# Autoencoders (AEs)

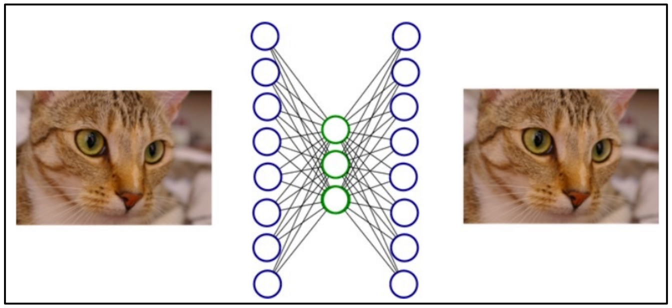
Thanks to Sargur Srihari, Fei-Fei Li, Justin Johnson, Serena Yeung, Sosuke Kobayashi, Yingyu Liang, Guy Golan, Song Han, Jason Brownlee, Jefferson Hernandez

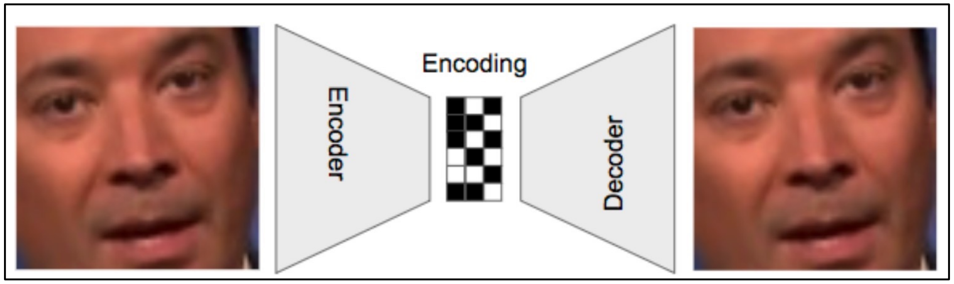
# Some Autoencoder Applications

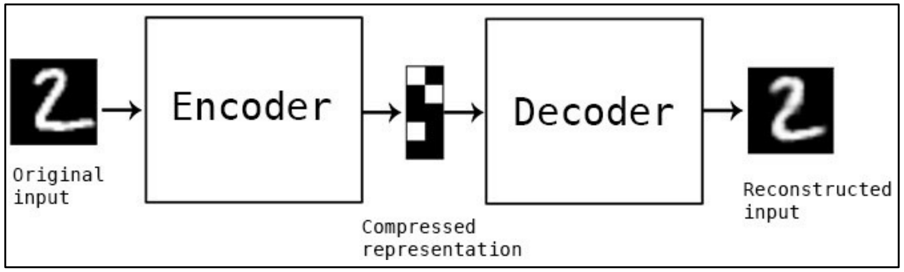
1. Dimensionality Reduction
2. Image Compression
3. Image Denoising
4. Feature Extraction
5. Image generation
6. Sequence to sequence prediction
7. Encoders for transformers

# What is an Autoencoder (AE) ?

- A neural network trained using unsupervised learning
  - Trained to copy its input to its output
  - Learns an embedding ***h***

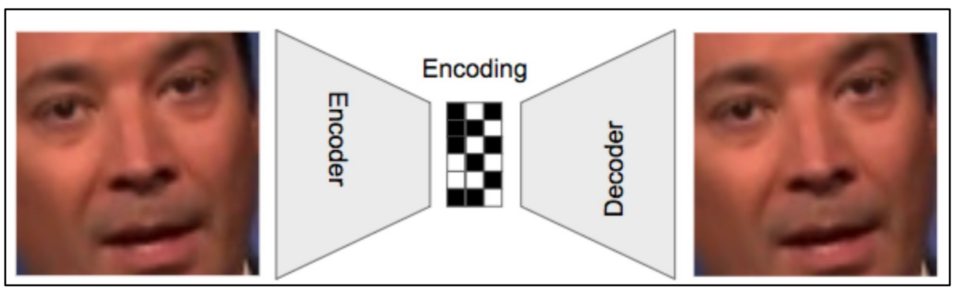




# Embedding is a Point on a Manifold

- An embedding is a low-dimensional vector
  - With fewer dimensions than the ambient space of which the manifold is a low-dimensional subset
- Embedding Algorithm
  - Maps any point in ambient space $\boldsymbol{x}$ to its embedding $\boldsymbol{h}$
  - Embeddings of related inputs form a manifold



