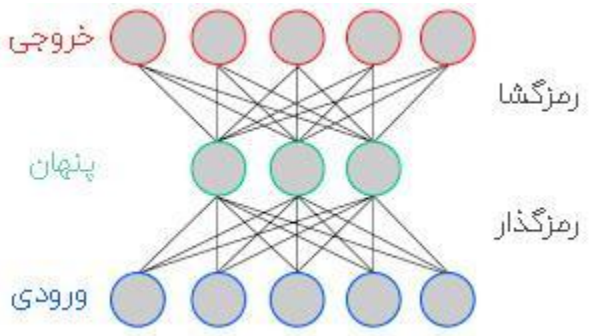
## Exercise 1

### How does the VAE architecture allow it to generate new data points, especially compared to associative auto-encoder, which cannot generate new data points?

An **Autoencoder** is a type of neural network that takes an image as input and reconstructs it using fewer bits. It consists of an encoder, a decoder, and a series of hidden layers between them. The basic idea behind an autoencoder is simple: set up an encoder and decoder as neural networks and learn the best encoding-decoding scheme through an iterative optimization process. In each iteration, the model is fed with data, the encoded-decoded output is compared to the original data, and the error is propagated through the architecture to update the network weights.




There are various types of autoencoders, such as **Variational Autoencoders (VAE)**, **Sparse Autoencoders**, and **Denoising Autoencoders**. The focus of our discussion here is the **Variational Autoencoder (VAE)**.

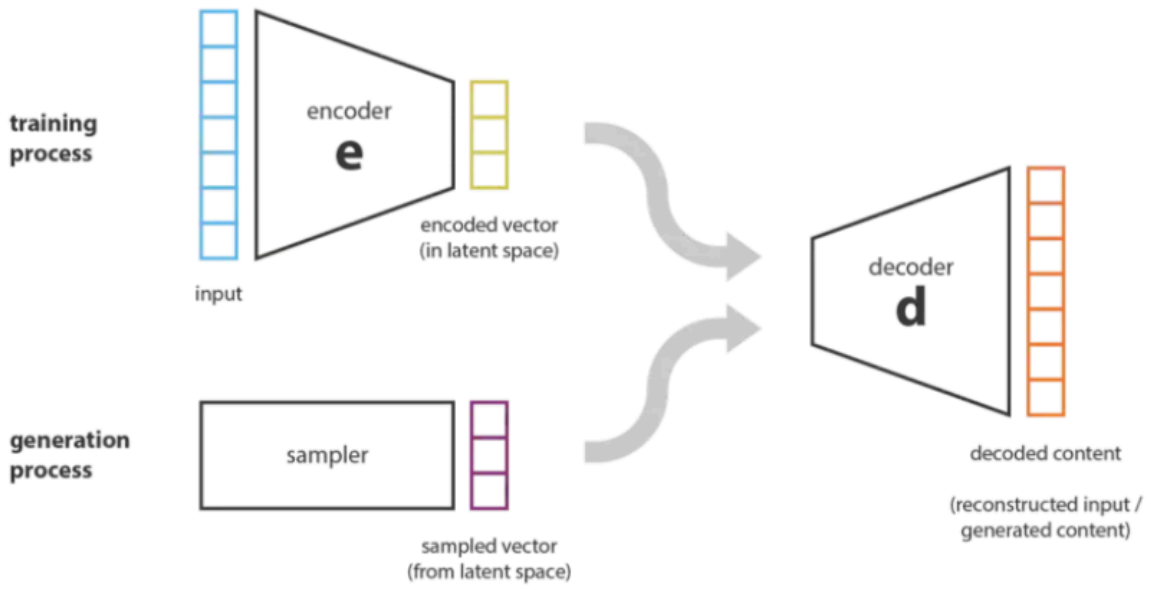
The encoder compresses the input, and the decoder tries to reconstruct the input from the compressed version provided by the encoder. An autoencoder is a neural network trained to copy its input to its output. The relationship between autoencoders and data generation is a bit more complex than just simple guesses. By adding non-linear functions (such as non-linear activation functions and more hidden layers), the autoencoder can learn fairly powerful representations of input data in lower dimensions with minimal information loss.

