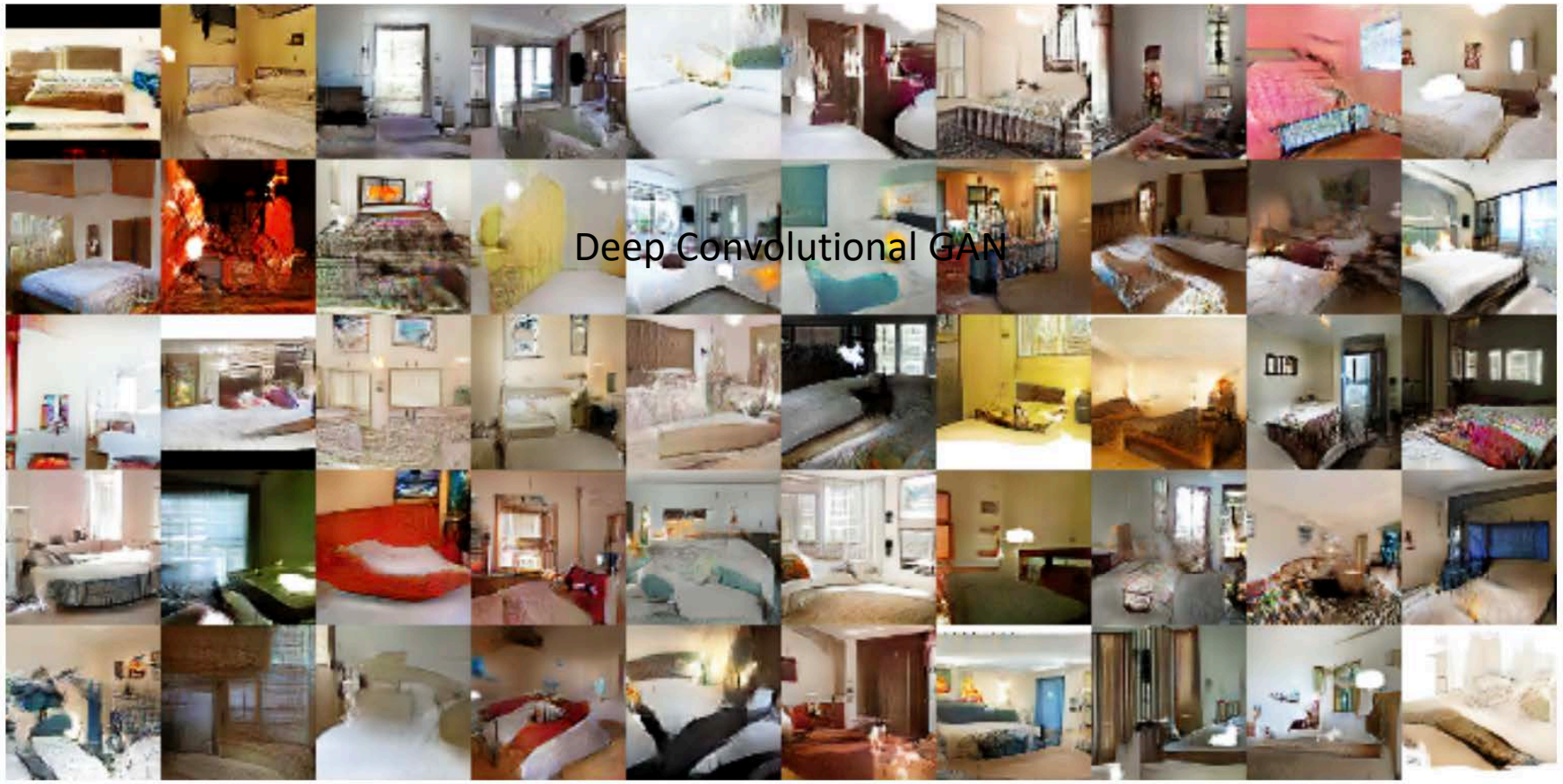
Radford et al, ICLR 2016



Woman with glasses

DCGAN: Deep Convolutional GAN

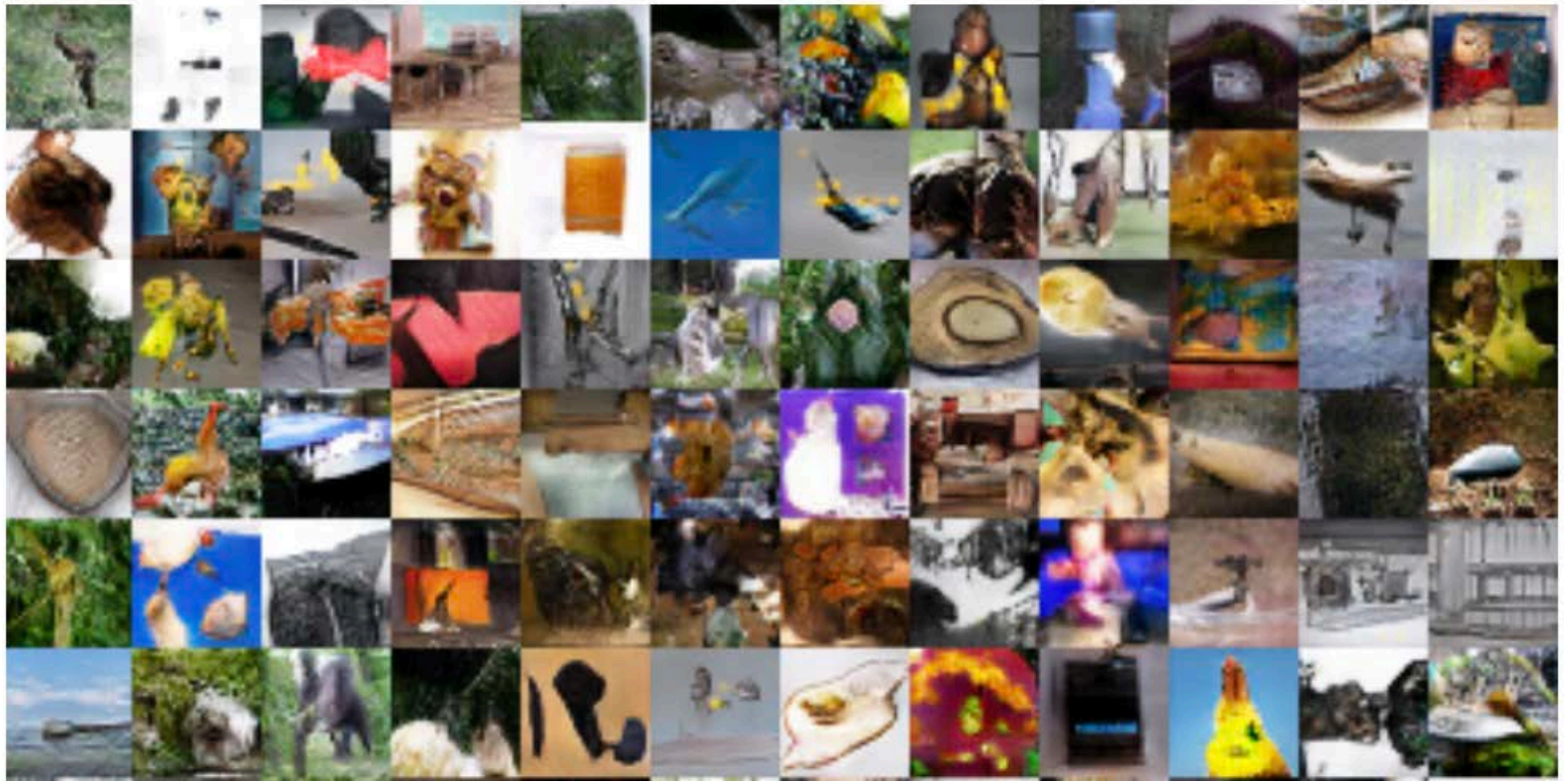
Results: LSUN Bedroom (3M Bedroom Images Only) shows very promising results



