Auto Encoders

What PCA wished it was

Introduction to Machine Learning, Spring 2026

Recall: Motivation for PCA

Recall: Motivation for PCA

Given data $\{x_1, \cdots, x_n\}, x_i \in \mathbb{R}^d$, *can we represent it in a lower dimension?*

Recall: Motivation for PCA

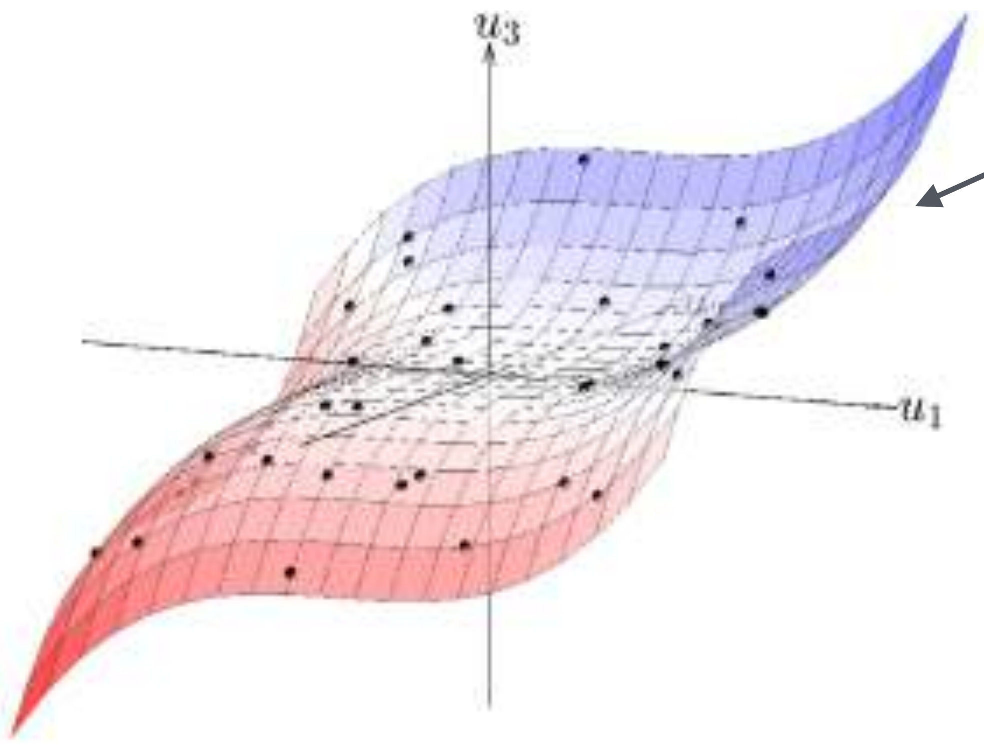
Given data $\{x_1, \cdots, x_n\}, x_i \in \mathbb{R}^d$, *can we represent it in a lower dimension?*

Lower dimension $\rightarrow$ computational efficiency, lower redundancy (list goes on)

What can go wrong?

What can go wrong?

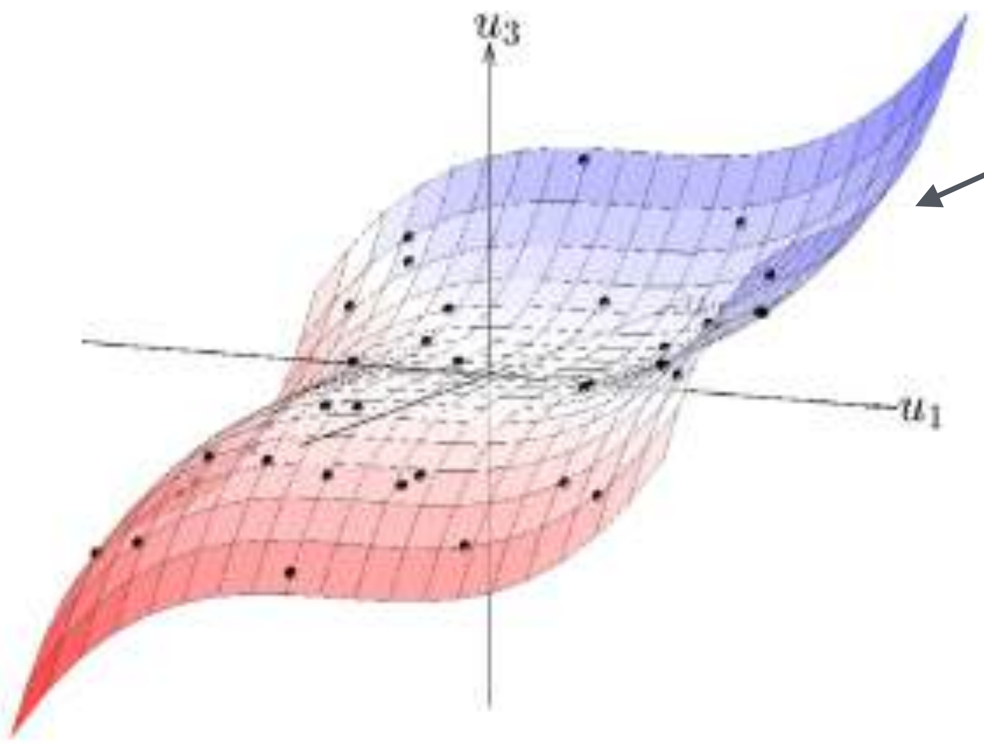
Assumes that data lies on a smaller dimensional *linear* manifold



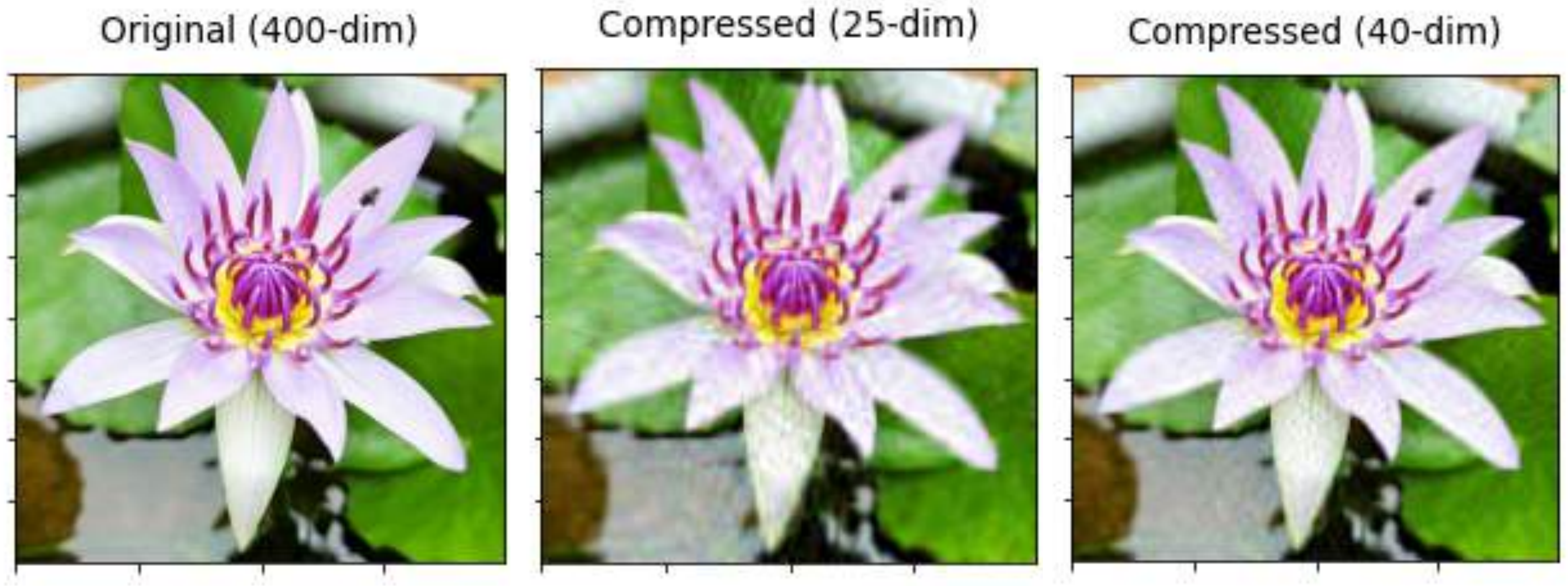
Not linear

What can go wrong?

Assumes that data lies on a smaller dimensional *linear* manifold



Not linear



It is lossy - the lower dimensional representations cannot be used to get the original data back.

Can we *learn*?

Can we *learn*?

We have build many tools in the course that allow us to use data and *learn*, why not use them?

Can we *learn*?

We have build many tools in the course that allow us to use data and *learn*, why not use them?

Training needs input and output pairs. We only have some data points 

Can we *learn*?

We have build many tools in the course that allow us to use data and *learn*, why not use them?

Training needs input and output pairs. We only have some data points 

We need a clever way to convert this unsupervised learning task into a supervised learning task

