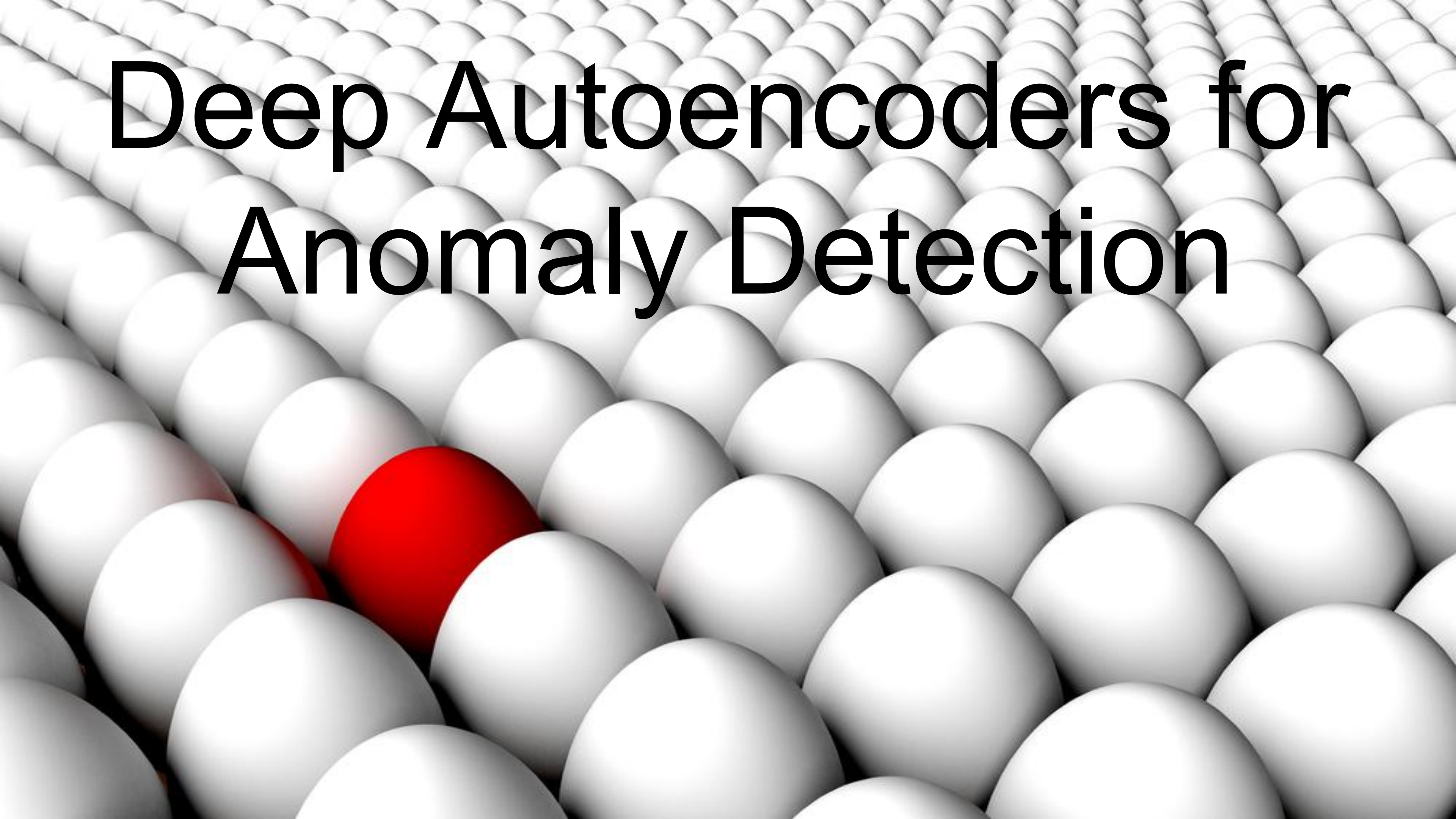
Deep Autoencoders for Anomaly Detection



Auto = “Self”

- Autoencoders utilize self-supervision, a form of unsupervised learning that aims to reconstruct input data
- Formally, an autoencoder approximates the identity function:

