Generative Adversarial Networks

Benjamin Striner[1]

[1]Carnegie Mellon University

April 20, 2020

Table of Contents

Table of Contents

Overview

Generative Adversarial Networks (GANs) are a powerful and flexible tool for generative modeling

- What is a GAN?
- How do GANs work theoretically?
- What kinds of problems can GANs address?
- How do we make GANs work correctly in practice?

Motivation

Generative networks are used to generate samples from an unlabeled distribution $P(X)$ given samples $X_1, \ldots, X_n$. For example:

- Learn to generate realistic images given exemplary images
- Learn to generate realistic music given exemplary recordings
- Learn to generate realistic text given exemplary corpus

Great strides in recent years, so we will start by appreciating some end results!

GANs (2014)

Output of original GAN paper, 2014 [GPM+14]

4.5 Years of Progress

GAN quality has progressed rapidly

https://twitter.com/goodfellow_ian/status/1084973596236144640?lang=en

Large Scale GAN Training for High Fidelity Natural Image Synthesis (2019)

Generating High-Quality Images [BDS18]

Figure 6: Samples generated by our BigGAN model at 512×512 resolution.

StarGAN (2018)

Manipulating Celebrity Faces [CCK+17]

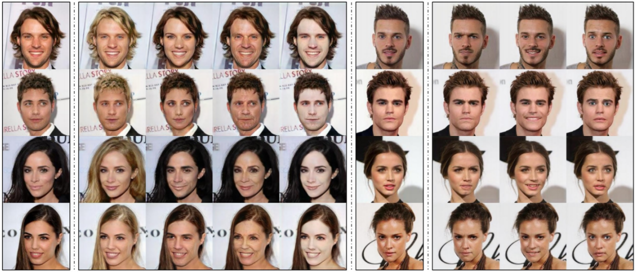


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator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

Progressive Growing of GANs (2018)

Generating new celebrities and a pretty cool video

https://www.youtube.com/watch?v=XOxxPcy5Gr4 [KALL17]

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

Unsupervised Image to Image Translation (2018)

Changing the weather

https://www.youtube.com/watch?v=9VC0c3pndbI [LBK17]

Table of Contents

Generative vs. Discriminative Networks

Given a distribution of inputs X and labels Y

- Discriminative networks model the conditional distribution $P(Y \mid X)$.
- Generative networks model the joint distribution $P(X, Y)$.

Why Generative Networks?

- Model understands the joint distribution $P(X, Y)$.
  - Can calculate $P(X \mid Y)$ using Bayes rule.
  - Can perform other tasks like $P(X \mid Y)$, generating data from the label.
  - "Deeper" understanding of the distribution than a discriminative model.
- If you only have X, you can still build a model. Many ways to leverage unlabeled data. Not every problem is discriminative.
- However, model for $P(X, Y)$ is harder to learn than $P(Y \mid X)$
  - Map from X to Y is typically many to one
  - Map from Y to X is typically one to many
  - Dimensionality of X typically $>>$ dimensionality of Y

Traditional Viewpoint

When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one.

Vapnik 1995

Alternative Viewpoint

> *(a) The generative model does indeed have a higher asymptotic error (as the number of training examples becomes large) than the discriminative model, but (b) The generative model may also approach its asymptotic error much faster than the discriminative model—possibly with a number of training examples that is only logarithmic, rather than linear, in the number of parameters.*

Ng and Jordan 2001

Implicit vs Explicit Distribution Modelling

- Explicit: calculate $P(x \sim X)$ for all x
- Implicit: can generate samples $x \sim X$

Why is explicit harder?

Explicit Distribution Modelling

- Dogs vs Cats: calculate probability X is a cat
- Faces: calculate probability X is a face

Implicit Distribution Modelling

- Dogs vs Cats: generate dog/cat labels at a ratio indicating confidence
- Faces: generate faces from the distribution of faces

Can you convert from explicit to implicit?

Can you go approximate explicit from implicit? How about implicit to explicit?

