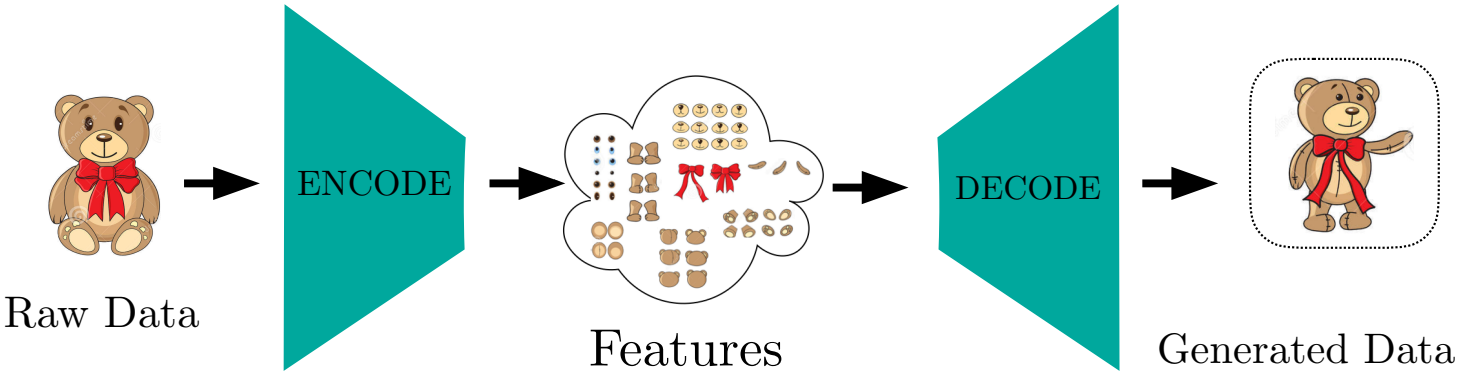
11-695: AI Engineering
GAN

LTI/SCS

Spring 2020

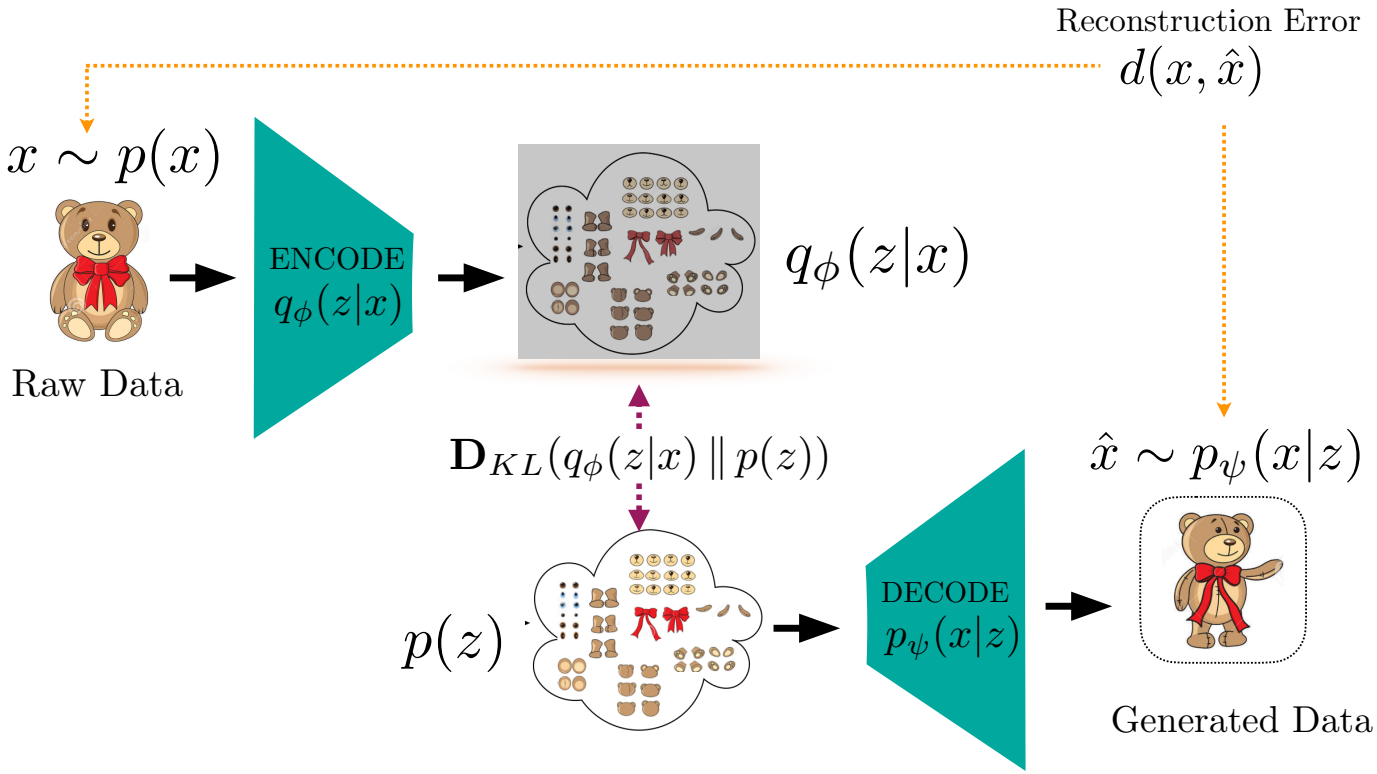
Outline

AutoEncoder



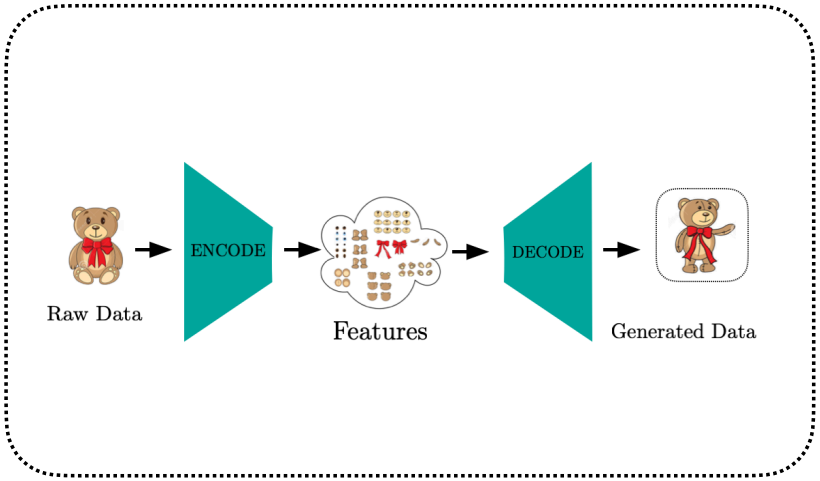
- We only have reconstruction loss
- No constraint is applied to the embedded representation

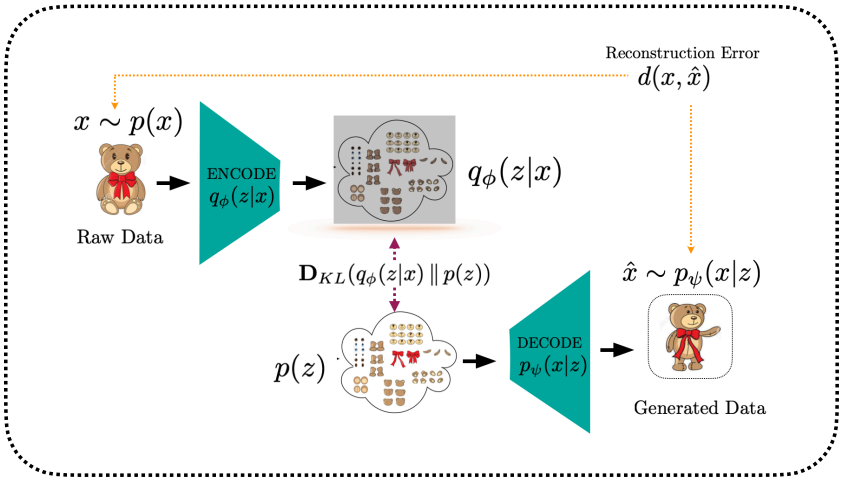
Variational AutoEncoder



- Reconstruction loss + KL loss
- Constraint for embedded representation: close to prior

