Generative Adversarial Networks – Part 2

11785 Deep Learning

Fall 2023

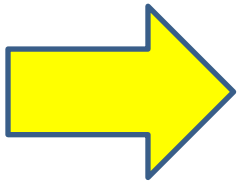
Jeel Shah, Harini Subramanyan

Slides borrowed from Abuzar Khan

Topics for the week

- Transformers
- GNNs
- VAEs
- GANs
- Connecting the dots

The problem



Try visiting:
https://thispersondoesnotexist.com

- From a large collection of images of faces, can a network learn to *generate* new portrait
  - Generate samples from the distribution of “face” images
    - How do we even characterize this distribution?

Discriminative vs Generative Models

Given a distribution of inputs X and labels Y.

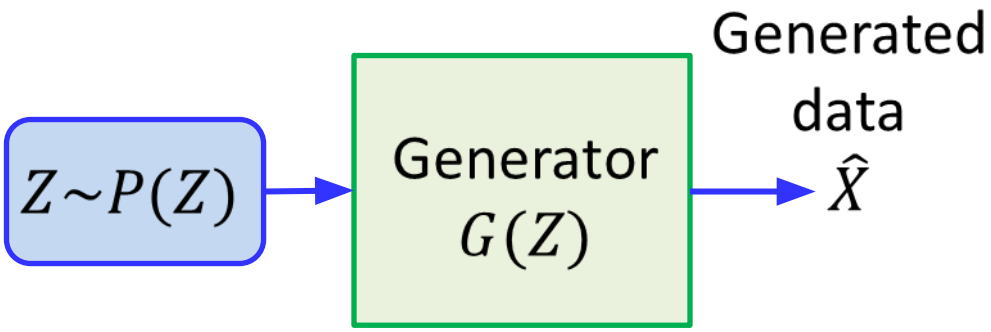
Discriminative models

- Discriminative models learn conditional distribution P(Y | X)
- Learns decision boundary between classes.
- Limited scope. Can only be used for classification tasks.
- E.g. Logistic regression, SVM etc.

Generative models

- Generative models learn the joint distribution P(Y, X)
- Learns actual probability distribution of data.
- Can do both generative and discriminative tasks.
- E.g. Naïve Bayes, Gaussian Mixture Model etc.
- Harder problem, requires a deeper understanding of the distribution than discriminative models.

What we have seen: VAE



- The decoder of a VAE is a generator!
- $\hat{X}$ → What input image X would have been encoded into the Z that I saw?
- Trained by **maximizing** the **likelihood of the data**
  - Likelihood maximization does not actually relate to whether the output *actually looks* like a face
- Can we make the training criteria a little more direct?

The problem

What's the easiest way to check if the produced output looks like a face or not?

The problem

What's the easiest way to check if the produced output looks like a face or not?

- You could just eyeball 👀 the results and use your understanding of what faces look like to evaluate what is generated.
- Unfortunately, **you** are not a differentiable function 🙃
- But what if we could, find a differentiable proxy for you and your non-differentiable-ness?

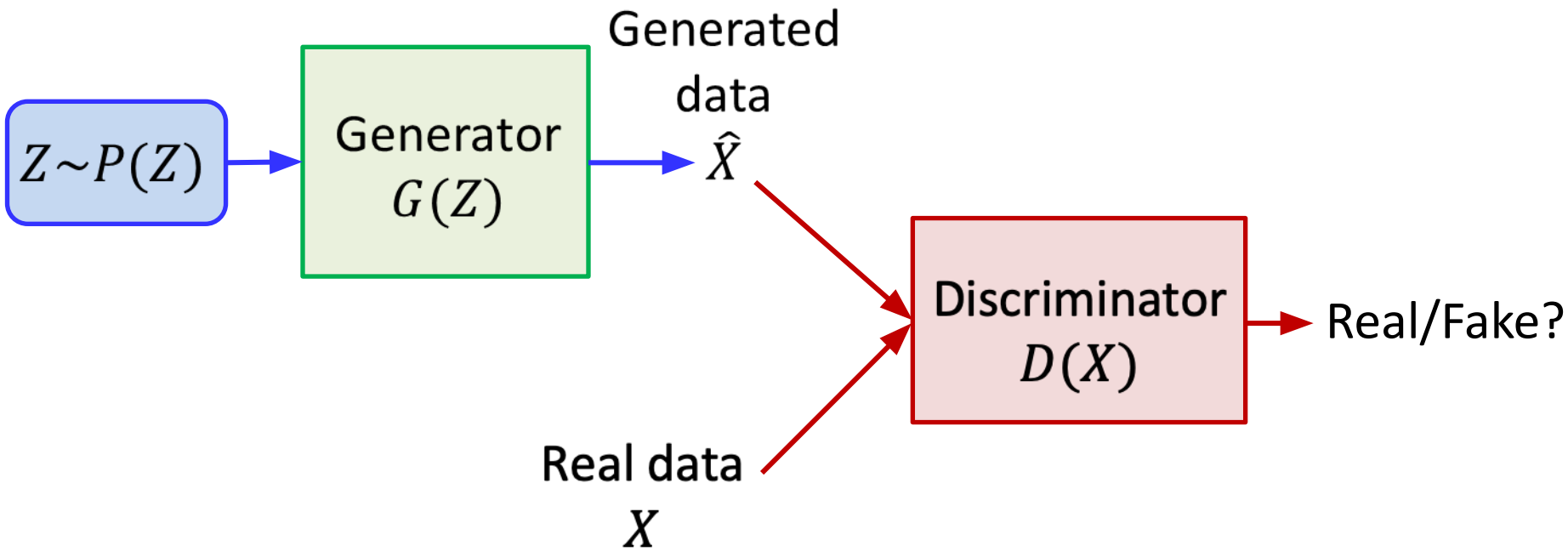
What are GANs

Generative Models which generate data similar to the training data .
E.g. Variational Autoencoders (VAE)

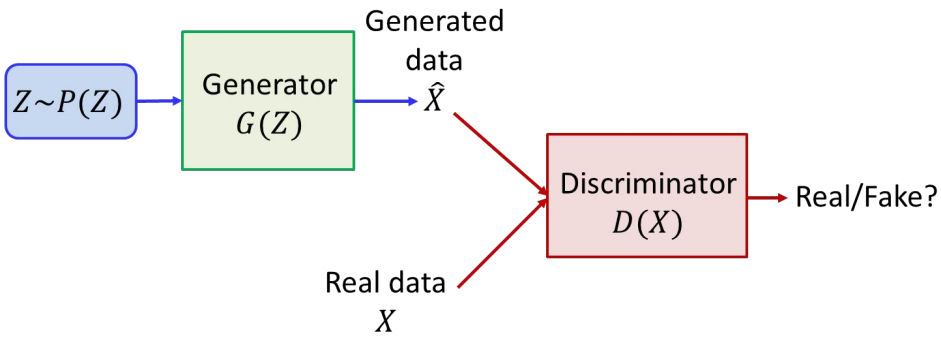
Neural Networks

Adversarial Training
GANS are made up of two competing networks (adversaries) that are trying beat each other.
A “game” is being played between the two.

What are GANs?

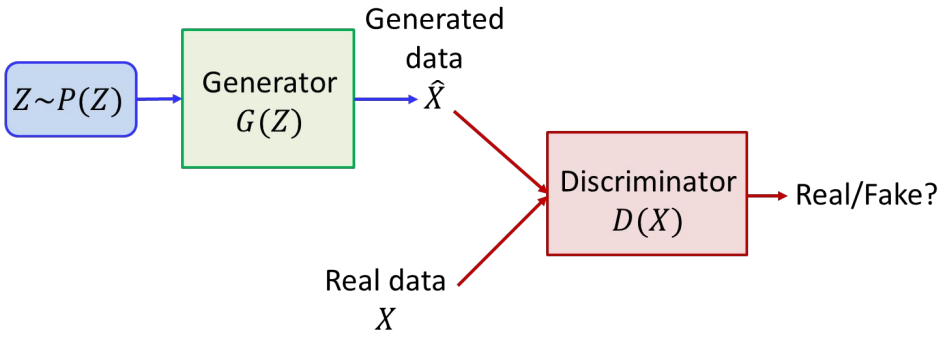


The GAN formulation



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

The GAN formulation

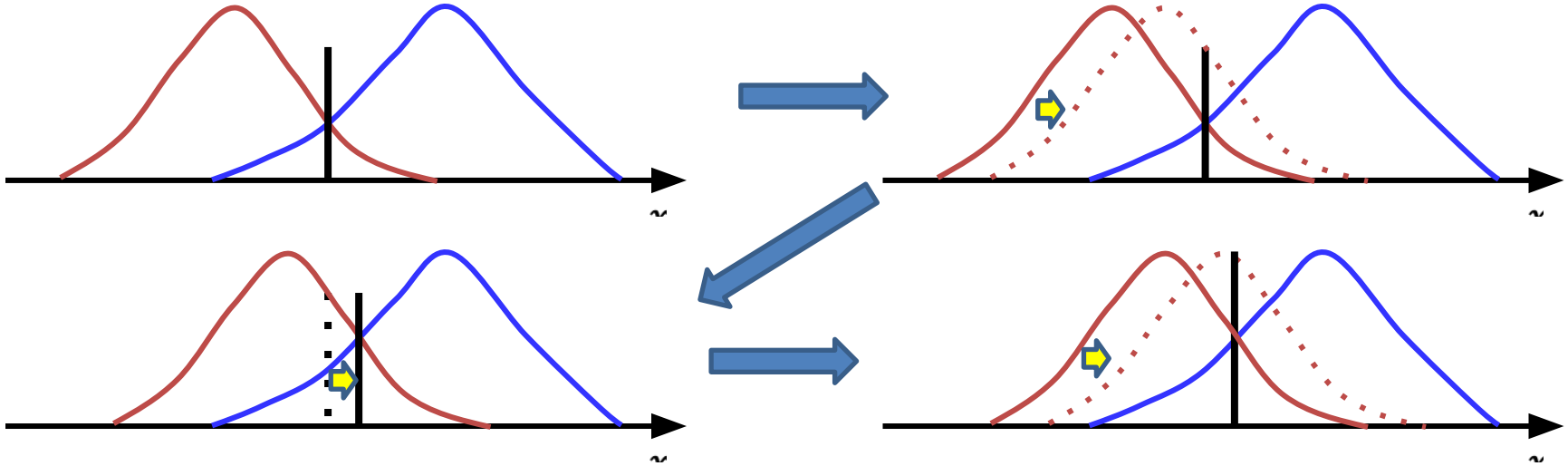


- The original GAN formulation is the following min-max optimization

$$\min_{G} \max_{D} \; E_{\boldsymbol{x} \sim \boldsymbol{P_X}} \log D(x) + E_{\boldsymbol{x} \sim \boldsymbol{P_G}} \log(1 - D(\boldsymbol{x}))$$

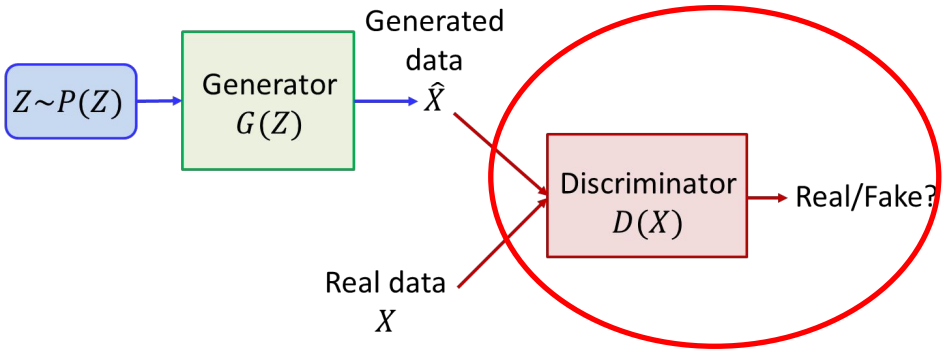
- Objective of D: Optimize for $D(x) = 1$ and $D(G(z)) = 0$
- Objective of G: Optimize for $D(G(z)) = 1$

The iterated learning



- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary “into” the real distribution
- Discriminator relearns new “perfect” boundary
- Generator shifts distribution past new boundary
- …
- In the limit Generator’s distribution sits perfectly on “real” distribution and the perfect discriminator is still random

Analysis of optimal behavior: The optimal discriminator

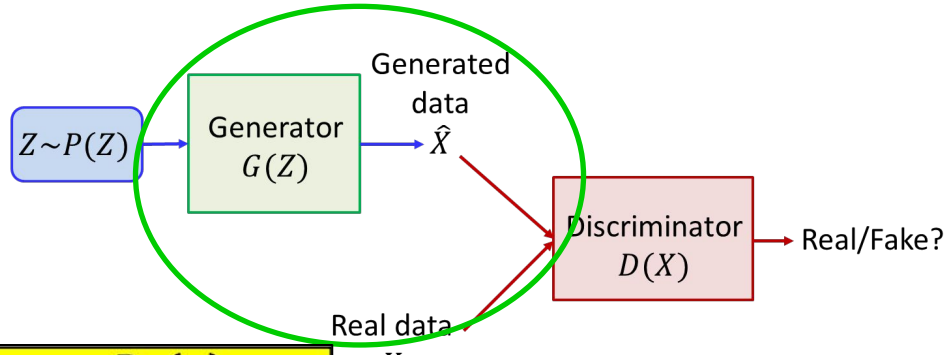


- The **optimal discriminator** would be a Bayesian classifier

$$D(x) = \frac{P_X(x)}{P_X(x) + P_G(x)}$$

  - Assuming uniform prior

Analysis of optimal behavior: The optimal generator



$$D(x) = \frac{P_X(x)}{P_X(x) + P_G(x)}$$

- The optimal generator:

$$\min_G 2D_{JSD}\left(P_X(x), P_G(x)\right) - \log 4$$

- The optimal generator minimizes the Jensen Shannon divergence between the distributions of the actual and synthetic data!
  - Tries to make the two distributions maximally similar

