



AutoEncoders

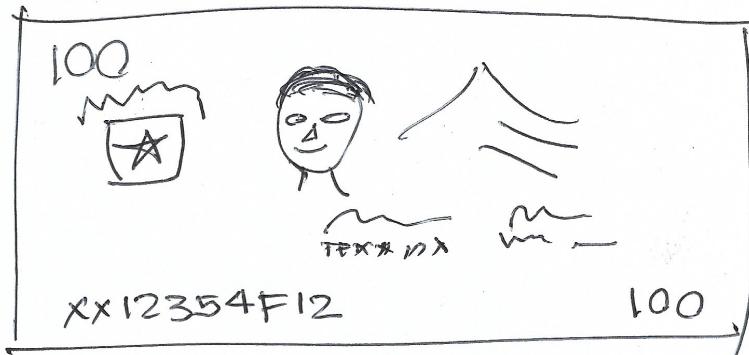
Rowel Atienza, Ph.D.
University of the Philippines
github.com/roatienza
2022

Exercise

Draw an image of a 100-peso bill

Draw from what you can remember only

Do not look and copy a 100-peso bill



AutoEncoder

A self-supervised/unsupervised network

Supply it with data, it will figure out the mapping from input to output

A neural network that attempts to copy its input to its output

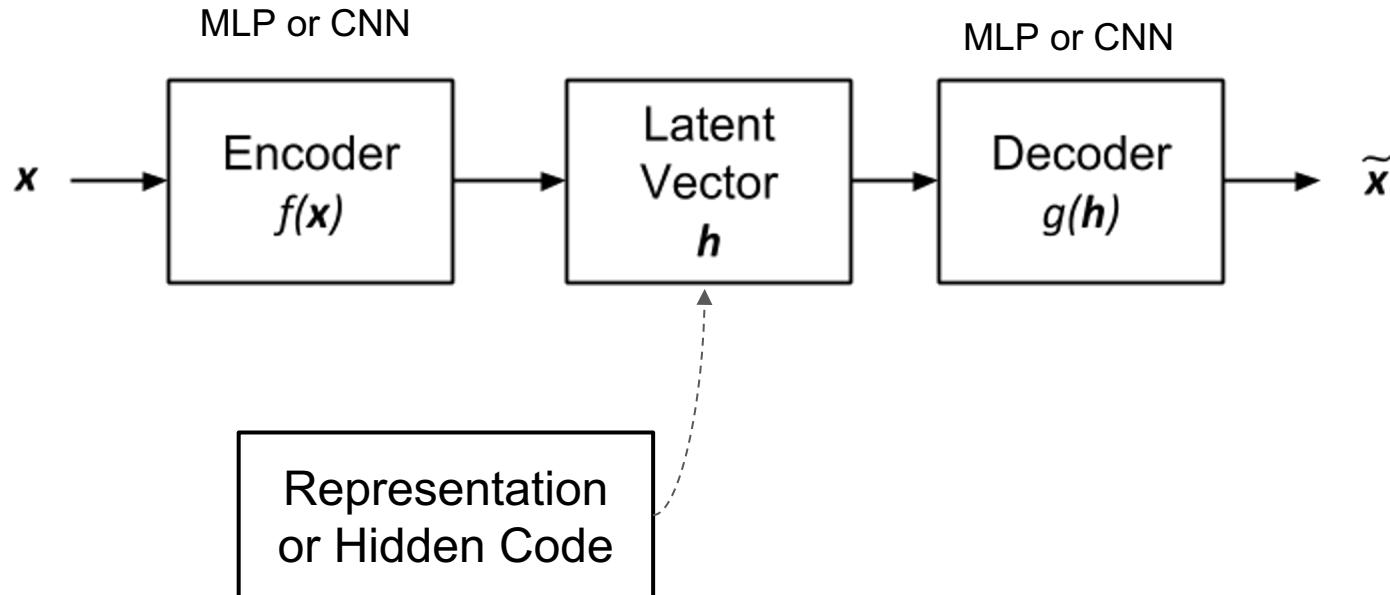
In doing so, it learns a hidden code, h , that represents its input

