AUTOENCODERS

RESEARCHGROUP
IPRODAM3D

Cristian López Del Alamo

2022

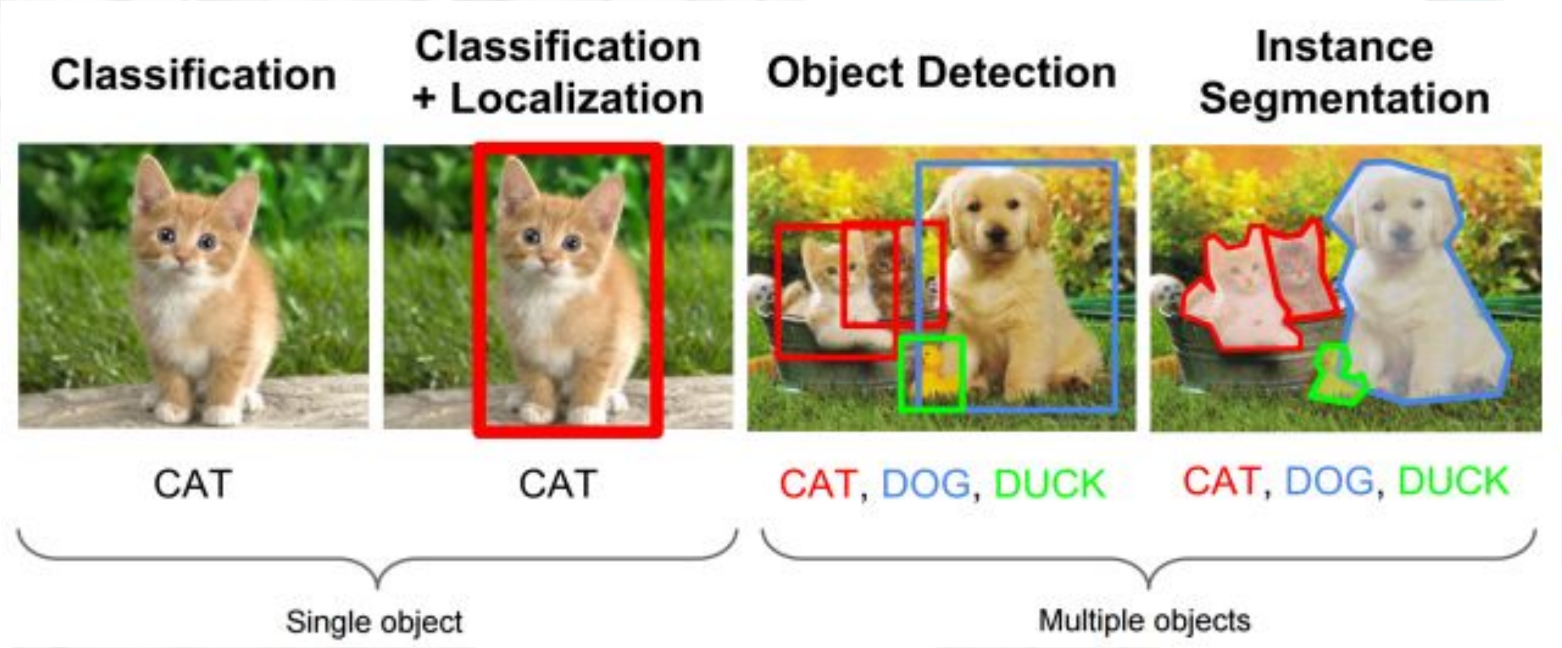
Introduction

- Objetivo
- Autoencoders
- Regularized Autoencoders
- Sparse Autoencoders
- Denoising Autoencoders

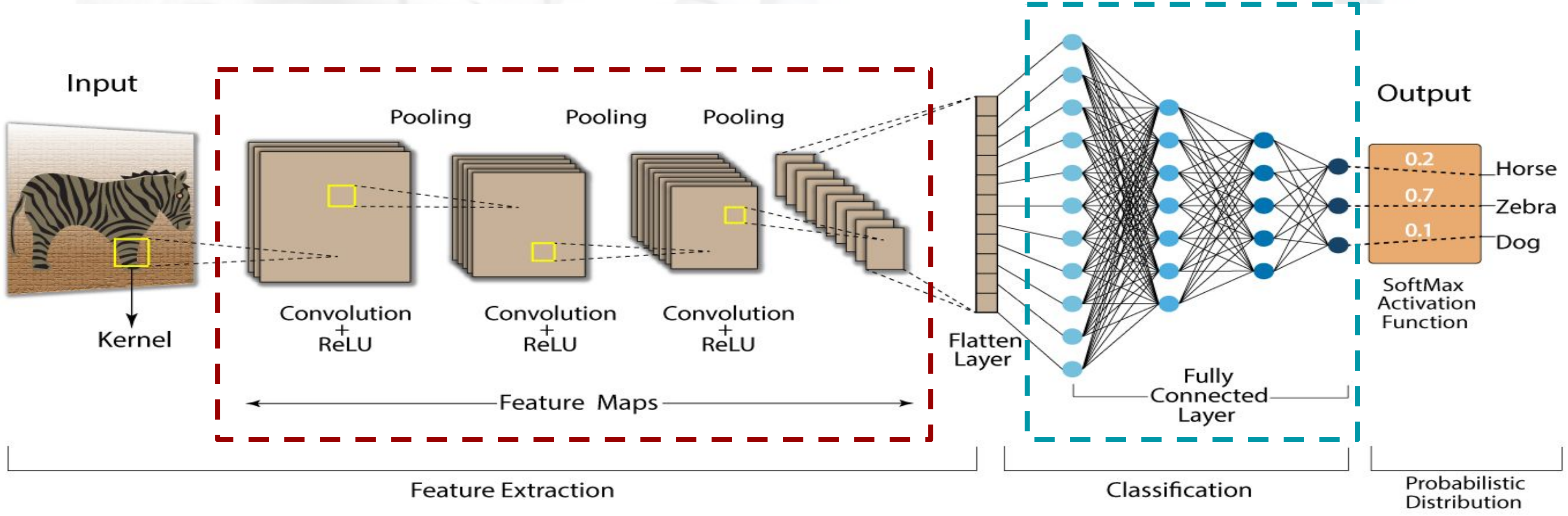
Objetivo

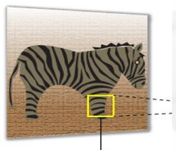
Entender el concepto, estructura y aplicaciones de los autoencoders

