Encoder-Decoder

Autoencoders for Computer Vision

Autoencoders (AE)

- A powerful tool used in machine learning for:
  - Feature extraction
  - Data compression
  - Image reconstruction

- Used for unsupervised learning tasks

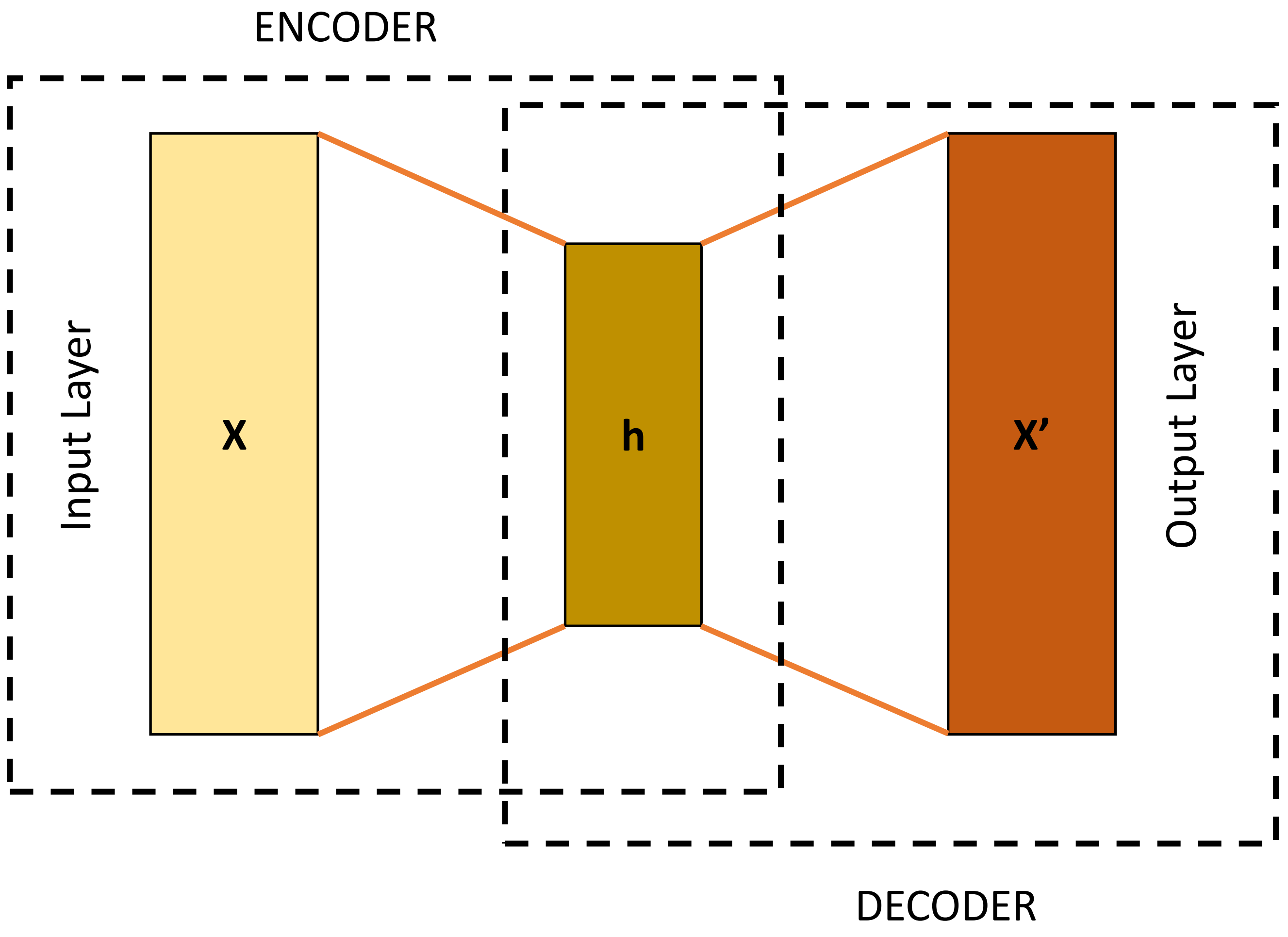
- An AE model has the ability to automatically learn complex features from input data
  - Popular method for improving accuracy of classification and prediction tasks

Autoencoders

- Autoencoders are neural networks
  - Can learn to compress and reconstruct input data, such as images, using a hidden layer of neurons
  - Learn data encodings in an **unsupervised** manner

- Consists of two parts:
  - Encoder: takes input data and compresses it into a lower-dimensional representation called **latent space**
  - Decoder: reconstructs input data from latent space representation

- In an optimal scenario, autoencoder performs as close to perfect reconstruction as possible