# Other Embeddings

All are dimensionally reduction methods:

Principle component analysis (PCA):

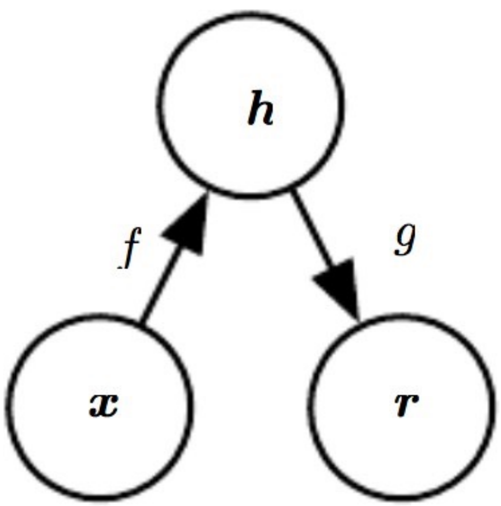
- PCA is a feature extraction technique — it combines the variables, and then it drops the least important variables while still retains the valuable parts of the variables
- Probably the most widely used embedding to date. The idea is simple: Find a linear transformation of features that maximizes the captured variance or (equivalently) minimizes the quadratic reconstruction error.

Multidimensional Scaling (MDS):

- Unsupervised ML methods that represent high-dimensional data in a lower dimensional space, while preserving the inter-point distances as best as possible.

# General Structure of an Autoencoder

- Maps an input $\boldsymbol{x}$ to an output $\boldsymbol{r}$ (called a reconstruction) through an internal representation code $\boldsymbol{h}$
  - Hidden layer $\boldsymbol{h}$ describes a code used to represent the input
- The network has two parts
  - The encoder function $\boldsymbol{h}=f(\boldsymbol{x})$
  - A decoder that produces a reconstruction $\boldsymbol{r}=g(\boldsymbol{h})$



# Autoencoders Differ from Classical Data Compression

- Autoencoders are data-specific
  - i.e., only able to compress data similar to what they have been trained on

- Different from MP3 or JPEG compression algorithm
  - These make general assumptions about "sound/images”, but not about specific types of sounds/images
  - Autoencoder for pictures of cats would do poorly in compressing pictures of trees
    - Features it would learn would be cat-specific
- Autoencoders are lossy
  - Their decompressed outputs will be degraded compared to the original inputs (similar to MP3 or JPEG compression).
  - This differs from lossless arithmetic compression
- Autoencoders are learned

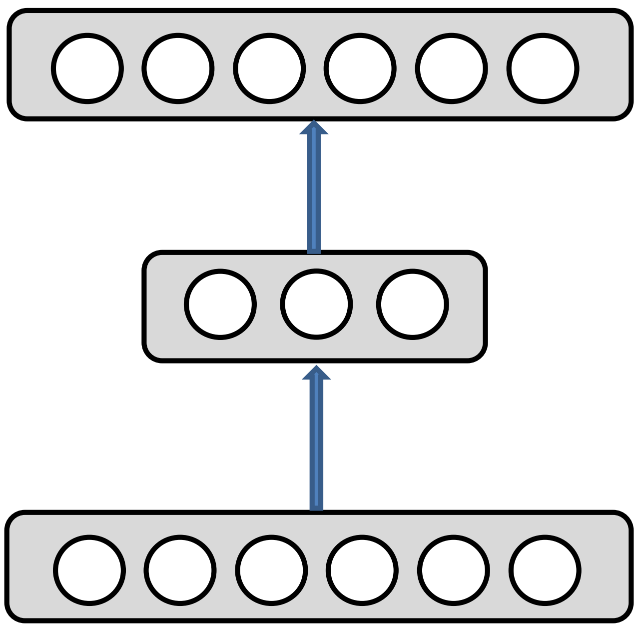
# What does an Autoencoder Learn?