Different data sources : Images

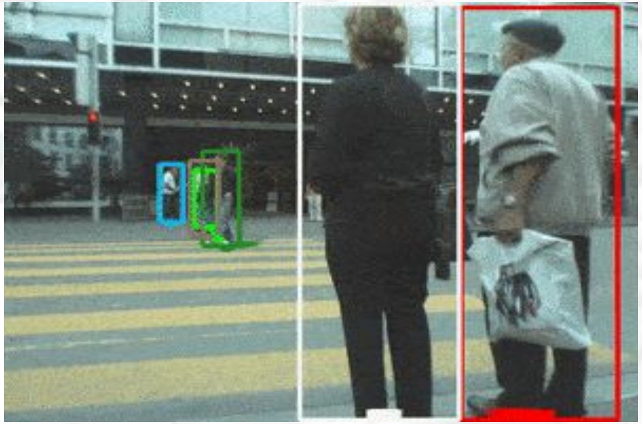


CNN

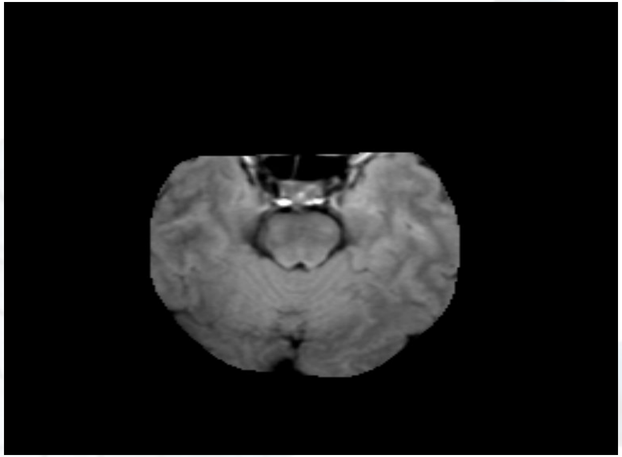


$f($  $) = \hat{y} \quad |y - \hat{y}|_p^p < \epsilon$

Different data sources

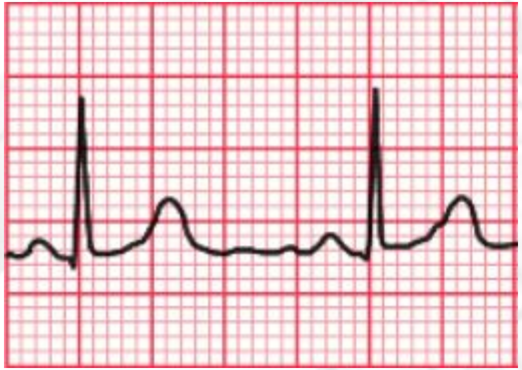


Tracking



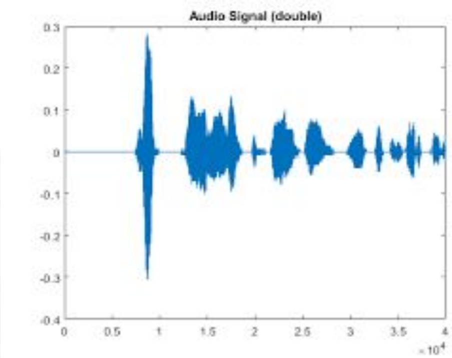
Brain Segmentation

Temporal Signals

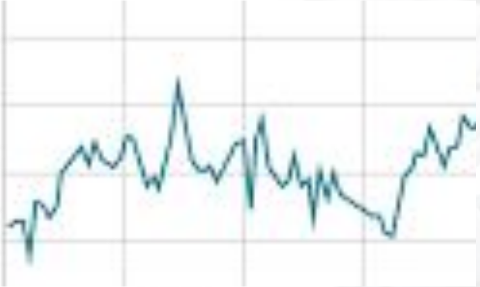


electrocardiogram

electroencephalogram



sound signal

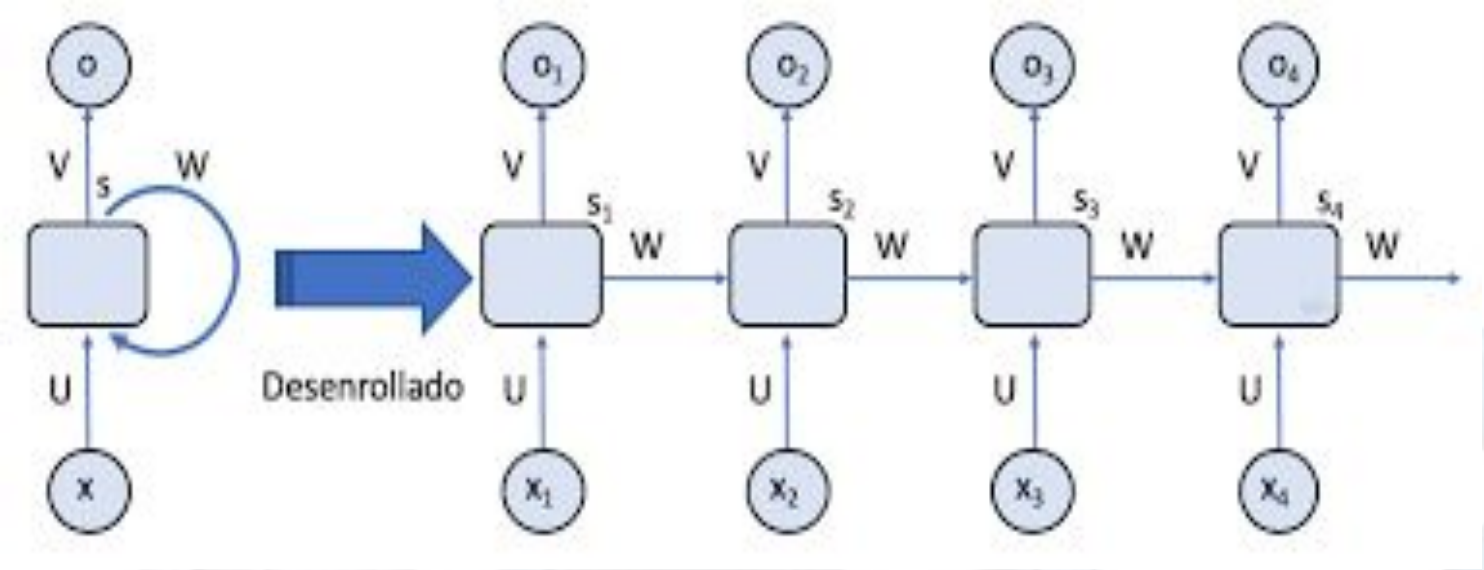


Sales information

ciencia de la **computación es el futuro**

Text

RNN

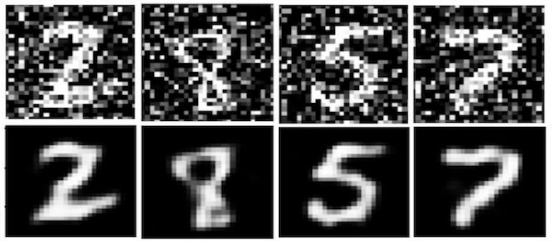


Recurrent Neural Networks

And for these applications?