A random problem

A random problem

Let's consider a completely different problem

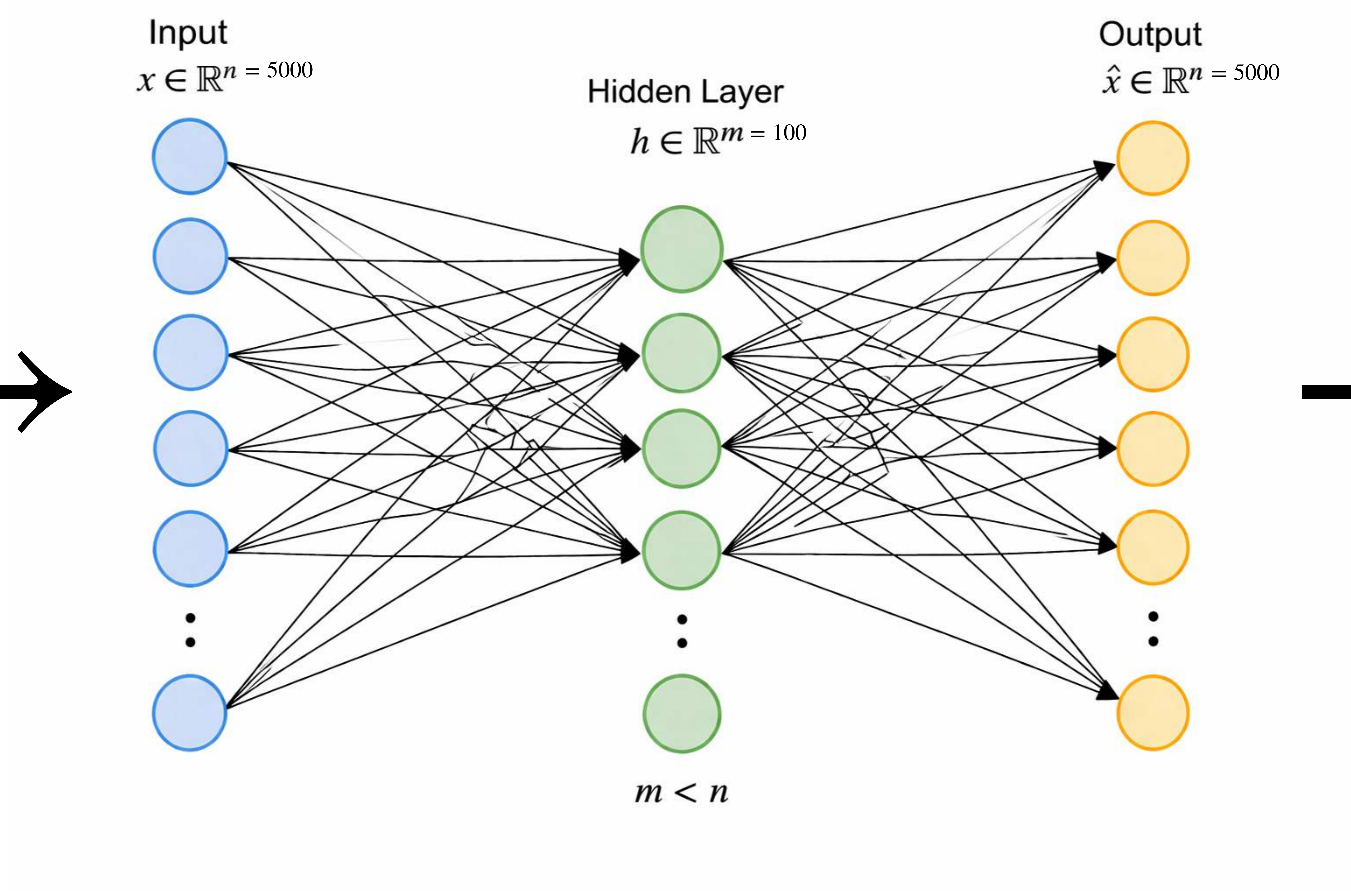
A random problem

Let's consider a completely different problem

Suppose I train a network, such that given x, it outputs.. x

A random problem

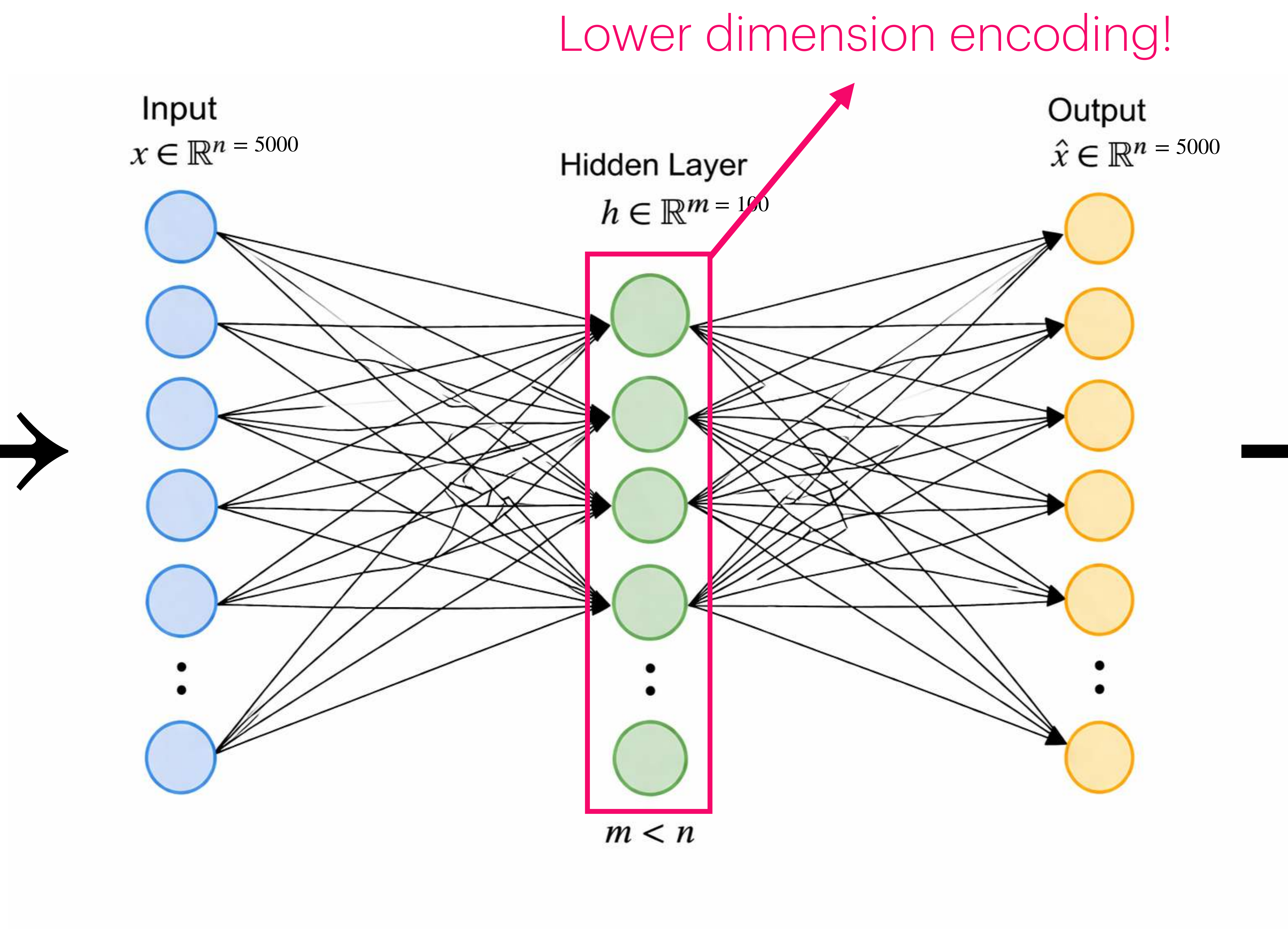
$x \rightarrow$



$\rightarrow x$

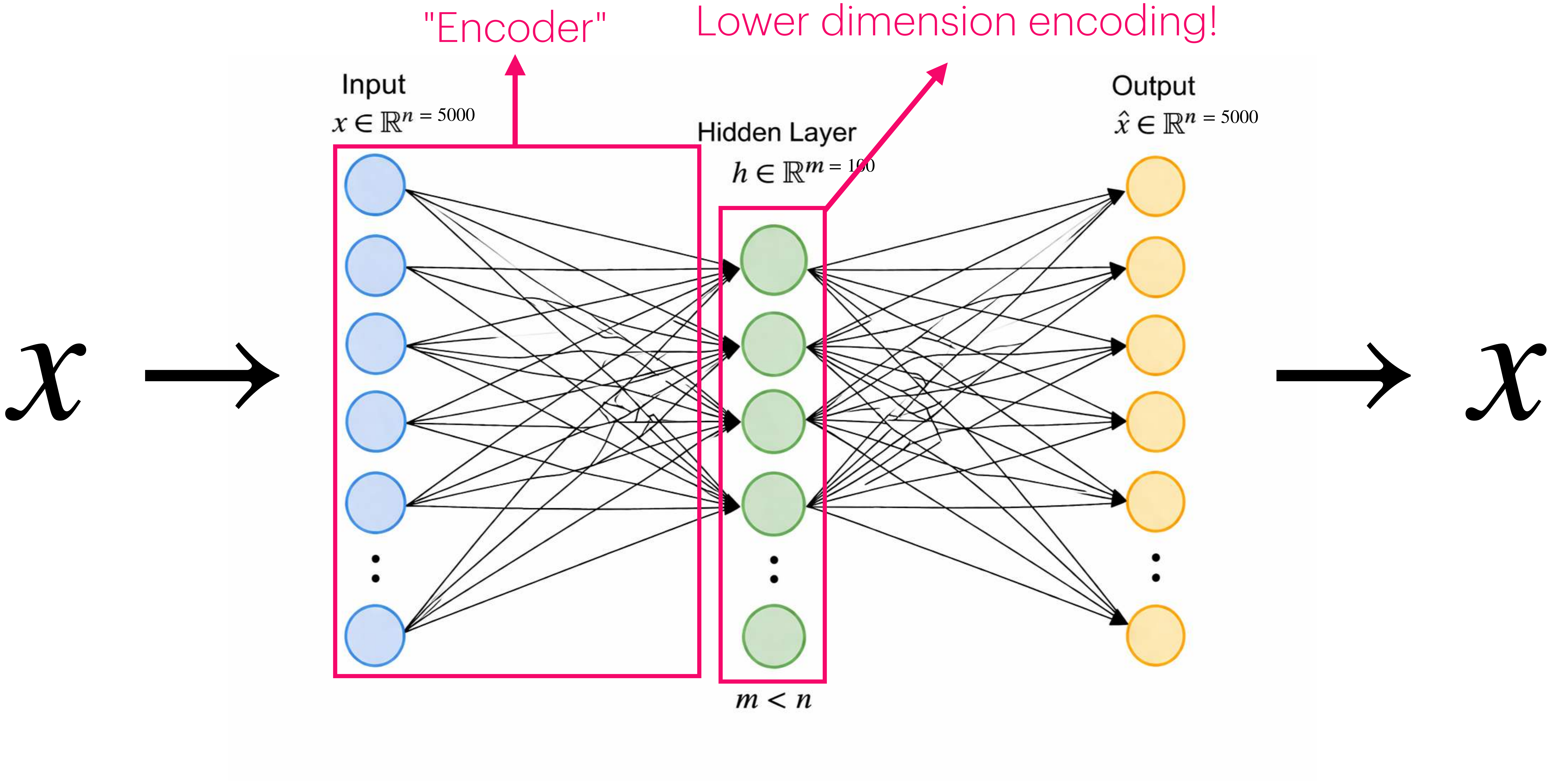
A random problem

$x \rightarrow$

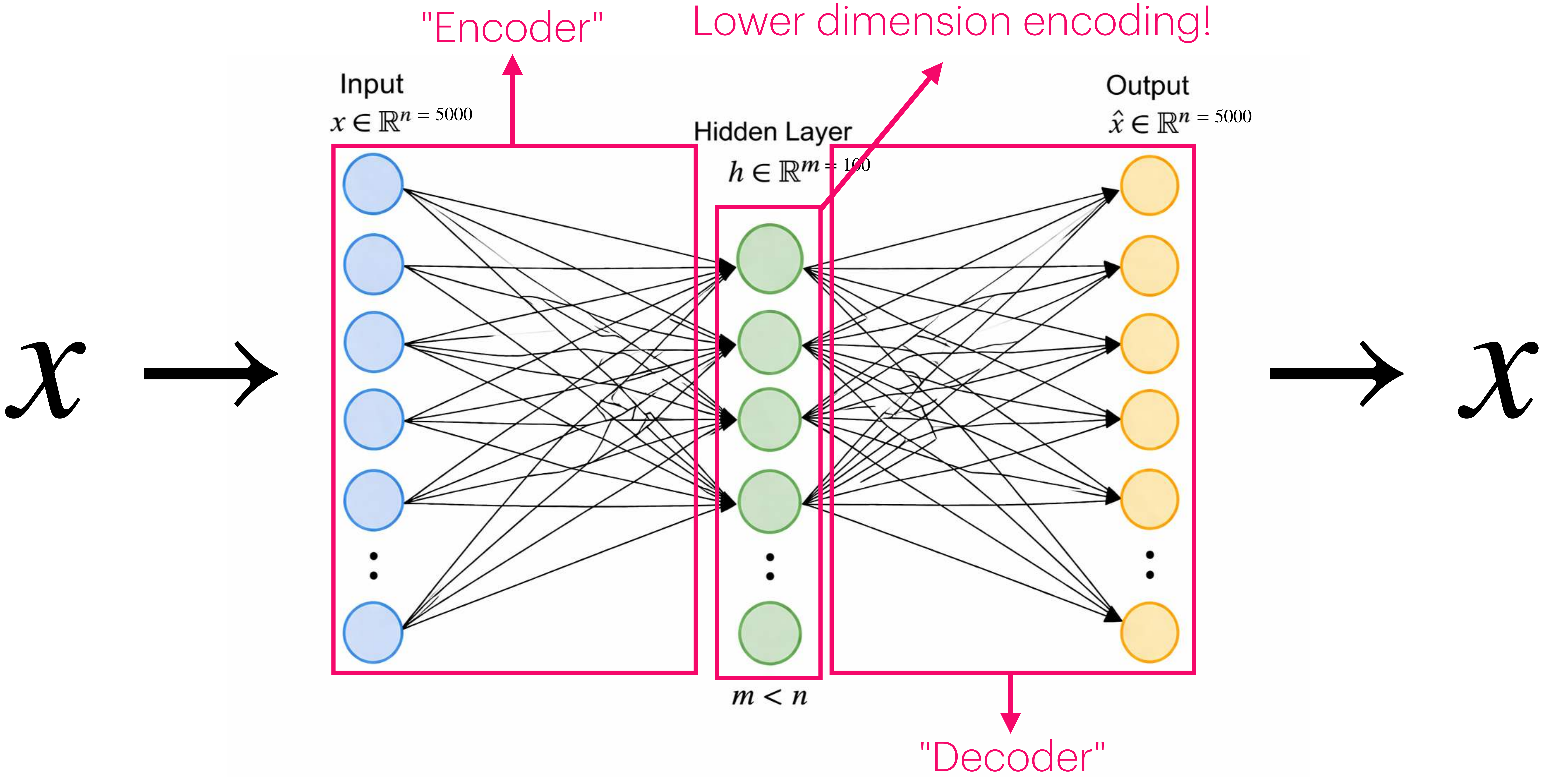


$\rightarrow x$

A random problem

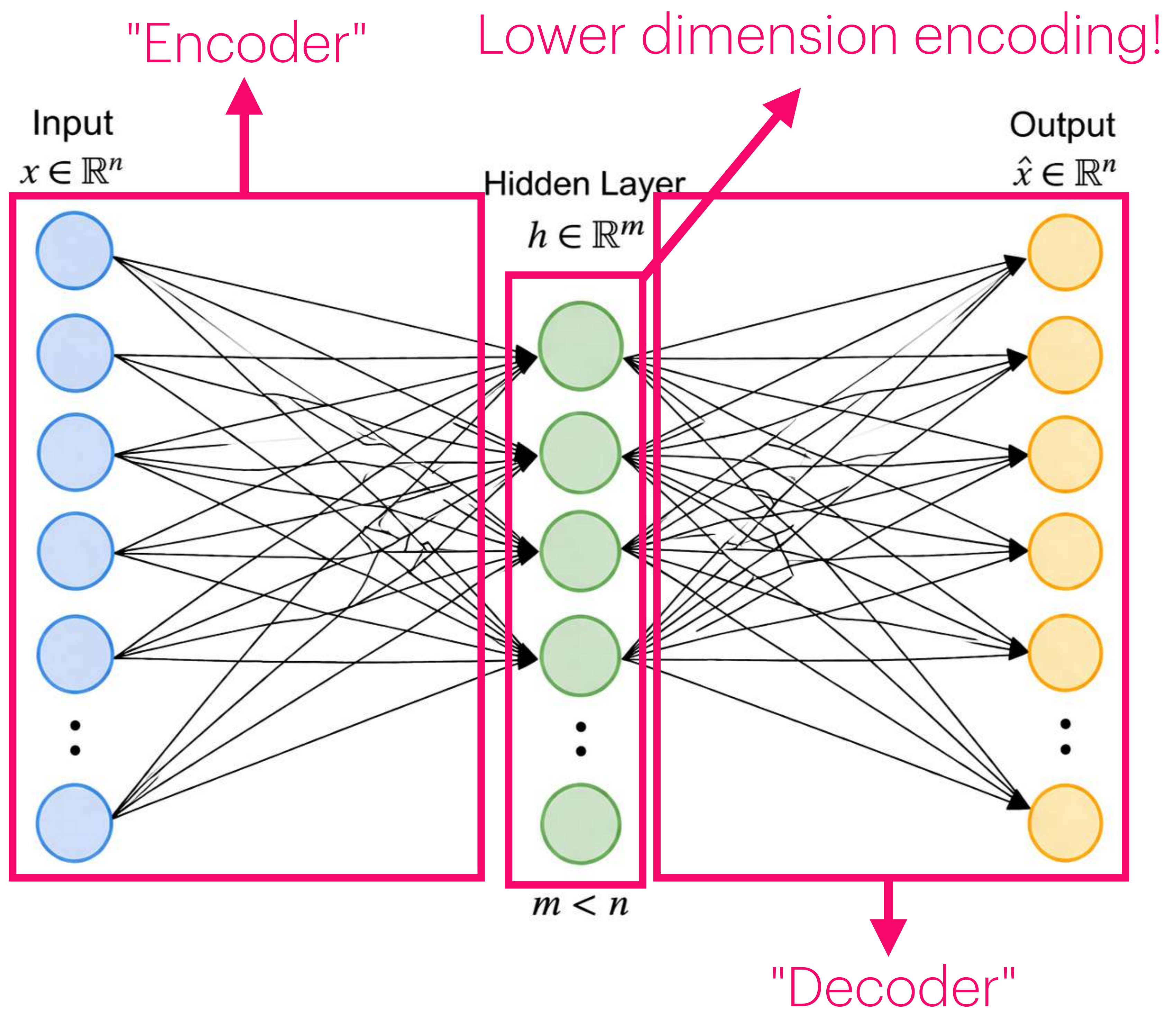


A random problem

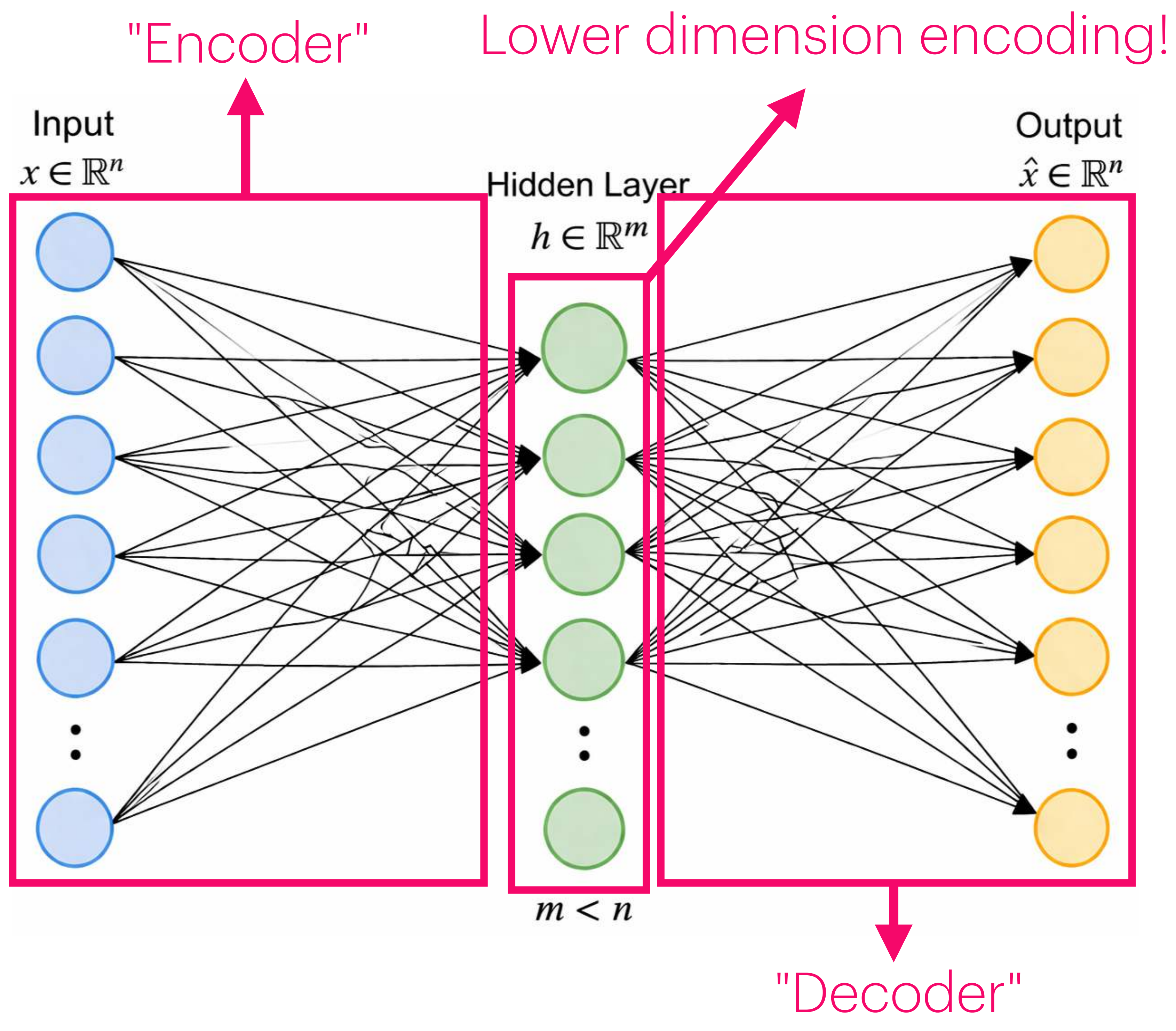


Simplifying the problem

Simplifying the problem

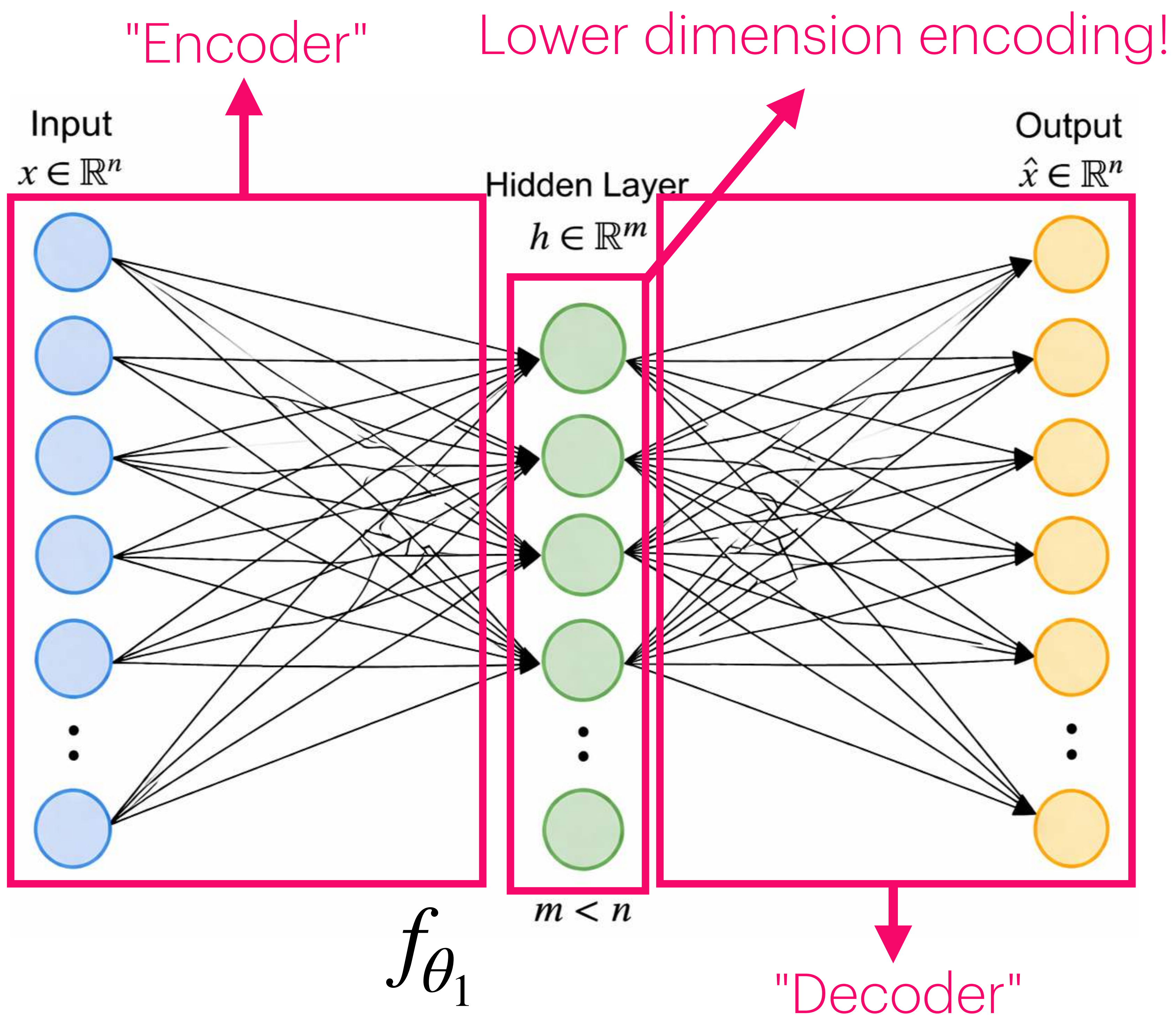


Simplifying the problem



$\rightarrow$

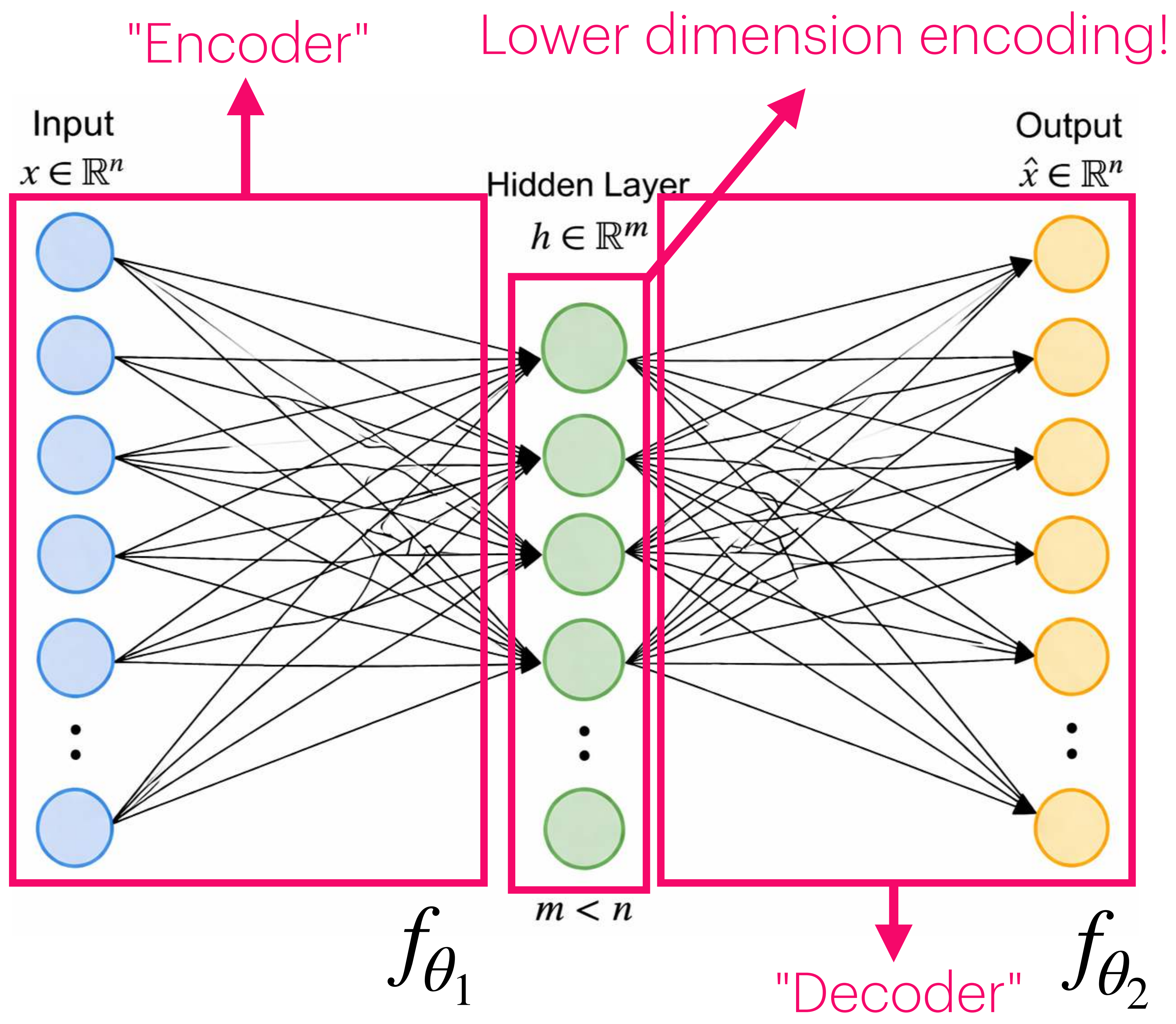
Simplifying the problem



$\rightarrow$

$f_{\theta_1} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ – the encoder

Simplifying the problem