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

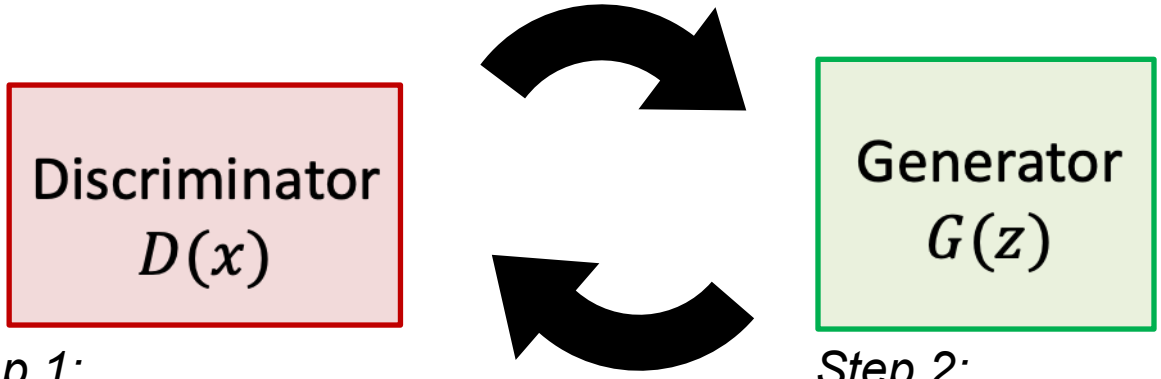
- Minimizing the Jenson-Shannon divergence between distributions of synthetic and true data
- Use a discriminator to optimize generator
- Simpler formulation
- Noisy and difficult optimization
- Sharper results

Training GANs

Notation

- Data sample: x
- Input noise vector: z
- Distribution of Real Data: P_X
- Distribution of Generated Data: P_G
- Distribution of Input noise vector: P_Z
- Generator: $G(z; \theta_G)$
- Discriminator: $D(x; \theta_D)$
- Generator output: $G(z), \hat{x}$
- Discriminator output: $D(x), D(G(z))$

How to Train a GAN?



Step 1:
Train the Discriminator
using the current Generator

Step 2:
Train the Generator
to beat the Discriminator

Optimize: $\min_G \max_D \mathbb{E}_{P_X} \log D(x) + \mathbb{E}_{P_z} \log(1 - D(G(z)))$

The discriminator is not needed after convergence

Training GANs
Training Algorithm

Algorithm 1: Minibatch stochastic gradient descent training for GANs

for num_epochs **do:**

  for k_steps **do:**

    $\{z^{(1)} \ldots z^{(m)}\} \sim P_Z$ (Sample m noise vectors)

    $\{x^{(1)} \ldots x^{(m)}\} \sim P_X$ (Sample m data points)

    $L_D \leftarrow \frac{1}{m}\sum_{i=1}^{m}\left[logD\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$

    $g_{\theta_D} \leftarrow \nabla_{\theta_D} L_D$

    $\theta_D \leftarrow \theta_D + \alpha \cdot g_{\theta_D}$

  end for

  $\{z^{(1)} \ldots z^{(m)}\} \sim P_Z$ (Sample m noise vectors)

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

  $g_{\theta_G} \leftarrow \nabla_{\theta_G} L_G$

  $\theta_G \leftarrow \theta_G - \alpha \cdot g_{\theta_G}$

end for

Goodfellow et. al. (2014), Generative Adversarial Networks

Training GANs
Training Algorithm

Algorithm 1: Minibatch stochastic gradient decent training for GANs

for num_epochs **do:**

  for k_steps **do:**

    {z(1)… z(m)} ~ P_Z (Sample

    {x(1)… x(m)} ~ P_X (Sample m data points)

    $L_D \leftarrow \frac{1}{m}\sum_{i=1}^{m}\left[logD\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$

    $g_{\theta_D} \leftarrow \nabla_{\theta_D} L_D$

    $\theta_D \leftarrow \theta_D + \alpha \cdot g_{\theta_D}$

  end for

  {z(1)… z(m)} ~ P_Z (Sample m noise vectors)

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

  $g_{\theta_G} \leftarrow \nabla_{\theta_G} L_G$

  $\theta_G \leftarrow \theta_G - \alpha \cdot g_{\theta_G}$

end for

Hyperparameter.
Goodfellow et al. used k=1

In practice, this saturates early in training. We can instead maximize log(D(G(z))) for better gradients.

$$\min_G \max_D \mathbb{E}_{P_X} \log D(x) - \mathbb{E}_{P_z} \log(D(G(z)))$$

$$\min_G \max_D \mathbb{E}_{P_X} \log D(x) - \mathbb{E}_{P_G} \log(D(x))$$

Goodfellow et. al. (2014), Generative Adversarial Networks

Training GANs
Theorems

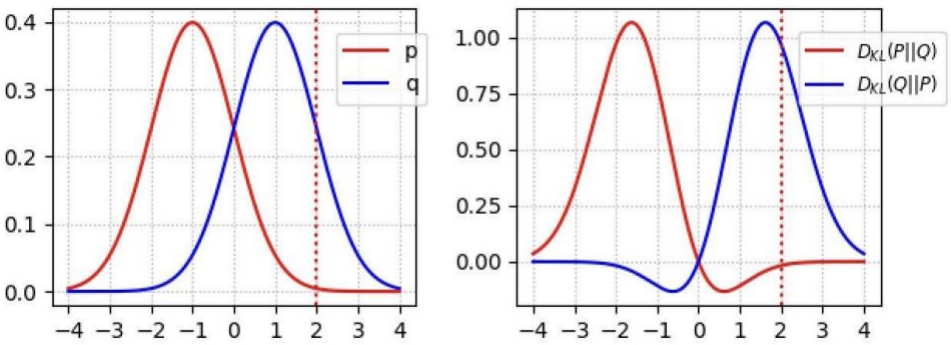
- $$L = 2D_{JS}\left(P_X(x), P_G(x)\right) - \log 4$$

Theorem 1: Global minima of the loss is achieved iff $P_X = P_G$ where $L = -log4$ and $D(x) = 1/2$

Theorem 2: if G and D have enough capacity, and at each step of **Algorithm 1**, the discriminator is allowed to reach its optimum given G, and P_G is updated so as to improve L, then P_G converges to P_X . [Proof in Section 4.2]

Goodfellow et. al. (2014), Generative Adversarial Networks

Training GANs
Distance Functions - KL



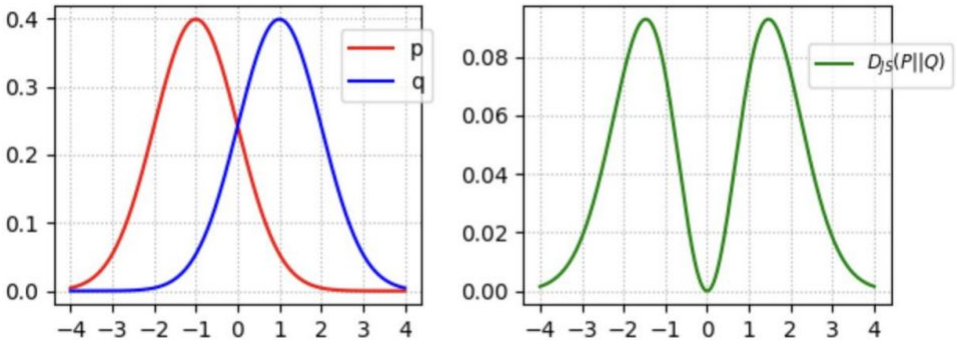
P: Real, X
Q: Fake, G

1. For $D_{KL}(P_X \,||\, P_G)$, red:
   1. High penalty (peak) where $P_X(x)$ is high but $P_G(x)$ is low.
   2. Low penalty where $P_X(x)$ is low but $P_G(x)$ is high.
      (It is fine to produce low quality images, but VERY BAD to drop image modes i.e. low diversity)

2. For $D_{KL}(P_G \,||\, P_X)$, blue:
   1. High penalty (peak) where $P_X(x)$ is low but $P_G(x)$ is high.
   2. Low penalty where $P_X(x)$ is high but $P_G(x)$ is low.
      (It is fine to produce samples that cover fewer image modes i.e. to have low diversity, but VERY BAD to have low quality)

Jonathan Hui, Why is it so hard to train GANs

Training GANs
Distance Functions - JS



$D_{JS}(P_X, P_G)$ was **supposed** to handle both issues: quality and diversity.

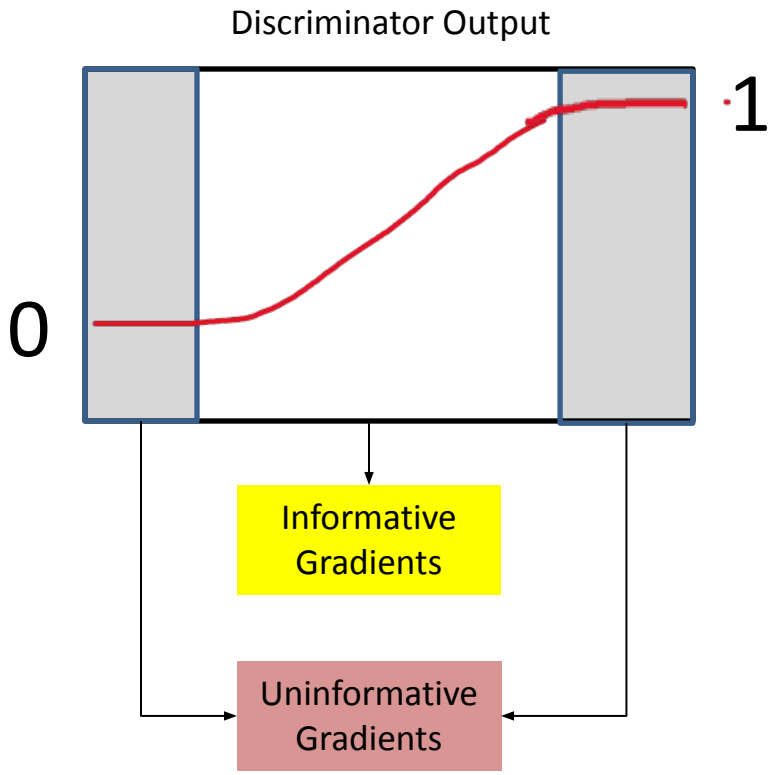
But it didn't quite happen.

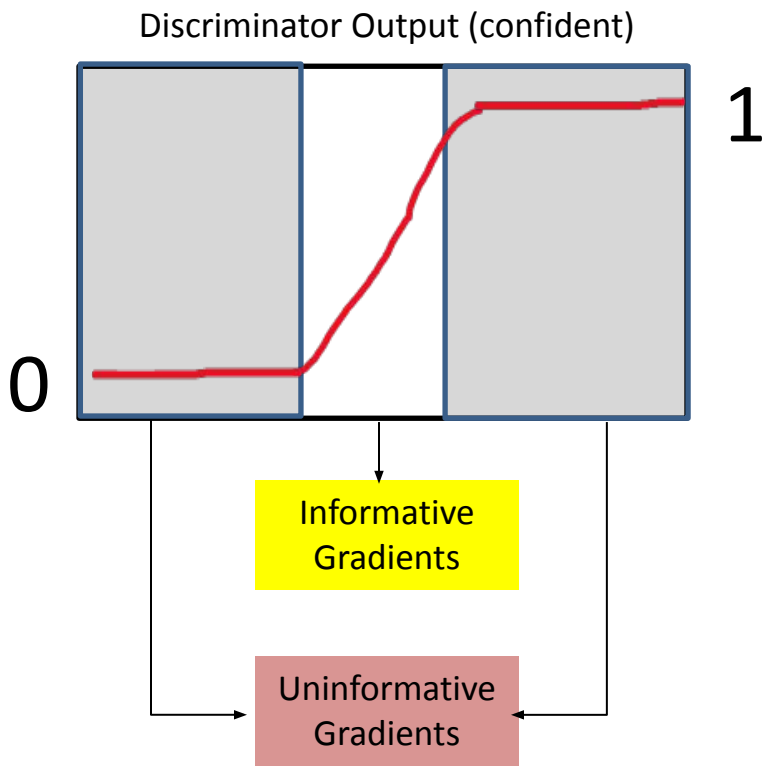
We need to look at what exactly causes this difficulty in training GANs, and why JS-Divergence is not a good enough metric for our purpose.

