Autoencoders

ML Instruction Team, Fall 2022

CE Department
Sharif University of Technology

Mahsa Yazdani
Arian Amani

Let's Think

Let's start with a question!

Any ideas?

How can we combine neural networks and unlabeled data to perform compression ?

The Power Of Unsupervised Learning

- Huge datasets compared to supervised learning (No need for labeling)
- Can find previously unknown patterns in data that are impossible with supervised learning

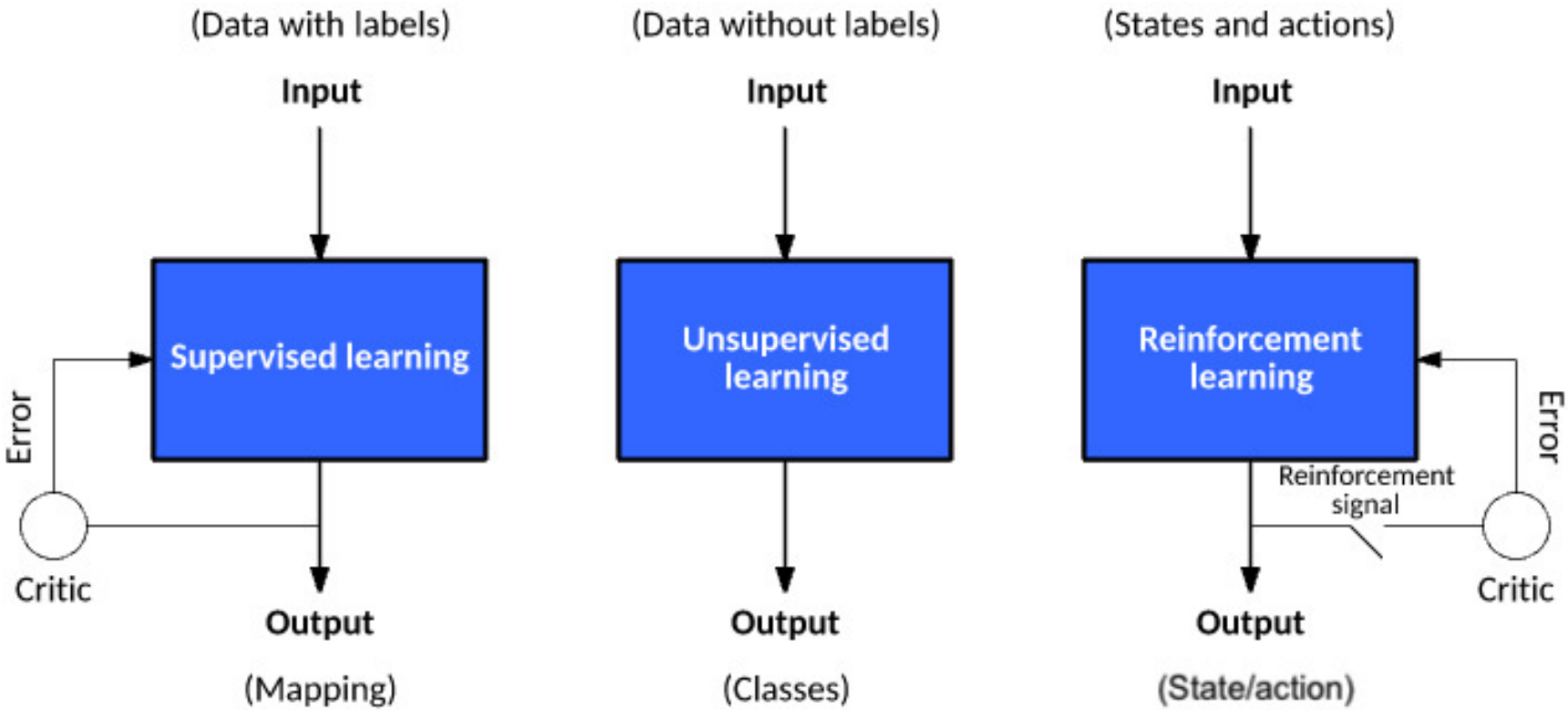
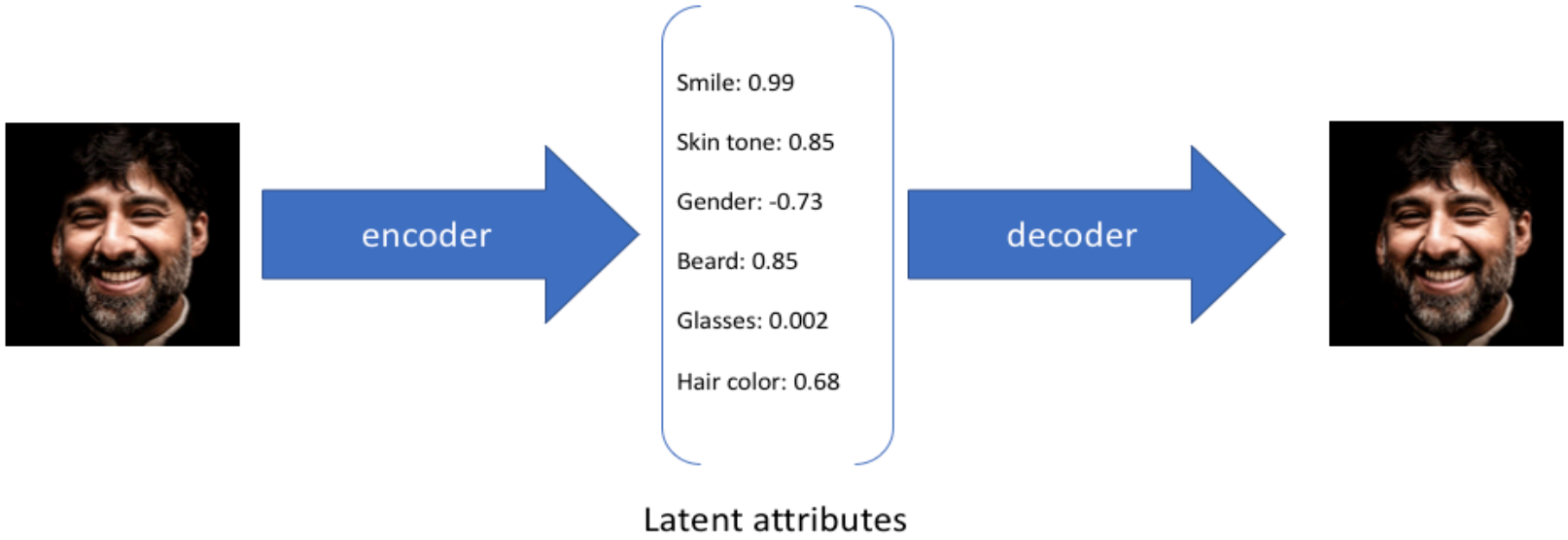


Figure: Types of Machine learning: Deep learning (supervised and unsupervised learning)(Jones [2017]), Source

Applications of Autoencoders

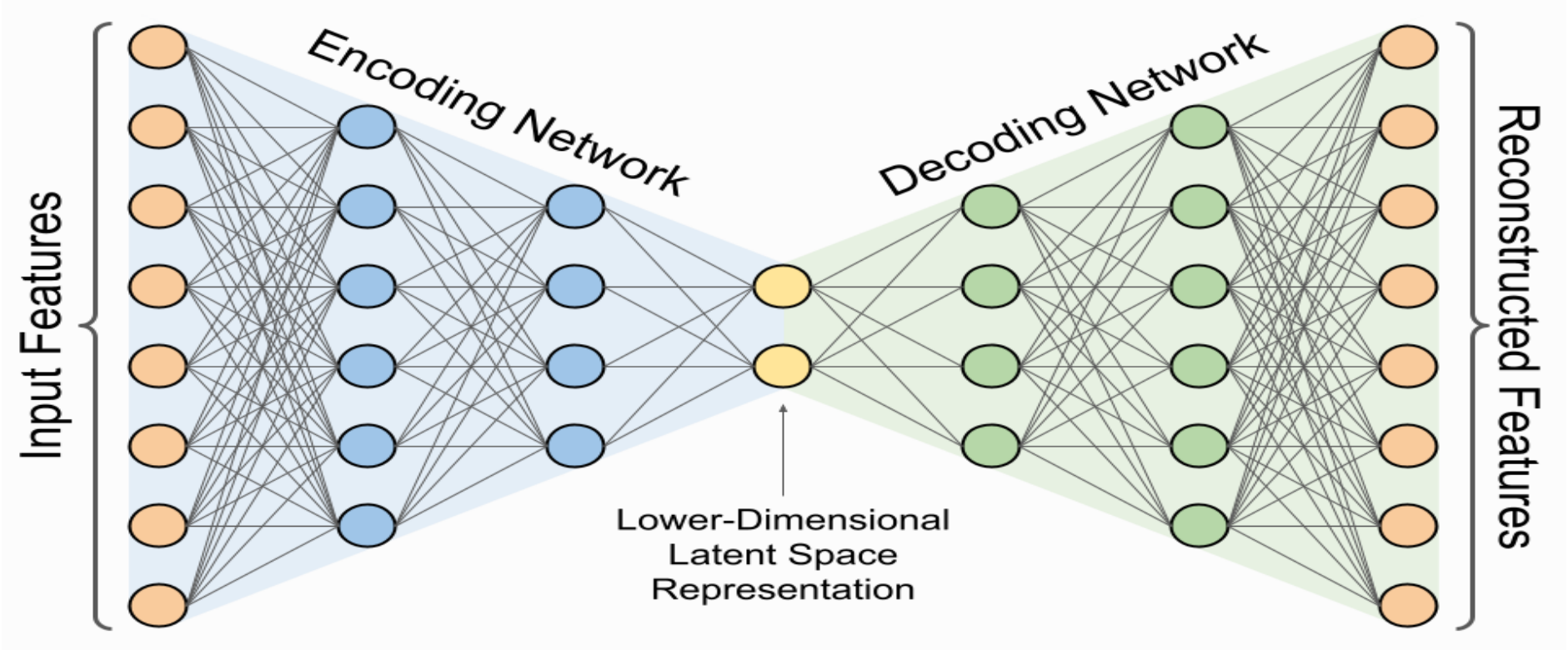
- Compression:
  - AEs can compress our input into a lower dimensional vector and try to reconstruct the original input from that vector



