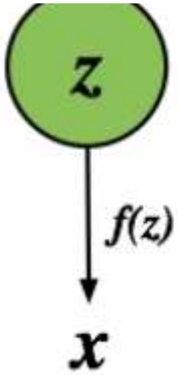
Lecture 15: Deep Generative Models IV: GANs

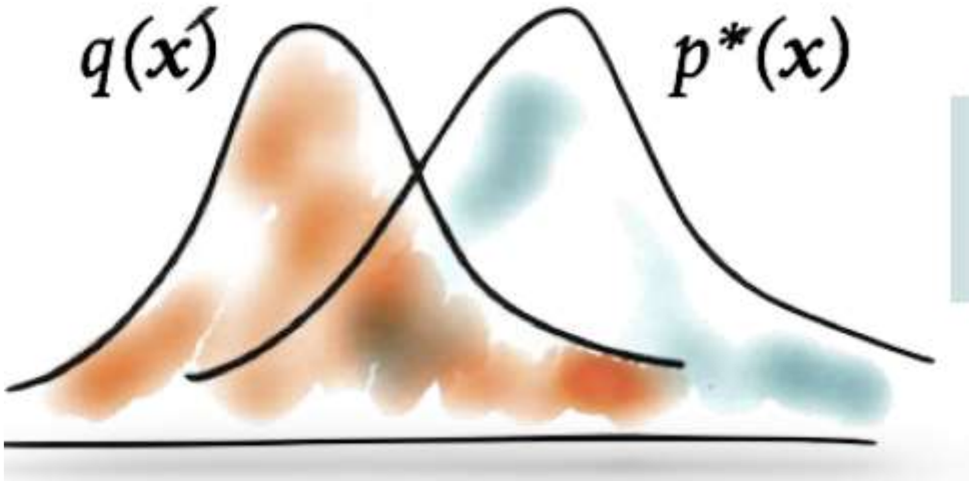
Lan Xu
SIST, ShanghaiTech
Fall, 2023

Review: Learning by comparison

- Basic idea

For some models, we only have access to an unnormalised probability, partial knowledge of the distribution, or a simulator of data.

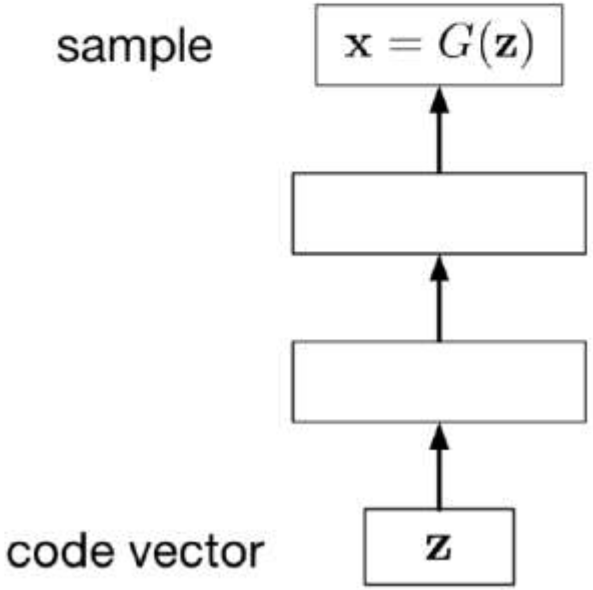




We compare the estimated distribution q(x) to the true distribution p*(x) using samples.

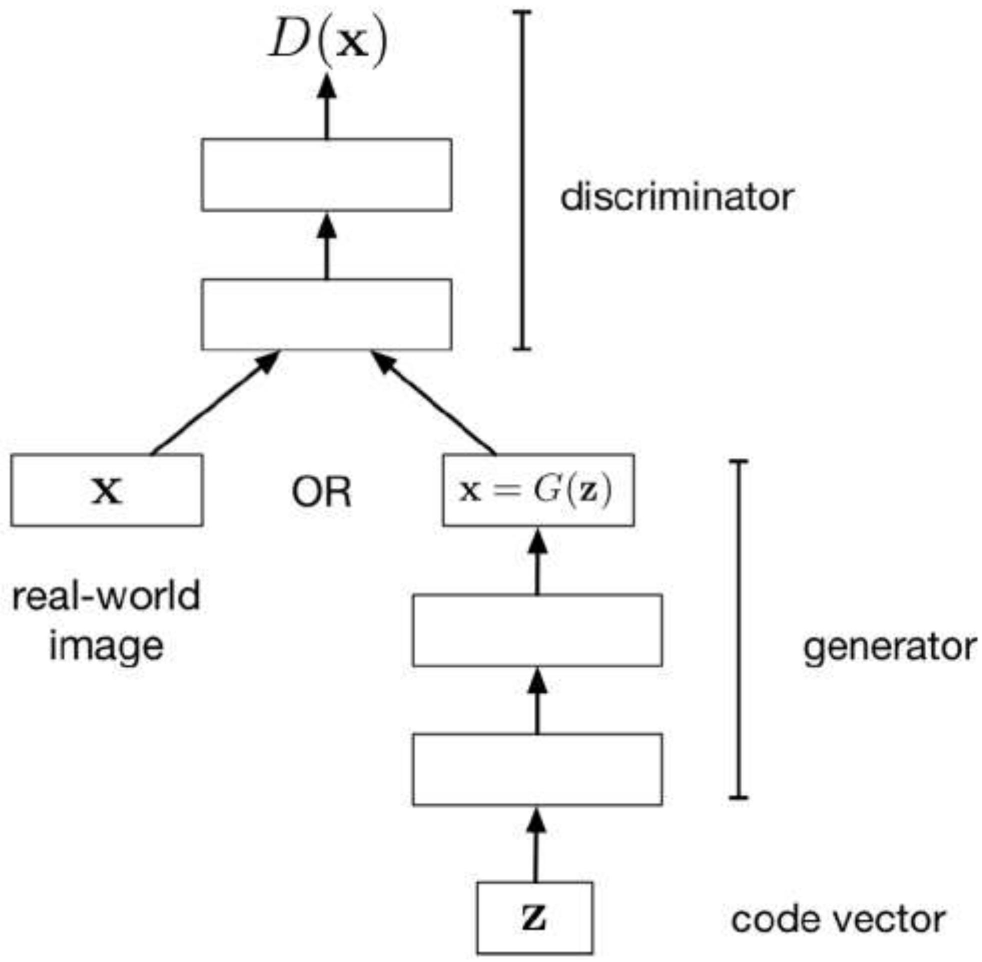
Review: Implicit Generative Models

- Implicitly define a probability distribution
- Start by sampling the code vector z from a fixed, simple distribution
- A generator network computes a differentiable function G mapping z to an x in data space



Review: Adversarial Learning

- Adversarial loss



Review: Two-player game

- Minimax formulation
  - The generator and discriminator are playing a zero-sum game against each other

$$\min_{G} \max_{D} \mathcal{J}_D$$

  - Using parametric models

Minimax objective function:

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

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Review: Learning procedure

- Minimax objective function

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

Alternate between:

1. **Gradient ascent** on discriminator

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

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Review: Theoretical property

- Adversarial loss for the optimality

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$$

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

$$= \min_G \left(E_{x\sim p_{data}}\left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + E_{x\sim p_G}\left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)}\right] - \log 4\right)$$

$$= \min_G \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2}\right) + KL\left(p_G, \frac{p_{data} + p_G}{2}\right) - \log 4\right)$$

$$= \min_G (2 * JSD(p_{data}, p_G) - \log 4)$$

Jensen-Shannon Divergence:

$$JSD(p,q) = \frac{1}{2} KL\left(p, \frac{p+q}{2}\right) + \frac{1}{2} KL\left(q, \frac{p+q}{2}\right)$$

Kullback-Leibler Divergence:

$$KL(p,q) = E_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$$

Review: Theoretical property

- Adversarial loss for the optimality

$$\min_G \max_D \left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

$$= \min_G (2 * JSD(p_{data}, p_G) - \log 4)$$

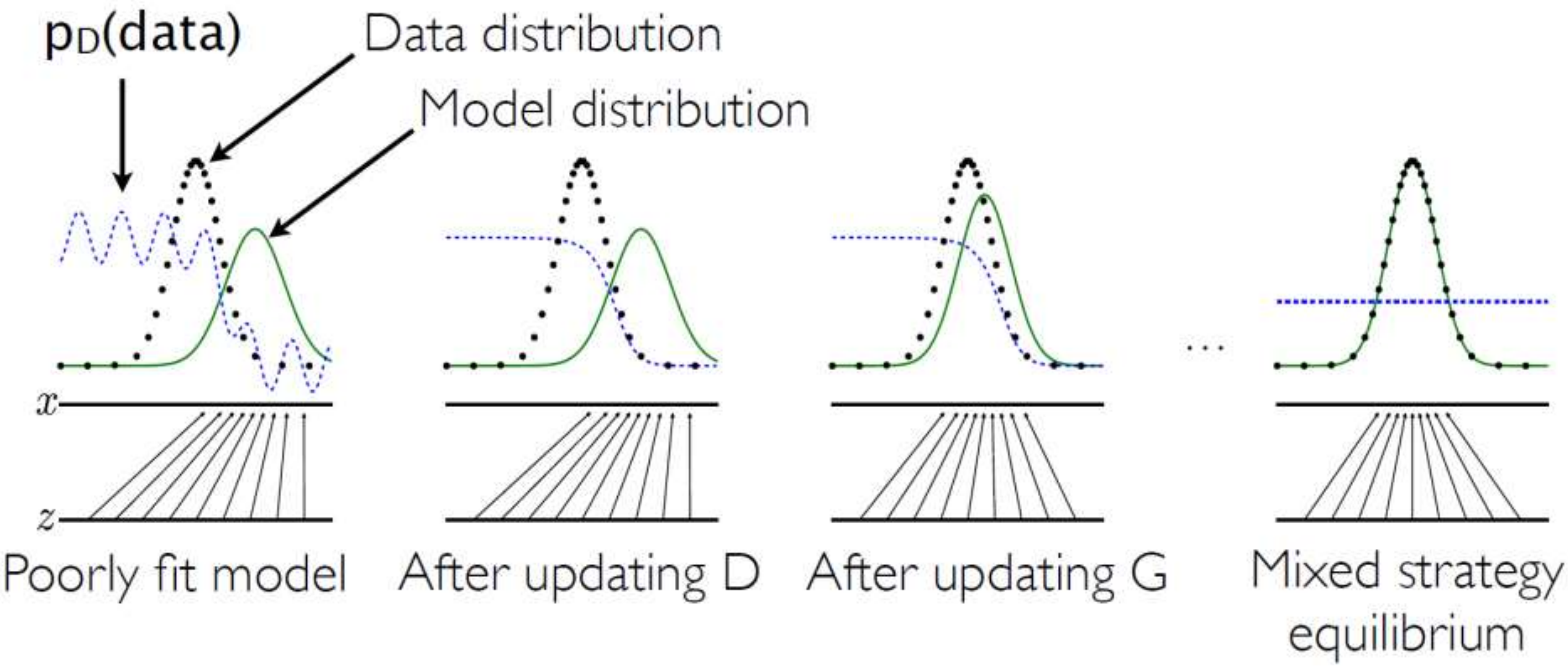
- Summary: the global minimum of the minimax game happens when:

1. $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$ (Optimal discriminator for any G)
2. $p_G(x) = p_{data}(x)$ (Optimal generator for optimal D)

- Caveats:

1. G and D are neural nets → depend on the network optimization!
2. ``Theoretical'' convergence to the optimal solution,

Training GANs



Training GANs

- Since GANs were introduced in 2014, there have been hundreds of papers introducing various architectures and training methods
- GAN Zoo: https://github.com/hindupuravinash/the-gan-zoo
- In general, training a GAN is tricky and unstable
- Many tricks:
  - S. Chintala, How to train a GAN, ICCV 2017 tutorial
  - https://github.com/soumith/talks/blob/master/2017-ICCV_Venice/How_To_Train_a_GAN.pdf

Generated Samples

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

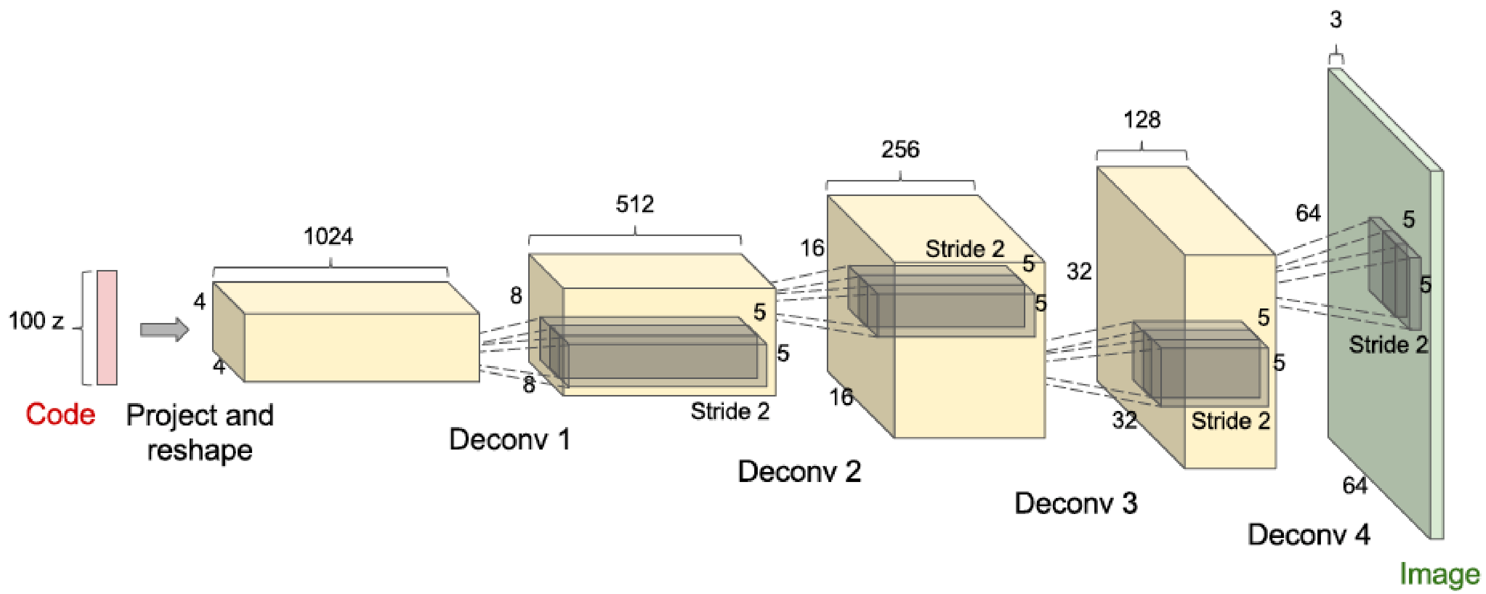
Generated Samples

Objects:

DCGAN

- GAN with convolutional architetures
  - Generator is an upsampling convolutional network
  - Discriminator is a convolutional network

Deep Convolutional GAN [Radford et al., 2015]