Jonathan Hui, Why is it so hard to train GANs

Difficulty in Training GANs

Issues with the Discriminator

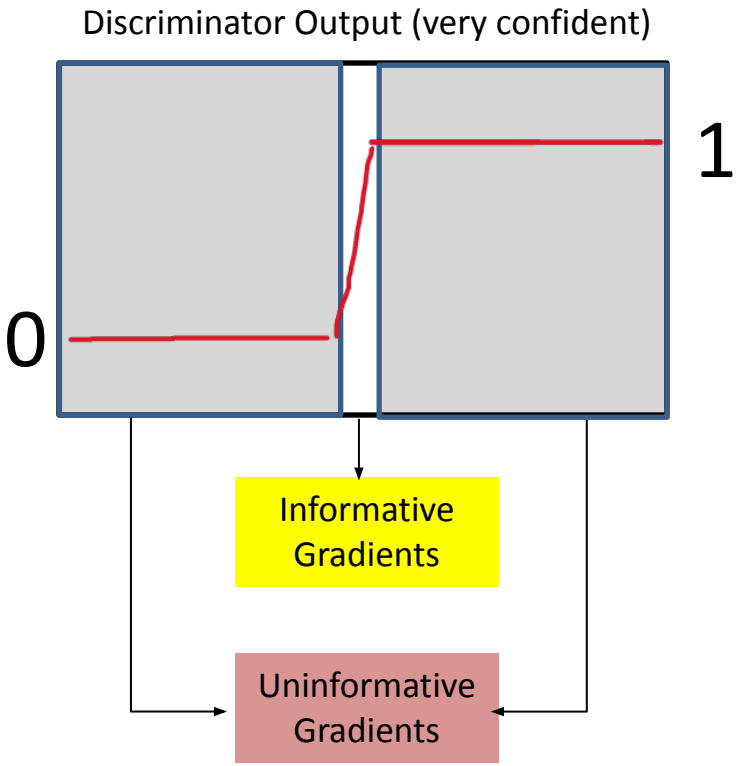


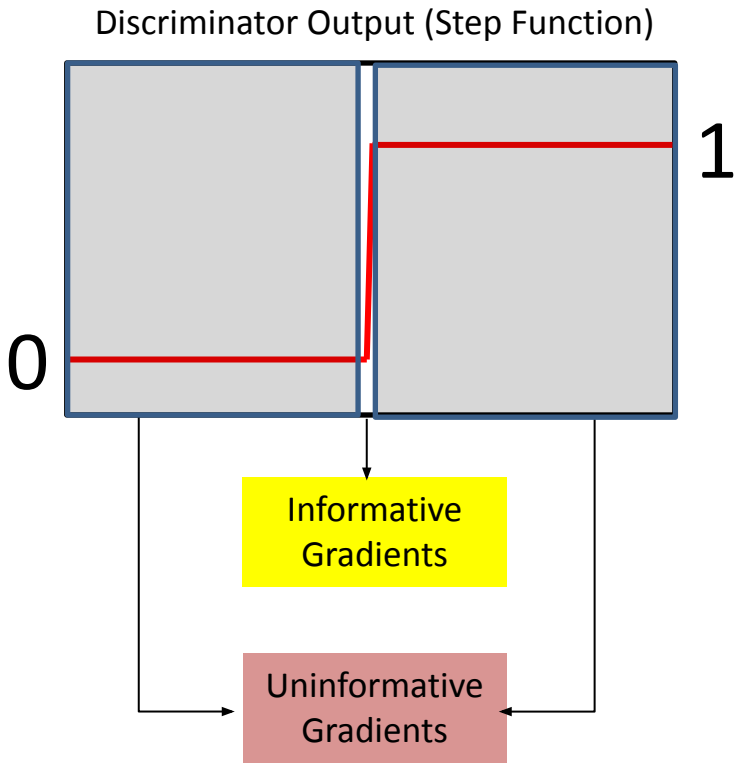


As the Discriminator gets more and more confident, the region that gives us informative gradients keeps shrinking.

Difficulty in Training GANs

Issues with the Discriminator





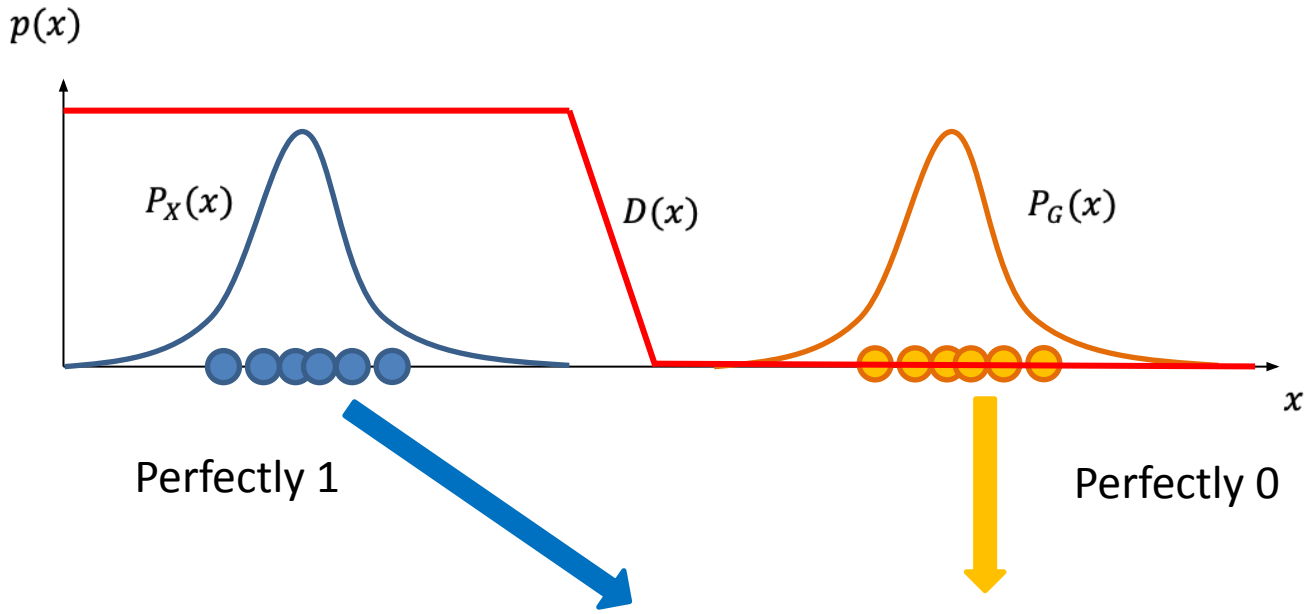
As the Discriminator gets more and more confident, the region that gives us informative gradients keeps shrinking.

… Till we eventually achieve a step-function which tells us nothing about how G should be changed for the desired effect on the model.

Difficulty in Training GANs

Uninformative Gradients

During early stages of training, when G is pretty-bad:



$$\min_G \max_D \mathbb{E}_{P_X} \log D(x) + \mathbb{E}_{P_z} \log(1 - D(G(z)))$$

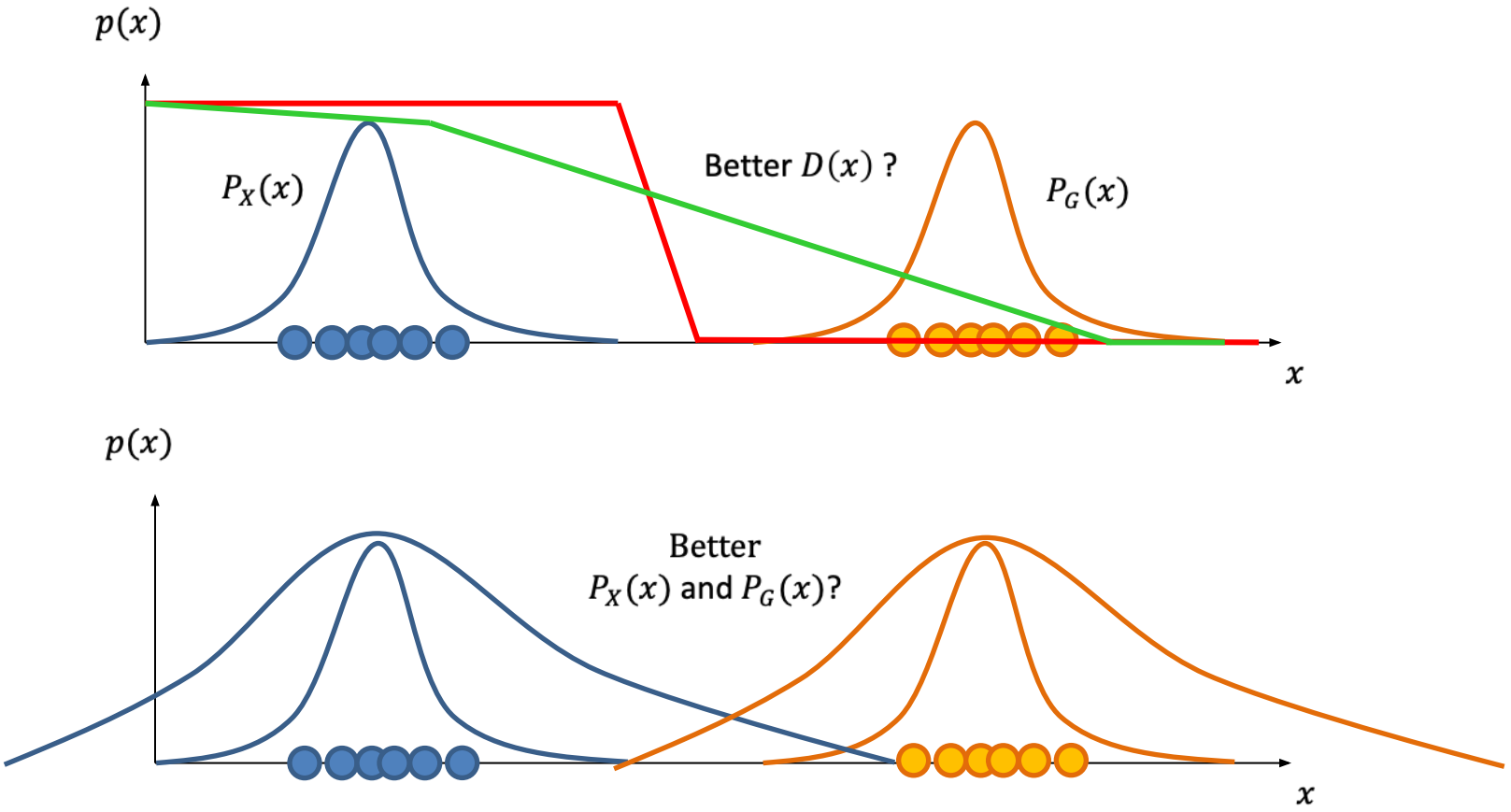
$$\approx \mathbb{E}_{P_X} \log(1) + \mathbb{E}_{P_z} \log(1)$$

Not very informative..

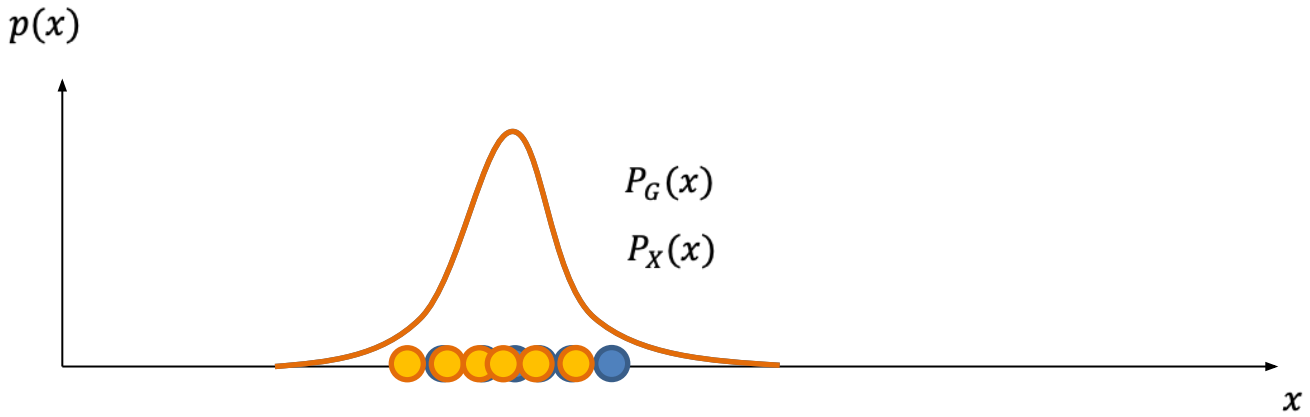
Difficulty in Training GANs

Uninformative Gradients

How can we improve on this?



What is the Goal?



Let us Consider the Worst case scenario,

Let there be 2 distributions P and Q such that they are only 1 for 1 value of x and 0 otherwise. Let $\boldsymbol{\theta}$ be the distance between the peaks of these distributions.

