# Computer Vision and Machine Learning in Industry 4.0: Use case (inserts and CNNs)

Laura Fernández Robles
Universidad de León (Spain)
l.fernandez@unileon.es

Supported by
eit Manufacturing
Funded by the European Union
EIT HEI Initiative
eit-hei.eu

# Credits

- Book: Deep Learning with Python (François Chollet).
- https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html
- https://deeplearning4j.org/neuralnet-overview
- MIT course: Introduction to Deep Learning: http://introtodeeplearning.com/
- https://brilliant.org/wiki/convolutional-neural-network/
- Convolutional Neural Networks for Visual Recognition (Stanford University): http://cs231n.stanford.edu/

# Table of contents



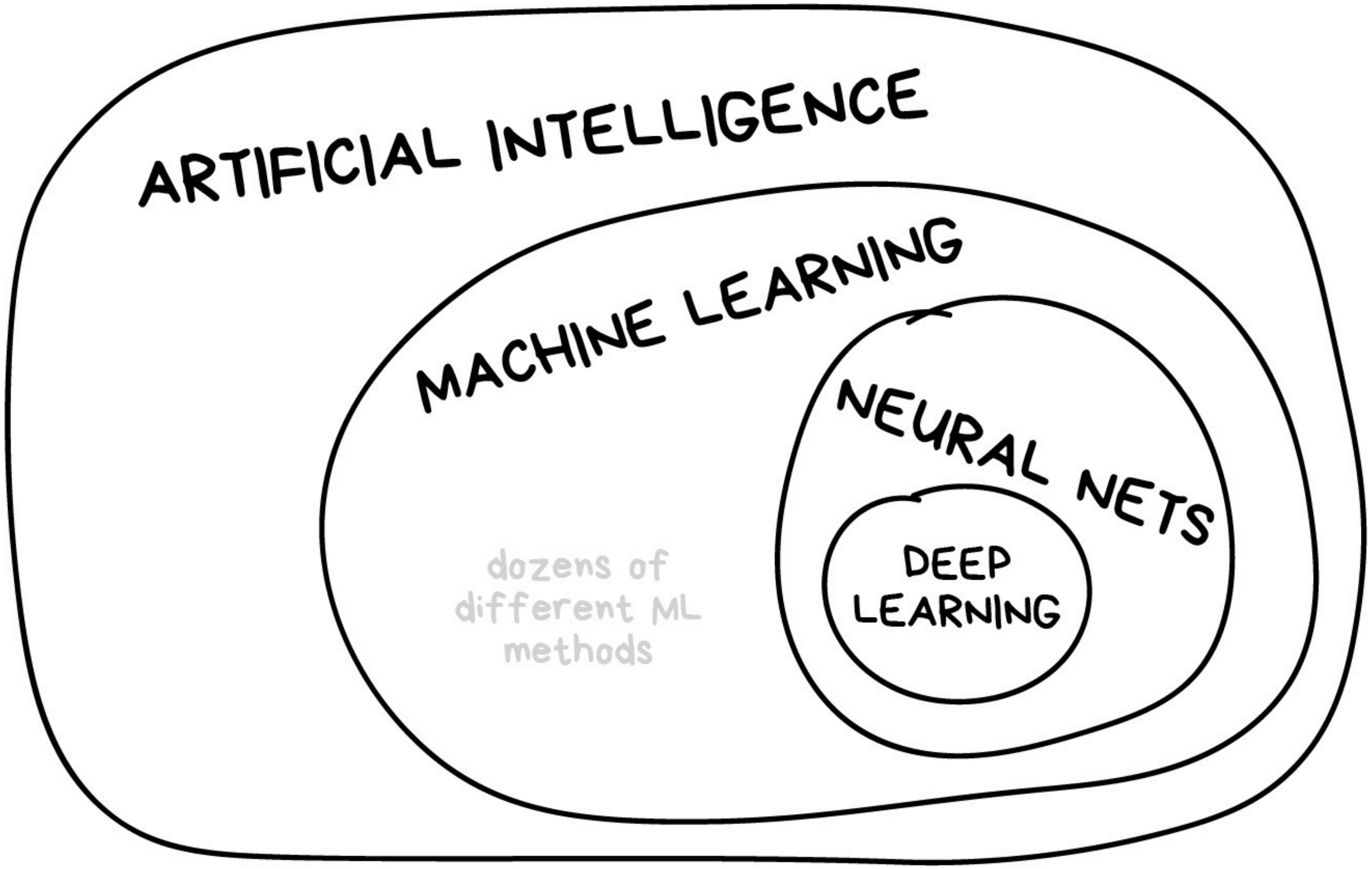