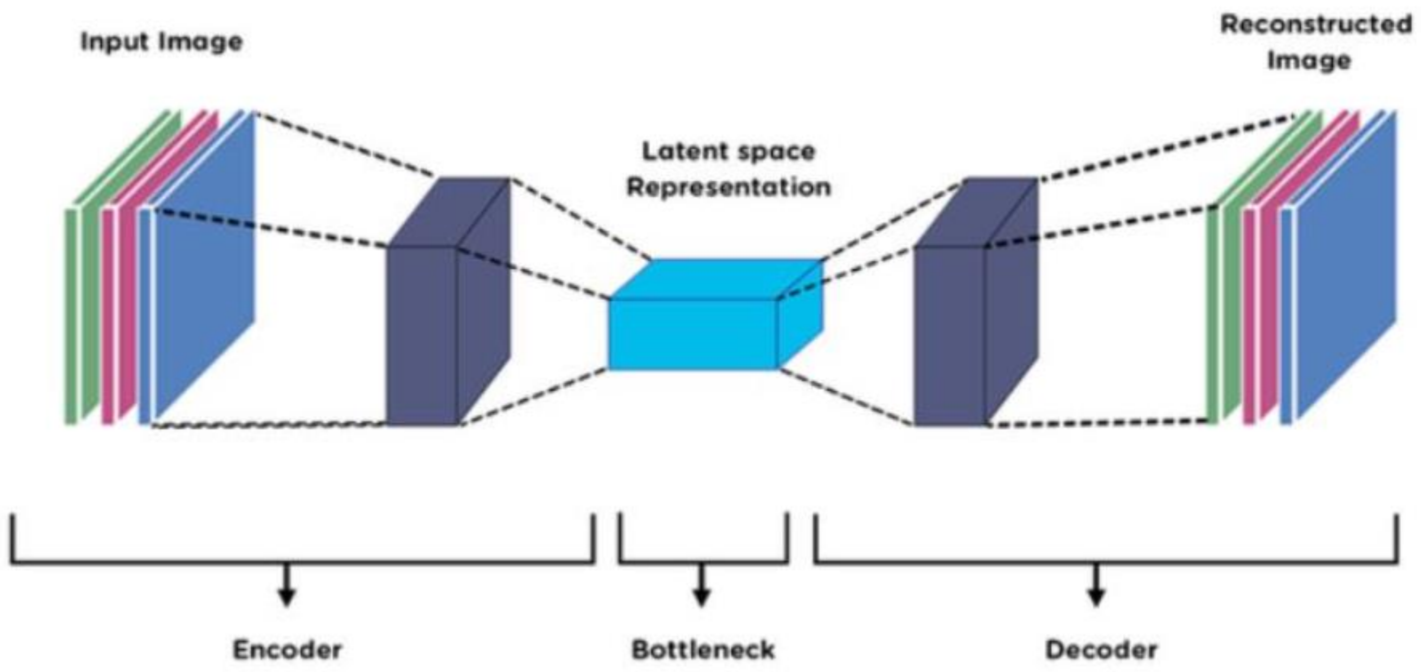

ENCODER
Input Layer
X
h
X'
Output Layer
DECODER

Loss Function and Reconstruction Loss

- Loss functions - critical role in training autoencoders and determining their performance
  - Most commonly used is **reconstruction loss**
  - Used to measure difference between model input and output
  - **Reconstruction error** calculated using various loss functions such as mean squared error, binary cross-entropy, or categorical cross-entropy
  - Applied method depends on type of data being reconstructed
- Reconstruction loss used to update weights of network during backpropagation to minimize difference between input and output
  - Goal: achieve low reconstruction loss
  - Low loss $\rightarrow$ model can effectively capture salient features of input data and reconstruct it accurately

AE in Computer Vision

- Input is an image and output is a reconstructed image
  - Input mage typically represented as a matrix of pixel values
  - Can be of any size, but is typically normalized to improve performance



Encoder

- **Encoder**: Compresses input image into a lower-dimensional representation, known as **latent space (**"bottleneck" or "code“)
- Encoder is:
  - Series of convolutional layers
  - Followed by pooling modules or simple linear layers, that extract different levels of features from input image
- Each layer:
  - Applies a set of filters to input image
  - Outputs a feature map that highlights specific patterns and structures in image

Encoder

- Every convolution layer composed of *n* convolution filters, $\{F_1^{(1)} \ldots F_n^{(1)}\}$ each with depth *D*
- Convolution of input volume $I = \{I_1, \ldots I_D\}$ with filters produces *n* activation maps

$$O_m(i,j) = a(\sum_{d=1}^{D} \sum_{u=-2k-1}^{2k+1} \sum_{v=-2k-1}^{2k+1} F_{m_d}^{(1)}(u,v) I_d(i-u, j-v)) \quad m = 1, .., n$$

  - $O(i,j)$ is pixel at position (i,j)
  - $2k+1$ is side of a square, odd convolutional filter
  - Every convolution wrapped by a non-linear function *a*

https://pgaleone.eu/neural-networks/2016/11/24/convolutional-autoencoders/

Encoder

$z_m = O_m = a(I * F_m^{(1)} + b_m^{(1)})$ b_m is bias for mth feature map

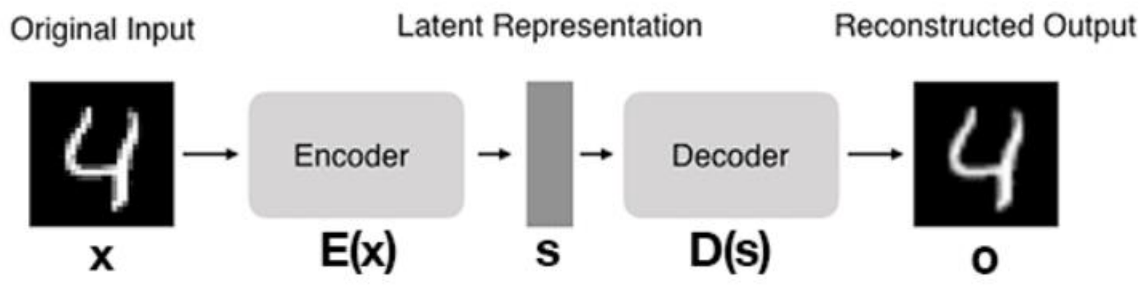
- Produced activation maps are encoding of input *I* in a low-dimensional space
- Convolution reduces output's spatial extent
  - Not possible to reconstruct volume with same spatial extent as input
  - Input padding such that *dim*(*I*) = *dim*(*decode*(*encode*(*I*)))