Generated Samples

Generated Bad Samples

- Problems with Global Structure and Counting

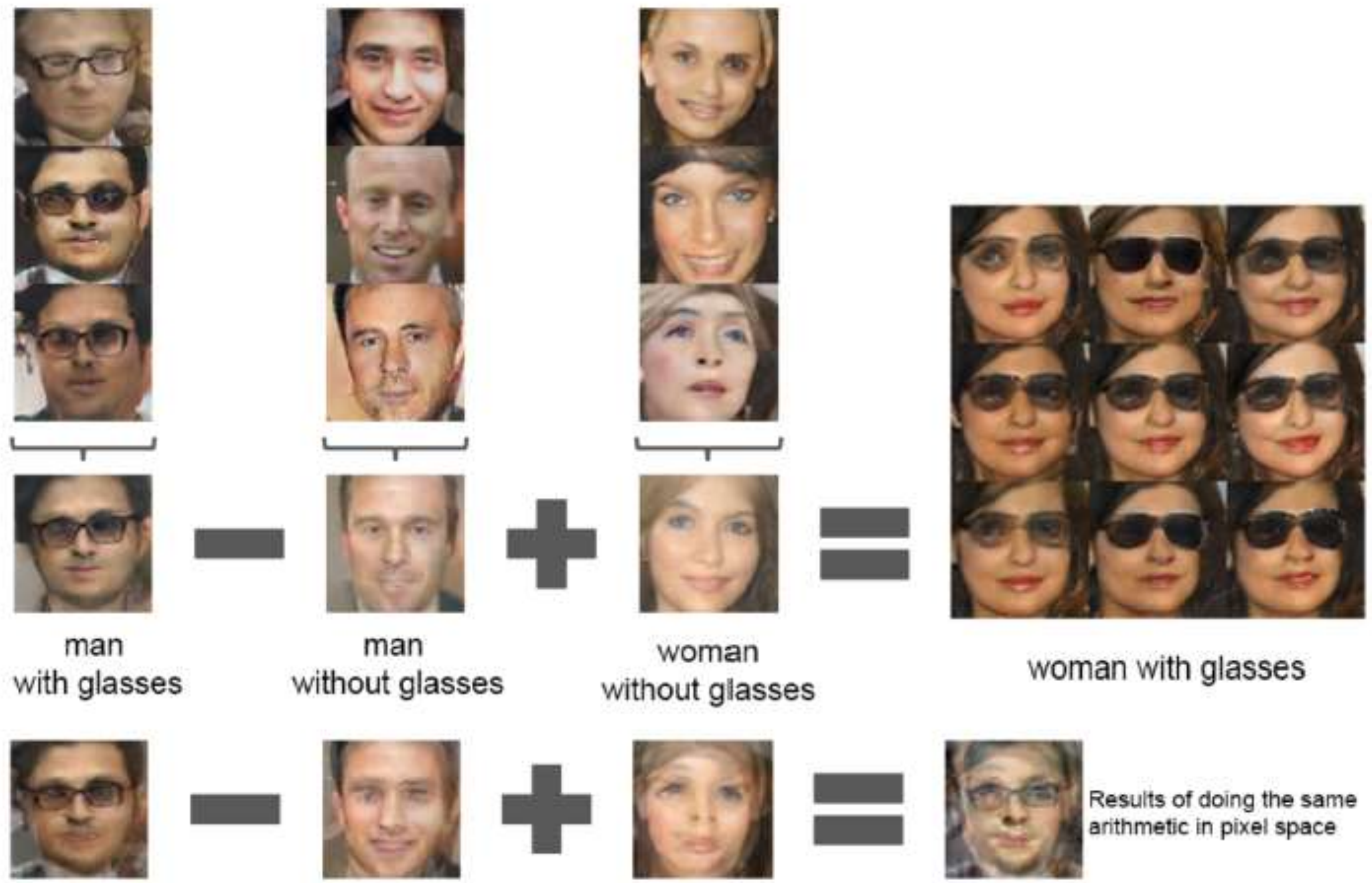
Walk Around Data Manifold

Interpolating between random points in laten space

Radford et al, ICLR 2016

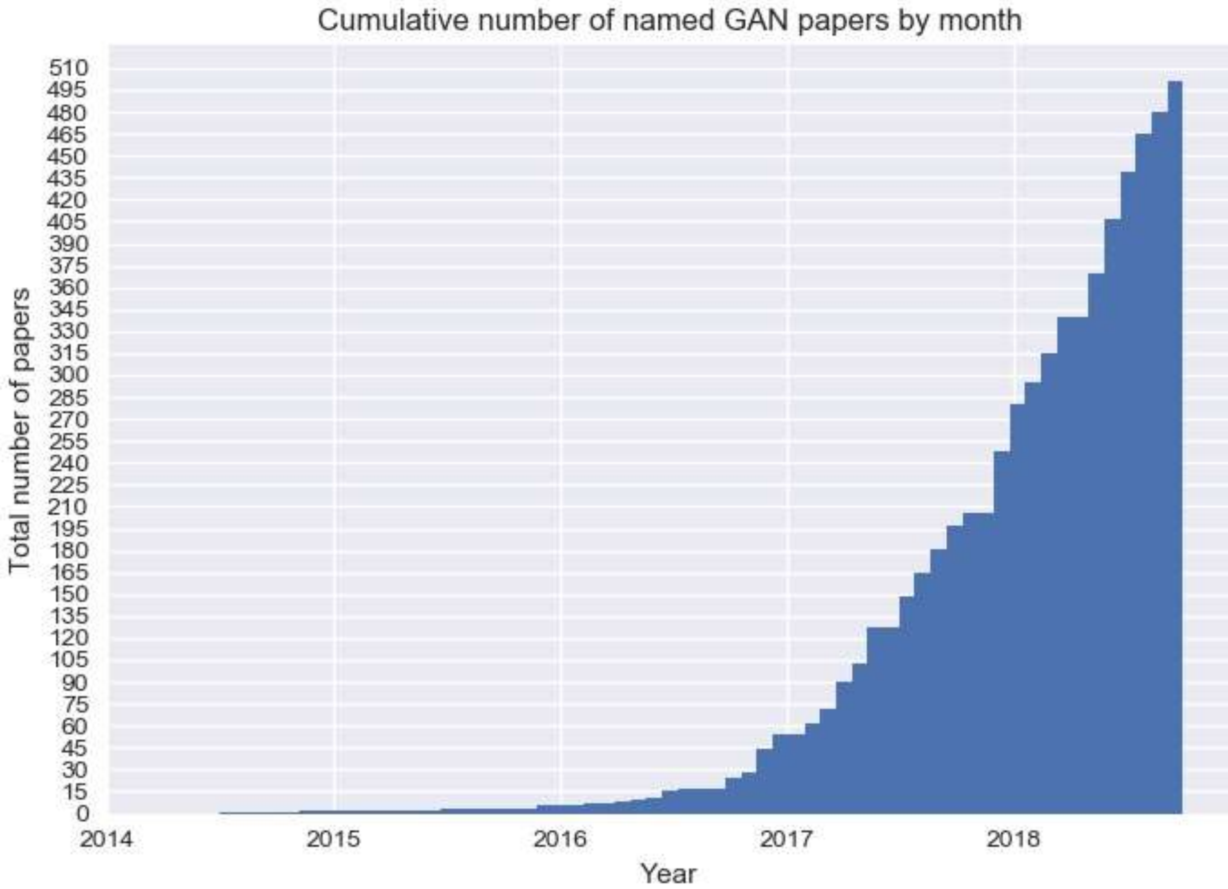
Walk Around Data Manifold

- Vector Arithmetic



Explosion of GANs

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



Evaluation metrics

- What makes a good generative model?
  - Each generated sample is indistinguishable from a real sample

  - Generated samples should have variety

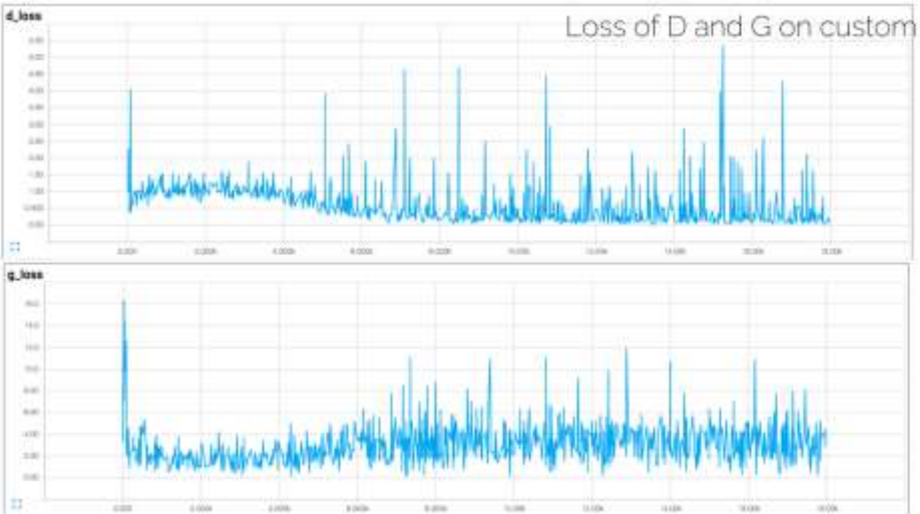
Images from Karras et al., 2017

Evaluation metrics

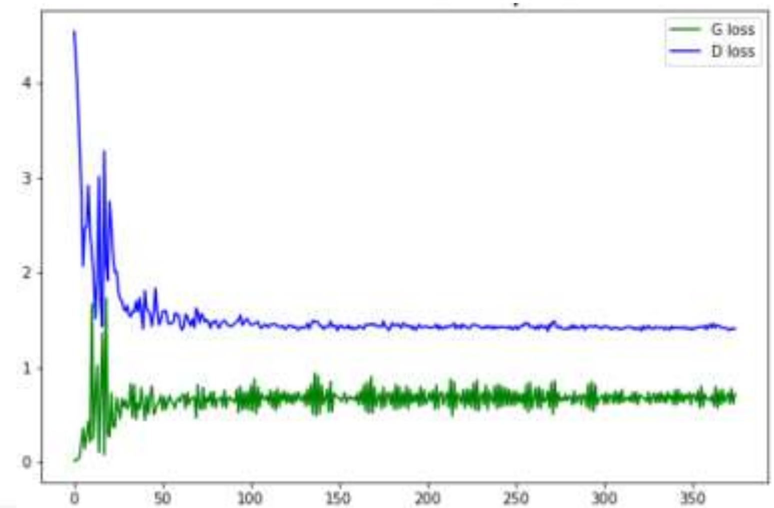
- How to evaluate the generated samples?
  - Cannot rely on the models' loss :-(
  - Human evaluation :-/
  - Use a pre-trained model :-)

Evaluation metrics

- “Good” Training Curves



Loss of D and G on custom dataset

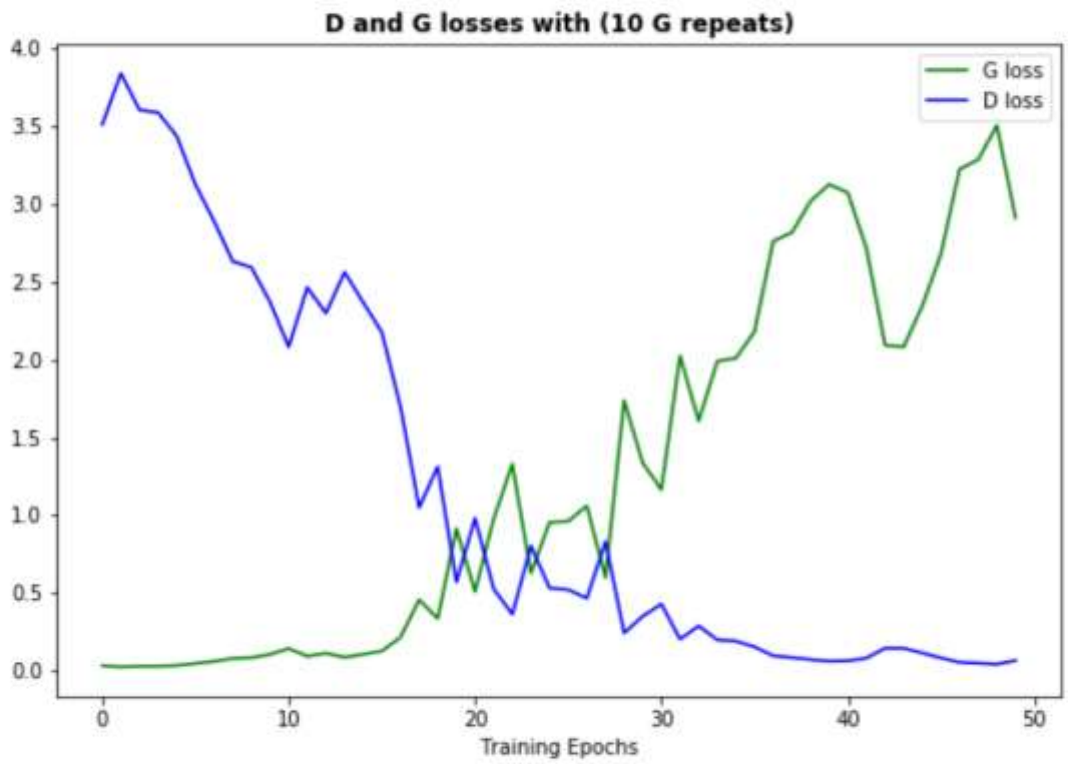


Generator's Error through Time

Discriminator's Error through Time

Evaluation metrics

- "Bad" Training Curves



Evaluation metrics

- Inception Score (IS) [Salimans et al., 2016]
  - Inception model p trained on ImageNet
  - Given generated image x, assigned the label y by model p

$$p(y|\boldsymbol{x}) \Rightarrow \text{low entropy (one class)}$$

  - The distribution over all generated images should be spread

$$\int p(y|\boldsymbol{x} = G(z))dz \Rightarrow \text{high entropy (many classes)}$$

  - Combining the above, we get the final metric:

$$\exp(\mathbb{E}_{\boldsymbol{x}} \mathrm{KL}(p(y|\boldsymbol{x}) || p(y)))$$

Evaluation metrics

- Frechet Inception Distance (FID) [Heusel et al. 2017]
  - Calculates the distance between real and fake data (lower the better)
  - Uses the embeddings of the real and fake data from the last pooling layer of Inception v3.
  - Converts the embeddings into continuous distributions and uses the *mean* and *covariance* of each to calculate their distance.

$$\mathrm{FID}(r,g) = ||\mu_r - \mu_g||_2^2 + \mathrm{Tr}(cov(r) + cov(g) - 2(cov(r)cov(g))^{\frac{1}{2}})$$

Evaluation metrics

- Comparisons

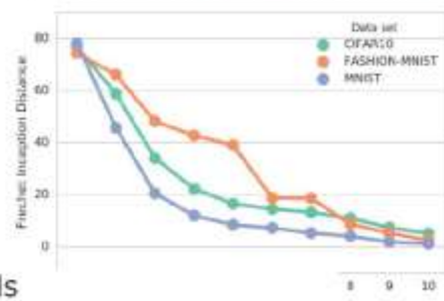
▪ IS vs FID

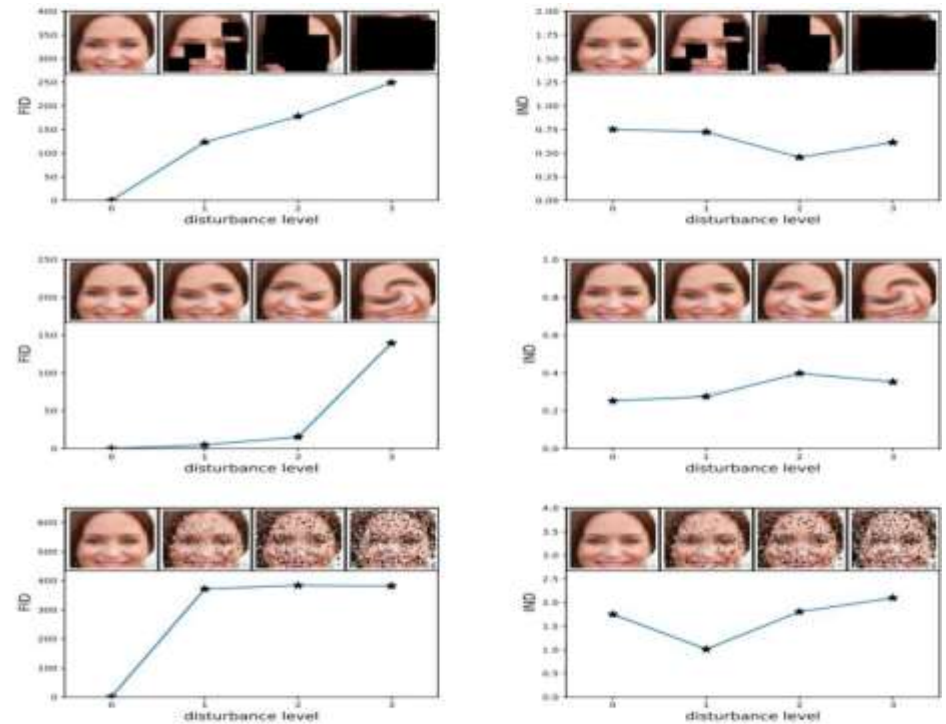
✓ FID considers the real dataset

✓ FID requires less sampling (faster) (~10k instead of 50k in IS)

