# This is CS4045!

GCR：dxuxugo

# Unsupervised Learning

“We expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.” - LeCun, Bengio, Hinton, Nature 2015

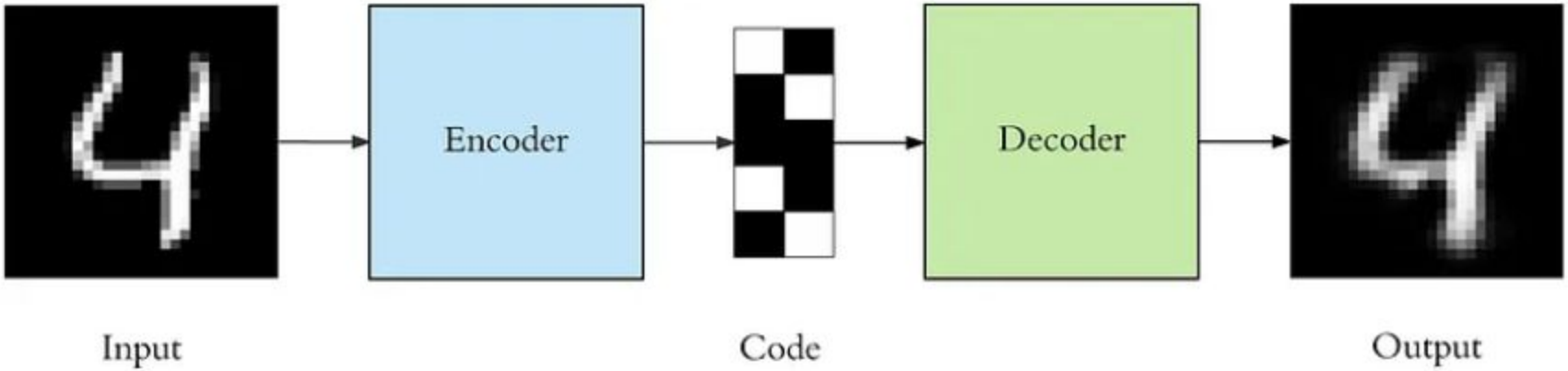
If intelligence was a cake, unsupervised learning would be the cake, supervised  learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake. - Yann LeCun, March 14, 2016 (Facebook)

# AutoEncoders

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output.

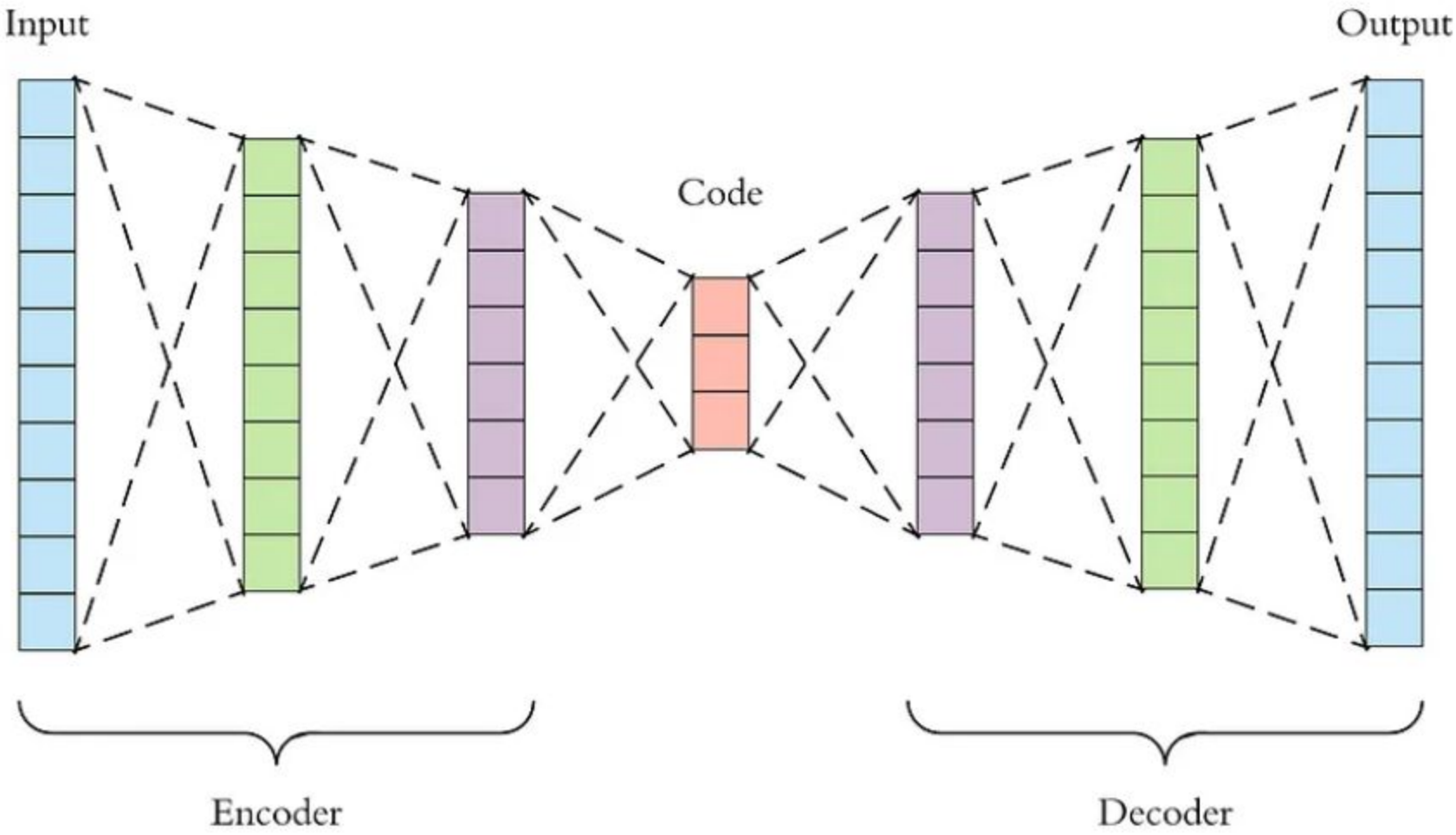
An autoencoder consists of 3 components: encoder, code and decoder.

Autoencoders are mainly a dimensionality reduction (or compression) algorithm.


Encoder
Decoder
Input
Code
Output

# Unsupervised????

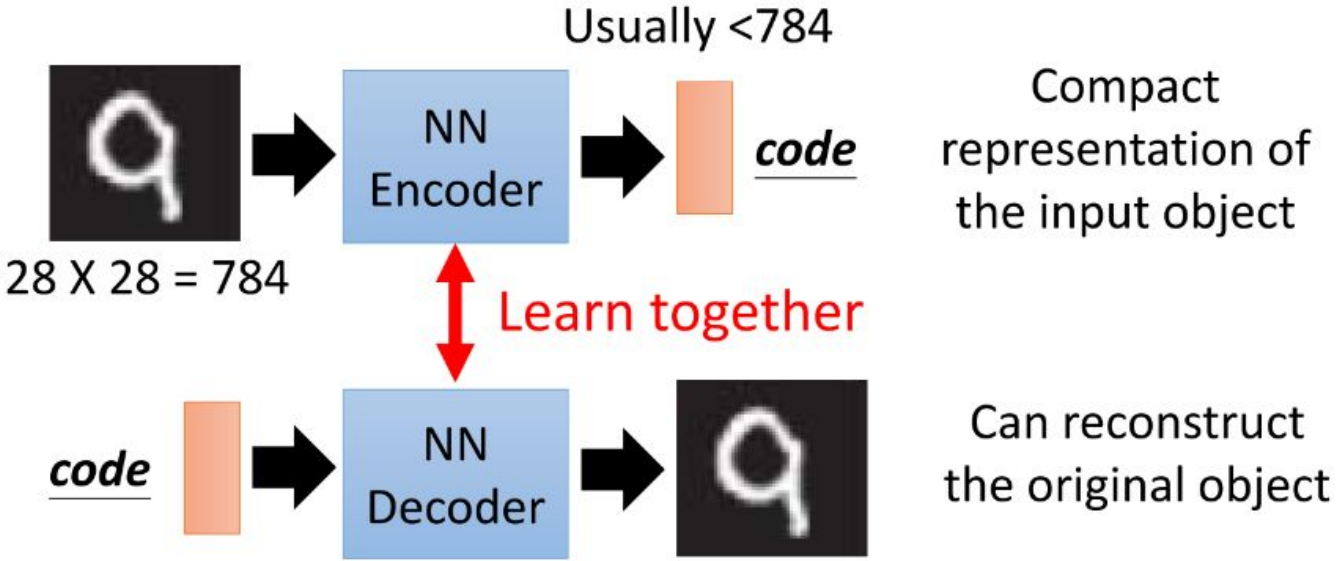
Autoencoders are considered an unsupervised learning technique since they don’t need explicit labels to train on. But to be more precise they are self-supervised because they generate their own labels from the training data.