Bottleneck

- **Bottleneck / Latent Representation**: Output of encoder is a compressed representation of input image in latent space
  - Captures most important features of input image
  - Typically a smaller dimensional representation of input image
  - Restricts flow of information to decoder from encoder - allowing only most vital information to pass through
  - Prevents neural network from memorizing input and overfitting data
  - Smaller the code, lower the risk of overfitting
  - Layer generally implemented as a linear layer or as a tensor if we use convolutions
  - If input data denoted as $\mathbf{x}$, then latent space representation $\mathbf{s} = E(\mathbf{x})$

Decoder

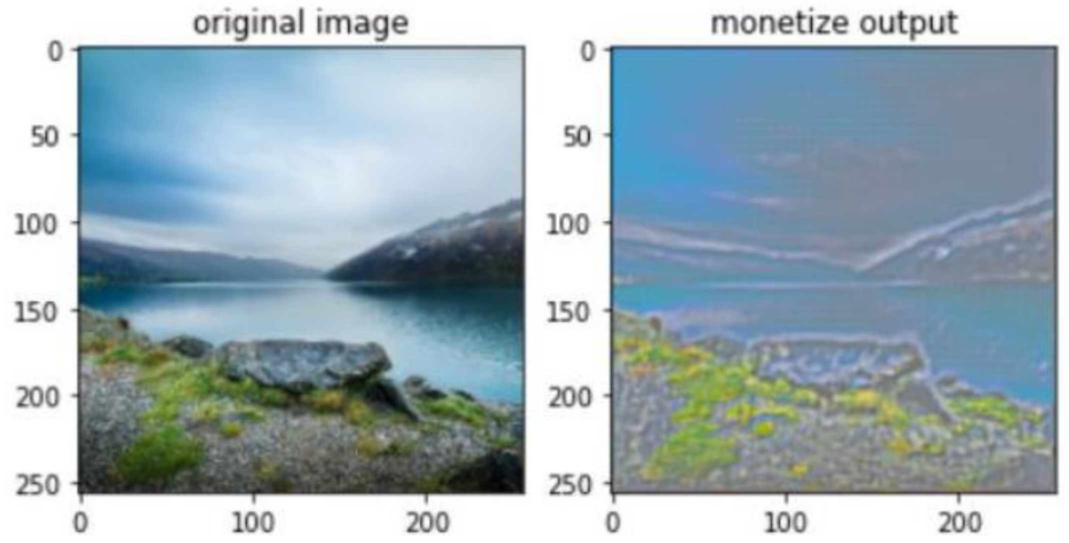
- **Decoder**: reconstructs input image from latent representation
    - Decoder is a set of multiple deconvolutional layers that gradually increase size of feature maps until final output is same size as input image
        - Usually implemented with transposed convolutions if we work with images or linear layers
    - Every layer applies a set of filters that up-sample feature maps
    - Output compared with ground truth
    - If output of decoder is $\mathbf{o}$, then $\mathbf{o} = D(\mathbf{s}) = D(E(\mathbf{x}))$



Decoder

- Output of decoder is a reconstructed image similar to input image
  - Reconstructed image may not be identical to input image
    - Since AE learns to capture most important features of input image in latent representation
  - These features can be used for tasks such as image classification, object detection, and image retrieval

AE converting input to a Monet style painting



https://www.analyticsvidhya.com/blog/2021/01/auto-encoders-for-computer-vision-an-endless-world-of-possibilities/

Decoder

- *n* feature maps $z_{m=1,..n}$ (latent representations) produced from Encoder used as input to decoder to reconstruct image *I*
- Hyper-parameters of decoding convolution fixed by encoding architecture
  - Filters volume $F^{(2)}$ with dimensions $(2k + 1, 2k + 1, n)$ to produce same spatial extent of *I*
  - Number of filters to learn: *D*
- Reconstructed image Ĩ result of convolution between feature maps

  $Z = \{z_{i=1}\}^n$ and $F^{(2)}$

  $$\tilde{I} = a(Z * F_m^{(2)} + b^{(2)})$$

- Loss function $L(I, \tilde{I}) = \frac{1}{2}||I - \tilde{I}||_2^2$

Dimensionality Reduction

- Dimensionality reduction - process of reducing number of dimensions in encoded representation of input data
- AE can learn to perform dimensionality reduction:
  - Training encoder to map input data to a lower-dimensional latent space
  - Decoder trained to reconstruct original input data from latent space representation
  - Size of latent space typically much smaller than size of input data - allowing for efficient storage and computation of data
- Through dimensionality reduction, AE can also help to remove noise and irrelevant features
  - Useful for improving performance of downstream tasks such as data classification or clustering

