# Deep Learning Architectures with self-supervision

University of Victoria - PHYS-555

# Main Idea

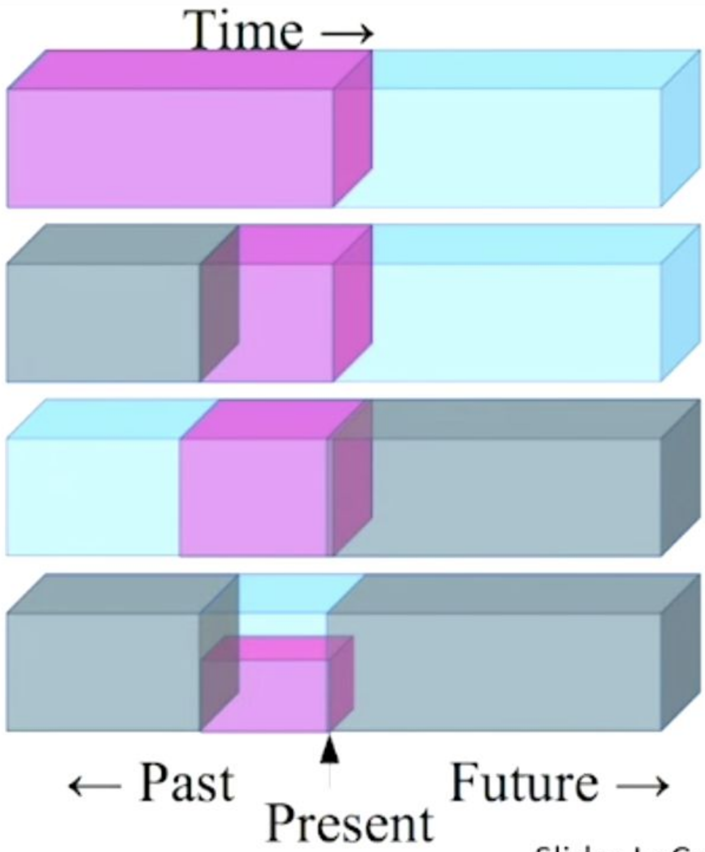
Apply supervision “internally” to the data, predict a subset of the information by exploiting the structure of the data.

Typical cases:

- **compress** and learn to reconstruct from compressed data
- **mask** data and learn to fill missing data
- learns special **pretext** tasks
- forces two samples known to be **similar** to have similar representations

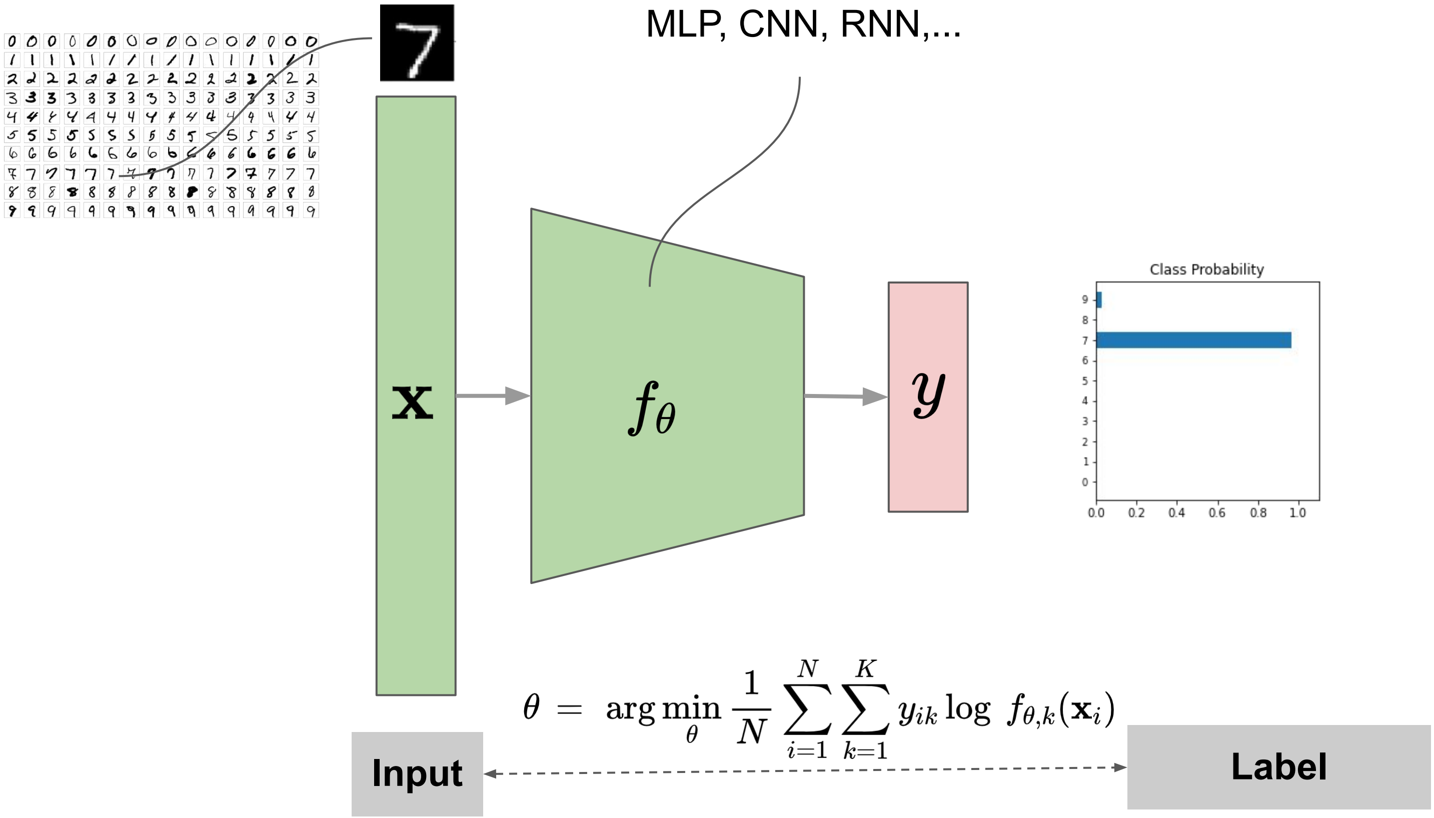
The learned representation maybe more data efficient by learning more semantics

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.