✓ FID more robust to noise and human judgement

✓ FID also sensitive to mode collapse





FID (lower is better) IS (higher is better)

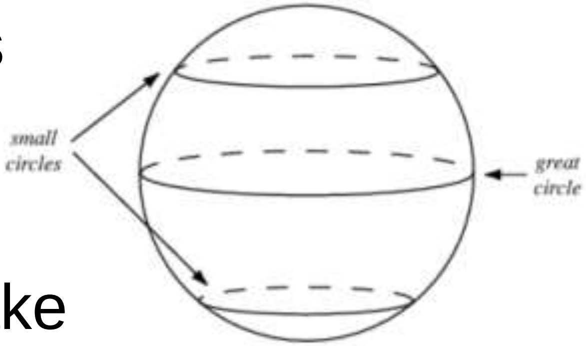
Images from Lucic et al., 2017 and Heusel et al., 2017

The GAN Zoo

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

  - 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
  - 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
  - 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
  - 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
  - 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
  - ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
  - ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
  - AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
  - acGAN - Face Aging With Conditional Generative Adversarial Networks
  - ACGAN - Coverless Information Hiding Based on Generative adversarial networks
  - acGAN - On-line Adaptative Curriculum Learning for GANs
  - ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
  - AdaGAN - AdaGAN: Boosting Generative Models
  - Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
  - AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
  - AdvGAN - Generating adversarial examples with adversarial networks
  - AE-GAN - AE-GAN: adversarial eliminating with GAN
  - AE-OT - Latent Space Optimal Transport for Generative Models
  - AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
  - AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
  - AffGAN - Amortised MAP Inference for Image Super-resolution
  - AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization

GAN Hacks



- https://github.com/soumith/ganhacks
- Normalize the inputs: [-1, 1], Tanh
- Use a spherical z; Use Batch Norm
- Different mini-batches for real and fake
- SGD for discriminator; ADAM for generator
- One-sided Label Smoothing

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

Some value smaller than 1; e.g.,0.9

- Avoid Sparse Gradients: no ReLU and MaxPooling

  LeakyReLU → good in both G and D

  Downsample → use average pool, conv+stride

  Upsample → deconv+stride, PixelShuffle

Why are GANs different

- GAN optimization is fundamentally different from other neural networks
  - Gradient descent is relatively well established
  - Loss functions don't change much
  - Most deep learning research has focused on new components to use within the standard single-player framework (dropout, batchnorm, relu, etc.)
- GANs are an area of research where the objectives and descent methods are still in flux

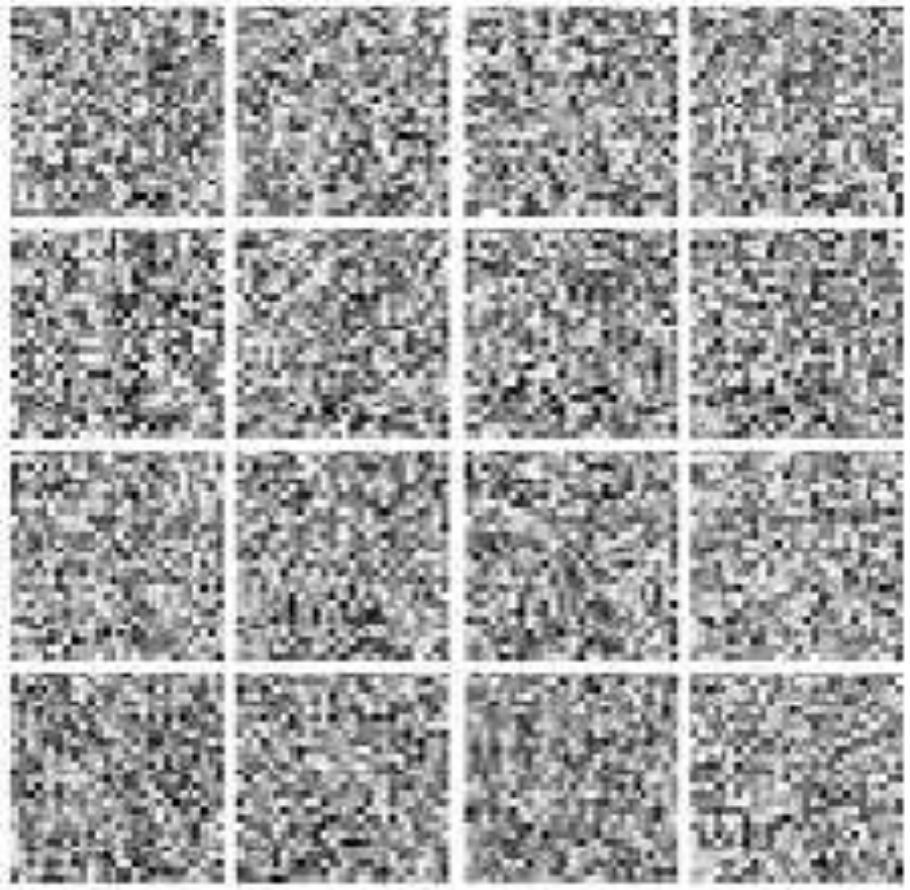
Potential causes of instability

- Several theories on why GANs are hard to train.
- Main contributing factors:
  - Adversarial optimization is a more general, harder problem than single-player optimization.
  - Two player games do not always converge using gradient descent.
  - There is a stationary point but no guarantee of reaching it.
  - Simultaneous updates require a careful balance between the two players.
  - Generated points tend to "herd" to probable regions, causing "mode collapse”.
  - Discriminator is highly nonlinear, gradient tends to be noisy or non-informative.

Common failures

- Difficult to train: no pain, no GAN!
- There are several common types of GAN failures that provide intuition into ways to make GANs better.
  - "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
  - Sometimes GANs enter into clear cycles. They seem to generate a single digit relatively well, then start generating a dierent digit, etc. Looks like "mode collapse" on a rotating set of samples, but it does not differentiate.
  - Sometimes hard to describe failures, but videos like this are relatively typical. https://www.youtube.com/watch?v=D5akt32hsCQ

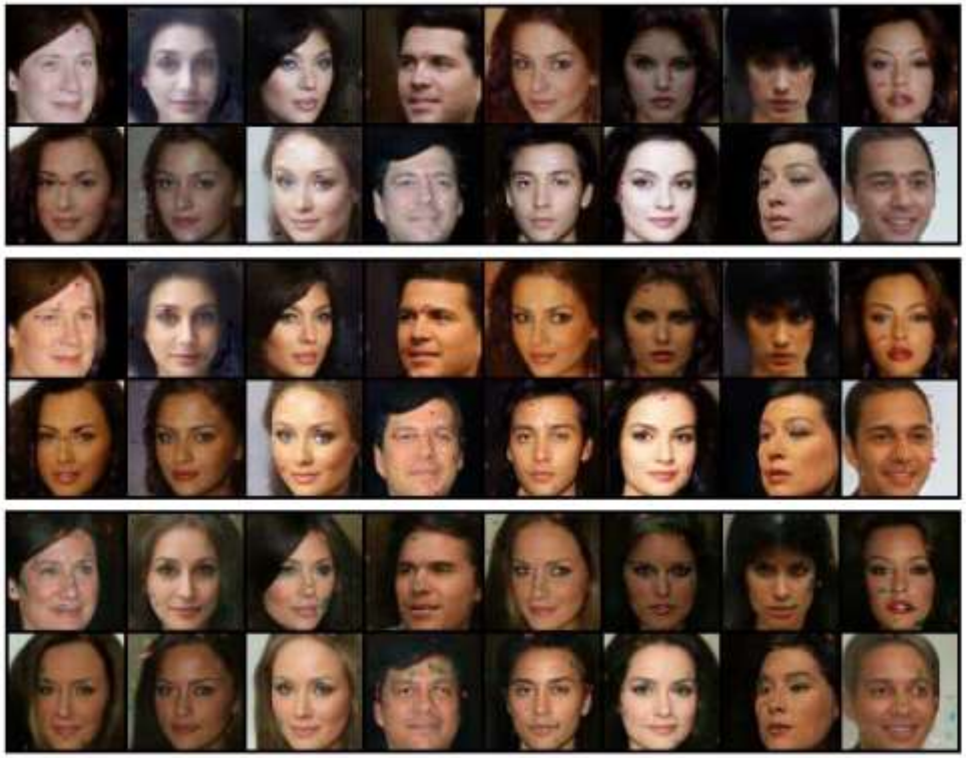
Common failures



Mode collapse example

Common failures

Mode collapse



Mode dropping

Outline

- Improving GAN training
  - WGANs
- Conditional GANs
  - Text-to-image: StackGANs
  - Image-to-image translation
- CycleGAN
  - Image-to-image translation with unpaired data

Acknowledgement: CMU, UofT, Stanford notes

Optimization techniques

- The main problem with GANs is that they are tricky to train
- There are many tricks to train them better
- 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

https://github.com/soumith/ganhacks

Wasserstein GAN

- Recall the GAN's formulation

  - Real data distribution P_r;
    Generator's distribution P_g, implemented as $x = G(z), z \sim P(z)$

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

  - **Discriminator**

    $$-\mathbb{E}_{x\sim P_r}[\log D(x)] - \mathbb{E}_{x\sim P_g}[\log(1 - D(x))] \qquad (1)$$

    $D(x)$: the probability that x from the real data rather than generator.

  - **Generator**

    $$\mathbb{E}_{x\sim P_g}[\log(1 - D(x))] \qquad \text{GAN}_0 \qquad (2)$$

    $$\mathbb{E}_{x\sim P_g}[-\log(D(x))] \qquad \text{GAN}_1 \qquad (3)$$

Wasserstein GAN

- Let's focus on the generator training

- Generator

$$\mathbb{E}_{x\sim P_g}[\log(1 - D(x))] \qquad \text{GAN}_0 \qquad (2)$$

$$\mathbb{E}_{x\sim P_g}[-\log(D(x))] \qquad \text{GAN}_1 \qquad (3)$$

Problems [Goodfellow et al., 2014]:

- P1: "In practice, GAN_0 may not provide sufficient gradient for G to learn well", GAN_1 is used instead. (log D trick)
- P2: "G collapses too many values of z to the same value of x" (Mode collapse in GAN_1)

Wasserstein GAN

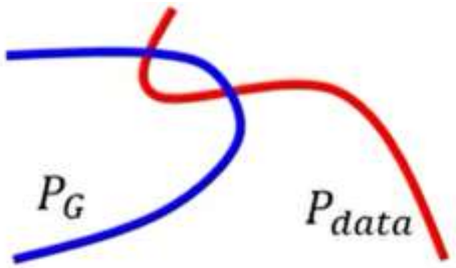
- P1:

In GAN$_0$, better discriminator leads to worse vanishing gradient in its generator

  - Reason: $$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

$$L = \mathbb{E}_{x \sim P_r}[\log D(x)] + \mathbb{E}_{x \sim P_g}[\log(1 - D(x))]$$

$$\Rightarrow \quad 2JS(P_r||P_g) - 2\log 2$$



  - If the supports of P_r and P_g almost have no overlap, then the JS divergence is 0 and there is no gradient info
  - The probability that the support of P_r and P_g almost have no overlap is 1

Wasserstein GAN

- P2:

GAN_1 is a conflicting/asymmetric objective, thus (1)unstable gradient (2) mode callapse

  - Reason: GAN_1 equals to optimize

$$KL(P_g||P_r) - 2JS(P_g||P_r)$$

  - Opposite signs for KL and JS
  - Mode dropping KL divergence

$KL(P_g||P_r)$ assigns an high cost to generating fake looking samples, and an low cost on mode dropping;
$KL(P_r||P_g)$ assigns an high cost to not covering parts of the data, and an low cost on generating fake looking samples;

Wasserstein GAN

- Re-think objective

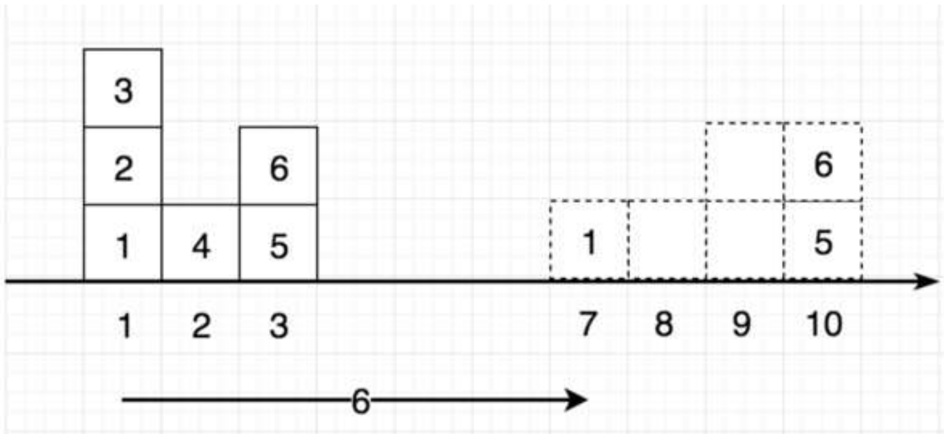
1. KL

$$KL(P||Q) = \mathbb{E}_P \log \frac{P}{Q}$$

2. JS

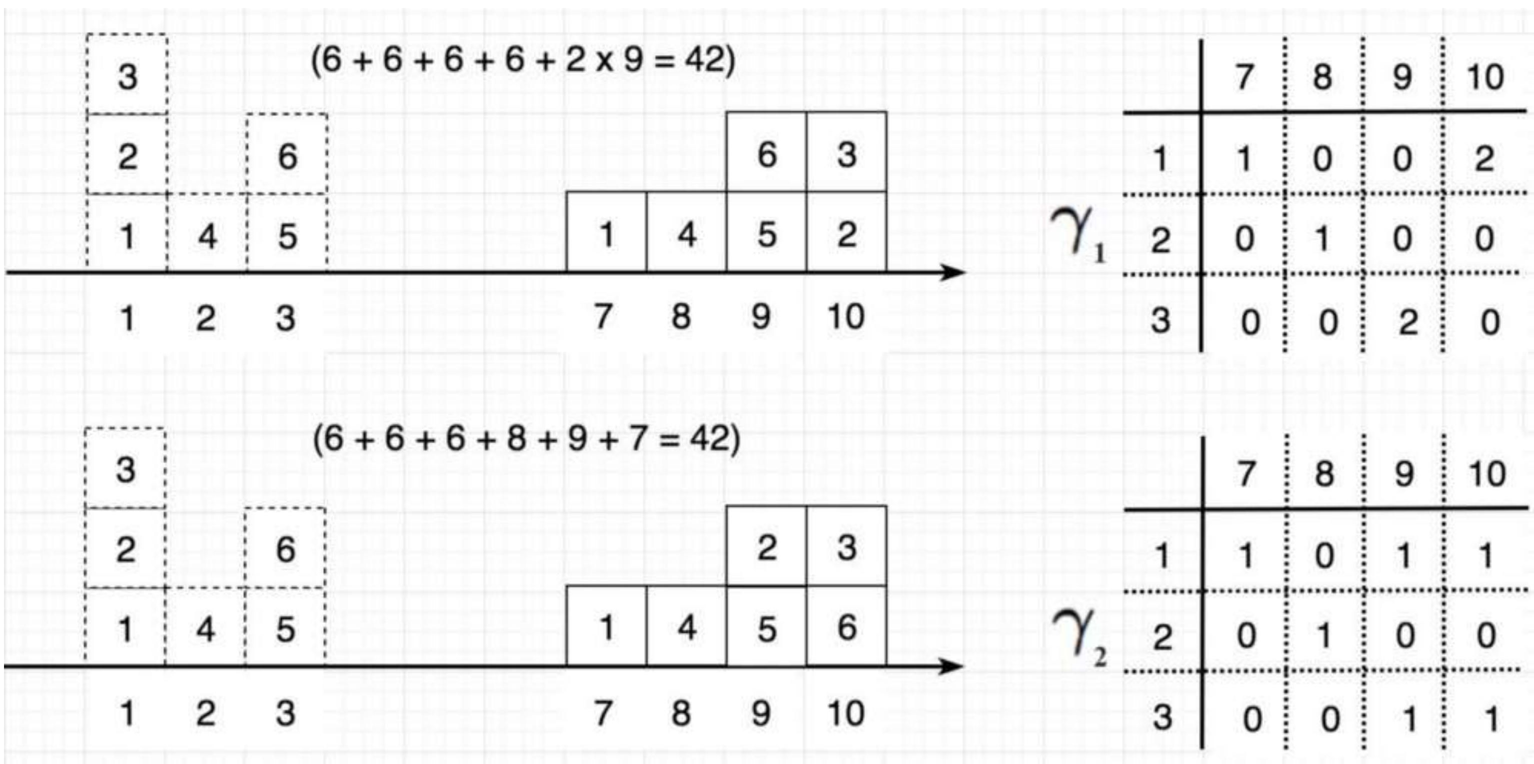
$$JS(P||Q) = \frac{1}{2}KL(P||\frac{P+Q}{2}) + \frac{1}{2}KL(Q||\frac{P+Q}{2})$$