Input
Output
Code
Encoder
Decoder

There are 4 hyperparameters that we need to set before training an autoencoder:

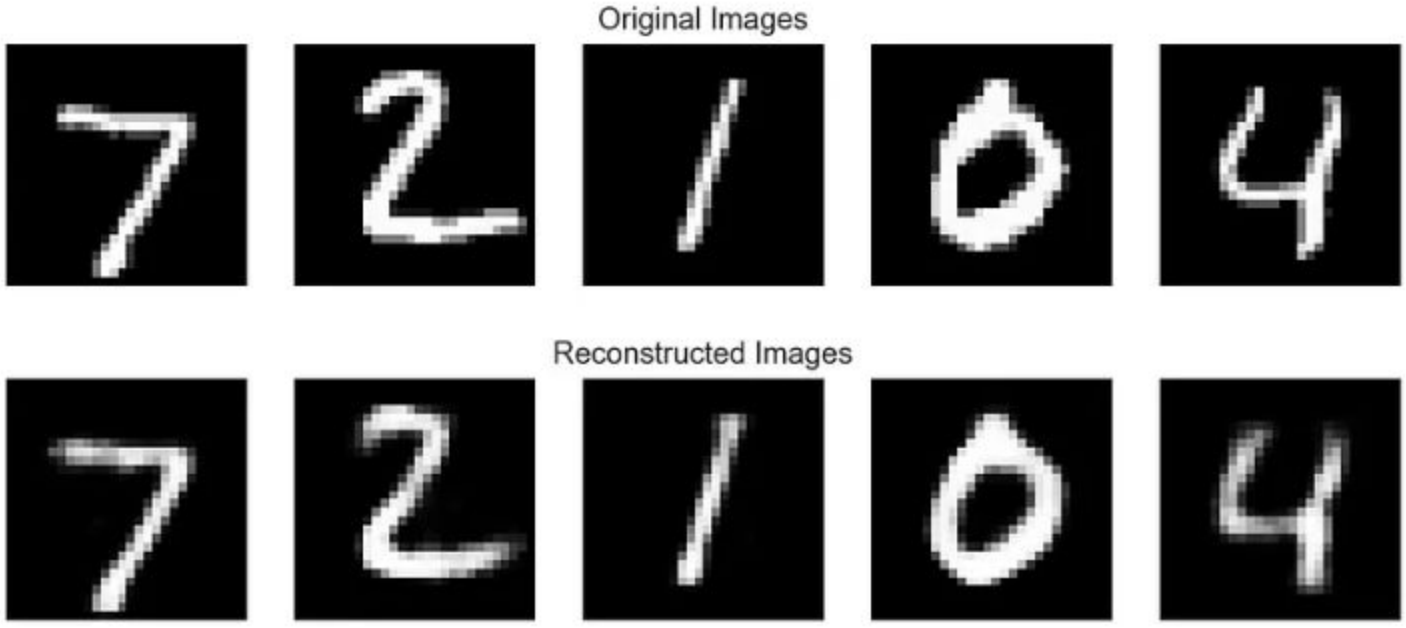
1. Code size
2. Number of layers
3. Number of nodes per layer
4. Loss function

# AutoEncoders



# Implementation

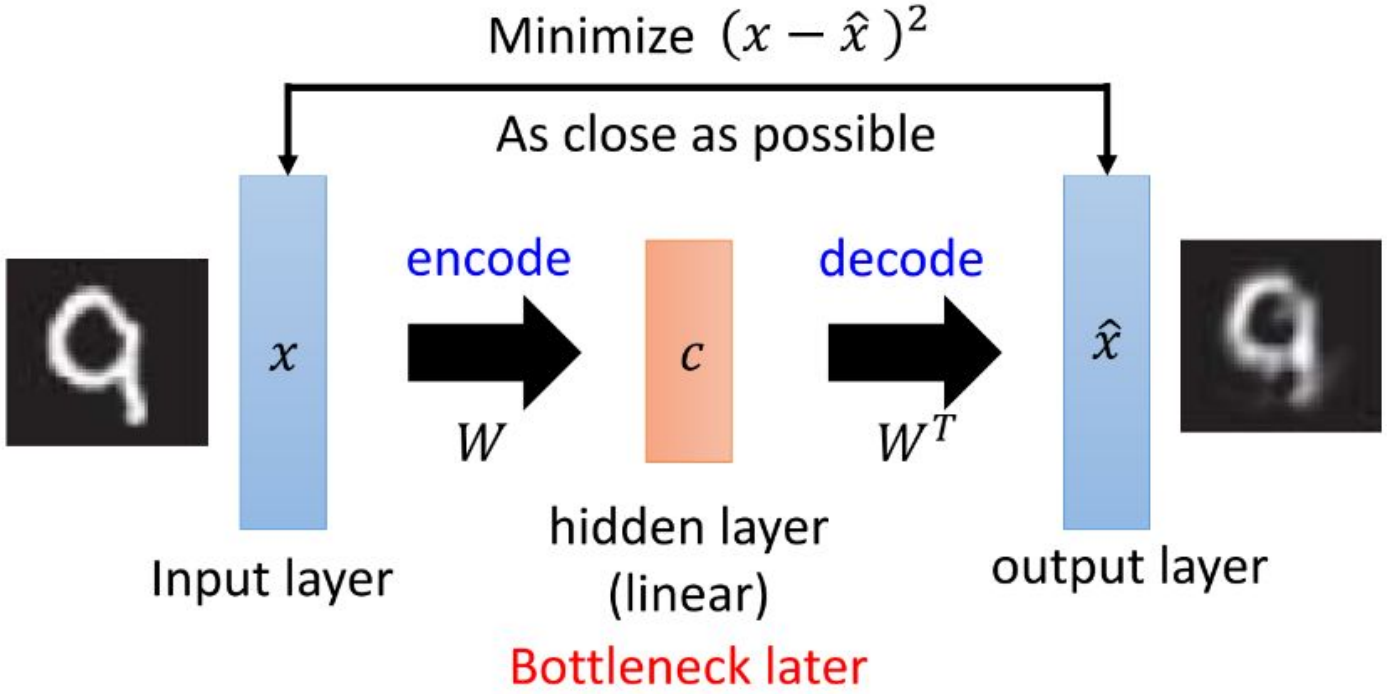
```
input_size = 784

hidden_size = 128

code_size = 32


input_img = Input(shape=(input_size,))

hidden_1 = Dense(hidden_size, activation='relu')(input_img)

code = Dense(code_size, activation='relu')(hidden_1)

hidden_2 = Dense(hidden_size, activation='relu')(code)

output_img = Dense(input_size, activation='sigmoid')(hidden_2)


autoencoder = Model(input_img, output_img)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

autoencoder.fit(x_train, x_train, epochs=5)
```


Original Images
Reconstructed Images

# A bit of Advice

We should be careful to not make it too powerful. Otherwise the autoencoder will simply learn to copy its inputs to the output, without learning any meaningful representation. It will just mimic the identity function. The autoencoder will reconstruct the training data perfectly, but it will be overfitting without being able to generalize to new instances, which is not what we want.

