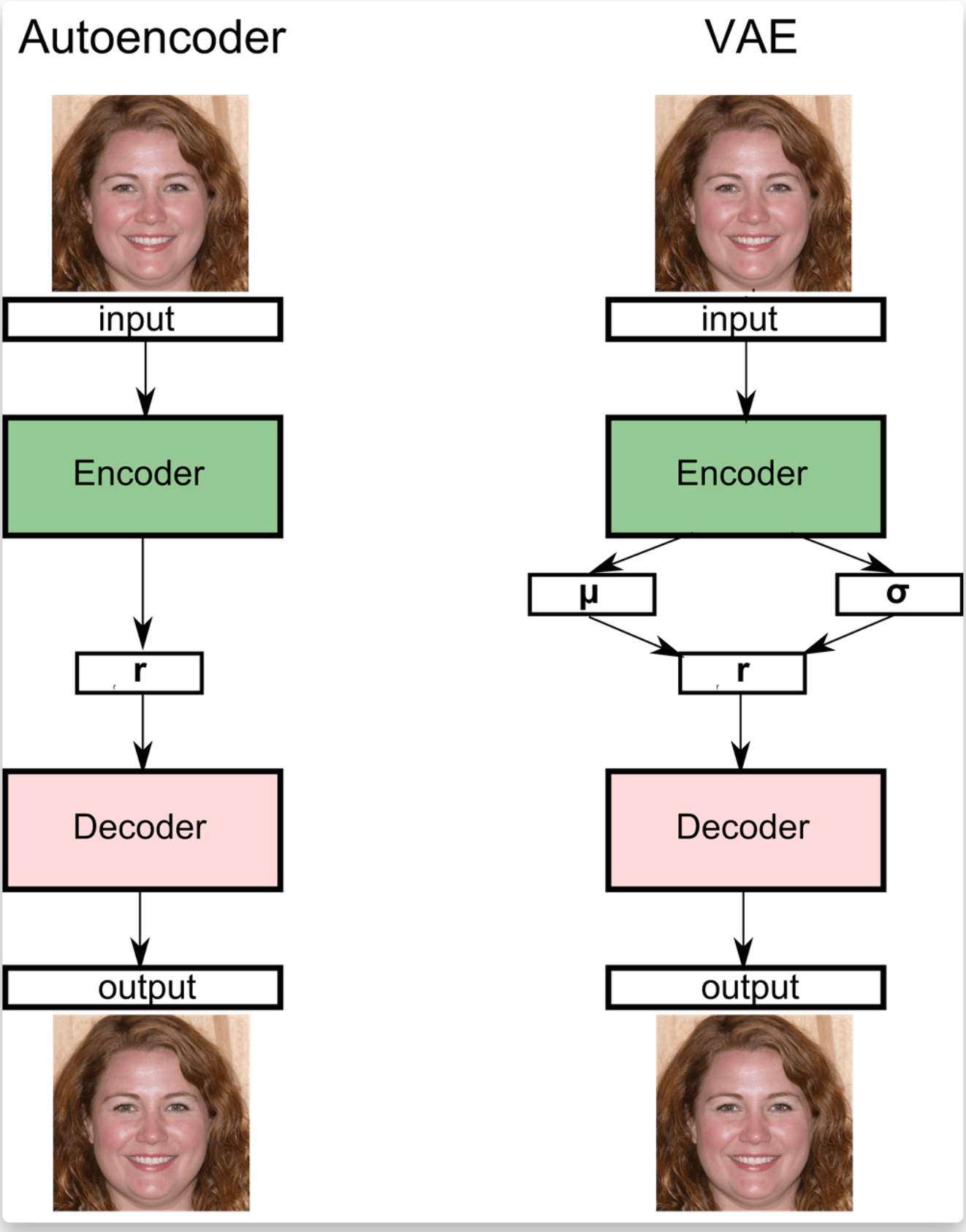
AE AND VAE


Autoencoder
input
Encoder
r
Decoder
output
VAE
input
Encoder
μ
σ
r
Decoder
output