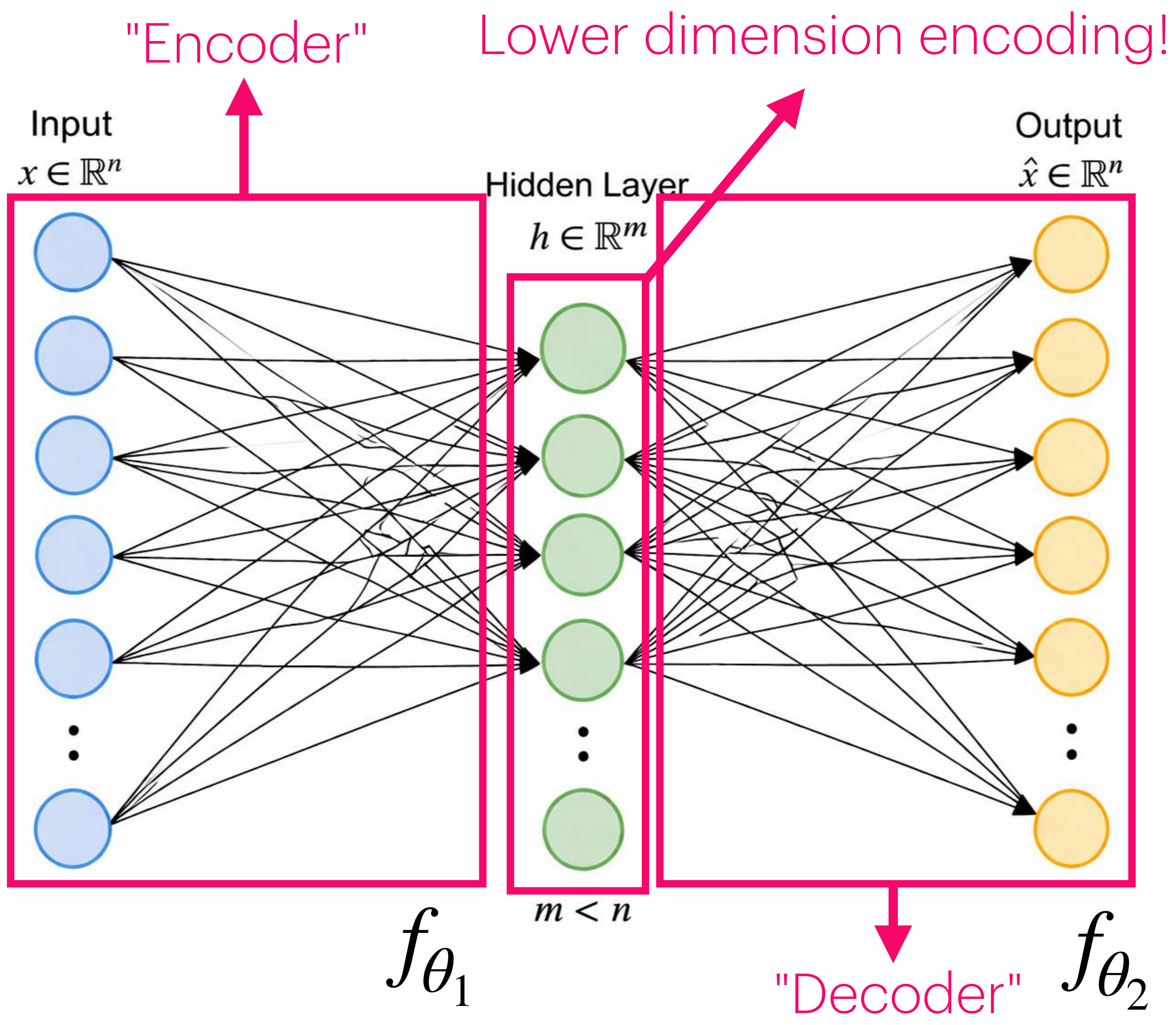


$\rightarrow$

$f_{\theta_1} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ – the encoder

$f_{\theta_2} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ – the decoder

Simplifying the problem

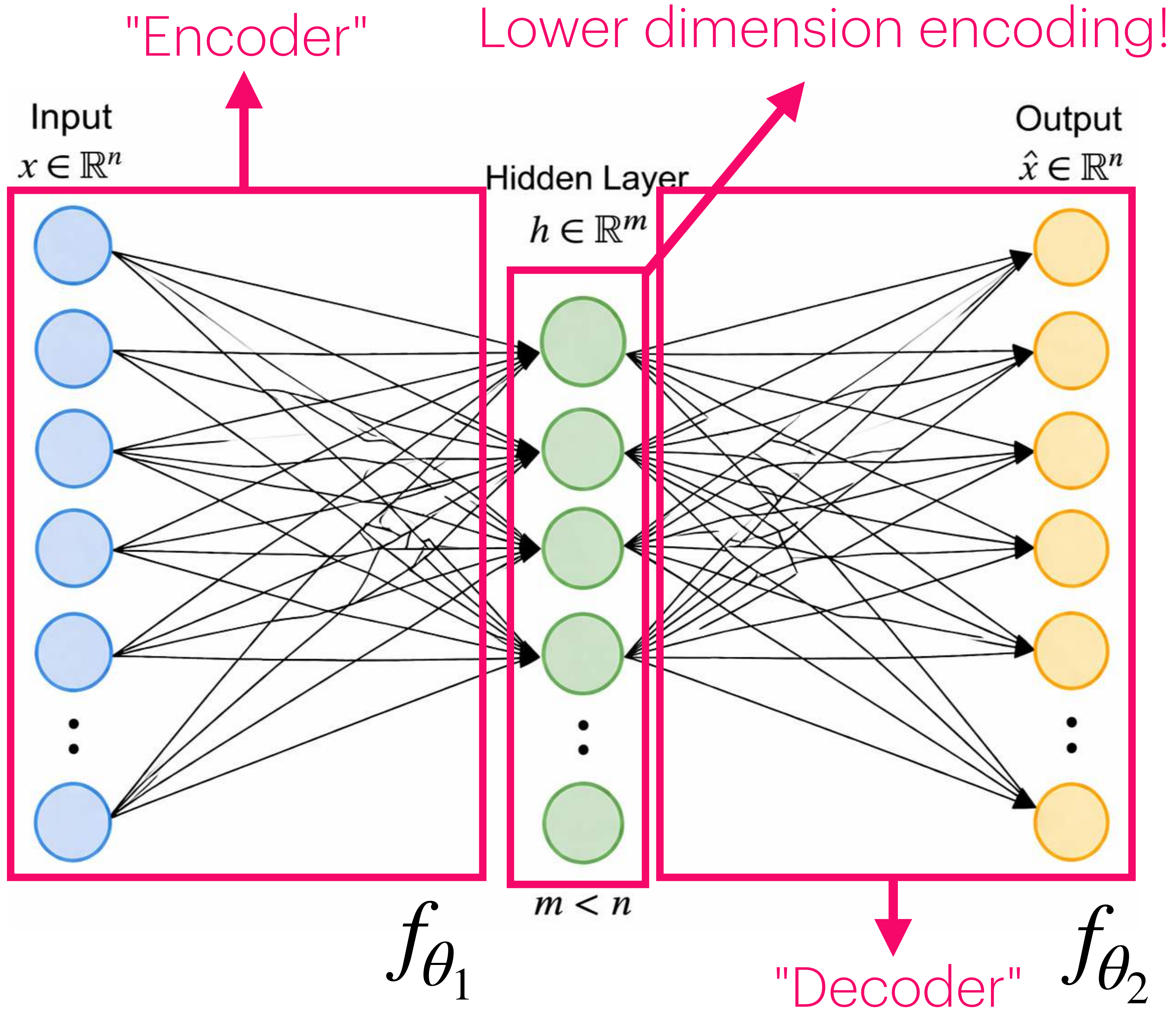


$f_{\theta_1} : \mathbb{R}^n \to \mathbb{R}^m$ – the encoder

$f_{\theta_2} : \mathbb{R}^m \to \mathbb{R}^n$ – the decoder

What do f_{θ_1} and f_{θ_2} even do?

Simplifying the problem



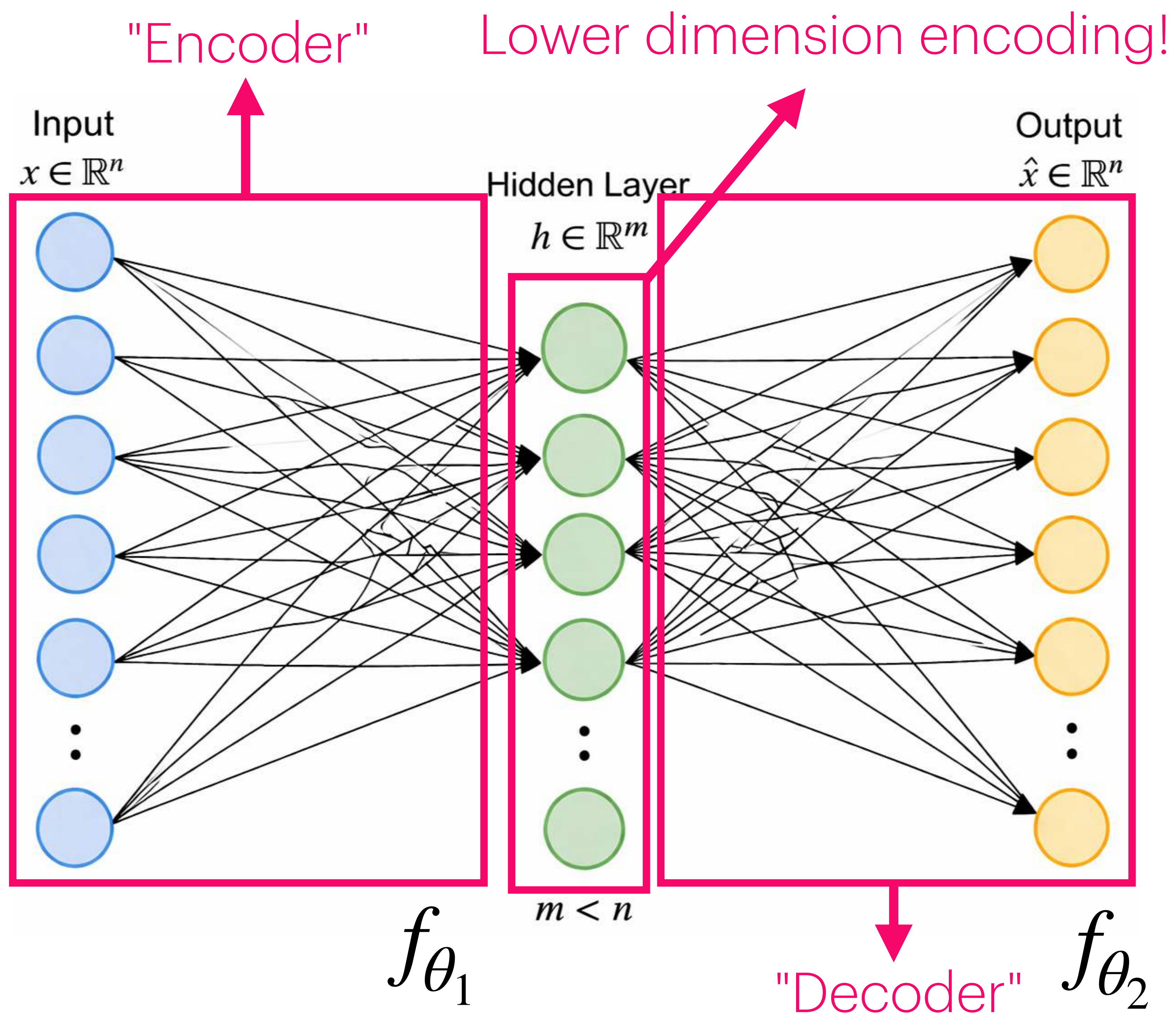
$f_{\theta_1} : \mathbb{R}^n \to \mathbb{R}^m$ – the encoder

$f_{\theta_2} : \mathbb{R}^m \to \mathbb{R}^n$ – the decoder

What do f_{θ_1} and f_{θ_2} even do?

We do not know.

Simplifying the problem



$f_{\theta_1} : \mathbb{R}^n \to \mathbb{R}^m$ – the encoder

$f_{\theta_2} : \mathbb{R}^m \to \mathbb{R}^n$ – the decoder

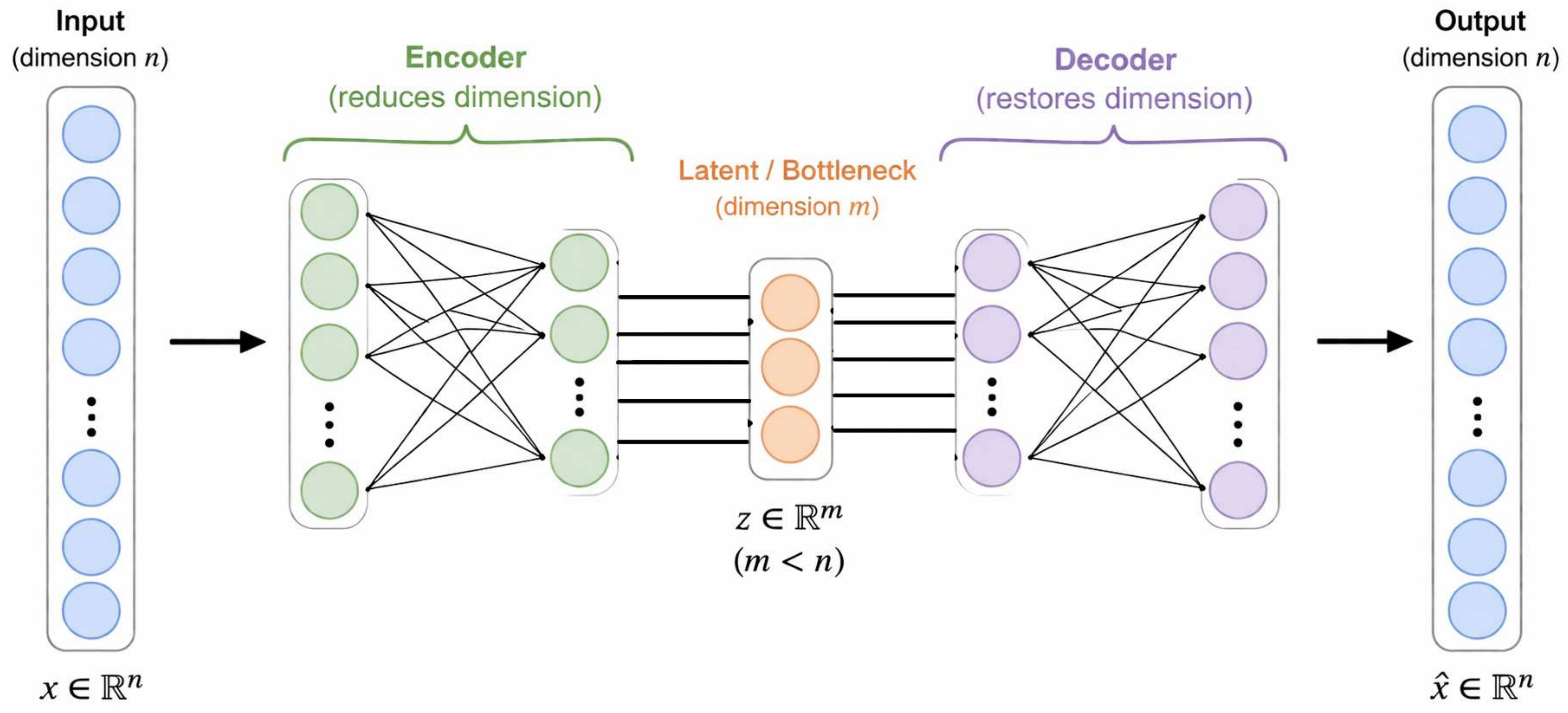
What do f_{θ_1} and f_{θ_2} even do?

We do not know.