# Recap: PCA

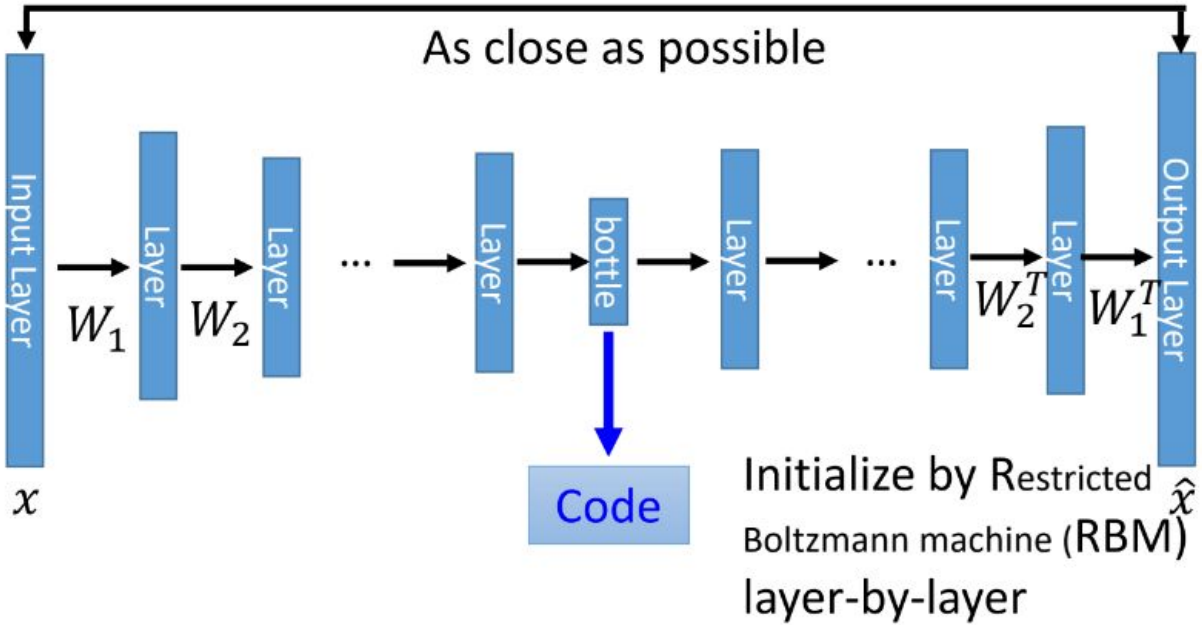



Output of the hidden layer is the code

# Deep Auto-encoder

Symmetric is not necessary.

