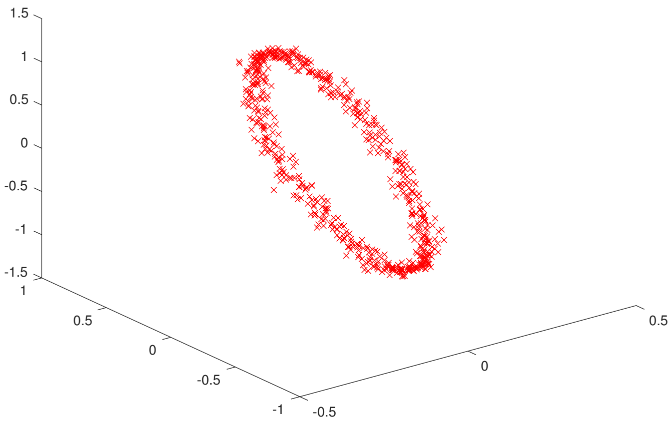
Non Linear Dimensionality Reduction

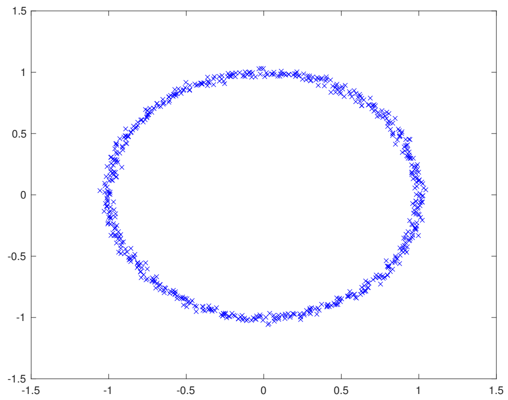
Pascal Fua
IC-CVLab

EPFL

Reminder: Dimensionality Reduction



$\mathbf{x}$



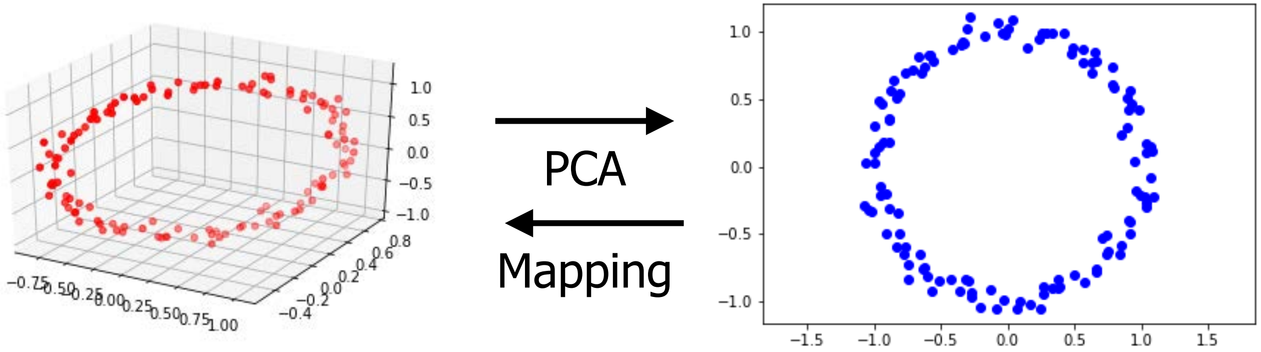
$\mathbf{y}$

Our goal is to find a mapping $\mathbf{y}_i = f(\mathbf{x}_i)$

- $\mathbf{x}_i \in \mathbb{R}^D$: High-dimensional data sample
- $\mathbf{y}_i \in \mathbb{R}^d$: Low-dimensional representation

How about a linear one $\mathbf{y}_i = \mathbf{W}^T\mathbf{x}_i$?

Reminder: Optimal Linear Mapping



$\mathbf{x}$

$\mathbf{y} = \mathbf{W}^{\mathbf{T}}(\mathbf{x} - \bar{\mathbf{x}})$

$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{W}\mathbf{y}$

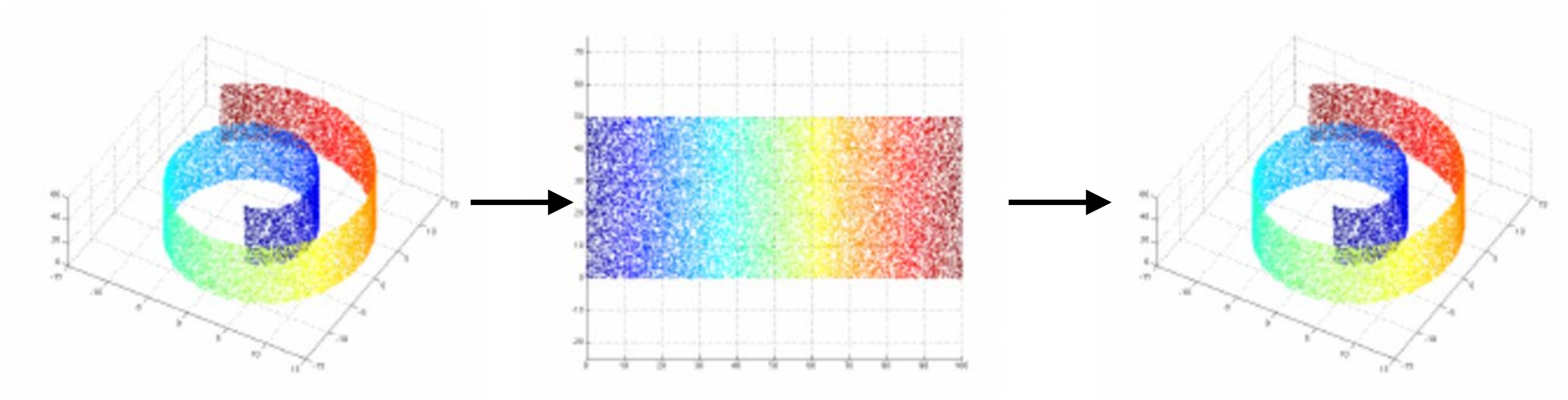
- This mapping incurs some loss of information.
- However, the corresponding rectangular matrix $\mathbf{W}$ is the orthogonal matrix that minimizes the reconstruction error

$$e = \|\hat{\mathbf{x}} - \mathbf{x}\|^2$$

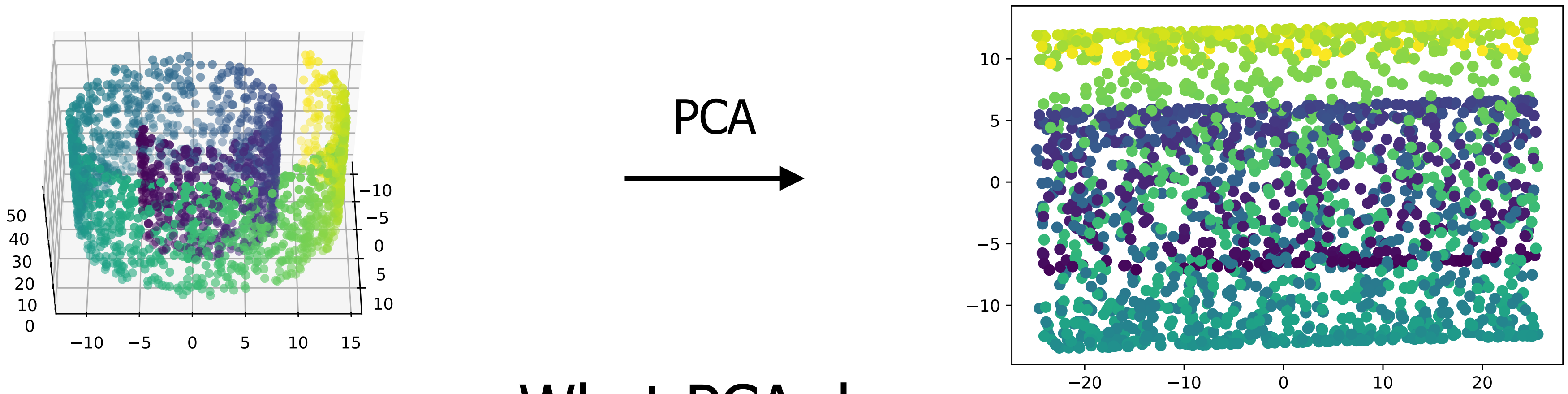
where

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{W}\mathbf{y} = \bar{\mathbf{x}} + \mathbf{W}\mathbf{W}^T(\mathbf{x} - \bar{\mathbf{x}})$$

Limitation of the Linear Model



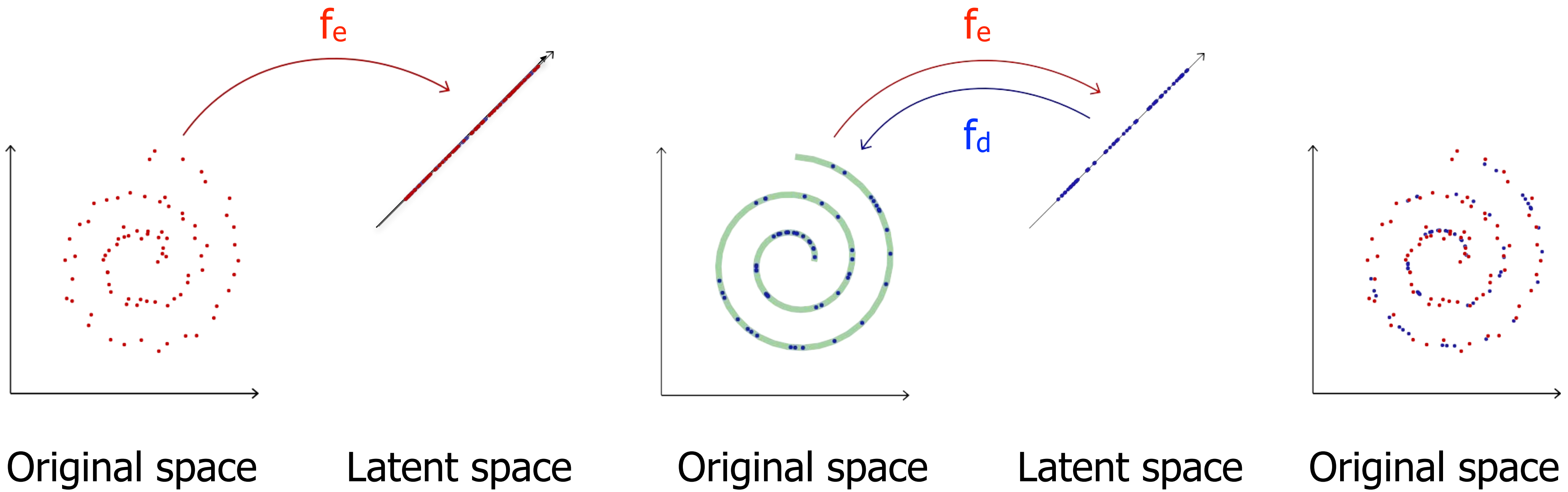
What it should do.



What PCA does.

—> Can we make the mappings non-linear so that cases like this one can be properly handled?

Latent Space



$$\mathbf{z} = f_e(\mathbf{x})$$

$$\hat{\mathbf{x}} = f_d(\mathbf{z})$$

$$\hat{\mathbf{x}} \approx \mathbf{x}$$

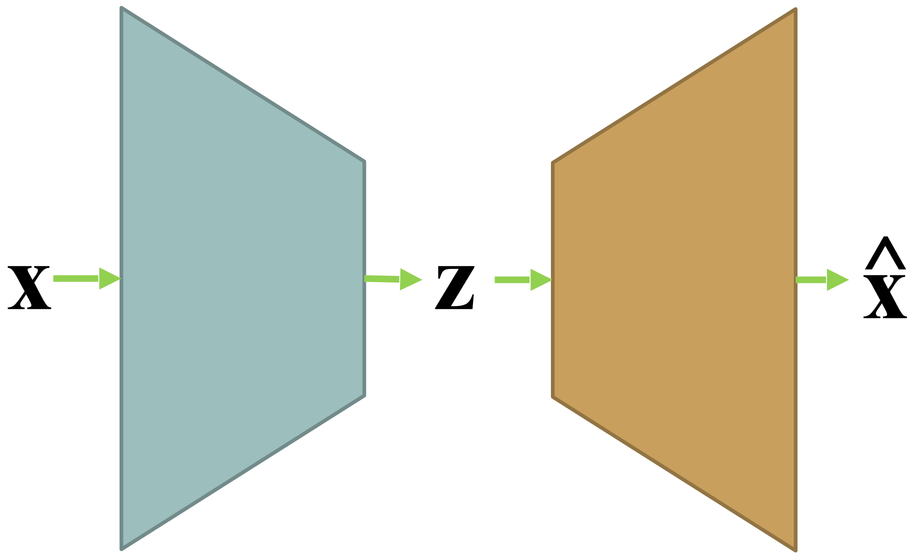
- Removes unnecessary degrees of freedom.
- Models correlations between the true ones.
- Makes it possible to denoise the original data.

