Autoencoder

Industrial AI Lab.
Prof. Seungchul Lee

Unsupervised Learning

- Many applications such as image synthesis, denoising, super-resolution, speech synthesis, compression, etc. require to go beyond classification and regression, and model explicitly a high dimension signal.

- This modeling consists of finding “meaningful degrees of freedom” that describe the signal, and are of lesser dimension.

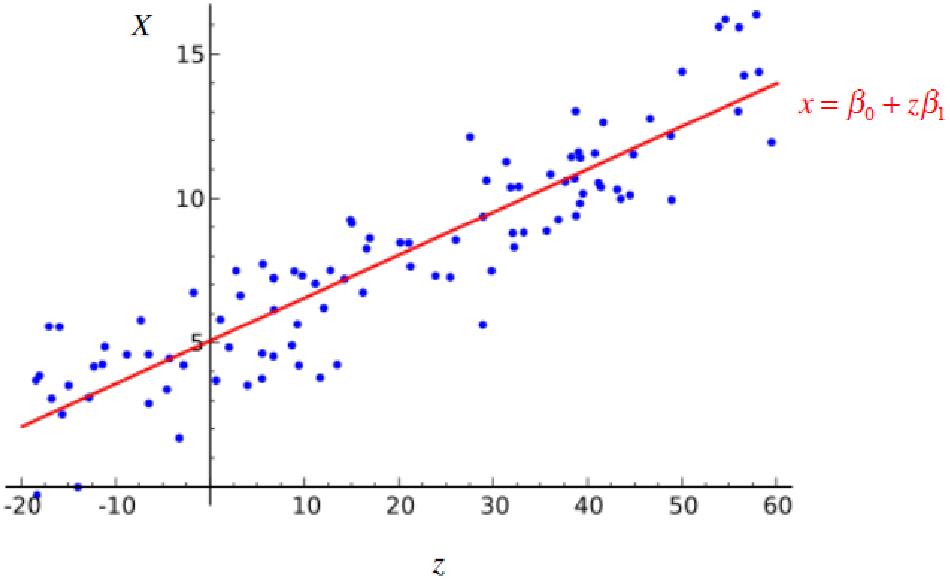
Unsupervised Learning

- Definition
  - Unsupervised learning refers to most attempts to extract information from a distribution that do not require human labor to annotate example
  - Main task is to find the 'best' representation of the data

- Dimension Reduction
  - Attempt to compress as much information as possible in a smaller representation
  - Preserve as much information as possible while obeying some constraint aimed at keeping the representation simpler

Recap: Linear Regression

- Most people think of linear regression as points and a straight line:

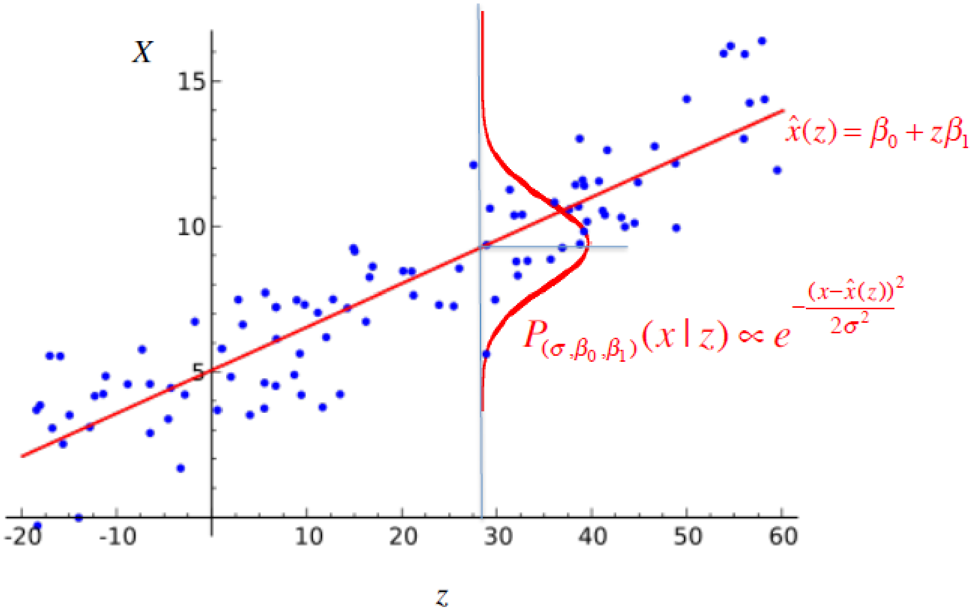


Recap: Linear Regression

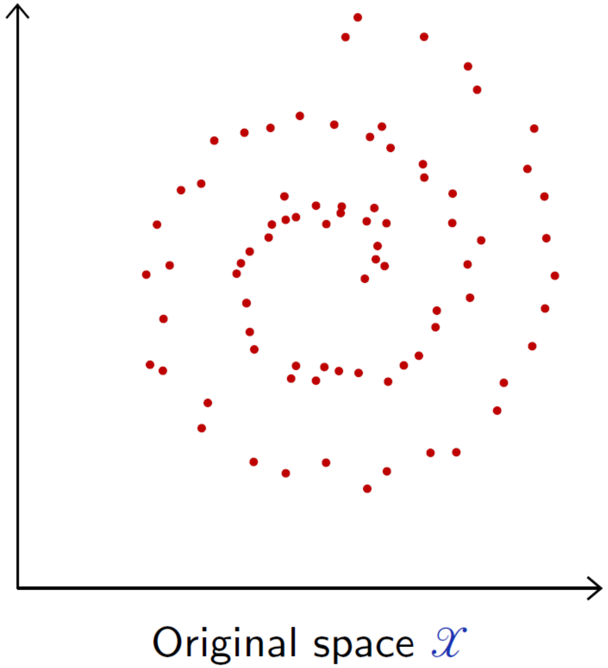
- Statisticians additionally have $P_\theta(X|Z)$
- True model
  - May not be too complicated as opposed to original data
- Observed data = true model + error
- Benefits of having an error model:
  - How likely is a data point
  - Confidence bounds
  - Compare models

- Q: how to find an unseen true model

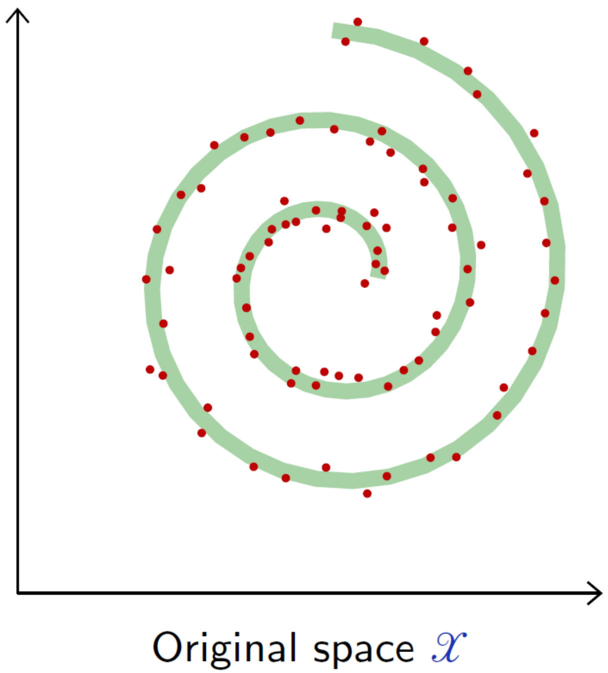
  (we never know the true model)