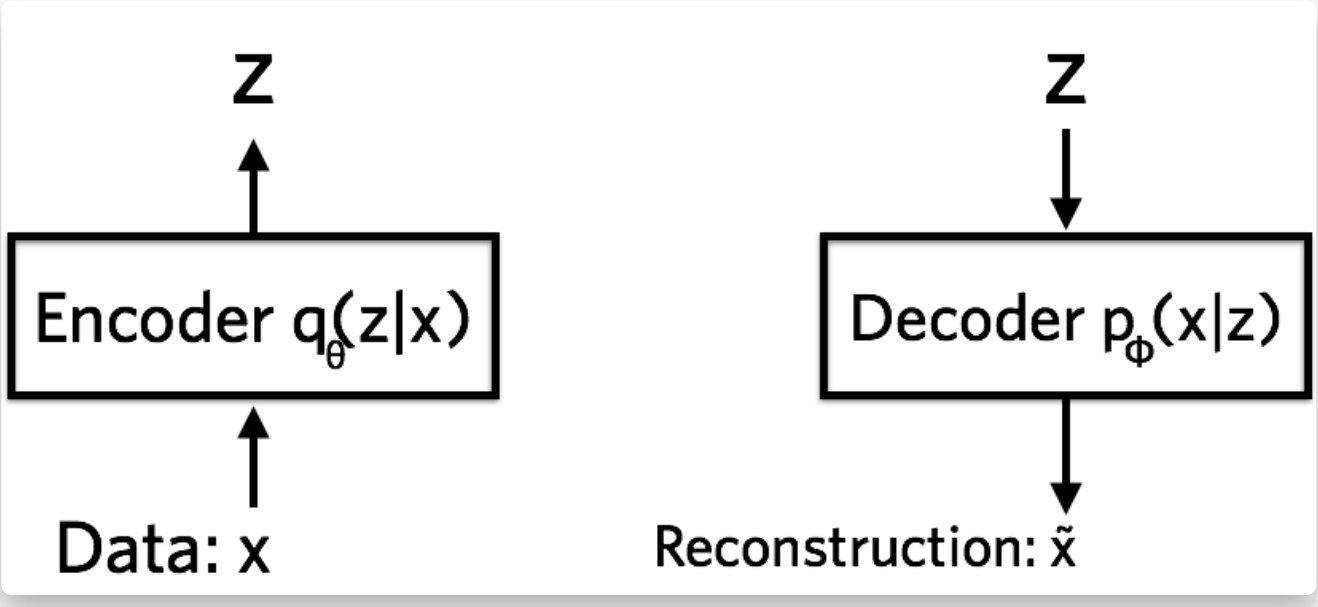
AE

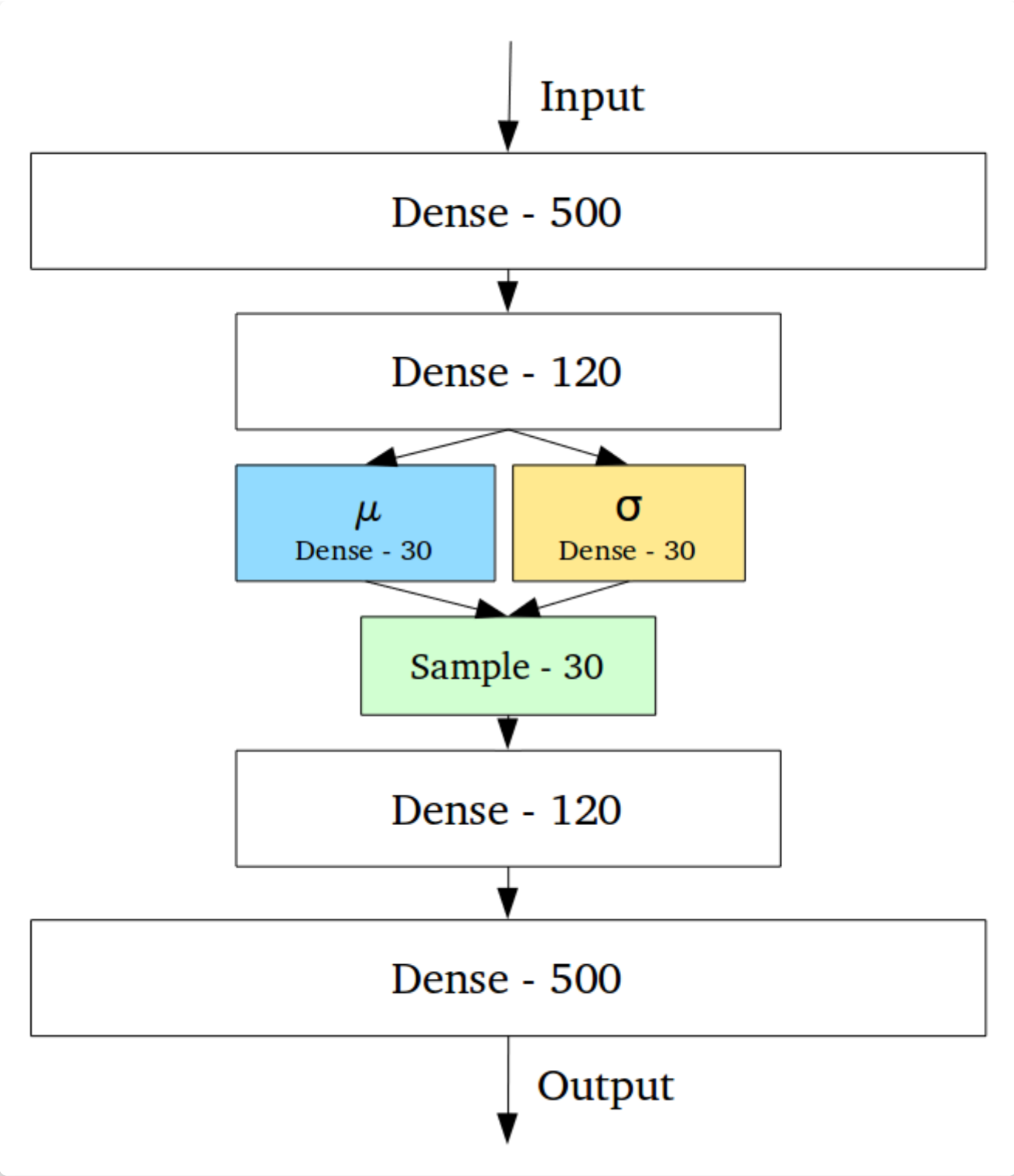
various models (undercomplete, sparse, denoising, contractive) which take data as input and discover some latent state representation of that data. More specifically, our input data is converted into an ***encoding vector*** where each dimension represents some learned attribute about the data. The most important detail to grasp here is that our encoder network is outputting a ***single value*** for each encoding dimension. The decoder network then subsequently takes these values and attempts to recreate the original input.

每个编码维度输出一个值。 解码器网络随后采用这些值并尝试重新创建原始输入。

VAE

H3 The neural net perspective





In neural net language, a variational autoencoder consists of an encoder, a decoder, and a loss function.

The ***encoder*** is a neural network. Its input is a datapoint x, **its output is a hidden representation** z and it has weights and biases θ,To be concrete, let's say x is a 28 by 28-pixel photo of a handwritten number. The encoder 'encodes' **the data which is 784-dimensional** into a latent (hidden) representation space z, which is much less than 784 dimensions. This is typically referred to as a 'bottleneck' because the encoder must learn an

efficient compression of the data into this lower-dimensional space. Let's denote the encoder $q_\theta(z \mid x)$. We note that the lower-dimensional space is stochastic: the encoder outputs parameters to $q_\theta(z \mid x)$ which is a **Gaussian probability density.** We can sample from this distribution to get noisy values of the representations z.

The **decoder** is another neural net. Its input is the representation z, it outputs the parameters to the probability distribution of the data, and has weights and biases ϕ ,The decoder is denoted(表示) by $p_\phi(x \mid z)$ Running with the handwritten digit example, let's say the photos are black and white and represent each pixel as 0 , 1 The probability distribution of a single pixel can be then represented using a Bernoulli distribution. The decoder gets as input the latent representation of a digit z and outputs 784 Bernoulli parameters, one for each of the 784 pixels in the image .The decoder 'decodes' the real-valued numbers in z into 784 real-valued numbers between 0 and 1. Information from the original 784-dimensional vector cannot be perfectly transmitted because the decoder only has access to a summary of the information (in the form of a less-than 784-dimensional vector z) how much information is lost?

We measure this using the reconstruction log-likelihood $\log p_\phi(x \mid z)$ **whose units are nats. This measure tells us how effectively the decoder has learned to reconstruct an input image** x **given its latent representation** z.

The ***loss function*** of the variational autoencoder is the negative log-likelihood with a regularizer. Because there are no global representations that are shared by all datapoints, we can decompose the loss function into only terms that depend on a single datapoint l_i The total loss is then $\sum_{i=1}^{N} l_i$ for N total datapoints. The loss function l_i **for datapoint** x_i is

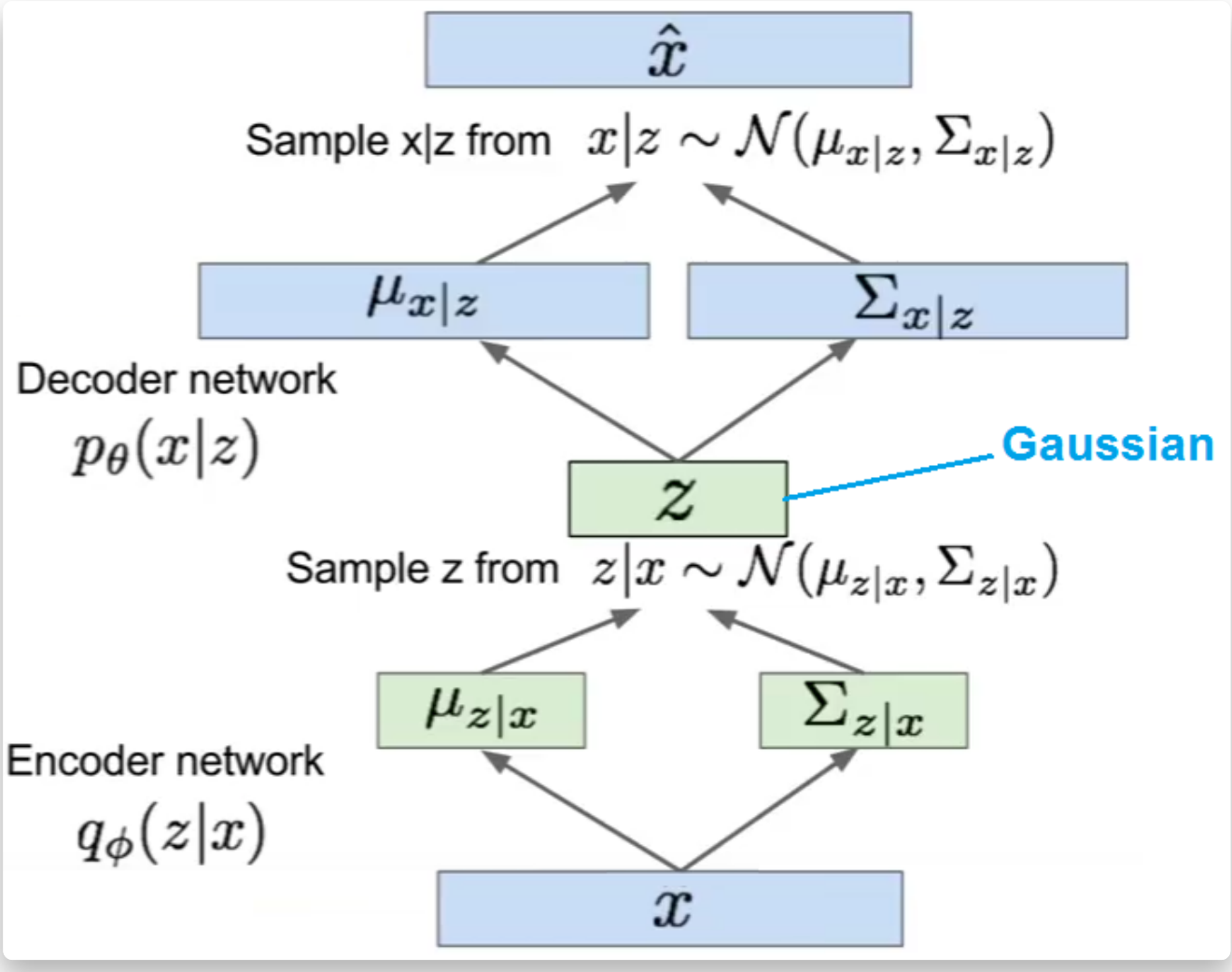
$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\phi\left(x_i \mid z\right)\right] + \mathbb{KL}\left(q_\theta\left(z \mid x_i\right) \| p(z)\right)$$