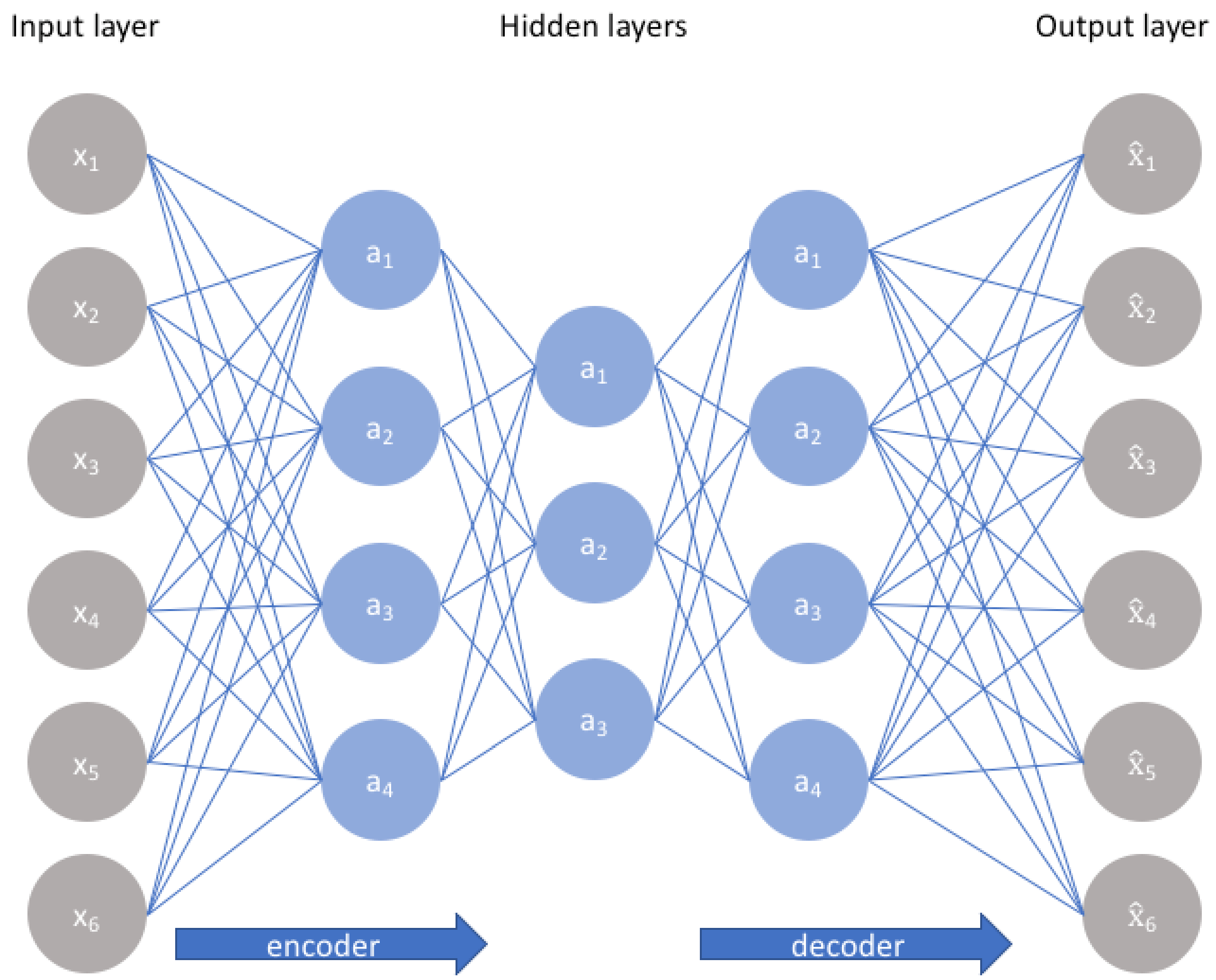
Contractive autoencoder

- Designed to learn a compressed representation of input data while being resistant to small perturbations in input
    - Achieved by adding a regularization term to training objective
    - This term penalizes network for changing output with respect to small changes in input

      $Loss = L(I, \tilde{I}) +$ regularizer
    - Can add a scaling parameter in front of regularization term to adjust trade-off between the two objectives