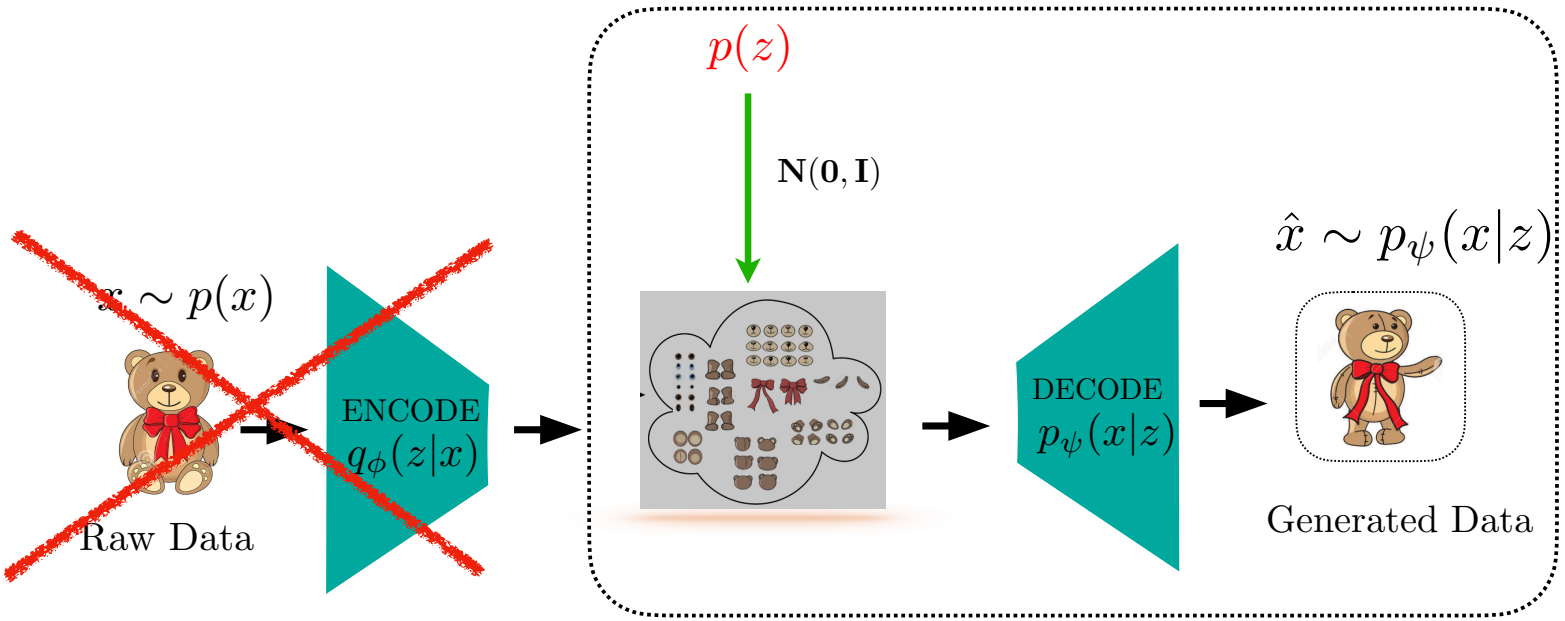
Some Observations





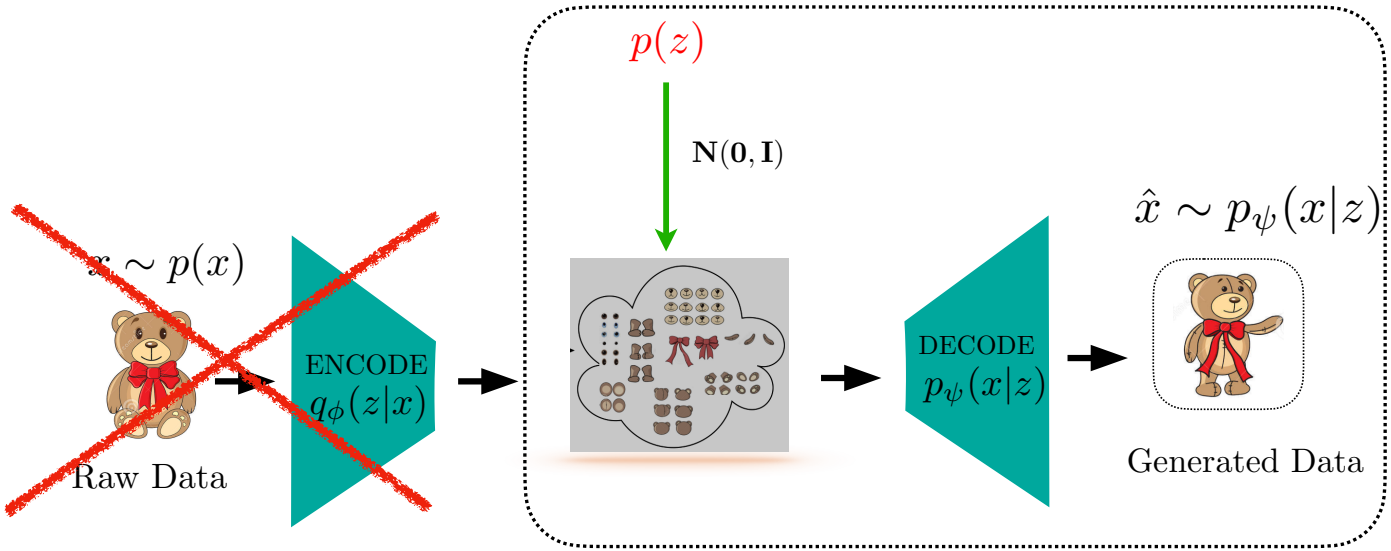
- Both approaches depend on how well you extract data
- MLE-based training
- With help from a constraint, new data can be generated