The first term is the reconstruction loss, or expected negative log-likelihood of the $i-th$ datapoint. The expectation is taken with respect to the encoder's distribution over the representations. this term encourages the decoder to learn to reconstruct the data. If the decoder's output does not reconstruct the data well, statistically we say that the decoder parameterizes a likelihood distribution that does not place much probability mass on the true data. For example, if our goal is to model black and white images and our model places high probability on there being black spots where there are actually white spots, this will

yield the worst possible reconstruction. Poor reconstruction will incur a large cost in this loss function.

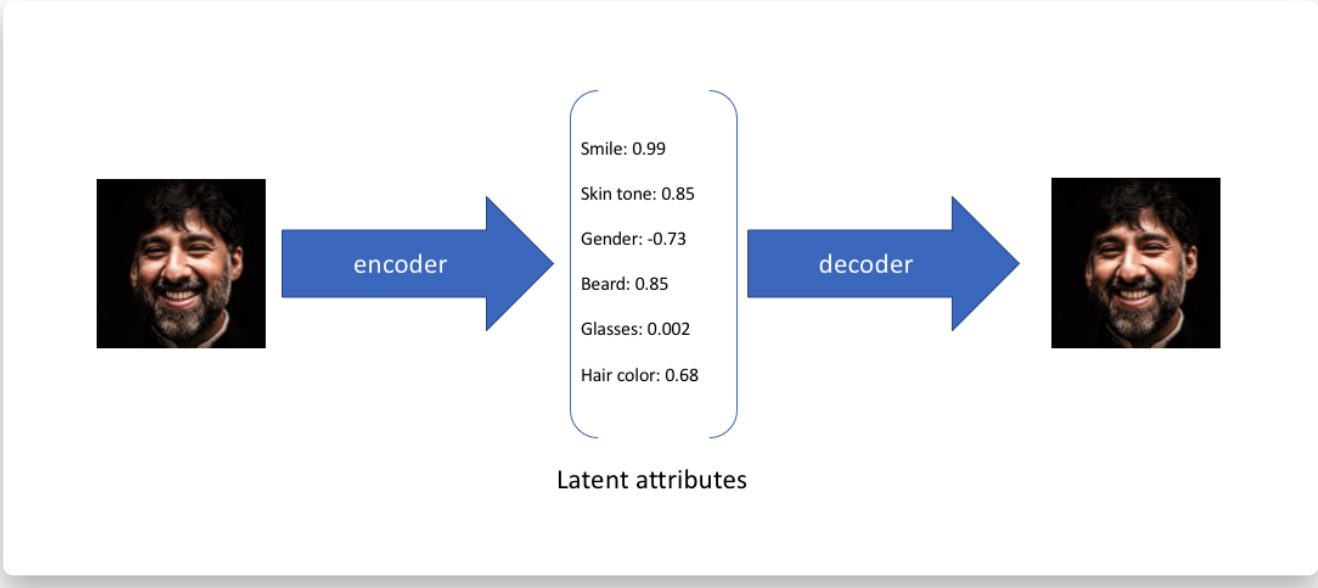
The second term is a regularizer that we throw in (we'll see how it's derived later). This is the KL divergence between the encoder's distribution $q_\theta(z \mid x)$ and $p(z)$ This divergence measures how much information is lost (in units of nats) when using q to represent p . It is one measure of how close q is to p.

In the variational autoencoder, p is specified as a standard Normal distribution with mean zero and variance one, or $p(z) = Normal(0, 1)$. if the encoder outputs represnetations z that are different than those from a standard normal distribution, it will receive a penalty in the loss. This regularizer term mean 'keep the representations z of each digit sufficently diverse'. If we didn't include the regularizer, the encoder could learn to cheat and give each datapoint a representation in a different region of Euclidean space. (如果不加KL 则当我们输出不同人写的相同数字时候，模型会认为这是不同的数字)