Now, let's examine the problem of generating data and look at the limitations of the current form of the autoencoder. Finally, we will introduce **Variational Autoencoders**.

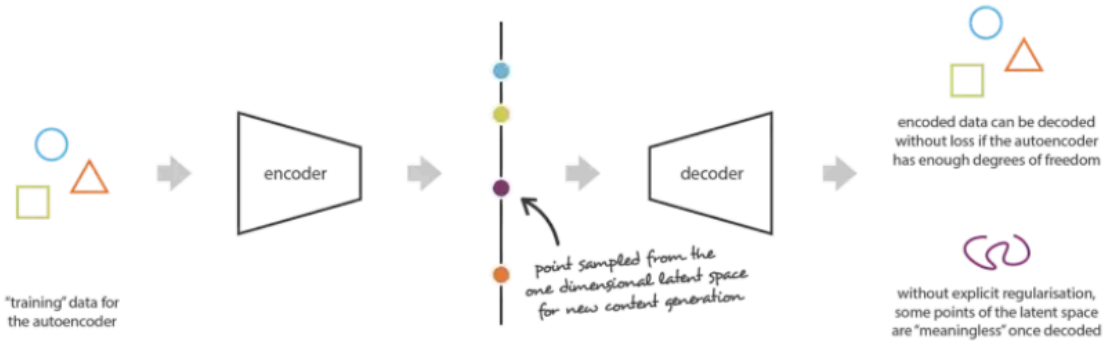
**Key Differences between Autoencoders and Variational Autoencoders**

In a standard autoencoder, the encoder outputs hidden vectors. However, in a **Variational Autoencoder (VAE)**, instead of outputting vectors in the hidden space, the encoder outputs parameters of a predefined distribution in the hidden space for each input. It imposes a constraint on this hidden distribution, forcing it to be normal (Gaussian).

The main difference between the two models is that, in a VAE, the input is encoded into two vectors instead of one. These two vectors are used to define a normal distribution, from which the latent representation of the input is sampled.



Once the autoencoder is trained, we have both an encoder and a decoder, but there's still no real way to generate new content. One way to think about it is that if the latent space is well-organized (organized well by the encoder during training), we can randomly pick a point from the latent space and decode it to obtain new data. The decoder, in this case, works similarly to a **Generative Adversarial Network (GAN)** generator.

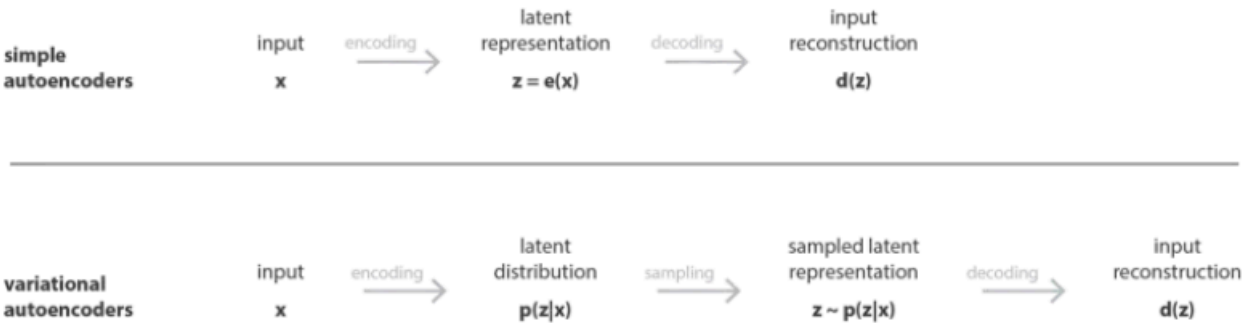



**VAE for Data Generation**

To use the decoder of the autoencoder for generative purposes, we need to ensure that the latent space is sufficiently organized. Thus, a **Variational Autoencoder** can be defined as a model where training is regularized to prevent overfitting, ensuring that the latent space has properties that make data generation possible.