$$f_{\theta_2}(f_{\theta_1}(x)) = x$$

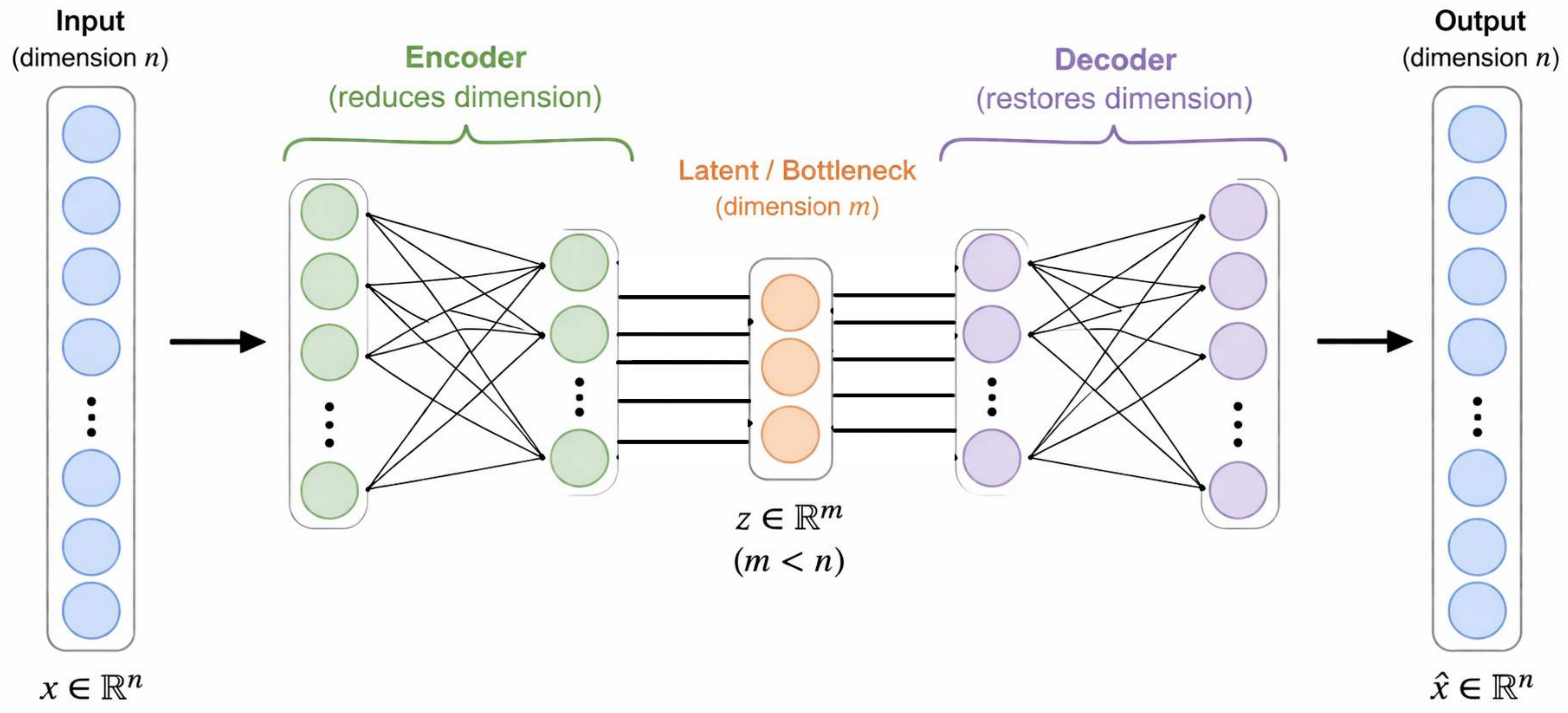
The architecture

The architecture

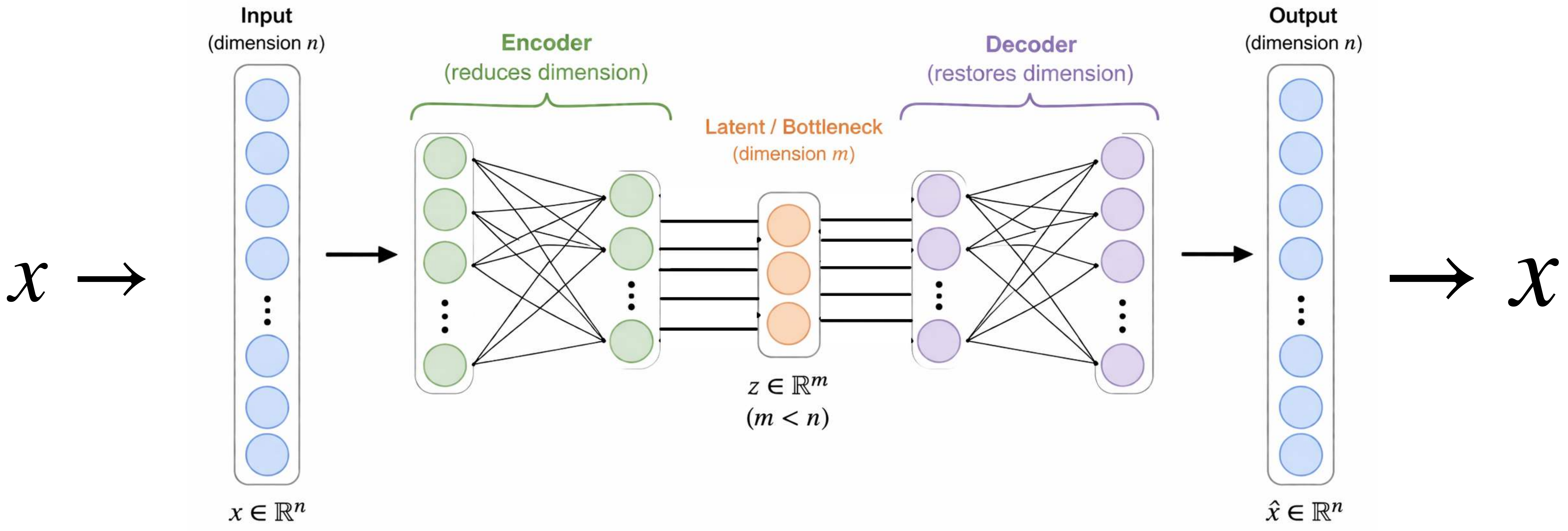


The architecture

$x \rightarrow$

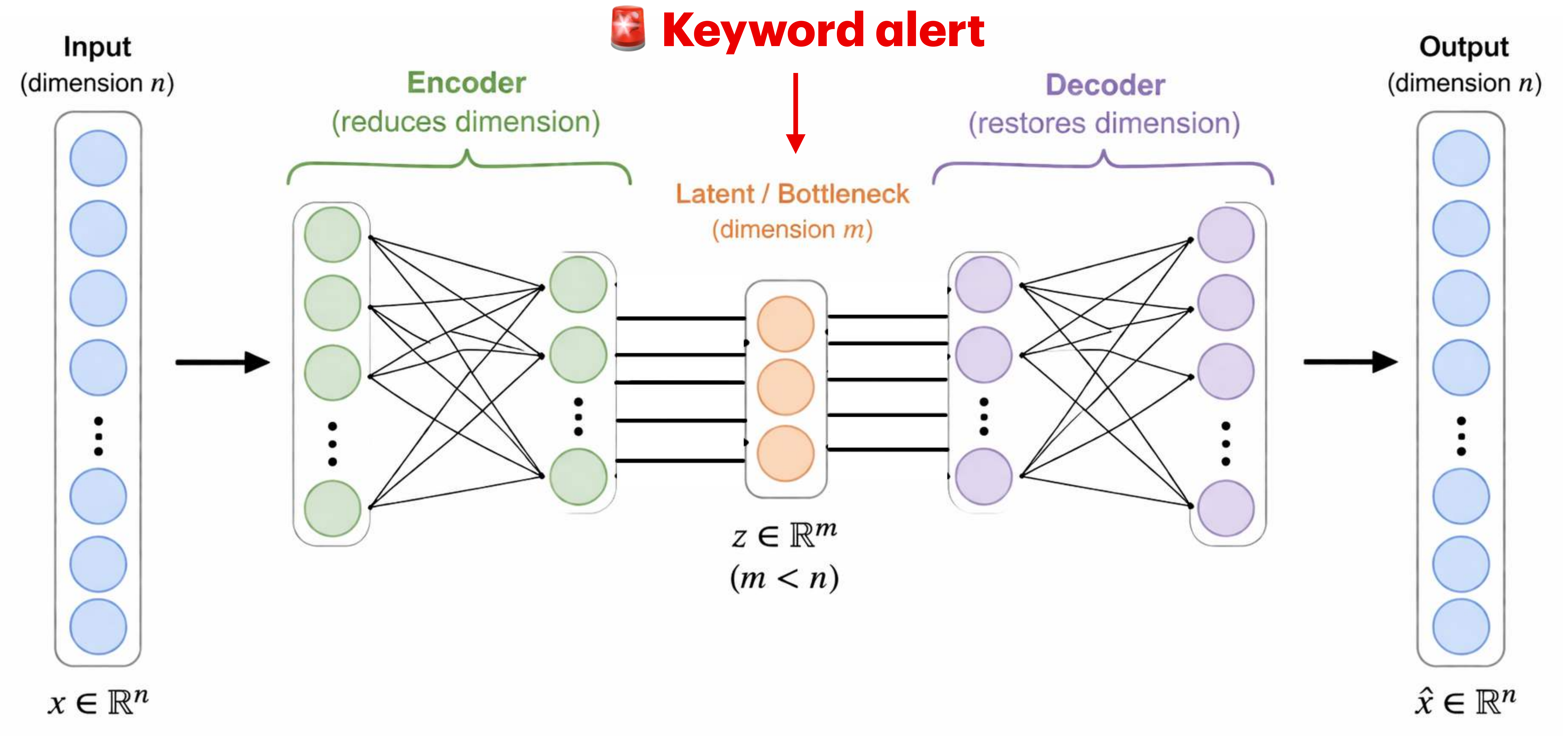


The architecture



The architecture

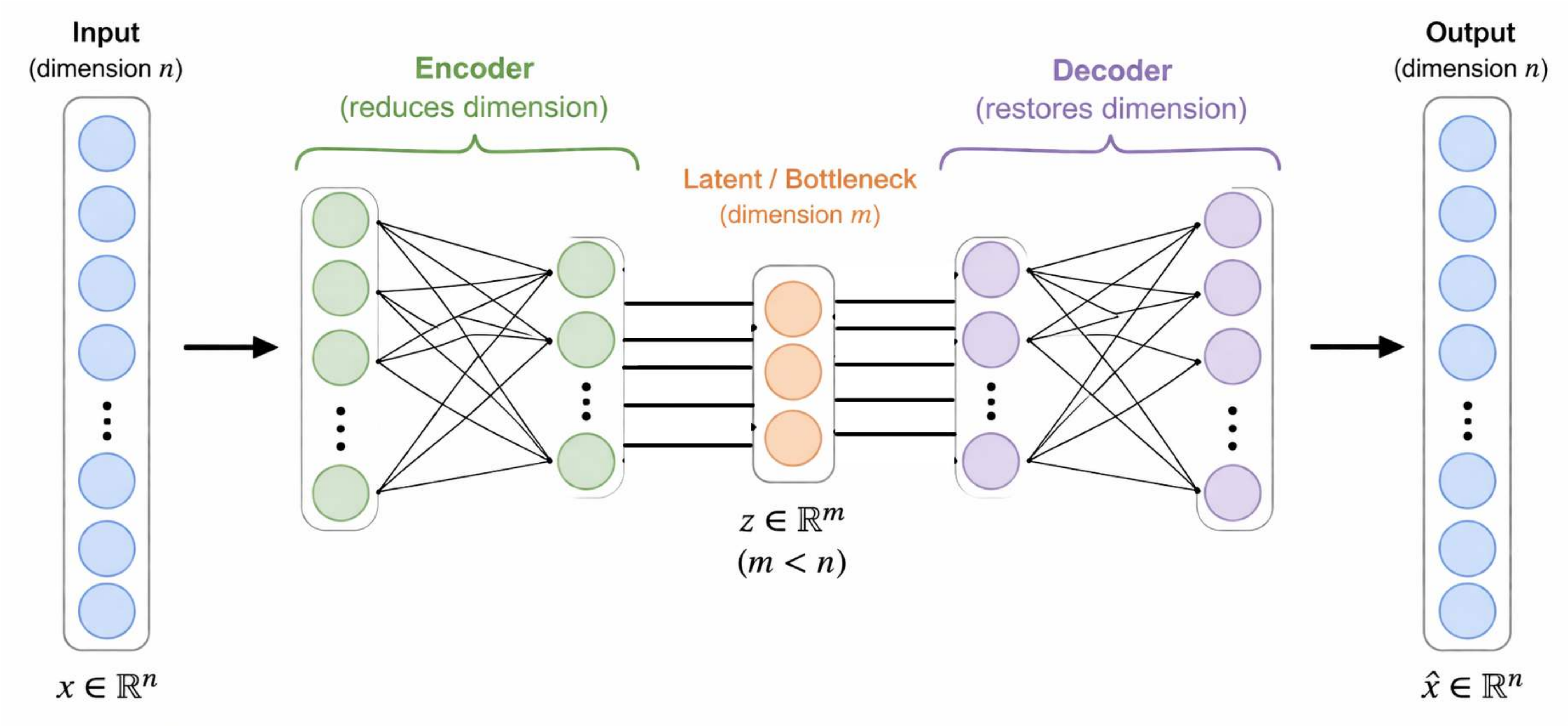
$x \rightarrow$



$\rightarrow x$

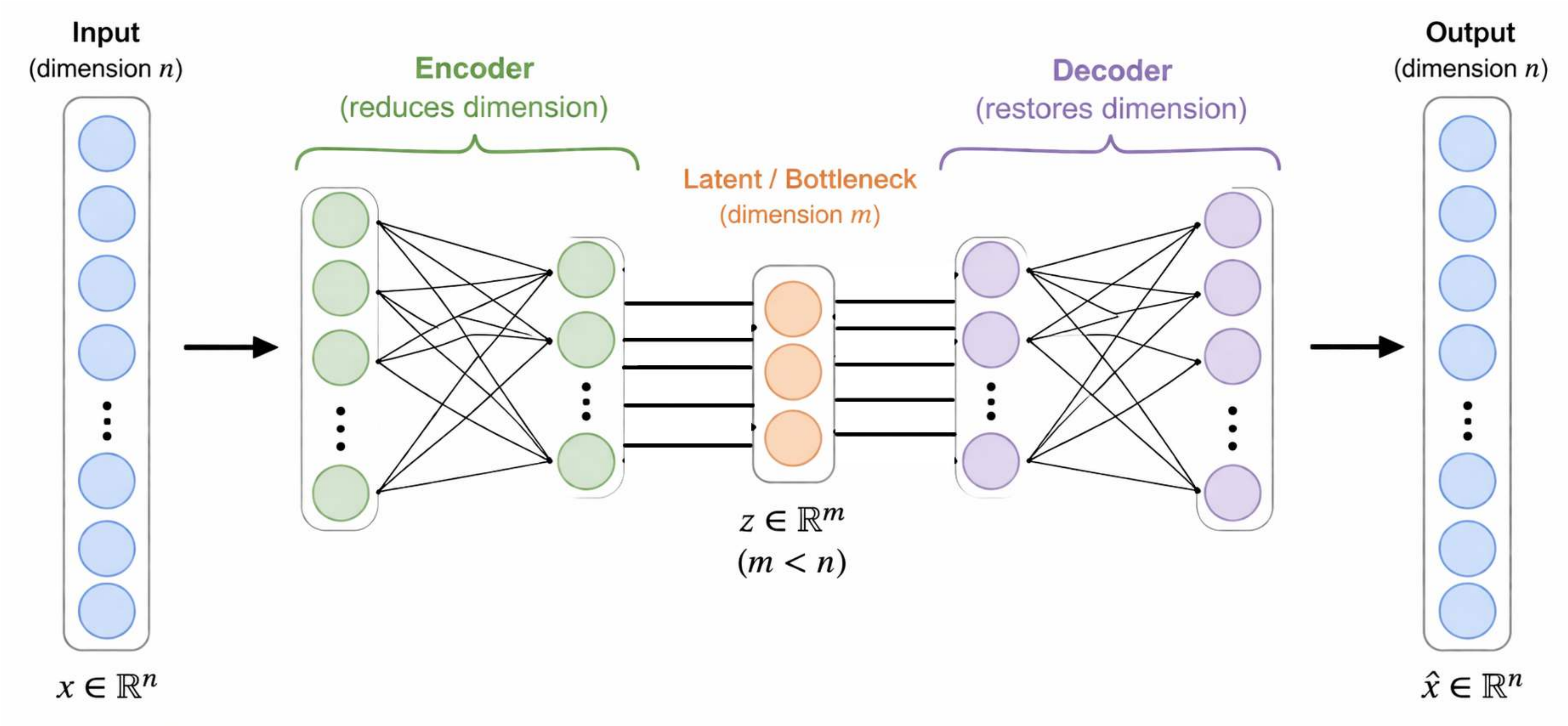
What can we control?

What can we control?

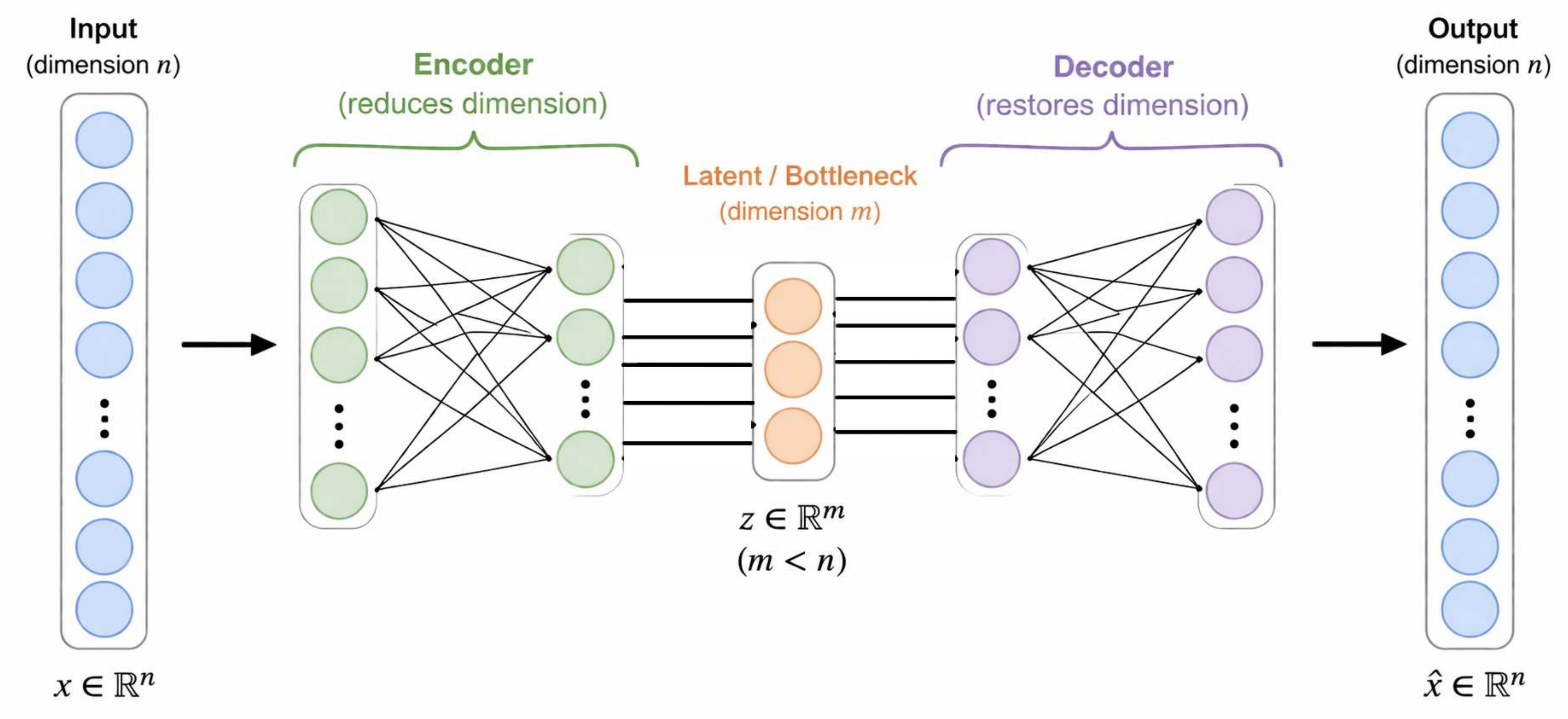


What can we control?

Hyperparameters?



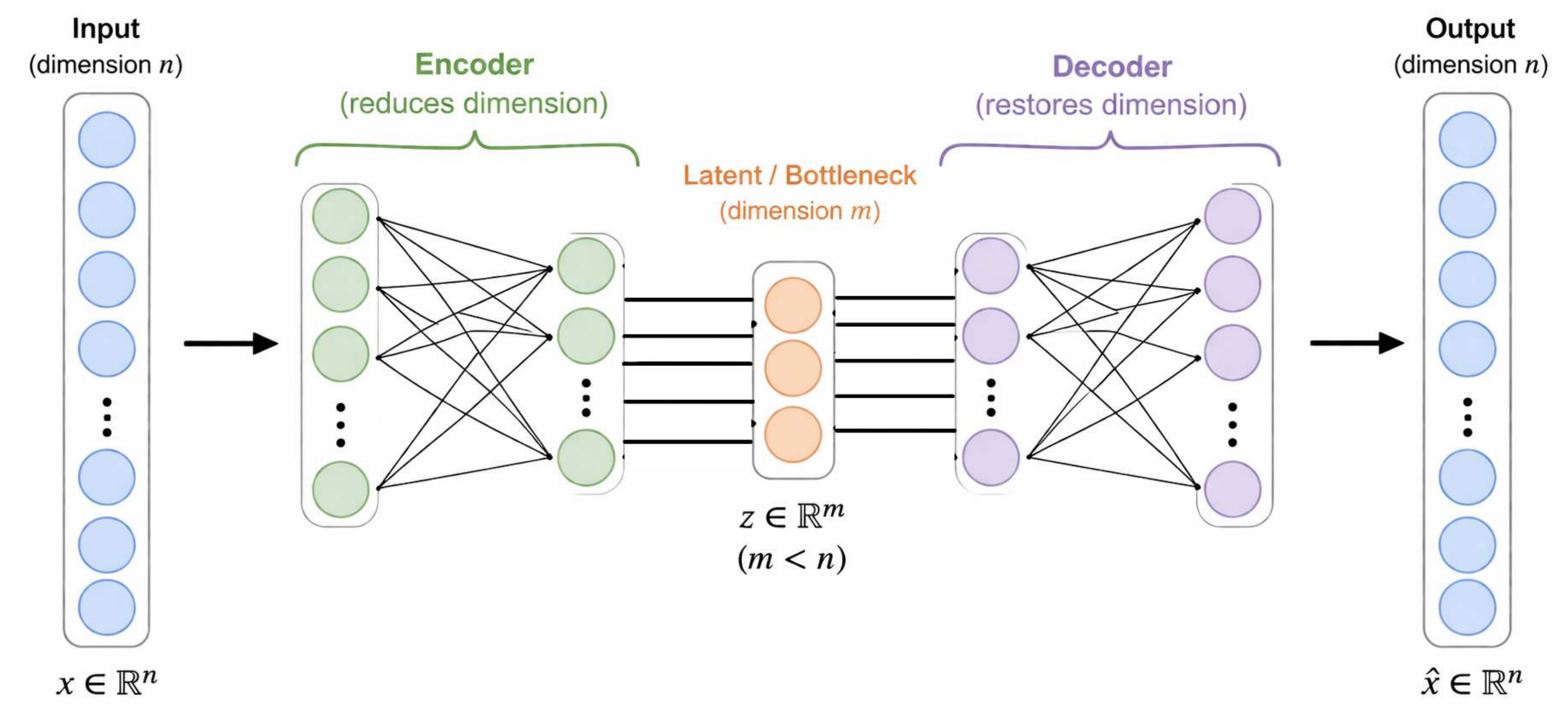
What can we control?



Hyperparameters?

Compression quality?

What can we control?

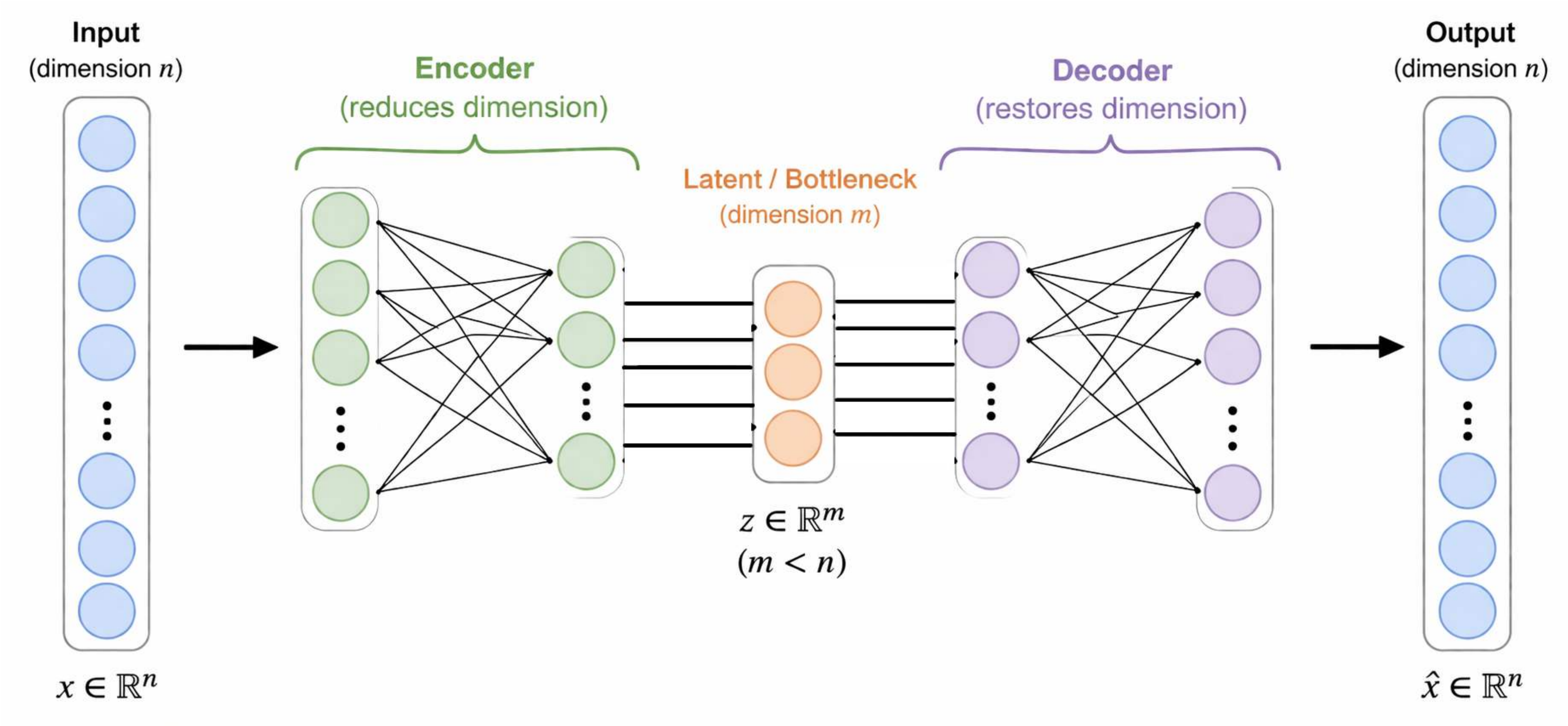


Hyperparameters?

Compression quality?

Expressiveness of neural network. Size of latent dimension. Training duration

What can we control?



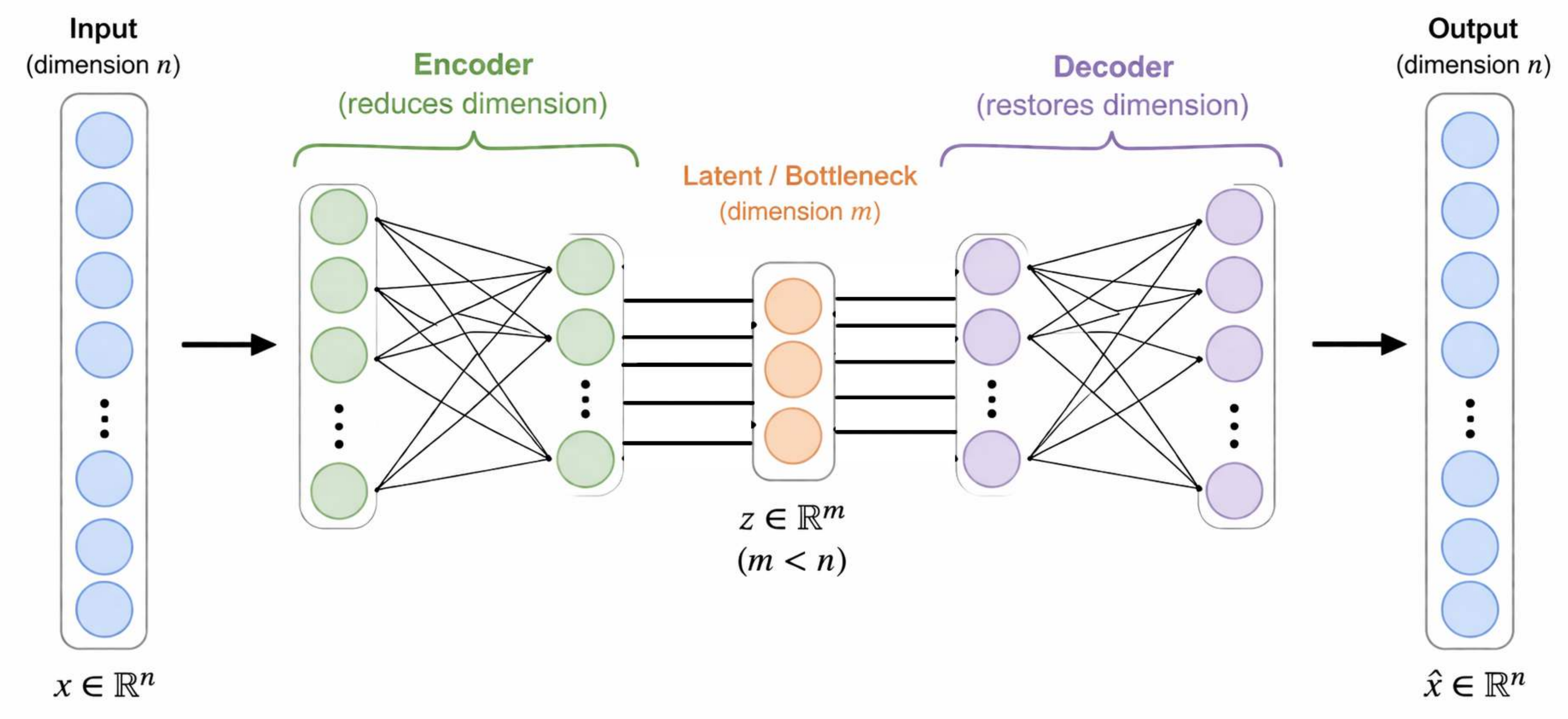
Hyperparameters?