$$f(x) = x$$

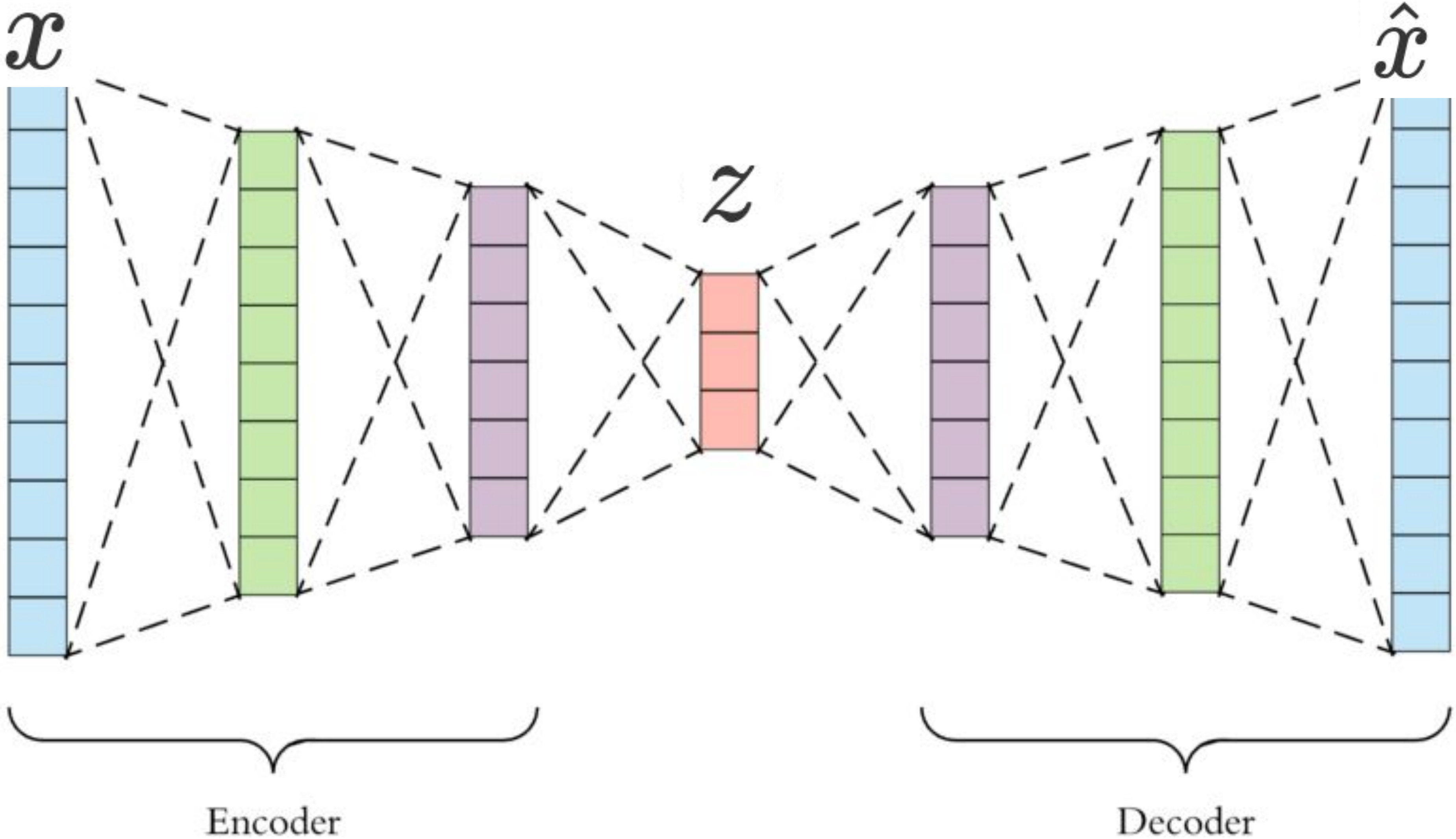
Why Learn a Trivial Function?

The identity function is trivial; why learn it?

