Autoencoder and Variational Autoencoders

Dr. Kamal Mammadov

October 8, 2024

Autoencoders: Definition

An autoencoder is a neural network designed to learn efficient representations of data. It is defined by the following components:

- **Two sets**:
  - The space of decoded messages $\mathcal{X}$ (input space).
  - The space of encoded messages $\mathcal{Z}$ (latent space).

  Typically, $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Z} = \mathbb{R}^n$, where $m > n$.

- **Two parameterized families of functions**:
  - The encoder family $E_\phi : \mathcal{X} \to \mathcal{Z}$, parameterized by ϕ.
  - The decoder family $D_\theta : \mathcal{Z} \to \mathcal{X}$, parameterized by θ.

- For any $x \in \mathcal{X}$, the latent representation is given by $z = E_\phi(x)$, referred to as the code or latent variable.

- For any $z \in \mathcal{Z}$, the (decoded) message is given by $x' = D_\theta(z)$.

Training an Autoencoder (1)

An autoencoder is trained to minimize the difference between its input and output. To judge its quality, we define:

- A reference probability distribution μ_{ref} over $\mathcal{X}$, typically given by the empirical distribution of a dataset $\{x_1, \ldots, x_N\} \subset \mathcal{X}$.
- A reconstruction quality function $d : \mathcal{X} \times \mathcal{X} \to [0, \infty]$, where $d(x, x')$ measures the difference between the original input x and the reconstructed output x'.

The loss function for the autoencoder is defined as:

$$L(\theta, \phi) := \mathbb{E}_{x \sim \mu_{\text{ref}}} \left[d(x, D_\theta(E_\phi(x))) \right]$$

The optimal autoencoder minimizes this loss:

$$\arg \min_{\theta, \phi} L(\theta, \phi)$$

Training an Autoencoder (2)

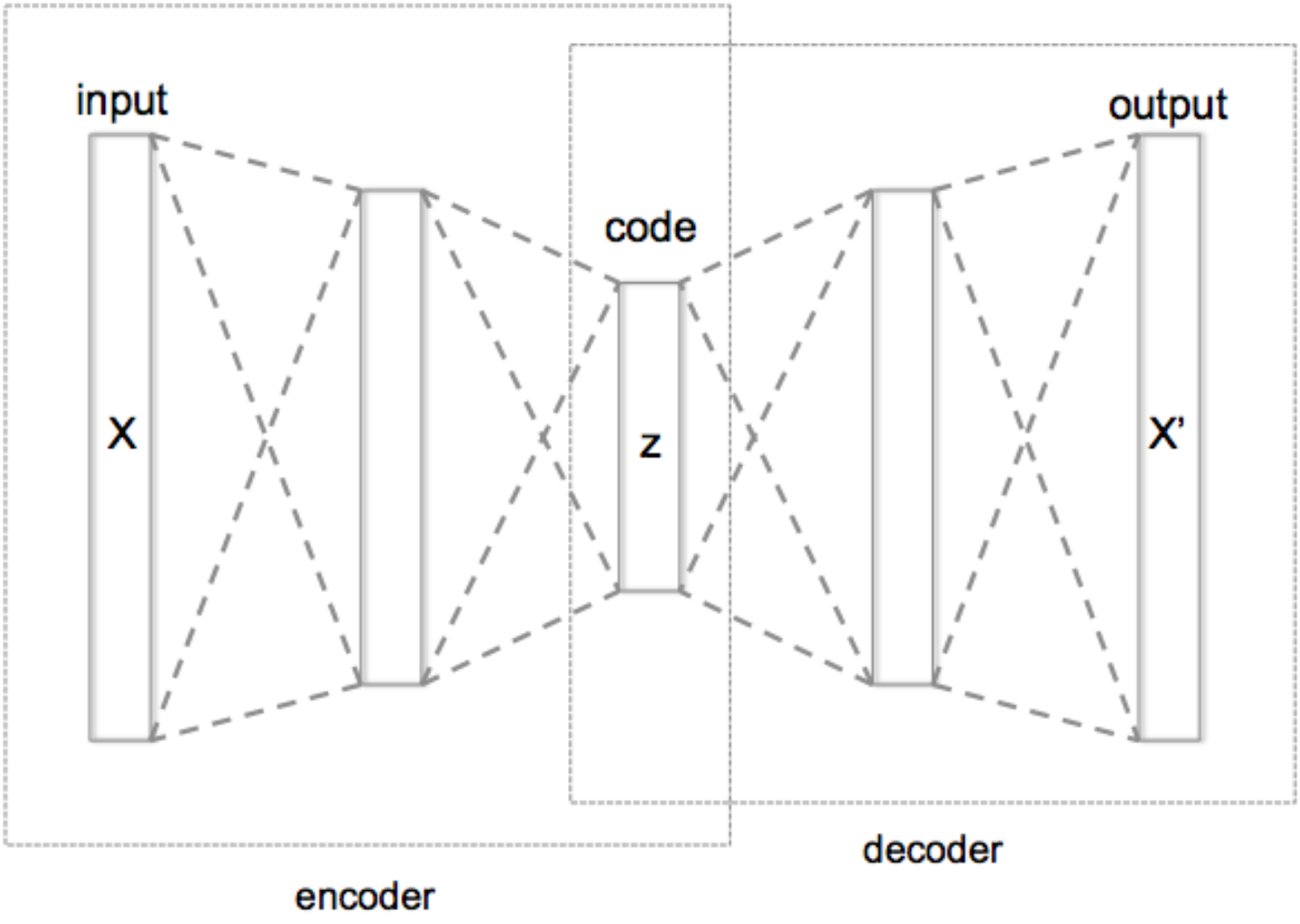
Typically, the reconstruction quality function is defined as $d(x, x') = \|x - x'\|_2^2$, which represents the squared Euclidean distance, commonly known as L2 loss.

When the reference distribution μ_{ref} is the empirical distribution of the dataset, training the autoencoder becomes a least-squares optimization problem. The objective is to minimize the average reconstruction error across all data points:

$$\min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \|x_i - D_\theta(E_\phi(x_i))\|_2^2$$

This is typically solved using gradient descent.

Schema of a basic autoencoder



Interpretation of Autoencoders

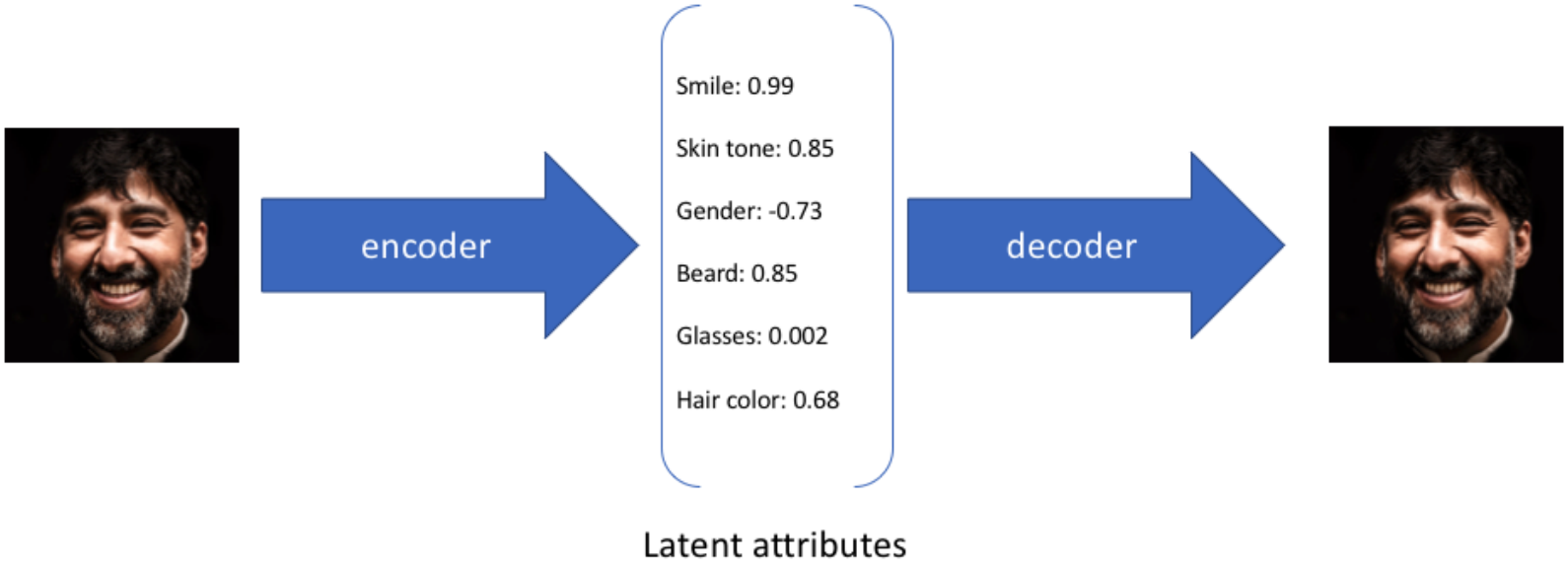
Basic Schema:

- An autoencoder consists of:
  - **Encoder**: Maps the input message to a code in the latent space.
  - **Decoder**: Reconstructs the message from the code.
- The goal is to achieve "near-perfect" reconstruction, as measured by the reconstruction quality function $d(x, x')$.

Undercomplete Autoencoder:

- The latent space $\mathcal{Z}$ has fewer dimensions than the input space $\mathcal{X}$, forcing compression of the message.
- At its best, an undercomplete autoencoder can perfectly reconstruct real messages, i.e., $D_\theta(E_\phi(x)) = x$ for x sampled from μ_{ref}.
- The decoder can then generate realistic messages by sampling from $\mathcal{Z}$.

Autoencoder compression example



Variational Autoencoder (VAE)

A Variational Autoencoder (VAE) is a generative model that extends the traditional autoencoder by framing the problem in a probabilistic setting.

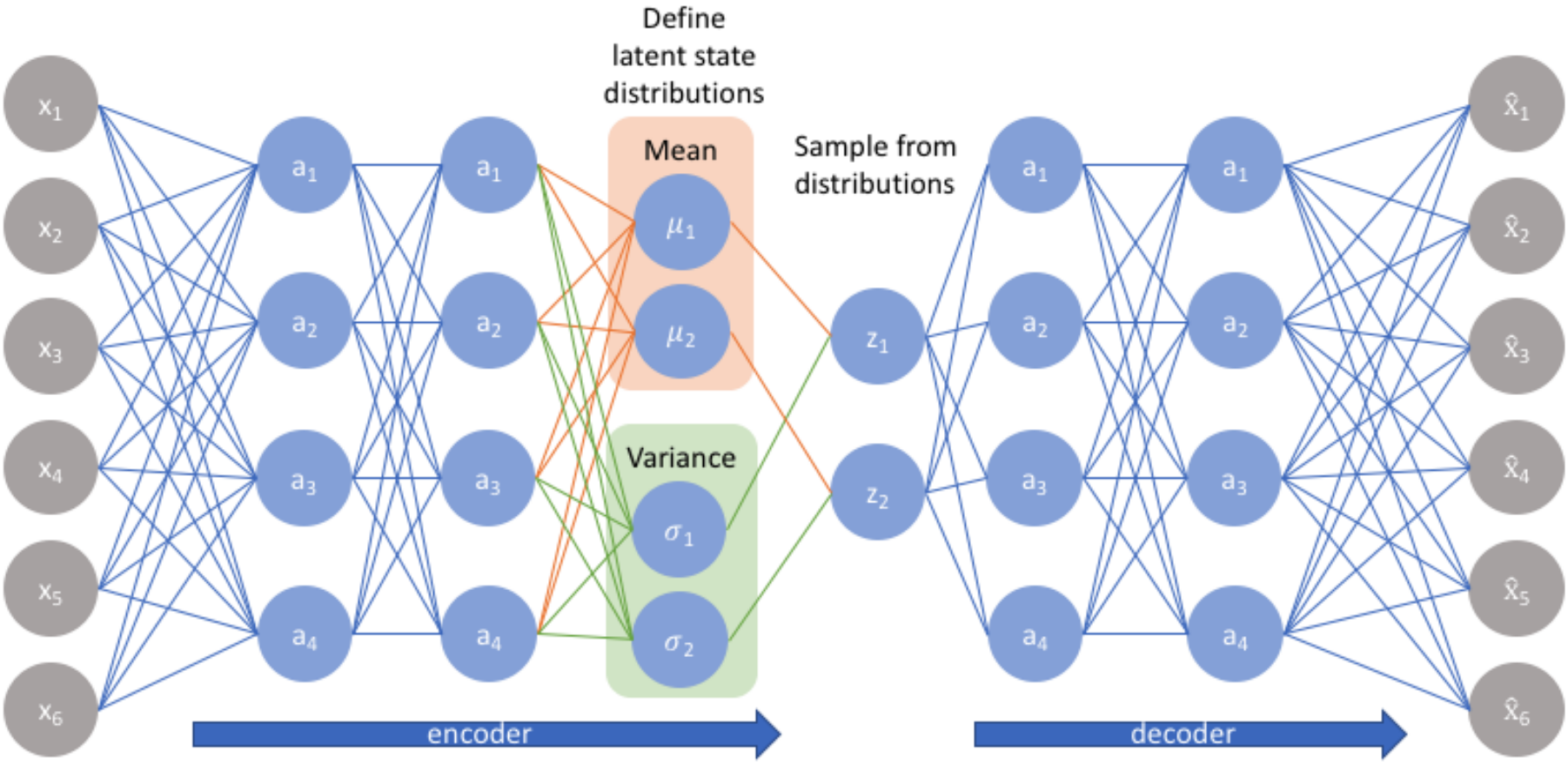
Problem Setting of a VAE:

- Traditional autoencoders map an input $x \in \mathcal{X}$ to a latent space $z \in \mathcal{Z}$ using an encoder $E_\phi(x)$ and then reconstruct it via a decoder $D_\theta(z)$.
- In VAEs, instead of encoding a single point, the encoder learns a **probability distribution** over the latent space, where the latent variable z follows a distribution, typically Gaussian.