- Let’s use a different distance between two distributions
  - Earth-mover distance/Wasserstein metric



Wasserstein GAN

- Earth-mover distance
  - Different transportation plan



  - The cost of the cheapest transportation plan

Wasserstein GAN

- Formal definition

Wasserstein

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

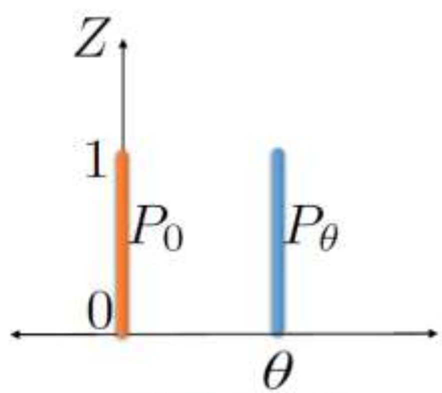
- $\Pi(P, Q)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are P and Q, respectively
- $\gamma(x, y)$ indicates a plan to transport "mass" from x to y, when deforming P into Q.
  The Wasserstein (or Earth-Mover) distance is then the "cost" of the **optimal** transport plan

Wasserstein GAN

- Examples of W-distance

P_0: distribution of $(0, Z)$, where $Z \sim U[0,1]$
P_θ: distribution of (θ, Z), where θ is a single real parameter

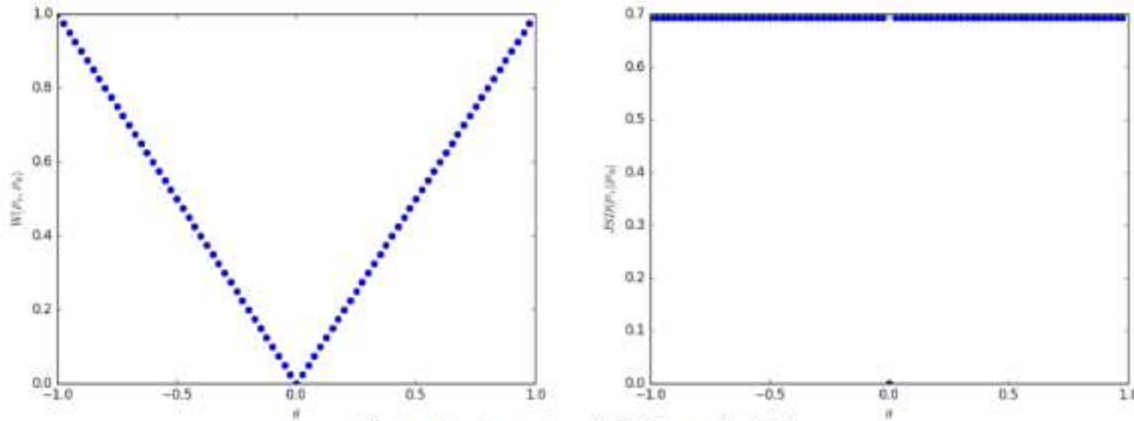
- $KL(P_0||P_\theta) = KL(P_\theta||P_0) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$
- $JS(P_0||P_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$
- $W(P_0||P_\theta) = |\theta|$



(a) Distributions

Wasserstein GAN

- Examples of W-distance



(b) Output of W and JS

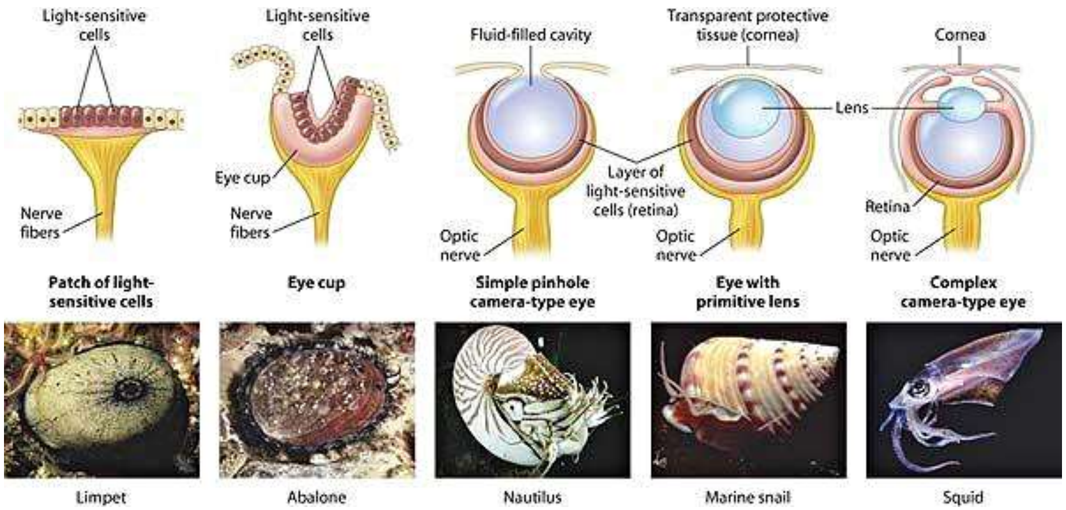
1. Observations

When the distributions are supported by low dimensional manifolds (such as P_r and P_g in GANs)

- KL or JS are binary, no meaningful gradient
- W is continuous and differentiable, hence always sensible

Wasserstein GAN

- Why EM distance?



P_{G_0} P_{data} …… $P_{G_{50}}$ P_{data} …… $P_{G_{100}}$ P_{data}

$JS(P_{G_0}, P_{data}) = log2$ $\quad$ $JS(P_{G_{50}}, P_{data}) = log2$ $\quad$ $JS(P_{G_{100}}, P_{data}) = 0$

$W(P_{G_0}, P_{data})$ $\quad$ $W(P_{G_{50}}, P_{data})$ $\quad$ $W(P_{G_{100}}, P_{data})$

Wasserstein GAN

Lipschitz Function

$\|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\|$

- Use W-distance in GAN
  - The infimum is highly intractable
  - Wasserstein distance has a duality form

$$W(P_r, P_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)]$$
$$= \frac{1}{K} \sup_{||f||_L \leq K} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)]$$

  where supremum is over all the K-Lipschitz functions

  - Consider a w-parameterized family of functions $\{f_w\}_{w\in W}$ that are all K-Lipschitz

$$W(P_r, P_g) = \max_{w\in W} \mathbb{E}_{x\sim P_r}[f_w(x)] - \mathbb{E}_{x\sim P_g}[f_w(x)]$$

  For example, $W = [-c, c]^l$

Wasserstein GAN

- Use W-distance in GAN
  - Loss for the discriminator

$$\mathbb{E}_{x\sim P_r}[f_w(x)] - \mathbb{E}_{x\sim P_g}[f_w(x)]$$

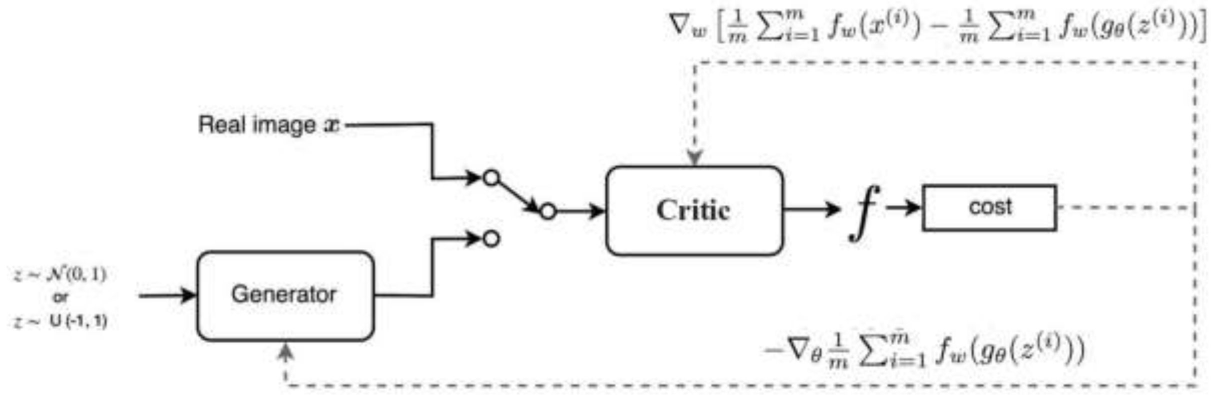
  - Loss for the generator

$$-\mathbb{E}_{x\sim P_g}[f_w(x)] = -\mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$$

  - Main difference
    - Remove the sigmoid of the last layer in D
    - Remove the log in the loss of D and G.
    - Clip the parameters of D in an inverval centered at 0.

GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -\log\left(D\left(G\left(z^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f(x^{(i)}) - f(G(z^{(i)}))\right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -f(G(z^{(i)}))$

Wasserstein GAN



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

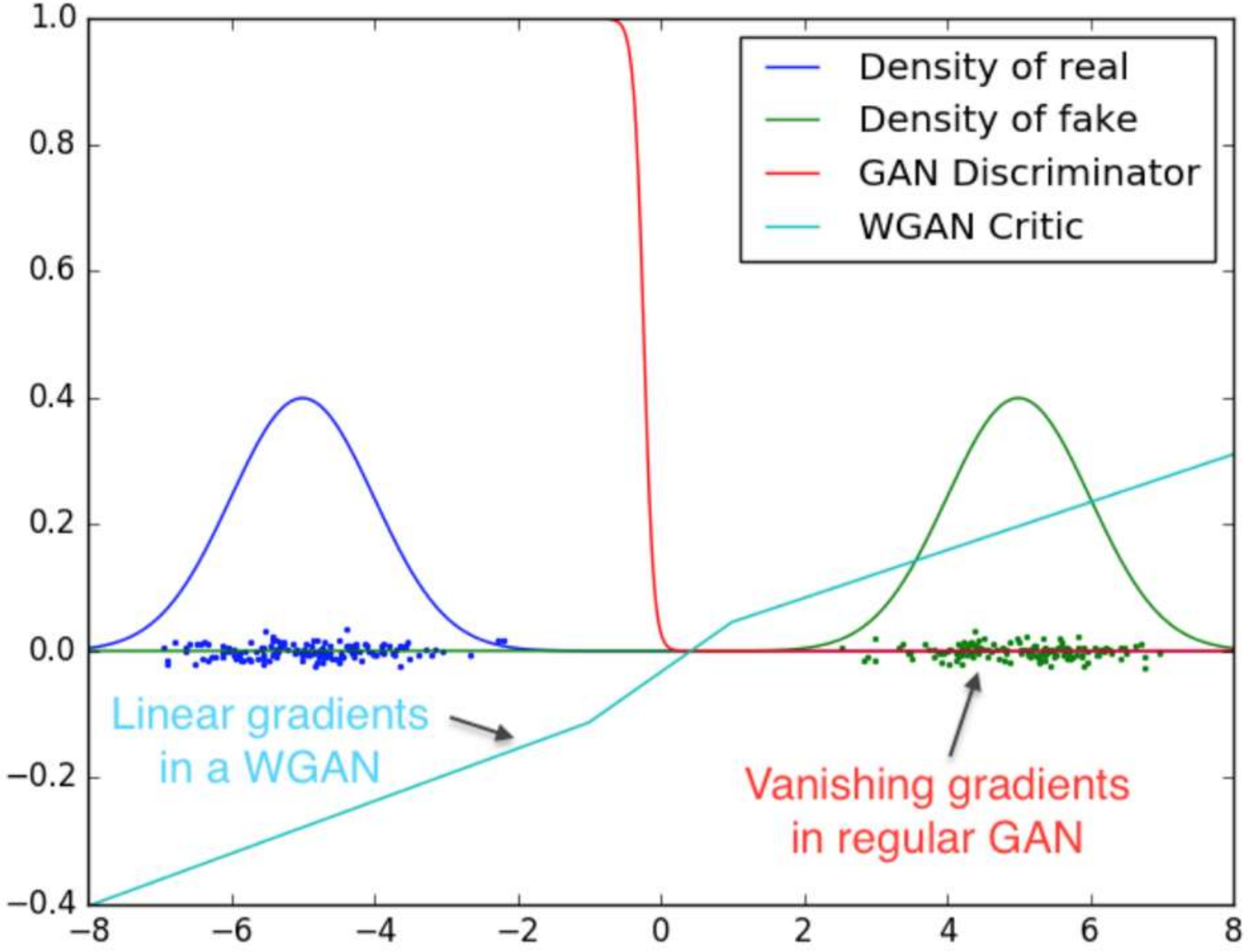
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

```
while θ has not converged do
    for t = 0, ..., n_critic do
        Sample {x^(i)}_{i=1}^m ~ P_r a batch from the real data.
        Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
        g_w ← ∇_w [ 1/m Σ_{i=1}^m f_w(x^(i)) − 1/m Σ_{i=1}^m f_w(g_θ(z^(i))) ]
        w ← w + α · RMSProp(w, g_w)
        w ← clip(w, −c, c)
    end for
    Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
    g_θ ← −∇_θ 1/m Σ_{i=1}^m f_w(g_θ(z^(i)))
    θ ← θ − α · RMSProp(θ, g_θ)
end while
```

