Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

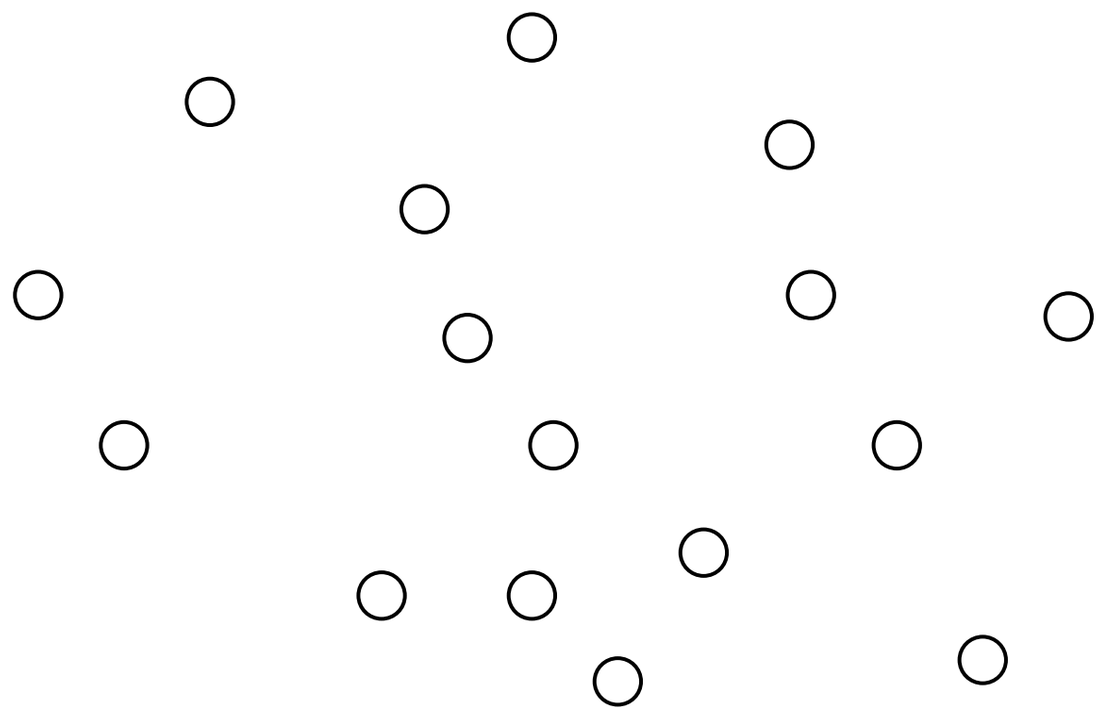
Today

- Intro. to Unsupervised Learning
  - Deep Unsupervised Learning
  - Matrix Completion, Approximation
  - Embedding
  - Clustering (next time)

Learning from raw data

What can be done without any labels?

Unsupervised learning



Unsupervised learning

- If you don’t have access to any labels, then you cannot do a Classification task or a Regression task, i.e. you cannot do Supervised Learning.
  - Note: there is a field called Semi-Supervised learning, where some of the points are labeled (we will see a special case, Active Learning / Interactive Learning, if time)
  - Note: you could choose one feature (i.e. column of the data) and call it the label, and do regression to predict that feature. But this might not be the desired learning task

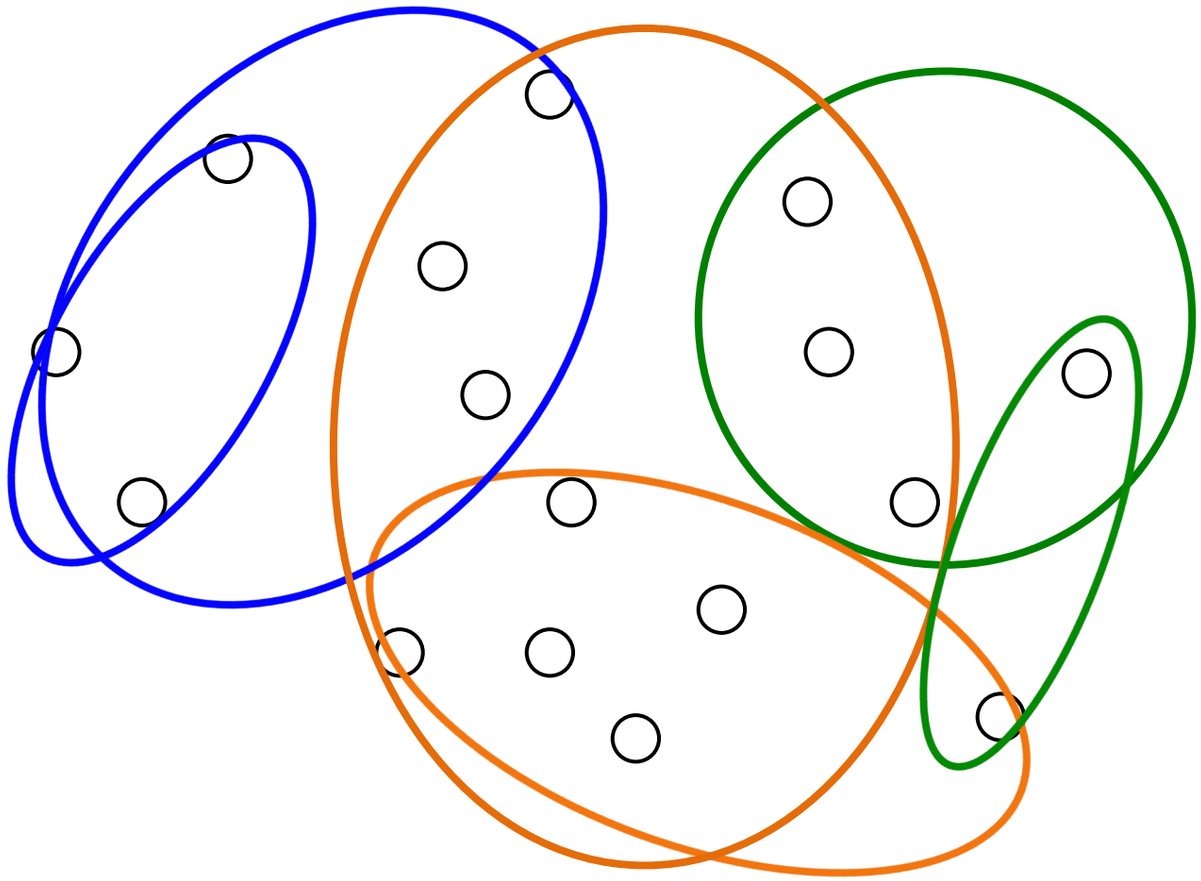
- When there are no labels, this is the Unsupervised Learning setting.