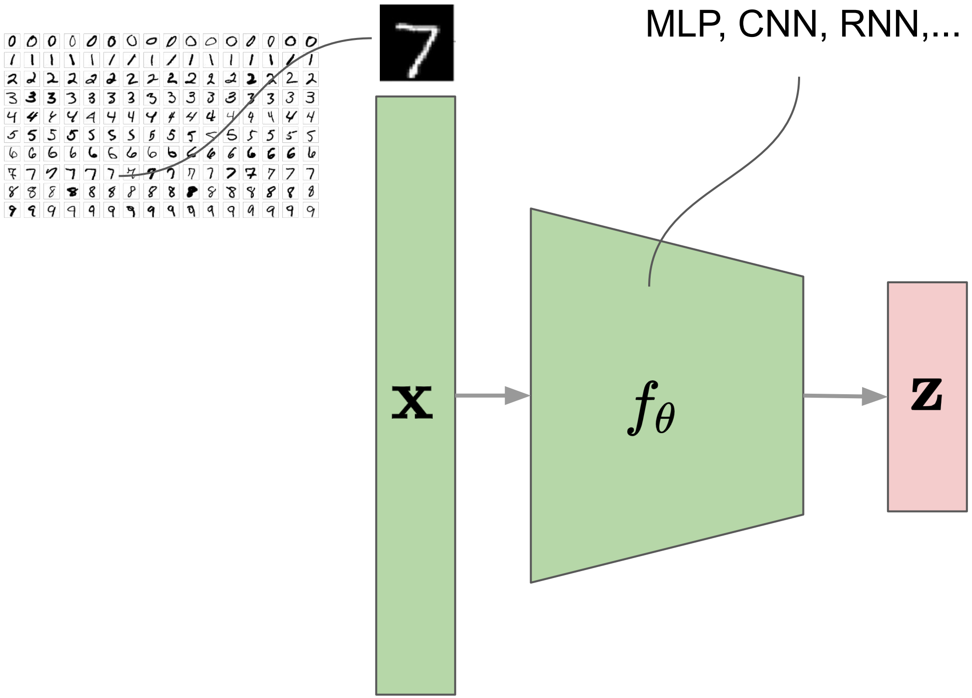



Slide: LeCun

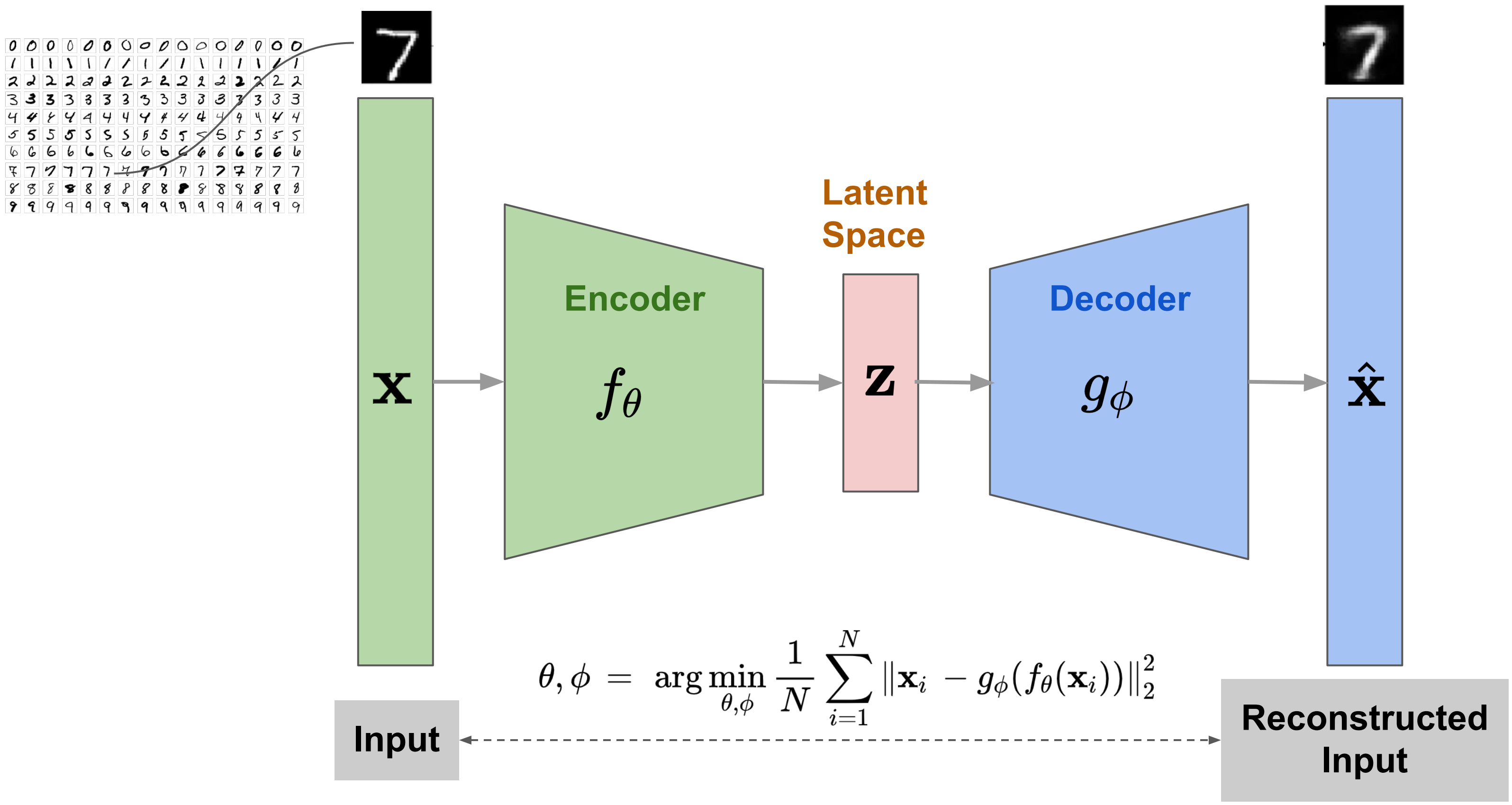
# Autoencoders



MLP, CNN, RNN,...
$\mathbf{x}$
$f_\theta$
$y$
Class Probability
9
8
7
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
$\theta = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \; f_{\theta,k}(\mathbf{x}_i)$
Input
Label




How can we get a compressed representation, less task specific?


Latent Space
Encoder
Decoder
$\mathbf{x}$
$f_\theta$
$\mathbf{z}$
$g_\phi$
$\hat{\mathbf{x}}$
$\theta, \phi = \arg\min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - g_\phi(f_\theta(\mathbf{x}_i))\|_2^2$
Input
Reconstructed Input

# Autoencoders

- Reconstruct the input data with a neural network
- Encoder can be anything
- Decoder does not have to be the same type as Encoder
- Shape and size of latent space is a hyper-parameter
- Convolutional Autoencoder:
  - Encoder = CNN
  - Decoder = CNN with transposed convolutions (or upsampling)

Homework: How is a UNet different?

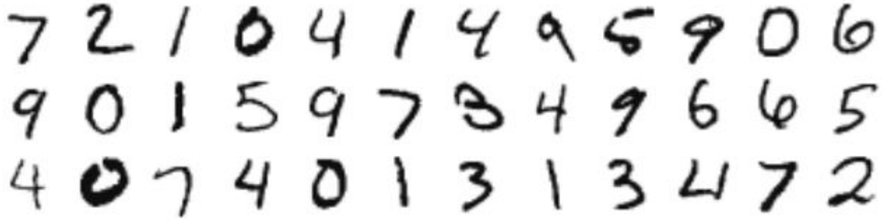
# Relation between PCA and Autoencoders

The encoder of a linear autoencoder is equivalent to PCA if we

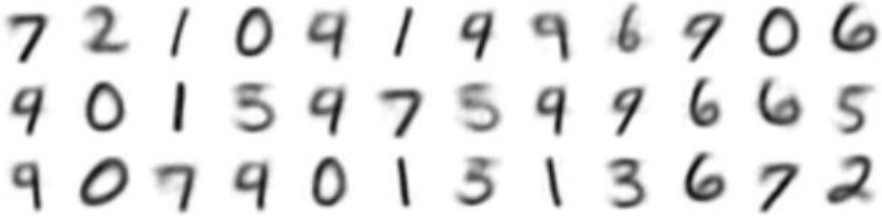
- use a linear encoder
- use a linear decoder
- use a MSE loss
- and normalize the inputs to xˆij = 1 √ m xij − 1 m Xm k=1 xk

Homework: prove it!

$X$ (original samples)

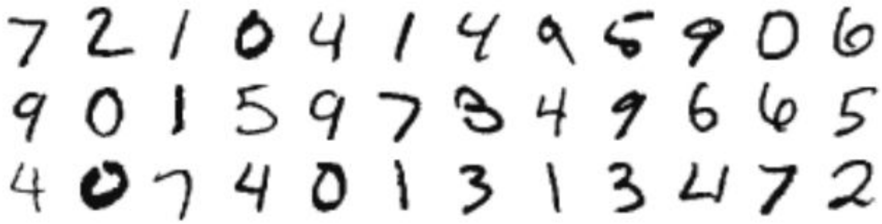


$g \circ f(X)$ (CNN, $d = 2$)

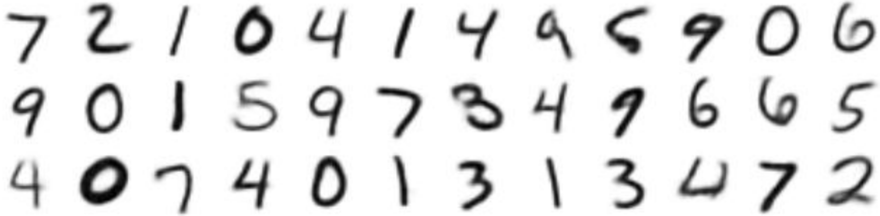


$g \circ f(X)$ (PCA, $d = 2$)

