“Creative” Neural Networks

The Hitchhiker’s Guide to ~~Deepfakes~~ Generative Modelling

Artem Maevskiy

Research Fellow – Institute for Functional Intelligent Materials @ National University of Singapore

MLHEP-2023

Erice, Apr 11 – 18, 2023

NUS National University of Singapore | Institute for Functional Intelligent Materials

INFN National Institute for Nuclear Physics

Practicum

EMFCSC — Ettore Majorana Foundation and Centre for Scientific Culture

Outline

- Intro:
  - Generative modelling: what it is and how it’s useful
  - Generative modelling: how it’s done
- Generative Adversarial Networks (GAN)
- Variational Autoencoders (VAE)
- Normalizing Flows (NF)
- Evaluating Generative Models

Variational Autoencoders

Modelling the probability density

- Try mapping latent code space to object space $\mathcal{Z} \to \mathcal{Y}$ with a NN model
- Latent code: $z \sim p_z$ (sampled from some fixed distribution)
- We can treat our model as a model for the probability density, e.g.:

$$p_\theta(y \mid z) = \mathcal{N}(y \mid \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

I.e. the network generates not just a single object, but rather the average object for the given latent code z

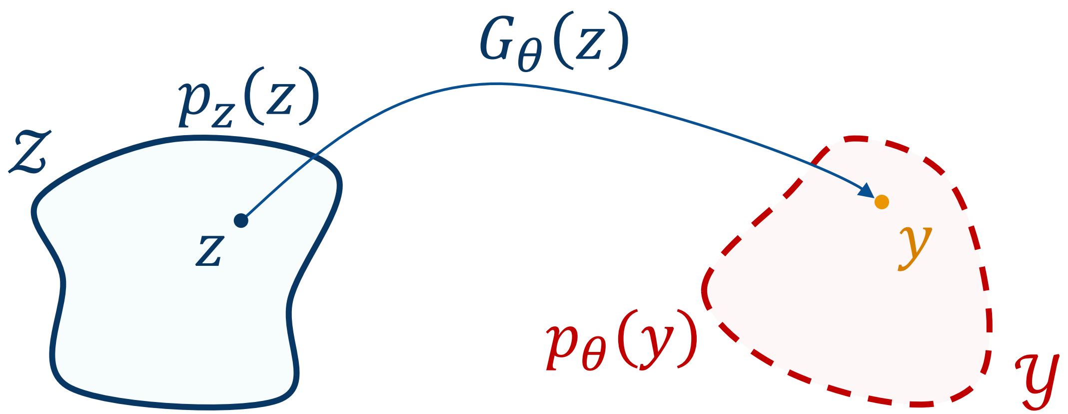
fixed parameter

$$p_\theta(y) = \int p_\theta(y, z) dz = \int p_\theta(y \mid z) p_z(z) dz = \mathop{\mathbb{E}}_{z \sim p_z} p_\theta(y \mid z)$$

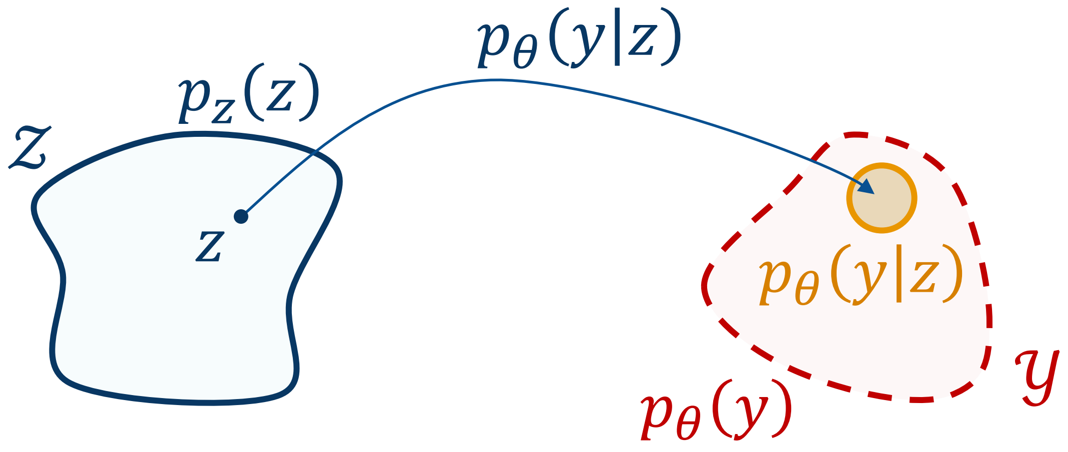
prior on z

Comparing with GAN generator

GAN generator:



VAE decoder:



How to train it?