- There are 4 Unsupervised Learning tasks:
  1. Clustering
  2. Embedding
  3. Learning a generative model
  4. Completion (estimating missing values), Approximation

Learning from raw data

What can be done without any labels?

Unsupervised learning



Clustering

Unsupervised learning

There are 4 Unsupervised Learning tasks:

1. Clustering
2. Embedding
3. Learning a generative model
4. Completion (estimating missing values), Approximation

1. Clustering: Reduce the (effective) number of data points, by partitioning points into "clusters." E.g.,

- Learning a mixture of Gaussians (Bayesian setting, EM algorithm)
  - This was also Learning a generative model
- Approximation algorithms for clustering
- Deep unsupervised learning for clustering

…

Unsupervised learning

There are 4 Unsupervised Learning tasks:

1. Clustering
2. Embedding
3. Learning a generative model
4. Completion (estimating missing values), Approximation

2. Embedding: Map into a different feature space

- Dimensionality reduction: reduce number of dimensions
- Feature selection (special case of the above), multiple methods
- Spectral methods
- Deep unsupervised learning for embedding
- Dimension can increase OR decrease, depending on design of architecture

Unsupervised learning

There are 4 Unsupervised Learning tasks:

1. Embedding
2. Clustering
3. Learning a generative model
4. Completion (estimating missing values), Approximation

3. Learning a generative model

- Generative model section of course: unsupervised: EM to learn a mixture model
- Deep unsupervised learning
    - Variational Autoencoder (VAE)
    - Generative Adversarial Network (GAN)

Deep Unsupervised Learning

Tasks | DL Models

- Embedding: Autoencoders, including VAE
- Clustering: Unsupervised DL for clustering
- Learning a generative model: VAE, GAN
- Approximation (Dictionary Learning) Autencoders

Deep Unsupervised Learning

- Supervised DL. Loss is a function of the label, y, and the network's output on input x.

Network output $f_W(x) = \hat{y}$

Loss function $\mathcal{L}(\hat{y}, y)$

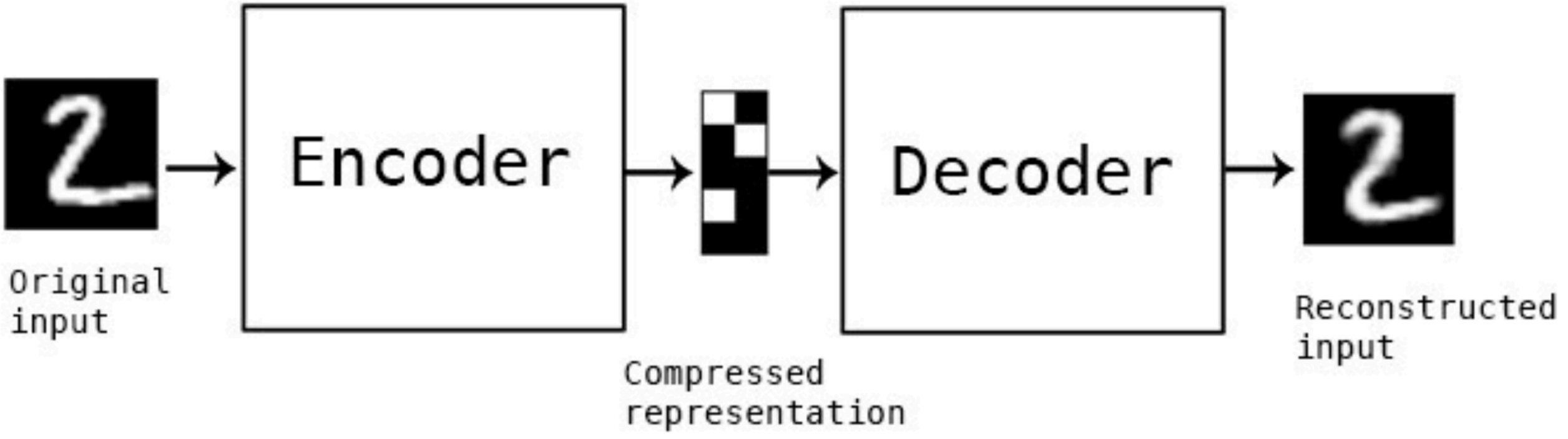
- Unsupervised DL. Loss is only a function of x, and the network's output on input x. There is no label, y.