Image Compression

Image Denoising

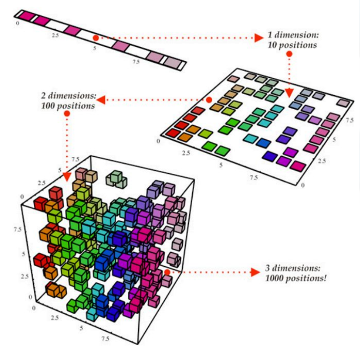


Feature Extraction

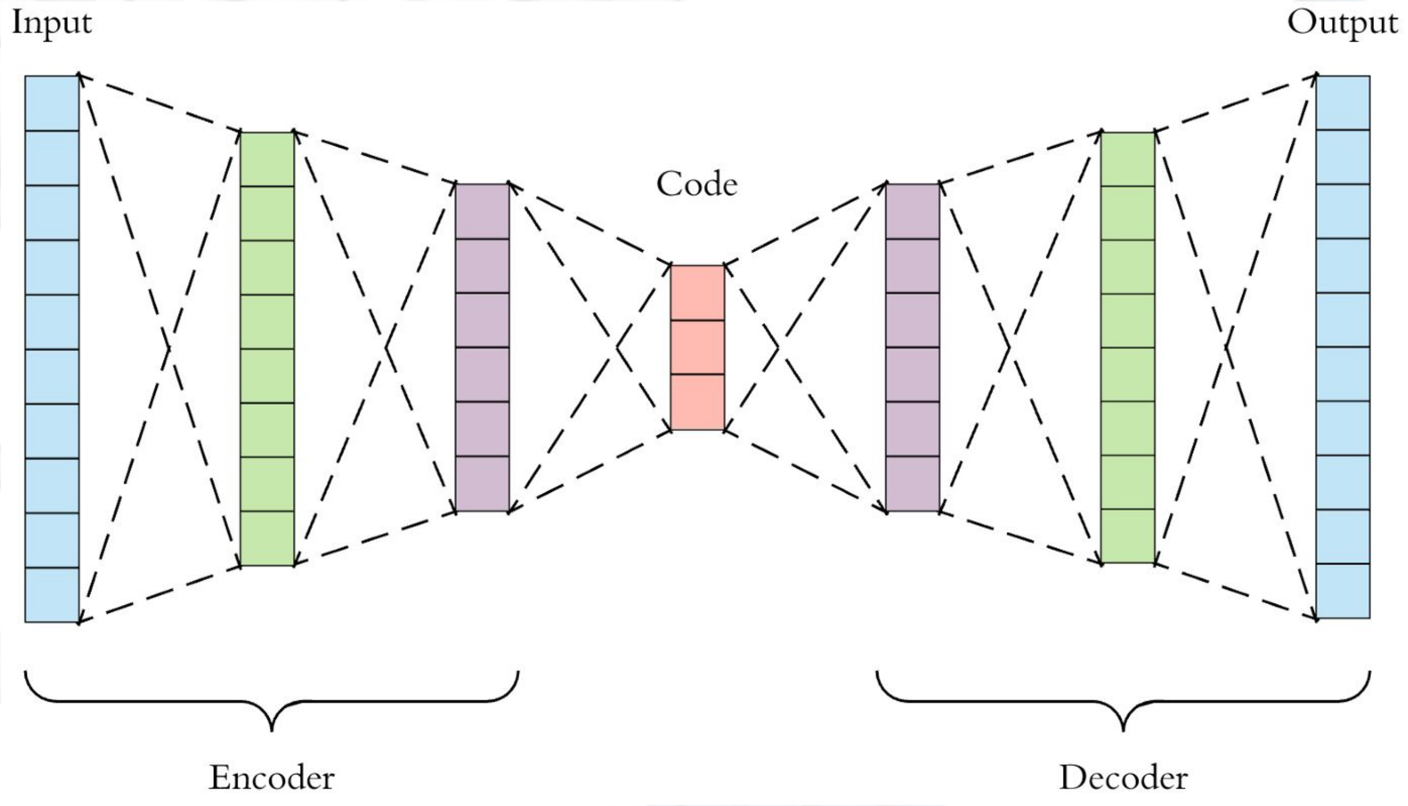
0.1	0.8	1.3	.3	3.1	4.2

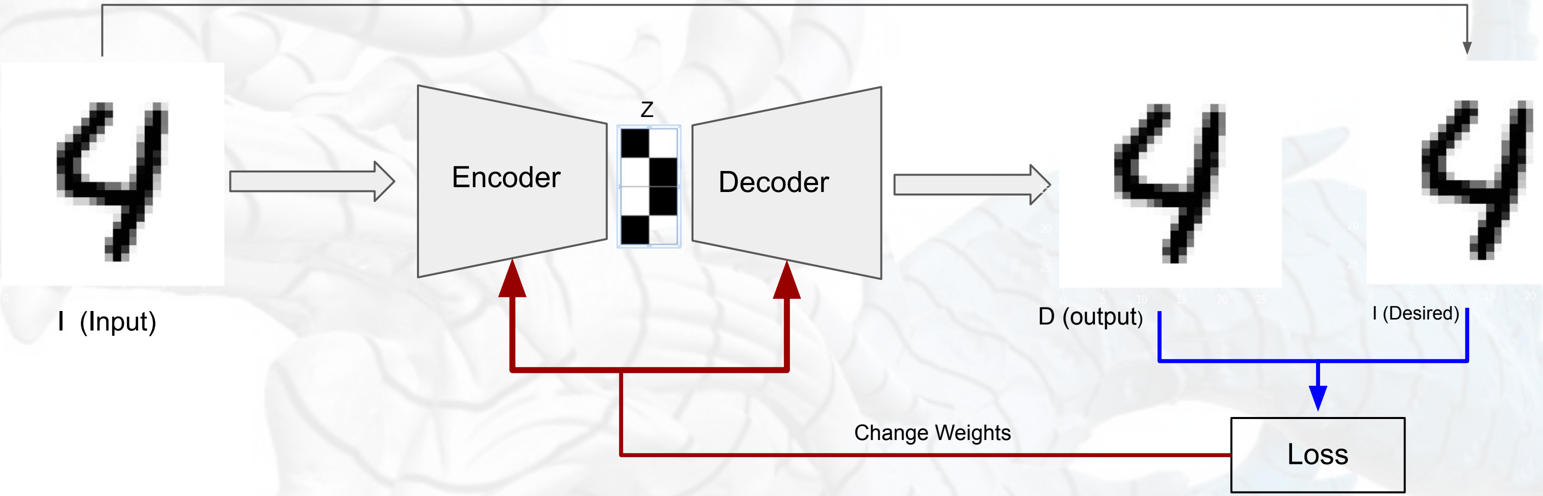
Image Inpainting

Dimensionality Reduction



AUTOENCODERS



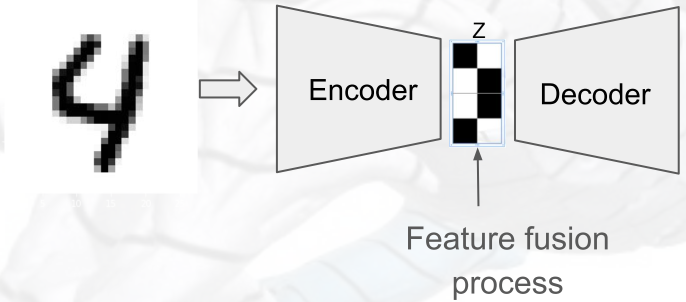


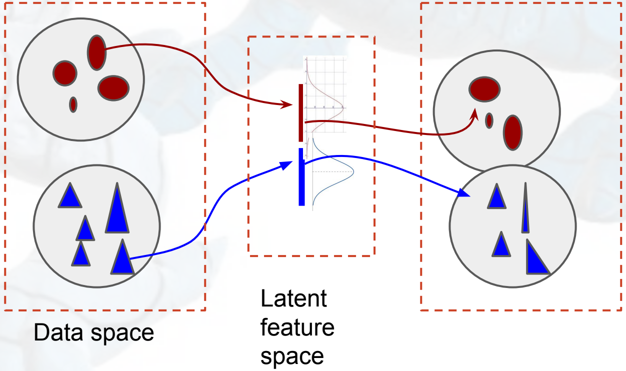
$$Z = Encoder(I) \wedge D = Decoder(Z)\ s.t\ I \approx D$$

$$Where\ I \in \mathbb{R}^n, Z \in \mathbb{R}^d\ and\ D \in \mathbb{R}^n \quad n > d$$

Autoencoder : Objective

The main objective is to perform a feature fusion process.



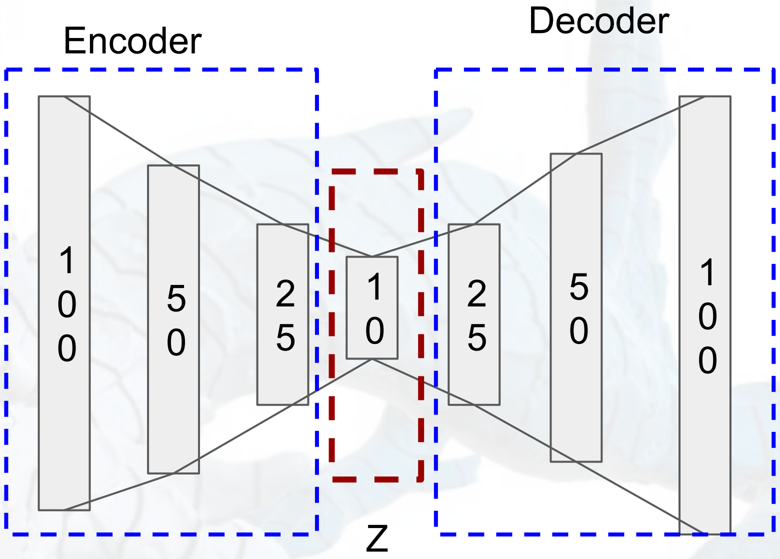


Autoencoder

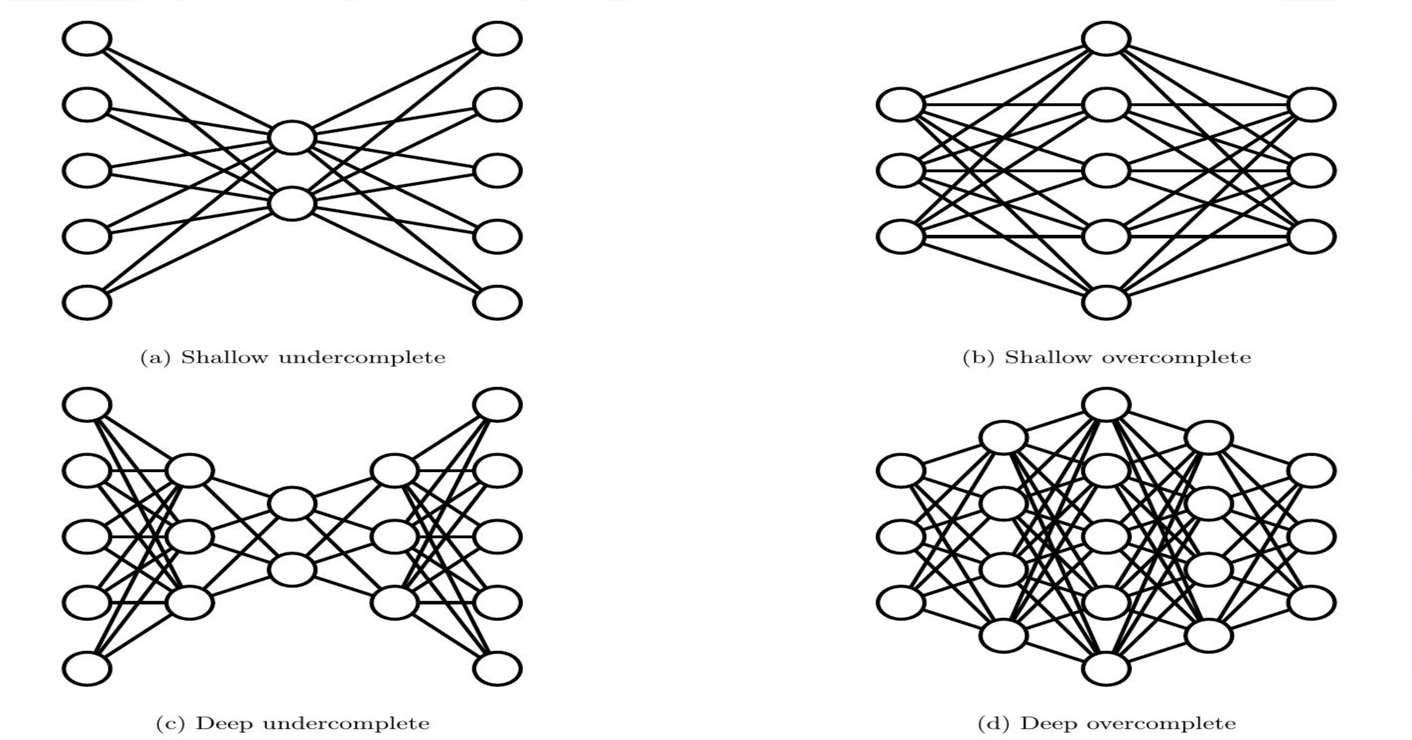
```python
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        #Encoder
        self.E1 = nn.Linear(in_features=100, out_features=50)
        self.E2 = nn.Linear(in_features=50, out_features=25)
        self.E3 = nn.Linear(in_features=25, out_features=10)
        #Decoder
        self.D1 = nn.Linear(in_features=10, out_features=25)
        self.D2 = nn.Linear(in_features=25, out_features=50)
        self.D3 = nn.Linear(in_features=50, out_features=100)

    def forward(self, Input):
        Input = F.relu(self.E1(Input))
        Input = F.relu(self.E2(Input))
        Z     = F.relu(self.E3(Input))

        Output = F.relu(self.D1(Z))
        Output = F.relu(self.D2(Output))
        Output = F.relu(self.D3(Output))
        return Output
```

