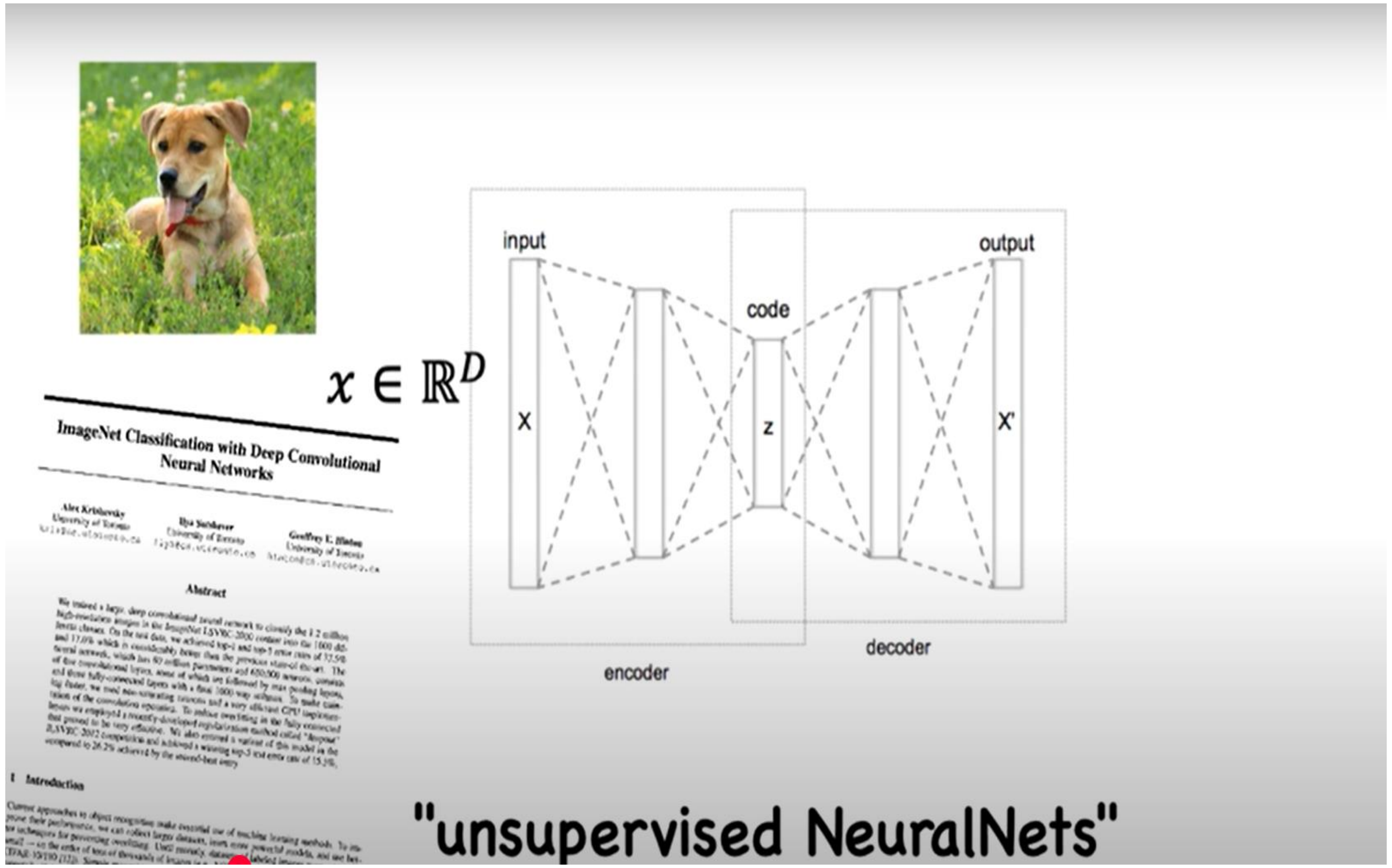
Autoencoders

Autoencoders

- Autoencoders are designed to reproduce their input, especially for images.
  - Key point is to reproduce the input from a learned encoding.