Linear vs NonLinear

- In PCA and LDA, the functions f_e and f_d are linear.
- We have seen that for classification non-linear functions are often more effective.
- Can we learn non-linear f_e and f_d?

- There are many ways to do this but we will focus on special purpose deep networks called autoencoders.

Autoencoders

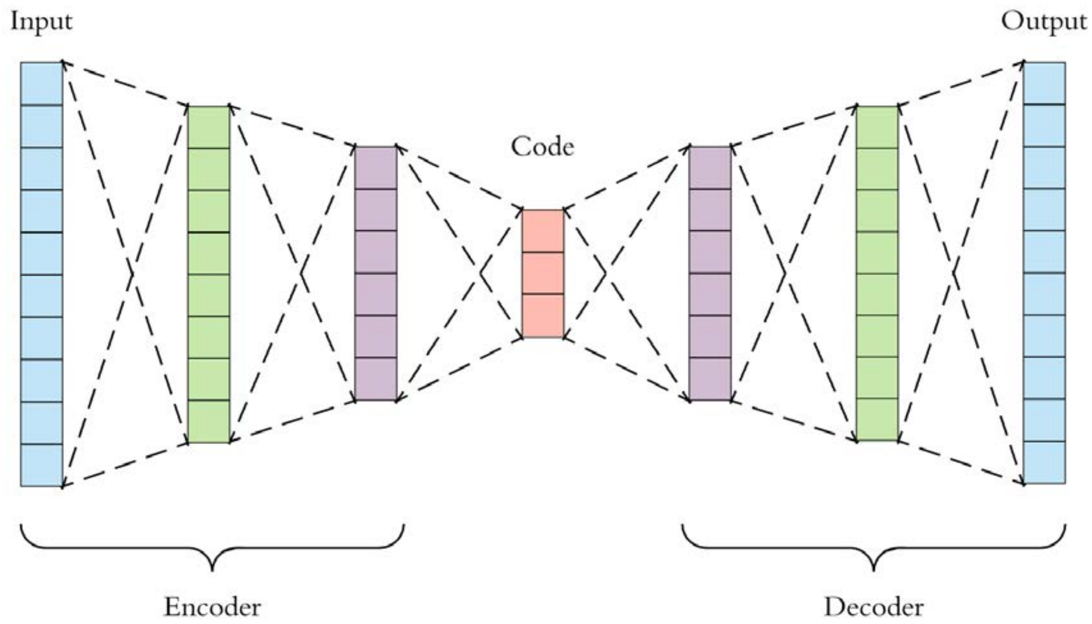


$$
\begin{aligned}
\mathbf{z} &= \sigma_e(\mathbf{W}_e\mathbf{x} + \mathbf{b}_e) \\
\hat{\mathbf{x}} &= \sigma_d(\mathbf{W}_d\mathbf{z} + \mathbf{b}_d)
\end{aligned}
$$

- $\mathbf{z}$ is the *latent vector* representation of $\mathbf{x}$.
- $\hat{\mathbf{x}}$ is the *reconstruction* of $\mathbf{x}$ and should be as similar to it as possible.
- $\mathbf{w}$ can be computed by minimizing $\sum_n \|\hat{\mathbf{x}_n} - \mathbf{x}_n\|^2$.

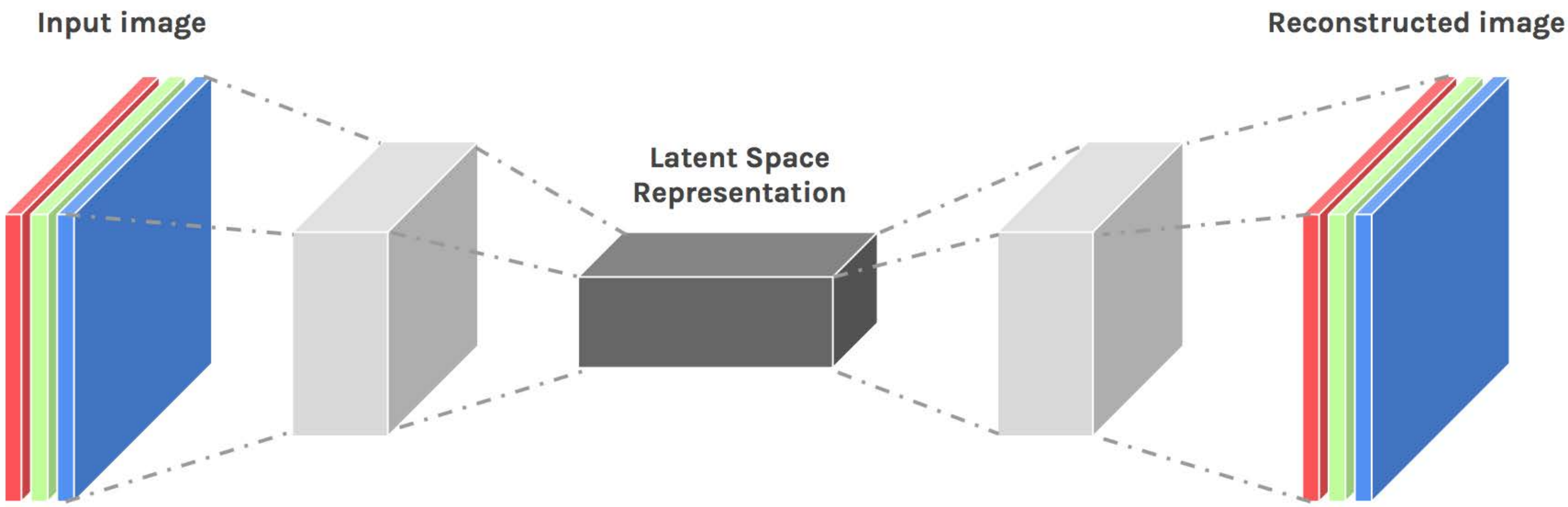
—> Unsupervised training.

Deep Autoencoder



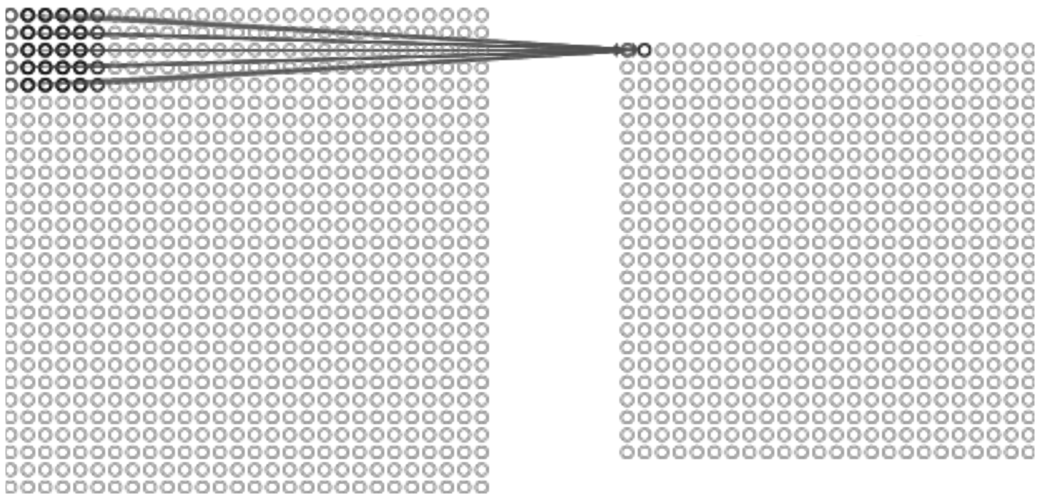
- There is no reason to restrict the encoder and decoder to consist of a single layer
- We can then create deep autoencoders by stacking multiple layers, each with an activation function.

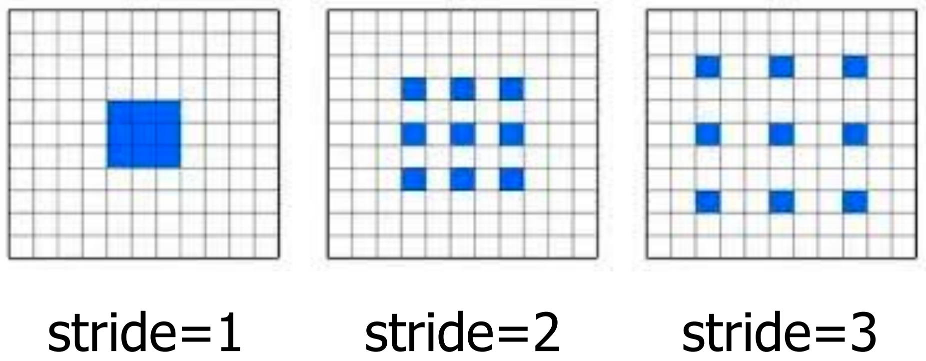
Convolutional Autoencoder



- When dealing with images, the layers can be convolutional ones.

Reminder: 2D Convolutional Layer

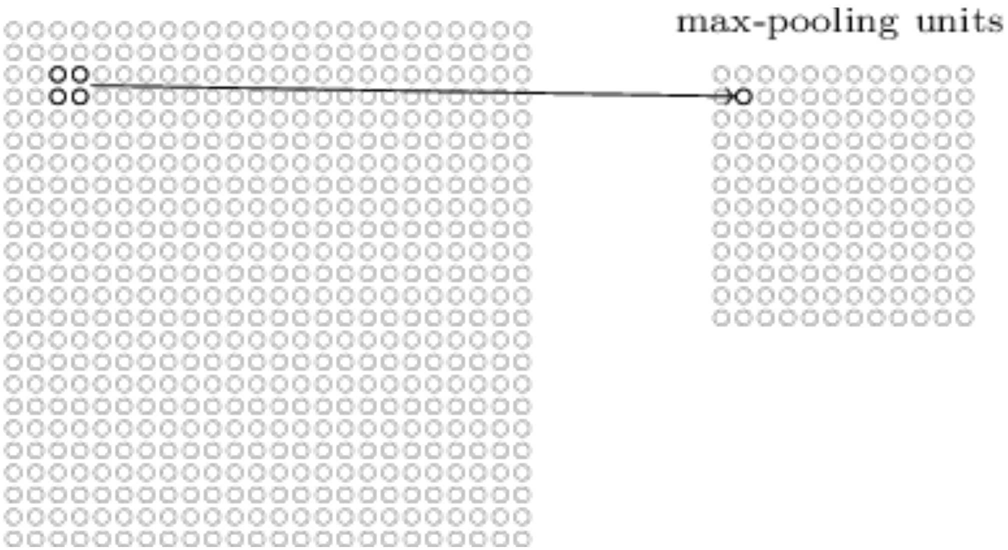


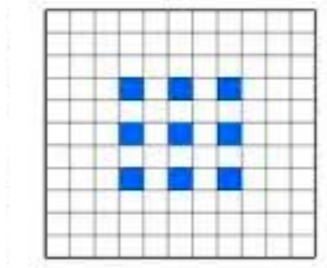


$$\sigma\left(b + \sum_{x=0}^{n_x} \sum_{y=0}^{n_y} w_{x,y} a_{i+x,j+y}\right)$$

- Using a stride > 1 reduces the image size.

