

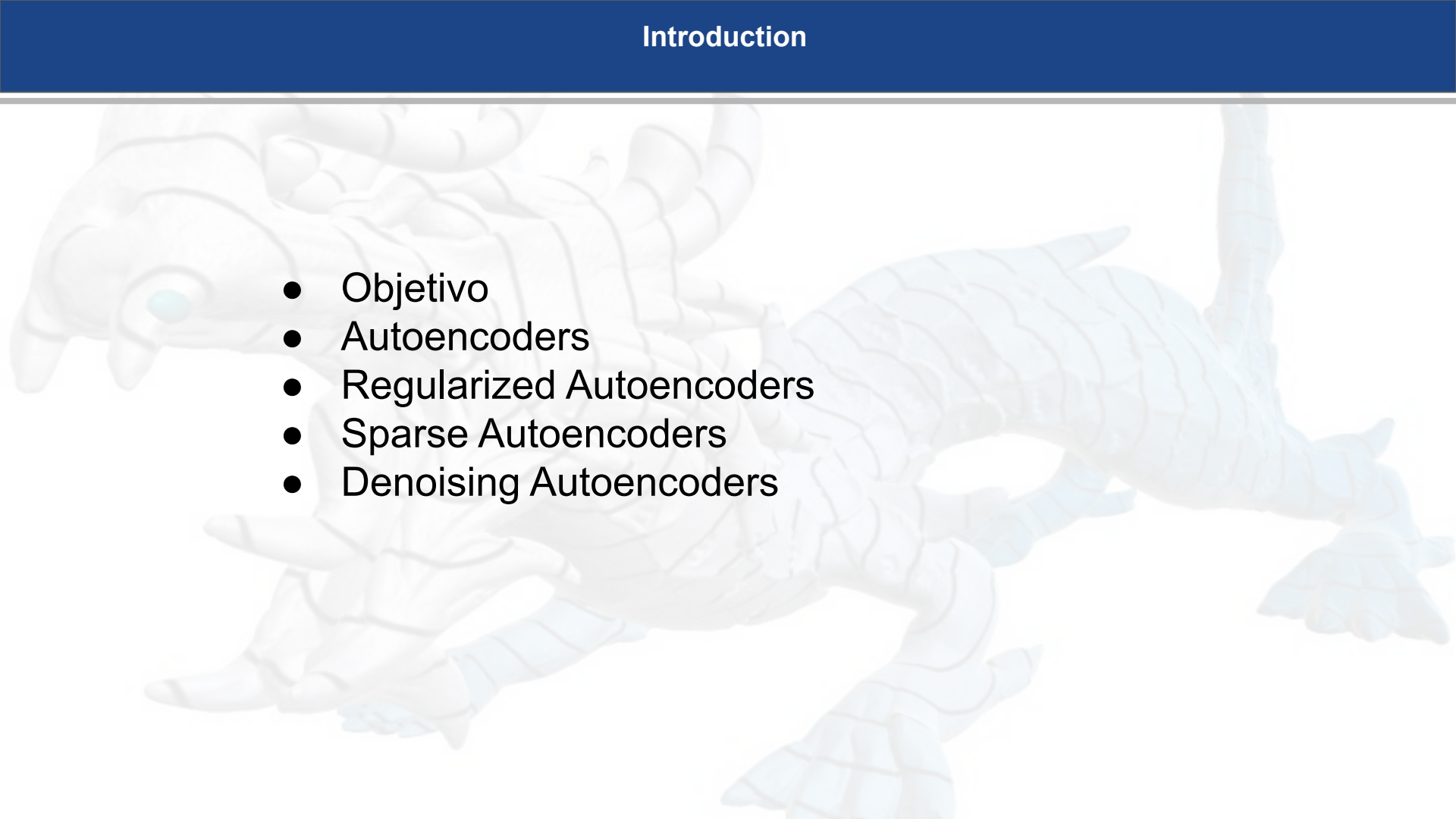


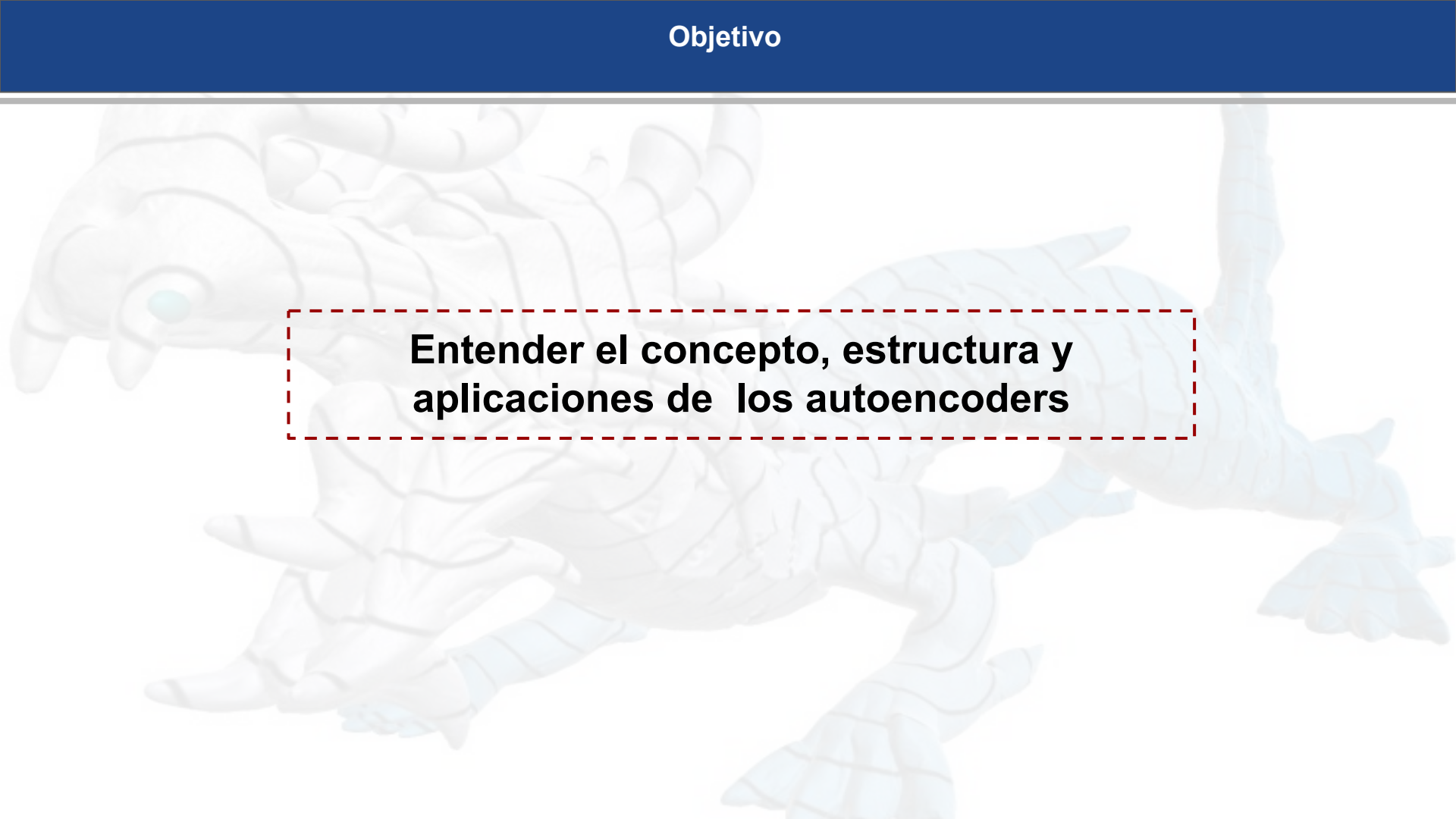
AUTOENCODERS

RESEARCHGROUP
I PRODAM3D

Cristian López Del Alamo

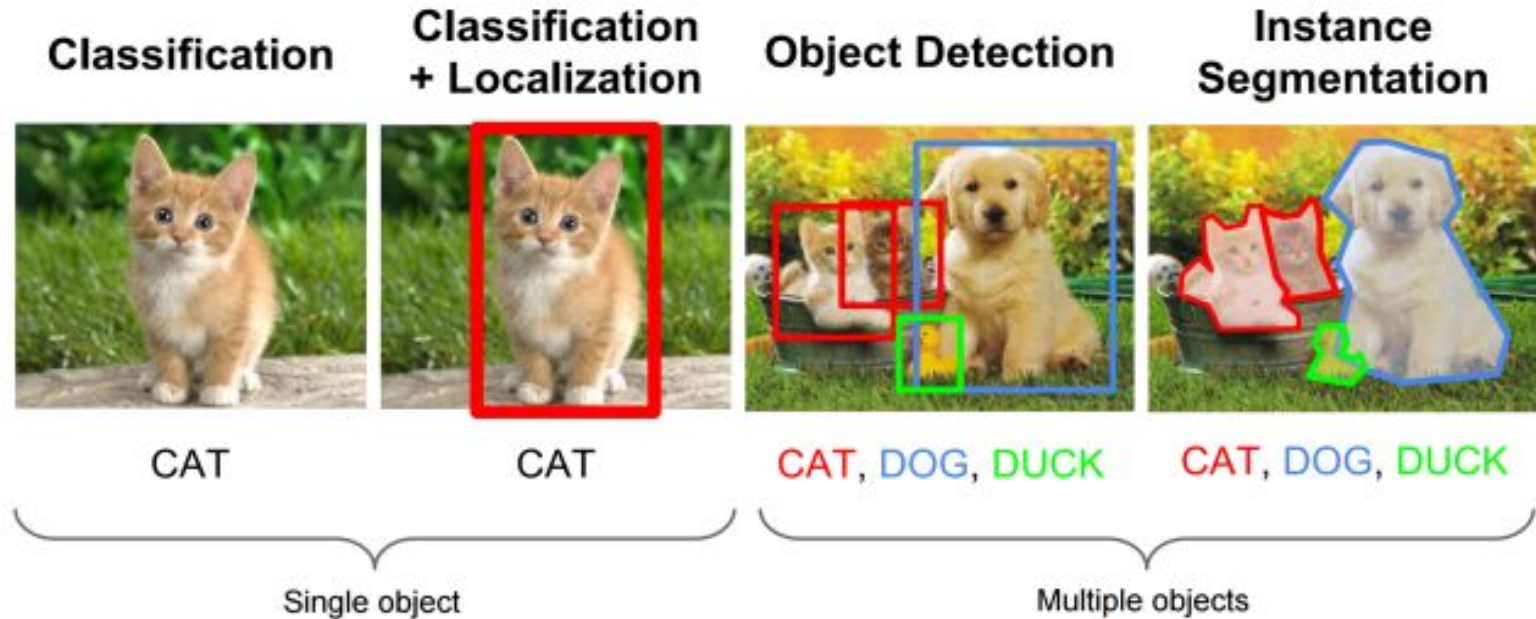
2022

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

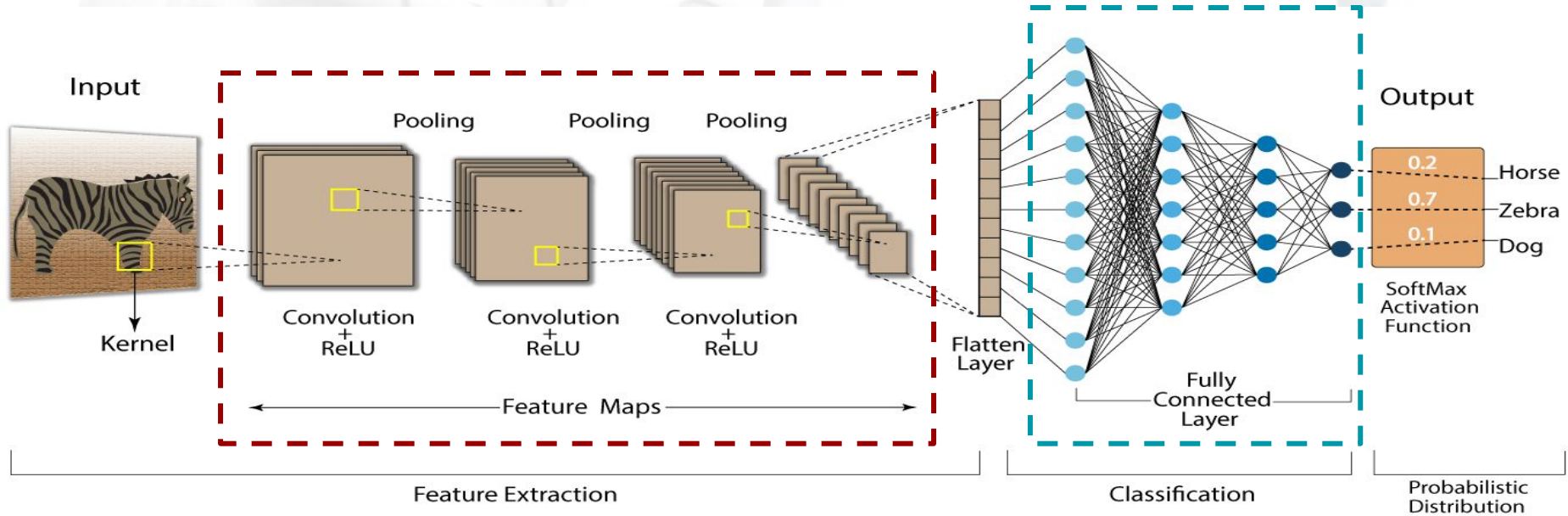


Entender el concepto, estructura y aplicaciones de los autoencoders

Different data sources : Images

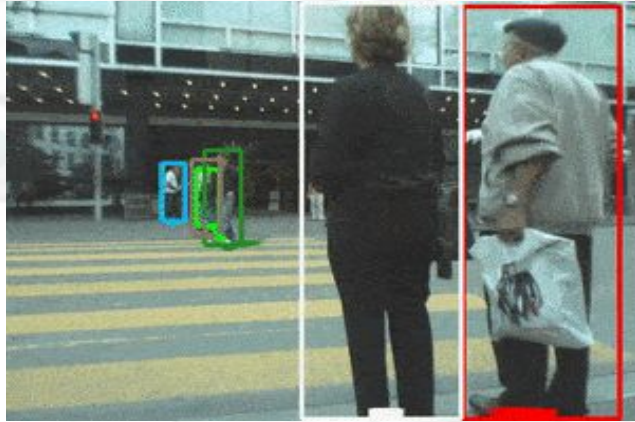


CNN

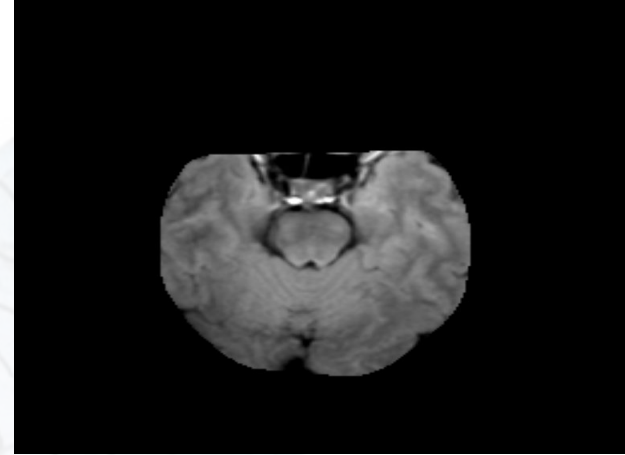


