Deep Learning (CS324)

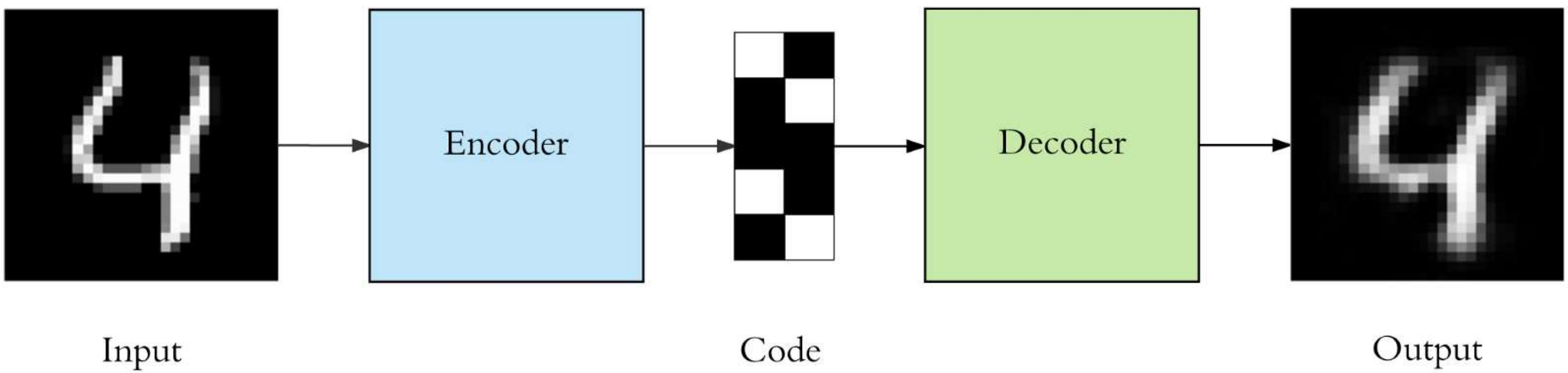
8. Variational autoencoders*

Prof Jianguo Zhang
SUSTech

*Based on https://jaan.io/what-is-variational-autoencoder-vae-tutorial/ and http://www.deeplearningbook.org chapter 20.10.3

Generation with autoencoders?

- Can we use an autoencoder to generate data given a code?



Ok. But why? Sample application

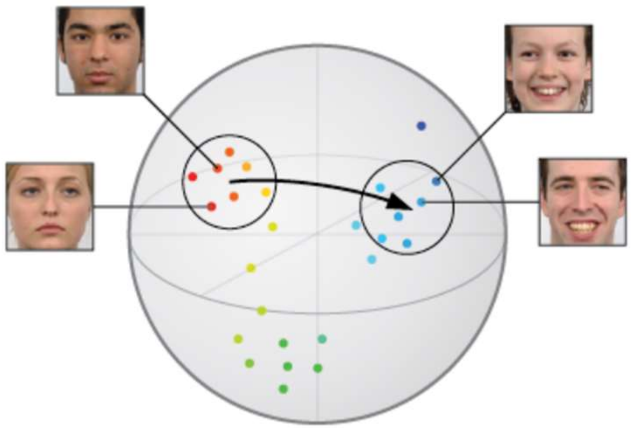
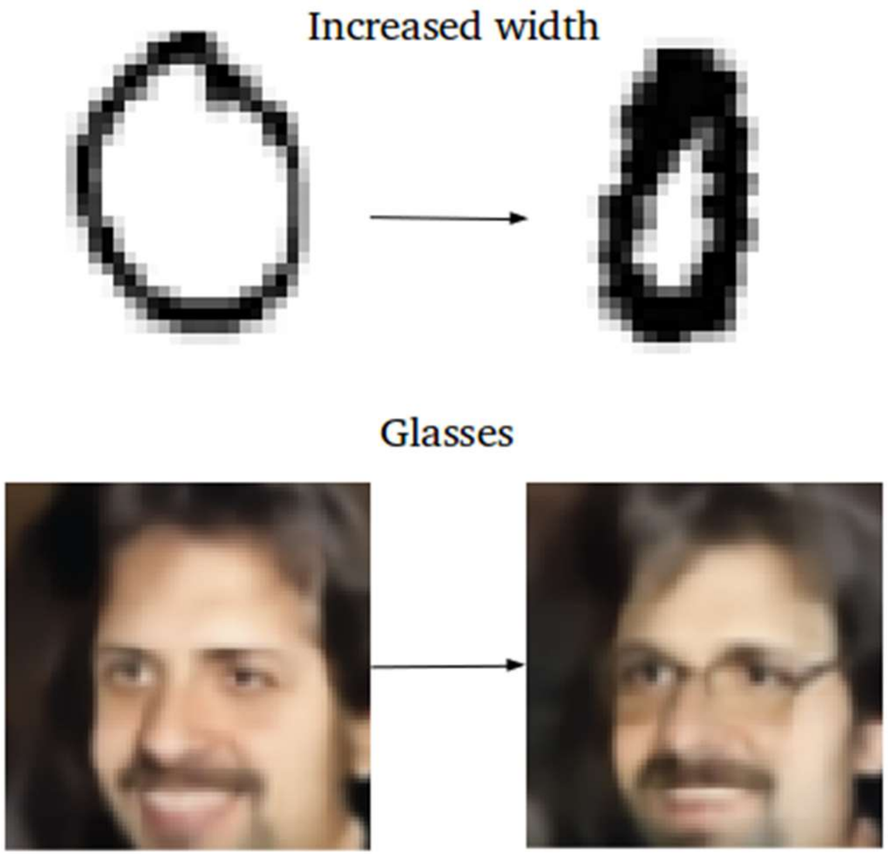


Figure 1: Schematic of the latent space of a generative model. In the general case, a generative model includes an encoder to map from the feature space (here images of faces) into a high dimensional latent space. Vector space arithmetic can be used in the latent space to perform semantic operations. The model also includes a decoder to map from the latent space back into the feature space, where the semantic operations can be observed. If the latent space transformation is the identity function we refer to the encoding and decoding as a reconstruction of the input through the model.

Exploring a specific variation of input data

Source: Sampling Generative Networks: https://arxiv.org/pdf/1609.04468.pdf

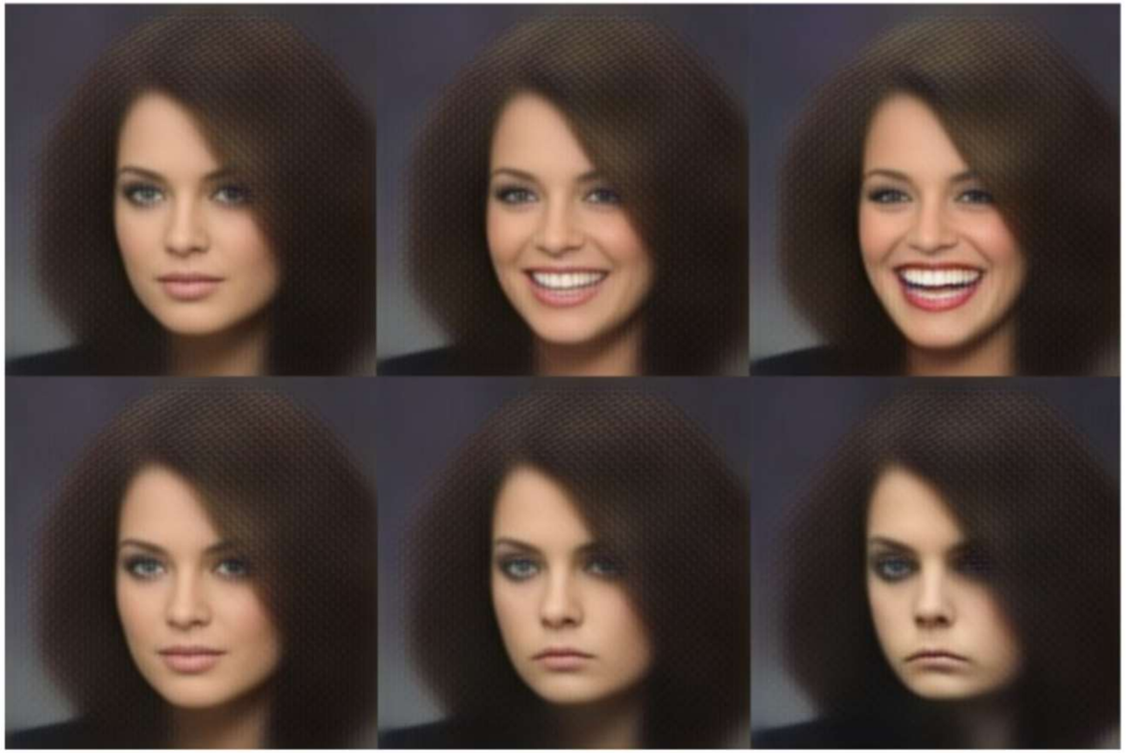
Ok. But why? Sample application



Exploring a specific variation of input data

Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

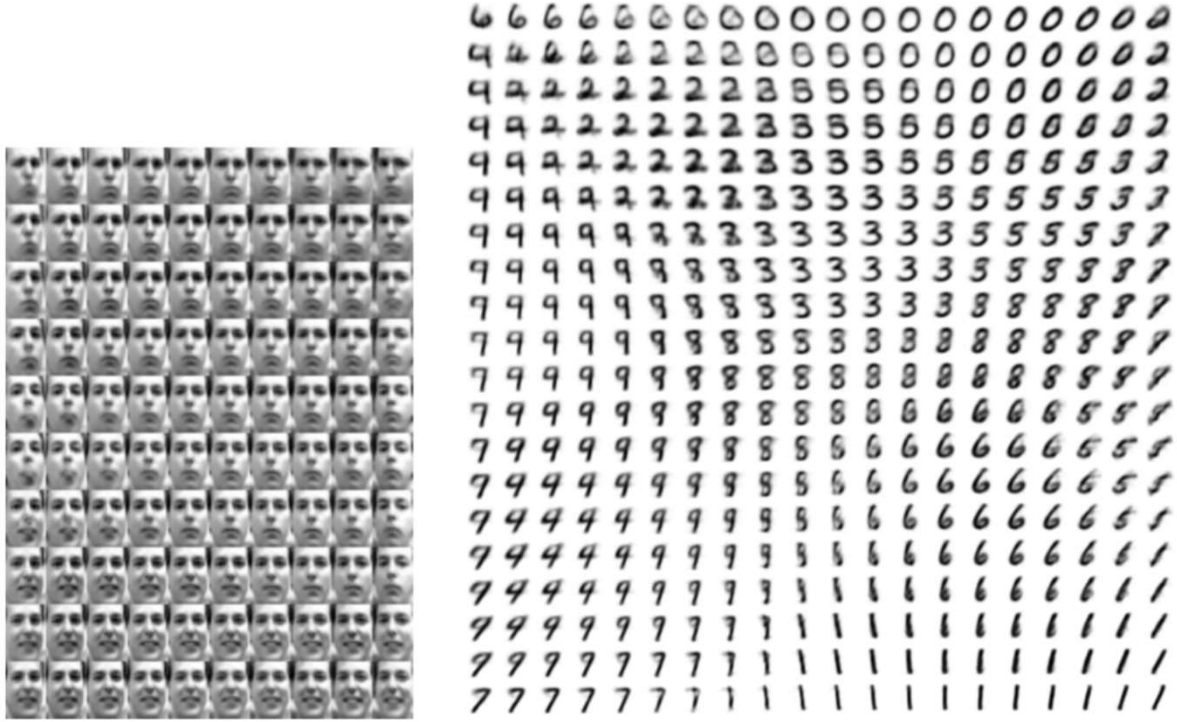
Ok. But why? Sample application



Exploring a specific variation of input data

Source: Sampling Generative Networks https://arxiv.org/pdf/1609.04468.pdf

Ok. But why? Sample application



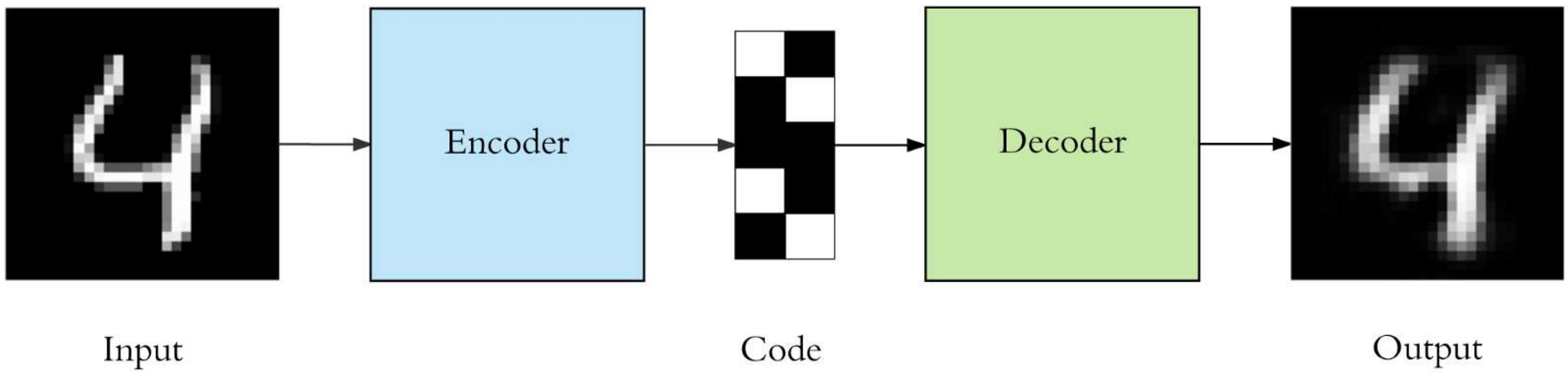
(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 2.7: Visualizations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Generation with autoencoders?

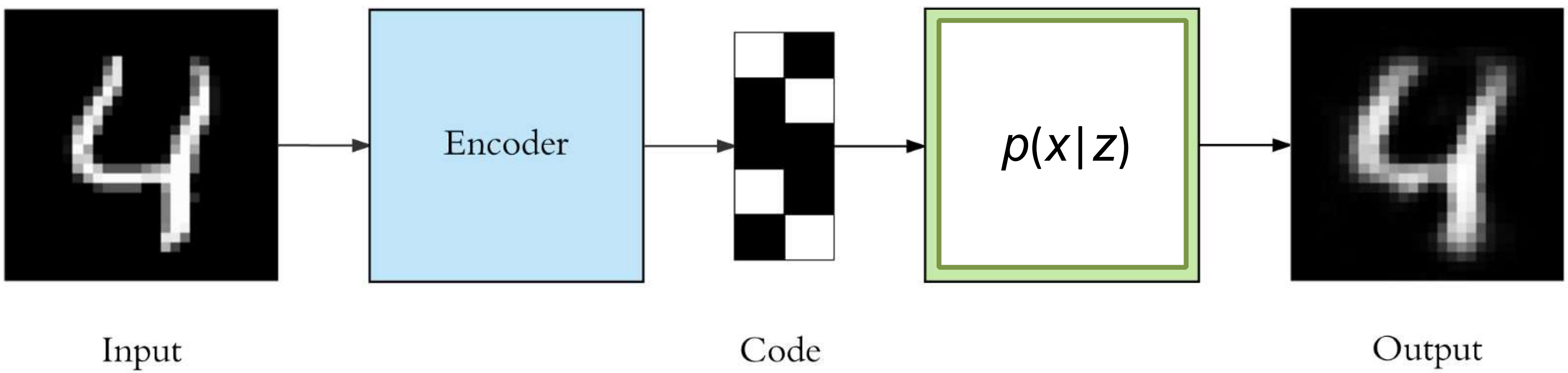
- Can we use an autoencoder to generate data given a code?



Kingma and Welling, Auto-encoding Variational Bayes, NIPS, 2013

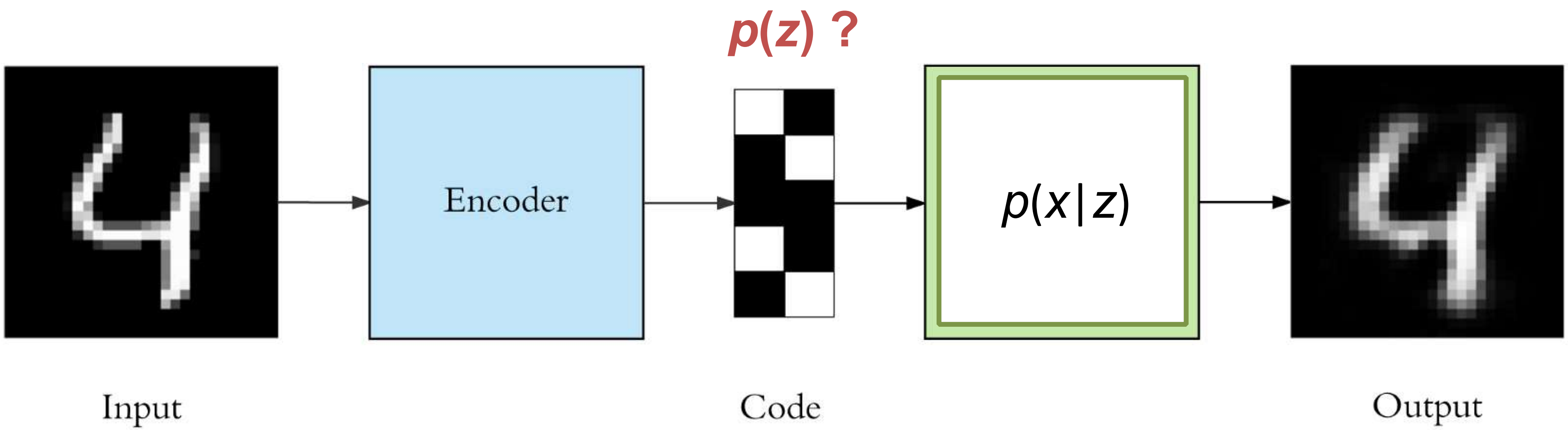
Generation with autoencoders?