Original Space and Latent Space

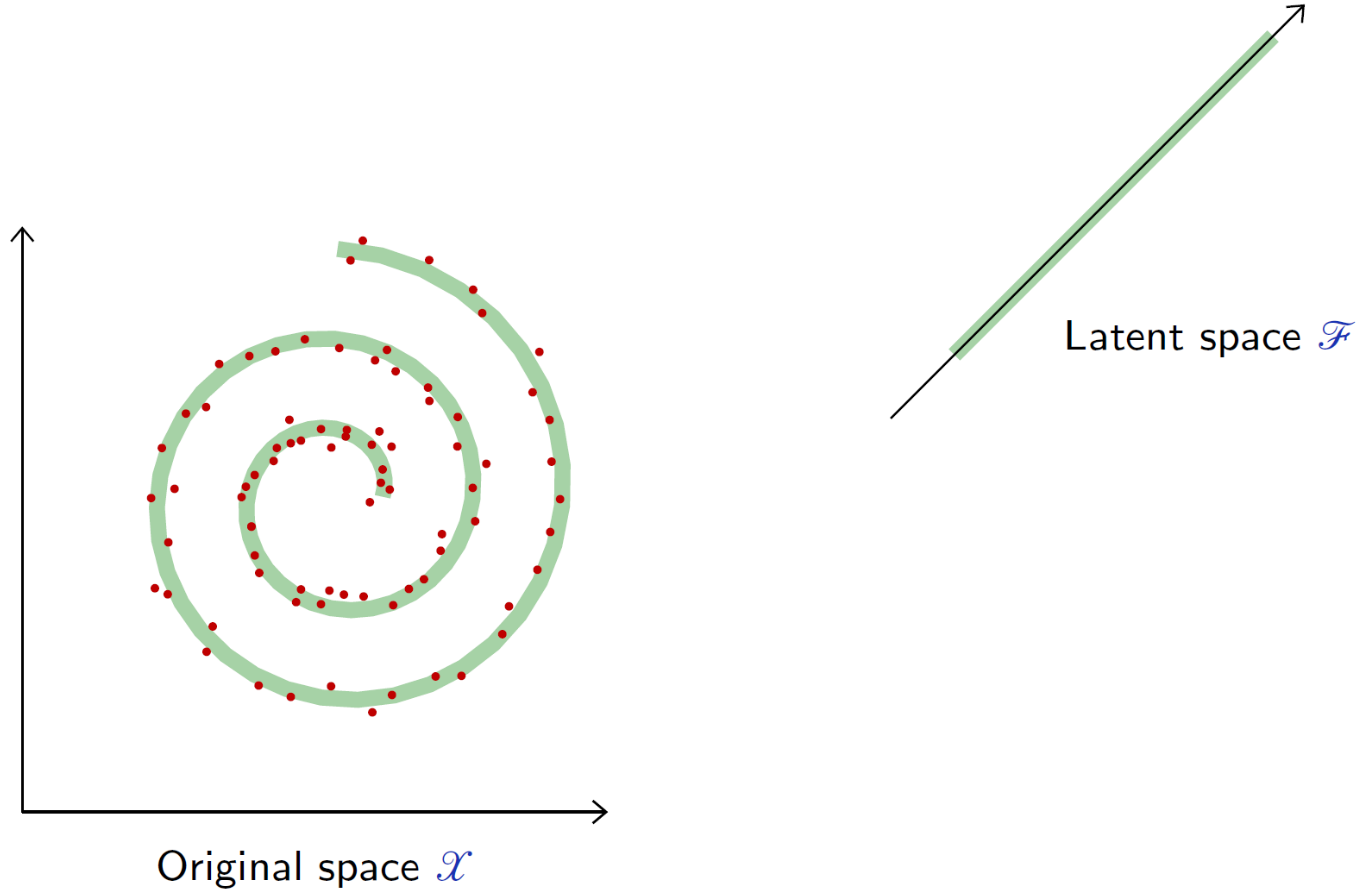


Original Space and Latent Space



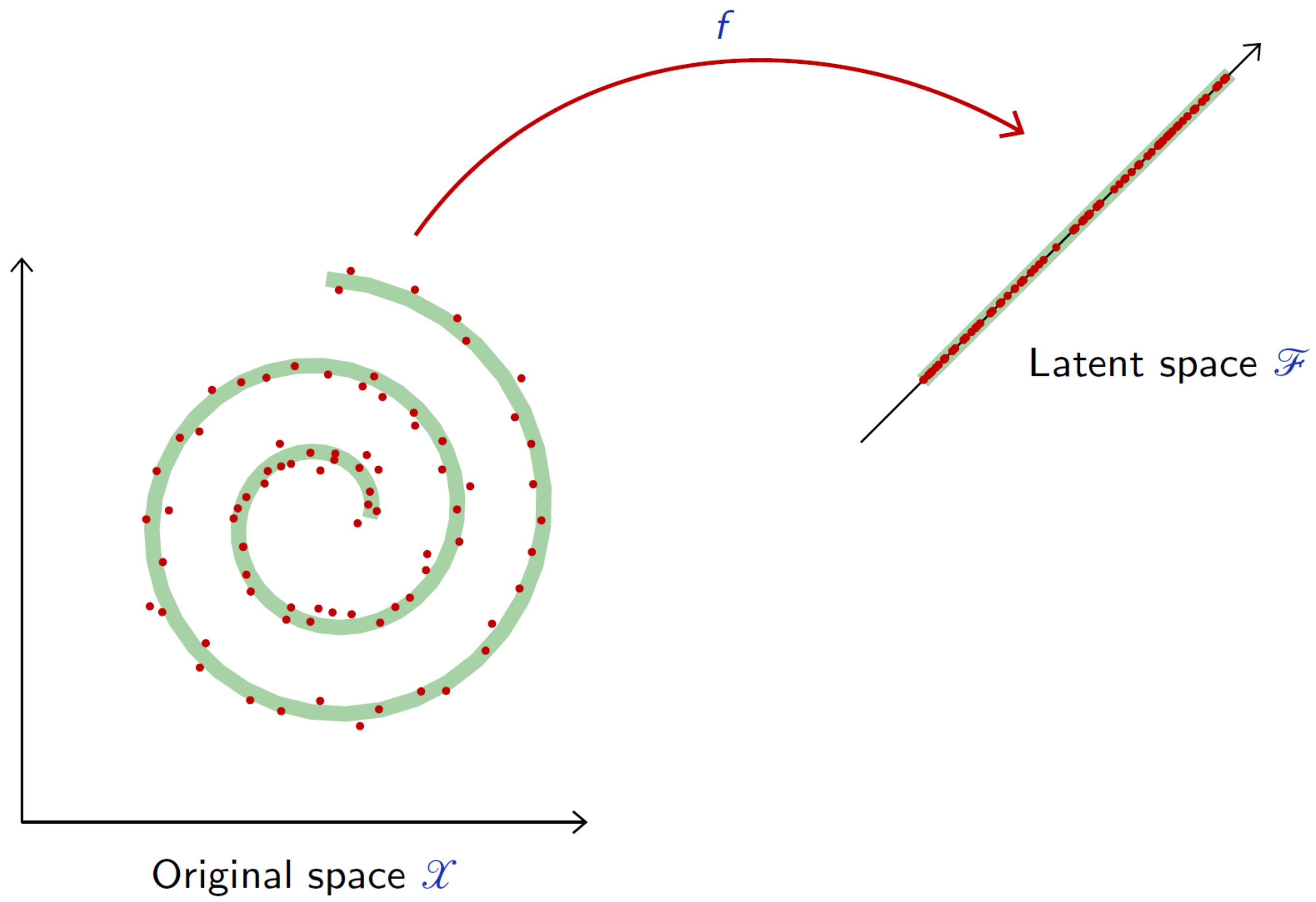
Source: Dr. Francois Fleuret at EPEL

Original Space and Latent Space



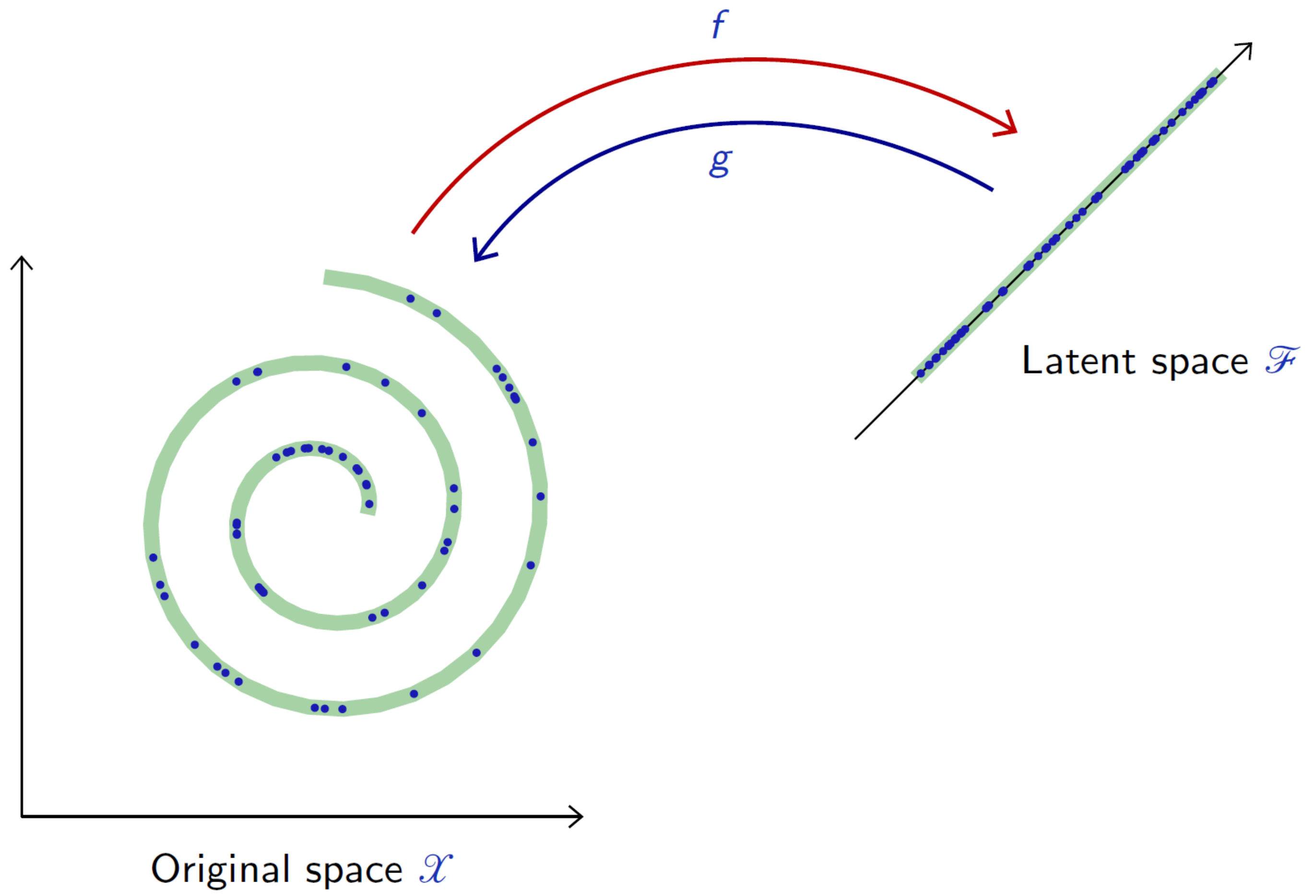
Source: Dr. Francois Fleuret at EPEL

Original Space and Latent Space



Source: Dr. Francois Fleuret at EPEL

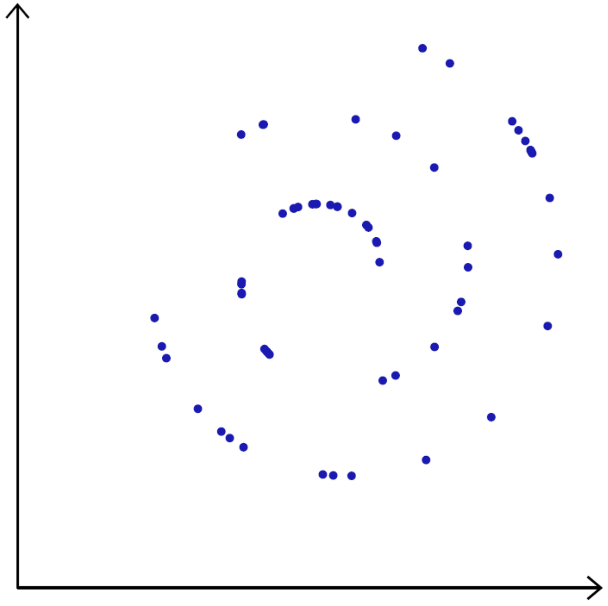
Original Space and Latent Space



Source: Dr. Francois Fleuret at EPEL

Original Space and Latent Space

- We can generate data



Source: Dr. Francois Fleuret at EPEL

Autoencoders

- It is like ‘deep learning version’ of unsupervised learning

- Definition
  - An autoencoder is a neural network that is trained to attempt to copy its input to its output
  - The network consists of two parts: an encoder and a decoder that produce a reconstruction