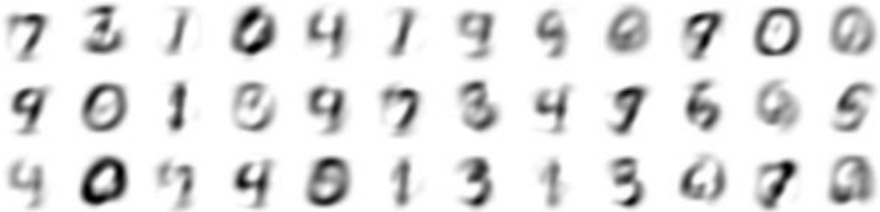
$X$ (original samples)



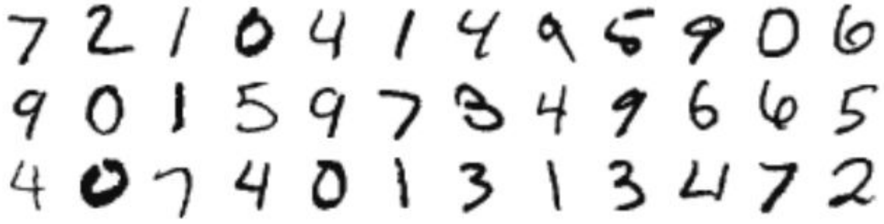
$g \circ f(X)$ (CNN, $d = 8$)



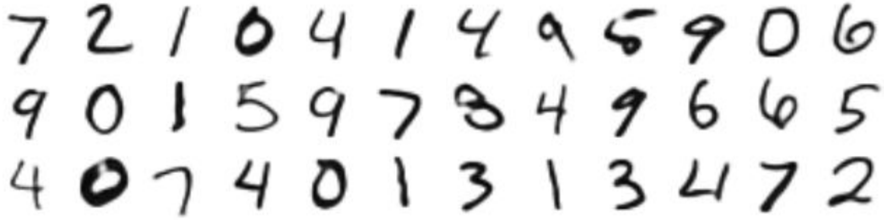
$g \circ f(X)$ (PCA, $d = 8$)



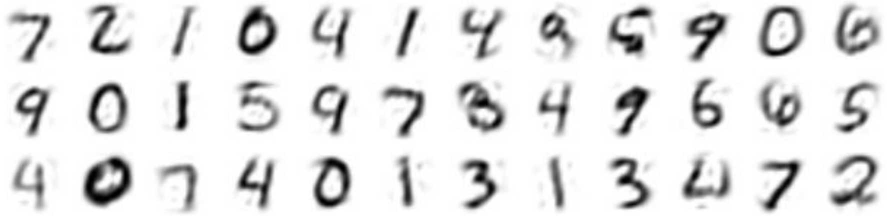
$X$ (original samples)



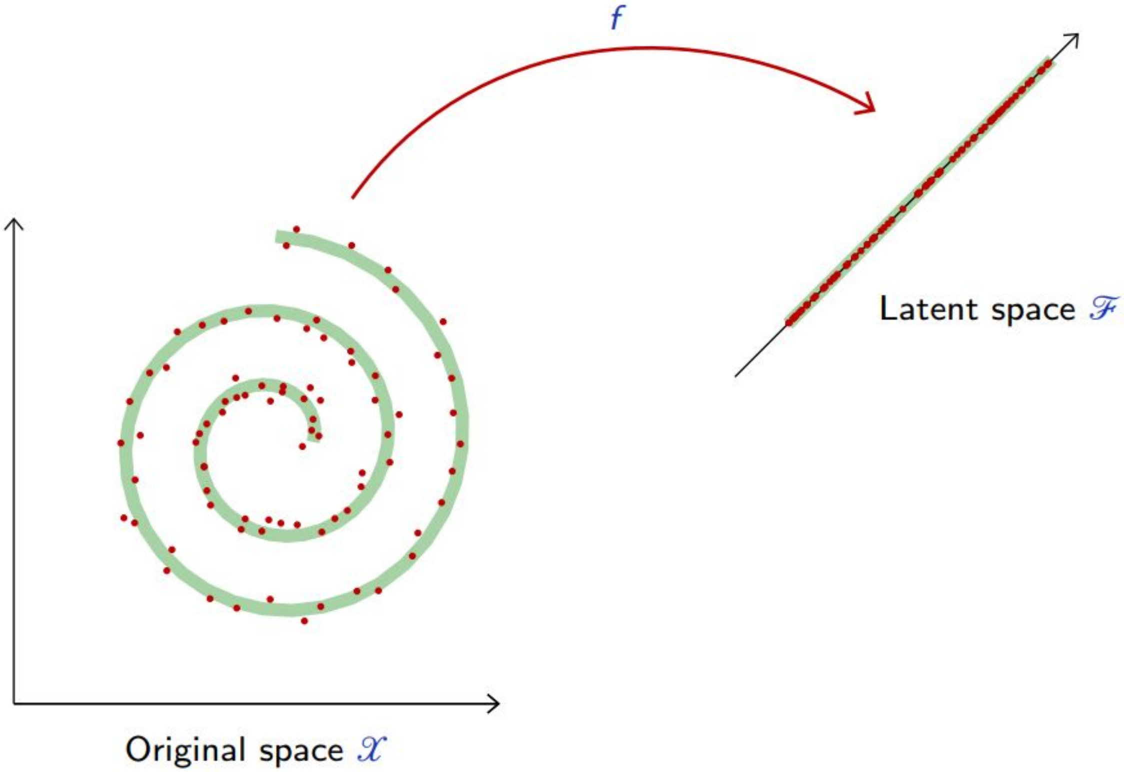
$g \circ f(X)$ (CNN, $d = 32$)



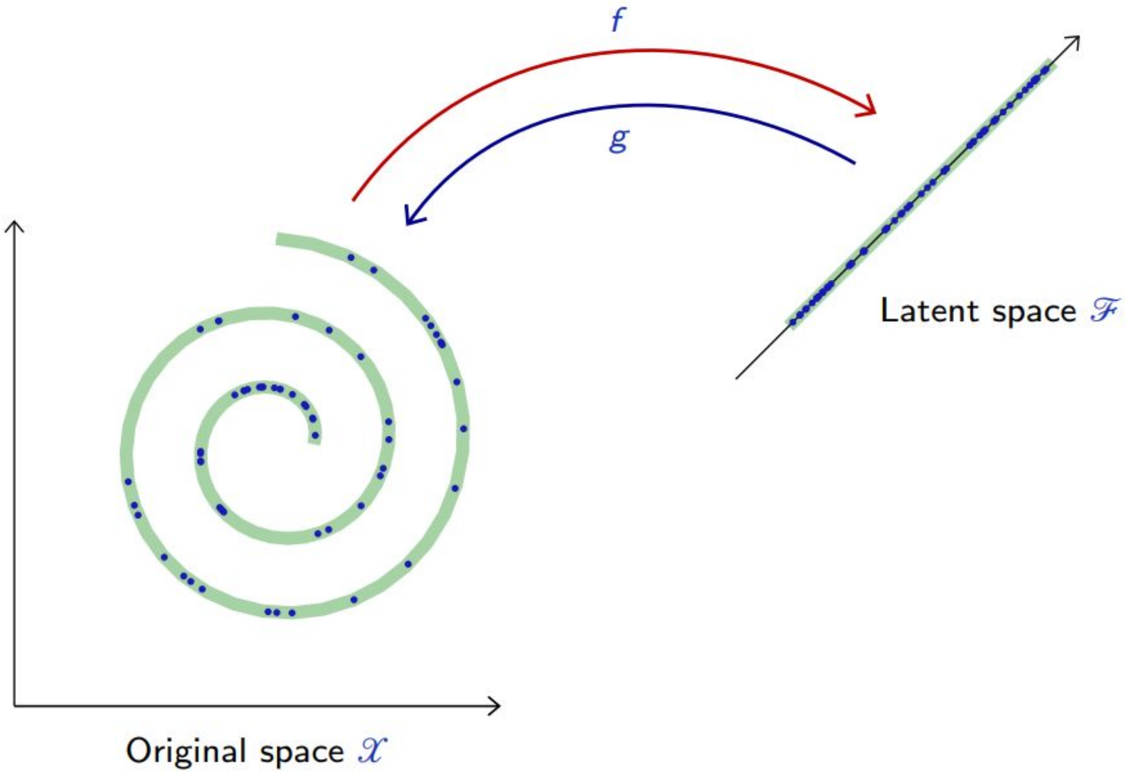
$g \circ f(X)$ (PCA, $d = 32$)