- Can we use an autoencoder to generate data given a code?
- Yes, but…



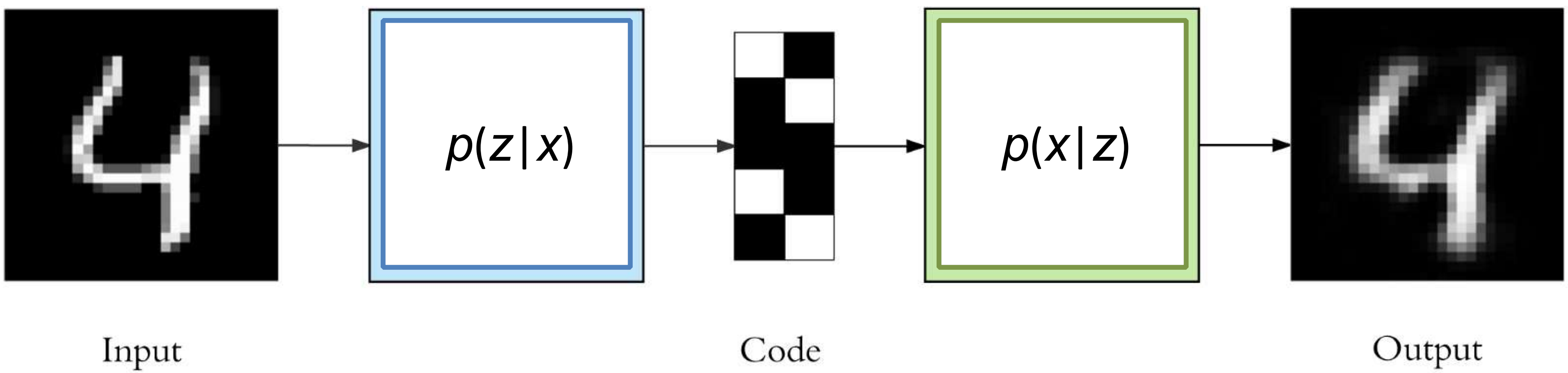
Generation with autoencoders?

- Can we use an autoencoder to generate data given a code?
- Yes, but…how to sample the code z first?



Generation with autoencoders?

- Can we use an autoencoder to generate data given a code?
- Yes, but…how to sample the code *z* first?

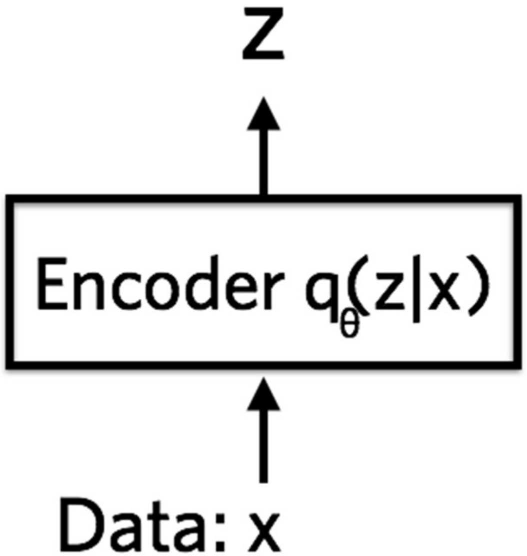


Variational autoencoder

- Let's have a closer look at our (variational) autoencoder components

VAEs: neural networks perspective

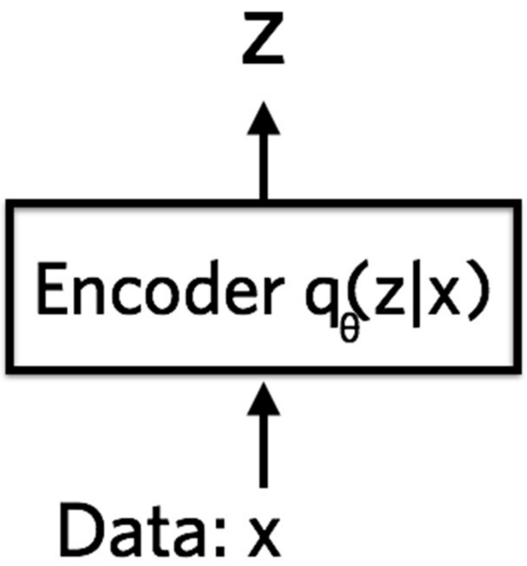
- Let's have a closer look at our (variational) autoencoder components



The Encoder is a neural network that takes a data point in input and outputs a hidden representation **z**, usually lower dimensional

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

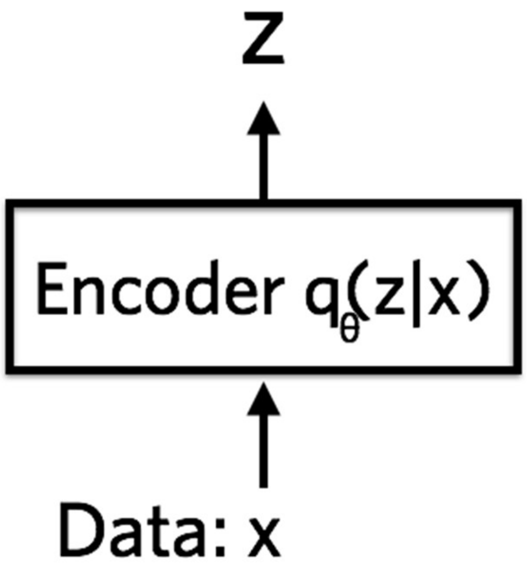


The Encoder is a neural network that takes a data point in input and outputs a hidden representation **z**, usually lower dimensional

More precisely, the Encoder outputs the parameters of a Gaussian probability density **$q_\theta(z|x)$**

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components



The Encoder is a neural network that takes a data point in input and outputs a hidden representation **z**, usually lower dimensional

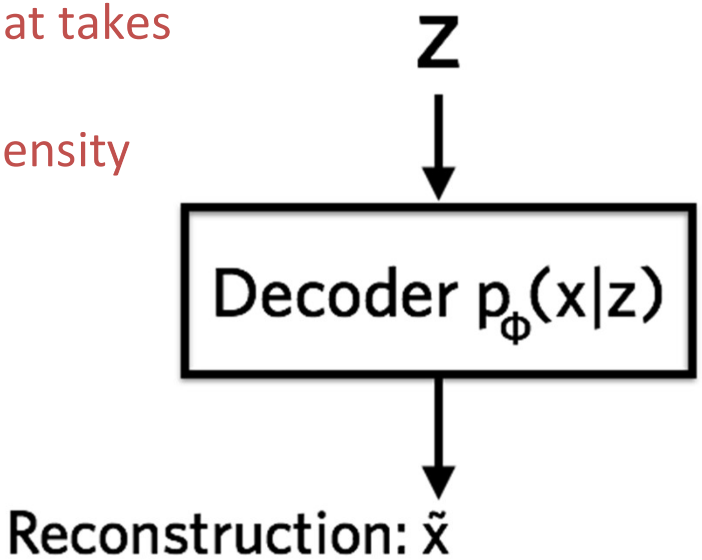
More precisely, the Encoder outputs the parameters of a Gaussian probability density **$q_\theta(z|x)$**

Then, we can sample a representation/code **z** from **$q_\theta(z|x)$**

VAEs: neural networks perspective

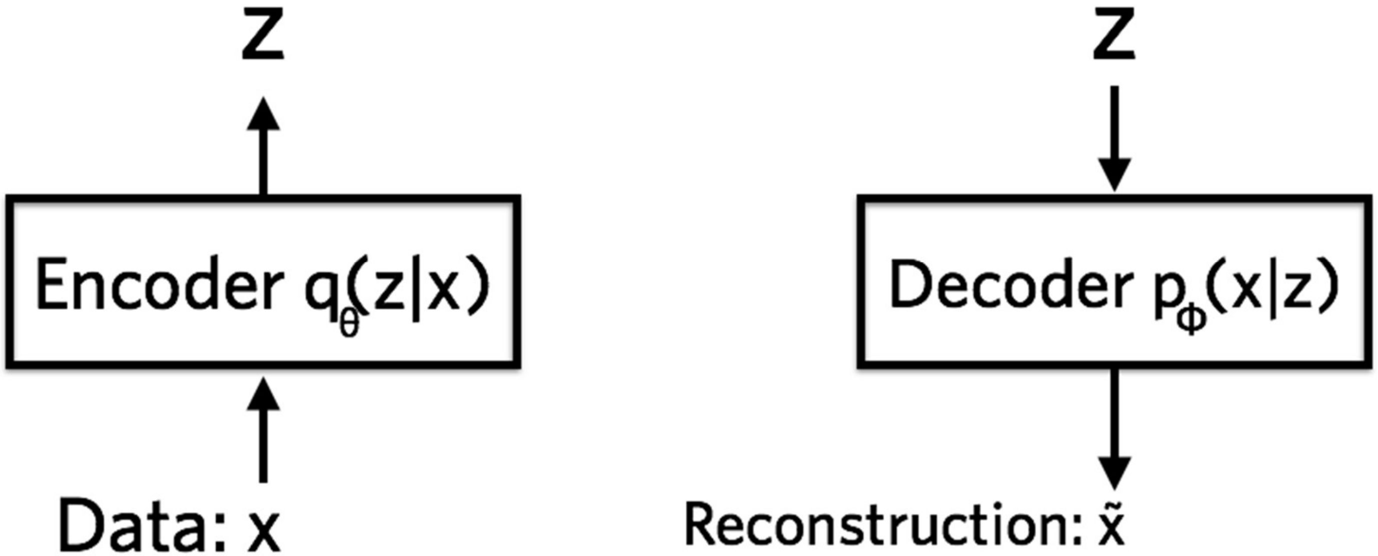
- Let's have a closer look at our (variational) autoencoder components

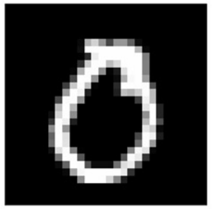
The Decoder is another neural network that takes the representation **z** in input and outputs the parameters of $\mathbf{p_{\phi}(x|z)}$, a probability density from which we can sample the data



VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

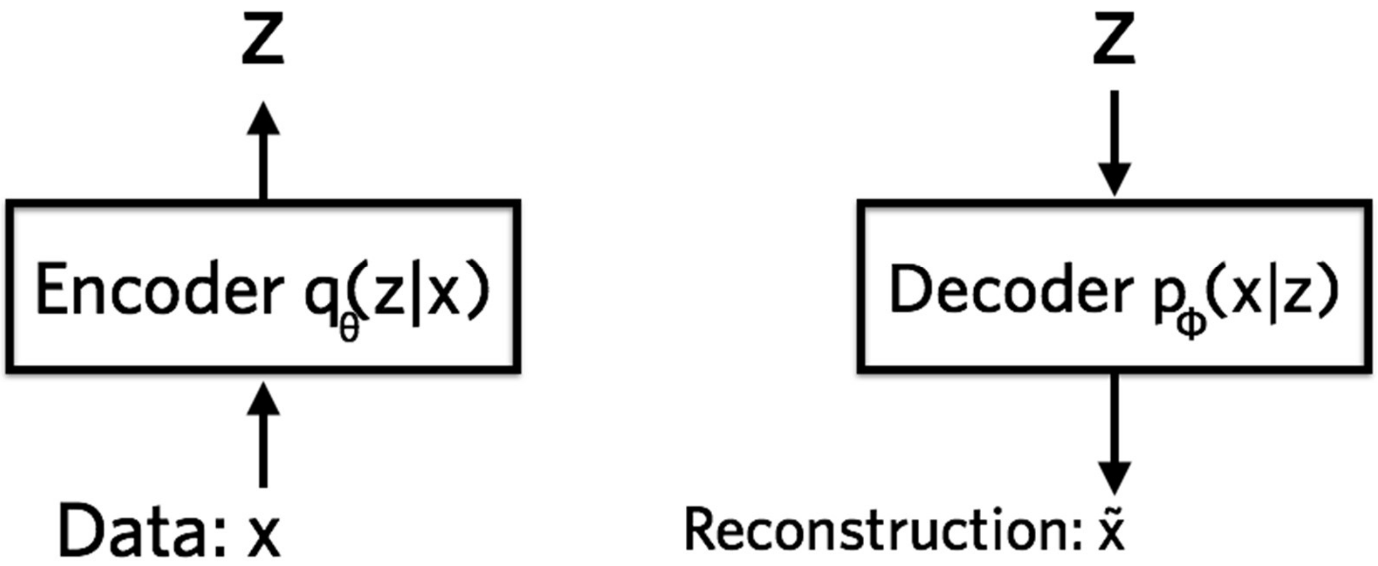




Input: 28 x 28 image of a digit
(784-dimensional binary image)

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

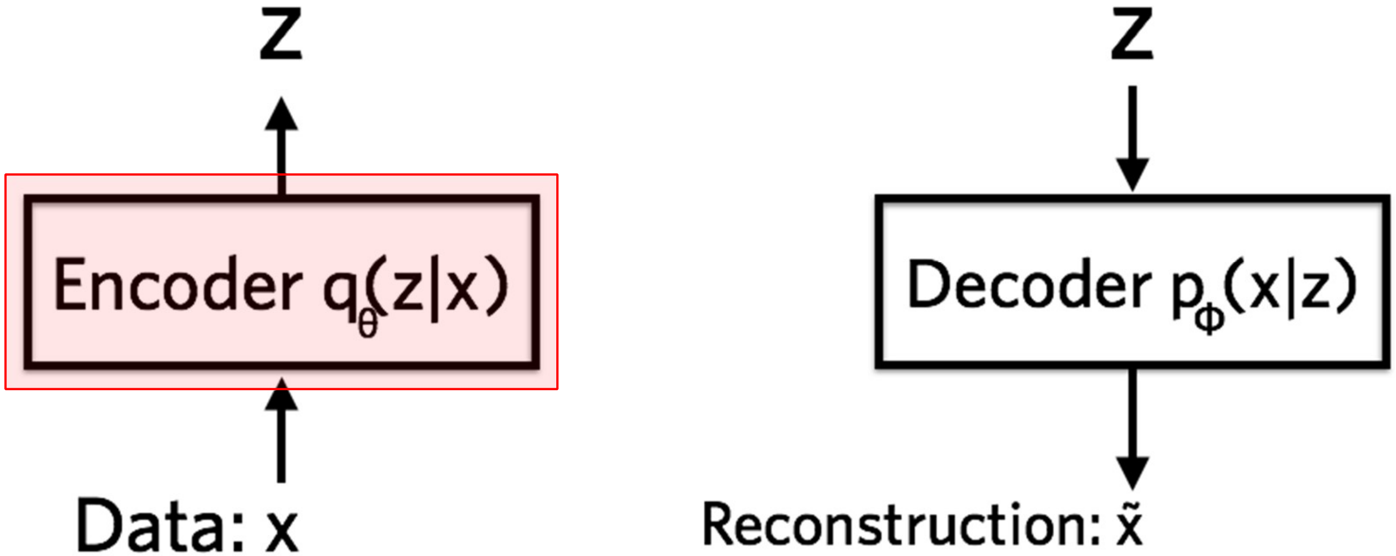


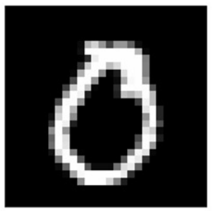
0 0 0 1
0 0 1 1
0 0 1 0 …

Input: 28 x 28 image of a digit
(784-dimensional binary image)

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

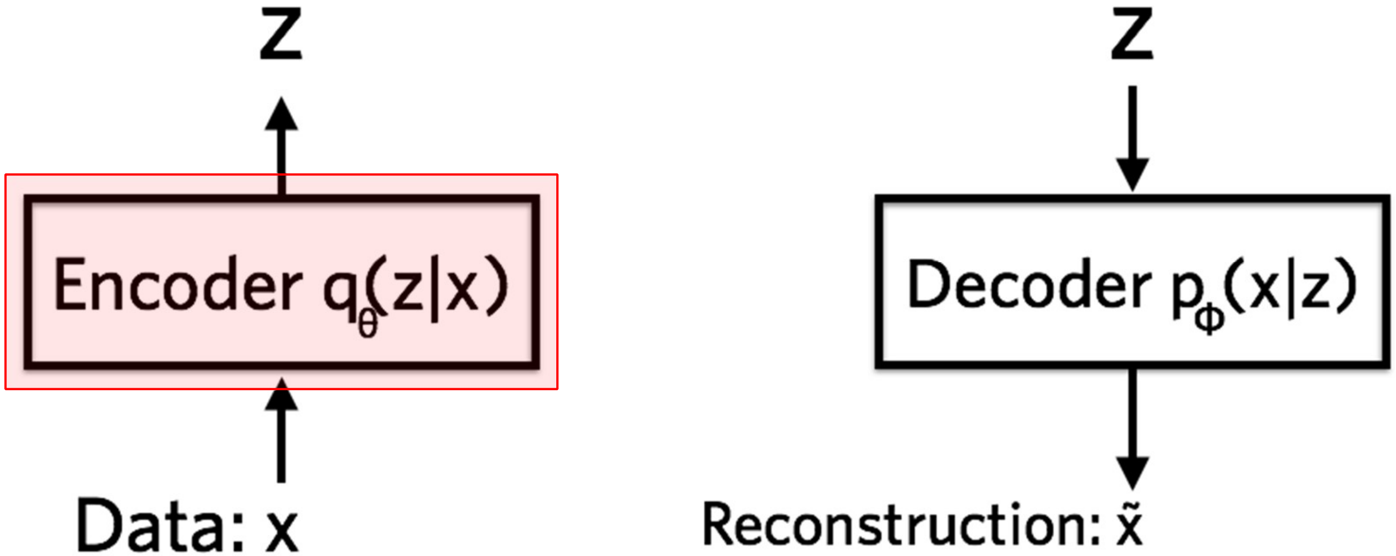


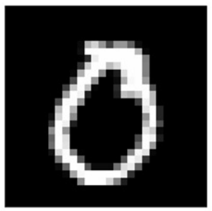


The input is fed to a network (the encoder) with its weights and biases (**θ**)

