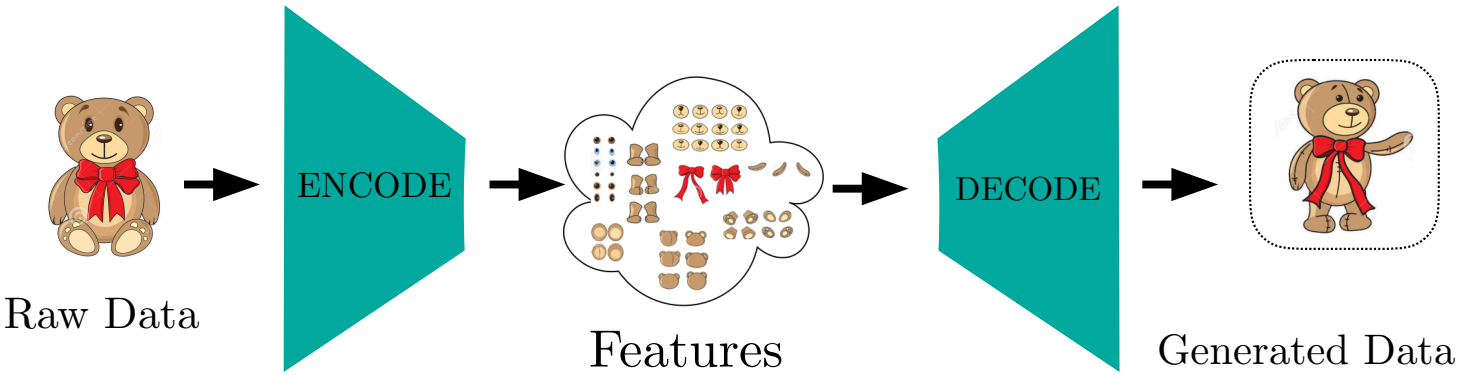
11-695: AI Engineering
GAN

Spring 2019

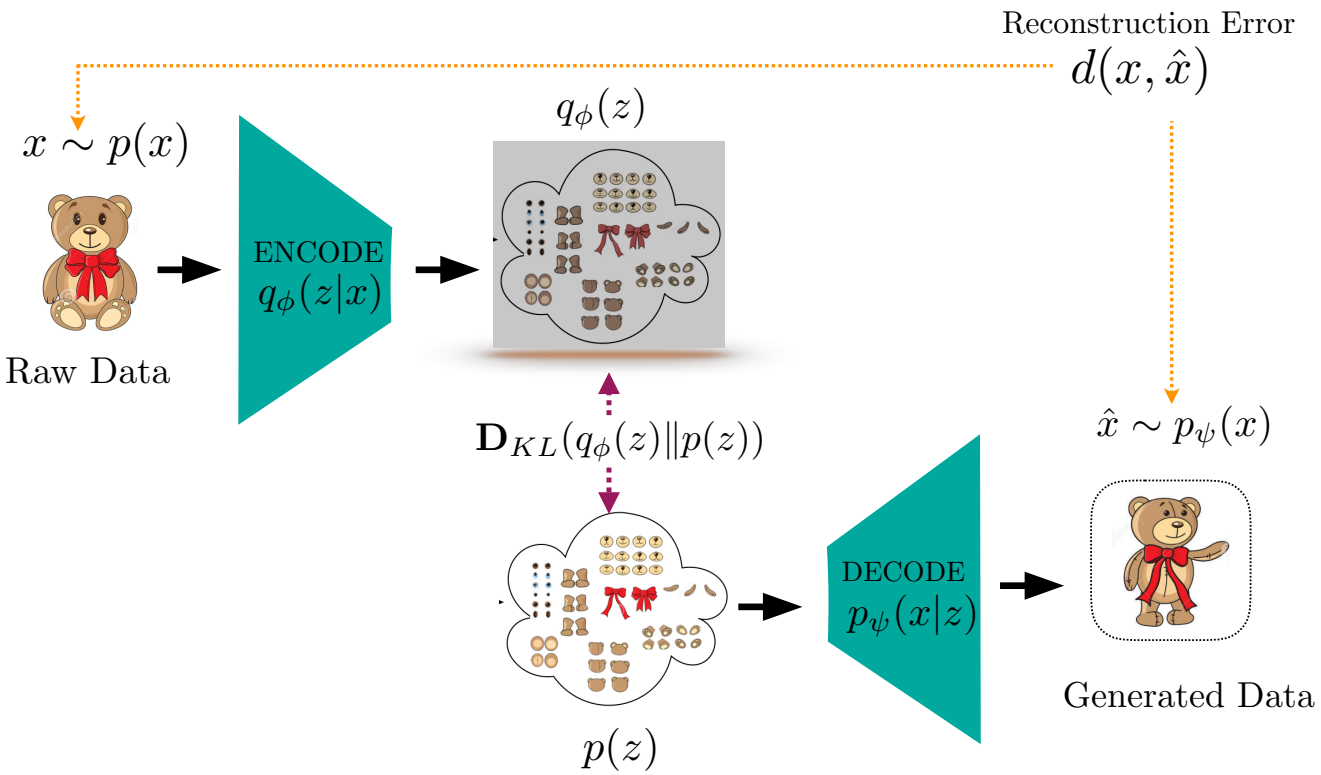
Outline

AutoEncoder



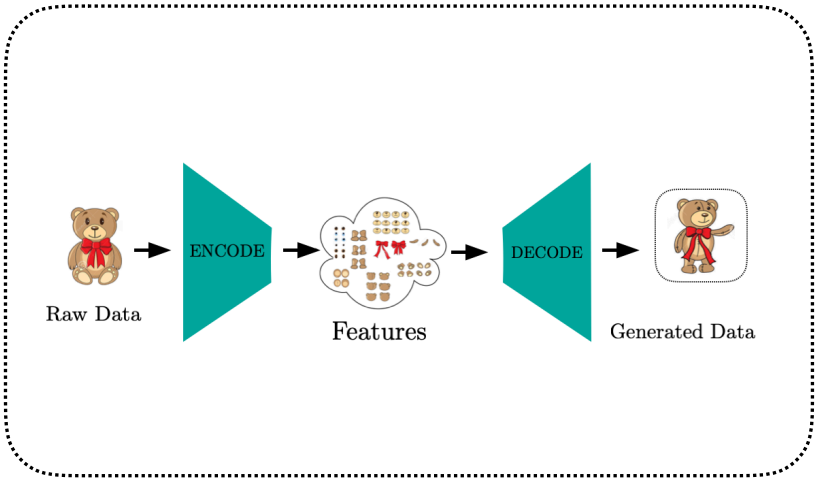
- We only have reconstruction loss
- Read more: Denoising AutoEncoder