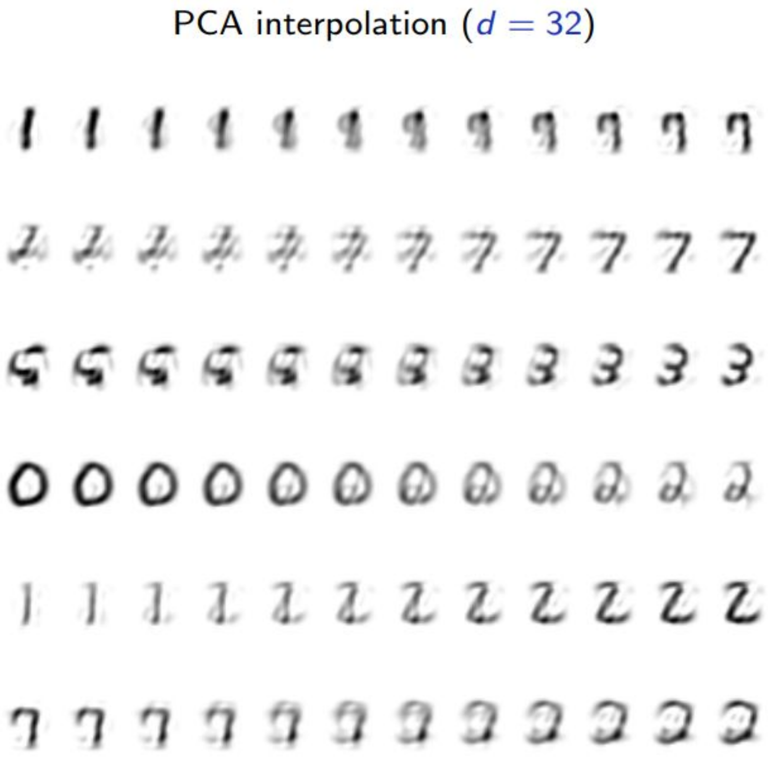
# Latent Space?



# Latent Space?



# Use of Autoencoder: Latent Code Interpolation

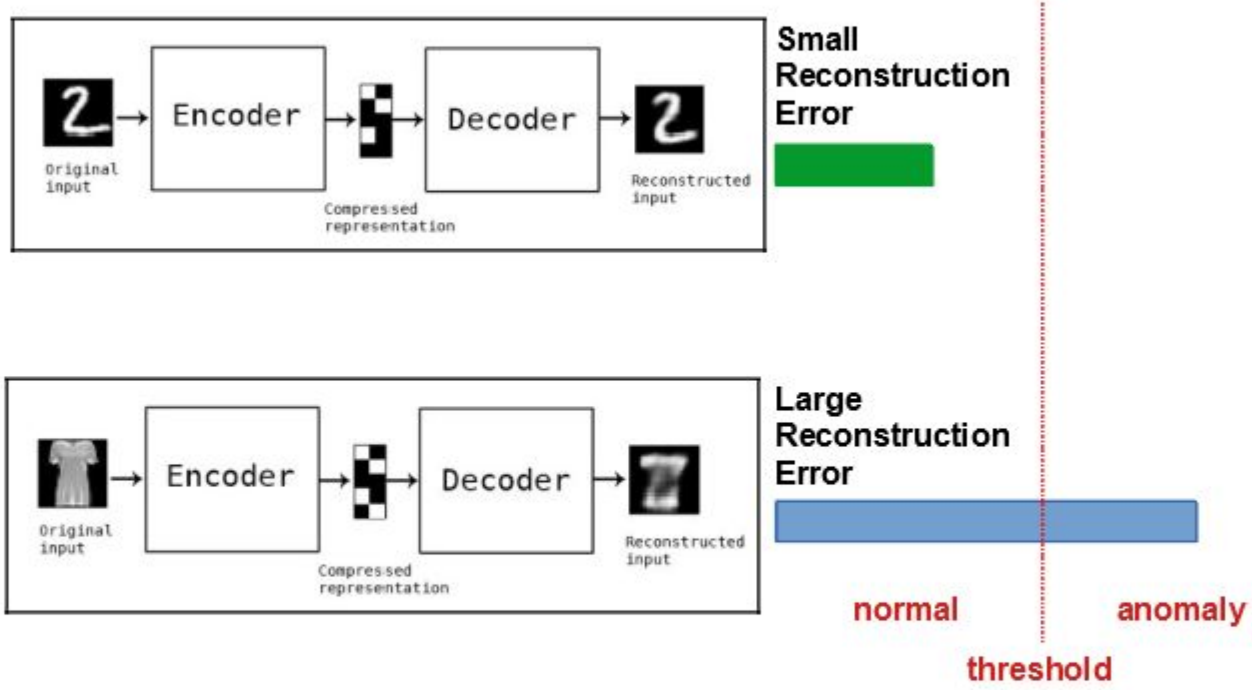


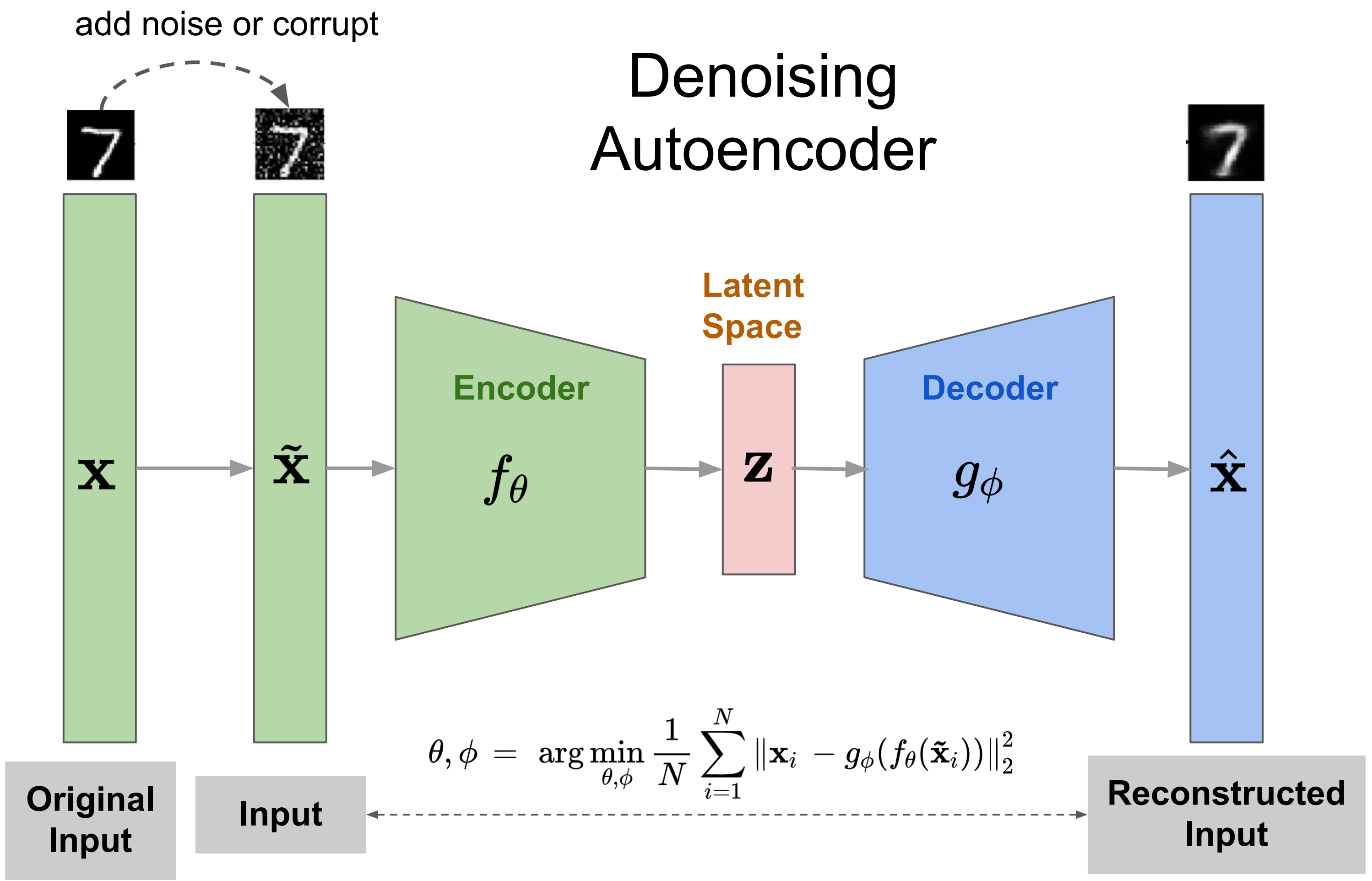
# Use of Autoencoder: Latent Code Interpolation