The network has 2 parts

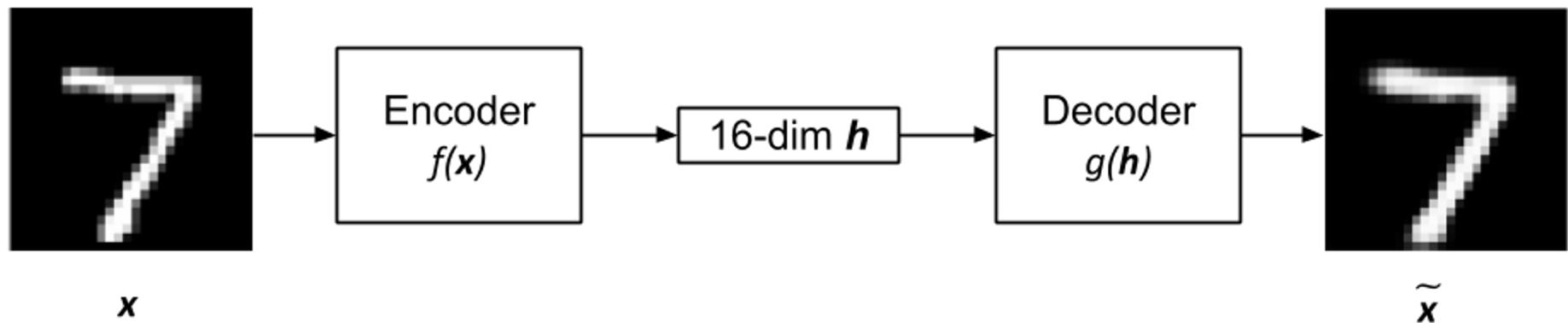
An encoder: $h = f(x)$

A decoder: $r = g(h)$

AutoEncoder



AutoEncoder



AutoEncoder

Autoencoder must not learn: $g(\mathbf{h}) = g(f(\mathbf{x})) = \mathbf{x}$

Instead it must learn to approximate \mathbf{x} only;

In doing so, it learns special properties of \mathbf{x}

Roughly, our brain compresses (lossy) information about what it senses



AutoEncoder

For example in MNIST, properties could be writing style, tilt, thickness, roundness of stroke, etc.

All properties needed to represent digits 0 to 9.

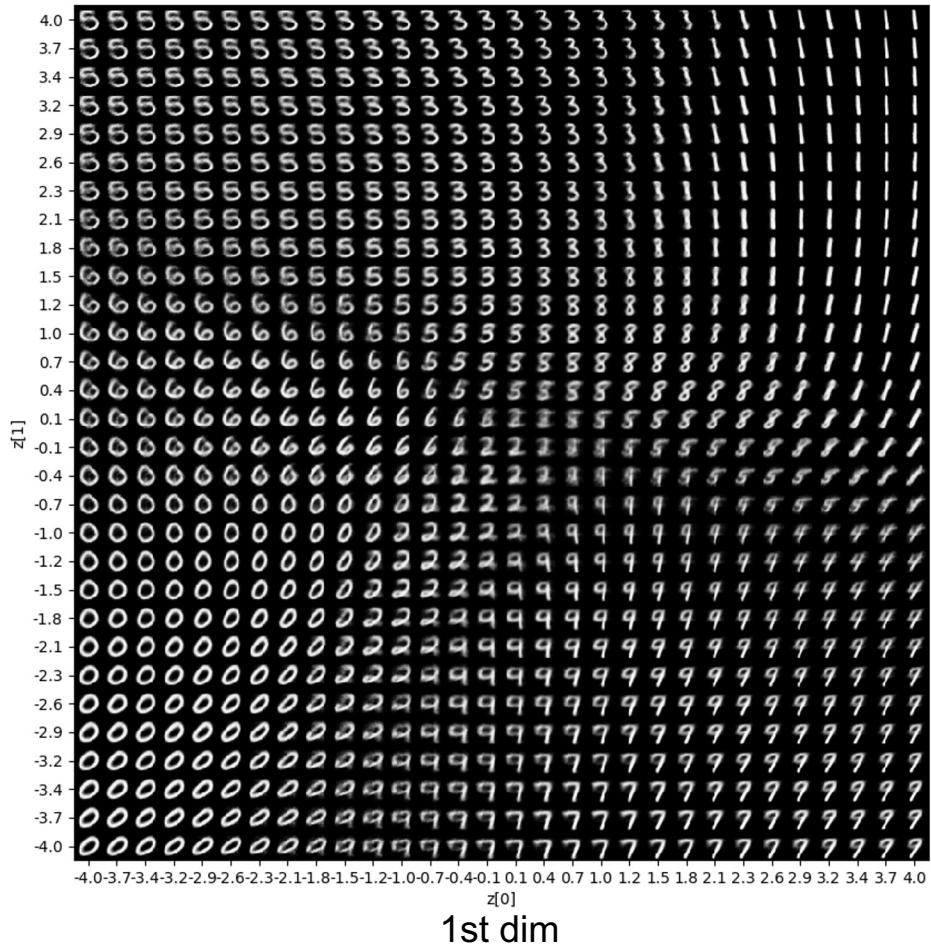
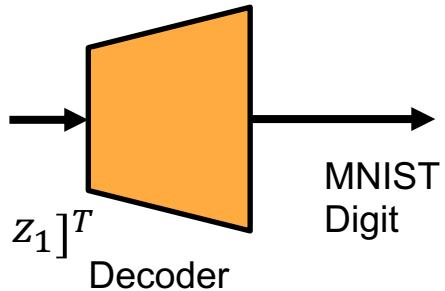
Autoencoder is not an identity function