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

DCGAN

- Results:
  - ImageNet-1k (1000 class Imagenet, 32x32 Center Crop)
  - Generated samples far from impressive, which will be improved by later work



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

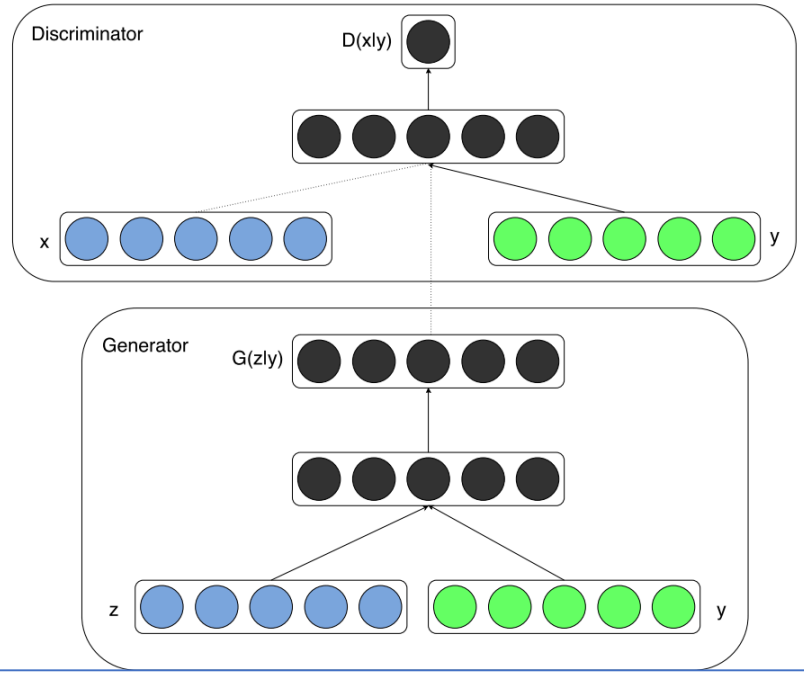
Next-Generation GANs

The GAN Zoo

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

Conditional GAN

On top of Gaussian Noise z, we can also add a label $\boldsymbol{y}$ to specify classes in the dataset that you want to sample from.



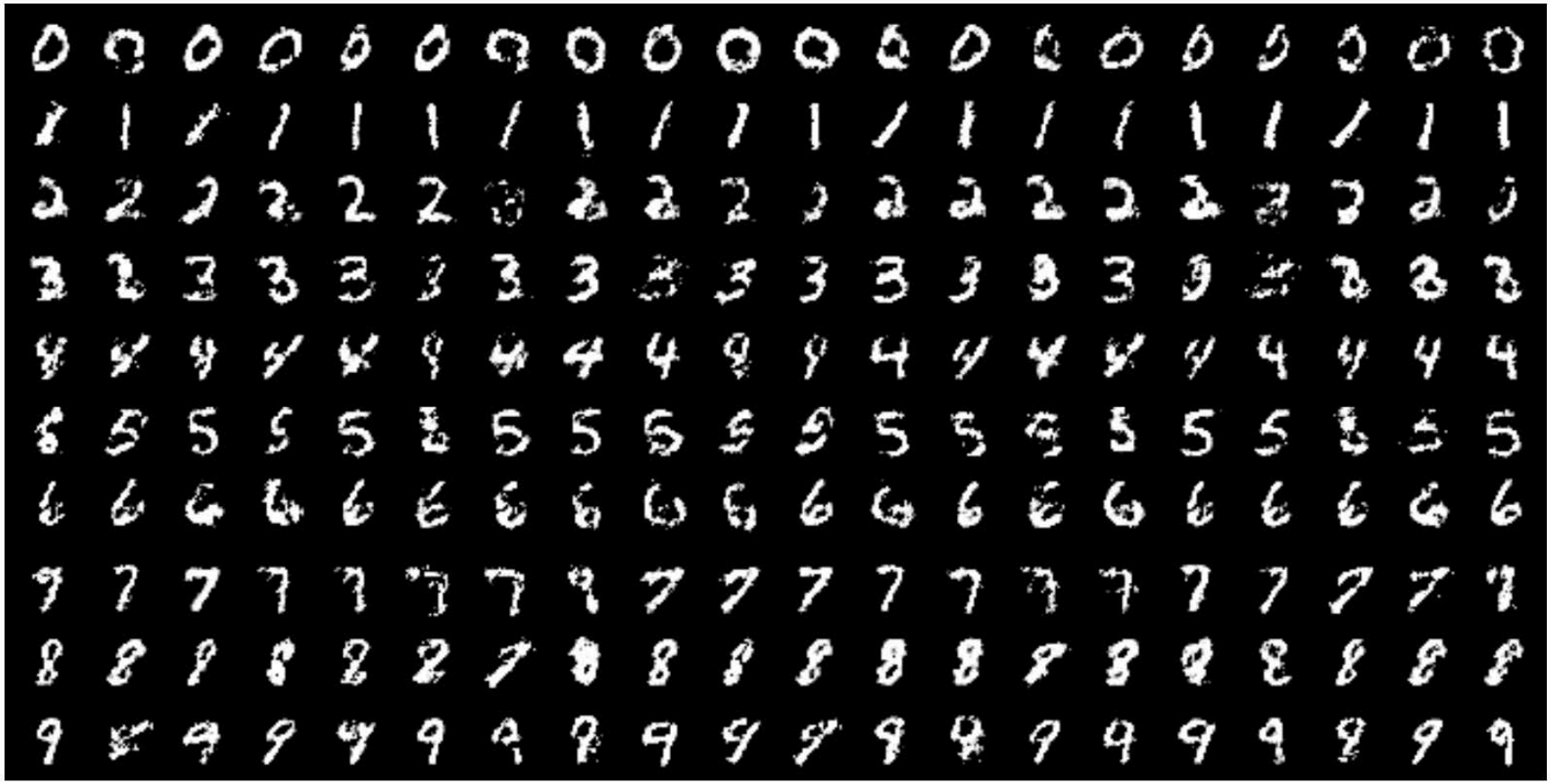
New Objective:

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

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

Conditional GAN

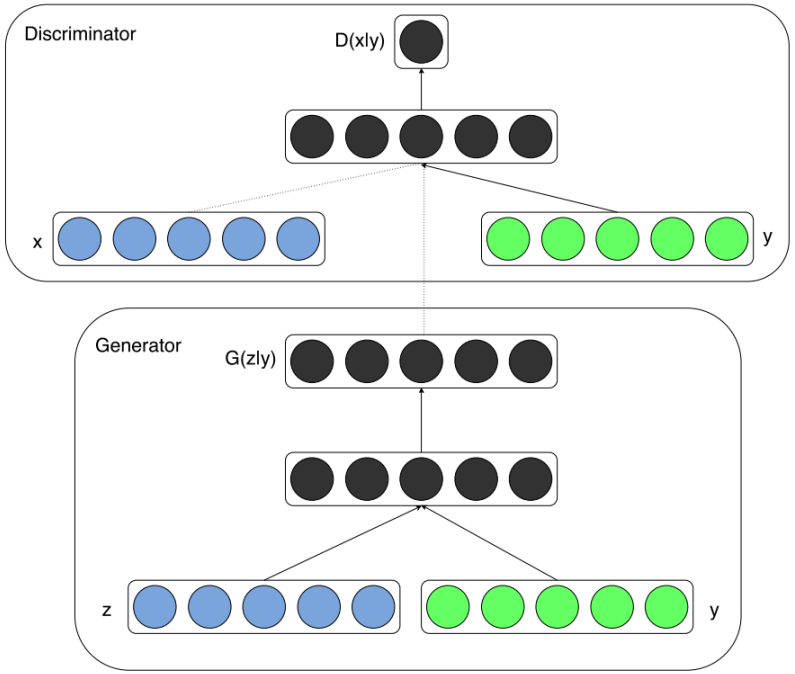
Generated samples on MNIST with class label = numbers



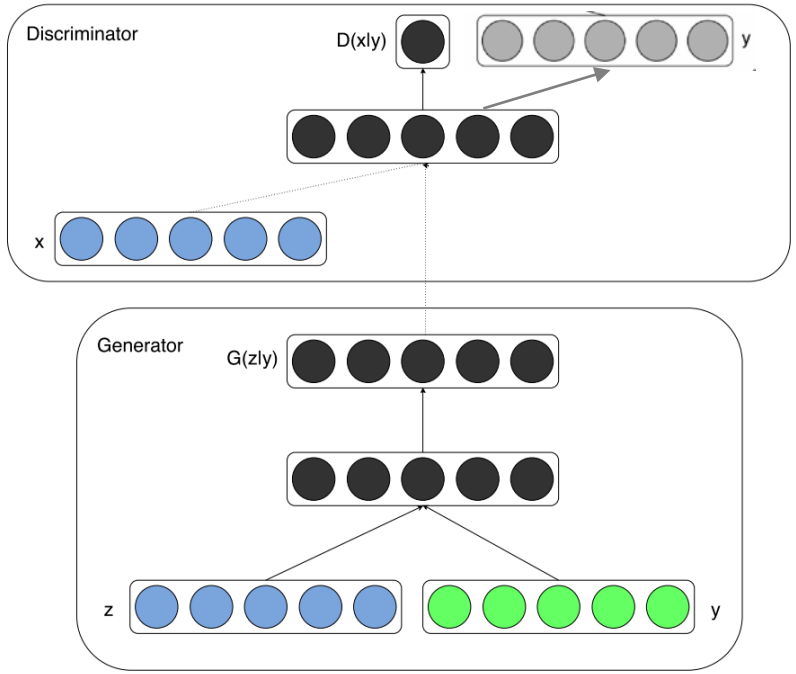
Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

ACGAN

Compared to CGAN, instead of adding labels as input to the discriminator, asks the discriminator to output a class label.



CGAN



ACGAN

Odena, A., Olah, C., & Shlens, J. (2017, August). Conditional image synthesis with auxiliary classifier gans.

ACGAN

Asks the discriminator to output a class label.