What about no feature extraction?



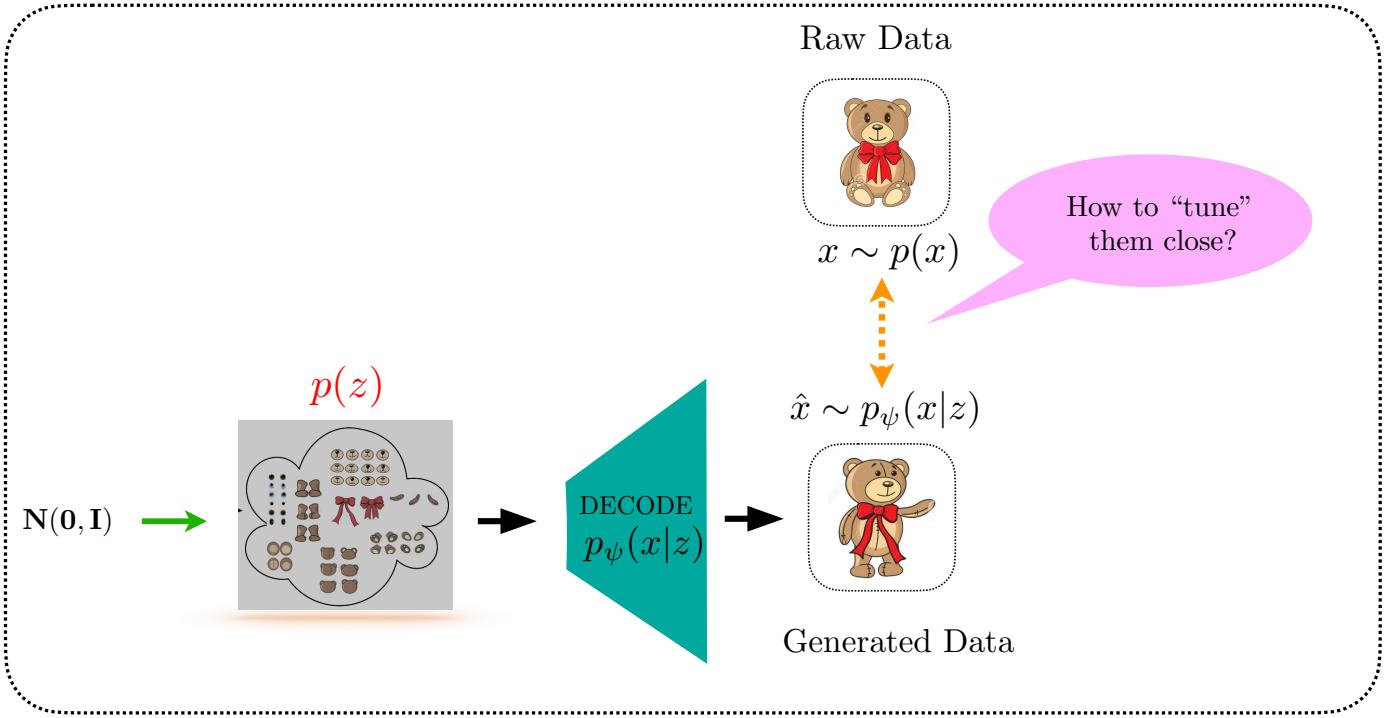
- What if we get rid of Encode part from data, *i.e.* we are left with generation only noise space
- Recall: We need to tune the noise.