Applications

AutoEncoder Decoder as a Generative Model

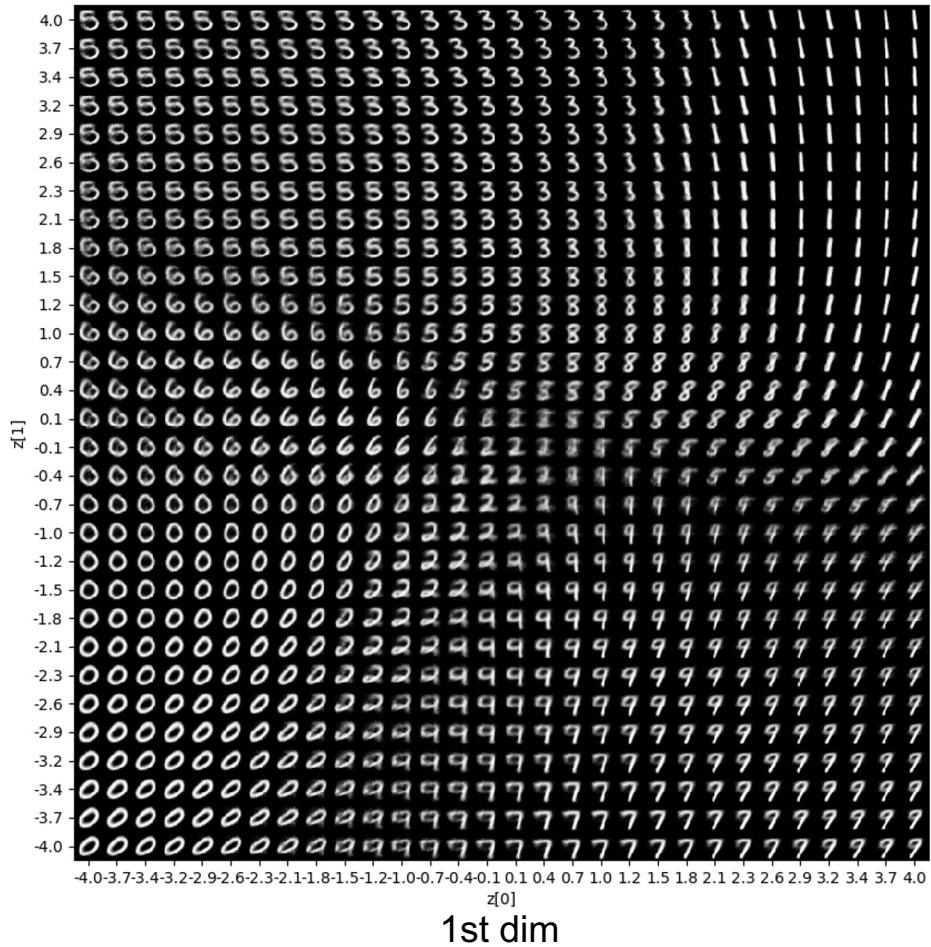
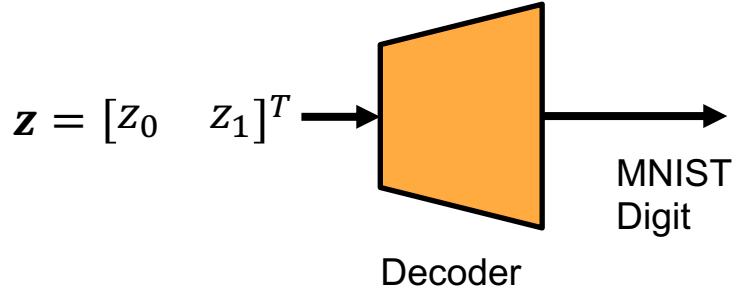
Digits over latent code (2-dim)

Anywhere we navigate the latent space, there is a corresponding decodable MNIST digit-like image.