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})] \quad (2)$$

$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})] \quad (3)$$

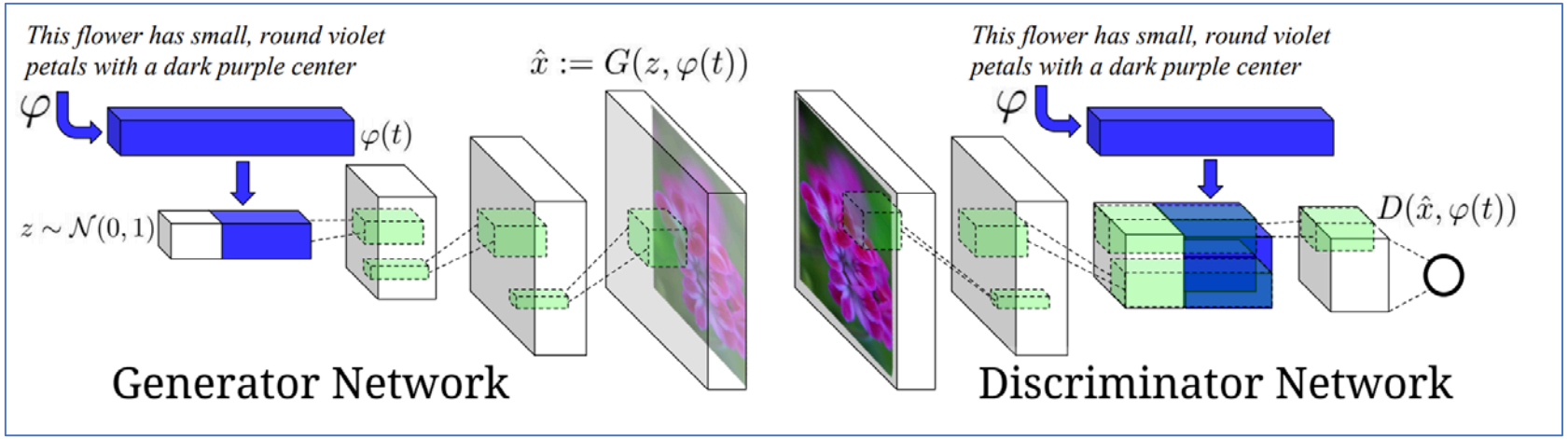
Discriminator: Maximize $L_c + L_S$

Generator: Maximize $L_c - L_S$

Odena, A., Olah, C., & Shlens, J. (2017, August). Conditional image synthesis with auxiliary classifier gans.

Text-to-Image Synthesis

- Text-to-Image Synthesis is a special case of CGAN (GAN-CLS)
  - y (label) is the sentence vector, either pre-trained or taken from a RNN encoder

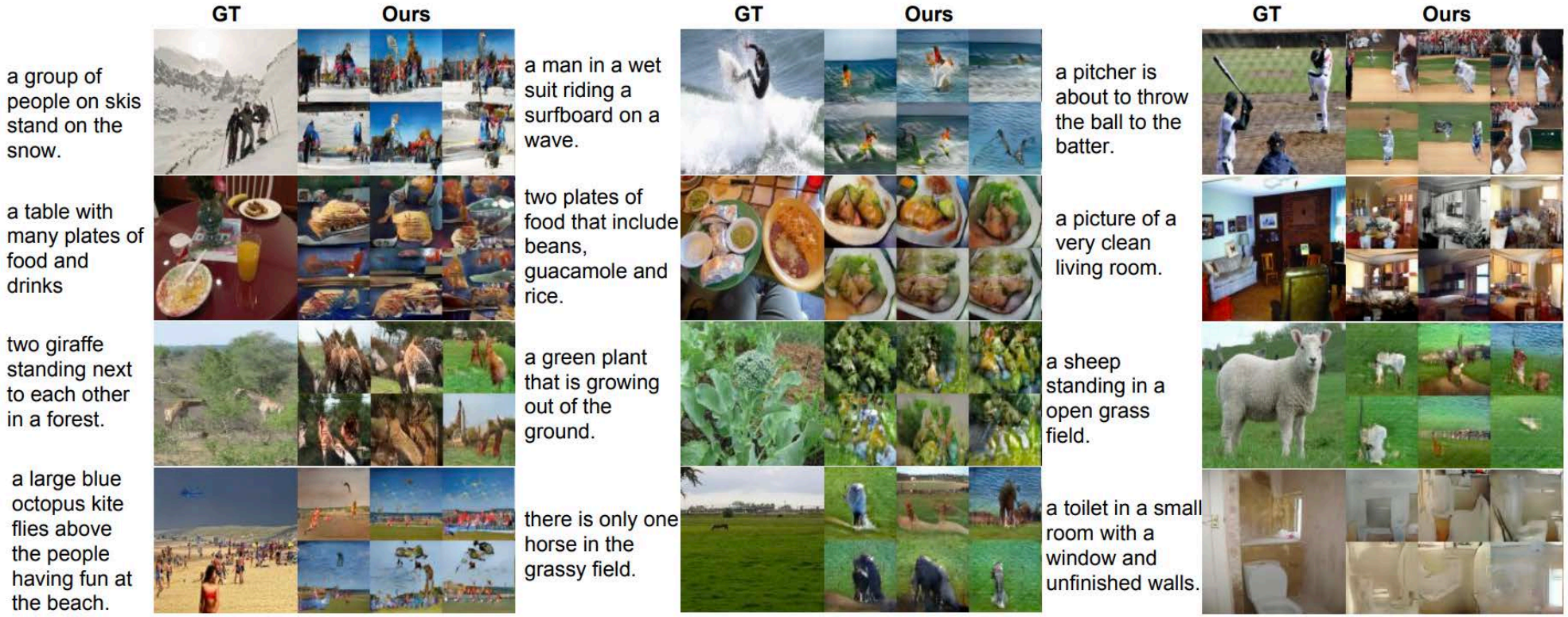


- Discriminator Loss function considers matching:

$$
\begin{aligned}
s_r &\leftarrow D(x, h) \ \{\text{real image, right text}\} \\
s_w &\leftarrow D(x, \hat{h}) \ \{\text{real image, wrong text}\} \\
s_f &\leftarrow D(\hat{x}, h) \ \{\text{fake image, right text}\} \\
\mathcal{L}_D &\leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2
\end{aligned}
$$

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis.

Text-to-Image Synthesis



Supports interpolation on y (sentence vector)

‘Blue bird with black beak’ →
‘Red bird with black beak’

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis.

Features and Challenges

- GANs can produce clear crisp results for many problems
- But they also have stability issues and are hard to train
- Problems such as “mode collapse” are frequent
  - Producing outputs with very low variability

VAEs vs GANs

VAEs

- Minimizing the KL divergence between distributions of synthetic and true data
- Uses an encoder to predict latent distributions to optimize generator
- More complex formulation
- Simpler optimization. Trains faster and more reliably
- Results are blurry

GANs

- Minimizing the Jenson-Shannon divergence between distributions of synthetic and true data
- Use a discriminator to optimize generator
- Simpler formulation
- Noisy and difficult optimization
- Sharper results

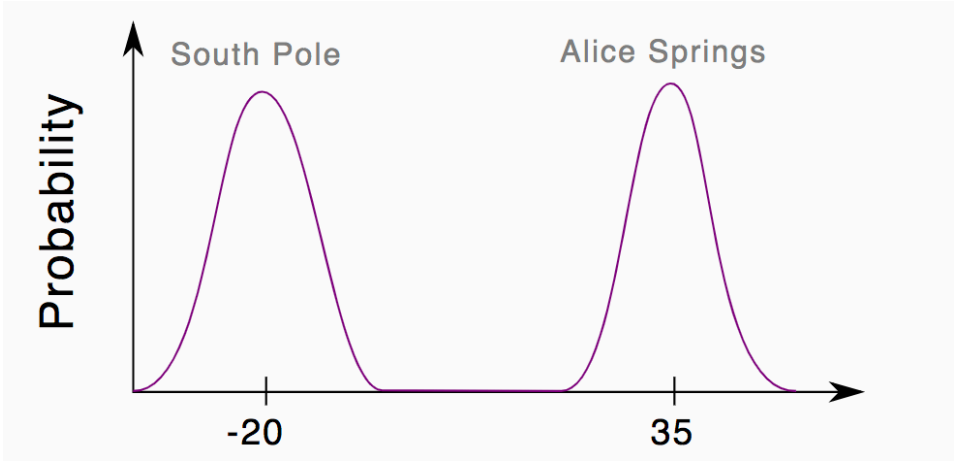
Common Problems with GANs

- Mode Collapse
  - Unrolled GAN
- Difficulty in Training
  - WGAN – Wasserstein Distance
- Difficulty in understanding global structure
  - StackGAN/BigGAN