Source

Applications of Autoencoders

- Dimensionality Reduction:
  - AEs can perform dimensionality reduction



Source

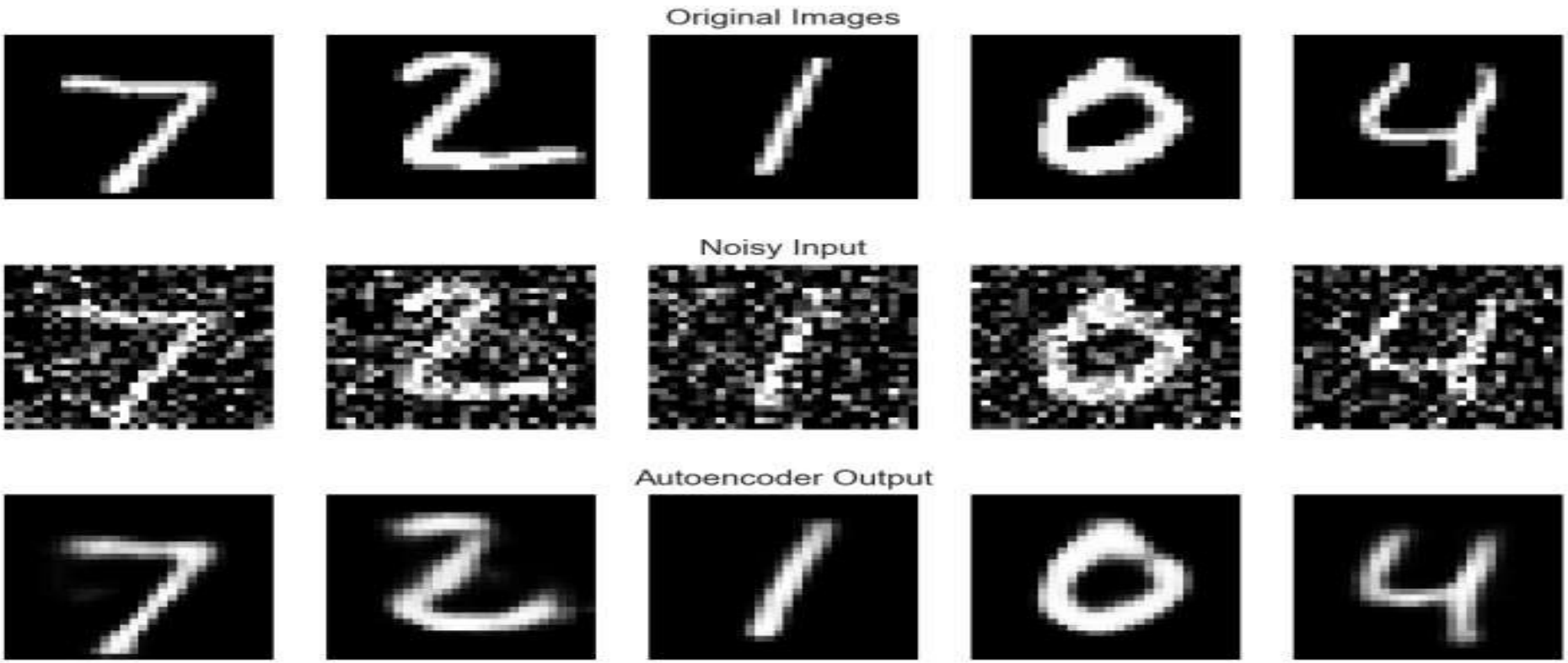
Applications of Autoencoders

- Image coloring and noise reduction

Source

Applications of Autoencoders

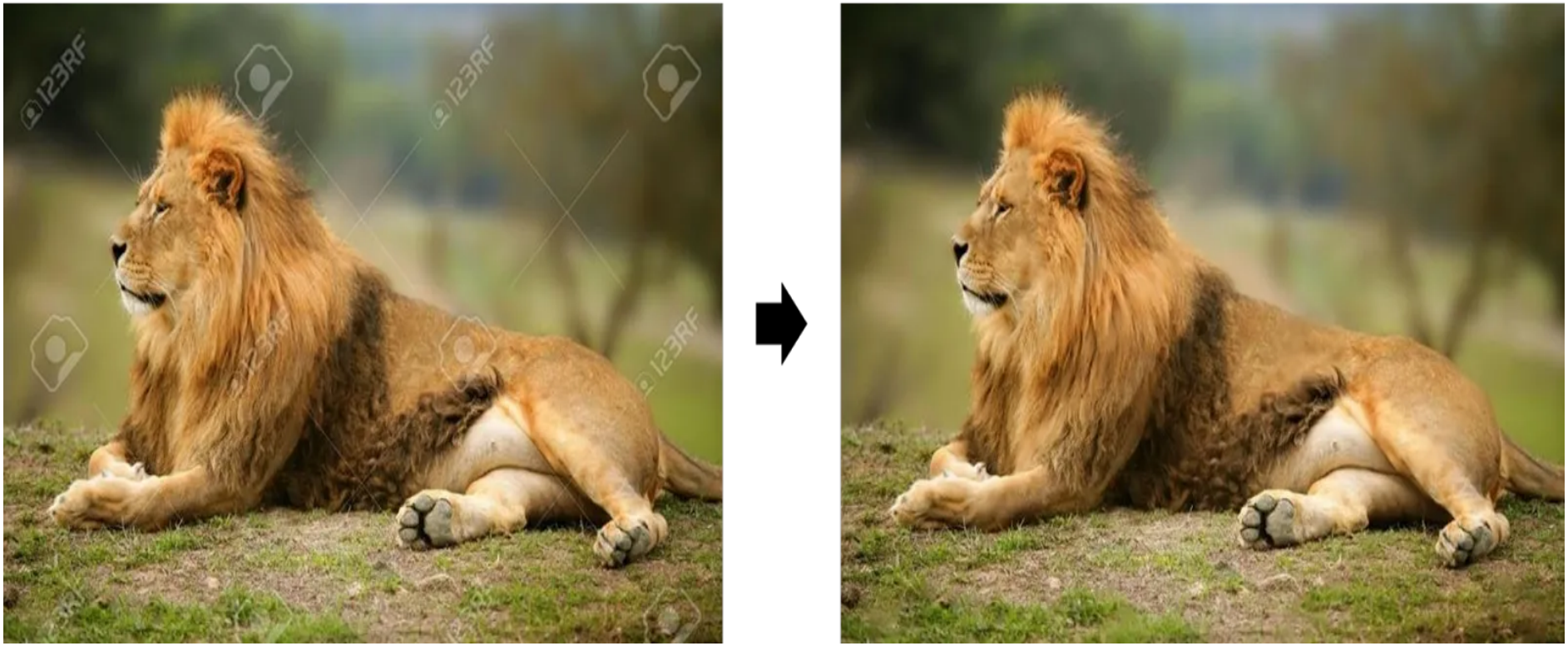
- Noise reduction



Source

Applications of Autoencoders

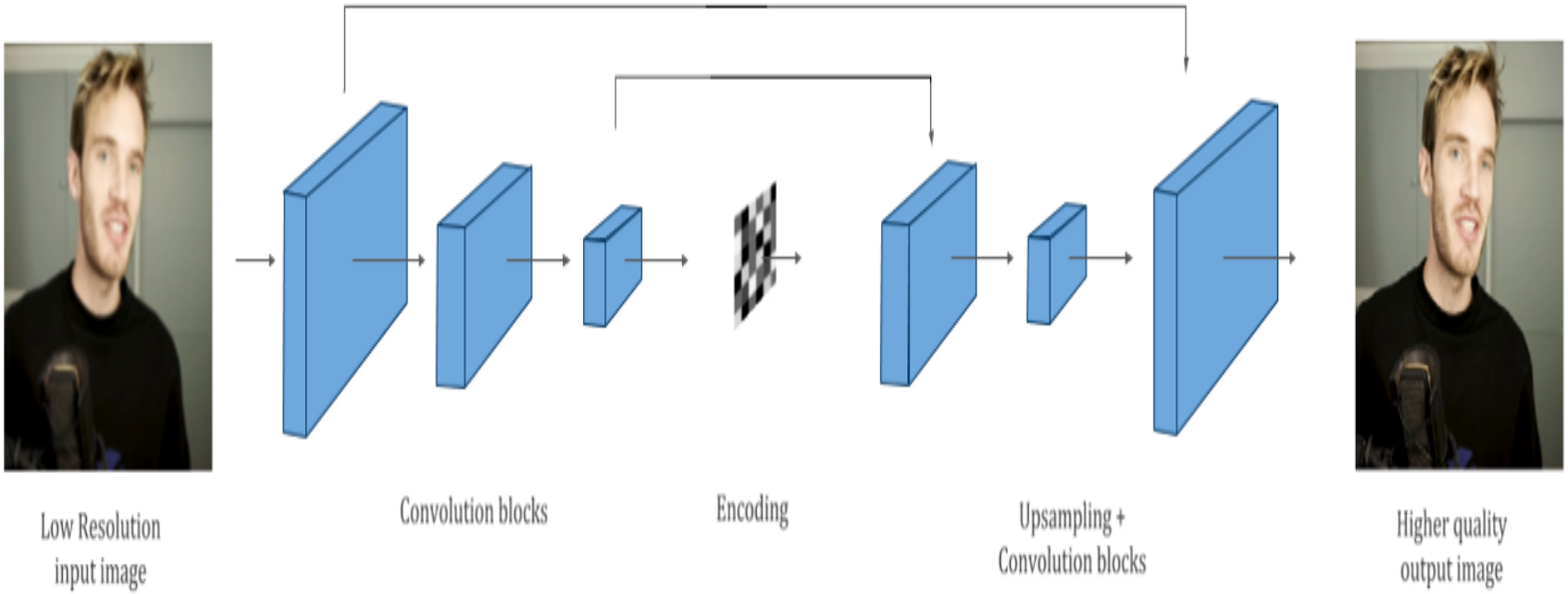
- Watermark removal



Source

Applications of Autoencoders

- Super-Resolution



Source

What is an Autoencoder?

- An autoencoder is a type of artificial neural network, capable of learning a low dimensional representation of the input data (codings), without supervision (unlabeled training data - unsupervised learning)
- Autoencoders take an input X and try to predict X. We use a bottleneck layer with a smaller dimension compared to the input, to use as the coding.

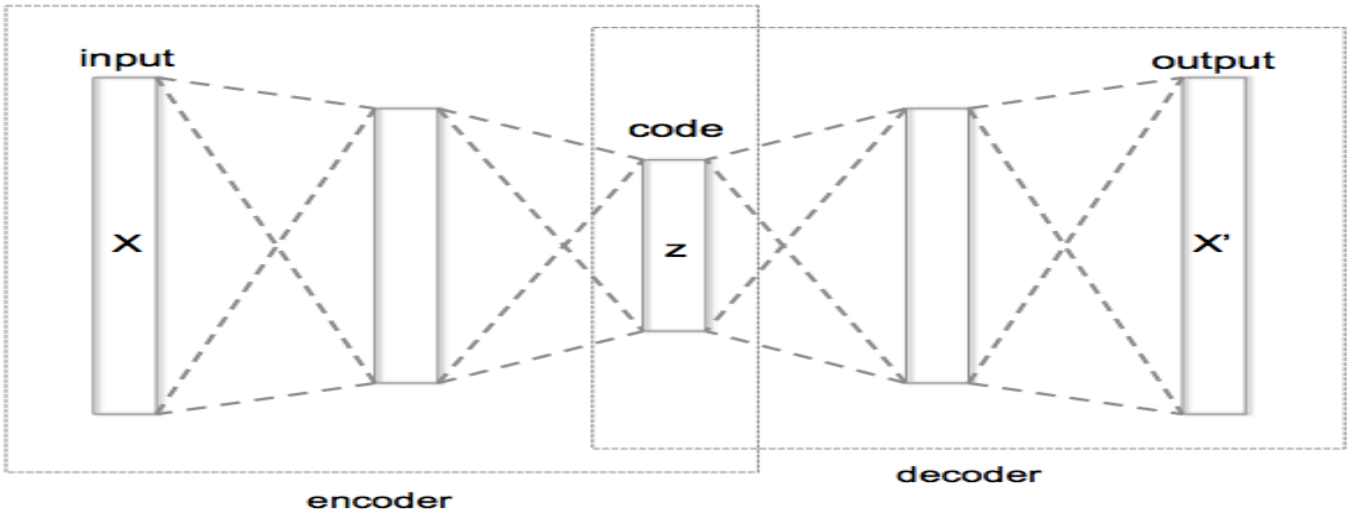


Figure: Schematic structure of an autoencoder with 3 fully connected hidden layers. The code (z - bottleneck) is the most internal layer, Source

Autoencoders: Architecture

- Autoencoders consist of 3 parts:
  1. Encoder : Function $f(x)$ that transforms input x to the latent variable z
  2. Bottleneck : Single layer of neurons that represent the encoding of our input, therefore the most important part of our model
  3. Decoder : Function $h(z)$ that tries to reconstruct input x from encoded latent variable $z \rightarrow h(z) = h(f(x)) = x'$