Application

The decoder is a generative model

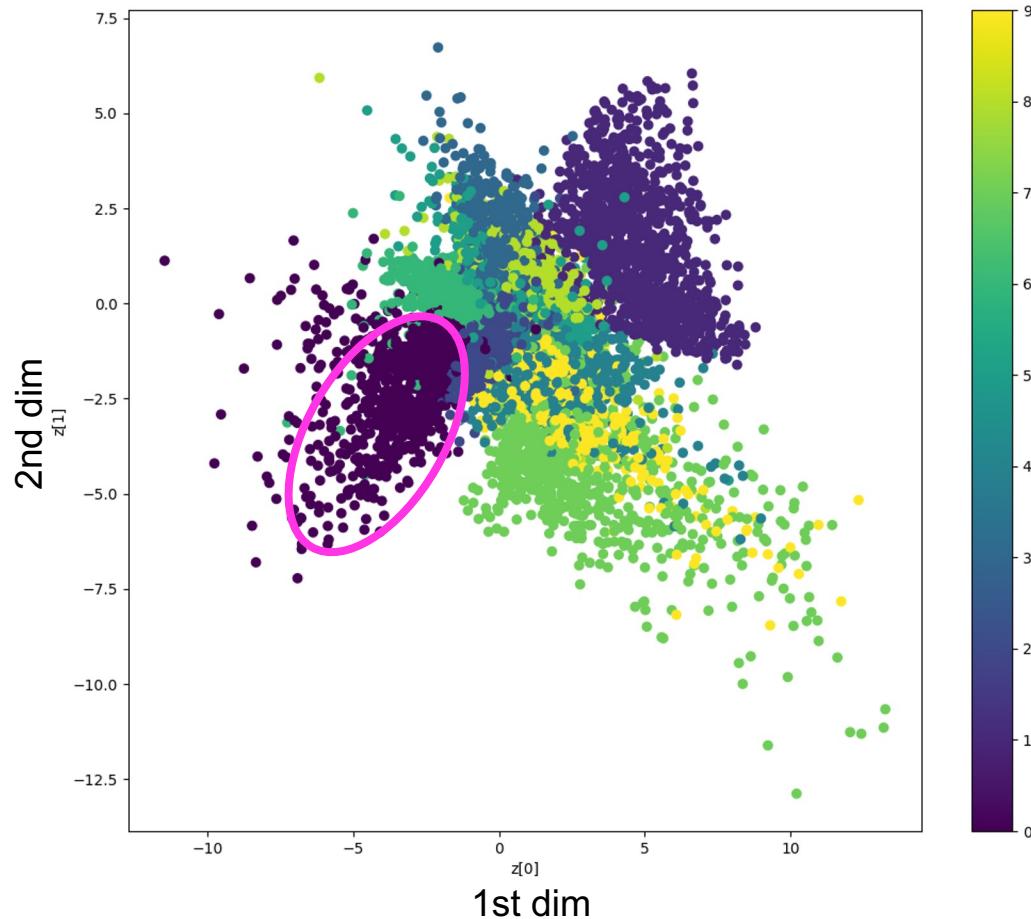


Applications

AutoEncoder Encoder as a
Clustering Network

Digits over latent code (2-dim)

Similar digits appear to cluster in one region.

