Autoencoders

Ludwig Krippahl

Autoencoders

Summary

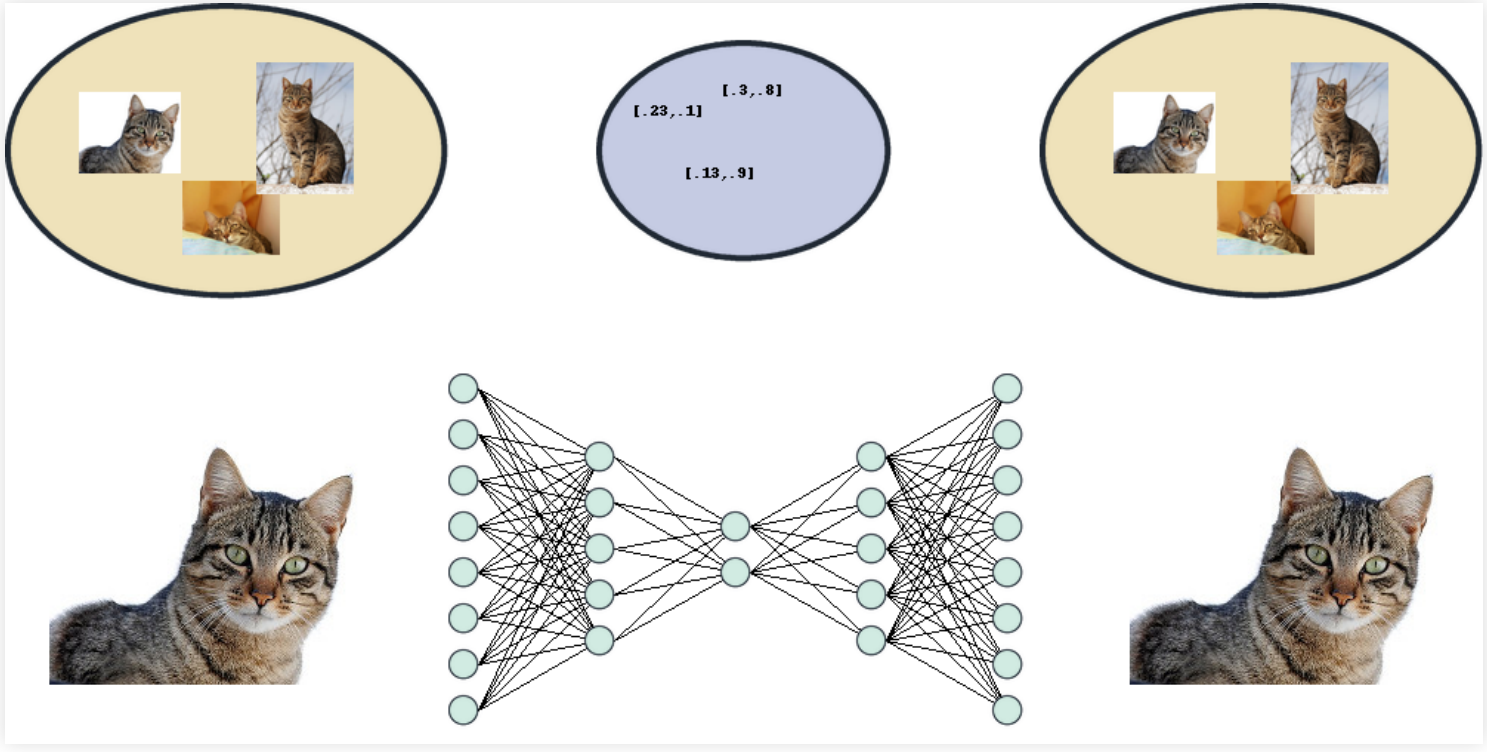
- What are Autoencoders
- Different restrictions on encoding
  - Undercompleteness
  - Regularization
  - Sparsity
  - Noise reconstruction
- Applications
- Example: dimensionality reduction

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

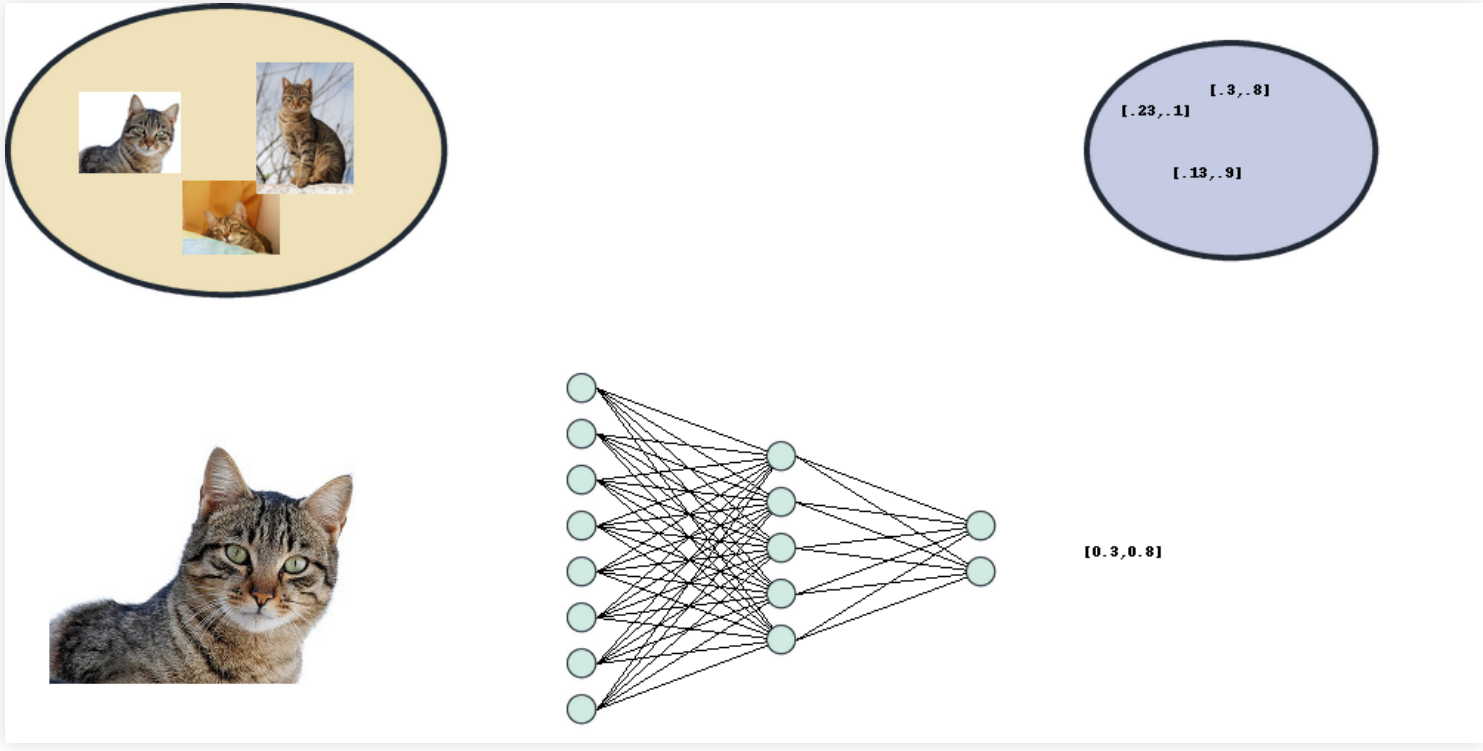


Cat images: Joaquim Alves Gaspar CC-SA

What are autoencoders?

Network trained to output the input (unsupervised)