Autoencoder interpolation ($d = 32$)

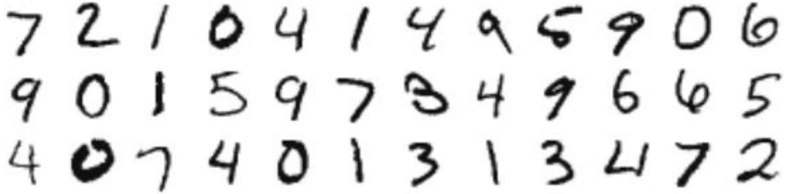
# Use of Autoencoder: Anomaly Detection

Trained
on MNIST



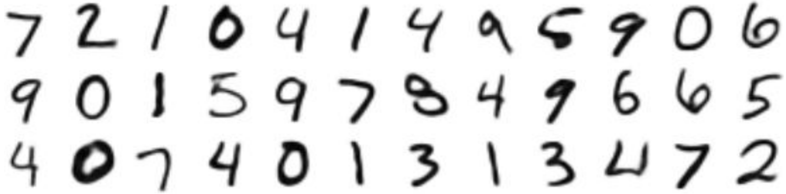

add noise or corrupt
Denoising
Autoencoder
Latent
Space
Encoder
$f_\theta$
Decoder
$g_\phi$
$\mathbf{x}$
$\tilde{\mathbf{x}}$
$\mathbf{z}$
$\hat{\mathbf{x}}$
$\theta, \phi = \arg\min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - g_\phi(f_\theta(\tilde{\mathbf{x}}_i))\|_2^2$
Original
Input
Input
Reconstructed
Input

Original

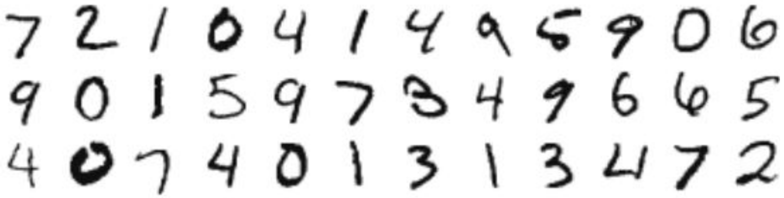


Corrupted ($p = 0.5$)

Reconstructed

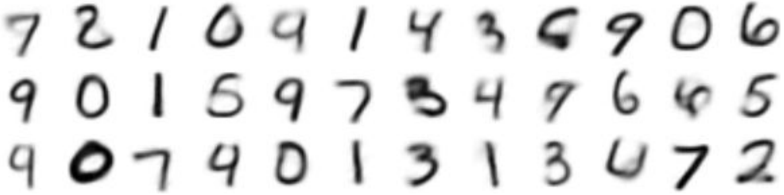


Original

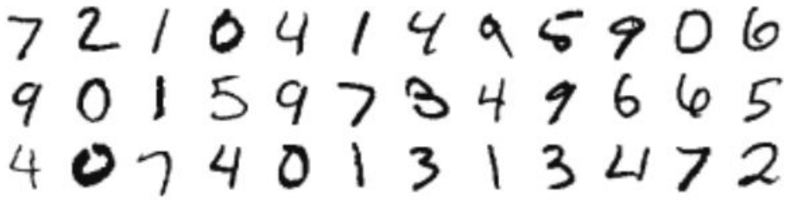


Corrupted ($p = 0.9$)

Reconstructed

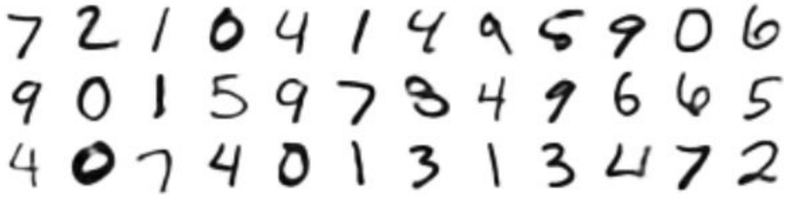


Original



Corrupted ($\sigma = 4$)

Reconstructed



# Other Regularized Autoencoders

Two main other main autoencoders mitigate overfitting with a regularizations functions: $\ell(\mathbf{x}_i, \hat{\mathbf{x}}_\mathbf{i}) + \Omega(h)$

- **Sparse Autoencoders** may help for classification. They have larger hidden layers, achieved by adding a sparsity constraint, such as:

$$\Omega(h) \;=\; \lambda \sum_i |h_i|$$

- **Contractive Autoencoders** encourages the latent code to be more robust data, achieved by adding a constraint on the Jacobian of the hidden layers:

$$\Omega(h) \;=\; \lambda \sum_{j=1}^{n} \sum_{l=1}^{k} \left( \frac{\partial h_l}{\partial x_j} \right)^2$$

# Self-supervision Pretext Task: Predicting Spatial Context

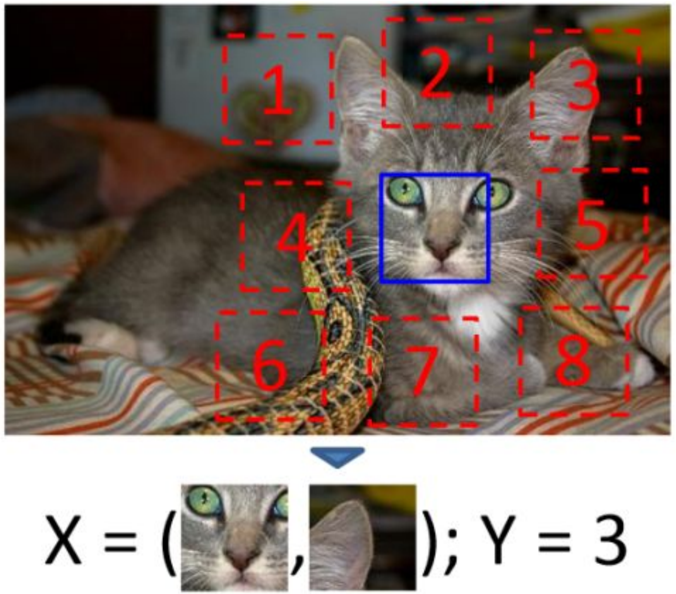
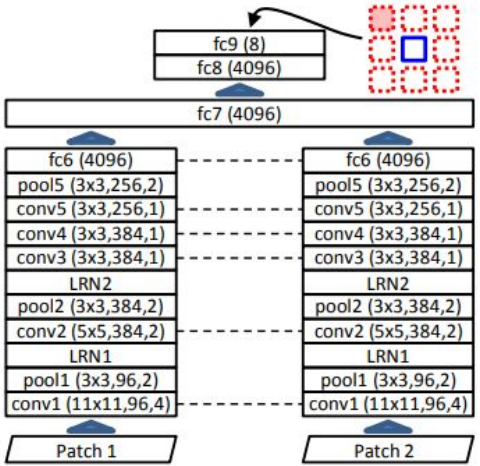



Figure 2. The algorithm receives two patches in one of these eight possible spatial arrangements, without any context, and must then classify which configuration was sampled.