$$f\left(\text{Input Image}\right) = \hat{y} \quad |y - \hat{y}|_p^p < \epsilon$$

Different data sources

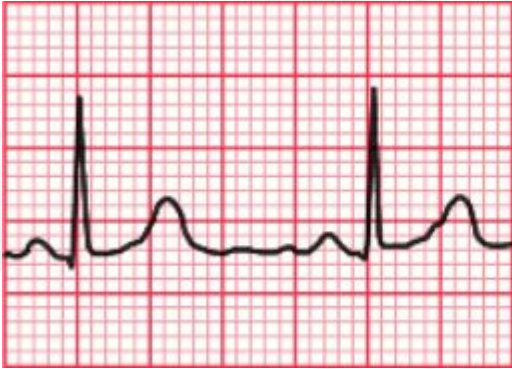


Tracking



Brain Segmentation

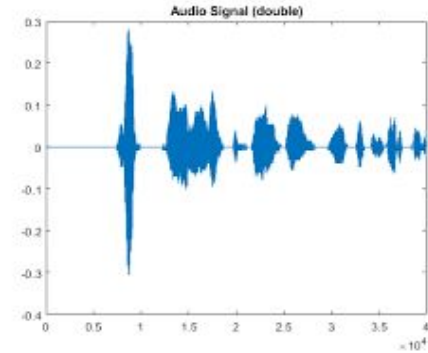
Temporal Signals



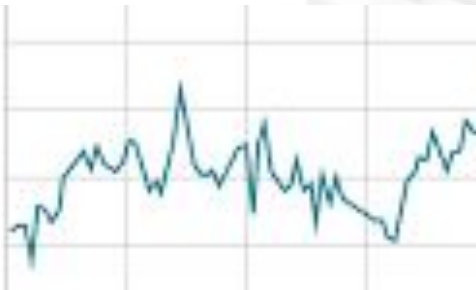
electrocardiogram



electroencephalogram



sound signal

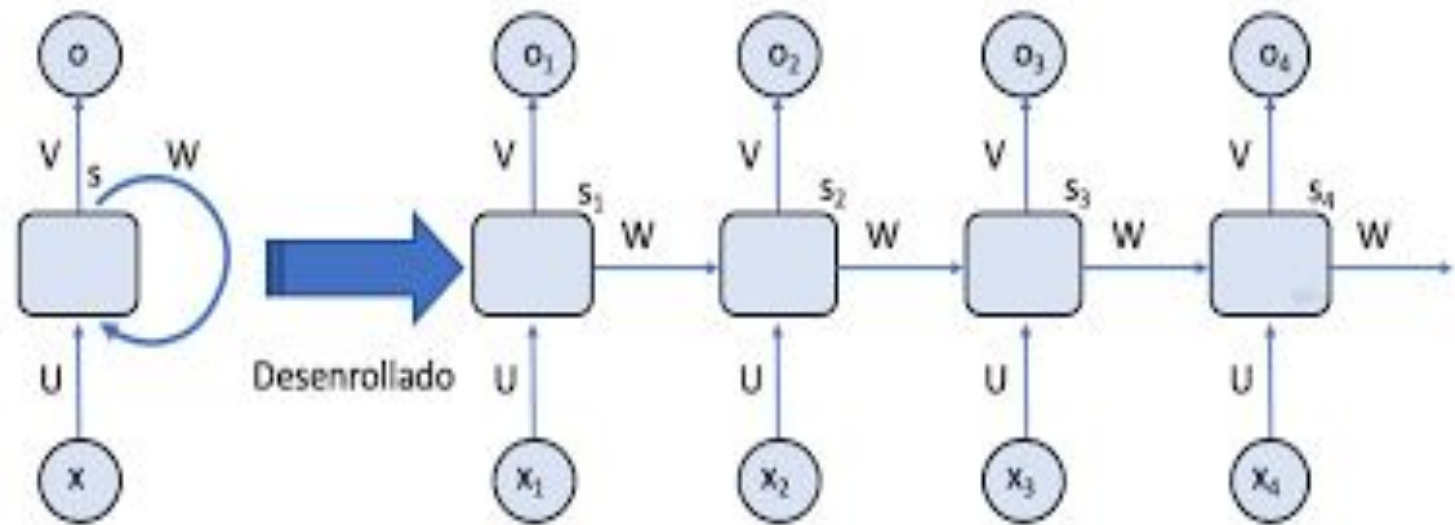


Sales information

ciencia de la computación es el futuro

Text

RNN



Recurrent Neural Networks

And for these applications?