- Of course, the auto-encoder can be deep



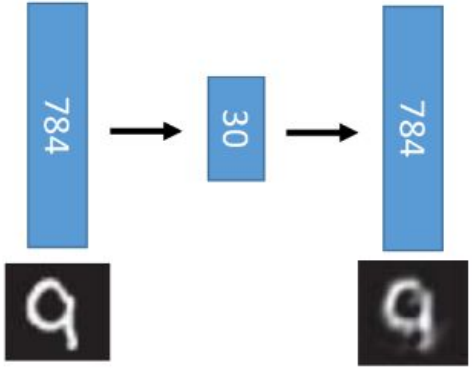

Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

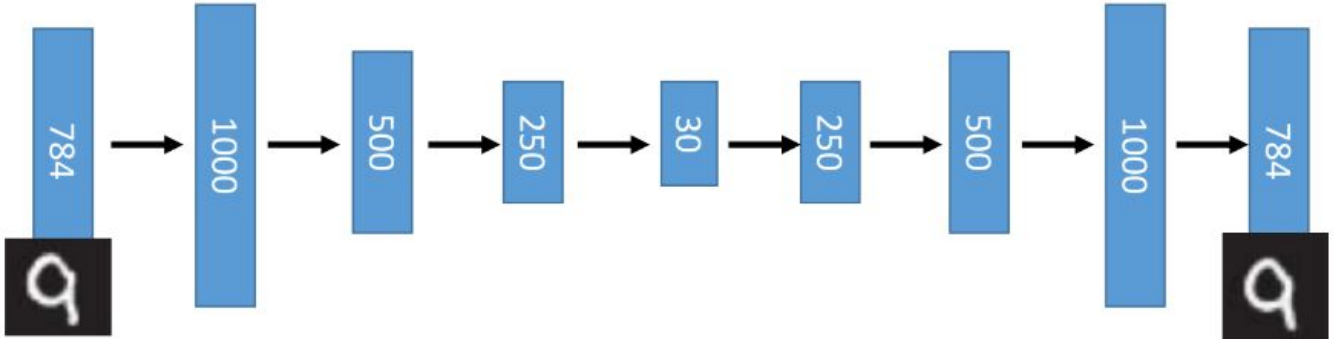
# Deep Auto-encoder

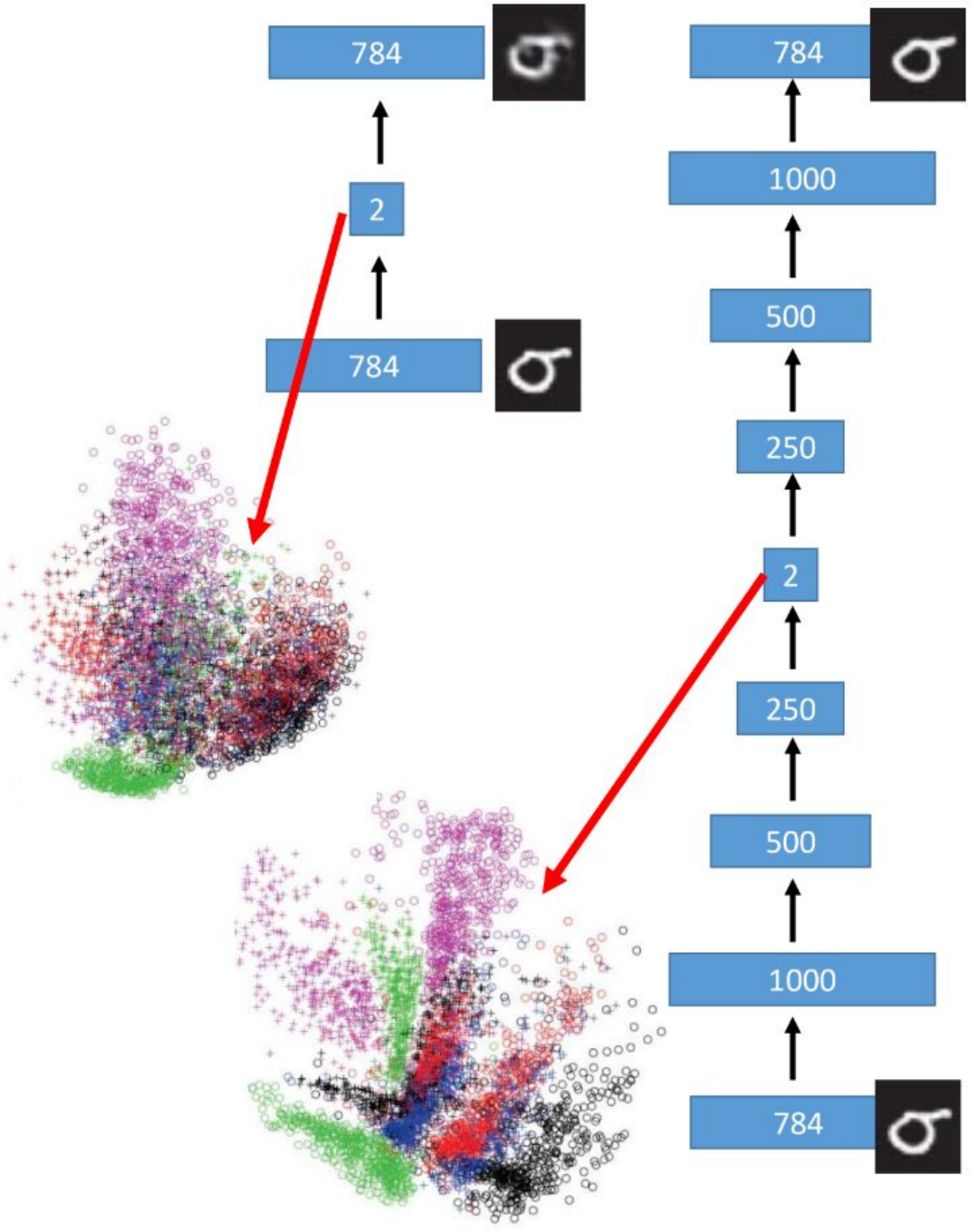
Original Image

PCA

Deep Auto-encoder



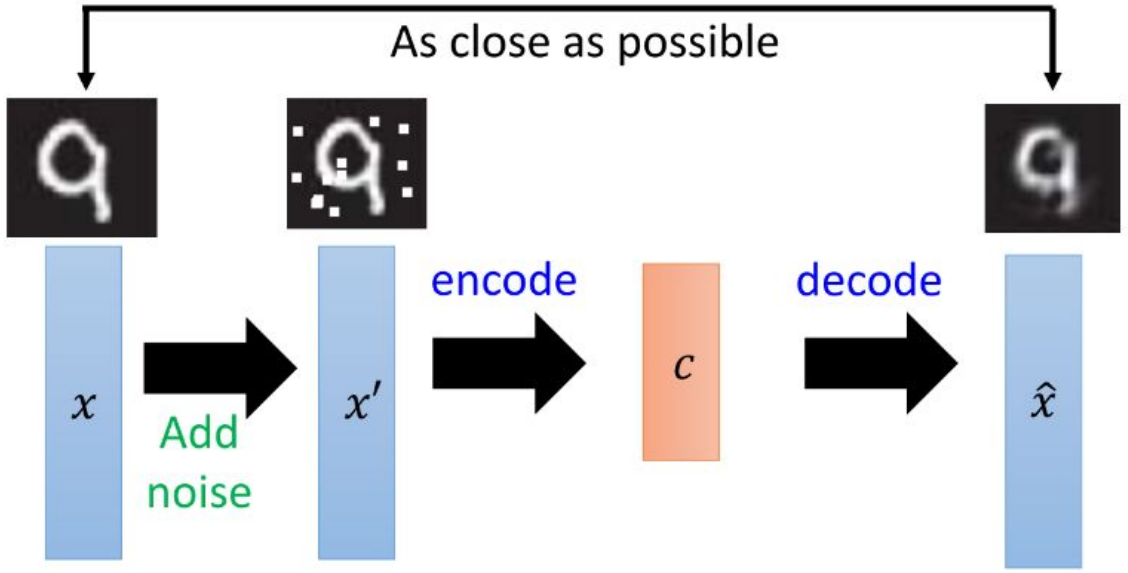

784
2
784
784
1000
500
250
2
250
500
1000
784

More: Contractive auto-encoder

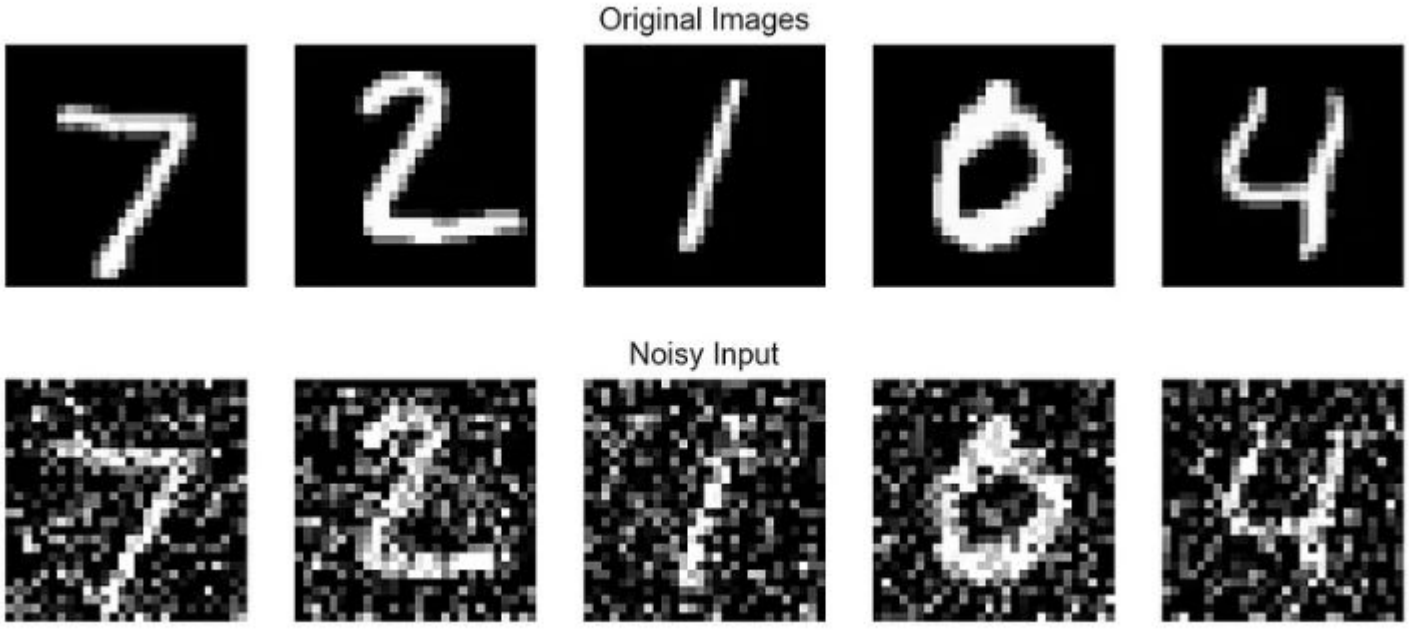
# Auto-encoder

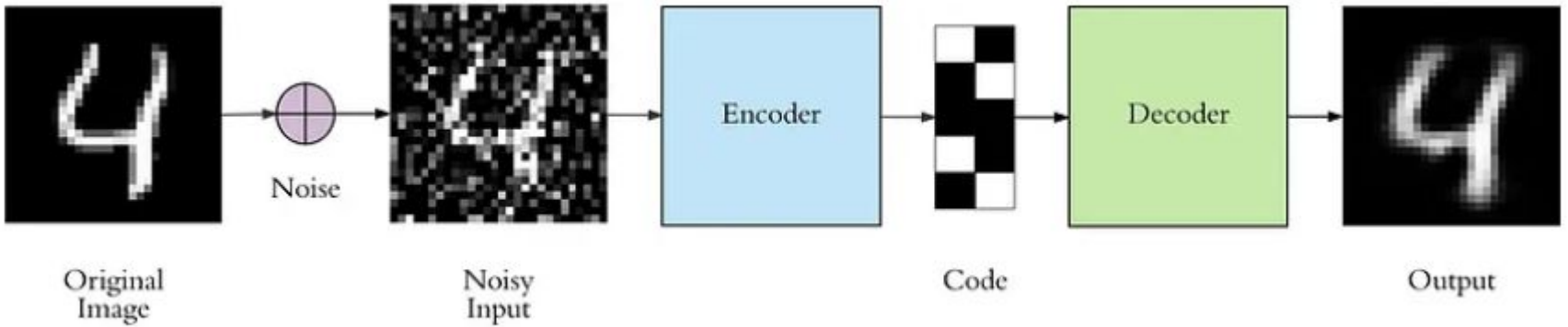
Ref: Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction." *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011.