So Far,

VAE: KL Divergence

$$D_{\mathrm{KL}}(p(x)\,||\,q(x)) = \sum_{x \in X} p(x) ln \quad \frac{p(x)}{q(x)}$$

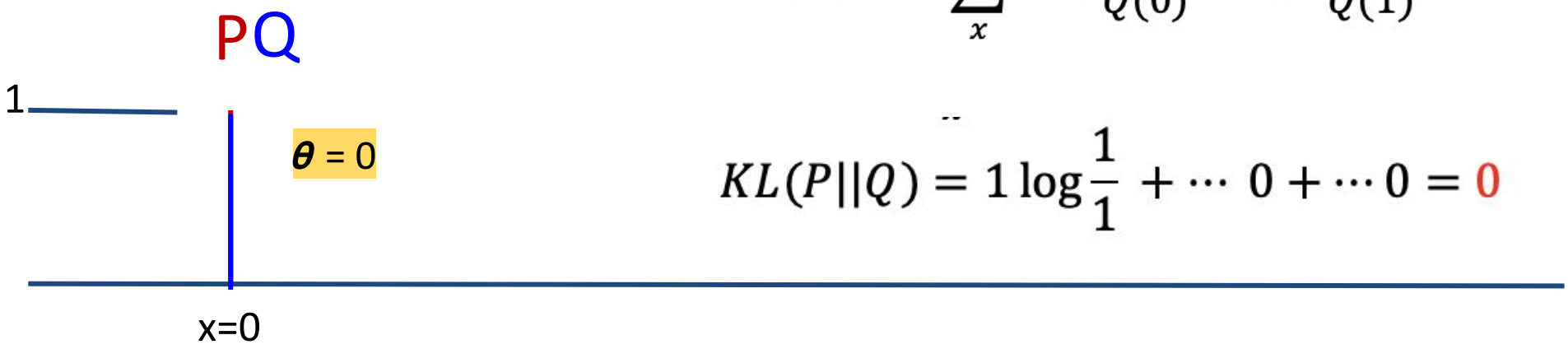
GAN: JSD

$$D_{JS}(p\|q) = \frac{1}{2} D_{KL}\left(p\|\frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q\|\frac{p+q}{2}\right)$$

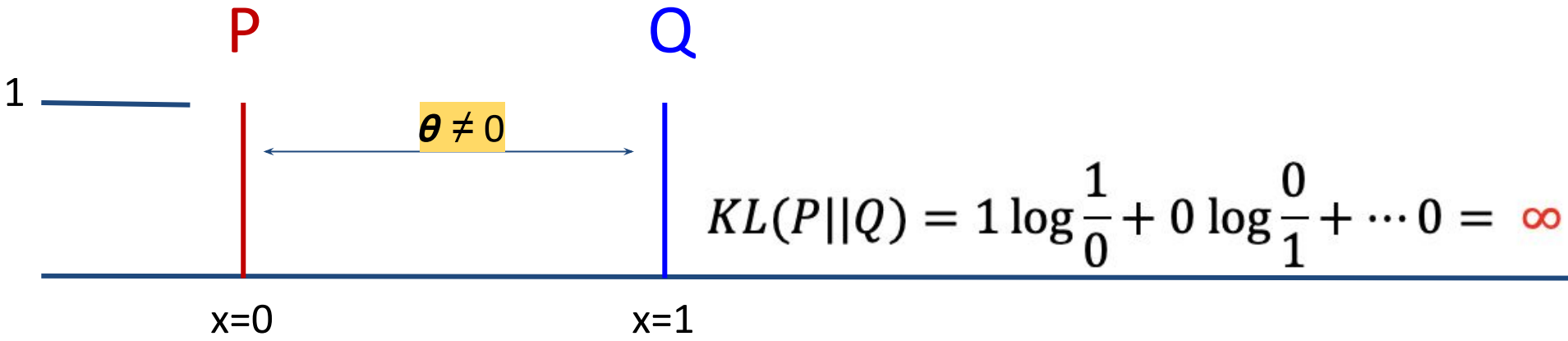
KL Divergence

$$KL(P||Q) = \sum_x \mathrm{P(x)} \frac{P(x)}{Q(x)}$$

$$KL(P||Q) = \sum_x \mathrm{P(0)} \frac{P(0)}{Q(0)} + \mathrm{P(1)} \frac{P(1)}{Q(1)} + \cdots$$



$$KL(P||Q) = 1 \log \frac{1}{1} + \cdots 0 + \cdots 0 = 0$$



$$KL(P||Q) = 1 \log \frac{1}{0} + 0 \log \frac{0}{1} + \cdots 0 = \infty$$

JS Divergence

$$m = \frac{P+Q}{2} \quad JSD(P||Q) = \frac{1}{2}\mathrm{KL}(\mathrm{P}||\mathrm{m}) + \frac{1}{2}\mathrm{KL}(\mathrm{Q}||\mathrm{m})$$

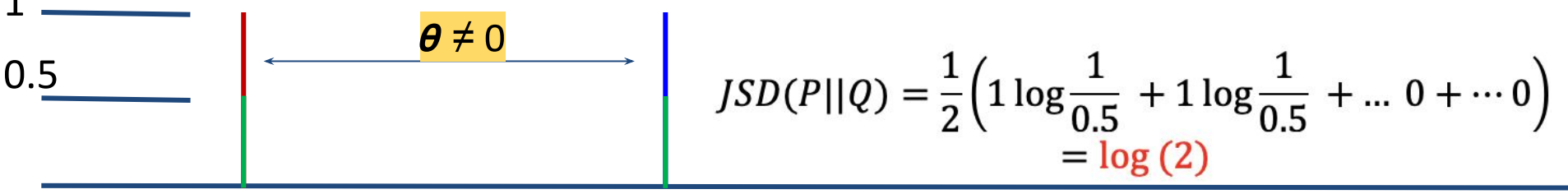
$$JSD(P||Q) = \frac{1}{2}\sum_x \mathrm{P}(0)\frac{P(0)}{m(0)} + \mathrm{P}(1)\frac{P(1)}{m(1)} + \cdots + \frac{1}{2}\sum_x \mathrm{Q}(0)\frac{Q(0)}{m(0)} + \mathrm{Q}(1)\frac{Q(1)}{m(1)} + \cdots$$

PQm

$$at\ x = 0, \quad m = \frac{1+1}{2} = 1$$

$$JSD(P||Q) = \frac{1}{2}(1\log\frac{1}{1} + \cdots\ 0 + \cdots 0) = 0$$

Pm Qm



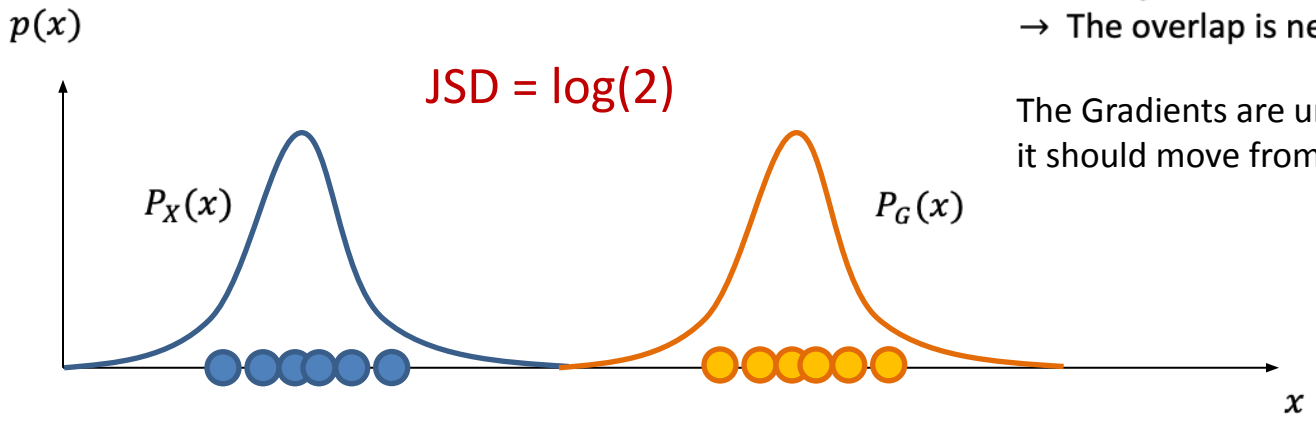
$$at\ x = 0, \quad m = \frac{1+0}{2} = 0.5$$

$$at\ x = 1, \quad m = \frac{0+1}{2} = 0.5$$

$$JSD(P||Q) = \frac{1}{2}\left(1\log\frac{1}{0.5} + 1\log\frac{1}{0.5} + \ldots\ 0 + \cdots 0\right)$$
$$= \log(2)$$

Difficulty in Training GANs

Troubles with JS Divergence



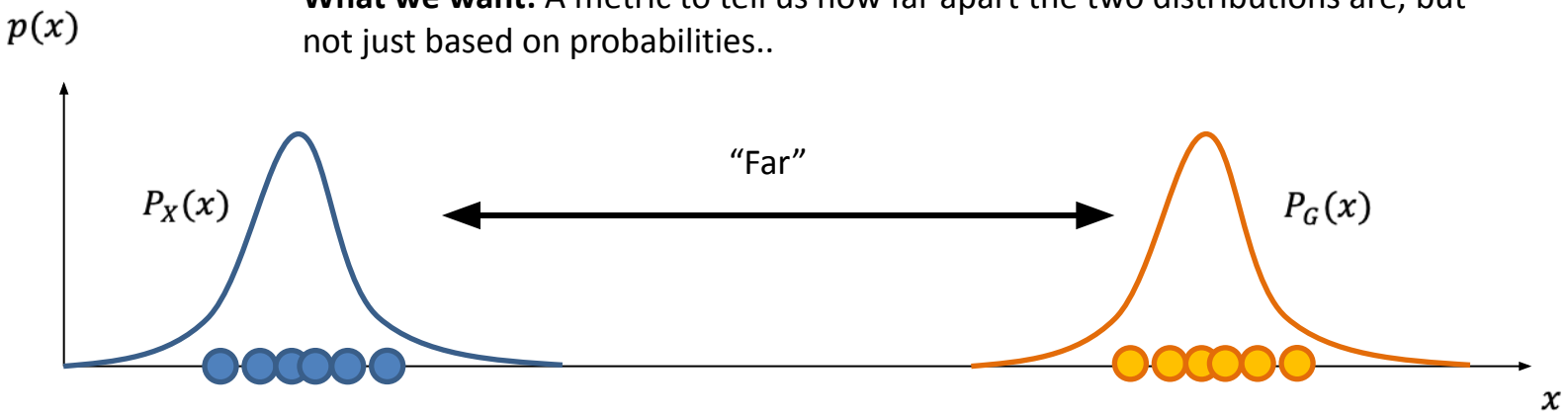
JS Divergence is nearly identical either way
→ The overlap is nearly 0.

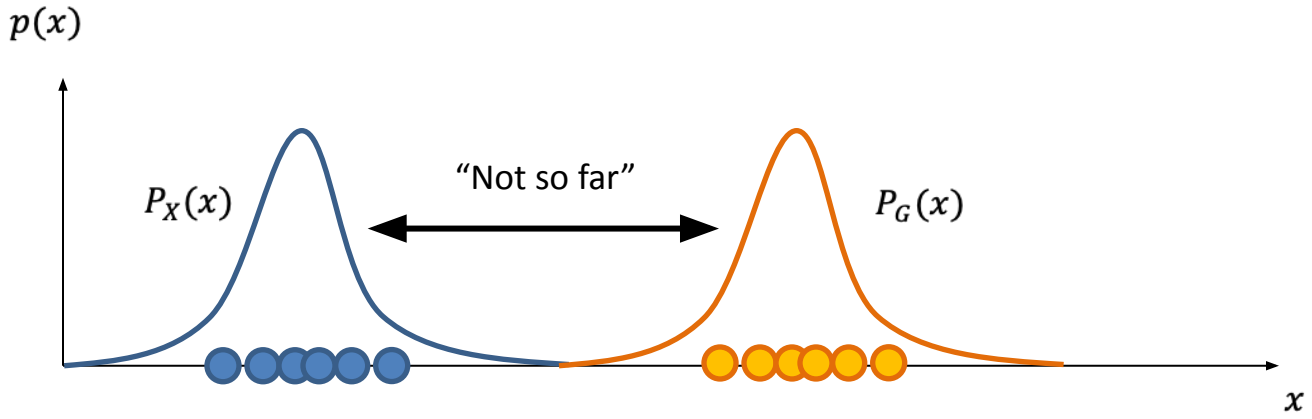
The Gradients are unable to tell the model that it should move from the first case to the second.

Improving GANs

A better distance metric than JSD

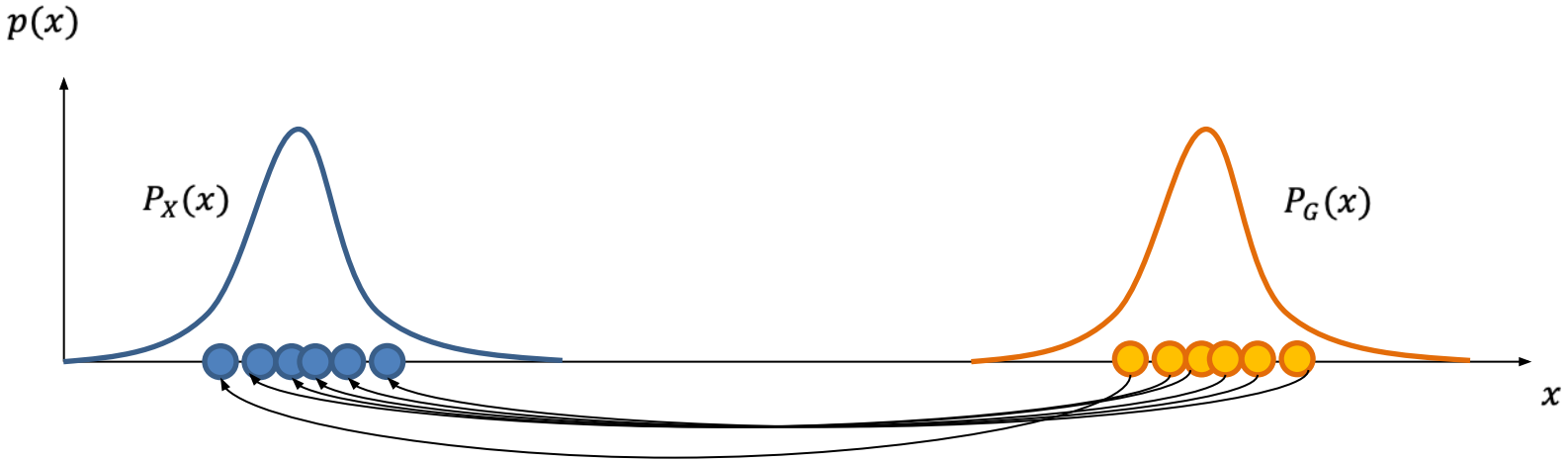
What we want: A metric to tell us how far apart the two distributions are, but not just based on probabilities..