Image Compression

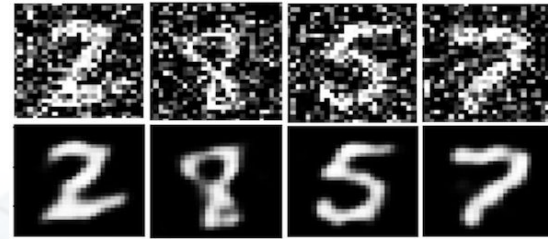


100%



53%

Image Denoising

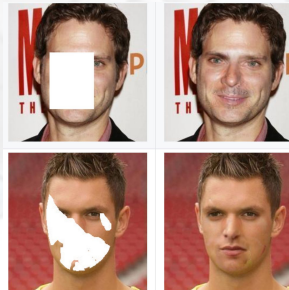


Feature Extraction

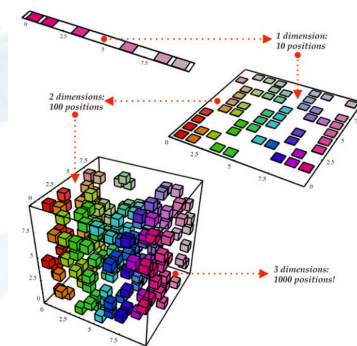


0.1	0.8	1.3	.3	3.1	4.2
-----	-----	-----	----	-----	-----

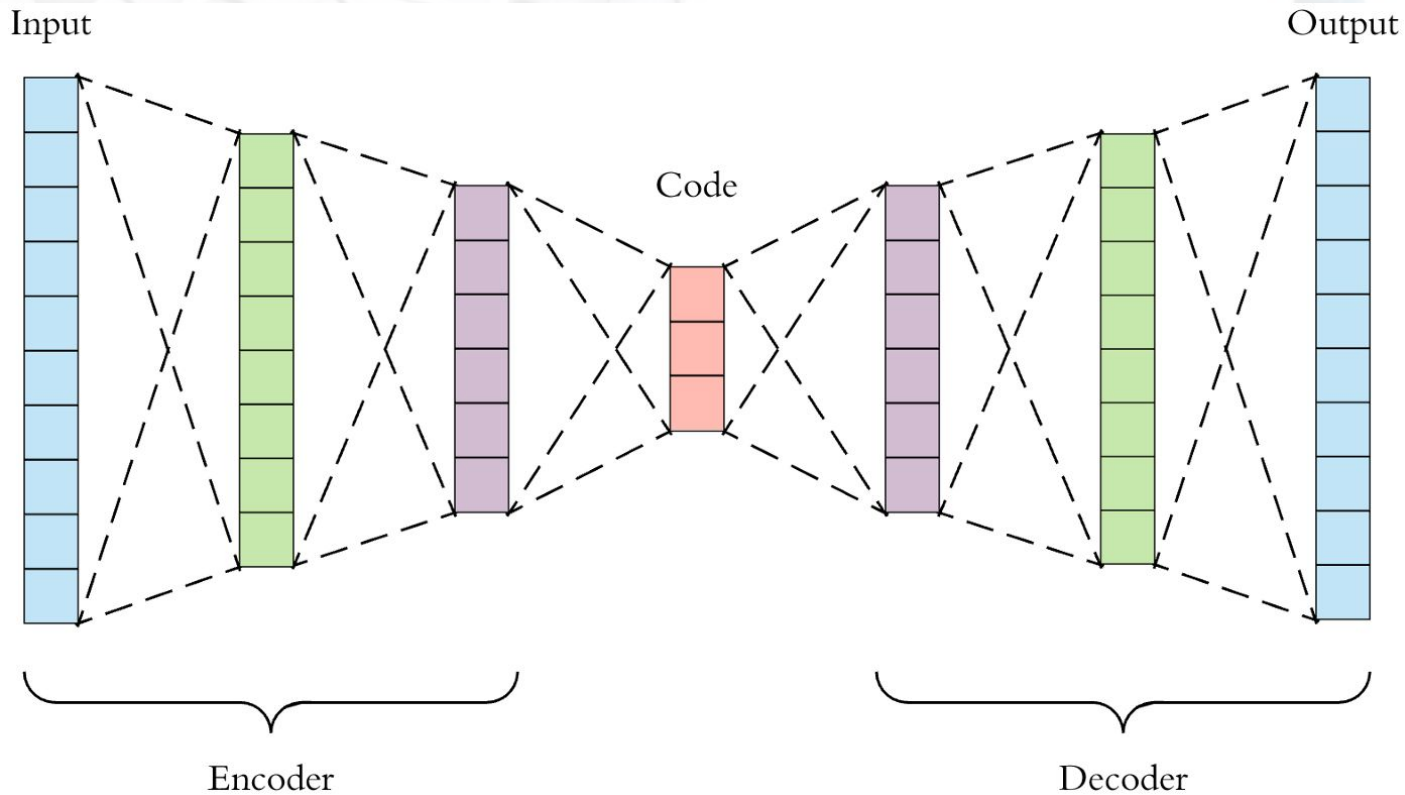
Image Inpainting



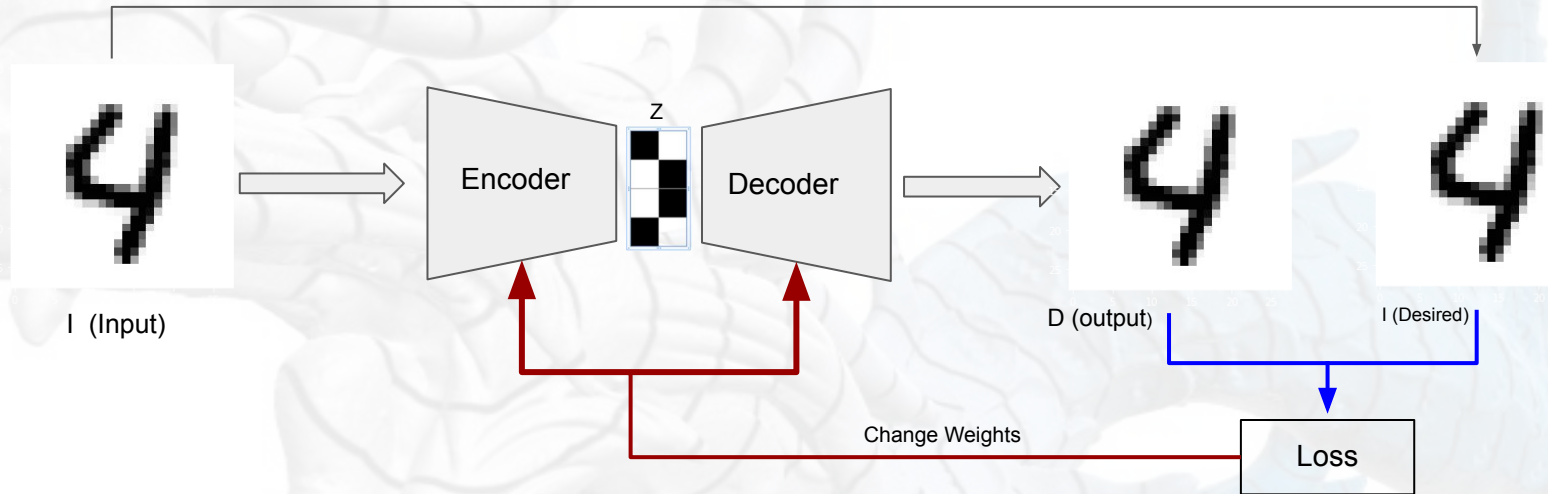
Dimensionality Reduction



AUTOENCODERS



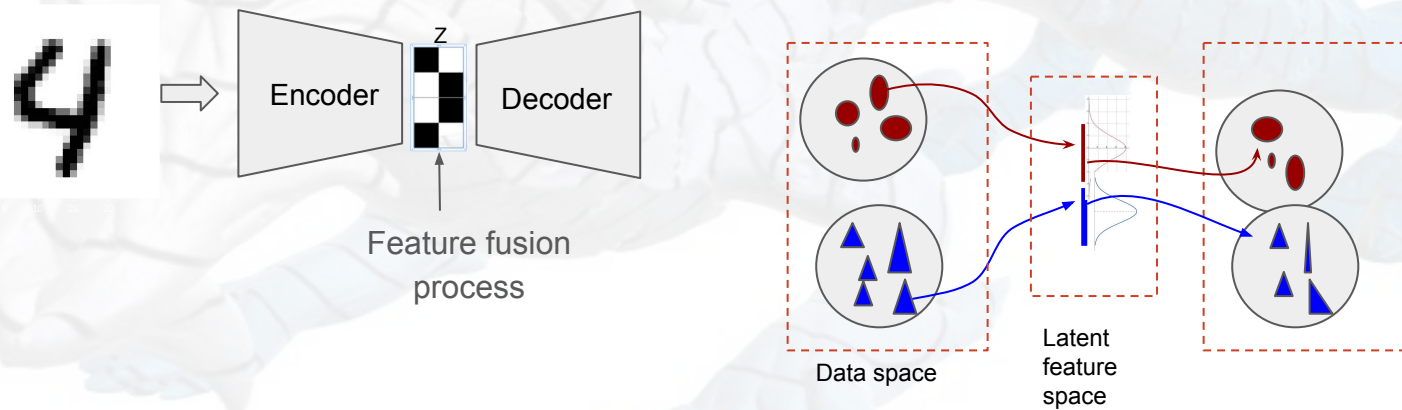
Autoencoder



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

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

The main objective is to perform a feature fusion process.

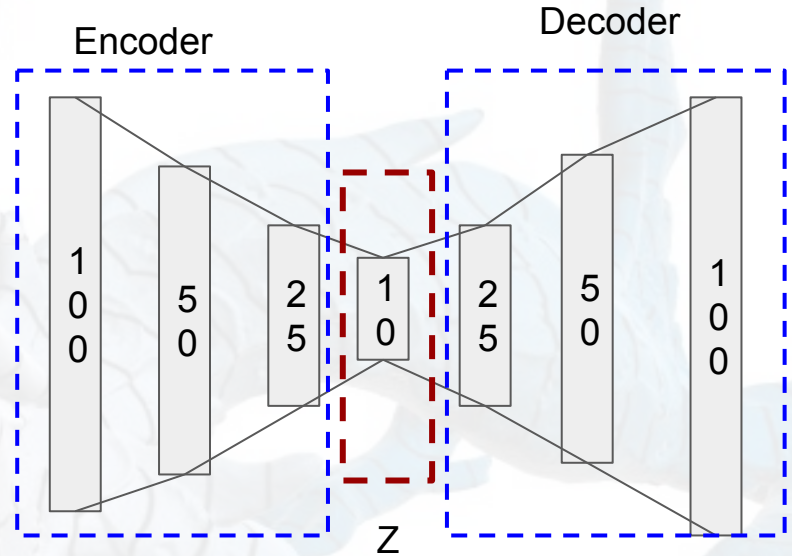


Autoencoder

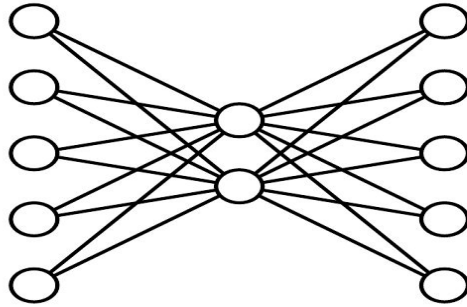
```
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        #Encoder
        self.E1 = nn.Linear(in_features=100, out_features=50)
        self.E2 = nn.Linear(in_features=50, out_features=25)
        self.E3 = nn.Linear(in_features=25, out_features=10)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

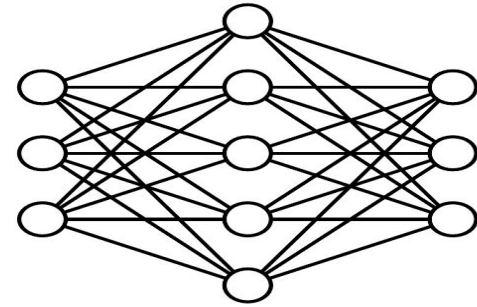
        Output = F.relu(self.D1(Z))
        Output = F.relu(self.D2(Output))
        Output = F.relu(self.D3(Output))
        return Output
```