- De-noising auto-encoder




Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.


Original Images
Noisy Input




Denoising autoencoder is trained as:

```
autoencoder.fit(x_train_noisy, x_train)
```

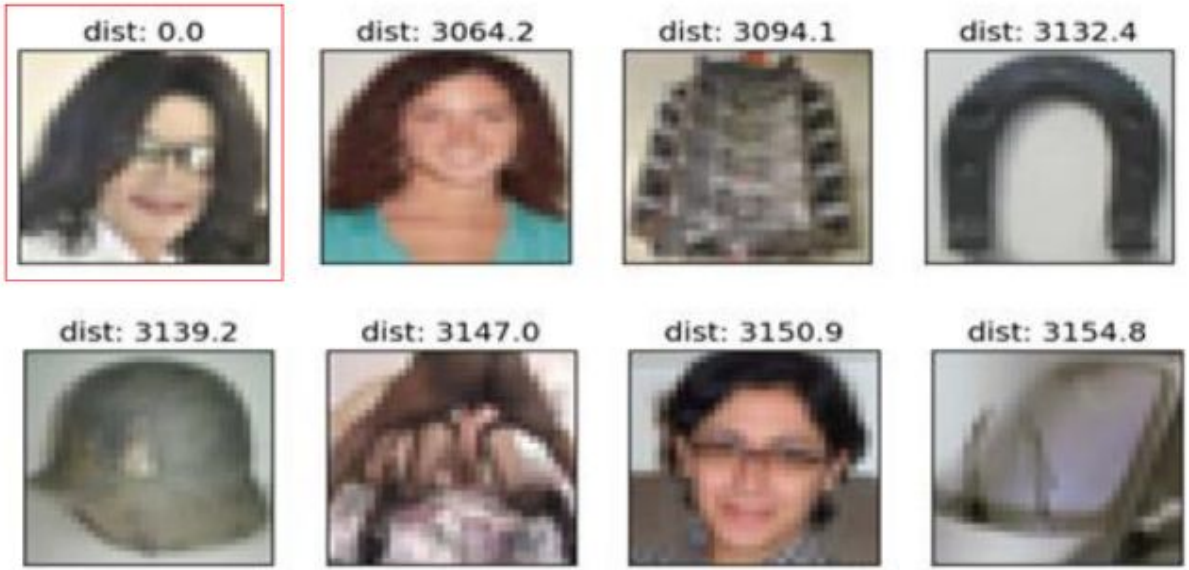
# Auto-encoder – Similar Image Search

Retrieved using Euclidean distance in pixel intensity space

(Images from Hinton's slides on Coursera)

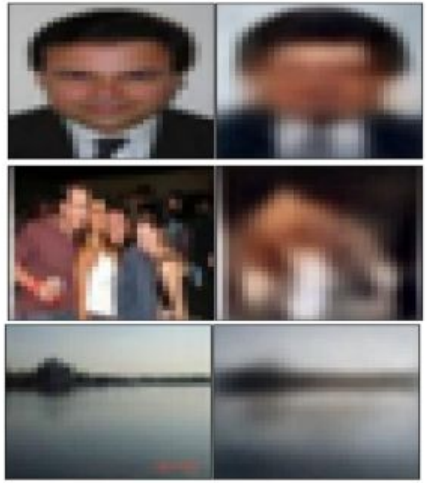
Reference: Krizhevsky, Alex, and Geoffrey E. Hinton. "Using very deep autoencoders for content-based image retrieval." *ESANN*. 2011.