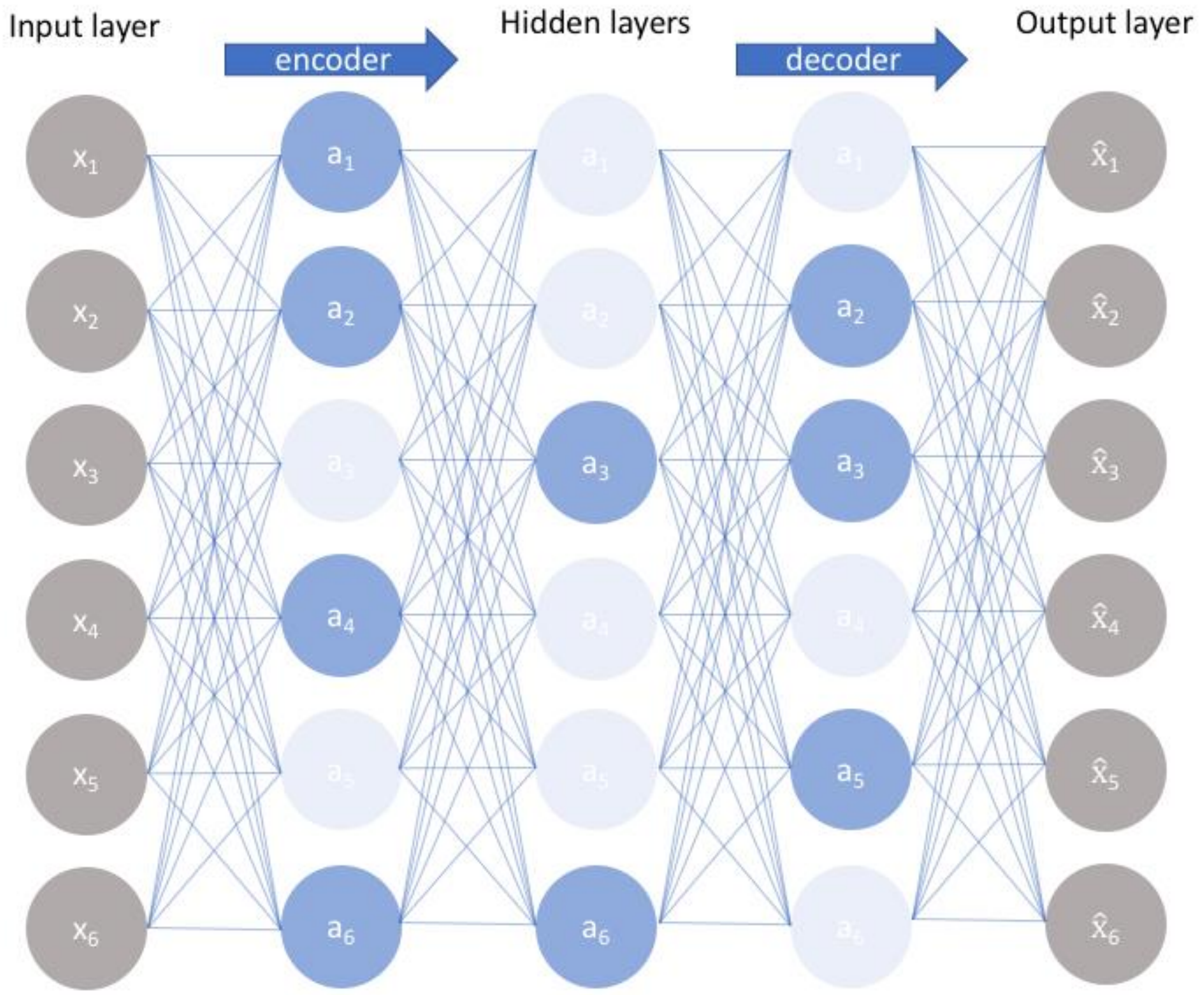
Undercomplete Autoencoder

- Constrain number of nodes present in hidden layer(s) of network
  - Limit amount of information that can flow through the network
  - Model can learn most important attributes of input data and how to best reconstruct original input from an "encoded" state
  - For higher dimensional data, autoencoders capable of learning a complex representation of data (manifold)
  - Can be used to describe observations in a lower dimensionality and correspondingly decoded into original input space


Input layer
Hidden layers
Output layer
x_1
x_2
x_3
x_4
x_5
x_6
a_1
a_2
a_3
a_4
a_1
a_2
a_3
a_1
a_2
a_3
a_4
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$
encoder
decoder

Sparse Autoencoder

- Added constraint on encoding process
  - Encoder network trained to produce sparse encoding vectors - have many zero values
    - Does not require reduction in number of nodes at hidden layer
- Forces network to identify only most important features of input data
  - Construct loss function to penalize *activations* within a layer
  - Encourage network to learn an encoding and decoding which only relies on activating a small number of neurons
  - Different approach towards regularization - we normally regularize *weights* of a network, not activations
- Sensitize individual hidden layer nodes toward specific attributes of input
  - Forced to selectively activate regions of network depending on input data



- **Opacity of a node corresponds with level of activation**
- **Individual nodes of a trained model which activate are *data-dependent***
- ***D*ifferent inputs will result in activations of different nodes through network**

Sparse Autoencoder

- Two main ways to impose sparsity constraint
  - Both involve measuring hidden layer activations for each training batch and adding some term to loss function to penalize excessive activations
- L1 Regularization: Add term to loss function to penalize absolute value of vector of activations *a* in layer h for observation *i*, scaled by tuning parameter λ

$$L(I, \hat{I}) + \lambda \sum_i |a_i^{(h)}|$$

Sparse Autoencoder

- KL-Divergence: measure of difference between two probability distributions
  - Define a sparsity parameter ρ which denotes average activation of a neuron over a collection of samples
  - This expectation can be calculated as

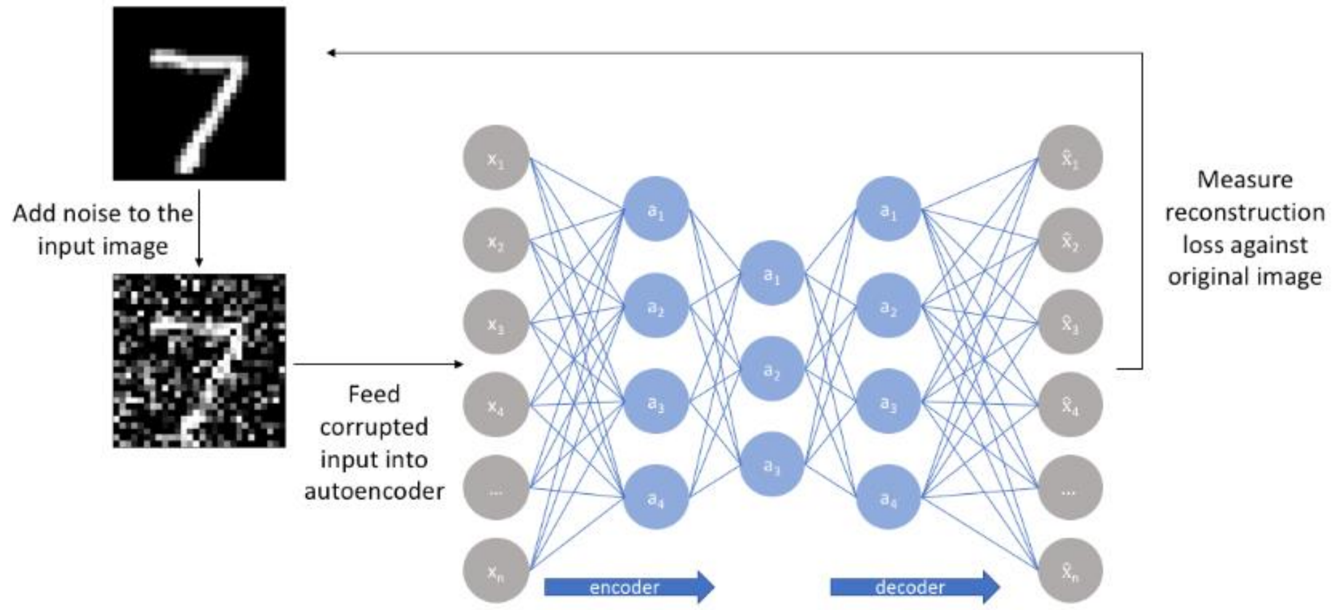
$$\widehat{\rho_j} = \frac{1}{m}\sum_{i}[\,a_i^{(h)}(x)]$$