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

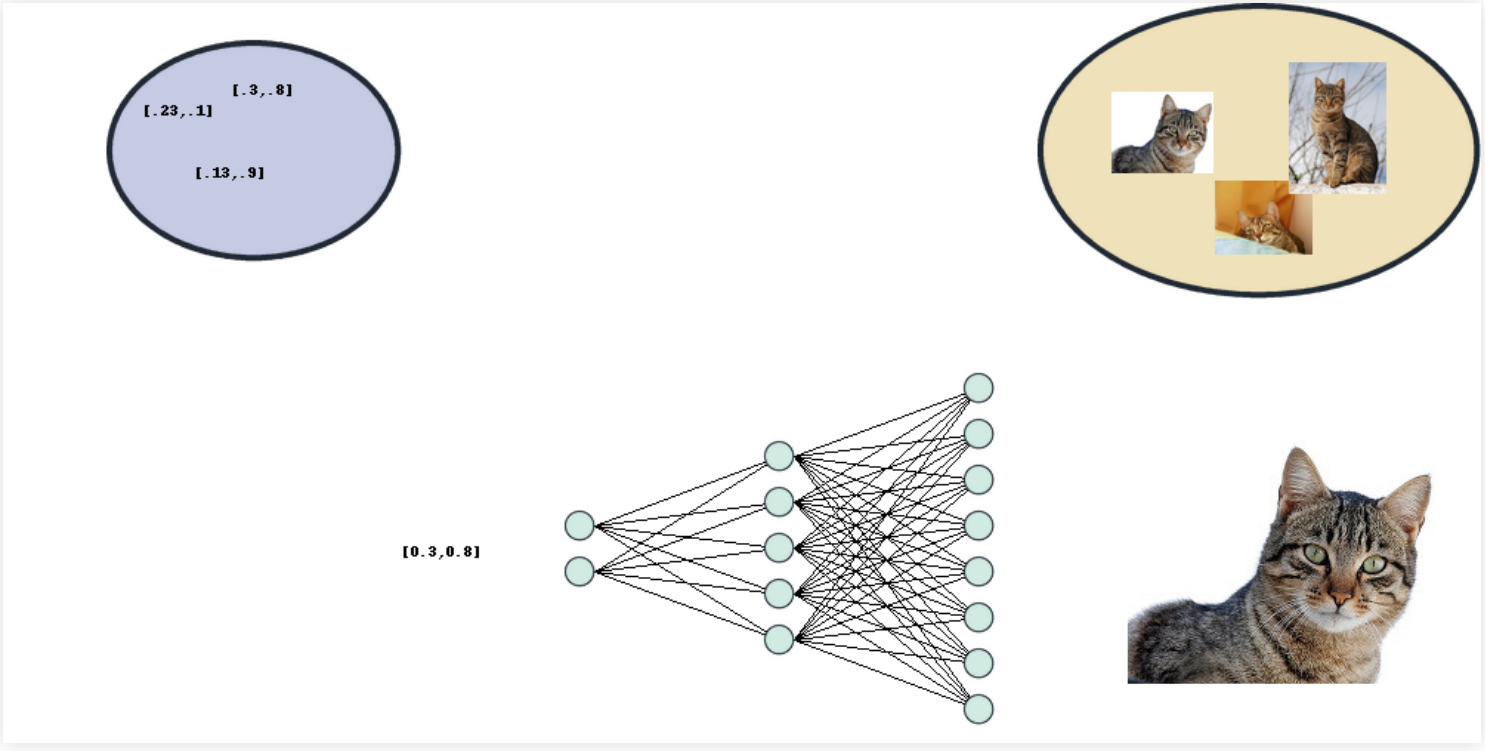


Cat images: Joaquim Alves Gaspar CC-SA

What are autoencoders?

Network trained to output the input (unsupervised)

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

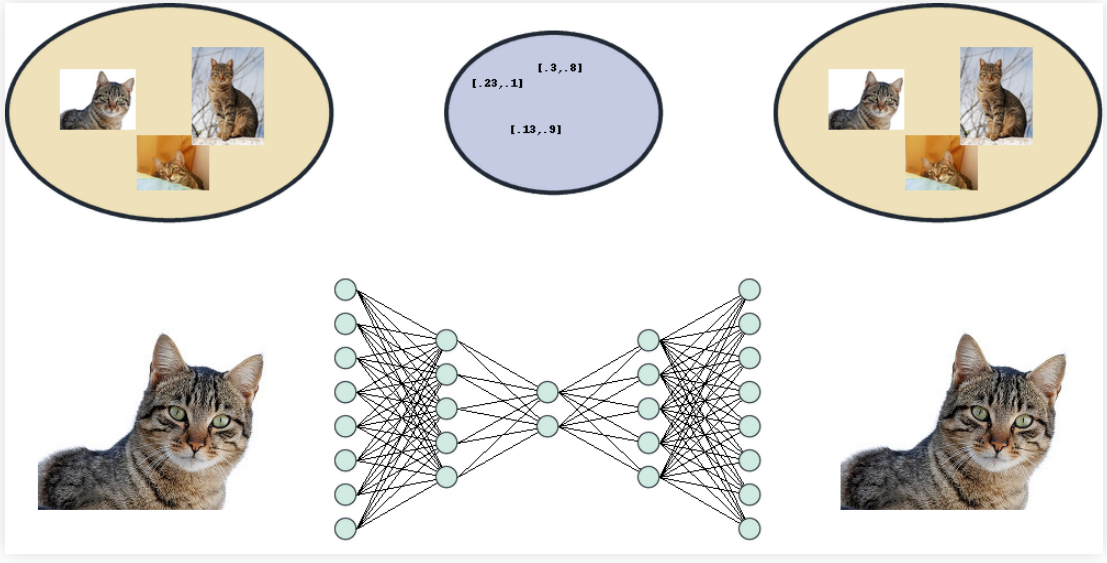


Cat images: Joaquim Alves Gaspar CC-SA

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 $x = g\,(f\,(x))$?



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 $x = g\,(f\,(x))$?
- 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 just feedfoward networks

- Can be trained with the same algorithms, such as backpropagation

- But since the target is x, they are unsupervised learners
- Need some "bottleneck" to force a useful representation

- Otherwise just copies values

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

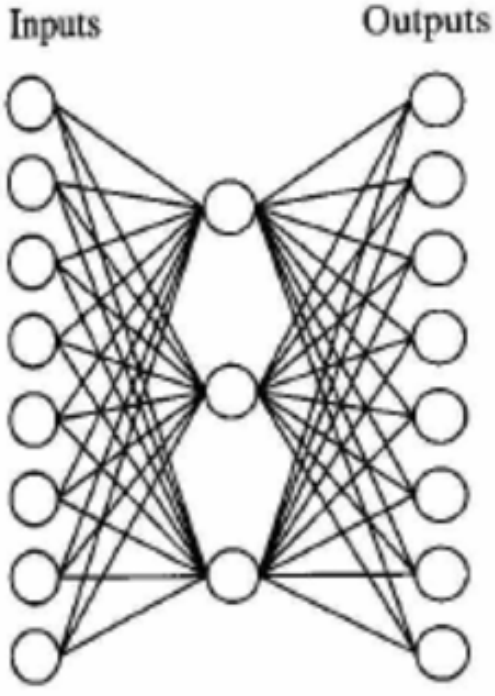
- With too much power, autoencoder can just index each training example and learn nothing useful:

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

Undercomplete Autoencoders

Autoencoder is undercomplete if h is smaller than x

- Mitchell's autoencoder, hidden layer of 3 neurons



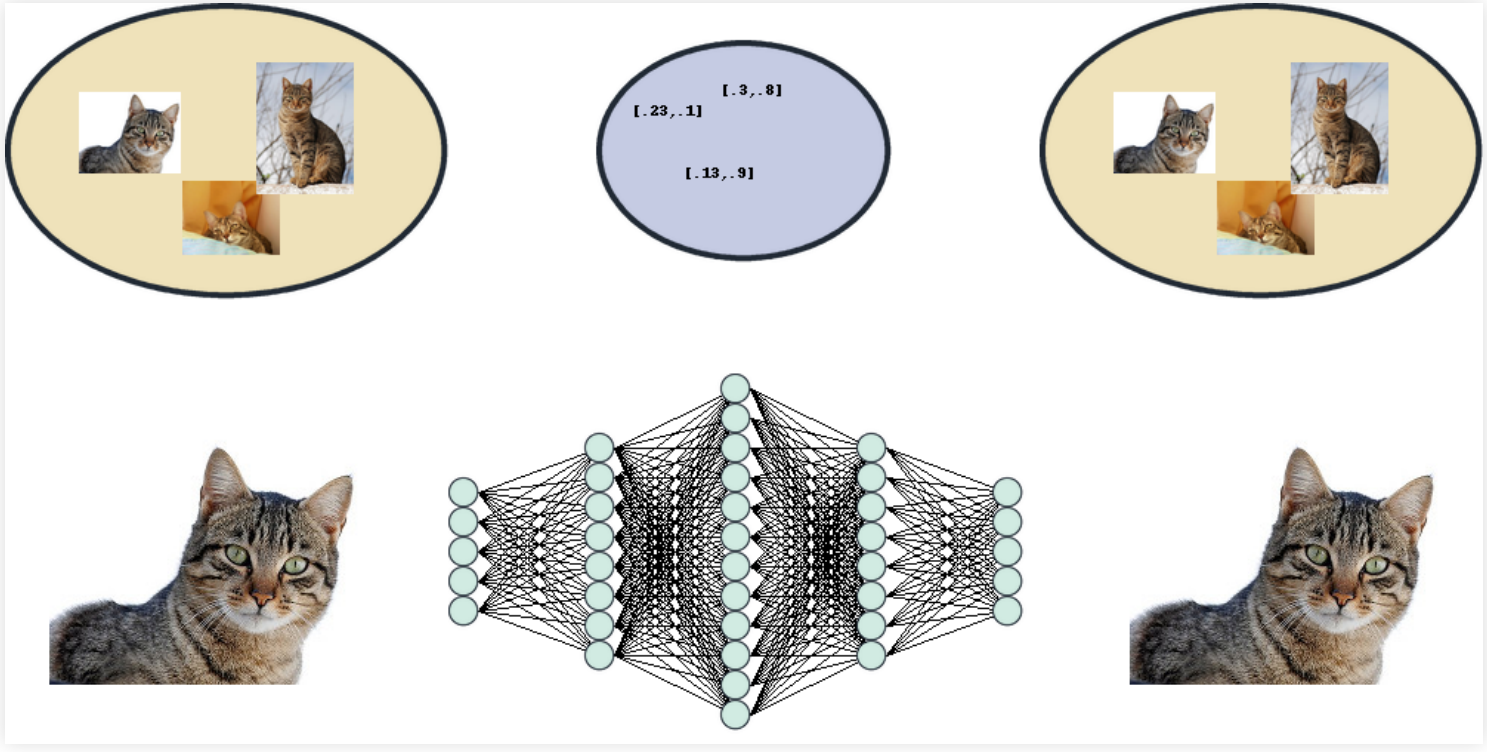
Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Tom M. Mitchell, Machine Learning, McGraw Hill 1997