Reminder: 2D Convolutional Layer





stride=2

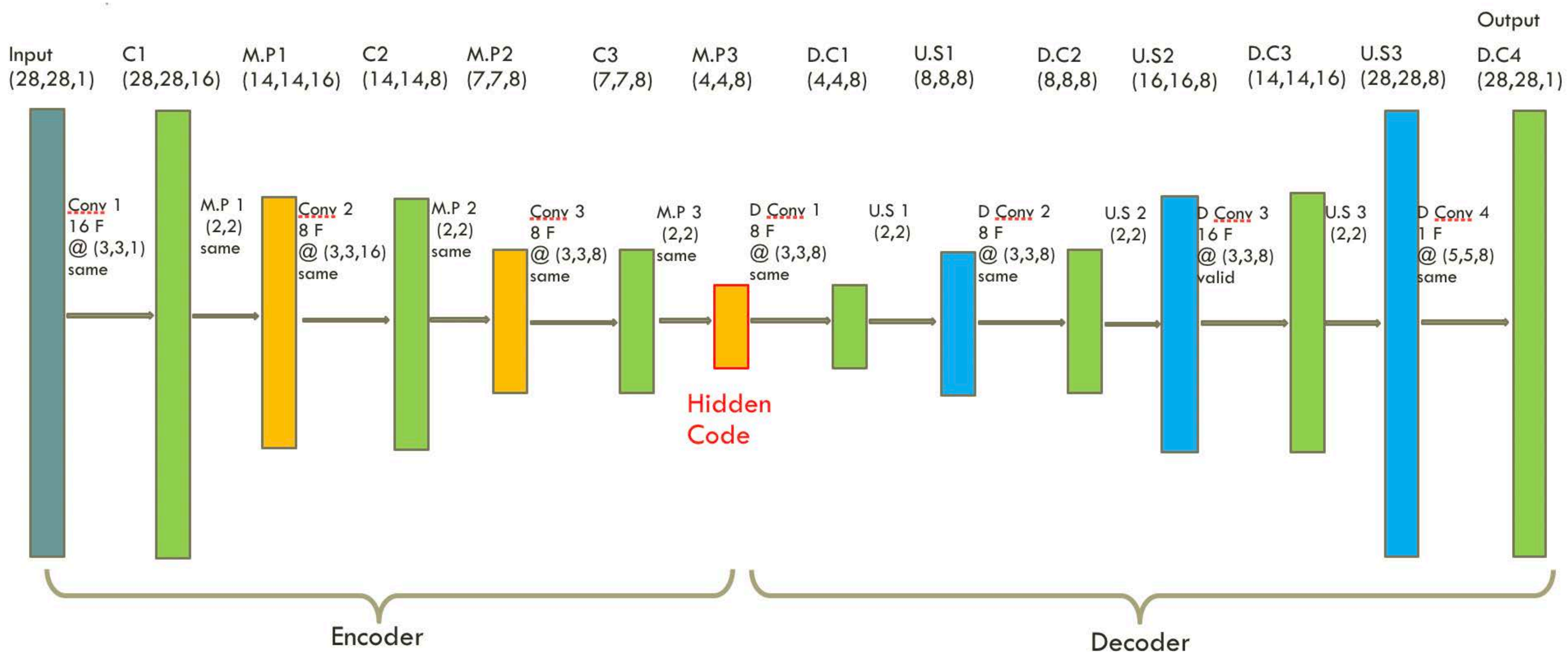
$$\sigma\left(b + \sum_{x=0}^{n_x} \sum_{y=0}^{n_y} w_{x,y} a_{i+x,j+y}\right)$$

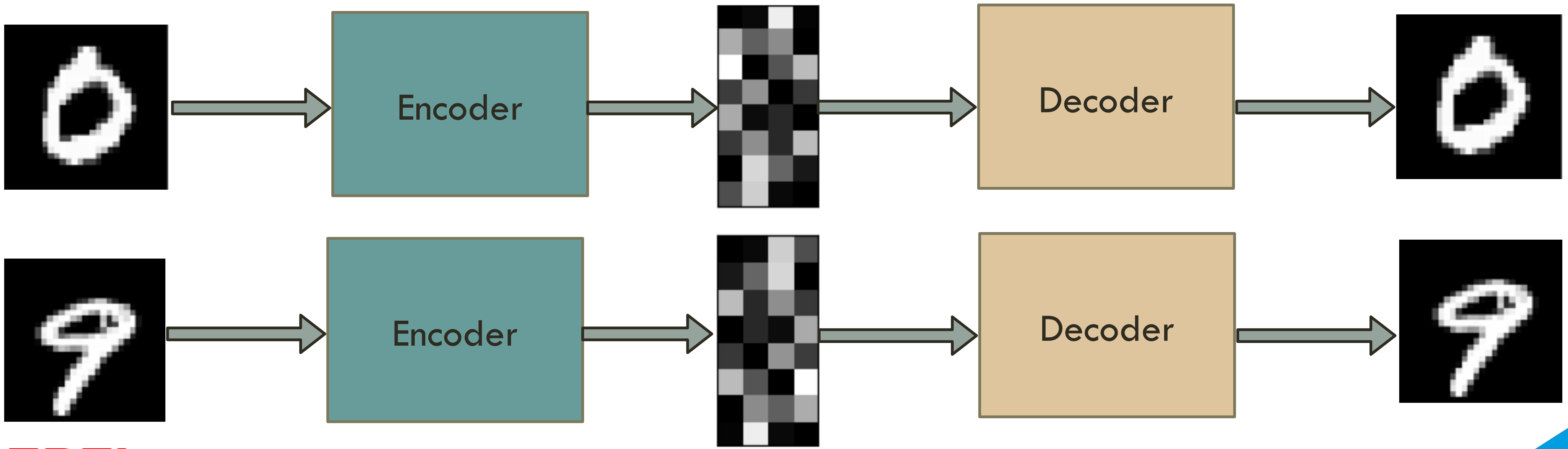
- Using a stride > 1 reduces the image size.
- Transposed convolution can be used to increase the image size.

PyTorch Translation

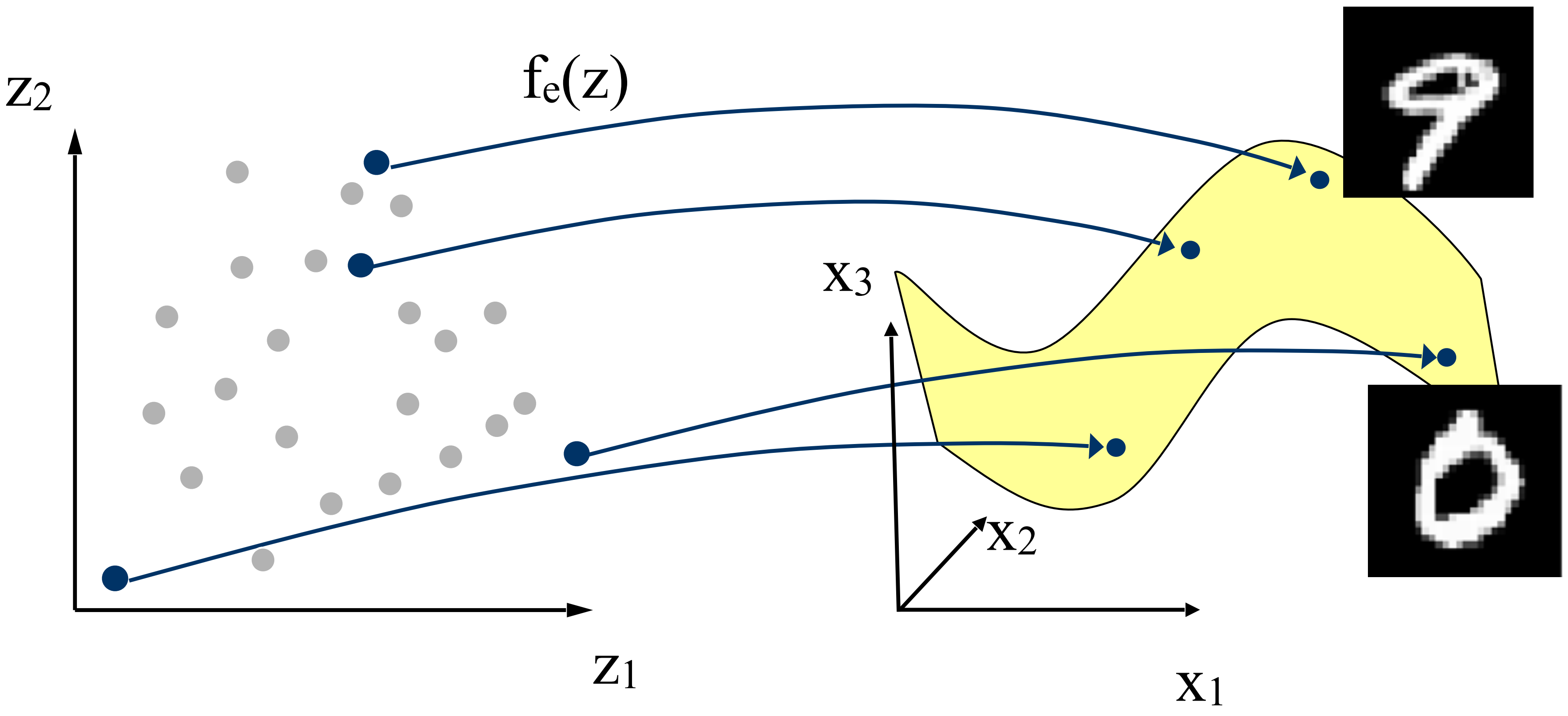
```
class Autoencoder(nn.Module):

  def __init__(nChannel=10,nHidden=50):
      self.convE1 = nn.Conv2d(1, nChannel,kernel_size=5,stride=2)
      self.convE2 = nn.Conv2d(nChannel,20,kernel_size=5,stride=2)
      self.convD1 = nn.ConvTranspose2d(nChannel,nChannel,kernel_size=4,stride=2,padding=1)
      self.convD2 = nn.ConvTranspose2d(nChannel,1 ,kernel_size=4,stride=2,padding=1)

  def encode(self,x):
      x = sigma(self.convE1(x))
      z = sigma(self.convE2(x))
      return z

  def decode(self, z):
      x = sigma(self.convD1(z)
      x = self.convD2(x)
      return x

  def forward(self,x):
      z = self.encode(x)
      return self.decode(z)
```

MNIST


Output
Input (28,28,1)
C1 (28,28,16)
M.P1 (14,14,16)
C2 (14,14,8)
M.P2 (7,7,8)
C3 (7,7,8)
M.P3 (4,4,8)
D.C1 (4,4,8)
U.S1 (8,8,8)
D.C2 (8,8,8)
U.S2 (16,16,8)
D.C3 (14,14,16)
U.S3 (28,28,8)
D.C4 (28,28,1)
Conv 1 16 F @ (3,3,1) same
M.P 1 (2,2) same
Conv 2 8 F @ (3,3,16) same
M.P 2 (2,2) same
Conv 3 8 F @ (3,3,8) same
M.P 3 (2,2) same
D Conv 1 8 F @ (3,3,8) same
U.S 1 (2,2)
D Conv 2 8 F @ (3,3,8) same
U.S 2 (2,2)
D Conv 3 16 F @ (3,3,8) valid
U.S 3 (2,2)
D Conv 4 1 F @ (5,5,8) same
Hidden Code
Encoder
Decoder


Encoder
Decoder
Encoder
Decoder

Latent Space



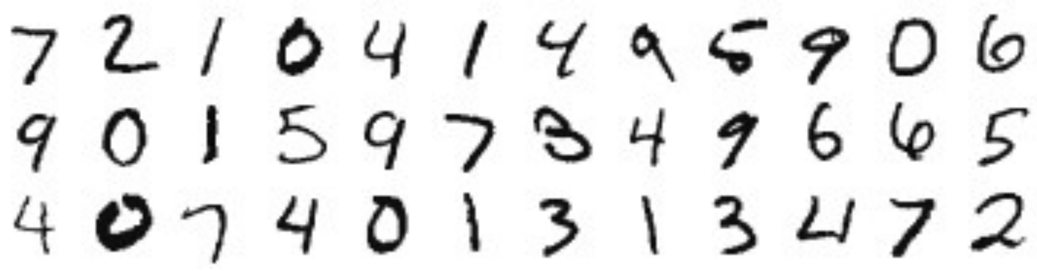
Exploring the Latent Space

Wandering through **z**-space and observing the effects in **x**-space

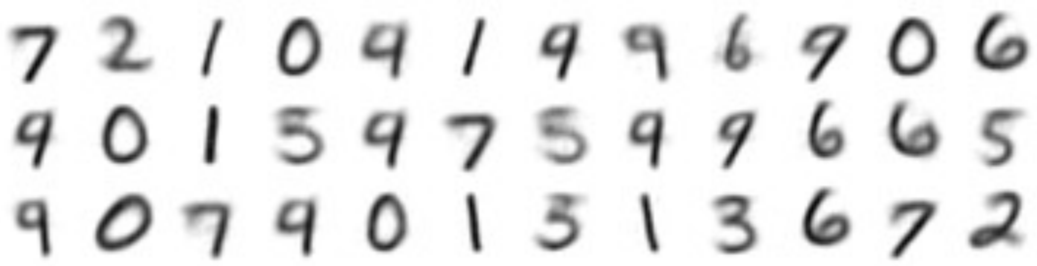
http://dpkingma.com/sgvb_mnist_demo/demo.html

Encoding / Decoding

X (original samples)



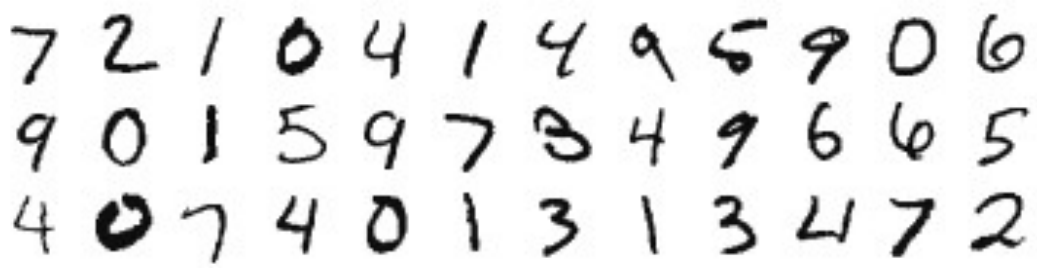
$g \circ f(X)$ (CNN, $d = 2$)



