Image Segmentation

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DECSAI

DaSCI
Instituto Andaluz de Investigación en
Data Science and Computational Intelligence

Readings

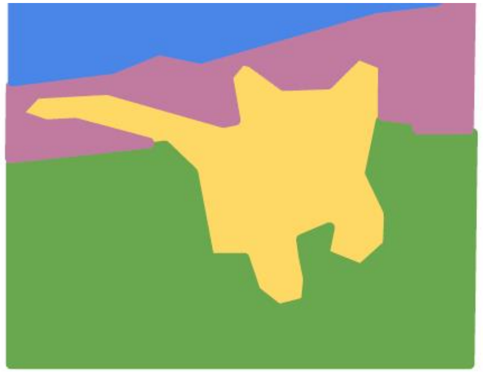
- Minaee et al. (2021). Image segmentation using deep learning: A survey. IEEE Trans. on Pattern Analysis and Machine Intelligence, 44(7), 3523-3542.
- Szeliski (2022), Chapter 6.4.
- Zhang, Lipton, Li and Smola (2023), *Dive into Deep Learning*, Chapter 14.9-14.11.
- Stanford University CS231n (2023): Deep Learning for Computer Vision. Lecture 11.

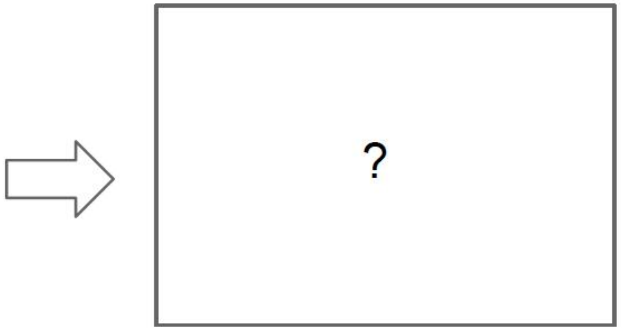
More classical approaches:

- Forsyth & Ponce (2012). Chapter 9.
- Sonka, Hlavac & Boyle (2015). Chapter 6 and 7.

Next slides are mainly based on those from [cs231n]

Semantic Segmentation: the problem





GRASS, CAT,
TREE, SKY, ...

Paired training data: for each training image, each pixel is labeled with a semantic category.

At test time, classify each pixel of a new image.

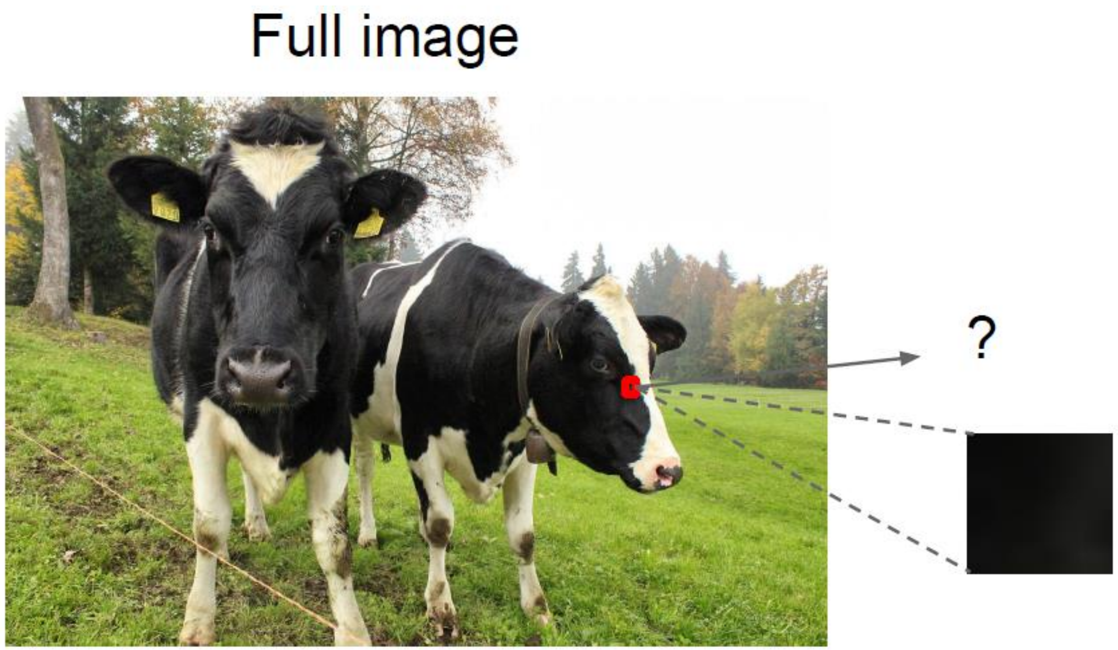
We want to predict a mask
(where the object of interest is present).

If we just have one type of object → binary mask
(0: background; 1: object)

Image Segmentation = Pixel-level Classification

When you perform the prediction for every pixel → dense prediction

Semantic Segmentation: the problem



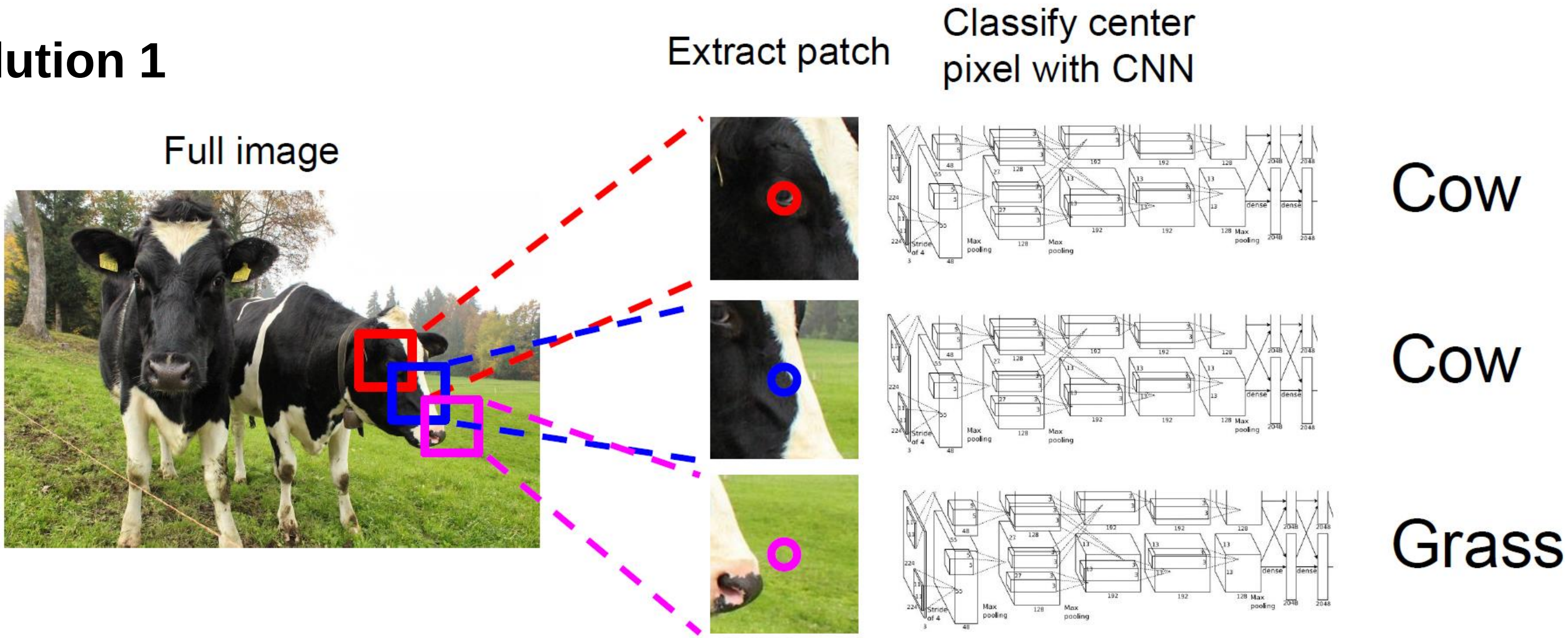
Can we perform segmentation using a single pixel?

Impossible to classify without any context

What context and how do we include it?

Semantic Segmentation

Solution 1



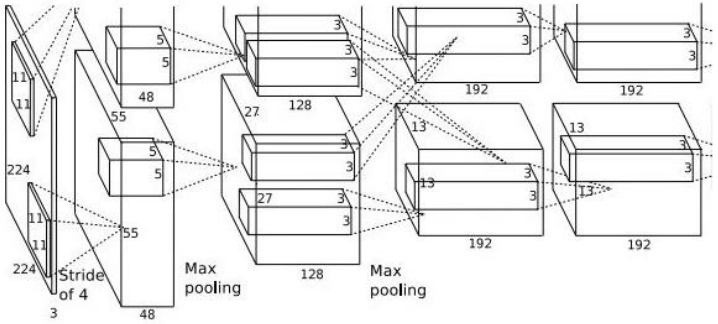
Extract patches/regions of fixed size and use a ConvNet to classify the central pixel!

Problems:
- Very inefficient! We treat every patch independently. Not reusing shared features between overlapping patches
- How to select patch size?