Regularized Autoencoders

An overcomplete autoencoder has $h \geq x$

- 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 $h \geq x$

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

Example: contractive autoencoder

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

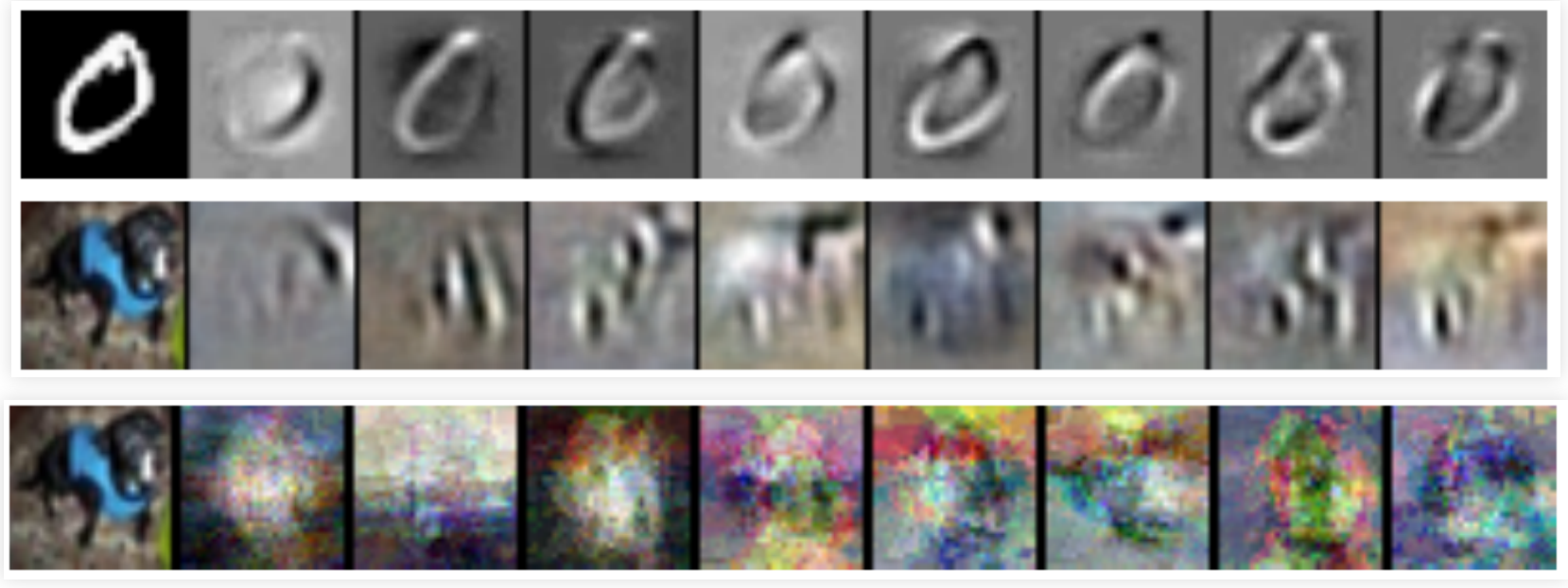
$$L(x, g(f(x))) + \lambda \sum_i \|\nabla_x h_i\|^2$$

- This makes h less sensitive to small variations in the input
  - More robust encoding; similar examples are close together
- 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

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)

Sparse Autoencoders

Force h to have few activations (sparse)

- Example: we want the probability of h_i firing

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

  to be equal to ρ (the sparseness parameter)

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

$$L(x, g(f(x))) + \lambda \sum_i \left(\rho \log \frac{\rho}{\hat{\rho}_i} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_i} \right)$$

- Note: sparseness may be due to small, nonzero, activations

- But it can correspond to zero activation in neurons if using ReLU

Sparse Autoencoders

Sparse autoencoders make neurons specialize

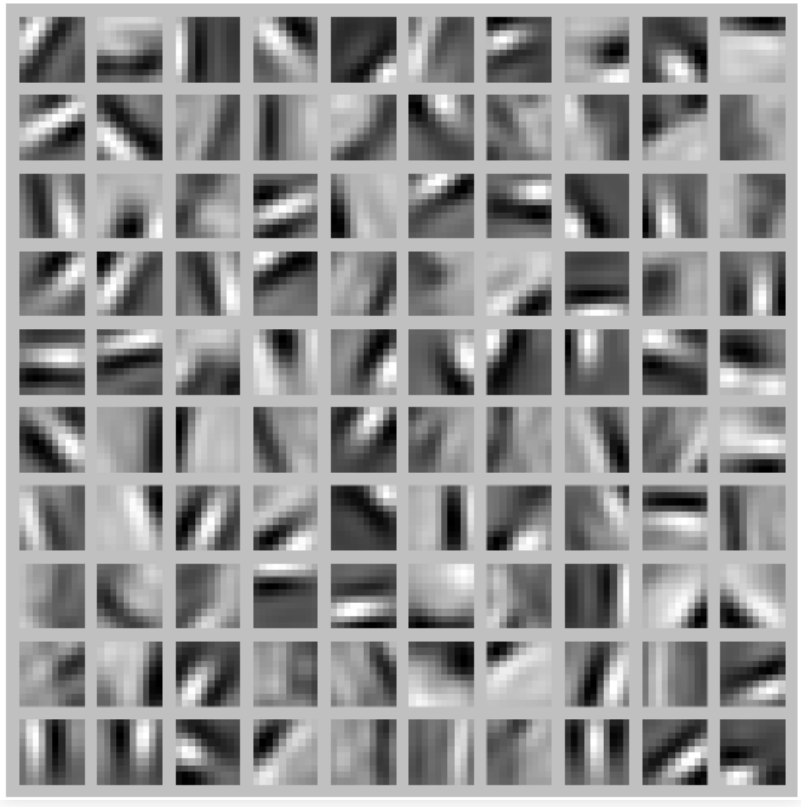
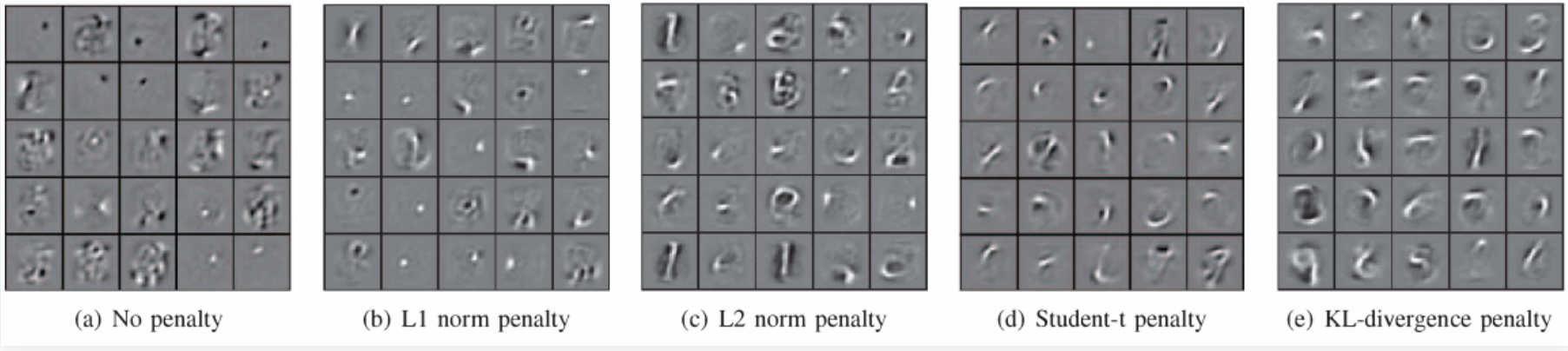