- Difficulty in Evaluation (as with many other generative models, but particularly for GANs)
  - Inception Score
  - Frechet Inception Distance
- Unpaired Conditional Generation
  - CycleGAN/XGAN

Mode Collapse

- Mode Collapse is a common problem in GAN, it can be illustrated by:

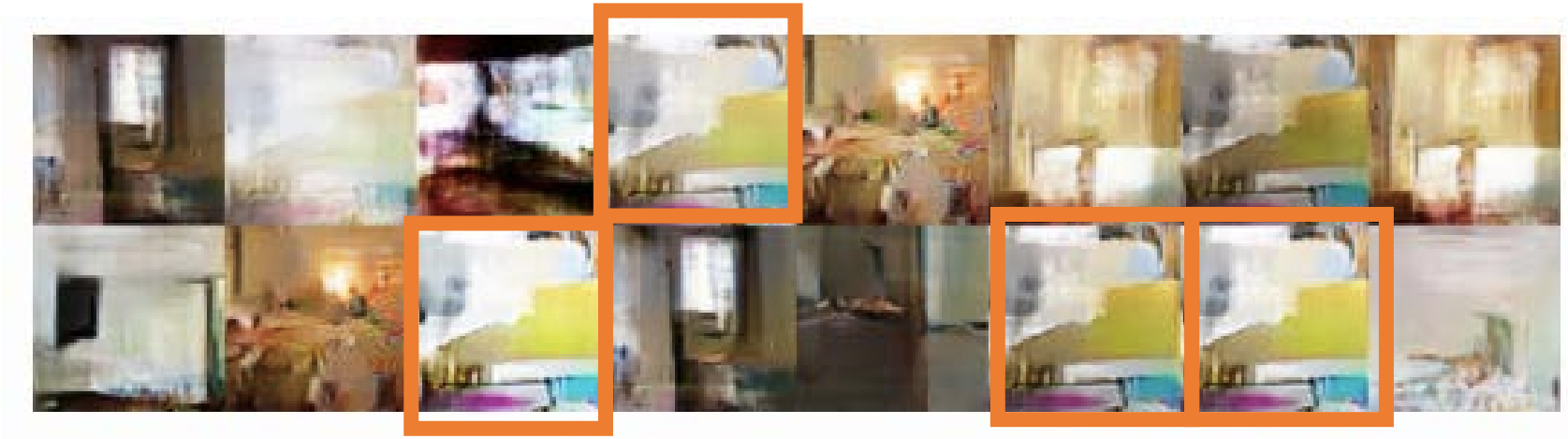


1) G Learns to generate all examples in south pole
2) D Learns all Alice springs temps. are real, but guess south pole as 0.5
3) G now switches to Alice springs temps., hence fooling D
4) D now knows south pole temps. are all real, switches the decision around.
5) Repeat

http://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/

Mode Collapse

- In practice, this happens frequently:

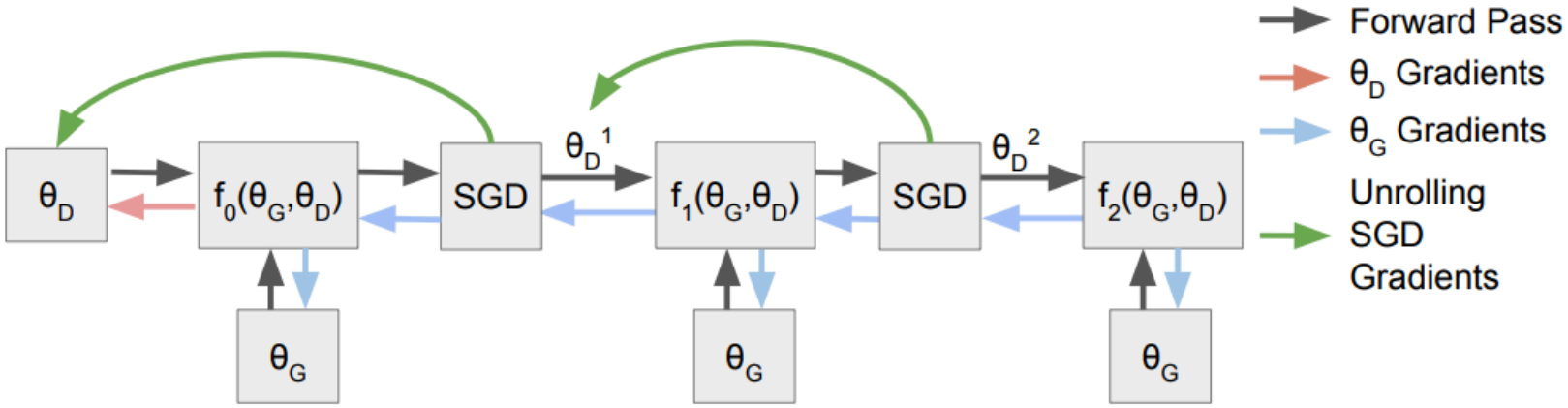


https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

Mode Collapse

Solutions:

- Experience replay: substituting an old discriminator/generator every couple of iterations
- Anticipate the gradients that the discriminator is taking, and allow generator to ‘look-ahead’ to prevent the cycle. (Unrolled GAN)



Metz, L., Poole, B., Pfau, D., & Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks.