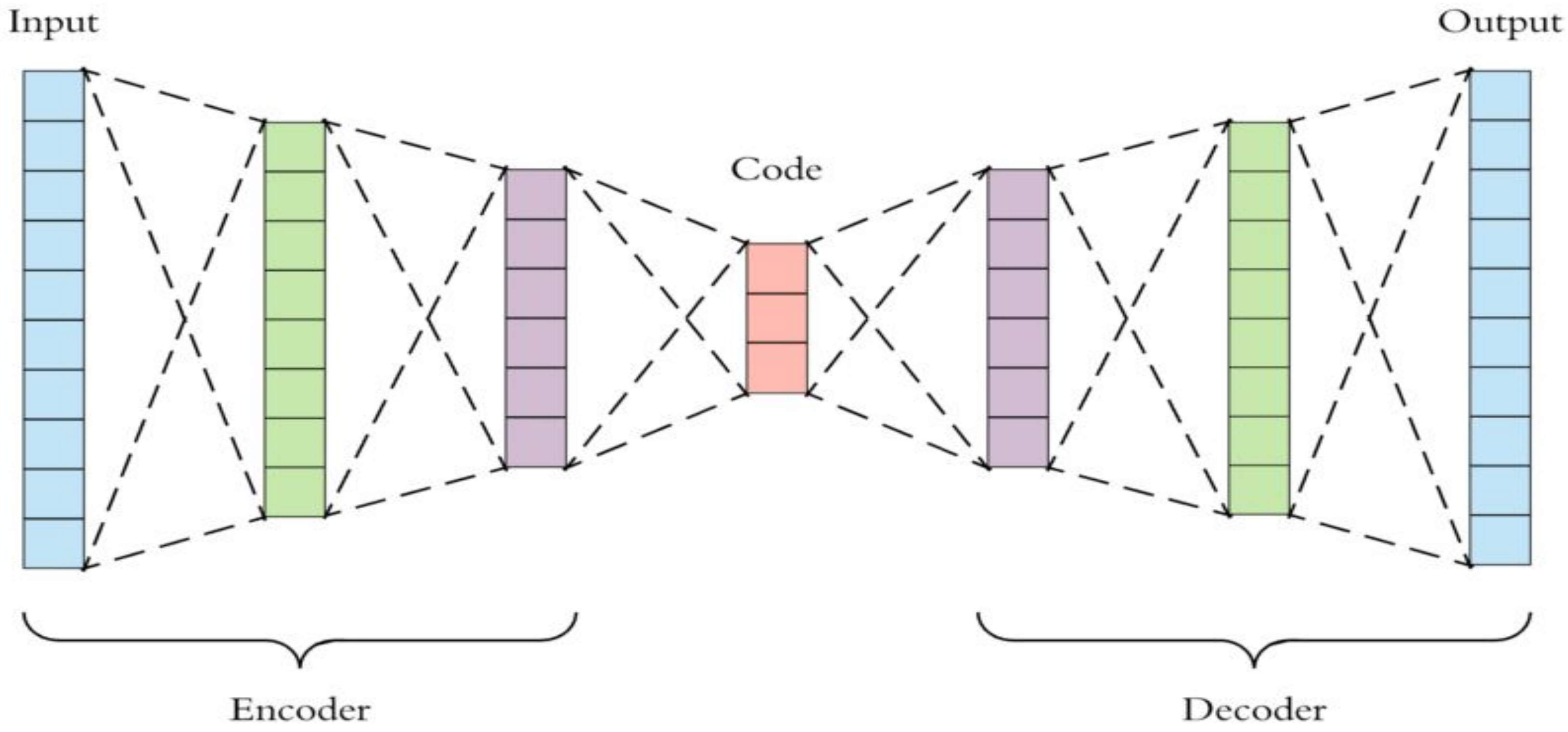
By placing constraints on a neural network architecture, specifically:

$$Z < X$$

Where **X is the raw feature space** and **Z is a latent representation space**, this becomes a non-trivial optimization problem where the model needs to learn some mapping from X to Z that **preserves information about an input x while reducing its dimensionality**.


x
z
$\hat{x}$
Encoder
Decoder

In this image, X is the number of input/output dimensions (X=10) and Z is the number of activation units in the “code” or encoding (Z=3). Note that Z < X.



Neural Network as a Composite Function

- A feed-forward neural network is a **composite function**, that is, it is composed of a **series of functions**.
- Each layer is a function.
- Autencoders exploit this property.

