# 11 - Autoencoders

**Ludwig Krippahl**

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

## Summary

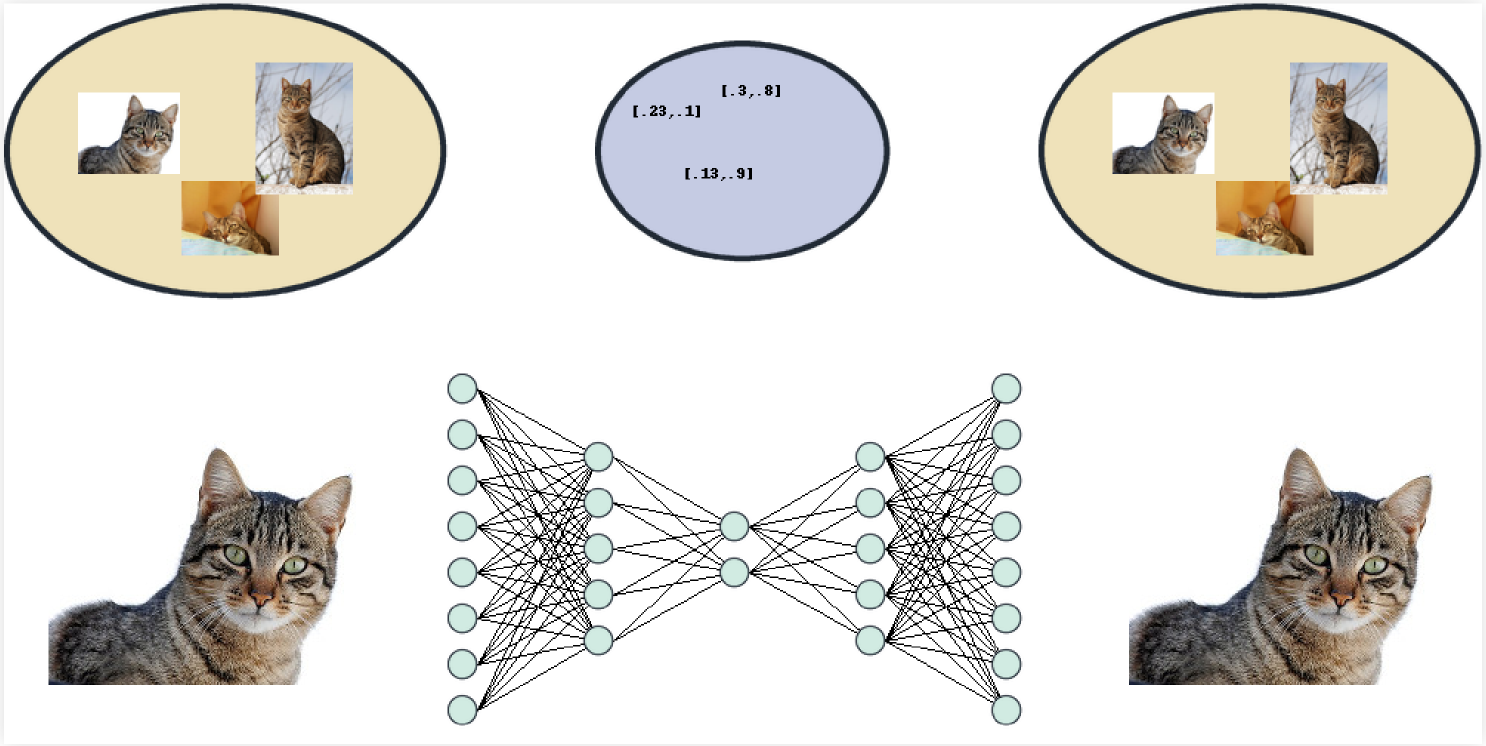
- What are Autoencoders?
- Different restrictions on encoding
  - Undercompleteness
  - Regularization
  - Sparsity
  - Noise reconstruction
- Convolutional Autoencoders
- Applications
  - Dimensionality reduction
  - Manifold learning
  - Anomaly detection