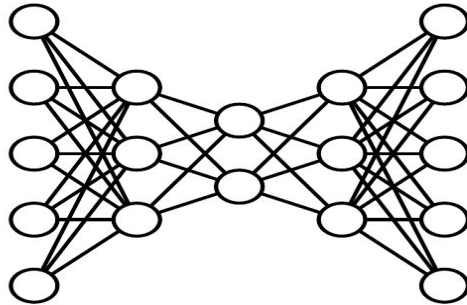
Autoencoder classification by their structure



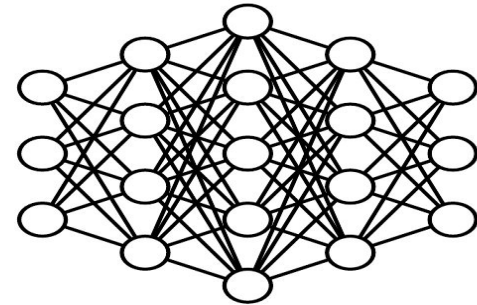
(a) Shallow undercomplete



(b) Shallow overcomplete



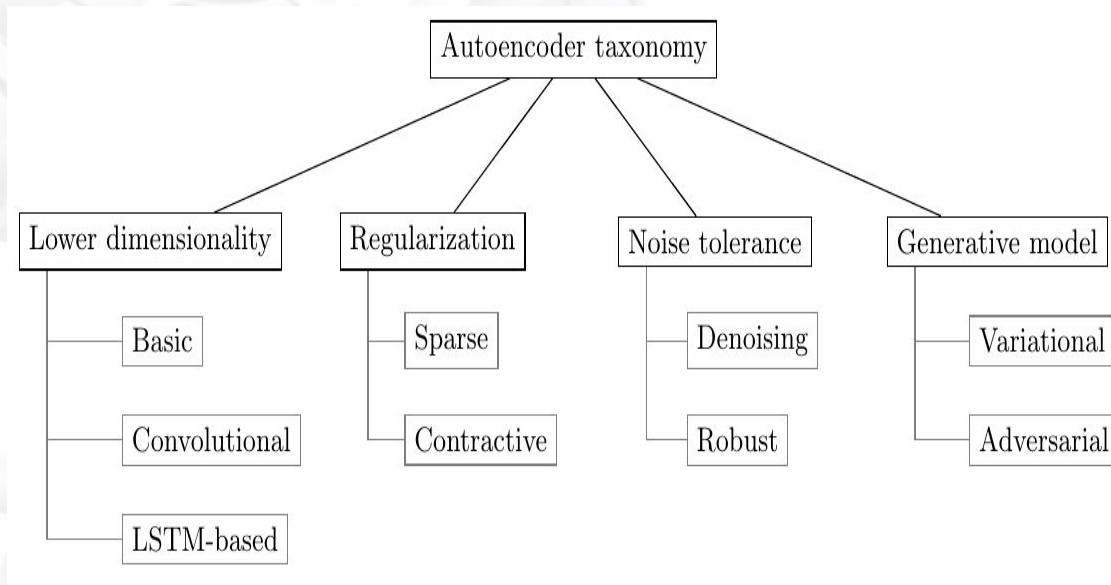
(c) Deep undercomplete



(d) Deep overcomplete

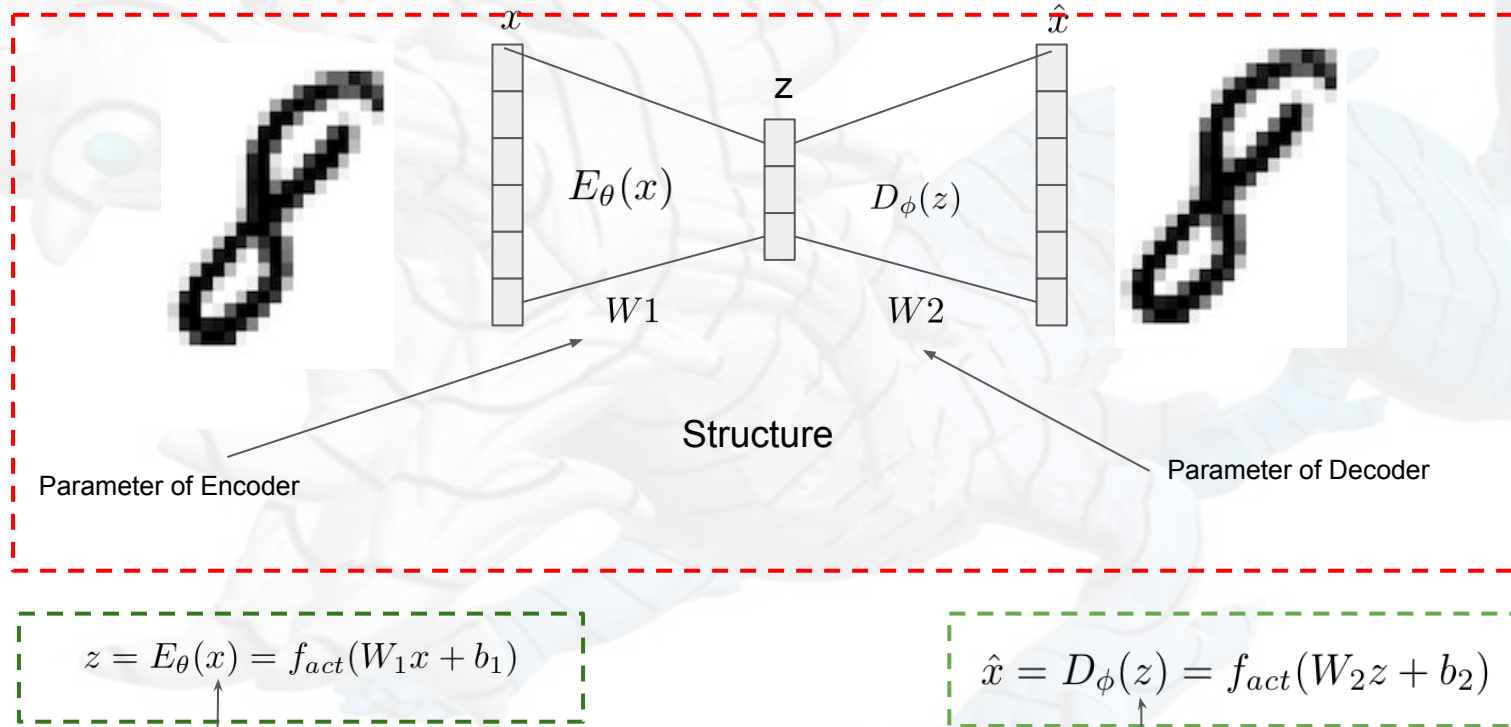
Autoencoder model according to their structure. ([David Charle, 2018](#))

Autoencoder classification by Taxonomy

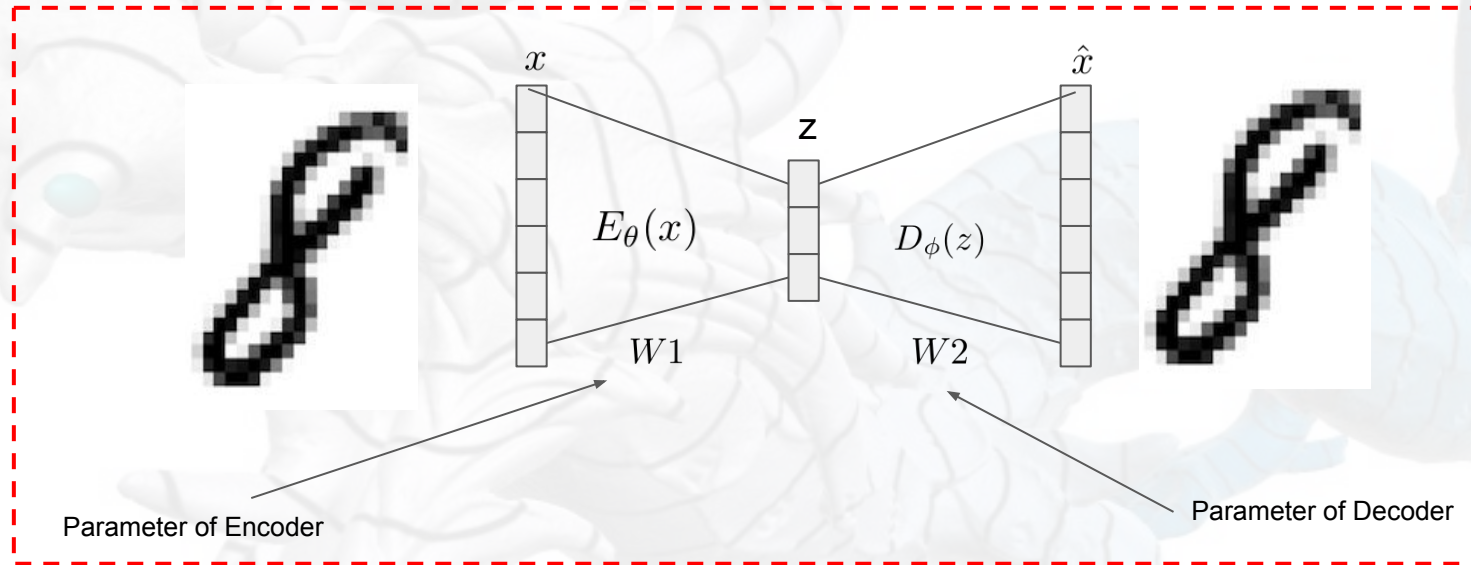


Autoencoder model according to their structure. ([David Charle, 2018](#))

Lower dimensionality: Basic autoencoder



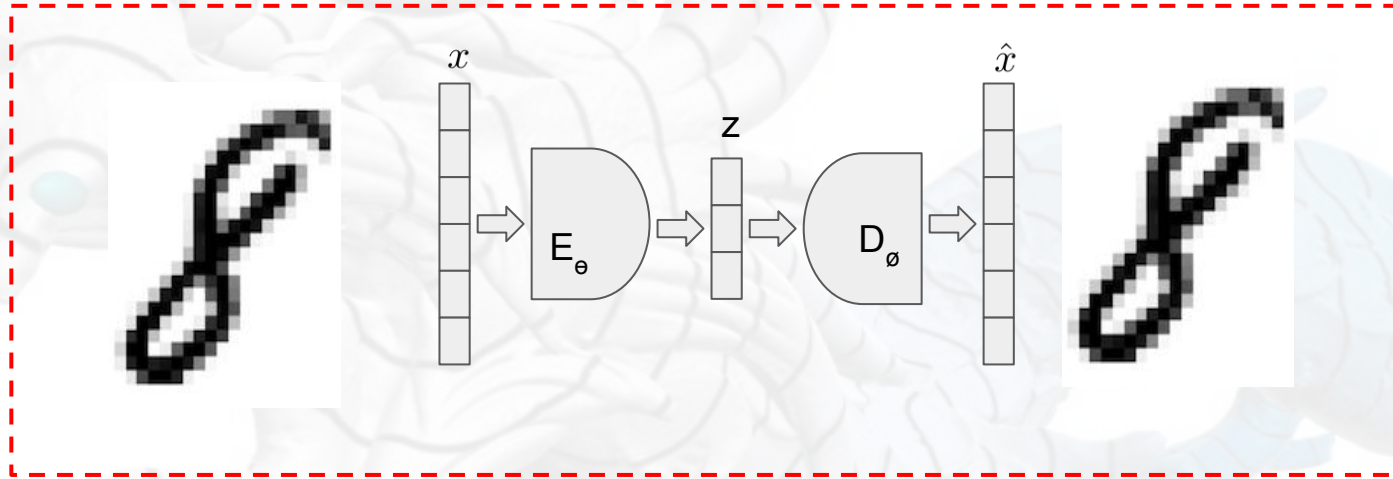
Lower dimensionality: Basic autoencoder



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

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

Lower dimensionality: Basic autoencoder



Regularization (weight decay)