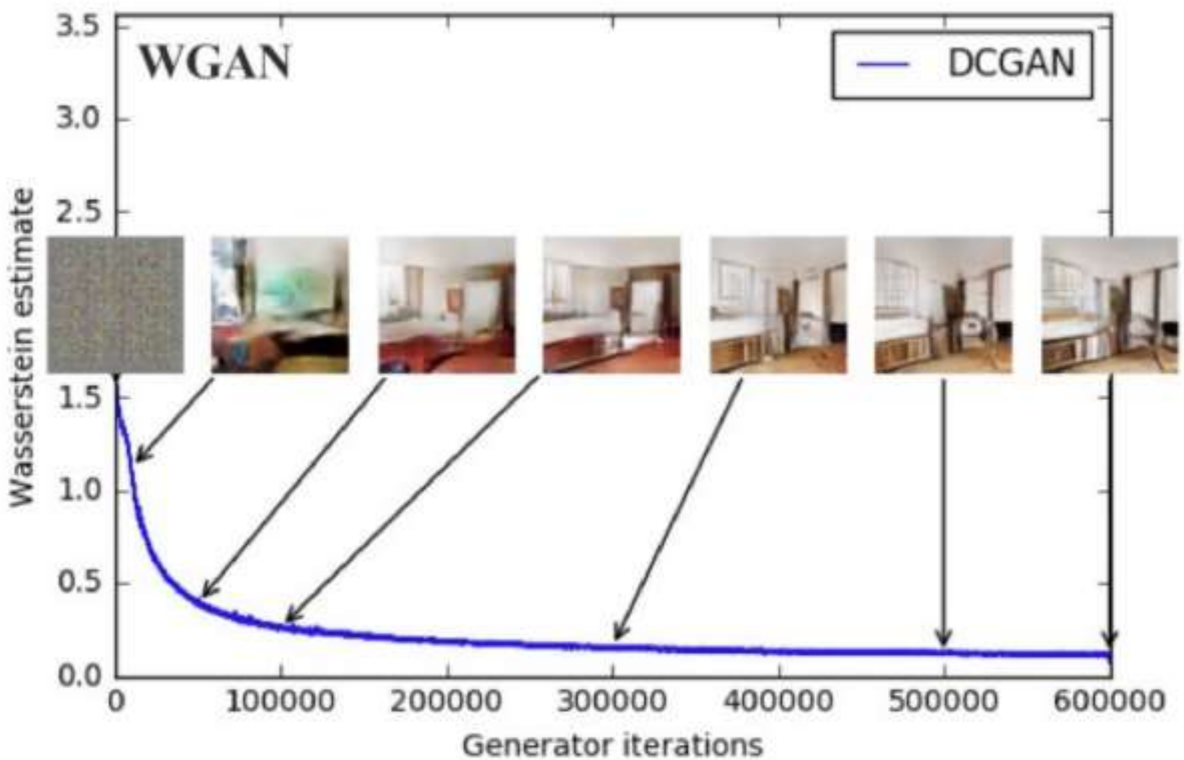
Wasserstein GAN

- Benefits



Wasserstein GAN

- Benefits



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

Wasserstein GAN

- Benefits
  - **A meaningful loss metric that correlates with the generator's convergence and sample quality.** WGAN algorithm attempts to train the critic relatively well before each generator update, the loss function at this point is an estimate of the EM distance.
  - It allows us to train the critic till optimality, and thus no longer need to balance generator and discriminator's capacity properly

A generator without batch normalization in DCGAN

  - In no experiment did the authors see evidence of mode collapse

A generator constrcuted with MLP

Outline

- Improving GAN training
  - WGANs
- Conditional GANs
  - Text-to-image: StackGANs
  - Image-to-image translation
- CycleGAN
  - Image-to-image translation with unpaired data

Acknowledgement: CMU, UofT, Stanford notes

Conditional GANs

- Conditional GANs include a label and learn P(X|Y)
  - Add conditional variable y into G and D
  - Objective function

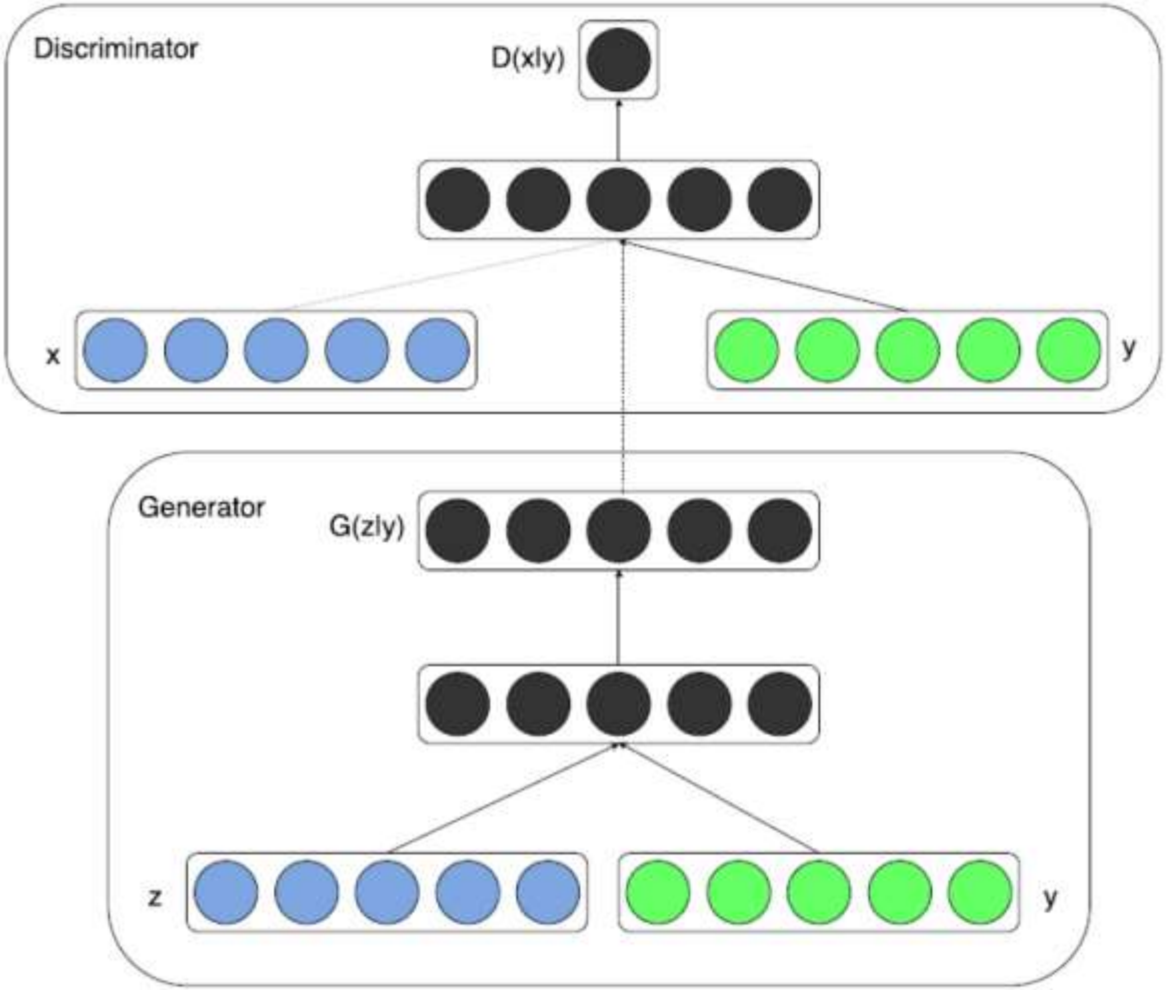
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\downarrow$$

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

Conditional GANs

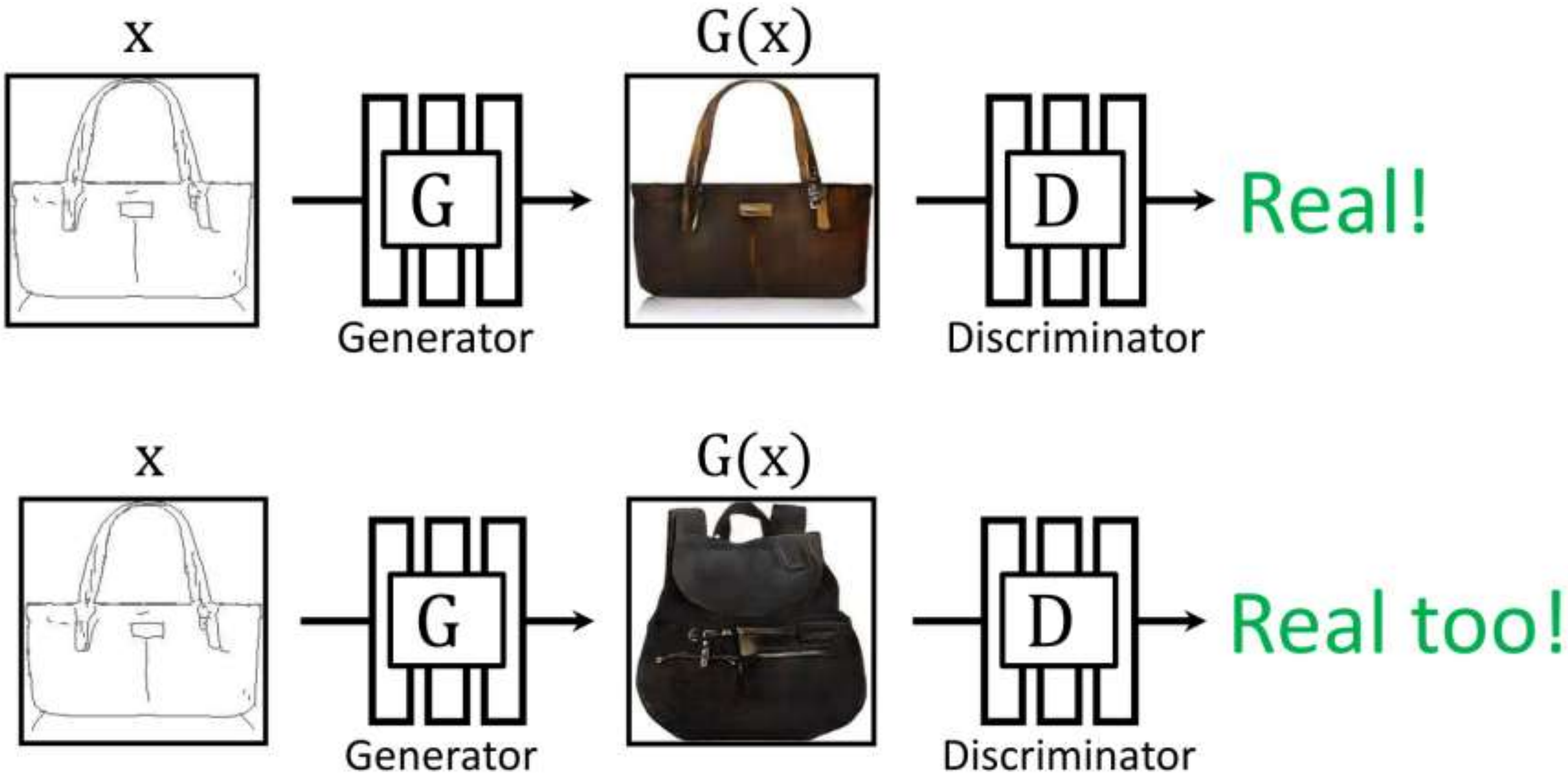
- Model architecture



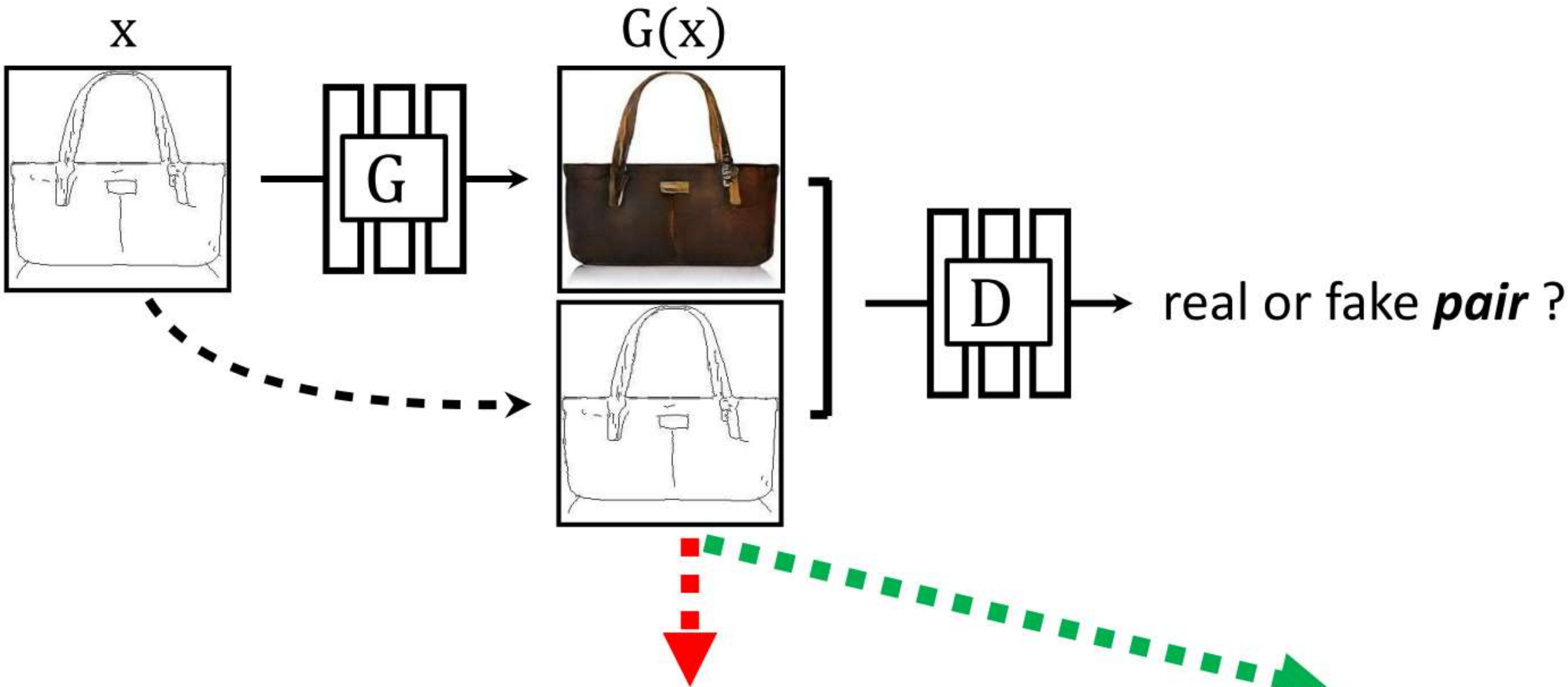
Conditional GANs

- Positive samples for D
  - True data + corresponding conditioning variable
- Negative samples for D
  - Synthesized data + corresponding conditioning variable
  - *True data + non-corresponding conditioning variable*