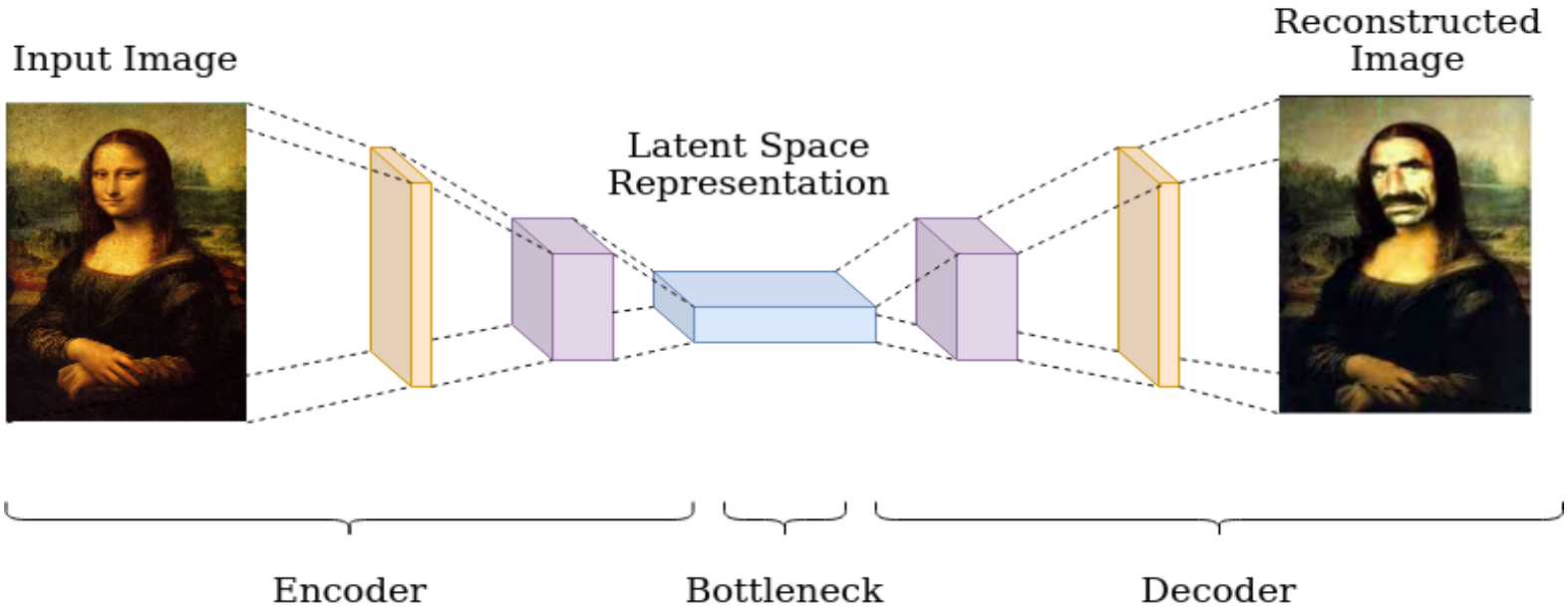


Figure: Autoencoder Architecture (with a little joke :)), Source

Linear vs. Non-Linear

- If your AE uses only linear activations and MSE loss function:
  - It will be performing PCA (Principal Component Analysis)
  - So we need non-linearity to get the most out of autoencoders

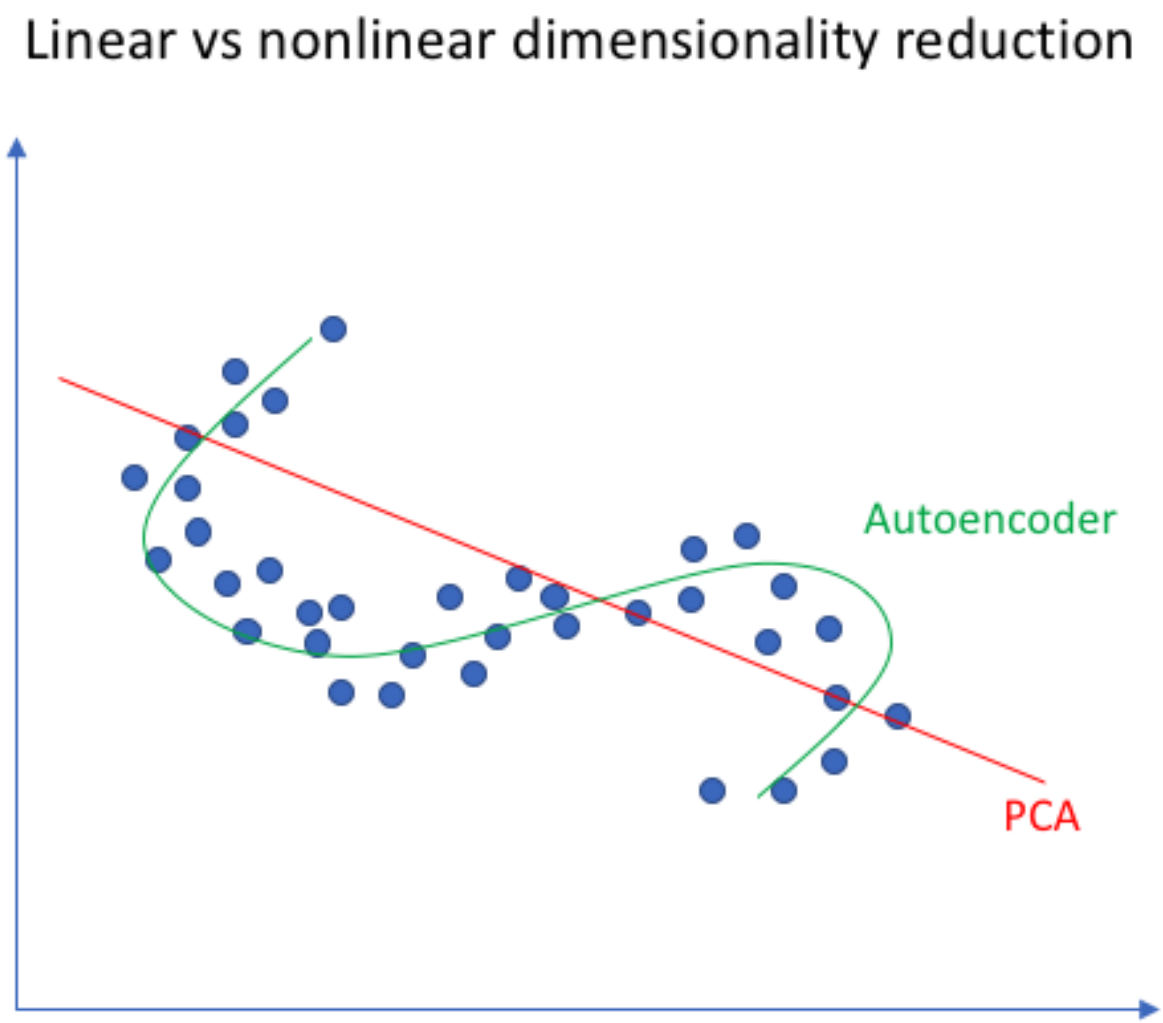
```
encoder = tf.keras.models.Sequential([tf.keras.layers.Dense(2,
    input_shape=input_shape)])

decoder = tf.keras.models.Sequential([tf.keras.layers.Dense(
    input_shape, input_shape=[2])])

pca_autoencoder = tf.keras.models.Sequential([encoder, decoder
    ])

pca_autoencoder.compile(loss="mse", optimizer=tf.keras.
    optimizers.SGD())
```

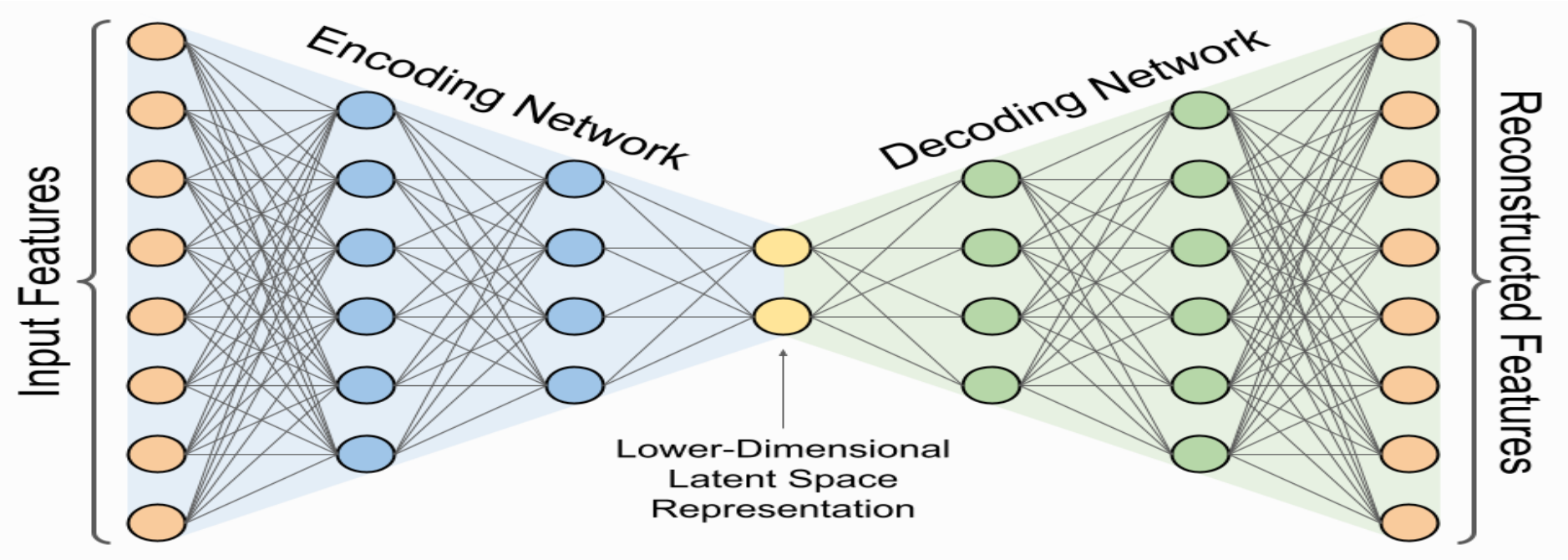
Linear vs. Non-Linear



Source

Stacked Autoencoders

- Autoencoders can have multiple hidden layers to learn more complex encoding/decoding functions - deep autoencoders
- Although, using too deep networks, can cause overfitting
  - Your model will just memorize points in the coding space for each training data, instead of learning a good latent representation of them



Source

Loss and Training

- We're trying to reconstruct our input
- Common loss functions for training autoencoders are:
  - L2: $loss(x, x') = \sum_{i=1}^{m} (x^{(i)} - x'^{(i)})^2$
  - Cross-Entropy

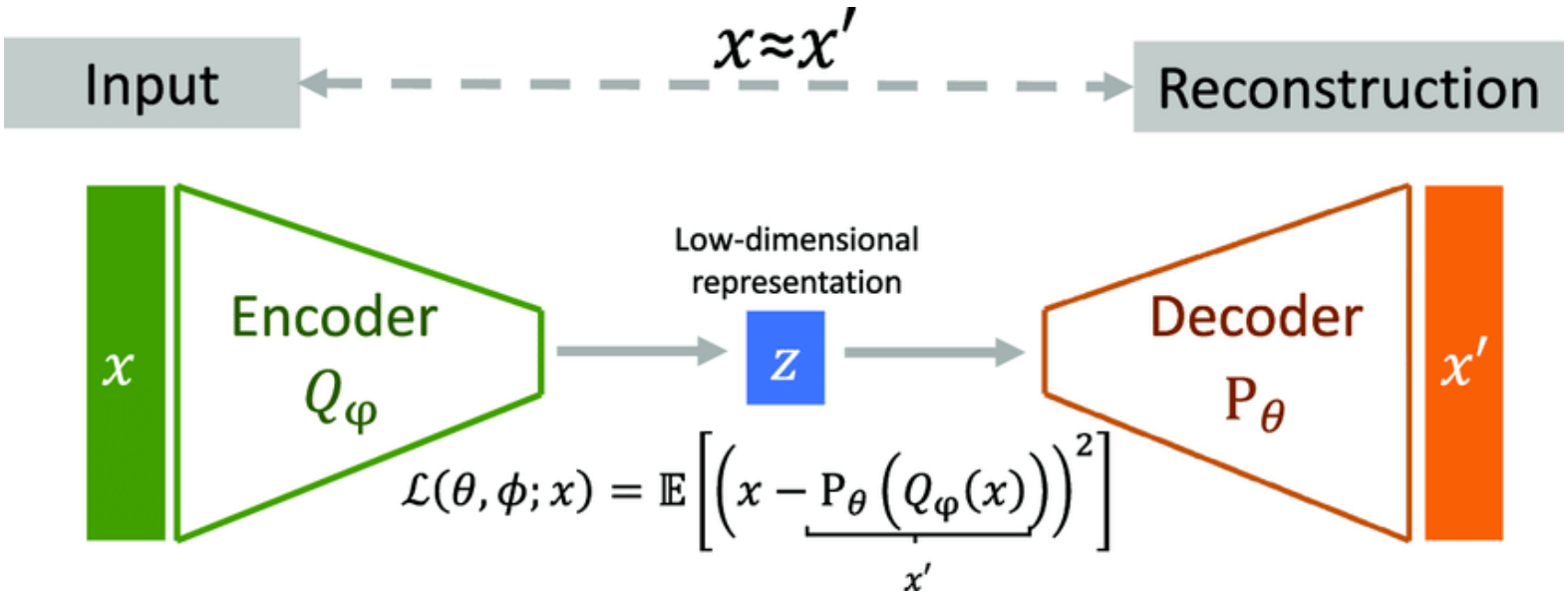


Figure: Schematic of an autoencoder architecture with mean-squared error reconstruction loss.[1]