Semantic Segmentation

Solution 2

Full image



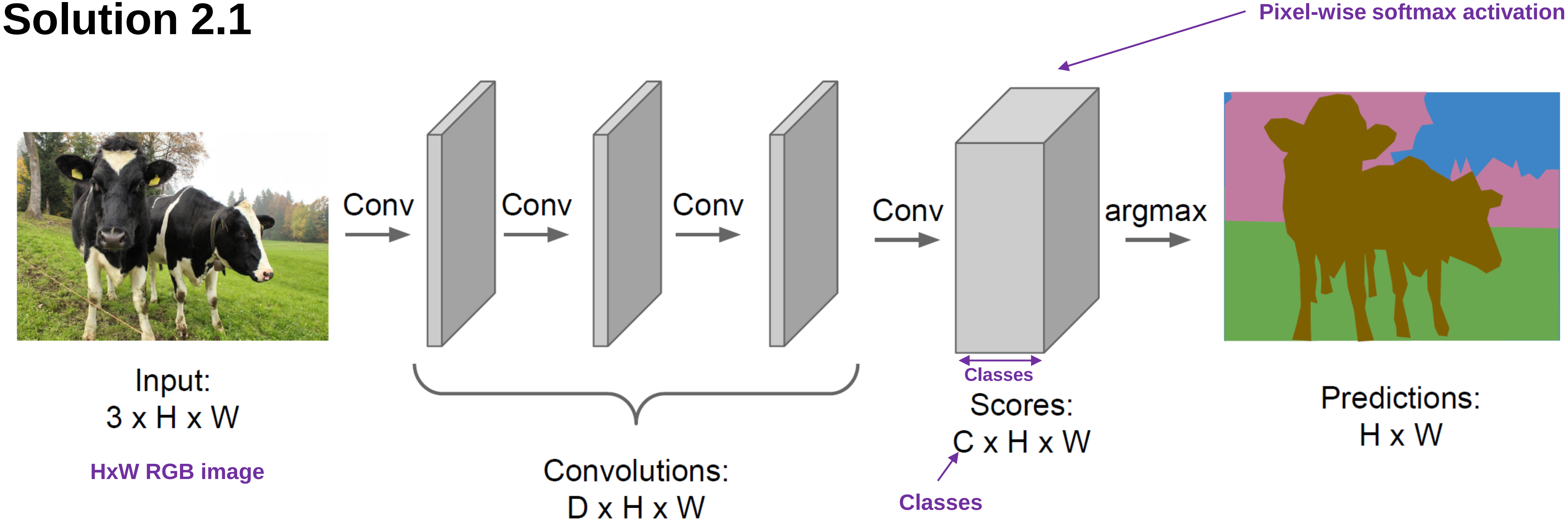
Encode the entire image with ConvNet, and do semantic segmentation on top!

Problem:

- classification architectures often reduce feature spatial sizes to go deeper, but **semantic segmentation requires the output size to be the same as input size**

Semantic Segmentation

Solution 2.1



Design a network with only **convolutional layers without downsampling operators** to make predictions for pixels all at once!!

Problem:

- convolutions at original image resolution will be **very expensive** (memory, number of operations,…)

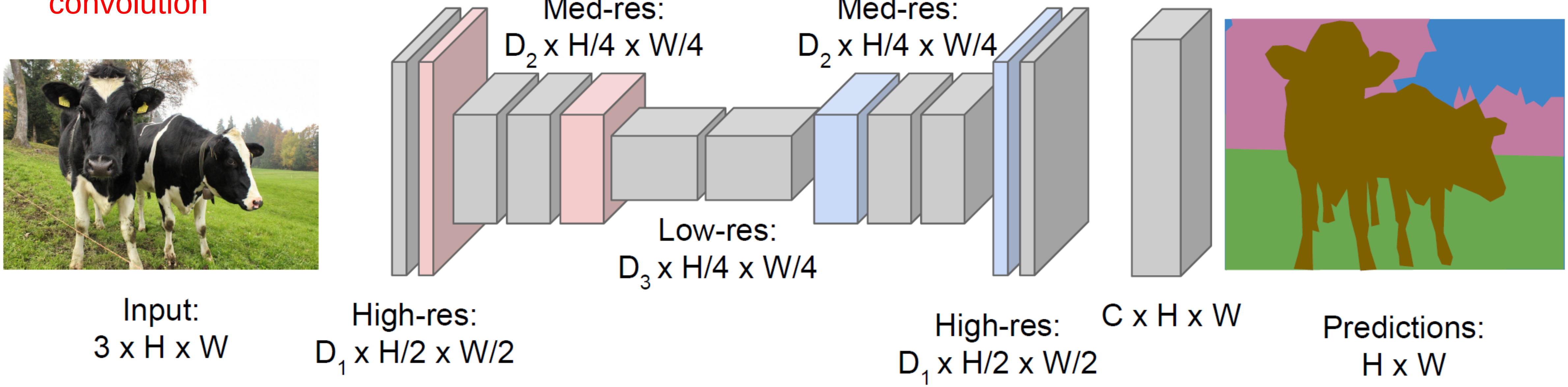
Fully Convolutional Neural Networks

Solution 2.2

Sometimes called hourglass-like models or, more frequently, encoder-decoder architectures

Downsampling:
Pooling, strided convolution

Upsampling:
???



Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Fully Convolutional Neural Networks

In-Network Upsampling: "unpooling"

Nearest Neighbor

1	2
3	4

Input: 2 x 2

→

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

We simply repeat every element

"Bed of Nails"

1	2
3	4

Input: 2 x 2

→

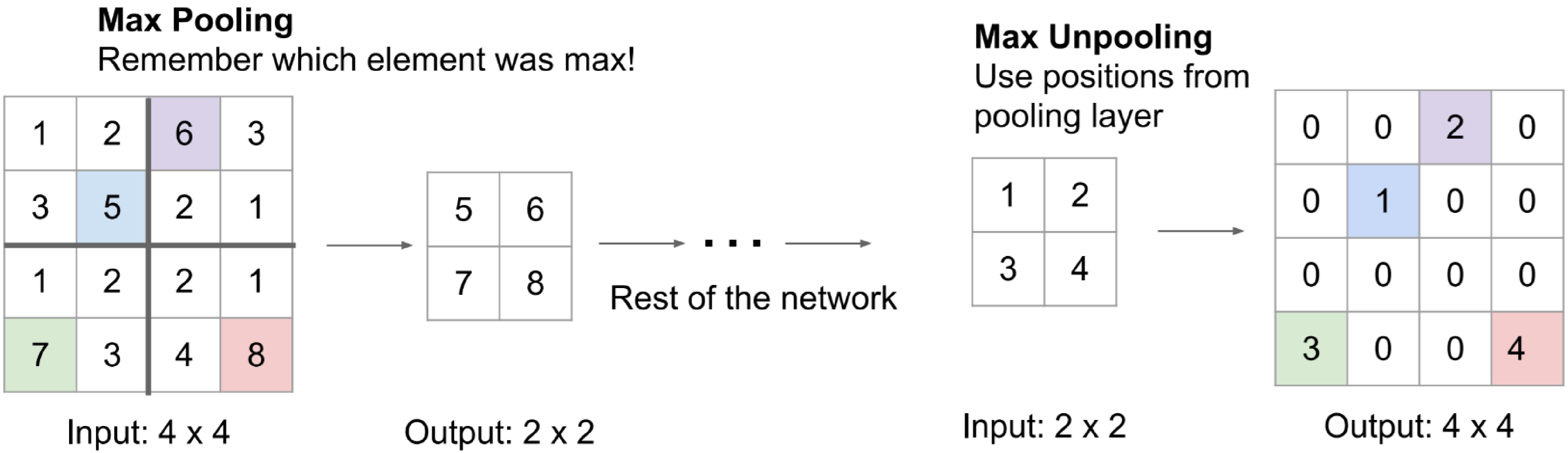
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

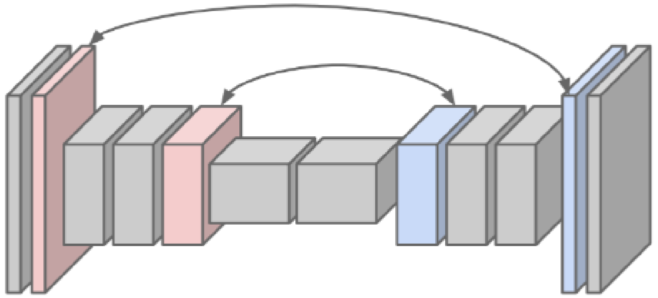
We simply place the value in a particular position in the output, filling the rest with zeros

Fully Convolutional Neural Networks

In-Network Upsampling: “max unpooling”.
Smarter “bed of nails” method. Rather than a predetermined/fixed location for the “nails”, we use the position of the maximum elements from the corresponding max pooling layer earlier in the network.



Corresponding pairs of downsampling and upsampling layers

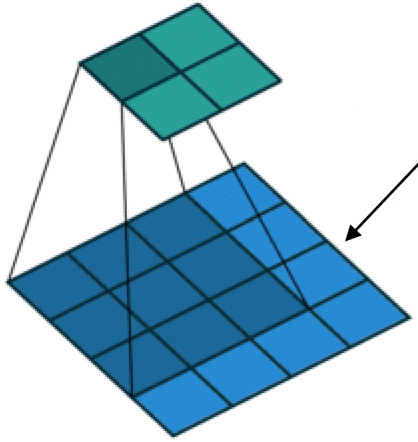


This works because Fully Convolutional Networks are often symmetric!!!

Fully Convolutional Neural Networks

Learnable Upsampling: Transposed Convolution.