The main distinction between VAEs and regular autoencoders is that in VAEs, we have more control over the hidden space between the encoder and decoder. We know the range of values in this space, which helps generate new data effectively. VAEs utilize a normal probability distribution to generate distributions in the hidden space instead of vectors of numbers. Unlike other models, a VAE does not aim for a one-to-one correspondence for data generation but rather aims to map from the same region of the latent space back to an output.

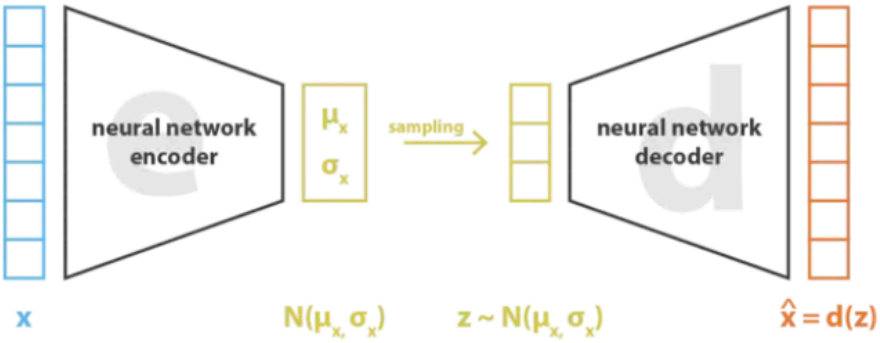
Initially, the input is encoded as a distribution in the latent space, then a point in this space is sampled from that distribution, and the sampled point is decoded. Finally, the error is computed and propagated back through the network.




**Regularization in VAE**

The encoded distributions are chosen to be normal so that the encoder learns to return the mean and covariance matrix describing these Gaussians. The reason an input is encoded as a distribution with some variance (rather than a single point) is to naturally enable the regularization of the latent space. The returned distributions from the encoder are forced to be close to a standard normal distribution.

Thus, the loss function minimized during the training of a VAE consists of a "reconstruction term" (in the final layer), which aims to make the encoding-decoding scheme as efficient as possible, and a "regularization term" (in the hidden layer), which organizes the latent space by making the distributions returned by the encoder close to a standard normal distribution.




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

The latent space for data generation must exhibit two key properties:

1. **Continuity**: Two nearby points in the latent space should not decode into two completely different data points.
2. **Completeness**: Any point sampled from the latent space should produce meaningful data when decoded.

## Exercise 2

Variational auto-encoders optimize a lower bound of the data likelihood for a given input sample $x$ $(i)$ such that

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x^{(i)})}[log\, p_\theta(x^{(i)}|z)] - D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)).$$

Explain the task of the KL−divergence term.

Write down the advantage of modeling $(z)$ and $q\phi(z|x\ (i)\ )$ by using Normal distribution with a diagonal covariance matrix.

Explain the task of the first term and its effect on the latent space.

In the previous question, we saw that the loss function minimized during the training of a **Variational Autoencoder (VAE)** consists of a "reconstruction term" (in the final layer), which aims to make the encoding-decoding scheme as efficient as possible, and a "regularization term" (in the latent layer), which organizes the latent space by making the distributions returned by the encoder closer to a standard normal distribution. This regularization is expressed as the **Kullback-Leibler (KL) divergence** between the returned distribution and a standard Gaussian, which will be further explained. We can understand that the KL-divergence between two Gaussian distributions has a closed-form solution, which can be directly expressed in terms of the means and covariance matrices of the two distributions. Essentially, KL-divergence is a measure of how close two probability distributions are. This term exists in the loss function to bring the two input probability distributions closer during the learning process.

$$D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)).$$

As we can see, there are two inputs:

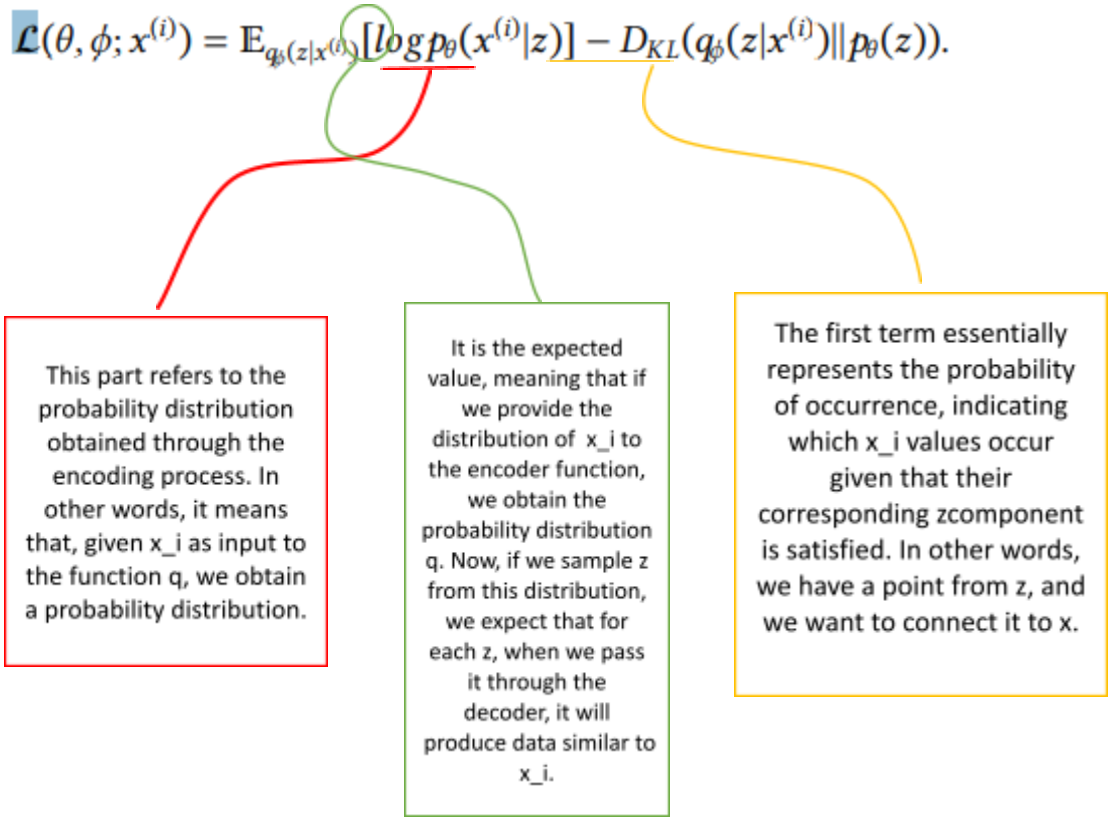
- The first is the probability distribution from the encoding process $\rightarrow q_\phi(z|x^{(i)})$
- The second is a fixed probability distribution that we choose to move the initial input closer to $\rightarrow p_0(z)$

The purpose of doing this is to expand the range of operation for generating new data. Without this term, the range would be quite similar to a regular autoencoder.

As we've seen, $p_0(z)$ is a fixed value that we define ourselves. Choosing this constant is important because on one hand, its continuity prevents two nearby points in the latent space from decoding into vastly different outputs, and on the other hand, its completeness ensures that the chosen probability distribution will generate meaningful outputs when sampled and decoded. If this constant has these characteristics, the latent space will be well-organized, but without them, the latent space will be disorganized. Therefore, to organize the latent space, we need to regularize both the covariance matrix and the means of the distributions returned by the decoder. Since in this process $p_0(z)$ is a normal distribution, the covariance matrix should approach the identity matrix to avoid **punctual distribution** issues.

Additionally, the means of the returned distributions should approach zero to prevent the encoded distributions from being too far apart from each other. In conclusion, having these two aspects in place will help maintain the two essential principles and improve the model overall.

Now, if we examine the effect of the first term, we can hypothesize that:




In other words, this term ultimately tries to adjust the weights of the encoder and decoder in such a way that the input data $p_i$ is reproduced in the output.