Retrieved using Euclidean distance in pixel intensity space




retrieved using 256 codes

# Auto-encoder – Similar Image Search



32x32

8192

4096

2048

1024

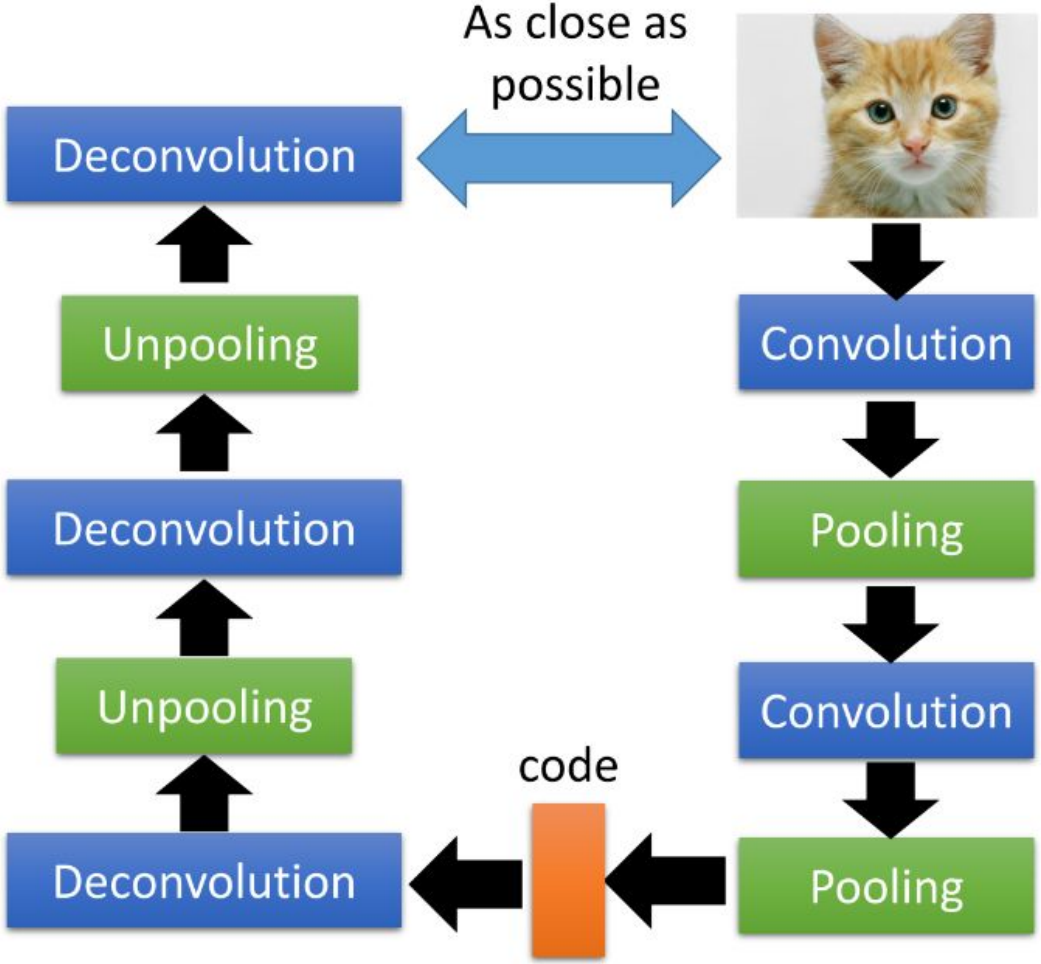
512

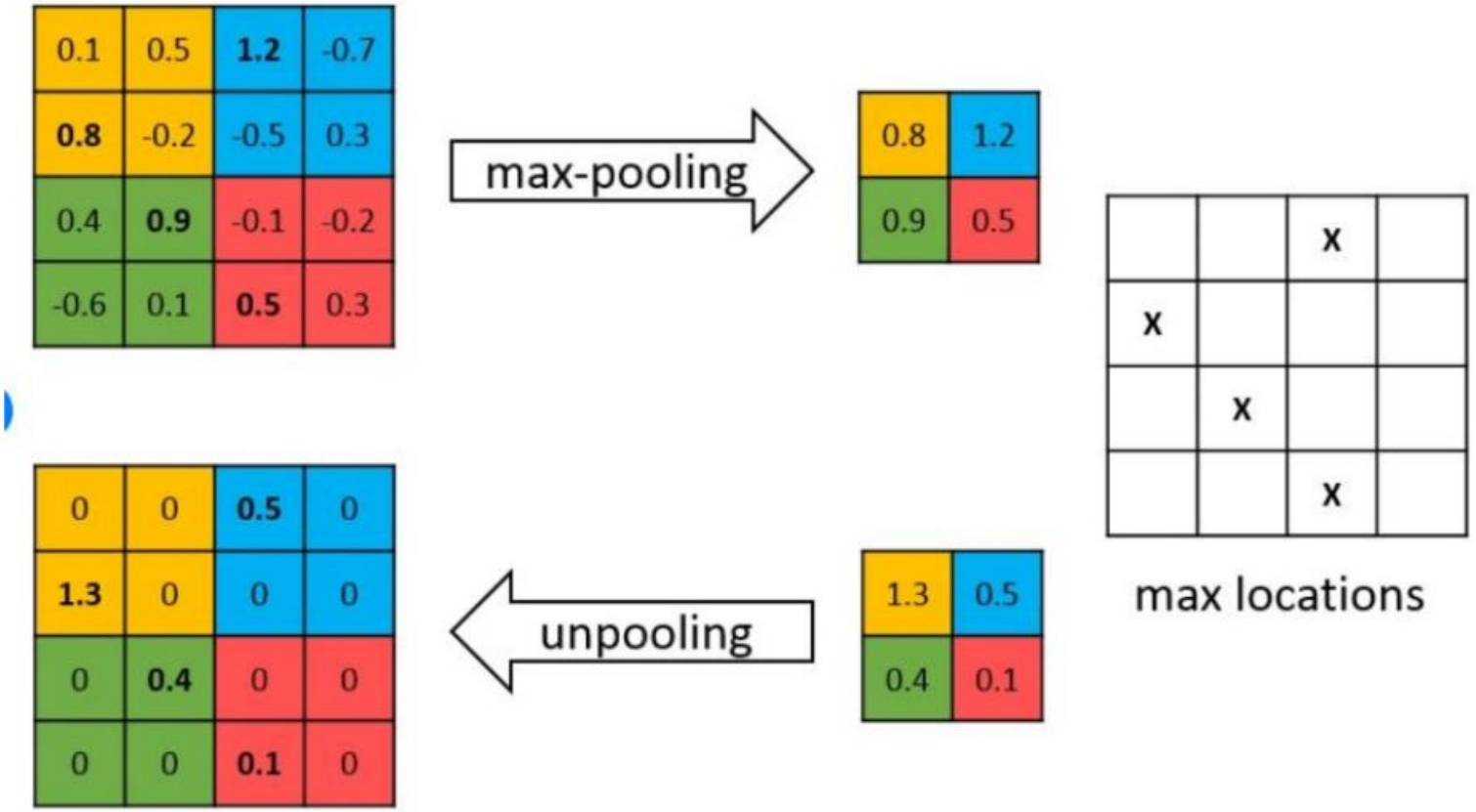
256

code

(crawl millions of images from the Internet)

# Auto-encoder for CNN

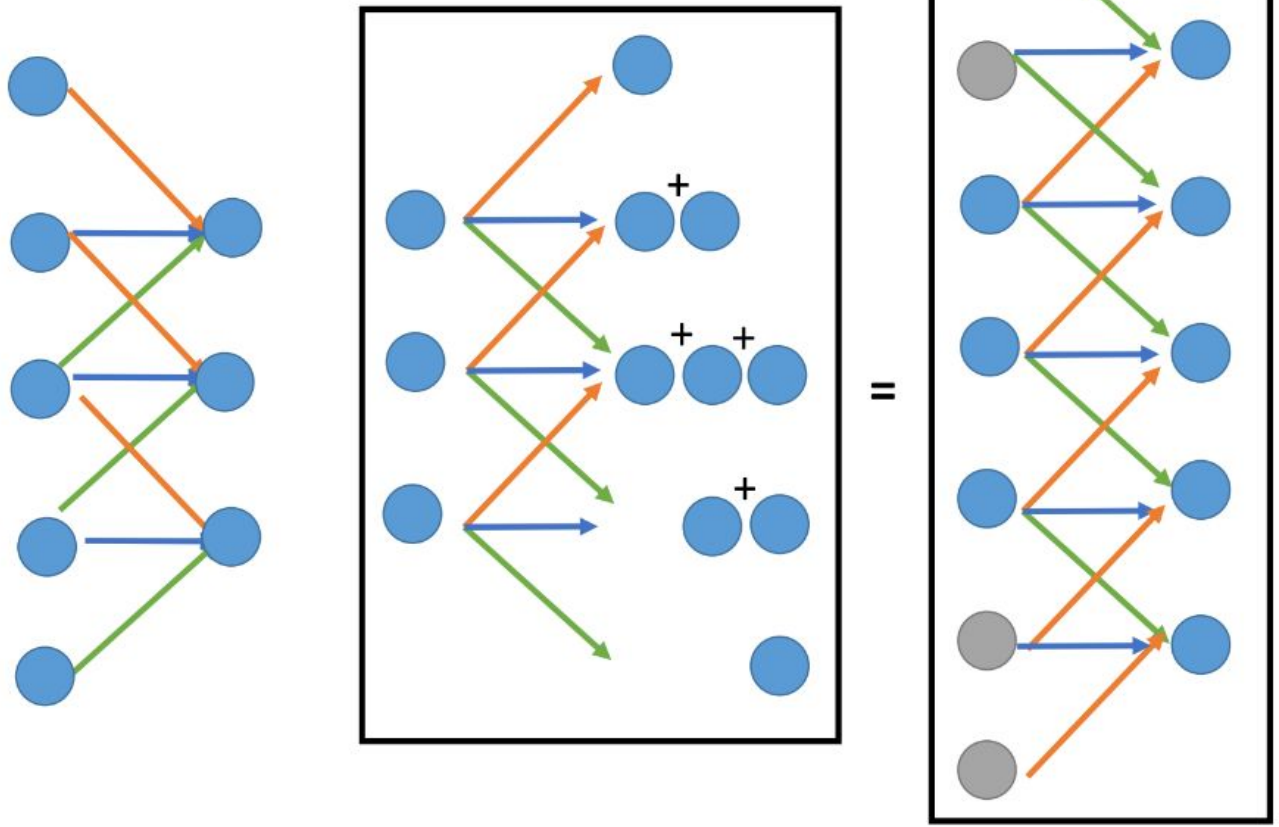





Pooling and unpooling layers. For each pooling layer, the max locations are stored. These locations are then used in unpooling layer.