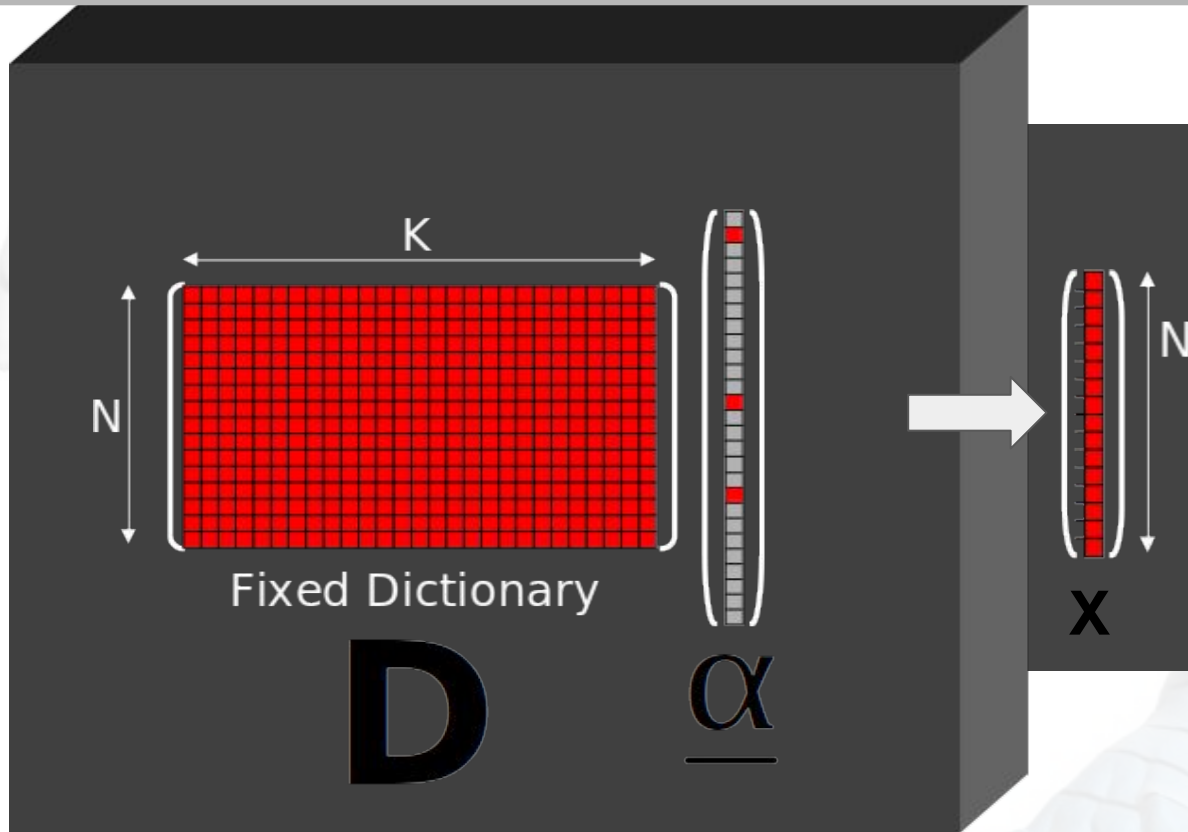
$$\mathcal{L}(x, \hat{x}) = \sum_{x \in S} \mathcal{L}(x, D_{\phi}(E_{\theta}(x))) + \lambda \sum ||w||_p^p$$

Autoencoder: Regularization

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

Loss Function

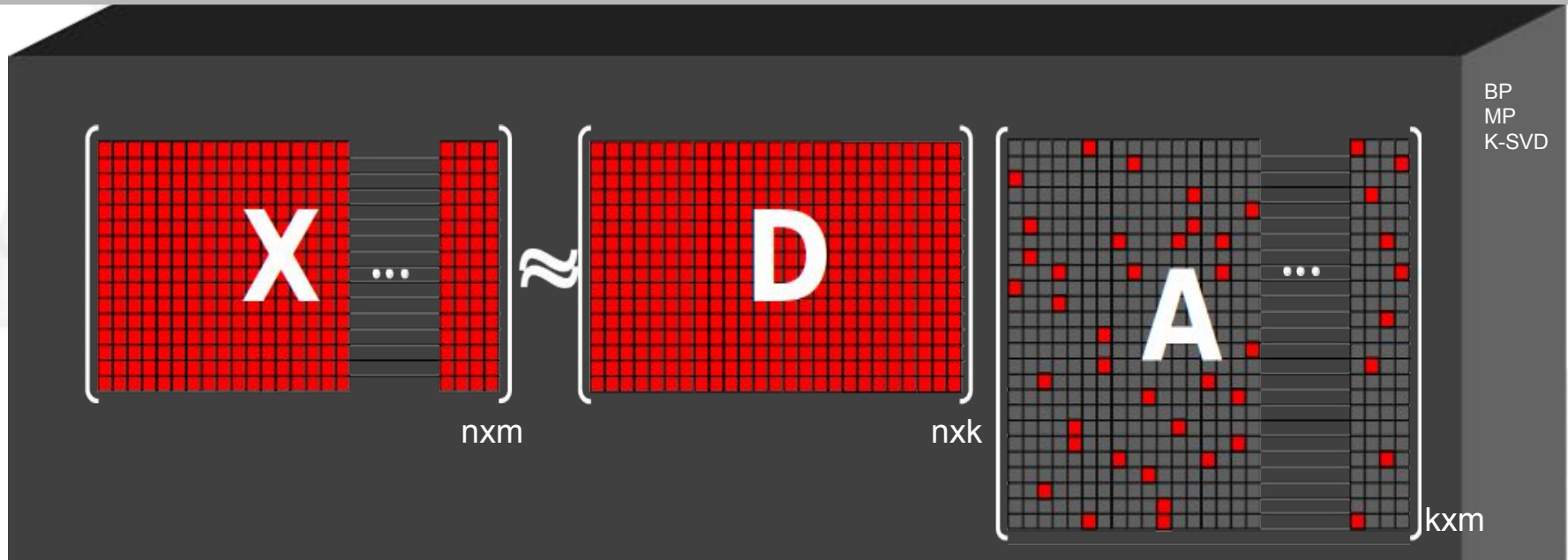
Regularization



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

$$X = D\alpha$$

Regularization: Sparse



$$\|DA - X\|_F^2 \forall j, \|\alpha_j\|_0 \leq L$$

Sparsity Applications

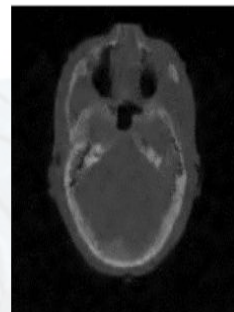
Inpainting



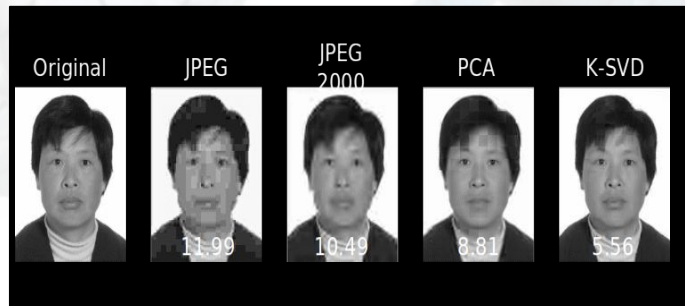
Image inpainting [2, 10, 20, 38] is the process of filling in missing data in a designated region of a still or video image. Applications range from removing objects from images to restoring damaged paintings and photographs. The goal is to produce a revised image in which the missing data is seamlessly merged into the image in a way that is not detectable by a typical viewer. Traditionally, inpainting has been done by professional artists. For photographs, inpainting is used to revert deterioration from aging, such as scratches and dust spots in film. It is also used to remove elements (e.g., removal of stamped text) from photographs, the infamous "airbrush" technique [20]. A current active area of research is inpainting of video sequences [39].



Denoising



Compression



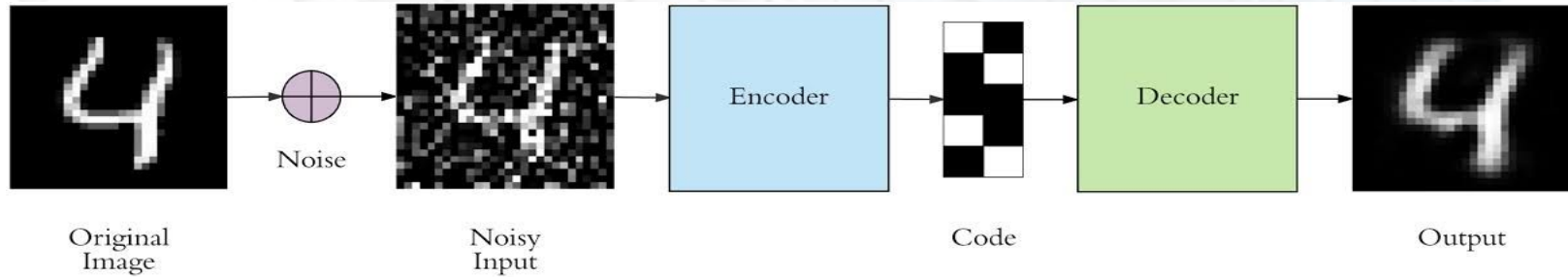
[source click](#)