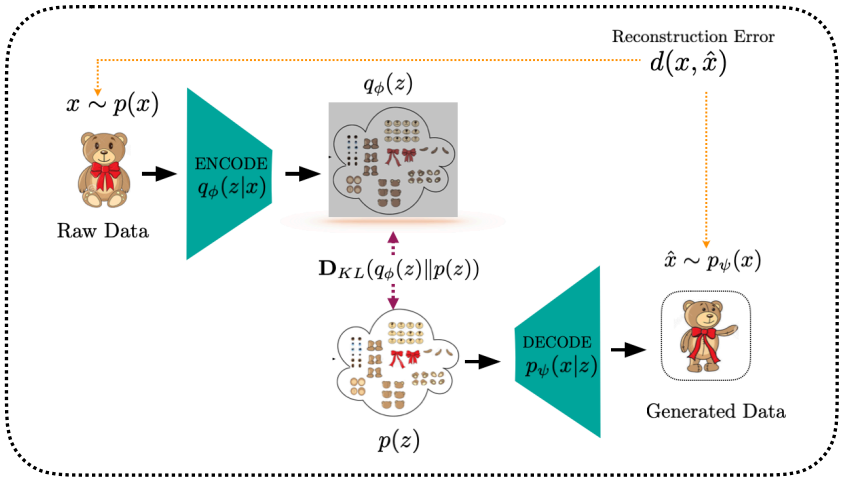
AE + Variational Approximation



- Still AutoEncoder, with KL term for variational vs. true distribution → Variational AutoEncoder
- We have ELBO loss which is: Likelihood + KL distance

