VAE

Encoder The encoder takes the input data and processes it to produce a representation in the latent space. This step involves learning the parameters (mean and variance) of a probability distribution (usually Gaussian) that represents the data in this latent space.

Specifically, for a given input x , the encoder outputs two things:

μ(x) and σ^2 (x), which are the mean and variance of the Gaussian distribution representing the latent space.

Sampling from the Latent Space To introduce randomness and ensure the model can generate different outputs, a sample z is drawn from the distribution N(μ(x),σ^2(x)N(μ(x),σ ^2 (x)). This step is crucial for the generative aspect of VAEs.

Because backpropagation cannot flow through a random node, the "reparameterization trick" is used. This involves sampling ϵ from a standard normal distribution and then computing z=μ(x) +σ(x)∗ϵz=μ(x)+σ(x)∗ϵ. This trick allows the gradient to bypass the randomness and makes it possible to train the model using gradient descent.

Decoder The sampled latent variable z is then passed to the decoder. The decoder's job is to reconstruct the input data from z. The output of the decoder is a reconstruction of the original input data, denoted as x'

The loss function in VAEs has two main components:

Reconstruction Loss: This measures how well the decoder can reconstruct the input data from the latent representation. It encourages the decoded samples to be as close as possible to the original inputs. This is often measured using mean squared error (for continuous data) or cross-entropy loss (for binary data).

KL Divergence: This component measures how much the learned latent distribution $q(z|x)$ (the encoder's output) diverges from the prior distribution $p(z)$ (often a standard Gaussian distribution). This term acts as a regularizer, encouraging the latent space to follow the desired distribution and preventing overfitting by not allowing the encoder to encode the input data too specifically to certain points in the latent space.

Diffusion model

Gradual Noising: The forward process incrementally adds Gaussian noise to an original data sample across a predefined number of steps, T. This process is described by a Markov chain, where each step slightly corrupts the data, moving it closer to pure noise. Mathematically, (see equation below) where ϵ is sampled from a standard normal distribution, and α_t controls the noise level at each step. $x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\epsilon \quad , \epsilon \sim \mathcal{N}(0, I)$

Diffusion Coefficients: The α _t parameters are crucial as they define how quickly the data is noised. These coefficients are predefined to ensure a smooth transition from data to noise over the T steps.

Denoising: The magic of diffusion models lies in the reverse process, where the model learns to denoise, starting from pure noise and gradually reconstructing the data by estimating the reverse diffusion steps. This process is parametrized by a neural network, which predicts the noise that was added at each step, effectively learning to generate data from the noise distribution.

Learning Objective: Training involves minimizing the difference between the original data and the

reconstructions at each step of the reverse process. The loss function typically includes a term for the likelihood of the reverse process (often modeled as the negative log-likelihood of the reverse Markov chain) and may incorporate additional terms to stabilize training and improve sample quality.