Pretraining

- You can use an autoencoder for pretraining on supervised problems with few labeled data

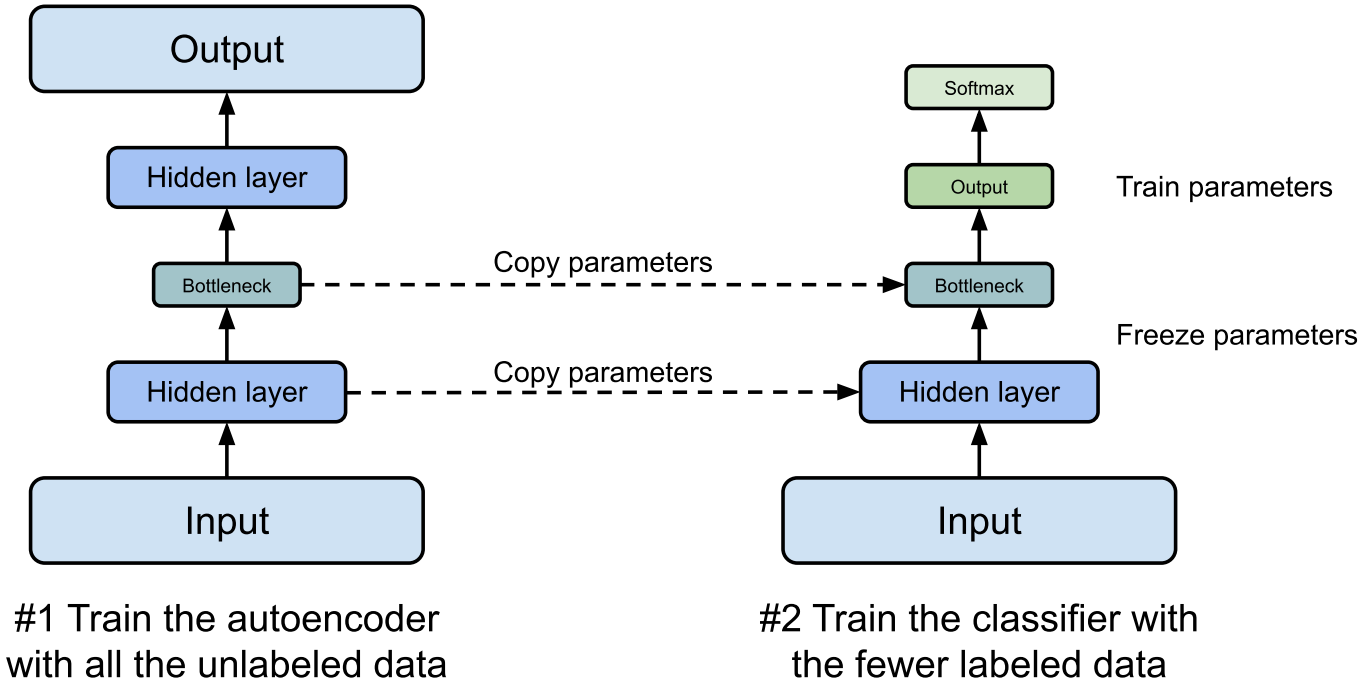
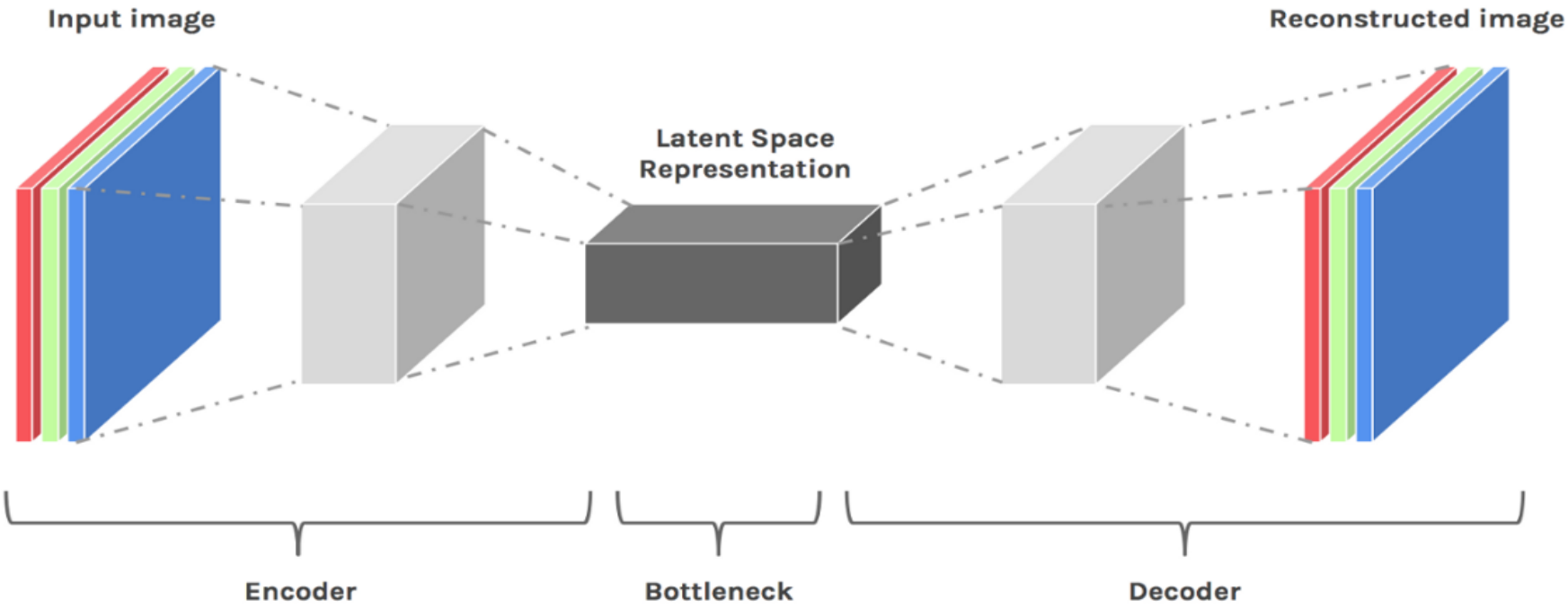


Figure: Using unsupervised learning for pretraining with autoencoders

Autoencoders & Images

- **Are normal autoencoders suitable for working with images?**
  - Encoder: A regular CNN
  - Decoder: A network that uses transpose convolutional layers (or upsampling layers with convolution layers)



Convolutional Encoder-Decoder architecture

Source

Denoising Autoencoders (DAE)

- An AE learns the identity function which brings the risk of overfitting
- A DAE forces the model to learn a more robust latent representation
- It can also be used to efficiently remove noise from noisy images

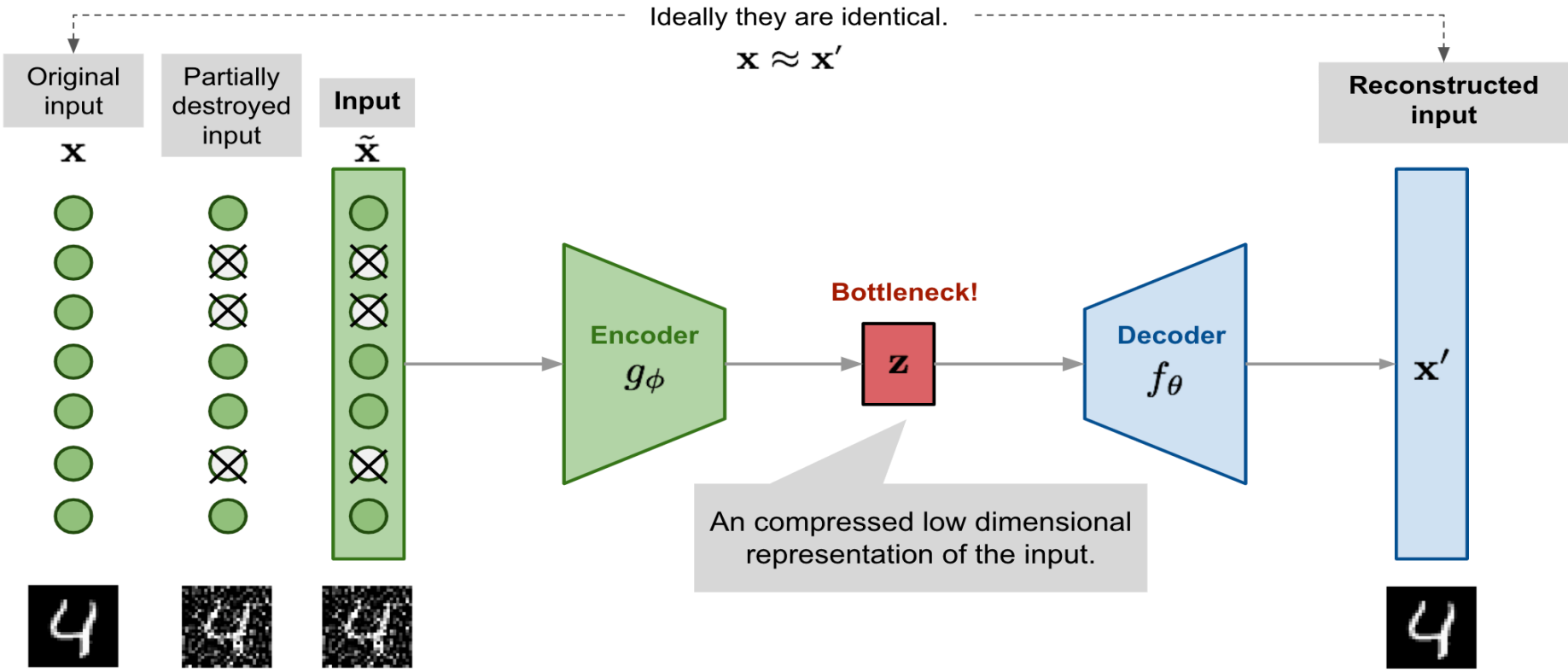
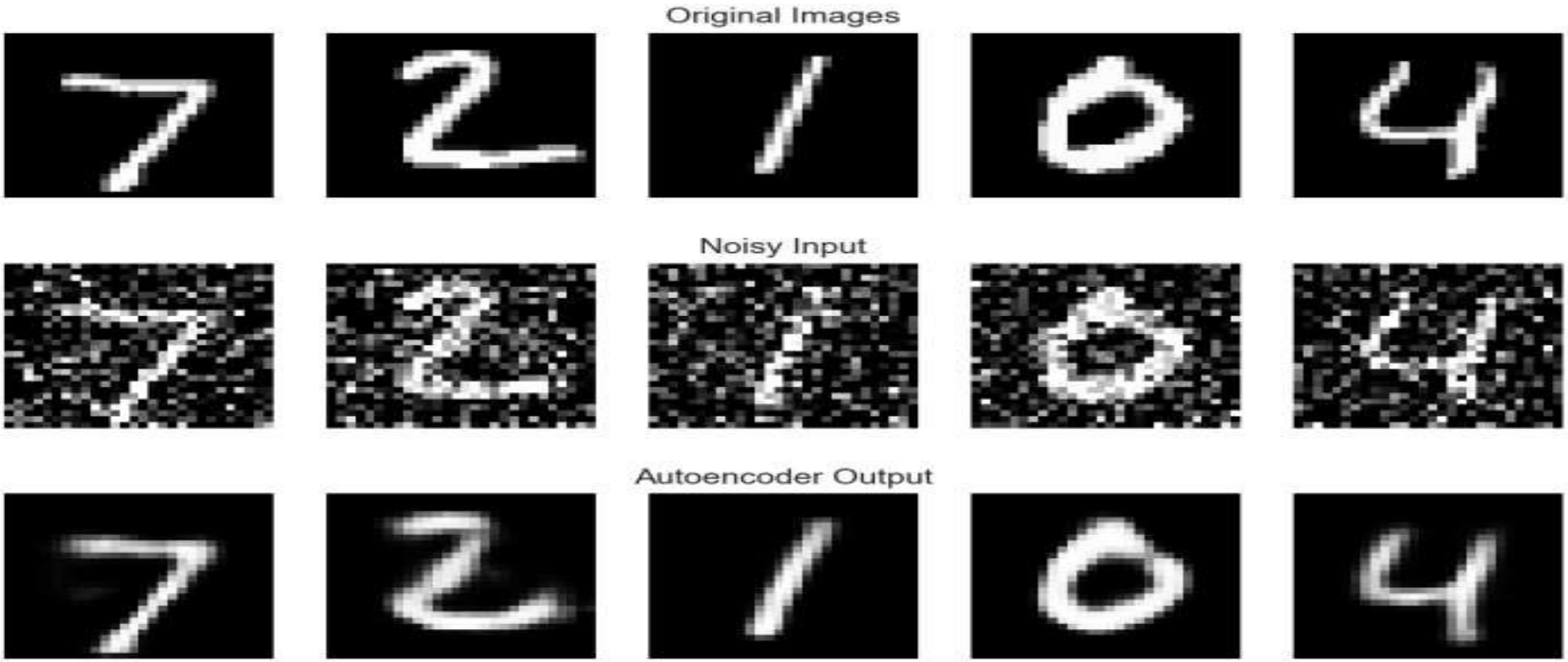