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

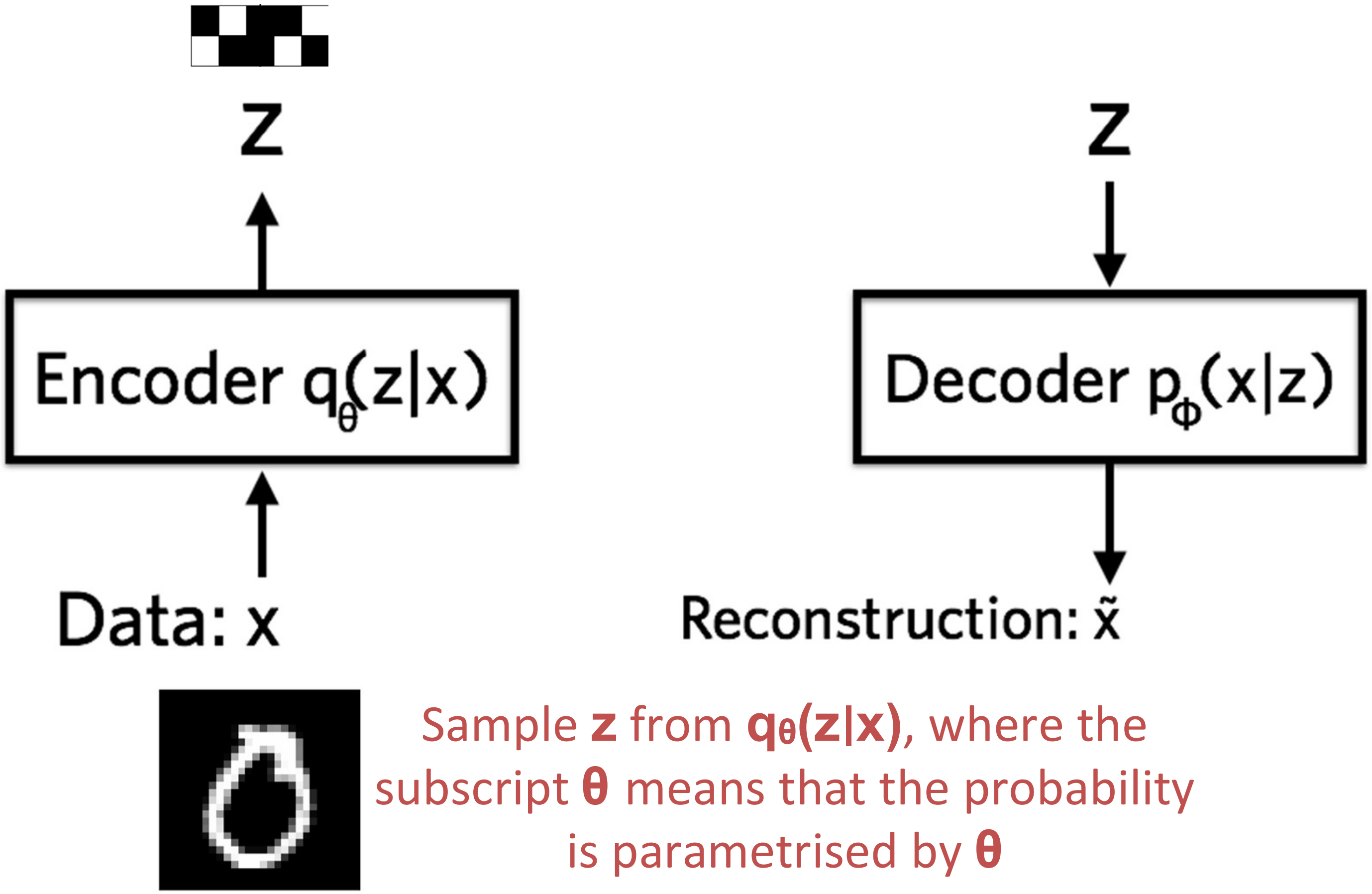




The network outputs mean and variance for **q_θ(z|x)**

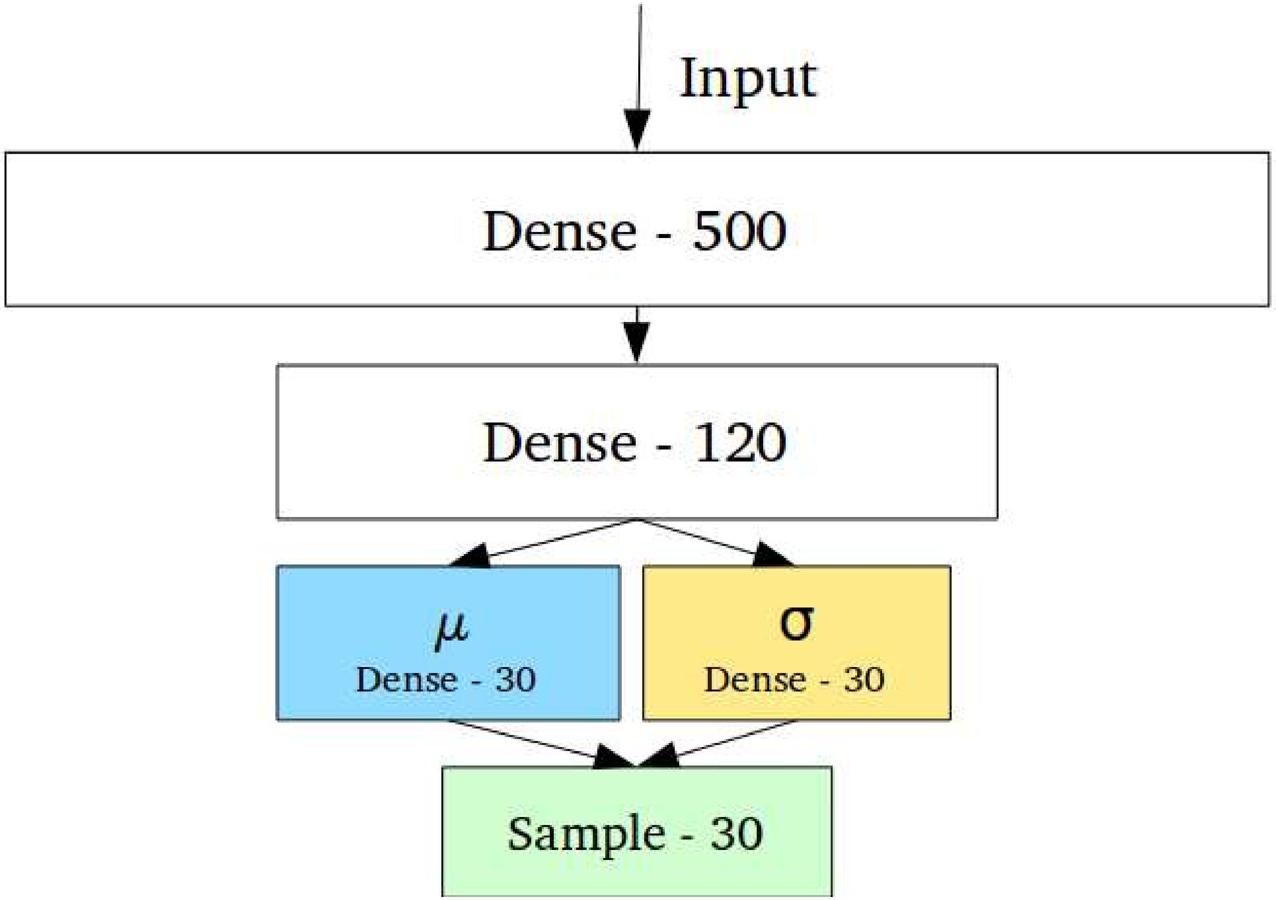
VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components



Sample **z** from $\mathbf{q_\theta(z|x)}$, where the subscript $\boldsymbol{\theta}$ means that the probability is parametrised by $\boldsymbol{\theta}$

VAEs: encoder



Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: encoder

Output μ $[0.1, 1.2, 0.2, 0.8, \ldots]$

Output $\boldsymbol{\sigma}$ $[0.2, 0.5, 0.8, 1.3, \ldots]$

Intermediate $\mathbf{X}$ $[X_{1} \sim N(0.1, 0.2^2), X_{2} \sim N(1.2, 0.5^2), X_{3} \sim N(0.2, 0.8^2), X_{4} \sim N(0.8, 1.3^2), \ldots]$

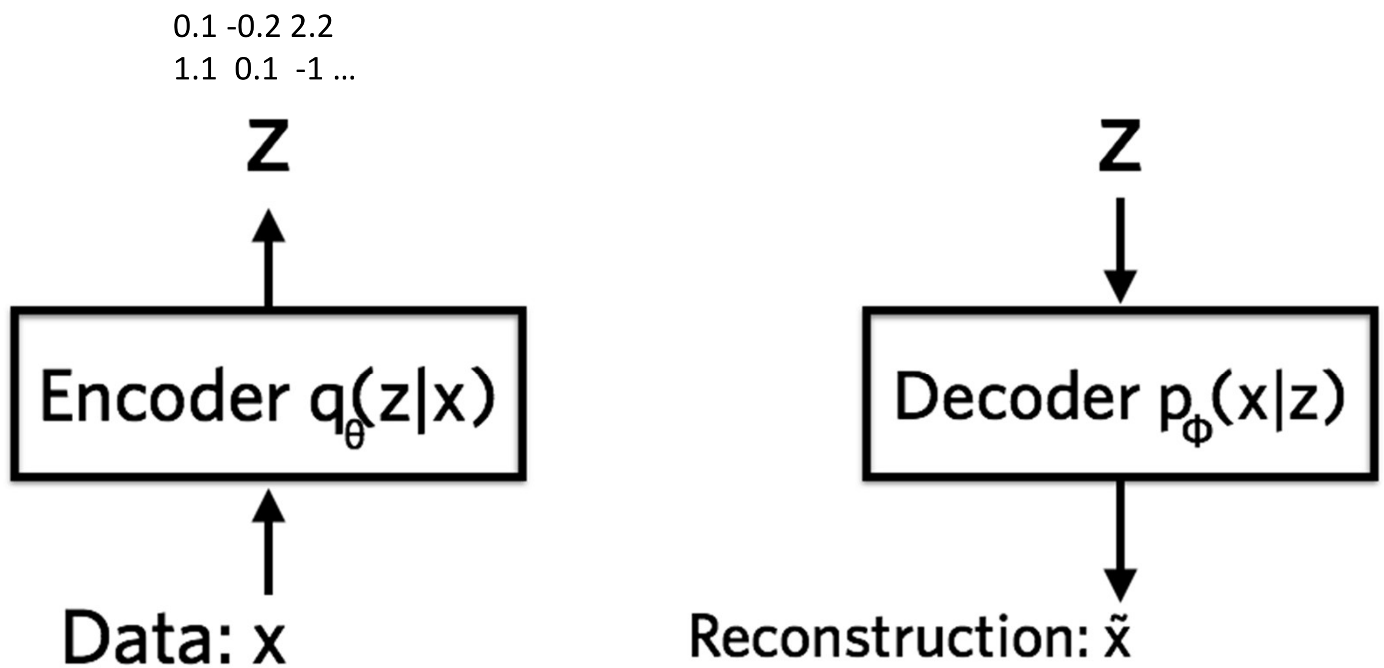
$\downarrow$ sample

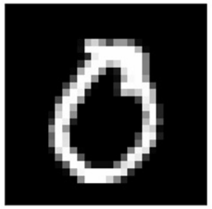
Sampled vector $[0.28, 1.65, 0.92, 1.98, \ldots]$

Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

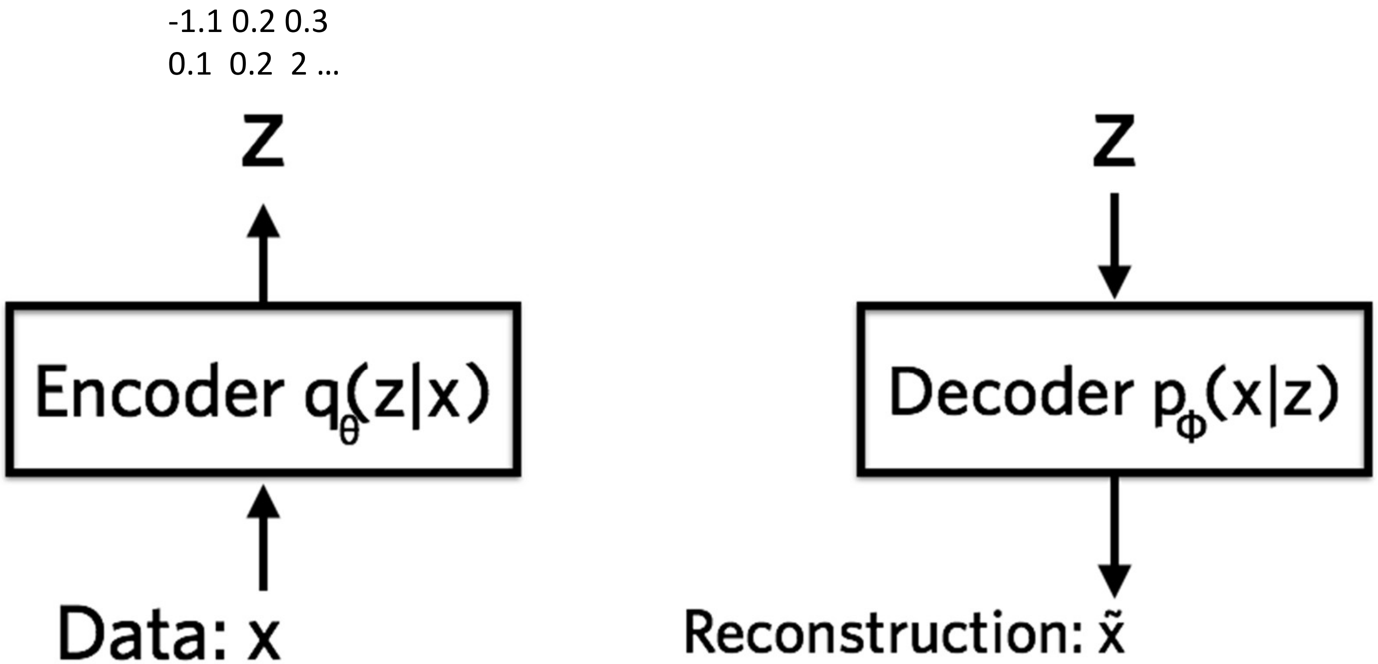


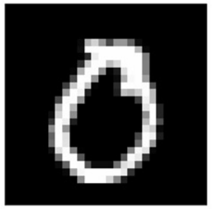


Note that different samples from **q_θ(z|x)** will in general yield different codes **z**

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

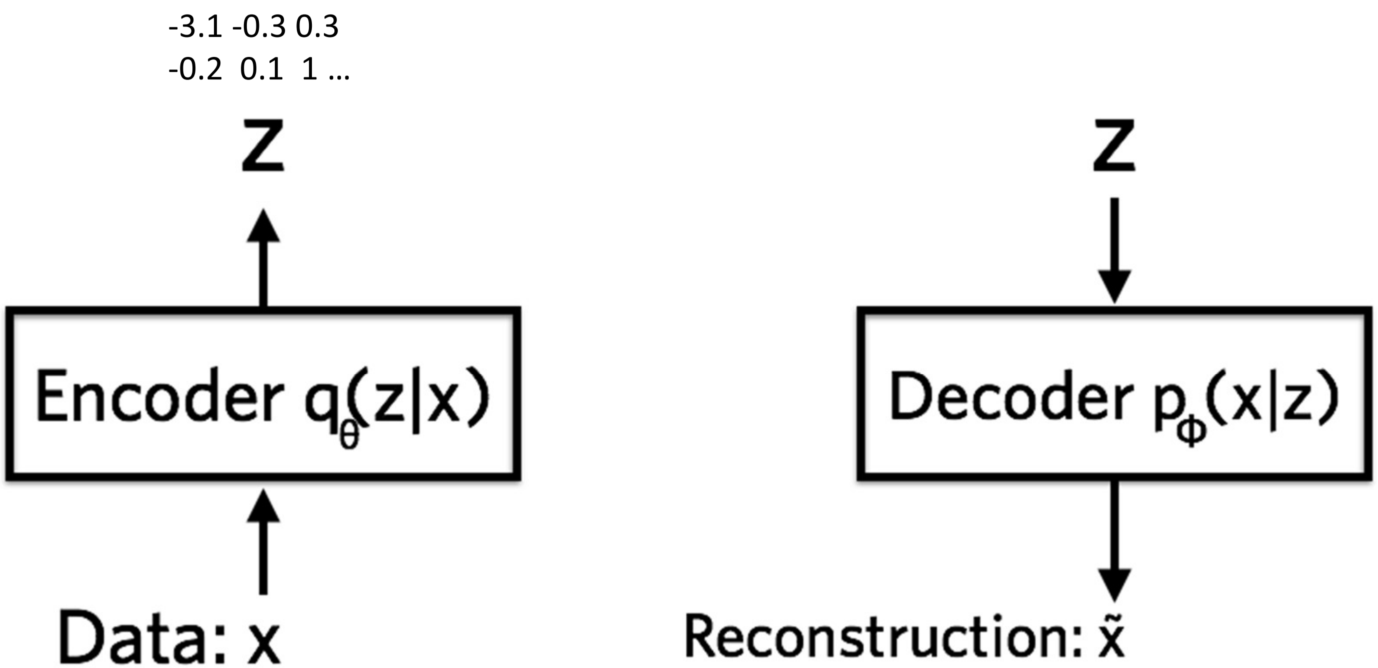


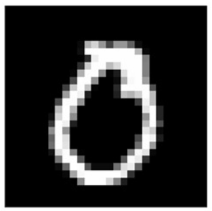


Note that different samples from **q_θ(z|x)** will in general yield different codes **z**