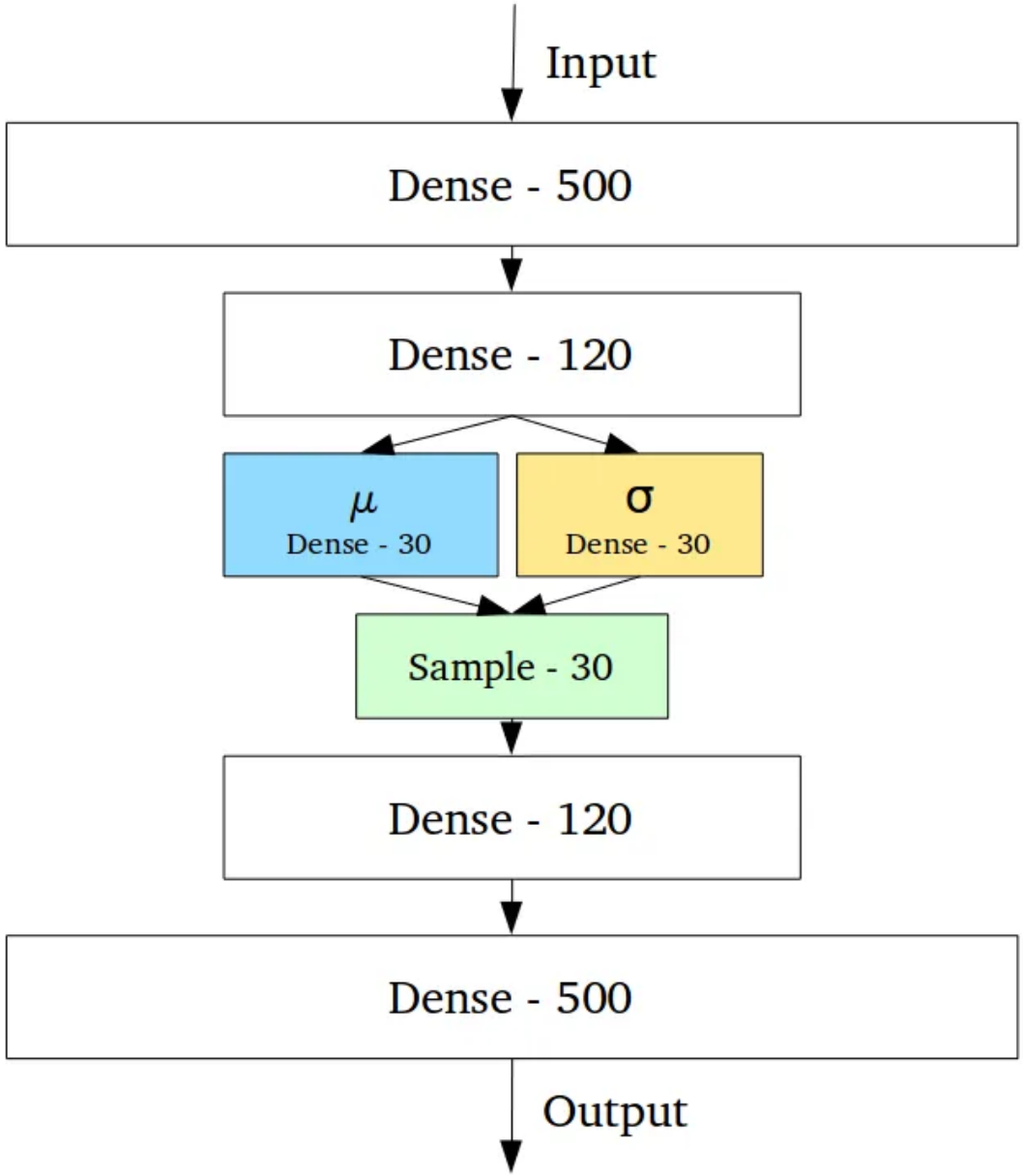
Encoder: Encoding into a Probability Distribution:

- The encoder $E_\phi(x)$ learns the parameters of a **Gaussian distribution** $q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x))$.
- The mean $\mu_\phi(x)$ and variance $\sigma^2_\phi(x)$ are outputs of the encoder's neural network, parameterizing the distribution over the latent variable z.

2d latent space Variational Autoencoder example



30d latent space VAE example



Preliminaries: Probability Density & Cumulative Distribution Functions

Probability Density Function (PDF)

- $f(x)$ describes likelihood of a continuous random variable
- Properties:
  1. $f(x) \geq 0$ for all x
  2. $\int_{-\infty}^{\infty} f(x)dx = 1$
- Probability in interval $[a, b]$:

$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

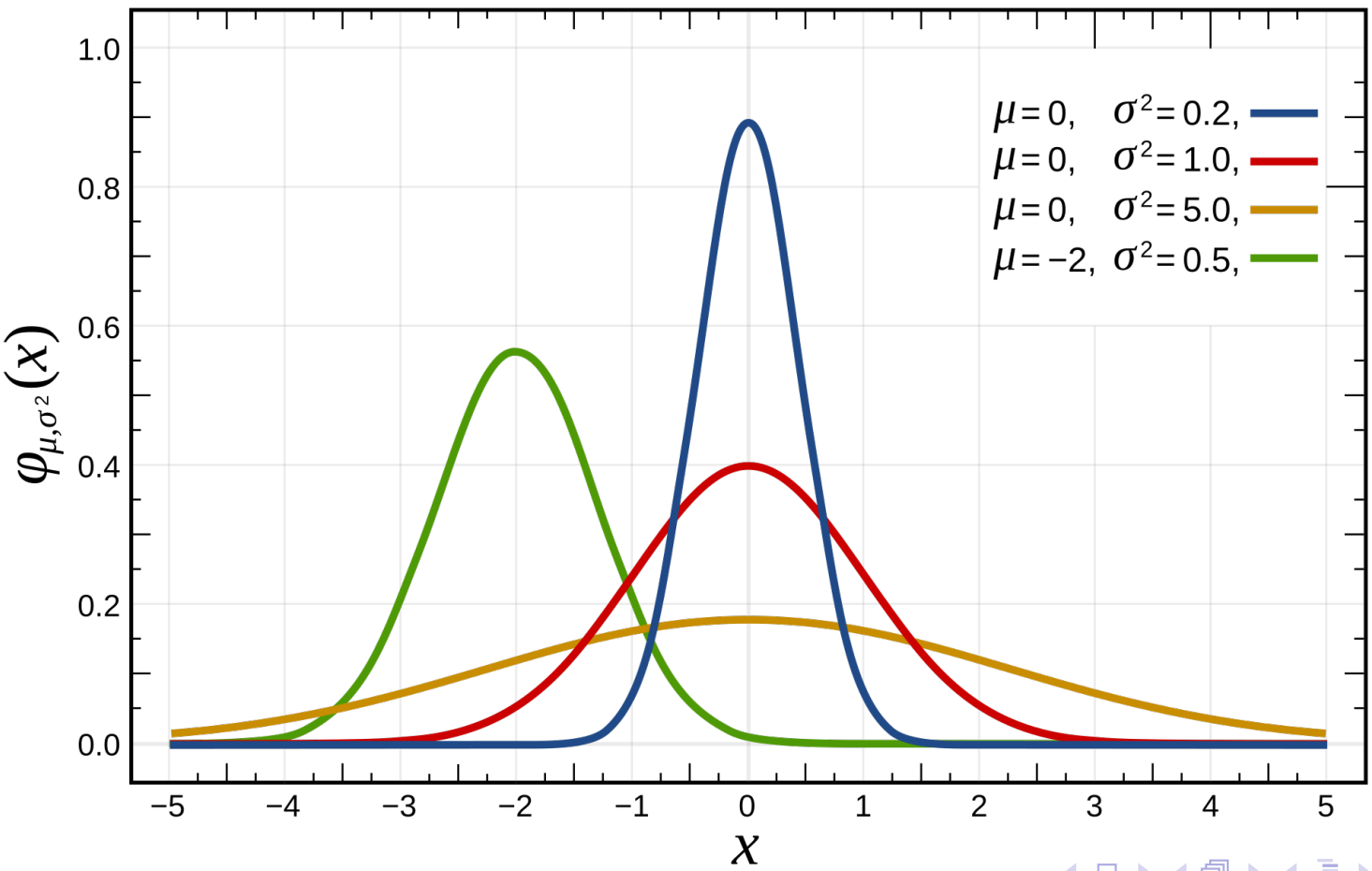
Cumulative Distribution Function (CDF)

- $F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)dt$
- Relationship: $f(x) = \frac{d}{dx}F(x)$
- Properties:
  1. $\lim_{x \to -\infty} F(x) = 0$
  2. $\lim_{x \to \infty} F(x) = 1$
  3. $F(x)$ is non-decreasing

Preliminaries: Gaussian (Normal) Distribution

The Gaussian distribution is characterized by two parameters, μ (mean), and σ (standard deviation). The PDF of a Gaussian distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Sampling and Reconstruction in VAE

Sampling from the Latent Space:

- To generate samples, we draw z from the Gaussian distribution learned by the encoder:

$$z \sim q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x))$$

- To enable backpropagation, VAEs use the **Reparameterization Trick**:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Decoder: Reconstructing from Latent Variable:

- The decoder $D_\theta(z)$ is a neural network that maps the latent variable z back to the data space:

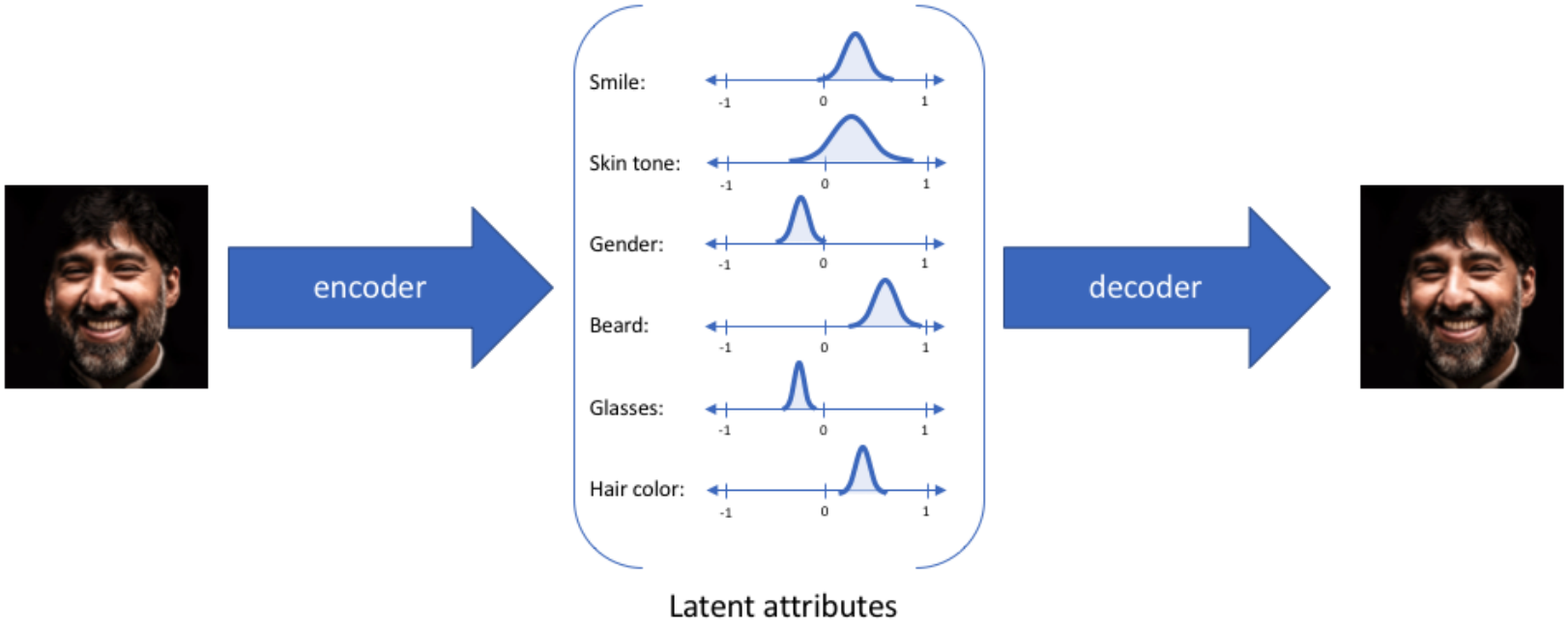
$$\hat{x} = D_\theta(z)$$

- So, we have:

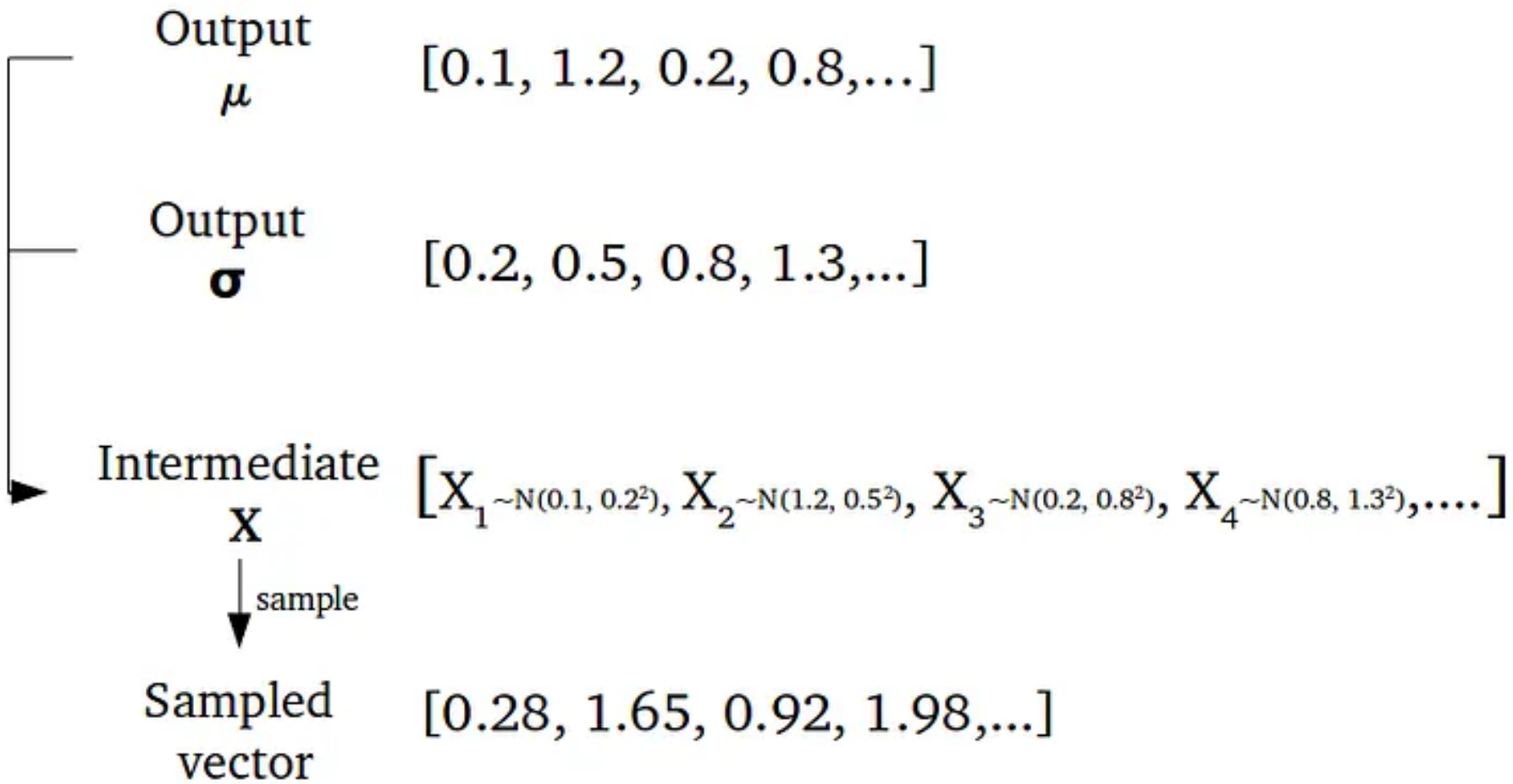
$$\hat{x} = D_\theta(z) = D_\theta(\mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon)$$

- This shows that the reconstruction $\hat{x}$ is a function of the encoder's outputs $\mu_\phi(x)$ and $\sigma_\phi(x)$, via the latent variable z.