Network output $f_W(x) = \hat{x}$

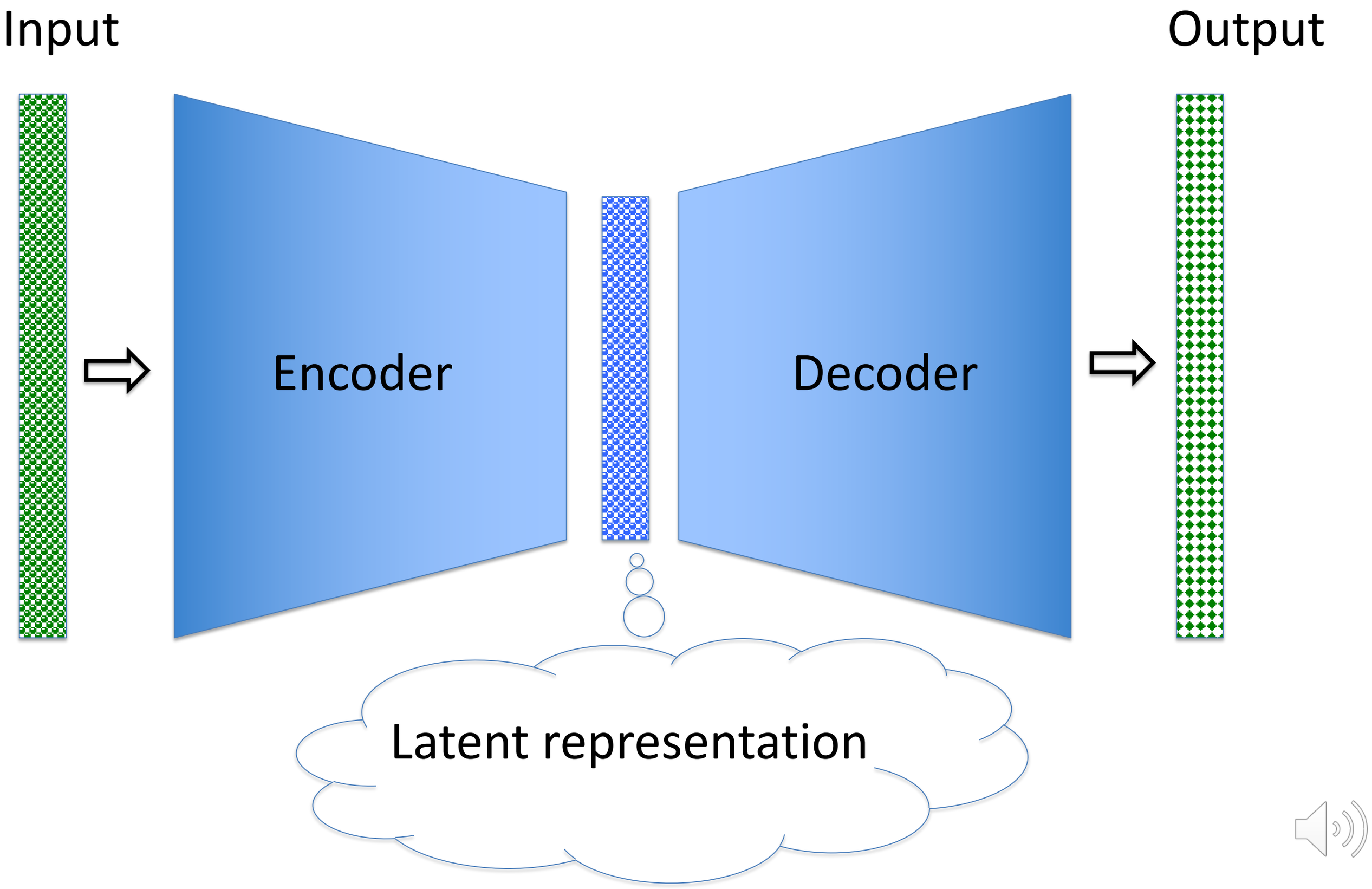
Loss function $\mathcal{L}(\hat{x}, x)$

Intro to Auto-encoders

- Want to train a neural-network in an unsupervised setting
  - Use the unlabeled data both as input, and to evaluate the output
- Hidden layer will learn feature representations of the input.



Autoencoder: The parameters of the encoder and decoder networks are trained to make the output approximate the input. After training on many input examples, the parameters of the bottleneck layer form a (compact) representation of the input distribution.