Problem with No Encoder



- ELBO $= \mathbb{E}_{q_\phi(z|x))}[log\, p(x|z)] - \mathbf{D}_{KL}(q_\phi(z|x)\|p(z))$
- We don't have $q(z|x) \rightarrow$ no base distribution to optimize against
- All we have: data $(x \sim p(x))$ to tune Decoder

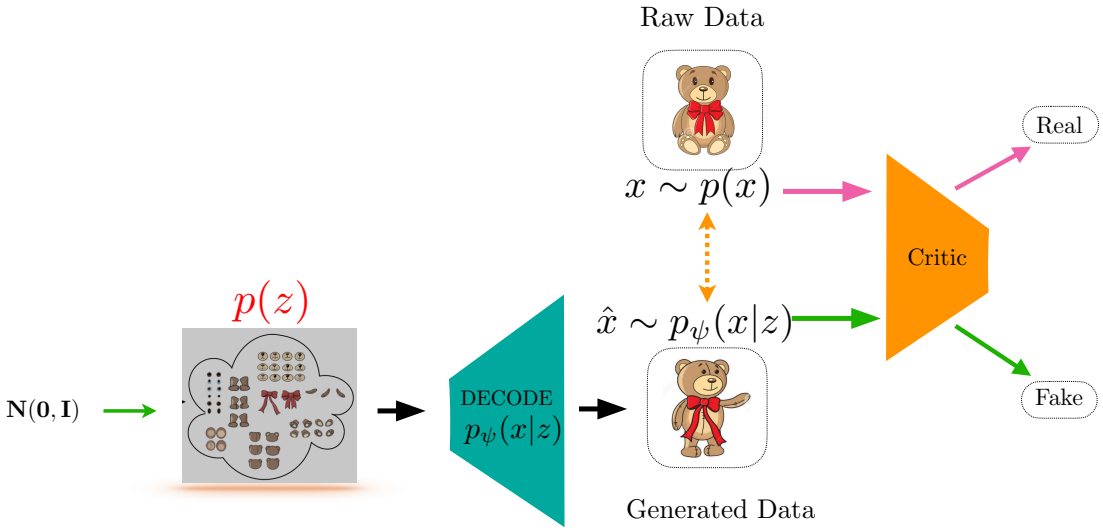
Replace Encoder with a Component



- Why don't we try to have a critic to evaluate how well the generator is?