Figure: Illustration of denoising autoencoder model architecture, Source

Denoising Autoencoders

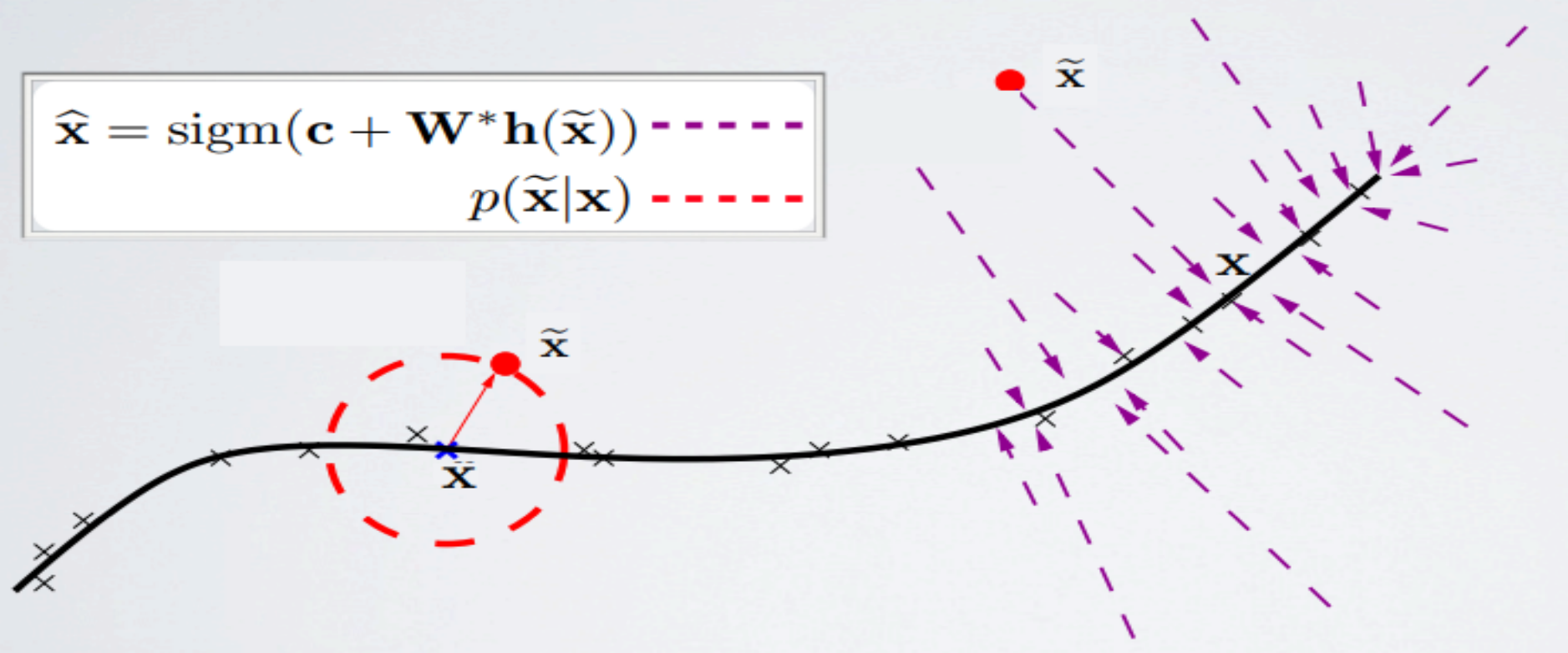
- Results of an autoencoder, predicting what the original data was, without even seeing it



Source

Denoising Autoencoders

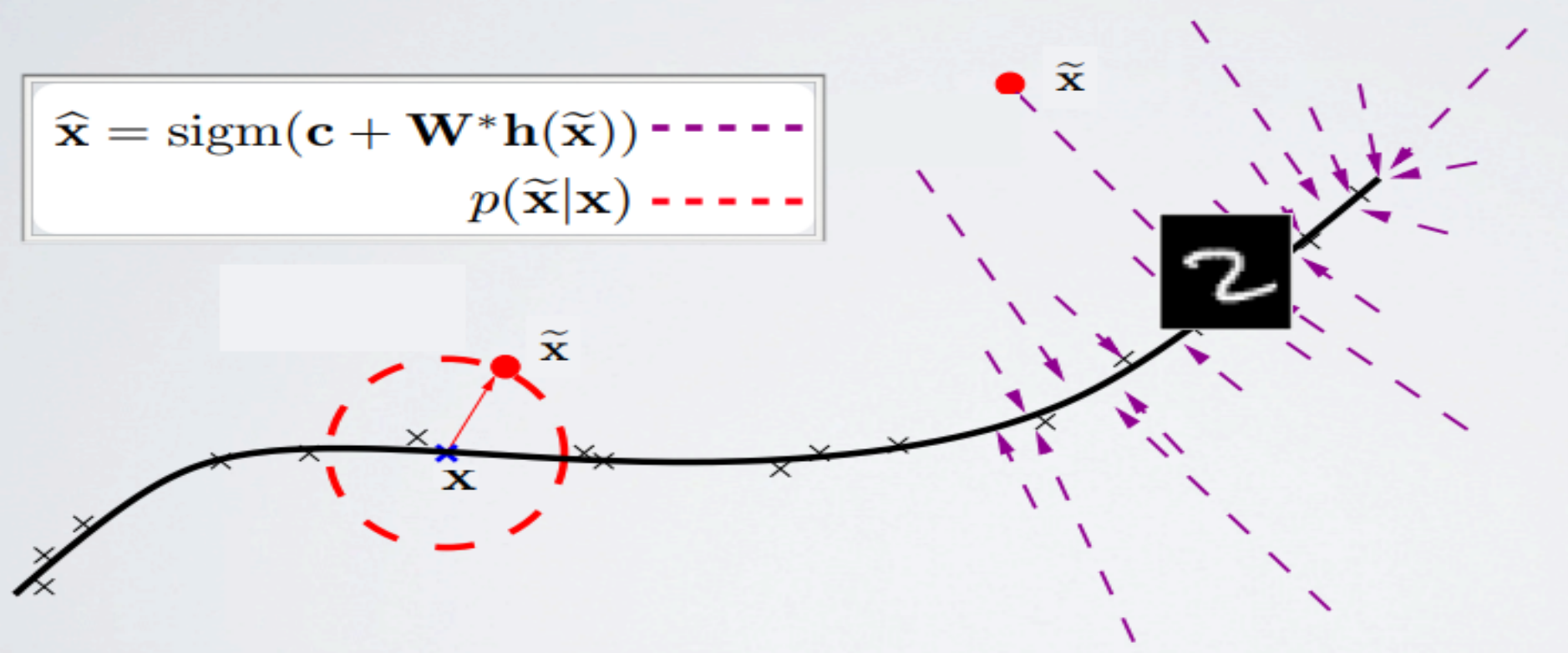
- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data (Refrence)



Source

Denoising Autoencoders

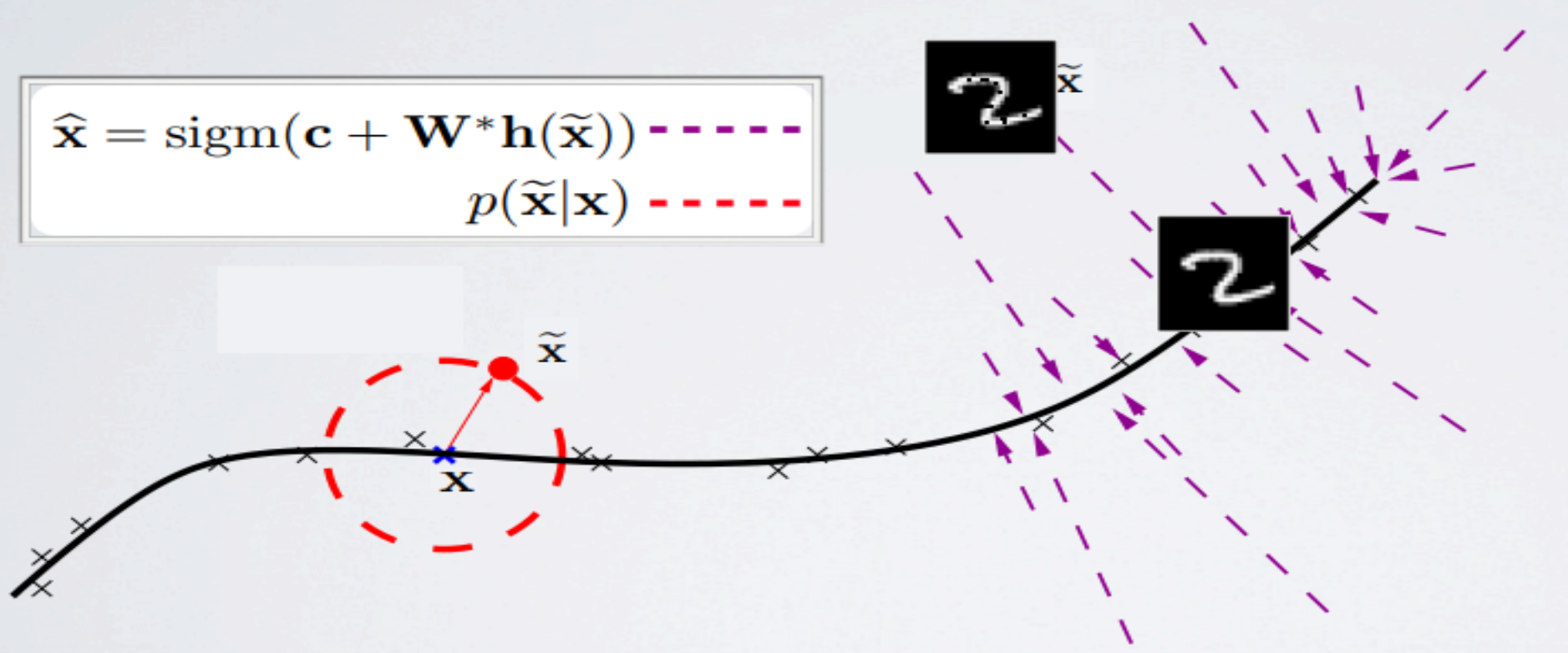
- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data (Refrence)