  - For *j*th neuron in layer h, activations for *m* training observations denoted individually as *x*
  - By constraining average activation of a neuron over a collection of samples - encouraging neurons to only fire for a subset of observations
  - compare ideal distribution ρ to observed distributions $\hat{\rho}$ over all hidden layer nodes

$$L(I, \hat{I}) + \sum_j KL(\rho || \hat{\rho})$$

Denoising autoencoder

- Designed to learn to reconstruct an input from a corrupted version of input
  - Corrupted input created by adding noise to original input
  - Network trained to remove noise and reconstruct original input



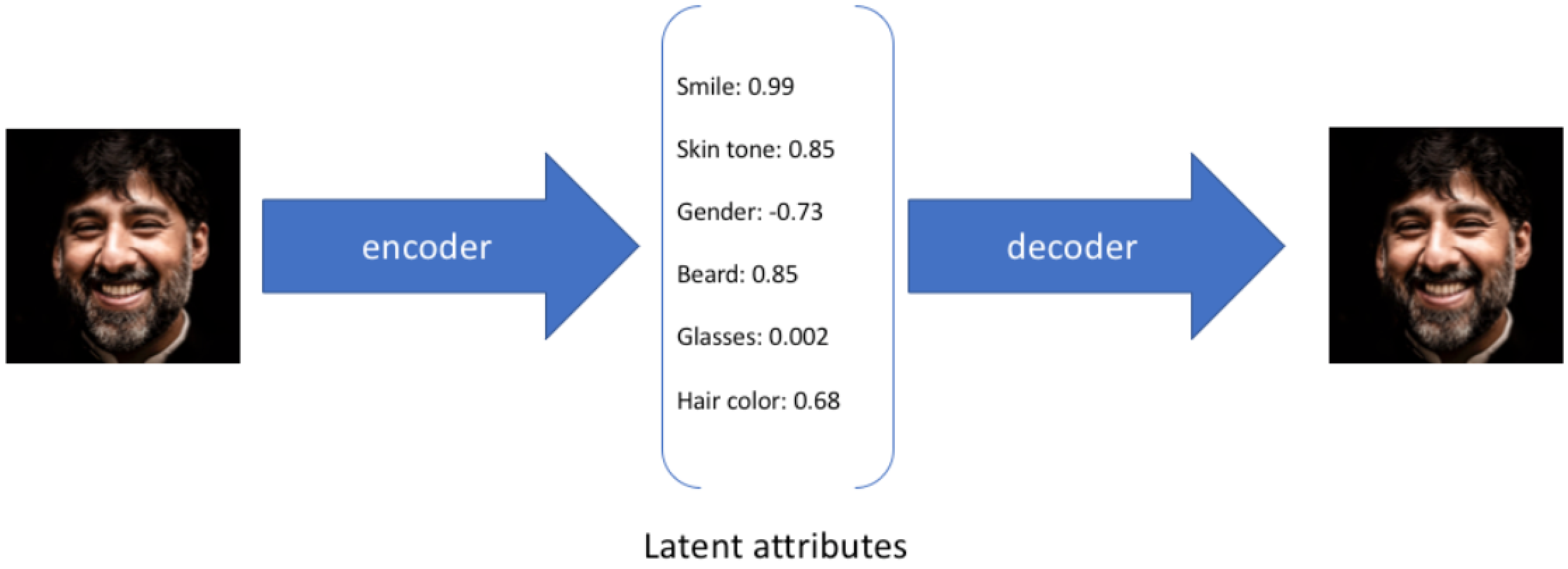
Denoising autoencoder

- Model is not able to develop a mapping which memorizes training data because input and target output are no longer same
- Rather, model learns a vector field for mapping input data towards a lower-dimensional manifold
  - If this manifold accurately describes the natural data, we have effectively "canceled out" added noise

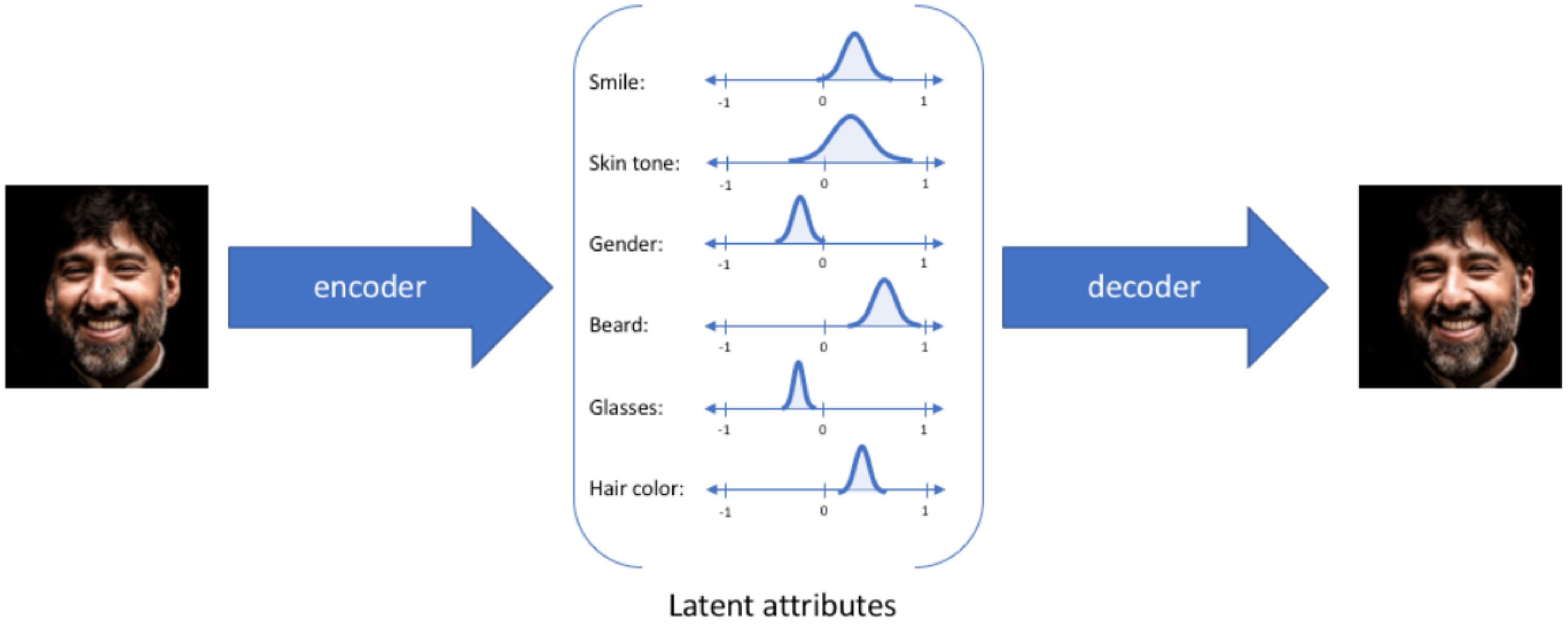
Variational Autoencoders (VAE)