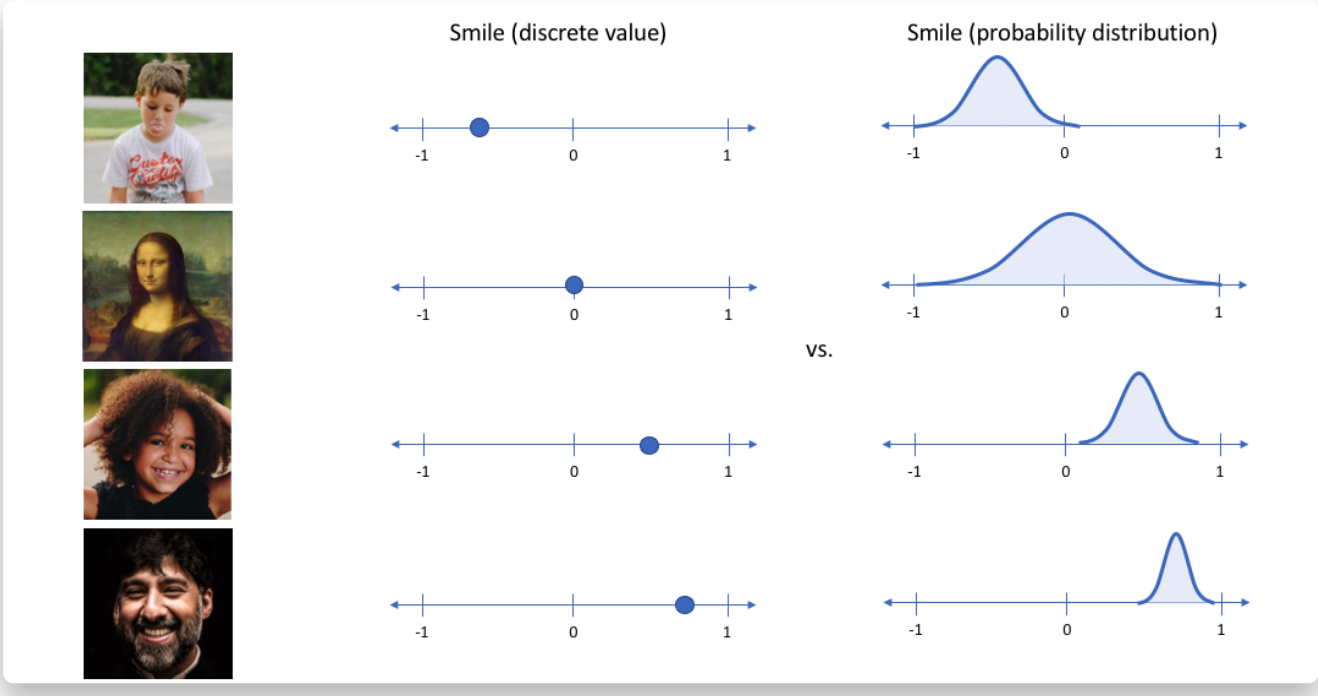


变分自动编码器 (VAE) 提供了一种概率方式来描述潜在空间(latent space)中的观察。 因此，我们不是构建一个输出单个值来描述每个潜在状态属性的编码器，而是制定我们的编码器来描述每个潜在属性的概率分布。

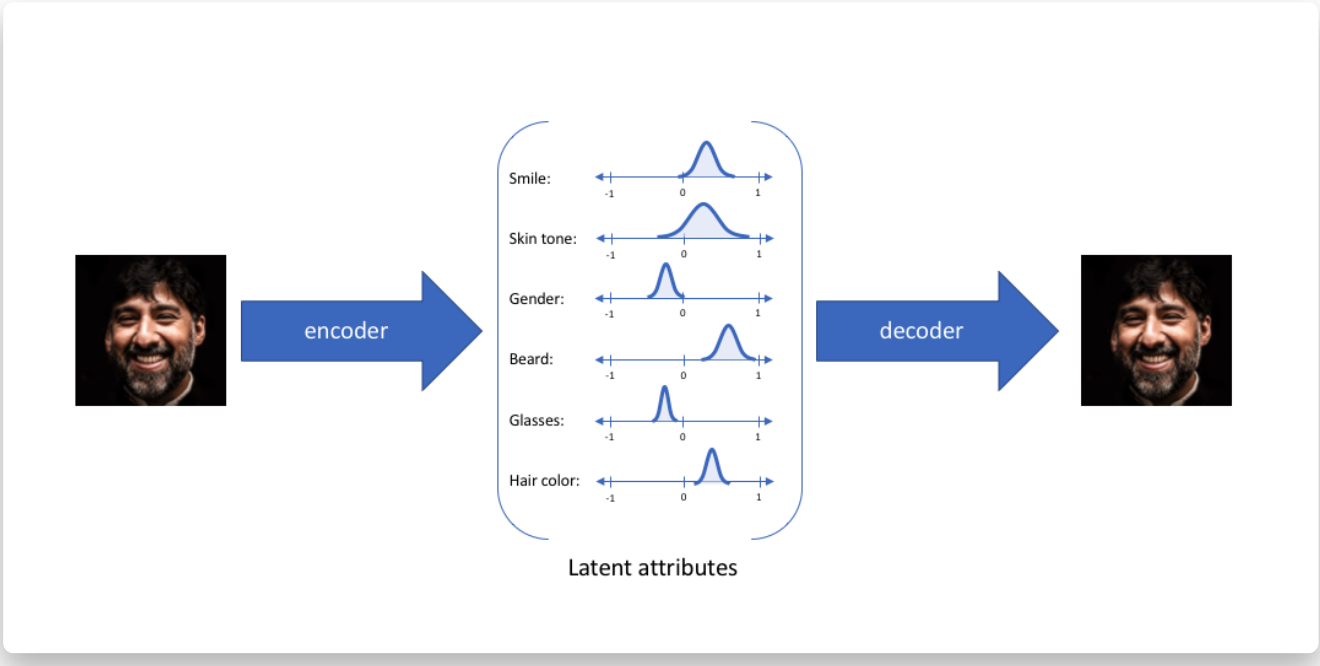
举个例子，假设我们已经在一个编码维度为 6 的大型面部数据集上训练了一个自动编码器模型。理想的自动编码器将学习面部的描述性属性，例如肤色，是否戴眼镜， 等等，试图以某种压缩表示形式描述观察结果。



在上面的示例中，我们使用单个值来描述每个属性，根据其潜在属性来描述输入图像。 然而，我们可能更愿意将每个潜在属性表示为一系列可能的值。 例如，如果您输入一张蒙娜丽莎的照片，您会为微笑属性分配什么单一值？ 使用变分自动编码器，我们可以用概率术语描述潜在属性。