[Doersh et al (2015)]

# Pretext Task: Learn to Solve Jigsaw Puzzles

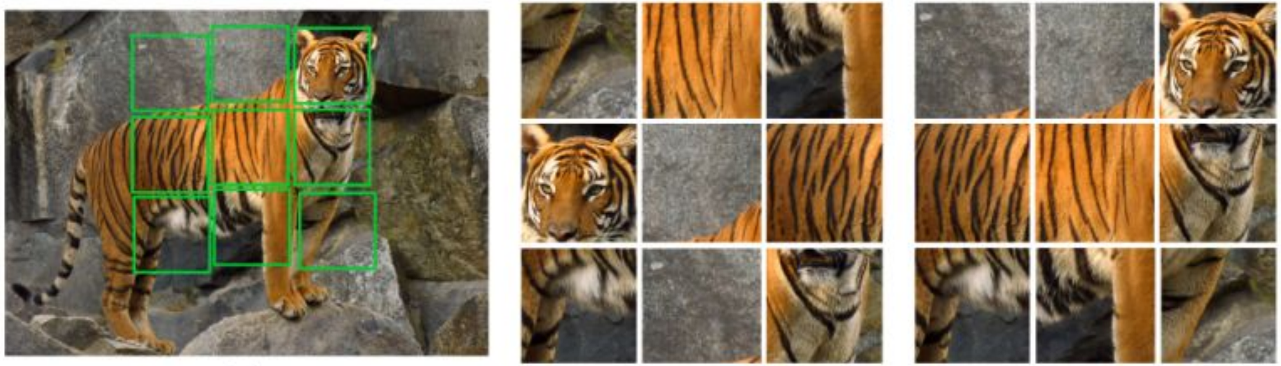
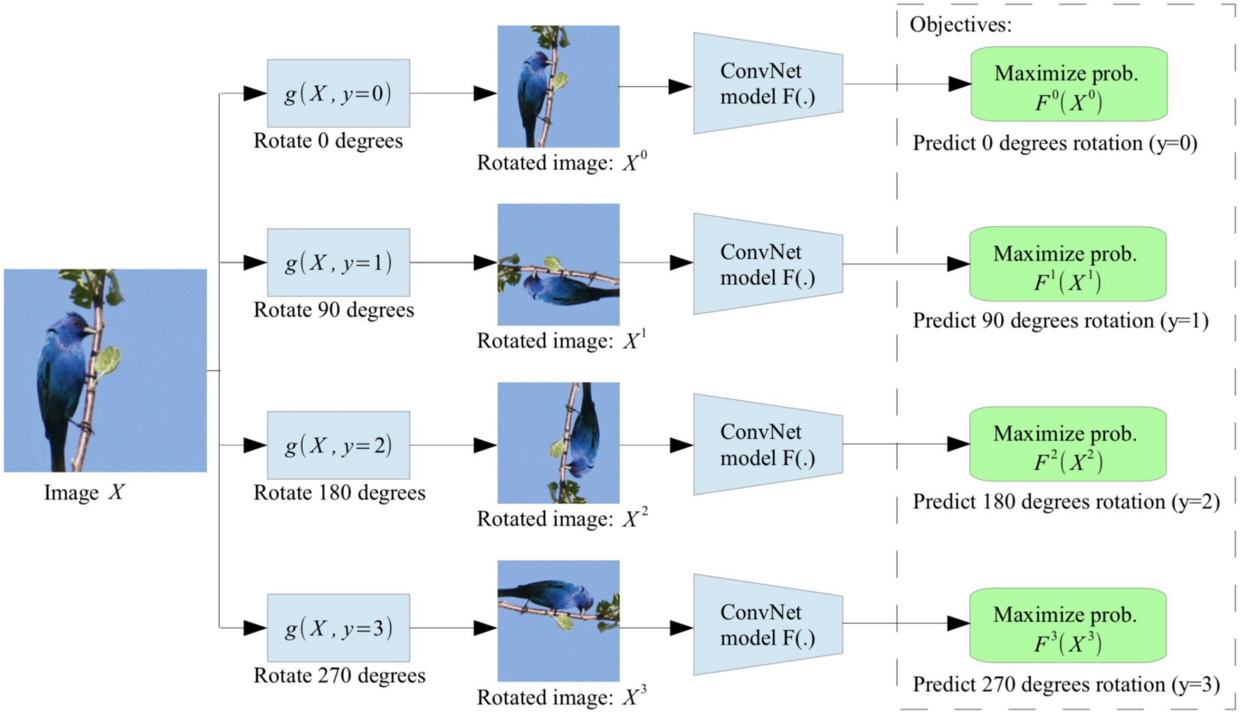


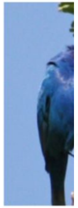
Fig. 1: Learning image representations by solving Jigsaw puzzles. (a) The image from which the tiles (marked with green lines) are extracted. (b) A puzzle obtained by shuffling the tiles. Some tiles might be directly identifiable as object parts, but others are ambiguous (*e.g.*, have similar patterns) and their identification is much more reliable when all tiles are jointly evaluated. In contrast, with reference to (c), determining the relative position between the central tile and the top two tiles from the left can be very challenging [10].

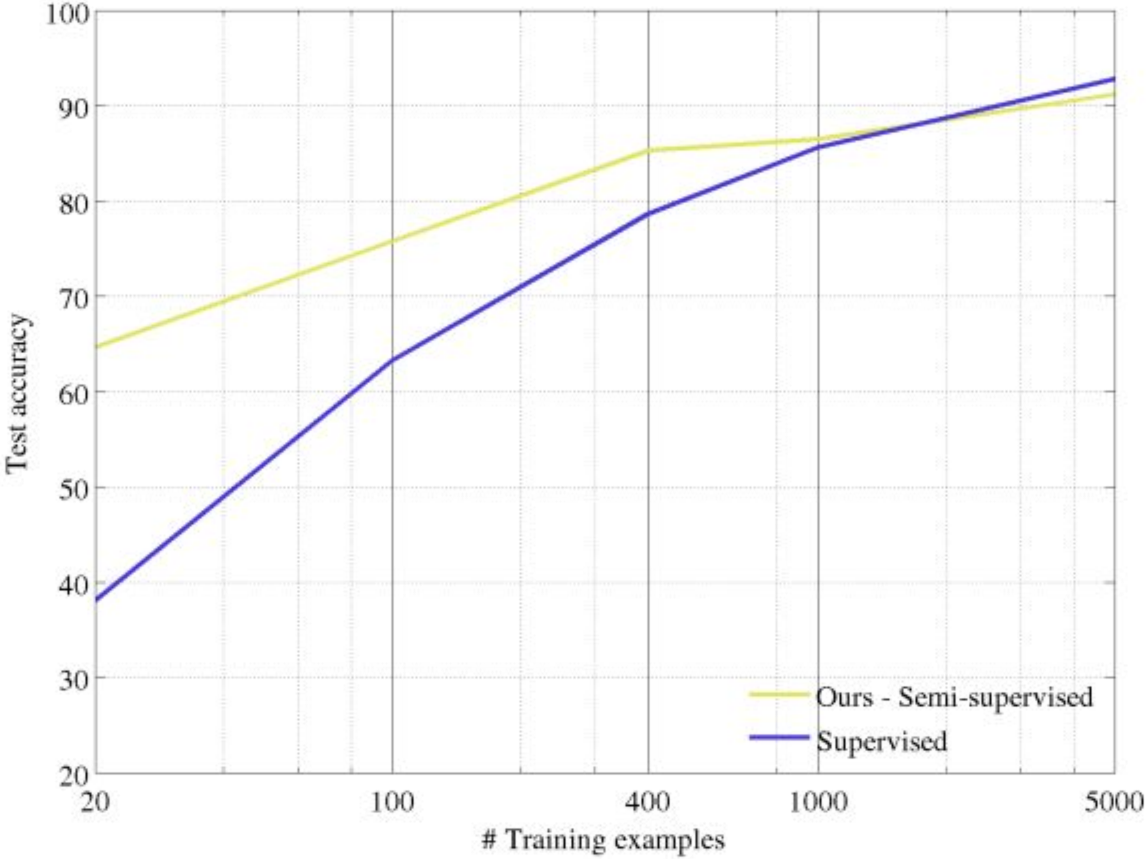
Noroozi et al (2016)

