CSC413 Neural Networks and Deep Learning

Lecture 11: Generative Adversarial Learning

Lecture 11

Mar 26/28, 2024

Table of Contents

Agenda

- Generative Adversarial Network (GAN)

Section 1

Generative Adversarial Networks

Generative Model

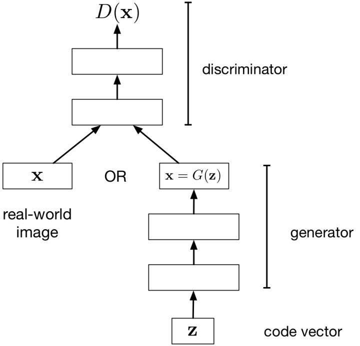
- A **generative model** learns the *structure* of a set of input data, and can be used to **generate** new data
- Examples:
  - RNN for text generation
  - Autoencoder
  - VAE

Blurriness of Autoencoder Images

- Blurry images, blurry backgrounds
- Why? Because the loss function used to train an autoencoder is the **mean square error loss** (MSELoss)
- To minimize the MSE loss, autoencoders predict the "average" pixel

Can we use a better loss function?

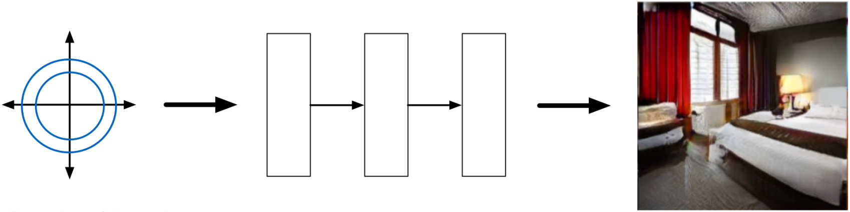
Generative Adversarial Network



- **Generator network**: try to fool the discriminator by generating real-looking images
- **Discriminator network**: try to distinguish between real and fake images

The loss function of the generator (the model we care about) is defined by the discriminator!

GAN Generator



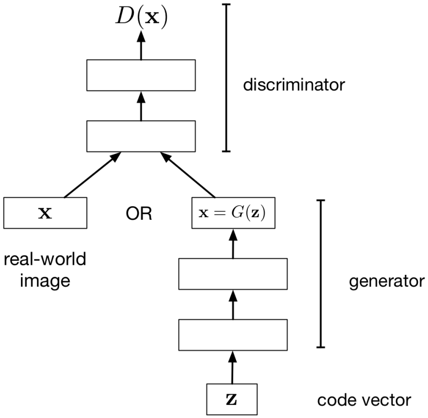
Each dimension of the code vector is sampled independently from a simple distribution, e.g. Gaussian or uniform.

This is fed to a (deterministic) generator network.

The network outputs an image.

- Generator's Input: a noise (i.e., random) vector
  - Q: Why do we need to input noise?
- Generator's Output: a generated image

GAN Architecture



- Discriminator Input: an image
- Discriminator Output: a binary label (real vs fake)

GAN: Generator and Discriminator

Generator:

- G: the generator neural network
- ϕ: the trainable parameters of the discriminator (we'll write G_ϕ if we want to make the dependency clear)
- z: a random noise vector
- $G(z)$ or $G_\phi(z)$: a generated image

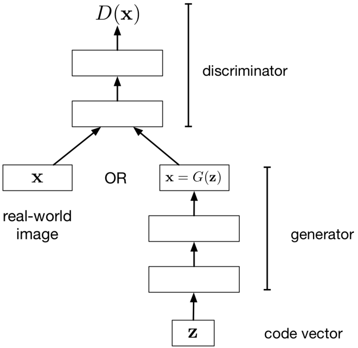
Discriminator:

- D: the discriminator neural network
- θ: the trainable parameters of the discriminator (we'll write D_θ if we want to make the dependency clear)
- x: an image (either real or fake)
- $D(x)$ or $D_\theta(x)$: the discriminator's determination of whether the image is real ($1 =$ real, $0 =$ fake)

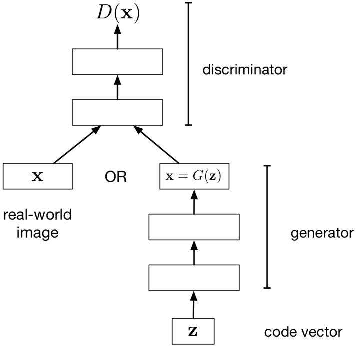
Two types inputs to the discriminator D

Questions:

- What does $D(x)$ with $x \sim D$ mean?
- What does $D(G(z))$ mean?