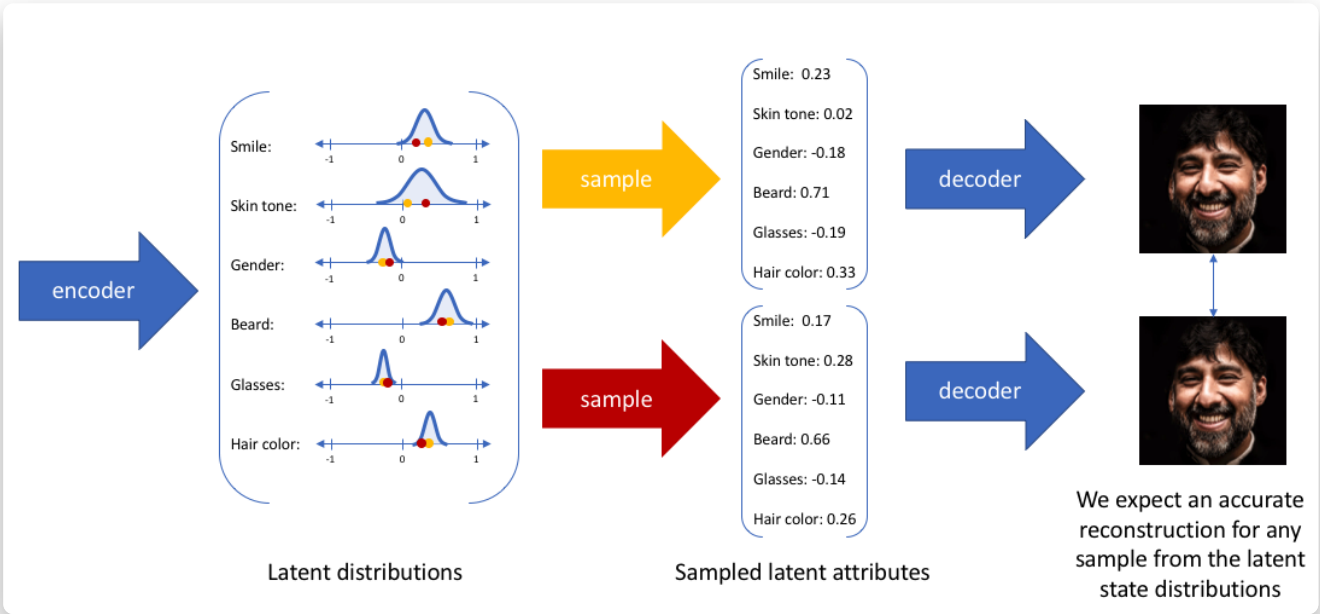


通过这种方法，我们现在将给定输入的每个潜在属性表示为概率分布。 当从潜在状态解码时，我们将从每个潜在状态分布中随机采样以生成一个向量作为我们解码器模型的输入。

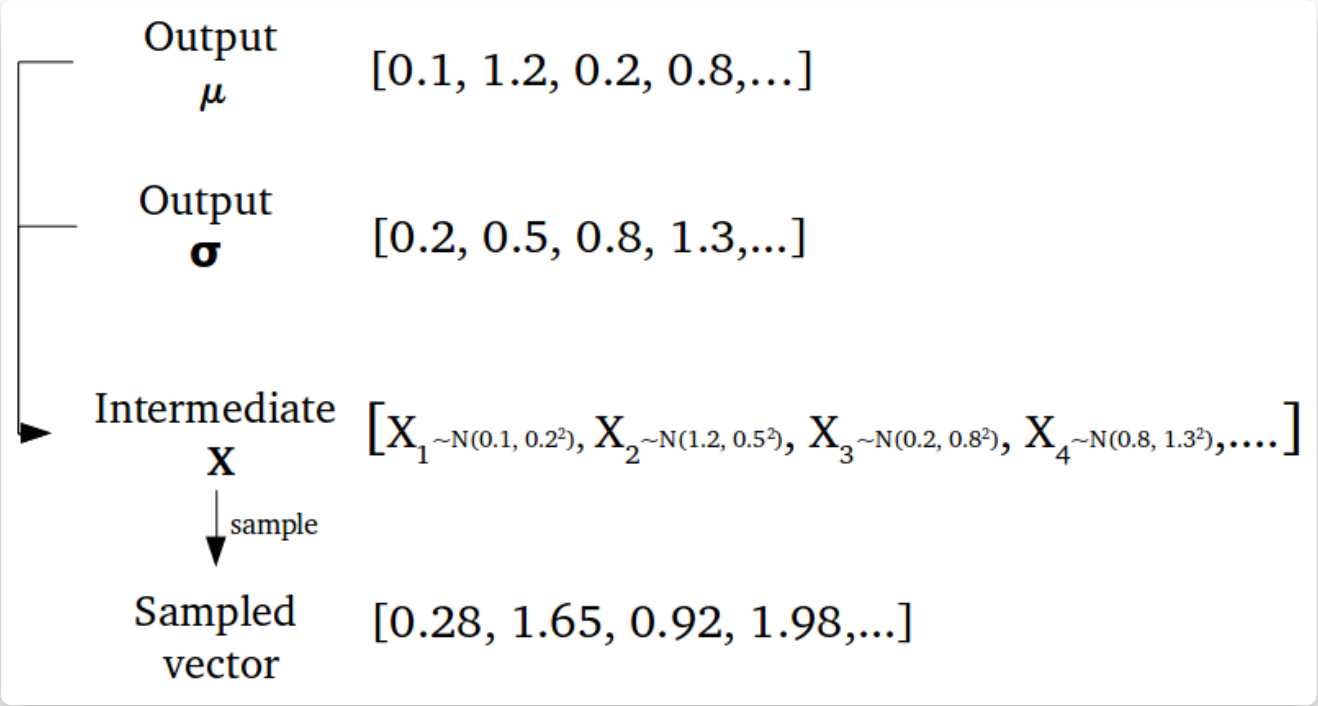


For variational autoencoders, the encoder model is sometimes referred to as the recognition model whereas the decoder model is sometimes referred to as the generative model.

通过构建我们的编码器模型以输出一系列可能的值（统计分布），我们将从中随机抽样以输入我们的解码器模型，我们实质上是在执行连续、平滑的潜在空间表示。 对于潜在分布的任何采样，我们期望我们的解码器模型能够准确地重建输入。 因此，潜在空间中彼此接近的值应该对应于非常相似的重建。



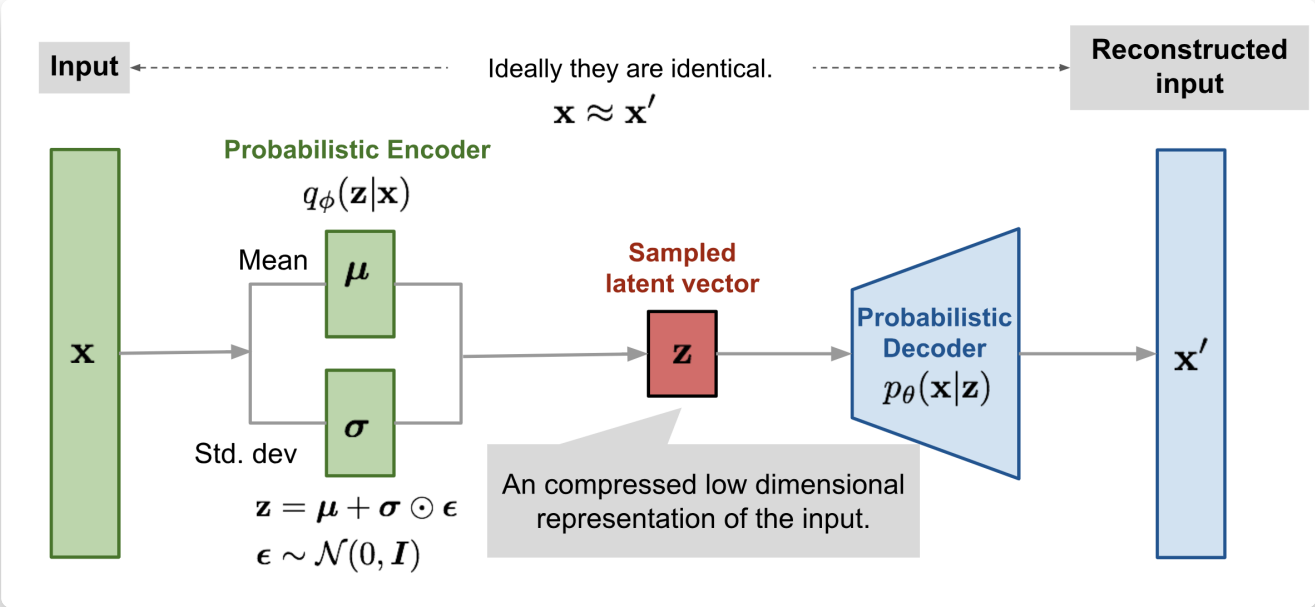
它们形成长度为 n 的随机变量向量的参数，其中 μ 和 σ 的第 i 个元素是第 i 个随机变量 X_i 的均值和标准差，我们从中采样，以获得采样编码 我们传递给解码器：



This stochastic generation means, that even for the same input, while the mean and standard deviations remain the same, the actual encoding will somewhat vary on every single pass simply due to sampling.

这种随机生成意味着，即使对于相同的输入，虽然均值和标准差保持不变，但实际编码在每次通过时都会因采样而有所不同。

STATISICAL MOTIVATION



The idea of **Variational Autoencoder** (Kingma & Welling, 2014), short for **VAE**, is actually less similar to all the autoencoder models above, but deeply rooted in the methods of variational bayesian and graphical model.