Table of Contents

GANs

Generative Adversarial Networks were introduced in 2014 [GPM+14]

- Goal is to model $P(X)$, the distribution of the training data
- Model can generate samples from $P(X)$
- Trained using a pair of "adversaries" (two players with conflicting loss functions)

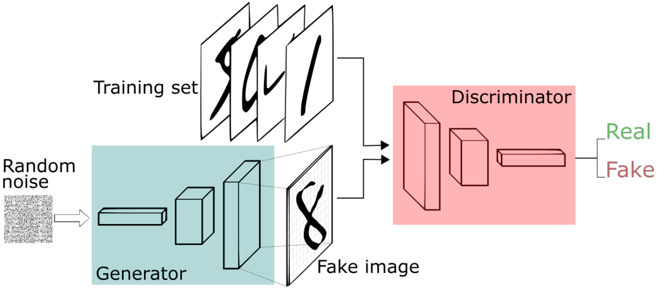
Generator

- The generator learns $P(X \mid Z)$; produce realistic looking output samples X given samples from a hidden space Z
  - Hidden representation Z is sampled from a known prior, such as a Gaussian
  - Generator function can be deterministic because composition of sampling from prior and the generator is stochastic
  - Generator maps between a simple known distribution and a complicated output distribution; learns a lower-dimensional manifold in the output space
  - However, no simple loss function available to measure the divergence between the generated distribution and the real distribution
  - Easy to measure distance between individual samples, harder to measure distance between complicated distributions
  - Instead of a traditional loss function, loss is calculated by a discriminator (another network)

Discriminator

- The discriminator is a secondary neural network that guides the generator
  - Trained to tell the difference between real and generated data
  - Generator tries to "confuse" the discriminator, so it can't tell the difference between real and generated data
  - From the perspective of the generator, the discriminator is like an adaptive loss function
- "Throwaway" network only really useful to train the generator

GAN Architecture Diagram



https://medium.freecodecamp.org/
an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394

Visualizing a GAN

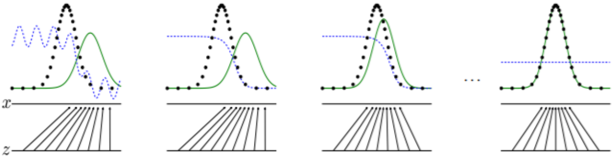


Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Min-Max Gaming

A GAN is defined by the following min-max game

$$\min_G \max_D V(D, G) = \mathbb{E}_X \log D(X) + \mathbb{E}_Z \log(1 - D(G(Z)))$$

- D wants $D(X) = 1$ and $D(G(Z)) = 0$
- G wants $D(G(Z)) = 1$

Min-Max Optimal Discriminator

What is the optimal discriminator?

$$f := \mathbb{E}_{X \sim P_D} \log D(X) + \mathbb{E}_{X \sim P_G} \log(1 - D(X))$$
$$= \int_X \left[P_D(X) \log D(X) + P_G(X) \log(1 - D(X)) \right] dX$$

Assuming we have an ideal function for the discriminator, it can output a different value for every X. So we optimize the following for each X.

$$\left[P_D(X) \log D(X) + P_G(X) \log(1 - D(X)) \right]$$

Min-Max Optimal Discriminator

Optimize by setting the derivative to 0.

$$\frac{\partial f}{\partial D(X)} = \frac{P_D(X)}{D(X)} - \frac{P_G(X)}{1 - D(X)} = 0$$

$$\frac{P_D(X)}{D(X)} = \frac{P_G(X)}{1 - D(X)}$$

$$(1 - D(X))P_D(X) = D(X)P_G(X)$$

$$D(X) = \frac{P_D(X)}{P_G(X) + P_D(X)}$$

Min-Max Optimal Discriminator

What is the optimal discriminator? How would you describe this in words?

$$D(X) = \frac{P_D(X)}{P_G(X) + P_D(X)}$$

Min-Max Optimal Discriminator

What is the optimal discriminator? How would you describe this in words?

$$D(X) = \frac{P_D(X)}{P_G(X) + P_D(X)}$$

If samples are equally likely to come from real or fake population, D is probability that the sample is real.
What would D be if the samples are definitely real? If they are definitely fake? If the real and fake distributions are the same?

Min-Max Optimal Value

What is value at the optimal discriminator?

$$
\begin{aligned}
f &:= \mathbb{E}_{X \sim P_D} \log D(X) + \mathbb{E}_{X \sim P_G} \log(1 - D(X)) \\
&= \mathbb{E}_{P_D} \log \frac{P_D(X)}{P_G(X) + P_D(X)} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{P_G(X) + P_D(X)}
\end{aligned}
$$

Can we rewrite this as a common divergence? Can anyone guess?