Convolution

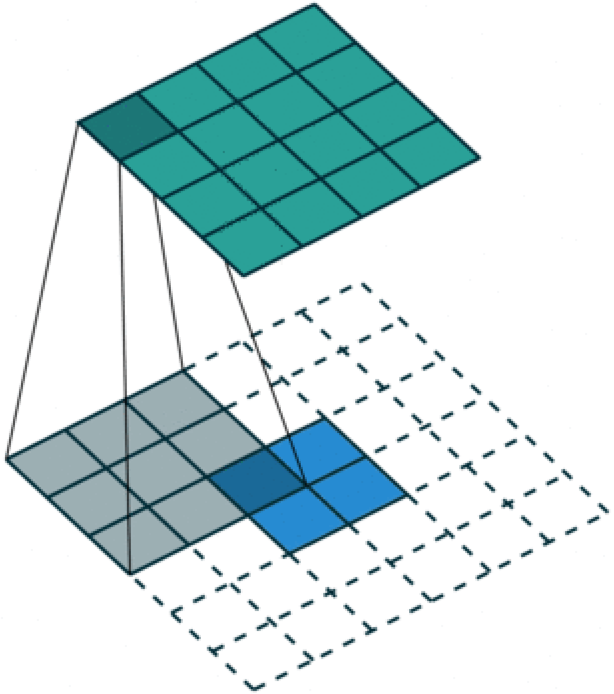


Convolution of a 3×3 kernel on a 4×4 input with unitary stride and no padding.

The transposed convolution can be considered as the operation that allows to **recover the shape (not the input values) of the initial feature map**.

Blue *maps are inputs, and* ***cyan*** *maps are outputs.*

Transposed convolution



The transpose will then have an output of shape 4×4 when applied on a 2×2 input.

https://github.com/vdumoulin/conv_arithmetic

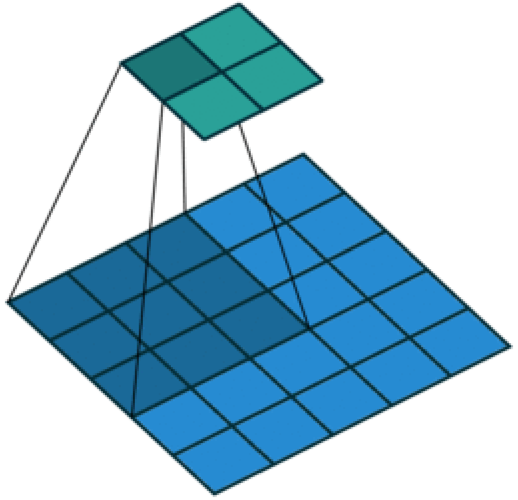
Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

You could also use, for instance, upsampling via bilinear interpolation, but experiments show that **it's commonly better to learn to upsample**. See Long et al., "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Fully Convolutional Neural Networks

Learnable Upsampling: Transposed Convolution.

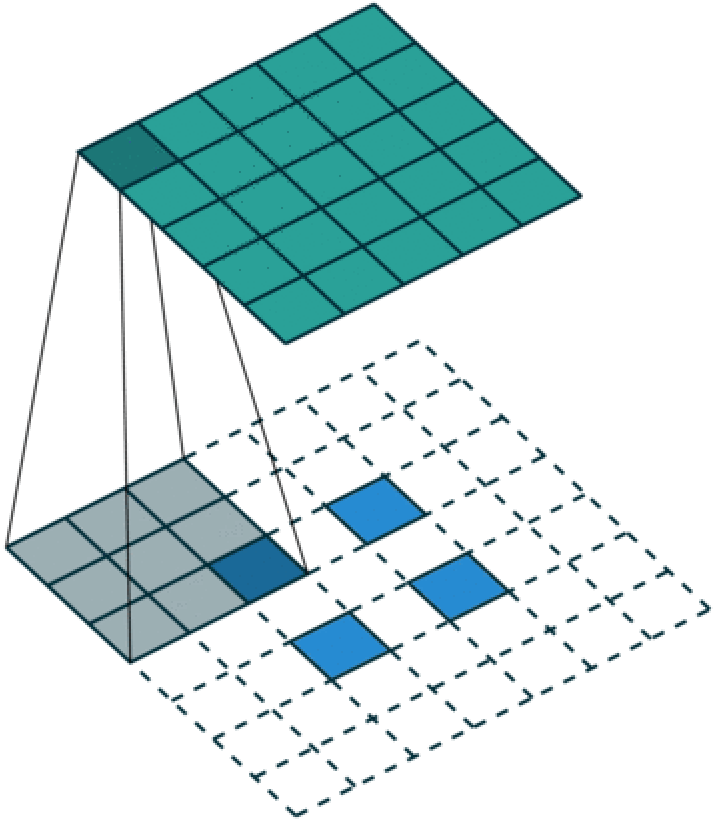
Convolution



Convolution of a 3×3 kernel on a 5×5
input with unitary stride and no padding.

Blue *maps are inputs, and*
cyan *maps are outputs.*

Transposed convolution

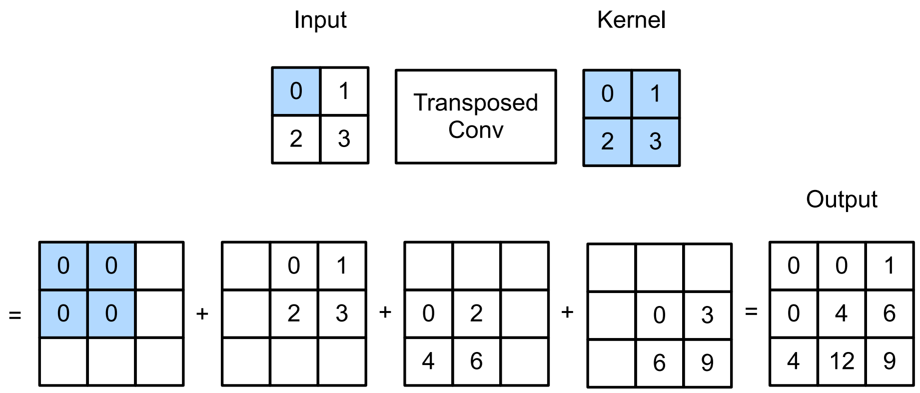


The transpose will then have an output of shape 5×5
when applied on a 2×2 input.

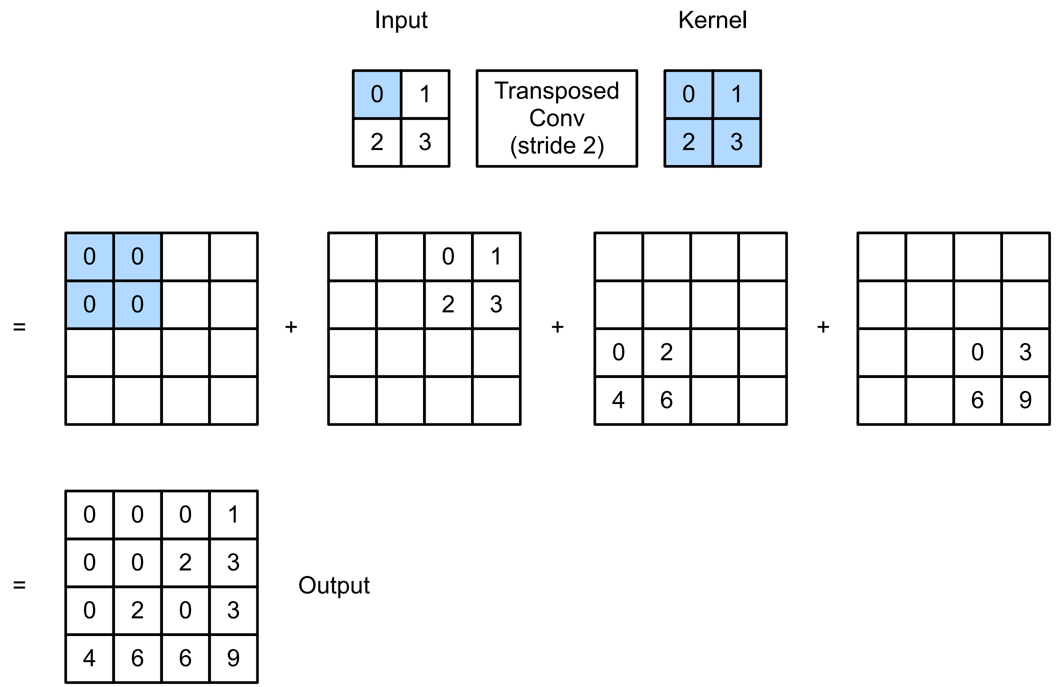
https://github.com/vdumoulin/conv_arithmetic
Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Fully Convolutional Neural Networks

Learnable Upsampling: Transposed Convolution.



Transposed convolution with 2x2 a kernel.



Transposed convolution with 2x2 a kernel with stride 2.

https://d2l.ai/chapter_computer-vision/transposed-conv.html

The transposed convolution is not the inverse of a convolution, and thus *deconvolution* does not seem a suitable name for the operation.

Fully Convolutional Neural Networks

Learnable Upsampling: Transposed Convolution.

We can express convolution in terms of a matrix multiplication. 1D Example:

Kernel

Input signal (6x1)

Output signal (2x1)

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Stride=2

Kernel size=3

Padding=1

Transposed convolution multiplies by the transpose of the same matrix.

Kernel^T

Input signal (2x1)

Output signal (6x1)

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

For filter sizes which produce an overlap in the output feature map, the overlapping values are simply added together.