Optimizing GAN: A Battle of Two Networks



- The Generator G should generate realistic looking images (or any other data type)
- The Discriminator D should discriminate between real and fake images

This is an adversarial battle between two networks.

Optimizing GAN: Optimizing the Generator

Optimize **Generator's** weights to:

- **maximize** the probability that...
  - discriminator labels a generated image as real
  - Q: What loss function should we use?

We wish to tune ϕ to increase $D_\theta(G_\phi(z))$

$$\min_{\phi} \mathbb{E}_z \left[\log\left(1 - D_\theta(G_\phi(z))\right)\right]$$

Optimizing GAN: Optimizing the Discriminator

Optimize **Discriminator's** weights to:

- **maximize** the probability that the
  - discriminator labels a real image as real
  - discriminator labels a generated image as fake
  - Q: What loss function should we use?

We wish to tune θ to:

- decrease $D_\theta(G_\phi(z))$
- increase $D_\theta(x)$, where $x \sim \mathcal{D}$ (the data distribution)

$$\max_\theta \mathbb{E}_{x\sim\mathcal{D}}\left[\log D_\theta(x)\right] + \mathbb{E}_z\left[\log\left(1 - D_\theta(G_\phi(z))\right)\right]$$

GAN Optimization

We optimize

$$\min_{\phi} \max_{\theta} \mathbb{E}_{x \sim \mathcal{D}} [\log D_\theta(x)] + \mathbb{E}_z [\log (1 - D_\theta(G_\phi(z)))]$$

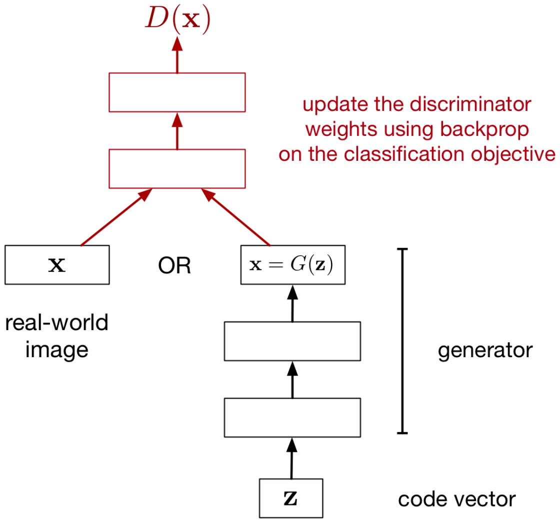
This is called the *minimax formulation* since the generator and discriminator are playing a zero-sum game against each other. That is why we have *adversarial* in the name.

Training GAN

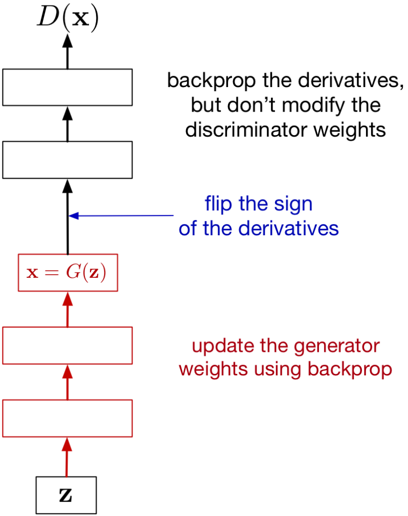
Alternate between:

- Training the discriminator
- Training the generator

Updating the discriminator

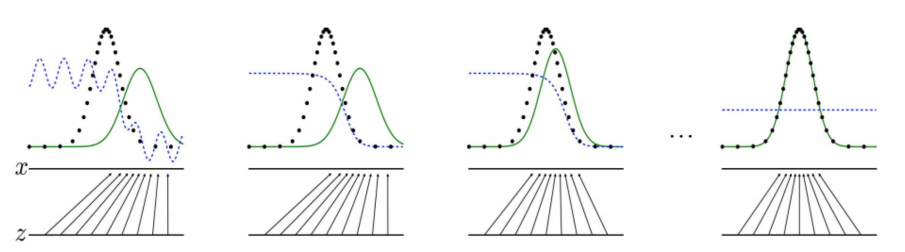


Updating the generator



GAN Alternating Training Visualized

Black dots is the data distribution $\mathcal{D}$, green line is the generator distribution $G(z)$, and blue dotted line is the discriminator:



1. The distributions $G(z)$ and $\mathcal{D}$ are quite different
2. The discriminator is updated to be able to better distinguish real vs fake
3. The generator is updated to be better match $\mathcal{D}$
4. If training is successful, $G(z)$ is indistinguishable from $\mathcal{D}$

GAN Demo

https://poloclub.github.io/ganlab/

A better cost function

- We introduced the minimax cost function for the generator:

$$\min_{\phi} \mathbb{E}_z \left[\log \left(1 - D_\theta(G_\phi(z))\right)\right]$$

- One problem with this loss function is *saturation*
- Recall from classification. When the prediction is really wrong
  - "Logistic + square error" gets a weak gradient signal
  - "Logistic + cross-entropy" gets a strong gradient signal
- Here, if the generated sample is really bad, the discriminator's prediction is close to 0, and the generator's cost is flat

A better generator cost function

Original minimax cost:

$$\min_{\phi} \mathbb{E}_z \left[\log\left(1 - D_\theta(G_\phi(z))\right)\right]$$

Modified generator cost:

$$\min_{\phi} \mathbb{E}_z \left[-\log D_\theta(G_\phi(z))\right]$$

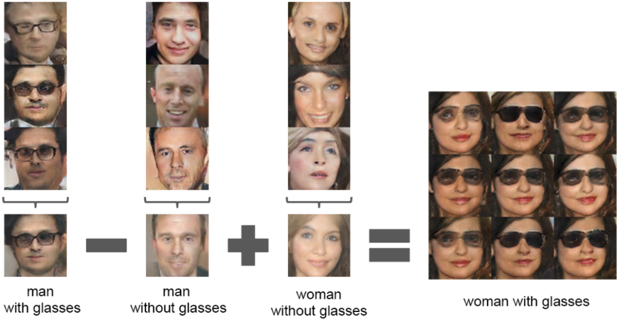
GAN Caveat

- Can work very well and produces crisp, high-res images, but **difficult to train**!
- Difficult to numerically see whether there is progress
  - Plotting the "training curve" (discriminator/generator loss) doesn't help much
- Takes a long time to train (a long time before we see progress)

GAN: Interpolation in z

Radford et al. (2016) https://arxiv.org/abs/1511.06434

GAN: Vector Arithmetic in z



Radford et al. (2016) https://arxiv.org/abs/1511.06434

GAN Samples (2019)

IMageNet object categories (by BigGAN, a much larger model, with a bunch more engineering tricks)

Brock et al., 2019. Large scale GAN training for high fidelity natural image synthesis

Mode Collapse

One prominent issue in training GAN is *mode collapse*

- The word “mode” here means “peak” or ” high-value local optimum”
- GAN model learns to generate one type of input data (e.g. only digit 1)
- Generating anything else leads to detection by discriminator
- Generator gets stuck in that local optima

Balance between Generator and Discriminator

If the discriminator is too good, then the generator will not learn due to **saturation**:

- Remember that we are using the discriminator like a “loss function” for the generator
- If the discriminator is too good, small changes in the generator weights won’t change the discriminator output
- If small changes in generator weights make no difference, then we can’t incrementally improve the generator

Wasserstein GAN

Idea: Use a different loss function.

Arjovsky et al. (2017) Wasserstein GAN. https://arxiv.org/abs/1701.07875

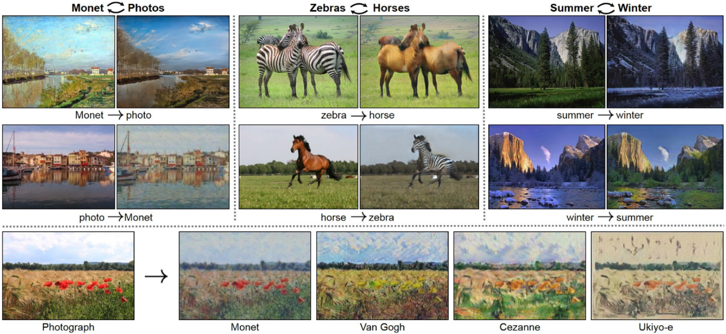
- Use the *Wasserstein distance* between the generator distribution and the data distribution

$$\min_{\phi} \overbrace{\max_{\theta : \|D_\theta\|_L \leq 1} \mathbb{E}_{x \sim \mathcal{D}} \left[D_\theta(x) \right] - \mathbb{E}_z \left[D_\theta(G_\phi(z)) \right]}^{\approx W(P_{\mathcal{D}}, P_G)}$$

- Reduces mode collapse, better measurement of progress
- Enforcing the Lipschitz continuity is done by clipping the weights of the discriminator - pushes weights towards the extremes
- Gradient penalties circumvent this issue; see Improved Training of Wasserstein GANs

Style Transfer with Cycle GAN

Style transfer problem: change the style of an image while preserving the content.