universidad de león

REGINNA 4.0

# Introduction

# Deep Learning





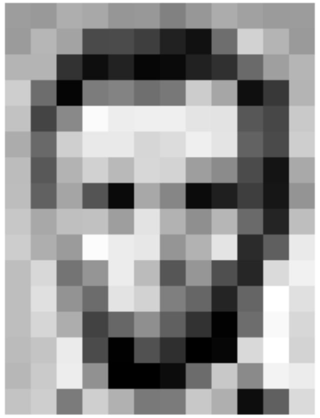

universidad de león

REGINNA 4.0

# The task in machine learning

**Example:**

Which US president is this?



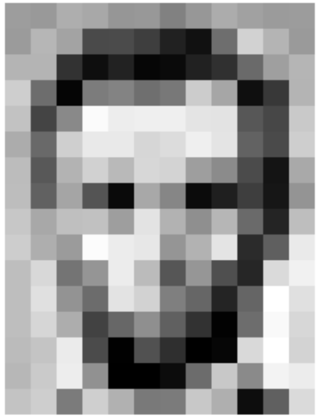
Input image

→ ???

# The task in machine learning

**Example:**

Which US president is this?



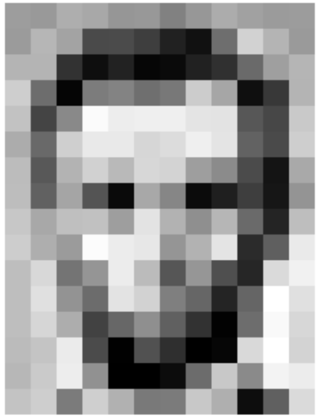
Input image

→ Abraham Lincoln

# The task in machine learning

**Example:**

Which US president is this?



Input image

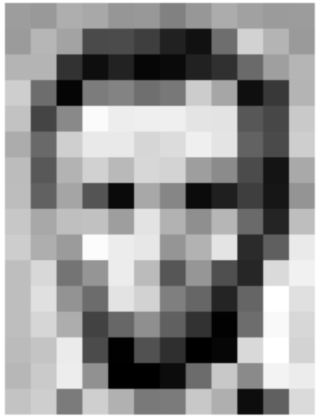
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

Pixel representation

| | |
|---|---|
| Lincoln | 0.8 |
| Washington | 0.1 |
| Jefferson | 0.05 |
| Obama | 0.05 |