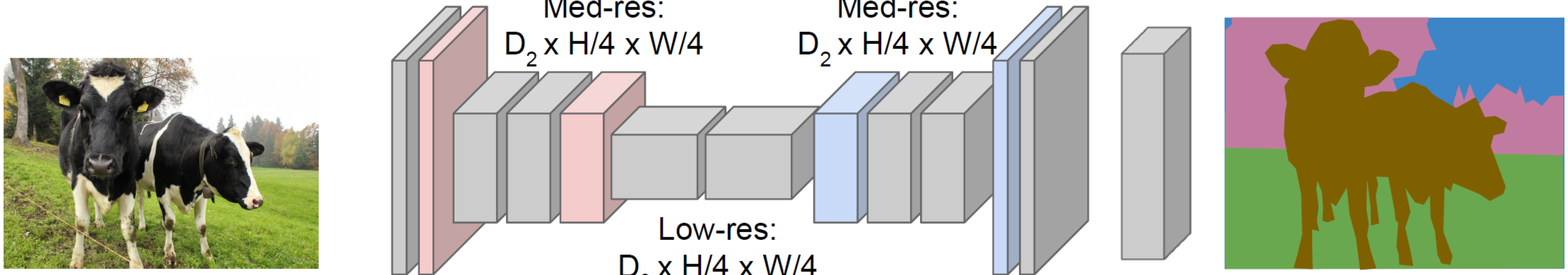
Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Fully Convolutional Neural Networks

Downsampling: Pooling, strided convolution

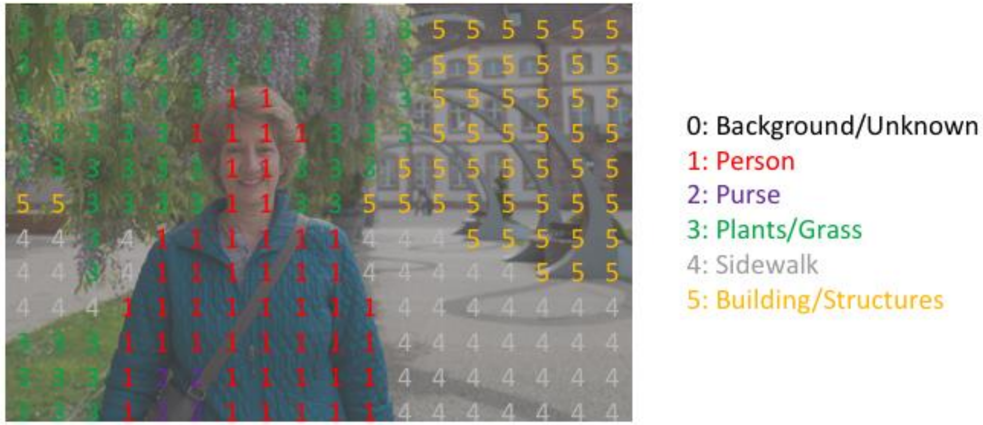
Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling: Unpooling or strided transposed convolution

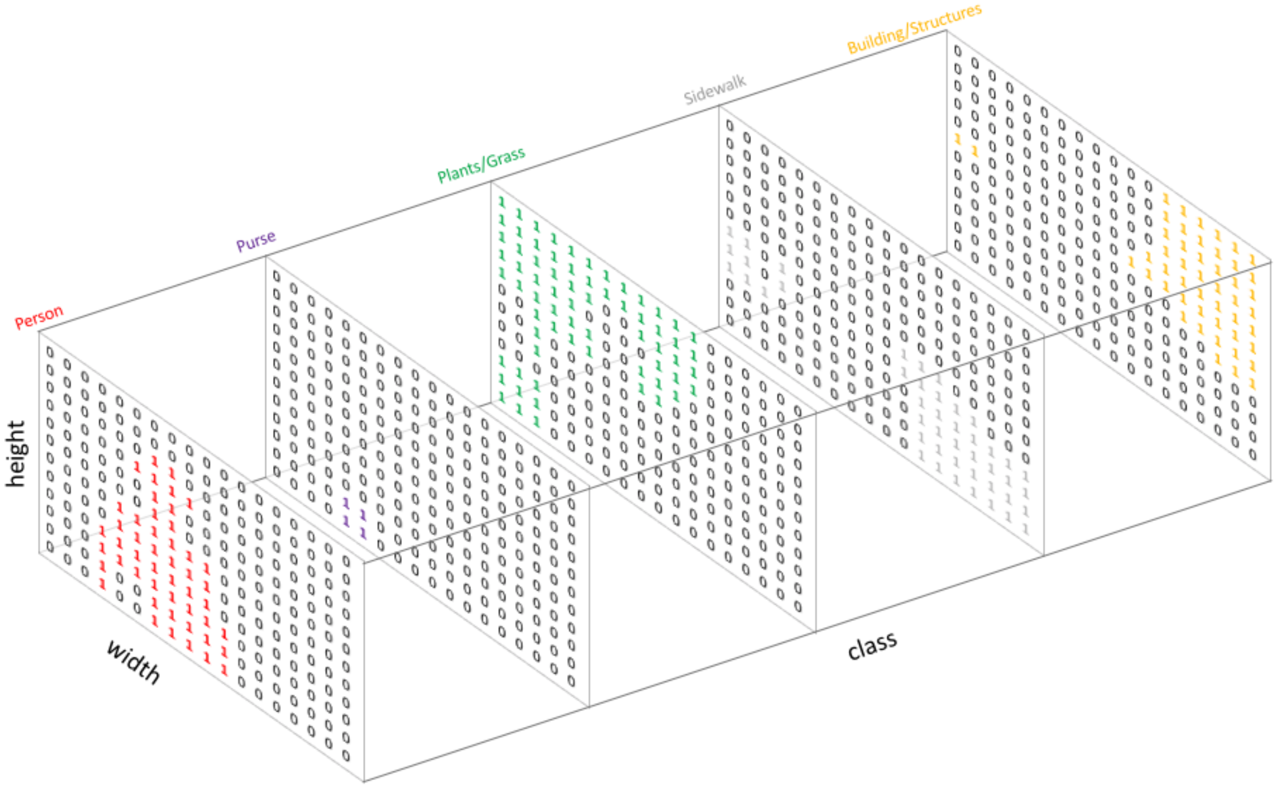


Fully Convolutional Neural Networks

What about the target labels employed for training?



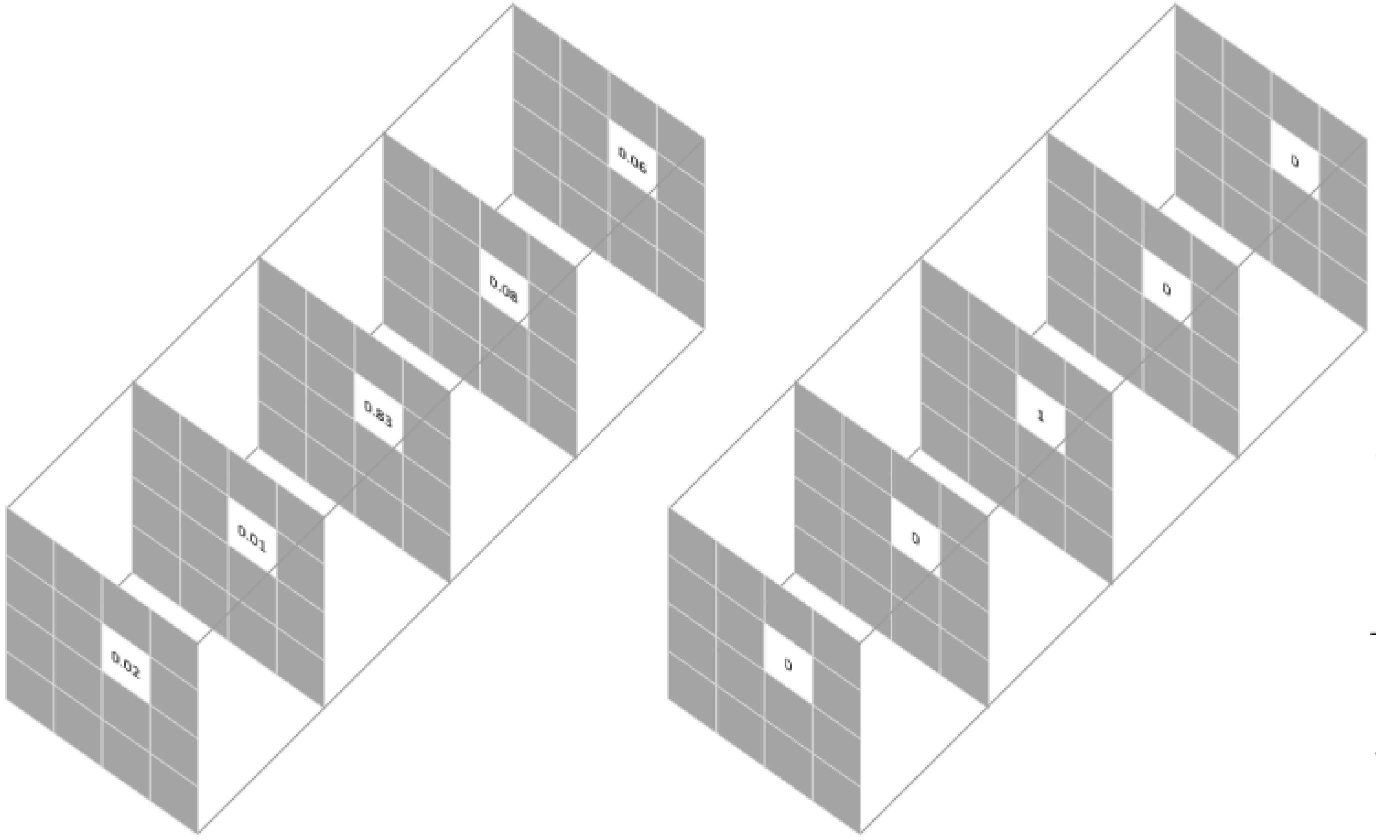
Our target is the one-hot encoding of the class labels (we create an output channel for each possible class)



We can train our segmentation model using a **pixel-wise cross-entropy loss** (having a softmax per pixel).

https://www.jeremyjordan.me/semantic-segmentation/

Fully Convolutional Neural Networks



Prediction for a selected pixel

Target for the corresponding pixel

Pixel-wise loss is calculated as the log loss, summed over all possible classes

$$-\sum_{classes} y_{true} \log\left(y_{pred}\right)$$

This scoring is repeated over all **pixels** and averaged

https://www.jeremyjordan.me/semantic-segmentation/

We're assuming equal learning to each pixel in the image. This can be a **problem if your classes have unbalanced representation in the image**, as **training can be dominated by the most prevalent class**.