- Learning $g(f(\boldsymbol{x}))=\boldsymbol{x}$ everywhere is not useful
- Autoencoders are designed to be **unable** to copy perfectly
  - Restricted to copying only approximately
- Autoencoders learn useful properties of the data
  - Forced to prioritize which aspects of input should be copied
- Can learn stochastic mappings
  - Go beyond deterministic functions to mappings $p_{\text{encoder}}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h})$

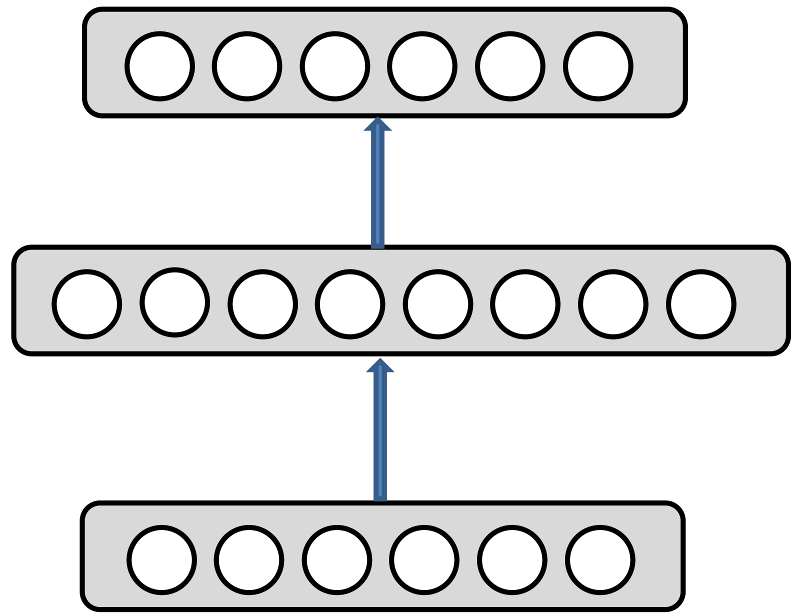
# Basic Types of Autoencoders (AEs)

We distinguish between two types of AE structures:

Undercomplete



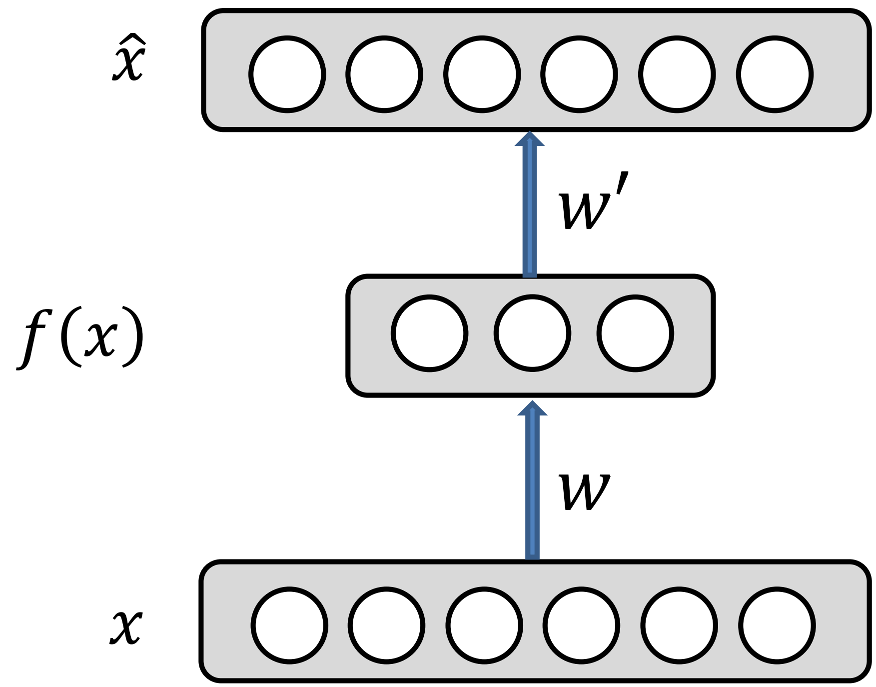
Overcomplete



# Undercomplete AE

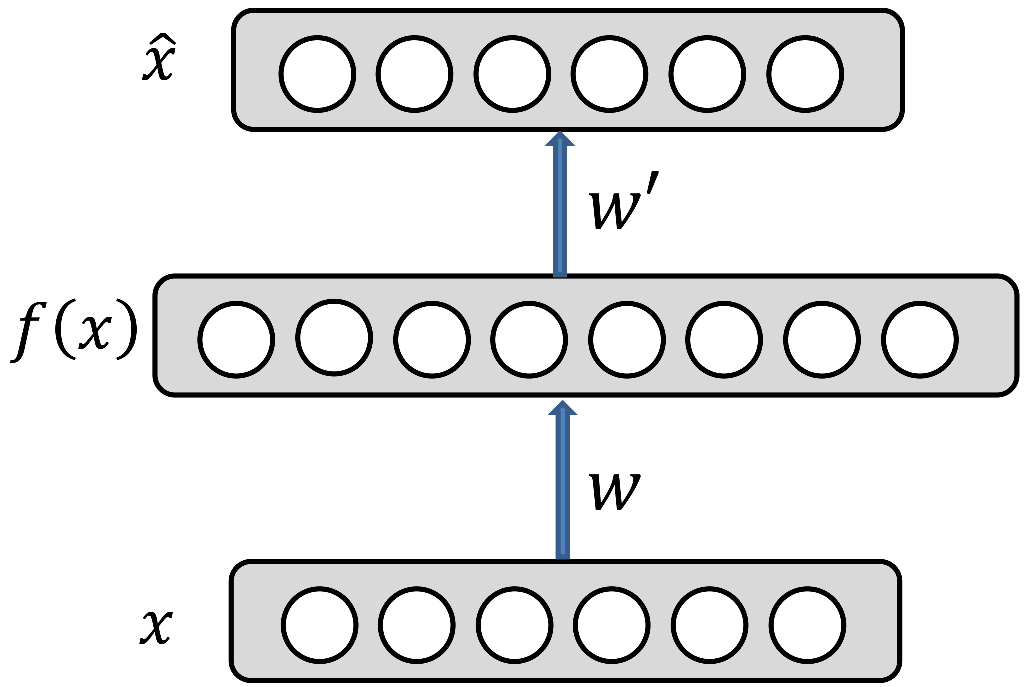
- Hidden layer is **Undercomplete** if smaller than the input layer
  - ❑Compresses the input
  - ❑Compresses well only for the training distribution

- Hidden nodes will be
  - ❑Good features for the training distribution.
  - ❑Bad for other types on input



# Overcomplete AE

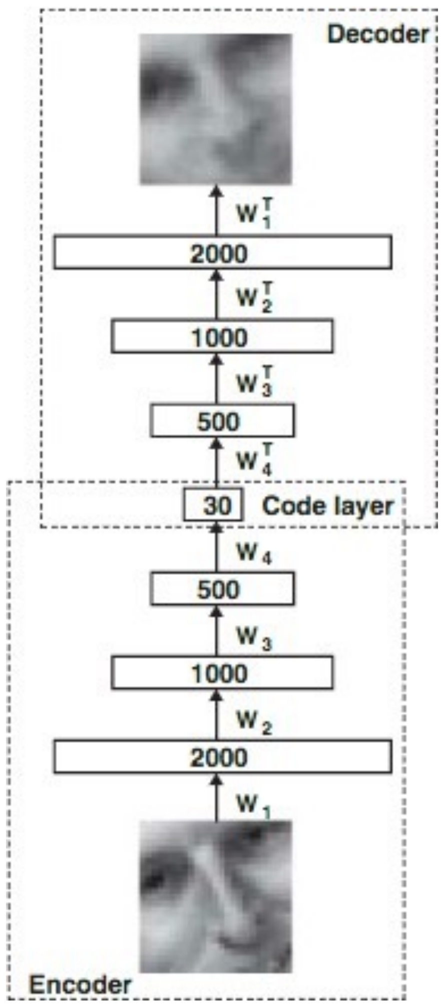
- Hidden layer is **Overcomplete** if greater than the input layer
  - ❑ No compression in hidden layer.
  - ❑ Each hidden unit could copy a different input component.
- No guarantee that the hidden units will extract meaningful structure.
- Adding dimensions is good for training a linear classifier (XOR case example).
- A higher dimension code helps model a more complex distribution.



# An autoencoder architecture

Decoder g

Encoder f




Weights W are learned using:

1. Training samples, and
2. a loss function

# Autoencoder Training Methods

1. Autoencoder is a feed-forward non-recurrent neural net
   - With an input layer, an output layer and one or more hidden layers
   - Can be trained using the same techniques
     - Compute gradients using back-propagation
     - Followed by minibatch gradient descent
2. Unlike feedforward networks, can also be trained using *Recirculation*
   - Compare activations on the input to activations of the reconstructed input
   - More biologically plausible than back-prop but rarely used in ML

# 1. Undercomplete Autoencoder

- Copying input to output seems useless but we have no interest in decoder output
- Want $\boldsymbol{h}$ to take on useful properties
- Undercomplete autoencoder
  - Constrain $\boldsymbol{h}$ to have lower dimension than $\boldsymbol{x}$
  - Force it to capture most salient features of training data