Let's label this distribution as p_θ , parameterized by θ . The relationship between the data input x and the latent encoding vector z can be fully defined by:

- Pior $p_\theta(z)$ 先验分布
- Likelihood $p_\theta(x|z)$
- Posterior $p_\theta(z|x)$ 后验

First, sample a $\mathbf{z}^{(i)}$ from a prior distribution $p_{\theta^*}(\mathbf{z})$ Then a value $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}\left(\mathbf{x} \mid \mathbf{z} = \mathbf{z}^{(i)}\right)$

The optimal parameter θ^* is **the one that maximizes the probability of generating real data samples:**

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta\left(\mathbf{x}^{(i)}\right)$$

Commonly we use the log probabilities to convert the product on RHS to a sum:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta\left(\mathbf{x}^{(i)}\right)$$

Now let's update the equation to better demonstrate the data generation process so as to involve the encoding vector:

$$p_\theta\left(\mathbf{x}^{(i)}\right) = \int p_\theta\left(\mathbf{x}^{(i)} \mid \mathbf{z}\right) p_\theta(\mathbf{z}) d\mathbf{z}$$

we would like to introduce a new approximation function to output what is a likely code given an input x , $q_\phi(\mathbf{z} \mid \mathbf{x})$, parameterized by ϕ

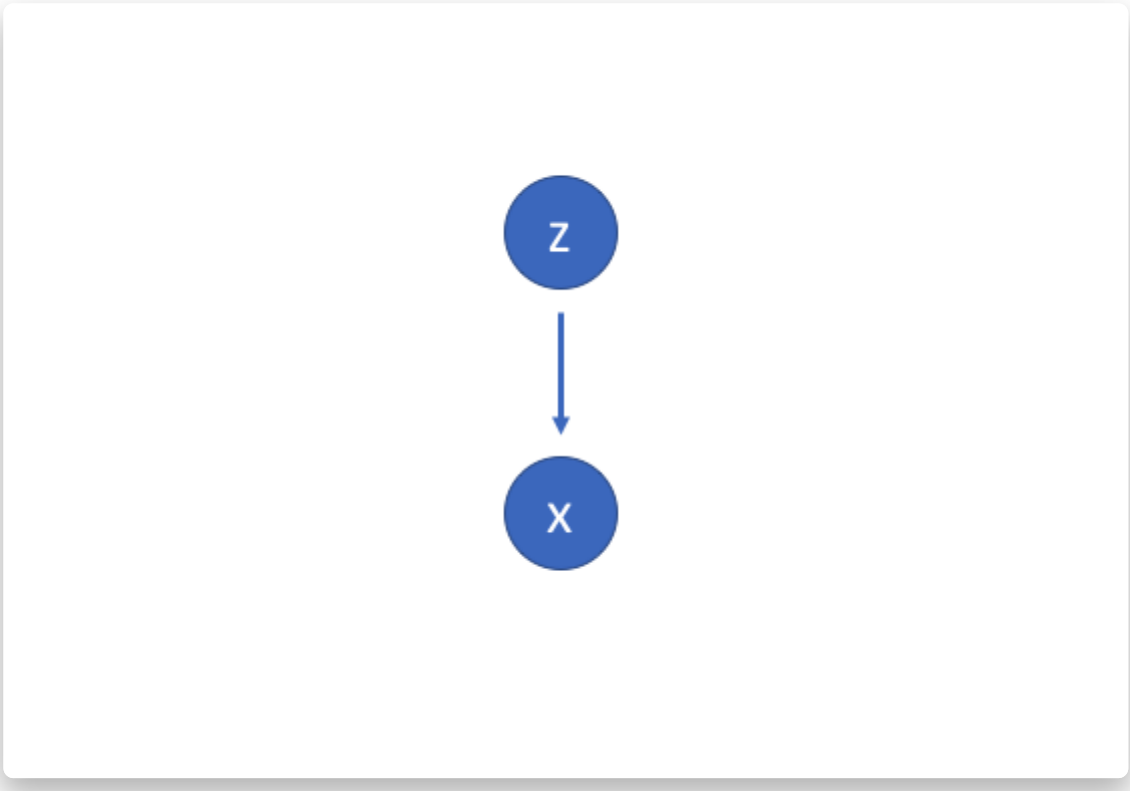
In the probability model framework, a variational autoencoder contains a specific probability model of data x and latent variables z. We can write the joint probability of the model as $p(x, z) = p(x \mid z)p(z)$ The generative process can be written as follows.

We can decompose this into the likelihood and prior: $p(x, z) = p(x \mid z)p(z)$

For each datapoint i :

- Draw latent variables $z_i \sim p(z)$
- Draw datapoint $x_i \sim p(x \mid z)$

Suppose that there exists some hidden variable z which generates an observation x.



We can only see x, but we would like to infer the characteristics of z. In other words, we'd like to compute $p(z \mid x)$.

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}$$

Unfortunately, computing $p(x)$ is quite difficult.

$$p(x) = \int p(x \mid z)p(z)dz$$

如果z是高维的那么可能会出现n重积分.

This usually turns out to be an intractable distribution. However, we can apply **variational inference** to estimate this value.

Variational inference approximates the posterior with a family of distributions $q_\lambda(z \mid x)$ The variational parameter λ indexes the family of distributions. For example, if q were Gaussian, it would be the mean and variance of the latent variables for each datapoint $\lambda_{x_i} = (\mu_{x_i}, \sigma^2_{x_i}))$

Let's approximate $p(z \mid x)$ by another distribution $q(z \mid x)$which we'll define such that it has a tractable distribution. If we can define the parameters of $q(z \mid x)$ such that it is very similar to $p(z \mid x)$, we can use it to perform approximate inference of the intractable distribution.

Recall that the KL divergence is a measure of difference between two probability distributions.

Thus, if we wanted to ensure that $q(z \mid x)$**was similar to** $p(z \mid x)$, we could minimize the KL divergence between the two distributions. （见推导pdf）

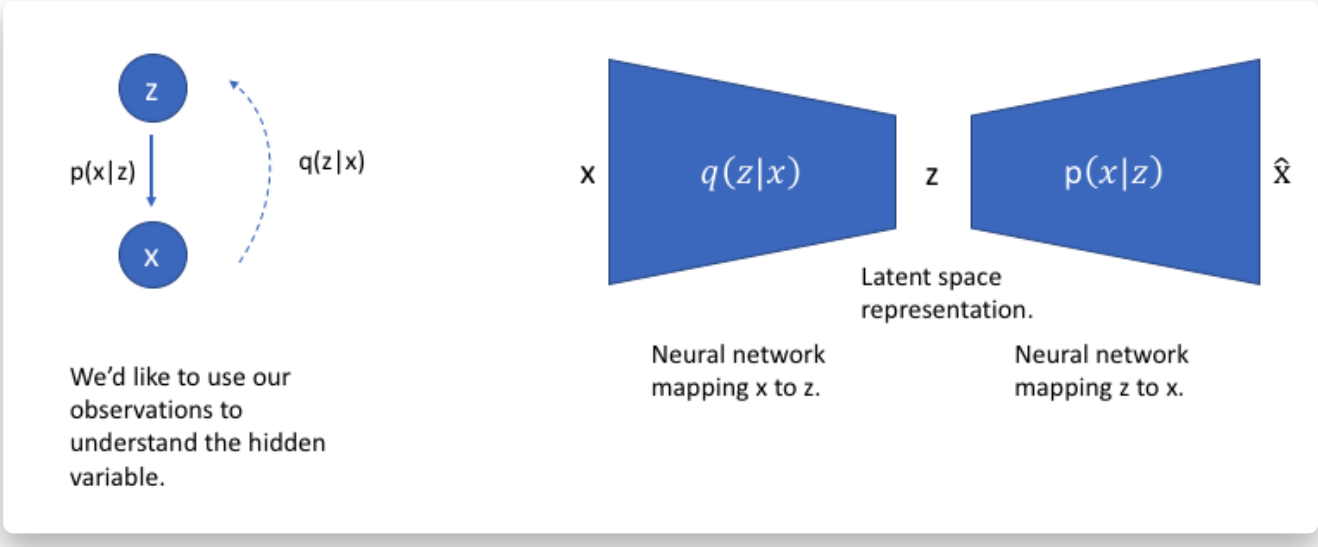
$$\mathbb{KL}\left(q_\lambda(z \mid x) || p(z \mid x)\right) = \\ \mathbb{E}_q\left[\log q_\lambda(z \mid x)\right] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

Dr. Ali Ghodsi goes through a full derivation(推导) here, but the result gives us that we can minimize the above expression by maximizing the following:

$$E_{q(z|x)} \log p(x \mid z) - KL(q(z \mid x) || p(z))$$

The first term represents the reconstruction likelihood， the second term ensures that our learned distribution q is similar to the true prior distribution p. 整体是L

To revisit our graphical model, we can use q to infer the possible **hidden variables** (ie. latent state) which was used to generate an observation. We can further construct this model into a neural network architecture where the encoder model learns a mapping from x to z and the decoder model learns a mapping from z back to x .