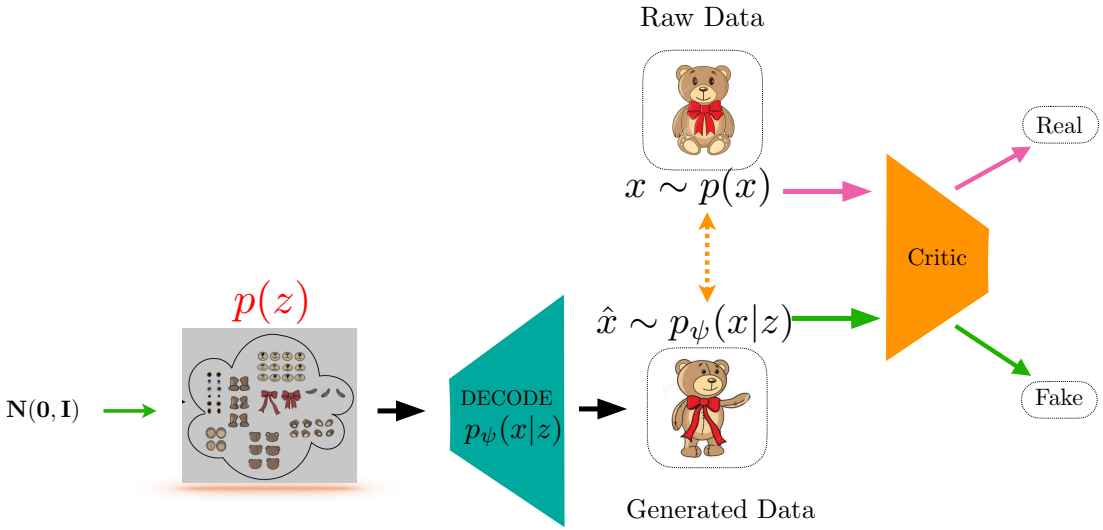
Outline

Requirements for Critic



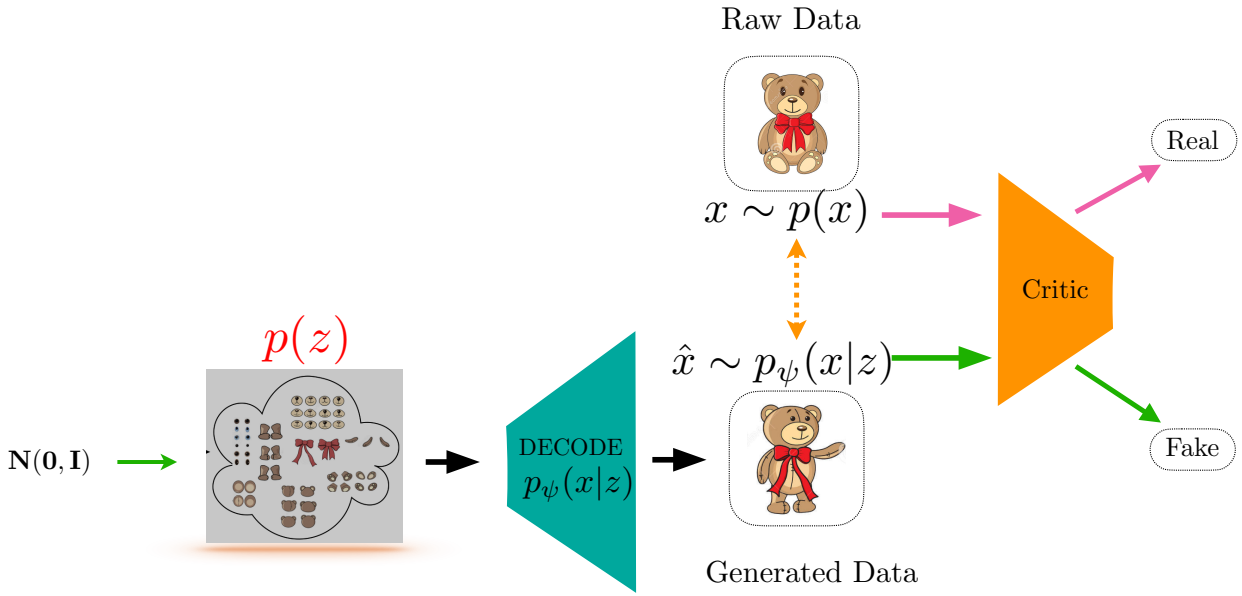
- It needs to make use of raw data, the only resource we have.
- Like Encoder, it should be able to extract features from data (of course, normally with a NN)
  - But only for a much simpler task: binary classification

Requirements for Critic



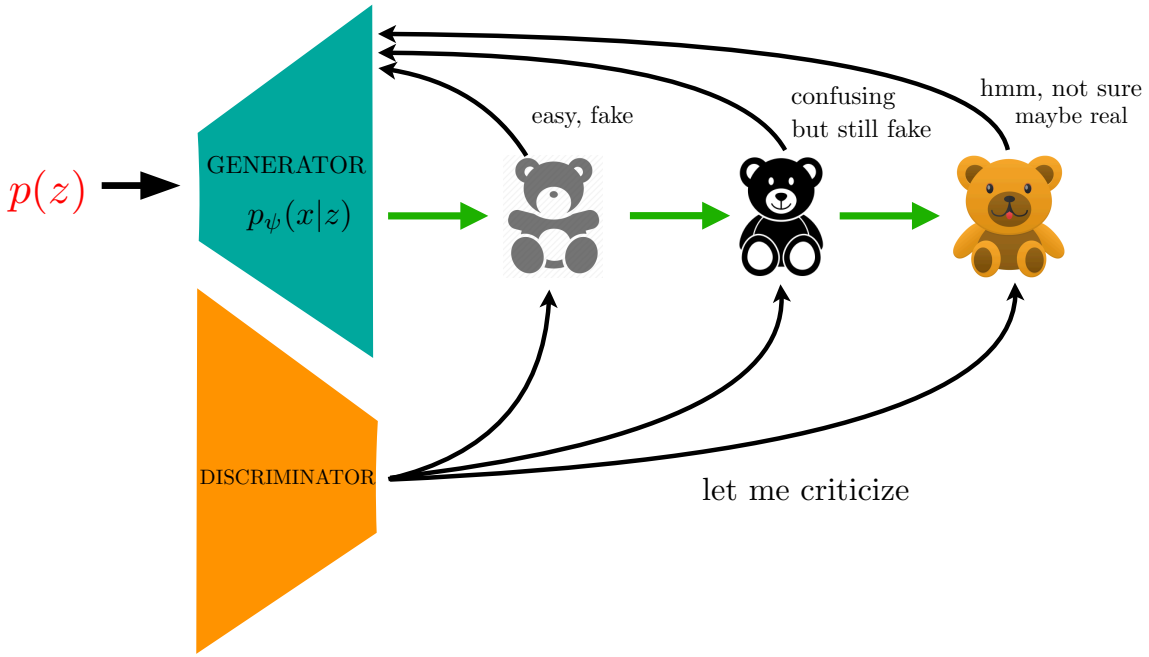
- It needs to make use of raw data, the only resource we have.
- Like Encoder, it should be able to extract features from data (of course, normally with a NN)
  - But only for a much simpler task: binary classification
  - So it looks like we have shifted Encoder from front to back