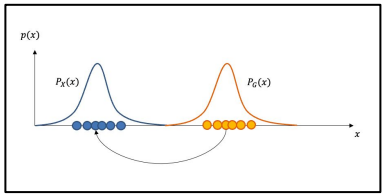
Improving GANs

A better distance metric than JSD



Imagine all the little points to be specs of dirt, and you want to move one pile of dirt over to another.

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?



Wasserstein GAN

Formulation

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \inf_{\gamma \in \Gamma(P_X, P_G)} \mathbb{E}_{(x,y)\sim\gamma(x,y)}[||x - y||]$$

Mapping of which x goes to which y

Distance between the points

Inf (Topic for real analysis):
Infimum, the closest thing to a lower bound you can get.
As far as we are concerned, an alternative to **min.**

Goal: Find the optimal mapping, or optimal plan, of moving the specs of dirt from one place to another.

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Formulation

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \inf_{\gamma \in \Gamma(P_X, P_G)} \mathbb{E}_{(x,y)\sim\gamma(x,y)}[||x - y||]$$

But…

1. We don't know what P_X is
2. γ could be super complex itself

Which make learning γ directly a very difficult problem.

This is where the Kantorovich-Rubinstein duality comes in:

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

Which looks a lot like the GANs we are familiar with!

$$\min_G \max_D \mathbb{E}_{P_X} \log D(x) - \mathbb{E}_{P_z} \log(D(G(z)))$$

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Formulation

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \inf_{\gamma \in \Gamma(P_X, P_G)} \mathbb{E}_{(x,y)\sim\gamma(x,y)}[||x - y||]$$

But…

1. We don't know what P_X is
2. γ could be super complex itself

Which make learning γ directly a very difficult problem.

This is where the Kantorovich-Rubinstein duality comes in:

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

Set of all 1-Lipschitz scalar functions:
$|f(x) - f(y)| \leq 1 \cdot |x - y|$

Basically, the function's slope is bounded by 1.

(K-Lipschitz is bounded by K)

Which looks a lot like the GANs we are familiar with!

$$\min_G \max_D \mathbb{E}_{P_X} \log D(x) - \mathbb{E}_{P_z} \log(D(G(z)))$$

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Formulation

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

How?
One way (not the best way) is to clip Weights:

1. One layer: $f(x;\theta) = ReLU(W_1 x + b_1)$
   if $W_{1,i,j} \in [-0.1, 0.1]$, then the slope can't be more than 0.1 (times the dimensionality)

2. Two layer: $f(x;\theta) = W_2 ReLU(W_1 x + b_1) + b_2$
   if $W_{1/2,i,j} \in [-0.1, 0.1]$, then the slope is still bounded (not 0.1 though)

We can't guarantee the function to be 1-Lipschitz, but some K-Lipschitz for some finite K.
And that's enough.

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Formulation

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

How?

1. Update D (θ_D) using gradient of $\mathbb{E}_{x \sim P_X}[D(x)] - \mathbb{E}_{z \sim P_Z}[D(G(z)]$
   (we interchange z and x in different equations for convenience, but they mean the same thing)

2. Clip all the weights in θ_D to some $[-c, c]$

3. Update G (θ_G) to minimize $-\mathbb{E}_{z \sim P_Z}[D(G(z)]$
   (or equivalently maximize $\mathbb{E}_{z \sim P_Z}[D(G(z)]$)

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Graphics

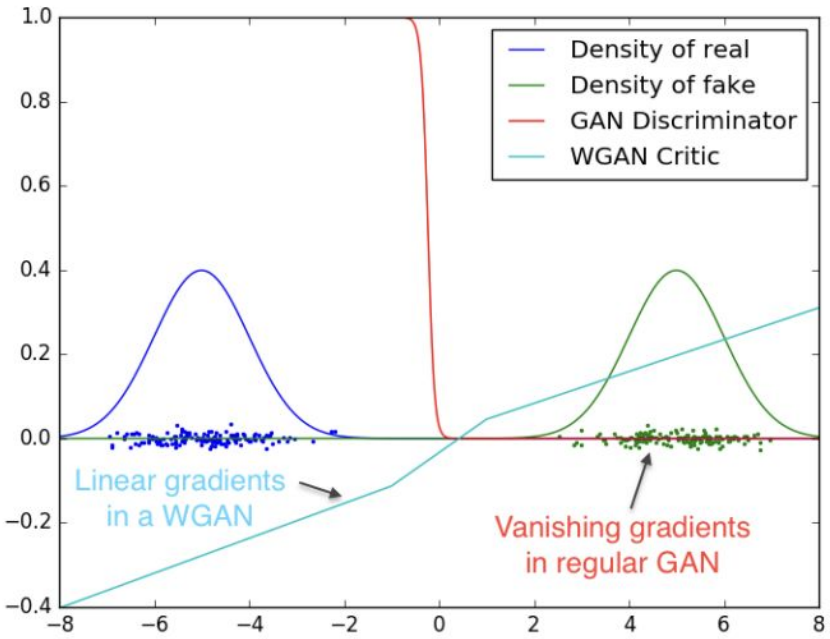


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Formulation

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

How?

1. Update D (θ_D) using gradient of $\mathbb{E}_{x \sim P_X}[D(x)] - \mathbb{E}_{z \sim P_Z}[D(G(z)]$
   (we interchange z and x in different equations for convenience, but they mean the same thing)

2. Clip all the weights in θ_D to some $[-c, c]$

3. Update G (θ_G) to minimize $-\mathbb{E}_{z \sim P_Z}[D(G(z)]$
   (or equivalently maximize $\mathbb{E}_{z \sim P_Z}[D(G(z)]$)

But
If the clipping parameter is too large, then it takes too long to train the Discriminator to optimality,
If too small, then the gradients vanish before reaching the generator.

Not the best way to do this..

Arjovsky et al, 2017, Wasserstein GAN

Wasserstein GAN

Gradient Penalty

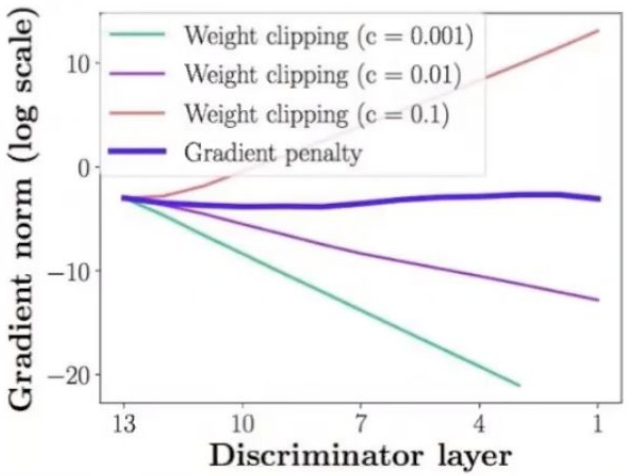
Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

What if: we instead directly attempt to include the Lipschitz constraint in the equation?

$$\mathbb{E}_{x \sim P_X}\left[D(x) - \lambda\left(\left|\left|\nabla_x D(x)\right|\right|_2 - 1\right)^2\right] - \mathbb{E}_{z \sim P_Z}[D(G(z)]$$

Basically: Push the norm of the gradient to be close to 1



Gulrajani et al, 2017, Improved Training of Wasserstein GANs

Wasserstein GAN

Spectral Norm

Earth mover's Distance or **Optimal Transport:**
How much distance you need to cover to move all the parts of one distribution to the other?

$$W(P_X, P_G) = \sup_{||f||_L \leq 1} \mathbb{E}_{P_X}[f(x)] - \mathbb{E}_{P_G}[f(x)]$$

$$W_l \leftarrow \frac{W_l}{\sigma(W_l)}$$

Where $\sigma(W_l)$ is the **spectral norm of** W_l

We won't go into details here, but it is sufficient to know that this is a more "principled" way to enforce the Lipschitz constraint on the weights.

It comes from the fact that the Lipschitz norm (the bound) for a composite function is upper bounded by the product of the Lipschitz norms of the composing functions.

Miyato et al, 2018, Spectral Normalization for Generative Adversarial Networks

Poll (@1704)

Which of the following is True:

A. KL Divergence can avoid exploding gradient problem.
B. Wasserstein Distance can fix vanishing gradient problem
C. JS Divergence can fix vanishing gradient problem
D. JS Divergence can avoid exploding gradient problem.

Poll (@1704)

Which of the following is True:

A. KL Divergence can avoid exploding gradient problem.
B. Wasserstein Distance can fix vanishing gradient problem
C. JS Divergence can fix vanishing gradient problem
D. JS Divergence can avoid exploding gradient problem.

GANs GANs Everywhere

Different Types of GANs

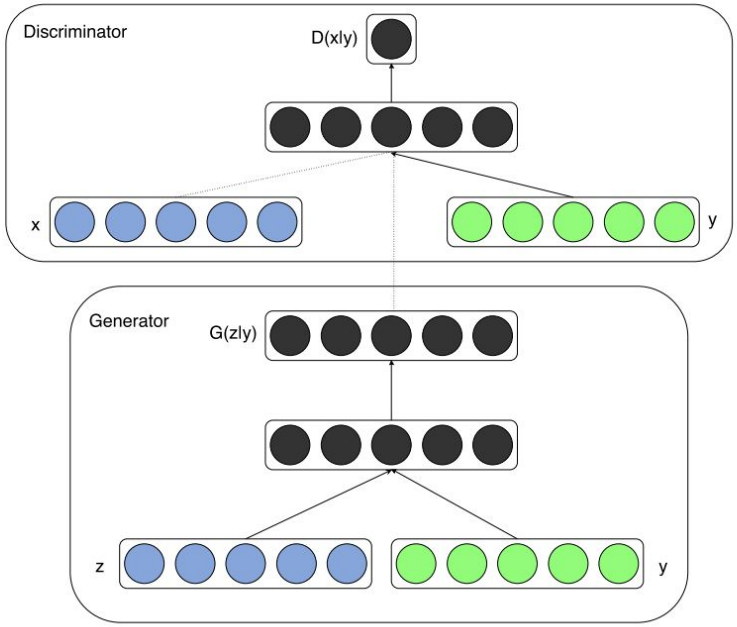
With the advent of WGANs and some other clever ways to give us easier training for GANs, there were several types of GANs proposed in literature:

1. Conditional GAN
2. LapGAN (Laplacian GAN, stacked Conditional GANs)
3. BiGAN (Involves both $z \rightarrow \hat{x}$ and $\hat{x} \rightarrow z$)
4. Recurrent Adversarial Network
5. Categorical GAN
6. InfoGAN
7. AAE
8. STAR-GAN
9. Pix2Pix
10. CycleGAN

Conditional GAN

What is it?