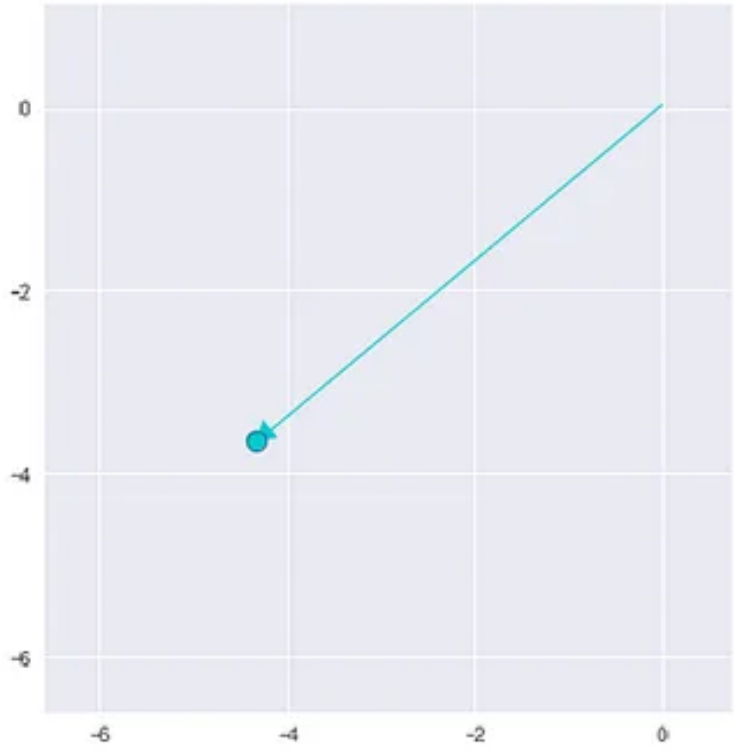
VAE compression example



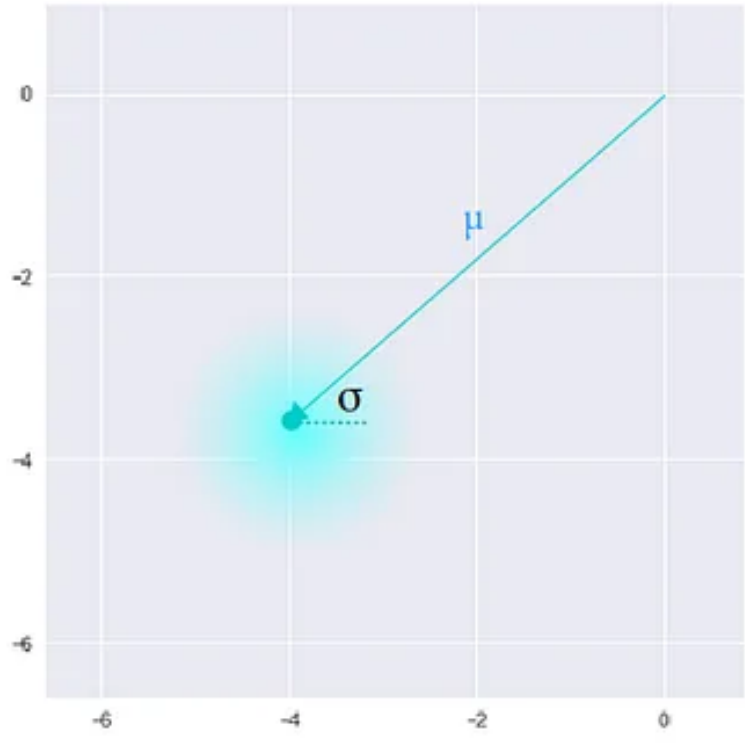
Sampling from a probabilistic VAE encoder



Comparing the difference between a deterministic encoder vs a probabilistic VAE encoder



Standard Autoencoder
(direct encoding coordinates)



Variational Autoencoder
(μ and σ initialize a probability distribution)

Reconstruction Loss in VAE

The **Reconstruction Loss** depends on:

- The difference between the input x and the reconstruction $\hat{x}$,
- The fixed variance σ^2 (usually set to 1), which scales the penalty for reconstruction errors.

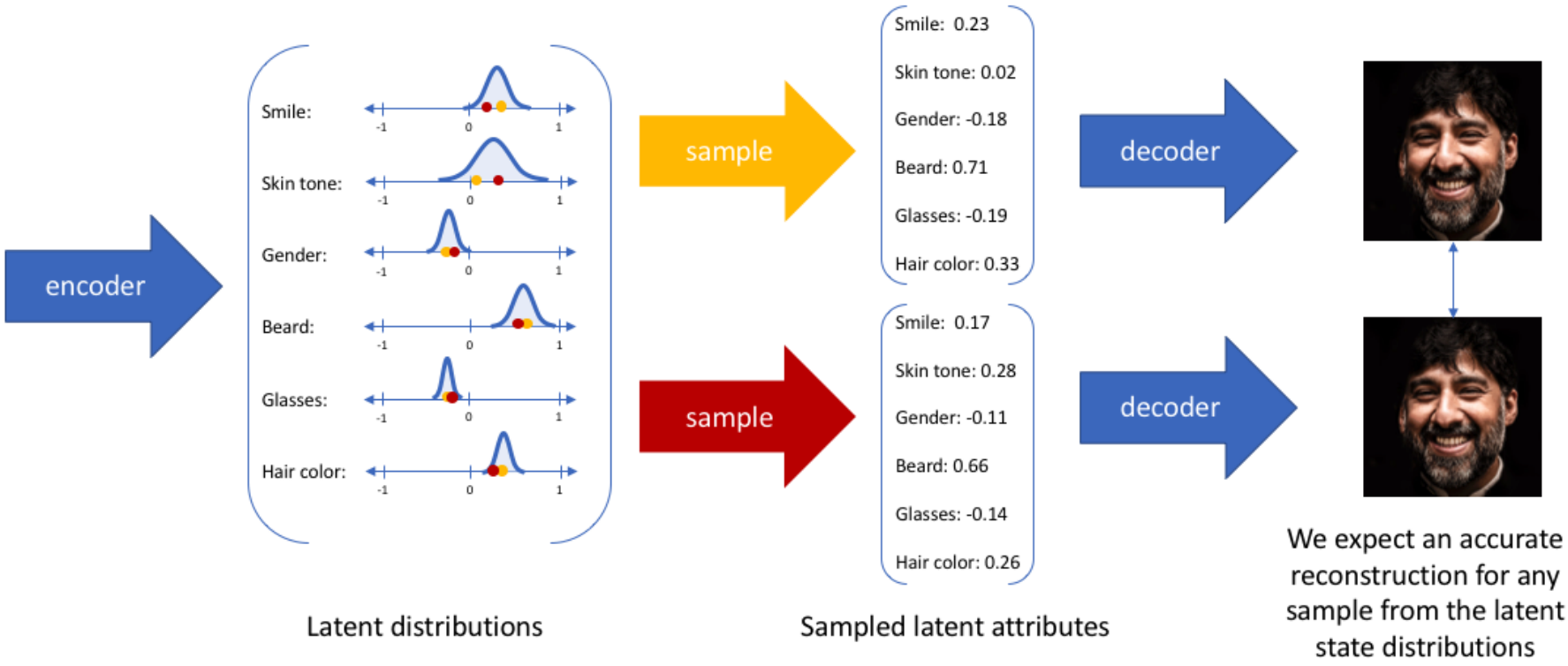
Reconstruction Loss:

$$\text{Reconstruction Loss} = \frac{1}{2\sigma^2} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$$

Where:

- $\hat{x} = D_\theta(z) = D_\theta(\mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon)$
- σ^2 is the fixed variance for reconstruction error (often set to 1),
- $\mu_\phi(x)$ and $\sigma_\phi(x)$ are the encoder's mean and variance,
- $\epsilon \sim \mathcal{N}(0, I)$ is a random sample from the standard normal distribution.

VAE Reconstruction Loss depends on sample



KL Divergence: Regularizing the Latent Space

KL Divergence encourages the latent space (i.e., the distribution over z) to follow a standard normal distribution $\mathcal{N}(0, I)$. This regularization prevents the model from overfitting and ensures a smooth, well-structured latent space.

Why KL Divergence?