Prediction

# The task in machine learning



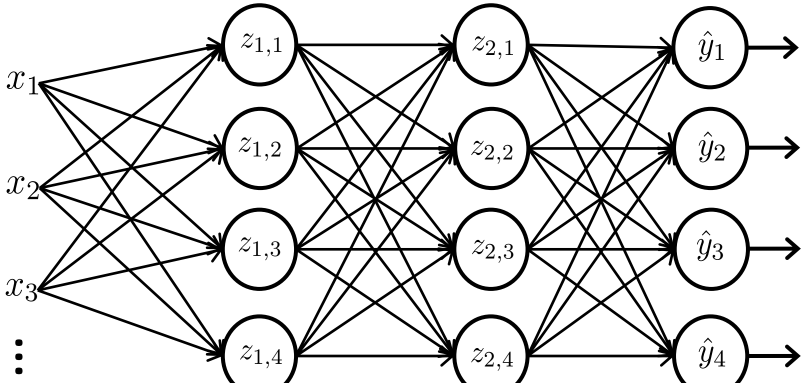
Input image

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

Pixel representation

$$\boxed{\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]}$$

Feature extraction

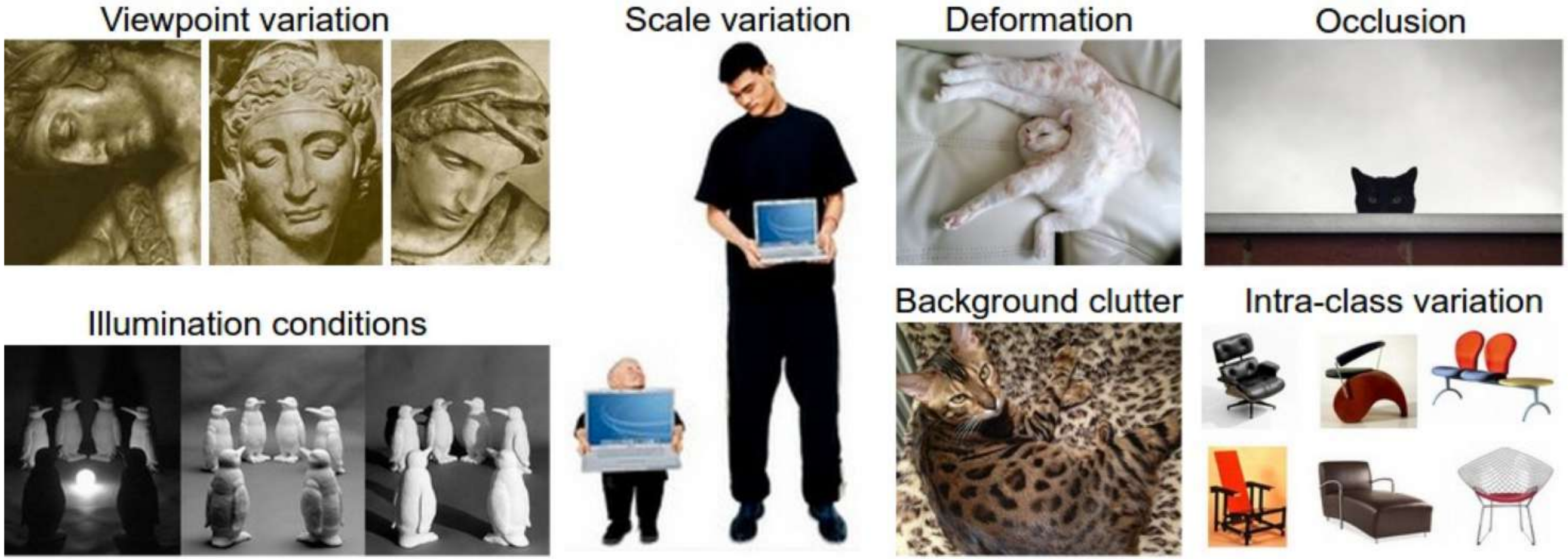



$$\begin{array}{l} \text{Lincoln} \\ \text{Washington} \\ \text{Jefferson} \\ \text{Obama} \end{array} \begin{bmatrix} 0.8 \\ 0.1 \\ 0.05 \\ 0.05 \end{bmatrix}$$