Sparse Autoencoder



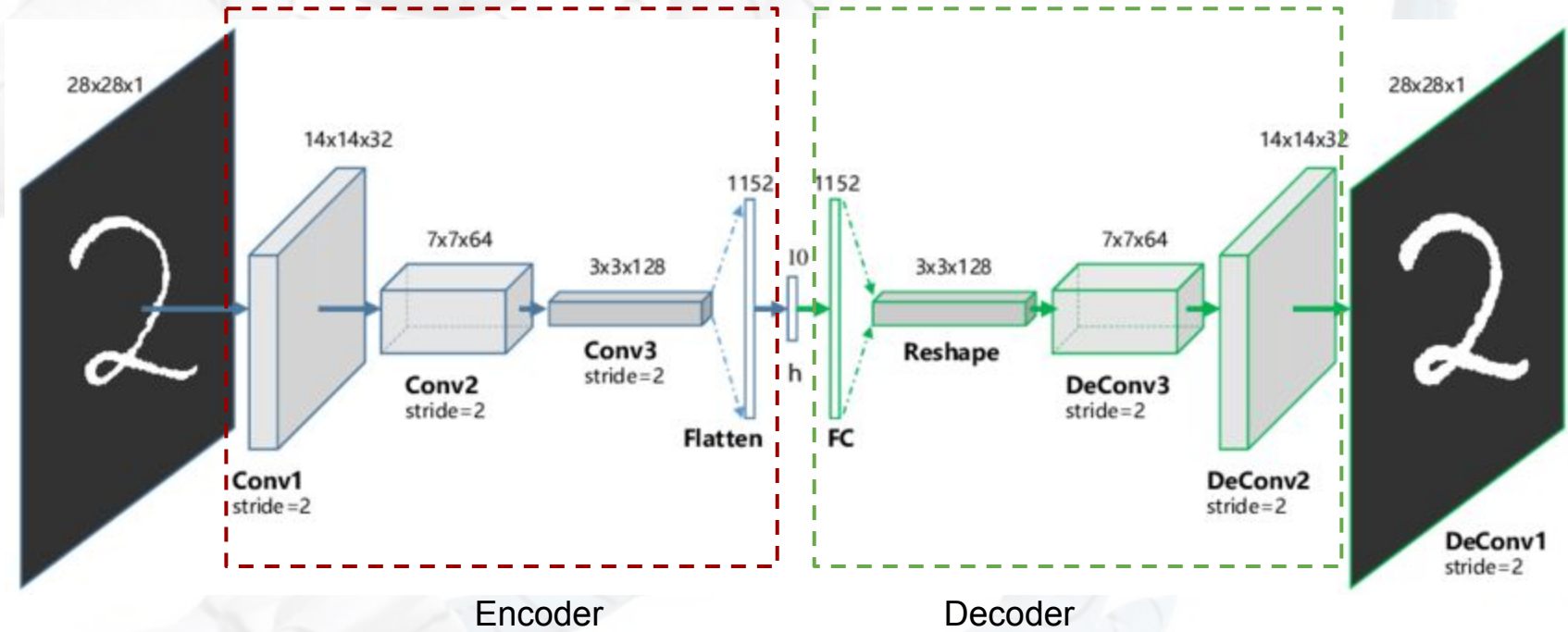
Denoising Autoencoders

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



[Pytorch Example](#)

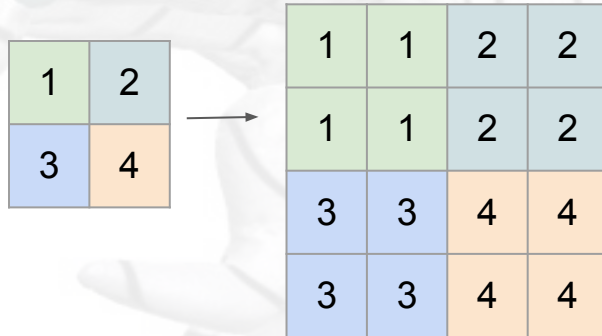
Convolutional Autoencoders



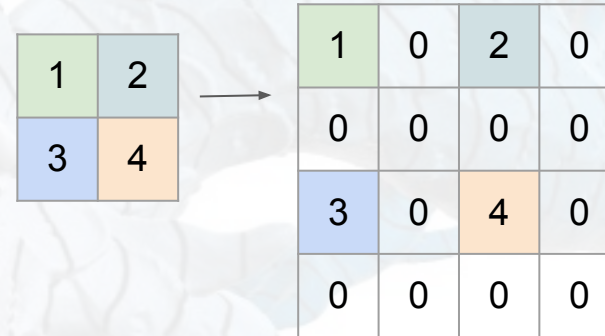
source: [click](#)

Convolutional Autoencoders: Unpooling

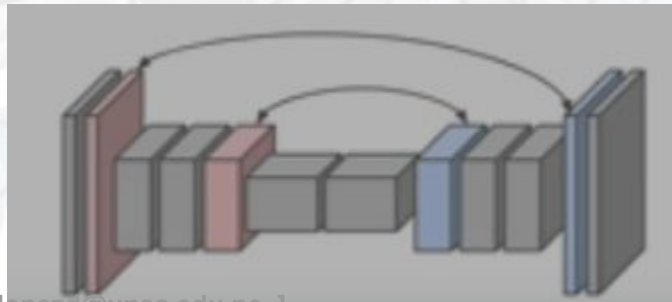
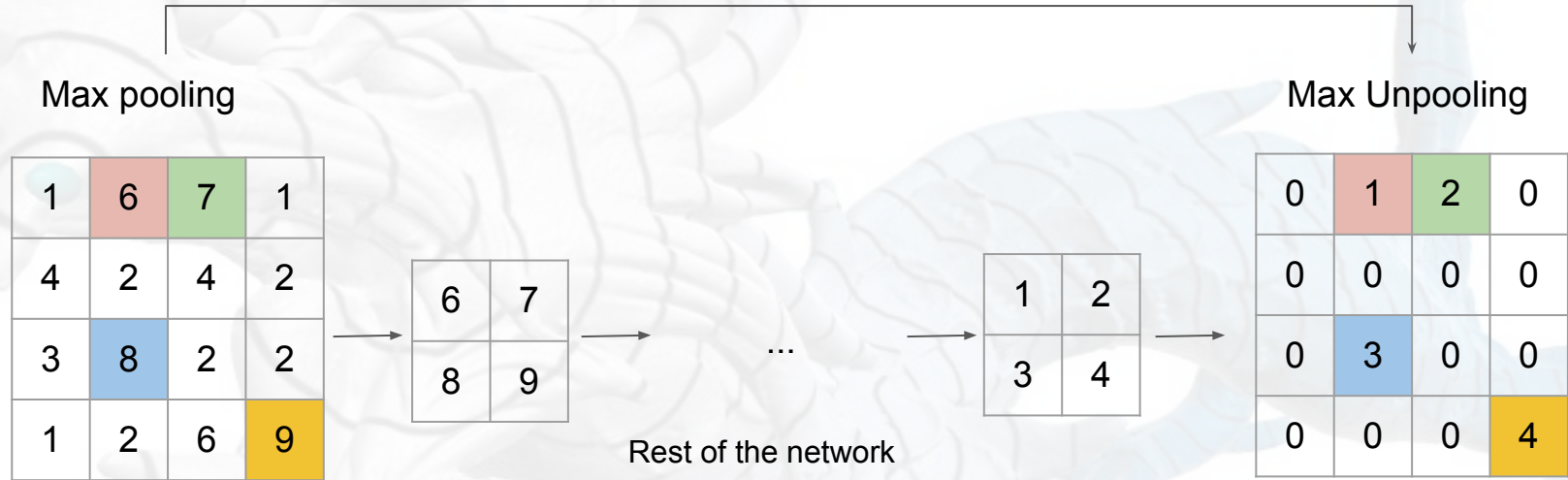
Nearest Neighbor



Bed of Nails



Convolutional Autoencoders : Max Unpooling

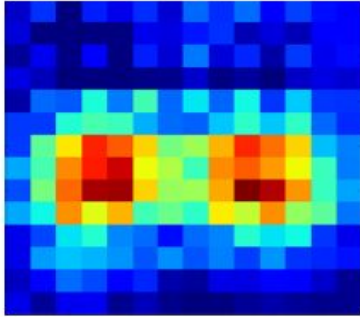


Source: [Video](#)

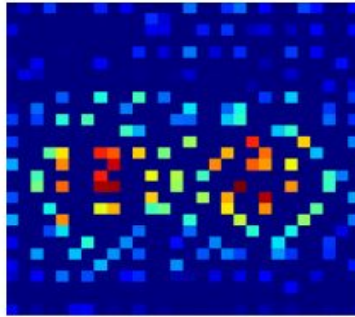
Convolutional Autoencoders : Unpooling



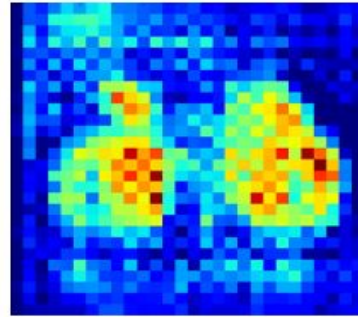
(a)



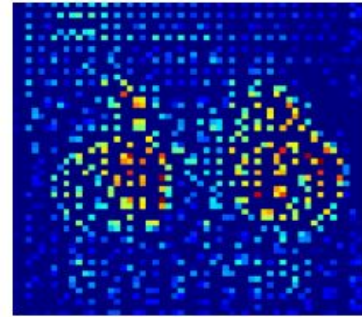
(b)



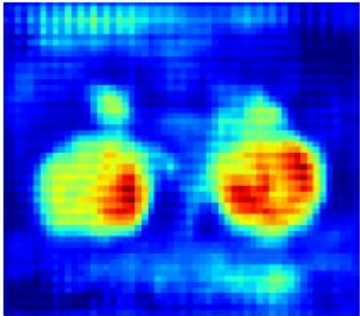
(c)



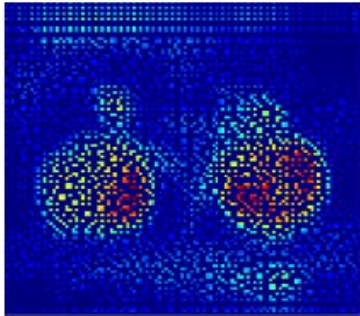
(d)



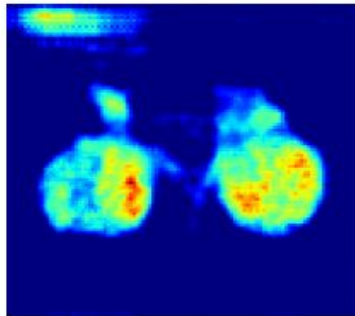
(e)



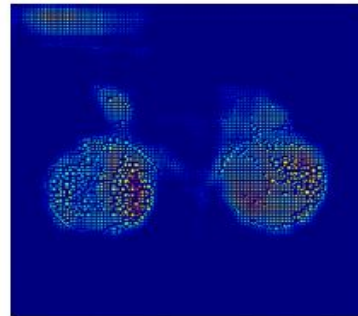
(f)



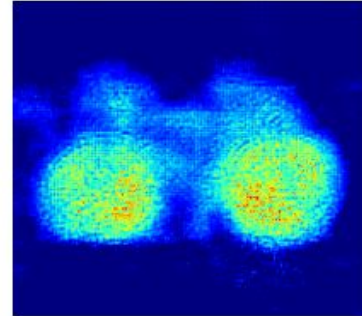
(g)