Source

Denoising Autoencoders

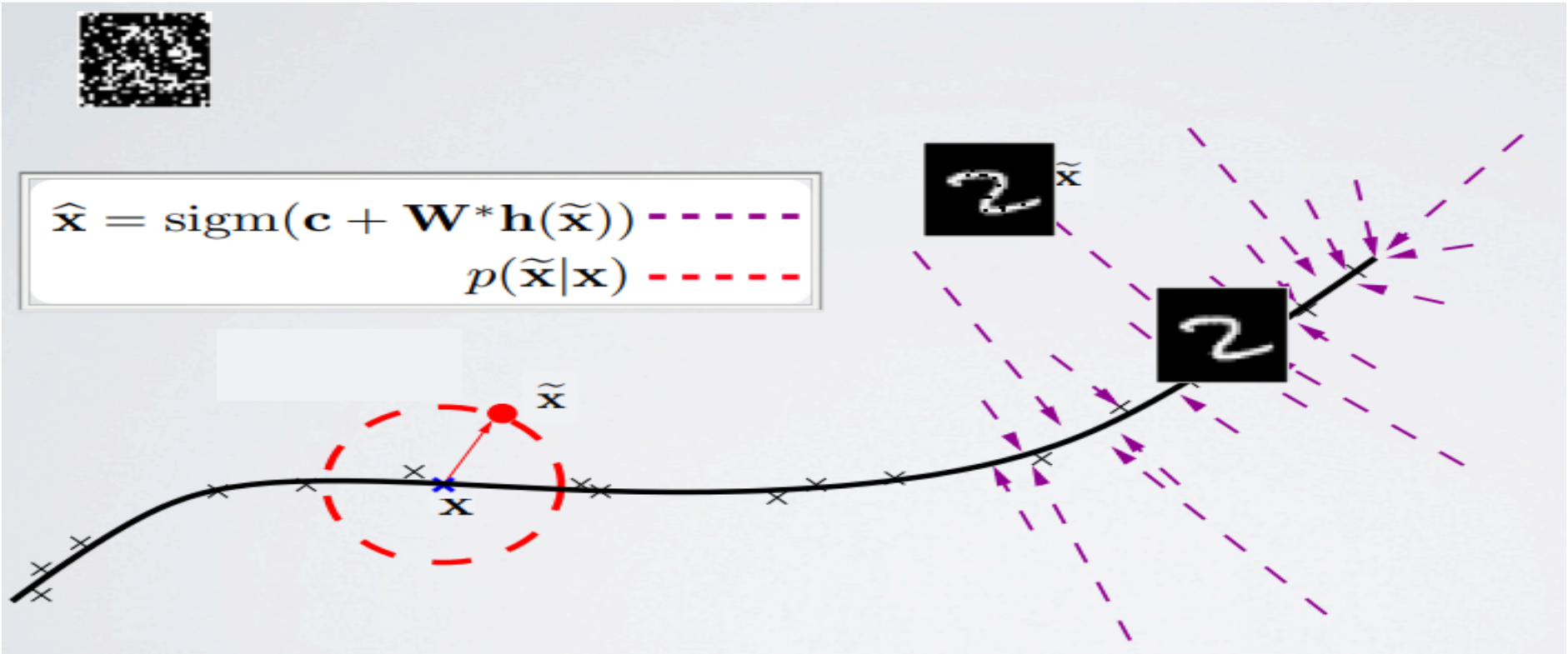
- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data (Refrence)



Source

Denoising Autoencoders

- DAEs won't simply memorize the inputs and outputs (More robustness)
- Intuitively, a DAE learns a projection from a neighborhood of our training data back onto the training data (Refrence)



Source

Autoencoder Generative Models

How can we generate NEW data with Autoencoders??

hint: Autoencoder learns the feature space!

Walking through an example

- We can try to decode a random point from the latent space to generate new data
- The freedom of AEs leads to a severe overfitting which results in some points of the latent space being meaningless once decoded

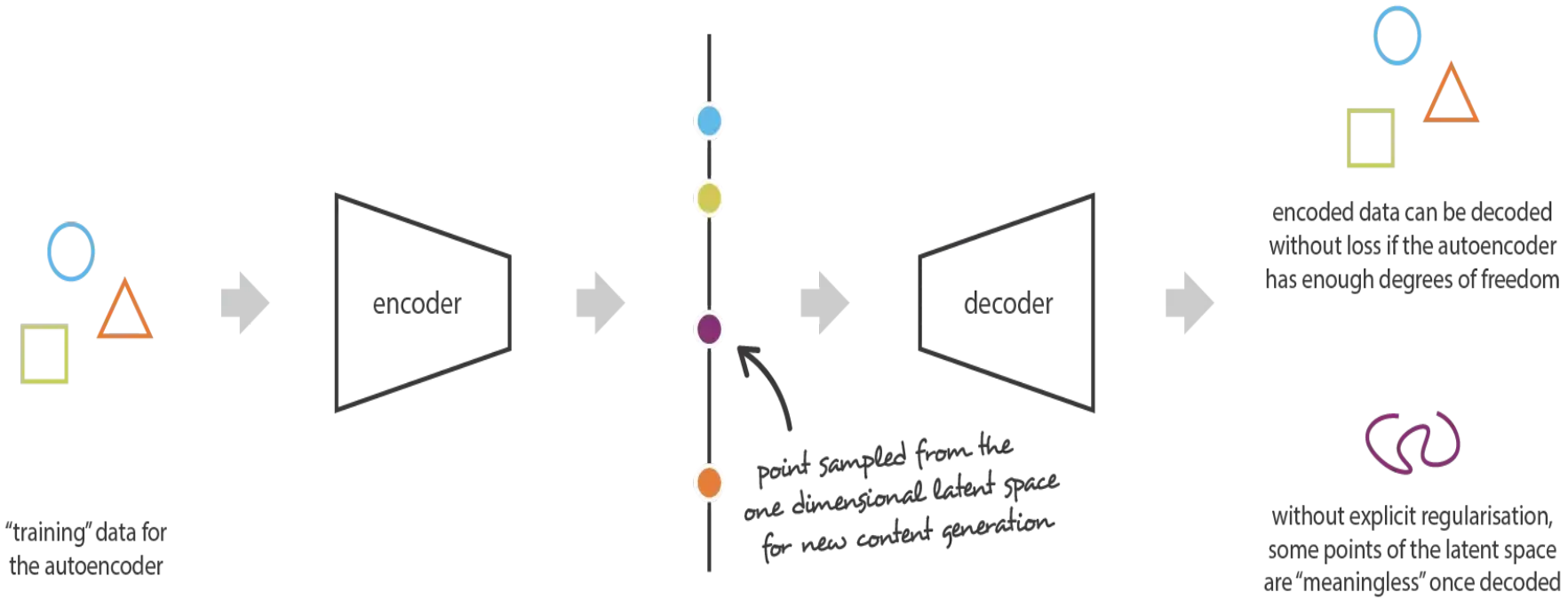


Figure: Irregular latent space prevent us from using autoencoder for new content generation, Source

Walking through an example

- Not all of the points in latent space have meaningful reconstructions

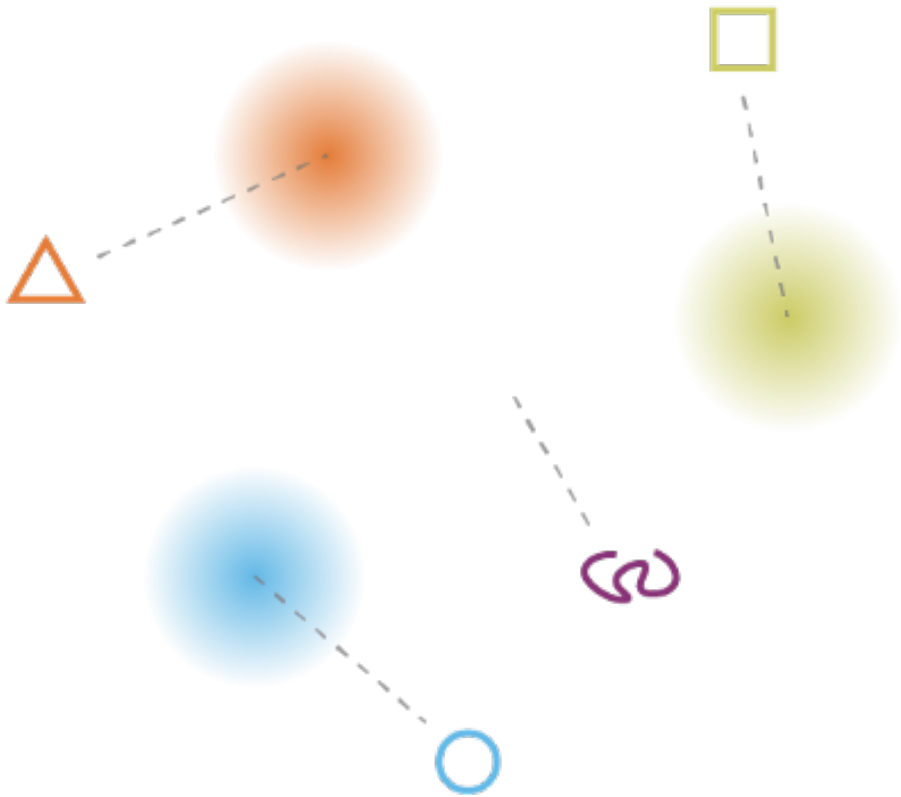


Figure: Latent space and the decoded outputs for some points in it, Source

Walking through an example

- What we want is something like the following picture. So that with sampling from the latent space, we can generate new shapes.

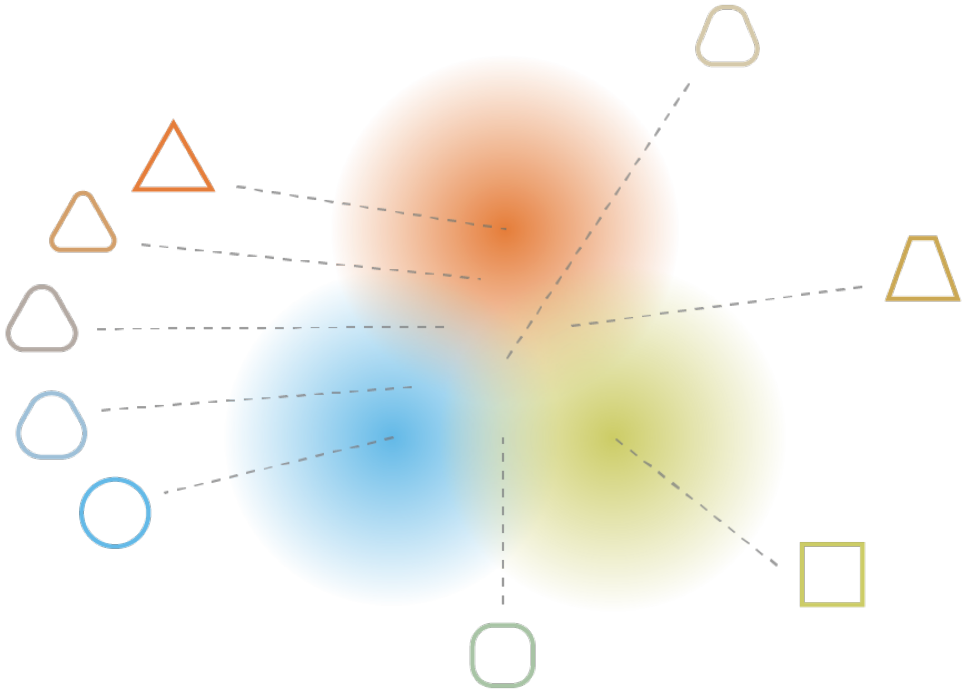


Figure: Regularisation tends to create a "gradient" over the information encoded in the latent space, Source

Variational Autoencoders

Ideas for solving this problem?