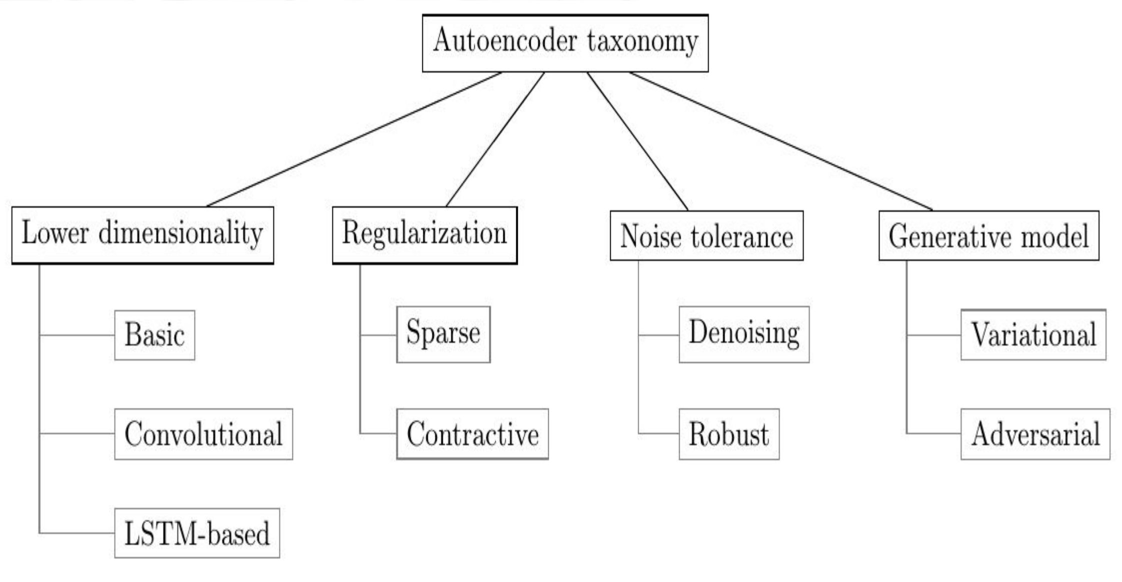


Autoencoder classification by their structure



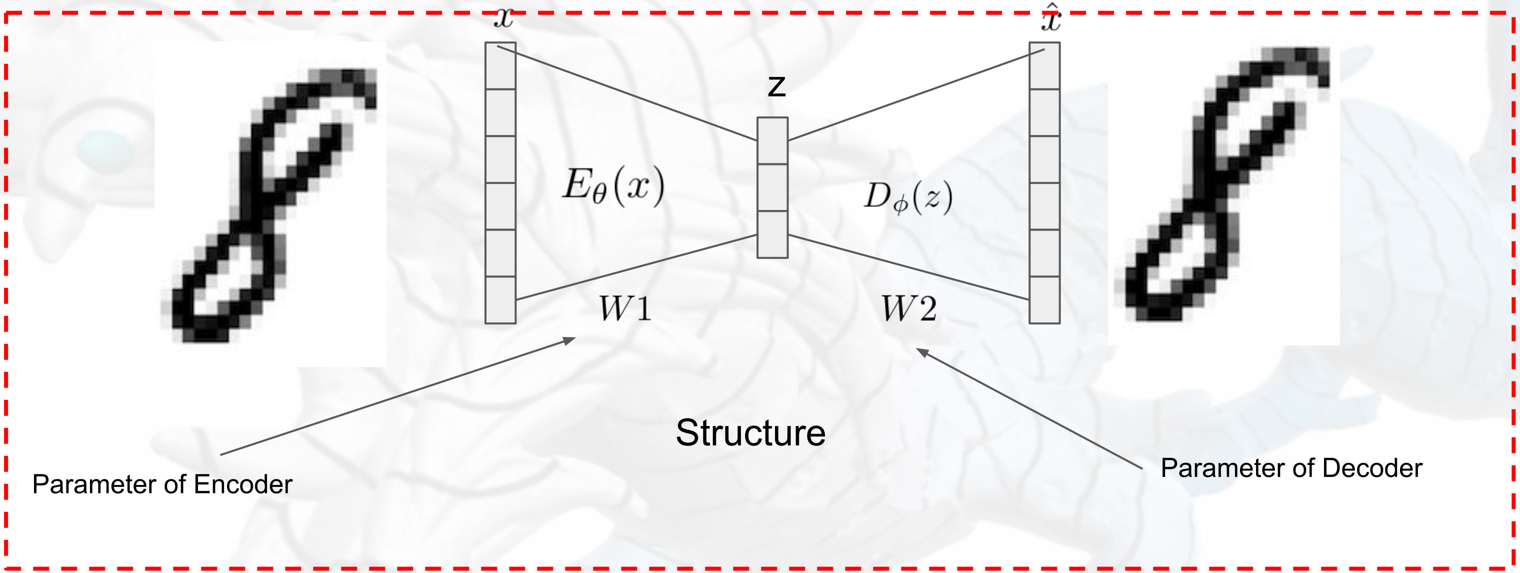
Autoencoder model according to their structure. (David Charte, 2018)

Autoencoder classification by Taxonomy



Autoencoder model according to their structure. (David Charte, 2018)

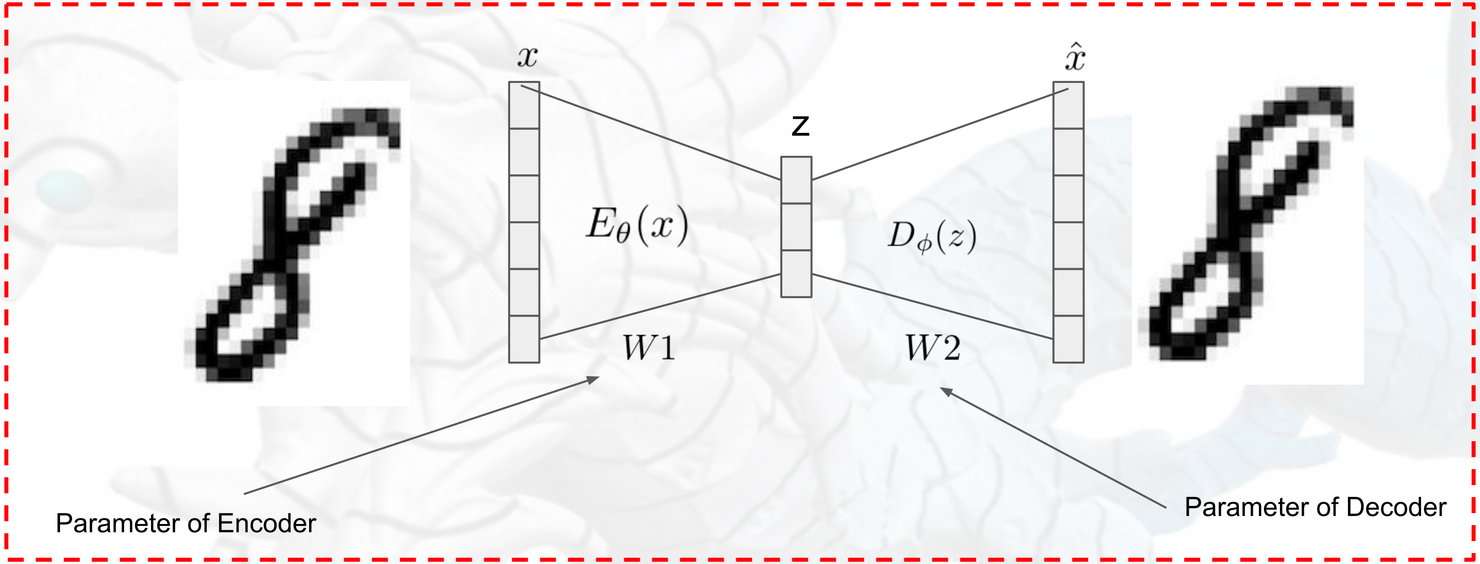
Lower dimensionality: Basic autoencoder



$$z = E_\theta(x) = f_{act}(W_1 x + b_1)$$

$$\hat{x} = D_\phi(z) = f_{act}(W_2 z + b_2)$$

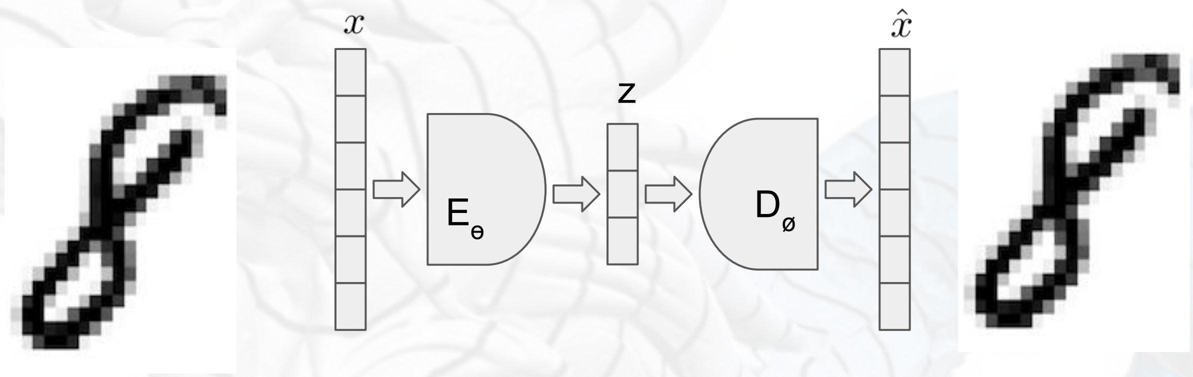
Lower dimensionality: Basic autoencoder



$$\mathcal{L}(x,\hat{x}) = \sum_{x \in S} \mathcal{L}(x, D_\phi(E_\theta(x)))$$

$$\mathcal{L}(x,\hat{x}) = ||x - \hat{x}||_2^2$$

Lower dimensionality: Basic autoencoder



Regularization (weight decay)

$$\mathcal{L}(x,\hat{x}) = \sum_{x \in S} \mathcal{L}(x, D_\phi(E_\theta(x))) + \lambda \sum ||w||_p^p$$

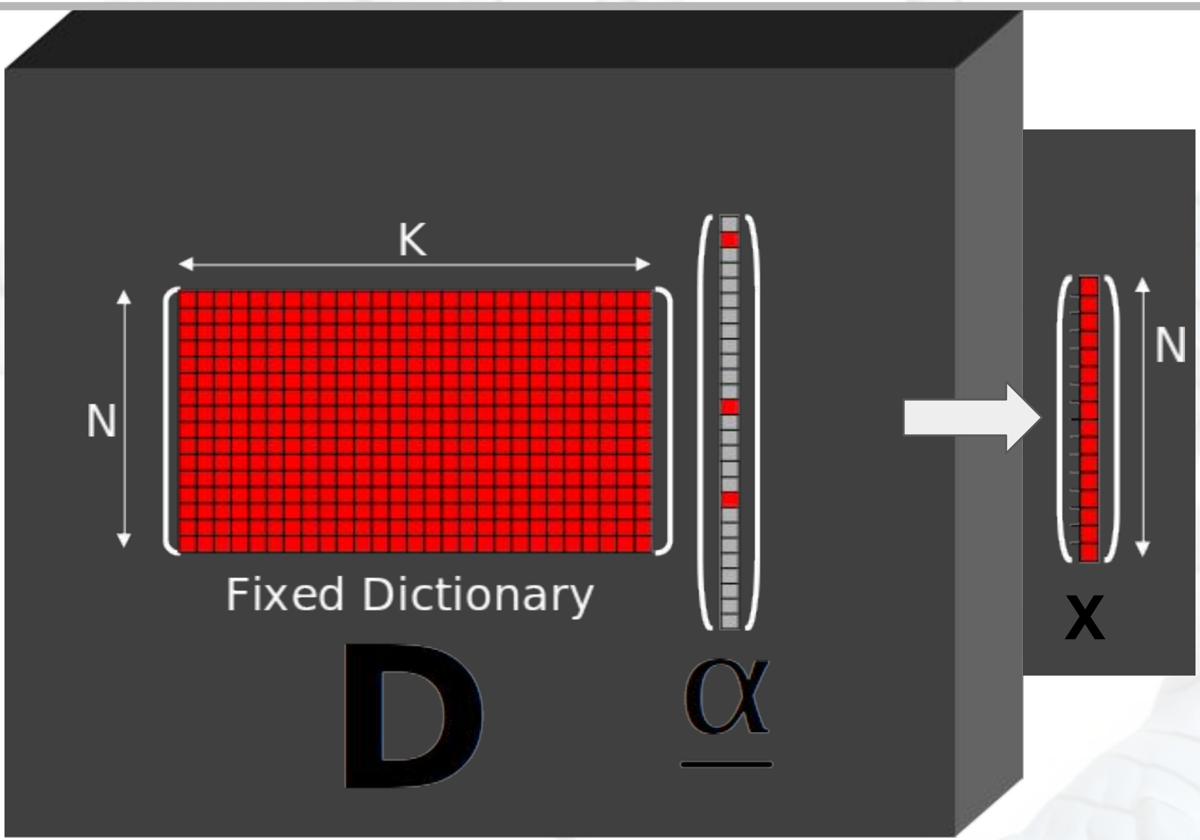
Autoencoder: Regularization

$$\mathcal{L}(x, \hat{x}) = \sum_{x \in S} \mathcal{L}(x, D_\phi(E_\theta(x))) + \lambda\Omega(W)$$