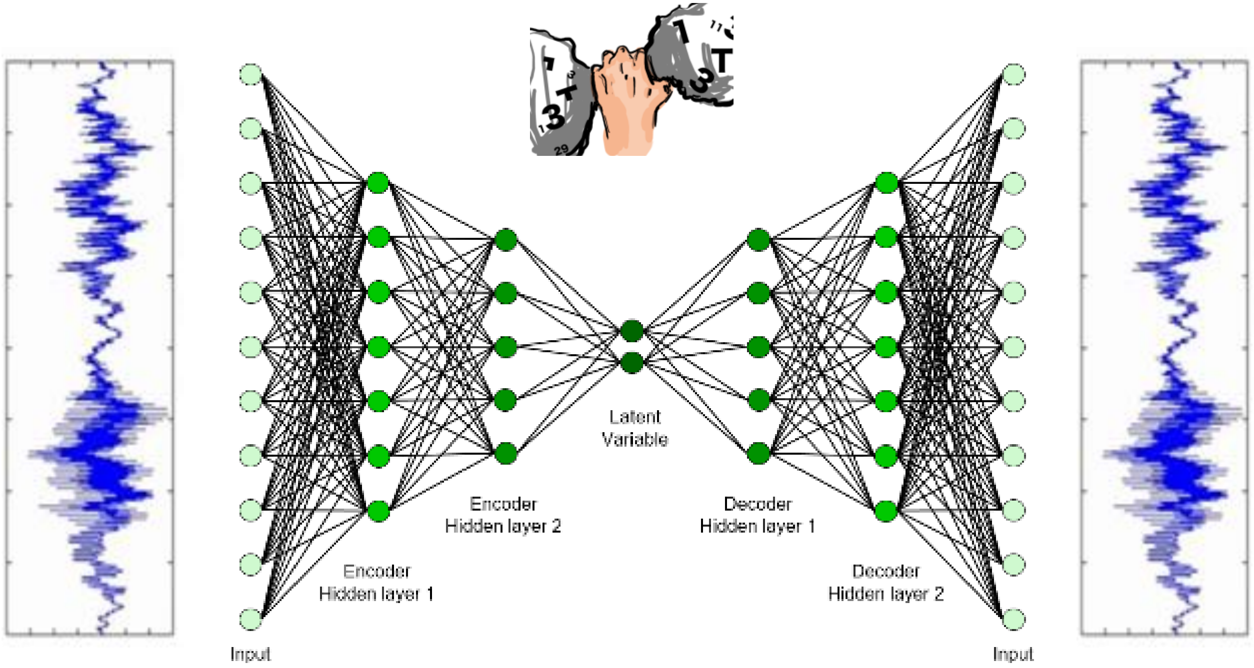
- Encoder and Decoder
  - Encoder function : $z = f(x)$
  - Decoder function : $x = g(z)$
  - We learn to set $g\big(f(x)\big) = x$

Autoencoder

- Dimension reduction
- Recover the input data

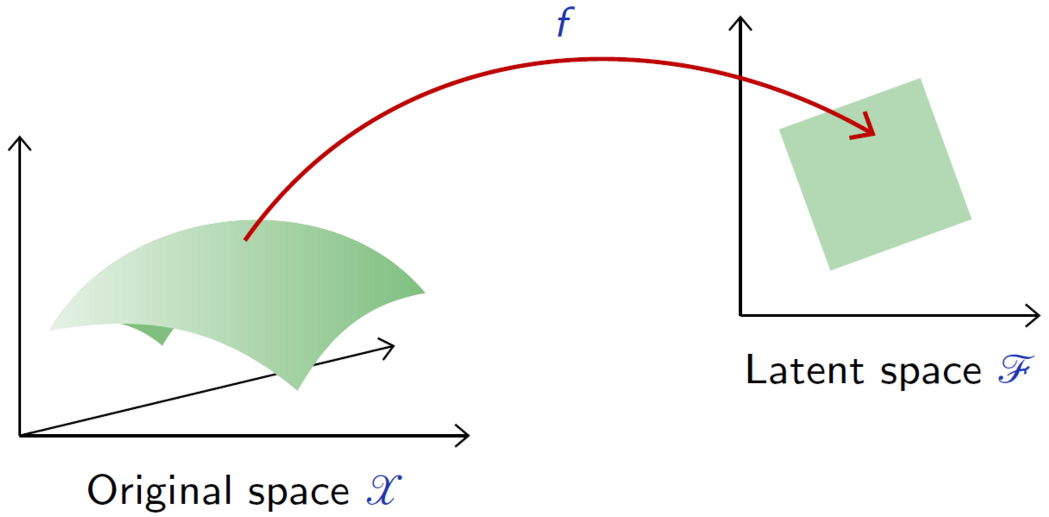
Autoencoder

- Dimension reduction
- Recover the input data
  - Learns an encoding of the inputs so as to recover the original input from the encodings as well as possible



Autoencoder

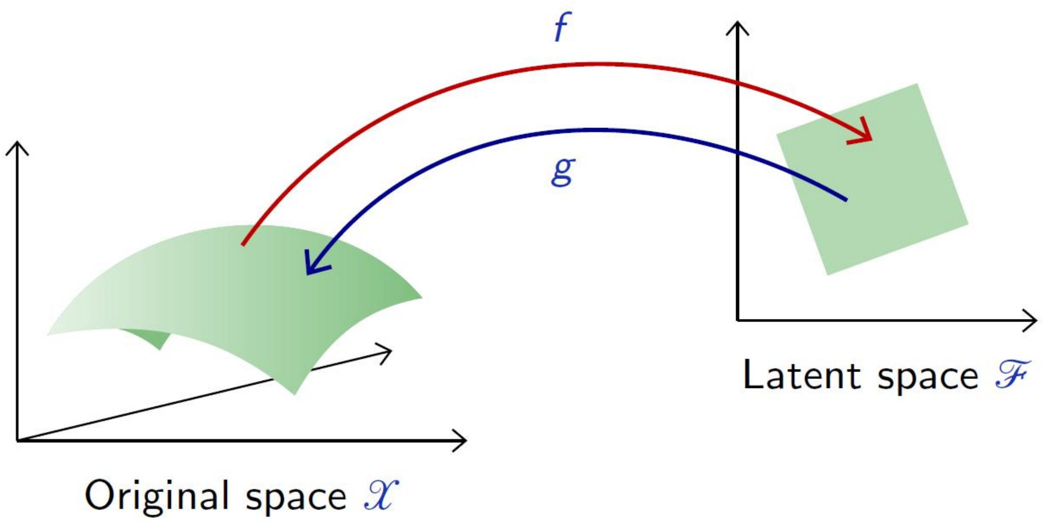
- Autoencoder combines an encoder f from the original space $\mathcal{X}$ to a latent space $\mathcal{F}$, and a decoder g to map back to $\mathcal{X}$, such that $g \circ f$ is [close to] the identity on the data



Source: Dr. Francois Fleuret at EPEL

Autoencoder

- Autoencoder combines an encoder f from the original space $\mathcal{X}$ to a latent space $\mathcal{F}$, and a decoder g to map back to $\mathcal{X}$, such that $g \circ f$ is [close to] the identity on the data



- A proper autoencoder has to capture a "good" parametrization of the signal, and in particular the statistical dependencies between the signal components.

Autoencoder

Let q be the data distribution over $\mathscr{X}$. A good autoencoder could be characterized with the quadratic loss

$$\mathbb{E}_{X \sim q}\left[\|X - g \circ f(X)\|^2\right] \simeq 0.$$

Given two parametrized mappings $f(\cdot\,;w)$ and $g(\cdot\,;w)$, training consists of minimizing an empirical estimate of that loss

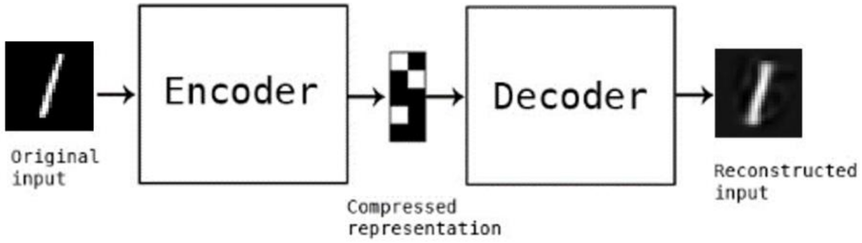
$$\hat{w}_f, \hat{w}_g = \underset{w_f, w_g}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \|x_n - g(f(x_n; w_f); w_g)\|^2 .$$

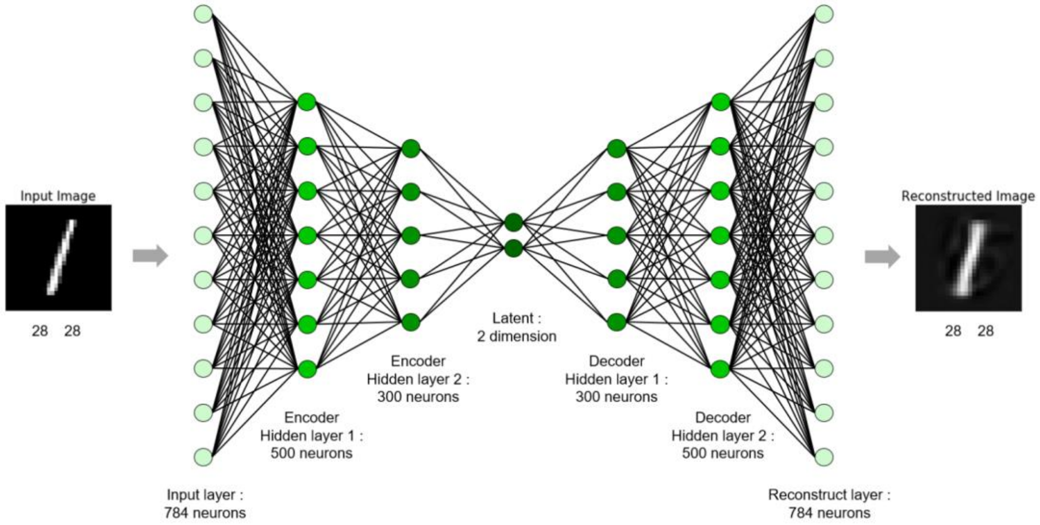
A simple example of such an autoencoder would be with both f and g linear, in which case the optimal solution is given by PCA. Better results can be achieved with more sophisticated classes of mappings, in particular deep architectures.