Image: Andrew Ng

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

Sparse Autoencoders

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



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

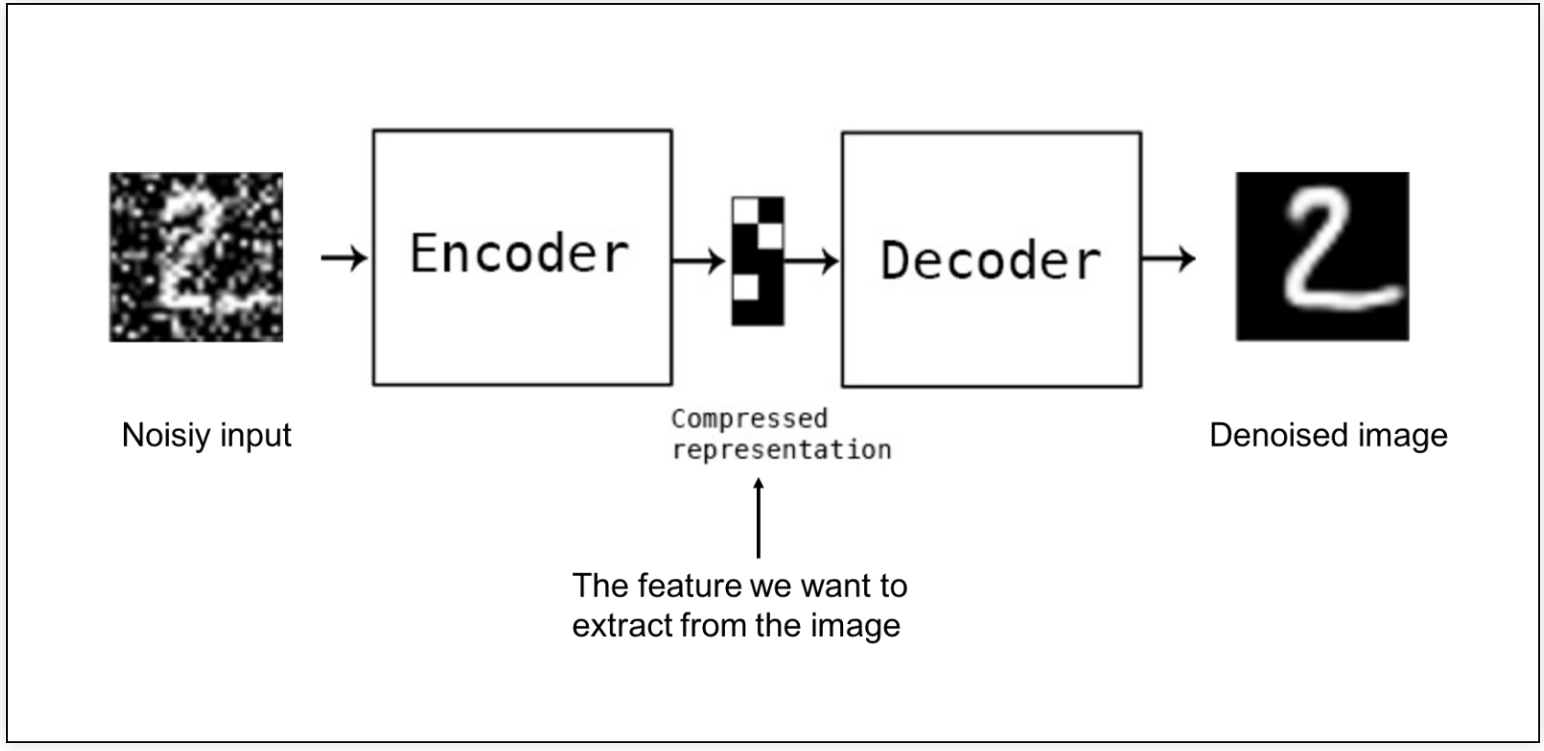
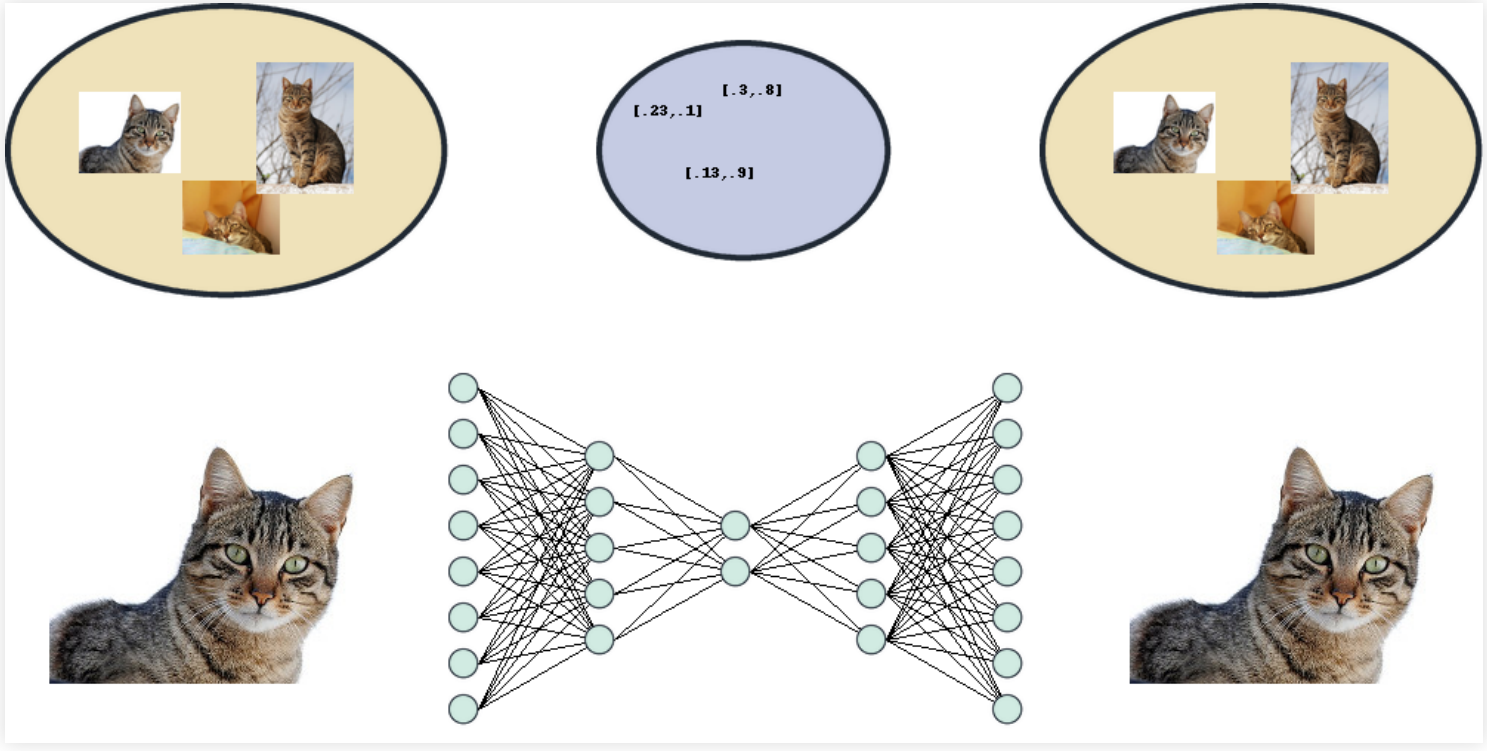


Image: Adil Baaj, Keras Tutorial on DAE

Generating Data

Generating Data

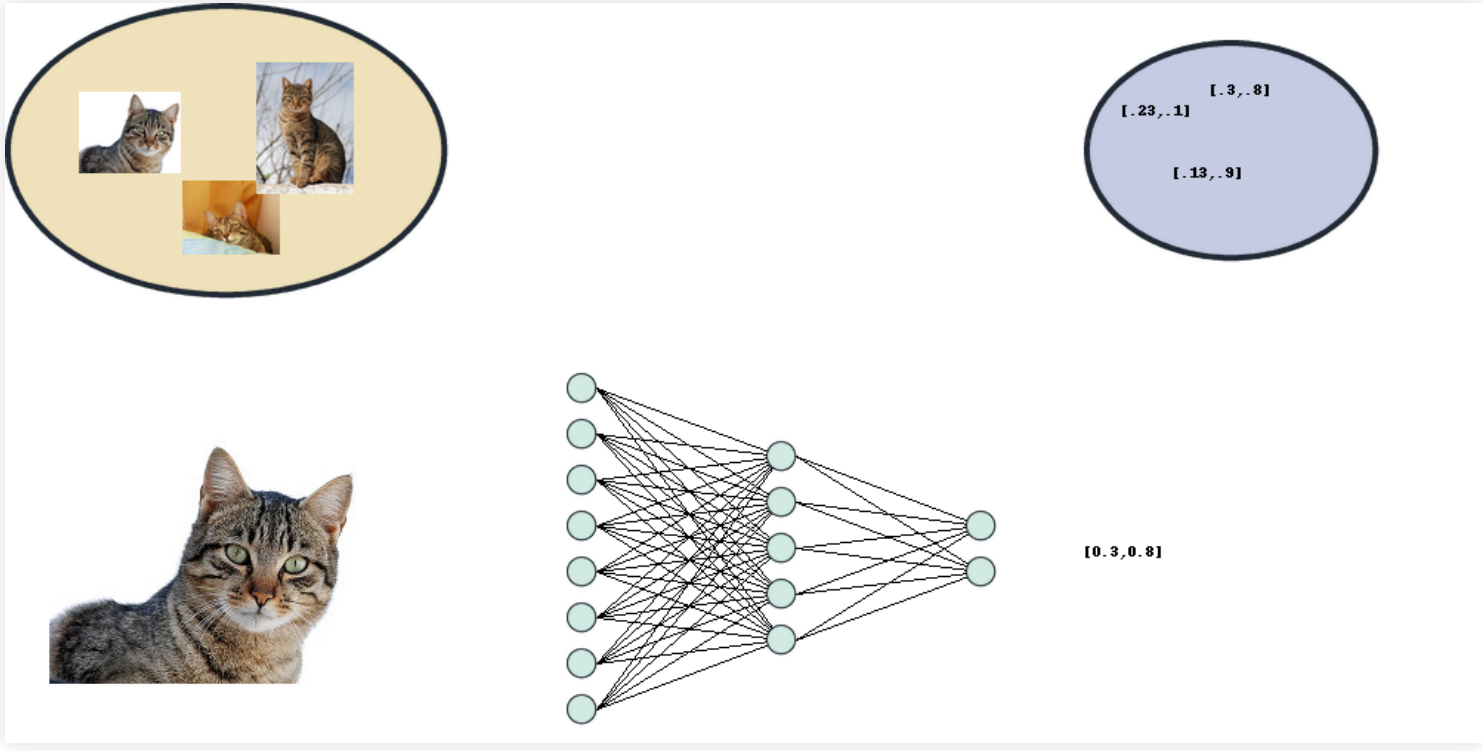
- Can we use autoencoders to generate new examples?