- Want to maximize likelihood on the training data:

$$\mathop{\mathbb{E}}_{y \sim p_{\mathrm{data}}} p_\theta(y) \to \max_\theta$$

- The problem is, we only know how to calculate $p_\theta(y|z)$ and $p_z(z)$, but not the integral:

$$p_\theta(y) = \int p_\theta(y \mid z) p_z(z) dz$$

The posterior

- Assume we're able to calculate (and sample from) the posterior $p_\theta(z|y)$
- Note that:

$$\log p_\theta(y) = \mathbb{E}_{z\sim p_\theta(z|y)} \log\left[p_\theta(y)\frac{p_\theta(z|y)}{p_\theta(z|y)}\right]$$

$$= \mathbb{E}_{z\sim p_\theta(z|y)}[\log p_\theta(y,z) - \log p_\theta(z|y)]$$

$$= \underbrace{\mathbb{E}_{z\sim p_\theta(z|y)} \log p_\theta(y,z)} + \underbrace{\mathcal{H}\left(p_\theta(z|y)\right)}$$

So, for the log-likelihood we're sampling not all z values, but only those corresponding to this particular y

Maximizing this encourages placing high probability mass on many z values that could've generated y

Approximate inference

- In practice, $p_\theta(z|y)$ is typically intractable
- Let's try to approximate it with another parametric distribution $q_\phi(z|y)$
  - E.g., $q_\phi(z|y) = \mathcal{N}\left(z\middle|\mu_\phi(y), \mathbb{I}\sigma_\phi^2(y)\right)$, where μ_ϕ and σ_ϕ^2 are outputs of a neural network
- And use it for the likelihood calculation, i.e.:

$$[\log p_\theta(y)]_{\mathrm{approx.},\phi} = \mathbb{E}_{z\sim q_\phi(z|y)} \log p_\theta(y, z) + \mathcal{H}\left(q_\phi(z|y)\right)$$

Approximate inference

- Let's check how bad this approximation is:

$$\log p_\theta(y) - [\log p_\theta(y)]_{\text{approx.},\phi} =$$

$$= \log p_\theta(y) - \mathbb{E}_{z\sim q_\phi(z|y)} \log p_\theta(y, z) - \mathcal{H}\left(q_\phi(z|y)\right)$$

$$= \mathbb{E}_{z\sim q_\phi(z|y)}\left[\log p_\theta(y) - \log p_\theta(y, z) + \log q_\phi(z|y)\right]$$

$$= \mathbb{E}_{z\sim q_\phi(z|y)}\left[-\log p_\theta(z|y) + \log q_\phi(z|y)\right]$$

$$= D_{\mathrm{KL}}\left(q_\phi(z|y) \| p_\theta(z|y)\right) \geq 0$$

Approximate inference

- We've shown that:

$$\log p_\theta(x) - [\log p_\theta(x)]_{\text{approx.},\phi} = D_{\text{KL}}\big(q_\phi(z|x) \| p_\theta(z|x)\big) \geq 0$$

- I.e., our approximate log-likelihood is the **lower bound** for the true log-likelihood
  - Also called evidence lower bound (ELBO) or variational lower bound
- The better q approximates the posterior – the closer the bound is to the actual log-likelihood
- Also, if we maximize the lower bound, we'll maximize the likelihood as well!

Alternative form

$$\mathrm{ELBO} = [\log p_\theta(y)]_{\text{approx.},\phi} = \mathbb{E}_{z\sim q_\phi(z|y)} \log p_\theta(y,z) + \mathcal{H}\left(q_\phi(z|y)\right)$$

$$= \mathbb{E}_{z\sim q_\phi(z|y)}\left[\log p_\theta(y|z) + \log p_z(z) - \log q_\phi(z|y)\right]$$

$$= \underbrace{\mathbb{E}_{z\sim q_\phi(z|y)} \log p_\theta(y|z)}_{\textbf{Data term}} - \underbrace{D_{KL}\left(q_\phi(z|y) \| p_z(z)\right)}_{\textbf{Regularizer}} \rightarrow \max_{\theta,\phi}$$

A few details

- Simplest choices for $p_z(z)$, $p_\theta(y|z)$ and $q_\phi(z|y)$:

$$p_z(z) = \mathcal{N}(z|\, 0, \mathbb{I})$$

$$p_\theta(y\mid z) = \mathcal{N}(y|\, \mu = G_\theta(z), \Sigma = \mathbb{I}\sigma^2)$$

$$q_\phi(z|y) = \mathcal{N}\left(z\middle|\, \mu_\phi(y), \mathbb{I}\sigma_\phi^2(y)\right)$$