Compression quality?

Expressiveness of neural network. Size of latent dimension. Training duration

Latent dimension?

What can we control?



Hyperparameters?

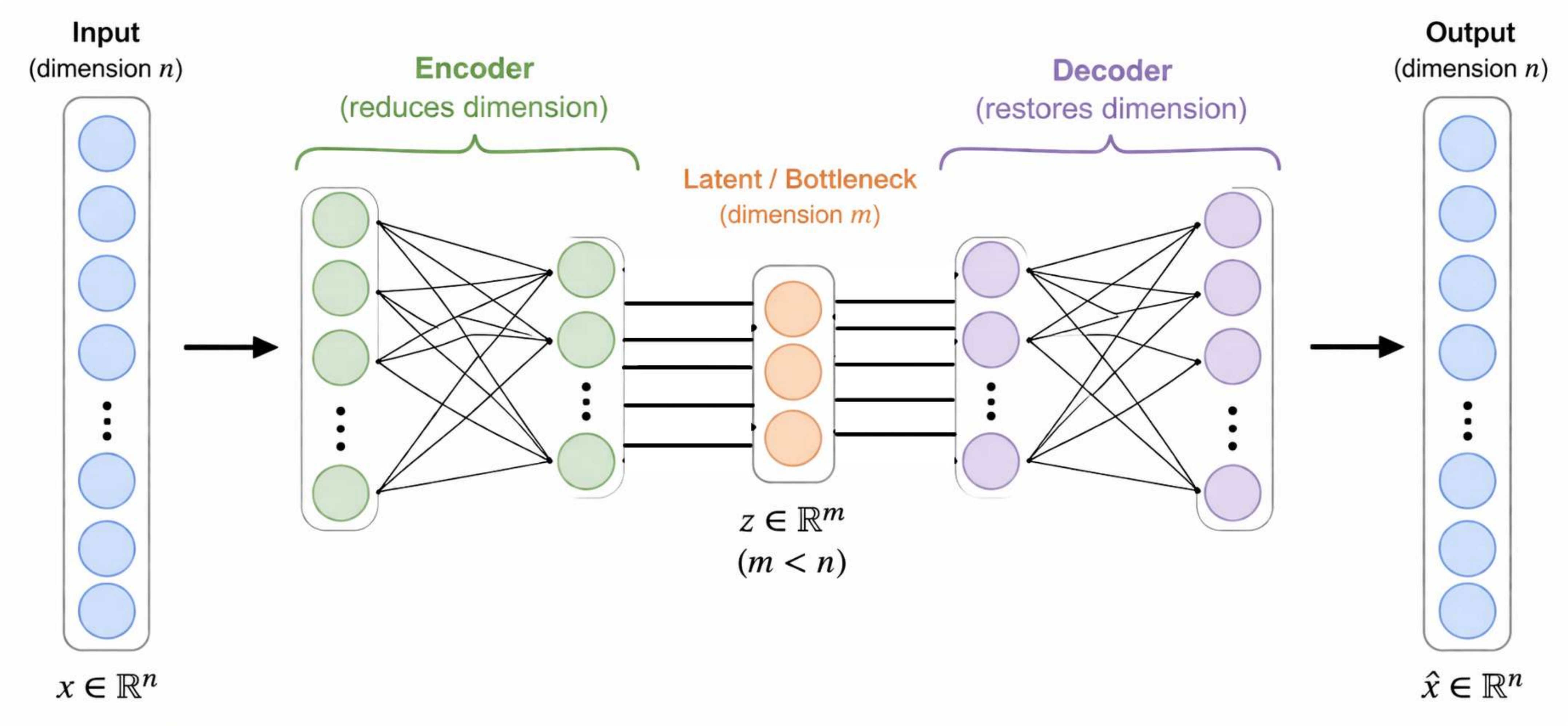
Compression quality?

Expressiveness of neural network. Size of latent dimension. Training duration

Latent dimension?

The value of m.

What can we control?



Hyperparameters?

Compression quality?

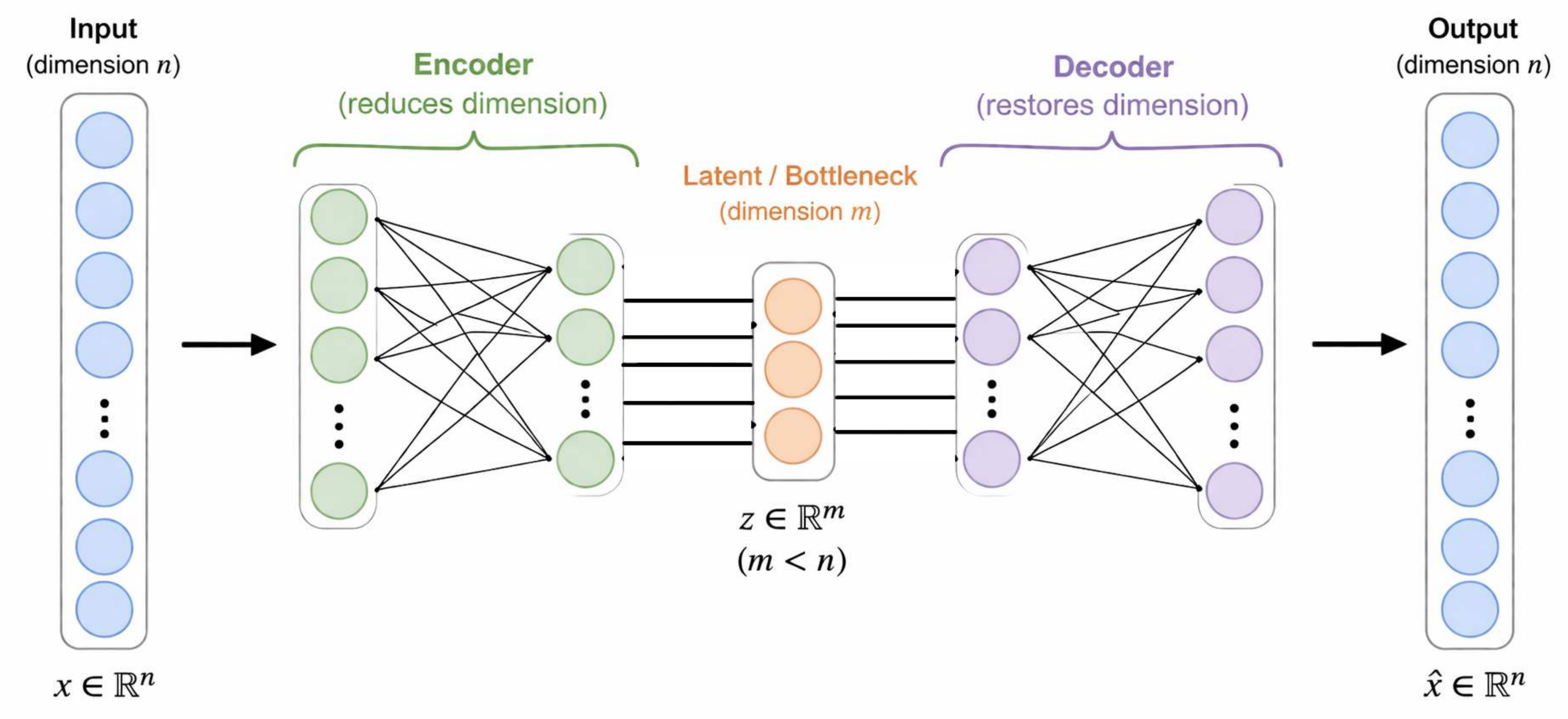
Expressiveness of neural network. Size of latent dimension. Training duration

Latent dimension?

The value of m.

Cost of encoding/decoding?

What can we control?



Hyperparameters?

Compression quality?

Expressiveness of neural network. Size of latent dimension. Training duration

Latent dimension?

The value of m.

Cost of encoding/decoding?

Architecture and size of Network

Note on neural networks

We have seen neural networks as compositions of smaller systems:

- Layers (weights + biases)
- Residual (skip) connections
- Convolution Layers

Note on neural networks

We have seen neural networks as compositions of smaller systems:

- Layers (weights + biases)
- Residual (skip) connections
- Convolution Layers

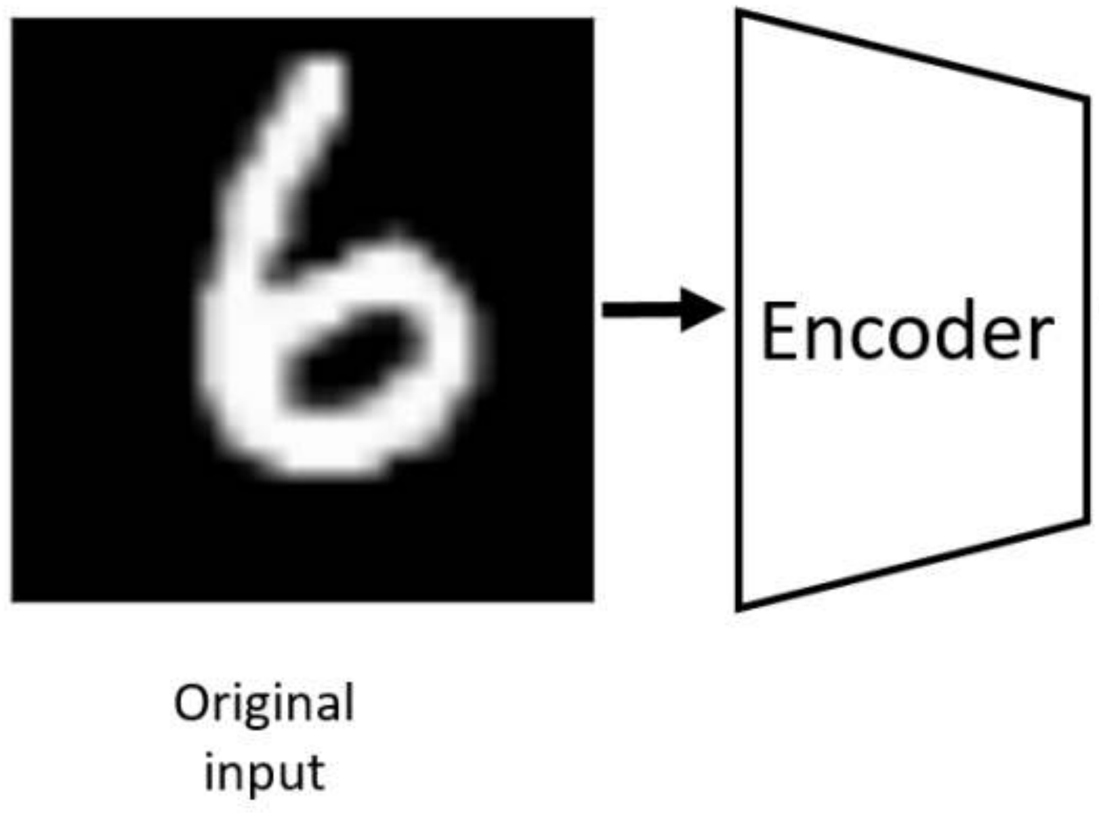
This shows a generalisation:

Neural networks can be built with neural networks as building blocks!