VAEs: neural networks perspective

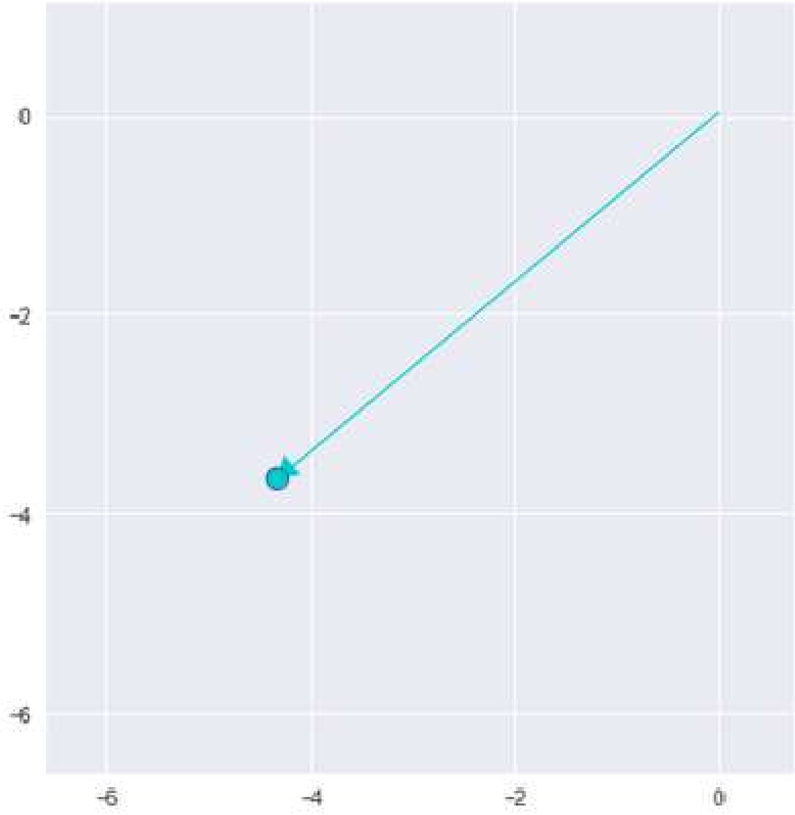
- Let’s have a closer look at our (variational) autoencoder components



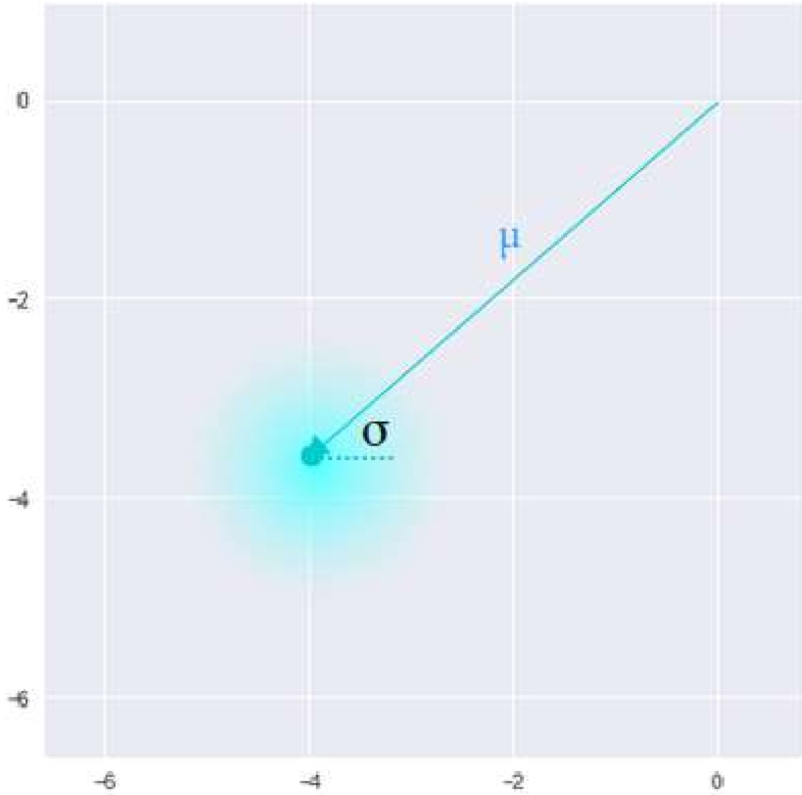


Note that different samples from **q$_\theta$(z|x)** will in general yield different codes **z**

Latent space: AE vs VAEs



Standard Autoencoder
(direct encoding coordinates)

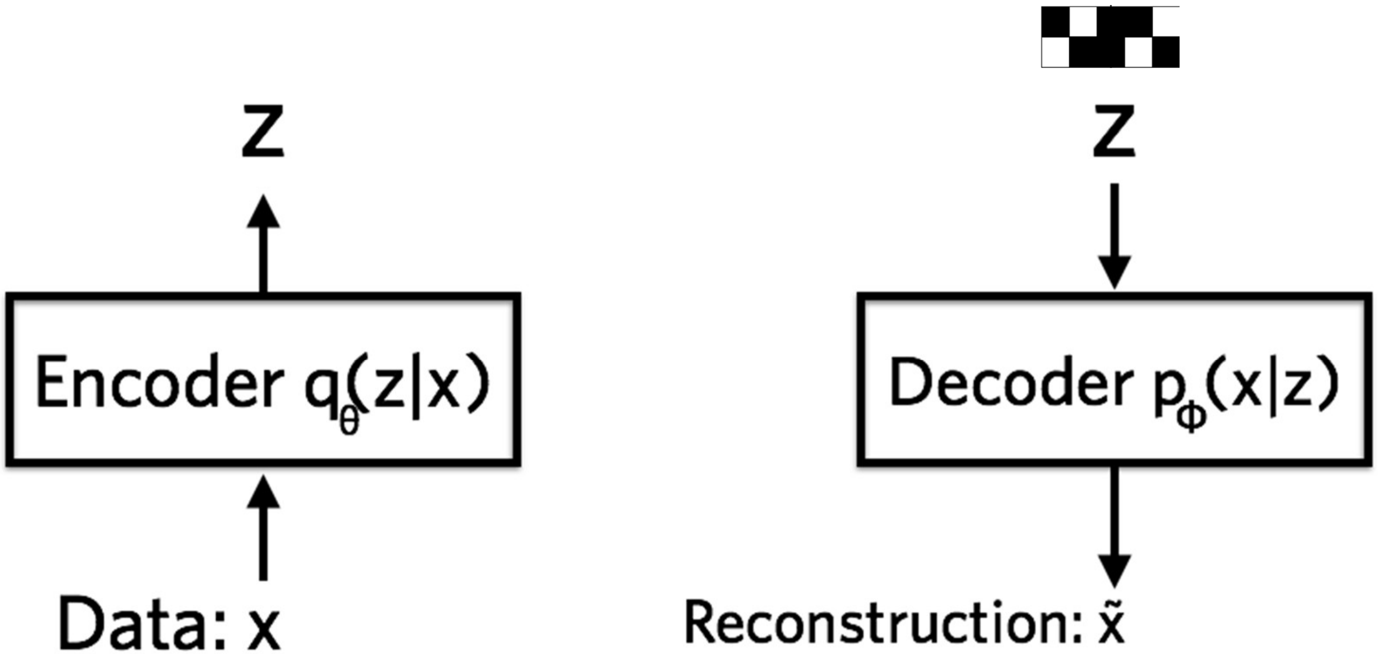


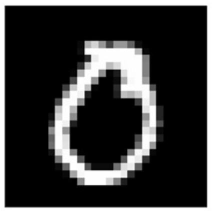
Variational Autoencoder
(μ and σ initialize a probability distribution)

Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

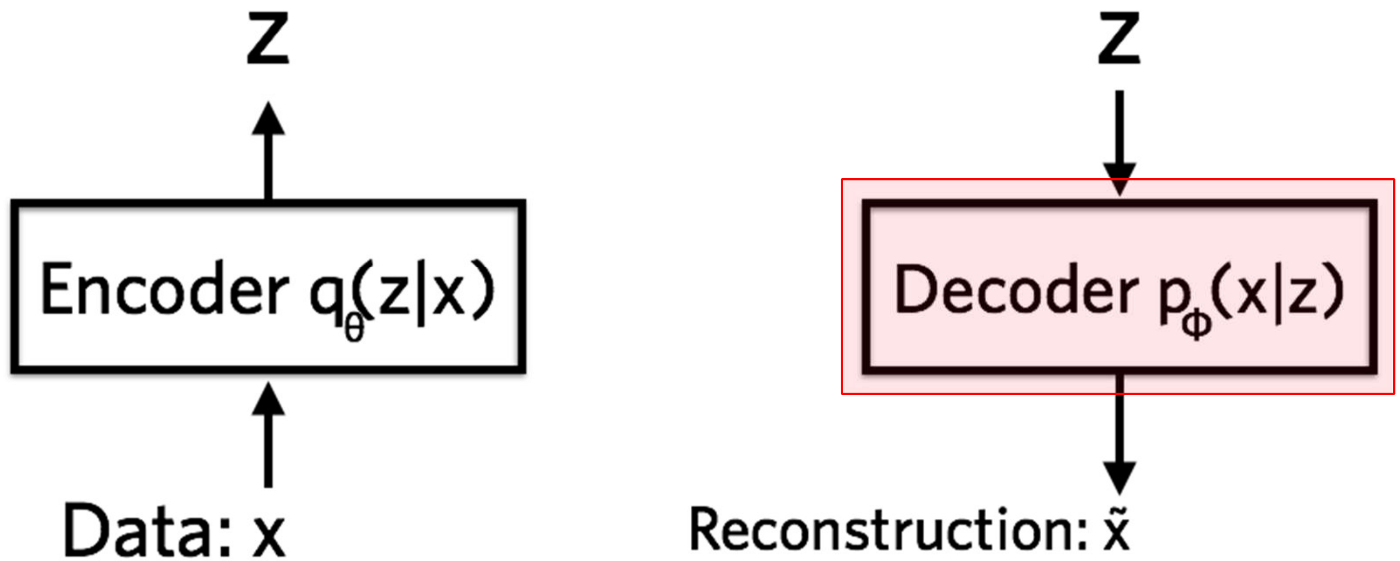


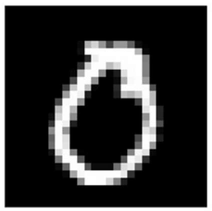


z is given in input to another neural network (the Decoder) with its weights and biases (**ϕ**)

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

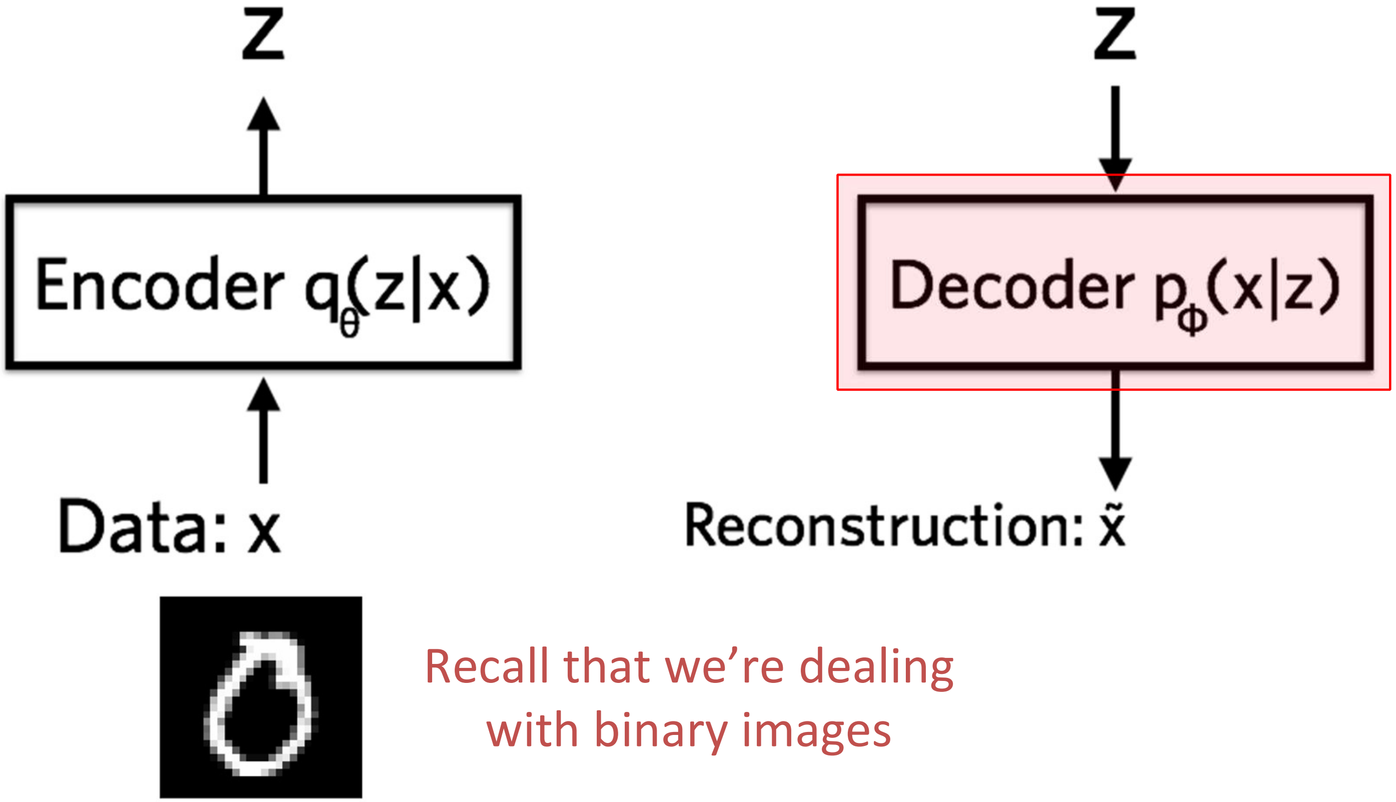




The network outputs the parameters of the distribution $\mathbf{p_\phi(x|z)}$

VAEs: neural networks perspective

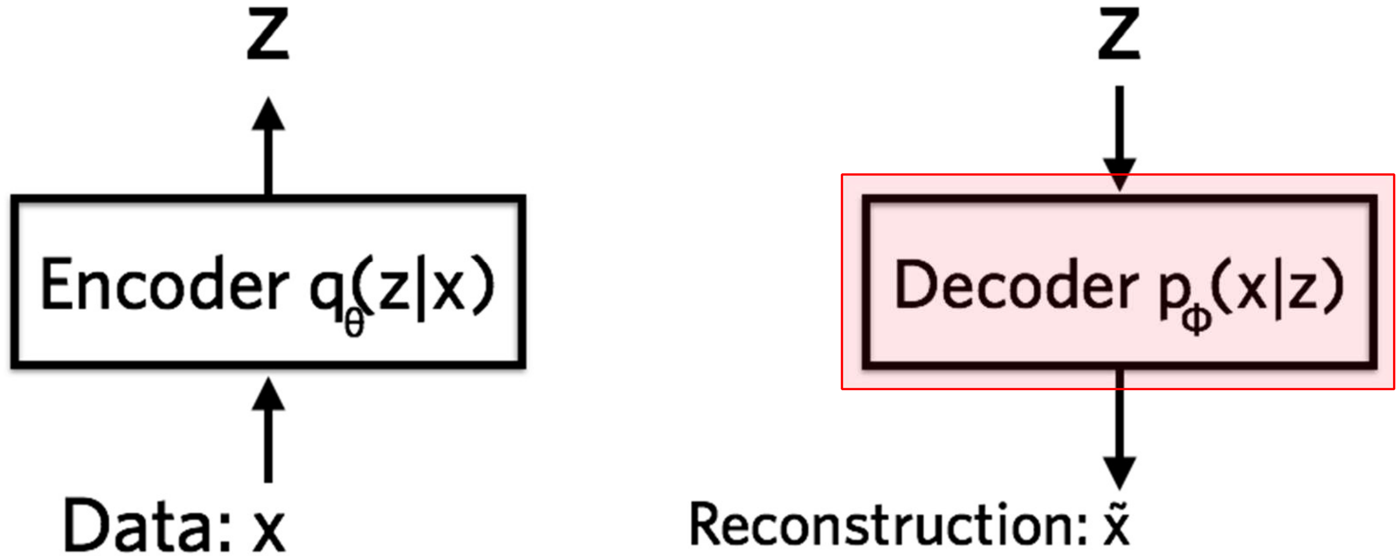
- Let's have a closer look at our (variational) autoencoder components