Possible solution:
- weighting strategy: enhance importance of minority classes

Sugino, Takaaki, et al. "Loss weightings for improving imbalanced brain structure segmentation using fully convolutional networks." *Healthcare*. Vol. 9. No. 8., 2021.

Other popular segmentation losses are also available, like the Dice loss or the Focal loss.

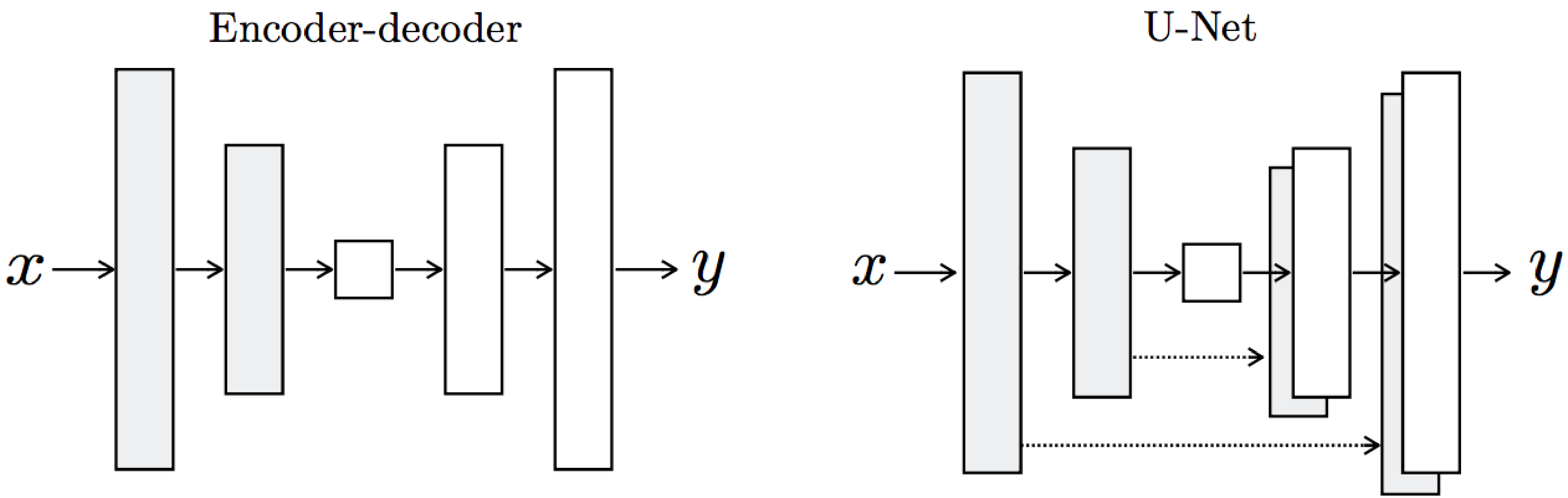
Image Classification vs Image Segmentation

- In Image Classification feature resolution decreases as we go deeper into the network.
  - Help reduce computational cost (memory, operations) while preserving transformation invariance.

- But in Image Segmentation spatial information is necessary.
  - **As we go deeper into the network**, **feature maps become increasingly good at encoding "what is in the image?" but information about "where in the image?" is lost**.

- Let's see two strategies to deal with this: U-Net and DeepLab

U-Net



[Image Source](#)

Ronneberger, Fischer, and Brox. "U-Net: Convolutional networks for biomedical image segmentation." *MICCAI 2015* (>75K citations)

U-Net

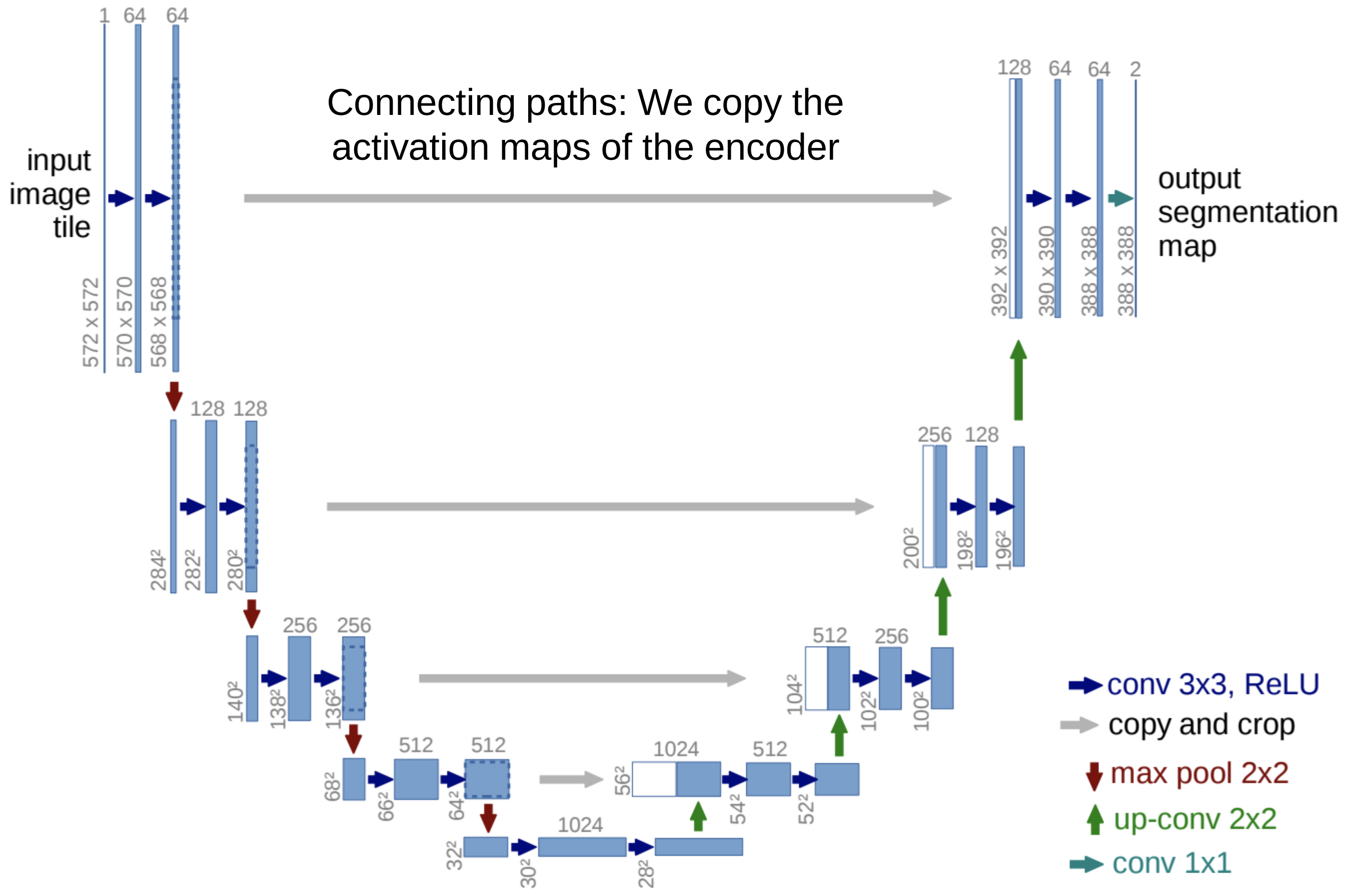


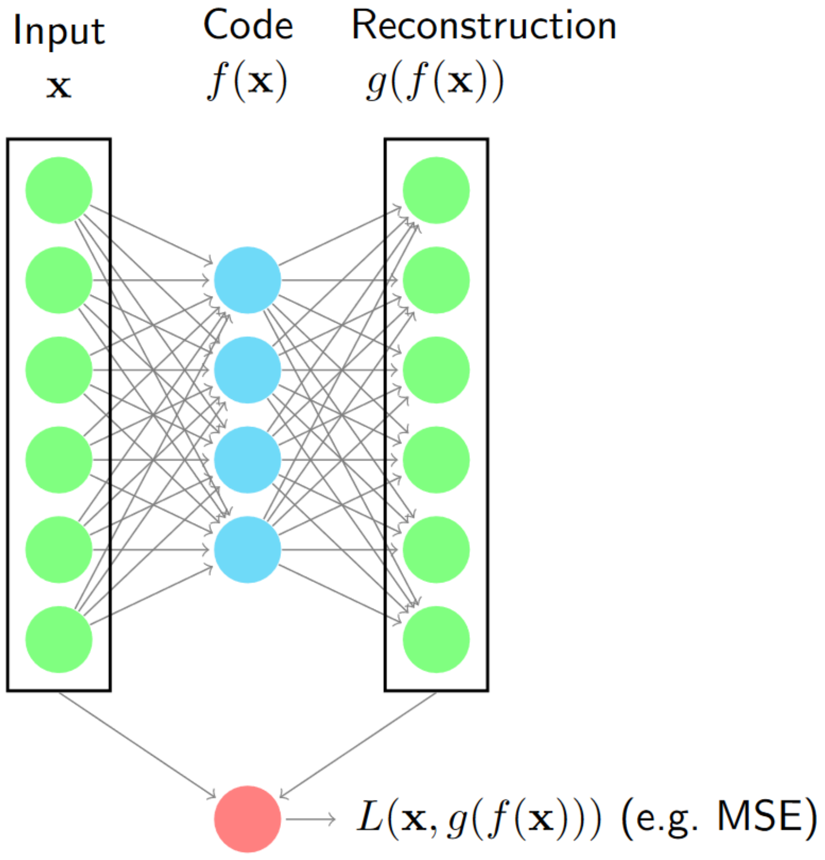
Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

- ConvNet with an encoder-decoder architecture (U-shape).
- Thanks to the connecting paths, we combine features from the encoder (details, spatial information) and the decoder (high-level semantic information).
- Successful when training with few images.
- It's also used in image super-resolution and image generation.
    - Many generative models (Stable Diffusion, DALL-E 2,…) use this type of U-Net architecture under the hood.