# Autoencoder with Linear Decoder +MSE is a PCA

- Learning process is minimizing a loss function

$$L(\boldsymbol{x}, g(f(\boldsymbol{x})))$$

  - where $L$ is a loss function penalizing $g(f(\boldsymbol{x}))$ for being dissimilar from $\boldsymbol{x}$
    - Exs: $L^2$ norm of difference: mean squared error
  - When the decoder $g$ is linear and $L$ is the mean squared error, an undercomplete autoencoder learns to span the same subspace as PCA
    - In this case the autoencoder trained to perform the copying task has learned the principal subspace of the training data as a side-effect

- Autoencoders with nonlinear $f$ and $g$ can learn more powerful nonlinear generalizations of PCA
  - But high capacity is not desirable

# Autoencoder Training Using a Loss Function

- Encoder $f$ and decoder $g$

  $f : X \rightarrow h$

  $g : h \rightarrow X$

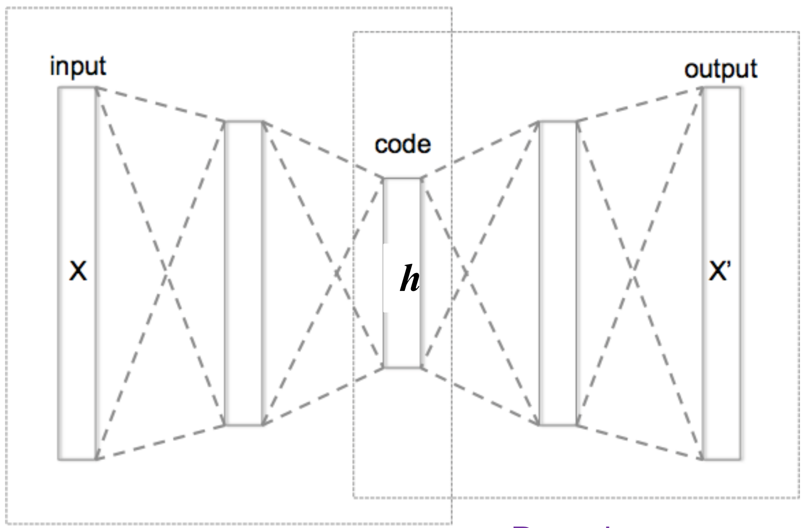
  $\arg\min_{f,g} \left\| X - (f \circ g) X \right\|^2$

- One hidden layer
  - Non-linear encoder
  - Takes input $x \in R^d$
  - Maps into output $h \in R^p$

  $h = \sigma_1(Wx + b)$

  $x' = \sigma_2(W'h + b')$

Autoencoder with 3 fully connected hidden layers




$\sigma$ is an element-wise activation function such as sigmoid or Relu

Trained to minimize reconstruction error (such as sum of squared errors)

$$L(\mathbf{x}, \mathbf{x}') = \left\| \mathbf{x} - \mathbf{x}' \right\|^2 = \left\| \mathbf{x} - \sigma_2(W^t(\sigma_1(W\mathbf{x} + \mathbf{b})) + \mathbf{b}') \right\|^2$$

Provides a compressed representation of the input $x$

# Encoder/decoder Capacity

- If encoder $f$ and decoder $g$ are allowed too much capacity
  - autoencoder can learn to perform the copying task without learning any useful information about the distribution of data
- Autoencoder with a one-dimensional code and a very powerful nonlinear encoder can learn to map $\boldsymbol{x}^{(i)}$ to code $i$.
  - The decoder can learn to map these integer indices back to the values of specific training examples
- Autoencoder trained for copying task fails to learn anything useful if $f/g$ capacity is too great

A model with too little capacity cannot learn the training dataset meaning it will underfit, whereas a model with too much capacity may memorize the training dataset, meaning it will overfit or may get stuck or lost during the optimization process.

The capacity of a neural network model is defined by configuring the number of nodes and the number of layers.