- Proposed in 2013 by Diederik P. Kingma and Max Welling
- Provides a probabilistic manner for describing an observation in latent space
  - Trained to learn a mapping from input data to a probability distribution in a lower-dimensional latent space
  - Generate new samples from this distribution
  - Commonly used in image and text generation tasks
- Latent space is continuous
  - Decoder can generate new data points that interpolate among training data points
- Difference from autoencoder:
  - Provides a statistical manner for describing samples of dataset in latent space
  - Encoder outputs a probability distribution in bottleneck layer instead of a single output value

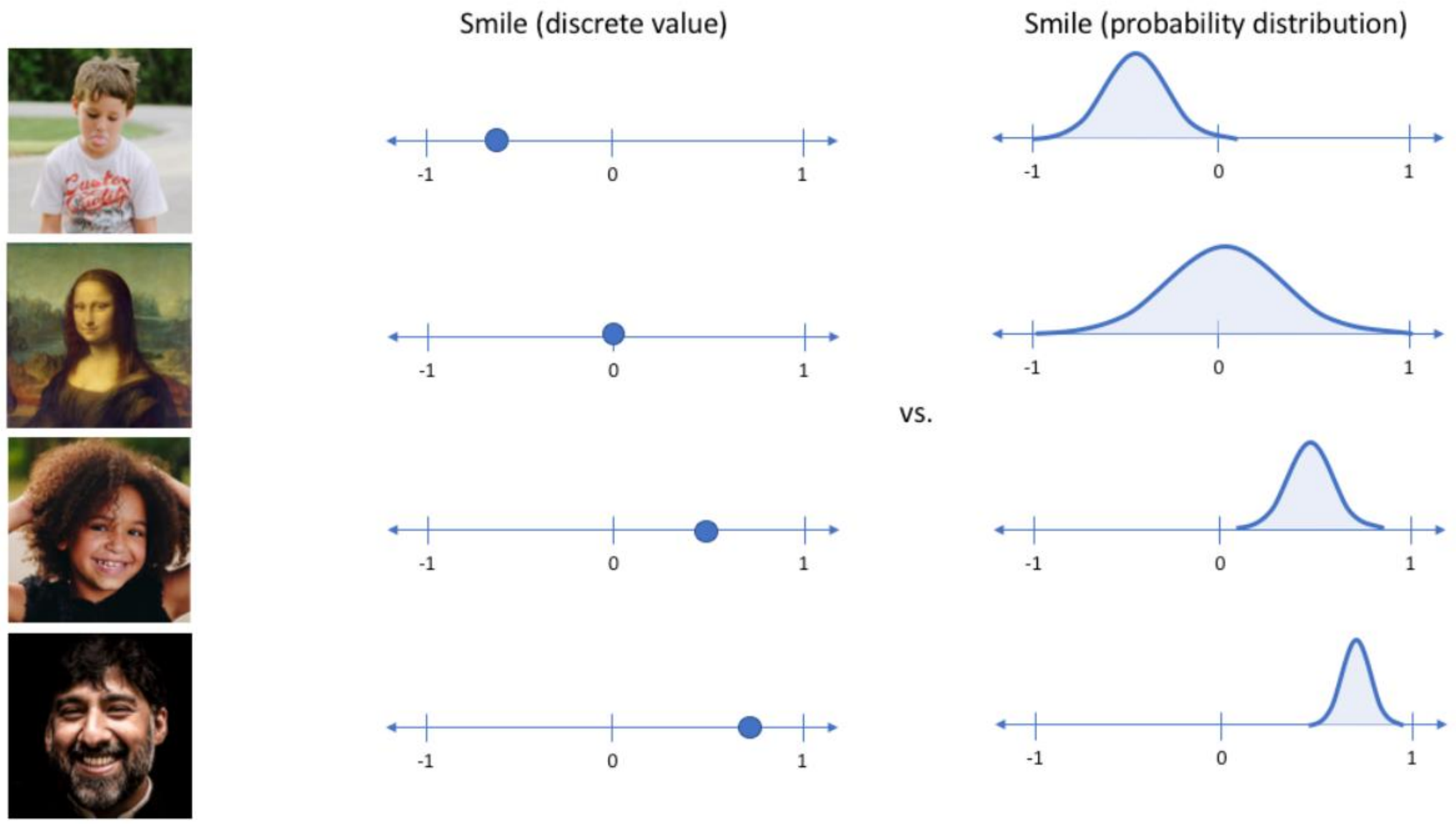
AE


encoder
Smile: 0.99
Skin tone: 0.85
Gender: -0.73
Beard: 0.85
Glasses: 0.002
Hair color: 0.68
decoder
Latent attributes