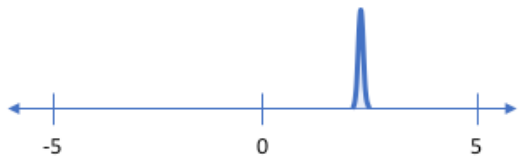
- It pushes the approximate posterior $q_\phi(z|x)$ to be close to the prior $\mathcal{N}(0, I)$.
- Regularizes the latent space to resemble a standard normal distribution.
- Helps prevent the latent space from becoming disjointed or irregular, enabling smooth sampling.

Distribution of latent variables are attracted to the unit normal distribution

Penalizing reconstruction loss encourages the distribution to describe the input

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

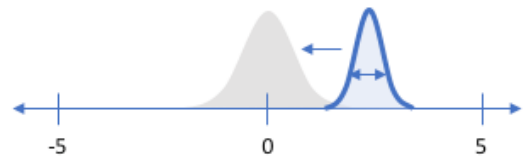
Without regularization, our network can "cheat" by learning narrow distributions



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

KL Divergence: Mathematical Formulation

The **KL Divergence** measures the difference between two distributions: the learned distribution $q_\phi(z|x)$ and the prior $\mathcal{N}(0, I)$.

$$\text{KL Divergence} = D_{\text{KL}}\left(q_\phi(z|x)||\mathcal{N}(0, I)\right)$$

For a Gaussian $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x))$ and prior $\mathcal{N}(0, I)$, the KL Divergence is:

$$D_{\text{KL}}(q_\phi(z|x)||p(z)) = \frac{1}{2}\sum_{j=1}^{n}\left(1 + \log(\sigma^2_{\phi,j}) - \mu^2_{\phi,j} - \sigma^2_{\phi,j}\right)$$

Where $\mu_{\phi,j}$ and $\sigma^2_{\phi,j}$ are the encoder's mean and variance for the j-th dimension of z.

KL Divergence: Intuition

Goal: Ensure that the approximate posterior $q_\phi(z|x)$ is close to the prior $\mathcal{N}(0, I)$, organizing the latent space.

Why it Matters:

- Without KL Divergence, the latent space could become disorganized and irregular.
- It helps maintain a continuous latent space where nearby points generate similar outputs.
- In image generation, it ensures sampling from the latent space produces realistic images.

Regularization through KL Divergence leads to a smooth latent space and prevents overfitting, improving the quality of generated samples.

VAE Objective: Reconstruction Loss + KL Divergence

The total objective function for training a **Variational Autoencoder (VAE)** combines the **Reconstruction Loss** and the **KL Divergence**:

$$\mathcal{L}_{\text{VAE}} = \frac{1}{2\sigma^2}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \frac{1}{2}\sum_{j=1}^{n}\left(1 + \log(\sigma_{\phi,j}^2) - \mu_{\phi,j}^2 - \sigma_{\phi,j}^2\right)$$

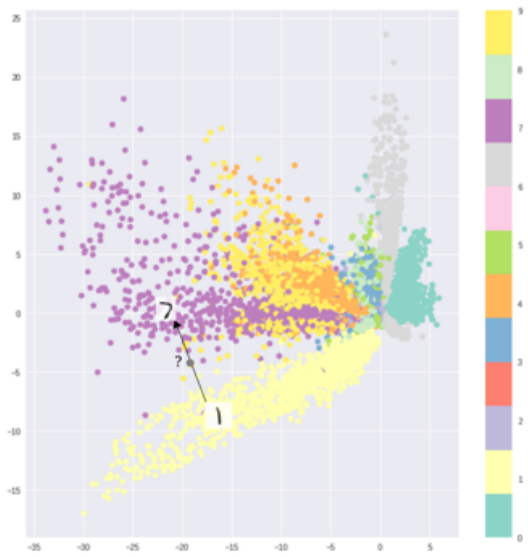
Where:

- $\hat{x} = D_\theta(z) = D_\theta(\mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon)$ is the reconstruction.
- $\frac{1}{2\sigma^2}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$ is the Reconstruction Loss.
- $\frac{1}{2}\sum_{j=1}^{n}\left(1 + \log(\sigma_{\phi,j}^2) - \mu_{\phi,j}^2 - \sigma_{\phi,j}^2\right)$ is the KL Divergence.
- σ^2 is a hyperparameter (often set to 1) that controls the trade-off between the Reconstruction Loss and KL Divergence.

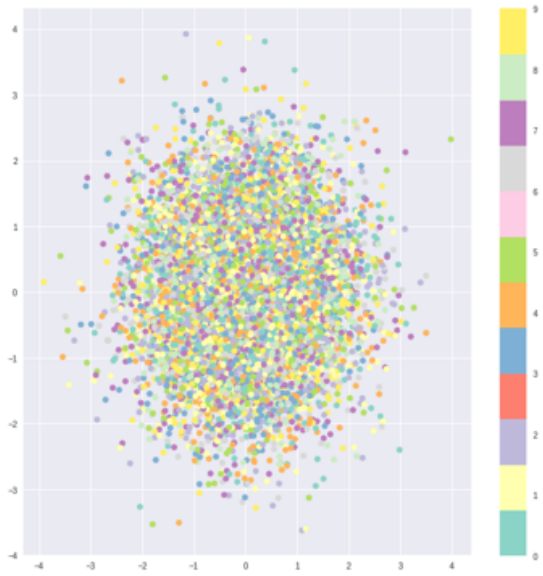
The VAE Neural Network is trained using back-propagation/gradient descent to minimize this total loss, balancing the reconstruction quality and latent space regularization.

Balancing Reconstruction Loss and KL Divergence (MNIST database)

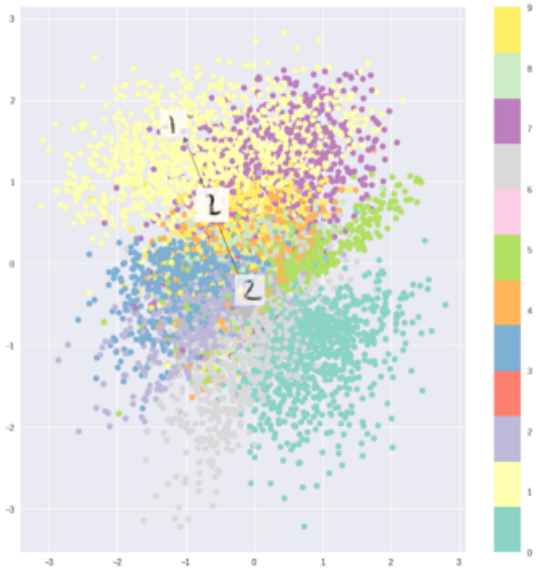
Only reconstruction loss



Only KL divergence



Combination



Learning a Smooth and Continuous Latent Space

In a VAE, the latent space is encoded probabilistically as a **Gaussian distribution** (mean μ, variance σ^2).

Why this is useful:

- Creates a smooth and continuous latent space.
- Small changes in z lead to gradual and meaningful changes in the output.
- Avoids disjointed or discontinuous latent spaces that are hard to explore for generating new data.

Handling Uncertainty and Variability in Data

Encoding x into a distribution (instead of a point) allows the VAE to capture the inherent variability in the data.

Why this is useful:

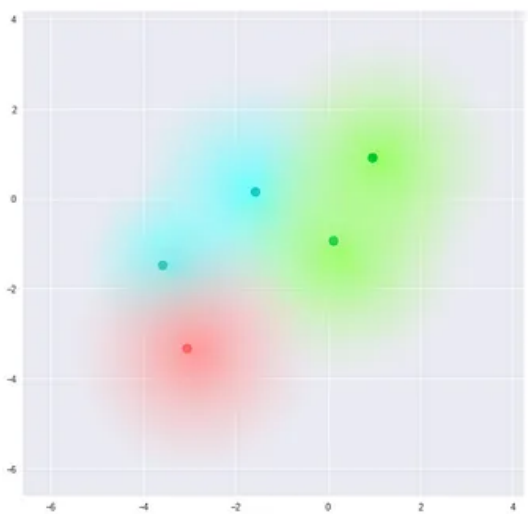
- Captures the range of possible representations of an input.
- Represents uncertainty in the data (e.g., slight variations in handwritten digits).
- Produces robust reconstructions and generates data that reflects the natural variation in the dataset.