Variational Autoencoders

- Instead of encoding the input into a single point (the latent representation), we encode it into a distribution over the latent space
- Then we can sample our latent representation from the latent distribution

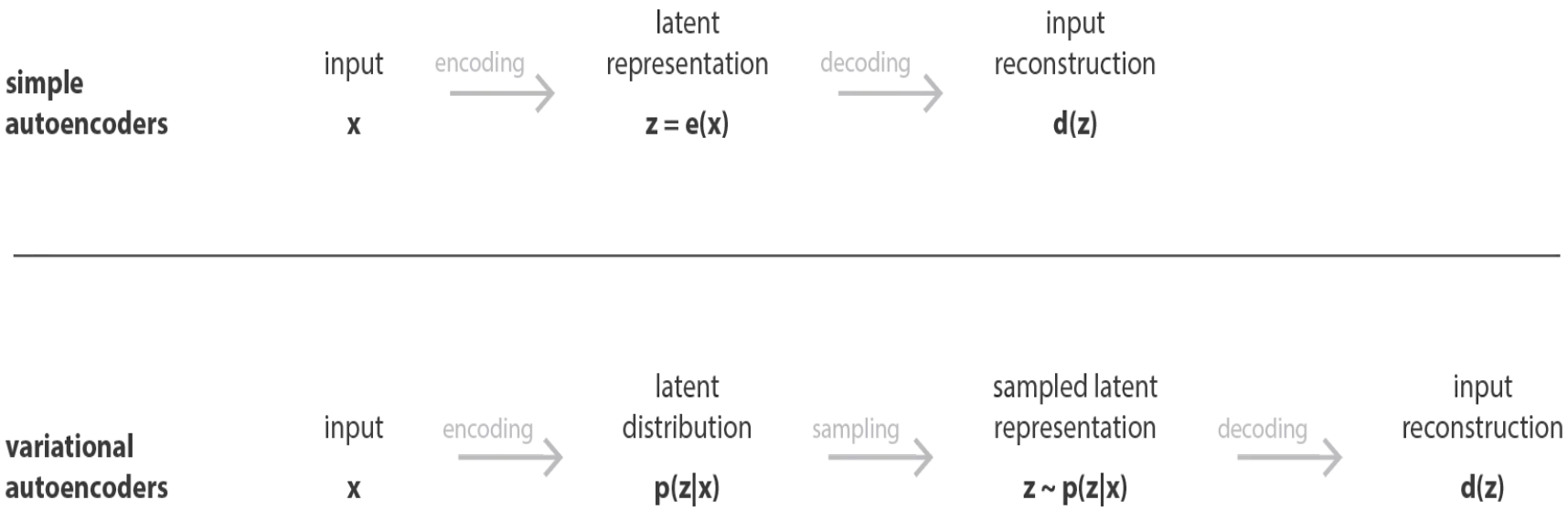
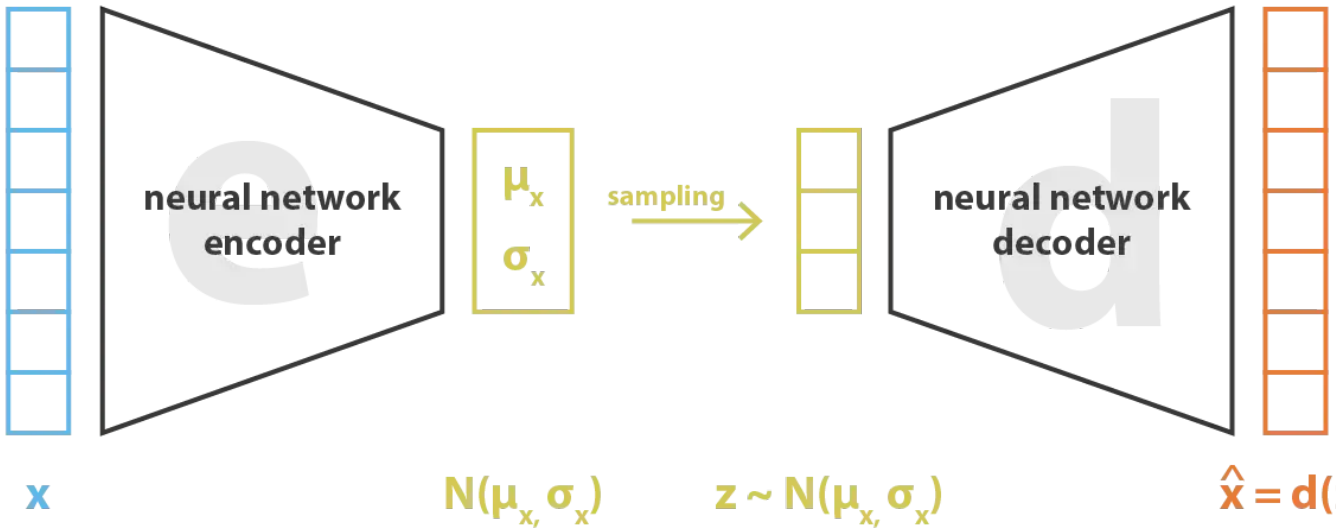


Figure: Difference between autoencoder (deterministic) and variational autoencoder (probabilistic), Source

Variational Autoencoders

- We will be encoding into, and sampling from normal distributions
- So our encoder needs to return the mean and the variance matrix, describing these Gaussians

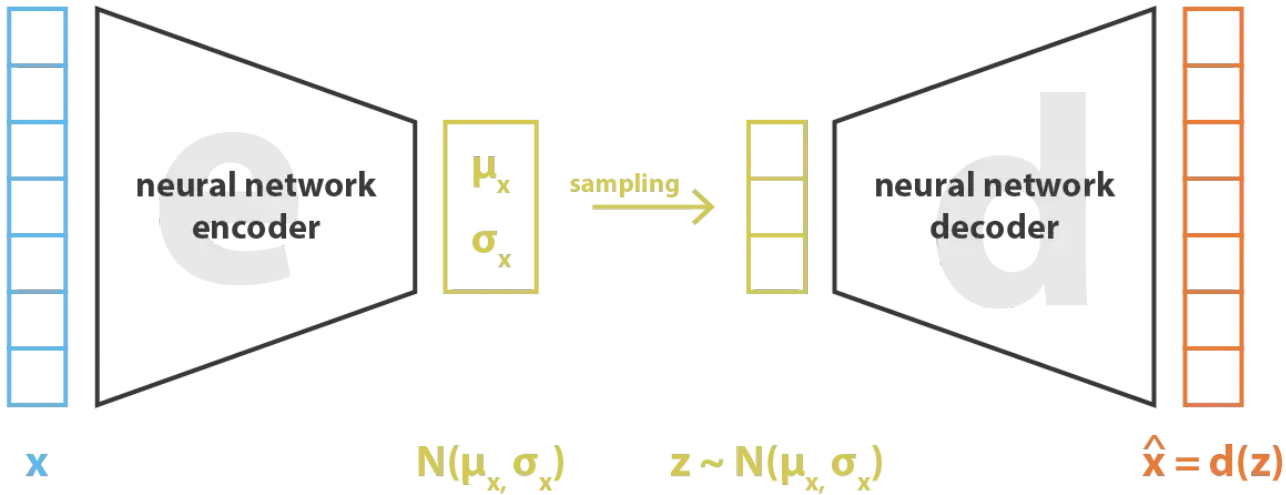


$$\text{loss} = ||x - \hat{x}||^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Figure: In variational autoencoders, the loss function is composed of a reconstruction term (that makes the encoding-decoding scheme efficient) and a regularization term (that makes the latent space regular), Source

Variational Autoencoders

- Encoding as distributions, will make it possible to express a good latent space regularization:
  - The distribution returned will be forced to be close to the standard normal distribution (Mean=0, Var=1) $\rightarrow \mathcal{N}(0, 1)$

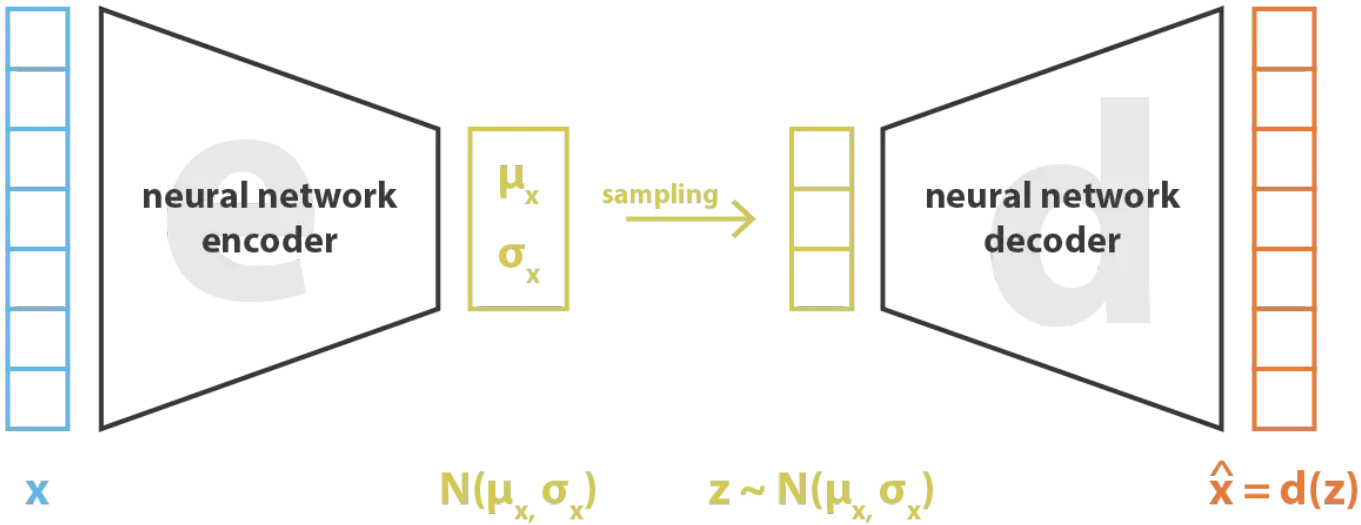


$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Figure: In variational autoencoders, the loss function is composed of a reconstruction term (that makes the encoding-decoding scheme efficient) and a regularisation term (that makes the latent space regular), Source

Variational Autoencoders

- The regularization term is expressed as the Kulback-Leibler divergence between the returned distribution and a standard Gaussian:
  - $KL(\mathcal{N}(\mu_x, \sigma_x)||\mathcal{N}(0, 1))$



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

Figure: In variational autoencoders, the loss function is composed of a reconstruction term (that makes the encoding-decoding scheme efficient) and a regularisation term (that makes the latent space regular), Source

Effect of regularization

- This regularity brings two main properties:
  - Continuity : Two close points in the latent space should not give two completely different contents once decoded
  - Completeness : For a chosen distribution, a point sampled from the latent space should give "meaningful" content once decoded

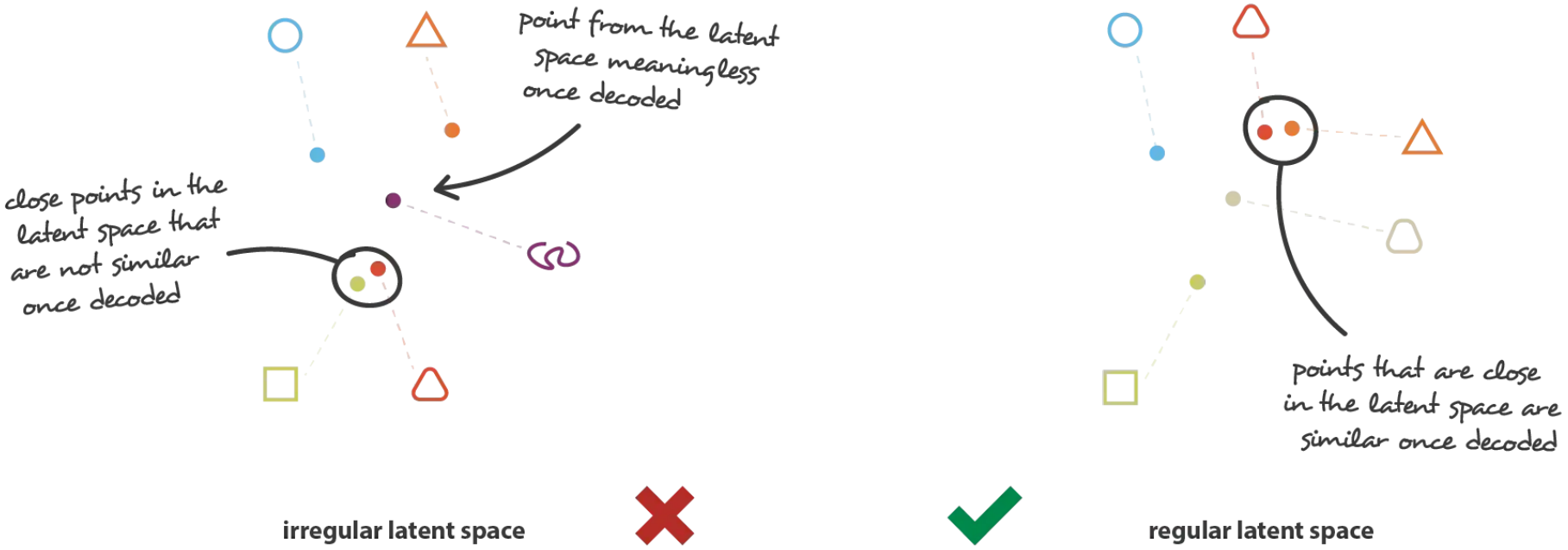


Figure: Difference between a "regular" and an "irregular" latent space, Source

In practice

- We will use two networks at the end of our encoder to calculate the mean and variance