Requirements for Critic



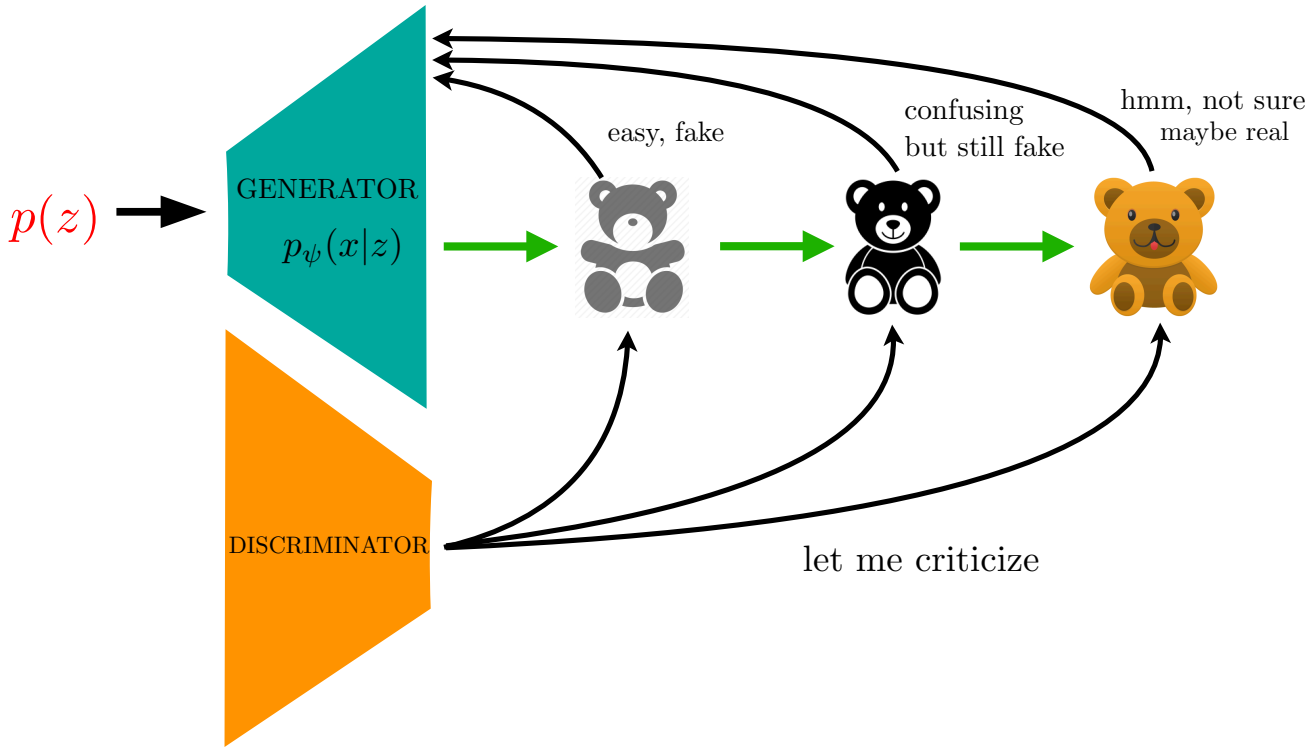
- Binary classification: isn't it too simple?
- Not quite, if we include a feedback loop, in that
  - Assume Critic is strict, and perfect, and constantly gives "feedback"
  - Generator uses feedback to get better overtime

Call them Generator and Discriminator



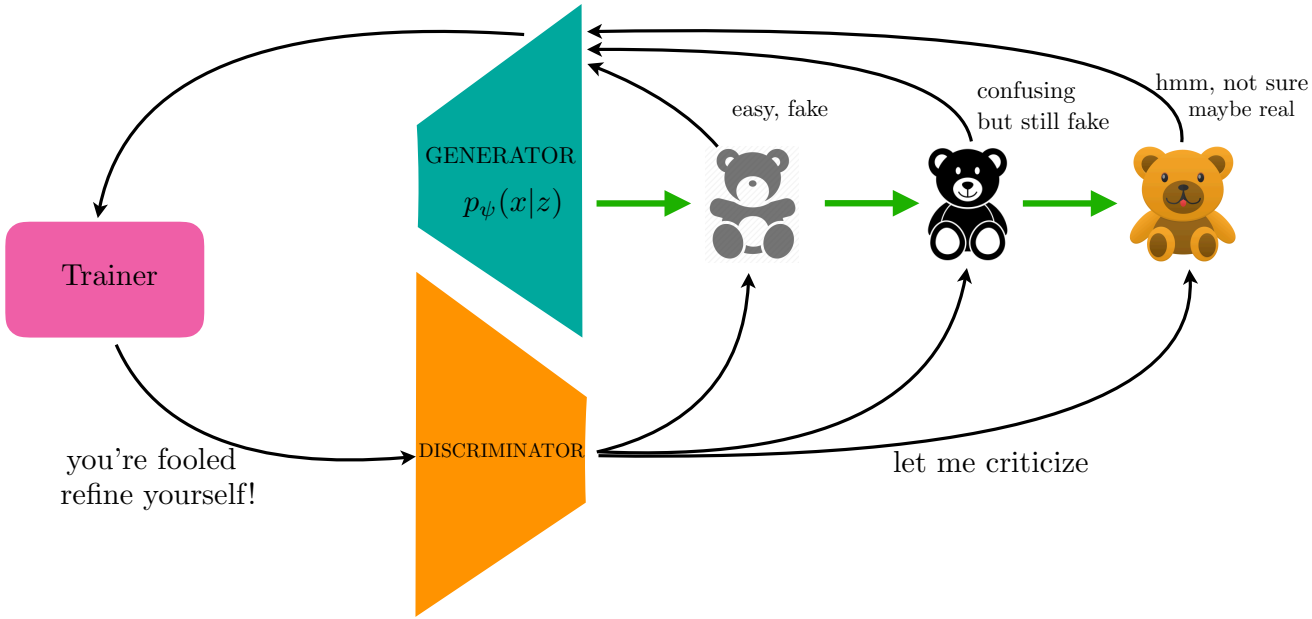
- Binary classification: isn't it too simple?
- Not quite, if we include a feedback loop, in that
  - Assume Critic is strict, and effective, and constantly gives "feedback"
  - Generator uses feedback to get better overtime