Conditional GANs



Conditional GANs



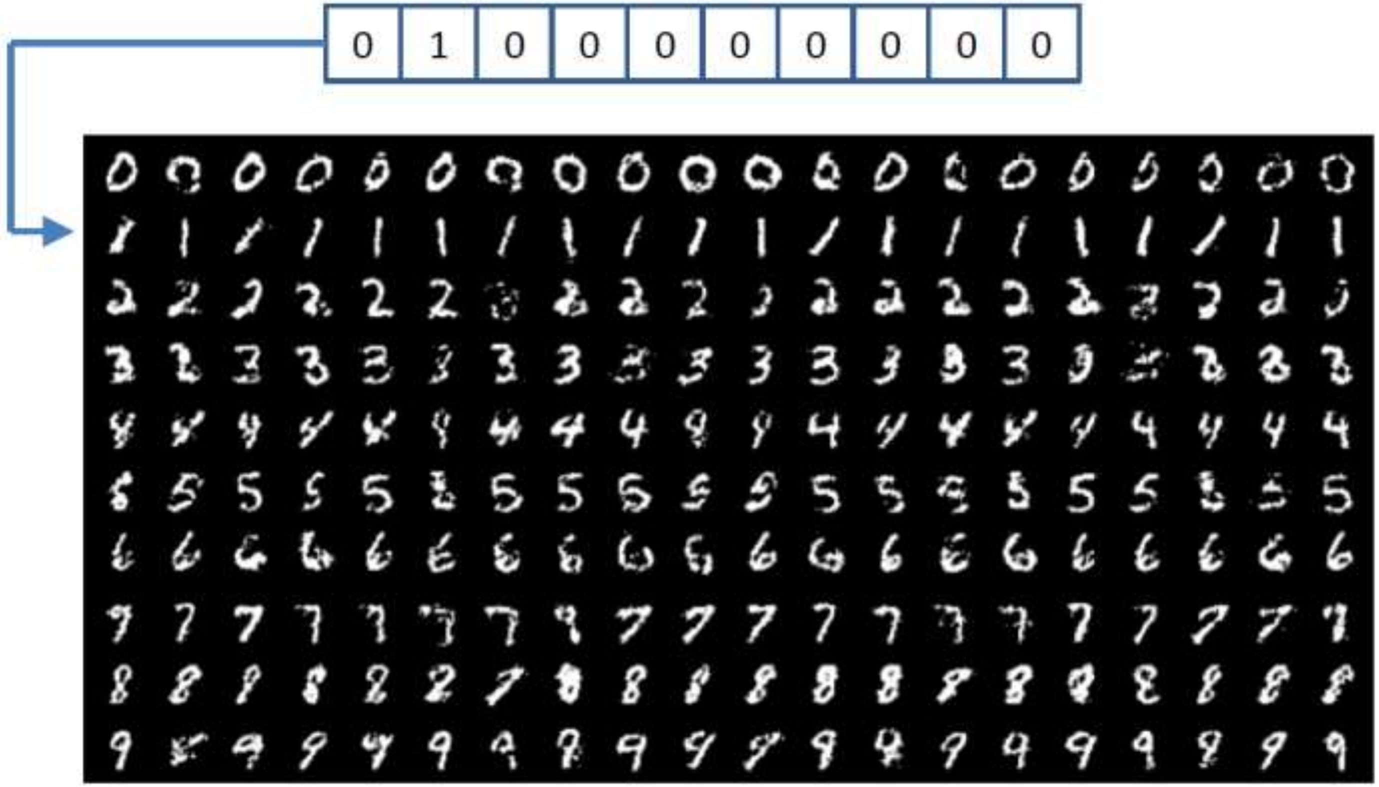
$$\min_G \max_D \mathbb{E}_{x,y}[\log D(\underbrace{x, G(x)}_{\text{fake pair}}) + \log(1 - \underbrace{D(x,y)}_{\text{real pair}})]$$

match joint distribution $p(G(x), y) \sim p(x, y)$

slides credit: Isola / Zhu

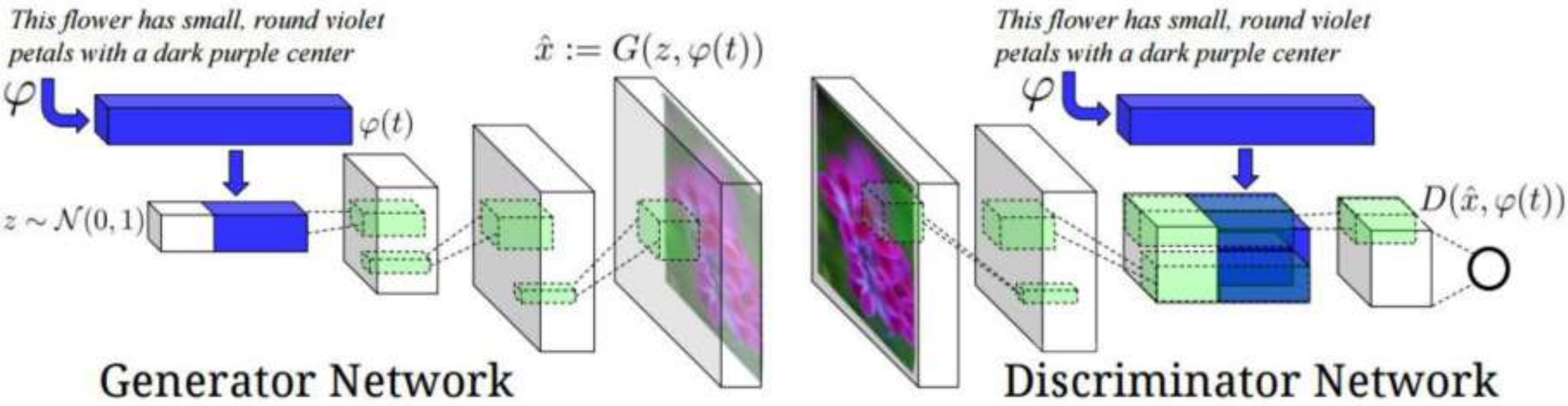
Conditional GANs

- MNIST example
  - Each row is conditioned on a different label.
  - A single neural network to generate all 10 digits



Mirza and Osindero 2016

Text-to-Image synthesis



Reed et al 2015

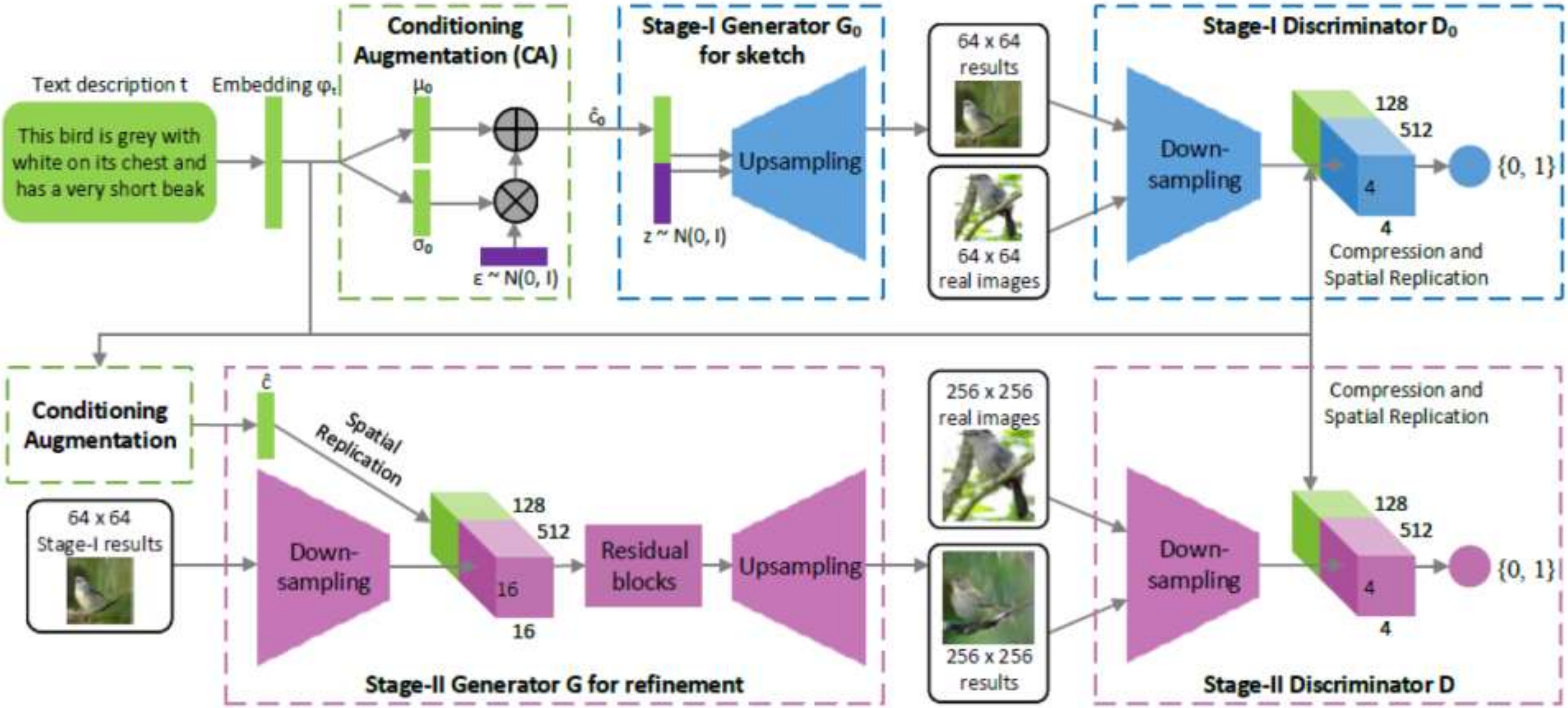
StackGAN

- A coarse-to-fine manner

Zhang et al. 2016

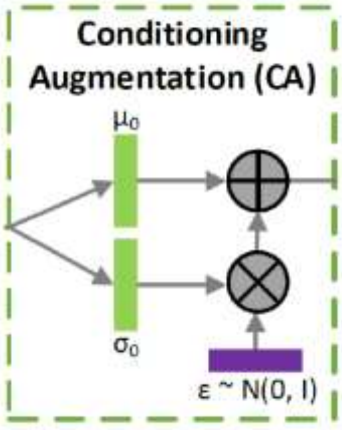
StackGAN

- Use stacked GAN structure

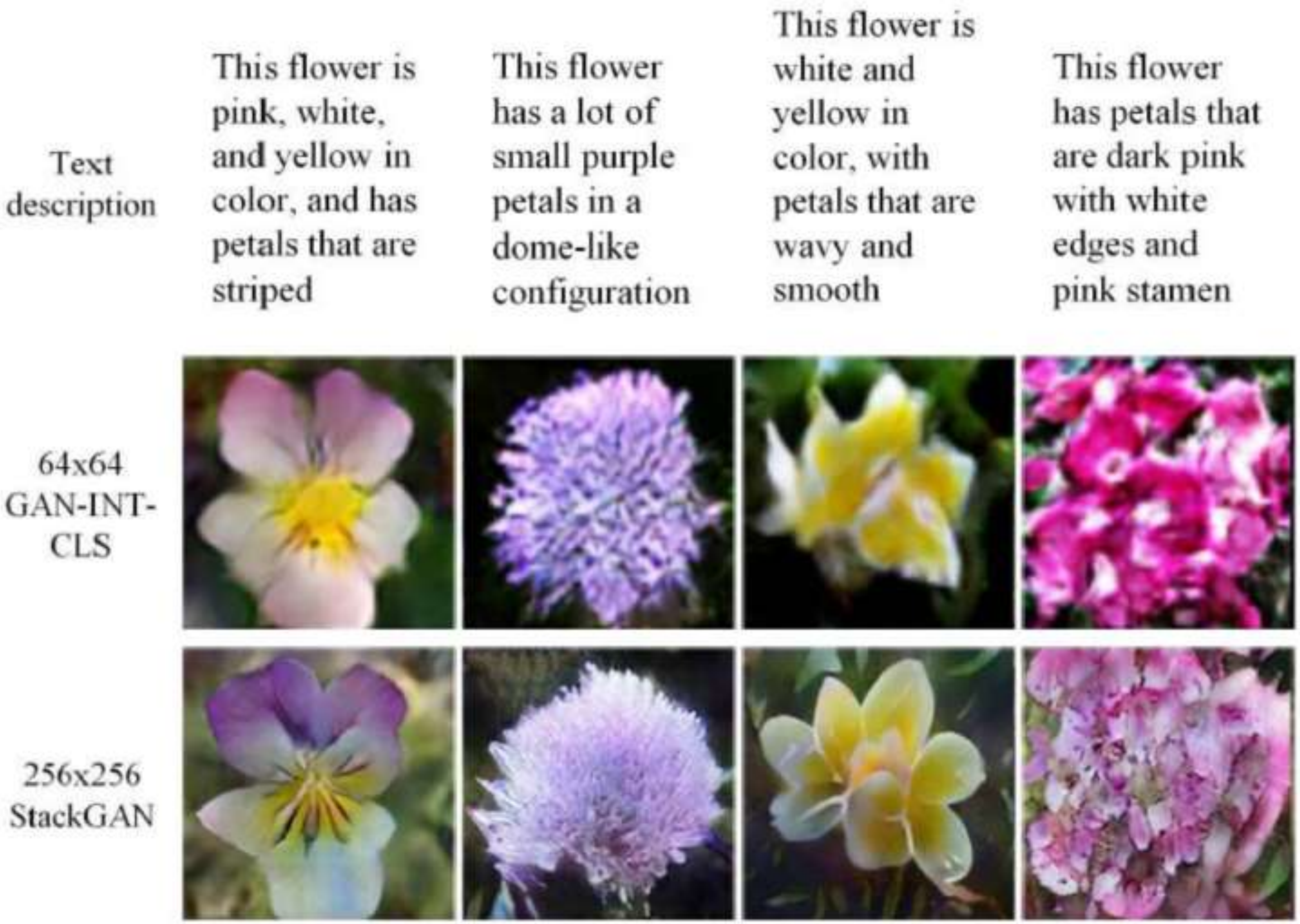


StackGAN

- Model design
  - Conditioning augmentation
    - Encoder with sampling step
  - No random noise vector for Stage-2
  - Conditioning both stages on text
  - Spatial replication for the text conditional variable
  - Negative samples for D
    - True images + non-corresponding texts
    - Synthetic images + corresponding texts



More StackGAN results



Conditional image synthesis

- Problem formulation
  - Input: original image (low-res or partially observed)
  - Output: target image (high-res or full image)
  - Often formulated as a regression problem

$$\tilde{Y} = F_{net}(X; W)$$

$$\min_W E_X[L(Y, \tilde{Y})]$$

  - However, conventional loss function usually leads to unsatisfactory results.
- Solution: Adding adversarial loss terms
  - Mapping to a distribution instead of a single image
  - Structural loss (distribution-wise) instead of point-wise loss

Image-to-image translation

- One-to-many or many-to-one mapping [Isola et al., 2016]

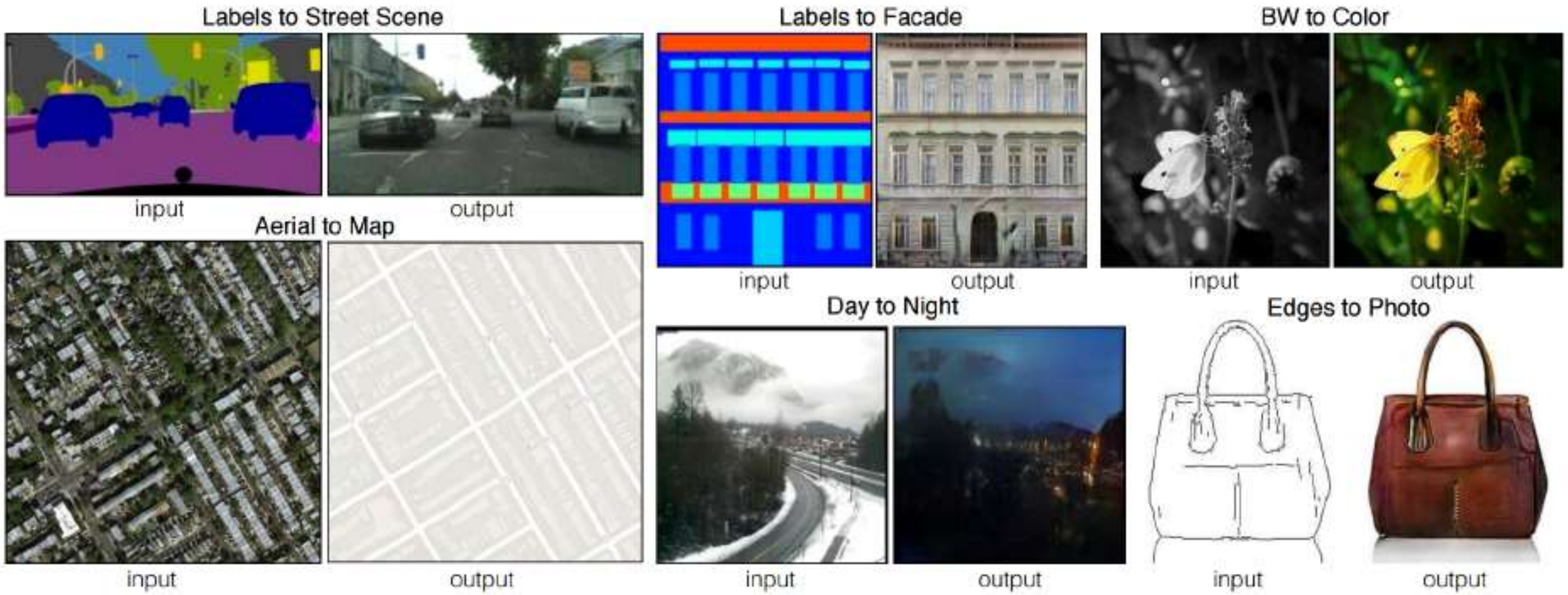


Image-to-image translation

- Combine the CGAN objective function with the L1 loss

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y\sim p_{data}(x,y),z\sim p_z(z)}[\|y - G(x,z)\|_1].$$