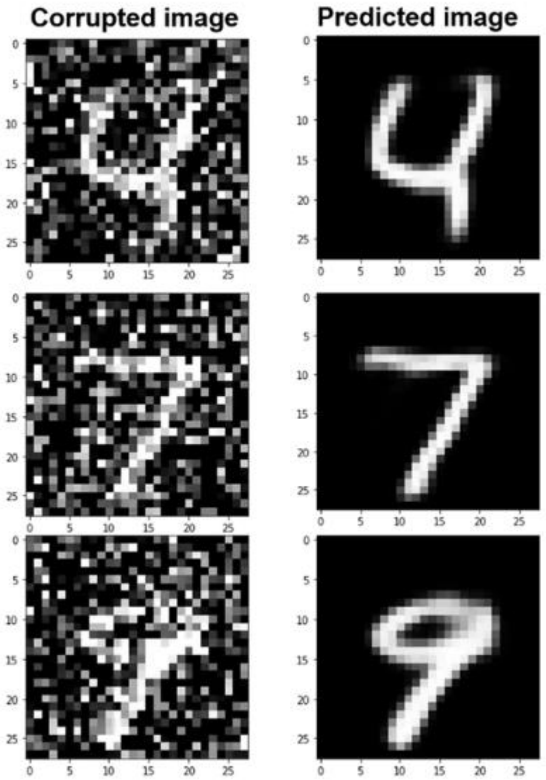
Ronneberger, Fischer, and Brox. "U-Net: Convolutional networks for biomedical image segmentation." *MICCAI 2015* (>75K citations)

Autoencoders

- These encoder-decoder architectures can be used for many different tasks, like image compression and denoising.



We want to learn an efficient encoding of input data.

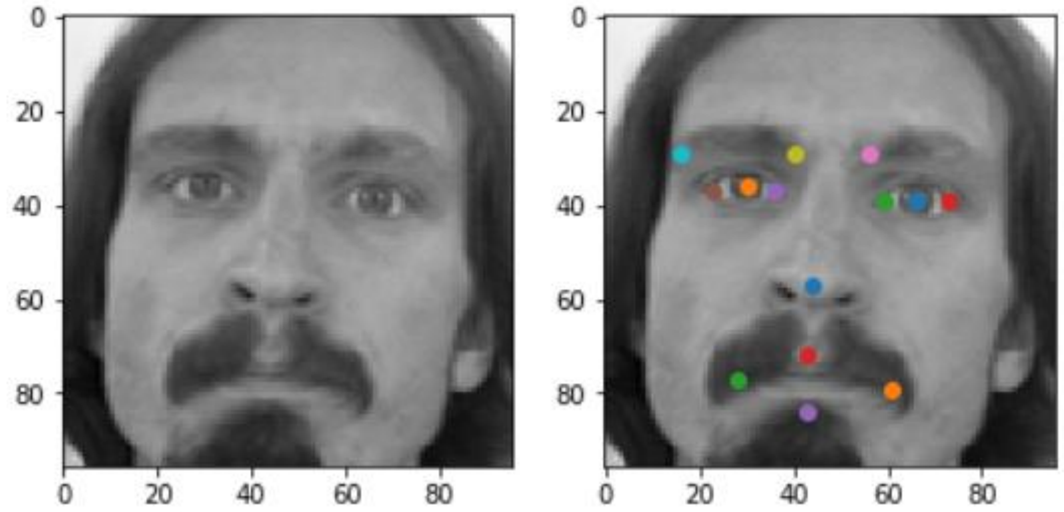


Denoising autoencoders:

- We corrupt input data on purpose, adding noise or masking some of the input values.
- The model is trained to predict the original, uncorrupted data.
- The autoenconder must undo this corruption.

Keypoint Regression

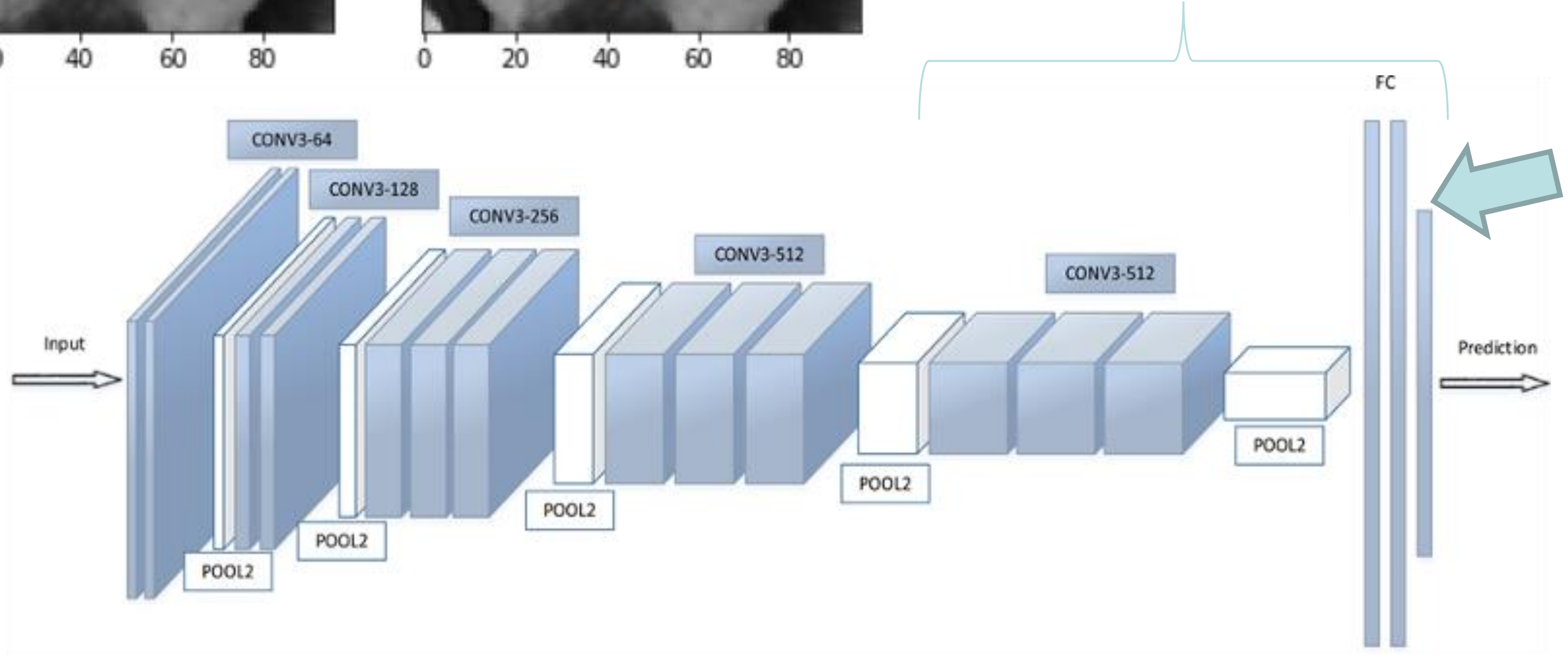
We want to locate keypoints in images (e.g., in faces)



Option 1:

Directly predict Cartesian coordinates (x,y) for each keypoint

Fine-tuning of last layers

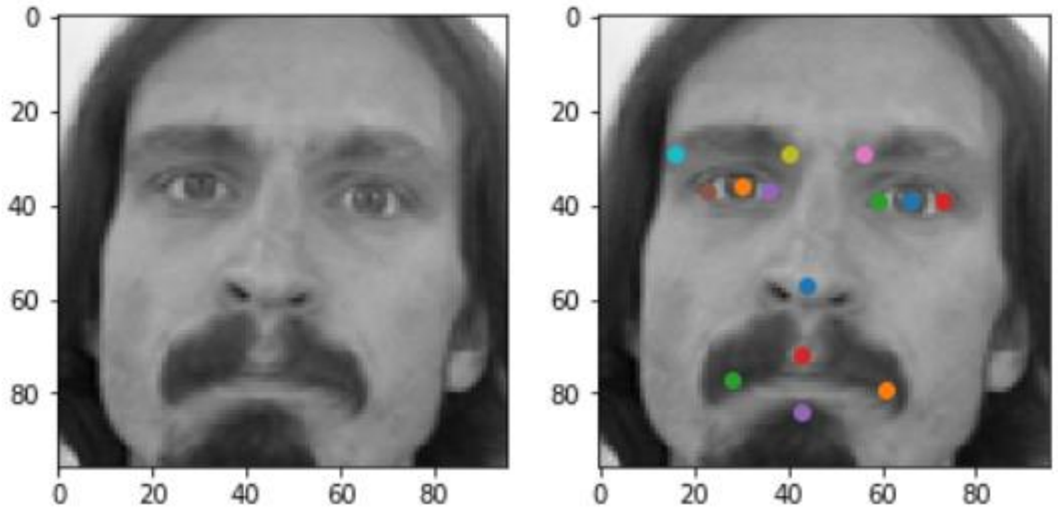


FC1000 is replaced by BN + a regression layer

We are forcing the network to map from the image to Cartesian coordinates. Maybe too complex. Likely to overfit.