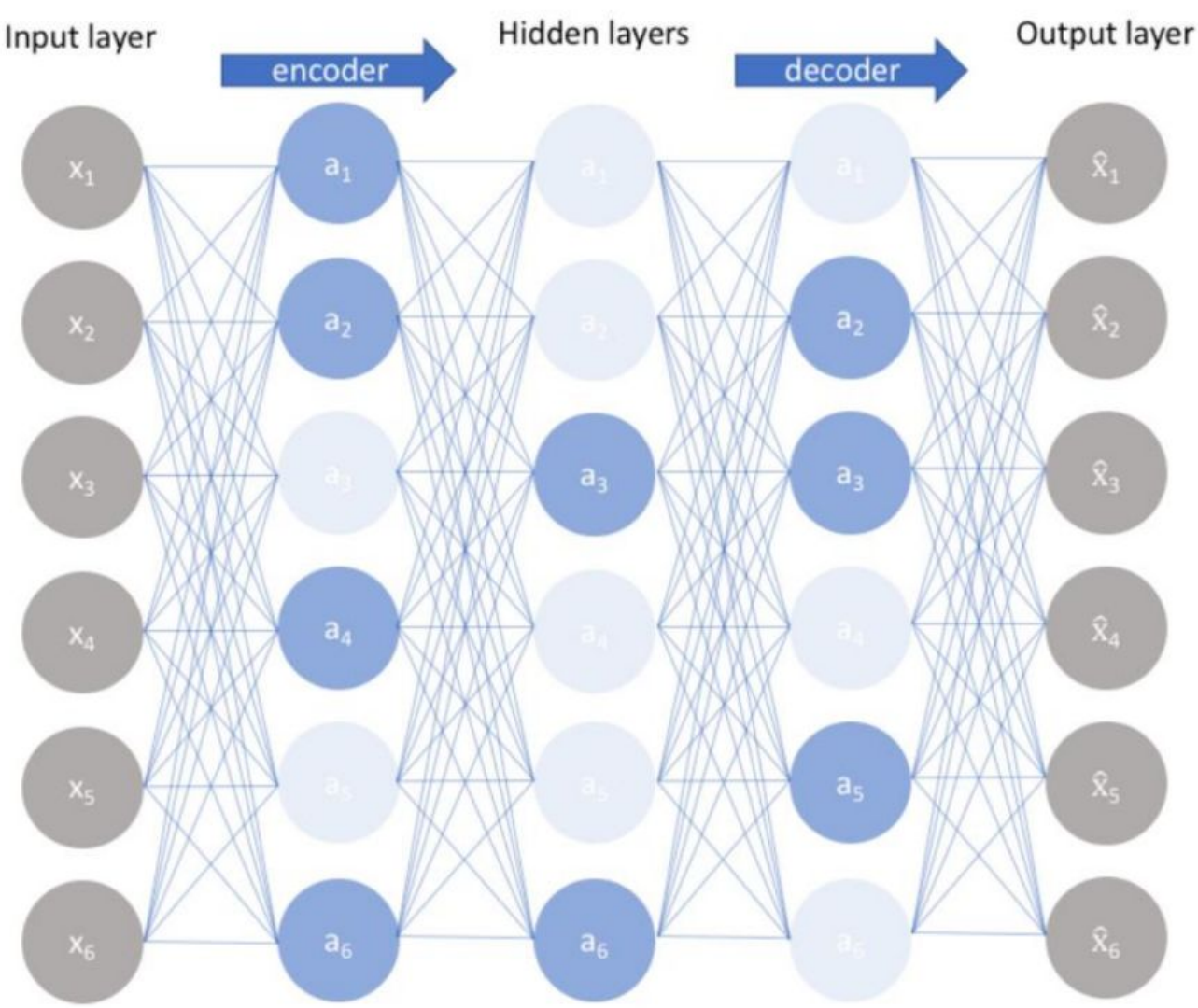
# CNN
# - Deconvolution

Actually, deconvolution is convolution.

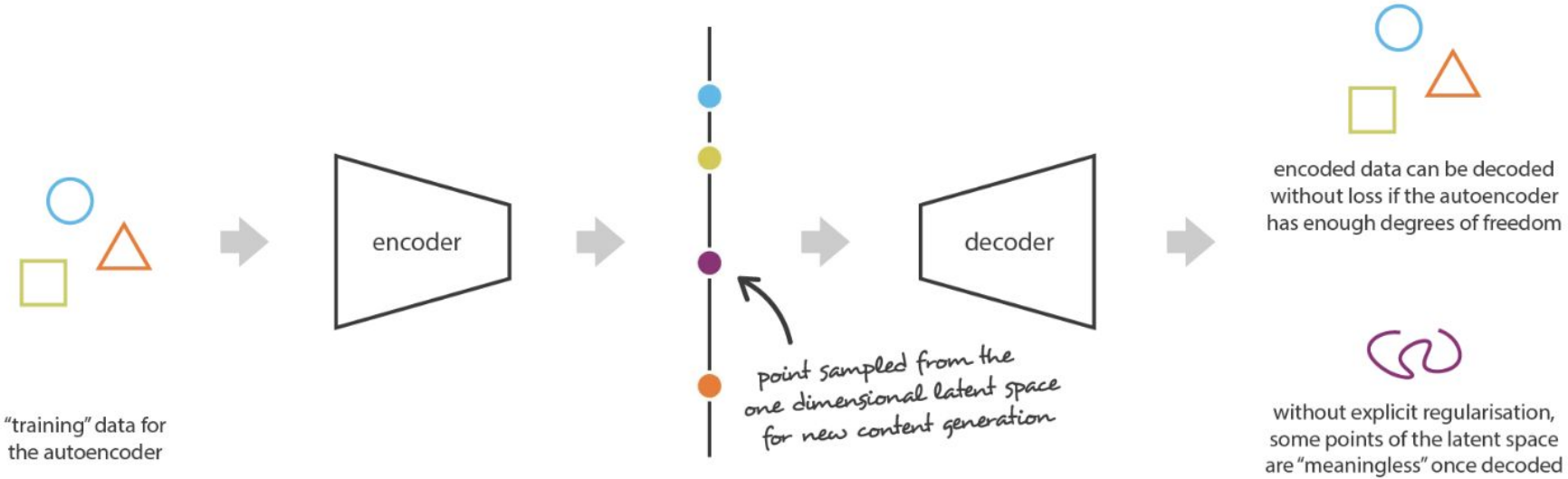


# Sparse AutoEncoder

- Use of Regularization (For better generalization)
- Introducing sparsity constraint such that only a fraction of the nodes would have nonzero values called active nodes
- Add a penalty term to the loss function such that only a fraction of nodes become active
- This forces the autoencoder to represent each input as a combination of small number of nodes and demands to discover interesting structure in the data


Input layer
encoder
Hidden layers
decoder
Output layer
x1
x2
x3
x4
x5
x6
a1
a2
a3
a4
a5
a6
a1
a2
a3
a4
a5
a6
a1
a2
a3
a4
a5
a6
x̂1
x̂2
x̂3
x̂4
x̂5
x̂6