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:

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

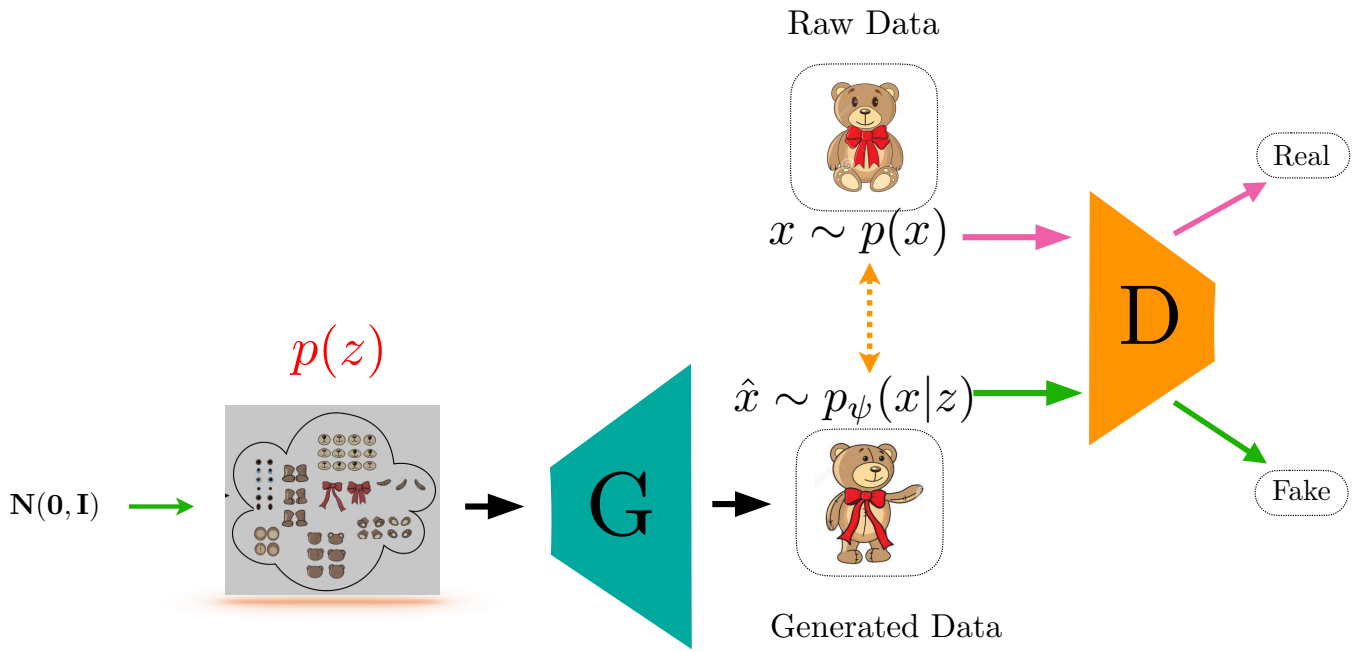
- Generator: tries to **minimize** that term:

$$\min_\psi [\mathbb{E}_{x\sim p(x)} \log\, D_\theta(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_\theta(G_\psi(z))] \tag{2}$$

$$= \min_\psi \mathbb{E}_{z\sim p(z)} \log(1 - D_\theta(G_\psi(z))] \tag{3}$$

Outline

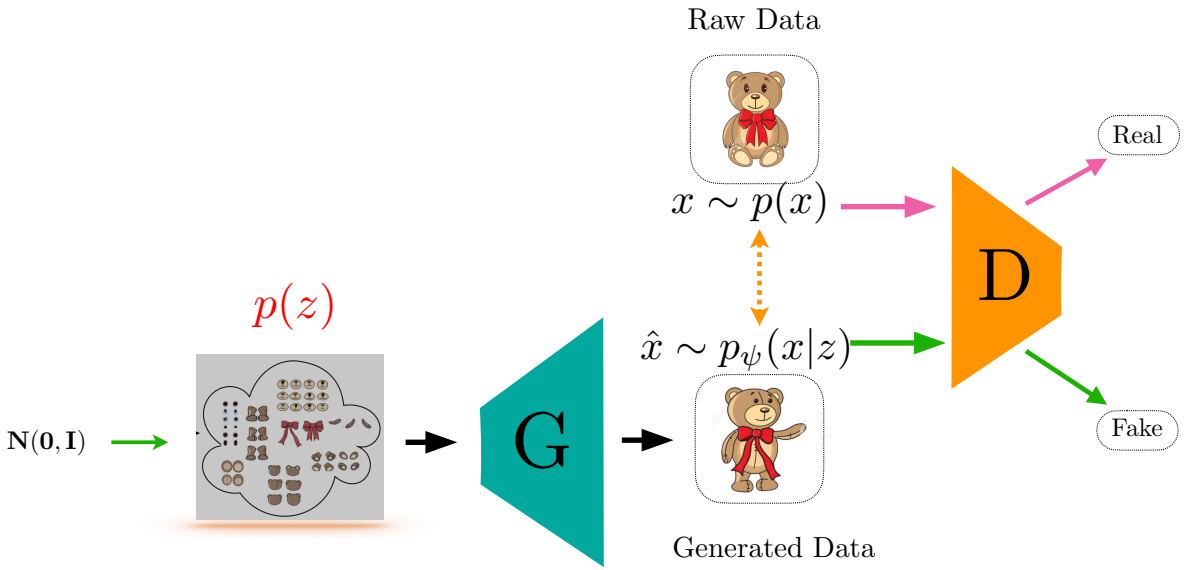
GAN[1]: We have two networks



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

[1] https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Objective for Training