Cat images: Joaquim Alves Gaspar CC-SA

Generating Data

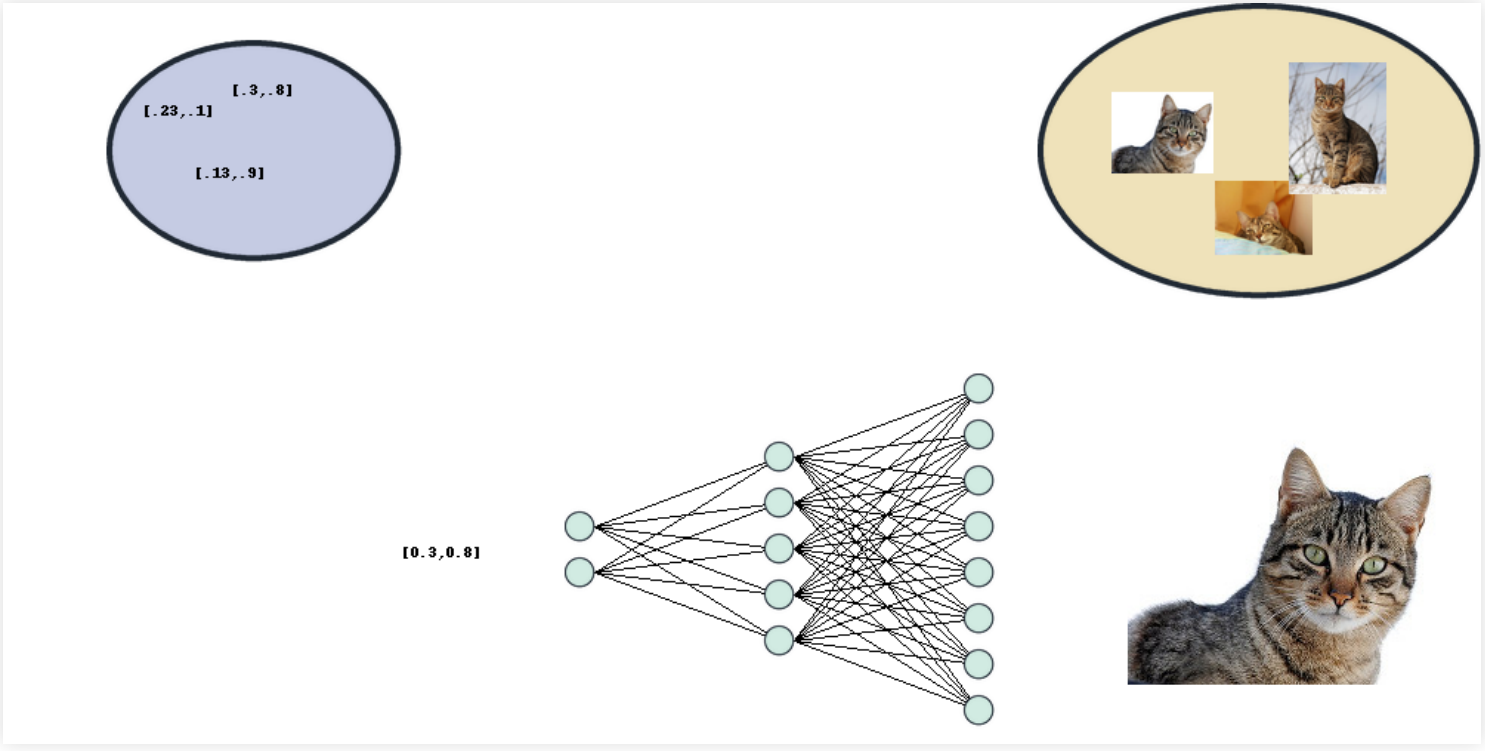
- Autoencoders create a latent representation from the data



Cat images: Joaquim Alves Gaspar CC-SA

Generating Data

- And then decode to recreate the data from this representation

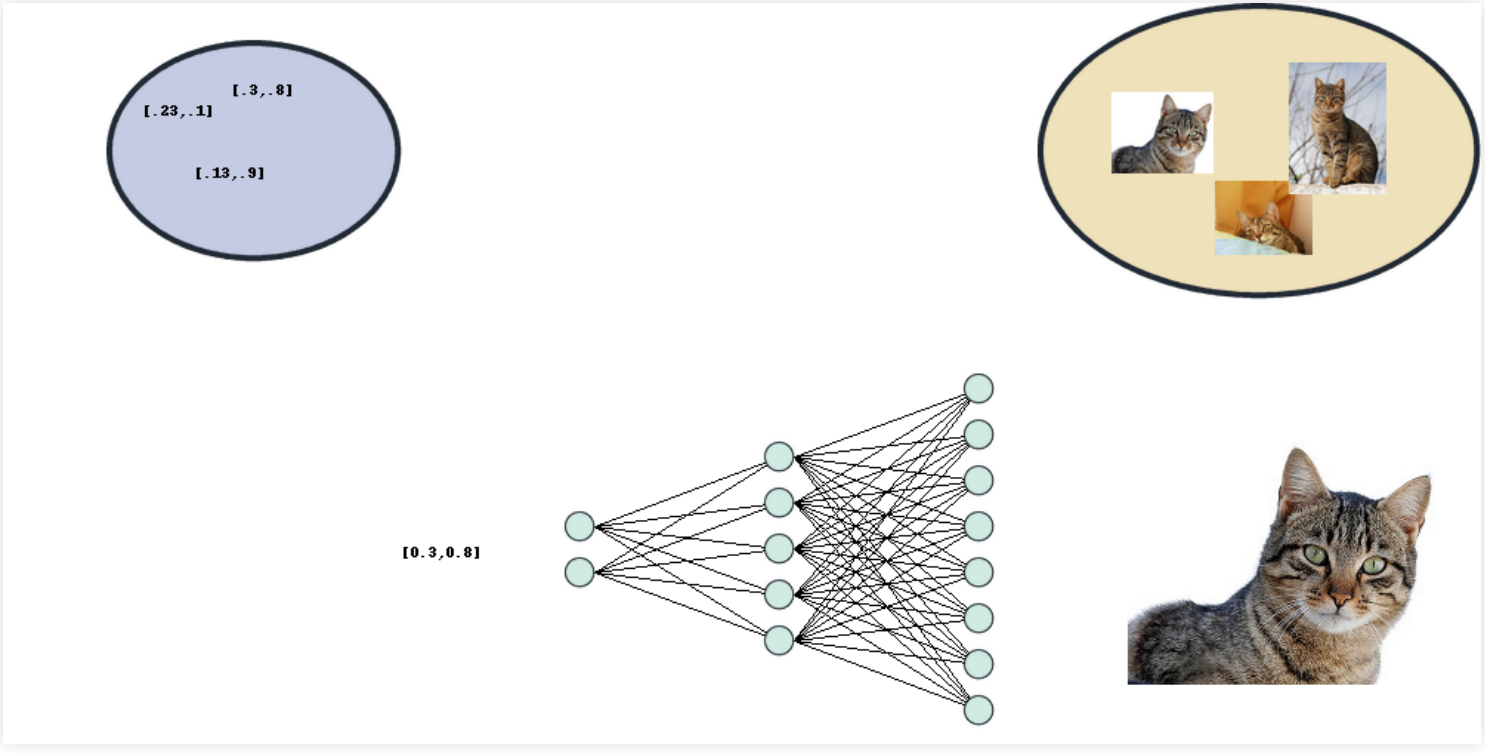


Cat images: Joaquim Alves Gaspar CC-SA

Generating Data

- Can we use the decoder to generate new examples?