Min-Max Optimal Value

What is value at the optimal discriminator?

$$
\begin{aligned}
f &= \mathbb{E}_{P_D} \log \frac{P_D(X)}{P_G(X) + P_D(X)} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{P_G(X) + P_D(X)} \\
&= \mathbb{E}_{P_D} \log \frac{P_D(X)}{2m(X))} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{2m(X)} \\
&= \mathbb{E}_{P_D} \log \frac{P_D(X)}{m(X))} + \mathbb{E}_{P_G} \log \frac{P_G(X)}{m(X)} - \log 4 \\
&= KL(P_D \| m) + KL(P_G \| m) - \log 4 \\
&= 2\left(\frac{1}{2} KL(P_D \| m) + \frac{1}{2} KL(P_G \| m)\right) - \log 4
\end{aligned}
$$

$$
m(X) := \frac{P_D(X) + P_G(X)}{2}
$$

Min-Max Optimal Generator

What is the optimal generator?

$$J = \min_G 2JSD(P_D \| P_G) - \log 4$$

$$JSD(A\|B) := \frac{1}{2}KL(A\|\frac{A+B}{2}) + \frac{1}{2}KL(B\|\frac{A+B}{2})$$

Minimize the Jensen-Shannon divergence between the real and generated distributions (make the distributions similar). Roughly average KL between A and B and an average distribution.

Min-Max Stationary Point

- There exists a stationary point
  - If the generated data exactly matches the real data, the discriminator should output 0.5 for all inputs. Why?
  - If the discriminator outputs 0.5 for all inputs, the gradient to the generator is flat, so the generated distribution has no reason to change.

Min-Max Stable Point

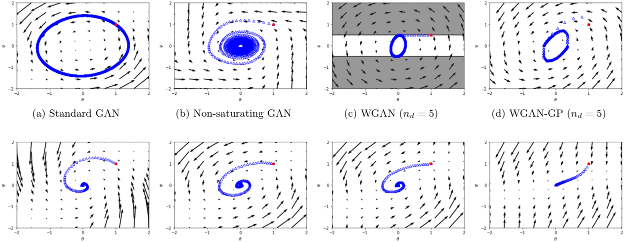
- The stationary point might not be stable (depends on exact GAN formulation and other factors)
  - If the generated data is near the real data, the discriminator outputs might be arbitrarily large
  - Generator may overshoot some values or oscillate around an optimum
  - Whether those oscillations converge or not depends on training details
- Imagine real data and generated data are separated by some minimal distance. A discriminator with unlimited capacity can still assign an arbitrarily large distance between these distributions.

Min-Max Optimization

- The hard part is that both generator and discriminator need to be trained simultaneously
- If the discriminator is under-trained, it provides incorrect information to the generator
- If the discriminator is over-trained, there is nothing local that a generator can do to get a marginal improvement
- The correct discriminator changes during training
- Discriminator and generator are trying to hit "moving targets"
- Significant research on techniques, tricks, modifications, etc. to help stabilize training

GAN Stability in Pictures

There are many variations of GANs that attempt to make the stationary point more stable



(a) Standard GAN (b) Non-saturating GAN (c) WGAN ($n_d = 5$) (d) WGAN-GP ($n_d = 5$)

(e) Consensus optimization (f) Instance noise (g) Gradient penalty (h) Gradient penalty (CR)

https://avg.is.tuebingen.mpg.de/projects/convergence-and-stability-of-gan-training

GAN Stability in Videos

GANs can be very sensitive to hyperparameters (more training details next time), as seen in these MNIST examples

- Good Hyperparameters https://www.youtube.com/watch?v=IUi0REAWj2c&t=4s
- Bad Hyperparameters https://www.youtube.com/watch?v=J8m1NXLwSKw
- More Advanced Method (WGAN-GP)

  https://www.youtube.com/watch?v=unXILX2wp1A

Perceptual Loss

- A discriminator might be able to address the ethereal issue of "perceptual distance"
  - Loss functions like L_2 are easy to implement and optimize
  - The L_2 distance is not very representative of images humans consider "similar"
  - Discriminator loss is much more flexible than L_1, L_2, etc.
  - For example, if discriminator includes a CNN, pooling, etc., then the loss will have some degree of shift invariance
- Although an idealized discriminator just calculates the *JS* divergence, a real discriminator calculates something much more complicated