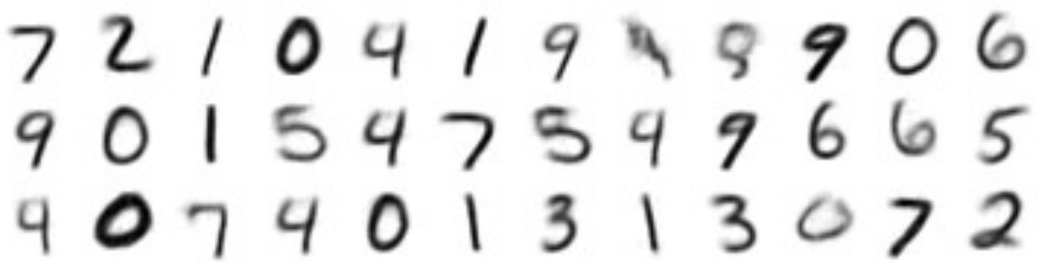
$g \circ f(X)$ (PCA, $d = 2$)

Encoding / Decoding

X (original samples)



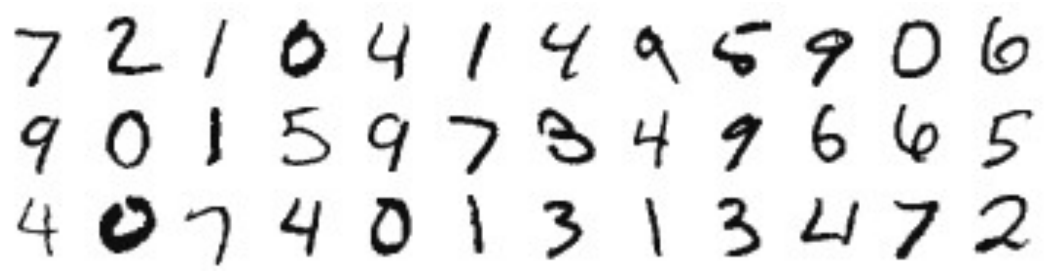
$g \circ f(X)$ (CNN, $d = 4$)



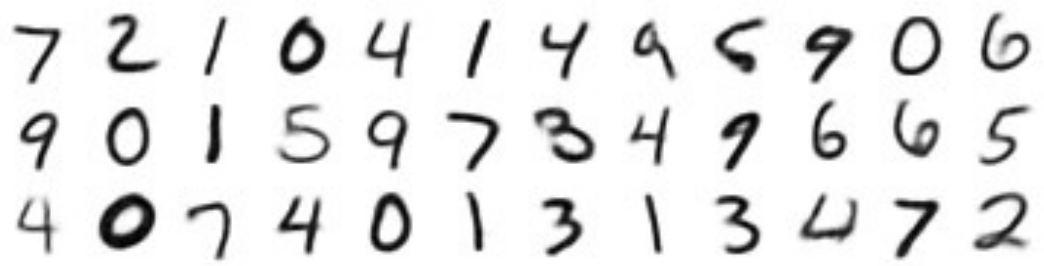
$g \circ f(X)$ (PCA, $d = 4$)

Encoding / Decoding

X (original samples)



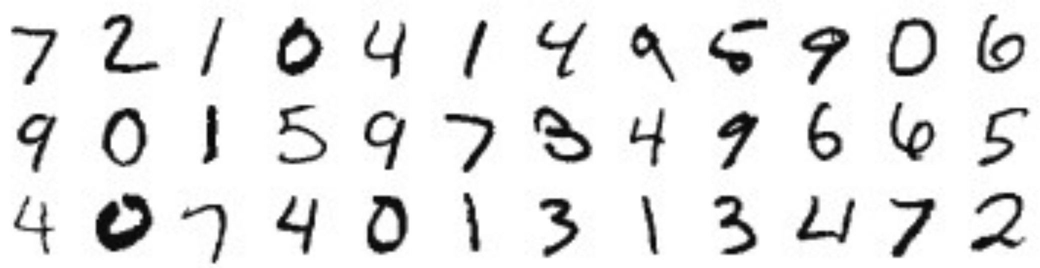
$g \circ f(X)$ (CNN, $d = 8$)



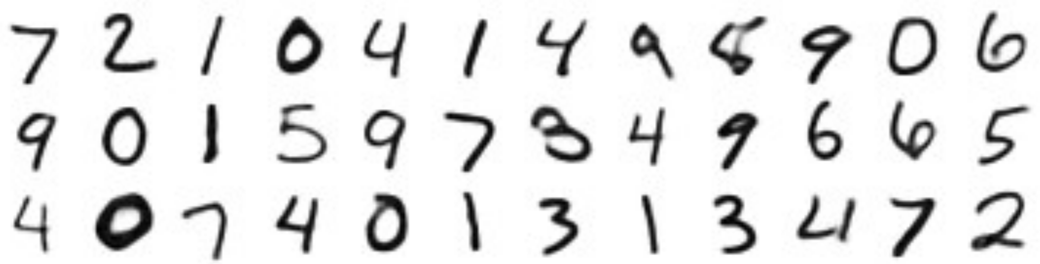
$g \circ f(X)$ (PCA, $d = 8$)

Encoding / Decoding

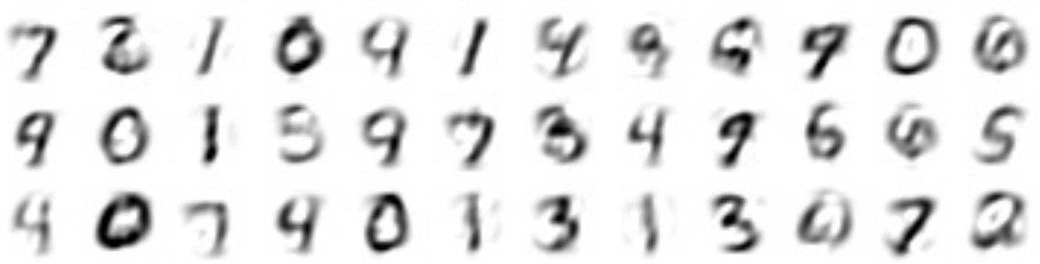
X (original samples)



$g \circ f(X)$ (CNN, $d = 16$)

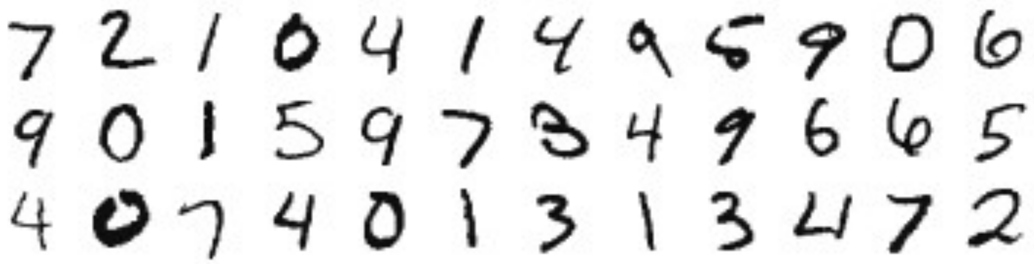


$g \circ f(X)$ (PCA, $d = 16$)

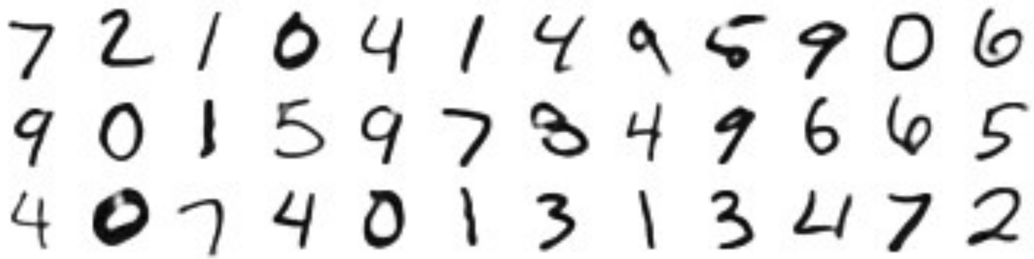


Encoding / Decoding

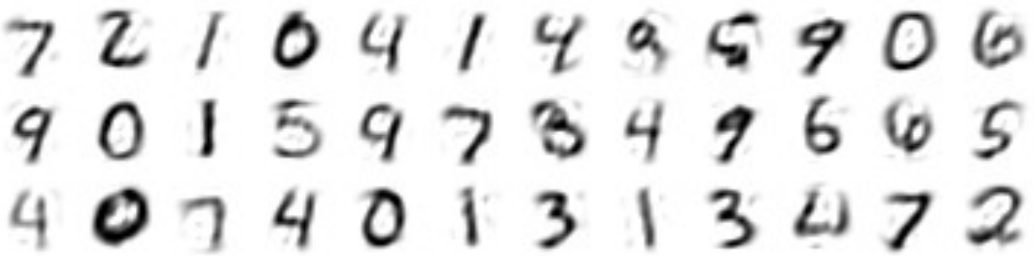
X (original samples)



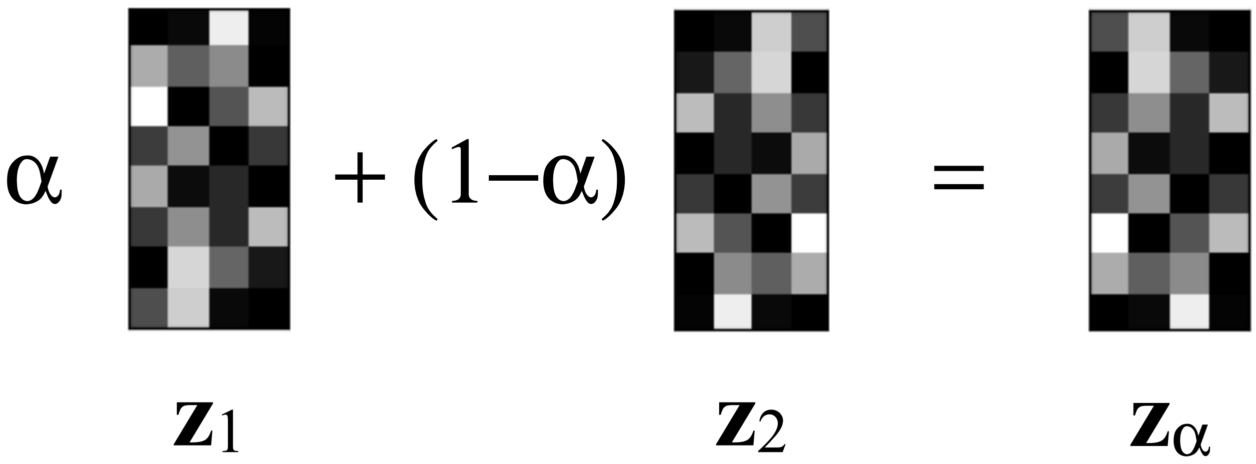
$g \circ f(X)$ (CNN, $d = 32$)

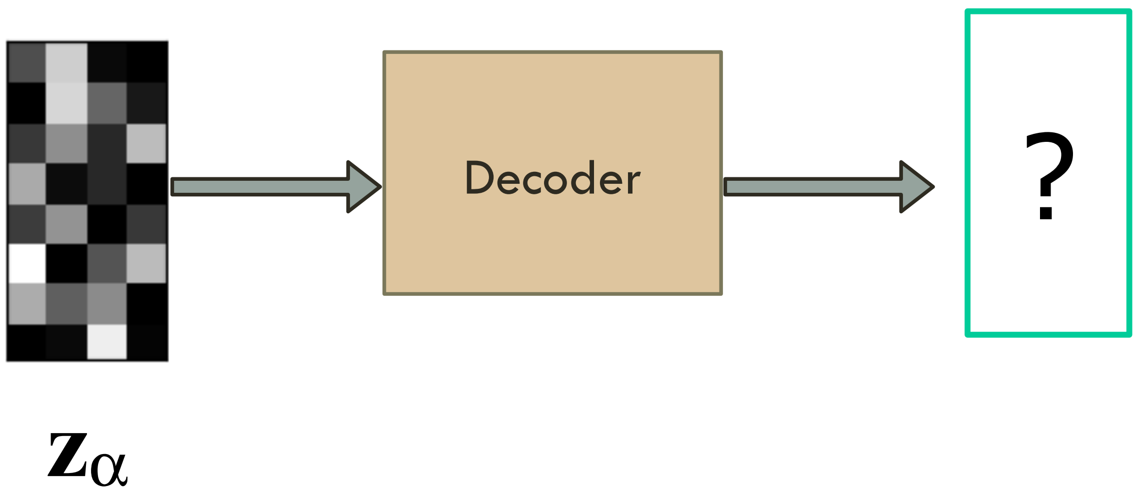


$g \circ f(X)$ (PCA, $d = 32$)