- Provides more control in what type of images are generated by the generator
- Are not strictly unsupervised learning algorithms because they require labeled data as input to the additional layer
- Allows us to generate the outputs for exactly the classes or labels that we desire



Mirza & Osindero, 2014, **Conditional Generative Adversarial Nets**

Conditional GAN

Injecting extra information

Mathematical expression for a simple GAN -

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

Mathematical expression for a Conditional GAN -

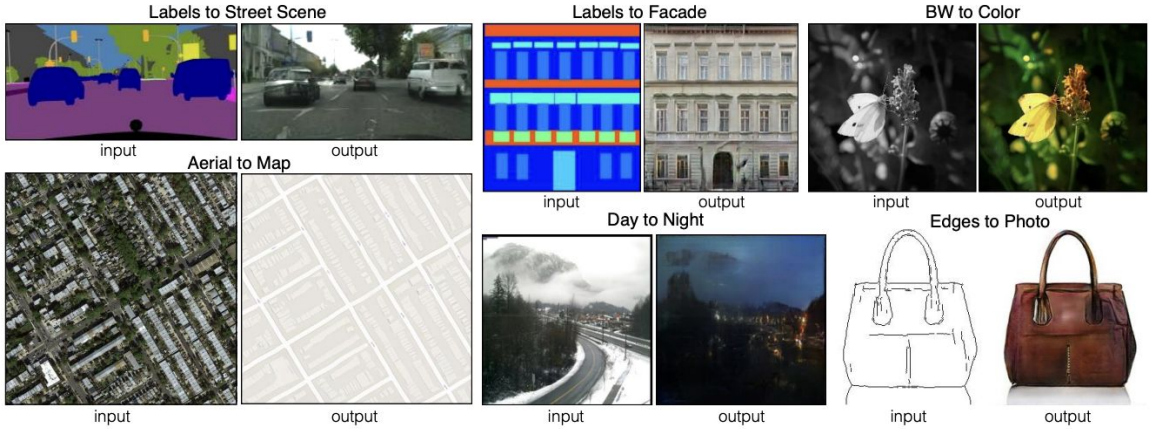
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

Implementing Conditional Generative Adversarial Networks (paperspace.com)

Conditional GAN

Applications

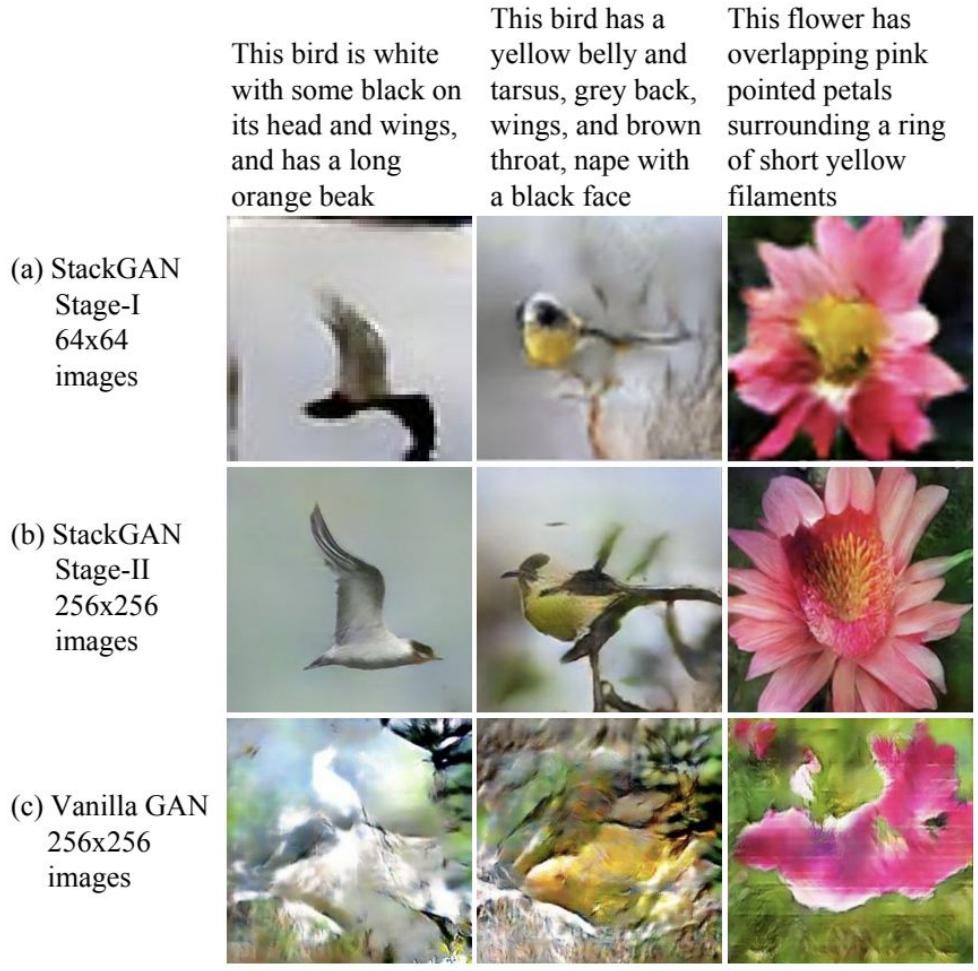
Image-to-image translation



https://arxiv.org/pdf/1611.07004.pdf

Conditional GAN

Applications



Text to image synthesis

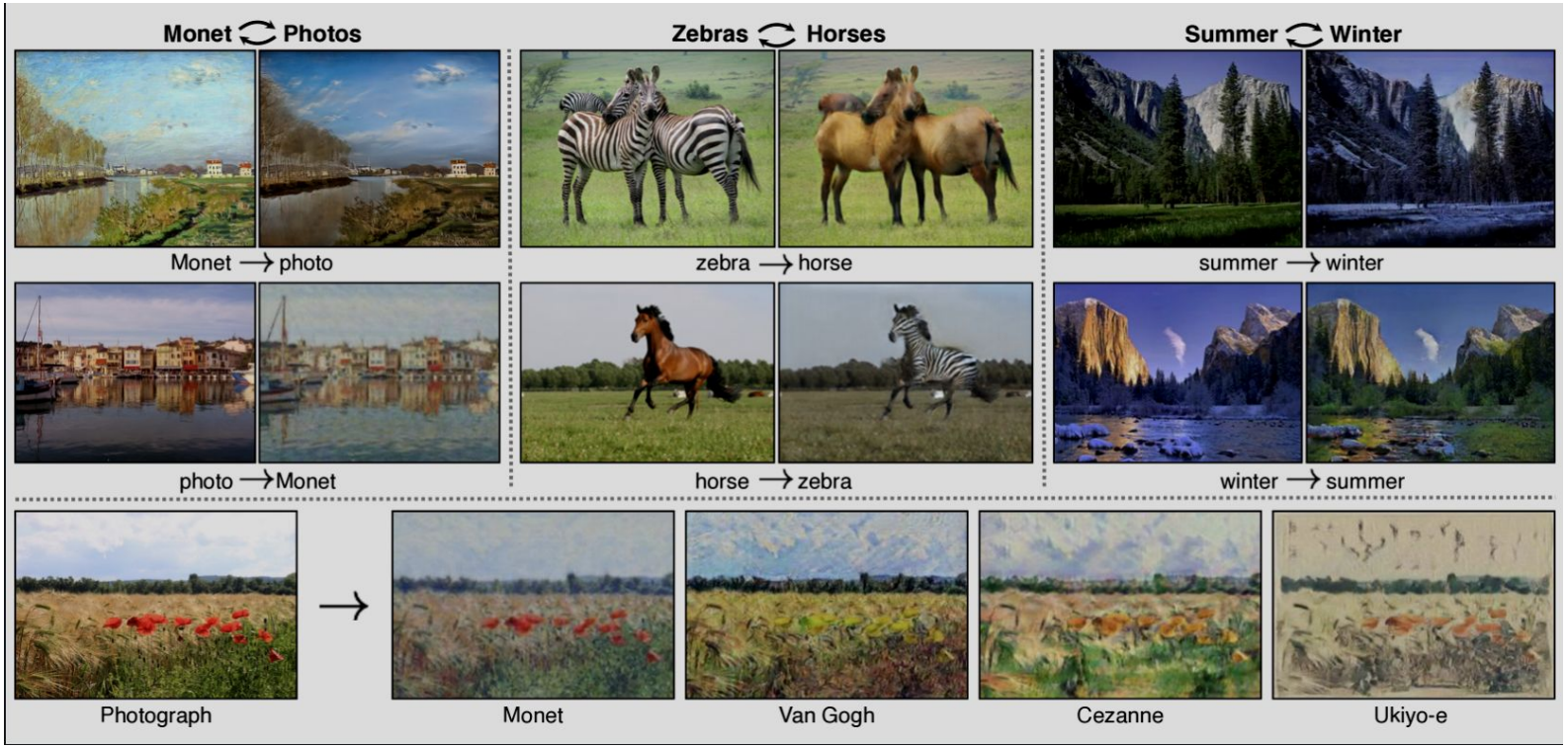
Figure 1. Comparison of the proposed StackGAN and a vanilla one-stage GAN for generating 256×256 images. (a) Given text descriptions, Stage-I of StackGAN sketches rough shapes and basic colors of objects, yielding low-resolution images. (b) Stage-II of StackGAN takes Stage-I results and text descriptions as inputs, and generates high-resolution images with photo-realistic details. (c) Results by a vanilla 256×256 GAN which simply adds more upsampling layers to state-of-the-art GAN-INT-CLS [26]. It is unable to generate any plausible images of 256×256 resolution.

https://arxiv.org/pdf/1612.03242.pdf

Cycle GAN

Translating Images

GANs can be used for much more than image generation.
By clever training, we can even use them for image translation.



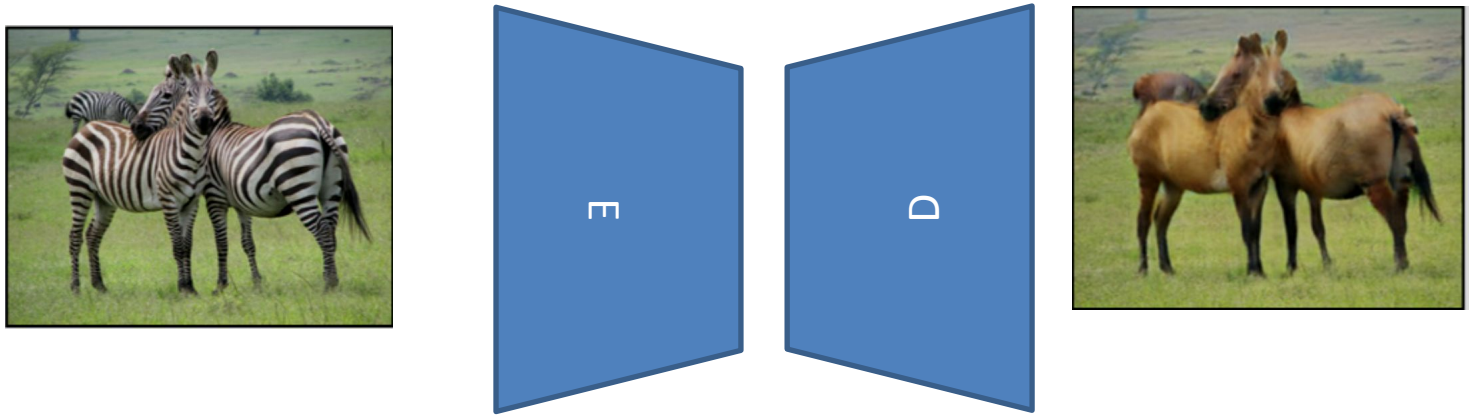
Zhu et al, 2017, **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks**

Why CycleGAN?

Lets try methods we have discussed so far

Cycle GAN

Autoencoder Alternative



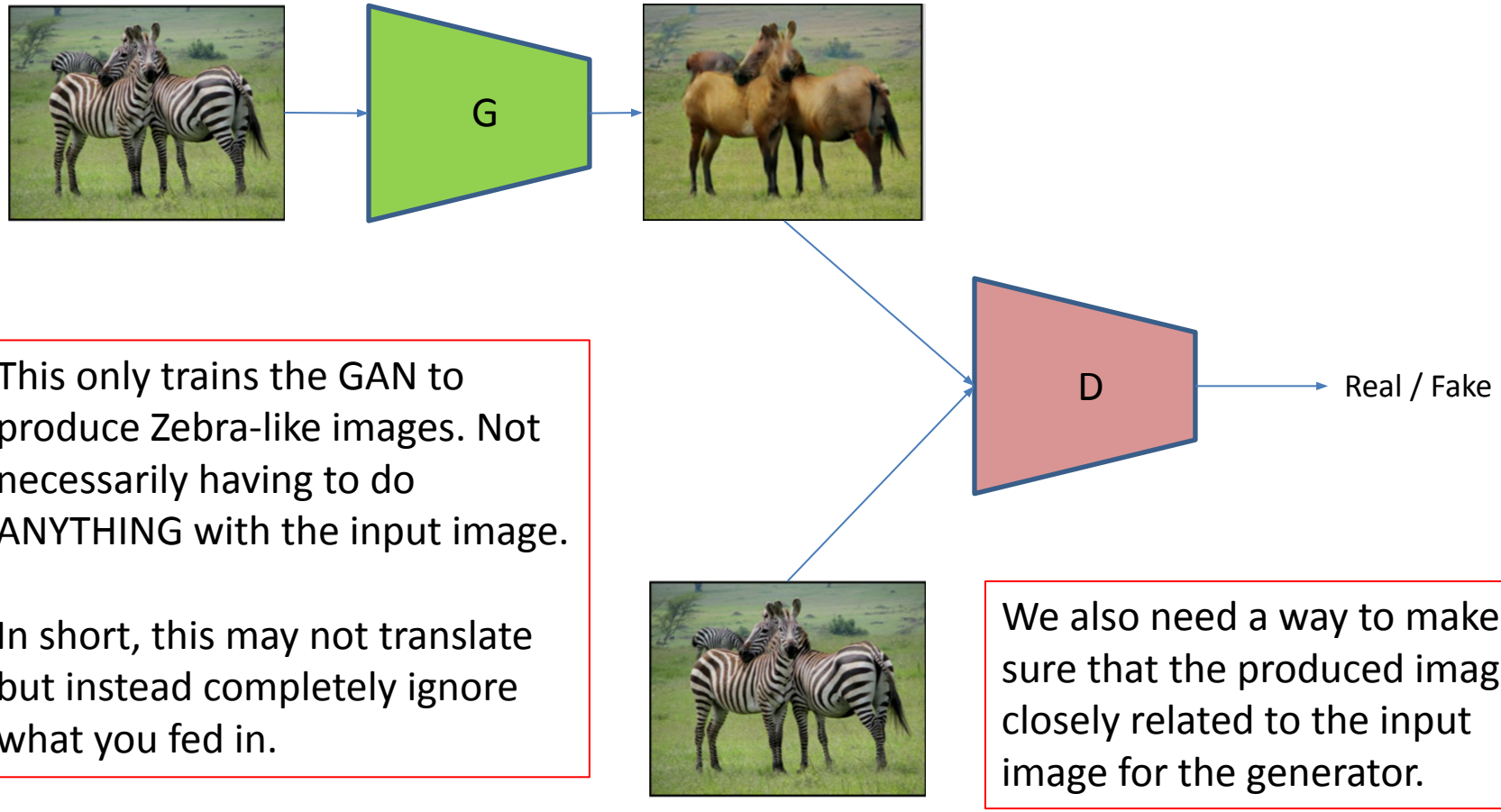
Unless you have paired images, achieving the above with a simple Autoencoder is simply not possible.