# Pretext Task: Predicting Image Rotation

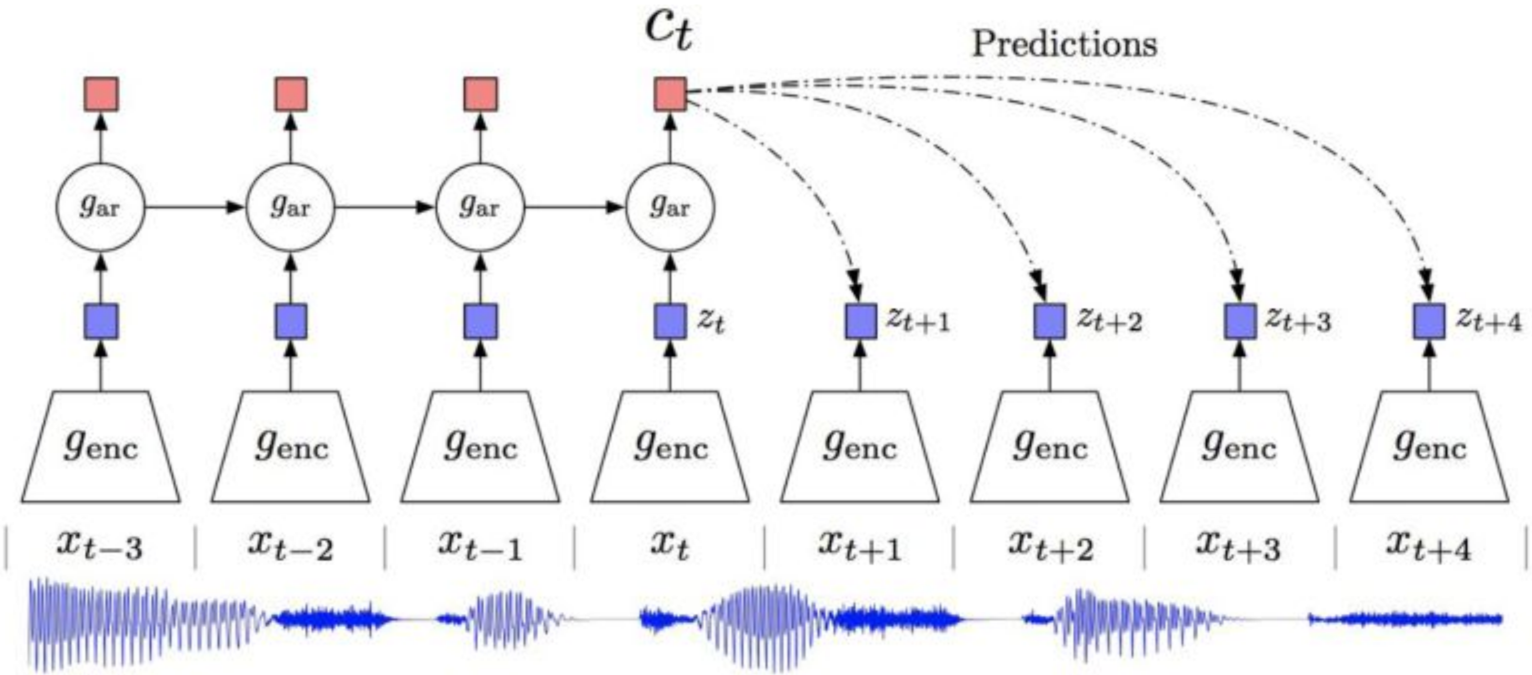


# Pretext Tas'




Test accuracy
100
90
80
70
60
50
40
30
20
20
100
400
1000
5000
# Training examples
Ours - Semi-supervised
Supervised

# Contrastive Predictive Coding




v.d. Oord et al (2018)

# What is Contrastive Learning?

Contrastive learning is a learning paradigm where we want to learn *distinctiveness.*

- *What makes two objects similar or different?*
- *When I train a network for some task, say classification, I am already forcing my network to learn discriminative features, right?*

Sometimes high-level features alone aren't enough to learn good representations, especially when *semantics* come into play.

Features like shape and color of the tail of a whale aren't enough to uniquely identify its species because the semantics for the tails of all whales are very similar.

From: Blog on [Medium]

# Learning similarity between samples with a distance

Goal: build a function $d_\theta(\mathbf{x}_1, \mathbf{x}_2)$ to quantify how “similar” two sample of data are

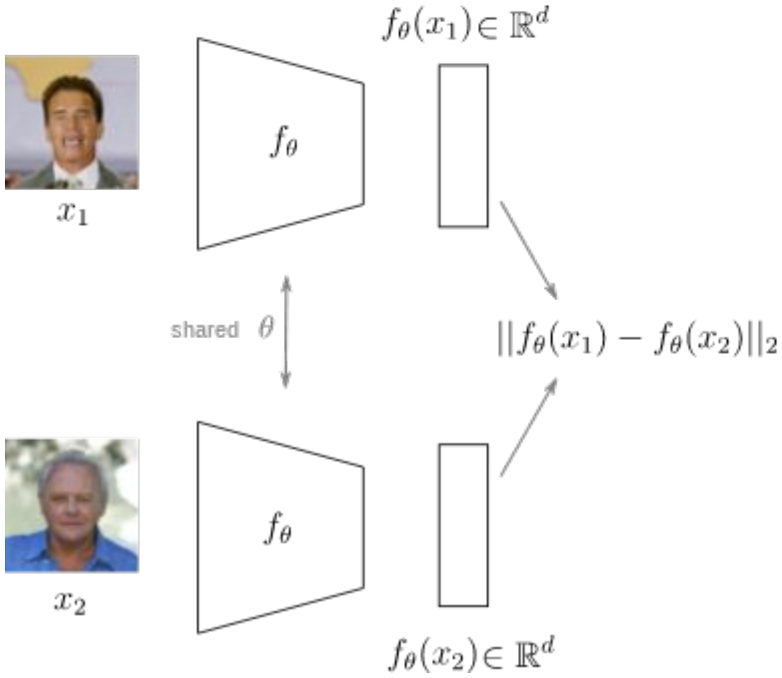
Example: a Euclidean distance between two representations of a NN $f_\theta$

$$d_\theta(\mathbf{x}_1, \mathbf{x}_2) = \|f_\theta(\mathbf{x}_1) - f_\theta(\mathbf{x}_2)\|_2$$

# Siamese Networks

- Can be used for classification: define a threshold $T$ to decide when two samples belong to the same class:

$$d_\theta(\mathbf{x}_1, \mathbf{x}_2) < T$$



# Example: SimCLR

**SimCLR** (Chen et al, 2020) proposed a simple framework for contrastive learning of visual representations. It learns representations for visual inputs by maximizing agreement between differently augmented views of the same sample via a contrastive loss in the latent space.

SimCLR works in the following three steps:

1. Randomly sample a mini-batch of N samples and each sample is applied with two different data augmentation operations, resulting in 2N augmented samples in total.

2. Given one positive pair, other 2(N-1) data points are treated as negative samples. The representation is produced by a base encoder NN

3. The contrastive loss is defined using cosine similarity. The loss operates on top of an extra projection of the representation rather than on the representation from the latent space directly. But only the representation h is used for downstream tasks.