- Intuition: we need to sample the right part of the latent space



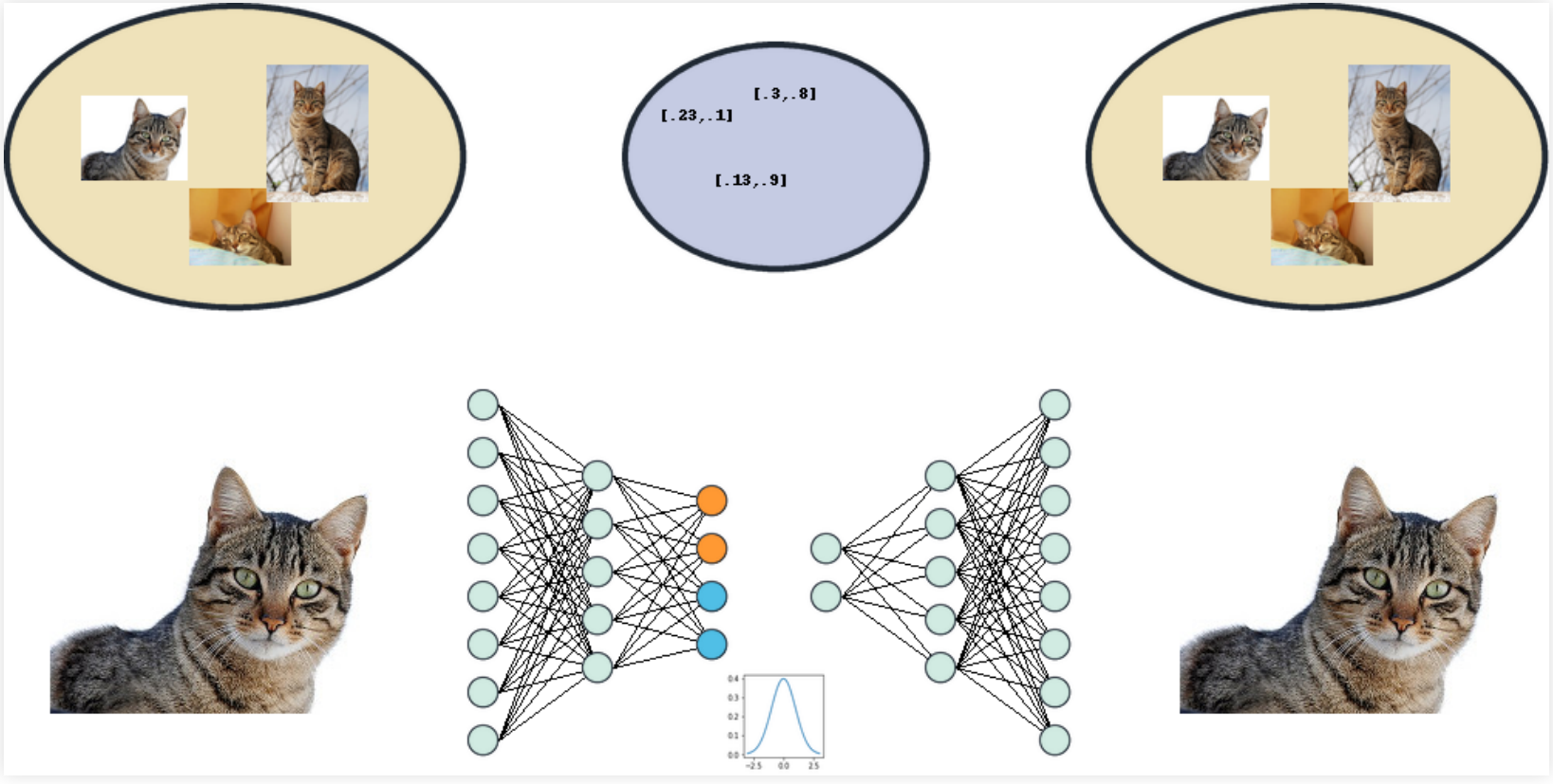
Cat images: Joaquim Alves Gaspar CC-SA

Generating Data

Variational Autoencoders

- Train the autoencoder to encode into a known distribution

- E.g. mixture of independent Gaussians



Variational Autoencoders

- How do we backpropagate through random sampling?
- Reparametrize: z is deterministic apart from a normally distributed error

$$z = \mu + \sigma \odot \epsilon \qquad \epsilon \sim \mathcal{N}(0, 1)$$

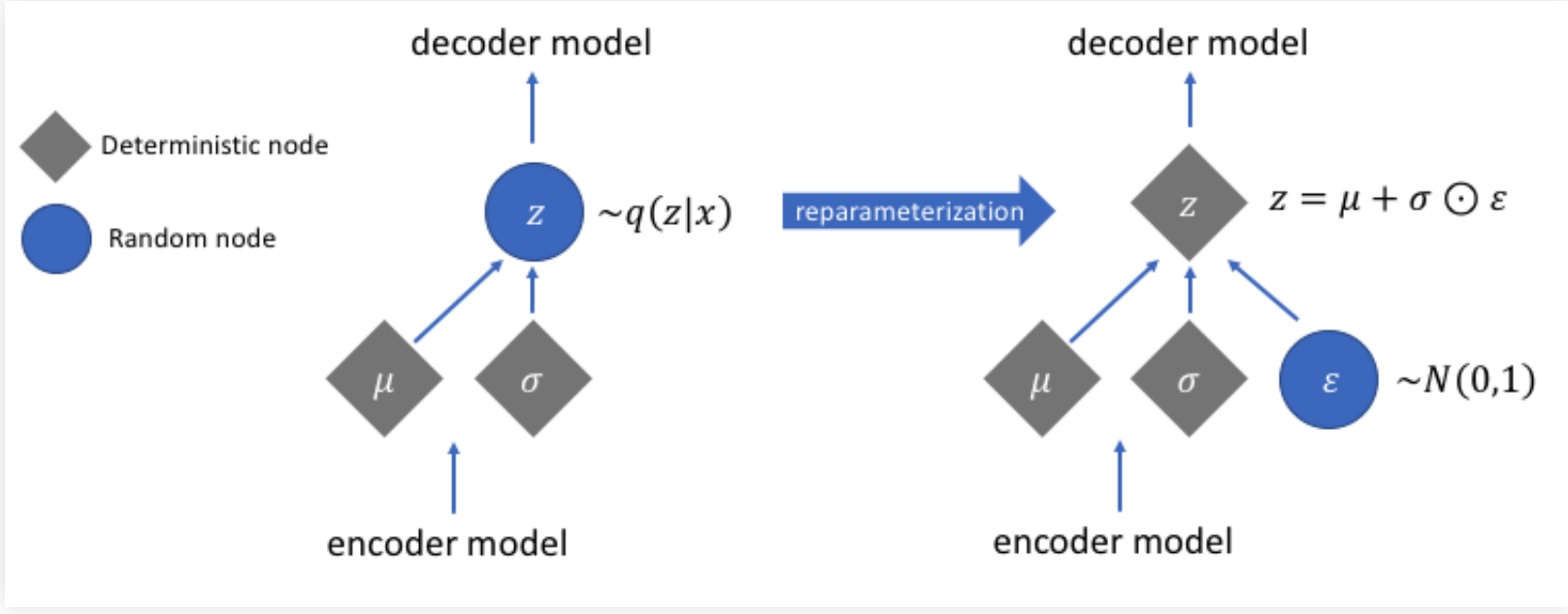


Image: Jeremy Jordan, Variational autoencoders.