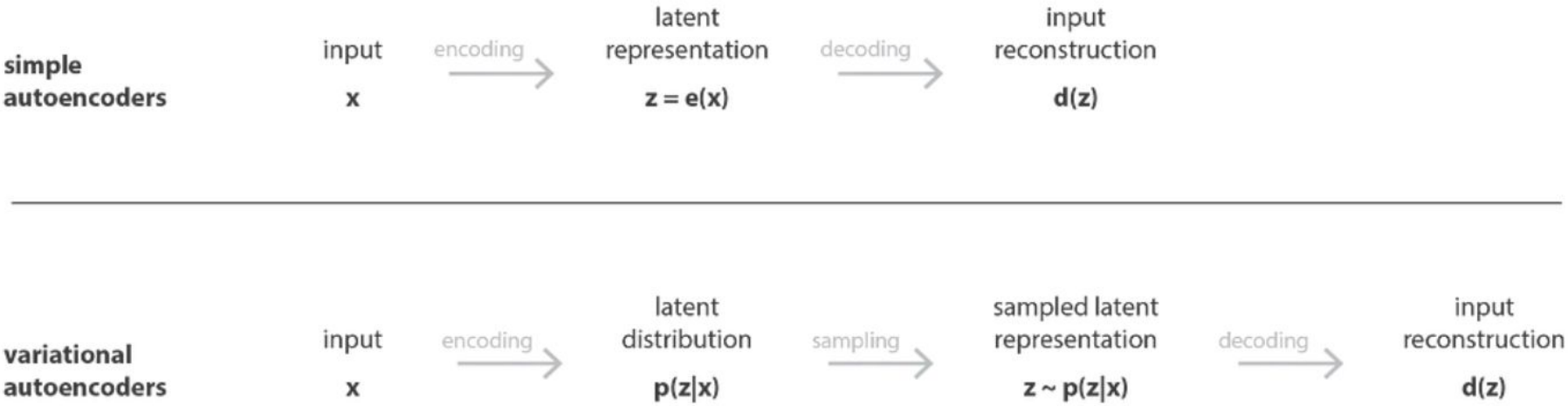
# VAE

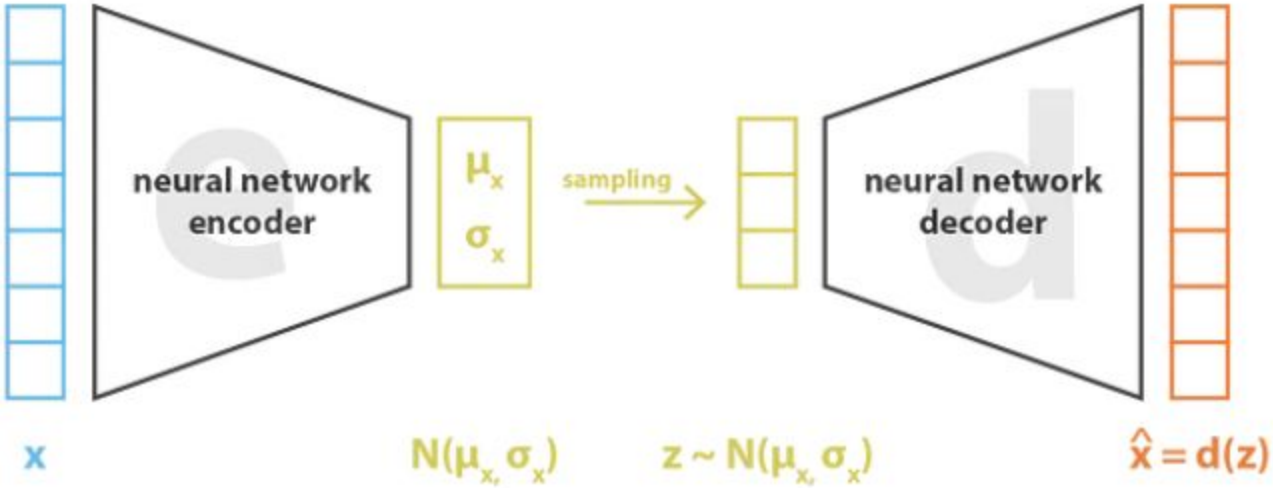



Irregular latent space prevent us from using autoencoder for new content generation.

# VARIATIONAL AUTOENCODER

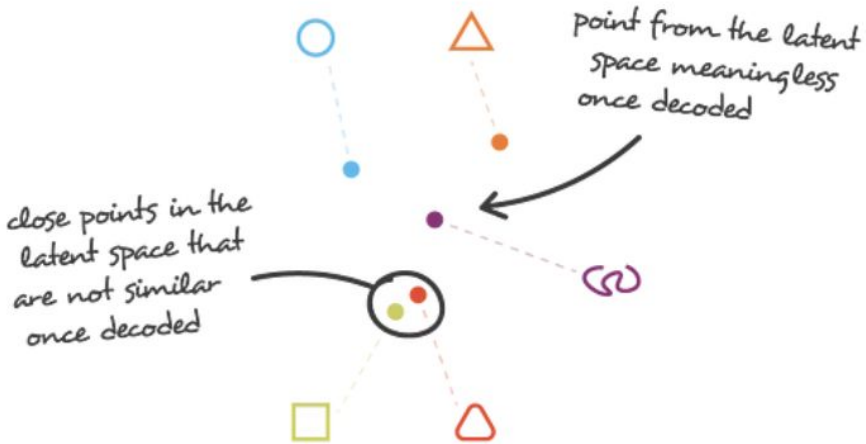
A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.


simple autoencoders
input
x
encoding
latent representation
z = e(x)
decoding
input reconstruction
d(z)
variational autoencoders
input
x
encoding
latent distribution
p(z|x)
sampling
sampled latent representation
z ~ p(z|x)
decoding
input reconstruction
d(z)

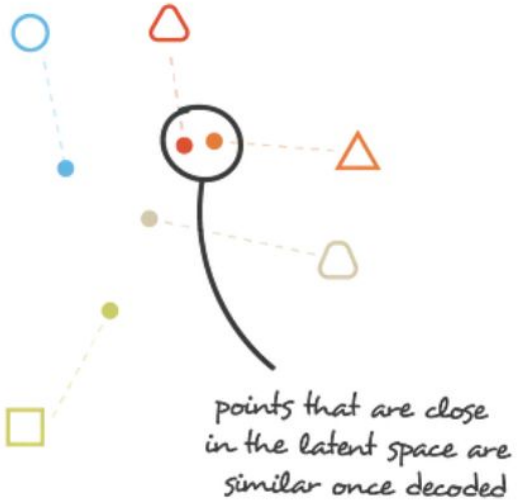



---

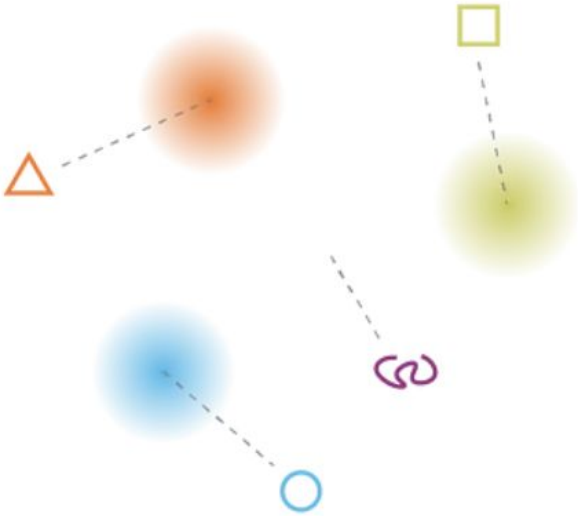
$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

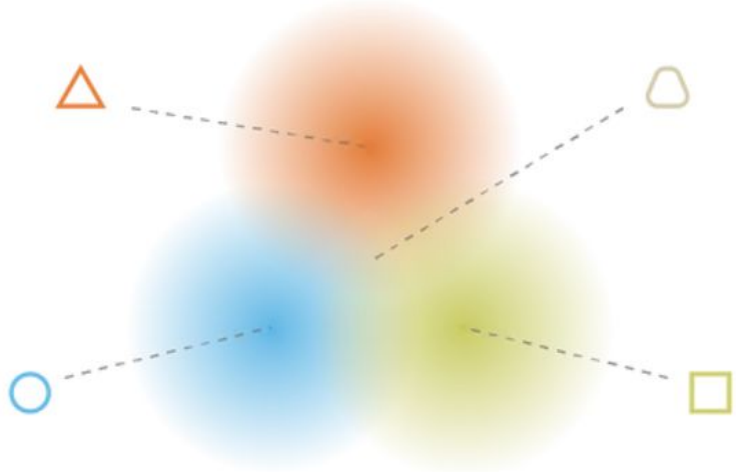



irregular latent space




regular latent space





**what can happen without regularisation**

**what we want to obtain with regularisation**

The returned distributions of VAEs have to be regularised to obtain a latent space with good properties.

# References

https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

# Thank You

# Let's Remember Each Other in Our Prayers :)

PICTURE
TIME

# This is CS4045!
# Signing Off

**GCR: dxuxugo**