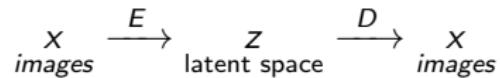


# Variational Autoencoder (VAE)

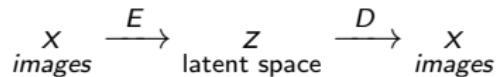
An *autoencoder*:



*variational*: trained in a specific way.

# Variational Autoencoder (VAE)

An *autoencoder*:



*variational*: trained in a specific way.

**Images**  $x \in \{0, 1\}^{28 \times 28}$  (can be discrete).

**Latent space vectors**  $z \in \mathbb{R}^d$  (continuous).

# Variational Autoencoder (VAE)



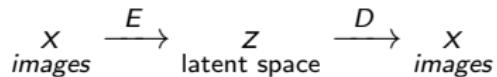
Two neural networks:

(We use  $X, Z$  for placeholders and  $x, z$  for actual values here).

**Encoder**  $E(x)$  outputs a distribution  $q_{\text{model}}(Z|x)$  of latents  $z$ , described by mean  $\mu$  and stddev  $\sigma$  for each of the  $d$  entries (a factorial distribution of  $d$  normal variables).

**Decoder**  $D(z)$  outputs a distribution  $p_{\text{model}}(X|z)$  of images  $x$ , described by outputting probabilities in  $[0, 1]$  for each pixel (a factorial distribution of height  $\times$  width bernoulli variables).

# Variational Autoencoder (VAE)



**Encoder**  $E(x)$  outputs a distribution  $q_{\text{model}}(Z|x)$  of latents  $z$ .

**Decoder**  $D(z)$  outputs a distribution  $p_{\text{model}}(X|z)$  of images  $x$ .

**Prior** distribution over latents:  $p_{\text{prior}}(Z) = \mathcal{N}(0, I^d)$  (fixed).

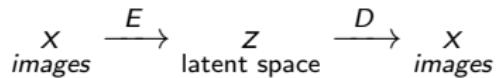
To generate images:

sample  $z \sim p_{\text{prior}}(Z)$ , then  $x \sim p_{\text{model}}(X|z)$ .

This gives a distribution over images  $p_{\text{model}}(X)$ :

$$p_{\text{model}}(x) := \mathbb{E}_{z \sim p_{\text{prior}}} p_{\text{model}}(x|z) = \int_z p_{\text{prior}}(z) p_{\text{model}}(x|z)$$

# Variational Autoencoder (VAE)



**Encoder**  $E(x)$  outputs a distribution  $q_{\text{model}}(Z|x)$  of latents  $z$ .

**Decoder**  $D(z)$  outputs a distribution  $p_{\text{model}}(X|z)$  of images  $x$ .

**Prior** distribution over latents:  $p_{\text{prior}}(Z) = \mathcal{N}(0, I^d)$  (fixed).

This gives a distribution over images  $p_{\text{model}}(X)$ :

$$p_{\text{model}}(x) := \mathbb{E}_{z \sim p_{\text{prior}}} p_{\text{model}}(x|z) = \int_z p_{\text{prior}}(z) p_{\text{model}}(x|z)$$

but also a **posterior** distribution  $p_{\text{model}}(Z|x)$ :

$$p_{\text{model}}(z|x) := \frac{p_{\text{model}}(x|z)p_{\text{prior}}(z)}{p_{\text{model}}(x)}$$

## How to optimize the decoder $p$ ?

If we just want to model the image distribution, we'd like to maximize the log likelihood, over params of decoder  $p_{\text{model}}(X|z)$ :

$$\text{log-likelihood} = \log p_{\text{model}}(\text{dataset}) = \mathbb{E}_{x \sim \text{dataset}} \log p_{\text{model}}(x)$$

We'll sample  $x \sim \text{dataset}$  and skip  $\mathbb{E}_{x \sim \text{dataset}}$  hereafter.

Unfortunately, using  $p_{\text{model}}(x) = \mathbb{E}_{z \sim p_{\text{prior}}} p_{\text{model}}(x|z)$  we'll usually just find  $z$  where  $p(x|z)$  is tiny (giving a bad estimate of  $p(x)$ ) and we'll try to increase it in random places everywhere, instead of learning some association between  $x$  and  $z$ .

So we want the encoder  $q_{\text{model}}(Z|x)$  as well  
(we often want it for other purposes, too).

## How to optimize the encoder $q$ ?

For a given  $x$ , the encoder should find  $z$  that are decoded to  $x$  with high probability. So arguably, we want  $q_{\text{model}}(Z|x)$  to be close to the posterior  $p_{\text{model}}(Z|x)$ . We can do this by minimizing

$$D_{KL}\left(q_{\text{model}}(Z|x) \parallel p_{\text{model}}(Z|x)\right) \geq 0.$$

That is, instead of maximizing just  $\log p_{\text{model}}(x)$ , we maximize the *evidence lower bound ELBO(x)* :=

$$\log p_{\text{model}}(x) - D_{KL}\left(q_{\text{model}}(Z|x) \parallel p_{\text{model}}(Z|x)\right) \leq \log p_{\text{model}}(x)$$

Here we could have an arbitrary weight in front of  $D_{KL}$  and usually you'd swap the arguments (usually you'd push the approximate  $q$  to have high mass where the 'true'  $p$  has high mass, instead of pushing  $q$  to have low mass where  $p$  has low mass).

Weight=1 and this  $D_{KL}$  order turns out to simplify the expression: expanding  $p_{\text{model}}(Z|x)$  gives a  $\log p_{\text{model}}(x)$  term that cancels out.