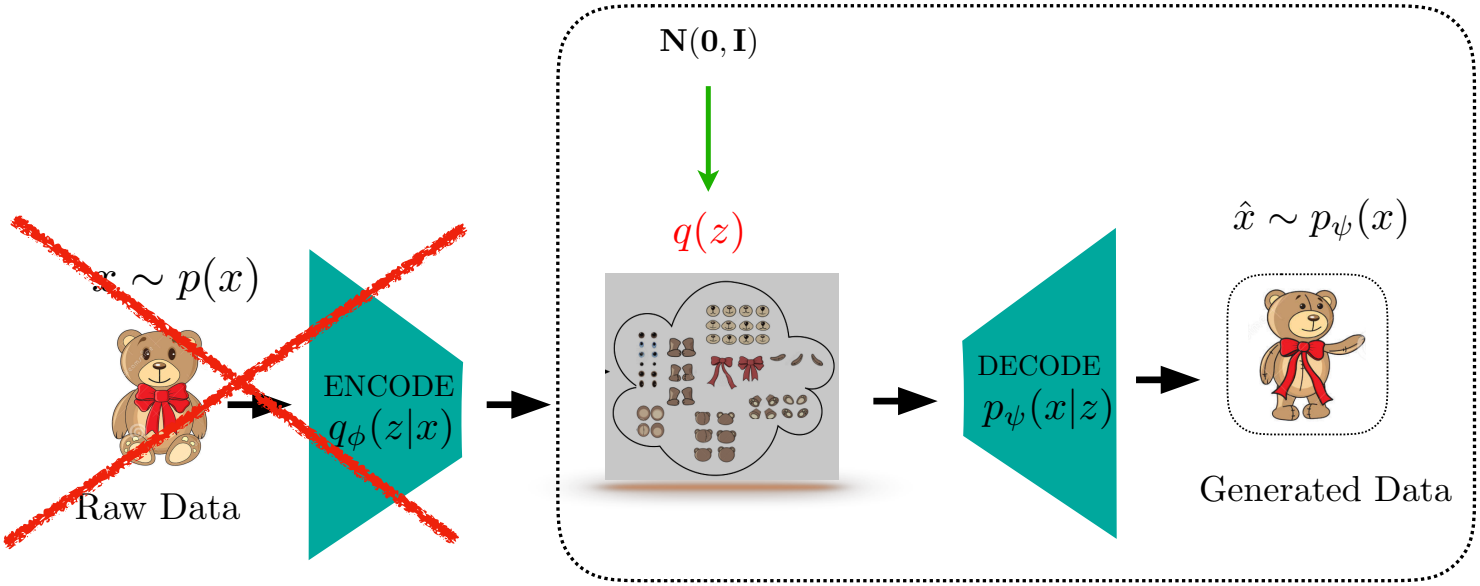
Some Observations





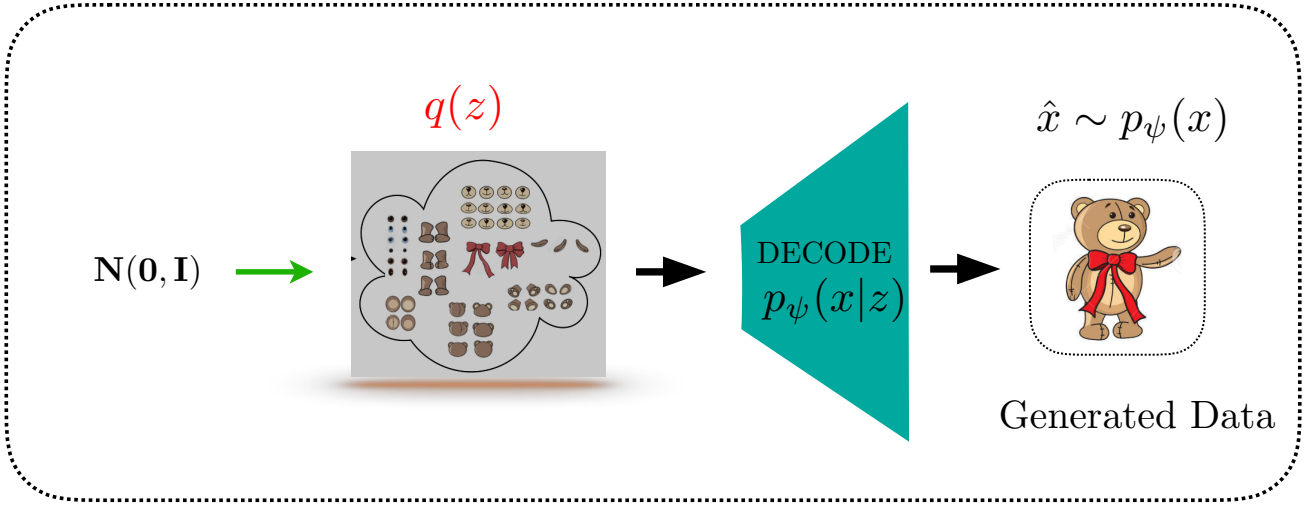
- Both approaches depend on how well you extract data, hence Likelihood-based approach
- With help from a noise (e.g. $\mathbf{N}(\mathbf{0}, \mathbf{I})$), data can be better generated

What about no likelihood?



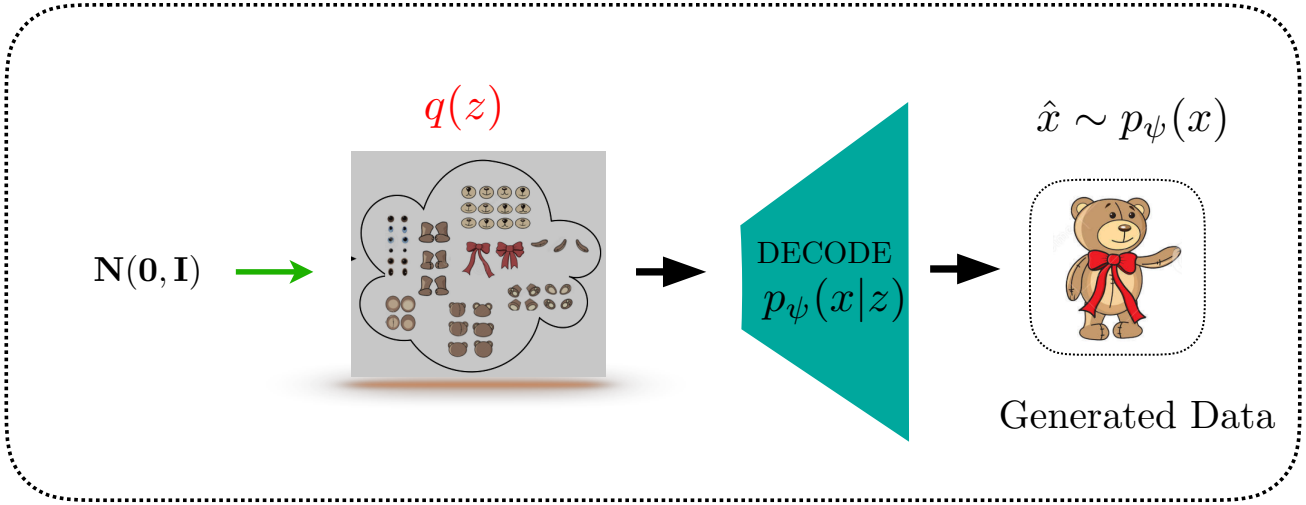
- What if we get rid of Encode part from data
- *i.e.* we are left with generation only noise space
- Recall: We need to tune the noise. How?
- Recall: Use KL distance from a variational distribution to the true distribution

Problem with No Encoder



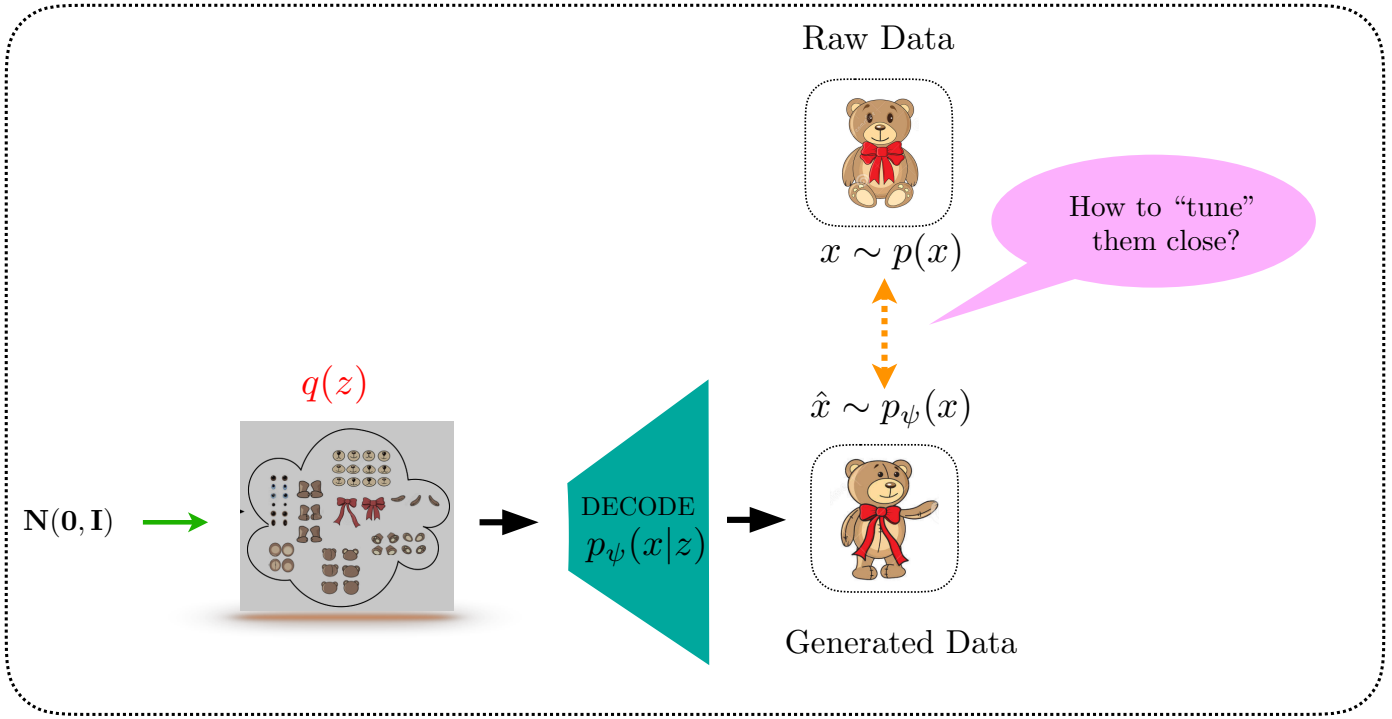
- ELBO $= log\, p(x) - \mathbf{D}_{KL}(q(z)\|p(z|x))$
- But we don't have $p(z|x)$ and so no base distribution to do a variational approximation on
- But: we need to make use of data $(x \sim p(x))$ to tune Decoder