Generative Modeling: Sampling from the Latent Space

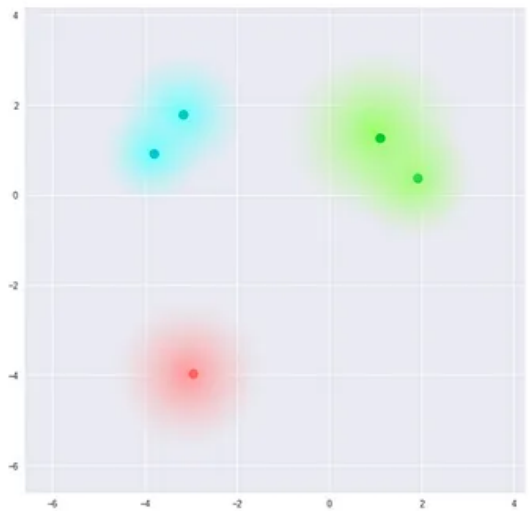
The probabilistic latent space allows us to **sample** from the latent distribution z to generate new data.

Why this is useful:

- After training, the latent space follows a standard normal distribution $\mathcal{N}(0, I)$.
- We can sample new z values and decode them to generate new, realistic data points.
- Deterministic encodings wouldn't provide a clear way to sample new data from the latent space.



What we require



What we may inadvertently end up with

Regularization via KL Divergence

The VAE uses the **KL Divergence** to regularize the latent space by comparing the learned distribution $q_\phi(z|x)$ with the prior $\mathcal{N}(0, I)$.

Why this is useful:

- Keeps the latent space close to a standard normal distribution.
- Prevents overfitting by encouraging a well-structured, compact latent space.
- Avoids over-representation or sparsity in certain regions of the latent space.

Improved Generalization

Probabilistic encoding allows the VAE to focus on the general structure of the data, rather than memorizing specific details.

Why this is useful:

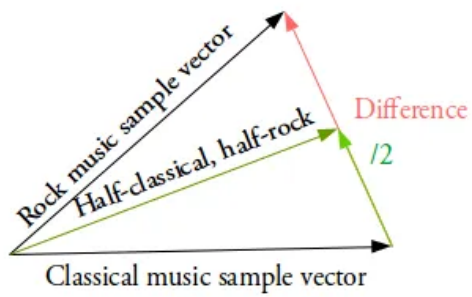
- The variance term allows the model to learn common patterns across the data.
- Results in better generalization to unseen data.
- Makes the model more robust to slight variations in the input data.

Interpolation in the Latent Space

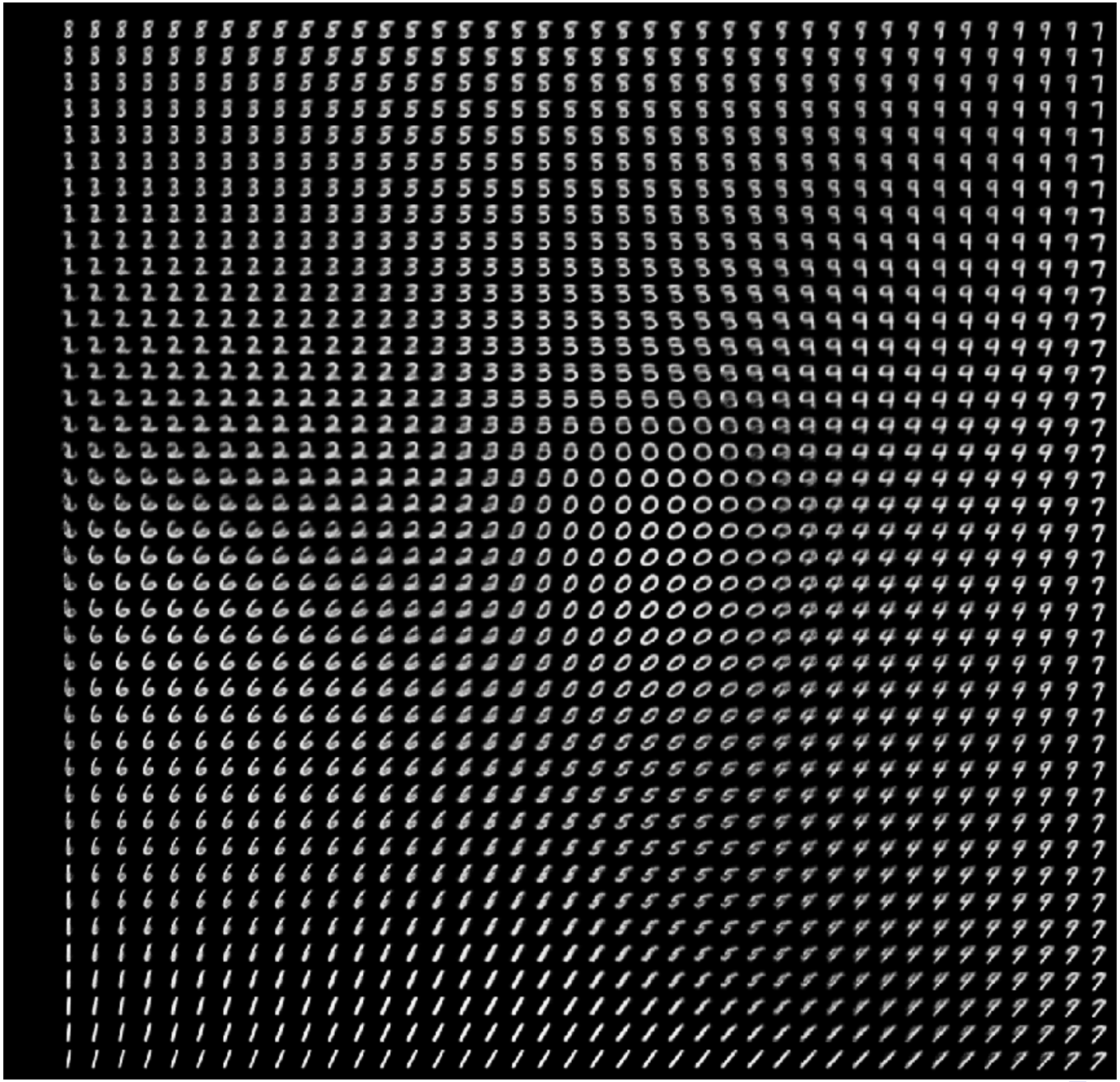
Probabilistic encoding enables smooth **interpolation** between different data points by moving through the latent space.

Why this is useful:

- Allows for generating intermediate data points between known samples.
- For example, interpolate between images of a '3' and a '5' by sampling along a path in the latent space.
- Deterministic encodings wouldn't allow for such smooth transitions.



VAE trained on MNIST database



Summary: Advantages of Probabilistic Encoding in VAEs

Encoding inputs probabilistically in VAEs provides several advantages:

- Smooth, continuous, and well-structured latent space.
- Captures uncertainty and variability in the data.
- Facilitates easy sampling of new data points.
- Improves generalization and allows for latent space interpolation.
- Regularization through KL Divergence ensures a well-behaved latent space.

This makes VAEs powerful tools for both data compression and generation.