Loss Function

Regularization

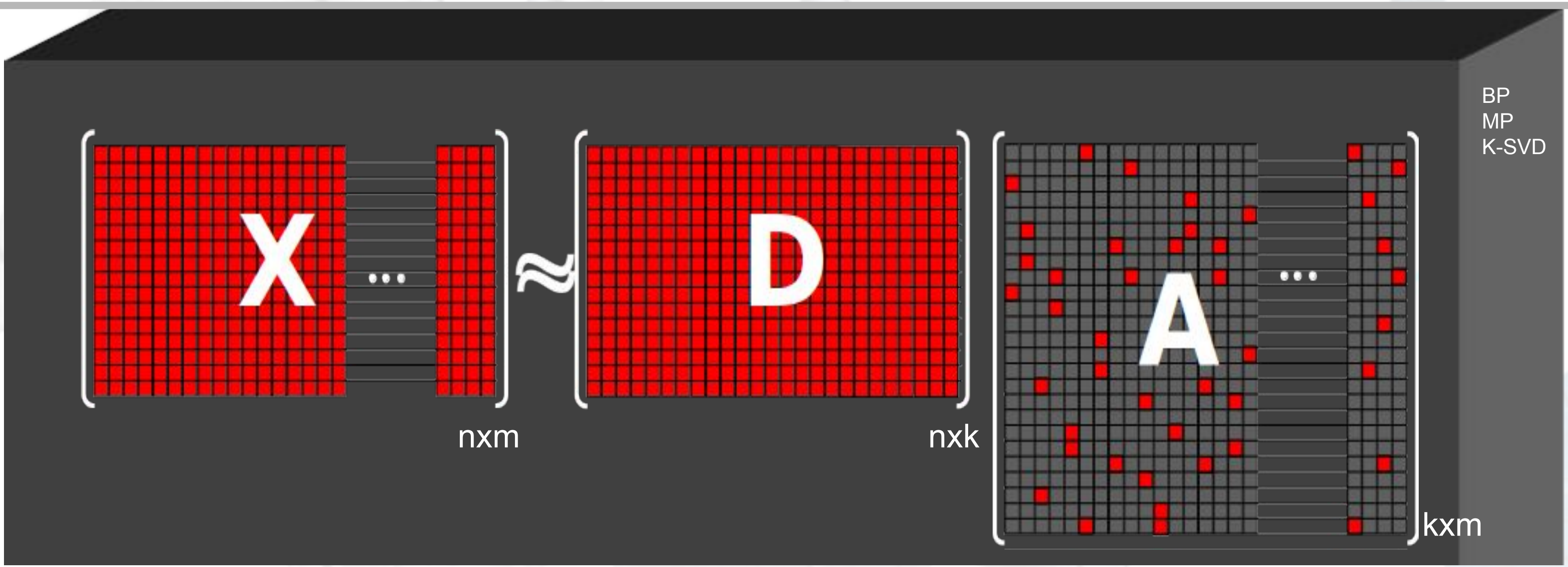
Regularization: Sparse



- Every column in **D** (dictionary) is a prototype signal (atom).

$$X = D\alpha$$

Regularization: Sparse

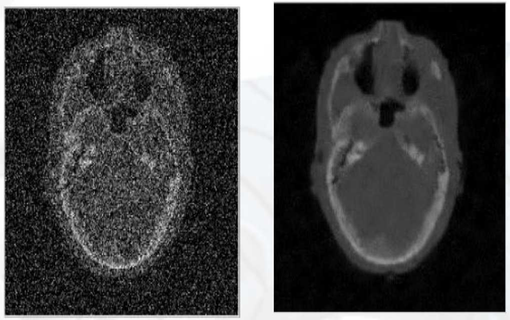


$$||DA - X||_F^2 \; \forall j, \; ||\alpha_j||_0 <= L$$

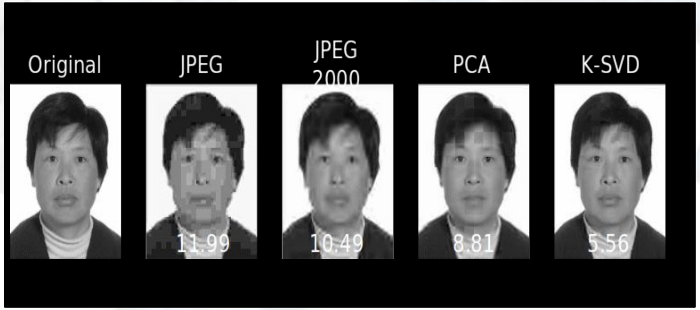
Sparsity Applications

Inpainting

Denoising



Compression

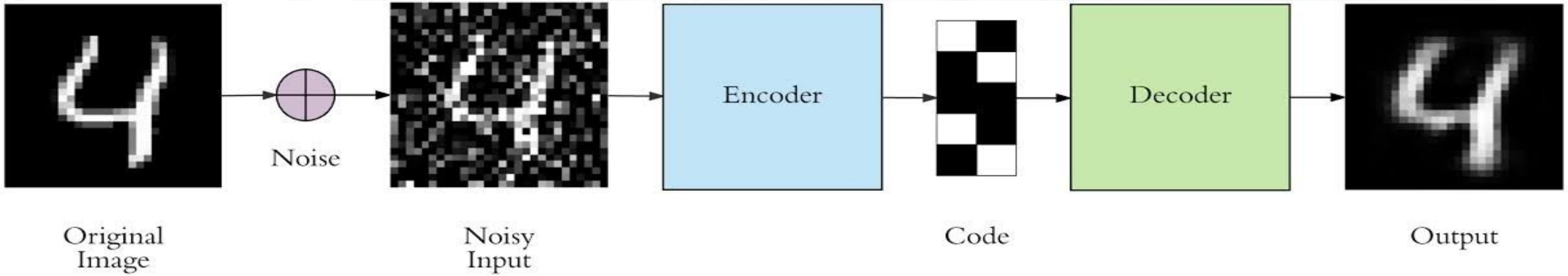


source click

Sparse Autoencoder

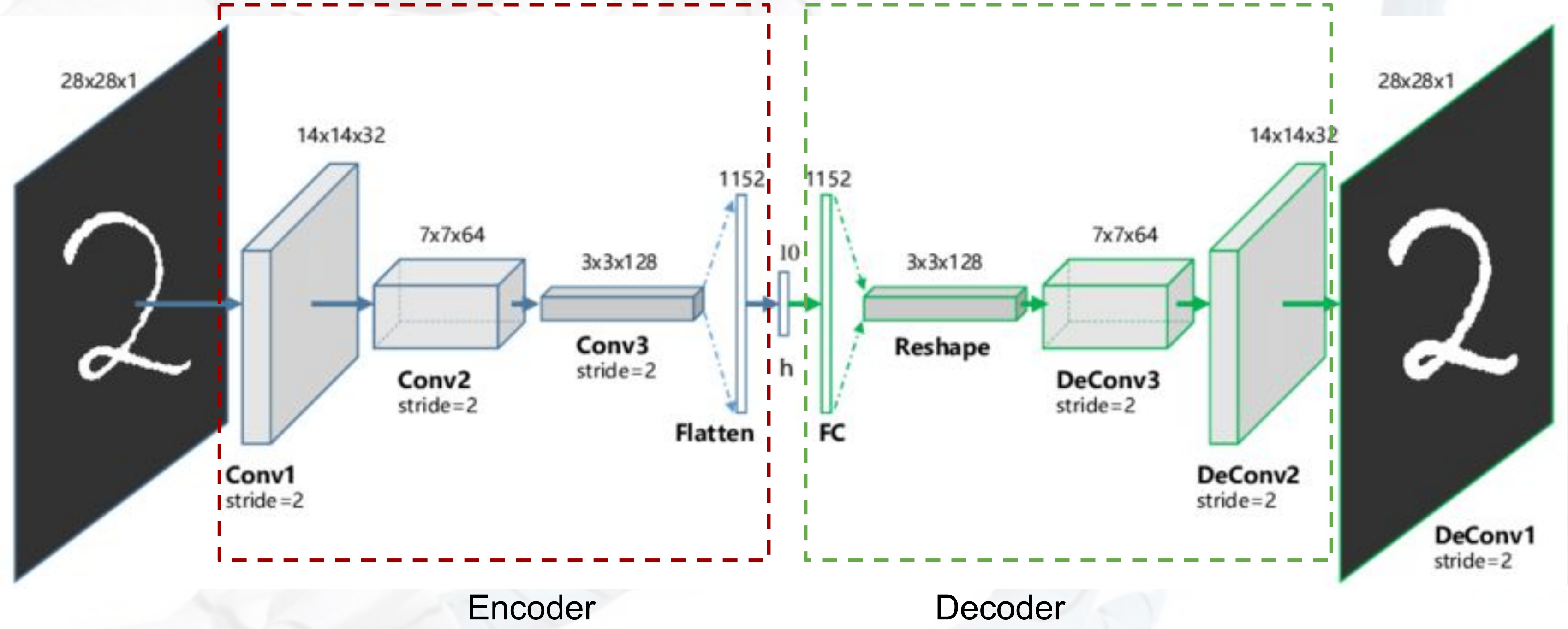
Denoising Autoencoders

Denoising autoencoder $\leftarrow$ $argmin_{\theta} \quad ||(x - \hat{x})||^2, \quad \hat{x} = D(z), z = E(x + \mathcal{N}(0,1))$



[Pytorch Example]