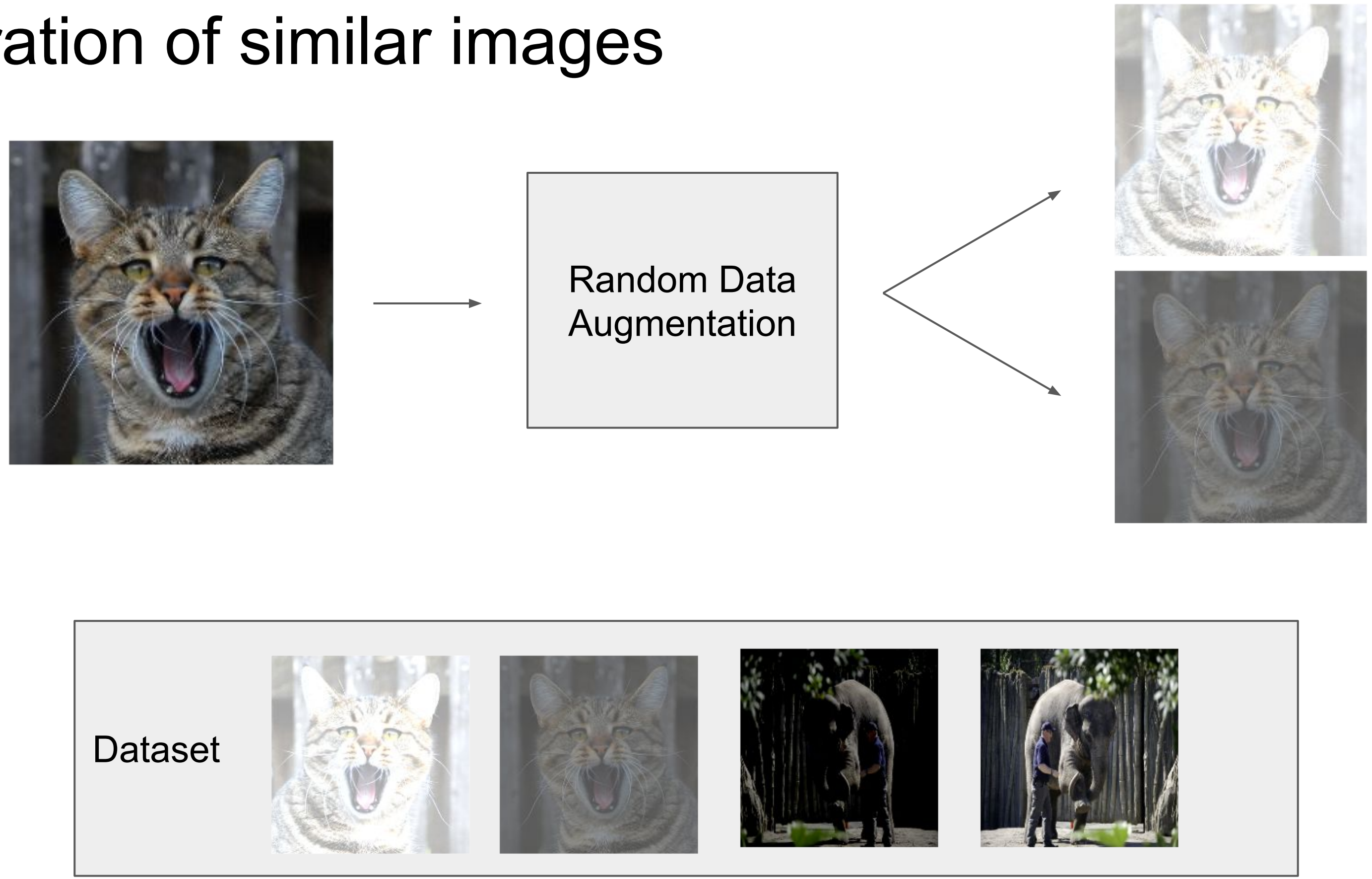
# How to generate pairs of similar data points?

A supervised approach would be to manually label them as similar or not

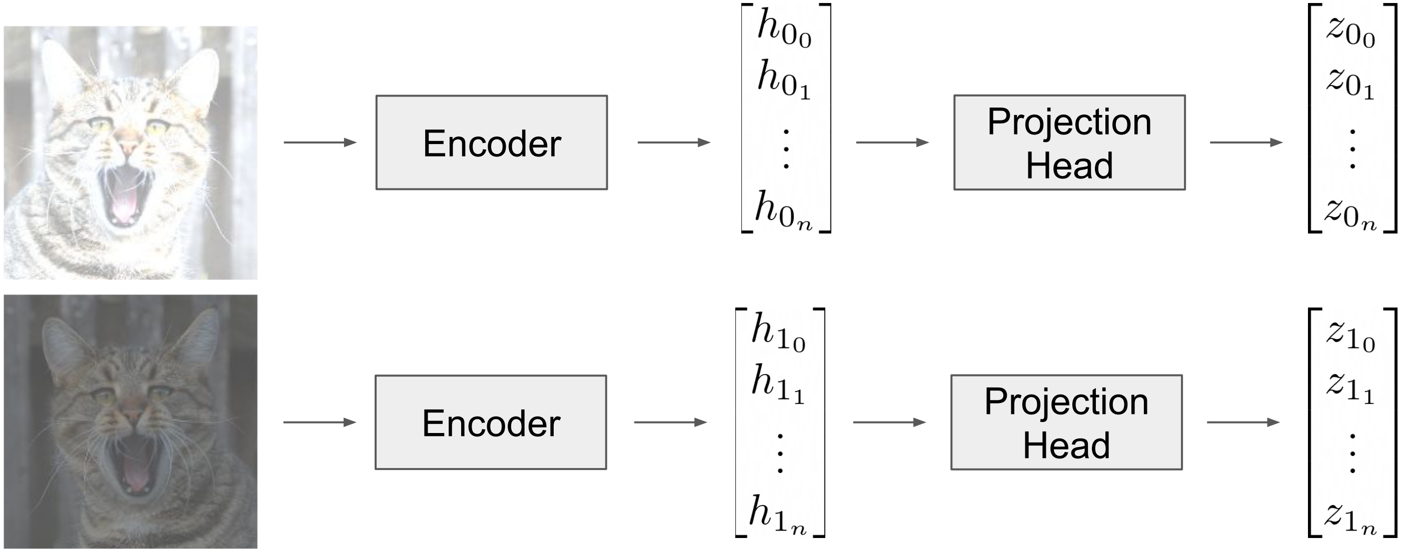
Similar

Dissimilar

# Generation of similar images



# Framework




Compare Similarity of the two embeddings!

**Resnet50 used as Encoder**

**Two layer MLP used to get embedding**

# Loss Calculation

For each data pair (embeddings z):

Compute Pairwise Similarity

$$s_{i,j} = \frac{z_i^T z_j}{\tau \parallel z_i \parallel \parallel z_j \parallel}$$

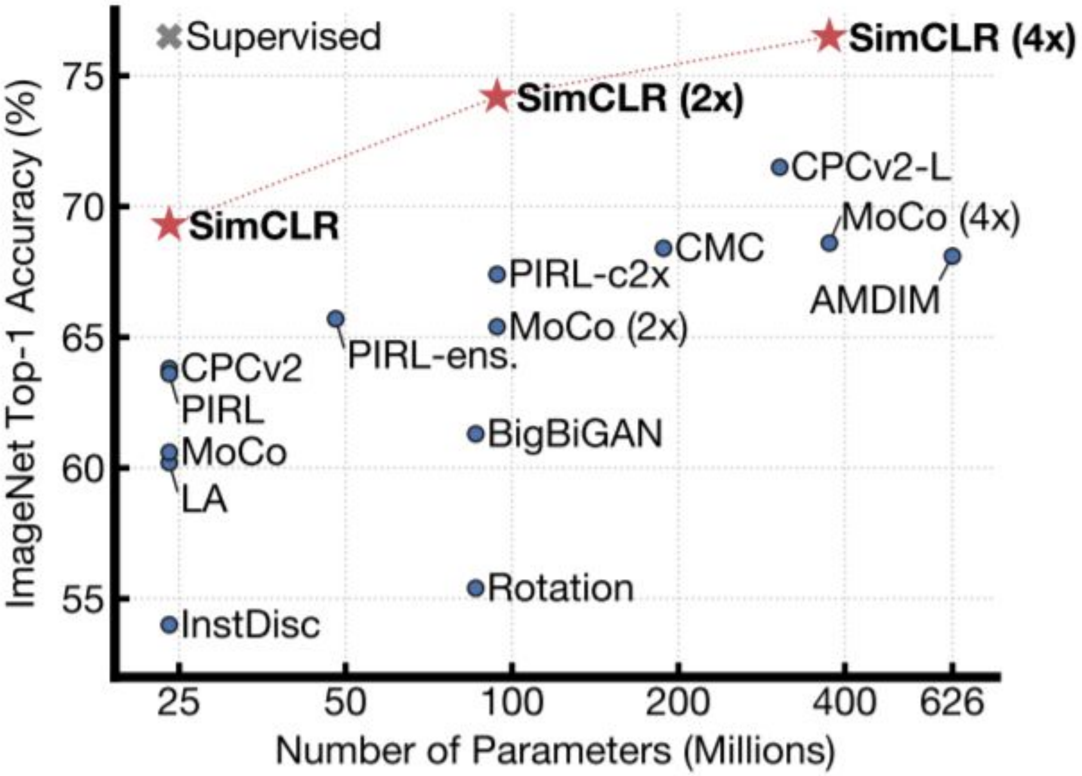
Compute loss

$$l_{i,j} = -log \frac{exp(s_{i,j})}{\sum_{k=1,k!=i}^{2N} exp(s_{i,k})}$$

Batch wise loss for positive pairs

$$L = \frac{1}{2N} \sum_{k=1}^{2N} [l(2k-1, 2k) + l(2k, 2k-1)]$$

# SimCLR Classification Results on ImageNet



# Example on Galaxies 3-band images

UMAP of the SimCLR representation

[Hayat et al. 2021](#)