## ELBO – derivation

$$p(x) := \mathbb{E}_{z \sim p(Z)} p(x|z) = \int_z p(z)p(x|z)$$

$$p(z|x) := \frac{p(x|z)p(z)}{p(x)}$$

---

$$\begin{aligned} ELBO(x) &:= \log p(x) - D_{KL}\left(q(Z|x) \middle\| p(Z|x)\right) = \\ &= \log p(x) - \mathbb{E}_{z \sim q(Z|x)} [\log q(z|x) - \log p(z|x)] = \\ &= \log p(x) - \mathbb{E}_{z \sim q(Z|x)} [\log q(z|x) - \log p(x|z) - \log p(z) + \log p(x)] = \\ &= \mathbb{E}_{z \sim q(Z|x)} [-\log q(z|x) + \log p(x|z) + \log p(z)] = \\ &= \mathbb{E}_{z \sim q(Z|x)} \left[ \log \frac{p(x|z)p(z)}{q(z|x)} \right] = \quad (\leftarrow \text{as seen on lecture}) \\ &= \mathbb{E}_{z \sim q(Z|x)} [\log p(x|z)] - D_{KL}\left(q(Z|x) \middle\| p(Z)\right) \end{aligned}$$

## ELBO – interpretation

$$ELBO(x) := \log p(x) - D_{KL}(q(Z|x) \| p(Z|x)) = \quad (1)$$

$$= \mathbb{E}_{z \sim q(Z|x)} [\log p(x|z)] - D_{KL}(q(Z|x) \| p(Z)) \quad (2)$$

The summands have some similarities but they are very different, only their difference is equal.

The left summand of (2) can be interpreted as *reconstruction loss*, pushing both  $p$  and  $q$  so that encoding-then-decoding gives back  $x$ .

The right summand of (2) can be interpreted as a regularization pushing  $q(Z|x)$  to the prior  $p(Z)$  (spread out, continuous).

(2) is easy to compute! Estimate the left by sampling  $z \sim q(Z|x)$ .

Compute the right analytically, as a function of the means  $\mu$  and std devs  $\sigma$  describing distributions  $q(Z|x)$  and  $p(Z)$  (the distributions themselves are simple, it doesn't matter that  $\mu, \sigma$  are NN outputs, for computing  $D_{KL}$ ).

## ELBO – a third view

$$ELBO(x) := \log p(x) - D_{KL}(q(Z|x) \| p(Z|x)) = \quad (1)$$

$$= \mathbb{E}_{z \sim q(Z|x)} [\log p(x|z)] - D_{KL}(q(Z|x) \| p(Z)) \quad (2)$$

Recall that we actually maximize  $\mathbb{E}_{x \sim \text{dataset } \mathcal{D}} ELBO(x)$ .

We can think of two processes that generate a pair  $(x, z)$ :

- ▶  $z \sim p(Z)$ , then  $x \sim p(X|z)$ , joint dist.  $p(x, z) = p(x|z)p(z)$
- ▶  $x \sim \mathcal{D}$ , then  $z \sim q(Z|x)$ , joint dist.  $q(x, z) = q(z|x)\mathcal{D}(x)$

It is easy to check that

$$\mathbb{E}_{x \sim \mathcal{D}} ELBO(x) = -D_{KL}(q(X, Z) \| p(X, Z)) + \text{const}$$

where  $\text{const} = H(\mathcal{D})$  does not depend on parameters of  $p$  or  $q$ .

“sample  $x$  from data and encode  $\simeq$  sample  $z$  from prior and decode”

## Reparametrization trick

$$ELBO(x) := \log p(x) - D_{KL}\left(q(Z|x) \parallel p(Z|x)\right) = \quad (1)$$

$$= \mathbb{E}_{z \sim q(Z|x)} [\log p(x|z)] - D_{KL}\left(q(Z|x) \parallel p(Z)\right) \quad (2)$$

While maximizing (2), it's important to optimize over parameters of  $q$  in the left summand as well. If we just sampled  $z \sim q(Z|x)$ , gradients wouldn't flow through that operation.

The *reparametrization trick* solves this by expressing  $z \sim q(Z|x)$  as a (deterministic, differentiable) function of  $x$  and noise  $\epsilon$  independent of  $x$ :

$$z = \mu(x) + \sigma(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I^d)$$

where  $\mu(x), \sigma(x)$  are outputs of the NN encoder.

## Side notes

In Bayesian statistics, VAE is also viewed as a practical way to compute posterior distributions. In this setting,  $p(x|z)$  and  $p(z)$  can be fixed; the goal is to find params for  $q(Z|x)$  that approximate  $p(Z|x)$  well.

This is called *variational inference*.

*Variational* refers to making small changes in functions to find an optimal function or a good approximation of a complicated function (here, finding a simple approximation  $q$  of the posterior).

*Inference* in this context often refers to computing posterior distributions, as needed in statistical inference (e.g. we infer stuff about hidden variables  $z$  given observed data  $x$ , modelling observations as  $X = f(Z)$ ).

*Mean field approximation* is a term you might encounter that just refers to approximating complicated distributions with factorial distributions (like we did here, both for  $q(Z|x)$  and  $p(X|z)$ ).

## Summary

- ▶ VAE is an autoencoder: we generate with a decoder, but we also train an encoder to help with the training.
- ▶ The model is trained by maximizing ELBO, interpreted as:
  - ▶ log-likelihood for decoder and  $D(q\| \text{posterior})$  for encoder (1)
  - ▶ reconstruction loss and “spread-out” regularization  $D(q\| \text{prior})$  (2)
  - ▶ “sample data  $x$  and encode  $\simeq$  sample prior  $z$  and decode” (3)
- ▶ The reparametrization trick allows gradients to flow through  $\mathbb{E}_{z \sim \text{encoder}(x)}$ .