Autoencoder with MNIST

Autoencoder with TensorFlow

- MNIST example
- Use only (1, 5, 6) digits to visualize in 2-D





Import Libraries and Load MNIST Data

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.neural_network import MLPRegressor
from sklearn.metrics import accuracy_score
```

- Only use (1, 5, 6) digits to visualize latent space in 2-D

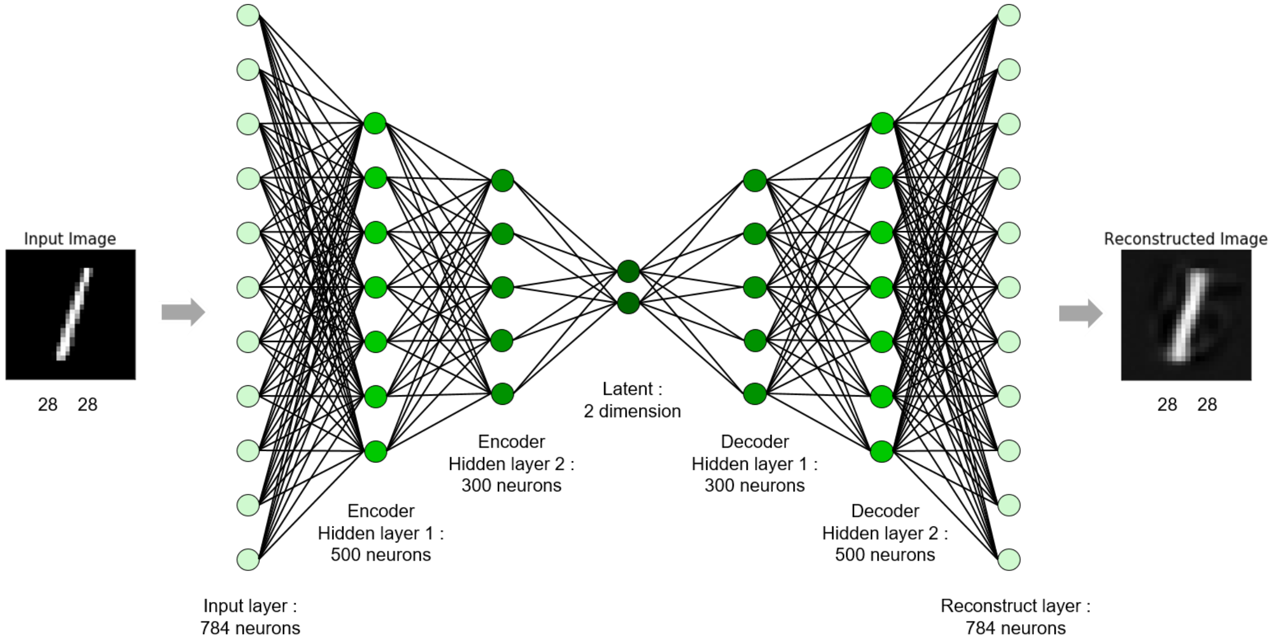
```
train_x = np.load('./data_files/mnist_train_images.npy')
train_y = np.load('./data_files/mnist_train_labels.npy')
test_x = np.load('./data_files/mnist_test_images.npy')
test_y = np.load('./data_files/mnist_test_labels.npy')

n_train = train_x.shape[0]
n_test = test_x.shape[0]

print ("The number of training images : {}, shape : {}".format(n_train, train_x.shape))
print ("The number of testing images : {}, shape : {}".format(n_test, test_x.shape))
```

```
The number of training images : 16583, shape : (16583, 784)
The number of testing images : 2985, shape : (2985, 784)
```

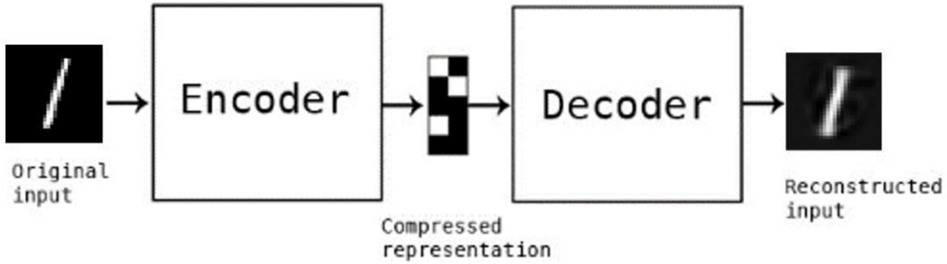
Structure of Autoencoder

- Input shape and latent variable shape
- Encoder shape
- Decoder shape