(h)



(i)

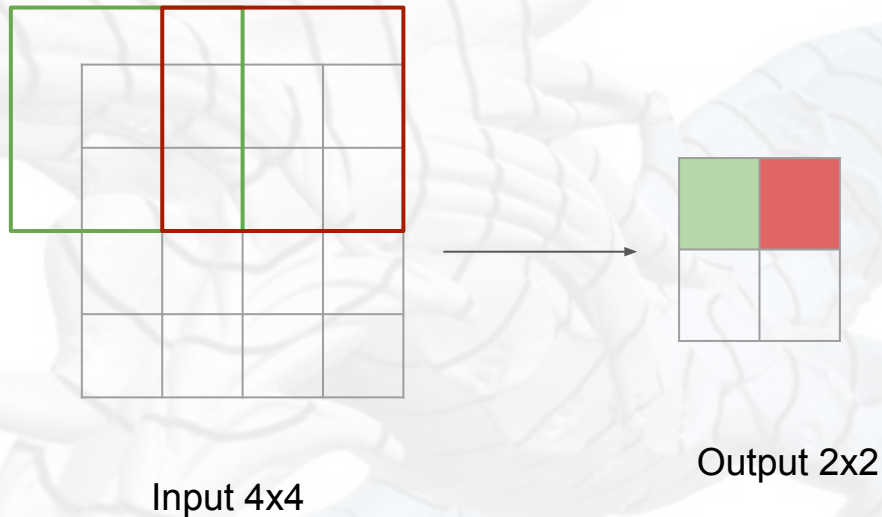


(j)

Source: [click](#)

Convolutional Autoencoders : Transpose Convolution

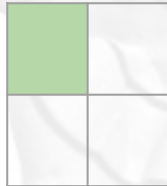
Kernel 3x3, stride 2, pad 1



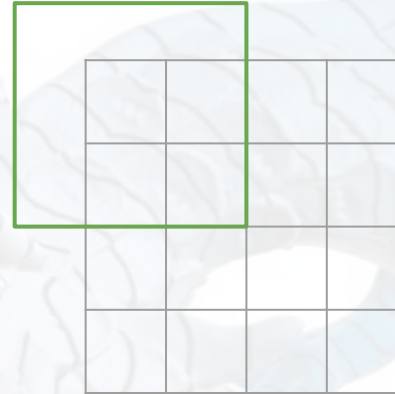
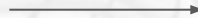
Source: [Video](#)

Convolutional Autoencoders : Transpose Convolution

Kernel 3x3, stride 2, pad 1



Input 2x2

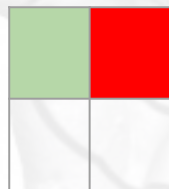


Output 4x4

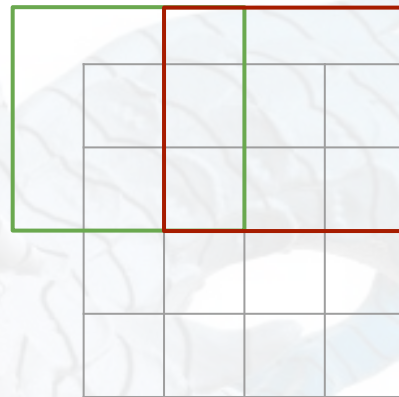
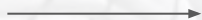
Source: [Video](#)

Convolutional Autoencoders : Deconvolution - Upconvolution

Kernel 3x3, stride 2, pad 1



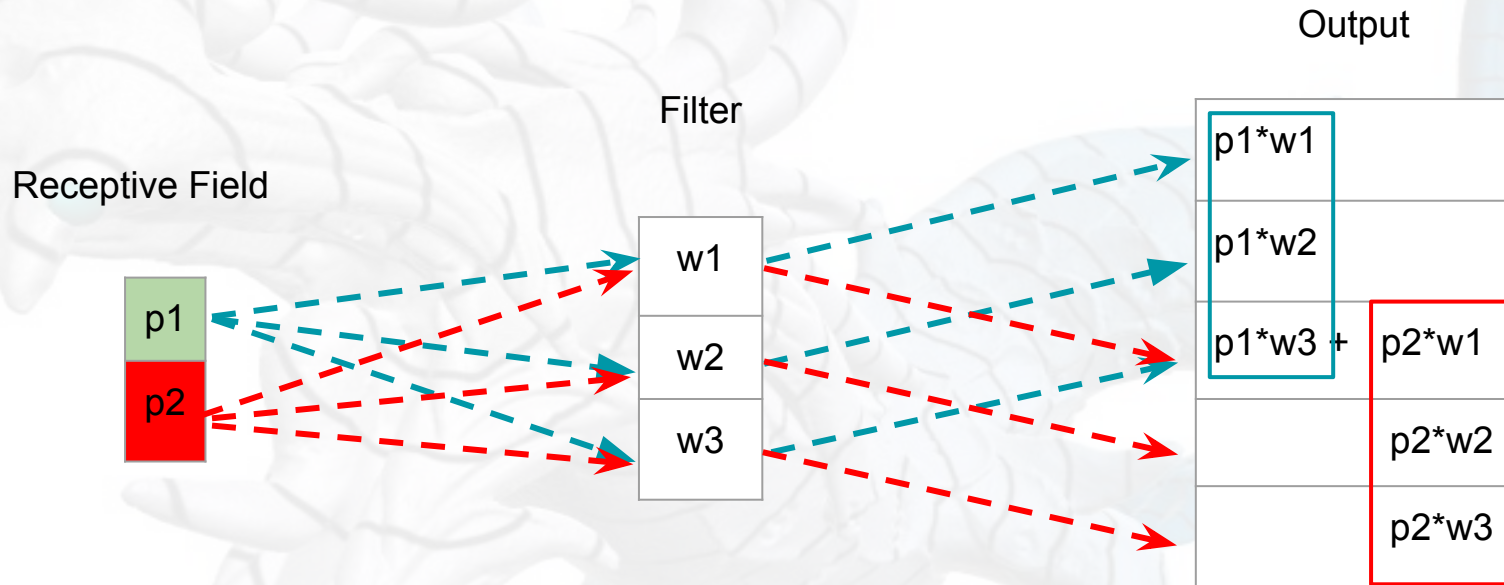
Input 2x2



Output 4x4

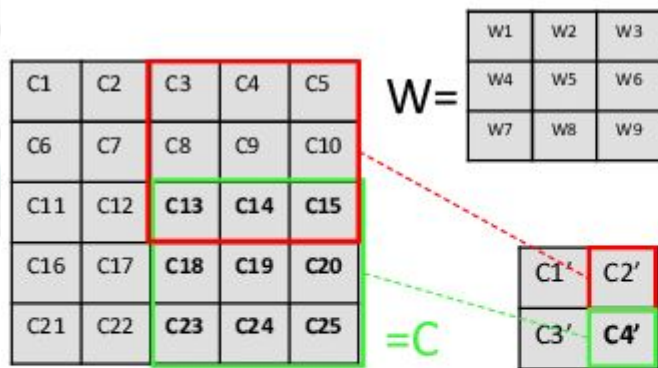
Source: [Video](#)

Convolutional Autoencoders : Deconvolution - Upconvolution



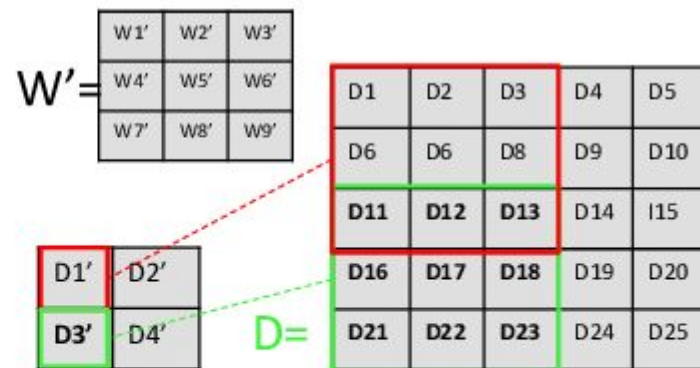
Source: [Video](#)

Convolutional Autoencoders : Deconvolution - Upconvolution



$$C4' = W * C$$

Convolution (stride=2)

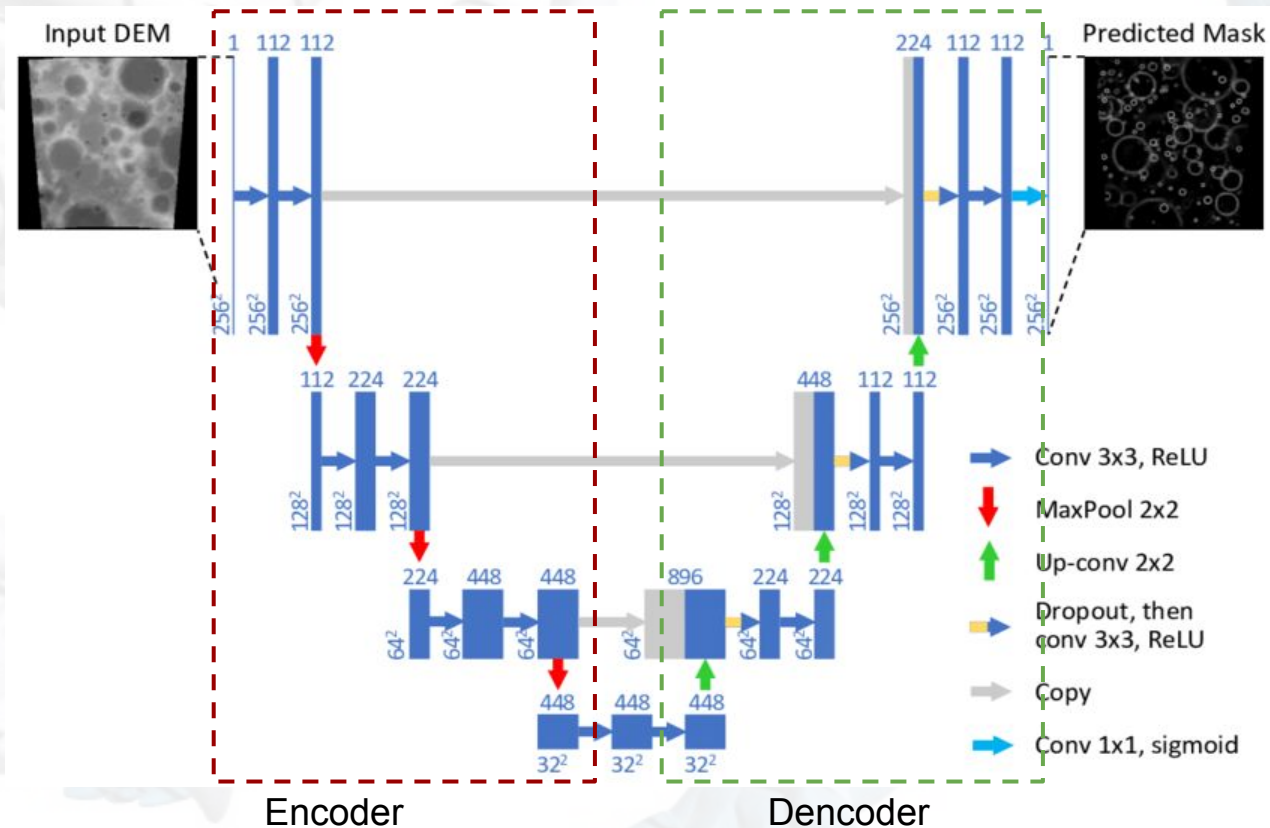


$$D = D3' \times W'$$

Deconvolution (stride=2)