- KL divergence for such case can be calculated analytically:

$$D_{KL}\left(\mathcal{N}(\mu_1, \sigma_1^2) \| \mathcal{N}(\mu_2, \sigma_2^2)\right) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_2 - \mu_1)^2}{2\sigma_2^2} - \frac{1}{2}$$

A few details

$$\underbrace{\mathbb{E}_{z\sim q_\phi(z|y)} \log p_\theta(y|z)}_{\text{Data term}} - \underbrace{D_{KL}\big(q_\phi(z|y)\big\|p(z)\big)}_{\text{Regularizer}} \to \max_{\theta,\phi}$$

- Sample y from training set
- Sample z from $q_\phi(z|y)$
- Calculate data term and regularizer
- Backpropagate + gradient ascent step

A few details

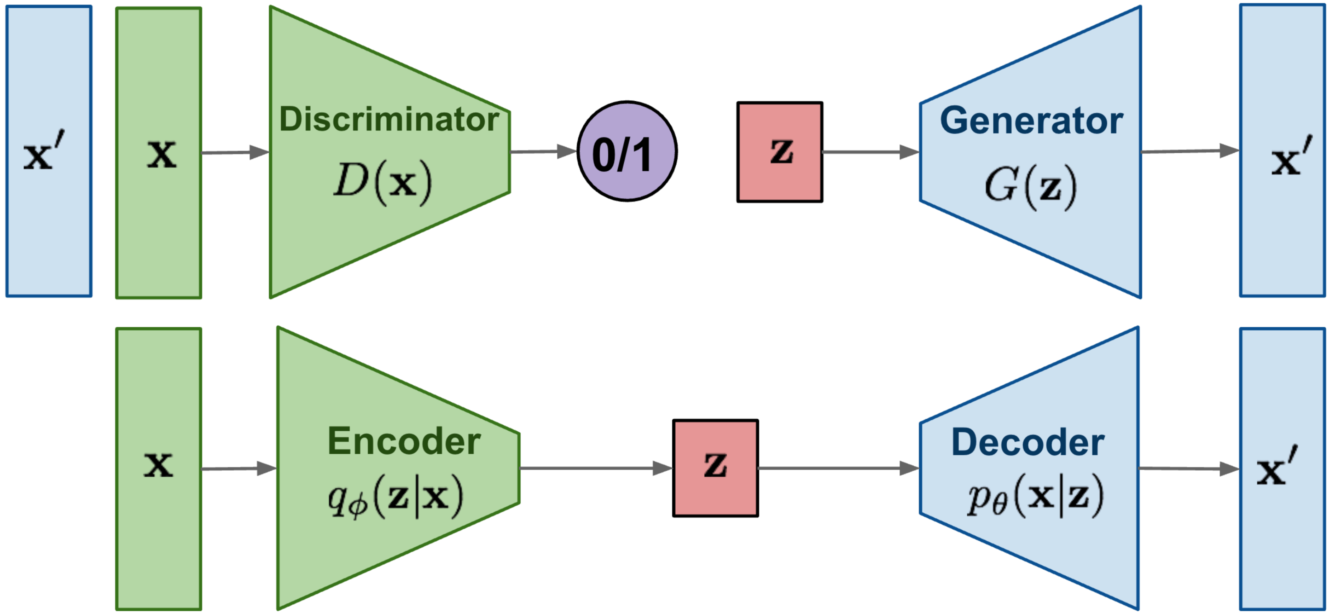
$$\underbrace{\mathbb{E}_{z\sim q_\phi(z|y)} \log p_\theta(y|z)}_{\text{Data term}} - \underbrace{D_{KL}\big(q_\phi(z|y)\|p(z)\big) \to \max_{\theta,\phi}}_{\text{Regularizer}}$$

- Sample y from training set
- Sample z from $q_\phi(z|y)$ ← How to backpropagate through this step?
- Calculate data term and regularizer
- Backpropagate + gradient ascent step