```python
# Shape of input and latent variable

n_input = 28*28

# Encoder structure
n_encoder1 = 500
n_encoder2 = 300

n_latent = 2

# Decoder structure
n_decoder2 = 300
n_decoder1 = 500
```



Build a Model

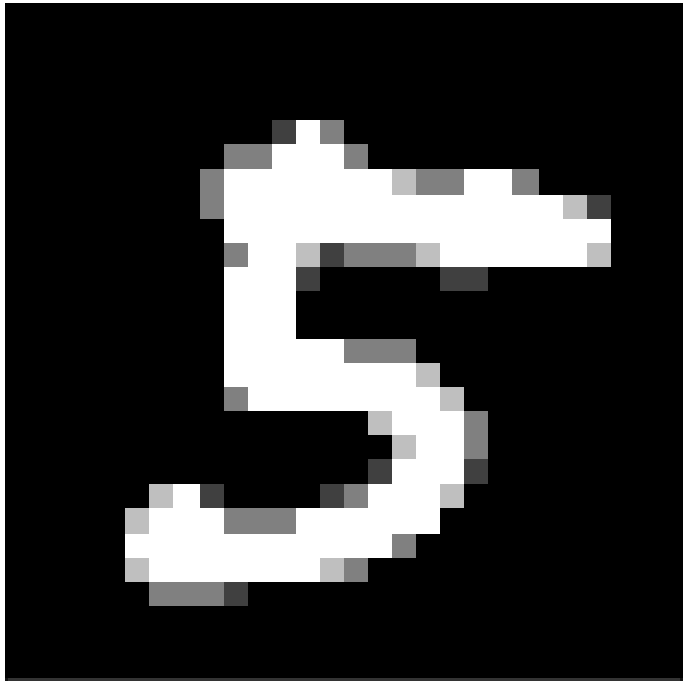



```python
reg = MLPRegressor(hidden_layer_sizes = (n_encoder1, n_encoder2, n_latent, n_decoder2, n_decoder1),
                   activation = 'tanh',
                   solver = 'adam',
                   learning_rate_init = 0.0001,
                   max_iter = 20,
                   tol = 0.0000001,
                   verbose = True)
```

Test or Evaluation

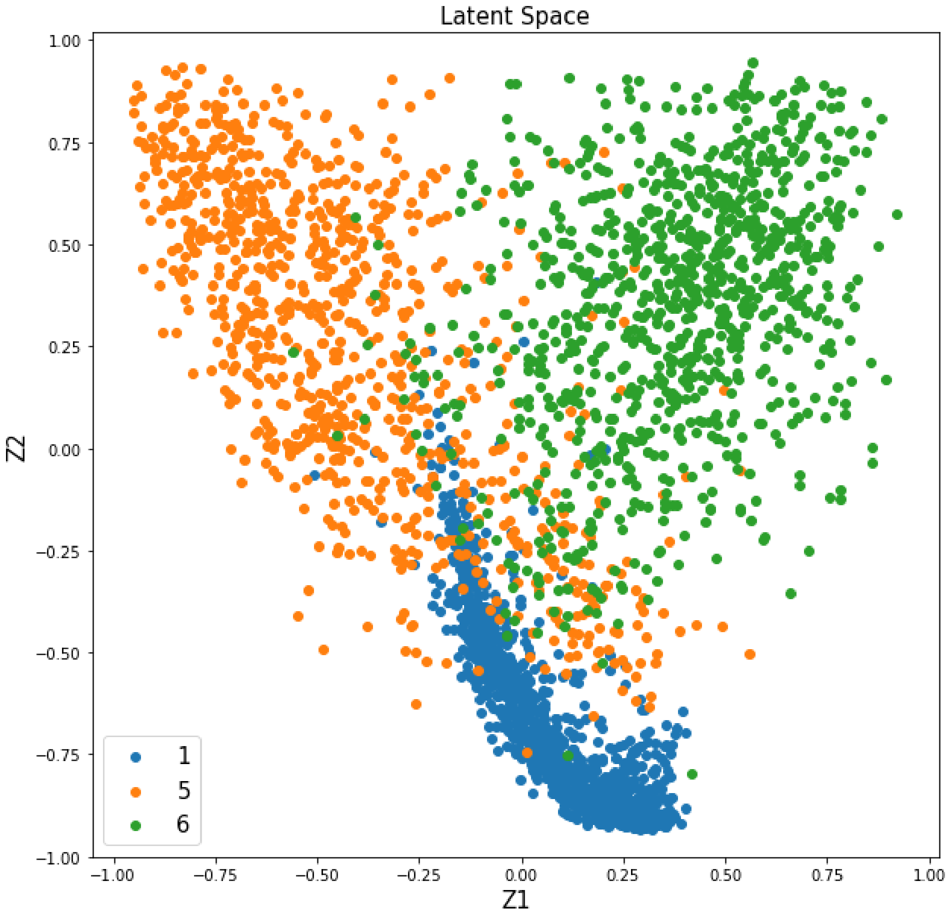
```
idx = np.random.randint(test_x.shape[0])
x_reconst = reg.predict(test_x[idx].reshape(-1,784))
```

Imput Image

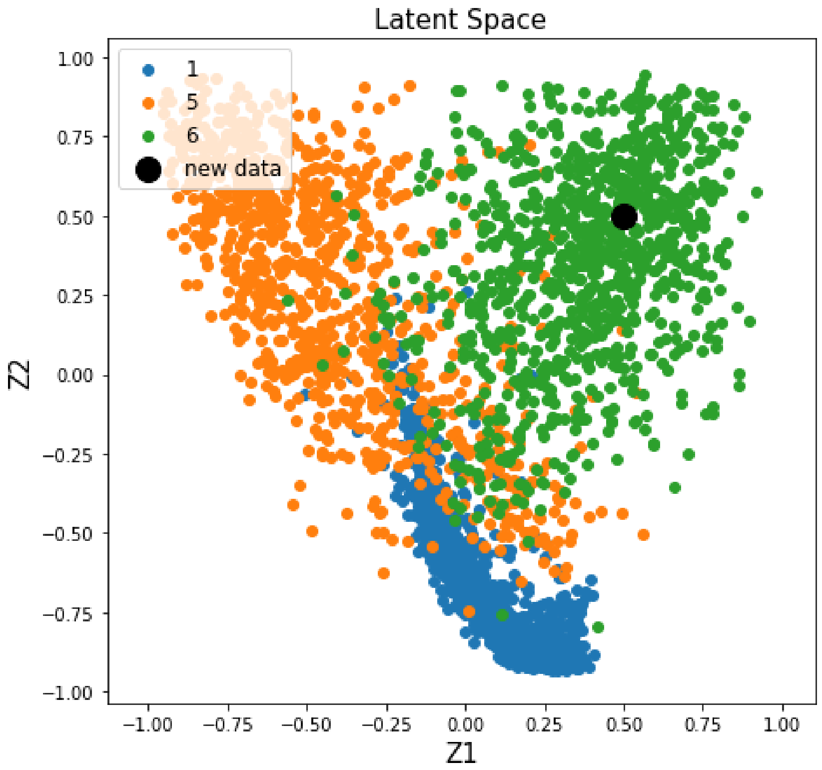


Distribution in Latent Space

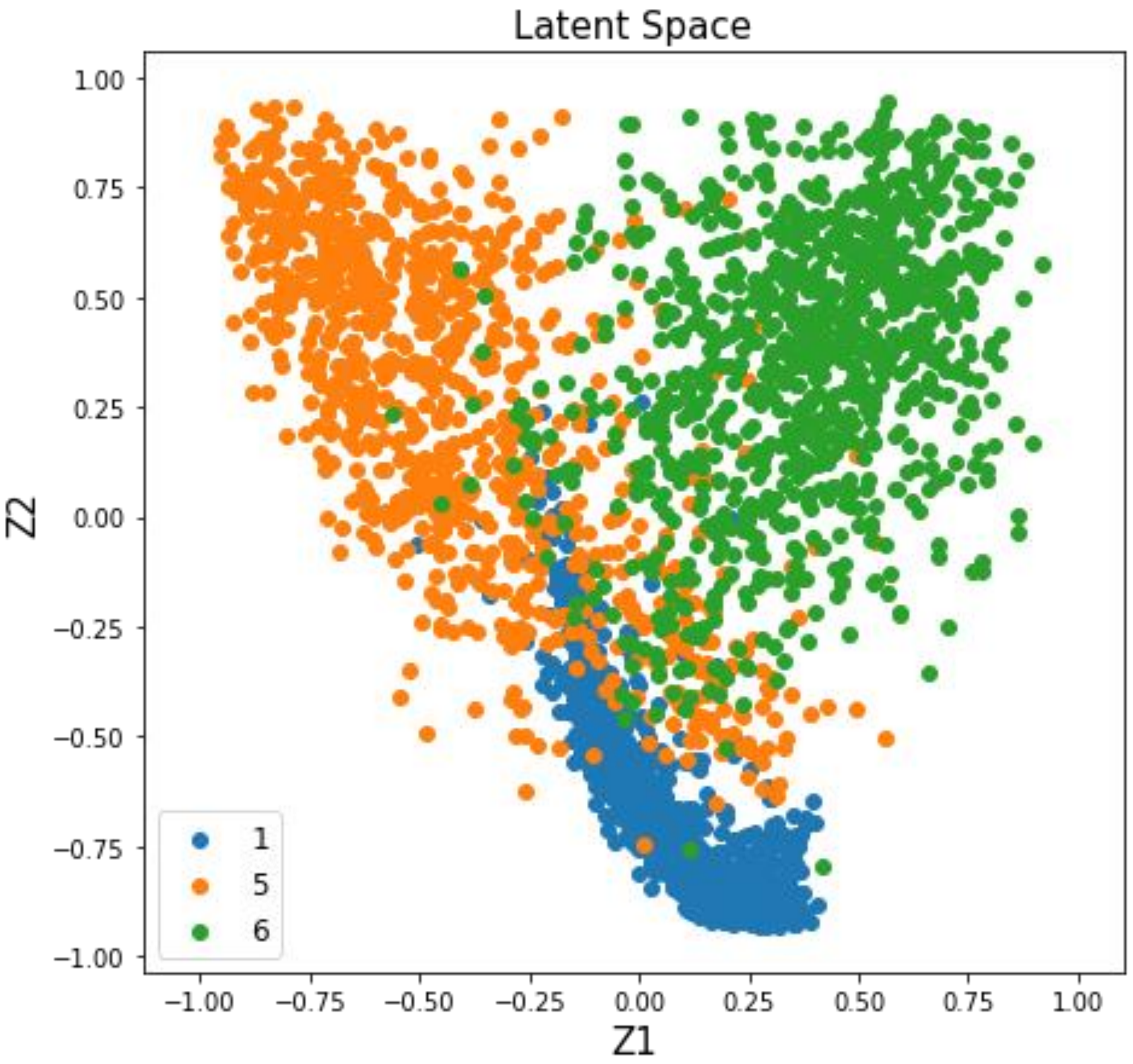
- Make a projection of 784-dim image onto 2-dim latent space