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, University of Toronto
Ilya Sutskever, University of Toronto
Geoffrey E. Hinton, University of Toronto

Abstract

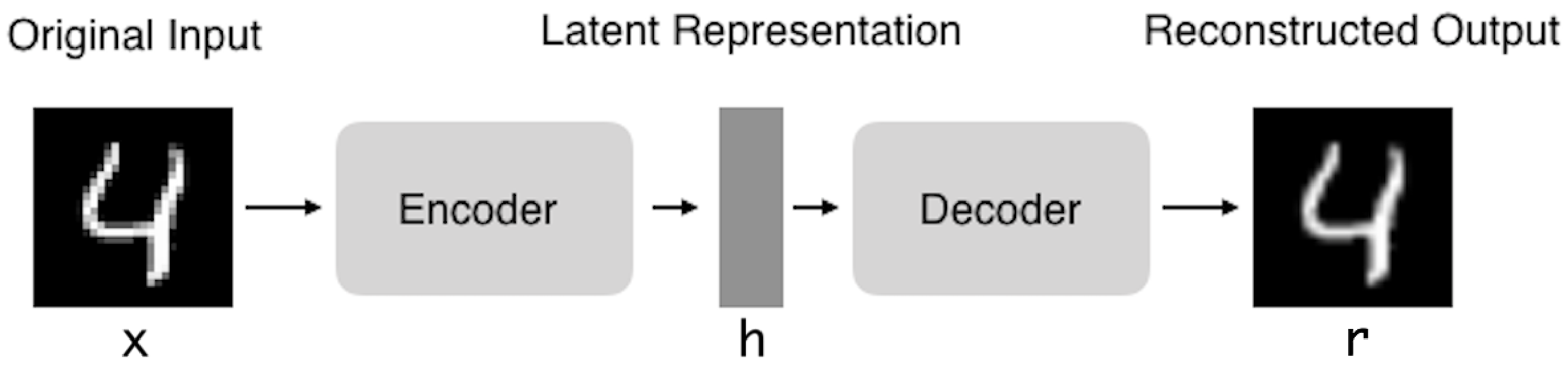
[illegible]

1 Introduction

[illegible]

Autoencoders: structure

- Encoder: compress input into a latent-space of usually smaller dimension. h = f(x)
- Decoder: reconstruct input from the latent space. r = g(f(x)) with r as close to x as possible



Autoencoders

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

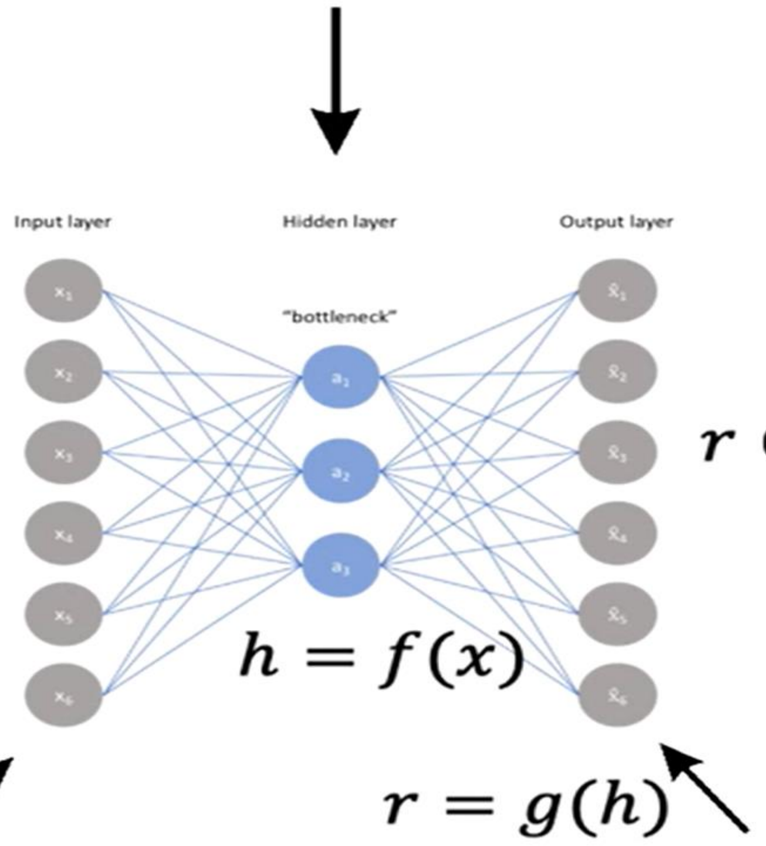
Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is

$x \in \mathbb{R}^D$



$h = f(x)$

$r = g(h)$

$\min Loss(x, r)$

$r \in \mathbb{R}^D$

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is

Autoencoders

Encoding

$$\mathbf{Z = W1 * X + b1}$$

where X is the input data, W1 is the encoding weight matrix, b1 is the encoding bias vector, and Z is the code.