Autoencoders

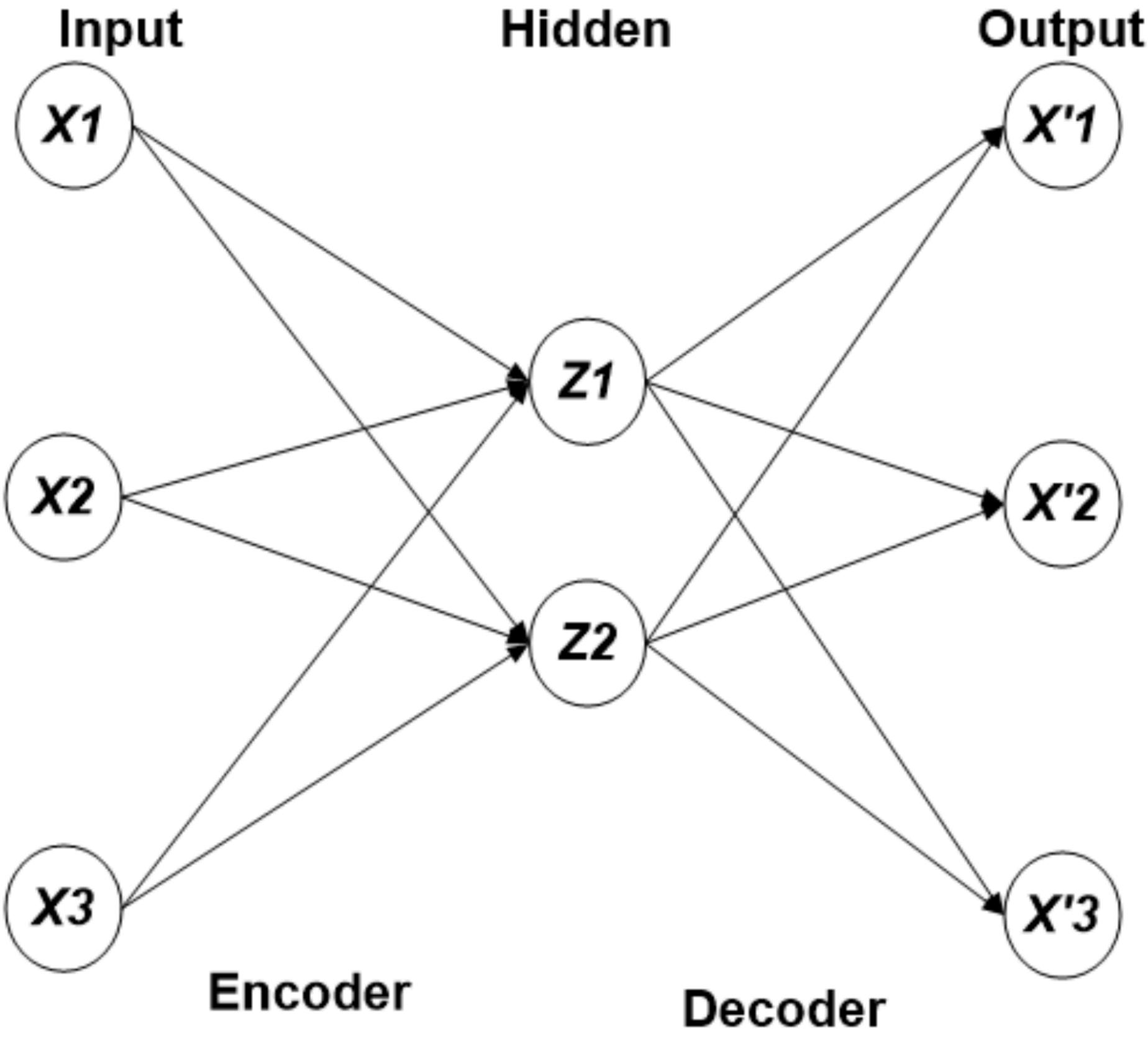
- Dense Autoencoder: the width of the bottleneck layer (the encoding) is smaller than the input width
  - So the embedding dimension is smaller than the input dimension (*i.e.* Dimensionality reduction)
  - So we get **compact** embeddings.

- Wide Autoencoder: the width of the encoding is larger than the input width
  - So the embedding dimension is larger than the input dimension (*Cf.* Kernel expansions)