Implicit Distributions

- Note that a generator implicitly learns a target distribution $P(X)$
  - Generator models $P(X \mid Z)$
  - Can draw samples from $P(X)$ by drawing samples from $P(Z)$ and calculating $P(X \mid Z)$
  - Not easy to actually marginalize over all Z and calculate $\mathbb{E}_Z P(X \mid Z)$ explicitly
  - So easy to draw samples, but requires sampling to calculate things like the likelihood of a given input

The Good, the Bad, and the Ugly

- **Good** GANs can produce awesome, crisp results for many problems
- **Bad** GANs have stability issues and open theoretical questions
- **Ugly** Many ad-hoc tricks and modifications to get GANs to work correctly

Table of Contents

GAN Evaluation

- The task of generating realistic-looking images is not as easily quantified as a task like correctly labeling images
- The distribution is implicit and we cannot easily evaluate by something like calculating the likelihood of a test set
  - Ask humans to compare and evaluate image quality
  - Sampling-based methods can approximately calculate the likelihood of a test set.
  - Neural networks trained for other purposes can be co-opted to evaluate GANs

Human Evaluations

The most direct answer to the question of whether generated data is "realistic-looking"

- Expensive
- Time consuming
- Not reproducible
- Yet maybe the only justifyable way to claim that generated data is "realistic"
- Maybe not so bad with MechanicalTurk, etc.

Approximate test set likelihood

A simple method to approximate the likelihood of a test set. However, not very accurate or efficient and requires a number of assumptions and hyperparameters.

- Cannot directly calculate $P(X)$, only $P(X \mid Z)$
- Therefore, pull many samples of Z and calculate $P(X \mid Z)$ for each, and then calculate the average probability
- If you generate a million images, and count how many of those match your test point, then you know the probability of the test point, sounds feasible ...?
- No image matches exactly, so generate a million images and place a Gaussian around each one. Convert your GAN to a GMM and calculate the probability under the GMM.
- Requires many samples, and some assumptions about a meaningful ball around each generated X

Evaluate with Discriminative Network

A standard discriminative network can be used to evaluate a GAN under some assumptions and some independence

- An Inception or other standard network is trained to classify real images into some number of labels
- A GAN is trained to generate images and is not given the labels
- If the GAN is generating images correctly
  - Inception should produce a wide variety of labels
  - Each label should have high confidence
- The "Inception Score" quantifies this intuition in terms of the entropy of each labeling and the entropy of the marginal labeling [SGZ+16]

Table of Contents

GANs and VAEs

GANs and VAEs are two large families of generative models that are useful to compare

- Generative Adversarial Networks (GANs) [this week] minimize the divergence between the generated distribution and the target distribution. This is a noisy and difficult optimization.
- Variational Autoencoders (VAEs) [next week] minimize a bound on the divergence between the generated distribution and the target distribution. This is a simpler optimization but can produce "blurry" results.

We will discuss some high-level comparisons between the two but save a deep-dive into VAEs for another time. There is also research on hybridizing the two models.

VAEs

- Similar to a typical autoencoders
  - Trained to reconstruct inputs
  - Encoder models $P(Z \mid X)$
  - Decoder models $P(X \mid Z)$
  - Hidden representation Z is learned by the model
- We encourage the marginal distribution over Z to match a prior $Q(Z)$
- Hidden representation during training is generated by encoder
- $\mathbb{E}_X P(Z \mid X) \approx Q(Z)$
- If our prior is something simple, then we can draw samples from the prior and pass them to the decoder.

Pros and Cons

- GANs produce “sharper” results
- VAEs train faster and more reliably
- VAEs require an analytical understanding of the prior and it’s KL divergence
- GANs only require the ability to sample from a prior
- VAEs learn an encoder decoder pair but GANs do not
- VAEs are more theoretically justified, the GAN zoo is more based on what works
- VAE generator trained on encoded data but evaluated on prior samples; GAN trained and evaluated on prior samples

Table of Contents

GAN Architectures

There are many variations of GANs for modeling different tasks. This is not meant to be exhaustive but a sample of the possibilities.