# What are autoencoders?

# What are autoencoders?

## Network trained to output the input (unsupervised)

- In the hidden layers, one layer learns a **code** describing the input

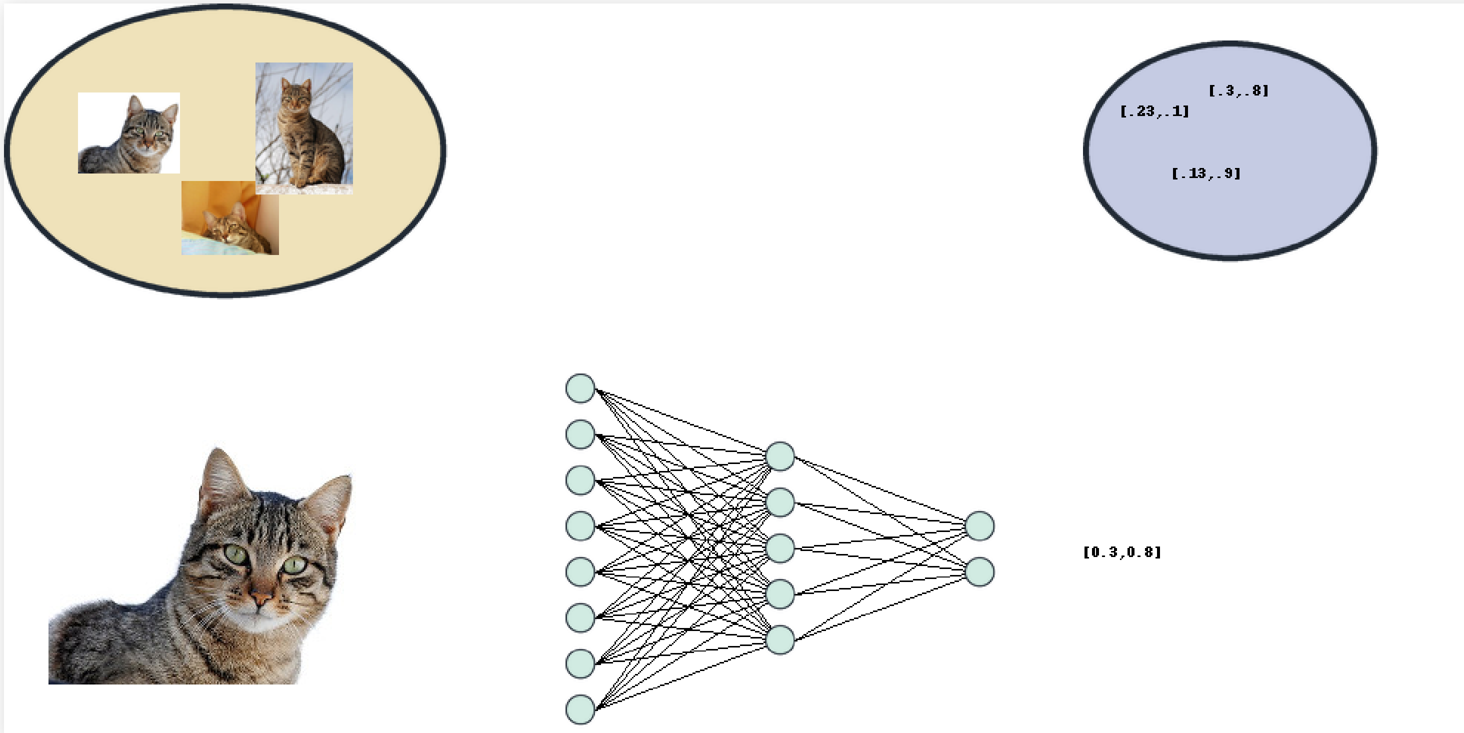



Cat images: Joaquim Alves Gaspar CC-SA

# What are autoencoders?

## Network trained to output the input (unsupervised)

- The encoder maps from input to **latent space**

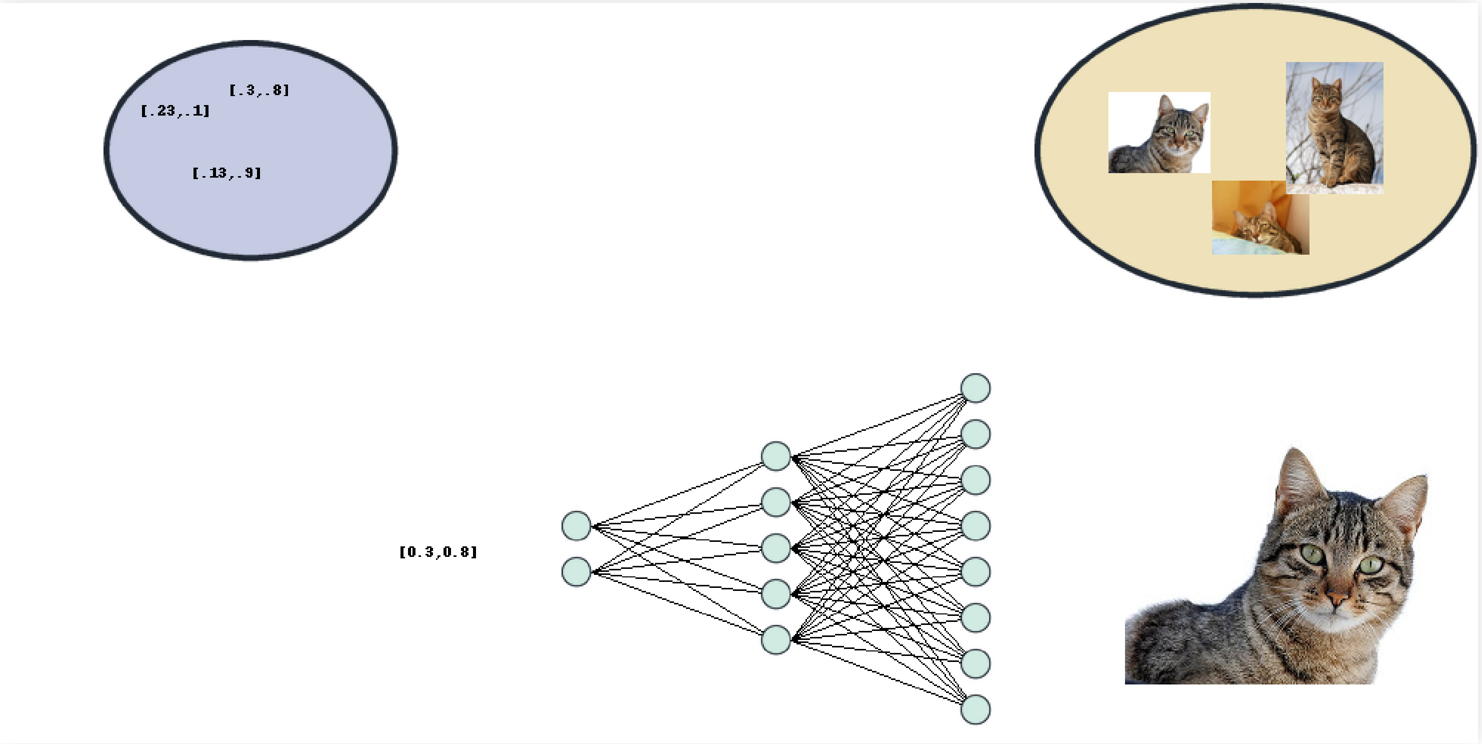



Cat images: Joaquim Alves Gaspar CC-SA

# What are autoencoders?

## Network trained to output the input (unsupervised)

- The decoder maps from **latent space** back to input space

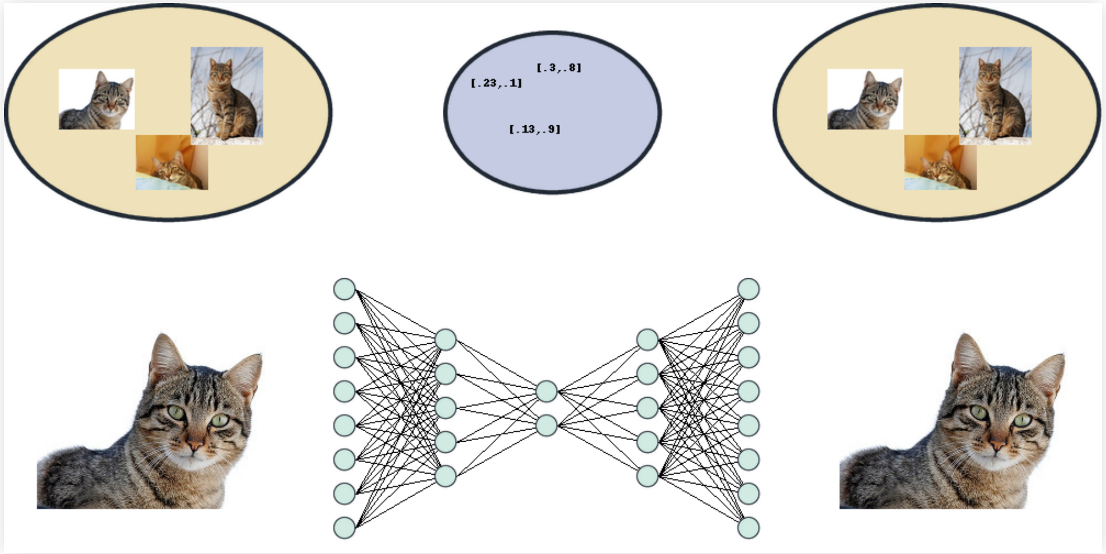



Cat images: Joaquim Alves Gaspar CC-SA

# What are autoencoders?

**Network trained to output the input (unsupervised)**

- Encoder, $h = f(x)$, and decoder, $x = g(h)$
- No need for labels, since the target is the input
- Why learn $x = g\left(f\left(x\right)\right)$?




Cat images: Joaquim Alves Gaspar CC-SA

# What are autoencoders?

## Network trained to output the input (unsupervised)

- Encoder, $h = f(x)$, and decoder, $x = g(h)$
- Why learn $x = g\left(f\left(x\right)\right)$?
- Latent representation can have advantages
  - Lower dimension
  - Capture structure in the data
  - Data compression
- As long as we can force the autoencoder to do something useful

# What are autoencoders?

**Network trained to output the input (unsupervised)**

- Encoder, $h = f(x)$, and decoder, $x = g(h)$
- Autoencoders are (usually) feedforward networks
  - Can be trained with backpropagation
- But since the target is $x$, they are unsupervised learners
- Need some "bottleneck" to force a useful representation
  - Otherwise just copies values
- Why a useful representation?
  - Captures the structure of the data
  - Provides good features
  - We'll see more of this in the next lecture

# Different types of autoencoders

# Undercomplete Autoencoders

**Autoencoder is undercomplete if $h$ is smaller than $x$**

- Forces the network to learn reduced representation of input
- Trained by minimizing a loss function

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

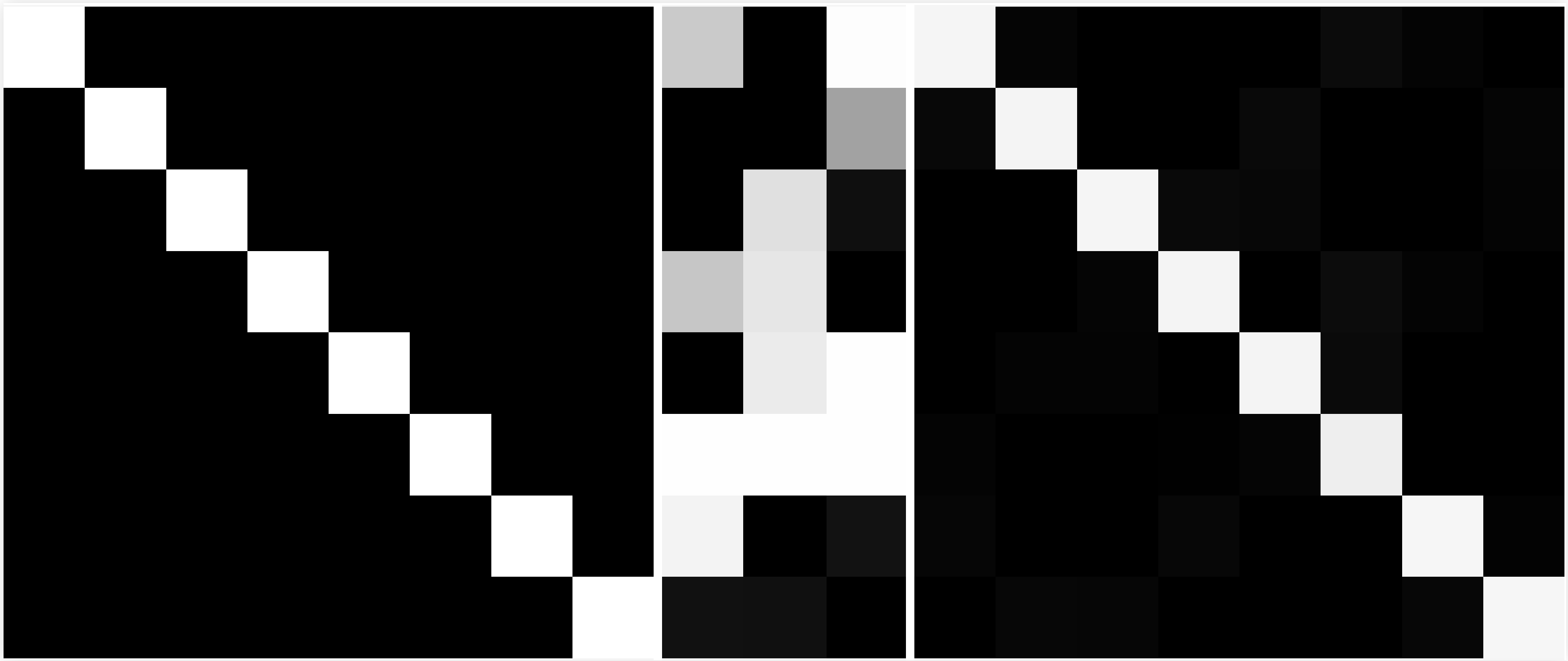
  that penalizes the difference between $x$ and $g(f(x))$
- If linear it is similar to PCA (without orthogonality constraint)
- With nonlinear transformations, an undercomplete autoencoder can learn more powerful representations
- However, we cannot overdo it
  - With too much power, autoencoder can just index each training example and learn nothing useful:

$$f(x_i) = i, \quad g(i) = x_i$$

# Undercomplete Autoencoders

**Autoencoder is undercomplete if $h$ is smaller than $x$**

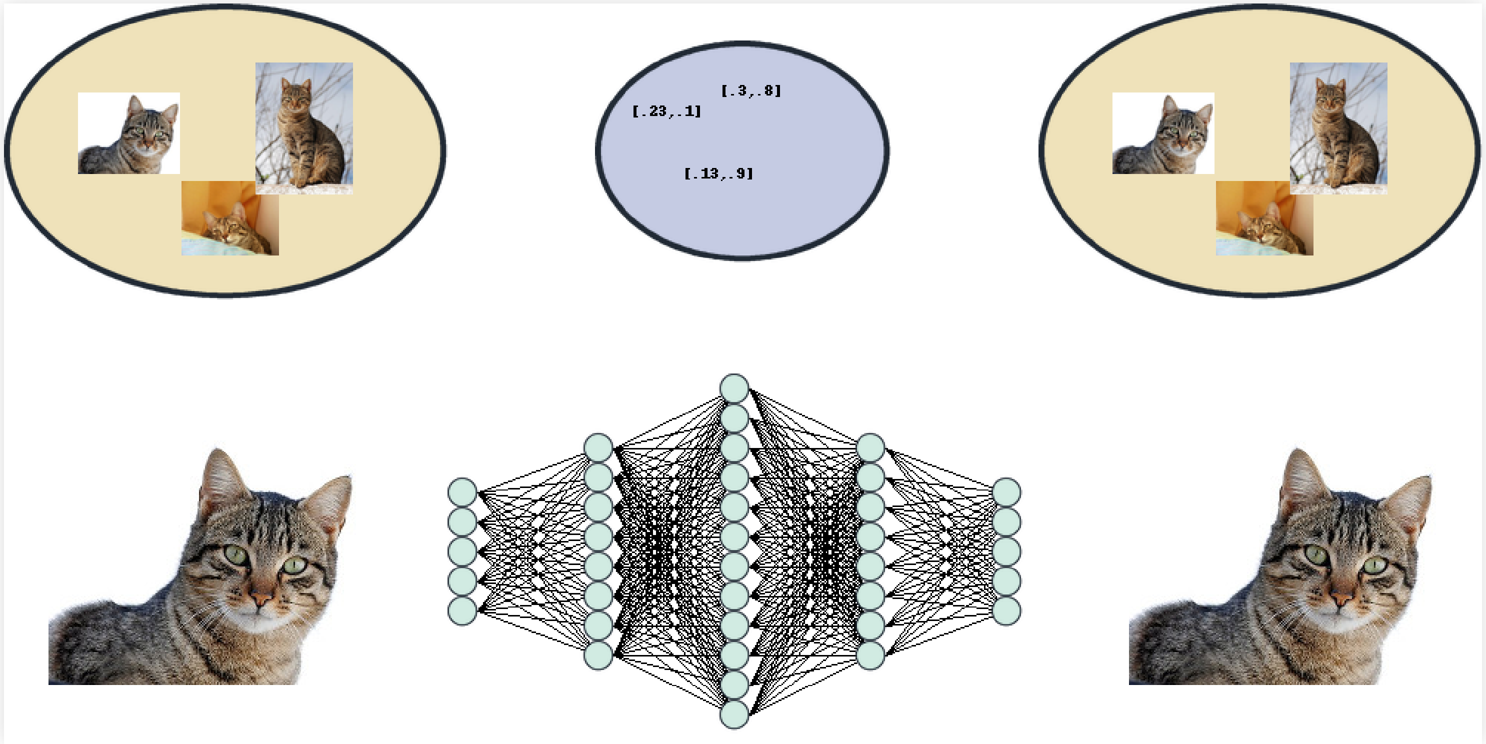
- Mitchell's autoencoder, hidden layer of 3 neurons



# Regularized Autoencoders

## An overcomplete autoencoder has $h$ larger than $x$

- This, by itself, is a bad idea as $h$ will not represent anything useful




Cat images: Joaquim Alves Gaspar CC-SA

# Regularized Autoencoders

**An overcomplete autoencoder has $h$ larger than $x$**

- But we can restrict $h$ with regularization
- This way the autoencoder also learns how restricted $h$ should be

## Sparse Autoencoder

- Force $h$ to have few activations
- Example: we want the probability of $h_i$ firing

$$\hat{p}_i = \frac{1}{m}\sum_{j=1}^{m} h_i(x_j)$$

to be equal to $p$ (the sparseness parameter)

- We can use the Kullback-Leibler divergence between Bernoulli variables as a regularization penalty

$$L(x, g(f(x))) + \lambda \sum_{i}^{n} \left( p \log \frac{p}{\hat{p}_i} + (1-p) \log \frac{1-p}{1-\hat{p}_i} \right)$$

- Note: sparseness may be due to small, nonzero, activations
  - But it can correspond to zero activation in neurons if using ReLU

# Regularized Autoencoders

## Sparse Autoencoder

- We can think of the distribution of $h$ as a prior assumption
- For example, the Laplace prior:

$$p(h_i) = \frac{\lambda}{2} e^{-\lambda |h_i|}$$

- The Laplace prior results in absolute value regularization penalty

$$L(x, g(f(x))) + \lambda \sum_i |h_i|$$

- This is analogous to the $L_1$ regularization but applied to the activation of the neurons in the coding layer
- Just like $L_2$ regularization comes from a Gaussian prior
  - (e.g. minimizing quadratic error)