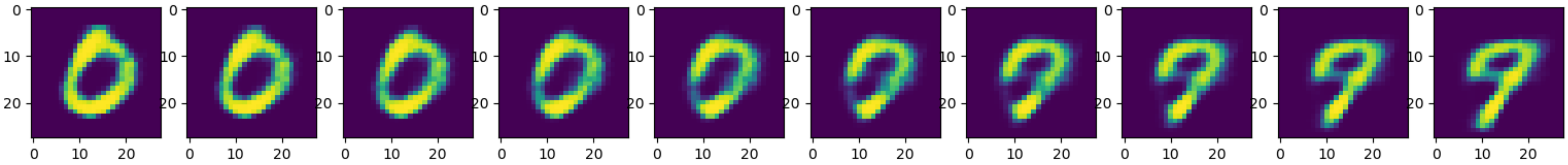


Interpolation

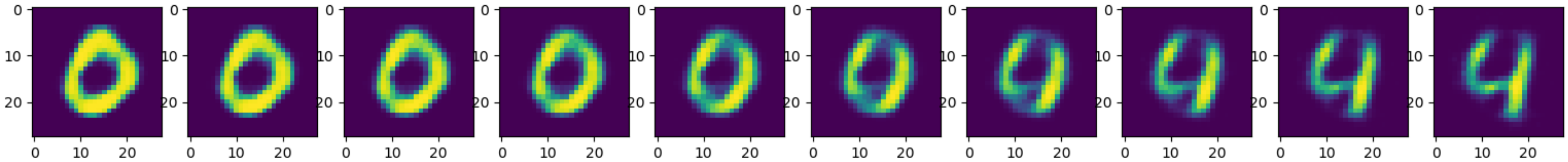




Interpolation

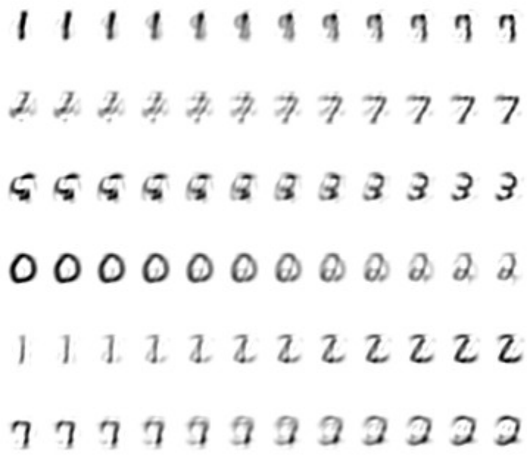


0 <—> 9

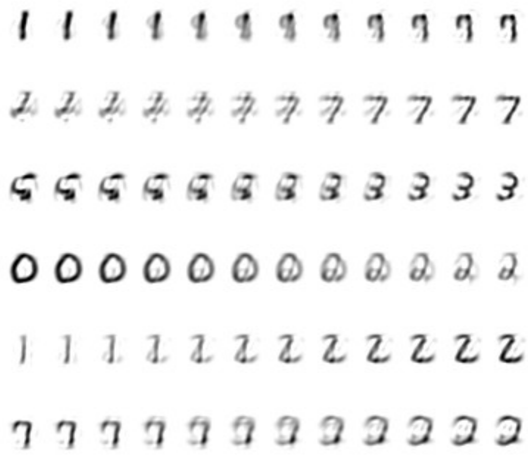


0 <—> 4

PCA vs Autoencoder Interpolation



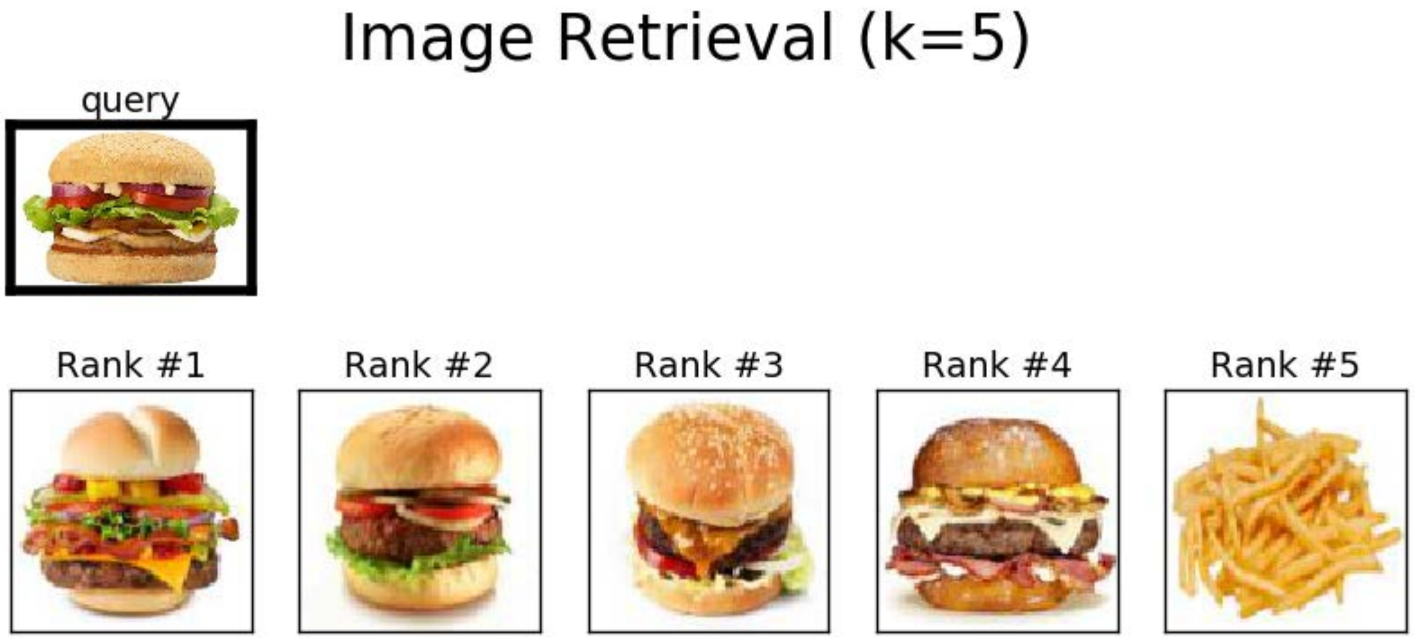
PCA (d=32)



Autoenc (d=8)

Autoenc (d=32)

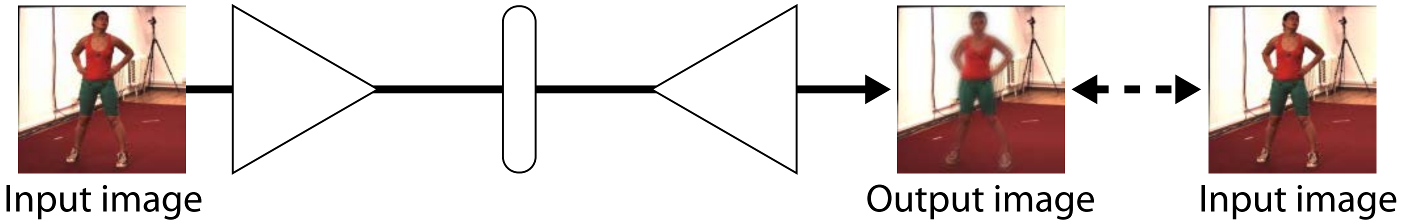
Application to Image Retrieval



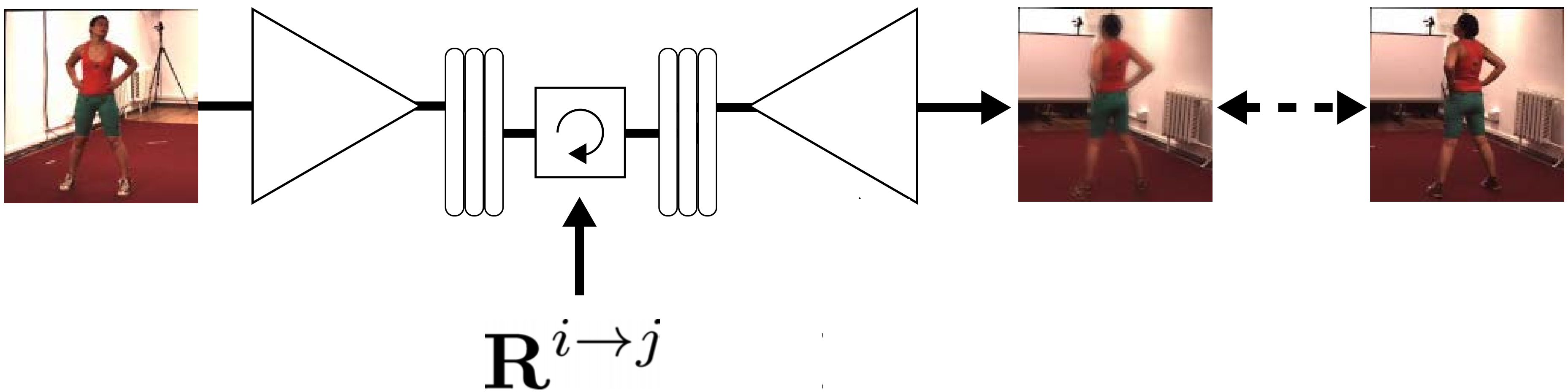
- A code provides a compact representation of the input.
- It can be used for retrieval in a large data collection, e.g., via by k nearest neighbors.

https://towardsdatascience.com/find-similar-images-using-autoencoders-315f374029ea