# Cases When Autoencoder Learning Fails

- When do autoencoders fail to learn anything useful:
  1. Capacity of encoder/decoder $f/g$ is too high
     - Capacity controlled by depth
  2. Hidden code $\boldsymbol{h}$ has dimension equal to input $\boldsymbol{x}$
  3. *Overcomplete* case: where hidden code $\boldsymbol{h}$ has dimension greater than input $\boldsymbol{x}$
     - Even a linear encoder/decoder can learn to copy input to output without learning anything useful about data distribution

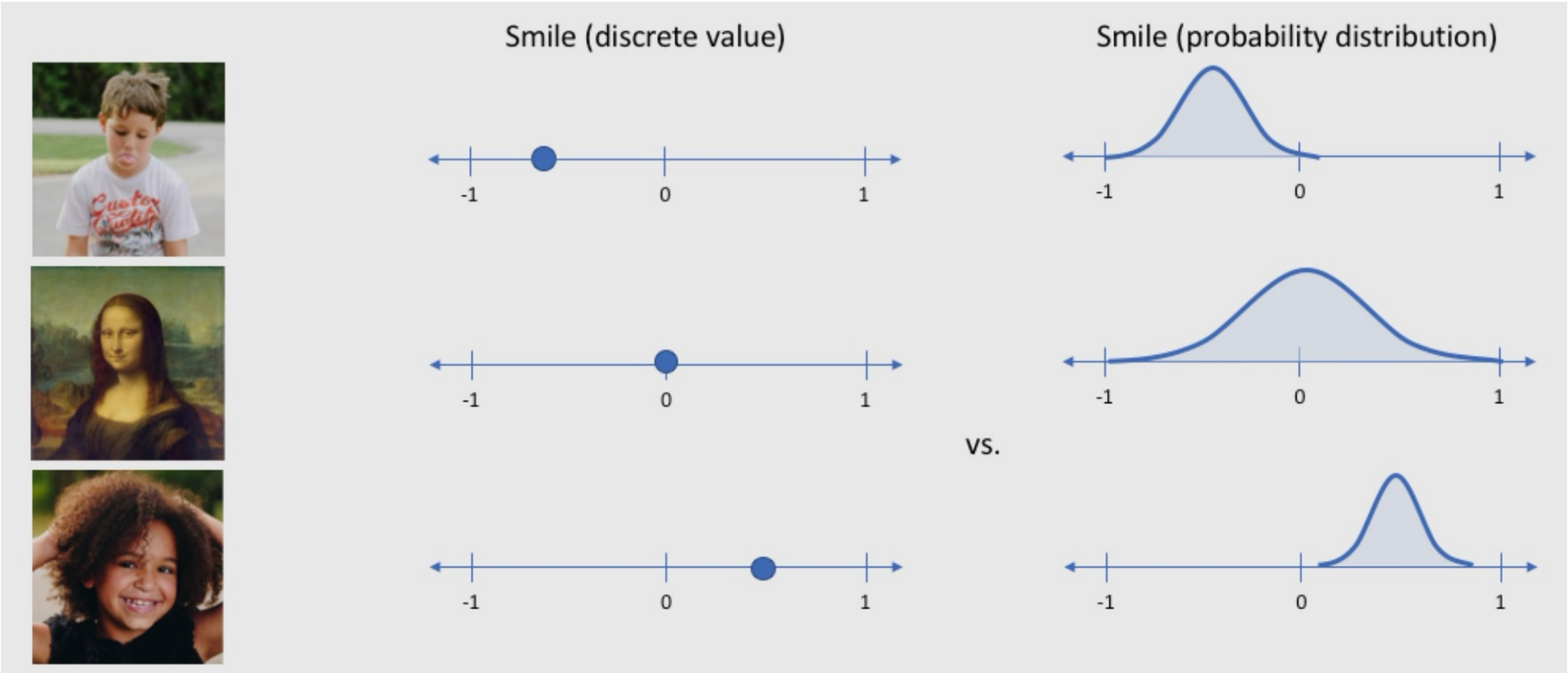
## 2. Correct AE Design: use Regularization

- Ideally, choose code size (dimension of $\boldsymbol{h}$) small and capacity of encoder $f$ and decoder $g$ based on complexity of distribution modeled
- *Regularized autoencoders*
  - Rather than limiting model capacity by keeping encoder/decoder shallow and code size small, use a loss function that encourages the model to have properties other than copy its input to output

# Regularized Autoencoder Properties

- Regularized AEs  have properties beyond copying input to output:
  - Sparsity of representation
  - Smallness of the derivative of the representation
  - Robustness to noise
  - Robustness to missing inputs
- Regularized autoencoders can be nonlinear and overcomplete
  - Still can learn something useful about the data distribution even if model  capacity is great enough to learn trivial identity function

# Latent variables treated as distributions




Source: https://www.jeremyjordan.me/variational-autoencoders/