## Sparse Autoencoder

- Sparse autoencoders make neurons specialize

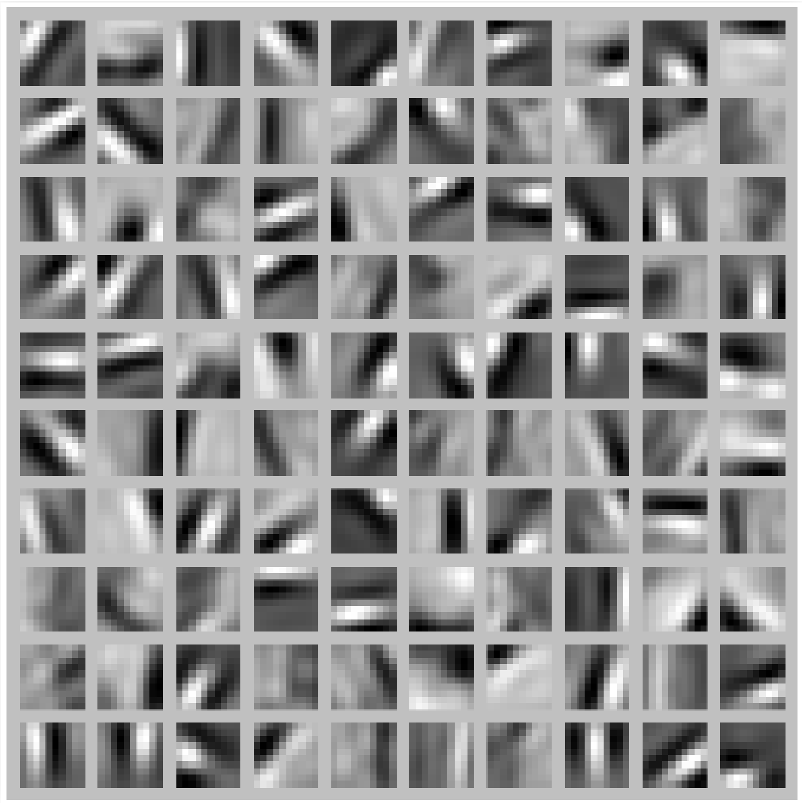


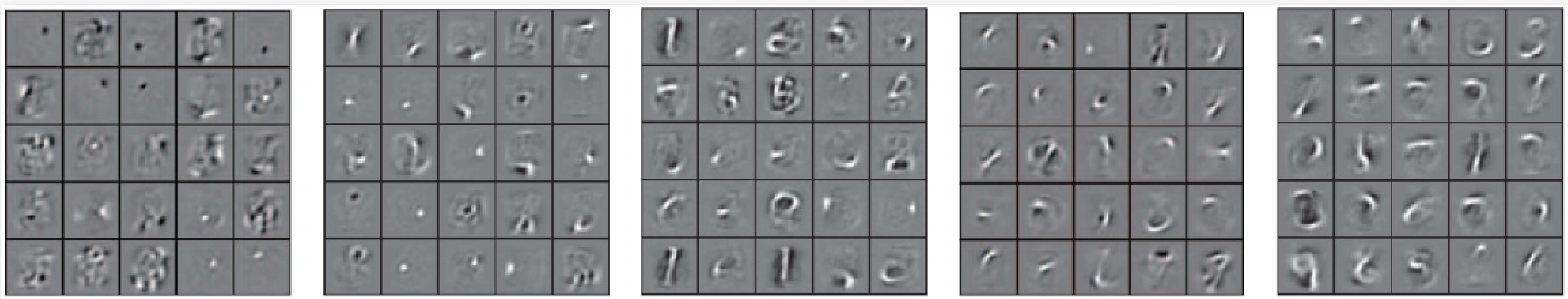
Image: Andrew Ng

- Trained on 10x10 images
- 100 neurons on $h$
- Images (norm-bounded) that maximize activation

# Regularized Autoencoders

## Sparse Autoencoder

- Sparse autoencoders trained on MNIST, different sparsity penalties
- (25 neurons in filter, images correspond to highest activation)



(a) No penalty (b) L1 norm penalty (c) L2 norm penalty (d) Student-t penalty (e) KL-divergence penalty

Niang et. al, Empirical Analysis of Different Sparse Penalties... IJCNN 2015,

## Denoising Autoencoders

- We can force $h$ to be learned with noisy inputs
- Output the original $x$ from corrupted $\tilde{x}$: $L(x, g(f(\tilde{x})))$

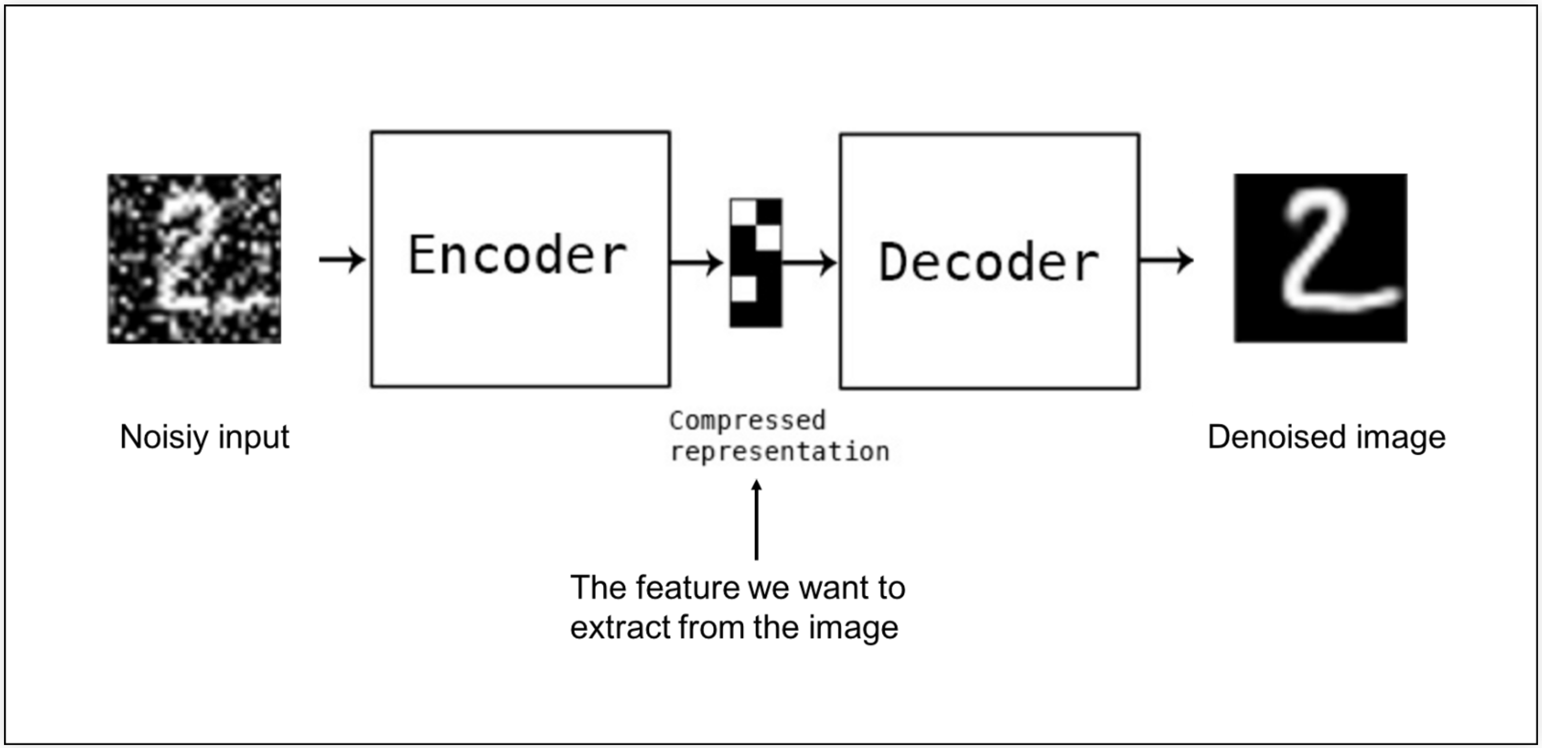



Image: Adil Baaj, Keras Tutorial on DAE

# Regularized Autoencoders

## Denoising Autoencoders

- We can force $h$ to be learned with noisy inputs
  - Output the original $x$ from corrupted $\tilde{x}$: $L(x, g(f(\tilde{x})))$
- This forces the autoencoder to remove the noise by learning the underlying distribution of $x$
- Algorithm:
  - Sample $x_i$ from $\mathcal{X}$
  - Apply corruption $C(\tilde{x}_i \mid x_i)$
  - Train with $(x, \tilde{x})$, minimizing $-\log p_{decoder}(x \mid h)$

## Contractive Autoencoder

- Penalize the derivatives of activation w.r.t. inputs

$$L(x, g(f(x))) + \lambda \sum_i ||\nabla_x h_i||^2$$

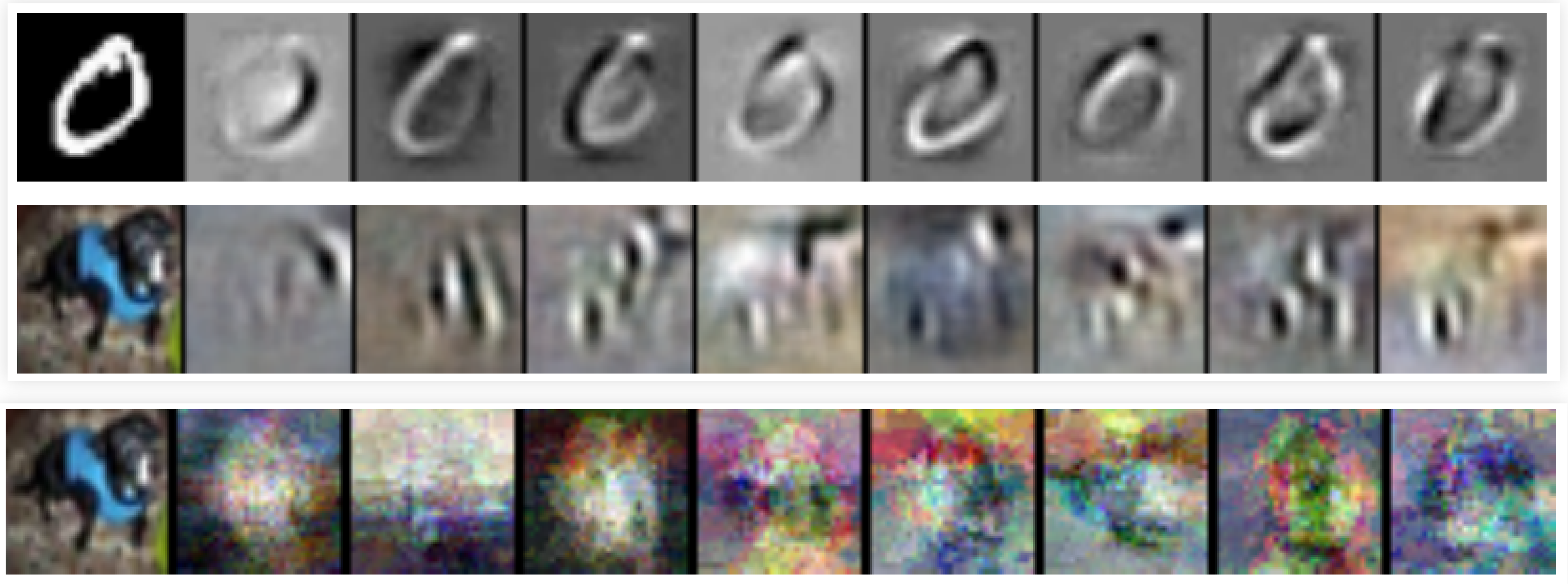
- This makes $h$ less sensitive to small variations in the input
  - More robust encoding; similar examples are grouped closer together
  - This is why it is called contractive (contracts the representation space)
- It also favours smaller weights
  - (the derivative of the activation w.r.t. the input depends on the weights)
- This restricts $h$ to important aspects of the input distribution

## Contractive Autoencoder

- Related to denoising autoencoder:
  - Denoising autoencoder makes $g\left(f\left(x\right)\right)$ less sensitive to perturbations in the input by forcing the encoder to denoise inputs
  - CAE makes $f\left(x\right)$ less sensitive to perturbations in the input by forcing the derivatives $\nabla_x h_i$ to be small
  - This makes them both contractive in the sense of "sqeezing" together similar inputs
- CAE can be expensive to compute
  - Computing the derivatives of $h$ w.r.t. $x$ is simple if only one hidden layer but can become expensive for deep autoencoders.