Problem with No Encoder



- ELBO $= log\, p(x) - \mathbf{D}_{KL}(q(z)\|p(z|x))$
- But we don't have $p(z|x)$ and so no base distribution to do a variational approximation on
- And we generate from a noise (random) space

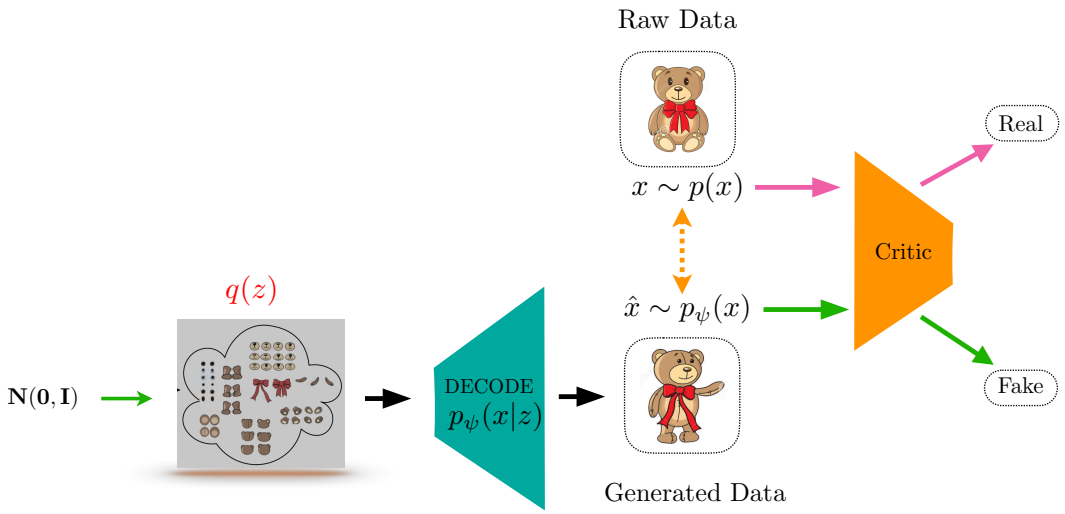
Replace Encoder with a New Component



- Why don't we try to have a critic to evaluate how well the generator is?
- (hence the "Adversarial")

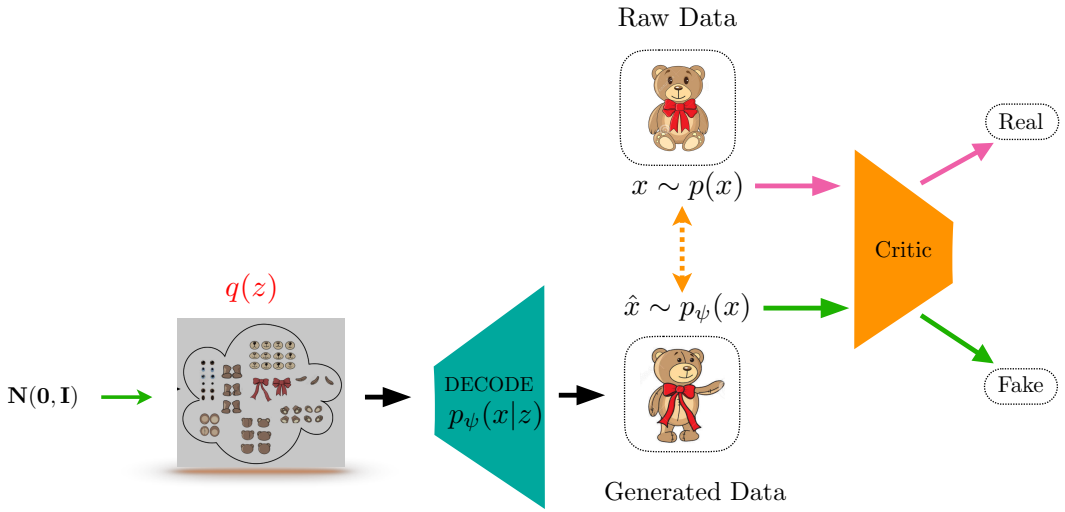
Outline

Requirements for Critic



- It needs to make use of raw data, the only resource we have, and to compare with.
- Design: like encoder, it should be able to extract features from data (of course, normally with a NN)
  - But only for a much simpler task: binary classification
  - So it looks like we have shifted encoder from front to back