- GAN
- Conditional GAN
- LapGAN
- Recurrent Adversarial Network
- Categorical GAN
- InfoGAN
- AAE
- BiGAN
- CycleGAN

GAN

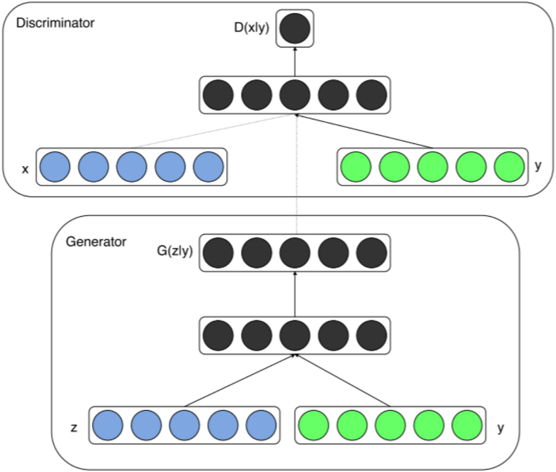
Unqualified, "GAN" typically refers to a simple model of $P(X)$ [GPM$^+$14]. This is a vanilla GAN. Think unsupervised generation of unlabeled images, video, etc.

Conditional GANs

A conditional GAN models $P(X \mid Y)$. For example, generate samples of MNIST conditioned on the digit you are generating. [MO14]. The model is constructed by adding the labels Y as an input to both generator and discriminator.

$$\min_G \max_D V(D, G) = \mathbb{E}_X \log D(X, Y) + \mathbb{E}_Z \log D(G(Z, Y), Y)$$

Conditional GAN Architecture



Conditional GAN Results

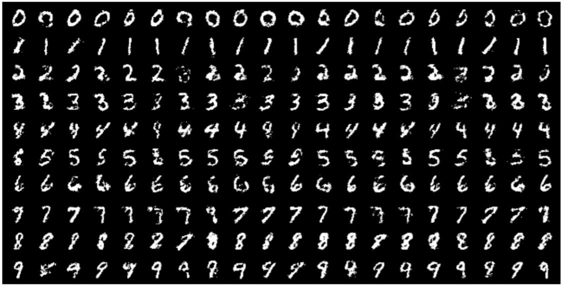
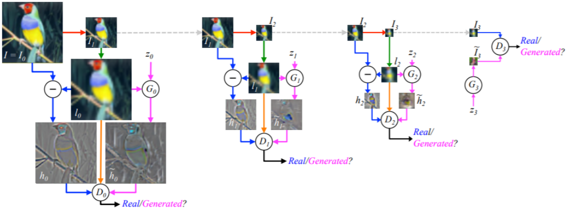


Figure 2: Generated MNIST digits, each row conditioned on one label

LapGAN

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]

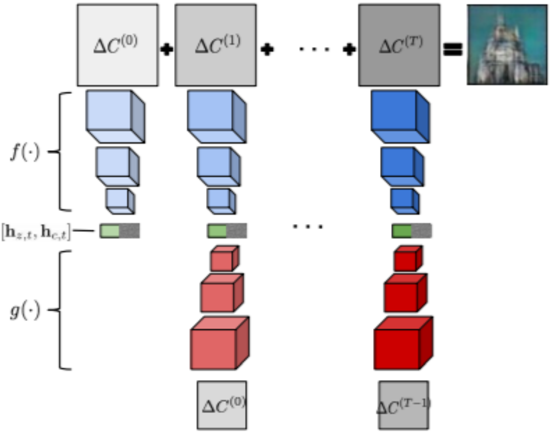
LapGAN Architecture



Recurrent Adversarial Networks

A recurrent adversarial network iteratively modifies a canvas to draw an image over several timesteps. The inputs to the generator are a sequence of prior samples. [IKJM16]

Recurrent Adversarial Network Architecture



Recurrent Adversarial Network Results

Images are generated over several timesteps

Figure 25. Drawing at different time steps on mnist samples.