Decomposition of a Feed-Forward NN

$$h(x) = z$$

Decomposition of a Feed-Forward NN

$$h(x) = z$$

$$g(z) = \hat{y}$$

Decomposition of a Feed-Forward NN

$$h(x) = z$$

$$g(z) = \hat{y}$$

$$f(x) = g(h(x)) = \hat{y}$$

Decomposition of an Autoencoder

$$enc(x|\theta_{enc}) = z$$

Decomposition of an Autoencoder

$$enc(x|\theta_{enc}) = z$$

$$dec(z|\theta_{dec}) = \hat{x}$$

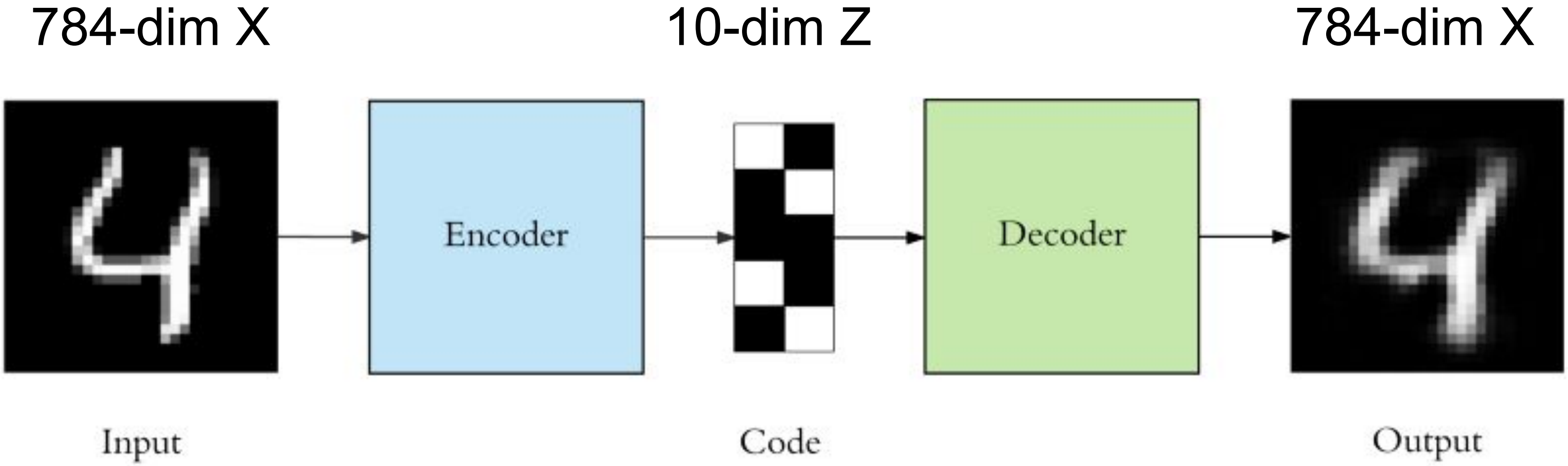
Decomposition of an Autoencoder

$$enc(x|\theta_{enc}) = z$$

$$dec(z|\theta_{dec}) = \hat{x}$$

$$f(x|\theta) = dec(enc(x|\theta_{enc})|\theta_{dec}) = \hat{x}$$

MNIST Example



The 10-dimensional encoding, or “embedding”, contains enough information for the decoder network to approximate the input.

Relation to Supervised Learning

In supervised learning, we learn some approximation of the function

$$F(x) = y$$

Where **x is a set of features** and **y is some known label or value**. The approximate function, or model, is defined by a set of parameters θ:

$$f(x|\theta) = \hat{y}$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

$$L(y, f(x|\theta))$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

$$L(y, f(x|\theta))$$

$$min_\theta \, L(y, f(x|\theta))$$

Self-Supervised Learning

In self-supervised learning, we learn some approximation of the identity function

$$f(x) = x$$

Where **x is a set of features**. This model is also defined by a set of parameters θ:

$$f(x|\theta) = \hat{x}$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

$$L(y, f(x|\theta))$$

$$min_\theta \, L(y, f(x|\theta))$$

Self-Supervised Learning

$$f(x|\theta) = \hat{x}$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

$$L(y, f(x|\theta))$$

$$min_\theta \, L(y, f(x|\theta))$$

Self-Supervised Learning

$$f(x|\theta) = \hat{x}$$

$$L(x, \hat{x})$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

$$L(y, f(x|\theta))$$

$$min_\theta \, L(y, f(x|\theta))$$

Self-Supervised Learning

$$f(x|\theta) = \hat{x}$$

$$L(x, \hat{x})$$

$$L(x, f(x|\theta))$$

Supervised Learning

$$f(x|\theta) = \hat{y}$$

$$L(y, \hat{y})$$

$$L(y, f(x|\theta))$$

$$min_\theta \, L(y, f(x|\theta))$$

Self-Supervised Learning

$$f(x|\theta) = \hat{x}$$

$$L(x, \hat{x})$$

$$L(x, f(x|\theta))$$

$$min_\theta \, L(x, f(x|\theta))$$

Applications

- Feature extraction
- De-noising
- Anomaly Detection
- Image Searches
- Image generation
- Image colorization
- Pixel Interpolation (increasing resolution of images)
- Synthetic Data generation
- Many more

Anomaly Detection

Learn some

$$p(x) \approx P(x)$$

Where P(x) is the true probability distribution of some source of data, p(x) is a learned model. When

$$p(x) << 1$$

(or p(x) is very close to 0) for some x, x is considered anomalous

Reconstruction Loss Relation to p(x)

$$L(x, \hat{x}) \propto \frac{1}{p(x)}$$

Intuitively, autoencoders are good at reconstructing “typical” examples; they struggle to reconstruct anomalous examples.

Autoencoder for Anomaly Detection

- Train autoencoder on a training set to convergence
- Define distribution of reconstruction loss on test set
- Define some threshold based on test set loss distribution
- For example, mean loss + 2 standard deviations
- Take new example, pass through autoencoder, compute reconstruction loss
- If loss > threshold, raise alarm
- Alternatively, inspect top N new examples ranked by reconstruction loss

Code Examples

- Credit Card Fraud Detection
- Handwritten Letter Detection with E-MNIST