Decoding:

$$\mathbf{X' = W2 * Z + b2}$$

where W2 is the decoding weight matrix, b2 is the decoding bias vector, and X' is the reconstructed data.

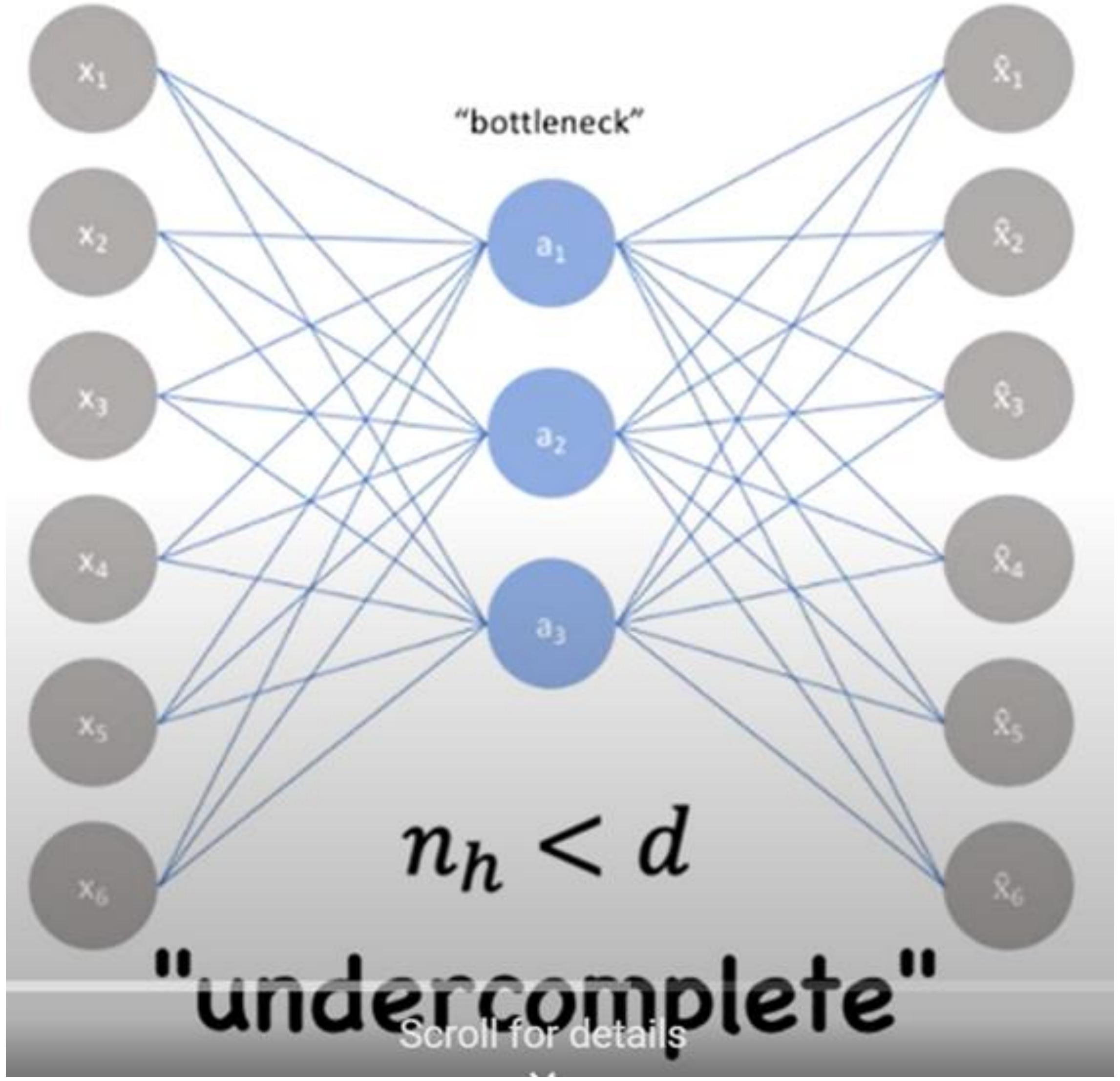
The goal of the linear autoencoder is to minimize the reconstruction error between X and X’.