Optional: Novel View Synthesis

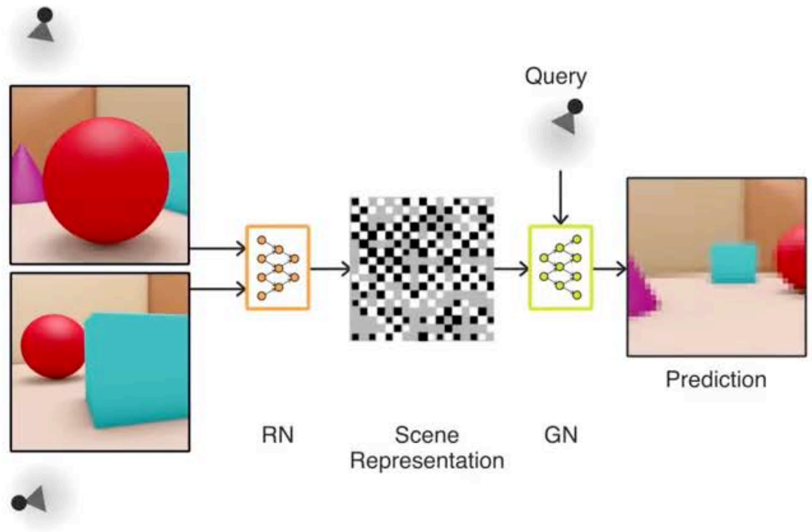


Conventional Autoencoder

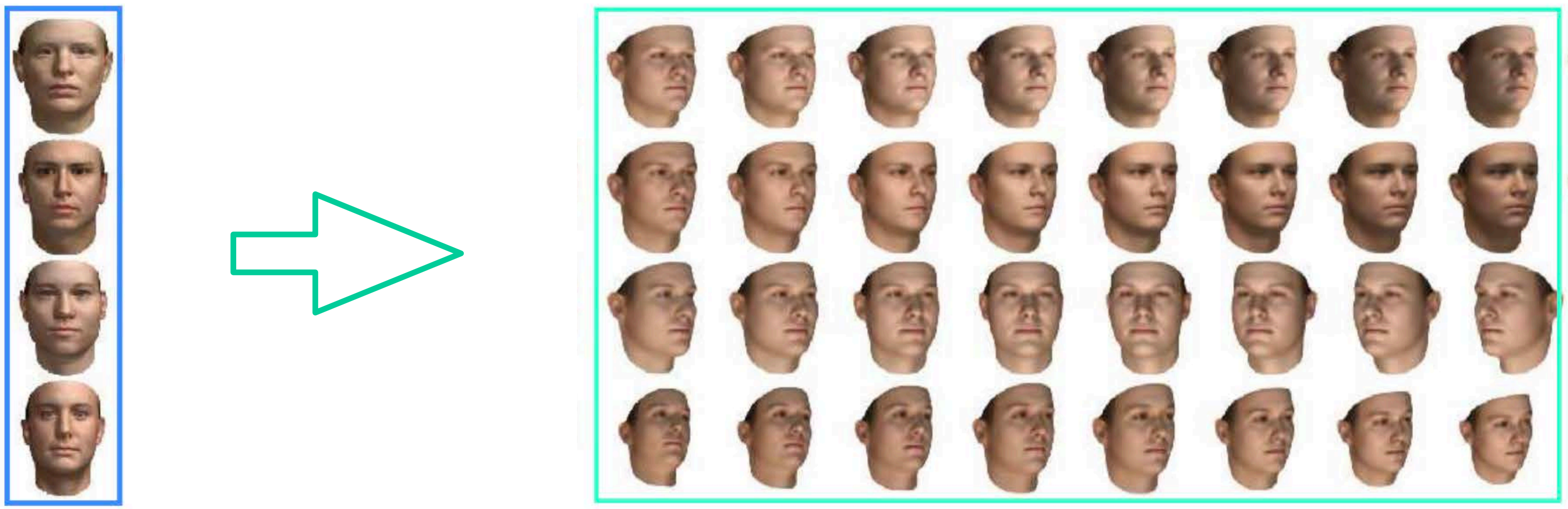


Rotation Aware Autoencoder

Optional: Novel View Synthesis

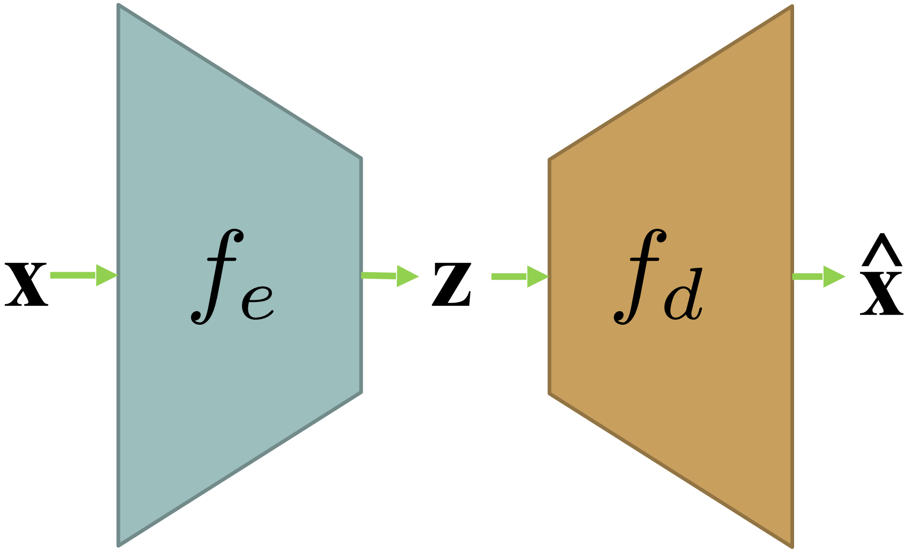


DeepMind



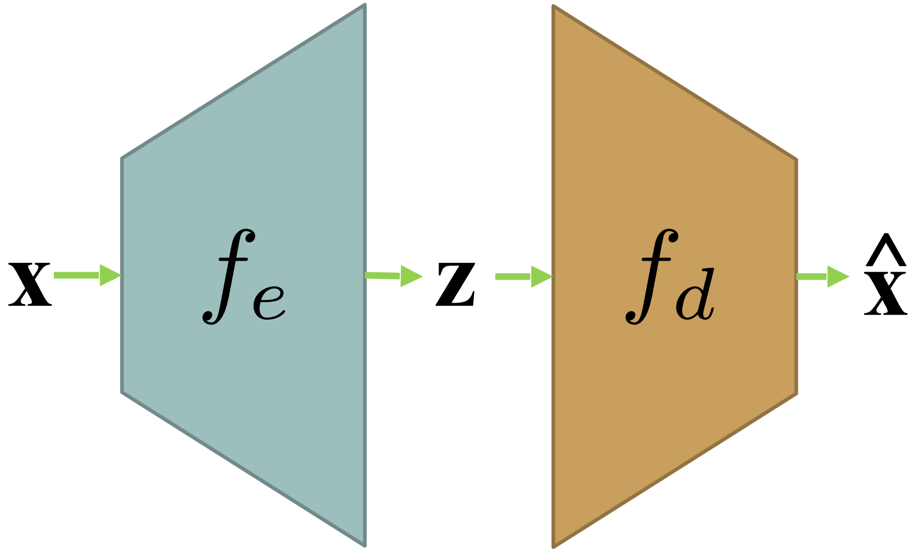
Worrall et al. ICCV’17

Under- vs Over-Complete



Undercomplete: dim(z) < dim(x)

- Compresses the input.
- Captures correlations.

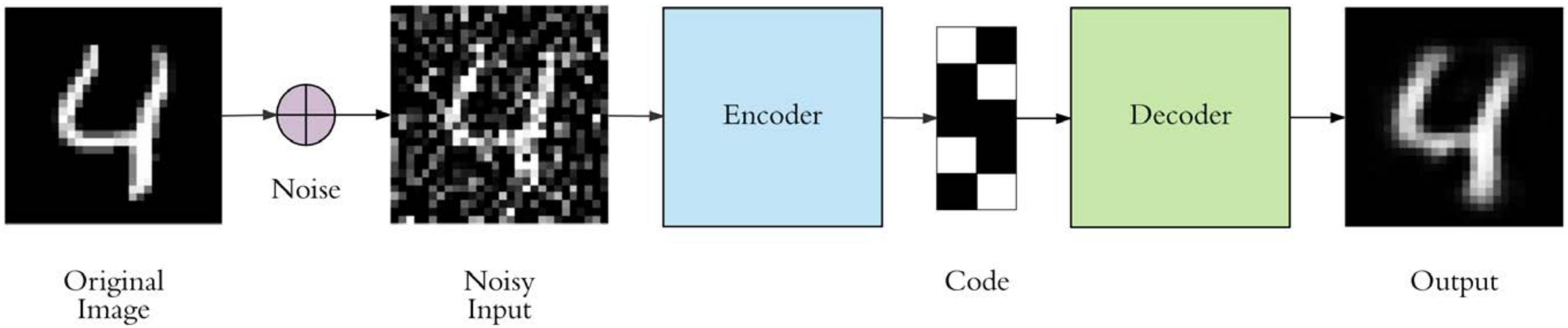


Overcomplete: dim(z) > dim(x)

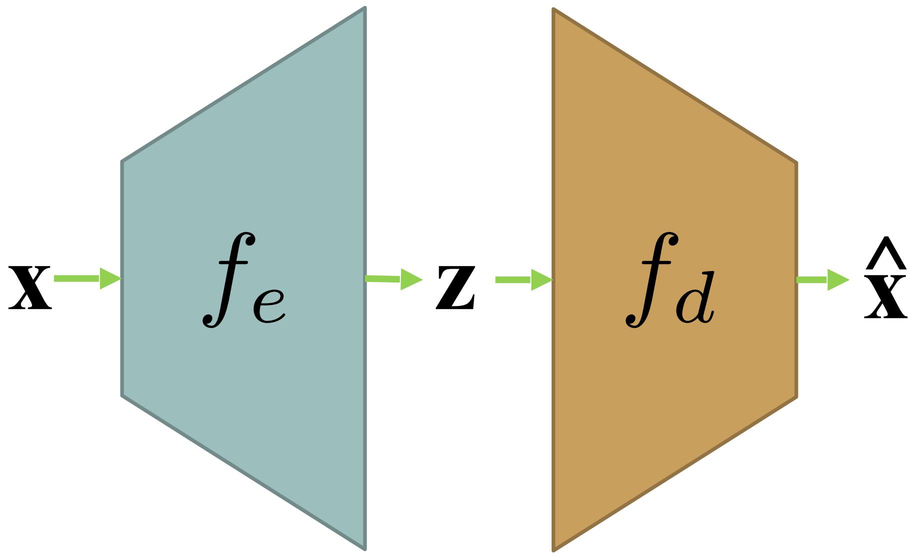
- Higher dimension can help.
- Degenerate solutions possible.
- Need a regularization term.

Denoising Autoencoders

- Having a low-dimensional latent representation encourages the autoencoder to learn an “intelligent” mapping.
- With dimensionality expansion, one could simply learn to copy the input.
- To avoid this, one can add noise to the input and aim to reconstruct a noise-free version of the input:



Contrastive Autoencoders



Overcomplete: dim(z) > dim(x)

Another way to avoid trivial solutions is to train with a regularization term:

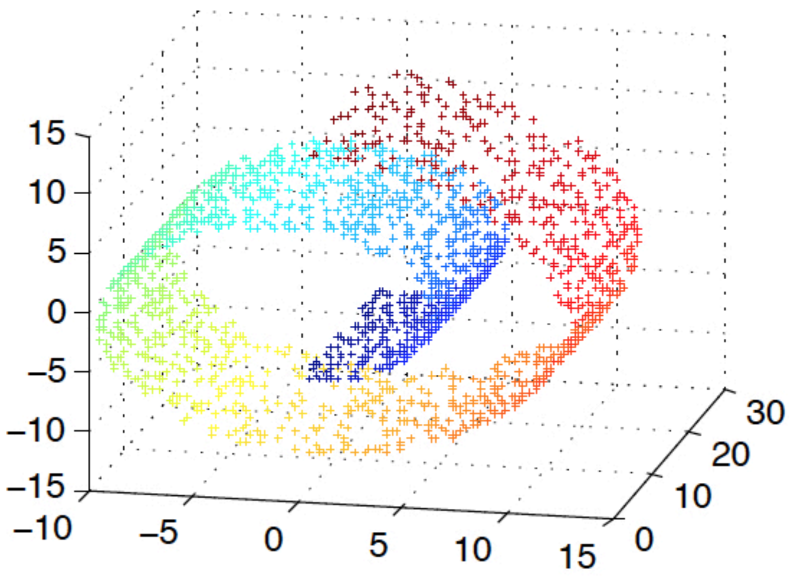
$$R(\mathbf{w}) = \sum_n L(\mathbf{x}_n, \mathbf{w}) + \lambda \Omega(\mathbf{x}_n, \mathbf{w}) \ ,$$

$$L(\mathbf{x}, \mathbf{w}) = \| f_d(f_e(\mathbf{x}, \mathbf{w})) - \mathbf{x} \|^2 \ ,$$

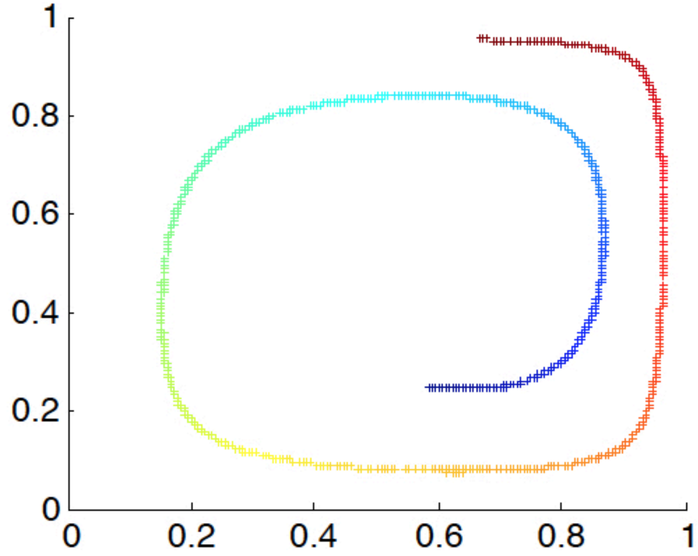
$$\Omega(\mathbf{w}) = \sum_{i,j} \left[\frac{\partial f_e(\mathbf{x}, \mathbf{w})_j}{\partial x_i} \right]^2 \ .$$

- When $\lambda \to 0$, $f_e \to$ identity.
- When $\lambda \to \infty$, $f_e \to$ constant.