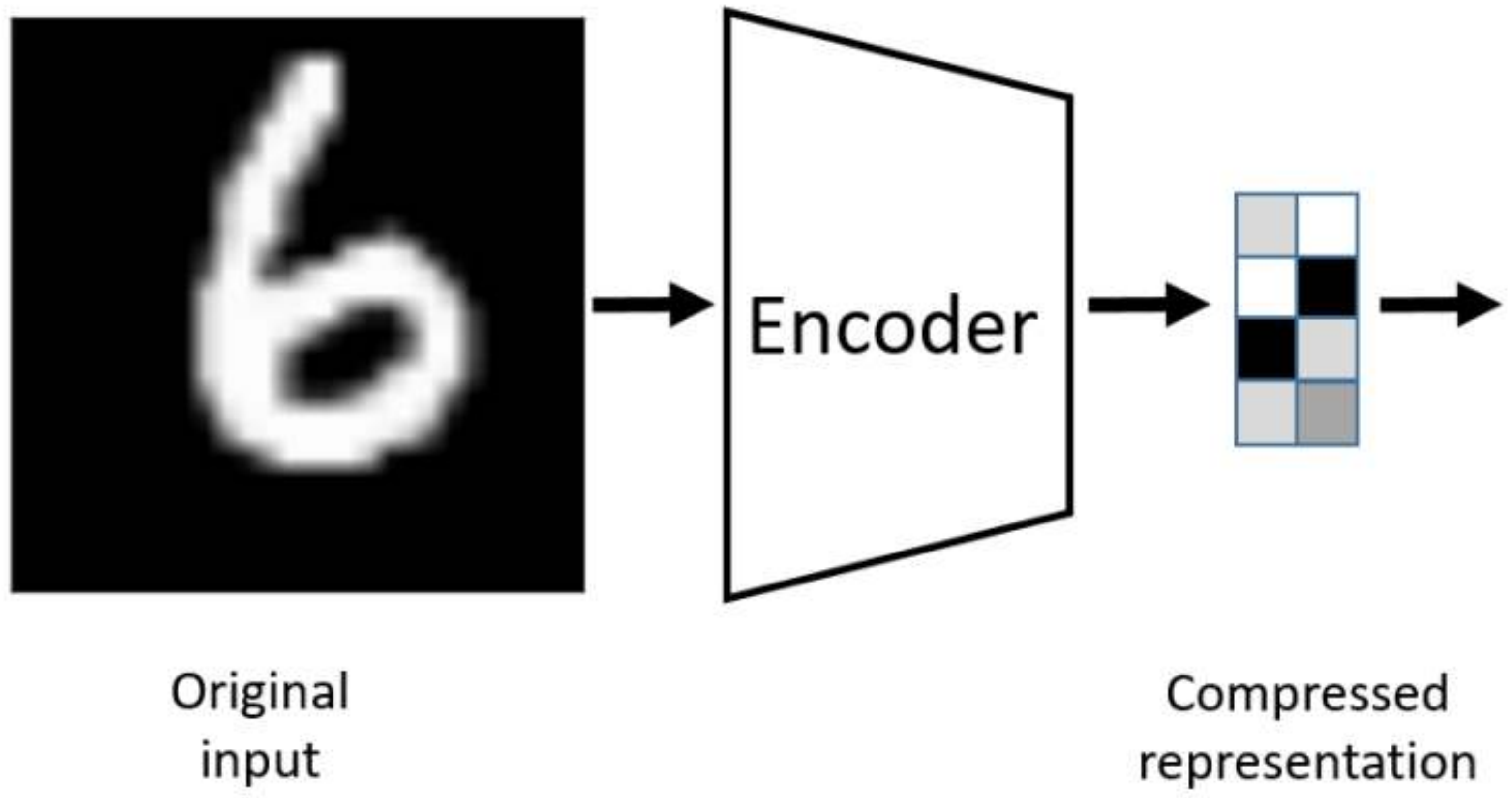
A concrete example

Original
input

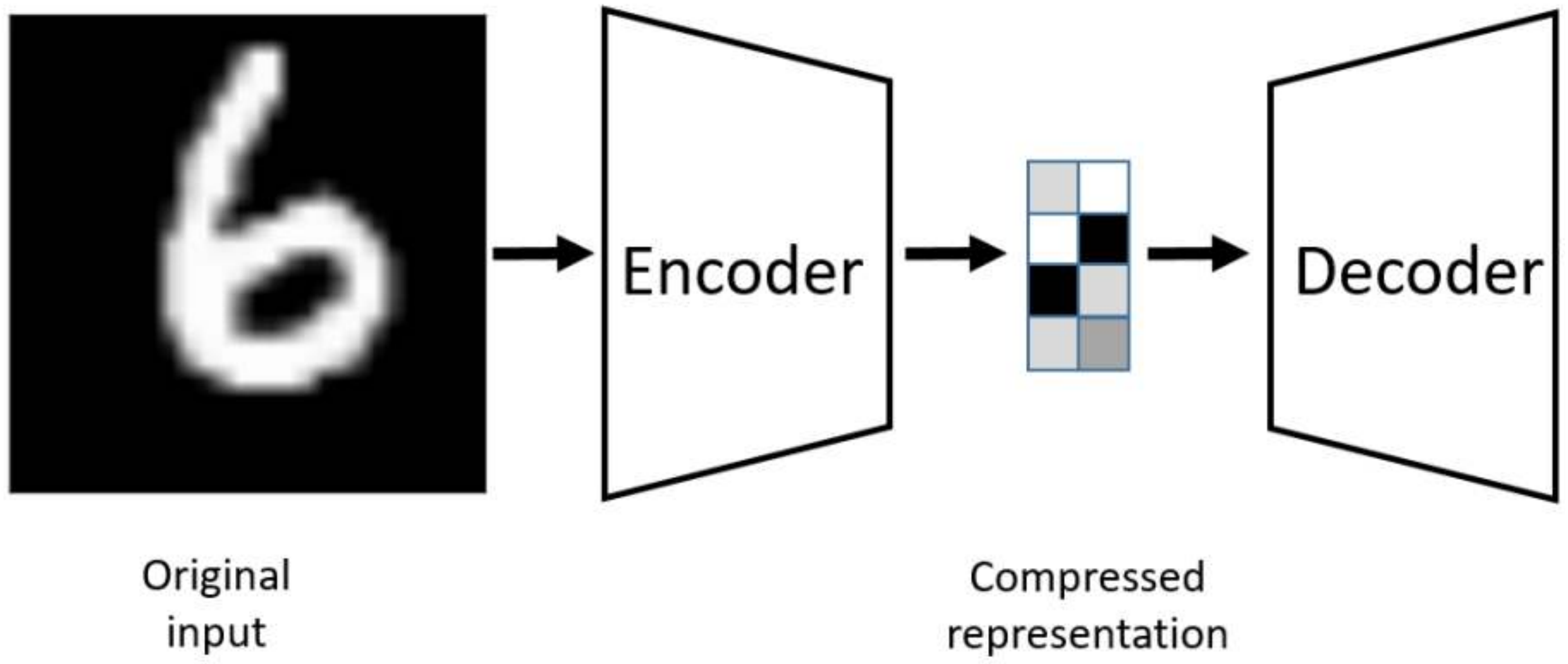
A concrete example



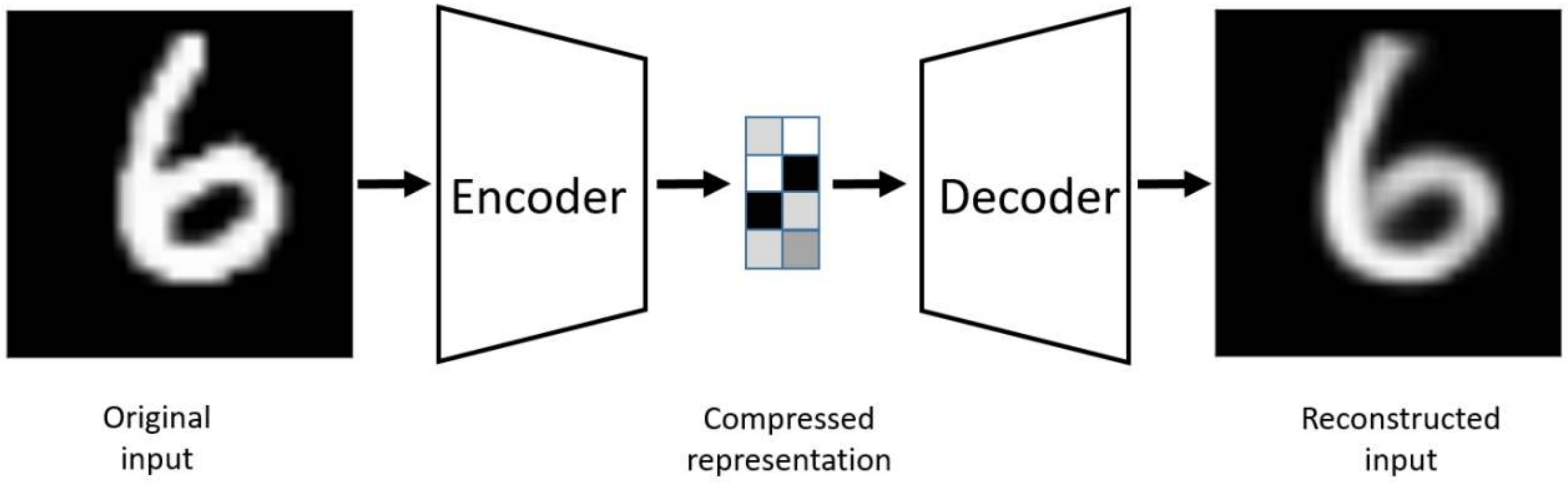
A concrete example



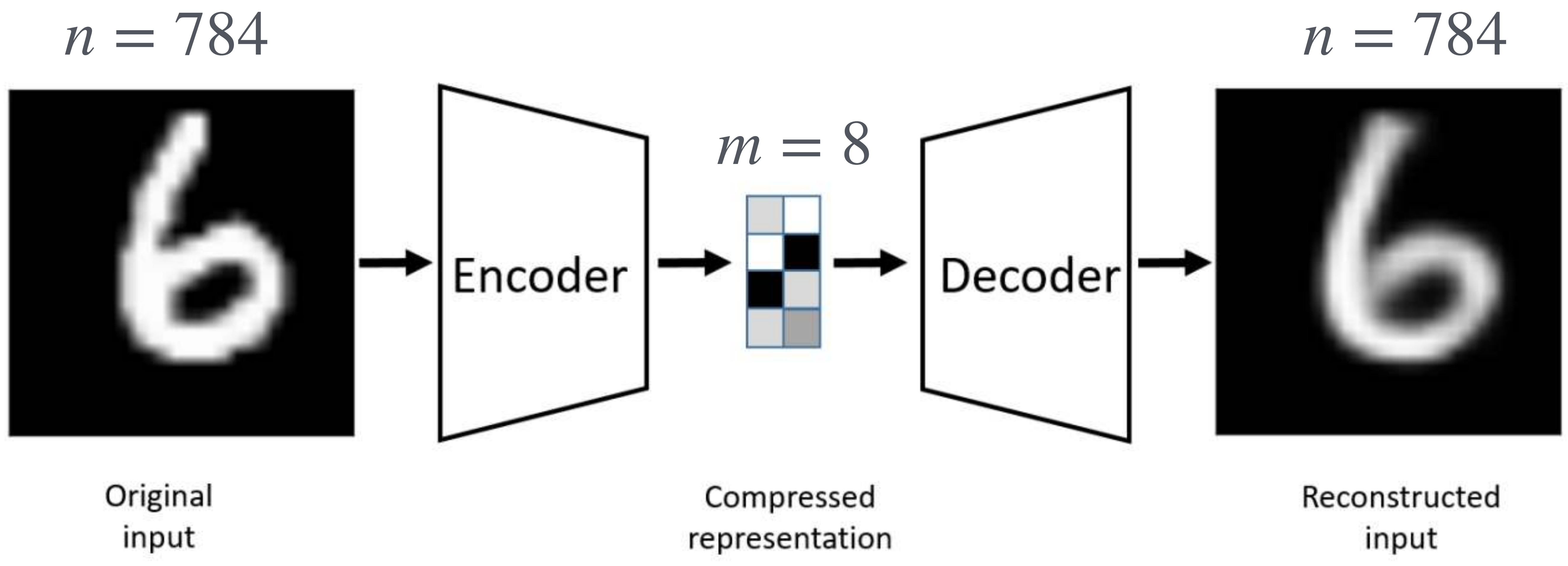
A concrete example



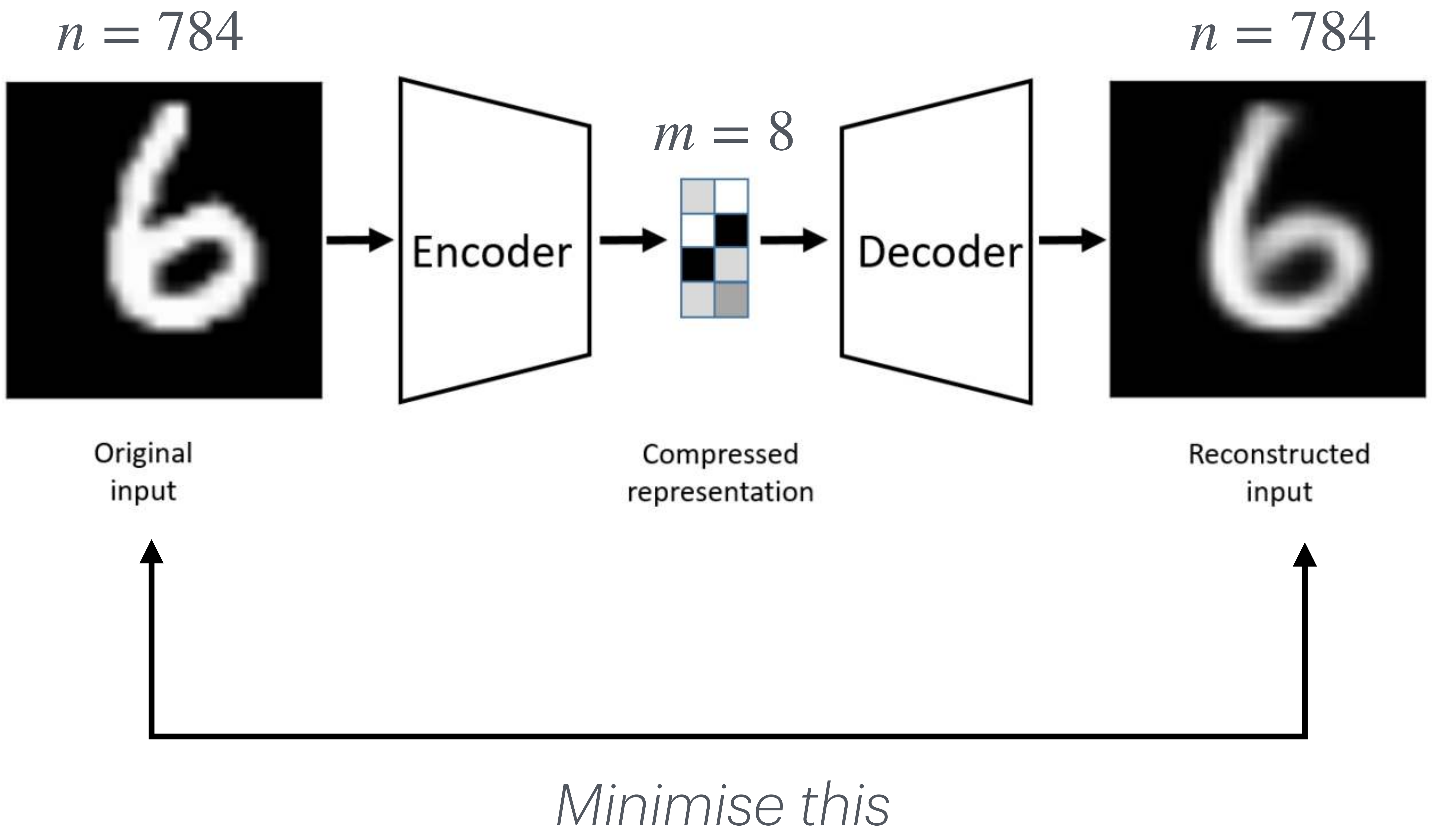
A concrete example



A concrete example



A concrete example



The optimisation problem

The optimisation problem

$$f_{\theta_2}(f_{\theta_1}(x)) = x$$

The optimisation problem

$$f_{\theta_2}(f_{\theta_1}(x)) = x$$

$$L(\theta_1, \theta_2) = \mathbb{E}_{x \sim p_{data}} \left[\left\| f_{\theta_2}\big(f_{\theta_1}(x)\big) - x \right\|_2^2 \right]$$

The optimisation problem

$$f_{\theta_2}(f_{\theta_1}(x)) = x$$

$$L(\theta_1, \theta_2) = \mathbb{E}_{x \sim p_{data}} \left[\left\| f_{\theta_2}\big(f_{\theta_1}(x)\big) - x \right\|_2^2 \right]$$

$$\hat{L}(\theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} \left\| f_{\theta_2}\big(f_{\theta_1}(x_i)\big) - x_i \right\|_2^2$$

The Algorithm

Given dataset X, encoder f_{θ_e}, decoder g_{θ_d}

for each epoch:

    for each mini-batch $B \subset X$:

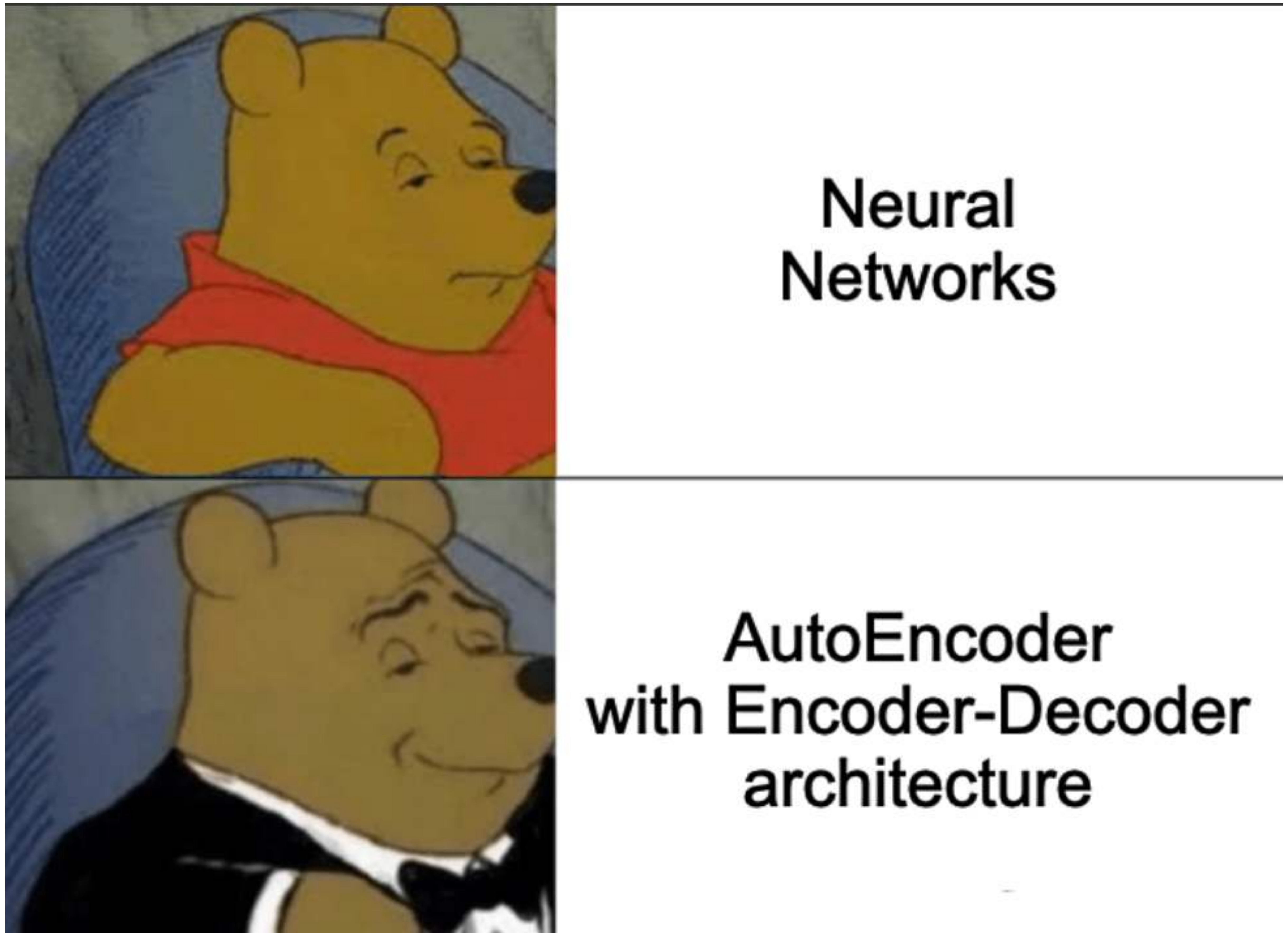
$$z = f_{\theta_e}(x)$$

$$\hat{x} = g_{\theta_d}(z)$$

$$\mathscr{L} = \|x - \hat{x}\|^2$$

$$\theta_e \leftarrow \theta_e - \eta \nabla_{\theta_e} \mathscr{L}$$

$$\theta_d \leftarrow \theta_d - \eta \nabla_{\theta_d} \mathscr{L}$$


Neural Networks
AutoEncoder with Encoder-Decoder architecture

Variational Auto Encoders

The whole secret lies in confusing the enemy, so that he cannot fathom our real intent.
- Sun Tzu, “The Art of War”

Deeper look

Deeper look

What does the the compression actually do?

Deeper look

What does the the compression actually do?

Autoencoders tend to have discrete latent spaces, where each input is mapped to some arbitrary point

So what?

So what?

If we take a random point in the latent space and decode it, we get garbage

So what?

If we take a random point in the latent space and decode it, we get garbage

The latent space does not capture the semantic meaning well

So what?

If we take a random point in the latent space and decode it, we get garbage

The latent space does not capture the semantic meaning well

Can we make it a nicer space, maybe continuous?

Hunt for a better space

Hunt for a better space

The encoder predicts a *fixed* output, causing this memorisation

Hunt for a better space

The encoder predicts a *fixed* output, causing this memorisation

Can we *vary* the output of the encoder?

Hunt for a better space

The encoder predicts a *fixed* output, causing this memorisation

Can we *vary* the output of the encoder?

Instead of encoding to fixed point, encode using some *randomness*

Randomness

Randomness

Randomness should be small enough for the decoder to reconstruct, but large enough for the encoder to not memorise

Randomness

Randomness should be small enough for the decoder to reconstruct, but large enough for the encoder to not memorise

But how?

Randomness

Randomness should be small enough for the decoder to reconstruct, but large enough for the encoder to not memorise

But how?

Just learn it

Learning the randomness to learn…

Learning the randomness to learn...

The randomness is parameterised by a Gaussian

Learning the randomness to learn…

The randomness is parameterised by a Gaussian

The encoder takes a data point and outputs the mean μ and the log variance $\log \sigma^2$ of the gaussian

Learning the randomness to learn…

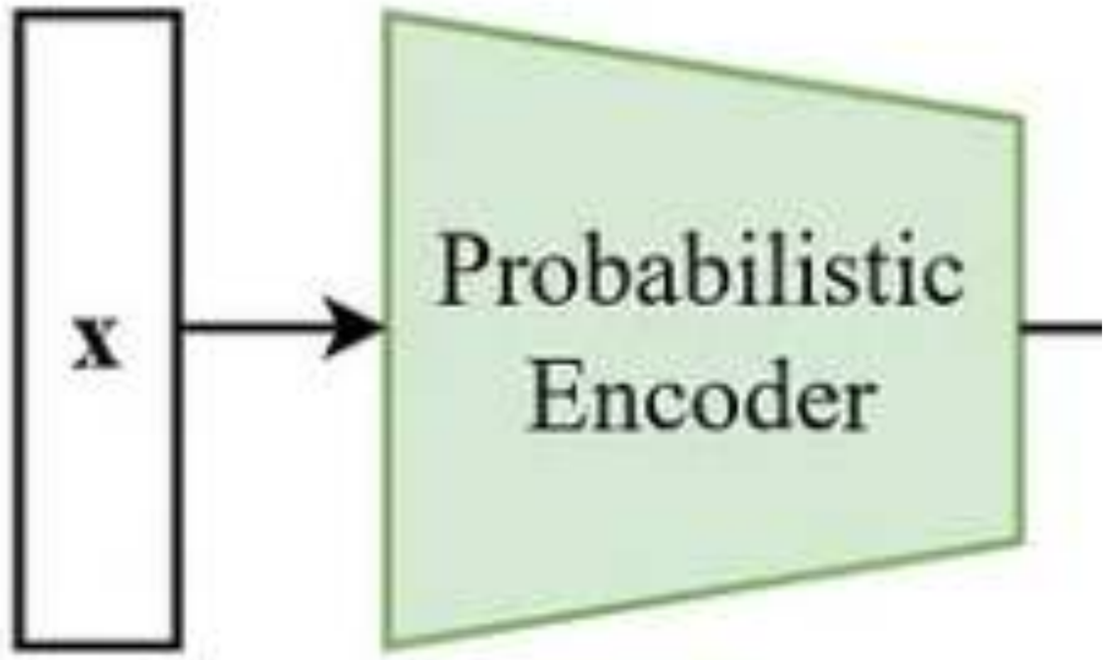
The randomness is parameterised by a Gaussian

The encoder takes a data point and outputs the mean μ and the log variance $\log \sigma^2$ of the gaussian

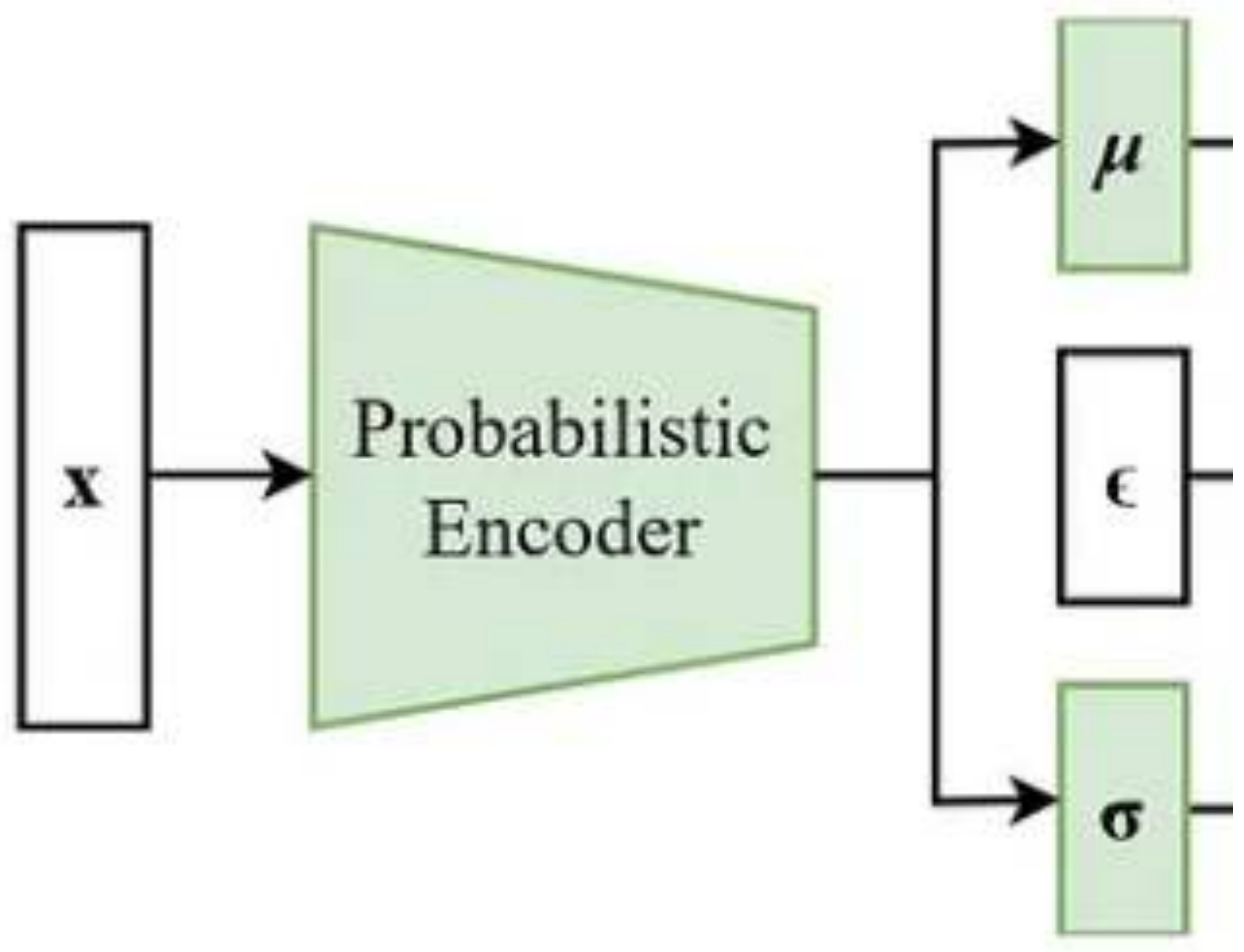
The latent is now a sample from this distribution