$$g(x) = g_2(g_1(x)) \qquad h(x) = h_2(h_1(x)) \qquad g_1(x) = h_1(x)$$

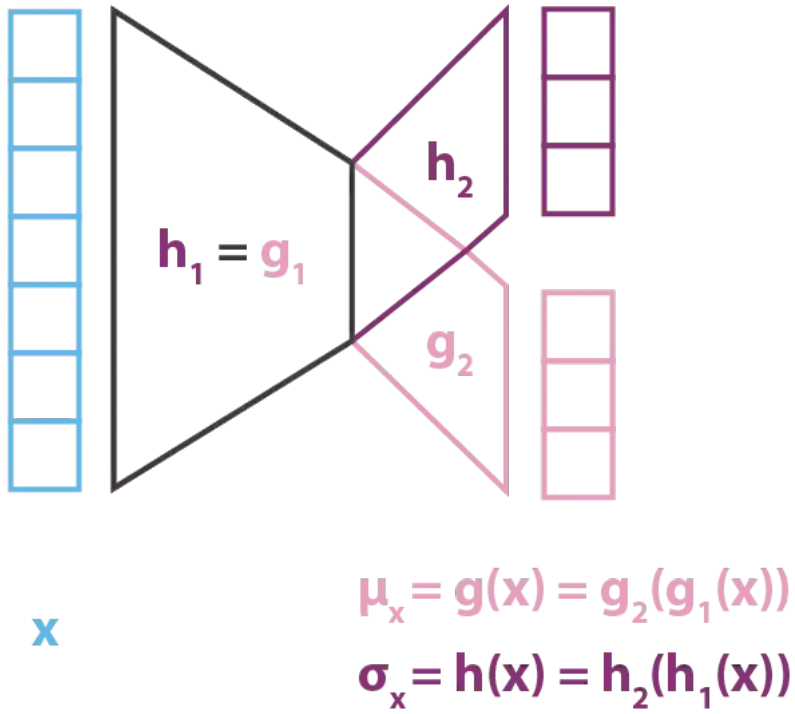


Figure: Encoder part of the VAE, Source

Reparameterization Trick

- Problem: We cannot perform backpropagation on the direct sampling operation

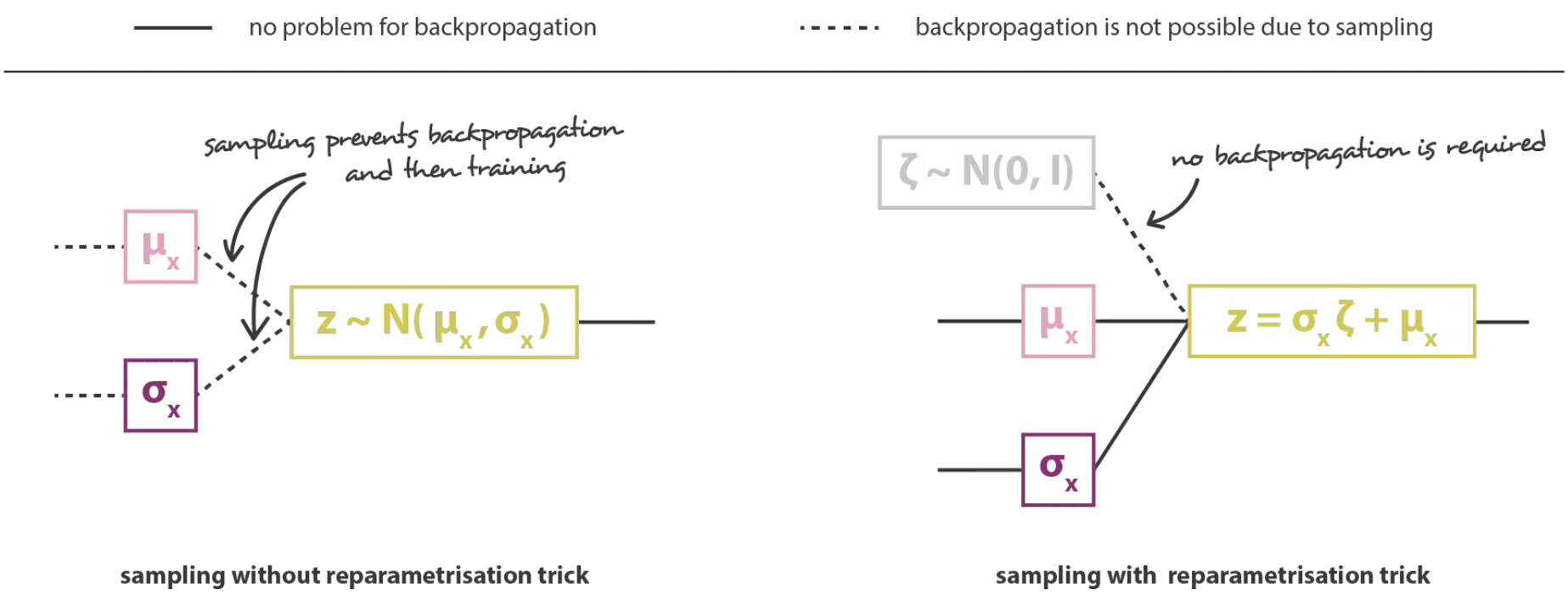


Figure: Illustration of the reparameterization trick, Source

Reparameterization Trick

- Solution: Reparameterization trick
  - We sample from a standard normal distribution and do the following operations to be able to track gradients

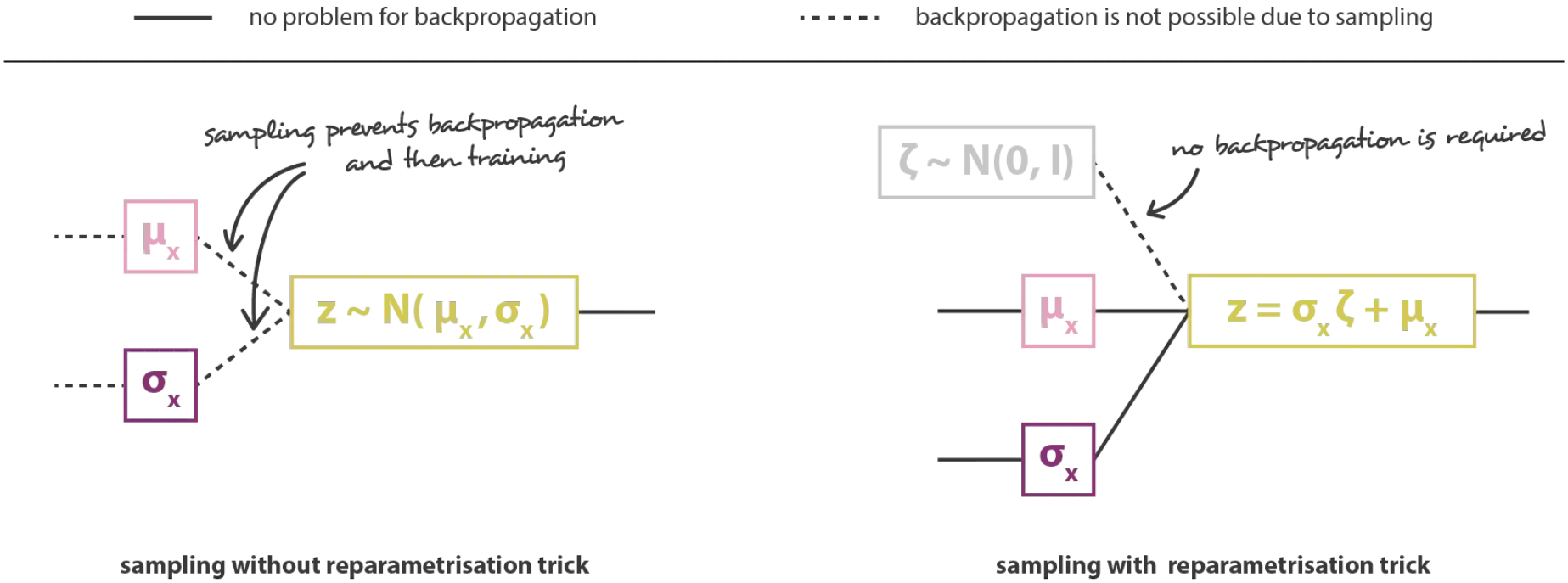
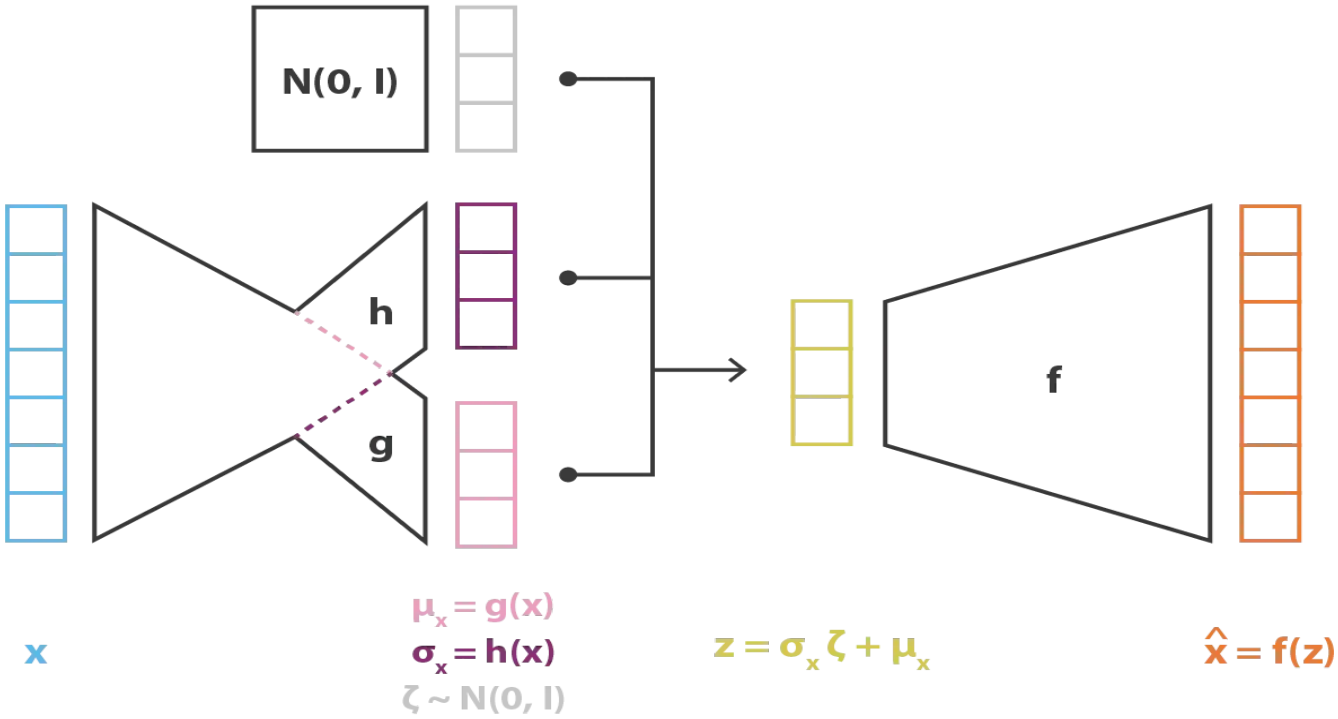


Figure: Illustration of the reparameterization trick, Source

The Whole Picture

- The whole picture of a simple variational autoencoder:



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

Figure: Variational Autoencoders representation, Source

Refrences

Yasemin Bozkurt, Tristan Bereau, and Joseph Rudzinski.
Interpretable embeddings from molecular simulations using gaussian mixture variational autoencoders.
Machine Learning: Science and Technology, 1, 03 2020.

Thank You!

Any Question?