Back to the Swiss Roll



Input

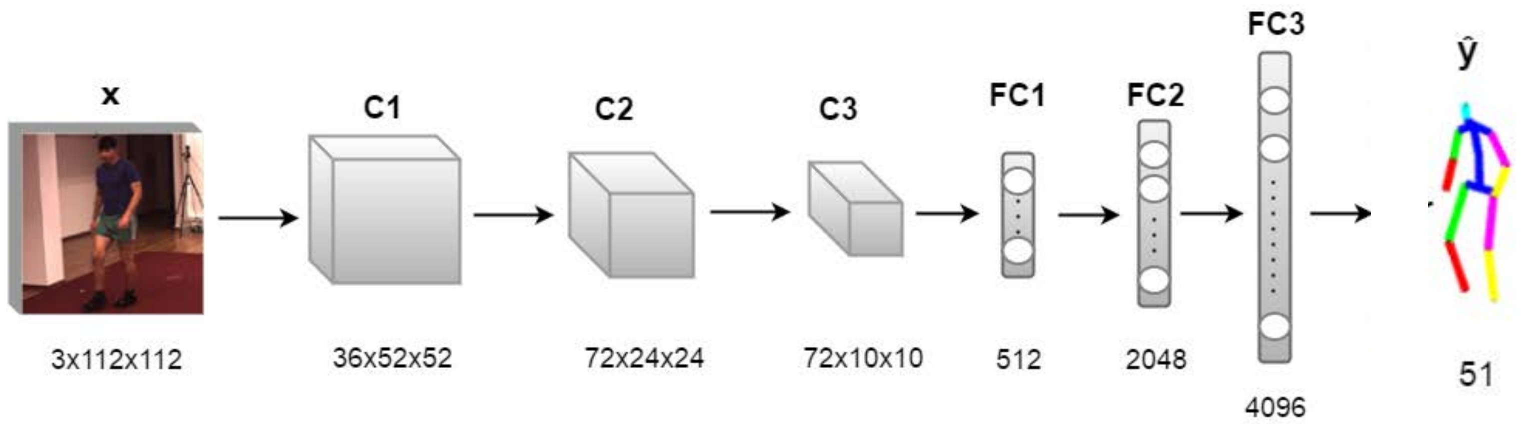


Latent representation

Embedding with Autoencoder Regularization

Wu et al., ECML-PKDD'13

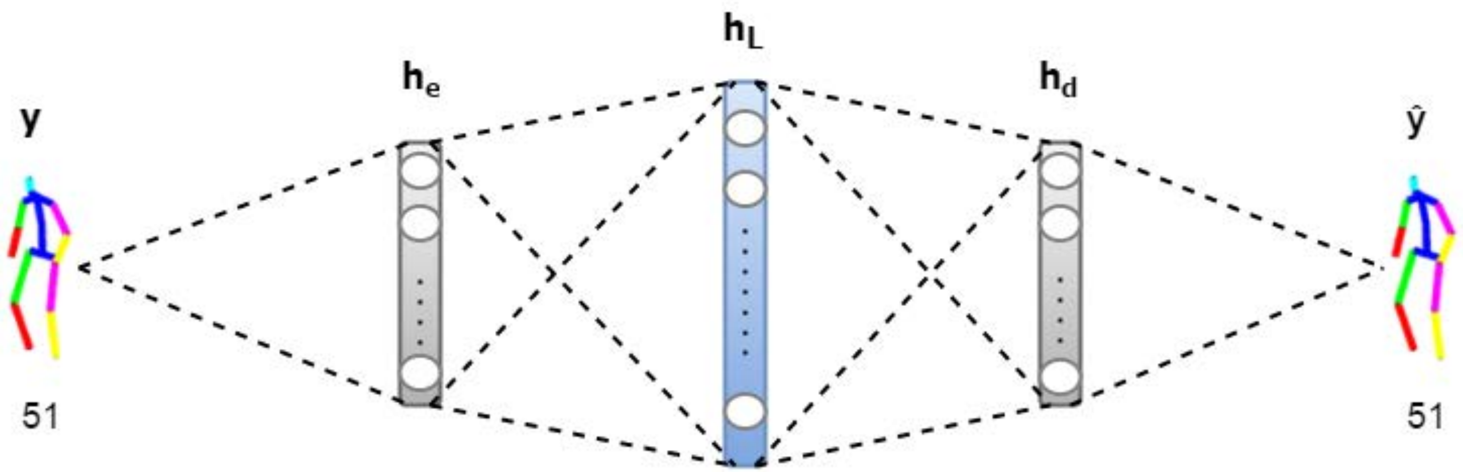
Optional: Body Pose Estimation



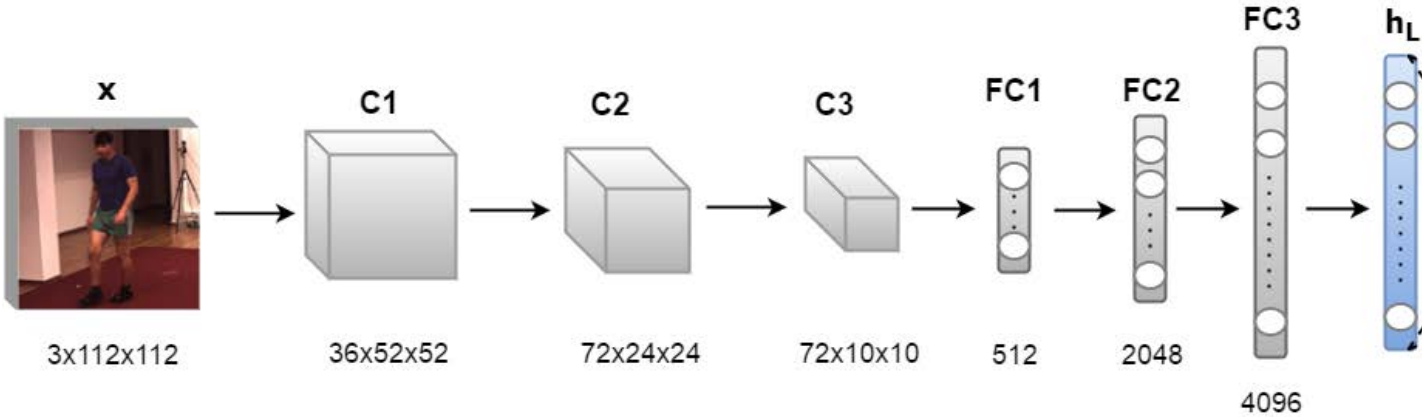
Input: $\mathbf{I}$

Output: $\{\mathbf{y}_j\}_{1 \leq j \leq J}$

Autoencoder:



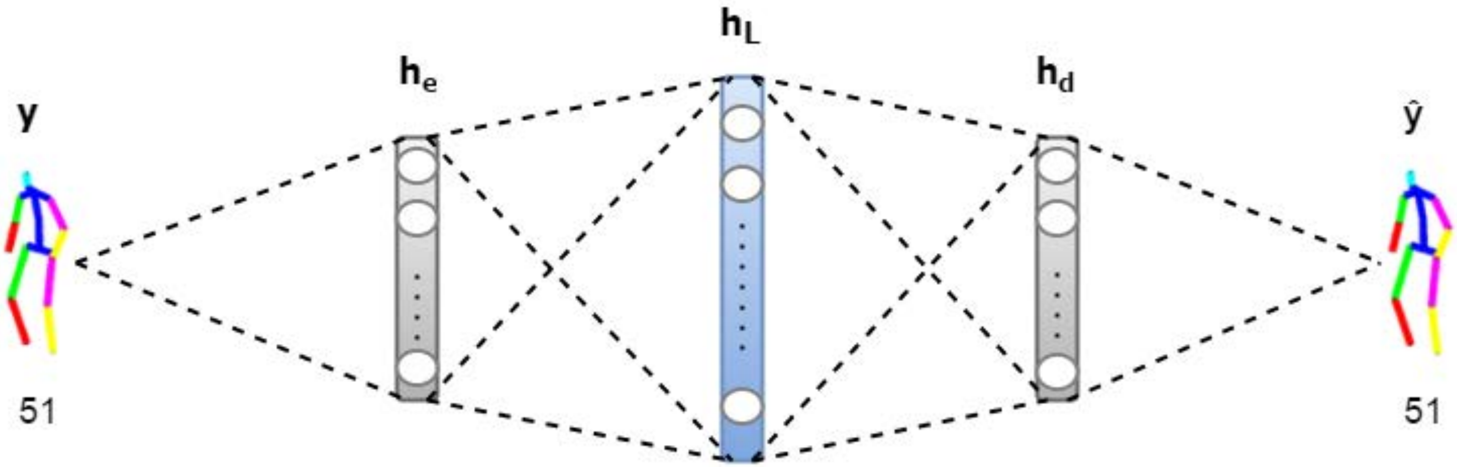
Optional: Structured Prediction



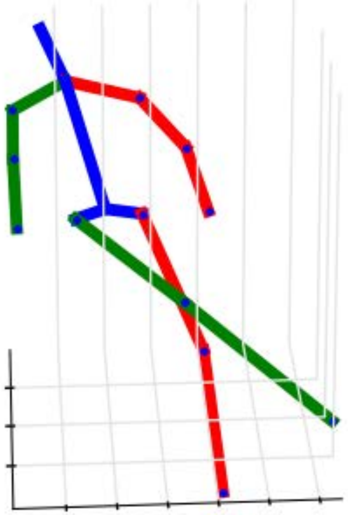
Input: $\mathbf{I}$

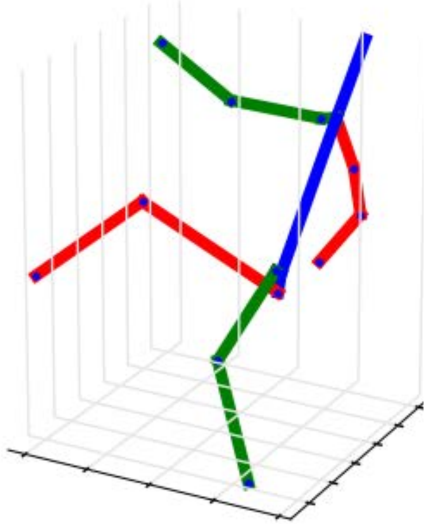
Output: $\{\mathbf{y}_j\}_{1 \leq j \leq J}$

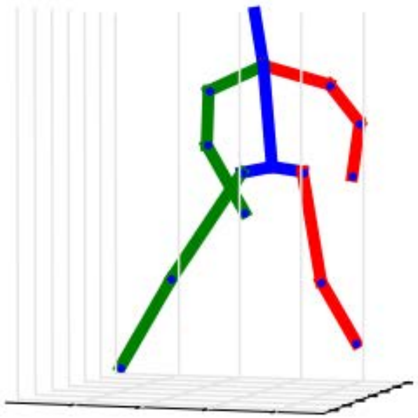
Autoencoder:



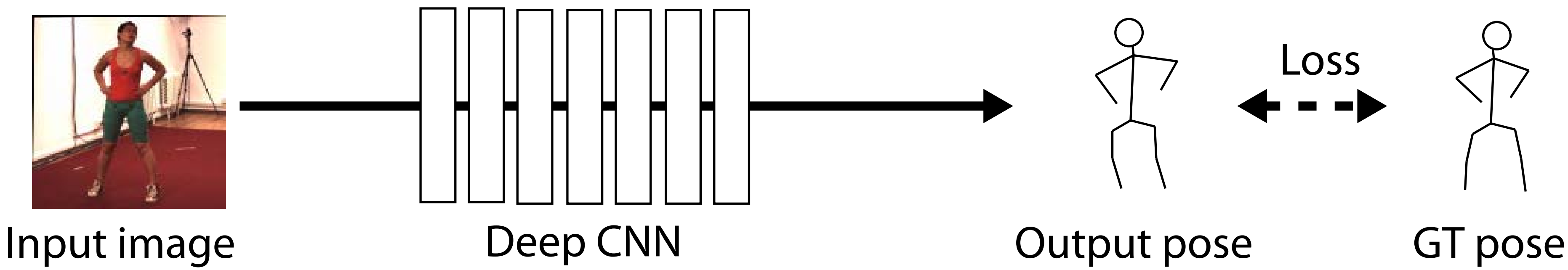
Monocular 3D Pose Estimation







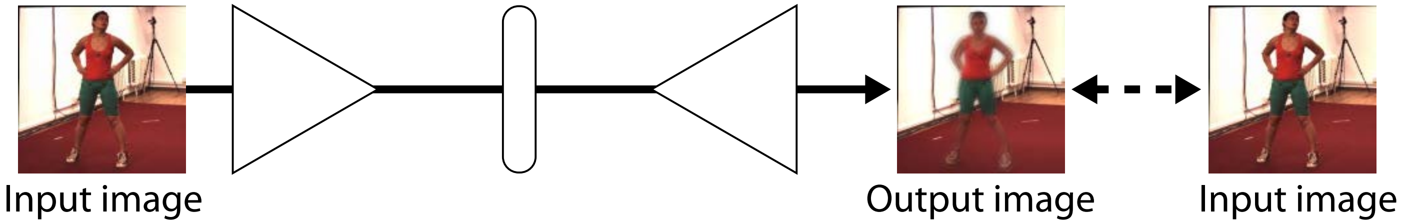
The Problem with Direct Estimation



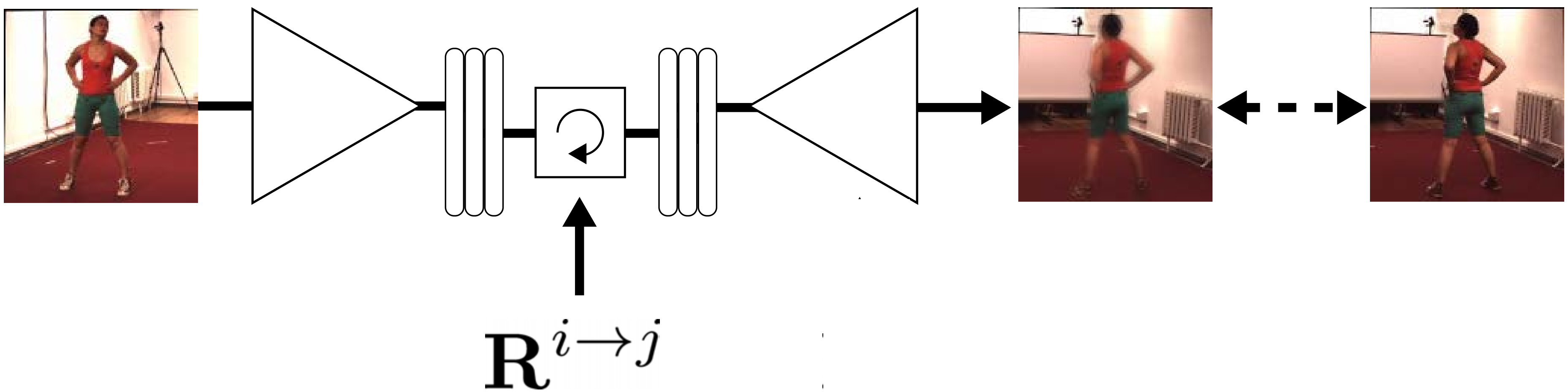
- The human body has many degrees of freedom.
- Going directly from image to 3D pose requires a very deep net.
- Training such a deep net requires a lot of training data.