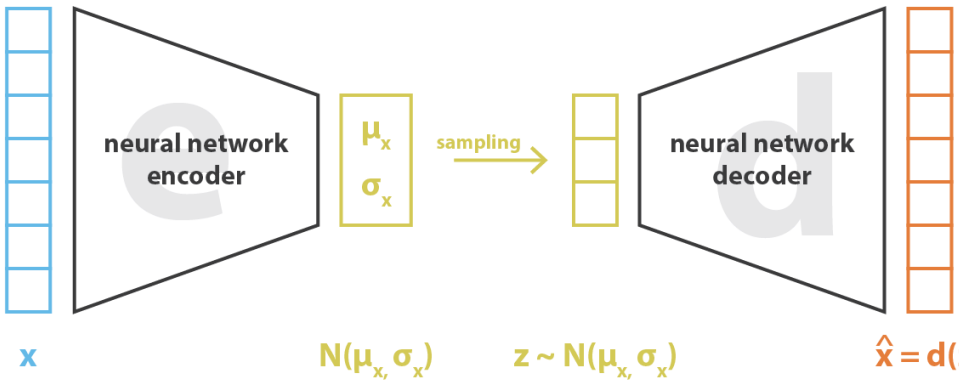


Our loss function for this network will consist of two terms, one which penalizes (惩罚) (reconstruction error (which can be thought of maximizing the reconstruction likelihood as discussed earlier) and a second term which encourages our learned distribution $q(z \mid x)$

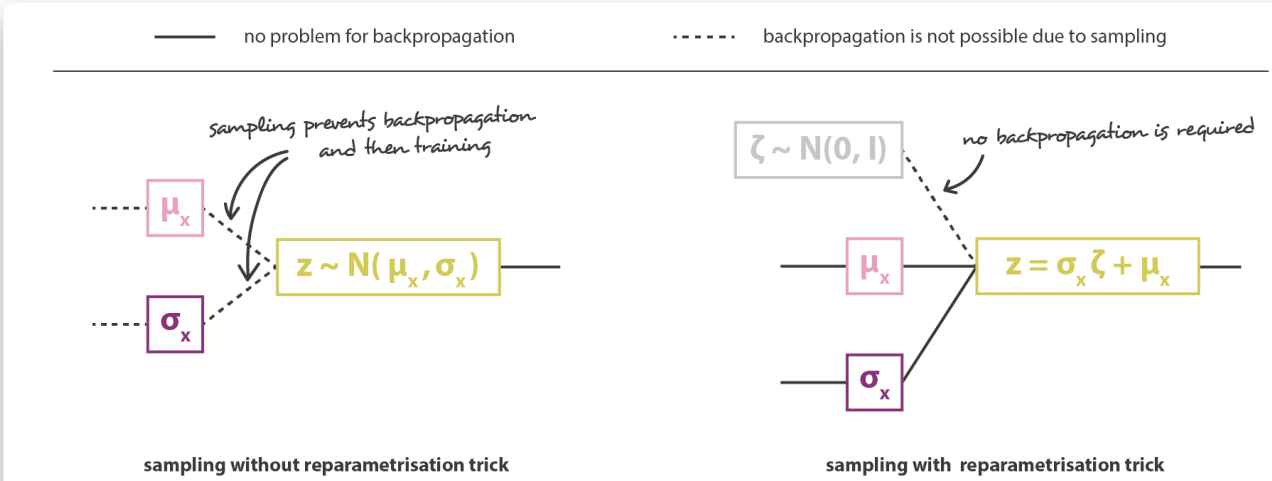
to be similar to the true prior distribution $p(z)$, which we'll assume follows a unit Gaussian distribution, for each dimension j of the latent space.

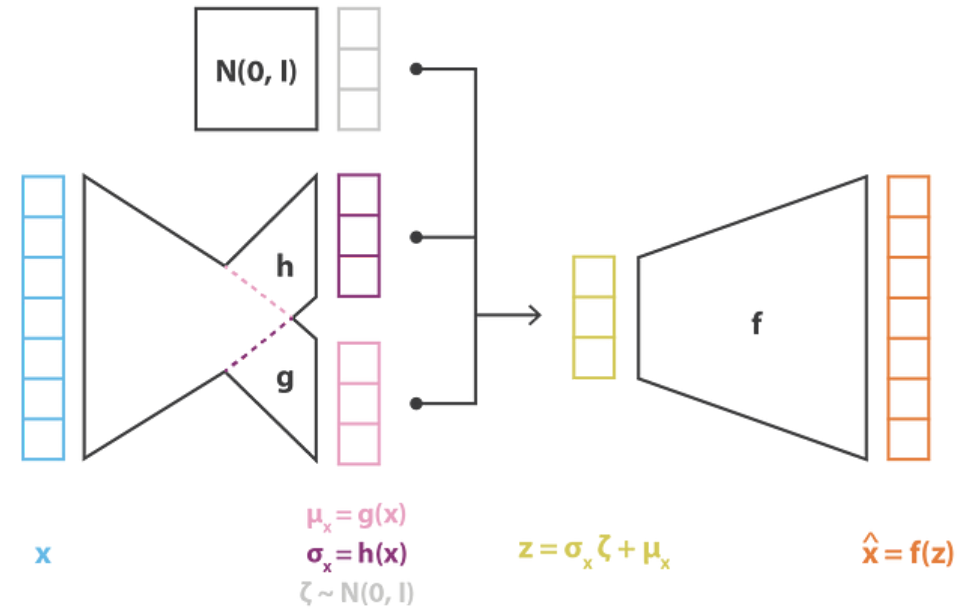
$$\mathcal{L}(x, \hat{x}) + \sum_j KL\left(q_j(z \mid x) || p(z)\right)$$

$$\mathcal{L}(x, \hat{x}) + \beta \sum_j KL\left(q_j(z \mid x) || N(0, 1)\right)$$



$$\text{loss} = ||x - \hat{x}||^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$