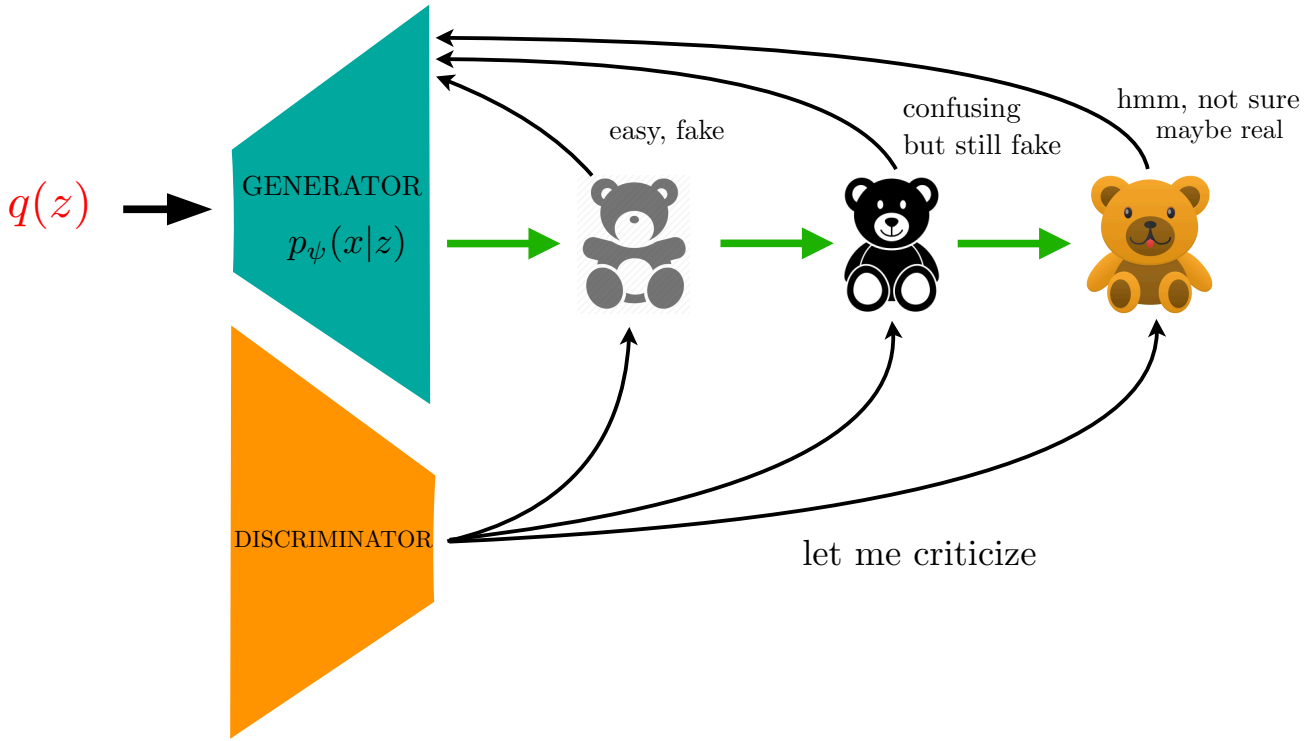
Requirements for Critic



- Binary classification: isn't it too simple?
- Not quite, if we include a feedback loop, in that
  - Assume Critic is strict, and perfect, and constantly gives "feedback"
  - Generator uses feedback to get better overtime