# Regularized Autoencoders

- CAE approximate the manifold where data is distributed
  - Allows for better learning with fewer data, compared to e.g. PCA



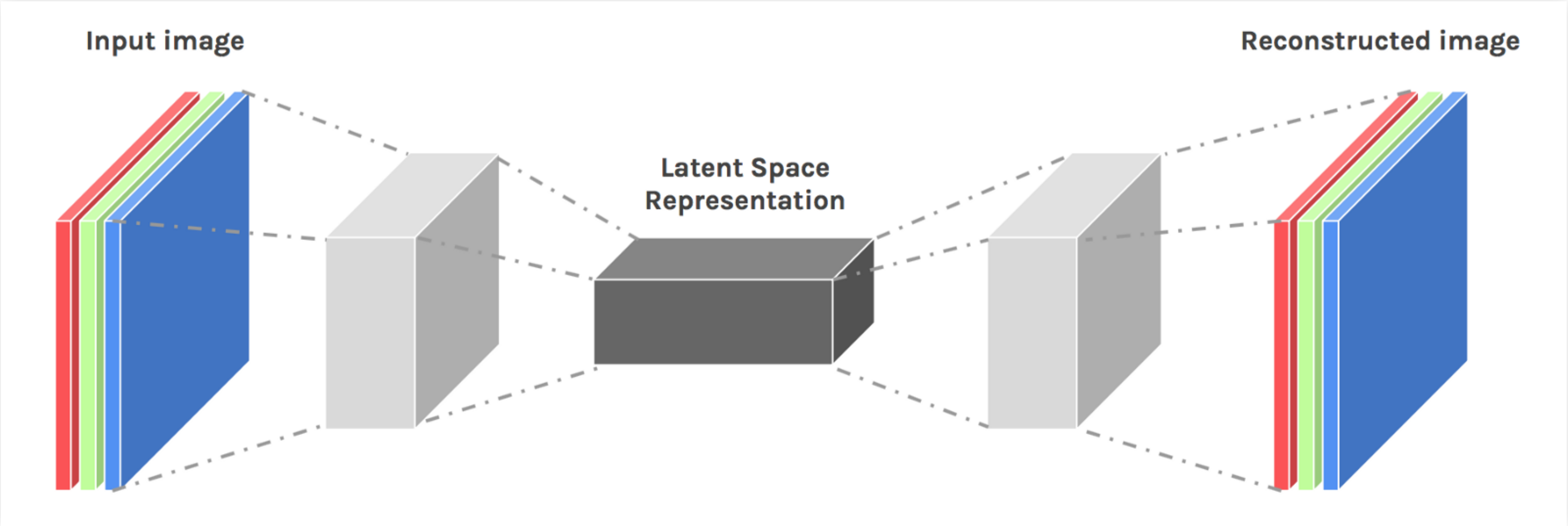
Leading SV of the Jacobian (Rifai et. al., Manifold Tangent Classifier, 2011)

# Convolutional Autoencoders

# Convolutional Autoencoders

## Use convolutions and "deconvolutions" to reconstruct

- Latent space is narrow, need to restore original dimensions




Barna Pásztor, Aligning hand-written digits with Convolutional Autoencoders

# Convolutional Autoencoders

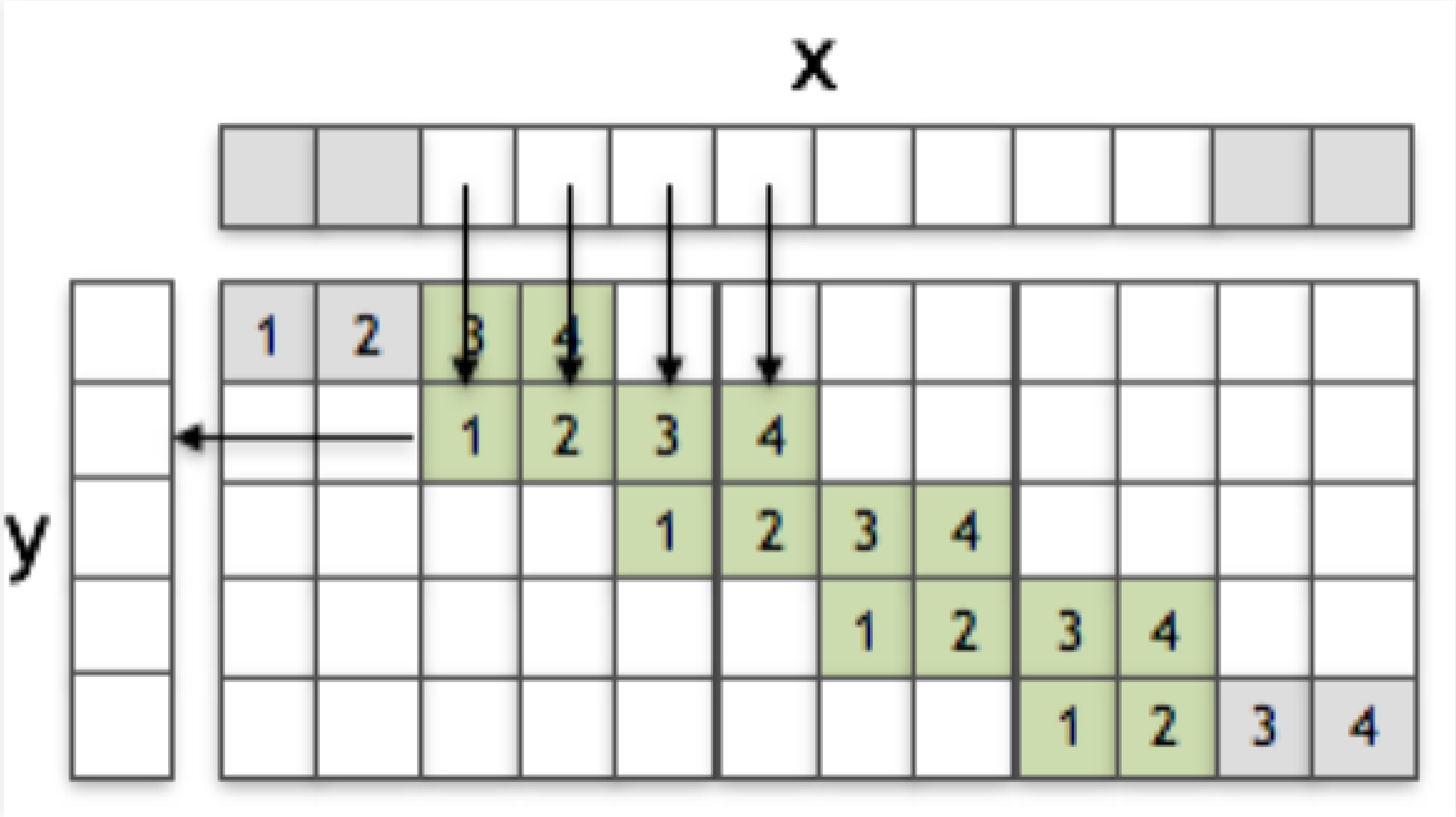
## Use "deconvolutional" layers

- Poor choice of name...
- Several options:

- Transposed convolution
- Fractional stride convolution
- Upsampling (NN or interpolation) followed by convolution
- Multiple convolutions followed by shuffling

# Convolutional Autoencoders

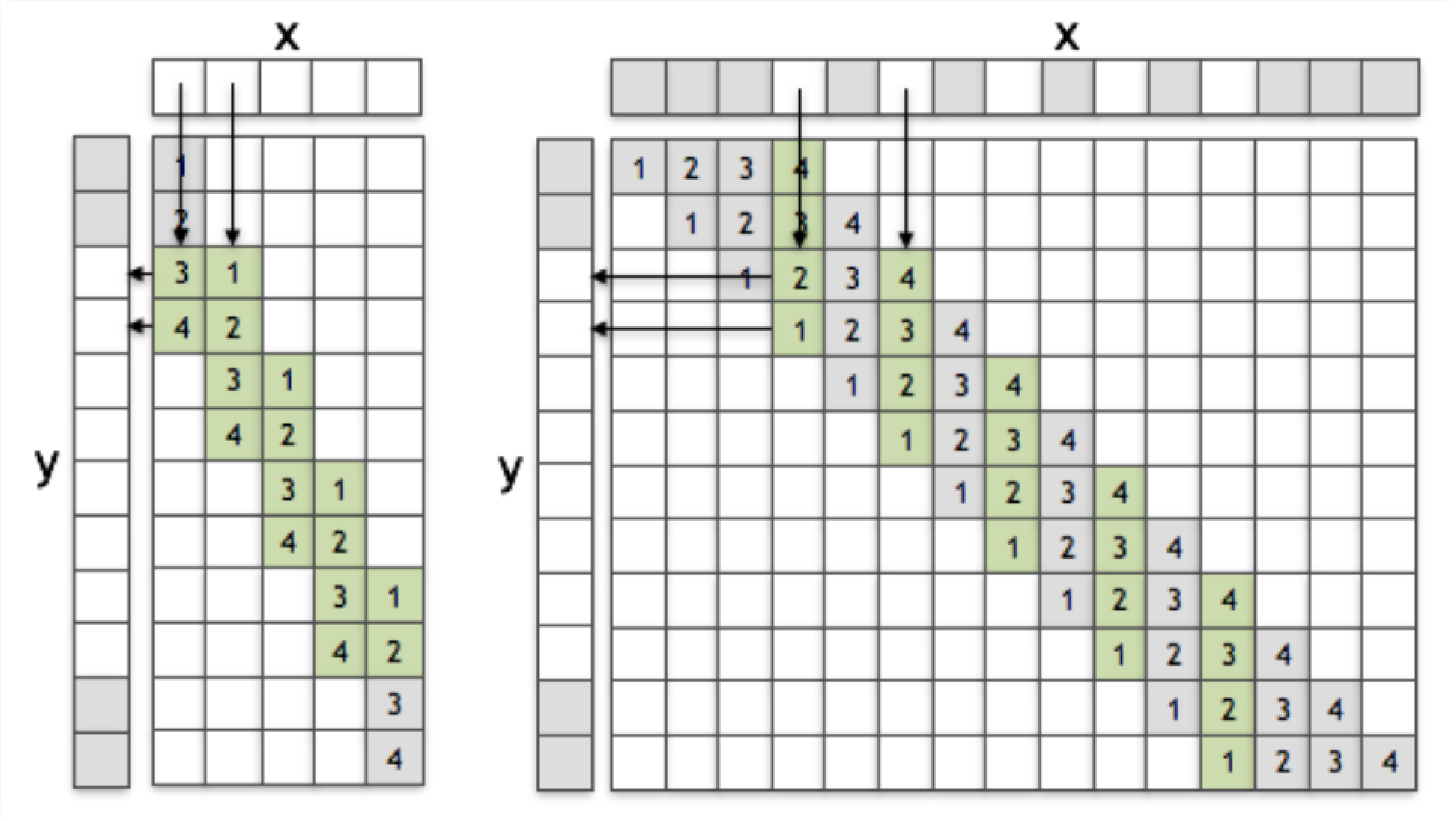
- Convolution in 1D, with padding and stride 2




Shi et. al., Is the deconvolution layer the same as a convolutional layer?