Can we learn a representation that has fewer degrees of freedom for specific activities?

Novel View Synthesis Revisited

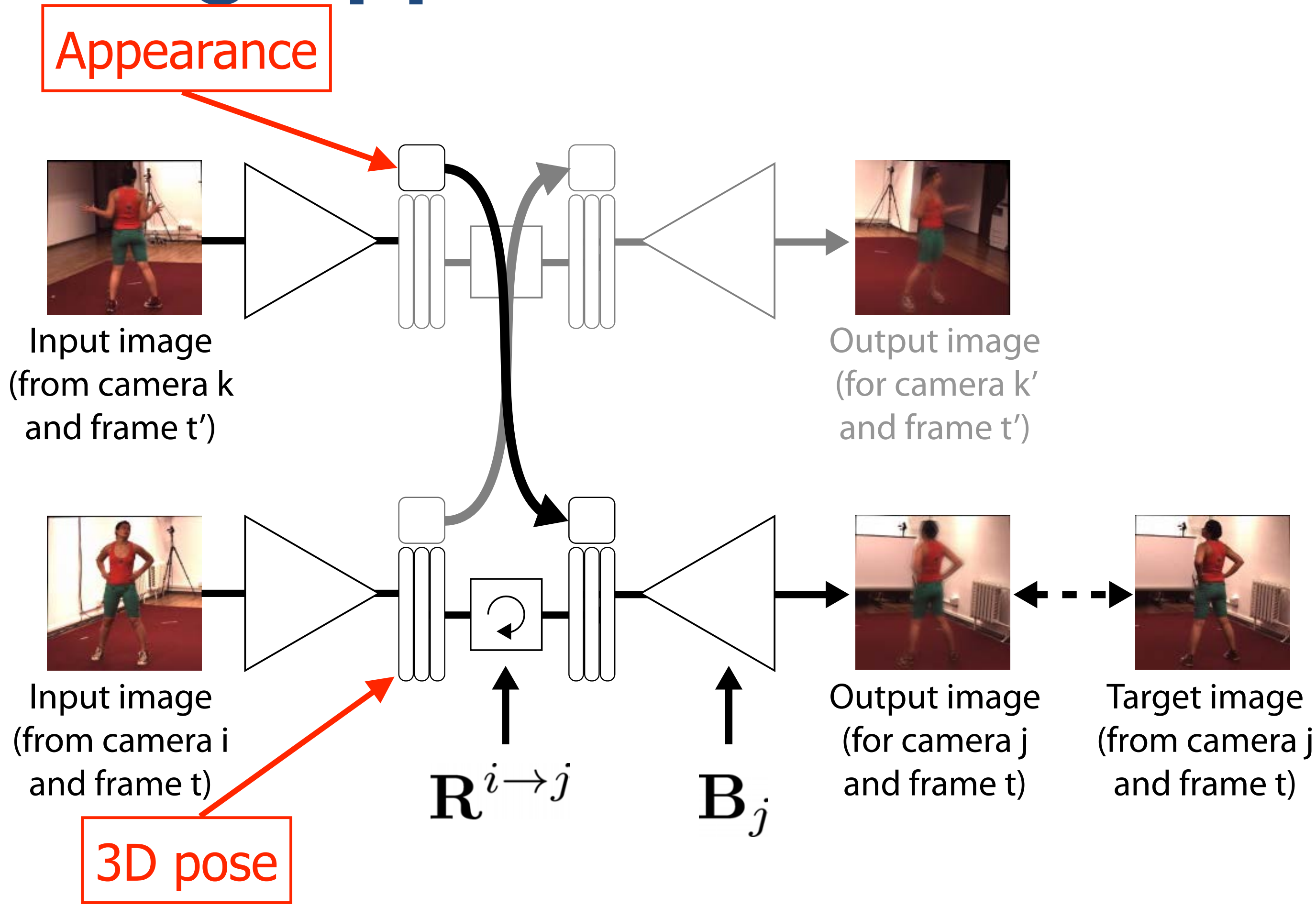


Conventional Autoencoder



Rotation Aware Autoencoder

Separating Appearance from Geometry



- The latent representation comprises a Nx3 matrix that encodes the 3D pose and a separate vector that models appearance.
- Before decoding the appearance vectors are swapped to ensure that they are similar in different images.

Latent Representation

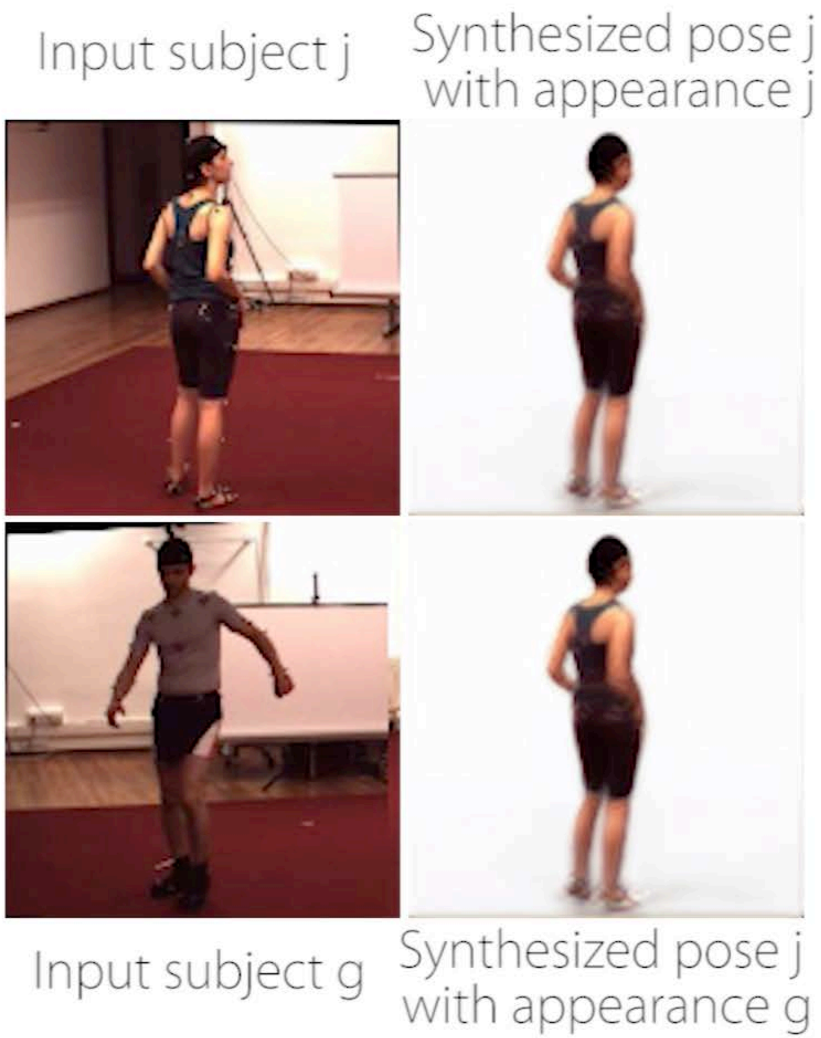
Input

Geometric encoding
(rotating point cloud)

Output

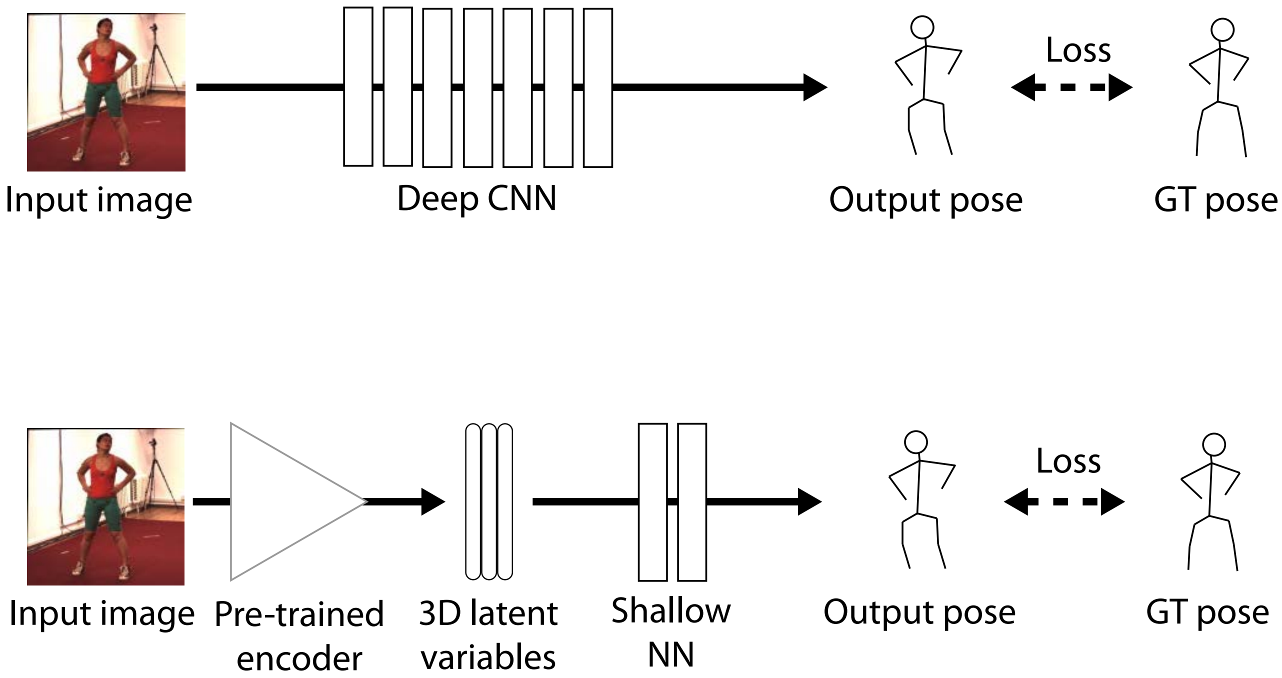
The test subject is reconstructed with the right pose but an approximate appearance.

Swapping Appearance



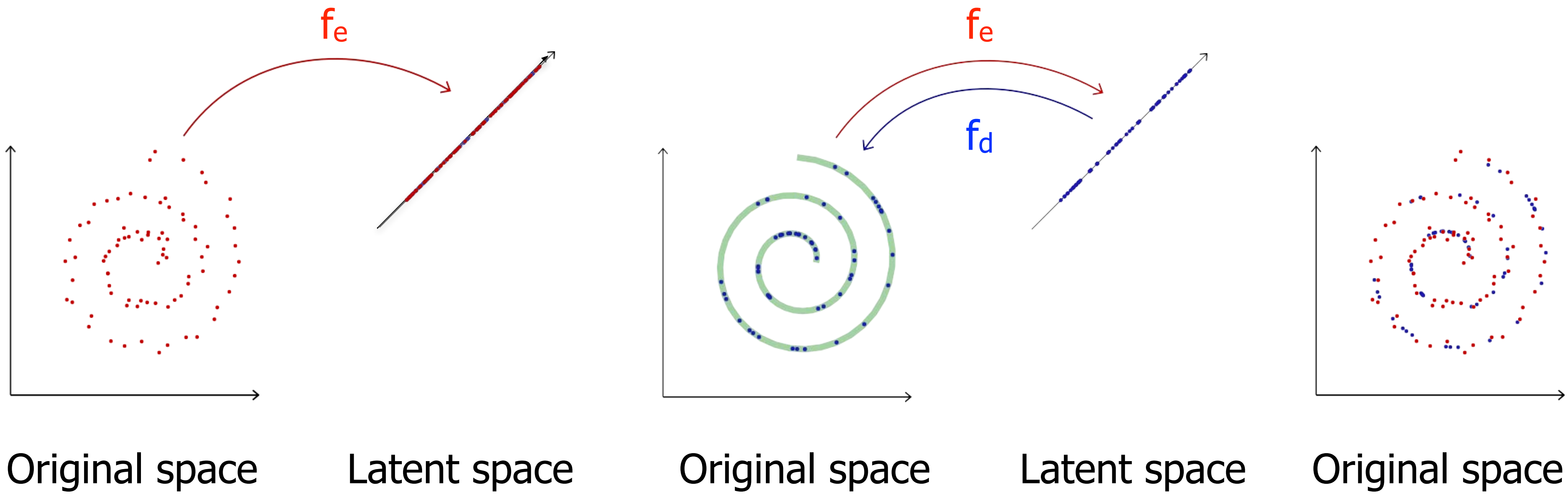
EPFL

Direct Estimation vs Using Latent Variables



- Training a shallow network requires less training data than a deeper one.
- This is important in specialized areas for which large training databases are hard to build.

Dimensionality Reduction



$$\mathbf{z} = f_e(\mathbf{x})$$

$$\hat{\mathbf{x}} = f_d(\mathbf{z})$$

$$\hat{\mathbf{x}} \approx \mathbf{x}$$

- Removes unnecessary degrees of freedom.
- Models correlations between the real ones.
- Makes it possible to denoise the original data.
- Can done linearly or non-linearly.