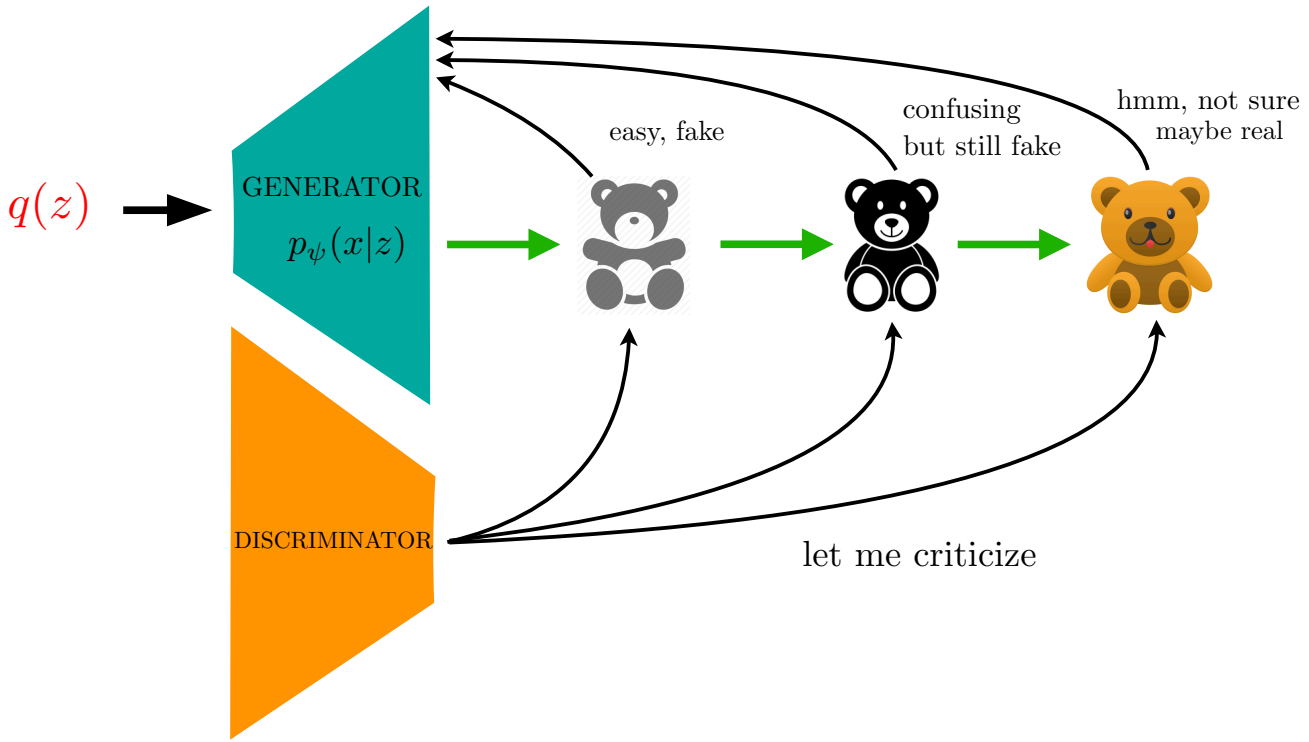
Call them Generator and Discriminator

Carnegie Mellon



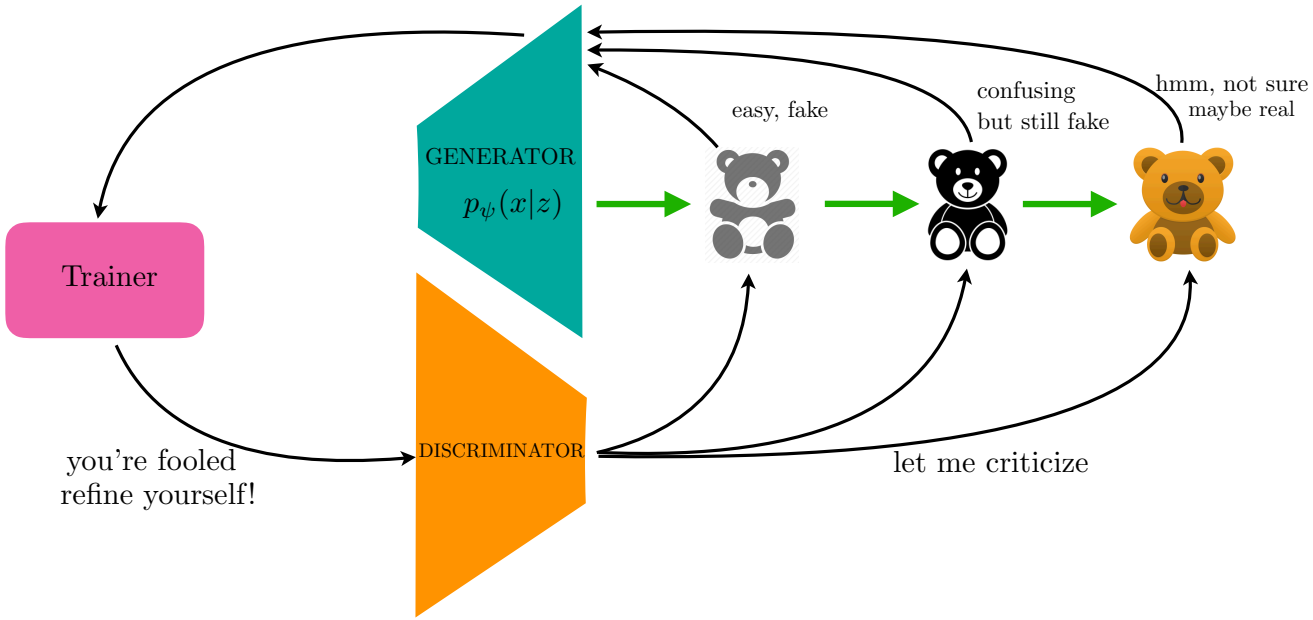
- Binary classification: isn't it too simple?
- Not quite, if we include a feedback loop, in that
  - Assume Critic is strict, and perfect, and constantly gives "feedback"
  - Generator uses feedback to get better overtime

We call the feedback process is a game



- Generator tries to fool the Discriminator
- Discriminator tries not to be fooled by Generator

Feedback to Discriminator?



- For sure we need! It's a NN and thus needs training
- As generator is better, discriminator has to be better too
- So both end up in a state where generated images are hard to classify

Let's formulate

- Data x, Noise z, Generator G, Discriminator D
  - Real data: $x \sim p(x)$
  - Generated (fake) data: $\hat{x} \sim p(z) = G_\psi(z)$
  - Discriminator classify $D_\theta(a)$ with input a
- Discriminator to, at the same time, **maximize**:
  - to tune $D_\theta(x)$ close to 1,
  - and $D_\theta(G_\psi(z))$ close to 0
- Generator, conversely, wants to **minimize** $D_\theta(G_\psi(z))$ to 1

It's called a MiniMax Game

Carnegie Mellon

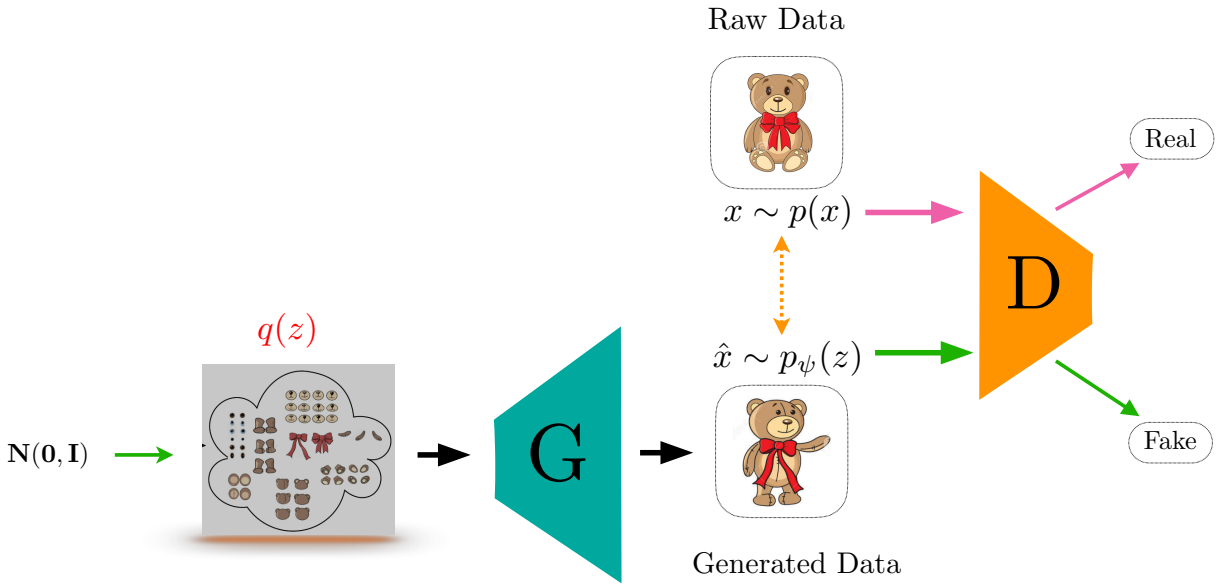
- Discriminator to **maximize**:
  - to tune $D_\theta(x)$ close to 1: $\mathbb{E}_{x\sim p(x)} log\, D_\theta(x)$
  - and $D_\theta(G_\psi(z))$ close to 0: $\mathbb{E}_{z\sim p(z)} log\, D_\theta(G_\psi(z))$
- Rewrite Discriminator:

  $\max_\theta[\mathbb{E}_{x\sim p(x)} log\, D_\theta(x) + (1 - \mathbb{E}_{z\sim p(z)} log\, D_\theta(G_\psi(z))]$
- Generator: tries to **minimize** that term:

  $\min_\psi \max_\theta[\mathbb{E}_{x\sim p(x)} log\, D_\theta(x) + (1 - \mathbb{E}_{z\sim p(z)} log\, D_\theta(G_\psi(z))]$

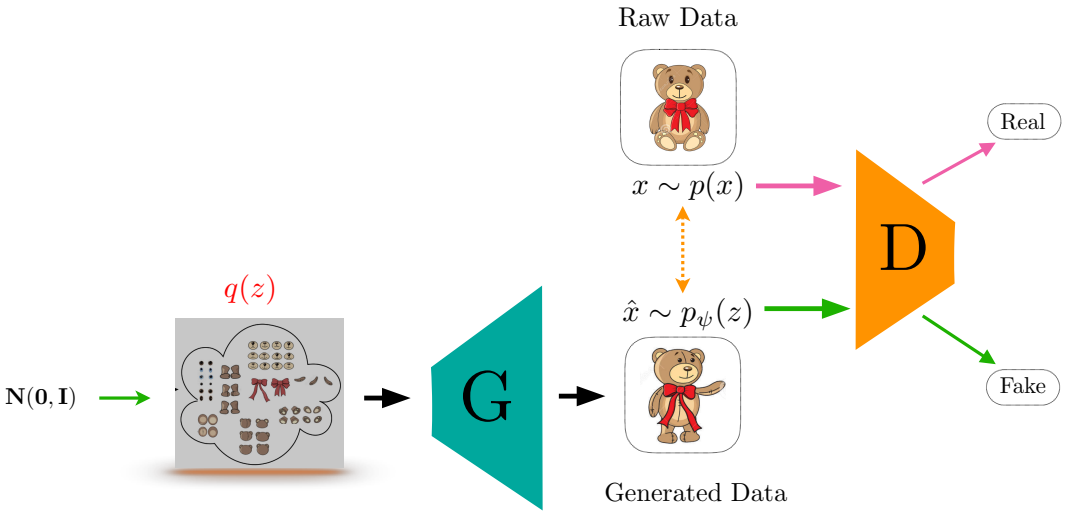
Outline

We have two networks



- A Generator: to fool a Discriminator
- A Discriminator: to play against Generator, hence "adversarial"

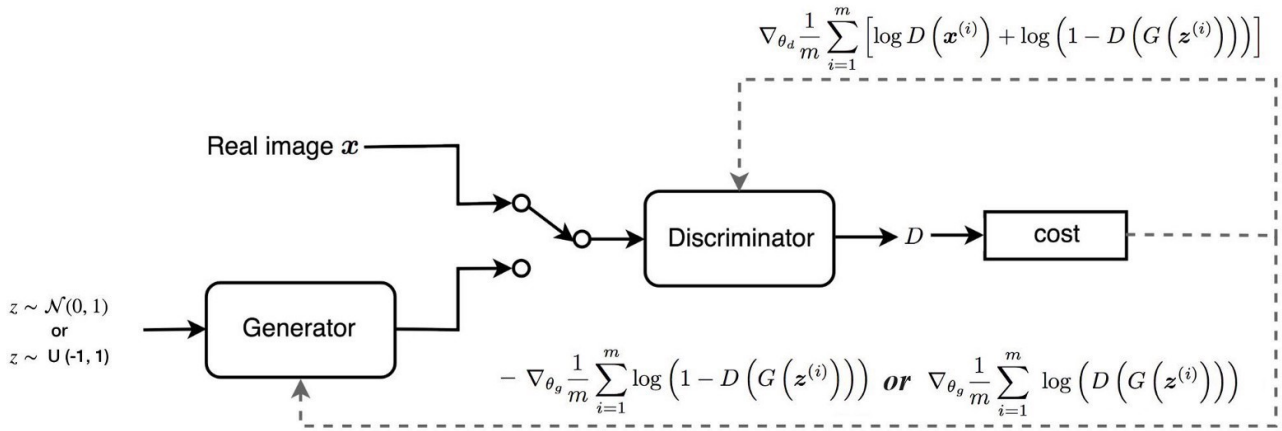
Training



- Objective:

  $\min_\psi \max_\theta [\mathbb{E}_{x \sim p(x)} log\, D_\theta(x) + (1 - \mathbb{E}_{z \sim p(z)} log\, D_\theta(G_\psi(z))]$
- We alternate in that:
  - Make G constant, let D **maximize** the objective
  - Make D constant, let G **minimize** the objective
- Question: single backprop?