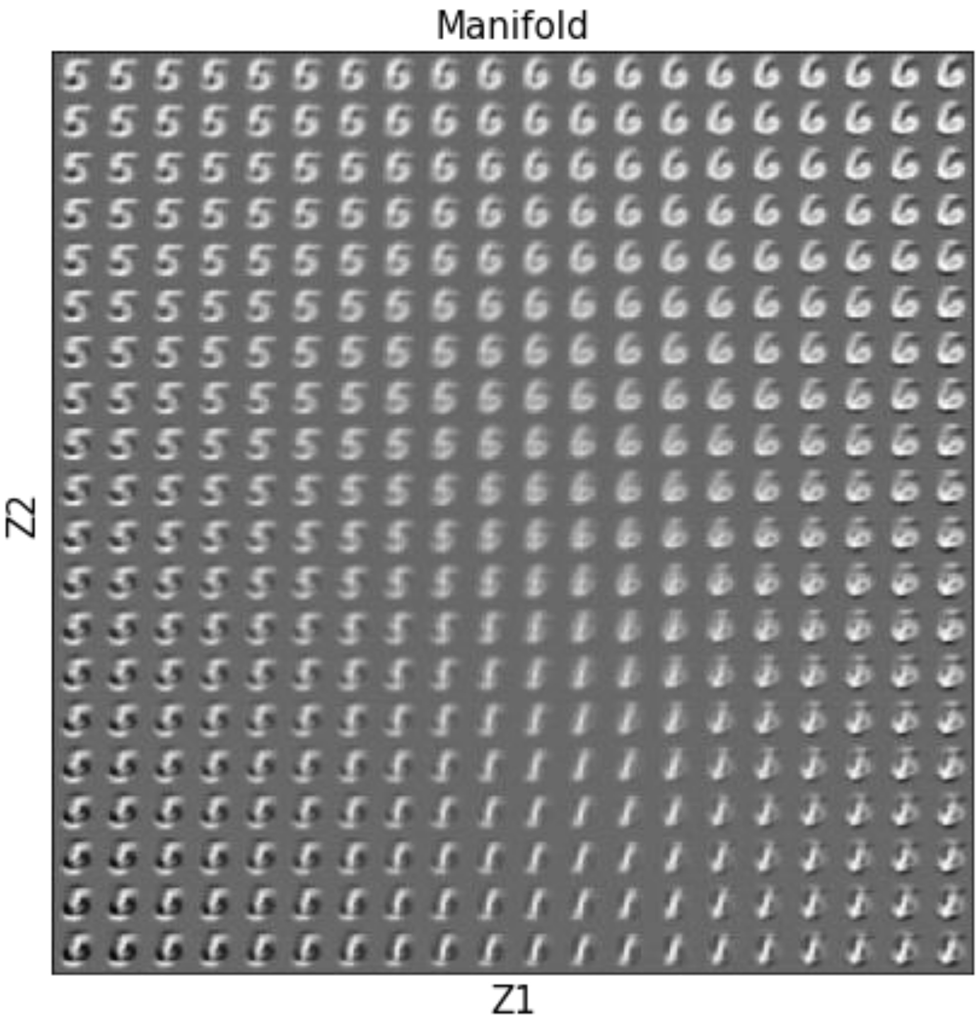


Data Generation



Visualization

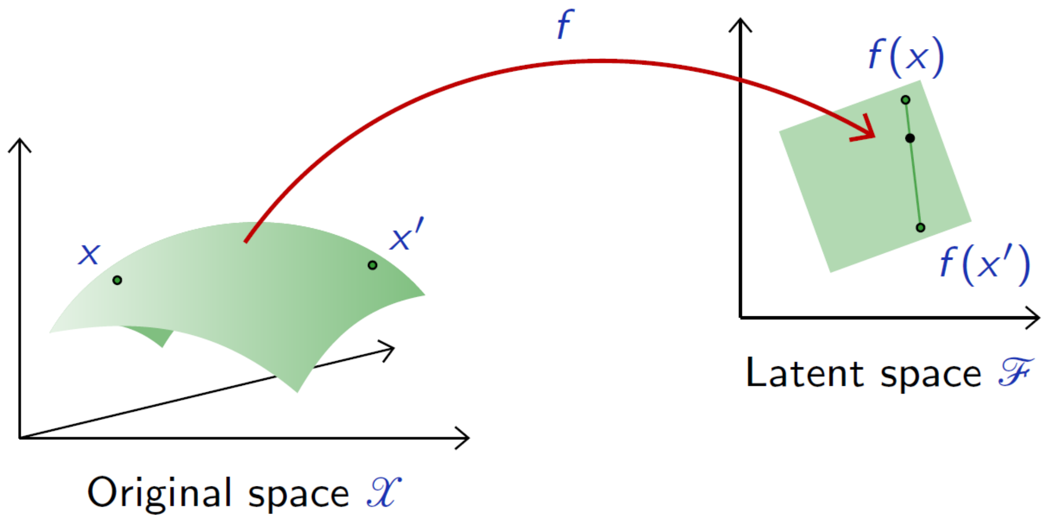




Autoencoder as Generative Model

Latent Representation

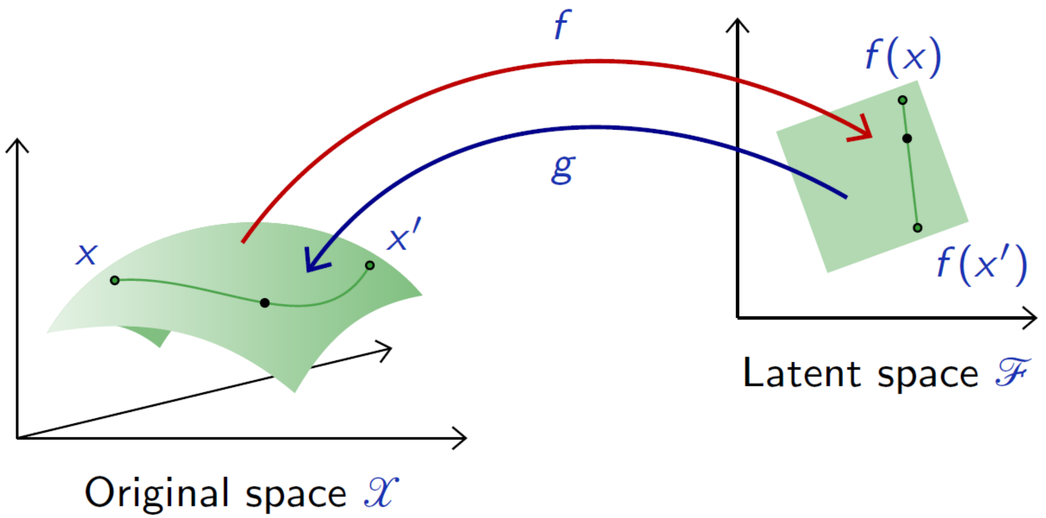
- To get an intuition of the latent representation, we can pick two samples x and x' at random and interpolate samples along the line in the latent space



Latent Representation

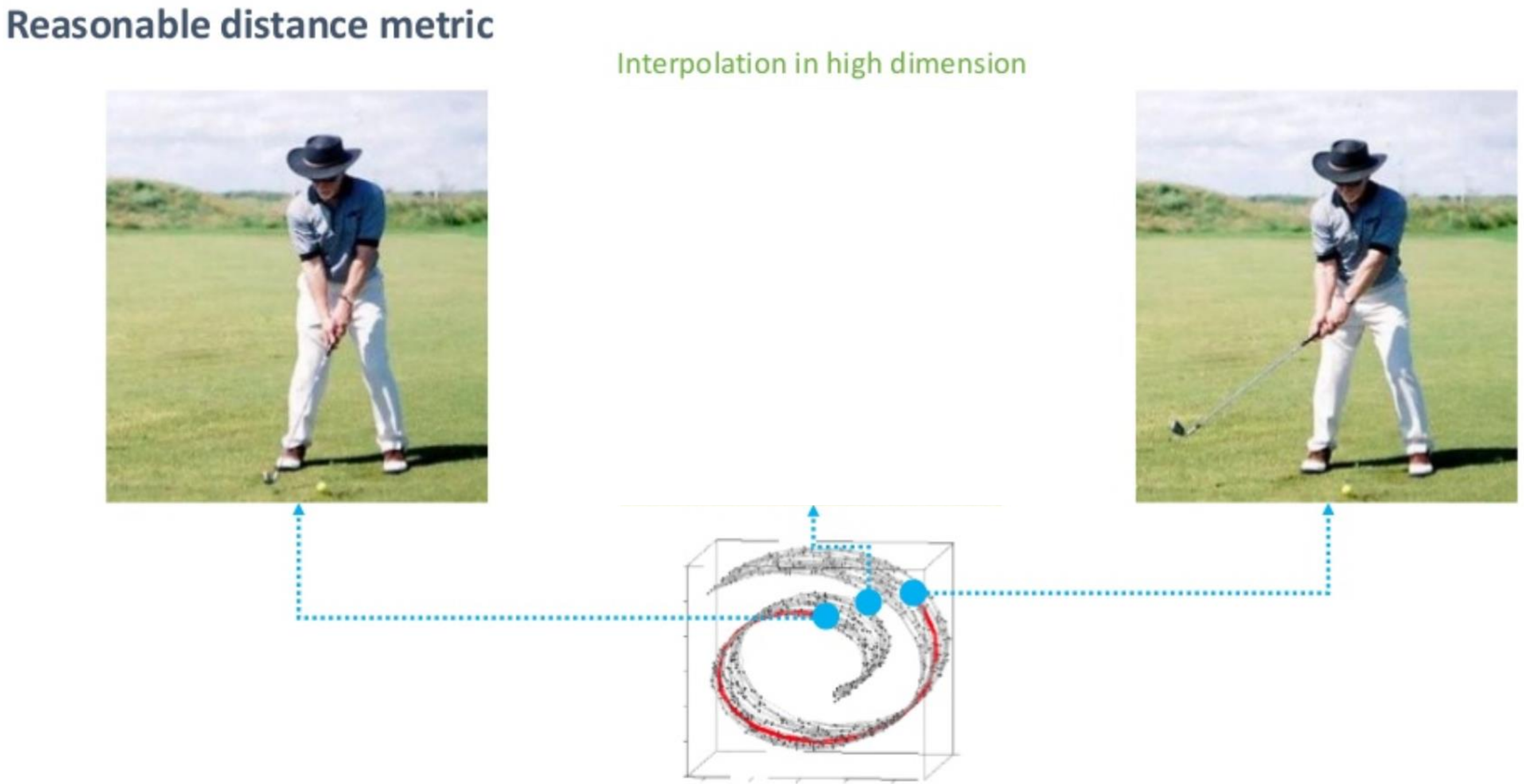
- To get an intuition of the latent representation, we can pick two samples x and x' at random and interpolate samples along the line in the latent space

$$\forall x, x' \in \mathscr{X}^2,\ \alpha \in [0,1],\quad \xi(x, x', \alpha) = g((1-\alpha)f(x) + \alpha f(x')).$$



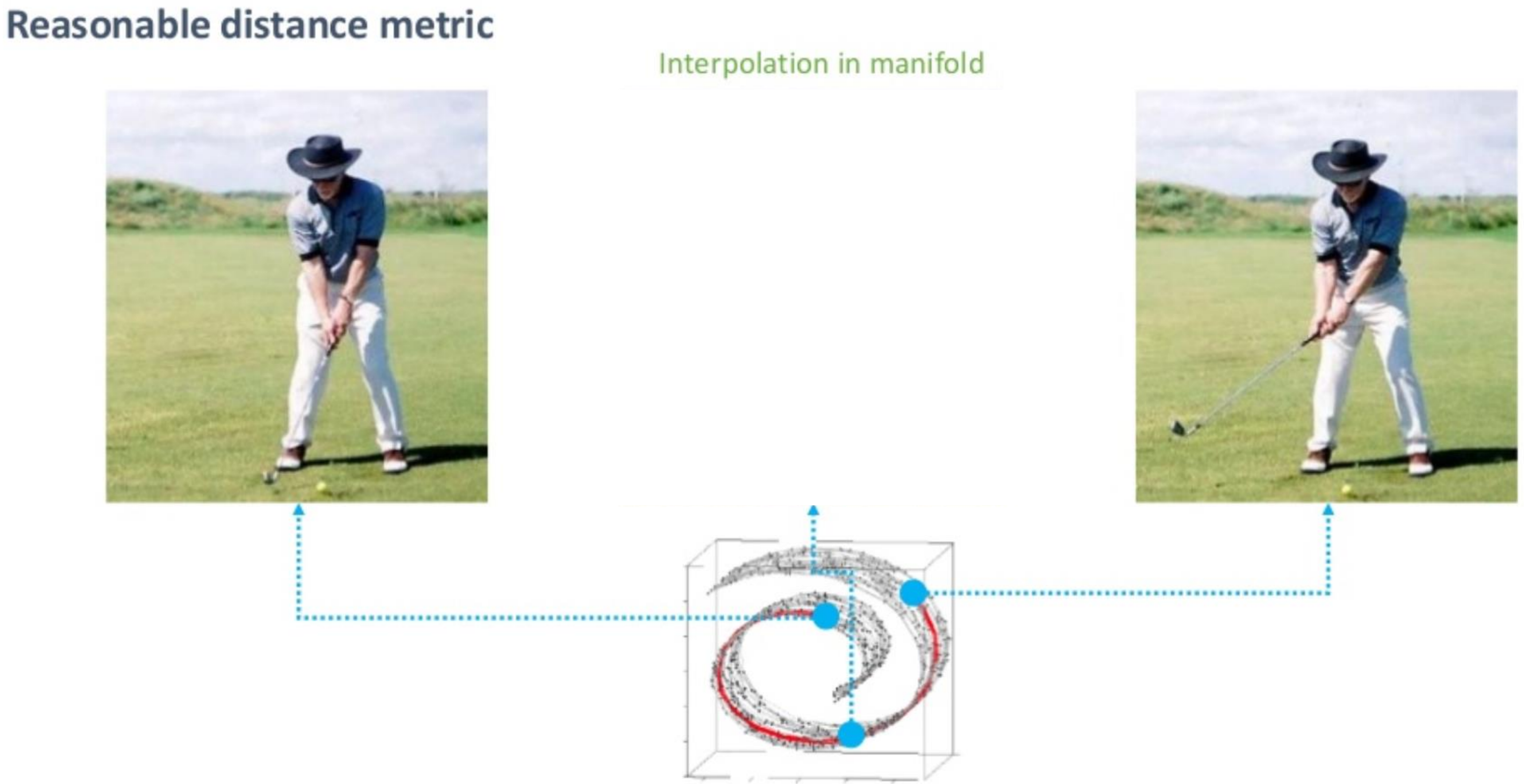
Source: Dr. Francois Fleuret at EPEL

Interpolation in High Dimension



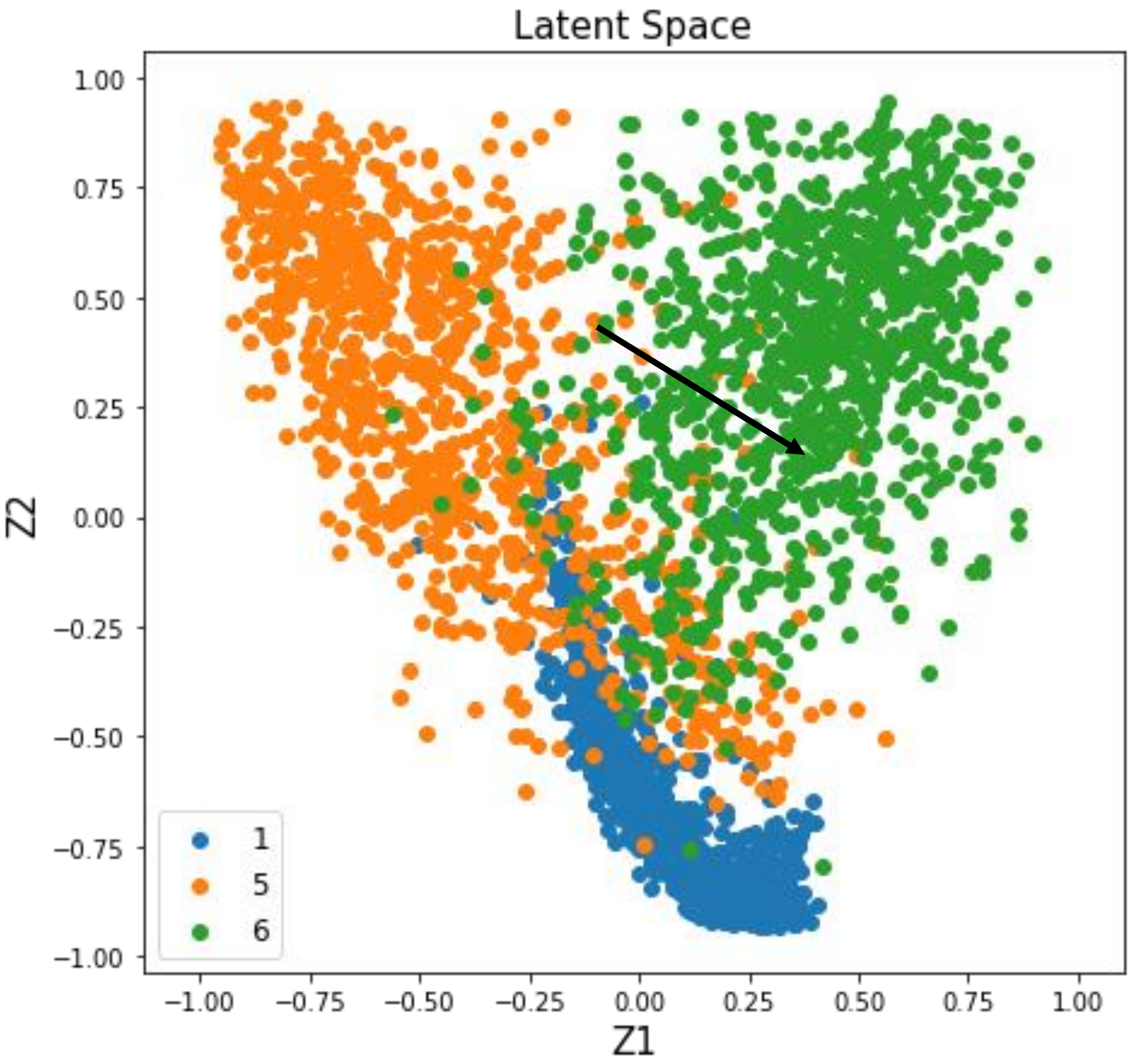
https://www.cs.cmu.edu/~efros/courses/AP06/presentations/ThompsonDimensionalityReduction.pdf