Difficulty in Training

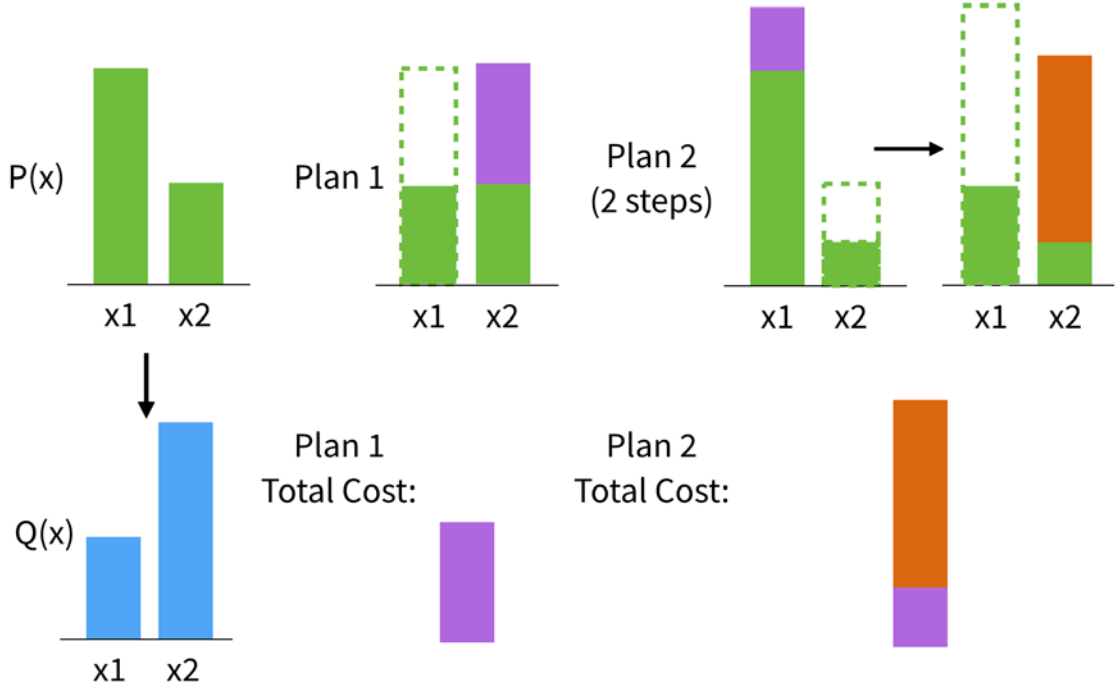
- The alternating training scheme using gradient descent might be difficult for convergence.
- WGANs: Instead of asking discriminator to make a decision, ask it to approximate a metric for computing the minimum distance to match the distributions p_{model} to p_{data}, in this case, using the Wasserstein distance.
- Intuition of Wasserstein Distance (Earth-Mover/EM distance):
  - Assume both distributions as piling up dirt over a region D
  - Earth Mover's distance is the minimum cost of turning one pile into another.

Intuition of Wasserstein Distance

Goal: Move mass from P(x) to match Q(x)

Cost: Mass moved in total × Distance to move (1 in our case)

P(x), Q(x): Discrete Distributions



However: There can be many plans (e.g. Plan 2) with higher costs, especially in continuous distributions

Wasserstein Distance: **Minimum** cost to match the distributions (**Plan 1**)

Wasserstein GAN

- For an image generation task, Wasserstein distance measures the minimum cost for matching the distributions of $g_\theta(z)$ and x
- The discriminator now attempts find a good 'critic' function f_w (Intuition: realism score of the generated samples) that quantifies distance between the distributions of generated images and real images
- Critic f_w Objective (Maximize):

$$\left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$$

- Generator g_θ attempts to maximize the realism score of the synthetic examples:

$$\frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$$

Wasserstein GAN

To build a W-GAN:

1. Remove the Sigmoid activation at the D (now 'critic' f_w) output layer, so it now outputs a linear score
2. Clip the weights to a fixed range after each gradient update step, to enforce the Lipschitz constraint (see paper for details)
3. Use new learning objectives:

Ascend gradients for w, parameters of the critic:

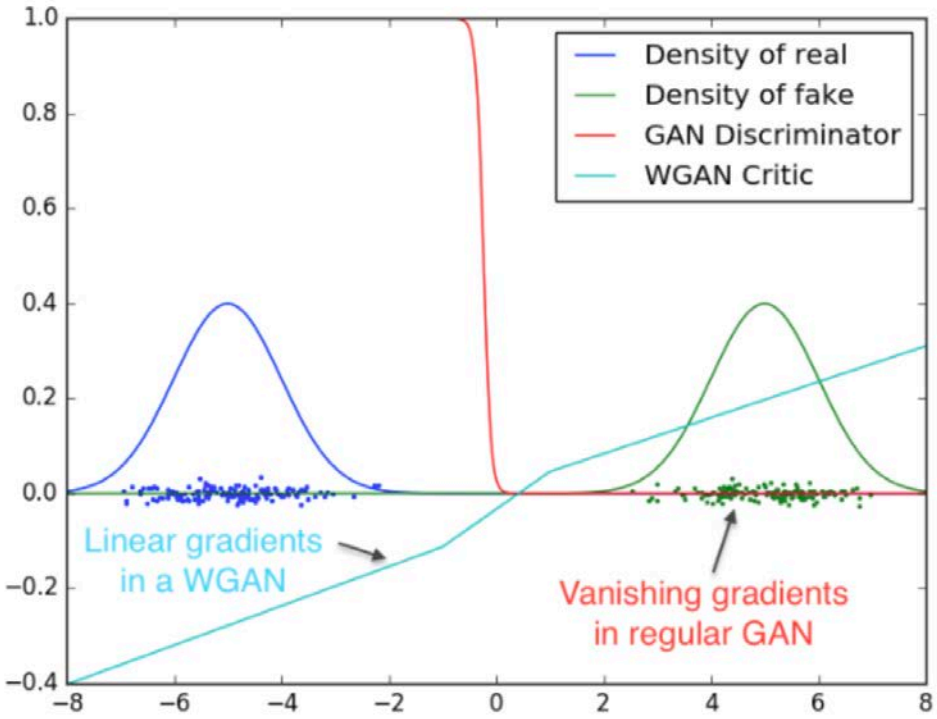
$$\nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$$

Ascend gradients for θ , parameters of the generator:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$$