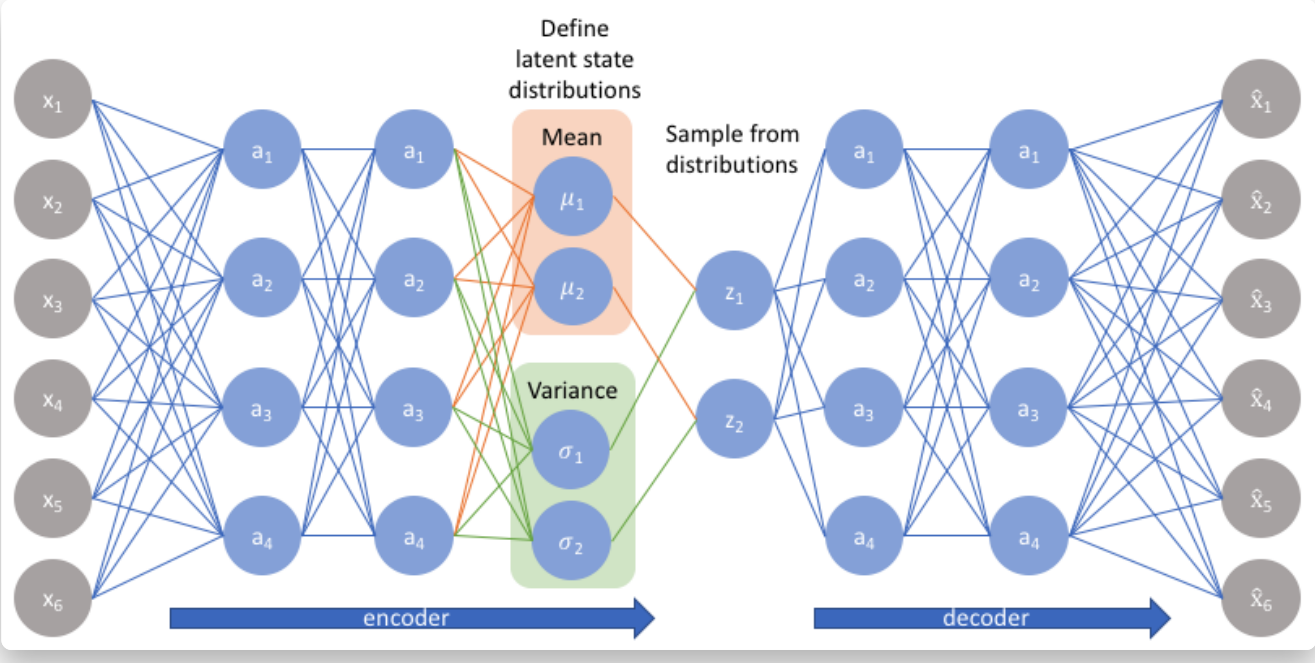
$$\text{loss} = C||x - \hat{x}||^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = C||x - f(z)||^2 + KL[N(g(x), h(x)), N(0, I)]$$

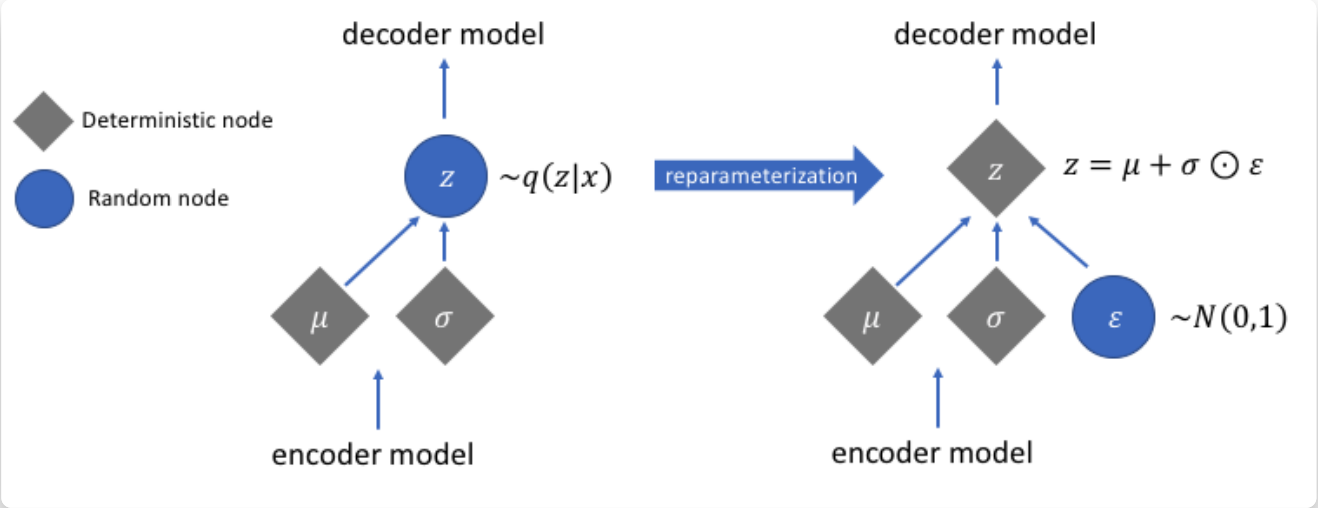
IMPLEMENTATION

Rather than directly outputting values for the latent state as we would in a standard autoencoder, the encoder model of a VAE will output parameters describing a distribution for each dimension in the latent space. Since we're assuming that our prior follows a normal distribution, we'll output ***two*** vectors describing the mean and variance of the latent state distributions. If we were to build a true multivariate Gaussian model, we'd need to define a covariance matrix describing how each of the dimensions are correlated. However, we'll make a simplifying assumption that our covariance matrix only has nonzero values on the diagonal, allowing us to describe this information in a simple vector.

VAE 的编码器模型将输出描述潜在空间中每个维度分布的参数，而不是像我们在标准自动编码器中那样直接输出潜在状态的值。 由于我们假设我们的先验服从正态分布，我们将输出两个描述潜在状态分布的均值和方差的向量。 如果我们要建立一个真正的多元高斯模型，我们需要定义一个协方差矩阵来描述每个维度是如何相关的。 然而，我们将做一个简化的假设，即我们的协方差矩阵在对角线上只有非零值，允许我们在一个简单的向量中描述这个信息。

Our decoder model will then **generate a latent vector** by sampling from these defined distributions and proceed to develop a reconstruction of the original input.





With this reparameterization, we can now optimize the ***parameters*** of the distribution while still maintaining the ability to randomly sample from that distribution.

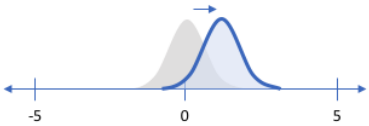
Screen-Shot-2018-03-18-at-4.39.41-PM

变分自动编码器的主要好处是我们能够学习输入数据的平滑潜在状态表示。 对于标准自动编码器，我们只需要学习一种允许我们重现输入的编码。 正如您在最左边的图中看到的，仅关注重建损失确实允许我们分离出类别（在本例中为 MNIST 数字），这应该允许我们的解码器模型能够再现原始手写数字，但是有一个 潜在空间内数据分布不均。 换句话说，潜在空间中的某些区域不代表我们观察到的任何数据。

另一方面，如果我们只关注确保潜在分布与先验分布相似（通过我们的 KL 散度损失项），我们最终会使用相同的单位高斯来描述每个观察，我们随后从中抽样来描述 可视化的潜在维度。 这有效地将每个观察结果视为具有相同的特征； 换句话说，我们未能描述原始数据。

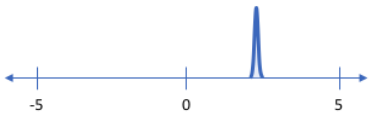
然而，当同时优化这两项时，我们鼓励我们描述观察的潜在状态，其分布接近先验但在必要时偏离以描述输入的显着特征。

Penalizing reconstruction loss encourages the distribution to describe the input



Our distribution deviates from the prior to describe some characteristic of the data

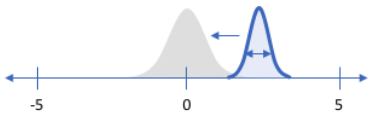
Without regularization, our network can “cheat” by learning narrow distributions



With a small enough variance, this distribution is effectively only representing a single value

Penalizing KL divergence acts as a regularizing force

Attract distribution to have zero mean



Ensure sufficient variance to yield a smooth latent space