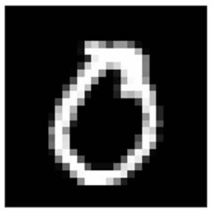


Recall that we're dealing with binary images

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

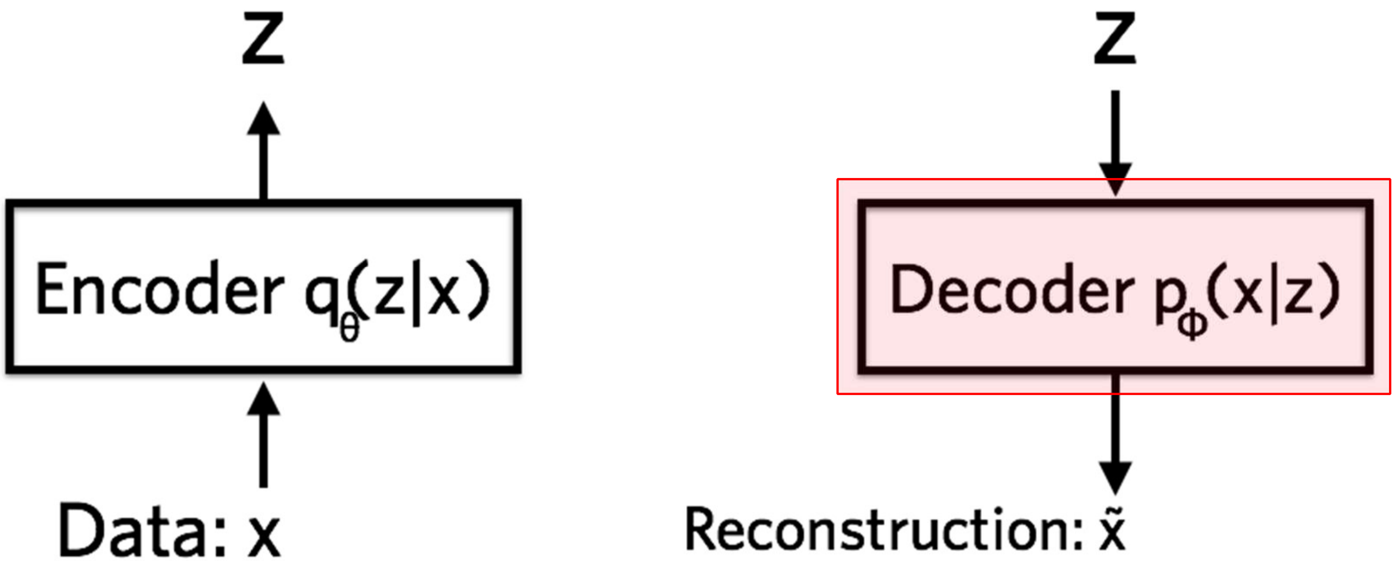




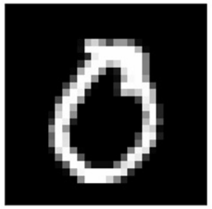
So we can teach our decoder to output 784 Bernoulli parameters (1 for each pixel)

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components



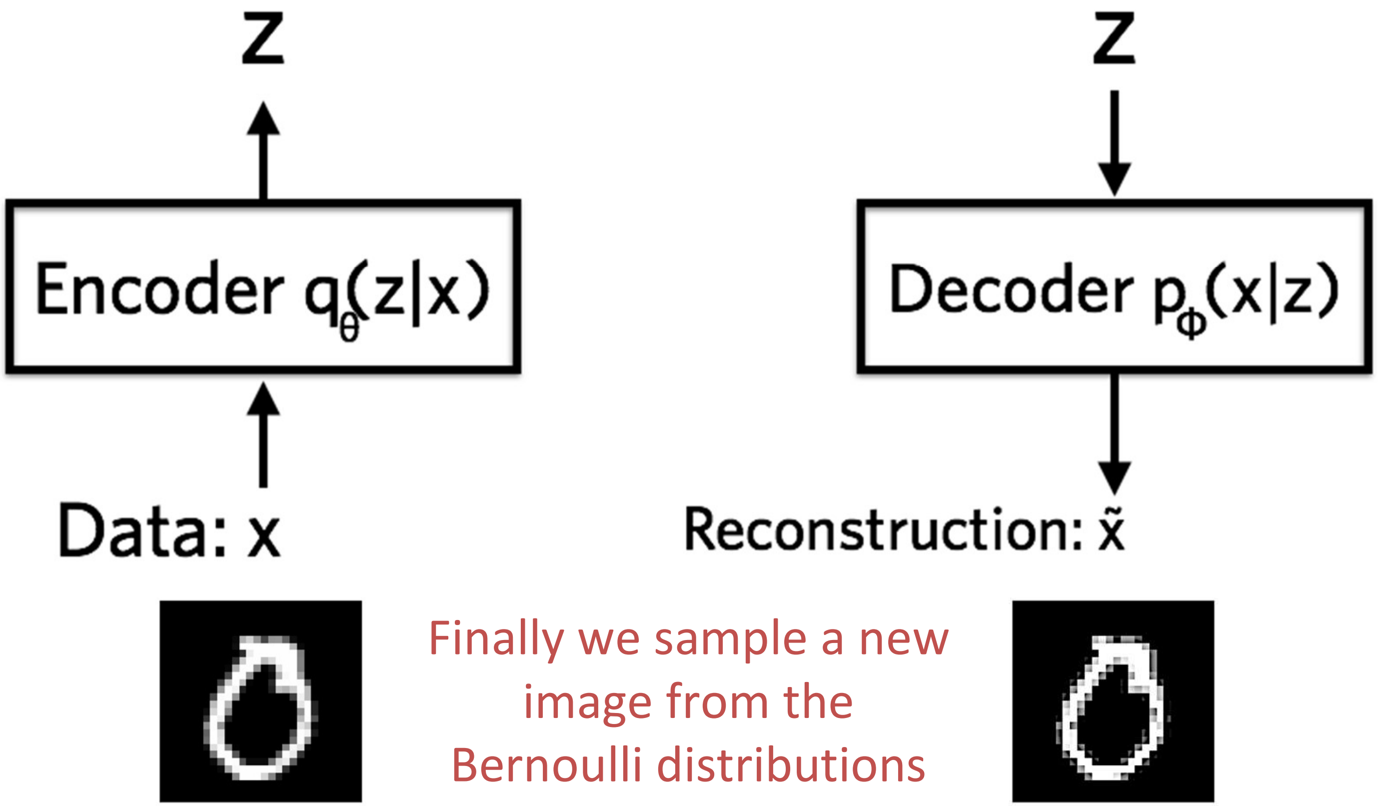
0 0.1 0.9
0 0.3 0.8
0 0.1 0.8 …



So we can teach our decoder to output 784 Bernoulli parameters (1 for each pixel)

VAEs: neural networks perspective

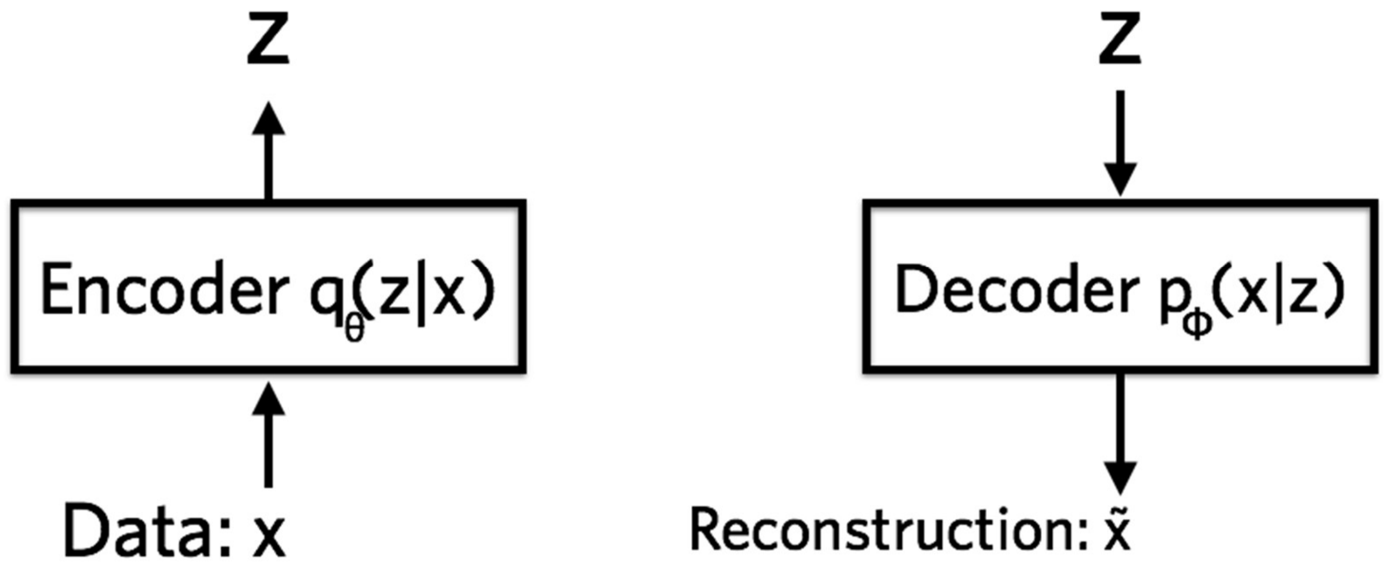
- Let’s have a closer look at our (variational) autoencoder components



Finally we sample a new image from the Bernoulli distributions

VAEs: neural networks perspective

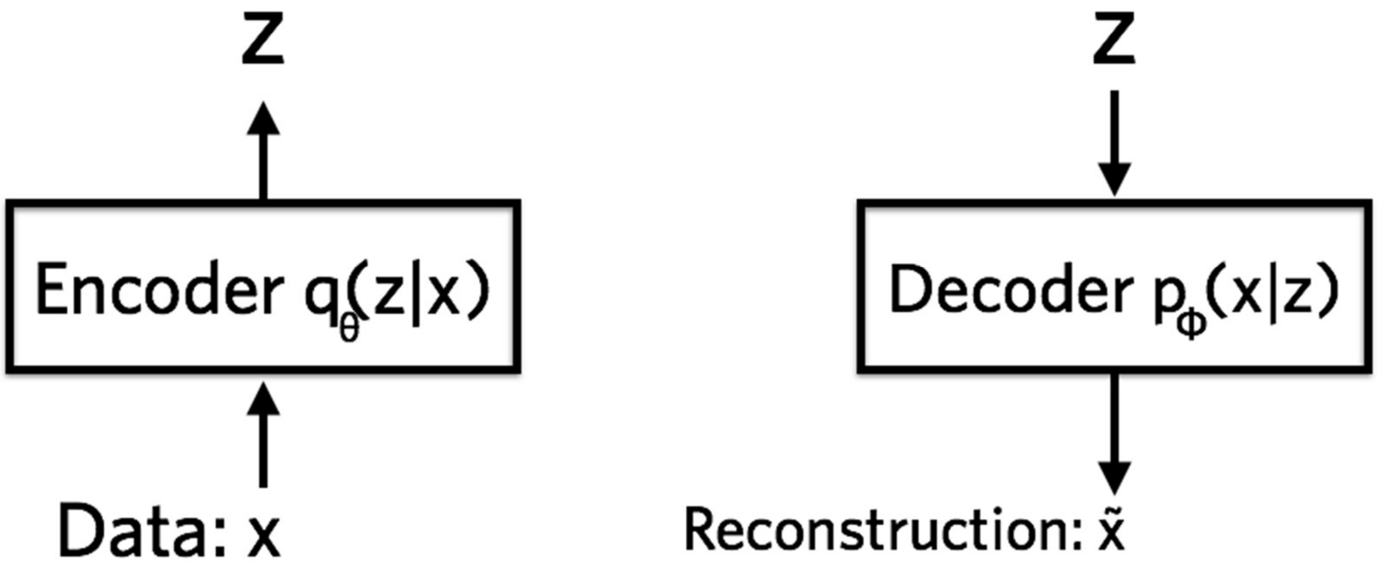
- Let’s have a closer look at our (variational) autoencoder components



How much information is lost when going from the low-dimensional representation **z** to the higher dimensional reconstructed **x**?

VAEs: neural networks perspective

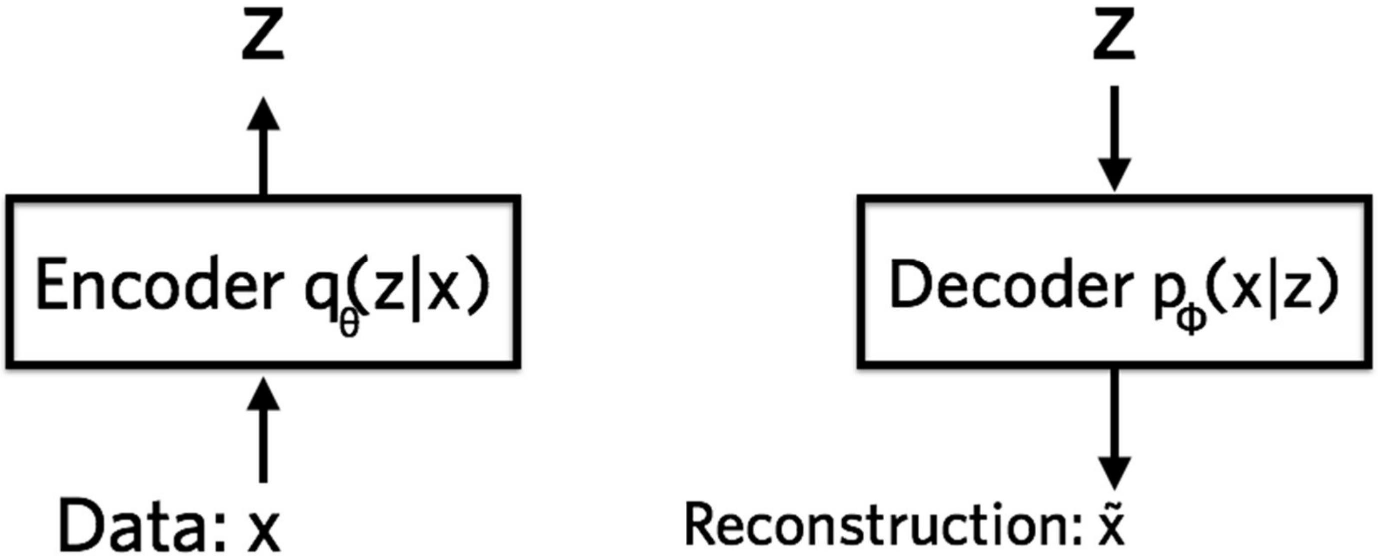
- Let’s have a closer look at our (variational) autoencoder components



This can be measured using the reconstruction log-likelihood **log p_ϕ(x|z)**
It tells us how effectively the decoder has learned to reconstruct **x** given **z**

VAEs: neural networks perspective

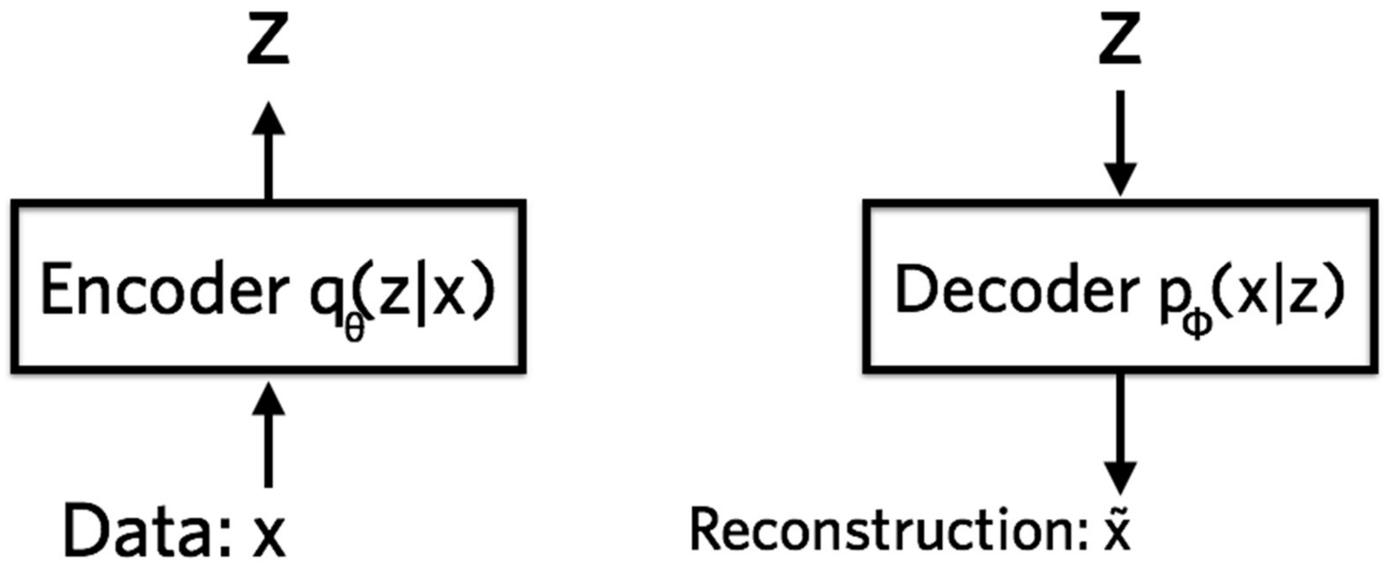
- Let's have a closer look at our (variational) autoencoder components