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda\mathcal{L}_{L1}(G).$$

- Use the U-net structure for the generator

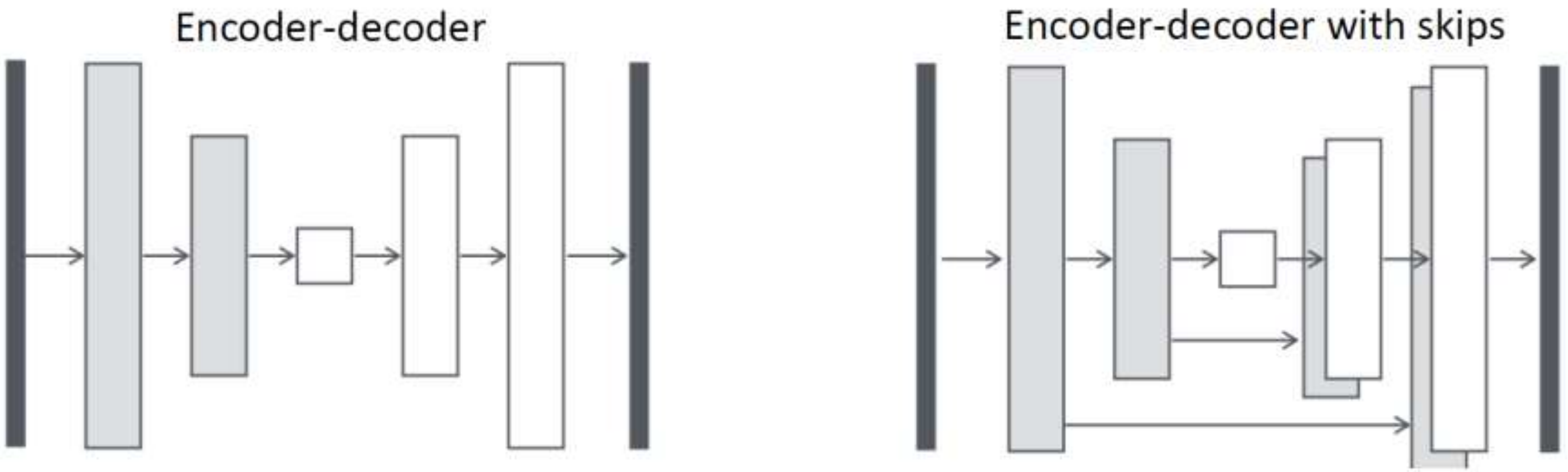


Image-to-image translation

- Patch-based discriminator
  - Separate each image into N x N patches
  - Train a patch-based discriminator

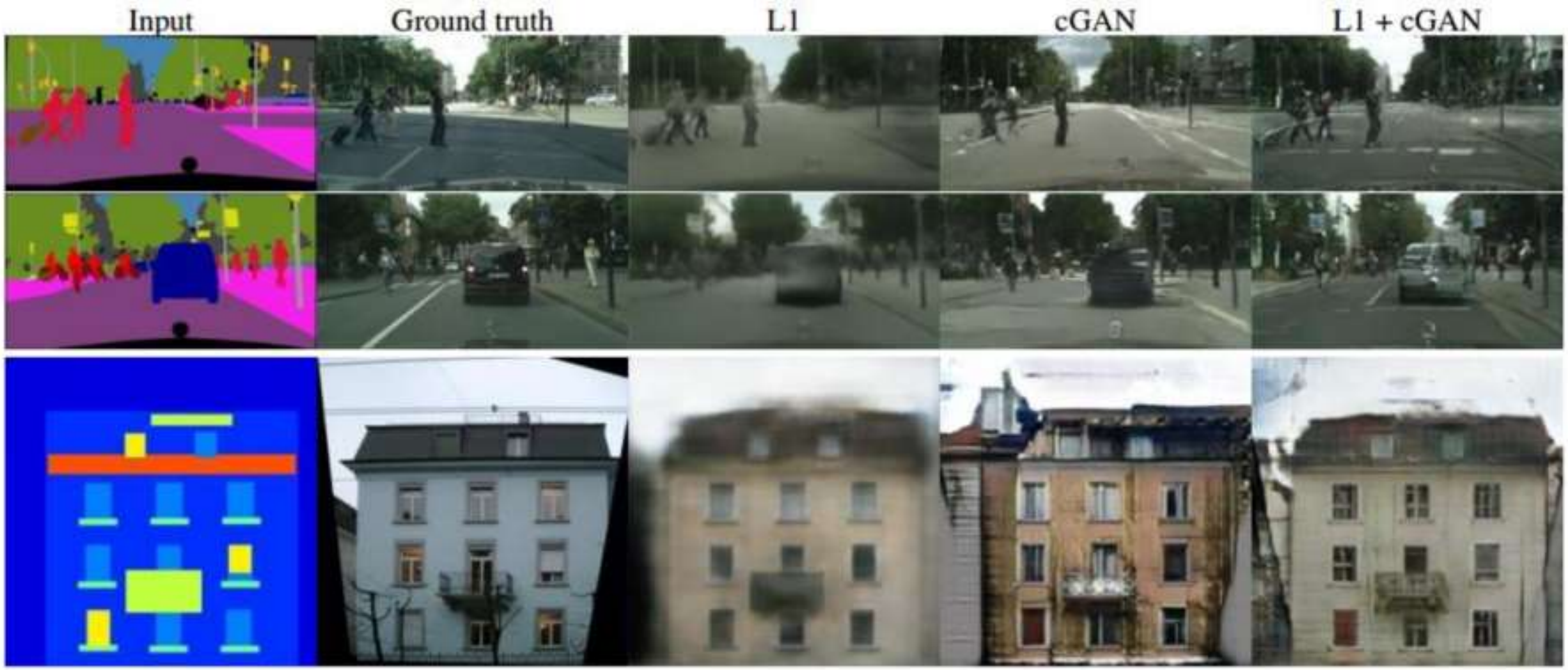
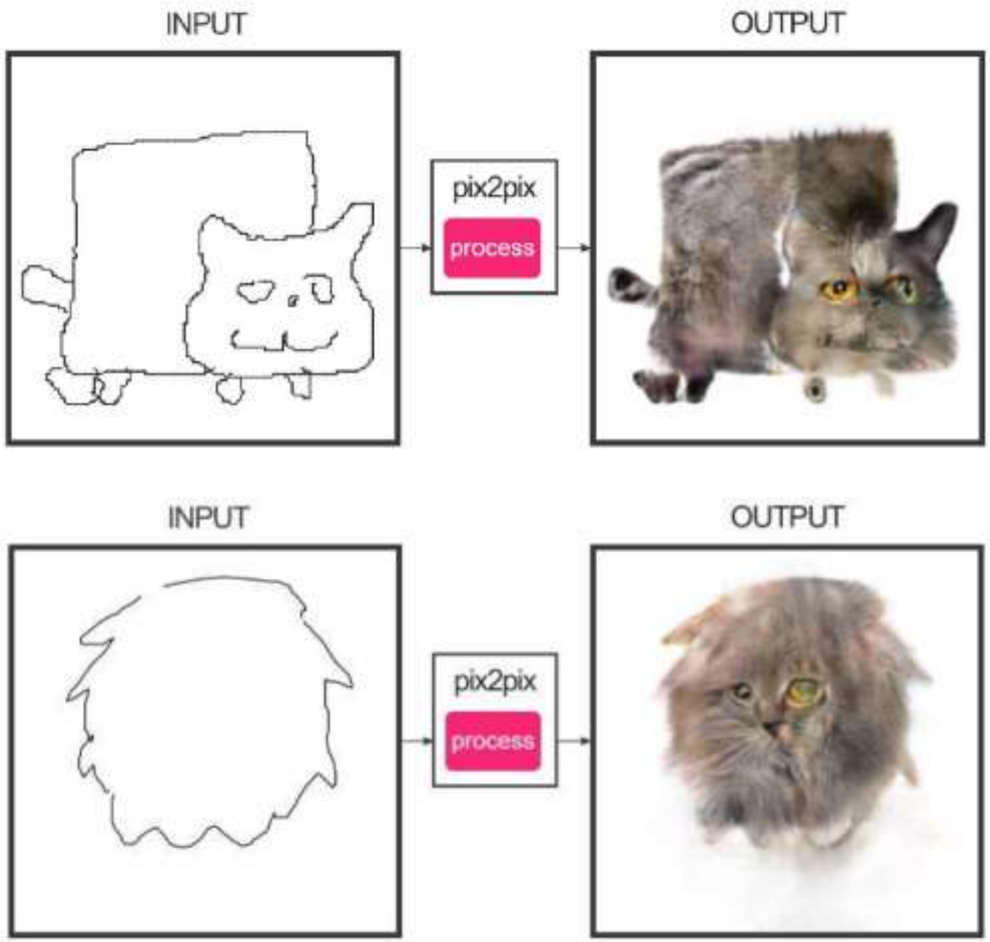


Image-to-image translation

- More results

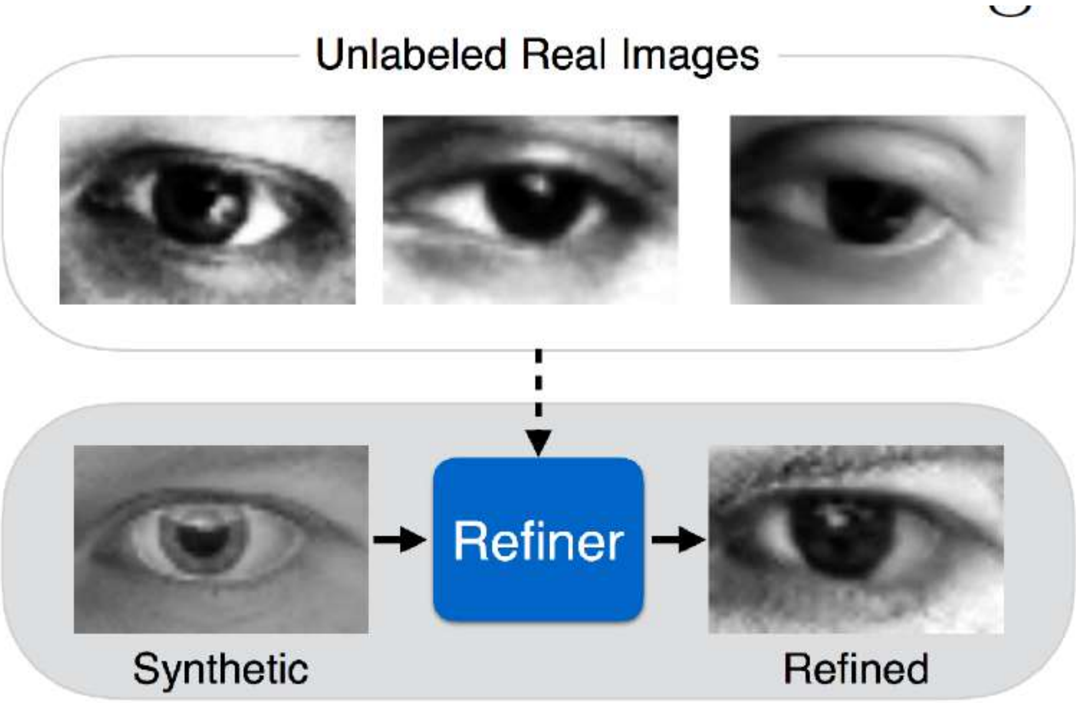
Image-to-image translation

- More results



Other im2im translation

- CGANs for simulated training data



(Shrivastava et al., 2016)

Sim-to-real synthesis

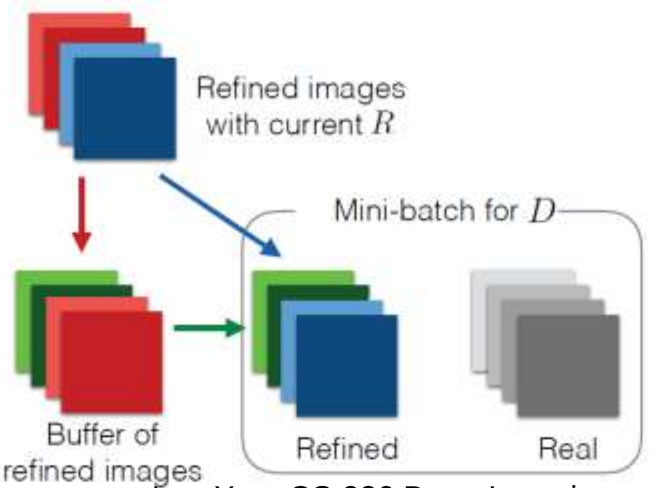
- Refiner network

$$\tilde{\mathbf{x}} = R_\theta(\mathbf{x})$$

- Learning objective

$$L_D(\phi) = -\sum_i \log(D_\phi(\tilde{\mathbf{x}}_i)) - \sum_j \log(1 - D_\phi(\mathbf{y}_j))$$

$$L_R(\theta) = -\sum_i \underbrace{\log(1 - D_\phi(R_\theta(\mathbf{x}_i)))}_{\text{Realistic style}} + \underbrace{\lambda \|\psi(R_\theta(\mathbf{x}_i)) - \psi(\mathbf{x}_i)\|_1}_{\text{Label information (content)}}$$



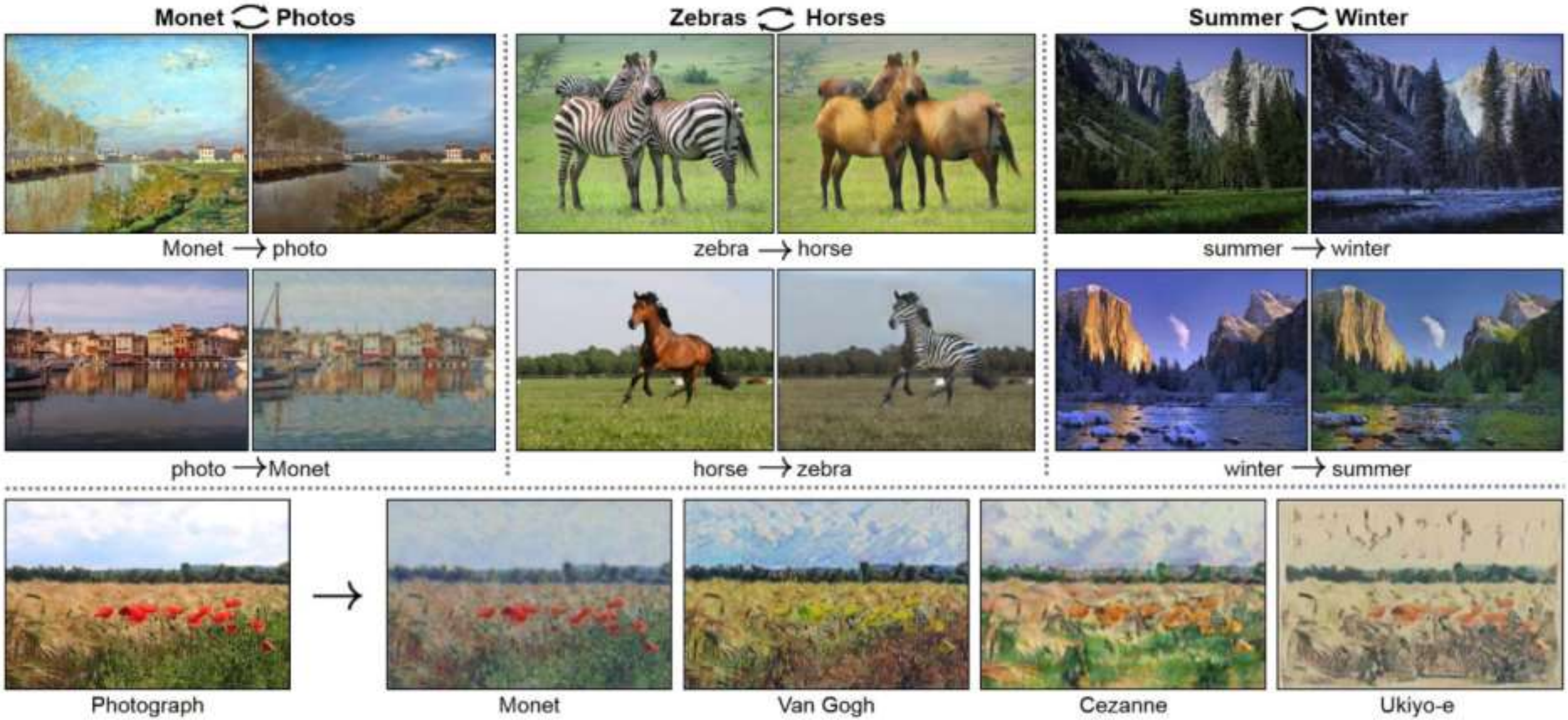
Outline

- Improving GAN training
  - WGANs
- Conditional GANs
  - Text-to-image: StackGANs
  - Image-to-image translation
- CycleGAN
  - Image-to-image translation with unpaired data