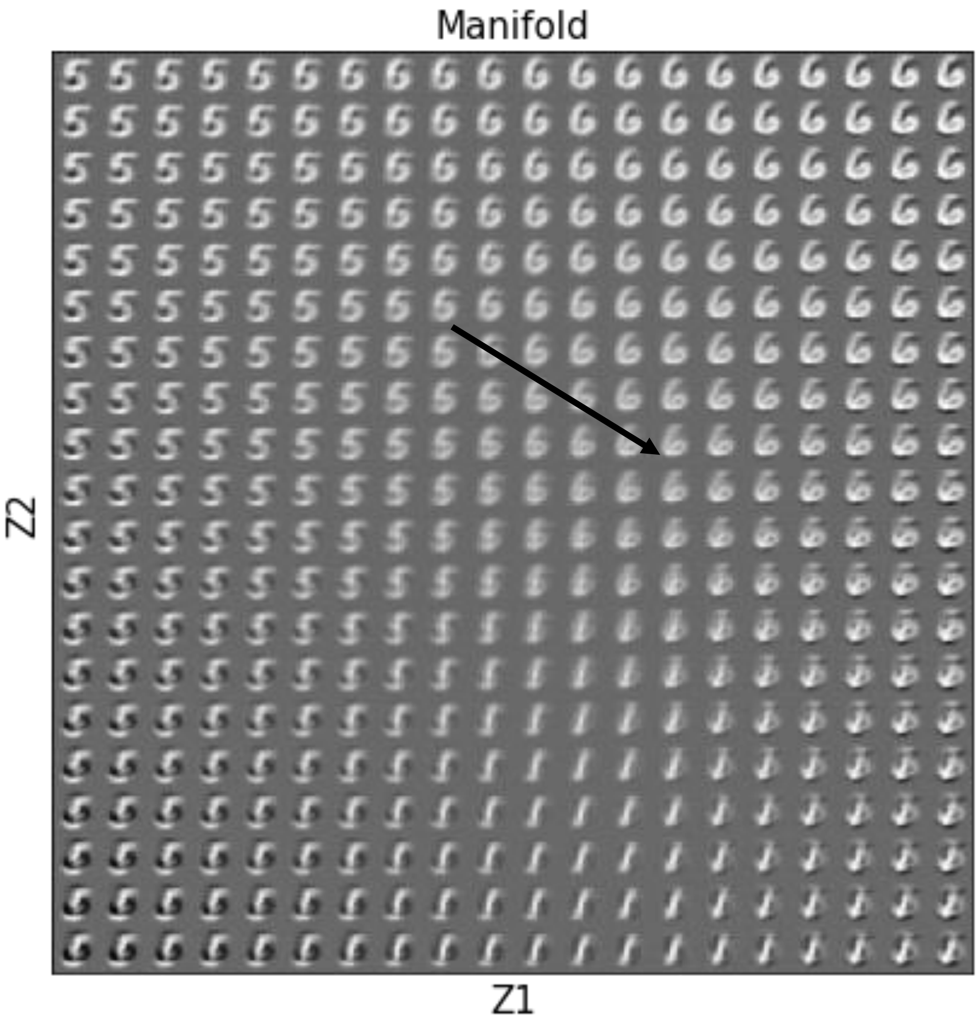
Interpolation in Manifold



https://www.cs.cmu.edu/~efros/courses/AP06/presentations/ThompsonDimensionalityReduction.pdf

MNIST Example: Walk in the Latent Space





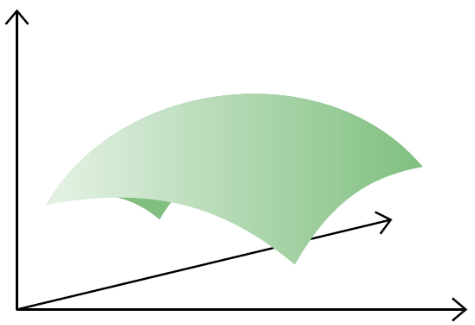
Generative Capabilities

- We can assess the generative capabilities of the decoder g by introducing a [simple] density model q^Z over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with g.

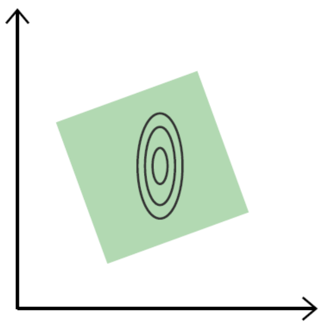
We can for instance use a Gaussian model with diagonal covariance matrix.

$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.



Original space $\mathscr{X}$



Latent space $\mathscr{F}$

Source: Dr. Francois Fleuret at EPEL

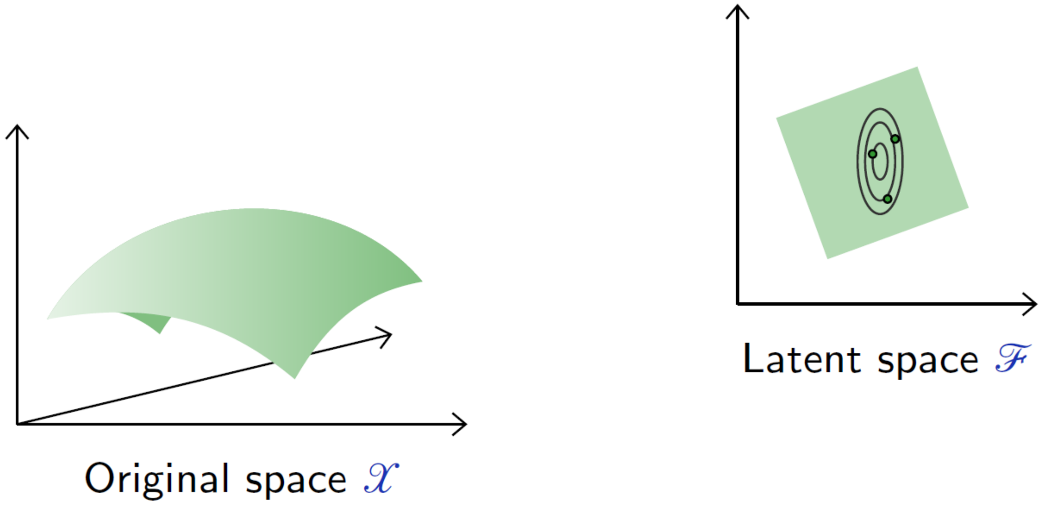
Generative Capabilities

- We can assess the generative capabilities of the decoder g by introducing a [simple] density model q^Z over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with g.