An example

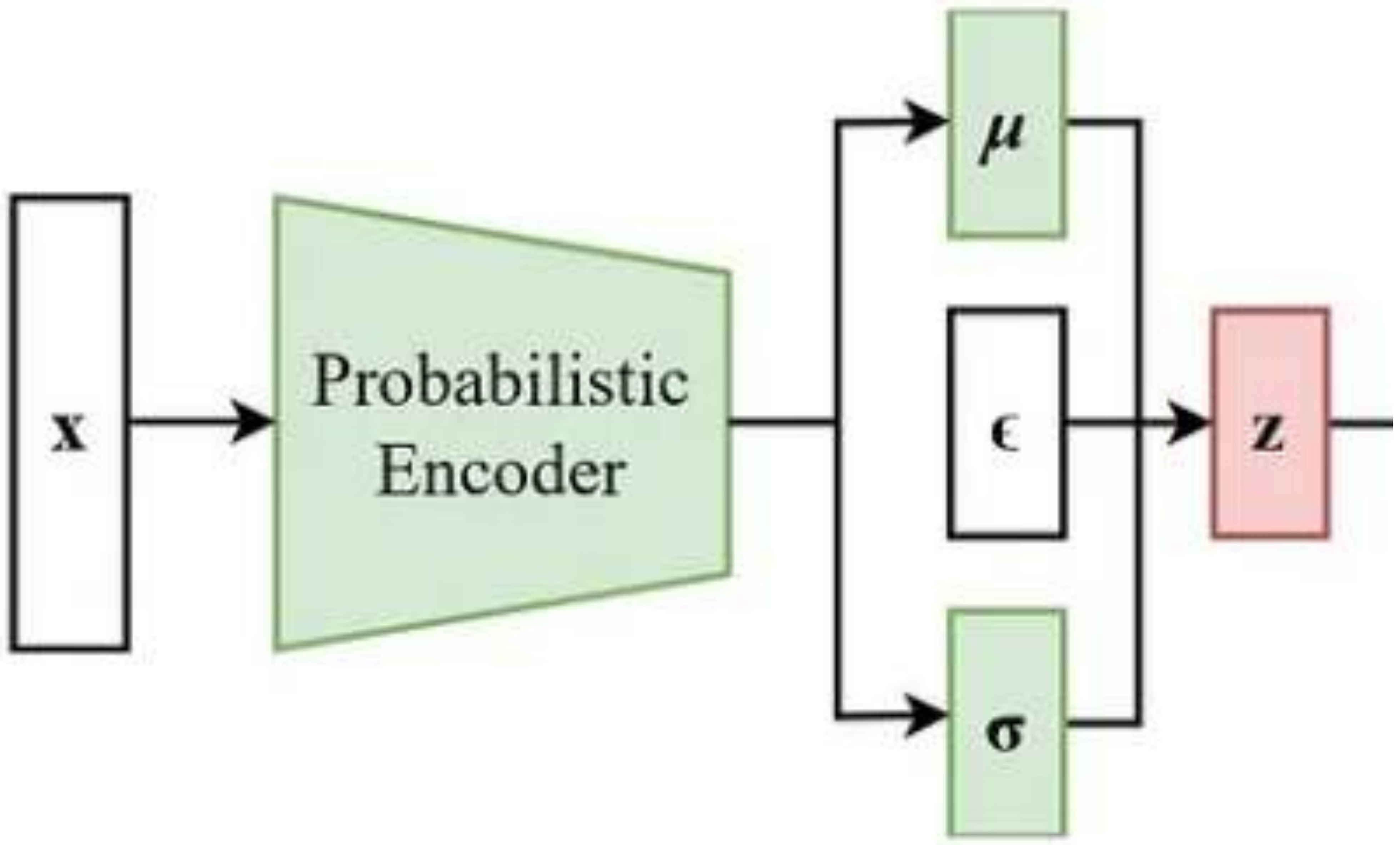
An example



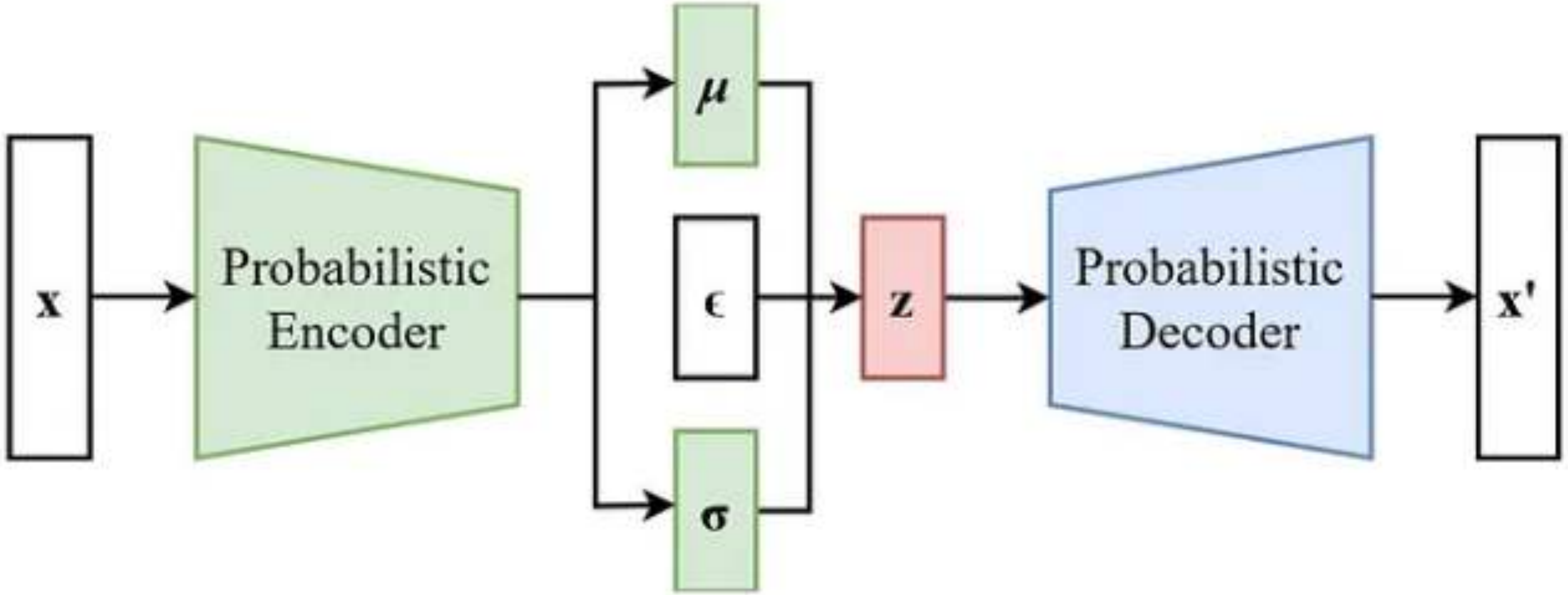
An example



An example



An example



What can do wrong?

What can do wrong?

Model always sets the variance to 0, becomes an auto encoder

What can do wrong?

Model always sets the variance to 0, becomes an auto encoder

Can we penalise the model for being too “discrete”?

The penalty

The penalty

Force the encoder to produce outputs close to a standard Gaussian by adding a KL loss

The penalty

Force the encoder to produce outputs close to a standard Gaussian by adding a KL loss

$$\mathrm{KL}\left(p \| q\right) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Measures how close two distributions are

The closed form

$$p = \mathcal{N}(\mu_p, \sigma_p^2 I), \quad q = \mathcal{N}(\mu_q, \sigma_q^2 I)$$

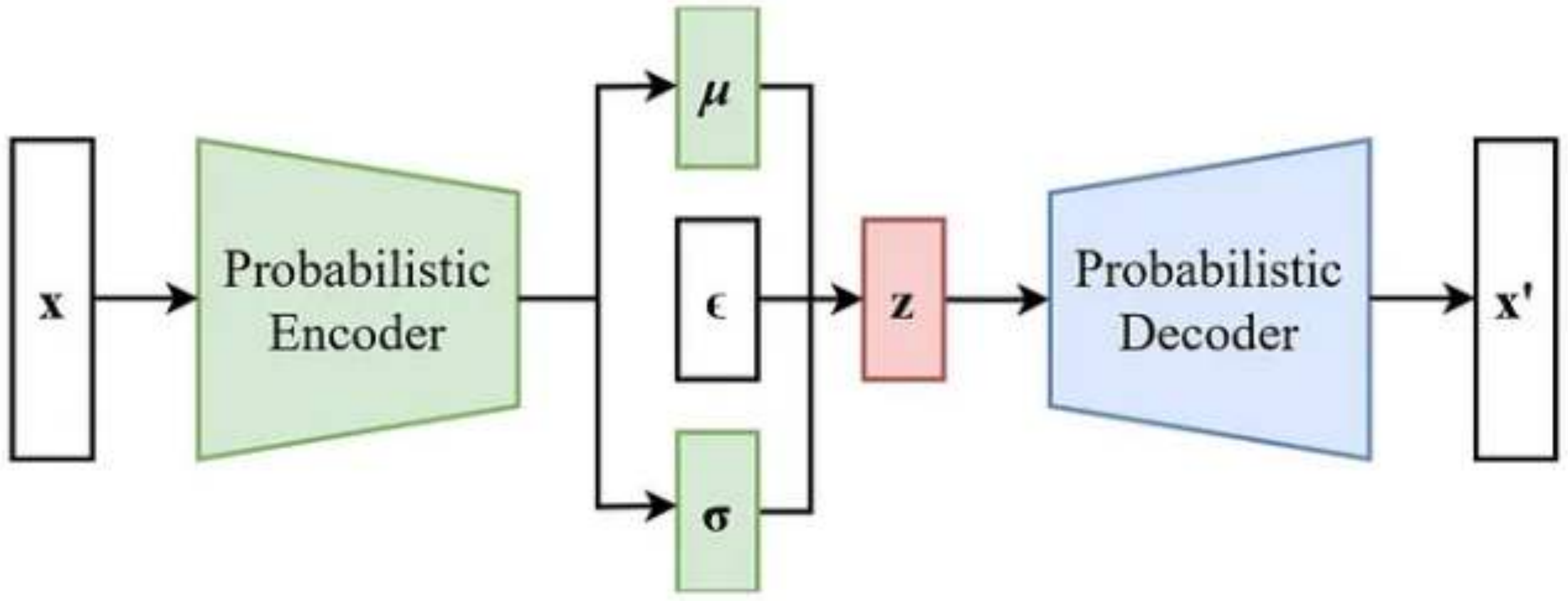
The closed form

$$\mathrm{KL}\left(p\|q\right) = \frac{1}{2}\left[d\frac{\sigma_p^2}{\sigma_q^2} + \frac{\|\mu_p - \mu_q\|^2}{\sigma_q^2} - d + d\log\left(\frac{\sigma_q^2}{\sigma_p^2}\right)\right]$$

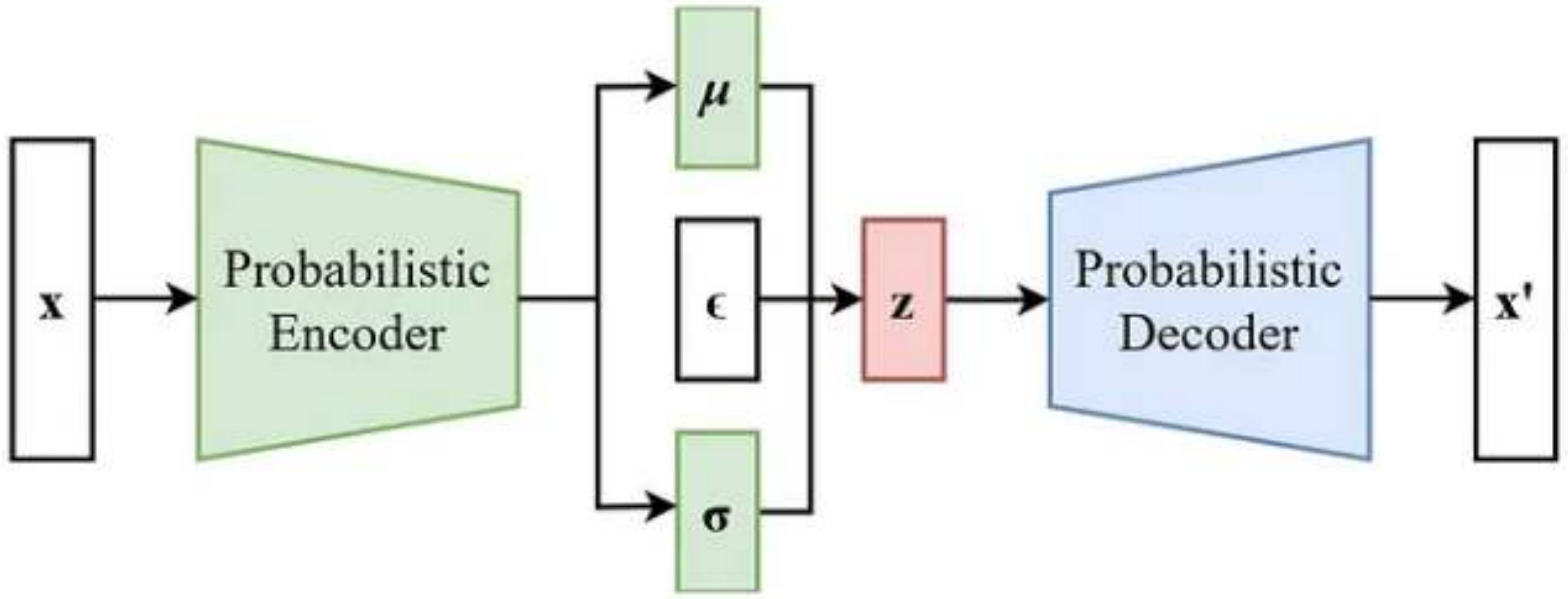
$$p = \mathcal{N}(\mu_p, \sigma_p^2 I), \quad q = \mathcal{N}(\mu_q, \sigma_q^2 I)$$

Finally…

Finally...

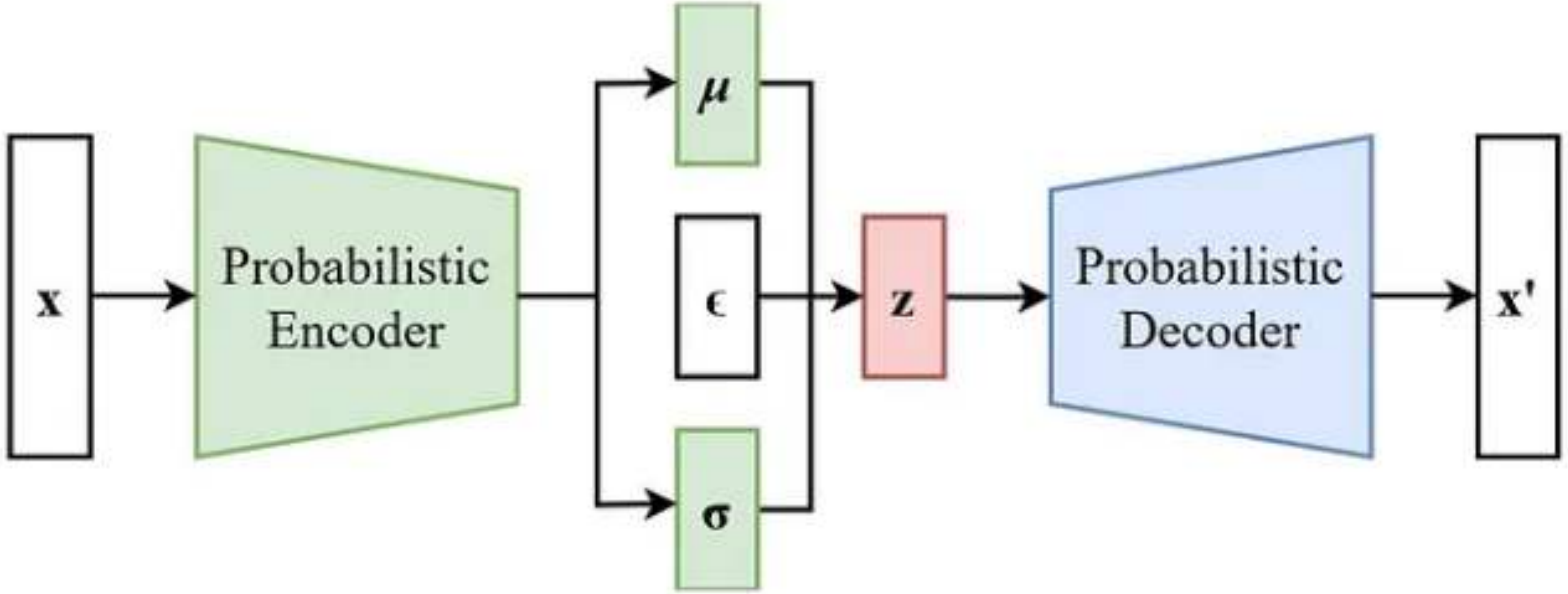


Finally...



$$L(\theta_1, \theta_2) = \|x - f_{\theta_2}(z)\|_2^2 + \beta \cdot D_{KL}(q_{\theta_1}(z \mid x) \parallel \mathcal{N}(0,I))$$

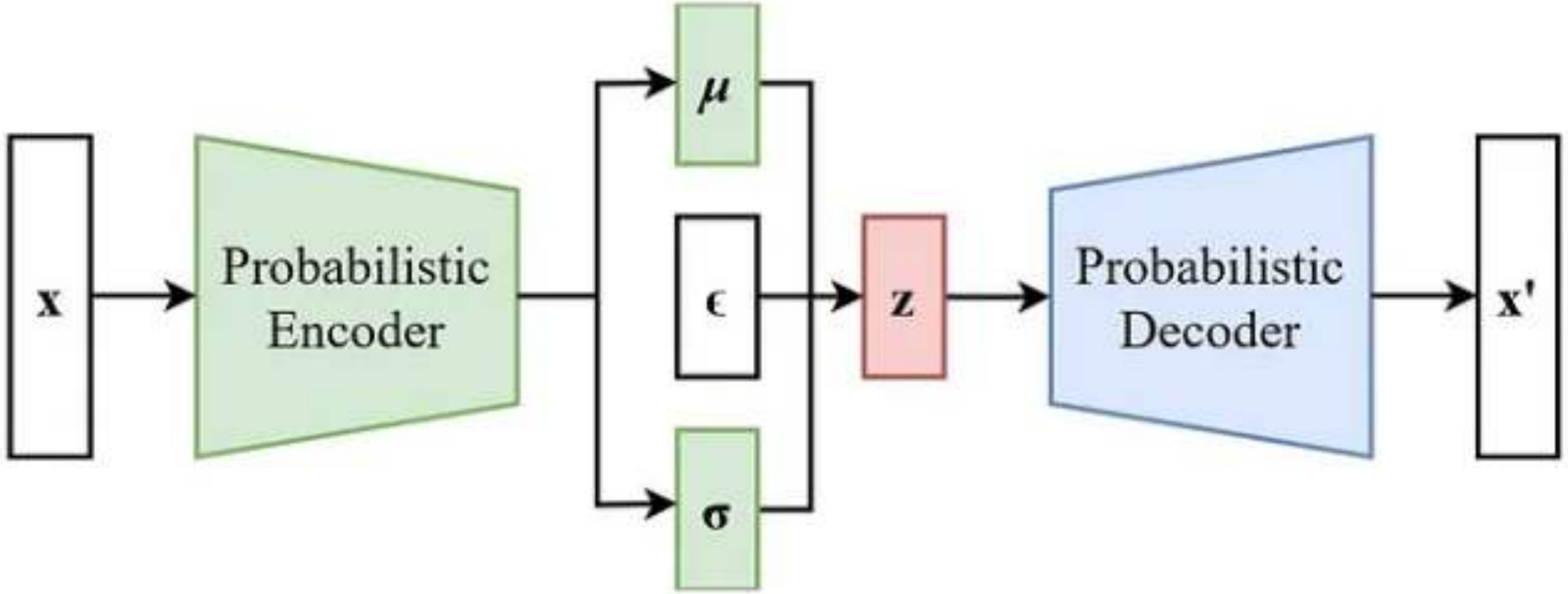
Finally...



$$L(\theta_1, \theta_2) = \boxed{\|x - f_{\theta_2}(z)\|_2^2} + \beta \cdot D_{KL}(q_{\theta_1}(z \mid x) \parallel \mathcal{N}(0,I))$$

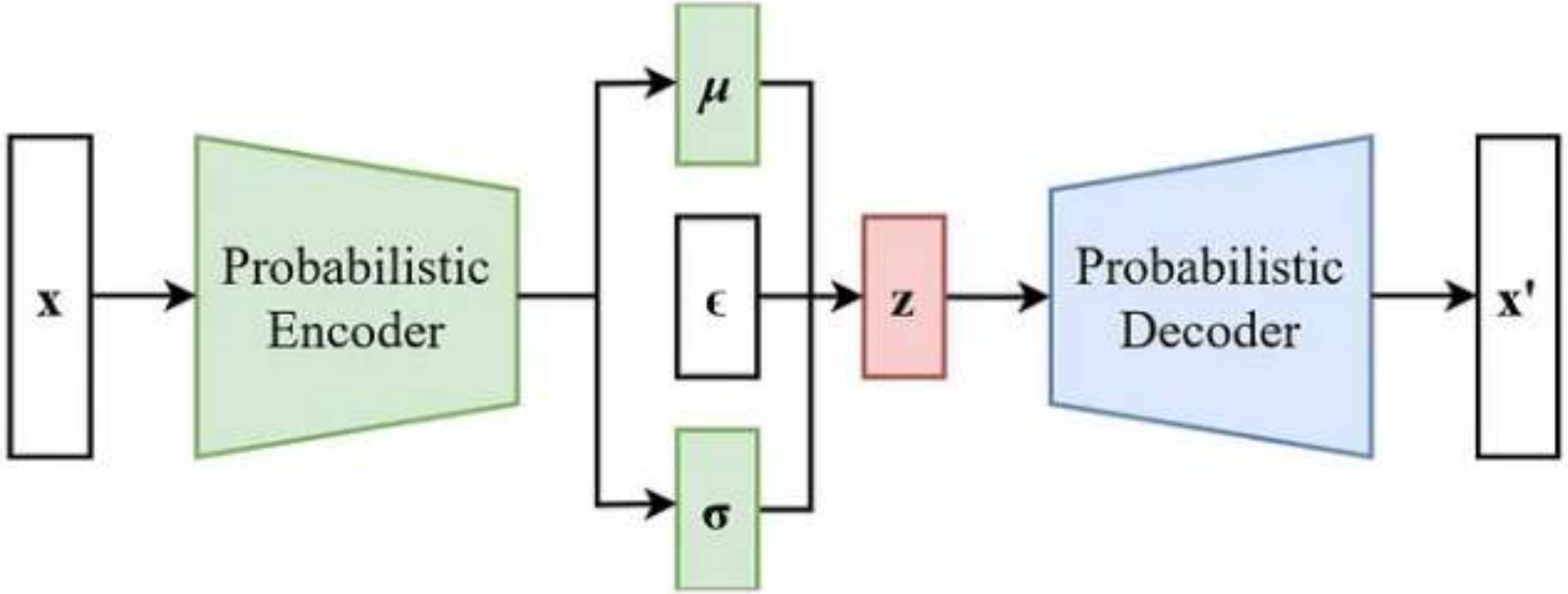
Recon. Loss

Finally...