Backprop and Batching

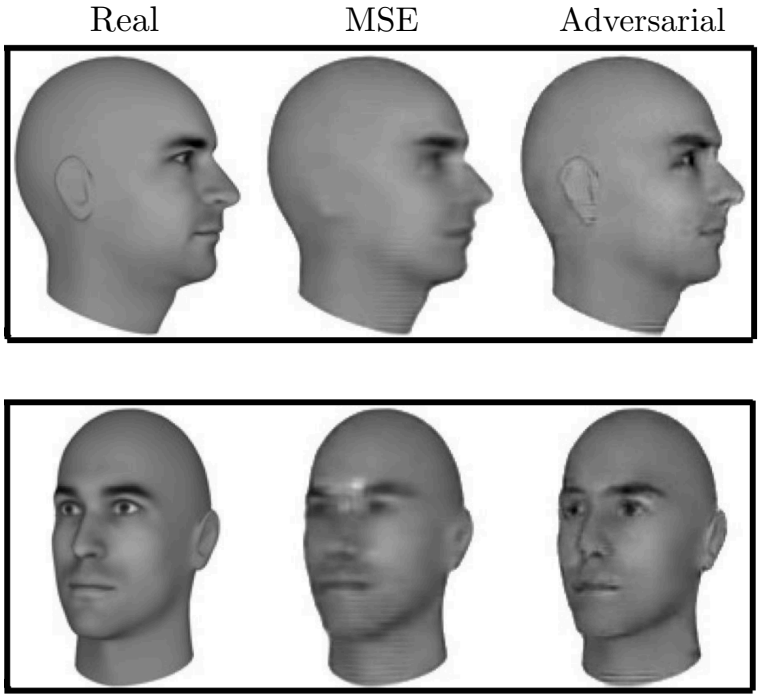


- Empirical expectation for gradient of D_θ:
  $\nabla_\theta = \frac{1}{m} \sum_{i=1}^{m} [log D_\theta(x^{(i)}) + log(1 - D_\theta(G_\psi(z^{(i)})))]$
- Empirical expectation for gradient of G_ψ:
  $\nabla_\psi = \frac{1}{m} \sum_{i=1}^{m} [log(1 - D_\theta(G_\psi(z^{(i)})))]$
- Batching for D_θ: half real and half fake.

Image credit: Johnathan Hui

GAN vs. Likelihood-based Generation

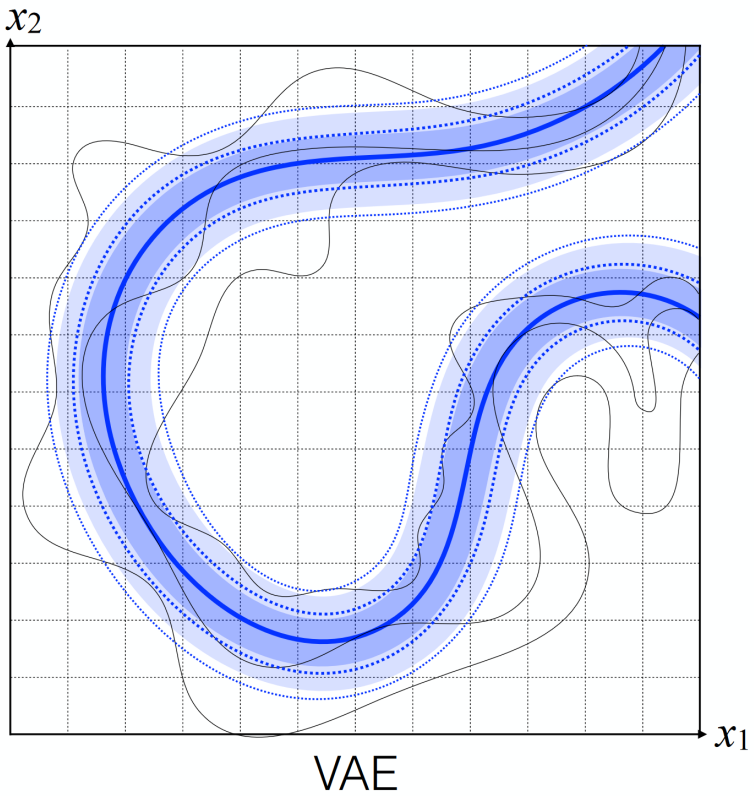
Carnegie Mellon



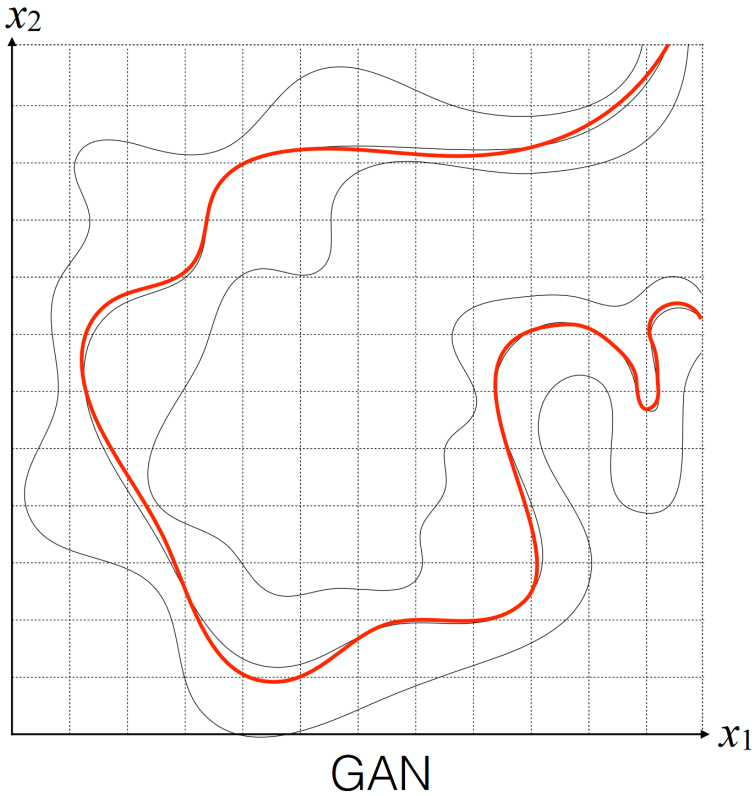
- MLE: blurry, seems to have the average effect
- GAN: sharp, seems to directly maximize the real-like generation (thanks to minimax theory)

Image credit: William Lotter *et. al.* 2015

GAN vs. VAE



VAE



GAN

Image credit: Quora.com (Louis Henri Franc)

High-resolution Image Generation

Carnegie Mellon

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.

Image credit: Karras *et al.* 2018