Convolutional Autoencoders



source: click

Convolutional Autoencoders: Unpooling

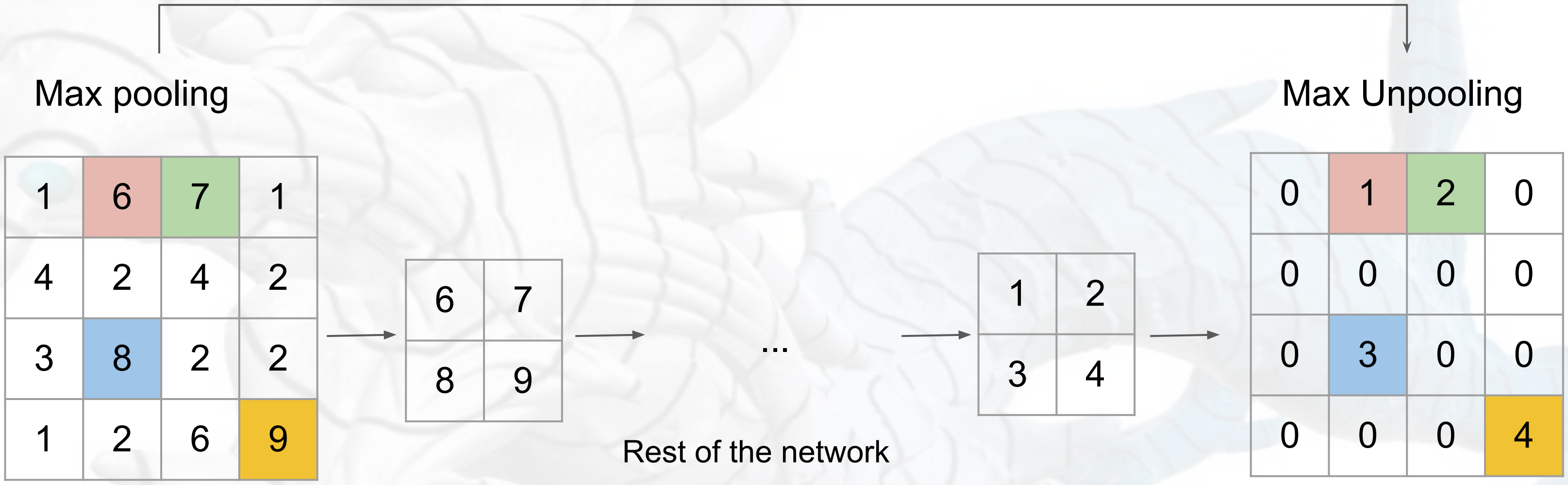
Nearest Neighbor

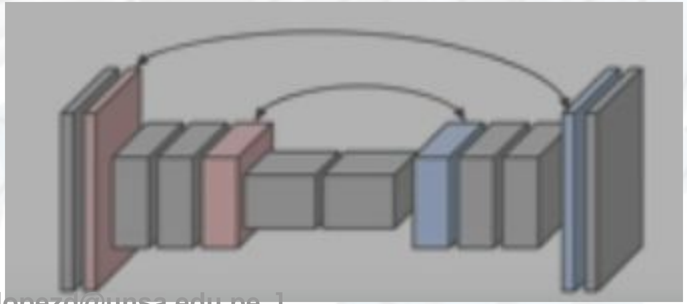
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix}$$

Bed of Nails

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

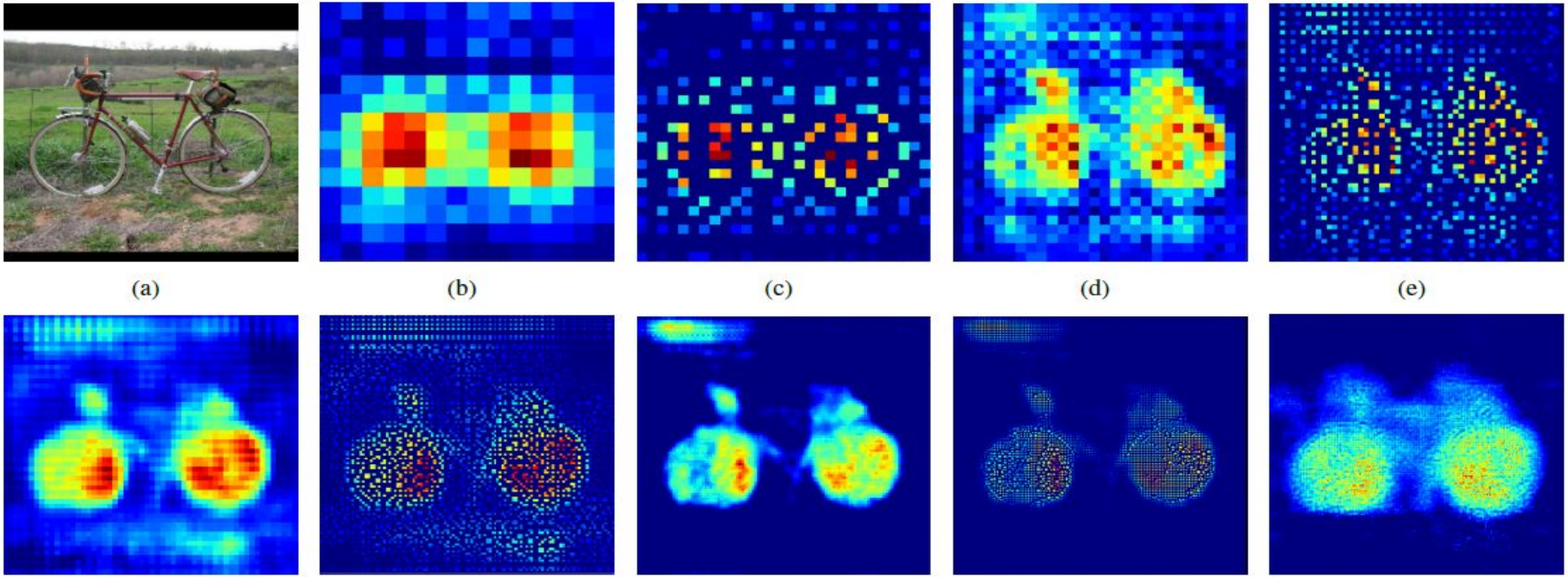
Convolutional Autoencoders : Max Unpooling





Source: [Video]

Convolutional Autoencoders : Unpooling



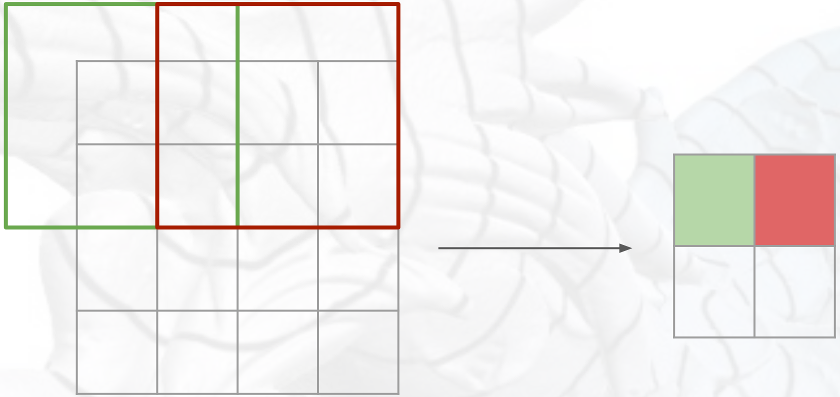
(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Source: [click]

Convolutional Autoencoders : Transpose Convolution

Kernel 3x3, stride 2, pad 1



Input 4x4

Output 2x2

Source: [Video]

Convolutional Autoencoders : Transpose Convolution

Kernel 3x3, stride 2, pad 1

Input 2x2

Output 4x4

Source: [Video]