# Convolutional Autoencoders

- Transposed and fractional stride Convolutions in 1D

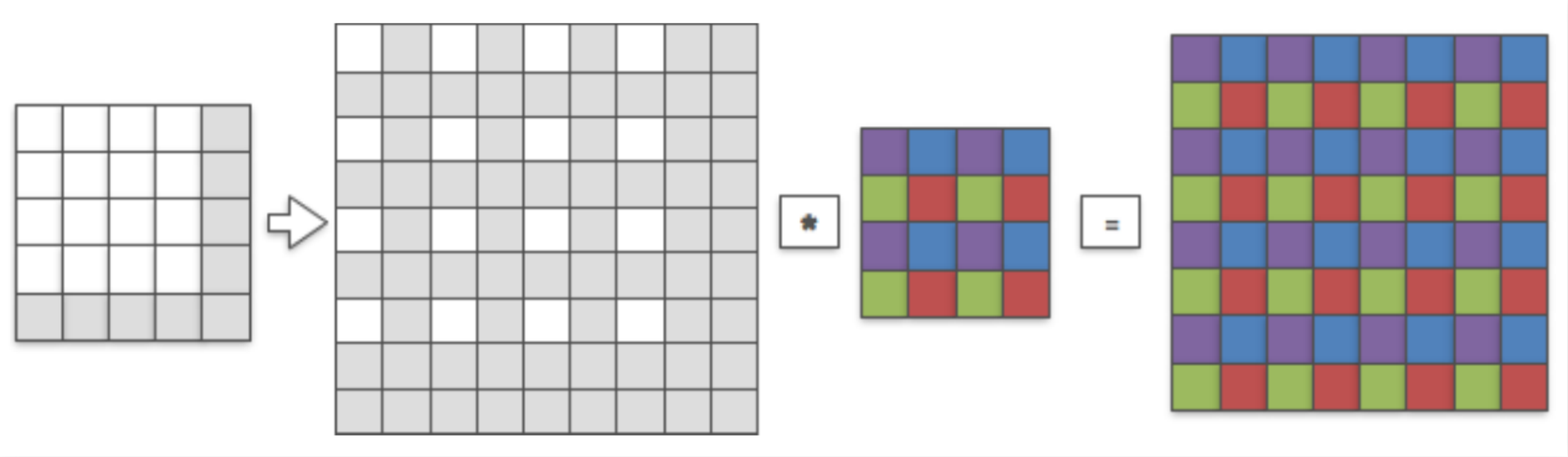



Shi et. al., Is the deconvolution layer the same as a convolutional layer?

# Convolutional Autoencoders

- Upsample followed by convolution (in 2D)

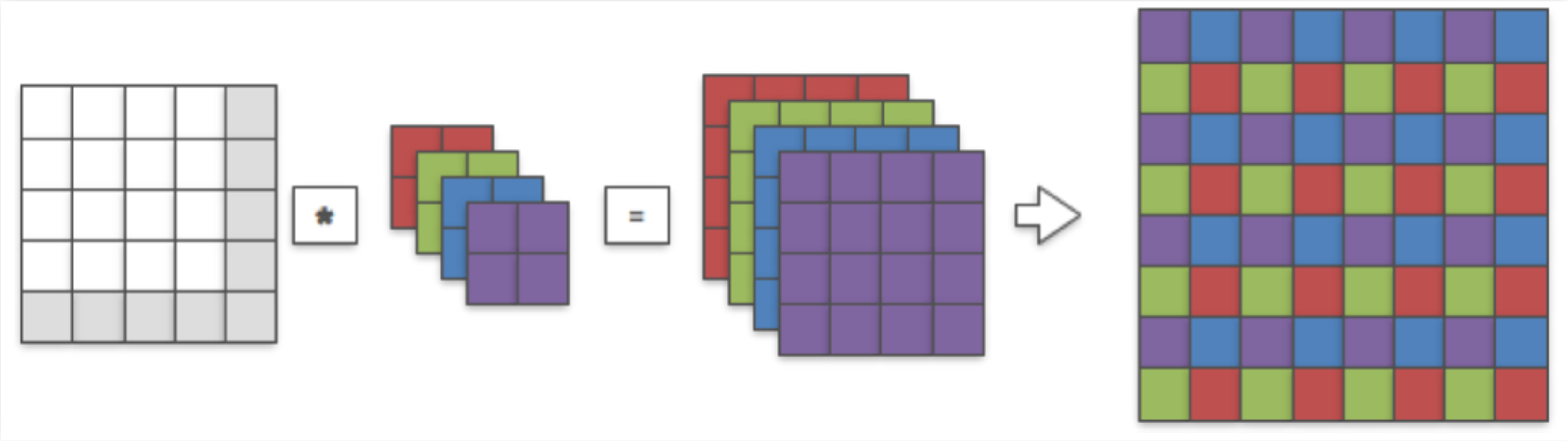
```
from tensorflow.keras.layers import UpSampling1D,UpSampling2D
UpSampling1D(size=2)
UpSampling2D(size=(2, 2), data_format=None, interpolation='nearest')
```




Shi et. al., Is the deconvolution layer the same as a convolutional layer?

# Convolutional Autoencoders

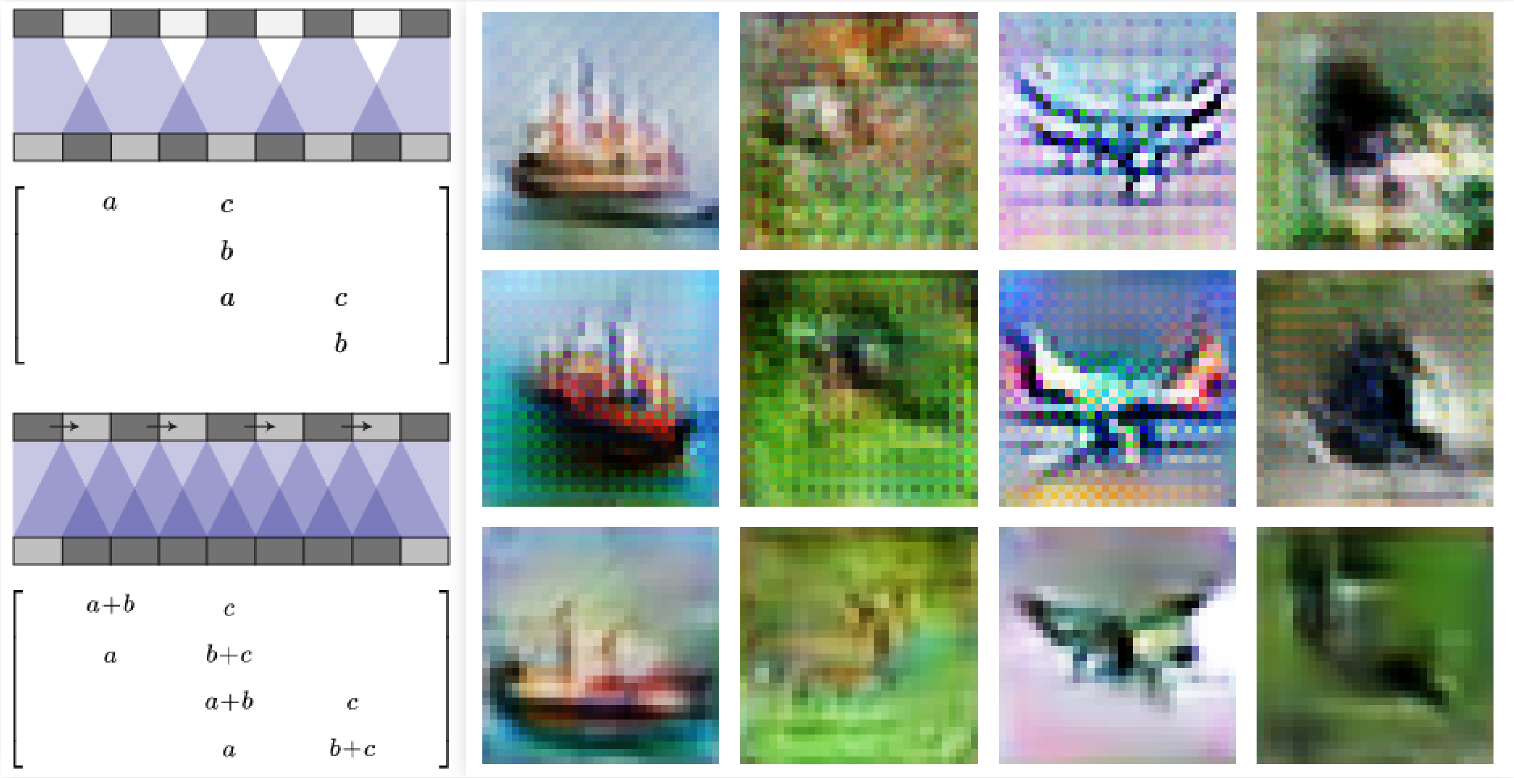
- Alternative: convolutions with several kernels and then shuffle




Shi et. al., Is the deconvolution layer the same as a convolutional layer?

# Convolutional Autoencoders

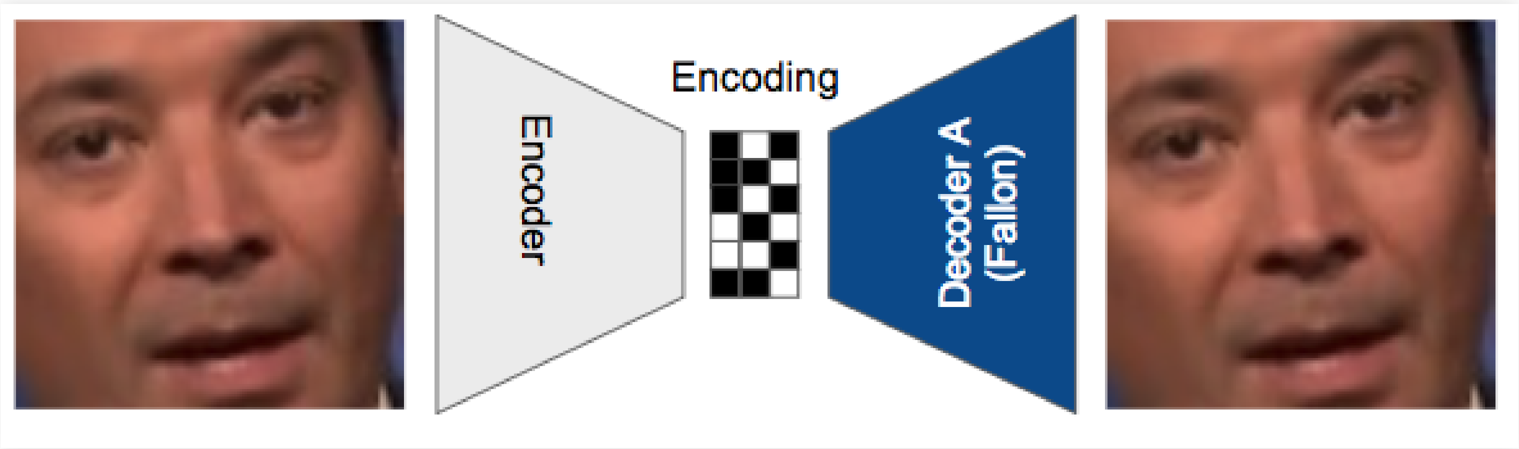
- Best to use upsampling followed by convolution
  - Transposed convolution generates artifacts due to overlaps




Odena, et al., "Deconvolution and Checkerboard Artifacts", Distill, 2016. http://doi.org/10.23915/distill.