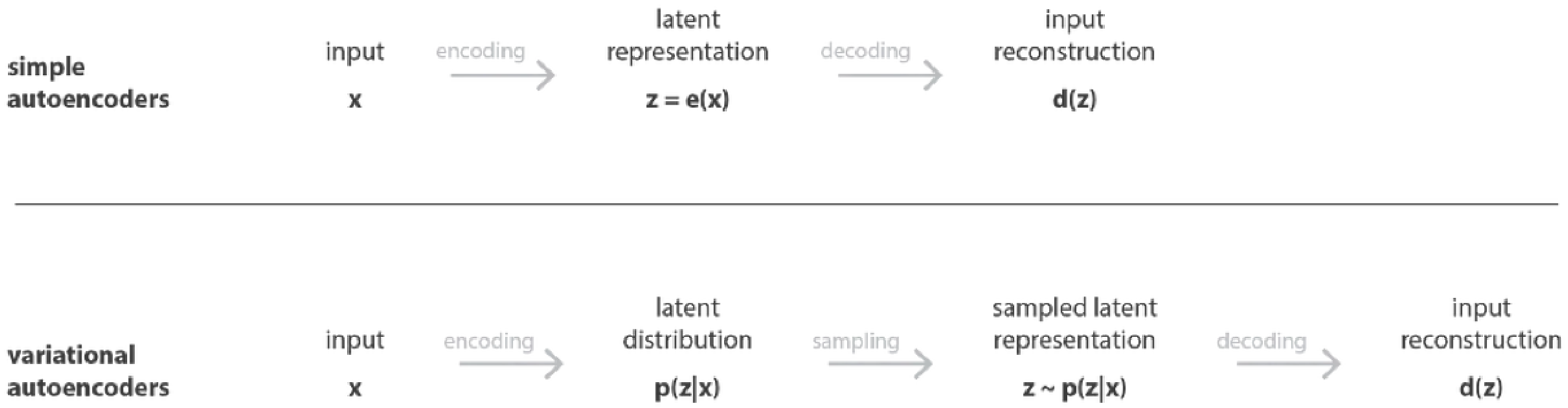
VAE


encoder
Smile:
Skin tone:
Gender:
Beard:
Glasses:
Hair color:
-1
0
1
decoder
Latent attributes

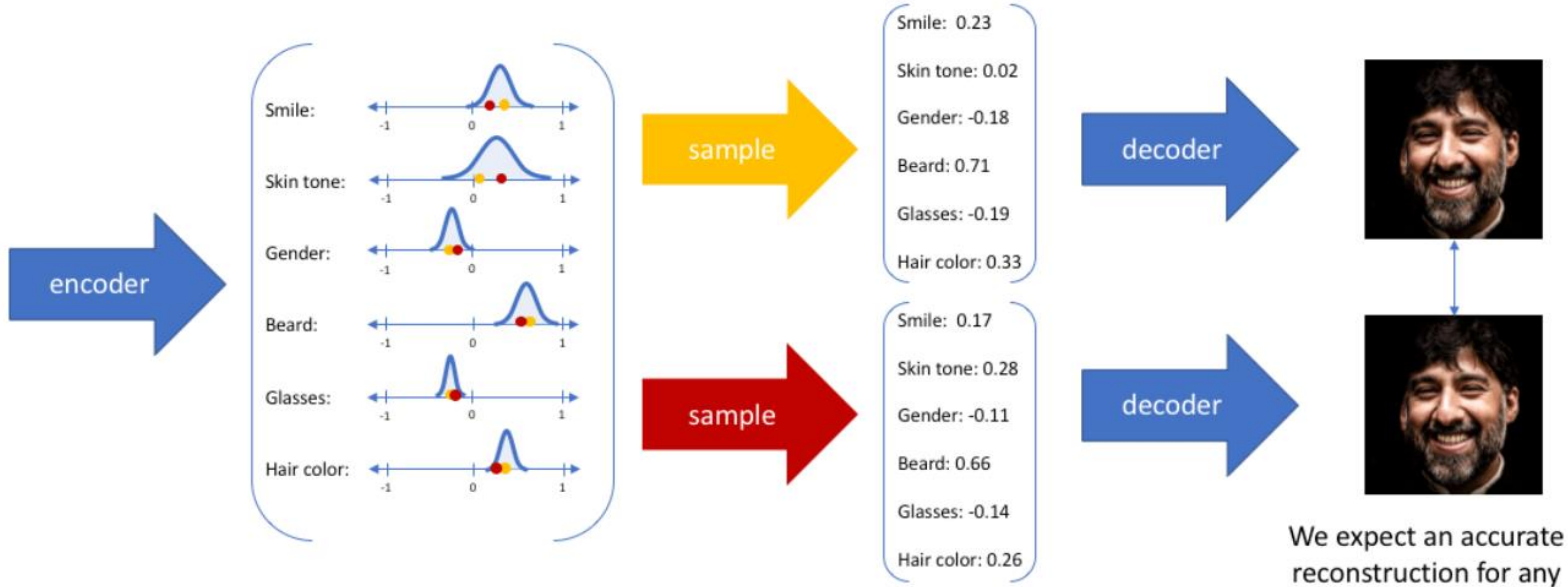
AE vs. VAE


Smile (discrete value)
Smile (probability distribution)
-1
0
1
vs.

AE vs VAE


simple autoencoders
input
x
encoding
latent representation
z = e(x)
decoding
input reconstruction
d(z)
variational autoencoders
input
x
encoding
latent distribution
p(z|x)
sampling
sampled latent representation
z ~ p(z|x)
decoding
input reconstruction
d(z)

VAE



Latent distributions

Sampled latent attributes

We expect an accurate reconstruction for any sample from the latent state distributions