Advantages of Wasserstein GAN

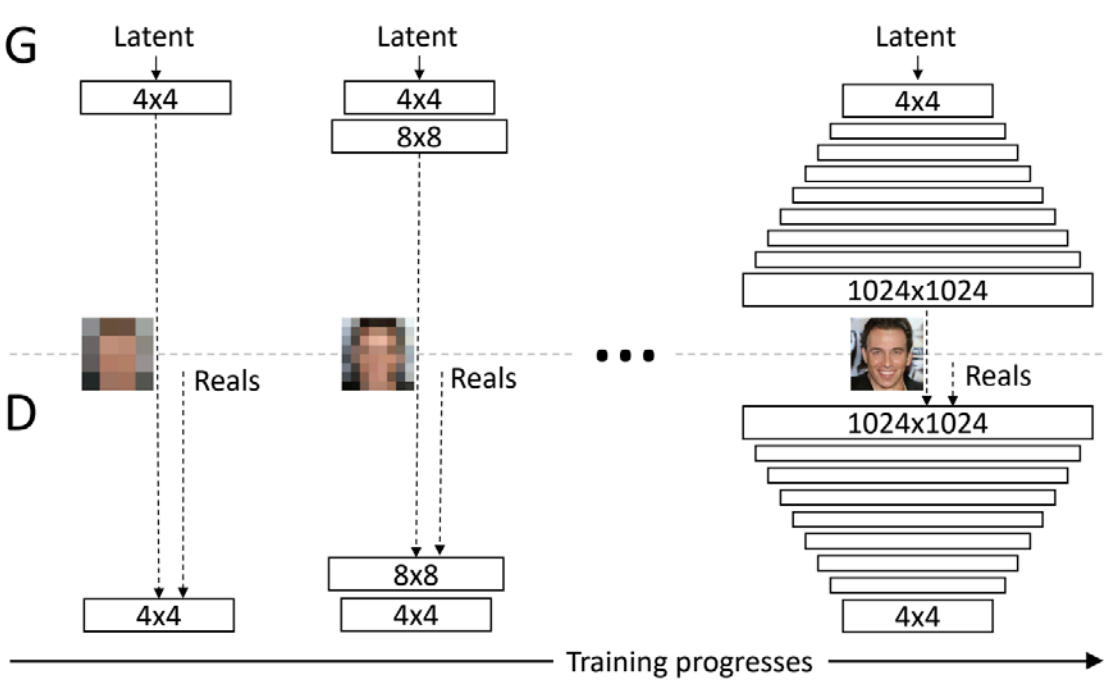
- Can keep learning until convergence: **gradients everywhere**
- More stable training/consistent convergence
- Train Critic till optimality: Avoids **Mode Collapse**
- A lot of theory behind it!



Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan.

Difficulty in Understanding Global Structure

- Progressive GANs
  - Incrementally add layers to GANs, each in charge of a certain resolution of output generations, and progressively learn the up-sampling of the images.
  - All previous layers remain trainable.



Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. ≈

Progressive GANs

Progressive GANs Results

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation.

Common Problems with GANs

- Mode Collapse
  - AdaGAN, Unrolled GAN
- Difficulty in Training
  - WGAN – Wasserstein Distance
- Difficulty in understanding global structure
  - StackGAN/BigGAN

- Difficulty in Evaluation (as with many other generative models, but particularly for GANs)
  - Inception Score
  - Frechet Inception Distance
- Unpaired Conditional Generation
  - CycleGAN/XGAN

Evaluating GANs

- Inception Score:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right),$$

  - Use the Inception v3 network to compute both $\mathrm{P}(y|x)$ and $\mathrm{P}(y)$,
  - $\boldsymbol{P}(\boldsymbol{y}|\boldsymbol{x})$ is the classification result of a single sample x
  - $P(y)$ is estimated by summing all classifications of $\boldsymbol{x}$
  - GANs with high Inception scores would have low entropy for $\mathrm{P}(y|x)$, and high entropy for $P(y)$
  - $P(y)$ stands for **diversity**, $\mathrm{P}(y|x)$ stands **for clearness**, meaning Inception network should be confident that there should be an object in the object.
  - A High KL divergence means the network can generate high-quality individual examples, and diverse overall examples.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans.

Evaluating GANs

- Inception Score:

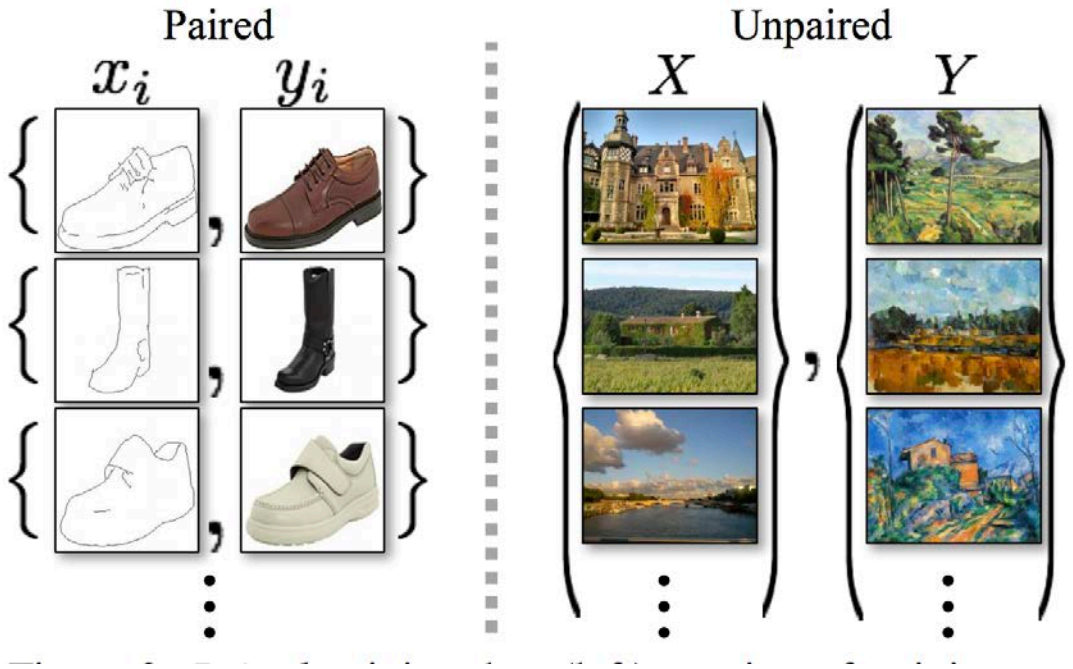
$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x} \sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right),$$

  - Works mostly for natural images
  - Large dataset of images + dense labels for a well-designed network to pre-train on.
  - Correlates well to human judgment: Inception Model