## Example: deep fakes

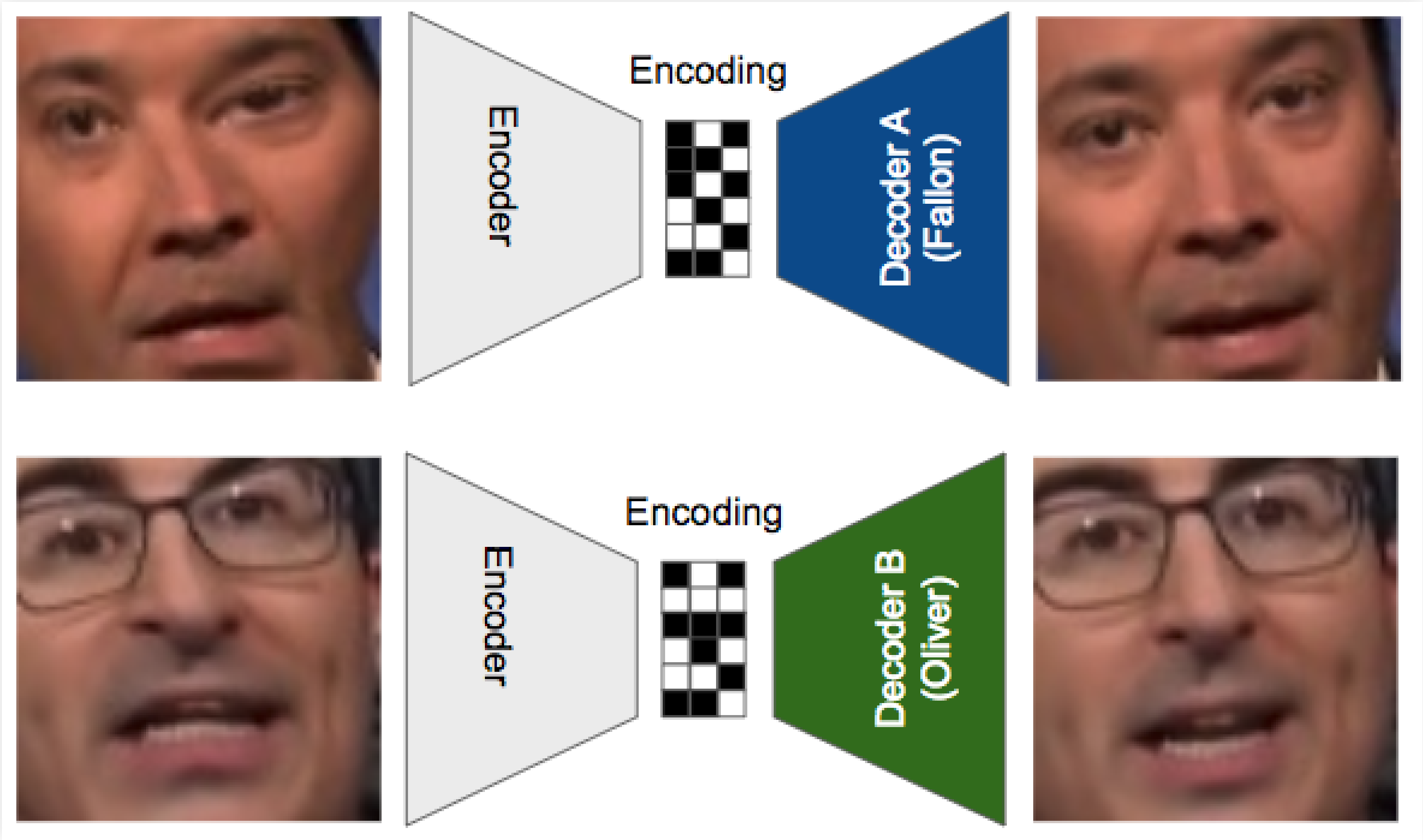
- Use autoencoders to convert one face into another
  - The decoder can reate a face from the correct latent representation
- Problem:
  - How to find the latent representation of the desired face?



## Example: deep fakes

- Use autoencoders to convert one face into another
  - The decoder can reate a face from the correct latent representation
- Problem:
  - How to find the latent representation of the desired face?
- Solution:
  - Train the same encoder on different sets of inputs
  - But for each set reconstruct with a specific decoder
  - Changing the decoder "translates" between sets

## Example: deep fakes




Gaurav Oberoi, Exploring DeepFakes, https://goberoi.com/exploring-deepfakes-20c9947c22d9

# Convolutional Autoencoders

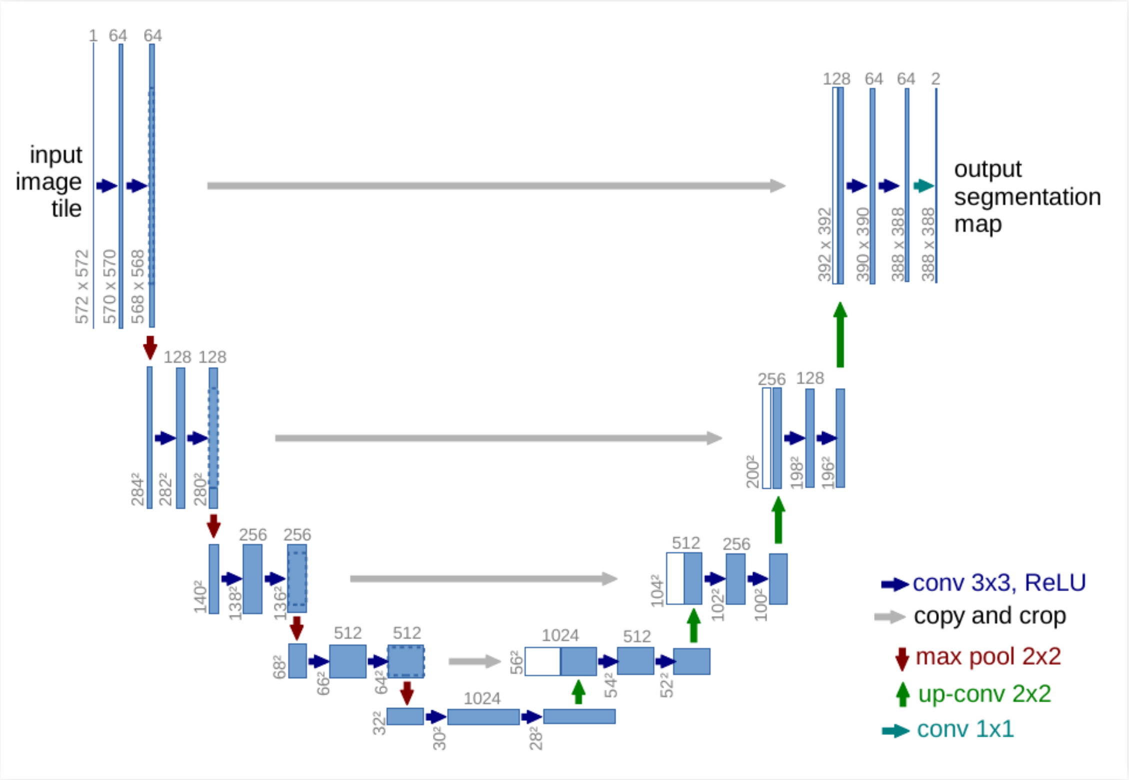
**Example: deep fakes**

- Train with images from videos
- Process video:
  - Input Fallon to encoder
  - Output Oliver using Oliver decoder


Gaurav Oberoi, Exploring DeepFakes, https://goberoi.com/exploring-deepfakes-20c9947c22d9

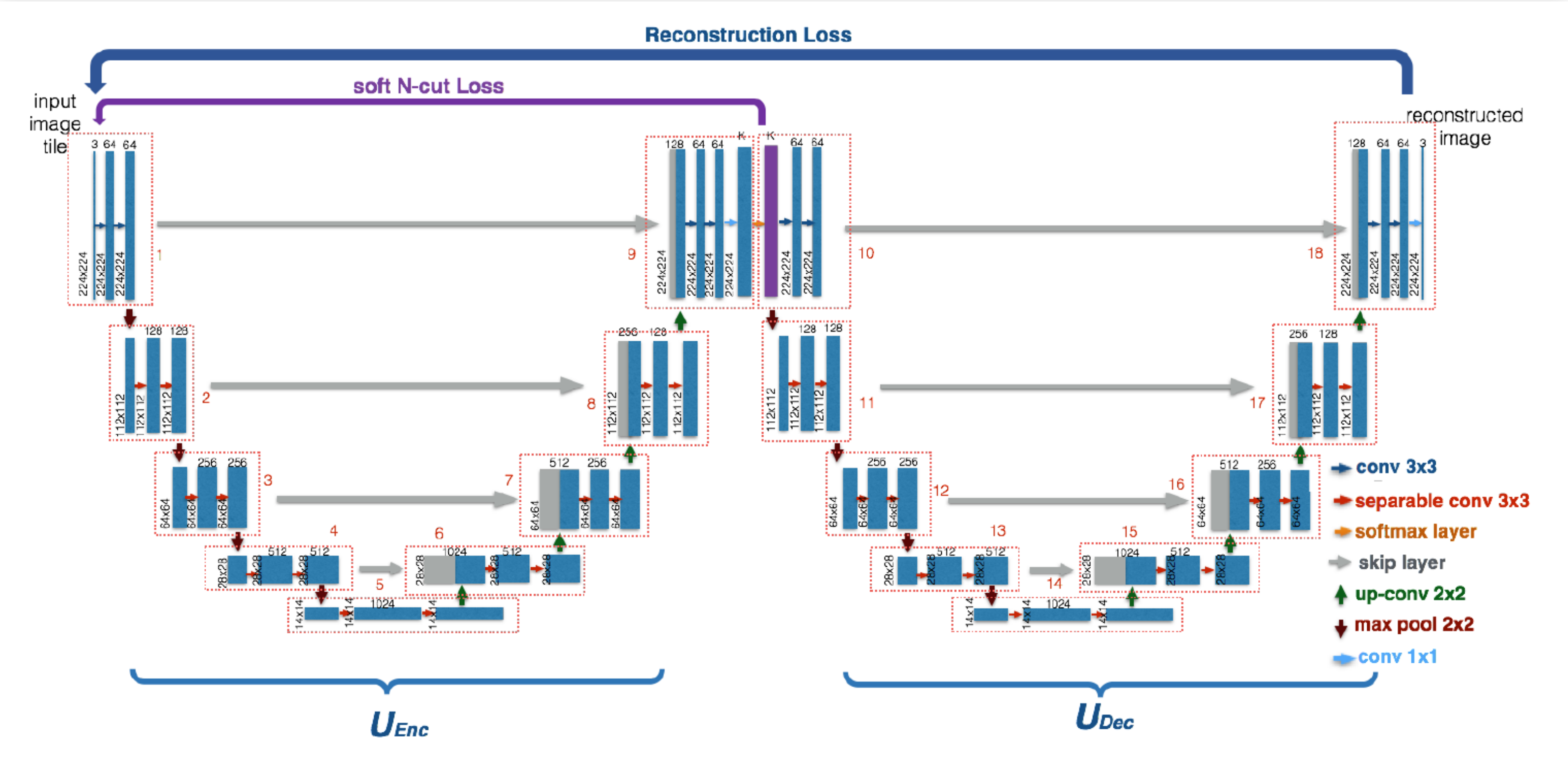
## Unsupervised Segmentation

- Remember U-Net for supervised segmentation:



## Unsupervised Segmentation

- W-Net uses two U-Net:




Xia et. al, W-Net: A Deep Model for Fully Unsupervised Image Segmentation

## Unsupervised Segmentation

- W-Net uses two U-Net, the reconstruction loss (quadratic error) plus a soft normalized cut loss

## Normalized cut

- If we split a graph into $A$ and $B$

$$cut(A,B) = \sum_{u \in A, v \in B} w(A,B)$$

- The normalized cut:

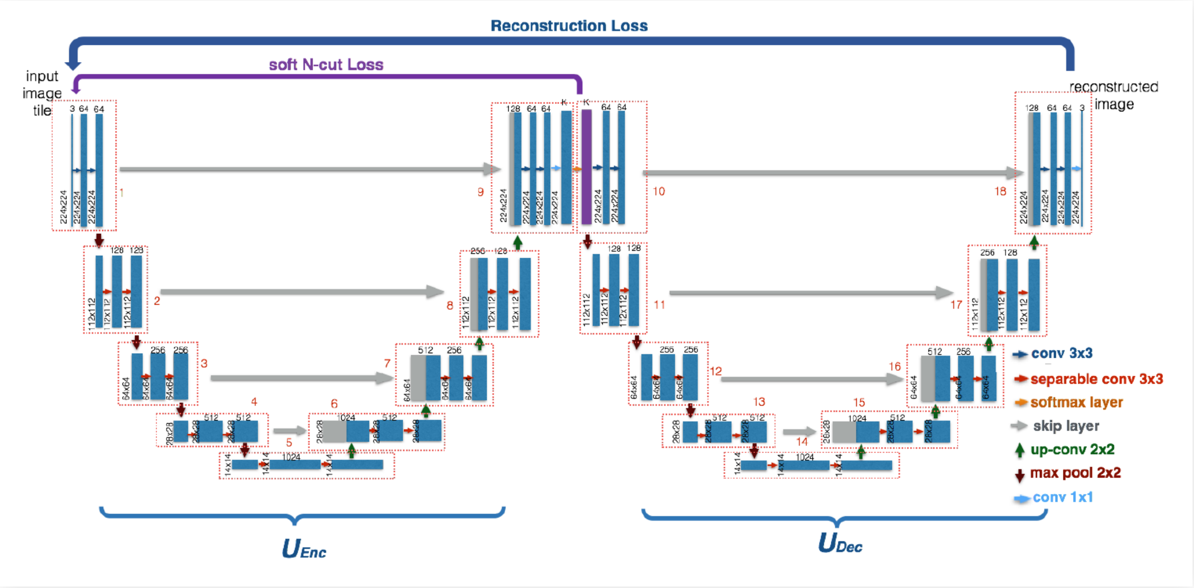
$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$assoc(X,V) = \sum_{u \in X, t \in V} w(u,t)$$

Shi et. al., Normalized Cuts and Image Segmentation

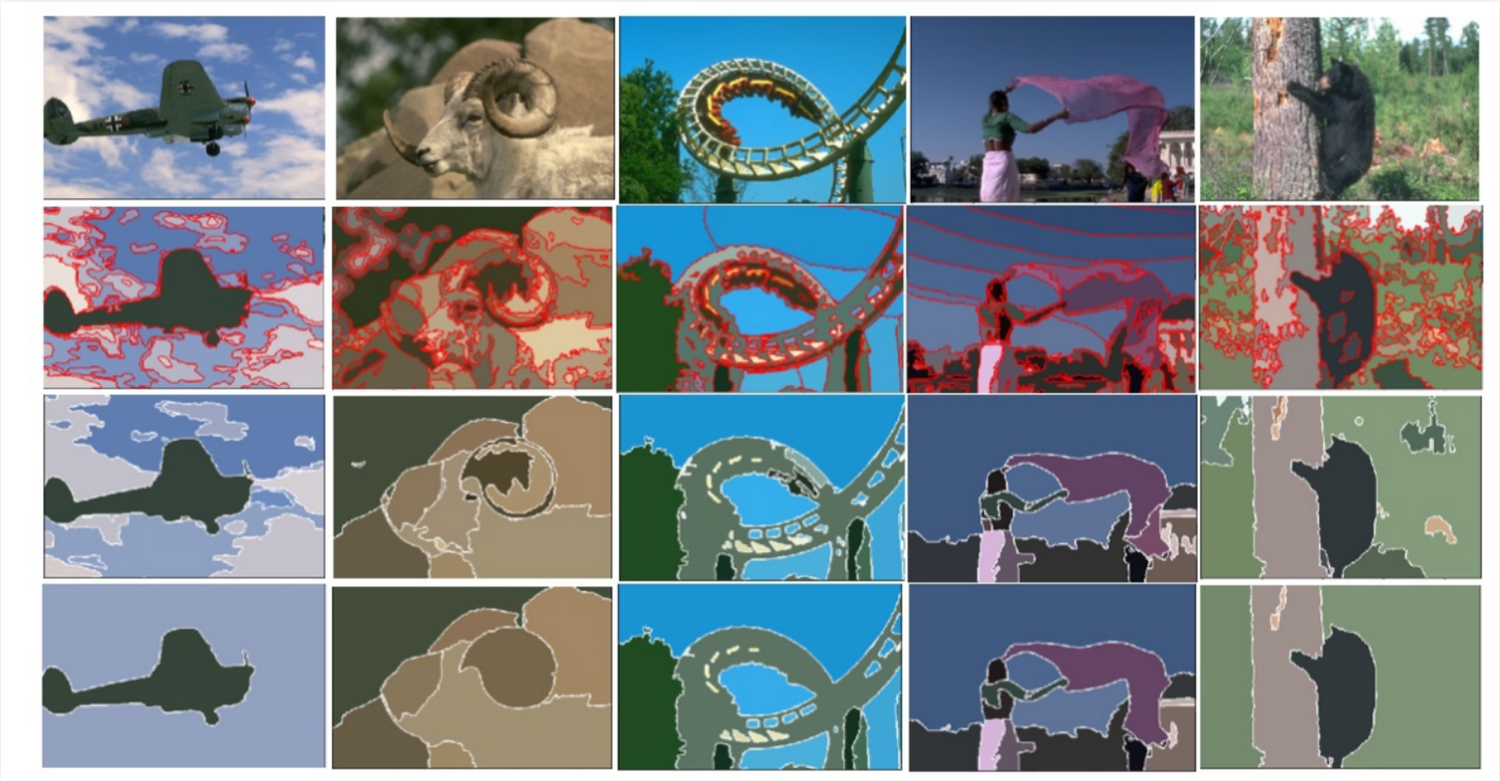
## Unsupervised Segmentation

- Normalized cut is not differentiable, W-Net uses soft normalized cut
- Training alternates, for each minibatch:
  - Update encoder to minimize soft normalized cut
  - Update whole net to minimize reconstruction loss (quadratic loss)



# Convolutional Autoencoders

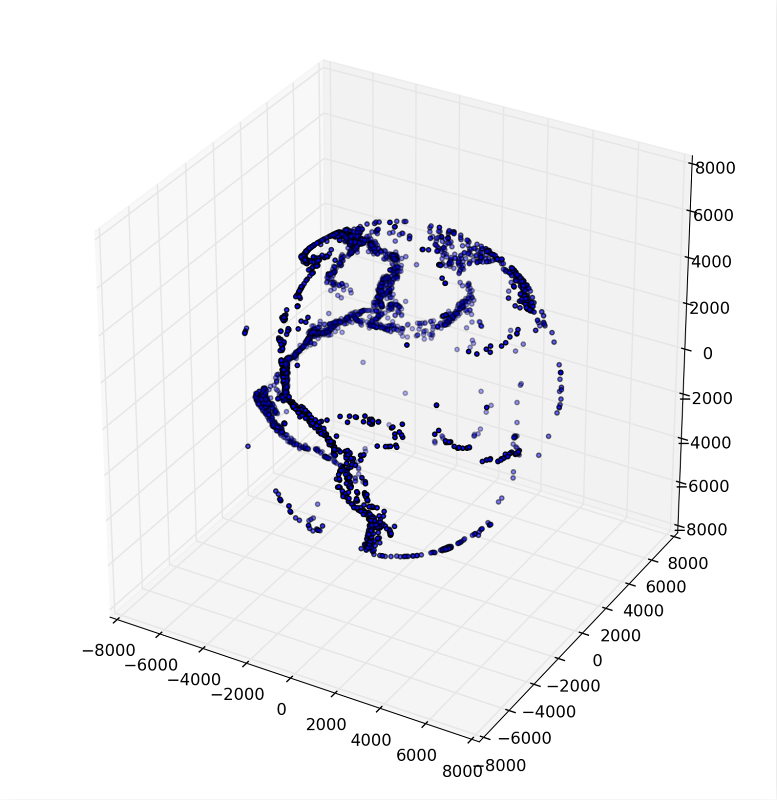
- Finally, post-processing for smoothing and clustering



# Applications

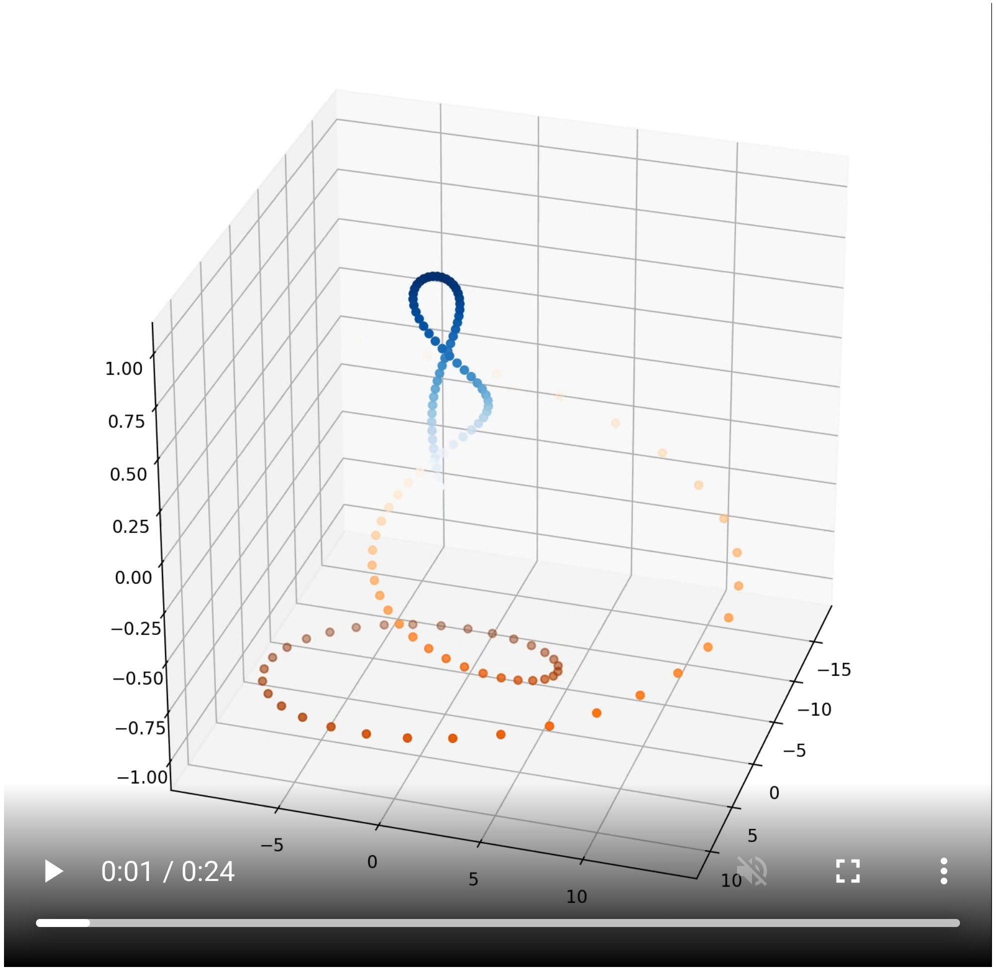
## Manifold

- A set of points such that the neighbourhood of each is homomorphic to an euclidean space
  - Example: the surface of a sphere