$$L(\theta_1, \theta_2) = \underbrace{\boxed{\|x - f_{\theta_2}(z)\|_2^2}}_{\text{Recon. Loss}} + \beta \cdot \underbrace{\boxed{D_{KL}(q_{\theta_1}(z \mid x) \parallel \mathcal{N}(0,I))}}_{\text{KL Penalty}}$$

Finally...



$$L(\theta_1, \theta_2) = \underbrace{\boxed{\|x - f_{\theta_2}(z)\|_2^2}}_{\text{Recon. Loss}} + \overbrace{\boxed{\beta}}^{\text{Regularization Strength}} \cdot \underbrace{\boxed{D_{KL}(q_{\theta_1}(z \,|\, x) \;\|\; \mathcal{N}(0,I))}}_{\text{KL Penalty}}$$

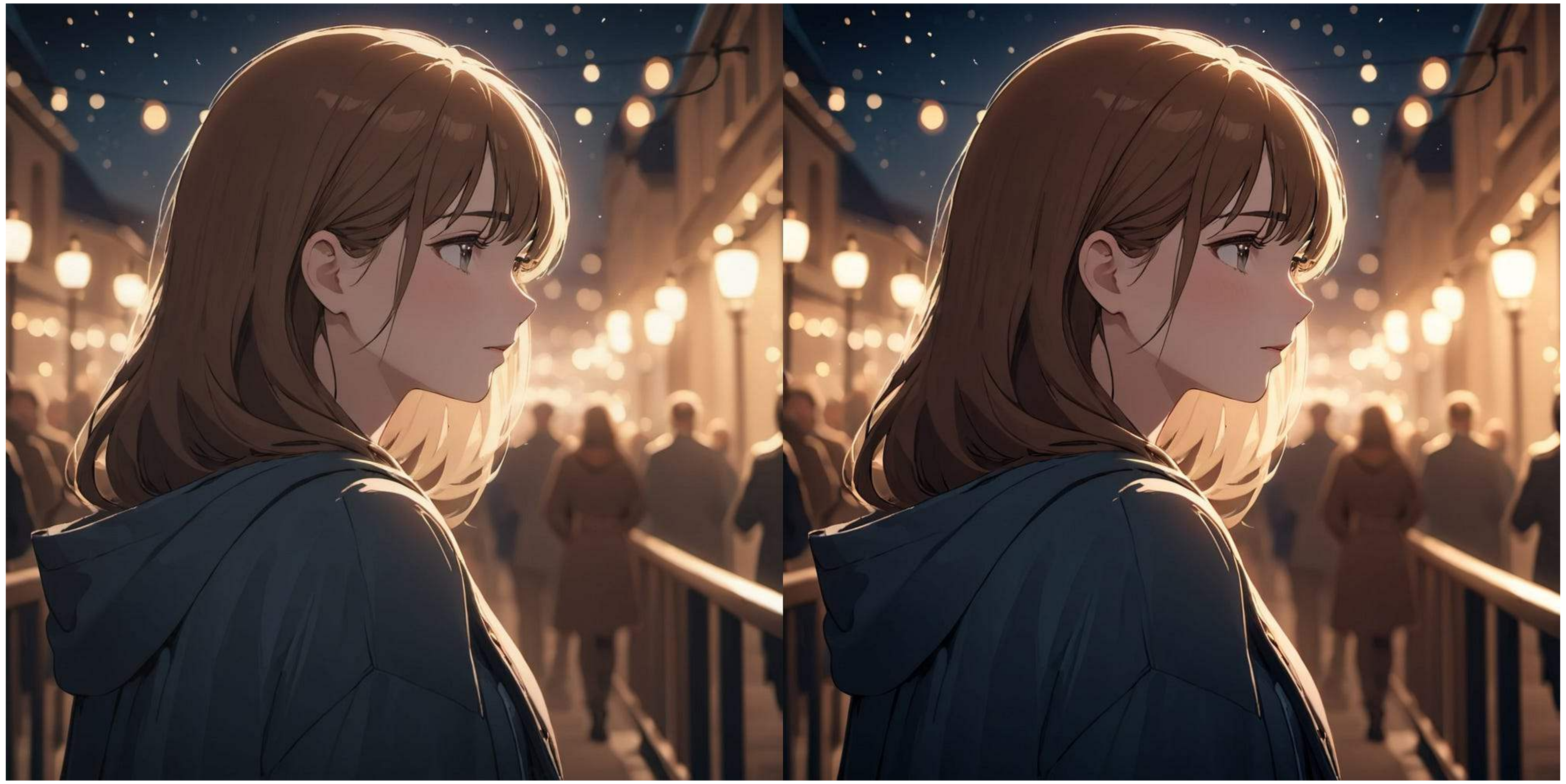
SOTA

SOTA

Stable Diffusion XL's VAE reduces the dimension by 48 times

SOTA

Stable Diffusion XL's VAE reduces the dimension by 48 times



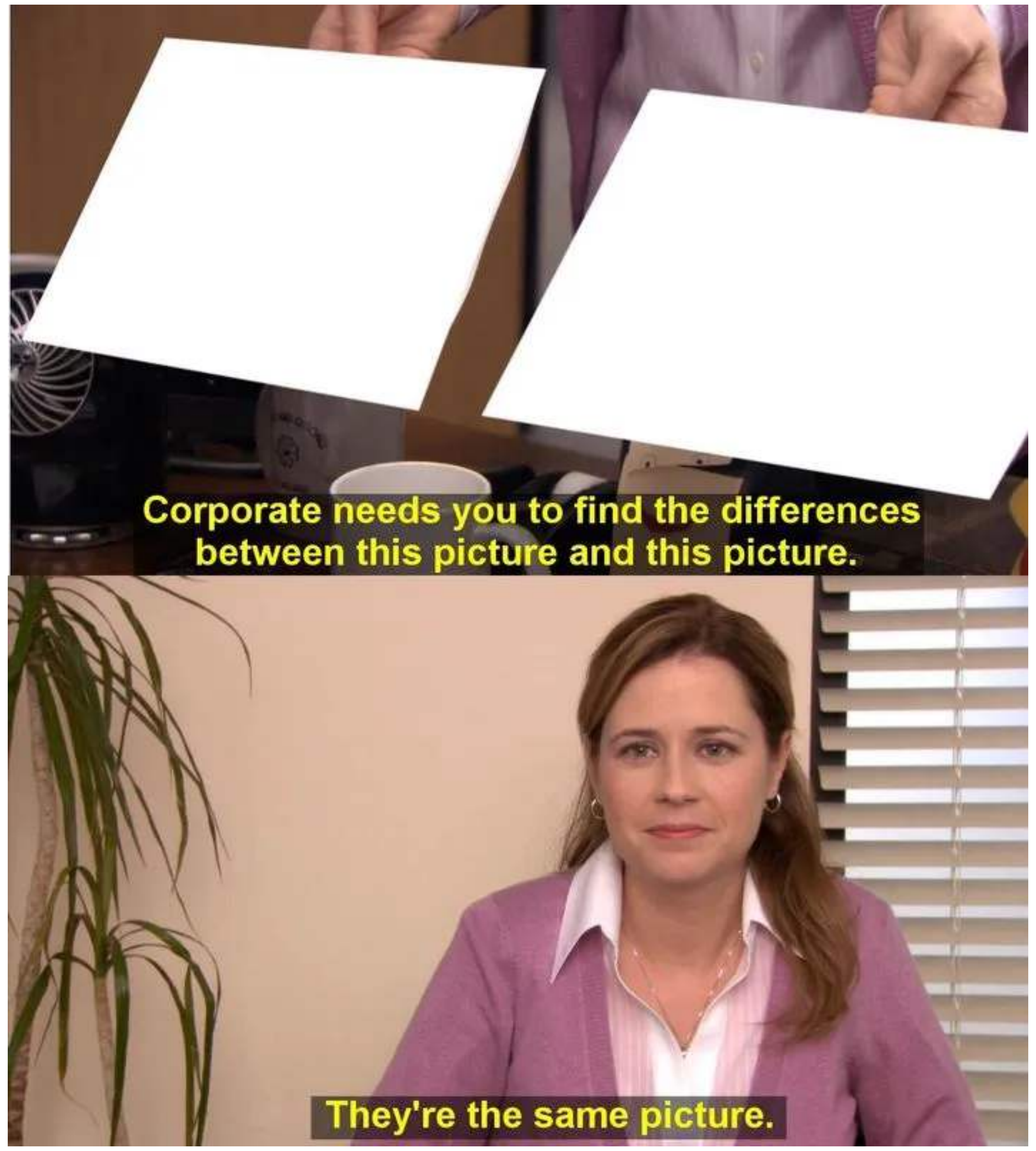
Original Reconstructed

SOTA

Stable Diffusion XL's VAE reduces the dimension by 48 times

Original

Reconstructed



Sequence Modelling

Sequence Modelling

Inputs are sequences $\{x_t\}$ and we wish to model them

Given$\{x_1, \cdots, x_{t-1}\}$ can we predict x_t?

Sequence Modelling

Inputs are sequences $\{x_t\}$ and we wish to model them

Given $\{x_1, \cdots, x_{t-1}\}$ can we predict x_t?

Examples:

- Stock prediction
- Next token prediction (language models!)
- Weather Prediction, etc

Sequence Modelling - Challenges

Sequence Modelling - Challenges

Variable length inputs

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Need to learn temporal order

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Need to learn temporal order

Remembering “context”

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Need to learn temporal order

Remembering “context”

Predictions need to somehow use the entire history

Approach #1: RNNs

Key Idea

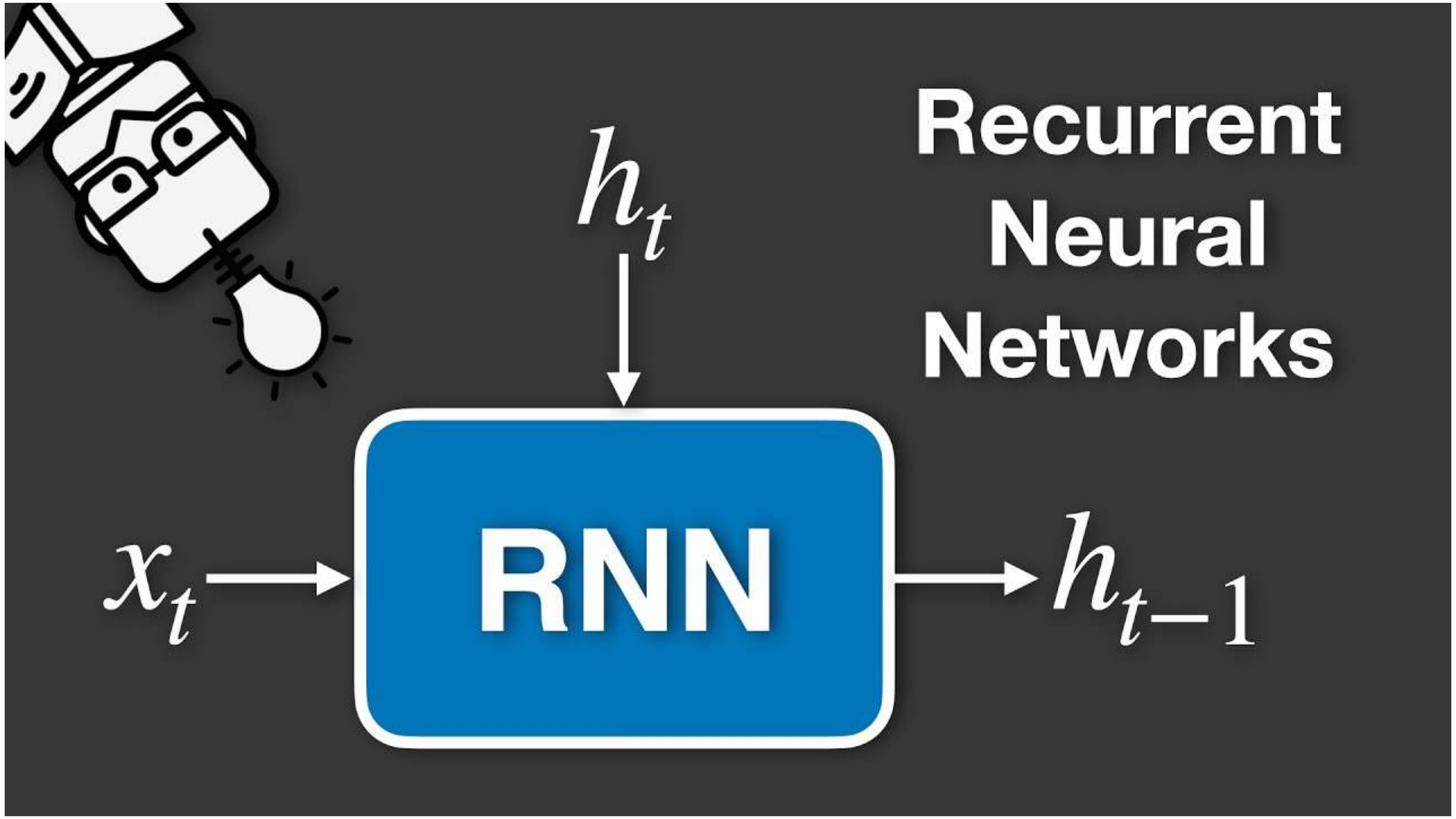
Represent the entire context as a hidden variable $\boldsymbol{h} \in \mathbb{R}^d$

Key Idea

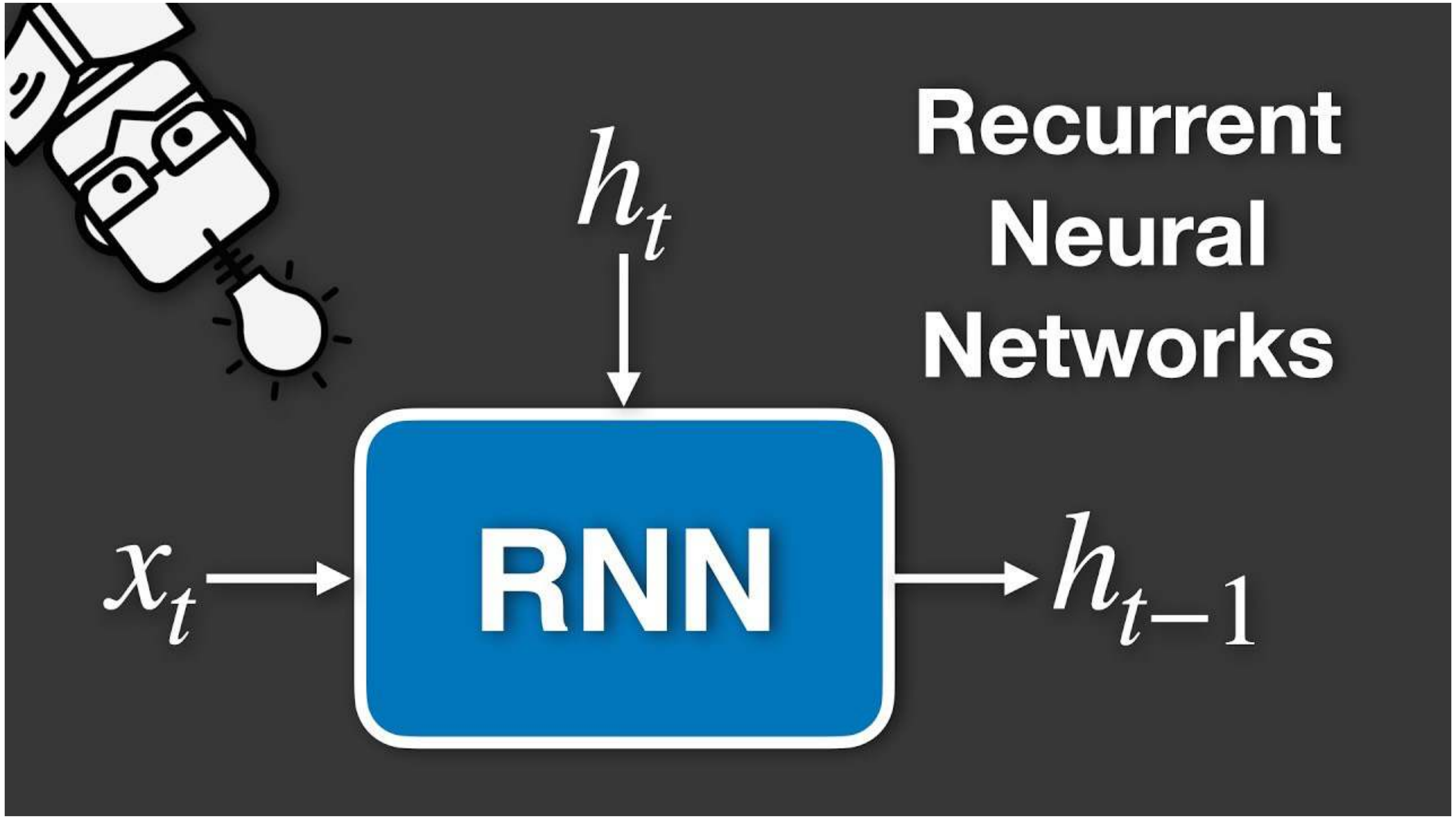
Represent the entire context as a hidden variable $\boldsymbol{h} \in \mathbb{R}^d$

Given the hidden variable, generate the next output

Recurrence?



Recurrence?



Pitfalls

Pitfalls

Forgets history: for a long context, the hidden state contains very little information about the starting tokens

Pitfalls

Forgets history: for a long context, the hidden state contains very little information about the starting tokens

Values and gradients either explode or vanish - unstable

Pitfalls

Forgets history: for a long context, the hidden state contains very little information about the starting tokens

Values and gradients either explode or vanish - unstable

Hard to parallelise - only run sequentially

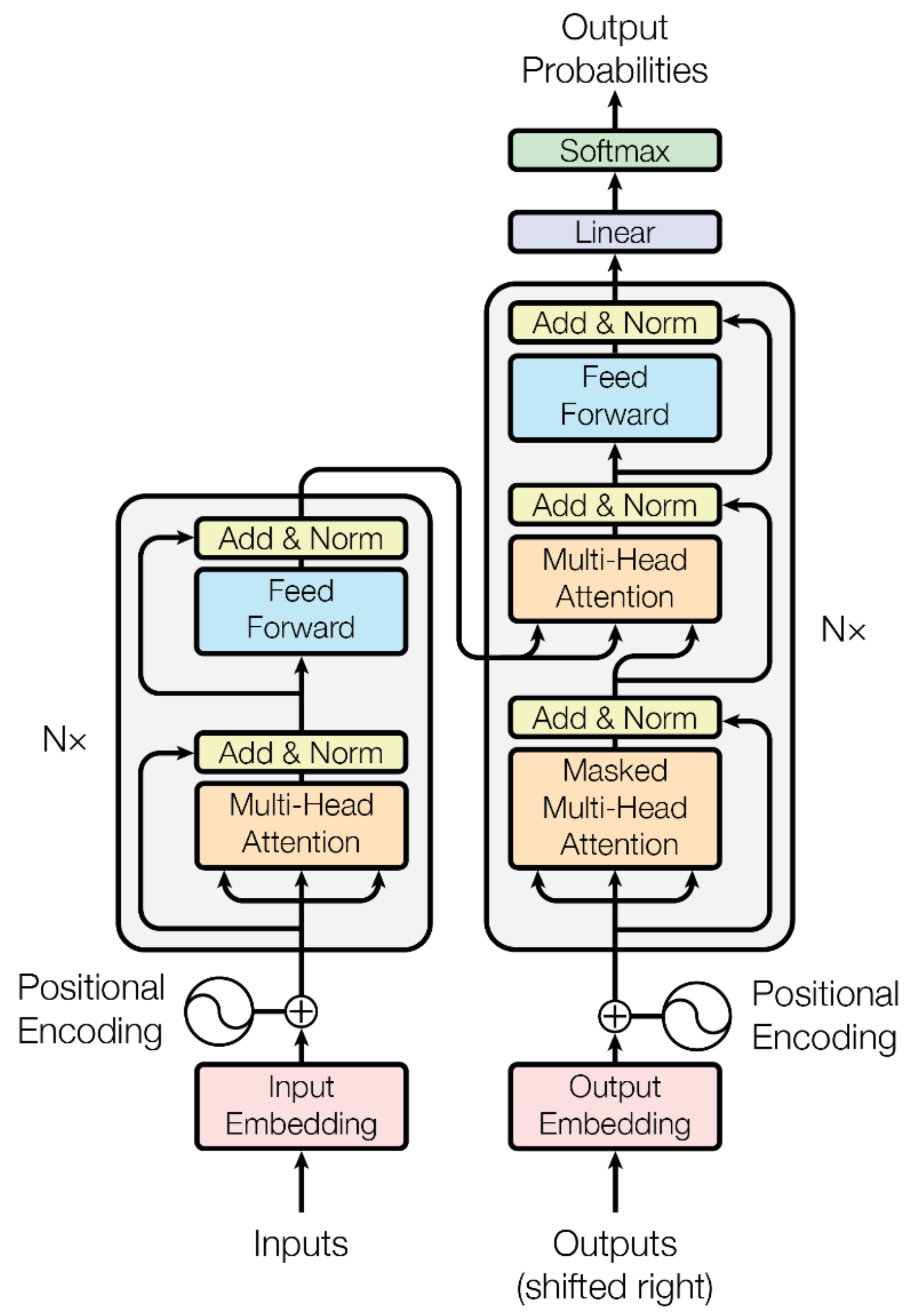
Increases cost of training and inference

What now?

What now?

What now?

Attention Is All You Need



What now?

Attention Is All You Need