Sounds nice, but we need a loss if we want to do back-propagation
and learn the optimal parameters of the two networks (encoder & decoder)

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

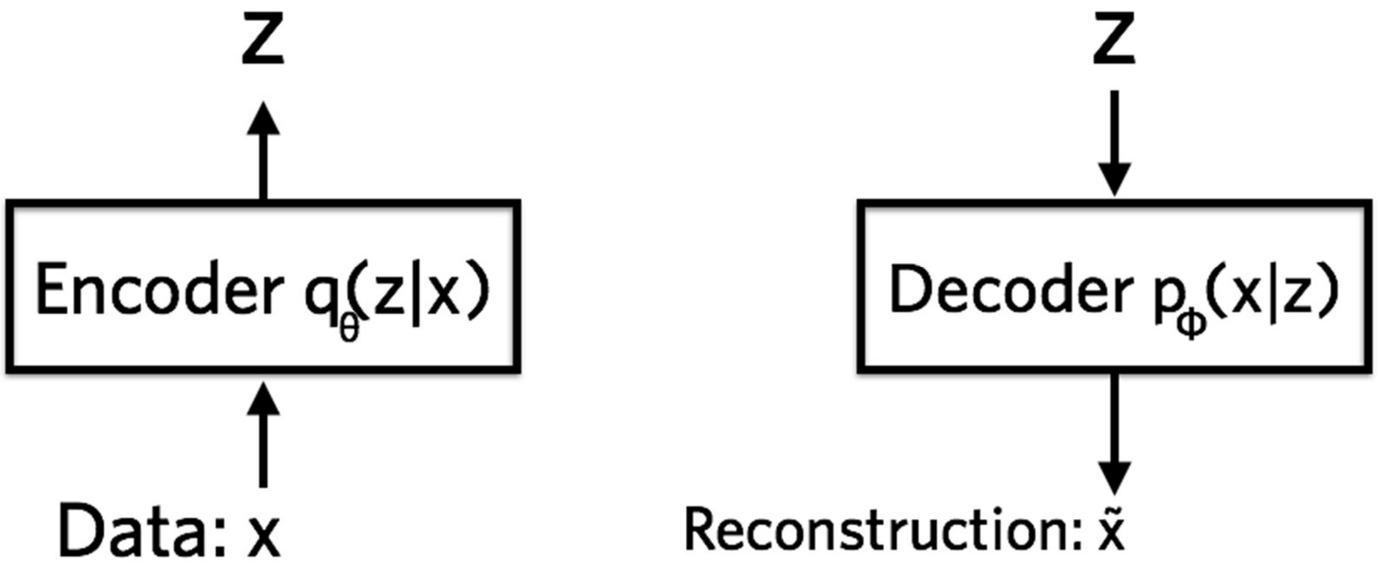


$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

loss function for i-th datapoint - the total loss is the sum of all the l_i

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

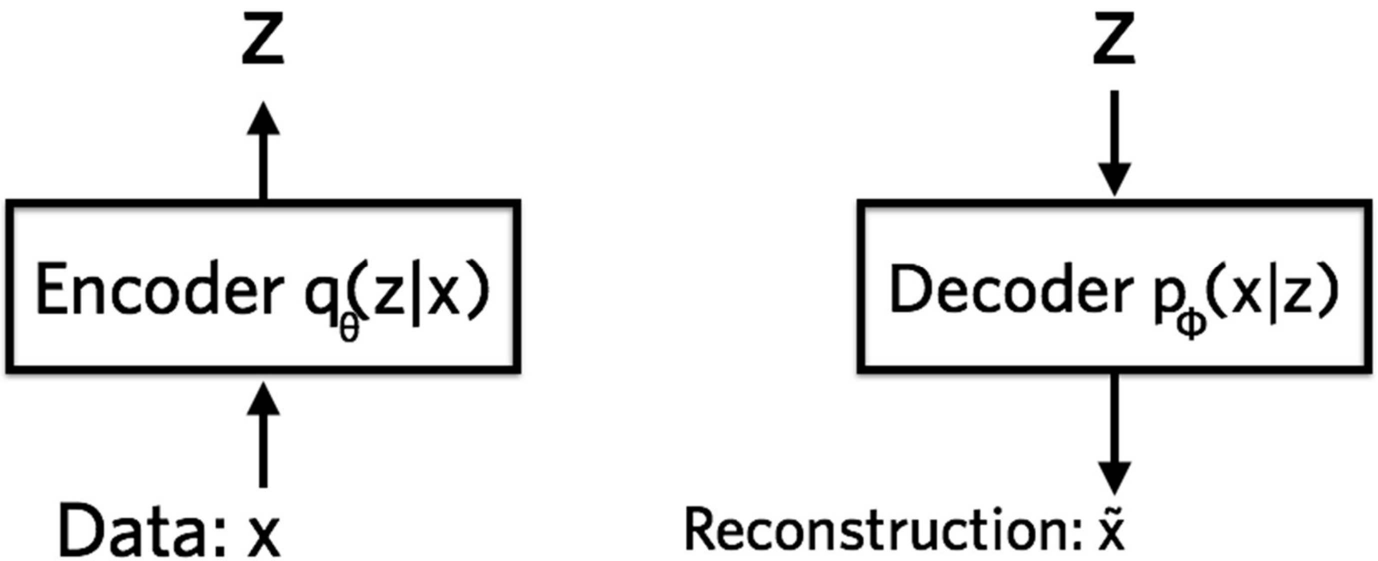


$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

Remember, for one data point we can sample many codes **z**

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

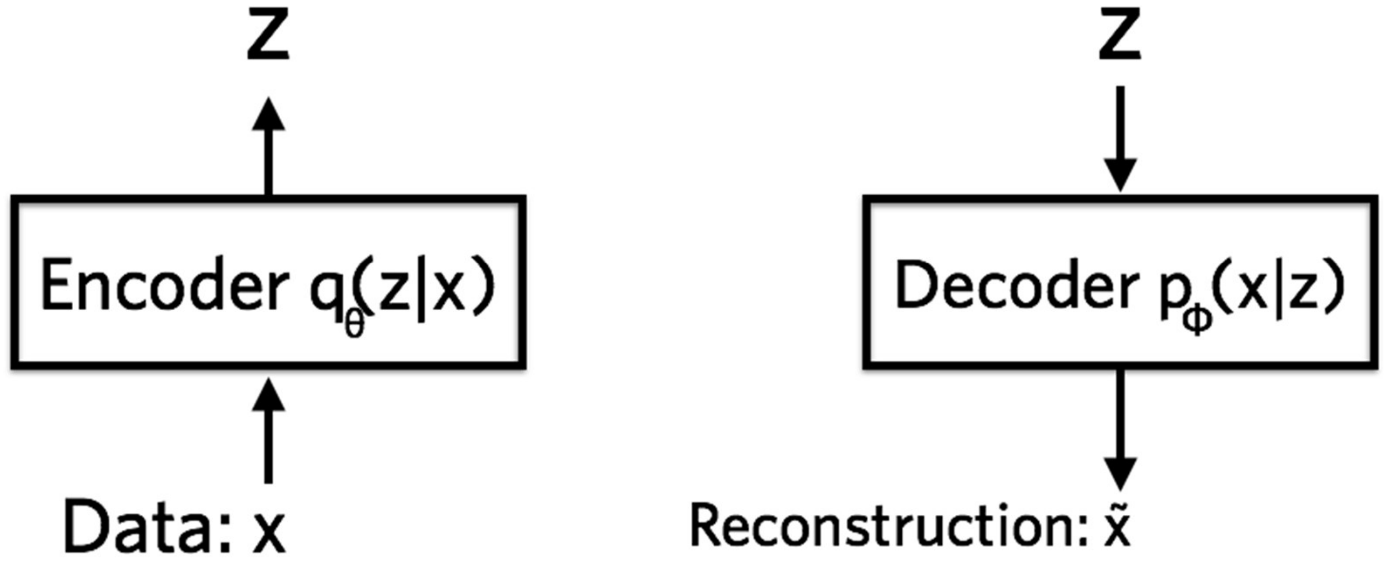


$$l_i(\theta, \phi) = \boxed{-\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]} + \mathbb{KL}(q_\theta(z \mid x_i) \,||\, p(z))$$

This is why here we have the expected log-likelihood (negative, we're minimising)

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

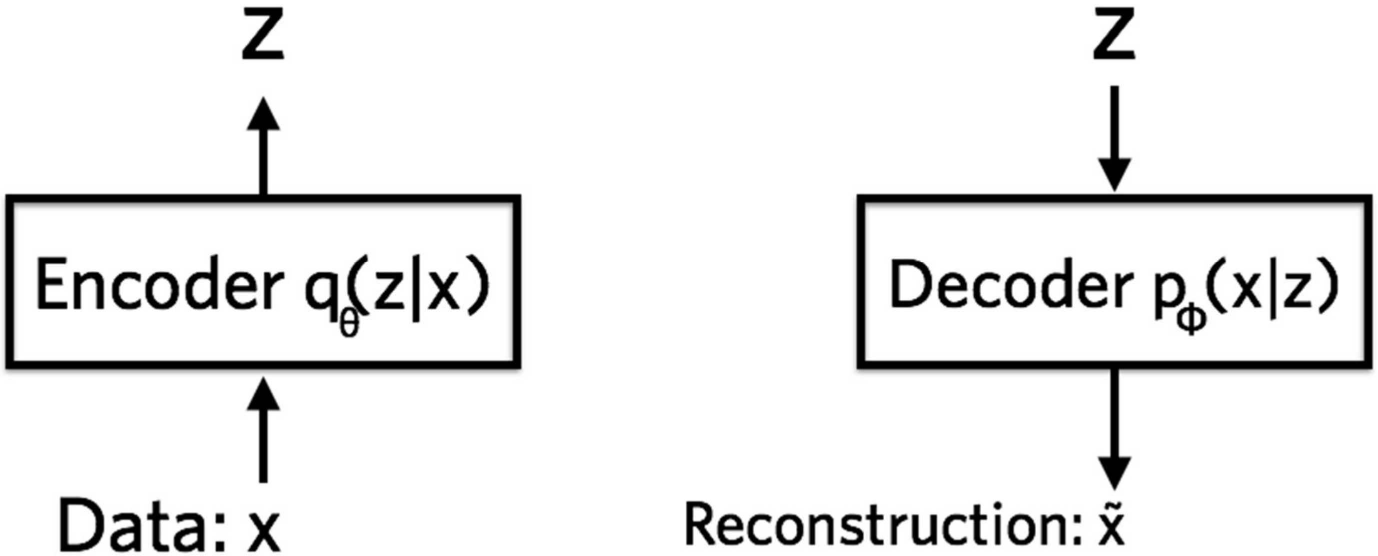


$$l_i(\theta, \phi) = \boxed{-\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]} + \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

This term encourages decoder to learn to reconstruct the data

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

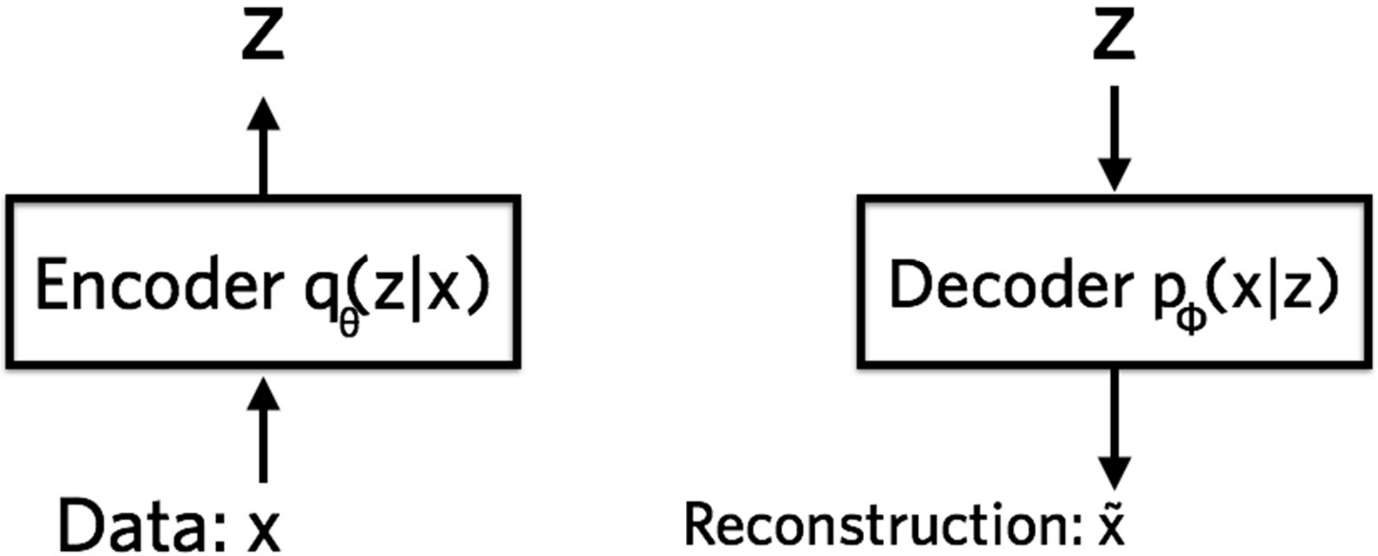


$$l_i(\theta,\phi) = \boxed{-\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)]} + \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

Bad encoder (e.g., high prob of black pixel when it should be white) = large cost

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

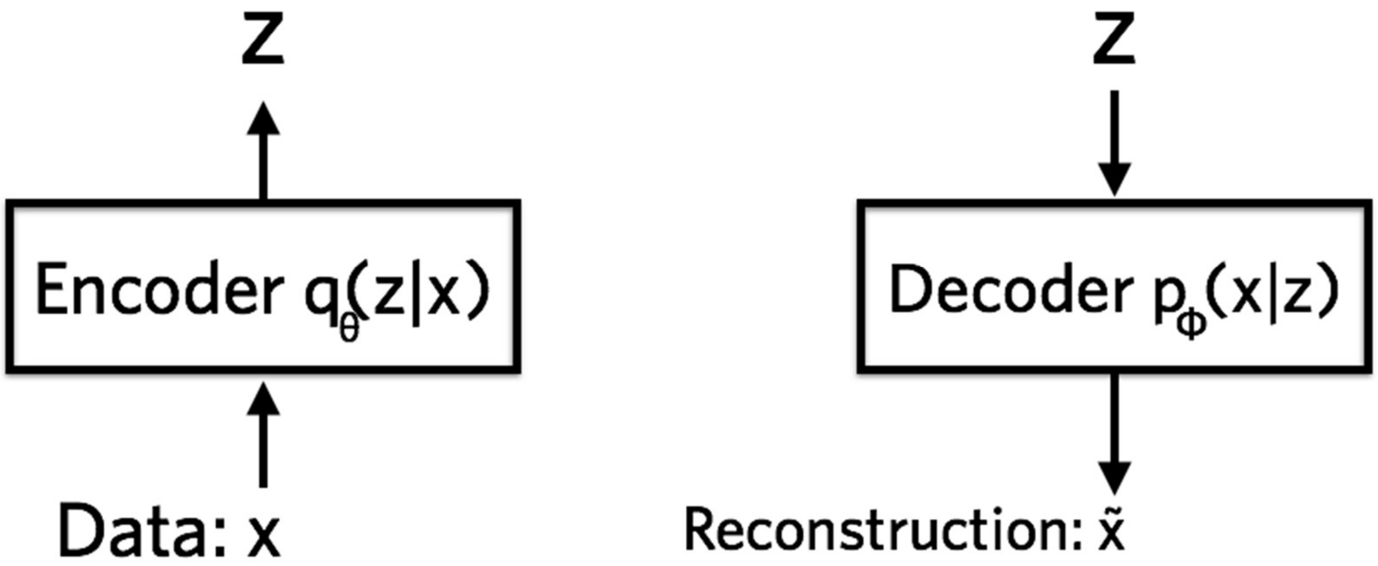


$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z \mid x_i)}[\log p_\phi(x_i \mid z)] \boxed{+ \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))}$$

Regularisation term - measures how close **q** is to **p**

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

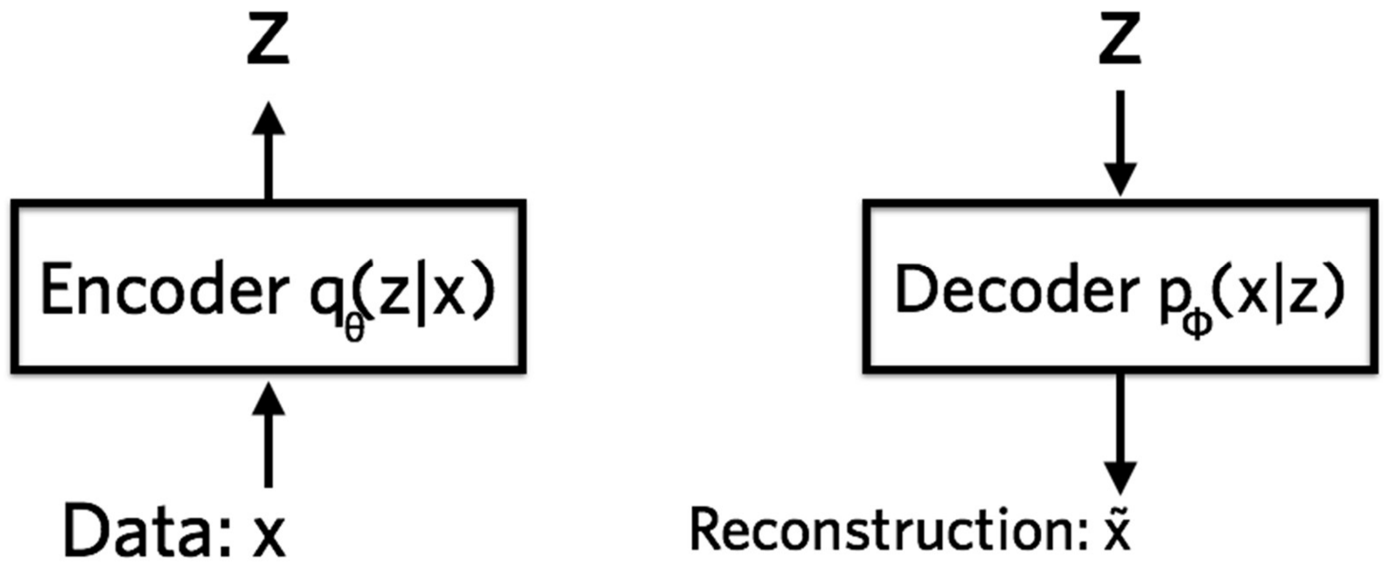


$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] \boxed{+ \mathbb{KL}(q_\theta(z \mid x_i) \,||\, p(z))}$$