Important difference between VAE & GAN

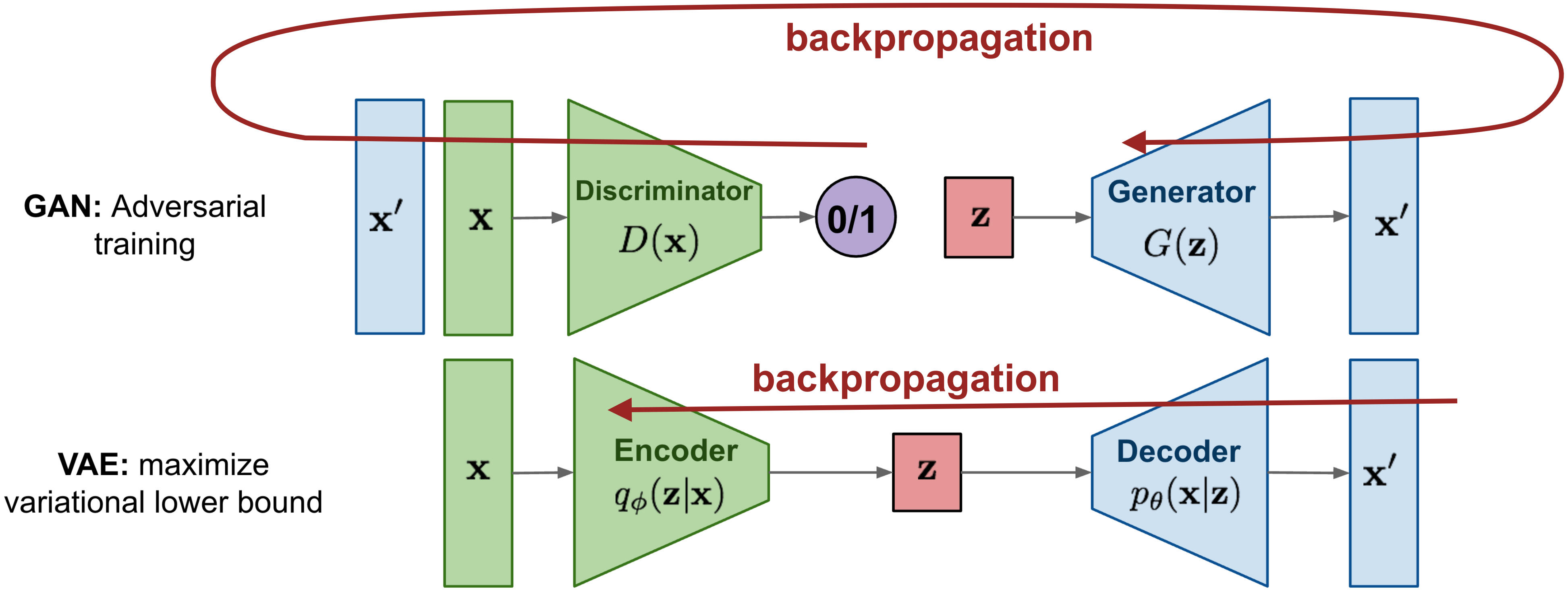
GAN: Adversarial training

VAE: maximize variational lower bound



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Important difference between VAE & GAN



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

VAE vs GANs

- In the tasks of image generation GANs are typically better
  - VAEs tend to produce blurry results due to the nature of the MSE loss
  - Note that MSE loss between images does not reflect our perception of image quality or similarity:

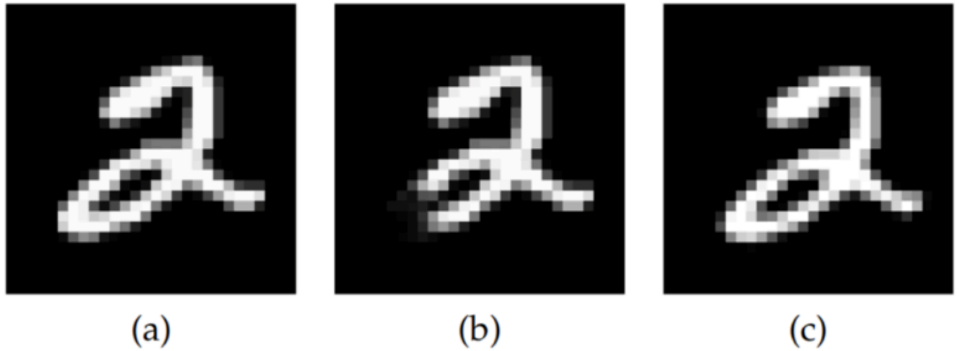


Image (b) — slightly altered image (a), image (c) — image (a) shifted by several pixels.
Under MSE metric, image (b) is much closer to (a), than (c) to (a).

  - There are some further advancements in VAEs that perform better (e.g., adversarial VAE)

VAEs vs GANs

- VAE is easier to train – no min-max game, just a single optimization objective
- The encoder gives you the mapping from objects to the latent representation
  - This lets you do things like interpolation between objects, analyzing latent space, etc.
- VAEs give you explicit access to the estimated data PDF