Variational Autoencoders

- VAE can learn to disentangle meaningful attributes

- We can force the independence of the latent variables

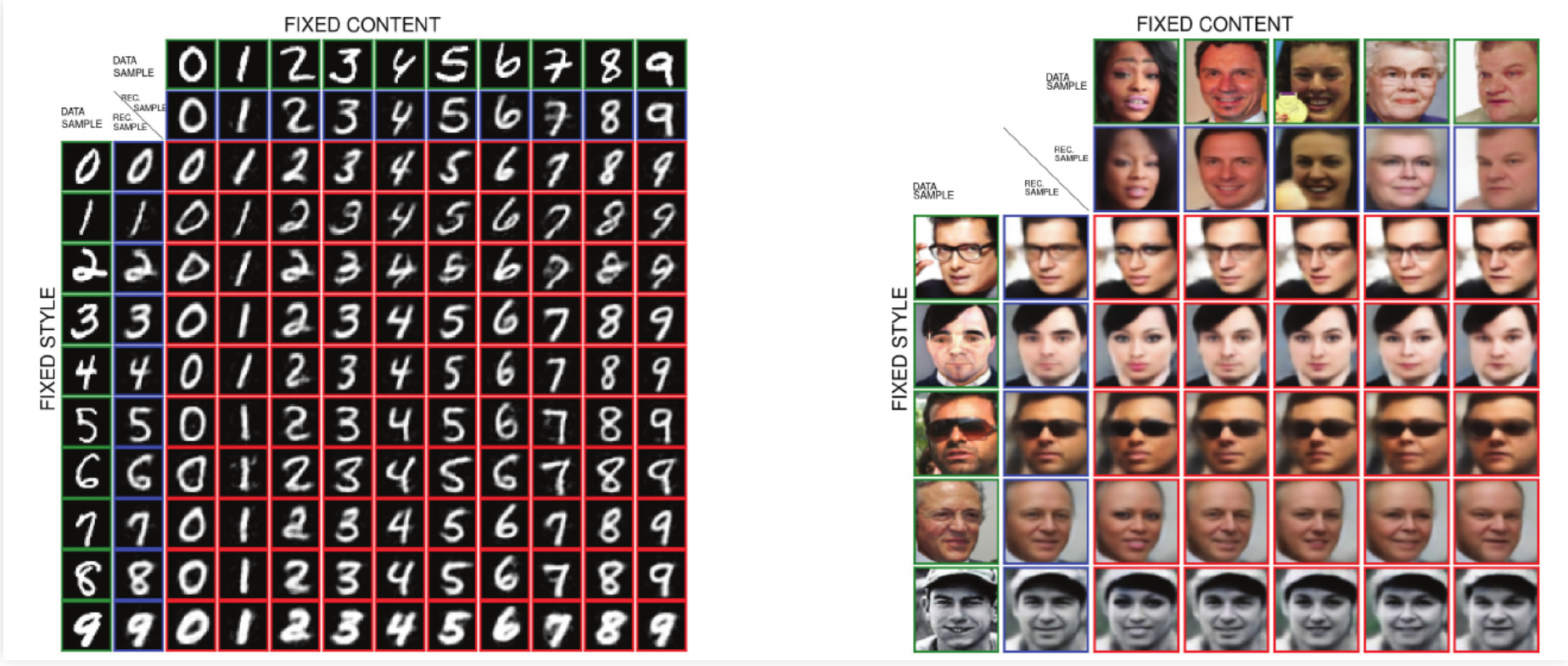


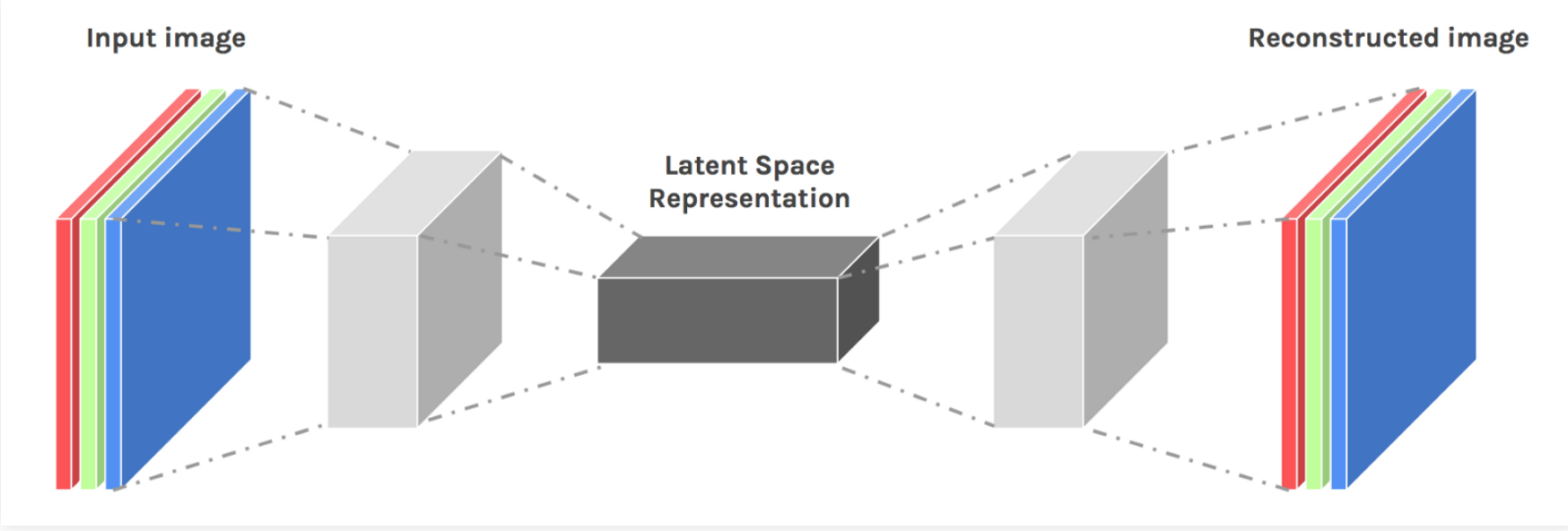
Image: Bouchacourt et. al., Multi-level variational autoencoder, 2018

Convolutional Autoencoders

Convolutional Autoencoders

Use convolutions and "deconvolutions" to reconstruct

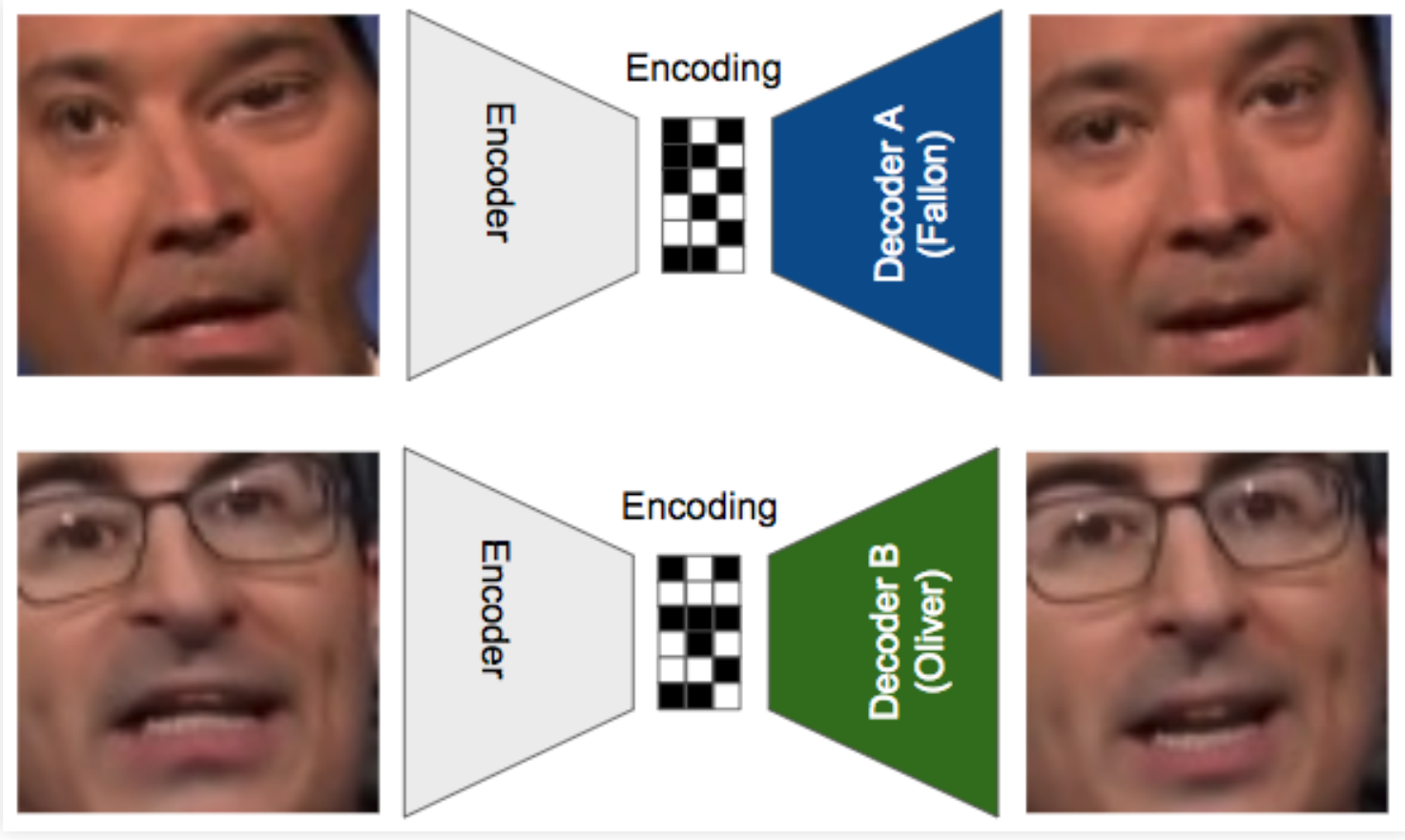
- Latent space is narrow, need to restore original dimensions



Barna Pásztor, Aligning hand-written digits with Convolutional Autoencoders

Convolutional Autoencoders

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

Simple example (with code)

Example

Dimensionality reduction

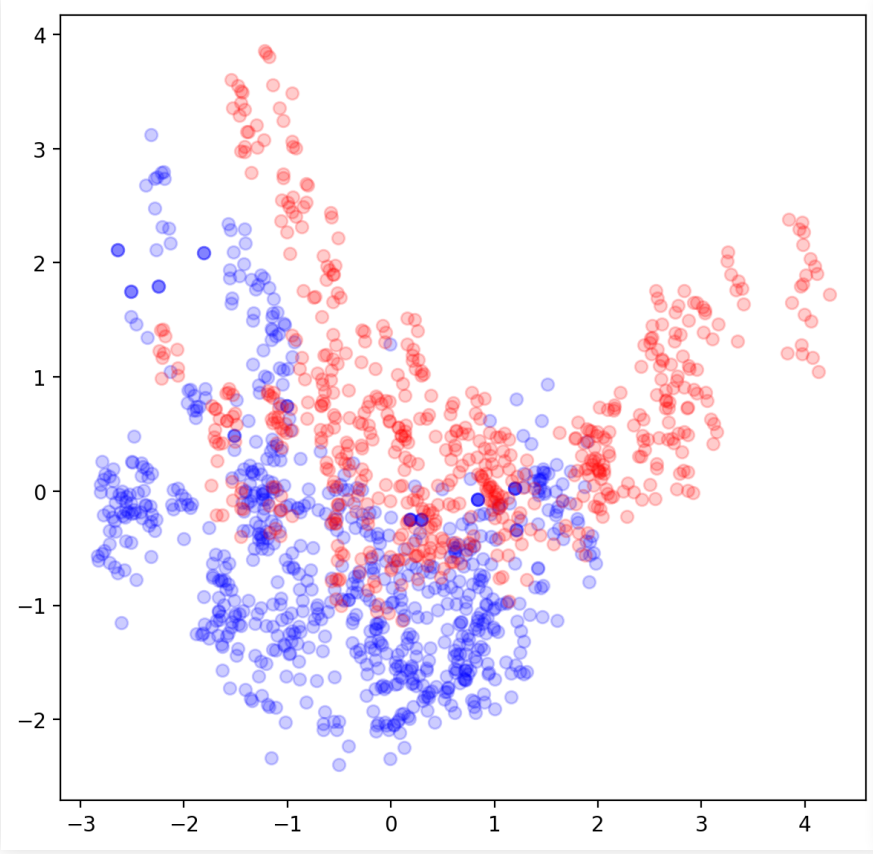
- Banknote authentication data set (UCI ML repository)
- Four features, numerical