In VAs we let **p(z) = Gaussian(0,1)**

VAEs: neural networks perspective

- Let's have a closer look at our (variational) autoencoder components

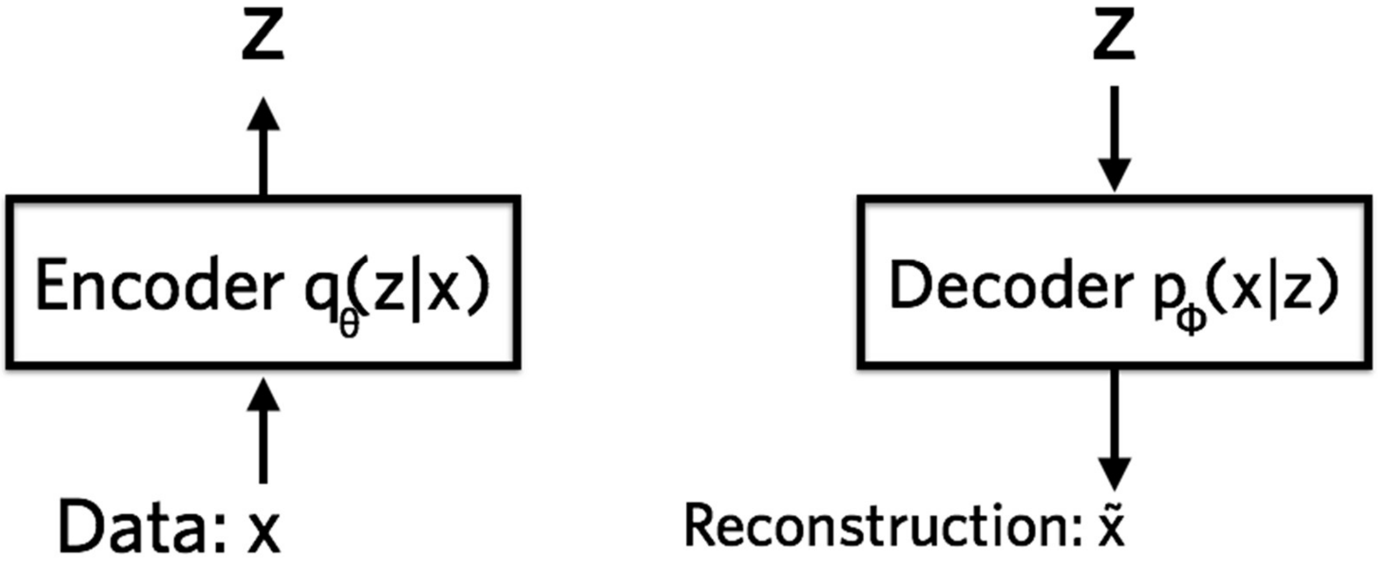


$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z \mid x_i)}[\log p_\phi(x_i \mid z)] \boxed{+ \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))}$$

This is added to make sure the encoder doesn't cheat and map each datapoint in different regions of the space

VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components

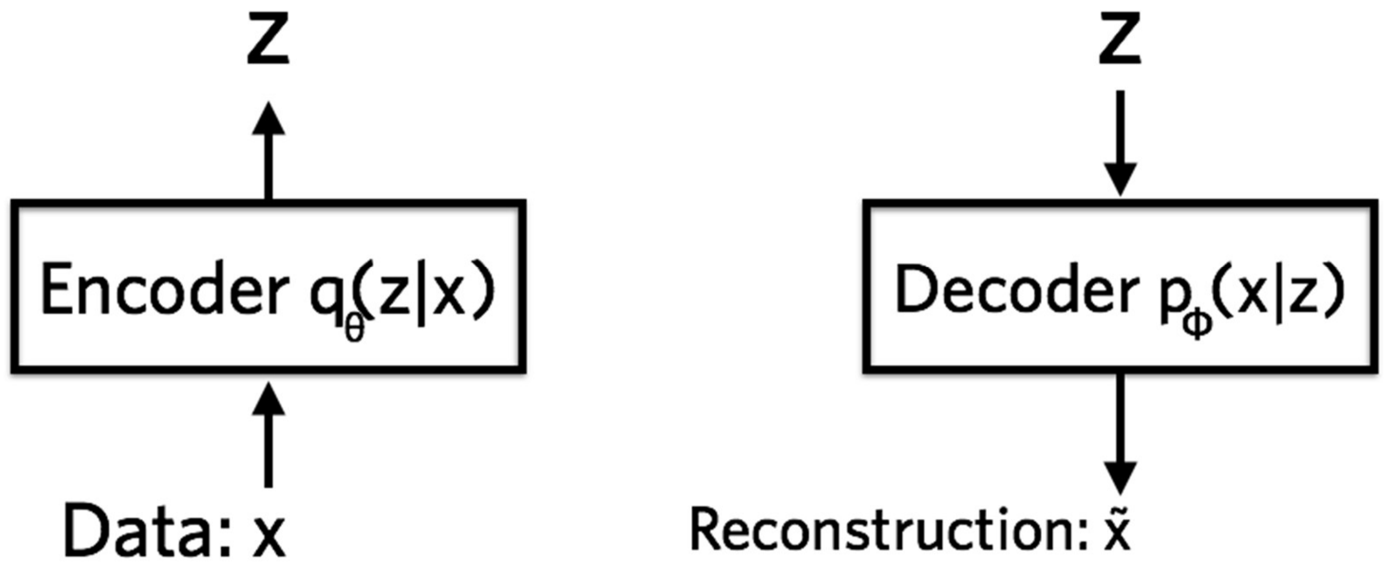


$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z \mid x_i)}[\log p_\phi(x_i \mid z)] \boxed{+ \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))}$$

This is bad because we want e.g. different images of same number to lie close in the embedding space

VAEs: neural networks perspective

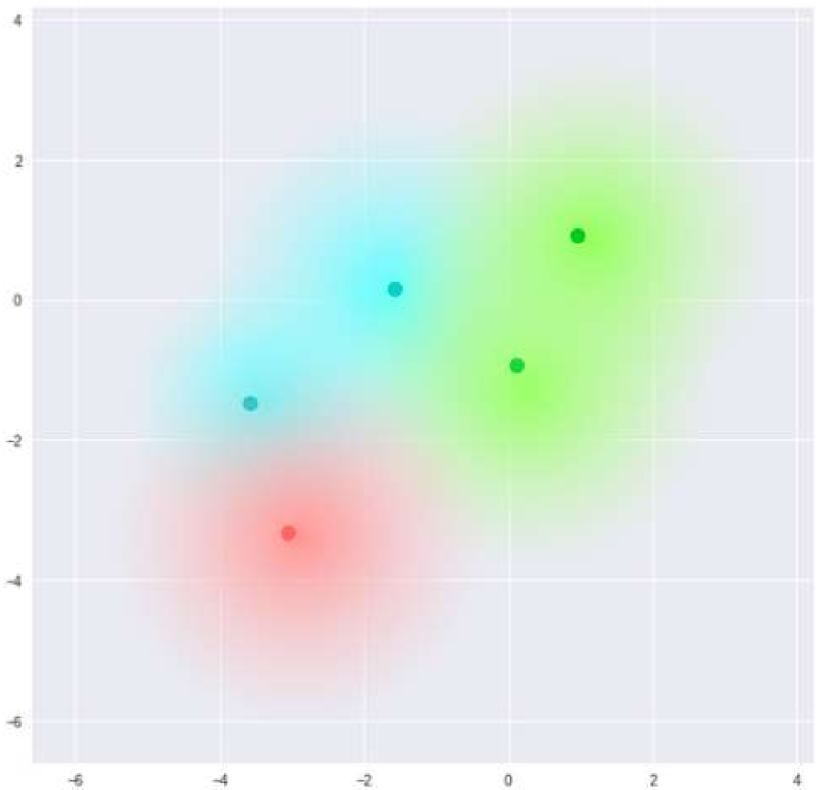
- Let’s have a closer look at our (variational) autoencoder components



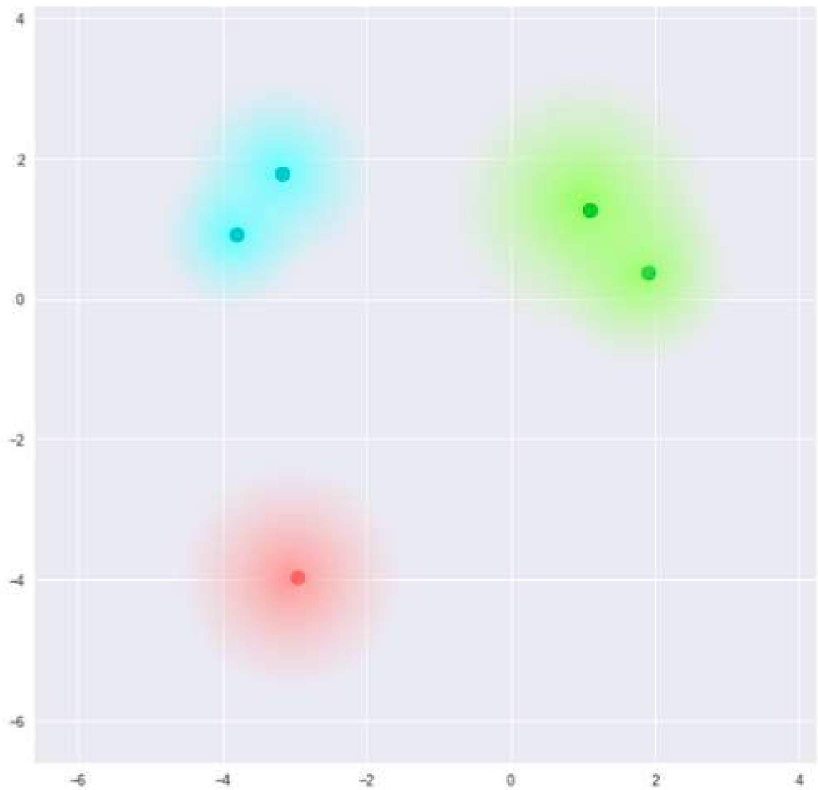
$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] \boxed{+ \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))}$$

The space has to be “meaningful”, so we penalise this behaviour

Meaning of the regularisation



What we require

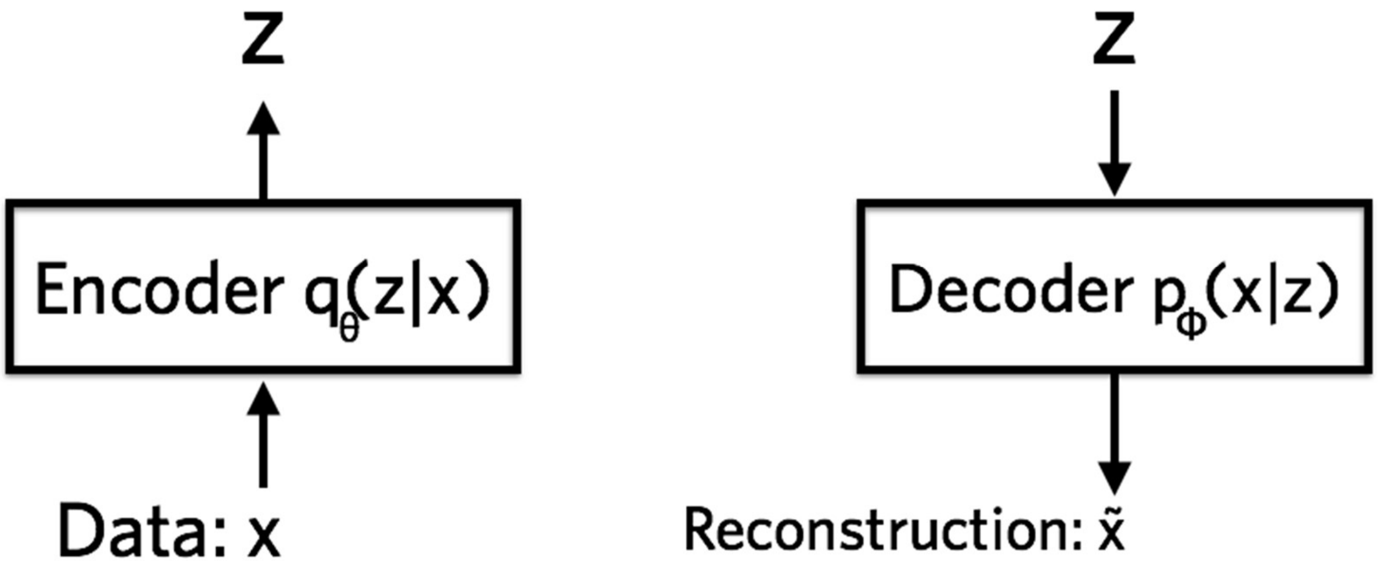


What we may inadvertently end up with

Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

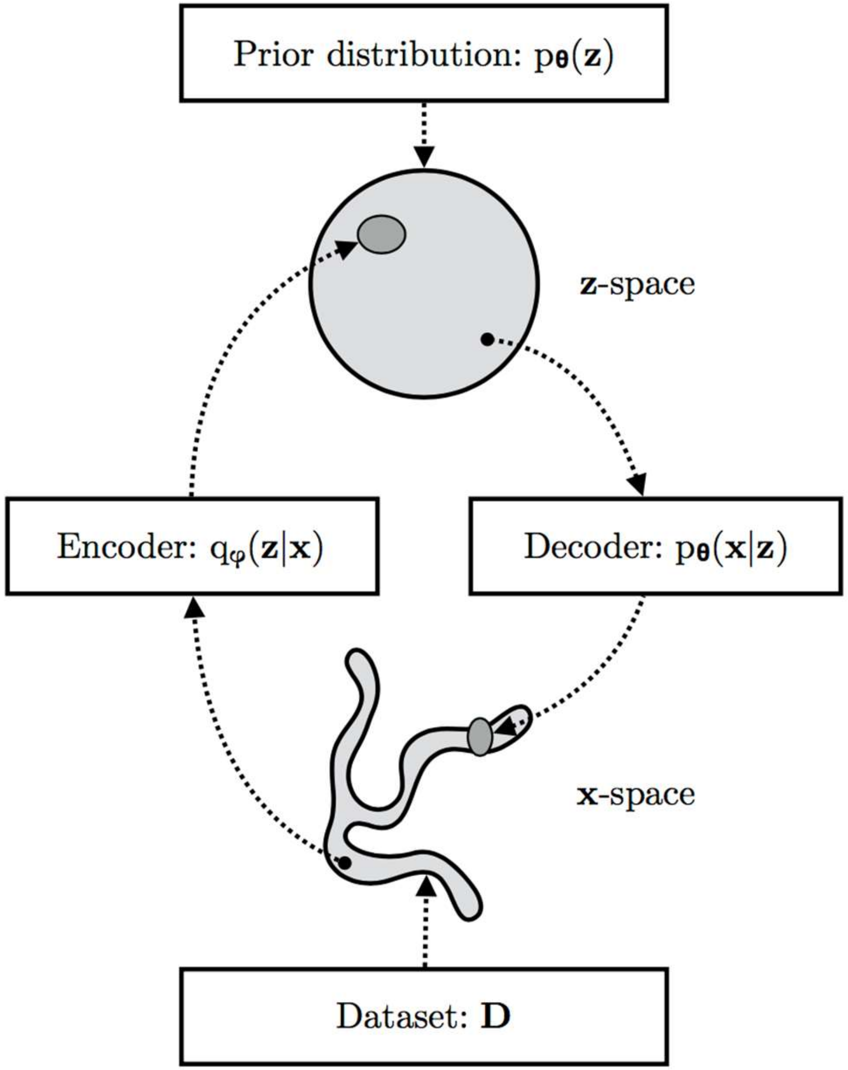
VAEs: neural networks perspective

- Let’s have a closer look at our (variational) autoencoder components



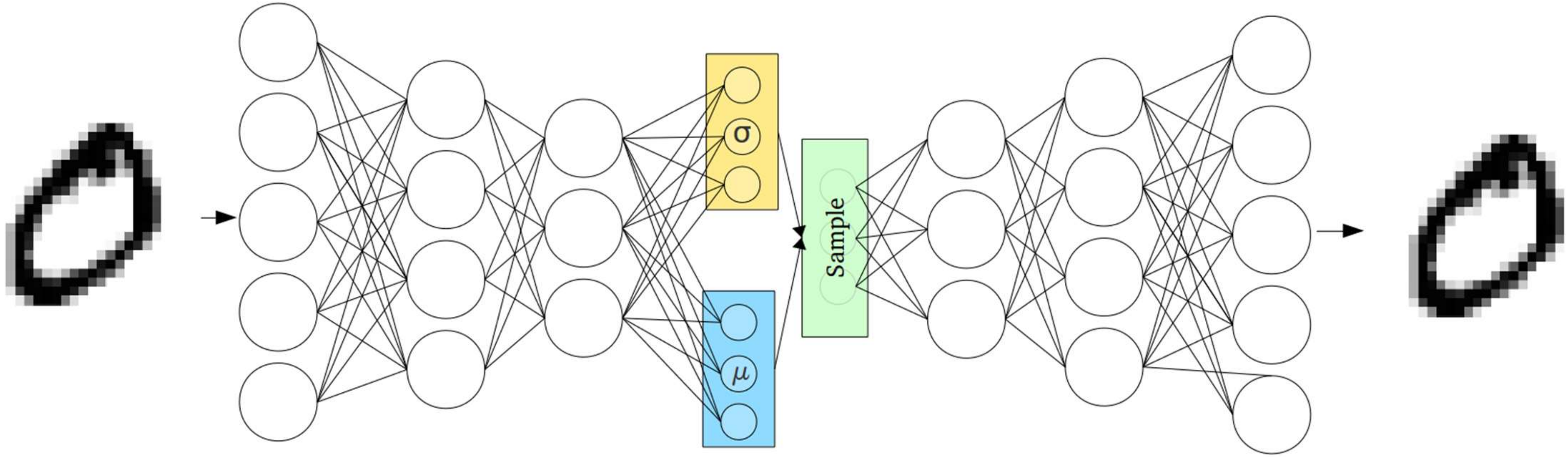
Given this architecture, we can use back-propagation to compute the gradients of the loss wrt the networks parameters and then optimise using any variant of gradient descent