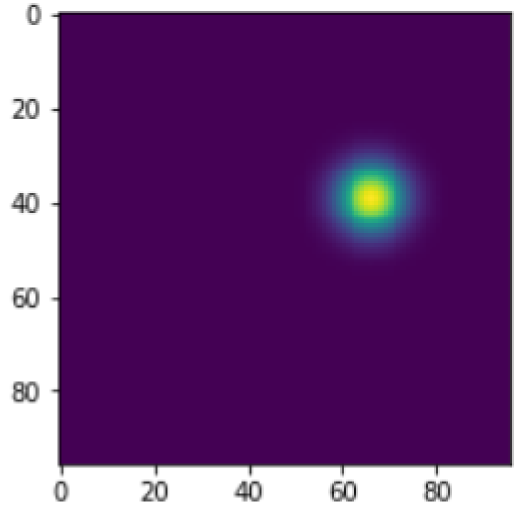
Keypoint Regression

We want to locate keypoints in images (e.g., in faces)



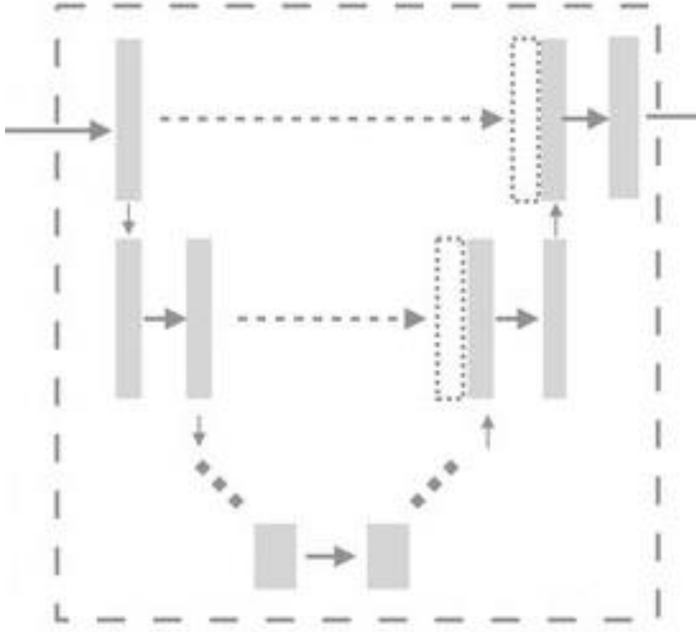
Option 2:

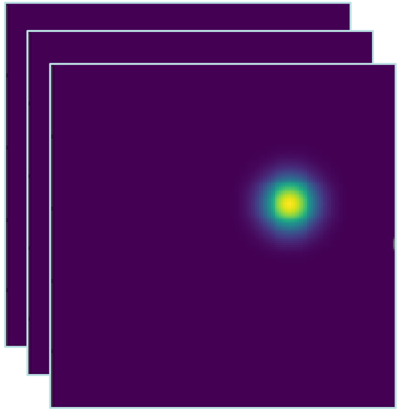
Perform heatmap regression for each keypoint.
Each keypoint is represented by a heatmap (smooth version of our Cartesian coordinates), that represents a wider range of "correct" answers (likelihood of a specific keypoint residing at that pixel).



Heatmaps generally work better than direct joint/keypoint regression.

Keypoint Regression





We train using a regression loss (MSE).

If we want to locate 3 keypoints in a HxW image:
- **output: 3xHxW tensor**, with **each channel representing one of our keypoints/classes**
- **each pixel would represent the likelihood** of that pixel being the keypoint we’re looking for
- to **retrieve the keypoints location**, we find the location of the highest activation in each output heatmap

DeepLabV3+

Xception backbone (depthwise separable convolutions)

+ atrous convolutions (larger receptive field, no increase in computational cost)

+ atrous spatial pyramid pooling (ASPP) (for multi-scale feature extraction)

+ encoder-decoder architecture (with BN, ReLU and 1x1,3x3 convolutions)

Check https://learnopencv.com/deeplabv3-ultimate-guide/ for details

Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. *ECCV 2018*

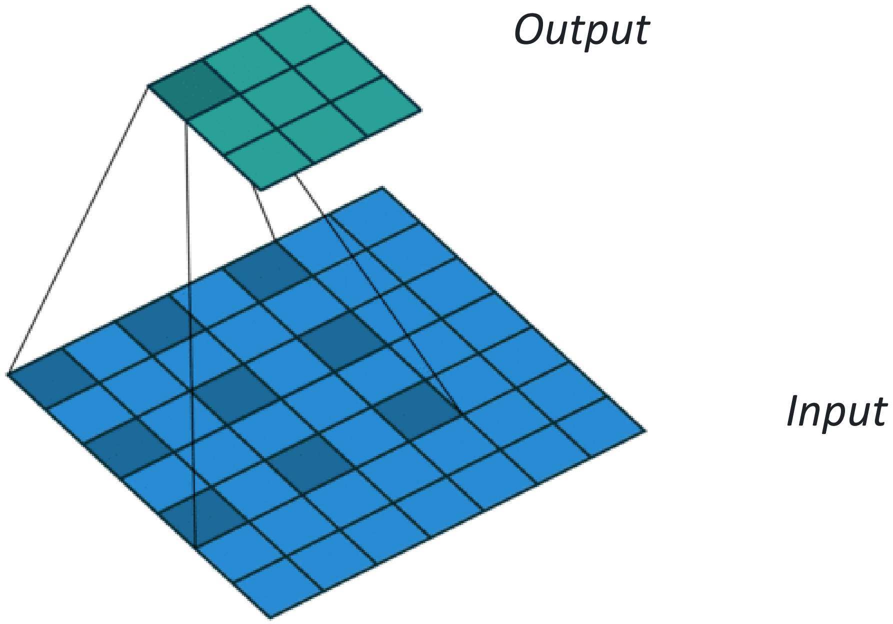
DeepLabV3+

Xception backbone (depthwise separable convolutions)

+ **atrous convolutions** (larger receptive field, no increase in computational cost)

+ atrous spatial pyramid pooling (ASPP) (for multi-scale feature extraction)

+ encoder-decoder architecture



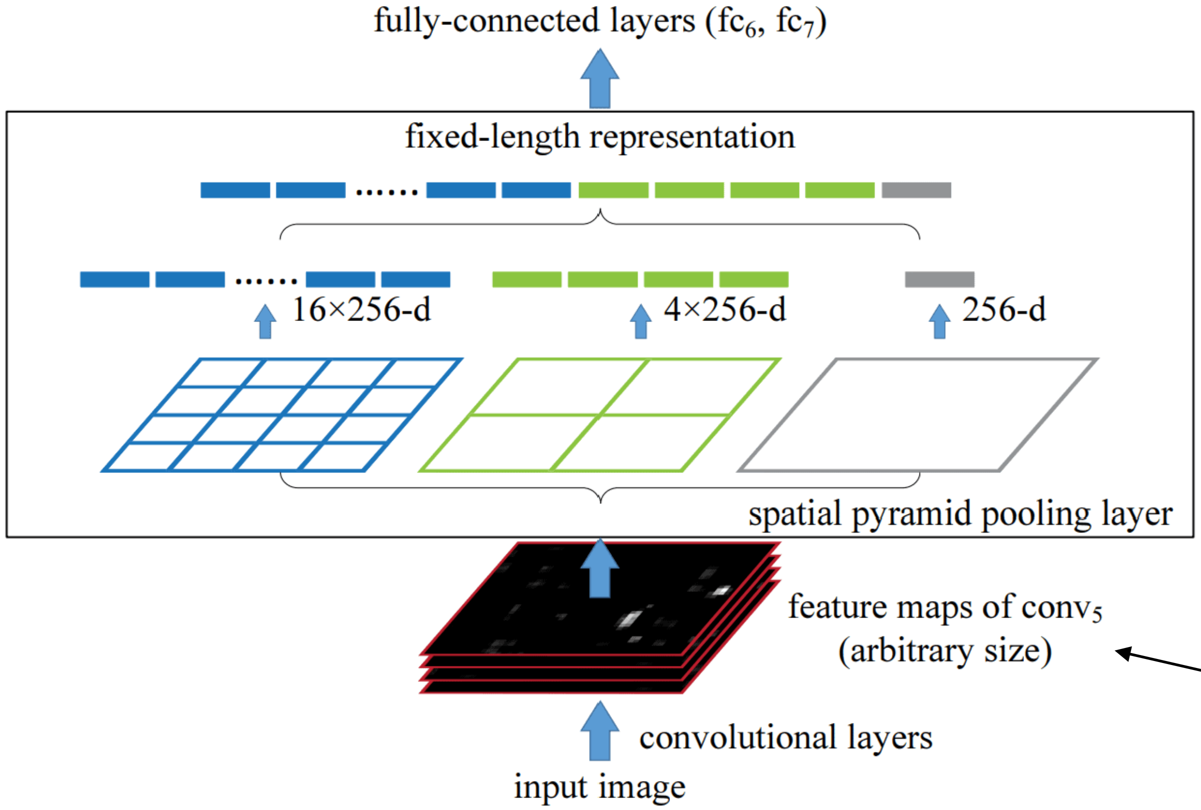
Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. *ECCV 2018*

DeepLabV3+

Xception backbone (depthwise separable convolutions)

+ atrous convolutions (larger receptive field, no increase in computational cost)

+ atrous **spatial pyramid pooling** (ASPP) (for multi-scale feature extraction)



Effective to resample features at different scales.