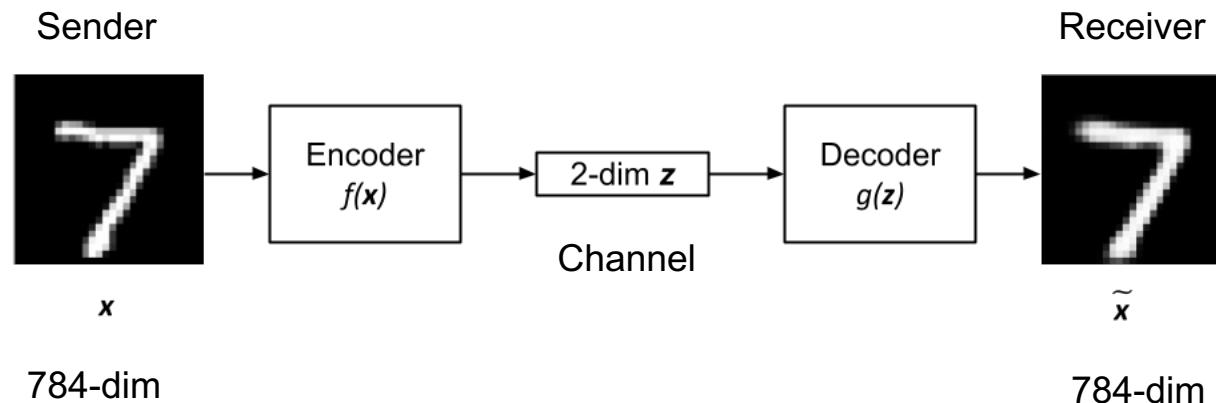


Other Applications

Dimensionality reduction (compression)

Feature learning

Many more...



The hidden code, h

Dimensionality is much less than x

For example, MNIST x could be $28 \times 28 = 784$ but h could be 2 or 4

The small dimension of h forces the neural network to learn only the most important features of x

How to train an autoencoder

The learning process is described by minimizing a loss function

$$L(x, g(f(x)))$$

L penalizes g as being dissimilar from x

Such as mean squared error (MSE)

AutoEncoder vs PCA

Principal Component Analysis (PCA) or SVD can only learn linear feature representation

Autoencoder can learn both linear and non-linear feature representations

Stochastic Encoder and Decoder

In AutoEncoder, the input is also the output

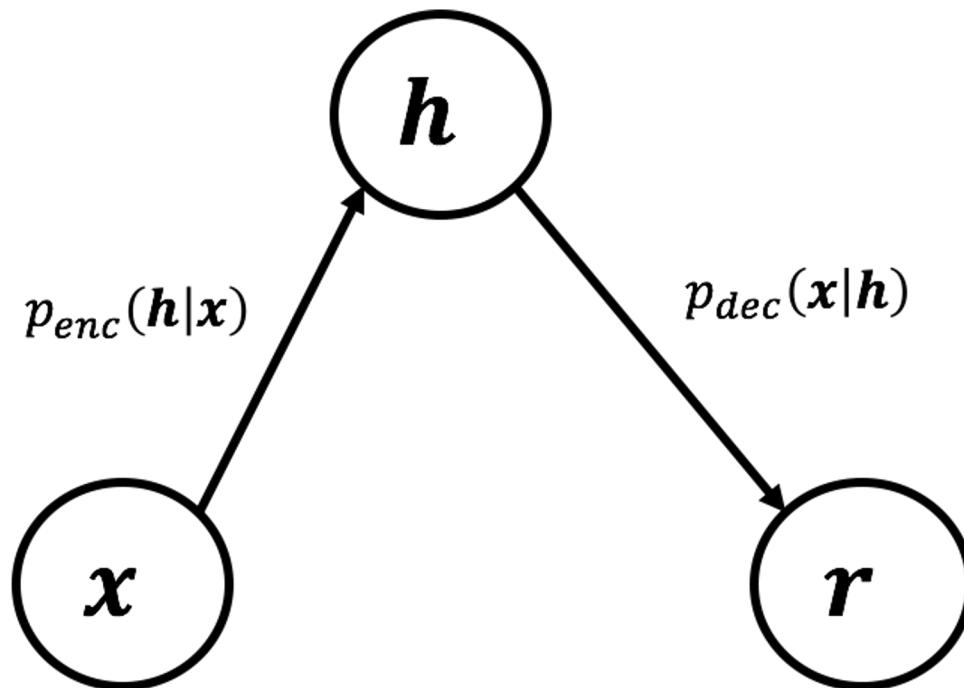
Encoder: $p_{enc}(\mathbf{h} | \mathbf{x})$

Decoder: $p_{dec}(\mathbf{x} | \mathbf{h})$

Similar to other networks, the goal of stochastic autoencoder is to minimize

$$-\log p_{dec}(\mathbf{x} | \mathbf{h})$$

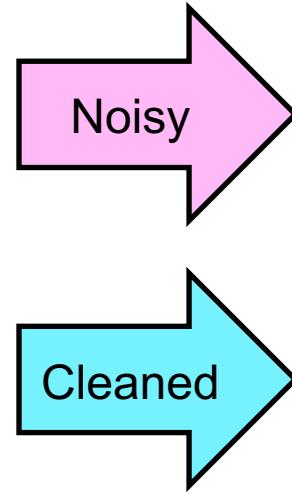
Encoding/Decoding Distribution Graphical Model