Convolutional Autoencoders : Deconvolution - Upconvolution

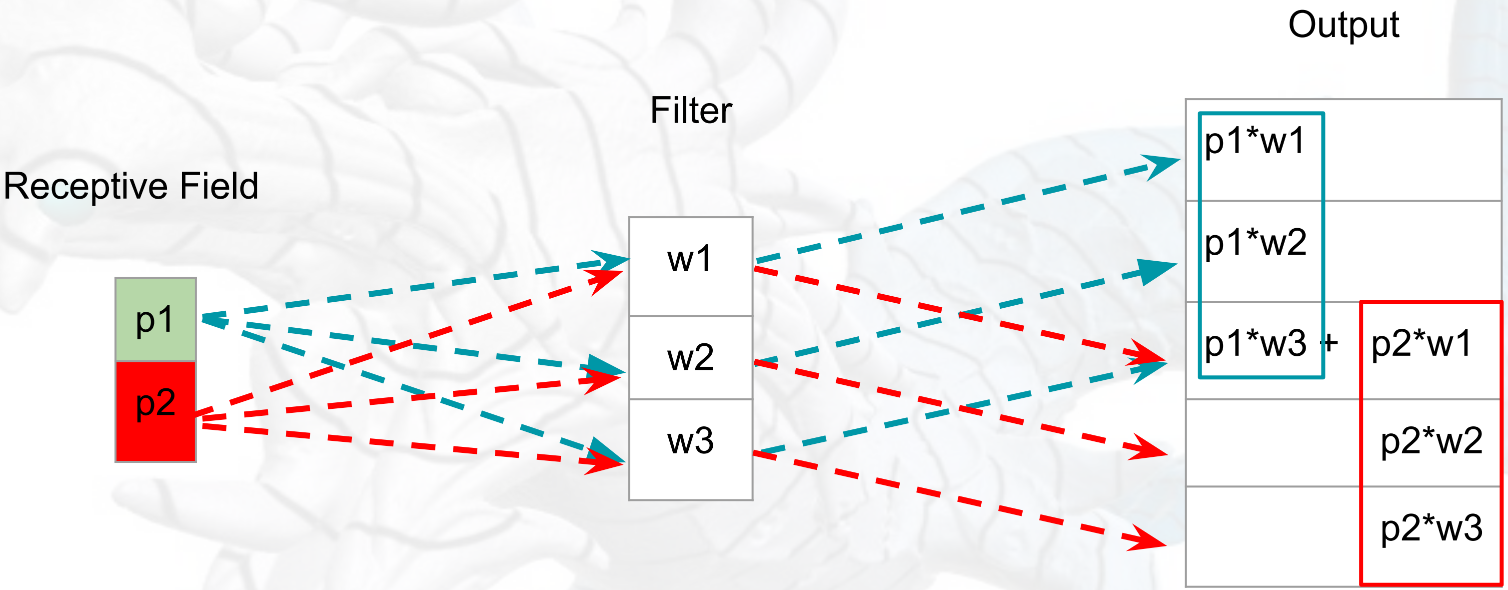
Kernel 3x3, stride 2, pad 1

Input 2x2

Output 4x4

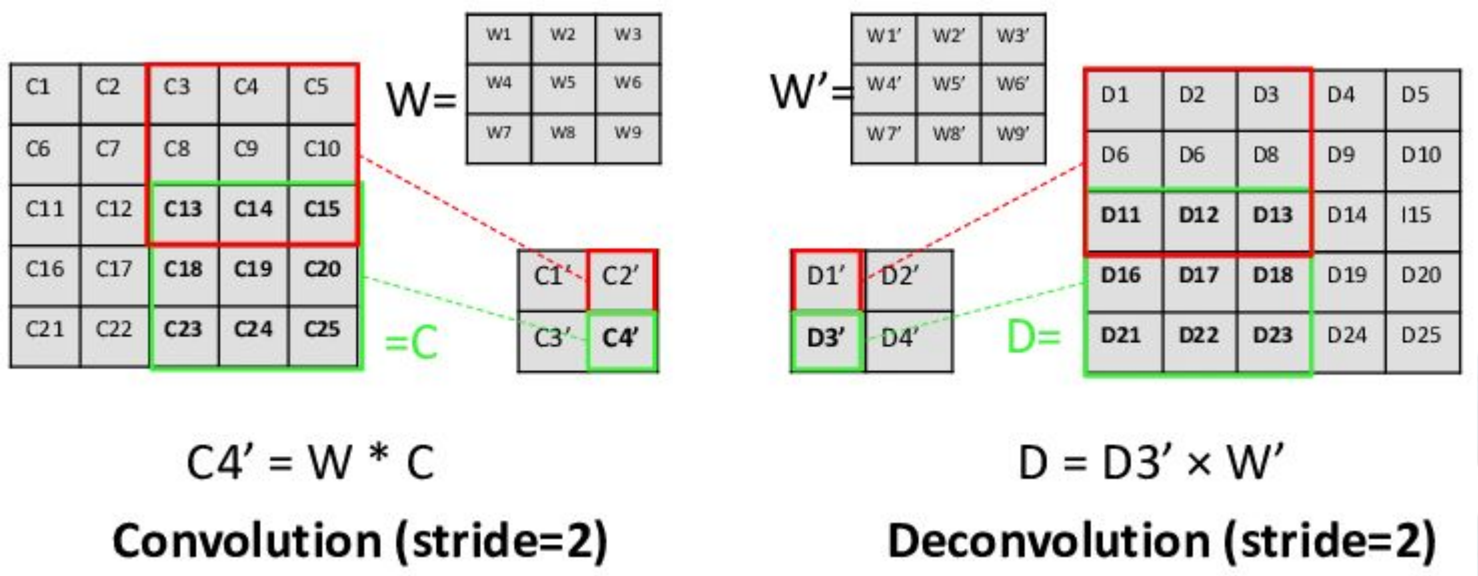
Source: [Video]

Convolutional Autoencoders : Deconvolution - Upconvolution



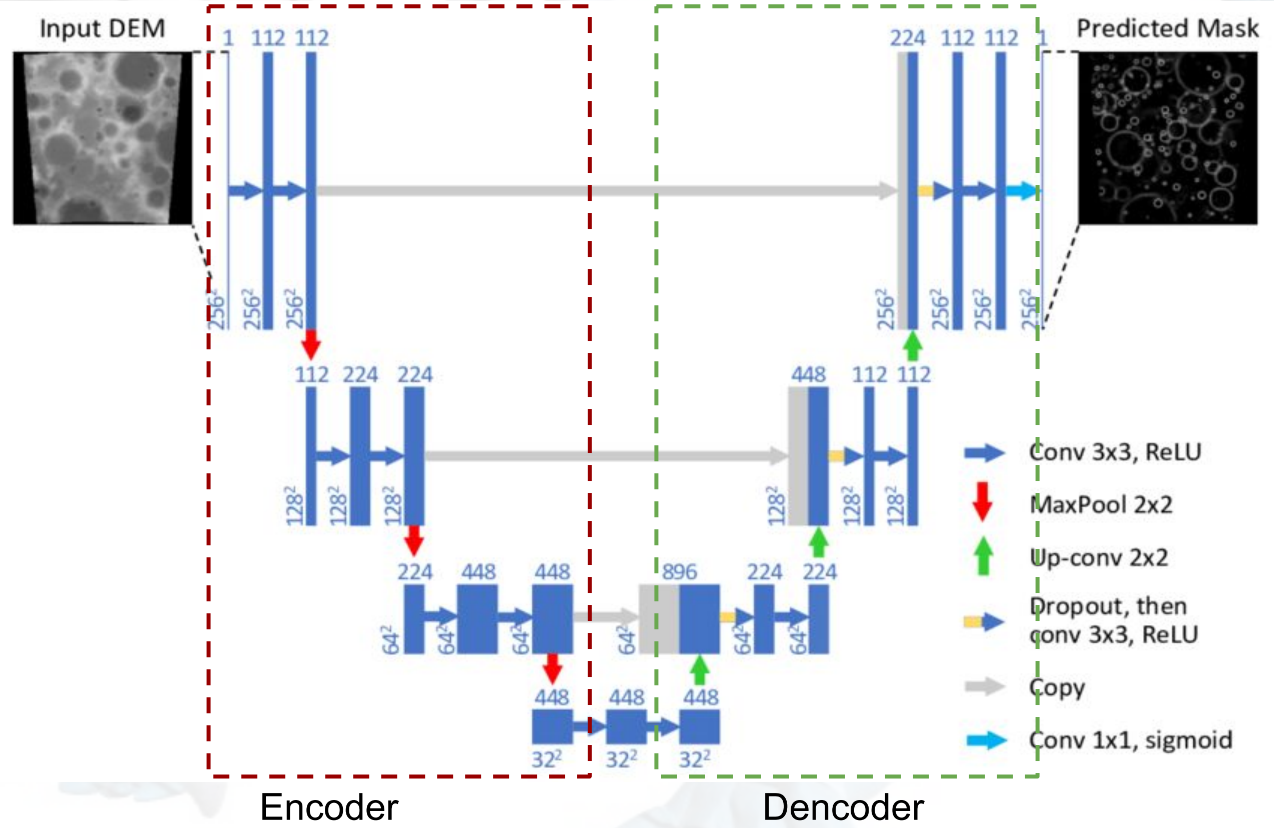
Source: [Video]

Convolutional Autoencoders : Deconvolution - Upconvolution



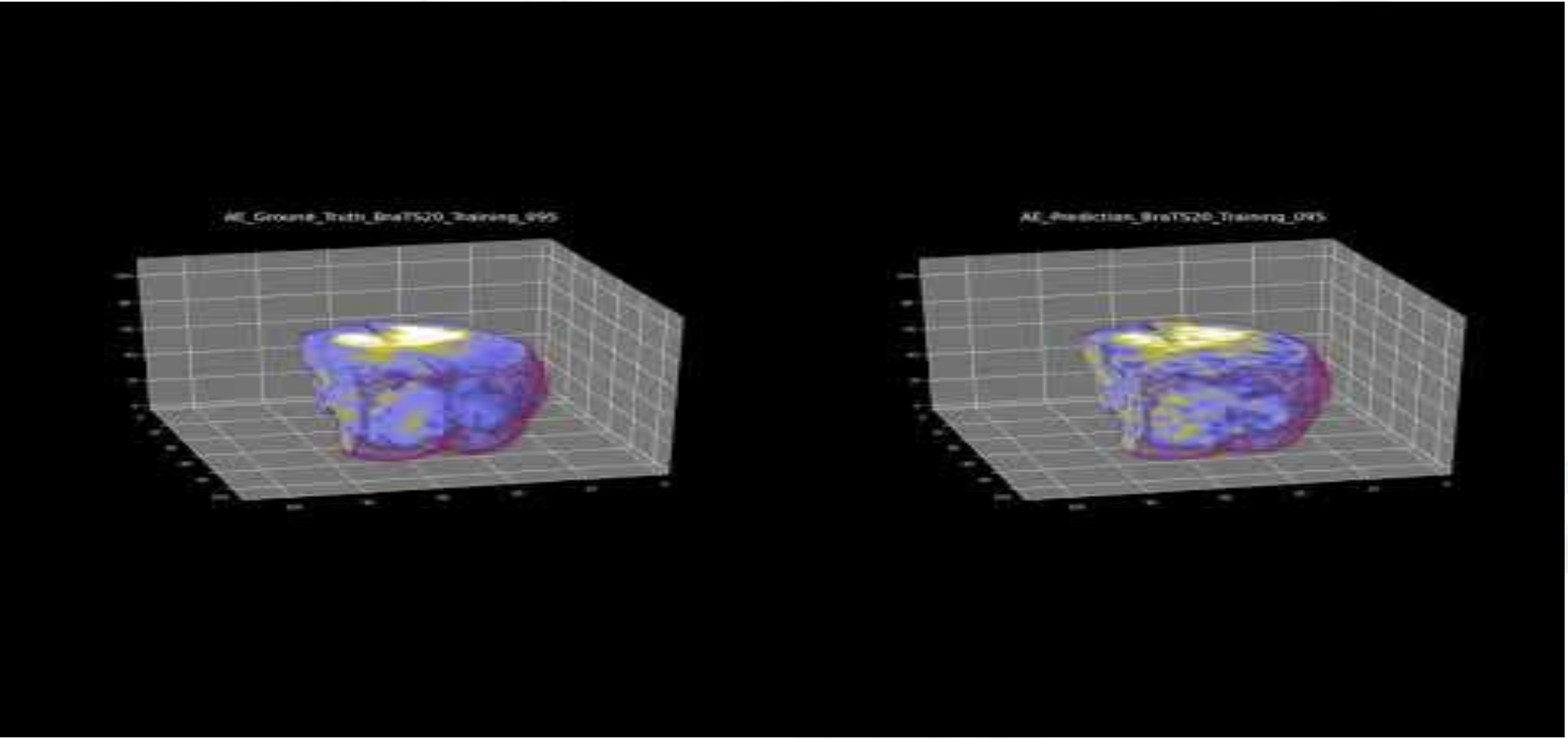
Source: [Video]