Provides a fixed-length representation, independent of the input image size.

ASPP = **SPP using atrous convolutions with different dilation rates**

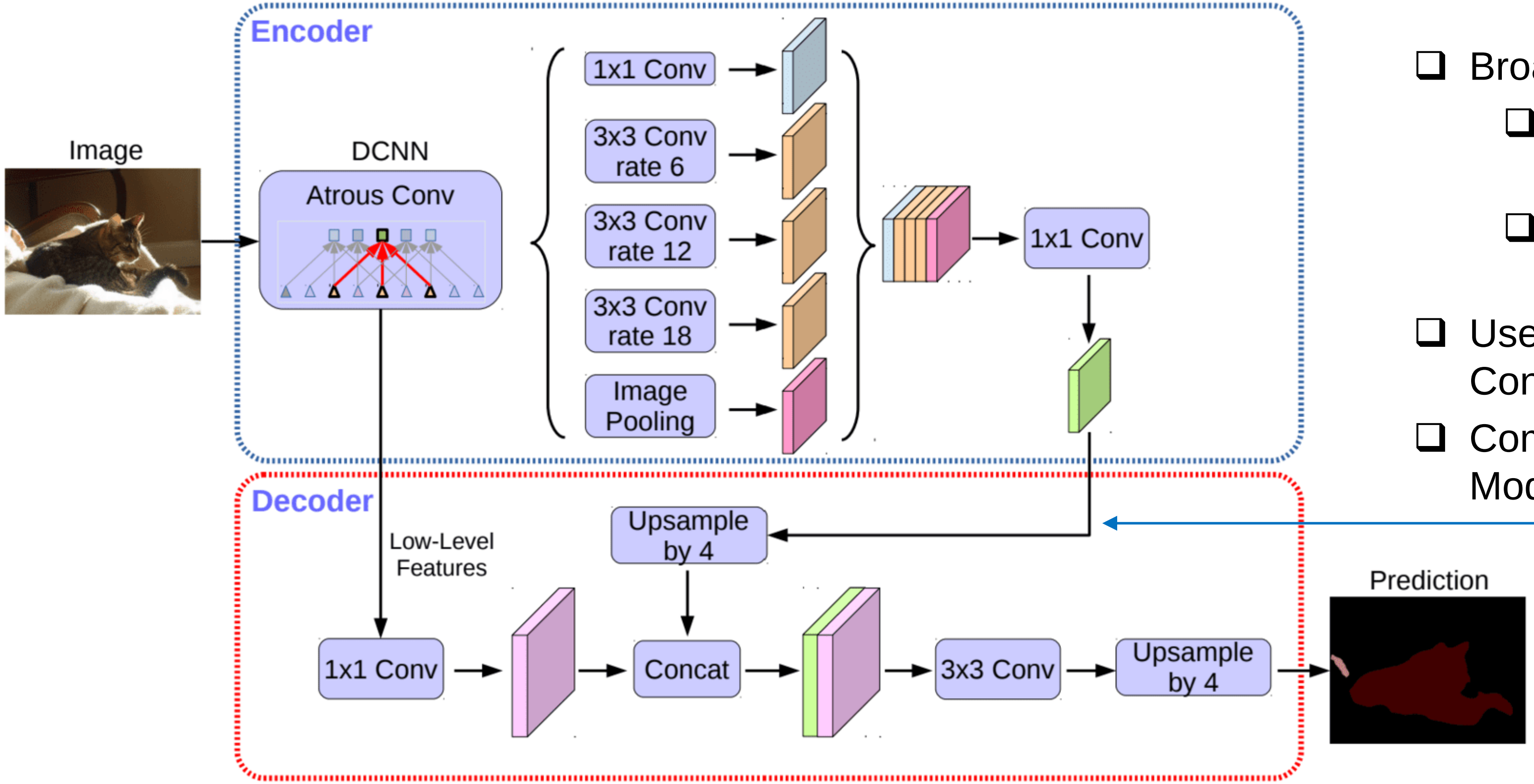
Different pooling sizes (from right to left): global, 4 bins, 16 bins

256 filters / activation maps

He et al. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE TPAMI*, *37*(9), 1904-1916.

DeepLabV3+

Full encoder-decoder architecture



- ❑ Broadly composed of two steps:
  - ❑ **Encoding phase:** to extract information from the image.
  - ❑ **Decoding phase:** to reconstruct output of appropriate dimensions.
- ❑ Use Atrous Convolution and Separable Convolutions to reduce computation.
- ❑ Combine Atrous Spatial Pyramid Pooling Modules and Encoder-Decoder Structure.

The encoder features are first bilinearly upsampled by a factor of 4 and then concatenated with the corresponding low-level features

Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. *ECCV 2018*

How do we evaluate segmentation?

- Accuracy: percentage of pixels correctly classified/segmented
  - Poor metric. E.g., if objects of interest only represent 10% of pixels in the image, and we systematically segment everything as background → acc=90%

- Dice Similarity Coefficient (DSC) or F1 score:

$$\mathrm{DSC} = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2TP}{2TP + FP + FN} \in [0,1]$$

Number of segmented pixels

Number of pixels belonging to the ground truth

TP: true positives | FP: false positives | FN: false negatives

  - You have one metric value per class. If you want a single overall value, you'd compute the mean of all these DSC values.

How do we evaluate segmentation?

- Dice Similarity Coefficient (DSC):

$$\mathrm{DSC} = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2TP}{2TP + FP + FN} \in [0,1]$$

Segmented pixels

Ground truth

TP: true positives | FP: false positives | FN: false negatives

- Jaccard Index (JI) or Intersection over Union (IoU): $\mathrm{JI} = \frac{|X \cap Y|}{|X \cup Y|} = \frac{DSC}{2-DSC}$

https://giou.stanford.edu/

$|X \cap Y|$

$|X \cup Y|$

How do we evaluate segmentation?

- Hausdorff Distance (HD):
  - largest of all distances from any point in the boundary of the resulting segmentation to the closest point in the ground truth.
  - we try to measure how close the contours are to each other. This is a good metric is boundary location accuracy is particularly relevant.

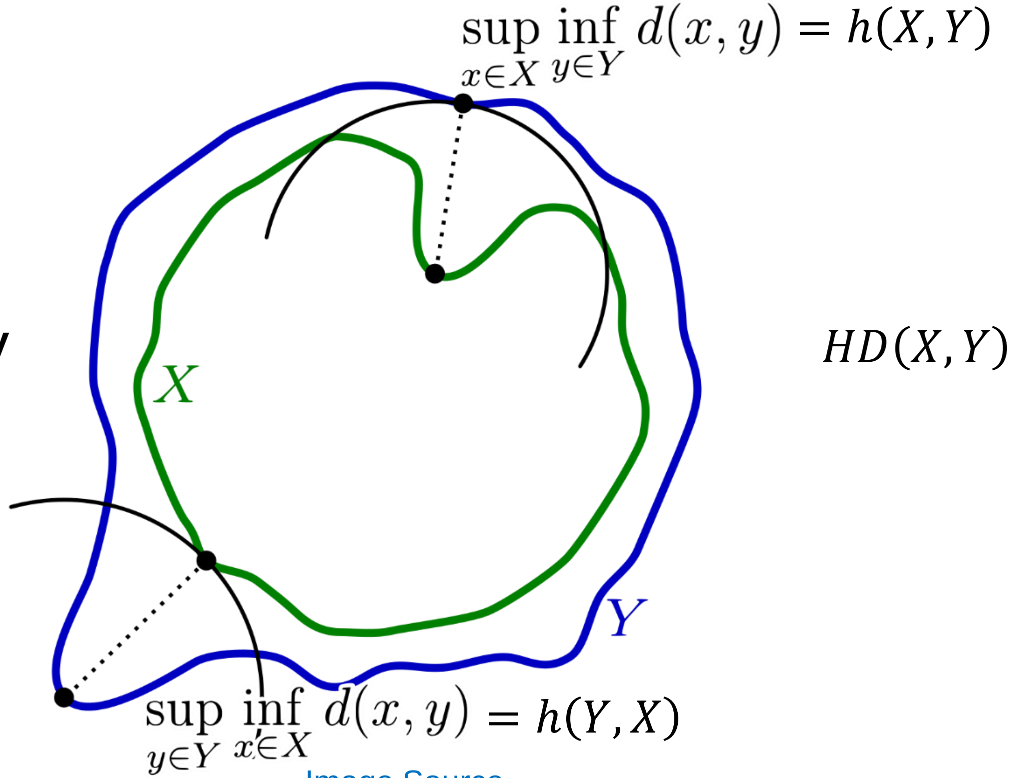


Image Source

$$HD(X, Y) = \max(h(X, Y), h(Y, X))$$

Image Segmentation

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DECSAI

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**