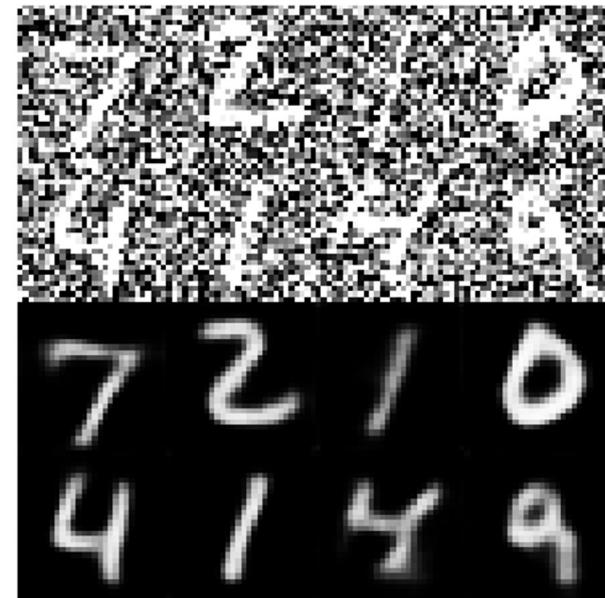
Applications

Denoising AutoEncoder

Can we recover clean data
given noised data?



Corrupted Input: 1st 2 rows, Denoised Output: last 2 rows



Denoising AutoEncoder

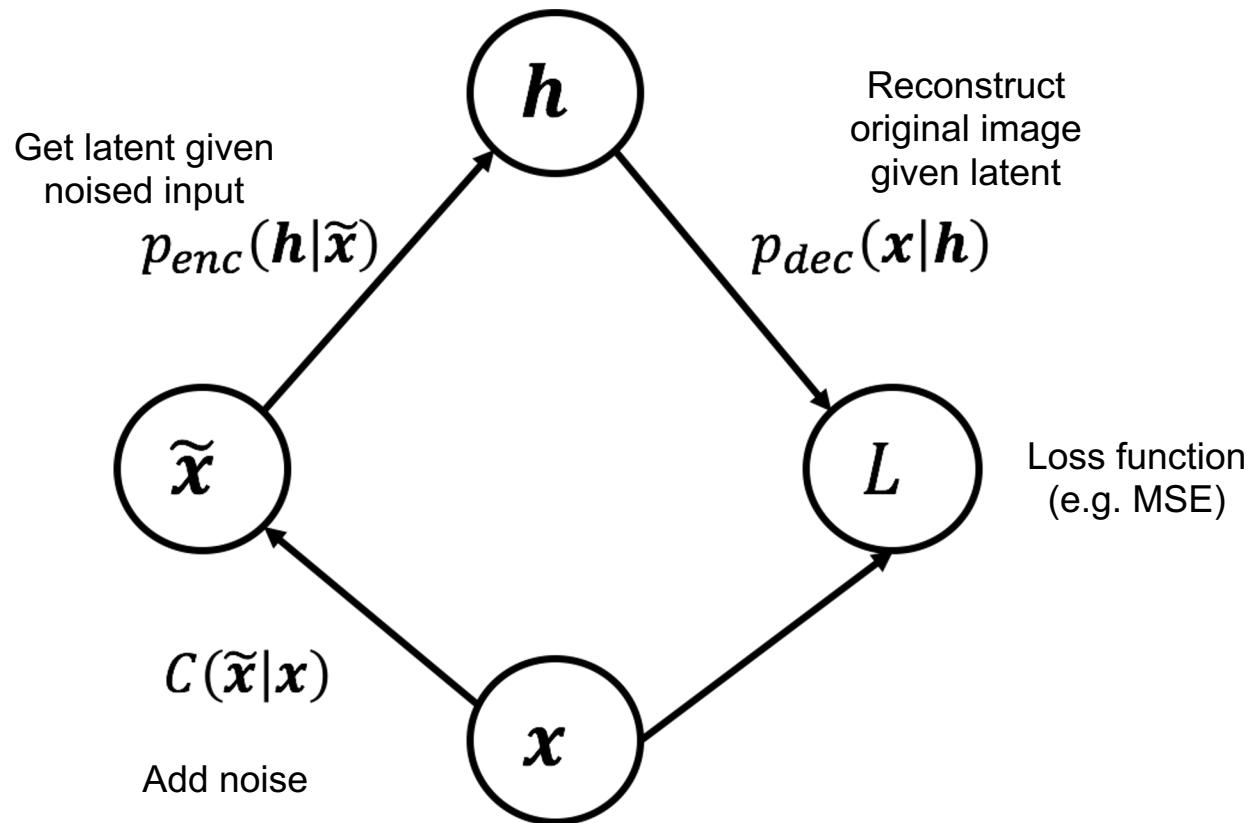
The idea is to recover the true distribution from a corrupted input:

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

\tilde{x} is the corrupted version of x

The network receives a corrupted input and generates a clean output (clean or uncorrupted version of input)

Denoising AutoEncoder Graphical Model



Denoising AutoEncoder

Sample a training example x from training data

Sample a corrupted version of x using corruption process

$$C(\tilde{x}|x)$$

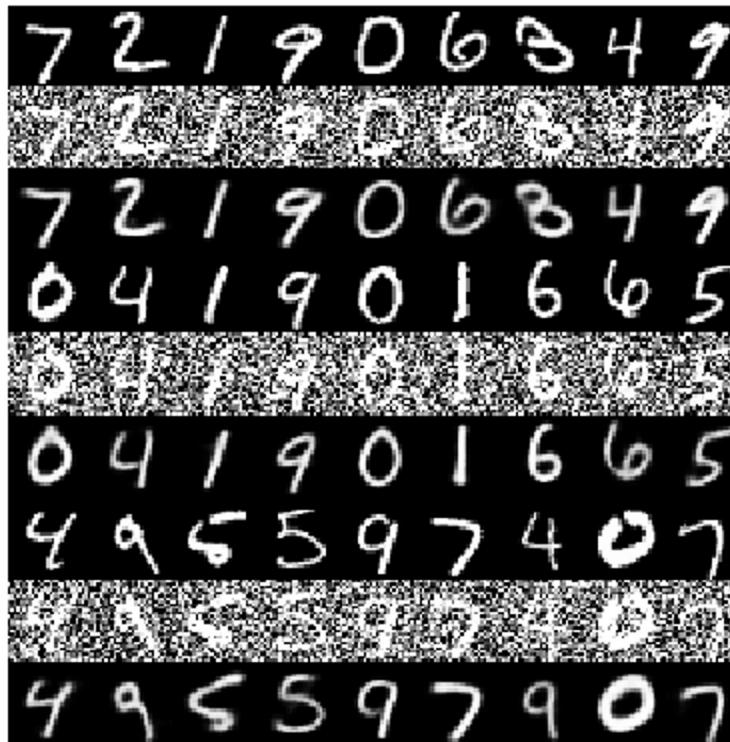
Use data points (\tilde{x}, x) to train the network to estimate autoencoder reconstruction distribution