Categorical GANs

A categorical GAN is useful for clustering and semi-supervised learning. Rather than a binary output, the discriminator produces a softmax output. The discriminator attempts to correctly label real data with low entropy and to produce high entropy labels for generated data. [Spr15]

CatGAN Results

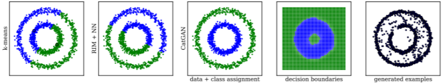


Figure 2: Comparison between k-means (left), RIM (middle) and CatGAN (rightmost three) – with neural networks – on the "circles" dataset with $K = 2$. Blue and green denote class assignments to the two different classes. For CatGAN we visualize class assignments – both on the dataset and on a larger region of the input domain – and generated samples. Best viewed in color.

InfoGANs

An InfoGAN learns both a decoder and a partial encoder. A secondary loss term is added to train an encoder to recover the hidden space from the output. The hidden space is split into c (information you care about) and z (noise you don't care about). [CDH+16]

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

The premise is that if you can recover z, then z will be meaningful and "disentangled"

InfoGAN Representations

InfoGAN learns meaningful representations

(a) Varying c_1 on InfoGAN (Digit type)

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

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

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

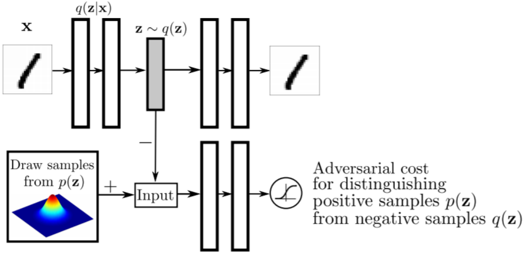
Adversarial Autoencoders

An adversarial autoencoder is like a combination of VAE and GAN. An encoder/decoder pair is trained to reconstruct X using hidden representation Z. [MSJG15]

- In VAE, encodings $\mathbb{E}_X P(Z \mid X)$ match prior $Q(Z)$ using bounds on KL divergence
- In AAE, encodings $\mathbb{E}_X P(Z \mid X)$ match prior $Q(Z)$ using discriminator to measure distance between the two distributions

If we have an autoencoder where the latent distribution is a known prior, then we can sample from Z directly, and now have a generative model.

AAE Architecture

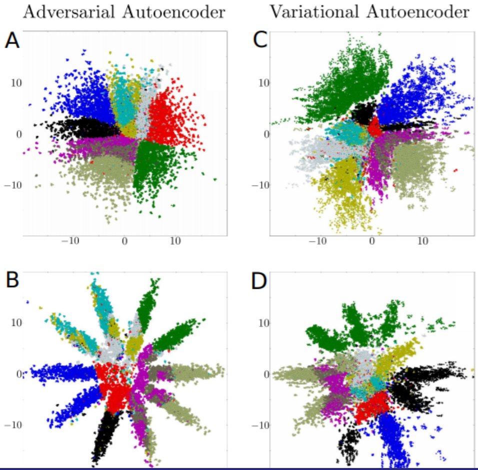


AAE vs. VAE

- Learns encoder/decoder pair instead of just decoder
- Discriminator works on latent space not input/output space, so easy to use on discrete inputs/outputs
- Latent space is strongly regularized to match prior exactly
- However, still requires a traditional loss function for reconstruction loss

AAE vs. VAE Visualized

AAE latent space matches prior better than VAE



BiGANs

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]

$$V(D, E, G) = \mathbb{E}_X \log D(X, E(X)) + \mathbb{E}_Z \log(1 - D(G(Z), Z))$$

This method simultaneously trains the pair and does not require any assumptions about the distance metric in either the hidden or output spaces.

BiGAN Architecture

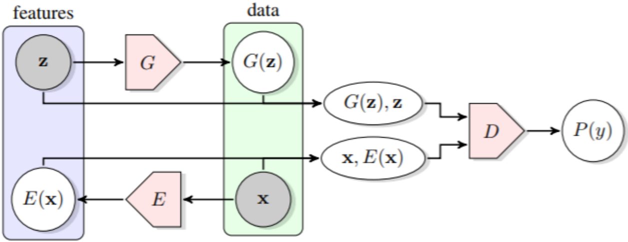


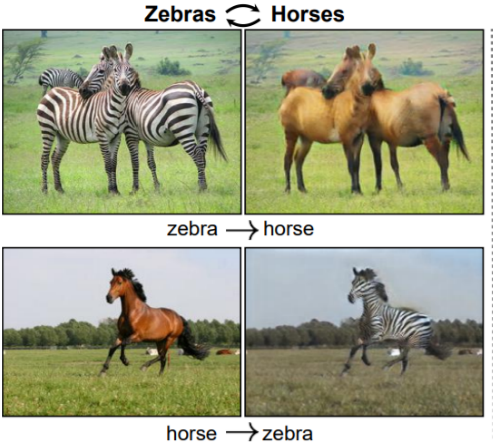
Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