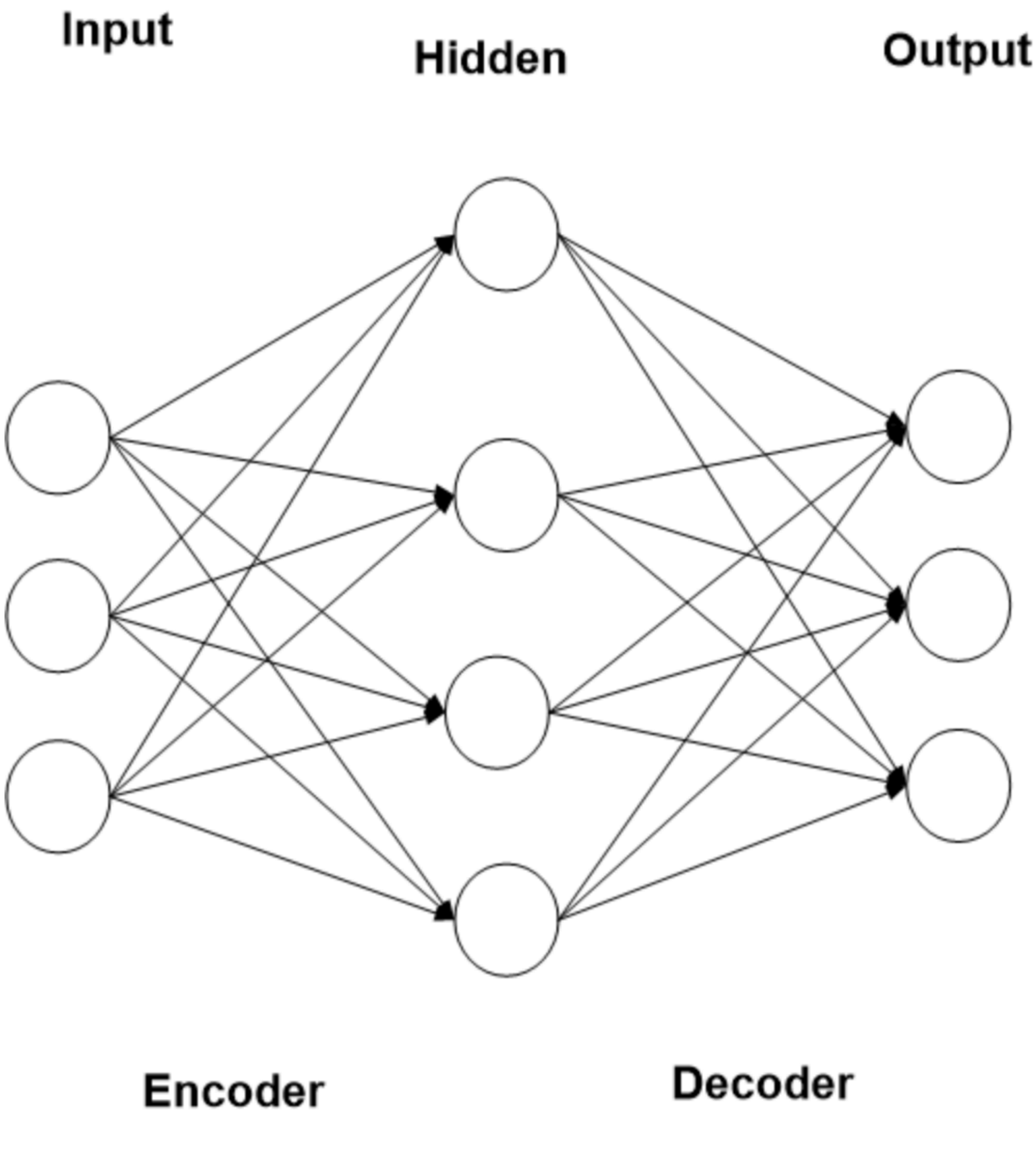
Q: Why would we want to do this?

A: We will get **sparse** embeddings. *Cf.* perception in the brain.