$$p_{recon}(x|\tilde{x}) = p_{dec}(x|h)$$

where h is the output of the encoder $h = f(\tilde{x})$

Denoising AutoEncoder on MNIST

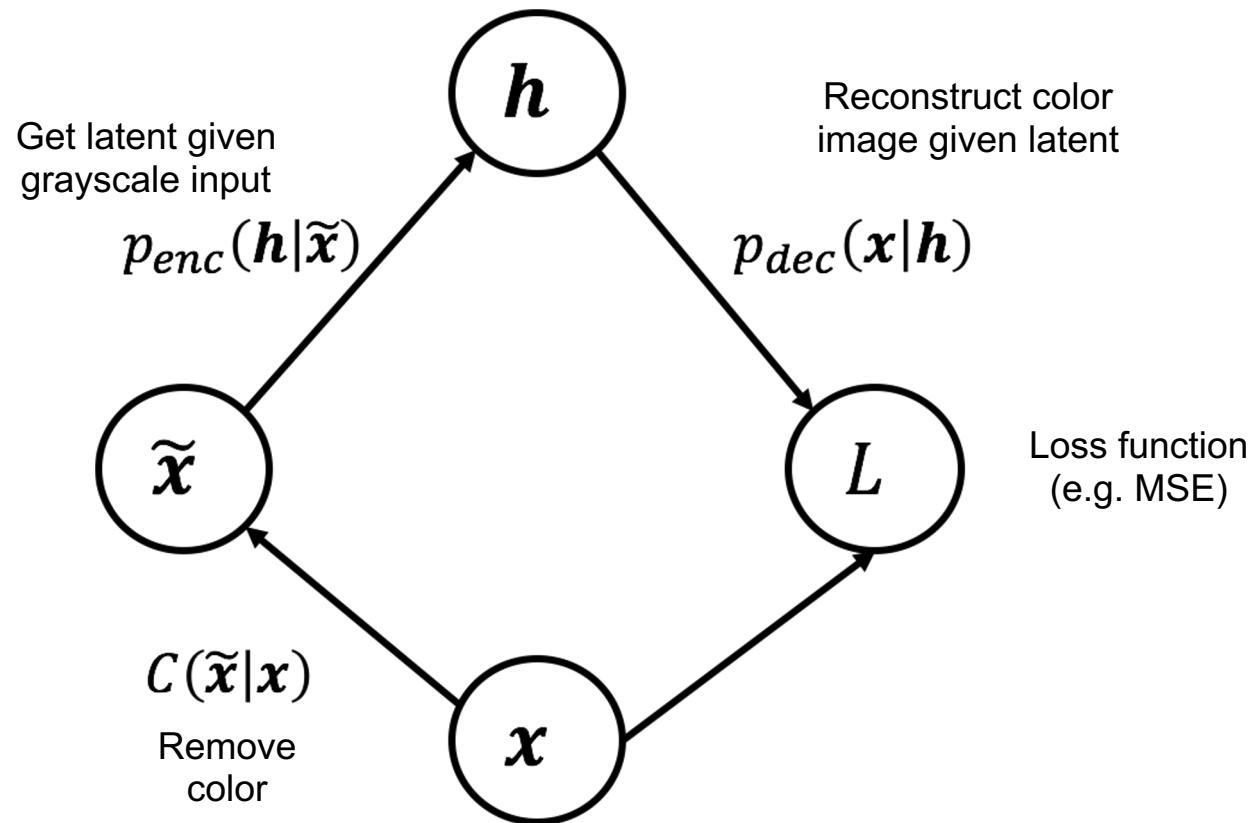
Original images: top rows, Corrupted Input: middle rows, Denoised Input: third rows



Colorization

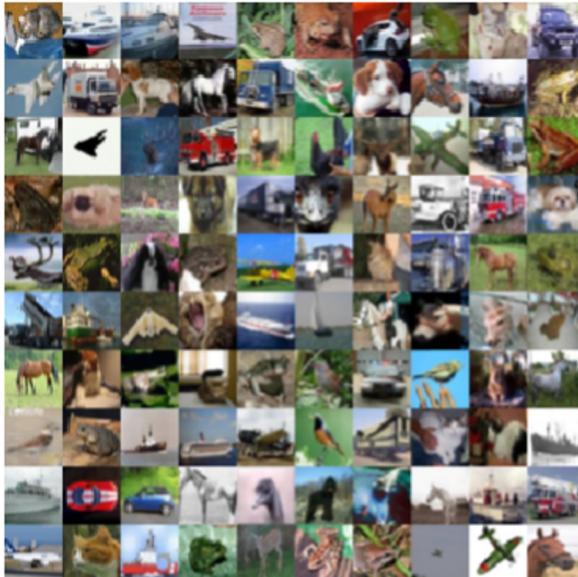


Colorization AutoEncoder



Colorization AutoEncoder on CIFAR10

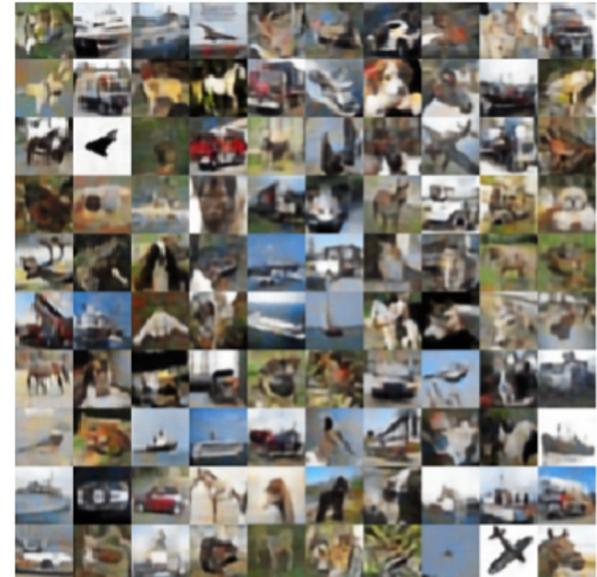
Test color images (Ground Truth)



Test gray images (Input)



Colorized test images (Predicted)



Code demo is next