- Objective of a minimax game between G and D:

$$\min_{\psi} \max_{\theta} [\mathbb{E}_{x \sim p(x)} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\psi(z))] \quad (4)$$

- We alternate in that:
  - Make G constant, let D **maximize** the objective
  - Make D constant, let G **minimize** the objective

Some notes about objective

- Objective of a minimax game between G and D:

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

- For a fixed G, the optimal discriminator D is:

$$D^*(x) = \frac{p_d(x)}{p_d(x) + p_g(x)} \tag{5}$$

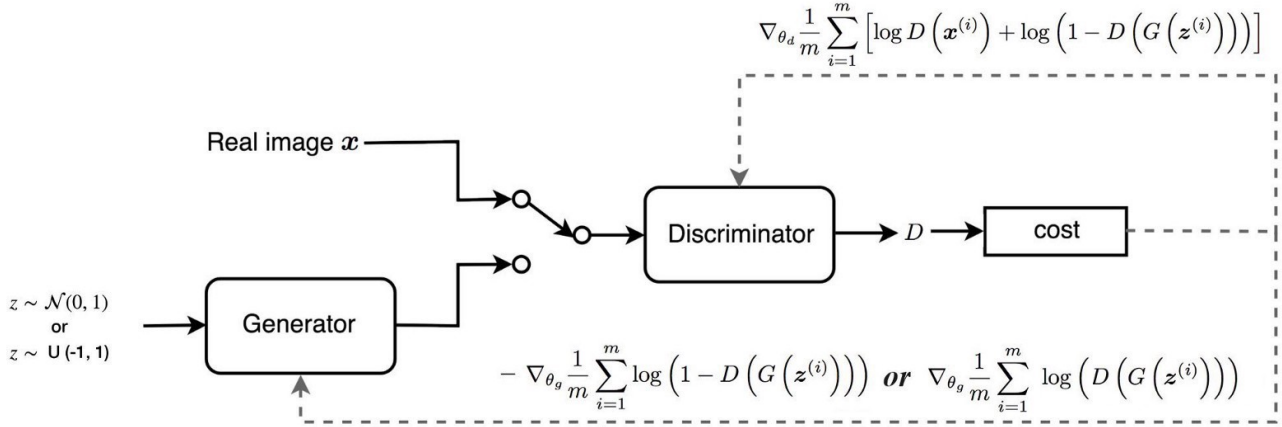
- At the optimal D^*, the objective of G is equivalent to:

$$C(G) = -\log(4) + \mathbf{D}_{KL}\left(p_d \| \frac{p_d + p_g}{2}\right) + \mathbf{D}_{KL}\left(p_g \| \frac{p_d + p_g}{2}\right) \tag{6}$$

$$= -\log(4) + 2\,\mathbf{D}_{JS}(p_d \| p_g) \tag{7}$$

- So the global optimum happens when $p_g == p_d$.

Backprop and Batching

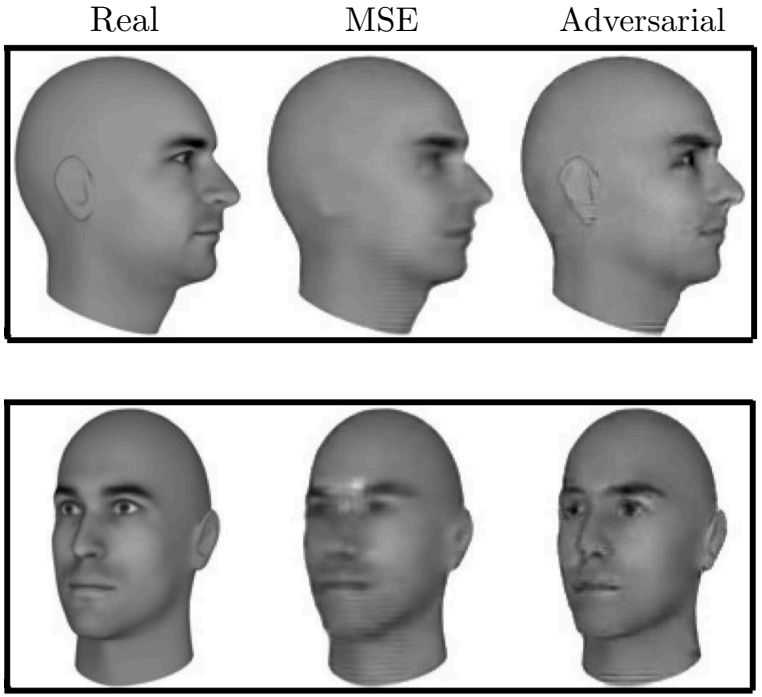


- Empirical expectation for gradient of D_θ:
  $\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} [\log D_\theta(x^{(i)}) + \log(1 - D_\theta(G_\psi(z^{(i)})))]$
- Empirical expectation for gradient of G_ψ:
  $\nabla_\psi \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D_\theta(G_\psi(z^{(i)})))]$
- Batching for D_θ: half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

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

High-resolution Image Generation

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