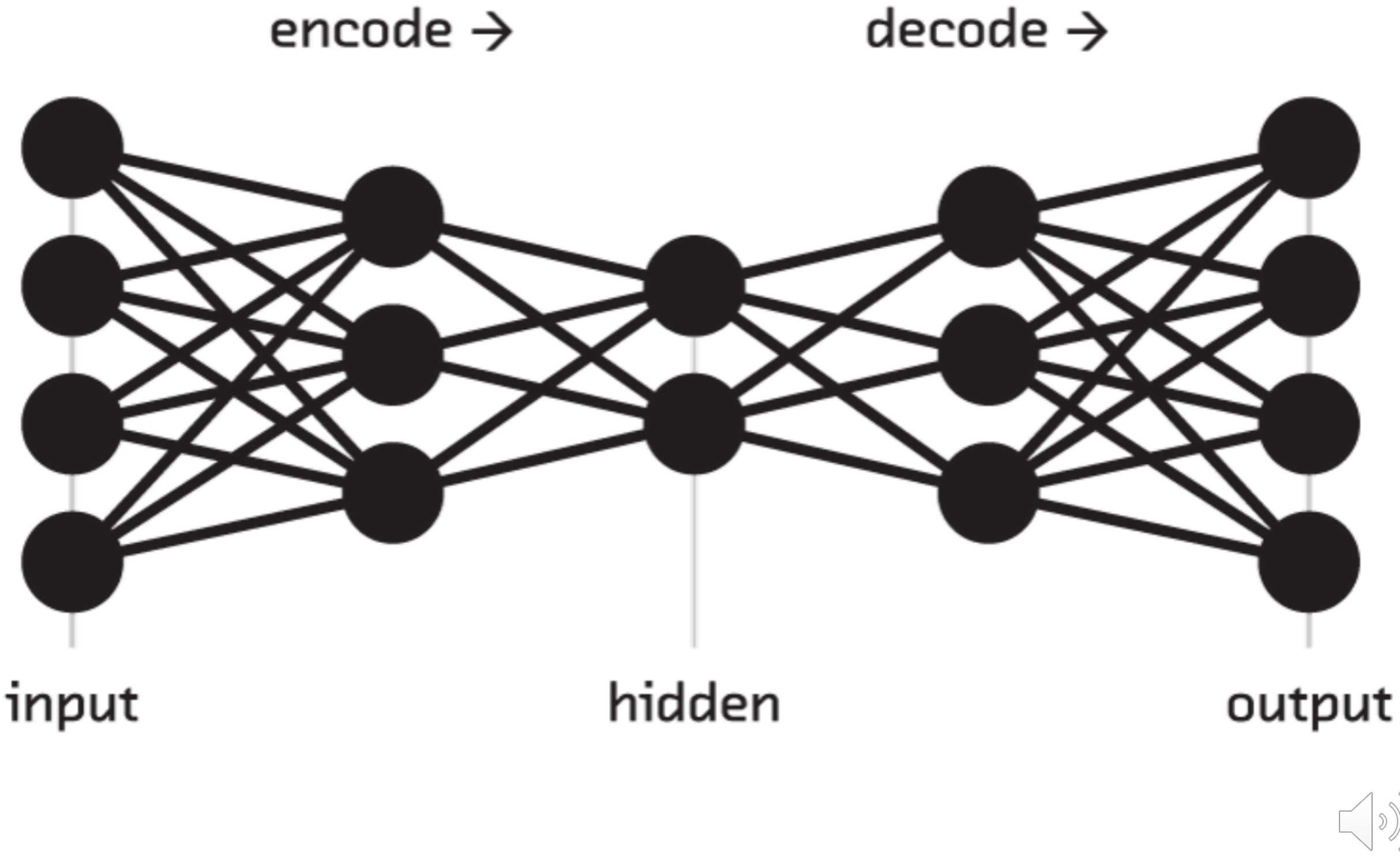
Dense Autoencoder



Wide Autoencoder



Stacked Autoencoder



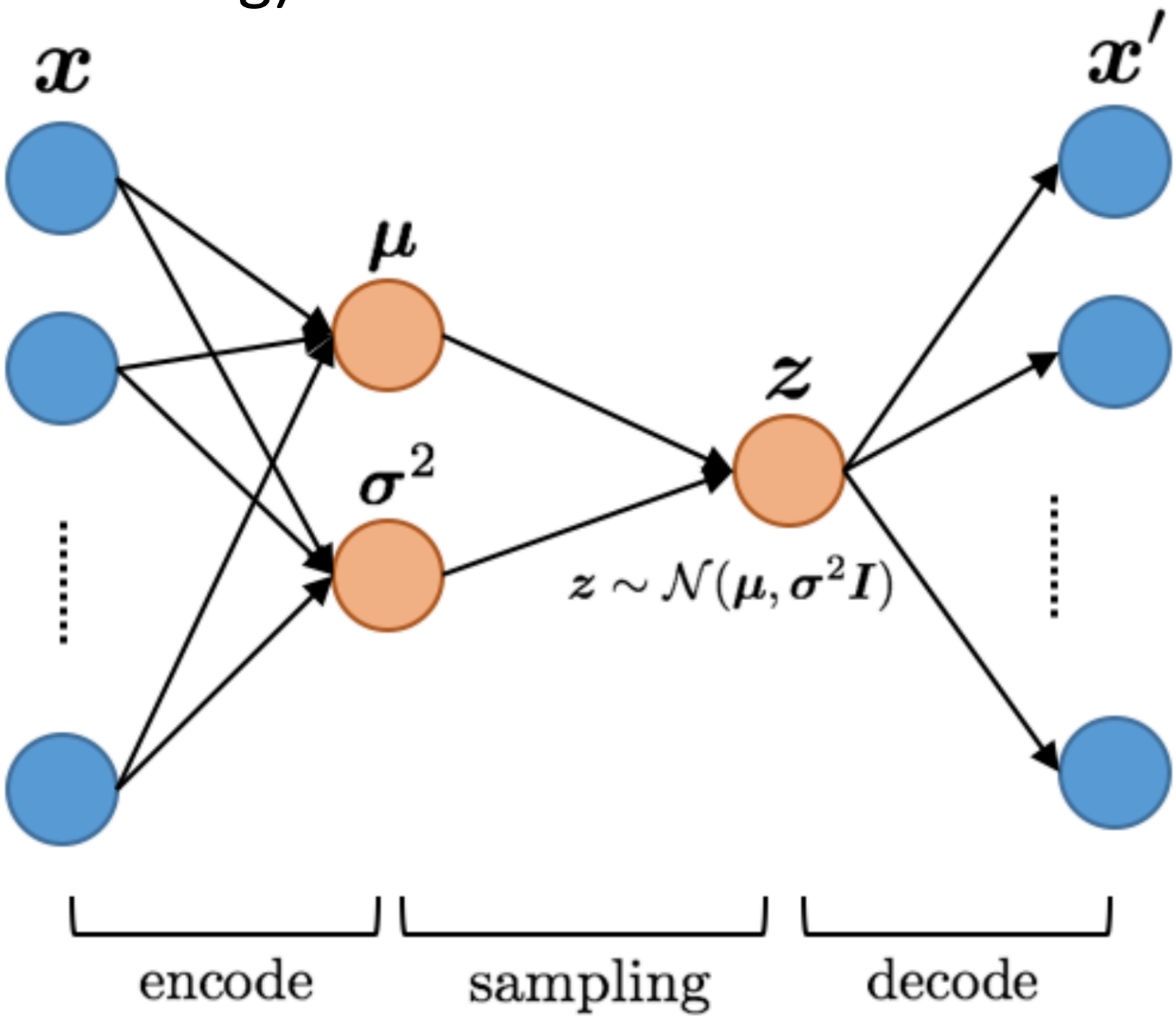
Generative Unsupervised DL

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GANs)