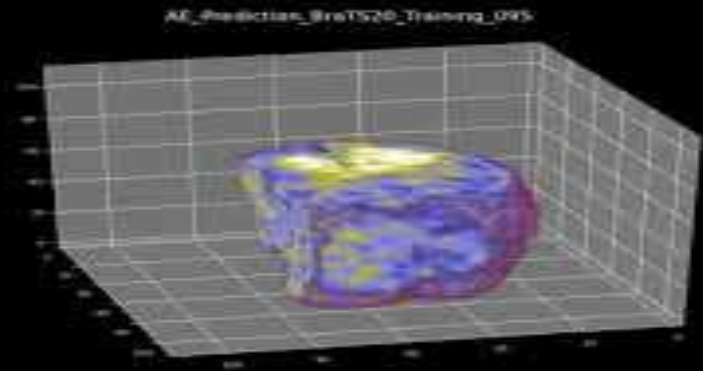
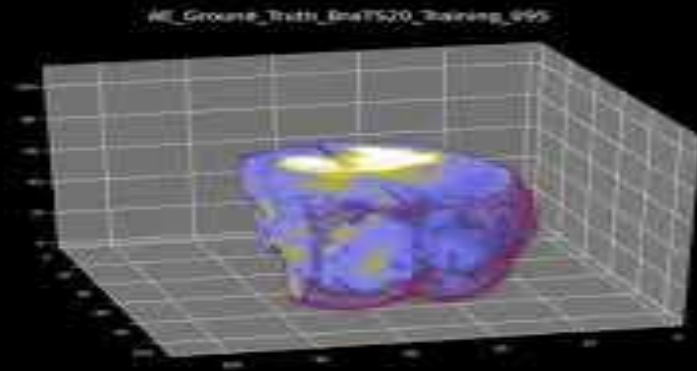
Source: [Video](#)

Convolutional Autoencoders

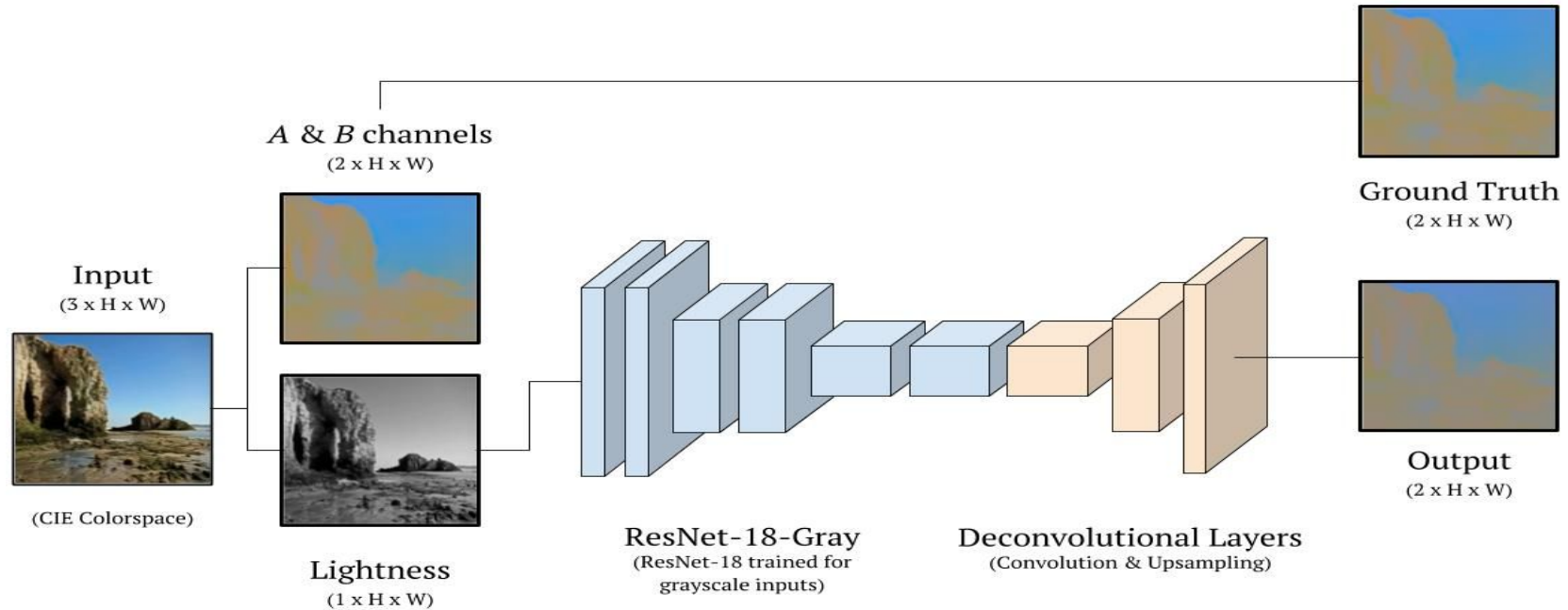


source: [Unet arxiv](#)

convolutional autoencoders example



convolutional autoencoders example



Source: [click](#)

convolutional autoencoders example



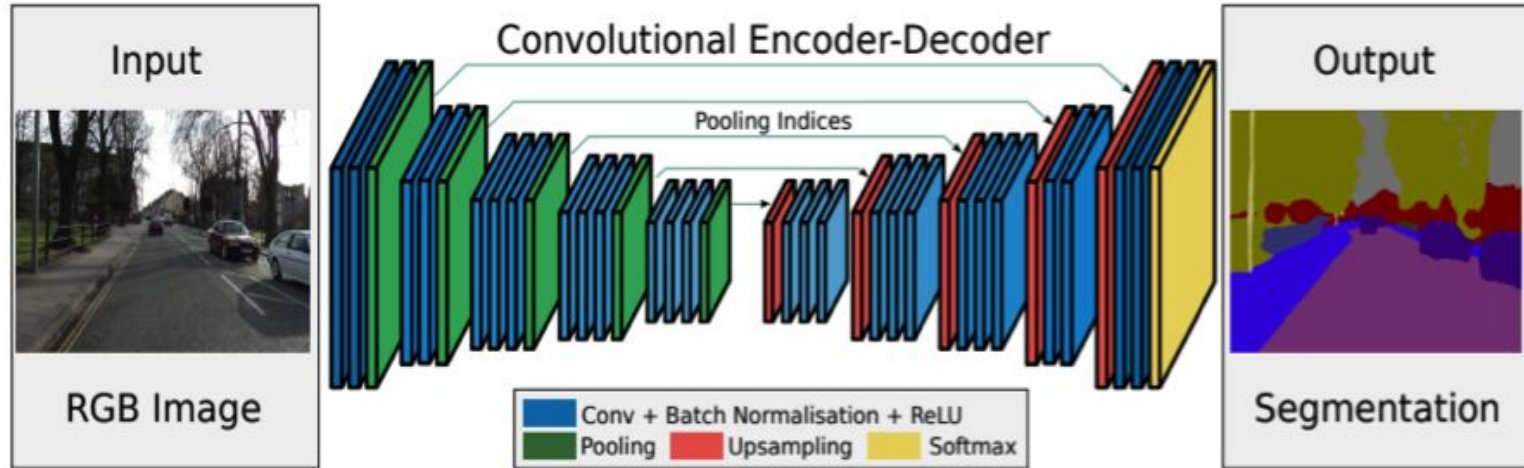
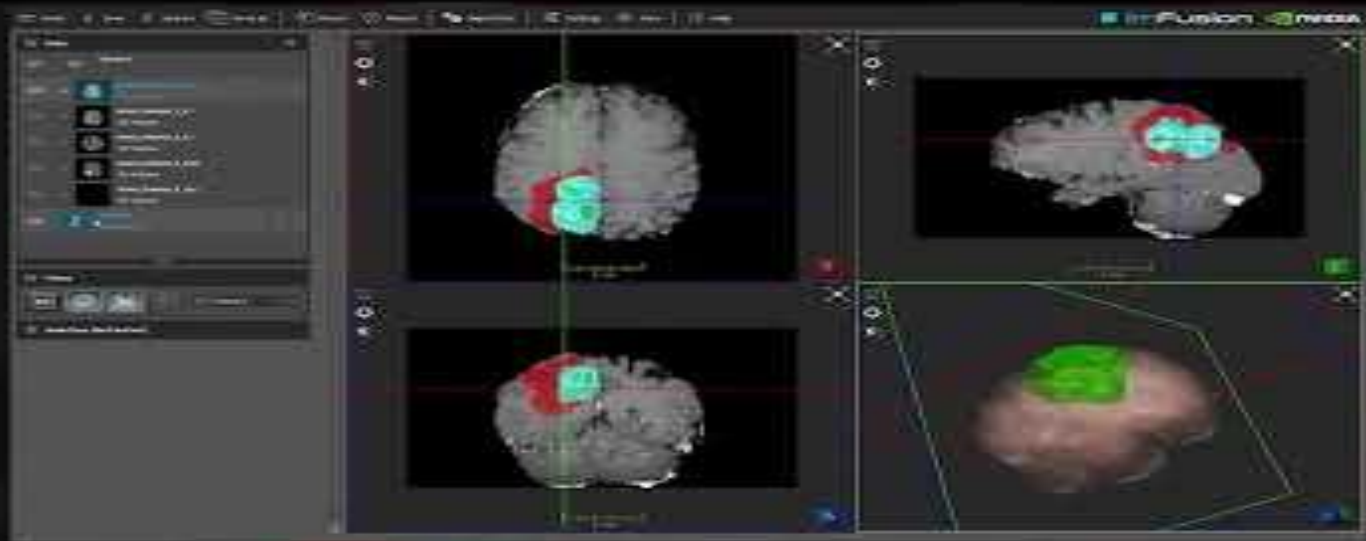


Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to 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



Due to a limited training dataset size,
a variational autoencoder branch is added.

convolutional autoencoders example



convolutional autoencoders example

[Pytorch Example](#)