Reparametrization Trick in VAEs

Problem:

- Need to sample from $q_\phi(z|x)$ & compute gradients
- Random sampling is not differentiable
- Cannot backpropagate through randomness

Solution: Reparametrize the random variable!

$$z \sim \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x))$$

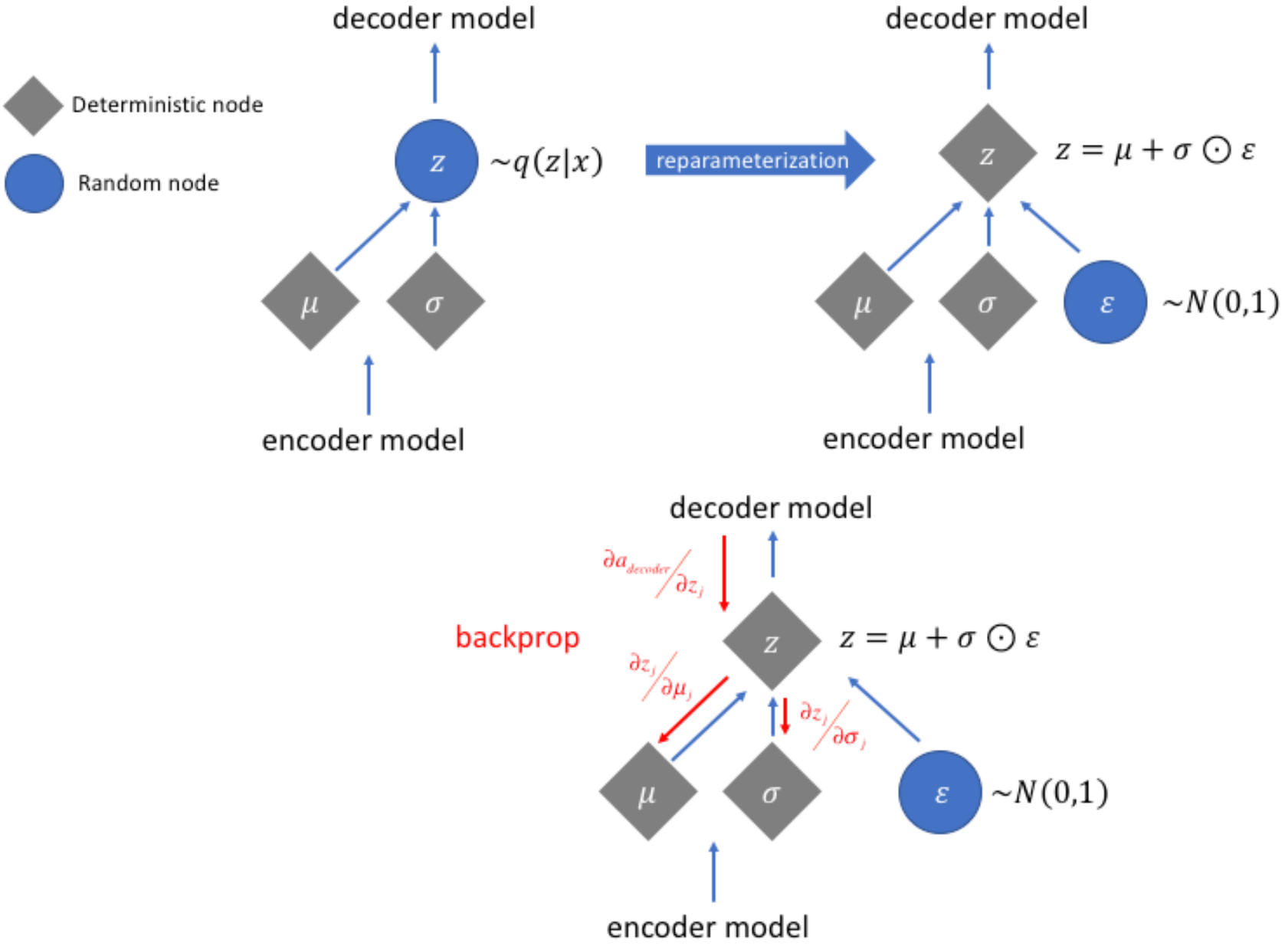
$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$$

where $\epsilon \sim \mathcal{N}(0, I)$

Why It Works:

- Isolates the randomness to a input noise ϵ, independent of the encoder weights ϕ.
- The mean μ_ϕ and variance σ_ϕ depend deterministically on ϕ.
- Enables gradients through μ_ϕ, σ_ϕ

Reparametrization Trick



Theorem Statement

Theorem: Let $\varepsilon \sim \mathcal{N}(0, 1)$ be a normal distribution with zero mean and unit variance, and let μ and σ be constants (with $\sigma > 0$). The random variable X defined by:

$$X = \mu + \sigma\varepsilon$$

is normally distributed with mean μ and variance σ^2.

Objective: Prove that $X \sim \mathcal{N}(\mu, \sigma^2)$ by deriving the PDF of X from first principles.

Step 1: CDF of X

The CDF of X is defined as:

$$F_X(x) = P(X \leq x)$$

Substitute $X = \mu + \sigma\varepsilon$:

$$F_X(x) = P(\mu + \sigma\varepsilon \leq x)$$

Rearranging:

$$F_X(x) = P\left(\varepsilon \leq \frac{x - \mu}{\sigma}\right)$$

Since $\varepsilon \sim \mathcal{N}(0, 1)$, we can express this as:

$$F_X(x) = \Phi\left(\frac{x - \mu}{\sigma}\right)$$

where $\Phi(z)$ is the CDF of the standard normal distribution.

Step 2: Differentiate the CDF to get the PDF

To find the PDF $f_X(x)$, we differentiate the CDF:

$$f_X(x) = \frac{d}{dx} F_X(x) = \frac{d}{dx} \Phi\left(\frac{x-\mu}{\sigma}\right)$$

By applying the chain rule:

$$f_X(x) = \phi\left(\frac{x-\mu}{\sigma}\right) \cdot \frac{1}{\sigma}$$

where $\phi(z)$ is the PDF of the standard normal distribution:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

Step 3: Final PDF of X

Substituting back $z = \frac{x-\mu}{\sigma}$:

$$f_X(x) = \frac{1}{\sigma} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Simplifying the exponent:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

This is the PDF of a normal distribution with mean μ and variance σ^2. Thus, $X \sim \mathcal{N}(\mu, \sigma^2)$.

Appendix: Theorem

Theorem

Let X be a continuous random variable with cumulative distribution function $F_X(x)$ and probability density function $f_X(x)$.

Then, for all x where $f_X(x)$ is defined, we have:

$$f_X(x) = \frac{d}{dx} F_X(x).$$

To prove this theorem, we will use the fact that the probability density function (PDF) $f_X(x)$ satisfies:

$$f_X(x) = \lim_{\Delta x \to 0} \frac{P(x \leq X \leq x + \Delta x)}{\Delta x}.$$

Proof - Part 1

By definition, the cumulative distribution function $F_X(x)$ is given by:

$$F_X(x) = P(X \leq x).$$

We can express $P(x \leq X \leq x + \Delta x)$ as a function of the CDF:

$$\begin{aligned} P(x < X \leq x + \Delta x) &= P(X \leq x + \Delta x) - P(X \leq x) \\ &= F_X(x + \Delta x) - F_X(x). \end{aligned}$$

Explanation:

- The probability $P(x < X \leq x + \Delta x)$ represents the chance that X falls within the interval $(x, x + \Delta x]$.
- We express this probability as the difference between the cumulative probabilities up to $x + \Delta x$ and up to x.

Proof - Part 2

Recall that the PDF satisfies

$$P(a \leq X \leq b) = \int_a^b f_X(x)dx$$

for any interval $[a, b]$. Therefore the probability density function $f_X(x)$ satisfies:

$$P(x \leq X \leq x + \Delta x) \approx f_X(x)\Delta x \quad \text{for small } \Delta x.$$

Therefore, we have:

$$f_X(x) = \lim_{\Delta x \to 0} \frac{P(x \leq X \leq x + \Delta x)}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{F_X(x + \Delta x) - F_X(x)}{\Delta x} = \frac{d}{dx} F_X(x).$$

Thus, $f_X(x) = F'_X(x)$.