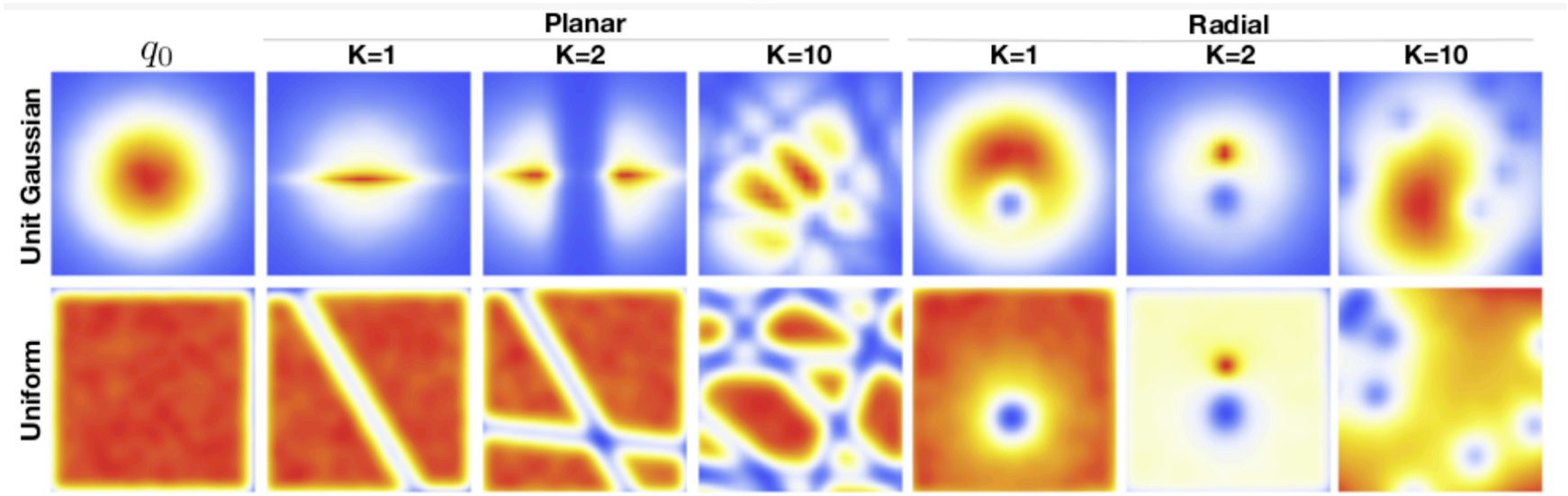
- The trained models can be used for:
  - Data generation: generating data similar to the training data
  - Anomaly detection: detecting when an example is different from the training data

Variational Autoencoder (VAE)

Learn a distribution over latent representations (instead of a single encoding).



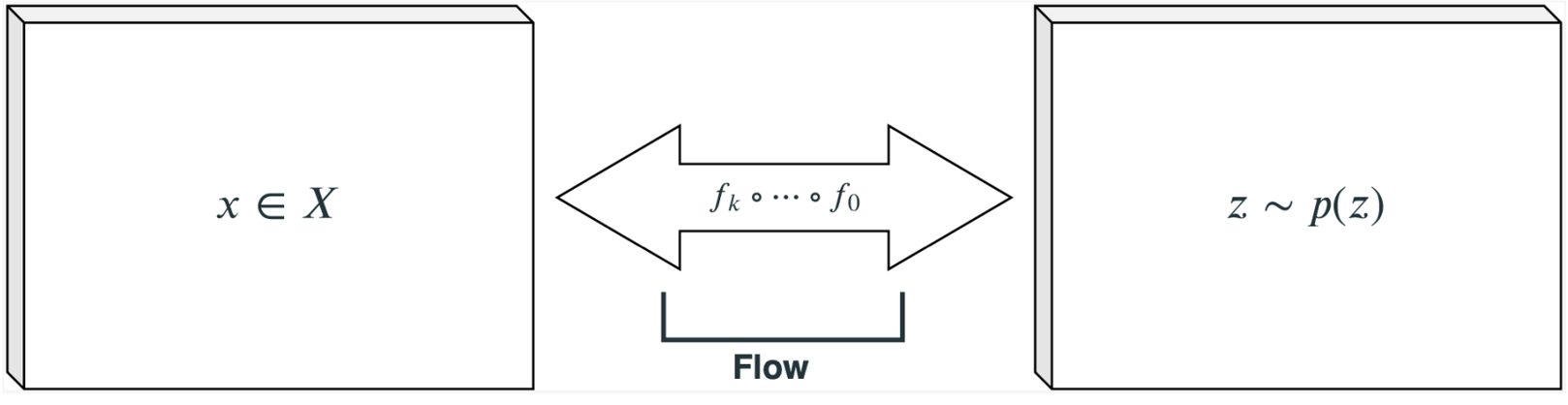
Normalizing Flows



[Rezende & Mohamed, 2015]

Learn a series of invertible transformations, $\{f_i\}$, from a simple prior on Z, to allow for more informative distributions on the latent space:

$$z_k = f_k \circ f_{k-1} \circ \cdots \circ f_1(z_0)$$



GANs: Generative Adversarial Networks

Two (2) neural networks are learning simultaneously, in competition with each other.