We can for instance use a Gaussian model with diagonal covariance matrix.

$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.



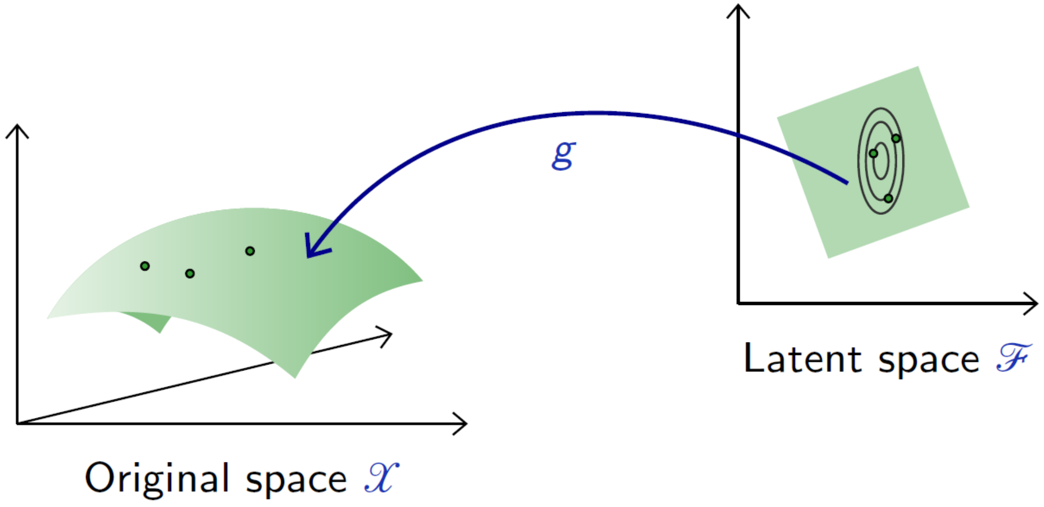
Generative Capabilities

- We can assess the generative capabilities of the decoder g by introducing a [simple] density model q^Z over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with g.

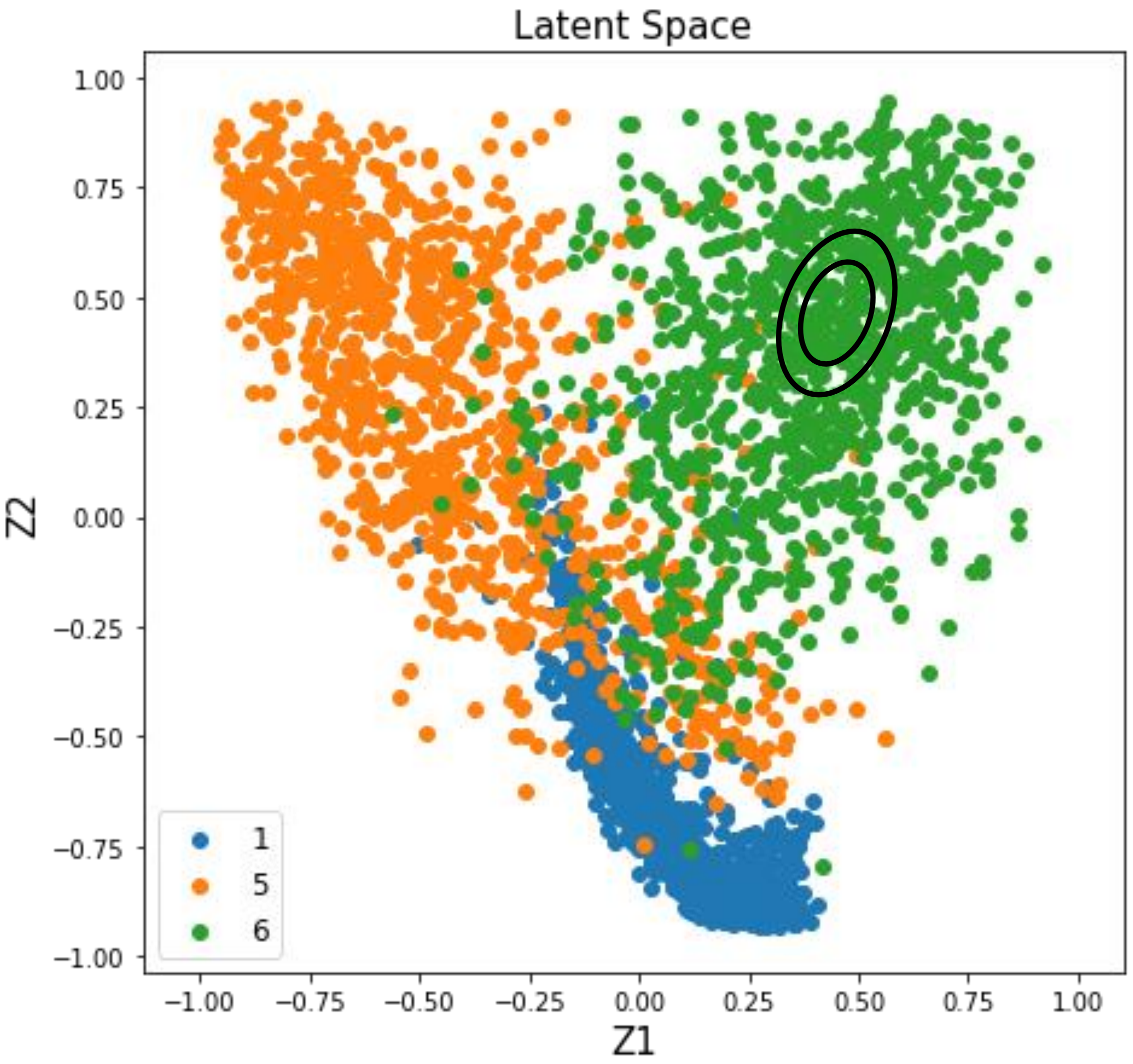
We can for instance use a Gaussian model with diagonal covariance matrix.

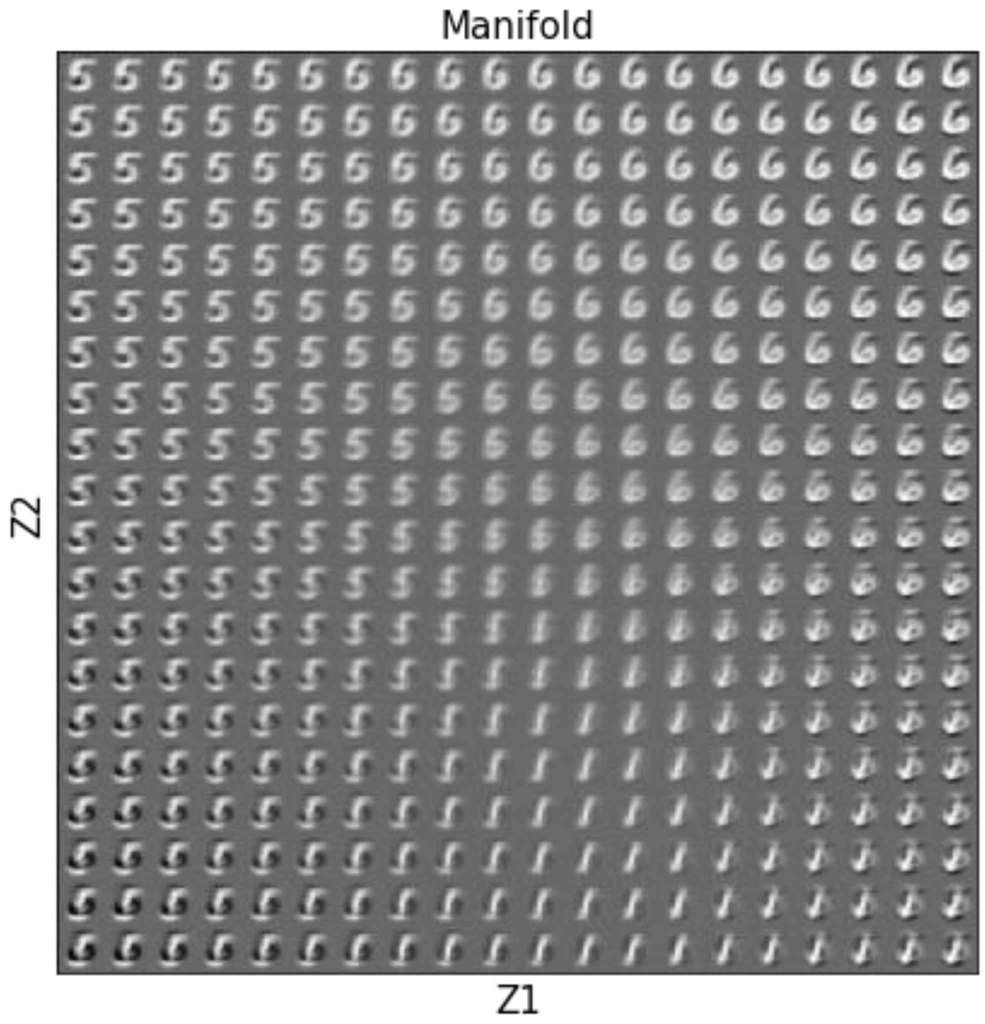
$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.



MNIST Example





Generative Models

- It generates something that makes sense.
- These results are unsatisfying, because the density model used on the latent space $\mathcal{F}$ is too simple and inadequate.
- Building a “good” model amounts to our original problem of modeling an empirical distribution, although it may now be in a lower dimension space.
- This is a motivation to VAE or GAN.