One common way to do this is to use the mean squared error (MSE) loss function

$$\text{loss} = 1/n * \sum(X - X')^2$$

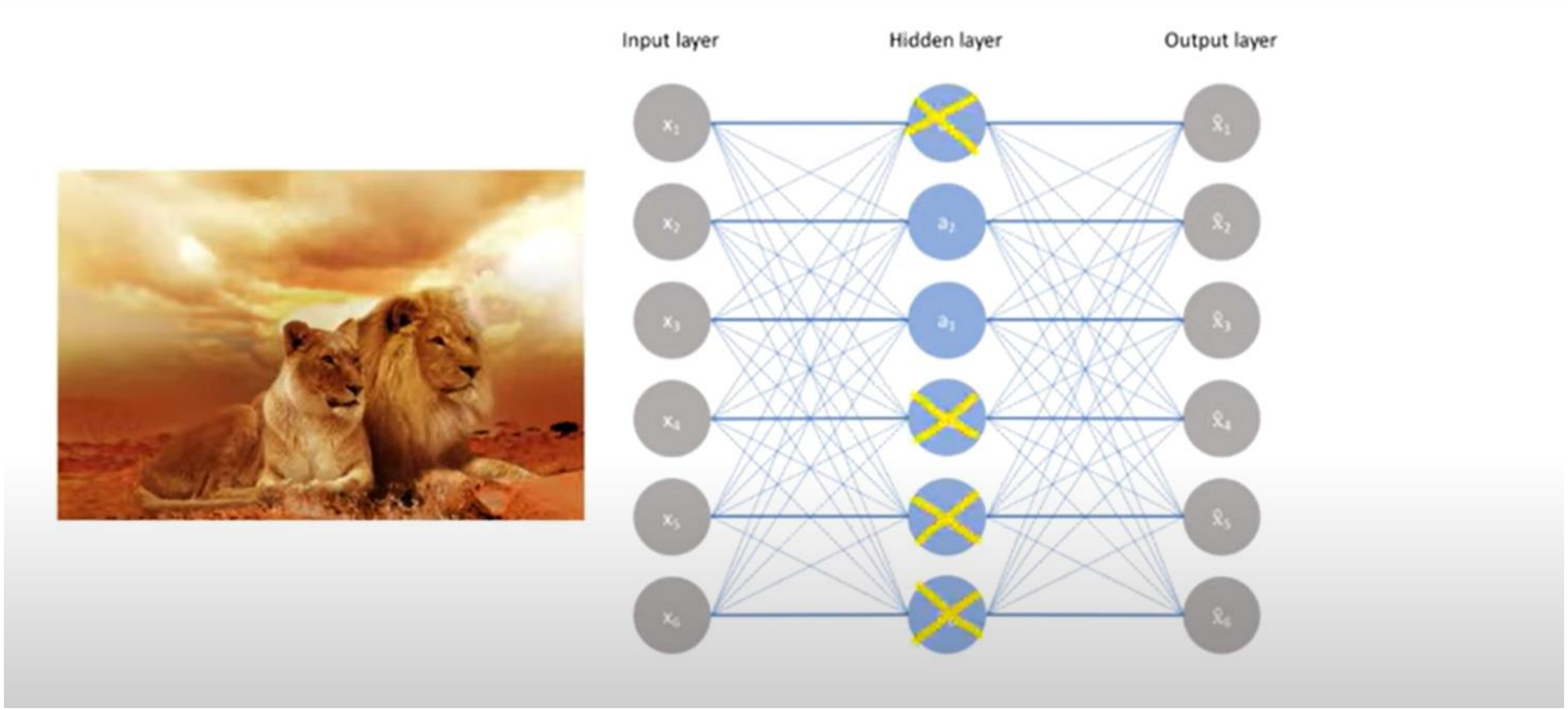
Under complete autoencoder

- The under complete autoencoder is forced to learn a bottleneck or compressed representation that cannot perfectly reconstruct the original input.
- The idea behind under complete autoencoders is to learn a more efficient and concise representation of the input data by capturing the most important and relevant information.
- The size of the code is smaller than the size of the input,
- Restricting the size of the code, the network is encouraged to learn a compressed representation that captures the most important features of the data and discards irrelevant information.
- undercomplete autoencoder can be used to reduce the dimensionality of the data, which can be useful for visualization, feature extraction, or as input to other machine learning models.
- undercomplete autoencoders can be difficult to train, especially if the code is too small or the data is high-dimensional.
- Regularization techniques such as L1 or L2 regularization, dropout, or sparsity constraints can be used to prevent overfitting and encourage the network to learn more informative representations.