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

Unpaired Conditional Image Generation

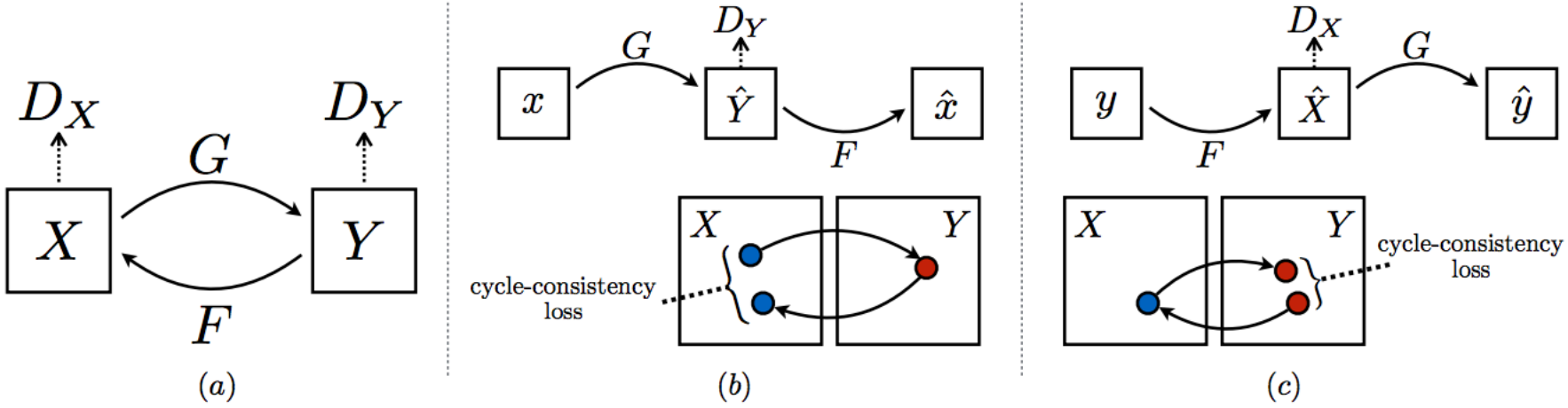
- So far, we have looked at paired image data for translating between image <-> image / text <-> image
- What if we only have **two sets of data that we know are in separate domains, but are not paired against each other** (e.g. artwork and photos)



Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

CycleGAN

- 2 generators + 2 discriminators, 1 pair for each domain
- Instead of having to use pairing, only relies on real images on the other domain for realism
- Use cycle-consistency loss for pairing instead (L1 distance)



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

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

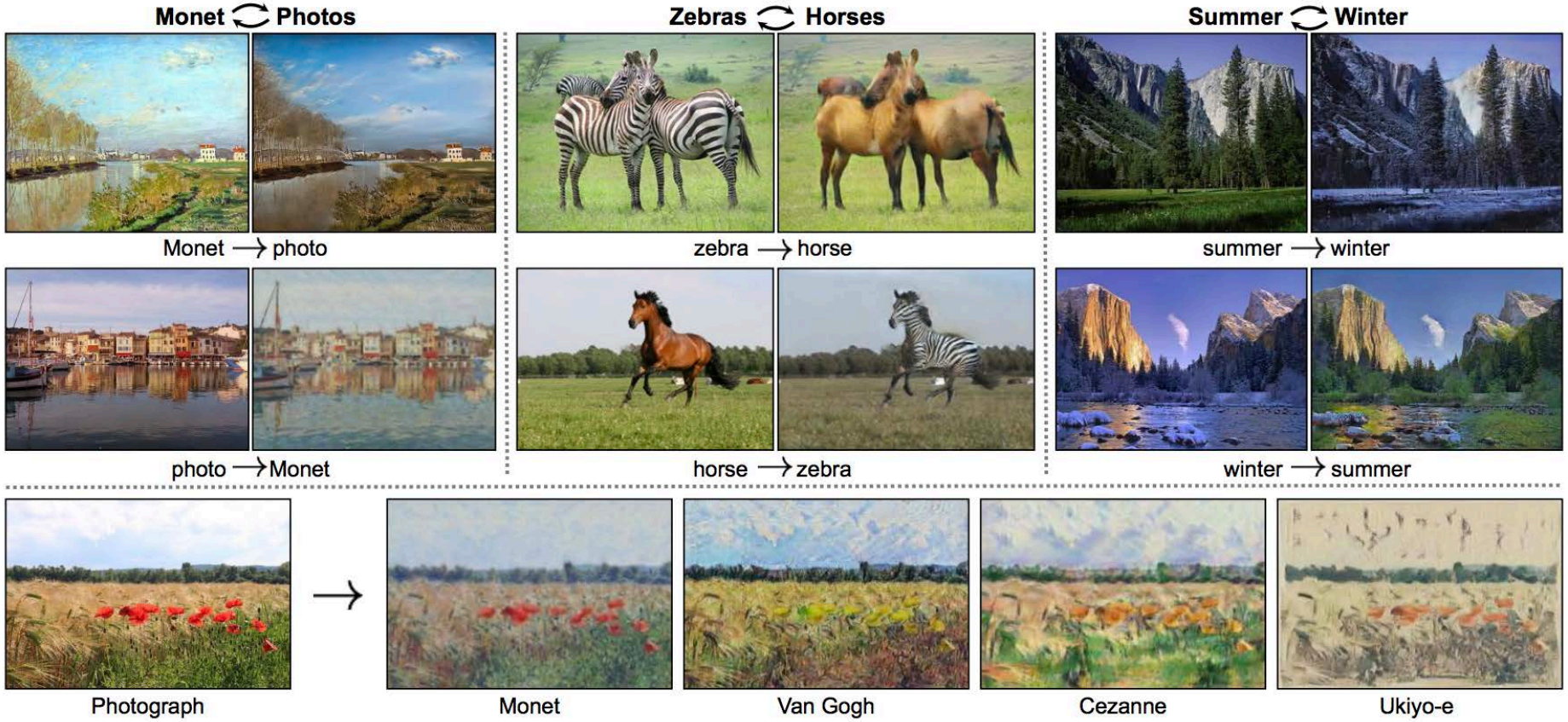
CycleGAN

- Two generator models:
  - generator (G) for generating images for Y
  - generator (F) for generating images for X

  ✓X-> G -> Y

  ✓Y -> F -> X
- Each generator has a corresponding discriminator model
- Loss:
  - Adversarial Loss
  - Cycle consistency loss
  - Identity Loss

CycleGAN



Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

GANs Summary

- Model generates samples **but does not estimate density** p(x) at image x.
- **2-player Mini-max game** between the discriminator and generator
- Original GAN minimizes JS-divergence
- Multiple **architectural improvements** for Generator/Discriminator itself: DCGAN/Progressive GAN allowed GANs to generate very impressive high-resolution
- Multiple **training scheme improvements:** WGAN for stable training
- Training can be unstable, many empirical ‘tricks’ might be required