VAEs architecture



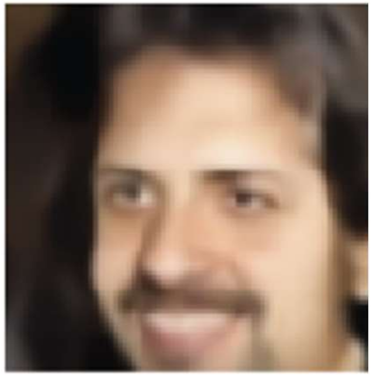
Source: D.P. Kingma Variational Inference and Deep Learning: A New Synthesis

VAEs architecture



Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: latent space exploration

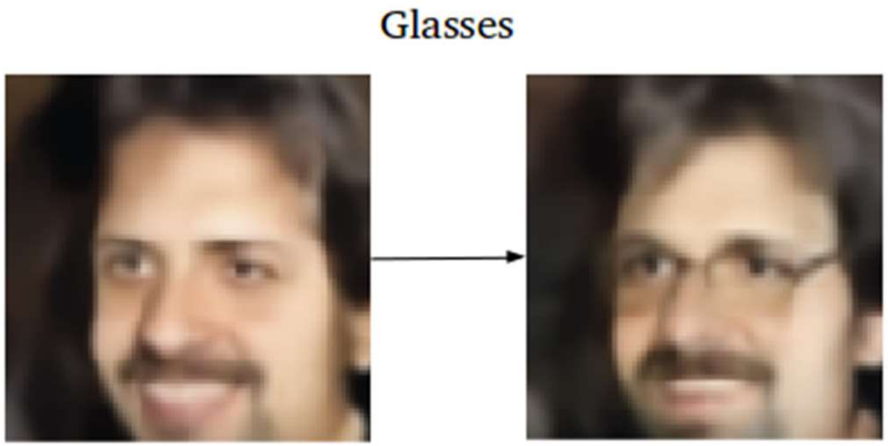


Want: find images between “face with glasses” to “face without glasses”

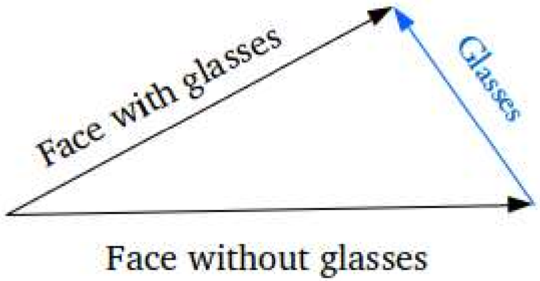
First: Find representation of faces in latent space (by giving it as input of encoder)

Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: latent space exploration

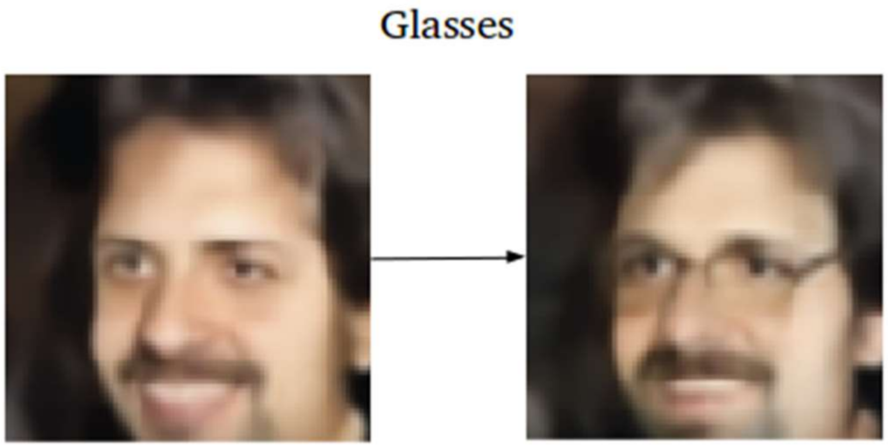


What we do: In the latent space, you computed the translation vector from “face without glasses” to “face with glasses” from the embeddings of two faces, without and with glasses

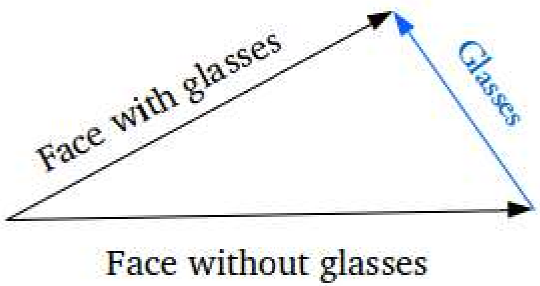


Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: latent space exploration

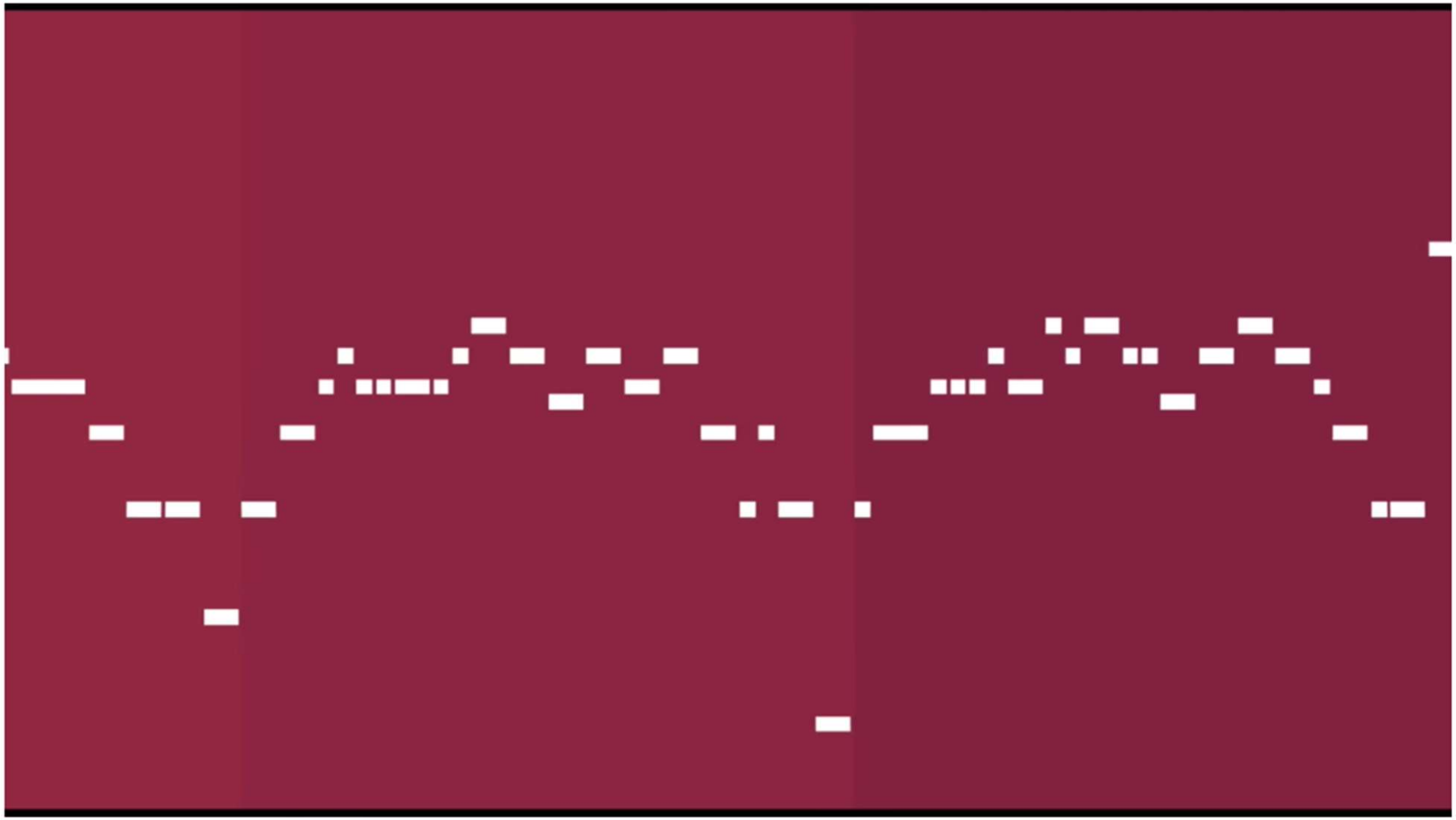


Add the translation vector to the latent representation then decode this representation to map it back to the image space



Source: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAEs: latent space exploration



Latent space interpolation

Source: https://magenta.tensorflow.org/music-vae

VAEs: neural networks perspective

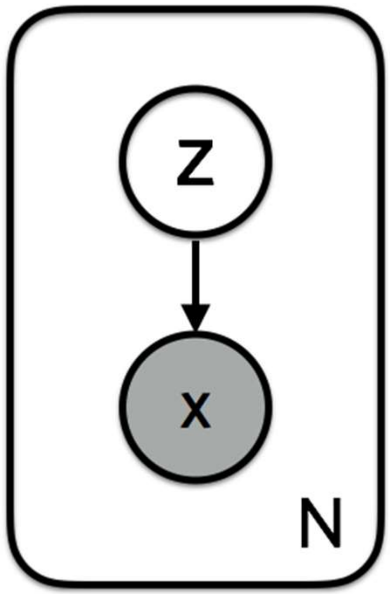
- …but why exactly is this called a “variational” autoencoder?

VAEs: probabilistic perspective

- To understand why, we need to look at variational autoencoders from a different perspective, i.e., that of a probability model

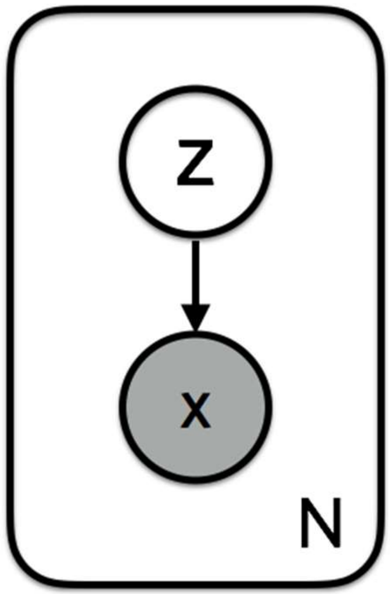
VAEs: probabilistic perspective

- To understand why, we need to look at variational autoencoders from a different perspective, i.e., that of a probability model
- Let there be a generative model of the data **x** and the latent variables **z** with joint probability **p(x,z) = p(x|z)p(z)**



VAEs: probabilistic perspective

- To understand why, we need to look at variational autoencoders from a different perspective, i.e., that of a probability model
- Let there be a generative model of the data **x** and the latent variables **z** with joint probability **p(x,z) = p(x|z)p(z)**
- First we sample **z** from prior **p(z)**
- Then we sample **x** from the likelihood **p(x|z)**



VAEs: probabilistic perspective

- In this context, learning is called *inference*
- We want to infer the optimal parameters of $\mathbf{p(x)}$
- In other words, we want to maximise $\mathbf{p(x)}$
- $\mathbf{\log p(x) = \log p(x,z)/p(z|x)}$, to be precise

VAEs: probabilistic perspective

- In this context, learning is called *inference*
- We want to infer the optimal parameters of **$p(x)$**
- In other words, we want to maximise **$p(x)$**
- **$\log p(x) = \log p(x,z)/p(z|x)$**, to be precise
- But **$p(z|x) = p(x,z)/p(x)$**, and computing **$p(x)$** takes an exponential time since **$p(x) = \int p(x|z)p(z)dz$**
- **$p(z|x)$** is called the posterior and its often intractable in this type of problems…

VAEs: probabilistic perspective

KL

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **$p(z|x)$** with a family of distributions **$q_\lambda(z|x)$** λ

Read more about it at: https://en.wikipedia.org/wiki/Variational_Bayesian_methods

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

$$\mathbb{KL}(q_\lambda(z \mid x) \;||\; p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

$$KL(q_\lambda(z|x) \,||\, p(z|x)) = E_q\left(\log \frac{q_\lambda(z|x)}{p(z|x)}\right)$$

$$= E_q(\log q_\lambda(z|x) - \log p(z|x))$$

$$= E_q(\log q_\lambda(z|x)) - E_q\left(\log \frac{p(z,x)}{p(x)}\right)$$

Constant

$$= E_q(\log q_\lambda(z|x)) - E_q(\log p(x,z)) + E_q(\log p(x))$$

$$= E_q(\log q_\lambda(z|x)) - E_q(\log p(x,z)) + \log p(x)$$

$$KL \geq 0 \Rightarrow \log p(x) \geq \underbrace{E_q(\log p(x,z)) - E_q(\log q_\lambda(z|x))}_{\text{ELBO (Evidence lower bound)}}$$

if want to max $\log p(x)$, then max ELBO.

(by Jiangao Zhang)

$$\text{ELBO} = E_q\left(\log p(x,z)\right) - E_q\left(\log q_\lambda(z|x)\right)$$

$$= E_q\left(\log p(x|z)\cdot p(z)\right) - E_q\left(\log q_\lambda(z|x)\right)$$

$$= E_q\left(\log p(x|z)\right) + E_q\left(\log p(z)\right) - E_q\left(\log q_\lambda(z|x)\right)$$

$$= E_q\left(\log p(x|z)\right) - E_q\left(\log \frac{q_\lambda(z|x)}{p(z)}\right)$$

$$= E_q\left(\log p(x|z)\right) - KL\left(q_\lambda(z|x) \,||\, p(z)\right)$$

---

* objective function of VAE.

* $p(z)$ is the prior, normally the Gaussian Distribution

* The first term is the maximum likelihood of the Decoder.

(by Jianyu Zhang)

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

We want this to be small

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions $\mathbf{q_\lambda(z|x)}$
- Of course we want our approximation to be good, i.e., close to the true posterior

$$q^*_\lambda(z \mid x) = \arg\min_\lambda \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

This is what we're looking for

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$q_\lambda^*(z \mid x) = \arg\min_\lambda \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

But this isn't good...

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

Let us introduce this function

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

Then we can write this

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **$p(z|x)$** with a family of distributions **$q_\lambda(z|x)$**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

KL is always >= 0, so to minimise KL we can maximise ELBO!

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))$$

Maximising ELBO means 1) **q** close to **p** and 2) higher **p** (better generator)

$$ELBO_i(\theta, \phi) = \mathbb{E}q_\theta(z \mid x_i)[\log p_\phi(x_i \mid z)] - \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

VAEs: probabilistic perspective

- Which is where the **variational** elements comes in!
- Variational inference approximates the posterior **p(z|x)** with a family of distributions **q_λ(z|x)**
- Of course we want our approximation to be good, i.e., close to the true posterior

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$

$$ELBO_i(\theta, \phi) = \mathbb{E}q_\theta(z \mid x_i)[\log p_\phi(x_i \mid z)] - \mathbb{KL}(q_\theta(z \mid x_i) \mid\mid p(z))$$

We link the network and probabilistic perspective by explicating the parameters of q and p and noting that the above is the (negative of the) loss function of a variational autoencoder

VAEs on the Web

- Various visualisations: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
- Interactive VAEs: https://www.siarez.com/projects/variational-autoencoder
- Morphing faces: http://vdumoulin.github.io/morphing_faces/online_demo.html
- MNIST demo: http://dpkingma.com/sgvb_mnist_demo/demo.html

VAEs on the Web

- Various visualisations: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
- Interactive VAEs: https://www.siarez.com/projects/variational-autoencoder
- Morphing faces: http://vdumoulin.github.io/morphing_faces/online_demo.html
- MNIST demo: http://dpkingma.com/sgvb_mnist_demo/demo.html
- Chemical molecule design with VAEs: https://github.com/aspuru-guzik-group/chemical_vae

- Next Topic
  - Generative adversarial networks