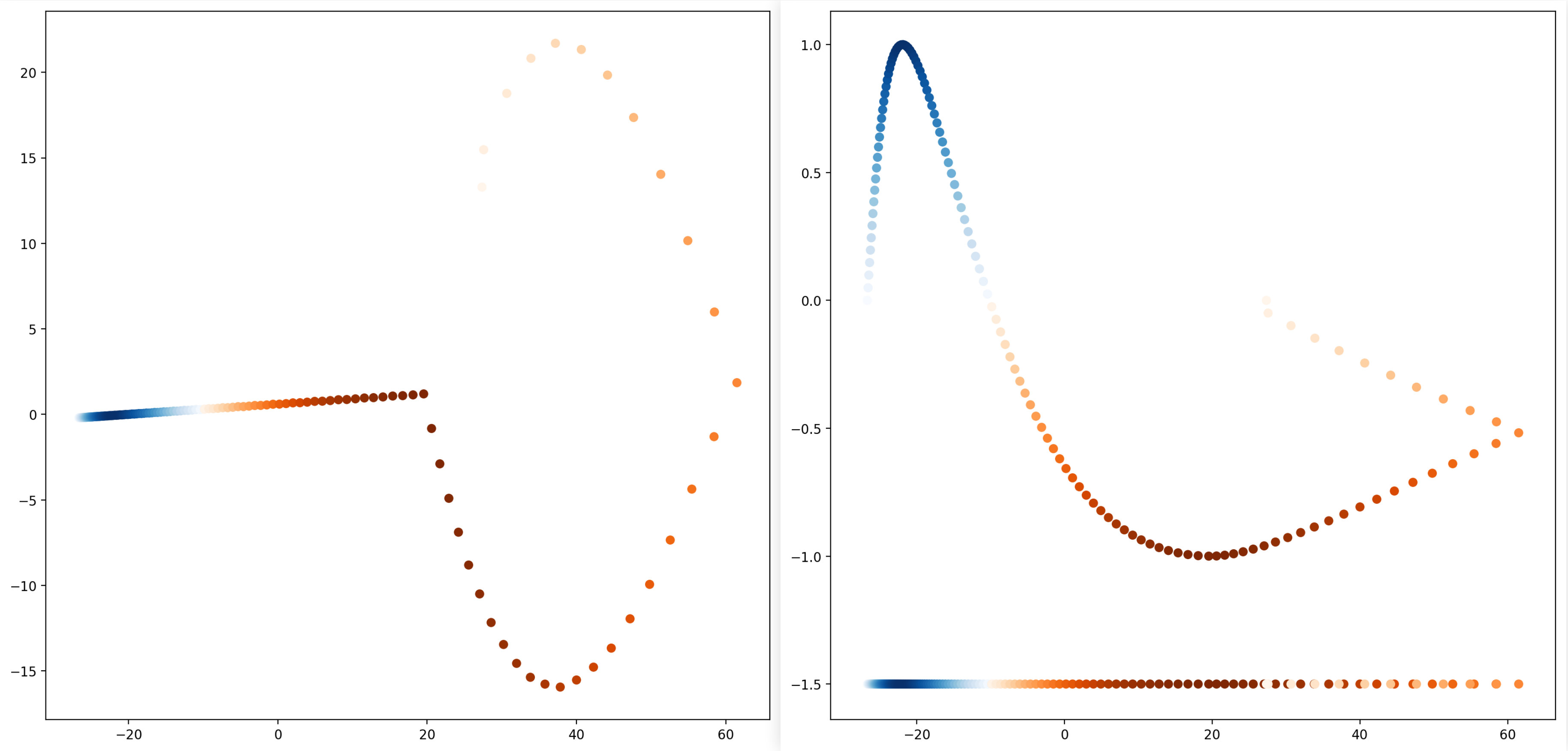
# Manifold Learning

- Data may cover a lower dimension manifold of the space



# Manifold Learning

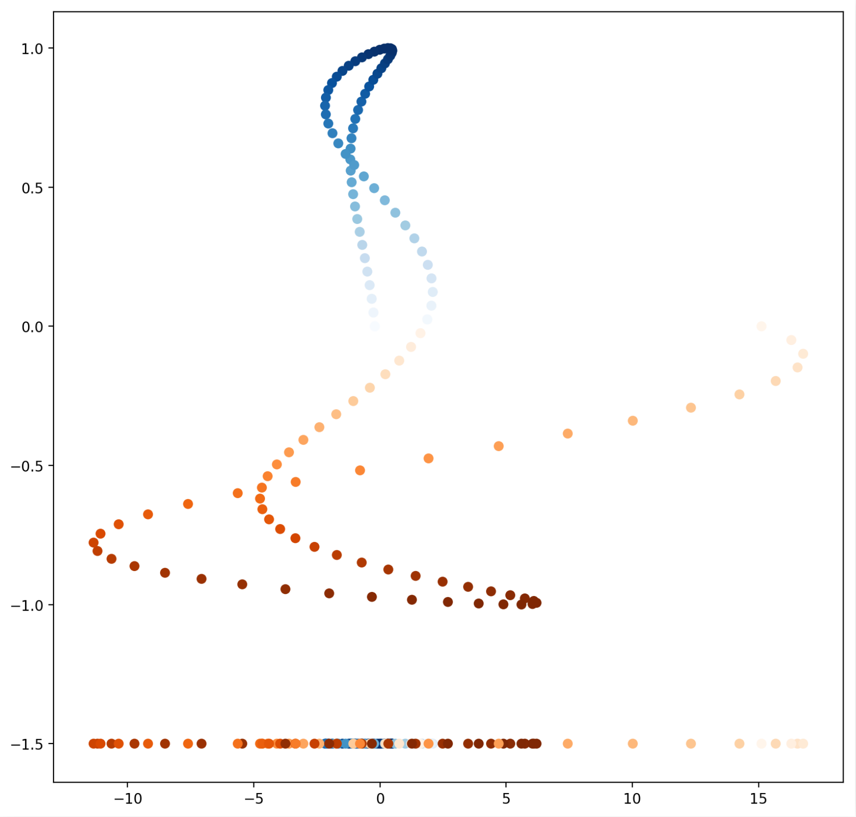
- Learn lower dimension embeddings of data manifold




FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

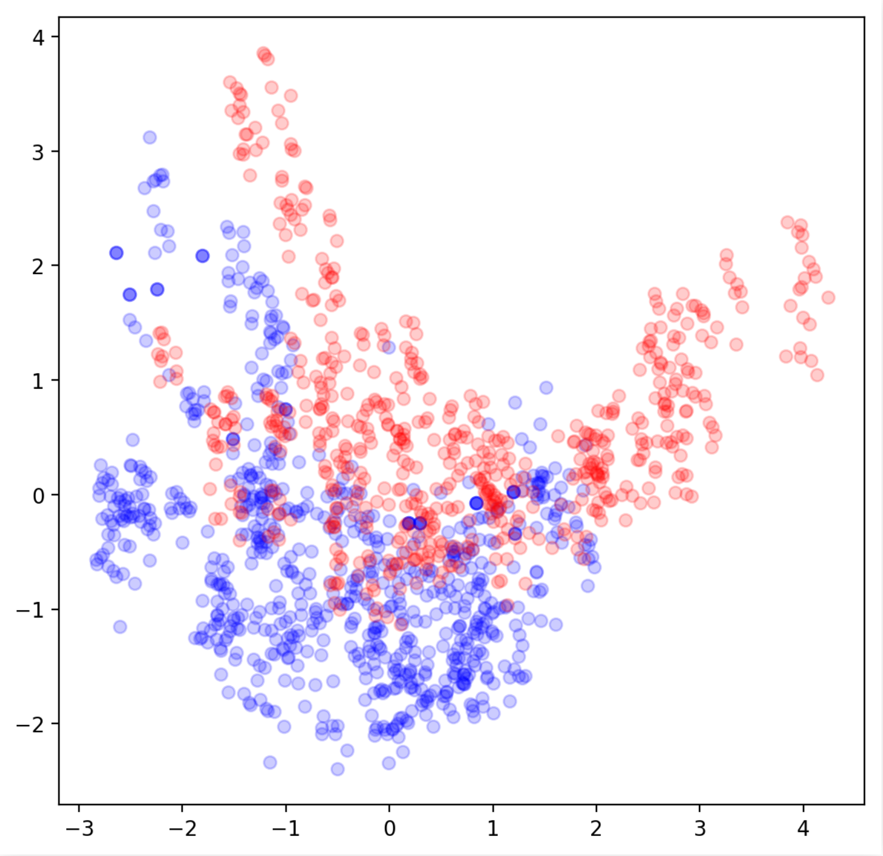
# Manifold Learning

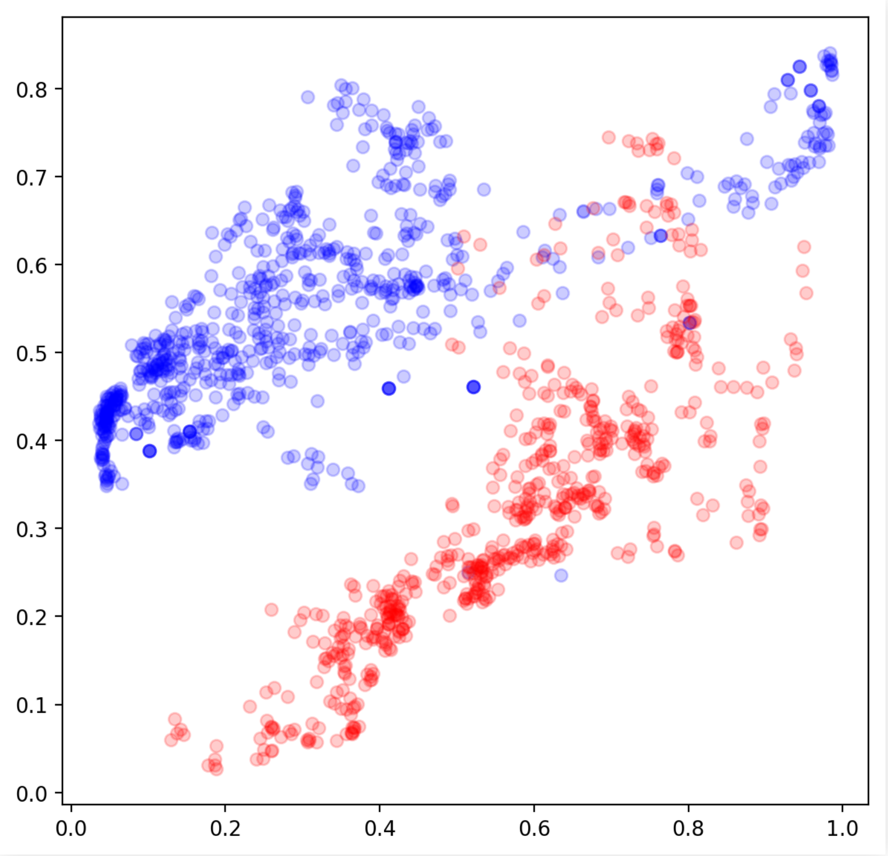
- Linear transformations do not work well for this
  - PCA just chooses a straight line along largest variance, cannot follow manifold
  - A linear autoencoder would e equivalent to PCA



# Dimensionality reduction

- Nonlinearity makes dimensionality reduction adapt to manifold
  - PCA vs autoencoder (4),6,4,2,4,6,(4) UCI bannknote dataset (4 features)



## Manifold learning and dimensionality reduction

- This works because we force the network in two opposite ways:
  - We demand the ability to reconstruct the input
  - But we also constrain how the network can encode the examples
- Undercompleteness is just one way of doing this

## Beware overfitting.

- If the autoencoder is sufficiently powerful, it can reconstruct the training data accurately but lose generalization power
- In the extreme, all information about reconstructing the training set may be in the weights and the latent representation becomes useless

## Detecting outliers

- Autoencoders learn a representation of the training data
  - The manifold depends on the distribution of $X$
- The reconstruction error for an anomaly will be higher
- This is good with unbalanced or partially labelled data
  - We have many normal examples but only a few exceptions
  - The anomalies are of different types
- We can train the autoencoder with only the normal data

# Summary

# Autoencoders

## Summary

- Autoencoders: learn the input in the output
  - Unsupervised learning
  - Copy with restrictions (dimension, regularization)
  - Or reconstruction (from corrupted inputs)
- Convolutional Autoencoders
- Applications

## Further reading:

- Goodfellow et.al, Deep learning, Chapter 14