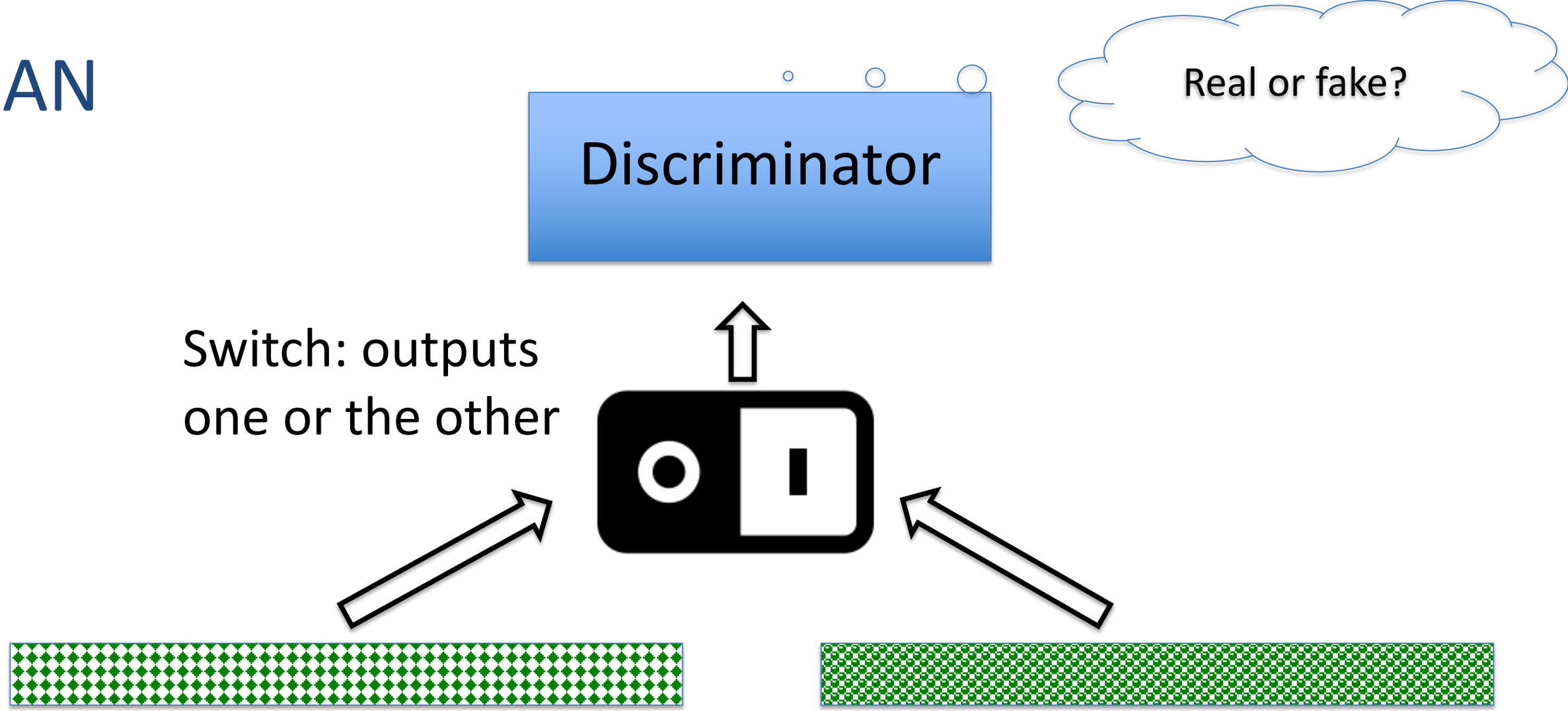
- The Generator, G, is learning to generate examples similar to the training data.

- The Discriminator, D, is learning to classify examples as
  - Real (from the training data)

    OR

  - Fake (generated by G).

GAN



Generative Adversarial Networks (GANs): The parameters of the Generator and Discriminator networks are trained jointly: D tries to discriminate between fake data and real; G tries to fool D. After training on many input examples, the Generator can be used to synthesize examples from the input distribution, and the Discriminator can be used to detect outliers.

Learning a GAN

- Generator, G, maps latent variable z, to data space.
  - Defined by network weight parameters W_g: $G(z; W_g)$

- Discriminator, D: X → R, outputs a probability that its input was a real data instance.
  - Defined by network weight parameters W_d: $D(x; W_d)$

- Fit the W's by optimization defined over both G and D:

$\min_G \max_D V(D, G)$, where:

$$V(D,G) = E_{x \sim P_{\text{data}}(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$

log-prob. that D **correctly** predicts **Real**

log-prob. that D **correctly** predicts **Fake**

Learning a VAE

- Encoder is a distribution, q, maps data space to latent space.
  - Defined by network weight parameters θ: $q_\theta(z|x)$

- Decoder is a distribution, p, maps latent space to data space.
  - Defined by network weight parameters ϕ: $p_\phi(x|z)$

- Fit the network weights θ and ϕ, by minimizing:

$$E_{q_\theta(z|x)}[\log p_\phi(x|z)] - \beta KL(q_{\theta(z|x)} \| p(z))$$

Reconstruction error on training data

Regularization term: Penalizes distance from a simple Prior

- Optimized using variant of EM, using a variational lower bound.

Unsupervised learning

There are 4 Unsupervised Learning tasks:

1. Clustering
2. Embedding
3. Learning a generative model
4. Completion (estimating missing values), Approximation

- (Sparse) Matrix (or Tensor) Completion
- Matrix Approximation (see also Dictionary Learning, Matrix Factorization, Compressive Sensing)
- Deep Learning approaches
  - Autoencoders for Dictionary Learning
  - DL for in-painting tasks