Acknowledgement: CMU, UofT, Stanford notes

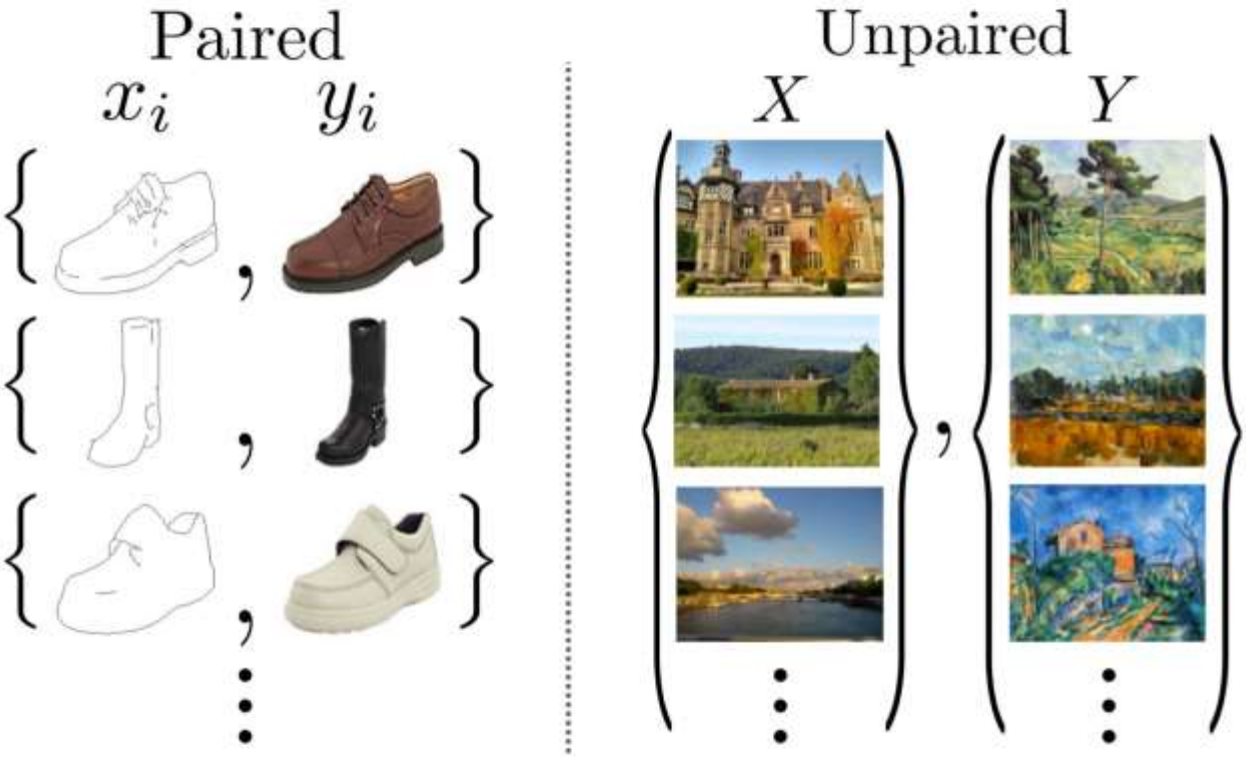
CycleGAN

- Image-to-image translation without paired data



CycleGAN

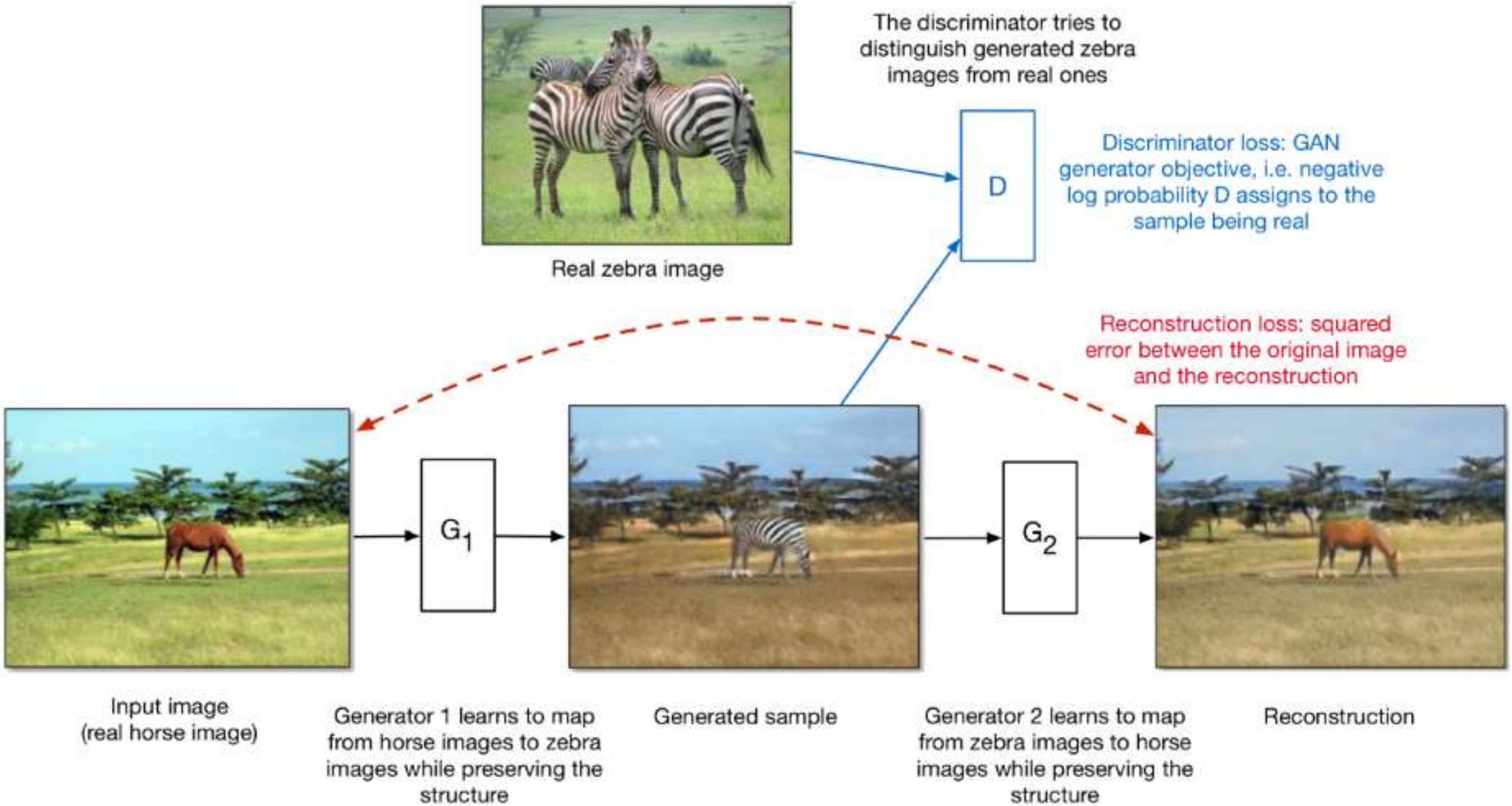
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.



CycleGAN

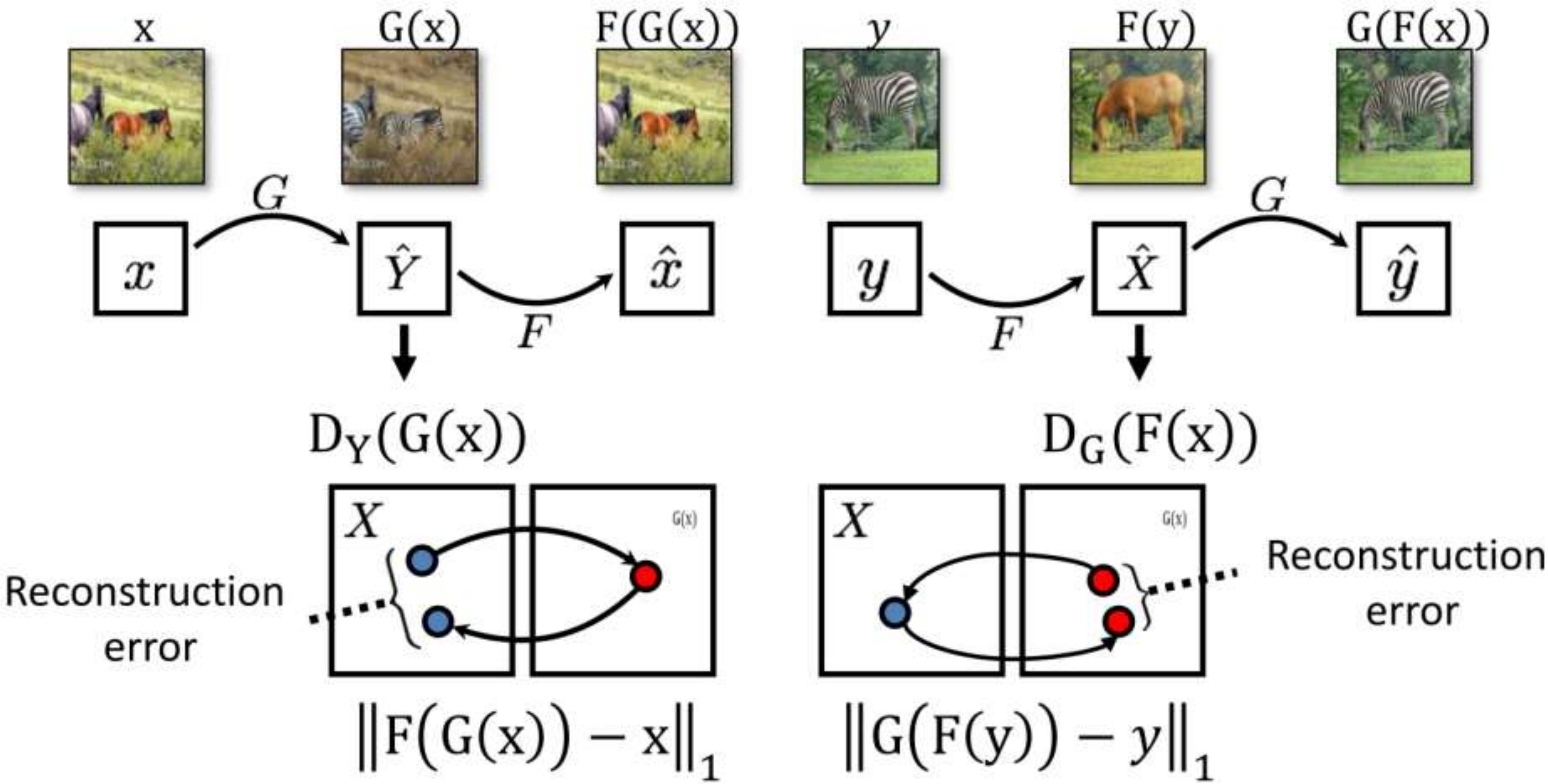
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - Make sure the generators are **cycle-consistent**: mapping from style 1 to style 2 and back again should give you almost the original image.

CycleGAN



Total loss = discriminator loss + reconstruction loss

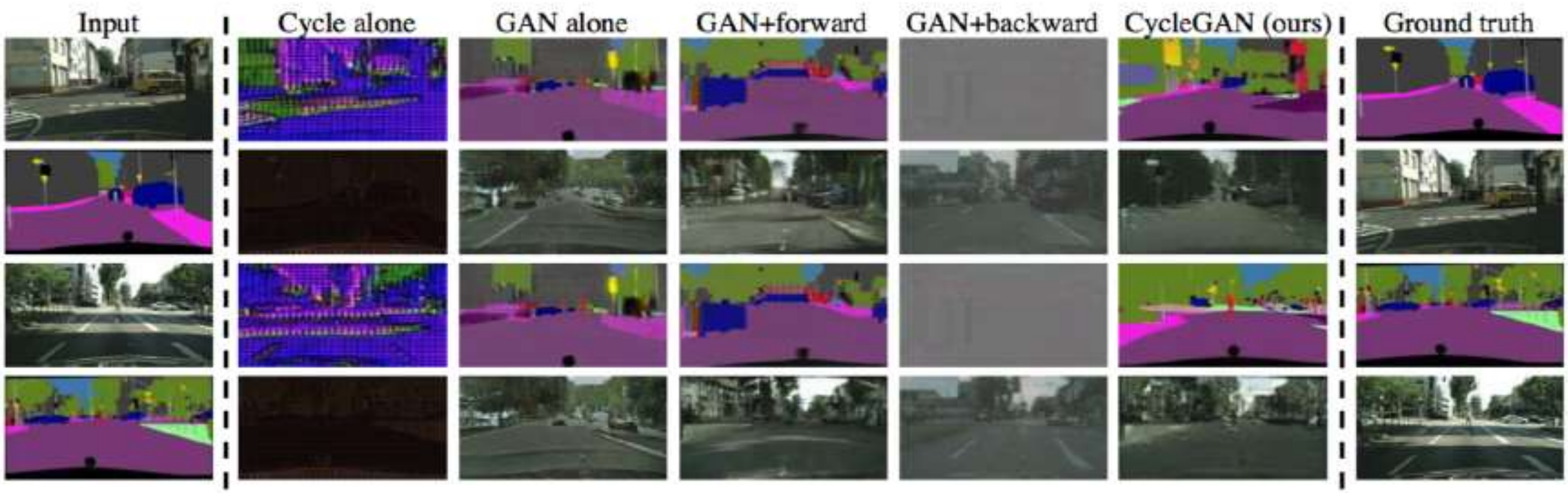
CycleGAN


x
G(x)
F(G(x))
y
F(y)
G(F(x))
G
F
G
F
x
$\hat{Y}$
$\hat{x}$
y
$\hat{X}$
$\hat{y}$
$D_Y(G(x))$
$D_G(F(x))$
X
X
Reconstruction error
Reconstruction error
$\|F(G(x)) - x\|_1$
$\|G(F(y)) - y\|_1$

CycleGAN

- Results

Style transfer between road scenes and semantic segmentations (labels of every pixel in an image by object category):



CycleGAN

- Results

- More details

https://hardikbansal.github.io/CycleGANBlog/

Summary

- Variants of GANs
  - Improving GANs
  - Conditional GANs: Conditional image synthesis
  - CycleGAN: Image-to-image translation with unpaired data
- Next time
  - Applications of GANs, Combination of VAE and GAN
- Keep working on the projects!