Sparse Autoencoder

Sparse Autoencoder – Uses sparsity constraints to force only a few neurons to be active, improving feature learning.



Overcomplete encoder

- An overcomplete autoencoder is a type of autoencoder that is trained to learn a compressed representation of the input data, where the size of the code is larger than the size of the input. In other words, the overcomplete autoencoder is capable of representing the input data in a more expressive way than an undercomplete autoencoder.

- overcomplete autoencoders is to learn a more rich and complex representation of the input data by allowing the code to have more dimensions than the input.

- Increasing the number of neurons in the hidden layer or layers, the network can learn to capture more complex patterns and features in the data.

- Overcomplete autoencoders can be useful for tasks such as denoising, where the goal is to remove noise or corruption from the input data.

- By learning a more expressive representation of the input, the overcomplete autoencoder can better differentiate between noise and signal, leading to improved denoising performance.

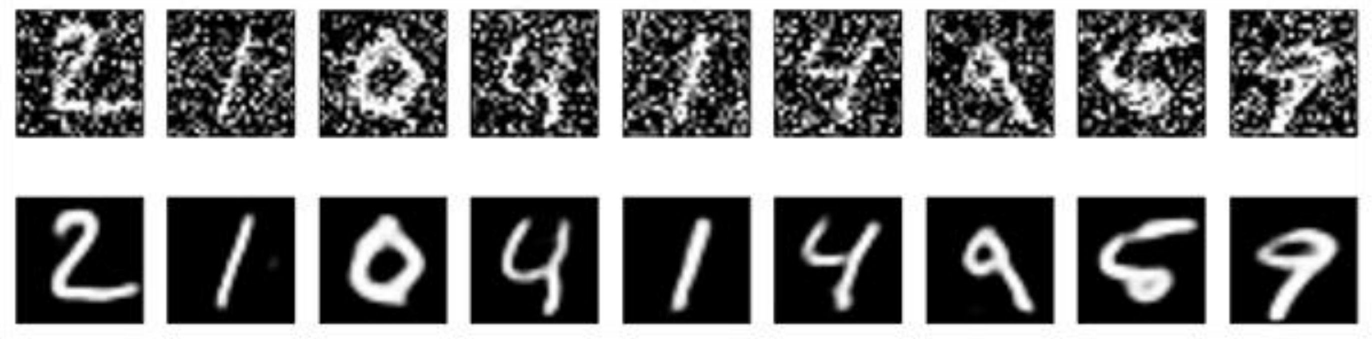
Properties of Autoencoders

- **Data-specific**: Autoencoders are only able to compress data similar to what they have been trained on.
- **Lossy:** The decompressed outputs will be degraded compared to the original inputs.
- **Learned automatically from examples:** It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

Denoising autoencoders

Denoising Autoencoder – Trained with noisy inputs to learn robust feature representations.

- Basic autoencoder trains to minimize the loss between x and the reconstruction g(f(x)).
- Denoising autoencoders train to minimize the loss between x and g(f(x+w)), where w is random noise.
- Same possible architectures, different training data.
- Kaggle has a dataset on damaged documents.



Denoising autoencoders

- Denoising autoencoders are a type of autoencoder that is trained to remove noise from corrupted input data.
- The goal of denoising autoencoders is to learn a representation of the input data that captures the underlying structure and removes the noise.
- The basic architecture of a denoising autoencoder is similar to a regular autoencoder, with an encoder that maps the input data to a compressed representation (code), and a decoder that maps the compressed representation back to the original input.
- Denoising autoencoders have been used in a variety of applications, such as image denoising, speech denoising, and text denoising.

Applications of Autoencoders

- **Dimensionality Reduction** (alternative to PCA)
- **Anomaly Detection** (e.g., fraud detection, medical diagnosis)
- **Denoising Images**
- **Feature Learning** for downstream ML tasks
- **Generative Modeling** (e.g., VAEs for image synthesis)