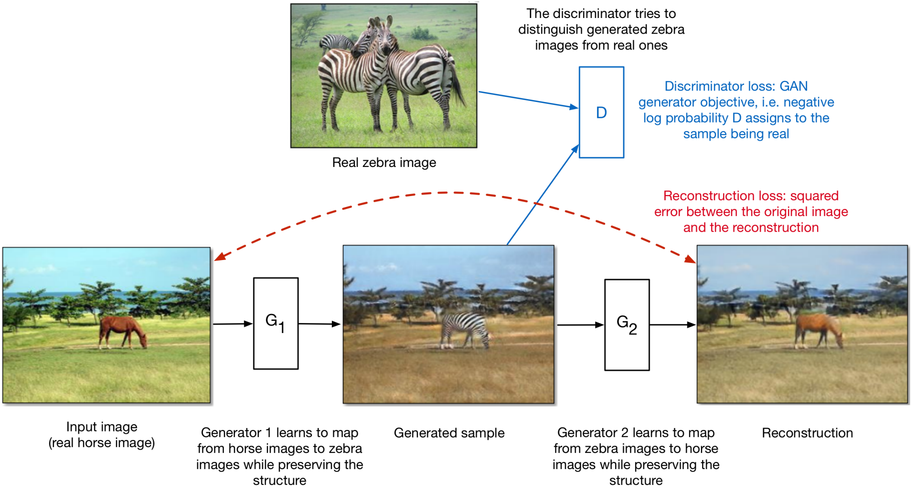


Data: Two unrelated collections of images, one for each style

Cycle GAN Idea

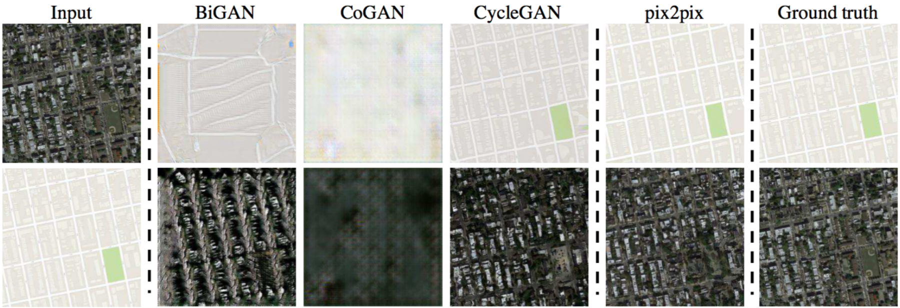
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated 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.

Cycle GAN Architecture



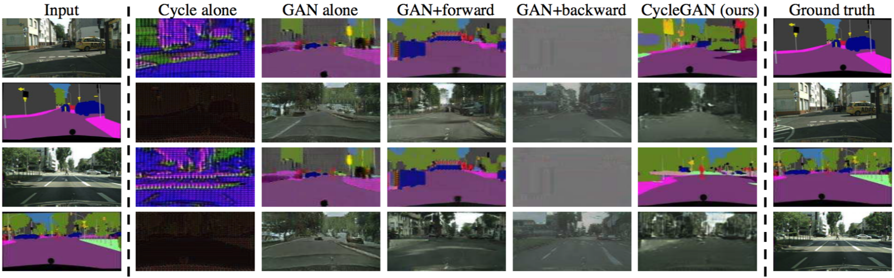
Total loss = discriminator loss + reconstruction loss

Cycle GAN: Aerial photos and maps



Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. See also the pix2pix paper.

Cycle GAN: Road scenes and semantic segmentation



Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Section 2

Next Week

Next Week

- Take-home test will be released next week.
  - We release it on Tuesday (April 2nd) and we collect it on Thursday (April 4th).
  - One goal of the take-home test is to give you an opportunity to review the material of this course one more time. This helps with consolidating the material of this course in your memory.
  - You should work alone. You can consult slides or books or papers, but you should not search for the solution over Google or use ChatGPT or similar systems.
    - We want you to fine-tune your weights in your brain.
  - It probably takes 3-4h to write it down, if you do not need to go back and re-learn things. To be safe, allocate 6-10h to it.
- We will probably have some guest speakers next week.
- Thank you for being with us this whole semester!