So, what if we just threw a GAN (or a WGAN) at it?

Zhu et al, 2017, **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks**

Cycle GAN

Regular GAN Alternative



This only trains the GAN to produce Zebra-like images. Not necessarily having to do ANYTHING with the input image.

In short, this may not translate but instead completely ignore what you fed in.

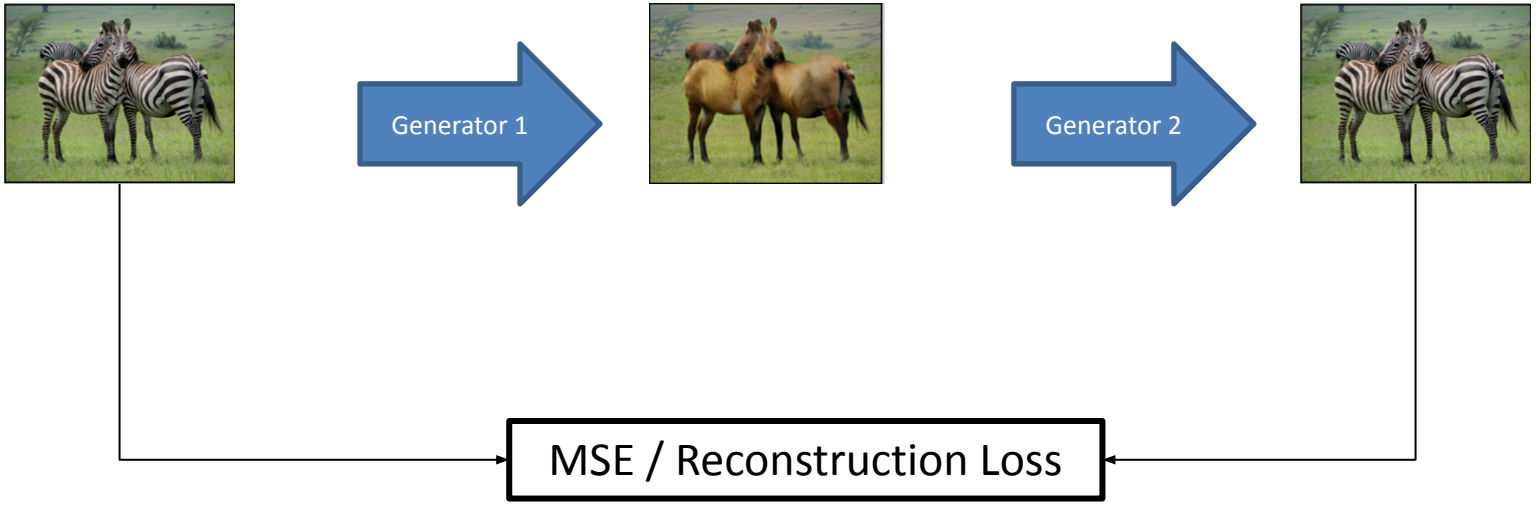
We also need a way to make sure that the produced image is closely related to the input image for the generator.

Zhu et al, 2017, **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks**

Cycle GAN

Cycle consistency Loss

This is where **Cycle GANs** come in.
On top of the regular GAN training to ensure that the generated image looks like a zebra, the cycle consistency loss (just a reconstruction loss) ensures that the generated image has retained enough information about the input enough to retrieve it using a different “special purpose” generator.

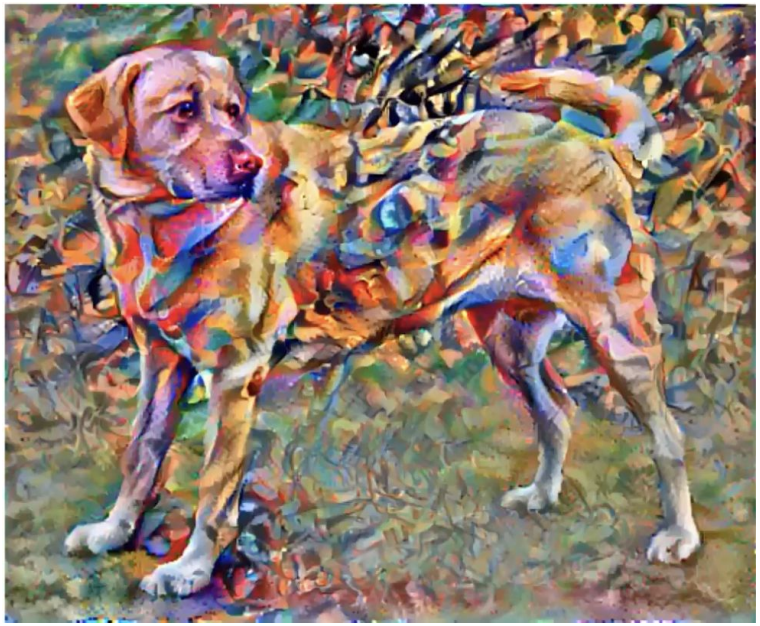


Zhu et al, 2017, **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks**

What we learn from CycleGAN

- There is no paired dataset of Zebras and Horses
- So there is no easy discriminative method to train Zebras from Horses
- But using GANs, we can train distributions to match.

Neural Style Transfer Results



Style transfer with CycleGAN

Comparing

- Neural Style Transfer
  - Need a content and style image
  - Specific & small number of images
  - More control
  - https://reinakano.com/arbitrary-image-stylization-tfjs/
- CycleGAN
  - Just need 2 domains of images. No need for specific content or style images
  - Many similar pictures
  - Specificity of images doesn't really matter

Comparing

For specific and small no. of images

- Neural Style Transfer
  - Need a content and style image
  - Specific & small number of images
  - More control
  - https://reinakano.com/arbitrary-image-stylization-tfjs/
- CycleGAN
  - Just need 2 domains of images. No need for specific content or style images
  - Many similar pictures
  - Specificity of images doesn't really matter

Comparing

- Neural Style Transfer
  - Need a content and style image
  - Specific & small number of images
  - More contro
  - https://reina stylization-tfjs/
- CycleGAN
  - Just need 2 domains of images. No need for specific content or style images
  - Many similar pictures
  - Specificity of images doesn't really matter

For specific and small no. of images

Specificity of images do not really matter

StarGAN

Generalization of CycleGAN

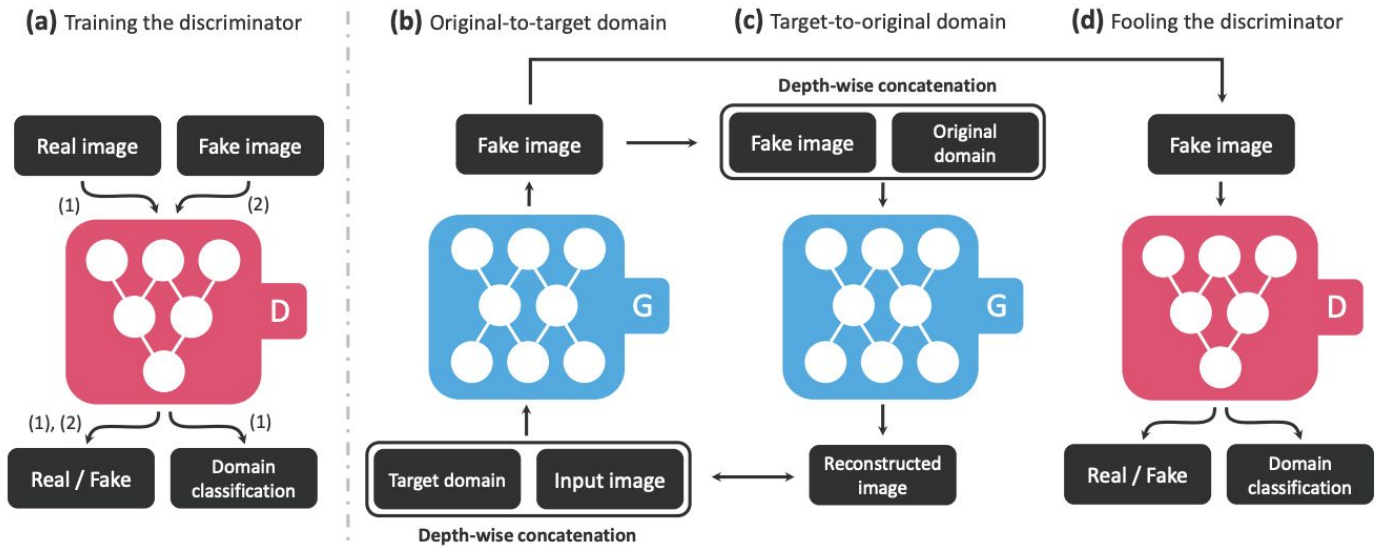


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G. **(a)** D learns to distinguish between real and fake images and classify the real images to its corresponding domain. **(b)** G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. **(c)** G tries to reconstruct the original image from the fake image given the original domain label. **(d)** G tries to generate images indistinguishable from real images and classifiable as target domain by D.

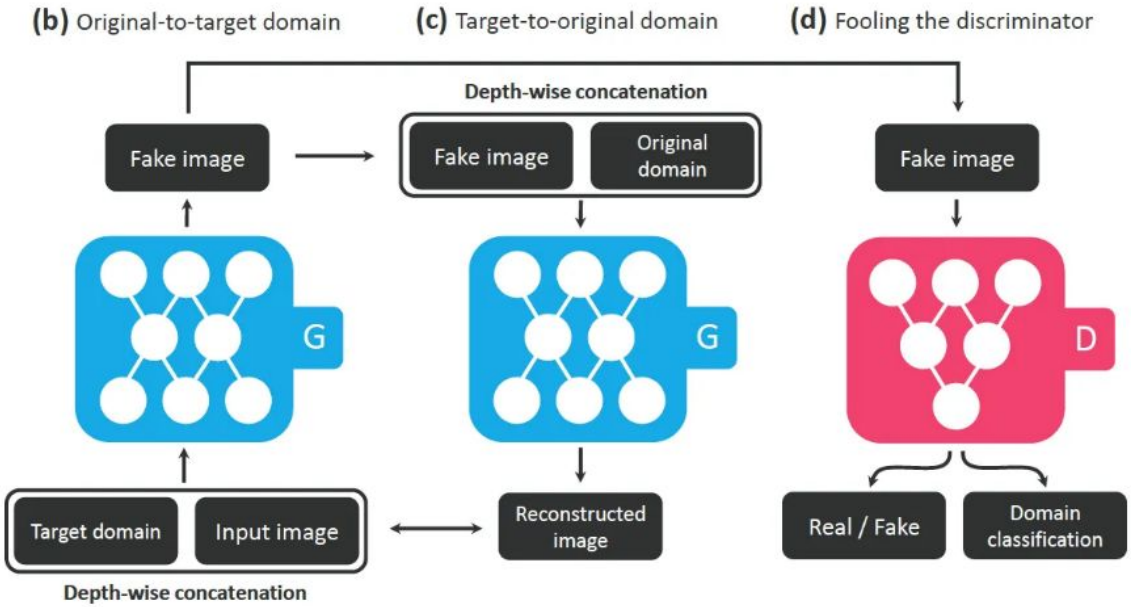
Given training data from two different domains, StarGANs learn to translate images from one domain to the other

For Example — changing the hair color (attribute) of a person from black (attribute value) to blond (attribute value).

Choi et al, 2017, **StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation**

StarGAN

How does it work?