CycleGAN

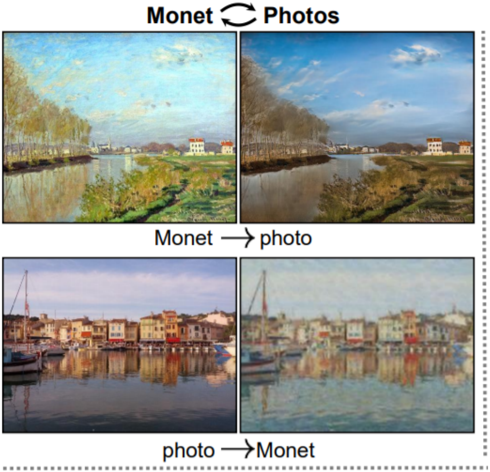
CycleGAN trains a pair of conditional GANs to perform image-to-image translation [ZPIE17].

- GAN A trained to convert from X to Y
- GAN B trained to convert from Y to X
- Additional "cycle-consistency" losses $\|Y - A(B(Y))\|_1$ and $\|X - B(A(X))\|$

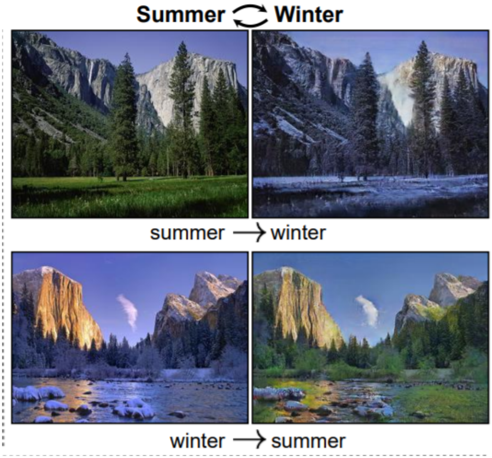
CycleGAN Results



CycleGAN Results



CycleGAN Results



CycleGAN Lesson

- There is no paired dataset of zebras and horses
- So no easy discriminative method to train zebras from horses
- But using GANs, can train distributions to match

References I

Andrew Brock, Jeff Donahue, and Karen Simonyan, *Large scale GAN training for high fidelity natural image synthesis*, CoRR **abs/1809.11096** (2018).

Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo, *Stargan: Unified generative adversarial networks for multi-domain image-to-image translation*, CoRR **abs/1711.09020** (2017).

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel, *Infogan: Interpretable representation learning by information maximizing generative adversarial nets*, CoRR **abs/1606.03657** (2016).

References II

- Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus, *Deep generative image models using a laplacian pyramid of adversarial networks*, CoRR **abs/1506.05751** (2015).
- Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell, *Adversarial feature learning*, CoRR **abs/1605.09782** (2016).
- I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Networks*, ArXiv e-prints (2014).

References III

- Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic, *Generating images with recurrent adversarial networks*, CoRR **abs/1602.05110** (2016).
- Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, *Progressive growing of gans for improved quality, stability, and variation*, CoRR **abs/1710.10196** (2017).
- Ming-Yu Liu, Thomas Breuel, and Jan Kautz, *Unsupervised image-to-image translation networks*, CoRR **abs/1703.00848** (2017).
- Mehdi Mirza and Simon Osindero, *Conditional generative adversarial nets*, CoRR **abs/1411.1784** (2014).

References IV

- Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow, *Adversarial autoencoders*, CoRR **abs/1511.05644** (2015).
- Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, *Improved techniques for training gans*, CoRR **abs/1606.03498** (2016).
- Jost Tobias Springenberg, *Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks*, arXiv e-prints (2015), arXiv:1511.06390.

References V

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, CoRR **abs/1703.10593** (2017).