Convolutional Autoencoders

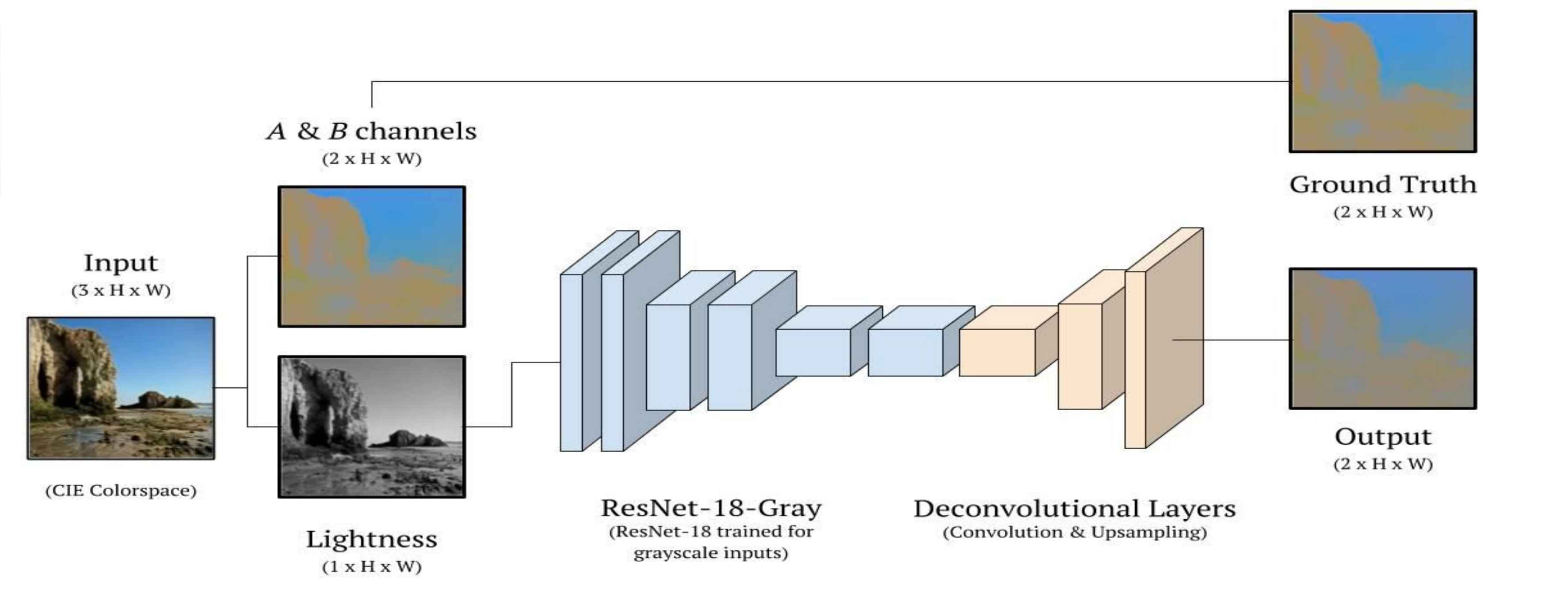


source: Unet arxiv

convolutional autoencoders example



convolutional autoencoders example



Source: [click]

convolutional autoencoders example

convolutional autoencoders example

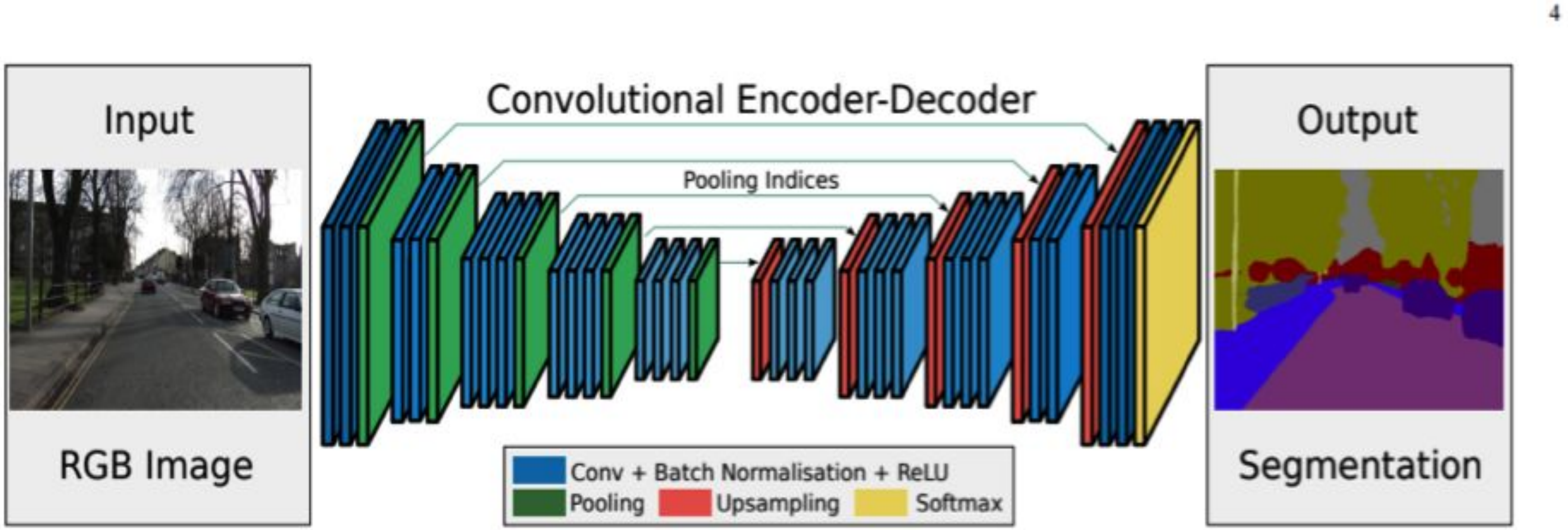
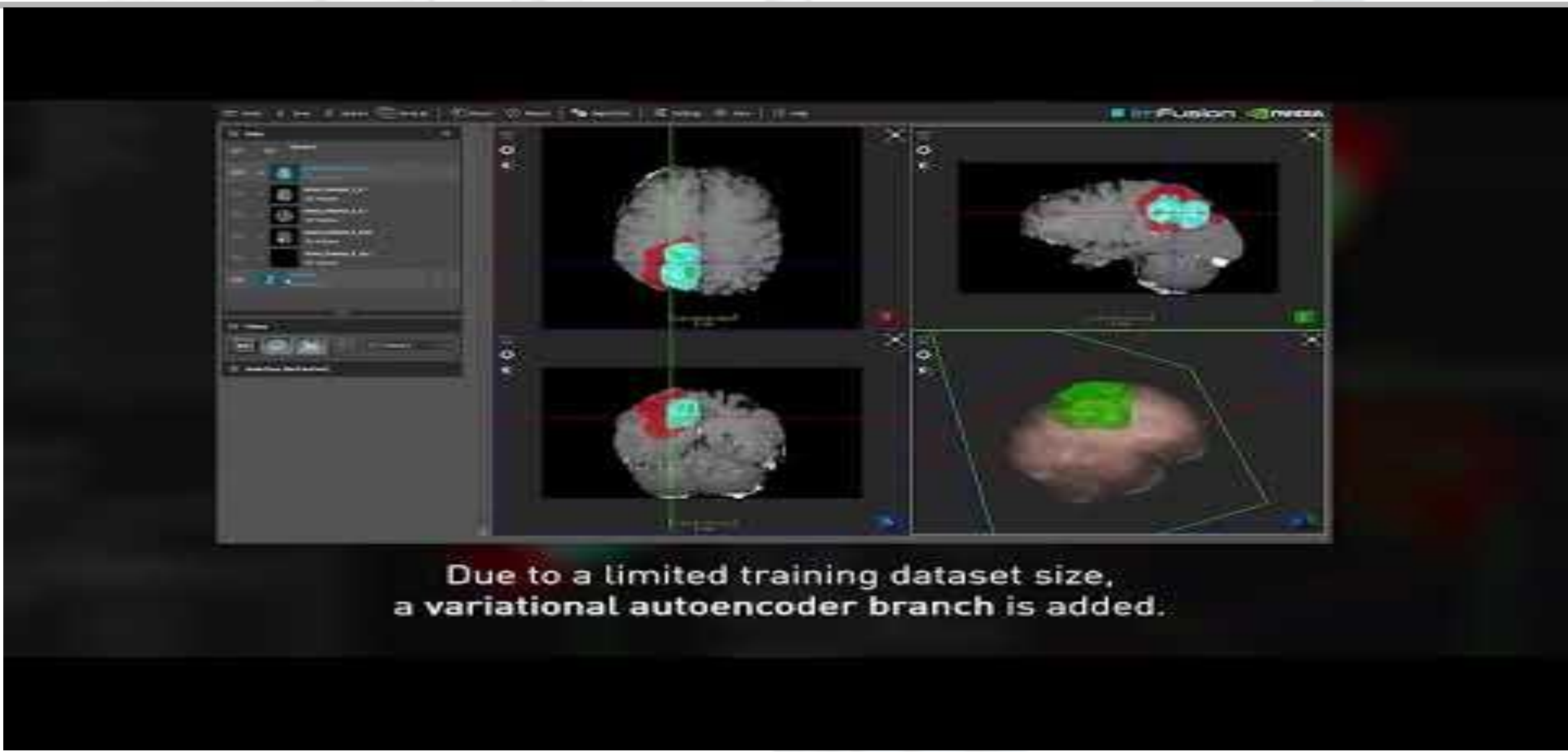


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

Image segmentation: click

convolutional autoencoders example



convolutional autoencoders example

convolutional autoencoders example

[Pytorch Example]