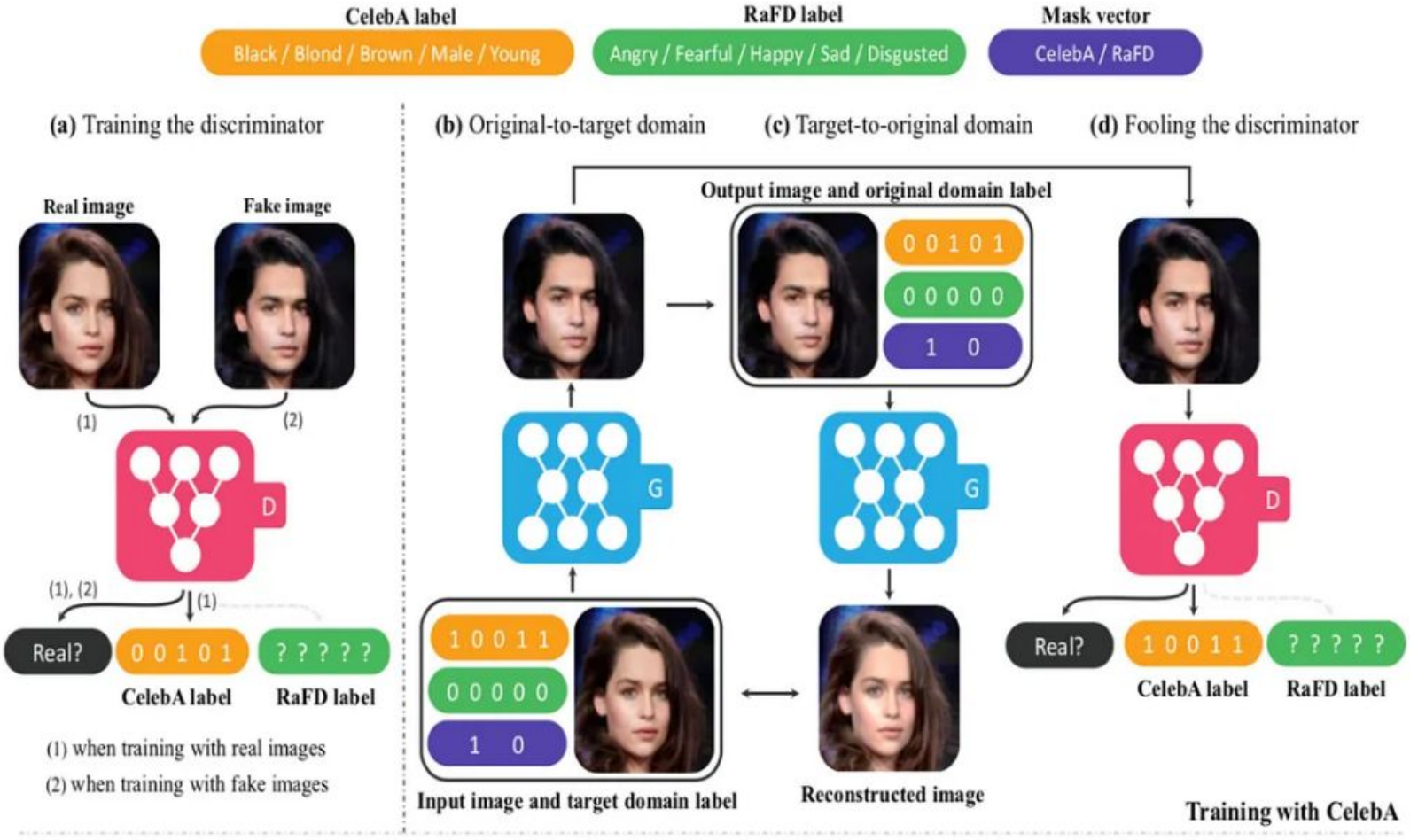


- G takes in as input both the image and target domain label and generates an fake image.
- It then tries to reconstruct the original image from the fake image conditioned on the original domain label.
- Discriminator tells if image is fake and classifies an image to its corresponding domain, so G will ultimately learn to generate realistic images corresponding to the given target domain. (d)

Choi et al, 2017, **StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation**

StarGAN

Image-to-Image Translation



Deep Learning Notes: StarGAN. From the perceived daunting task of… | by ashwin bhat | Medium

StarGAN

Objectives of StarGAN

Objectives of Discriminator:

1. Identify whether an image is fake or not.
2. Predict the target domain of the input image.
    - Uses an auxiliary classifier to learn the mapping of original image and its corresponding domain from the dataset

Choi et al, 2017, StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation

StarGAN

Objectives of StarGAN

Objectives of the Generator:

1. The images generated are realistic.
2. The images generated are classifiable as target domain by D.
3. Able to reconstruct the original image from the fake image given the original domain label. (Cycle consistency Loss)

Choi et al, 2017, StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation

BiGAN

Adversarial Feature Learning

A Bi-Directional Generative Adversarial Network trains an encoder/decoder pair in an elegant fashion. The discriminator tries to tell the difference between pairs of real data and encoded real data from data generated from prior samples and prior samples. [DKD16]

The BiGAN training objective is defined as a minimax objective

$$\min_{G,E} \max_{D} V(D, E, G) \tag{2}$$

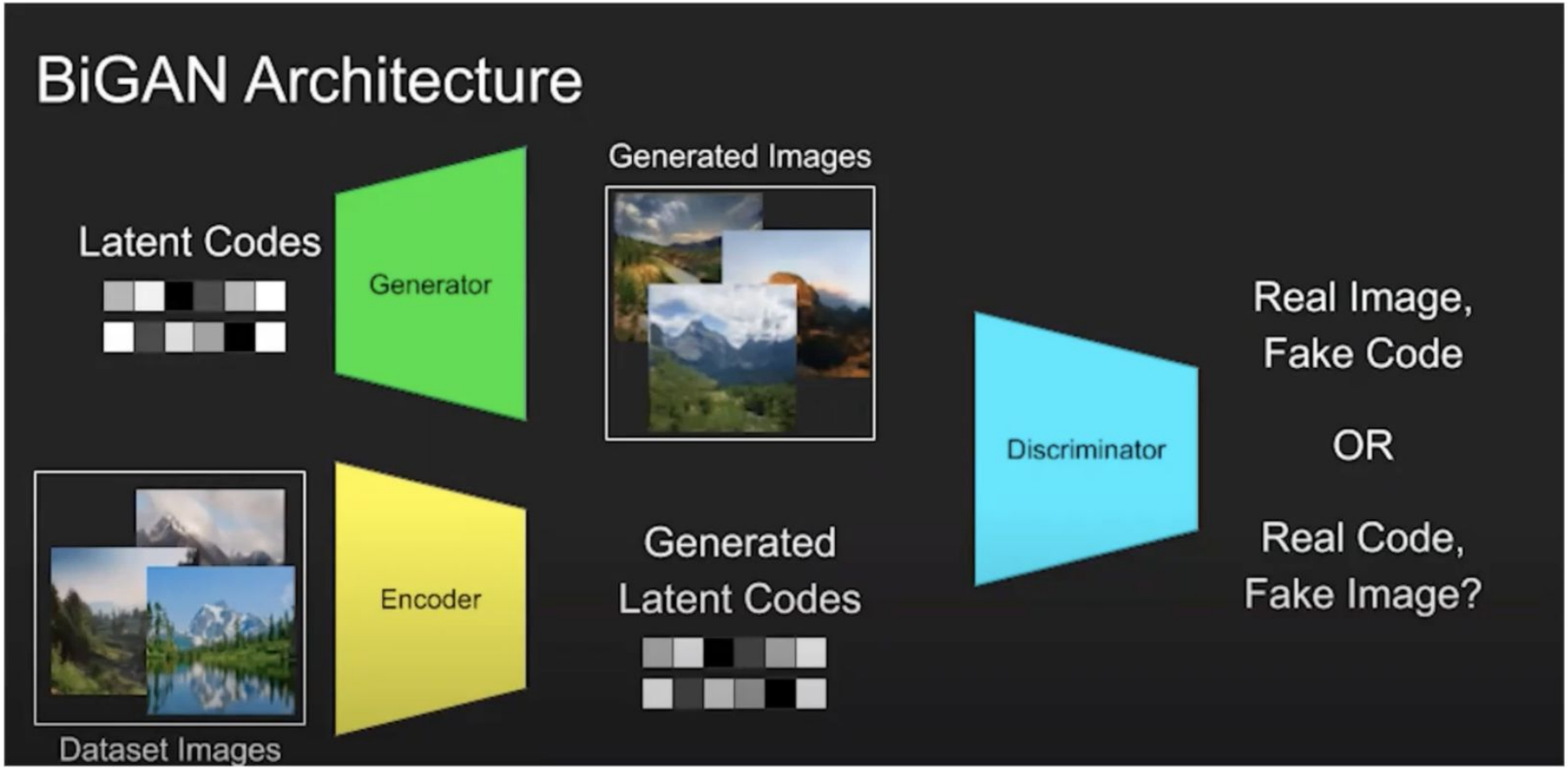
where

$$V(D,E,G) := \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{X}}}\Big[\underbrace{\mathbb{E}_{\mathbf{z}\sim p_E(\cdot|\mathbf{x})}\left[\log D(\mathbf{x},\mathbf{z})\right]}_{\log D(\mathbf{x},E(\mathbf{x}))}\Big] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}}\Big[\underbrace{\mathbb{E}_{\mathbf{x}\sim p_G(\cdot|\mathbf{z})}\left[\log\left(1-D(\mathbf{x},\mathbf{z})\right)\right]}_{\log(1-D(G(\mathbf{z}),\mathbf{z}))}\Big]. \tag{3}$$

Donahue et al, 2016, Adversarial Feature Learning

BiGAN

Adversarial Feature Learning



Donahue et al, 2016, Adversarial Feature Learning

LAPGAN

Stacking CGANs

A Laplacian GAN is constructed of a chain of conditional GANs, to generate progressively larger images. A GAN generates small, blurry images. A conditional GAN generates larger images conditioned on the smaller image, repeated until you reach the desired size.[DCSF15]

Denton et al, 2015, **Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**

LAPGAN

Stacking CGANs

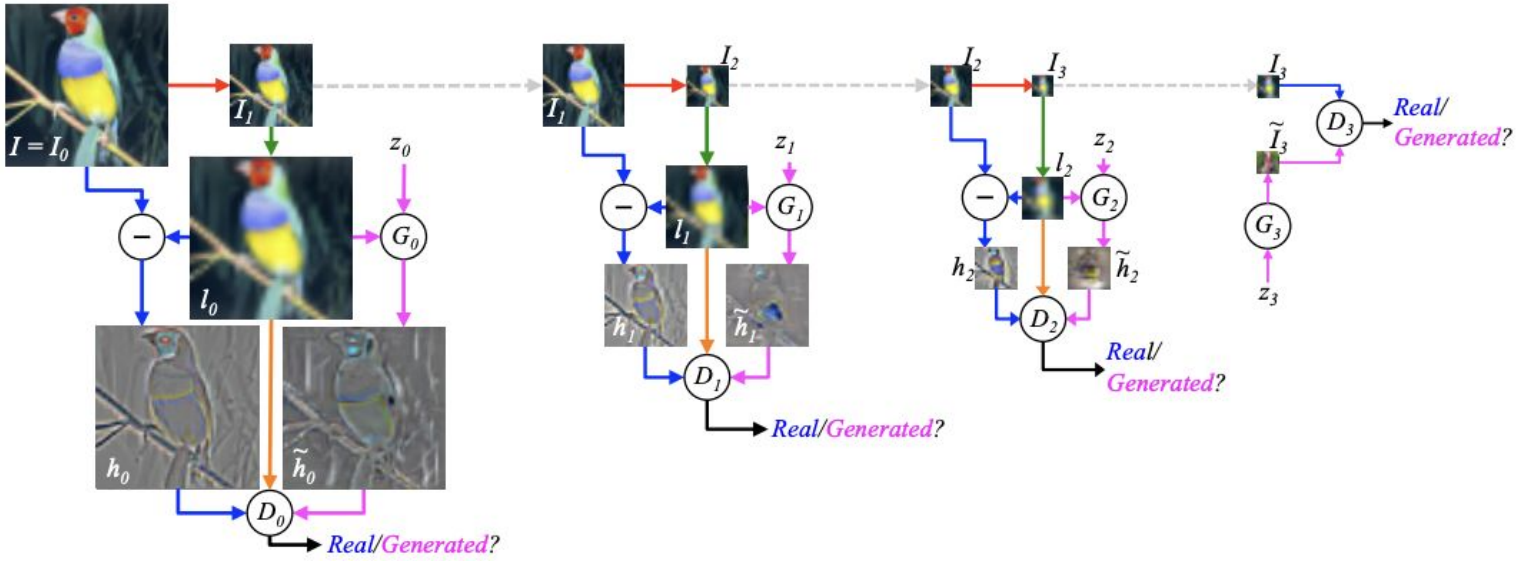


Figure 2: The training procedure for our LAPGAN model. Starting with a 64x64 input image I from our training set (top left): (i) we take $I_0 = I$ and blur and downsample it by a factor of two (red arrow) to produce I_1; (ii) we upsample I_1 by a factor of two (green arrow), giving a low-pass version l_0 of I_0; (iii) with equal probability we use l_0 to create *either* a real *or* a generated example for the discriminative model D_0. In the real case (blue arrows), we compute high-pass $h_0 = I_0 - l_0$ which is input to D_0 that computes the probability of it being real vs generated. In the generated case (magenta arrows), the generative network G_0 receives as input a random noise vector z_0 and l_0. It outputs a generated high-pass image $\tilde{h}_0 = G_0(z_0, l_0)$, which is input to D_0. In both the real/generated cases, D_0 also receives l_0 (orange arrow). Optimizing Eqn. 2, G_0 thus learns to generate realistic high-frequency structure $\tilde{h}_0$ consistent with the low-pass image l_0. The same procedure is repeated at scales 1 and 2, using I_1 and I_2. Note that the models at each level are trained independently. At level 3, I_3 is an 8×8 image, simple enough to be modeled directly with a standard GANs G_3 & D_3.

Denton et al, 2015, Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

GANs

And Many More…

Summary

- Training GANs is difficult for a lot of reasons
- Primarily, the issues of the discriminator capturing the wrong boundary, being too confident (and non differentiable), and getting saturated due to the formulation of the objective.
- We can remedy this by using Earth mover's metric – Wasserstein Distance
- The Lipschitz constraint of WGANs can be imposed in several ways including weight clipping, gradient penalties, and spectral norm.
- There are (so) many variants of GANs which do more than just image generation and are able to do image translation (or style transfer) as well!

References

- StarGAN — Image-to-Image Translation | by Pranoy Radhakrishnan | Towards Data Science
- Deep Learning Notes: StarGAN. From the perceived daunting task of… | by ashwin bhat | Medium