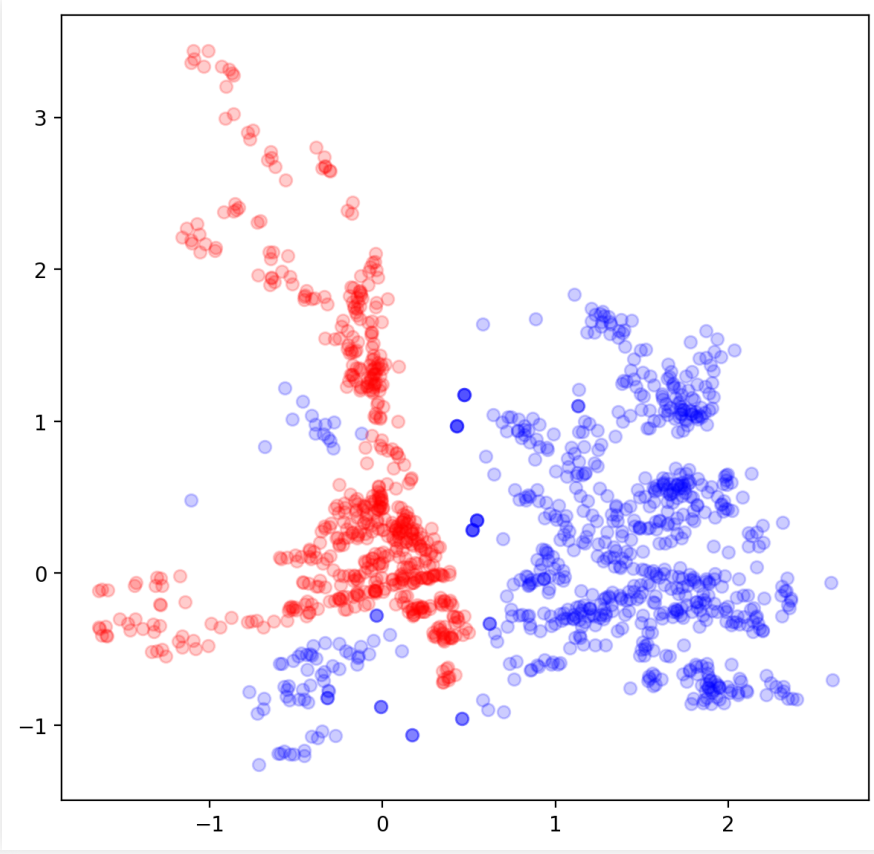
- variance of Wavelet Transformed image (continuous)
- skewness of Wavelet Transformed image (continuous)
- curtosis of Wavelet Transformed image (continuous)
- entropy of image (continuous)

- Two classes
- Autoencoder: 16,8,2,8,16

Example

Dimensionality reduction: PCA vs Autoencoder





Example

Dimensionality reduction with Autoencoder

```
import numpy as np
import tensorflow.keras.backend as K
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Activation,Dense,LeakyReLU

K.clear_session()

def get_data(file_name='TP1-data.csv'):
    """return data standardized"""
    mat = np.loadtxt(file_name,delimiter=',')
    Ys = mat[:,-1]
    Xs = mat[:,:-1]
    means = np.mean(Xs,0)
    stdevs = np.std(Xs,0)
    Xs = (Xs-means)/stdevs
    return Xs.astype(np.float32),Ys
```

Example

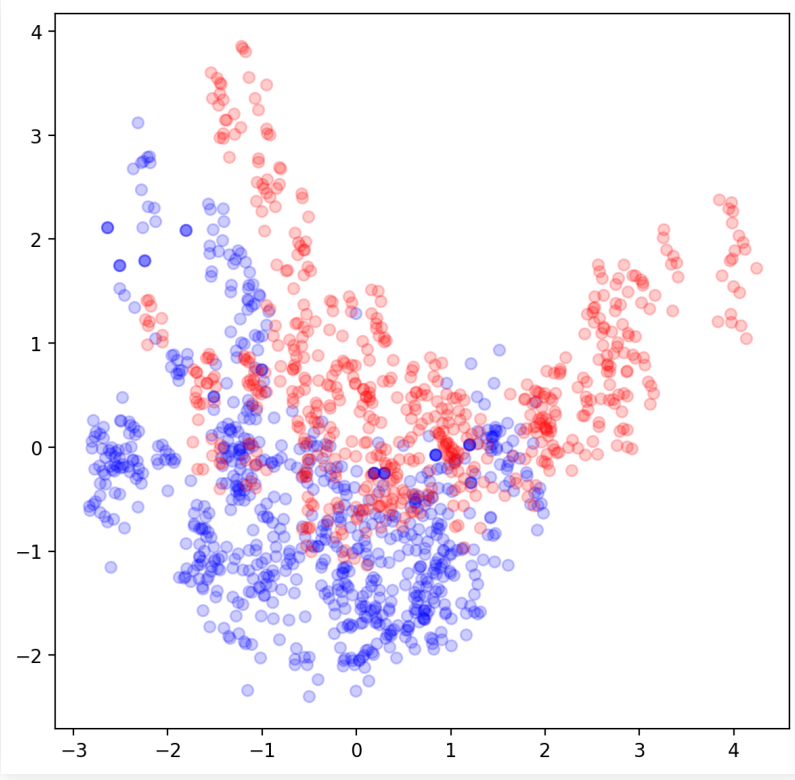
Dimensionality reduction with Autoencoder

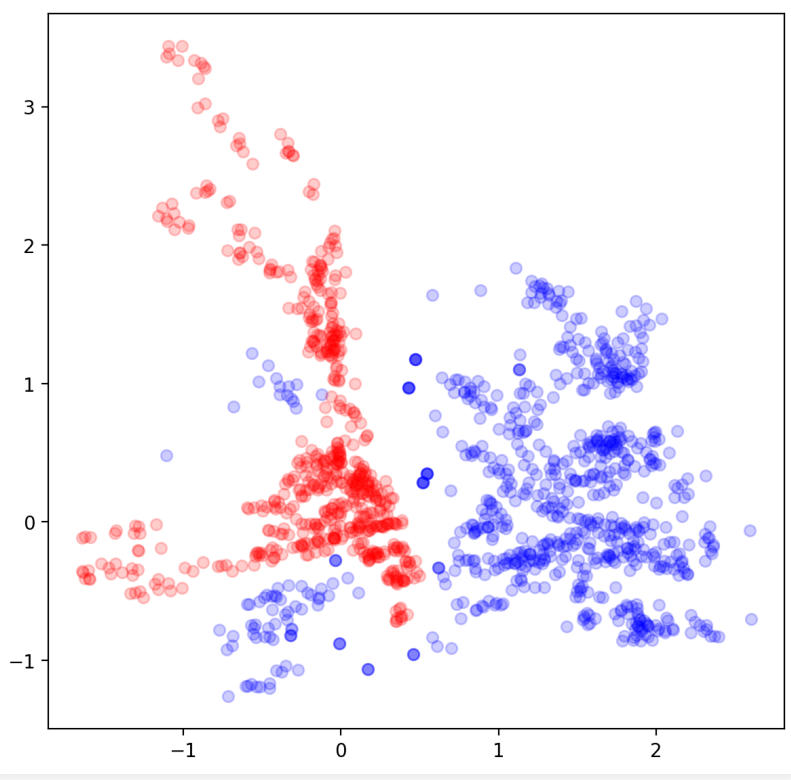
```
def create_model(layer_sizes,input_size):
    features = Input(shape = (input_size,), name = 'input',dtype='float32')
    layer = features
    for size in layer_sizes:
        layer = Dense(size)(layer)
        if size == min(layer_sizes):
            layer = Activation(LeakyReLU(),name ='encoder')(layer)
        else:
            layer = Activation(LeakyReLU())(layer)
    output = Dense(input_size)(layer)
    opt = SGD(0.1,momentum=0.9,decay = 0.1/5000)
    model = Model(inputs=[features], outputs=output)
    model.compile(optimizer=opt, loss='mean_squared_error')
    return model

Xs,Ys = get_data()
layers = [16,8,2,8,16]
model = create_model(layers,4)
model.summary()
model.fit(  [Xs], Xs, epochs=5000, batch_size=128)
```

Example

```
encoder = Model(inputs = model.get_layer('input').output,
                outputs = model.get_layer('encoder').output)
encoder.compile(optimizer='sgd', loss='mse')
encoded = encoder.predict(Xs)
```





Summary

Autoencoders

Summary

- Autoencoders: learn the input in the output
  - No labels required
- Constraints force useful encoding
  - Reduce dimensionality
  - Regularization
  - Denoising
  - Force probability distributions
- Transform or generate data