universidad de león

# Using machine learning: Manual feature extraction

## Problems?

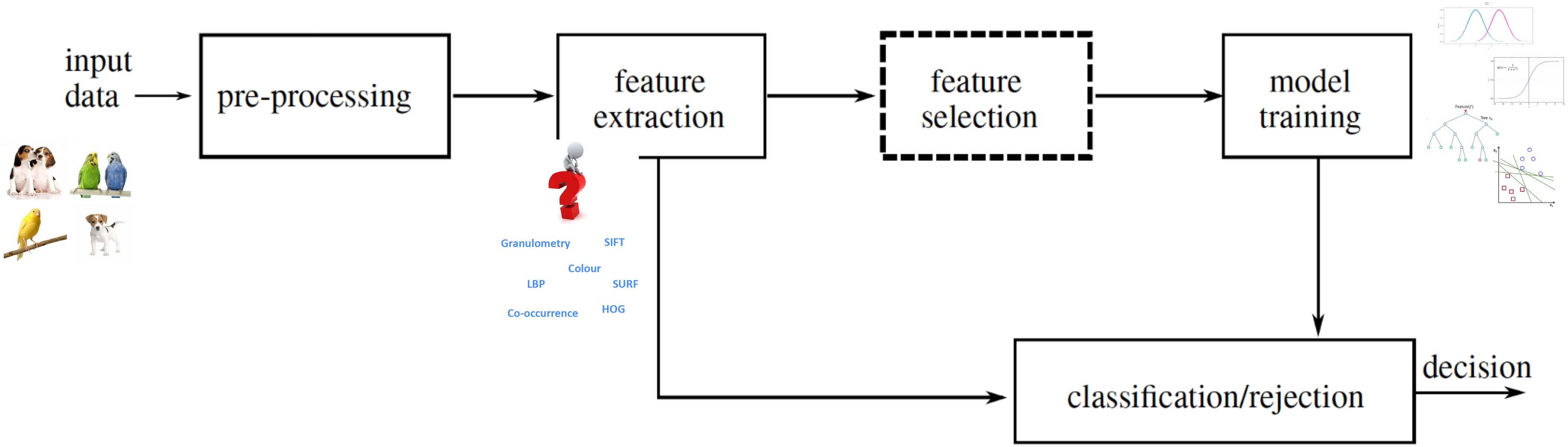




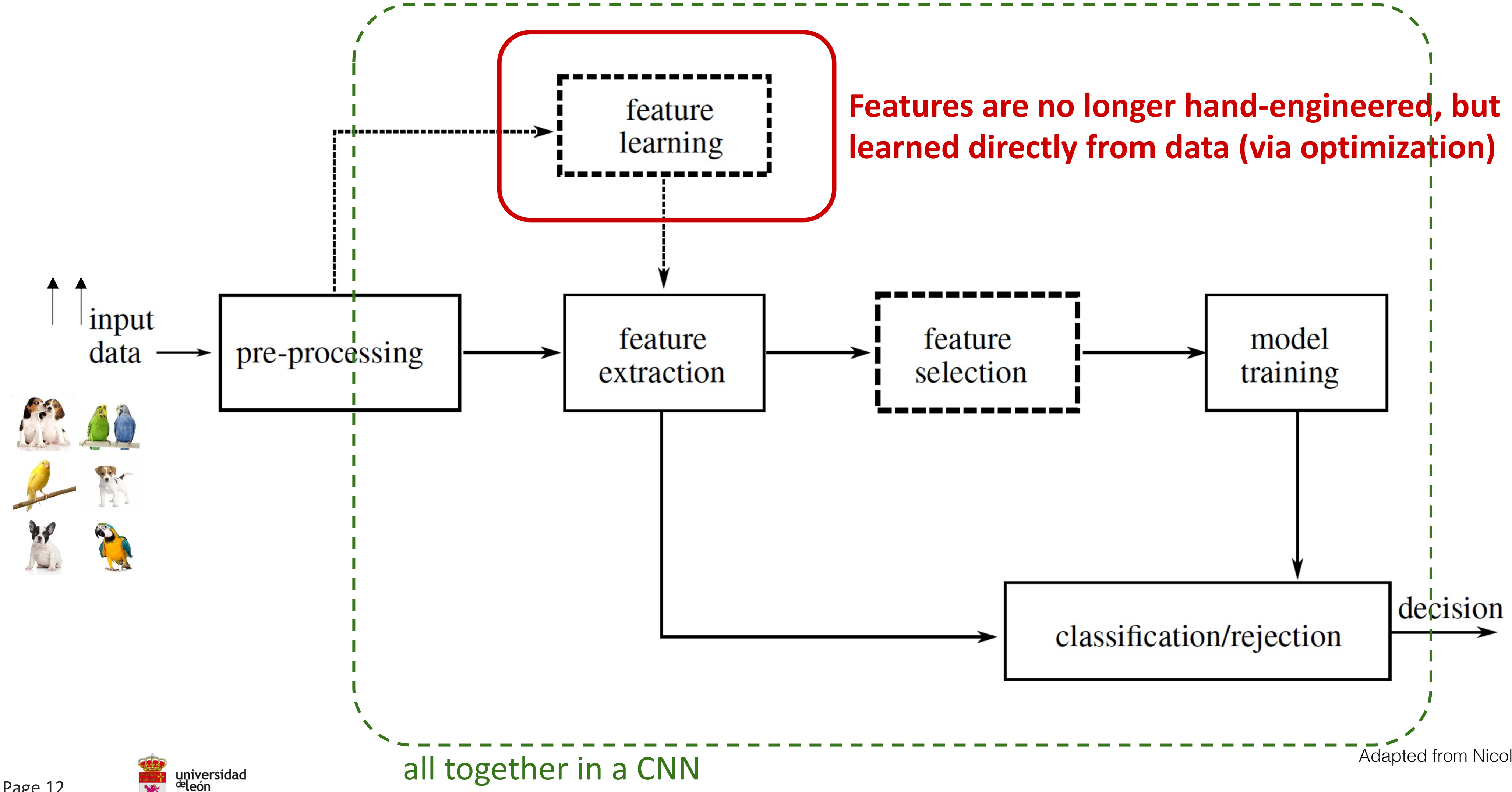
universidad de león

# Representation learning: where CNNs are great!

A classical supervised pattern recognition pipeline

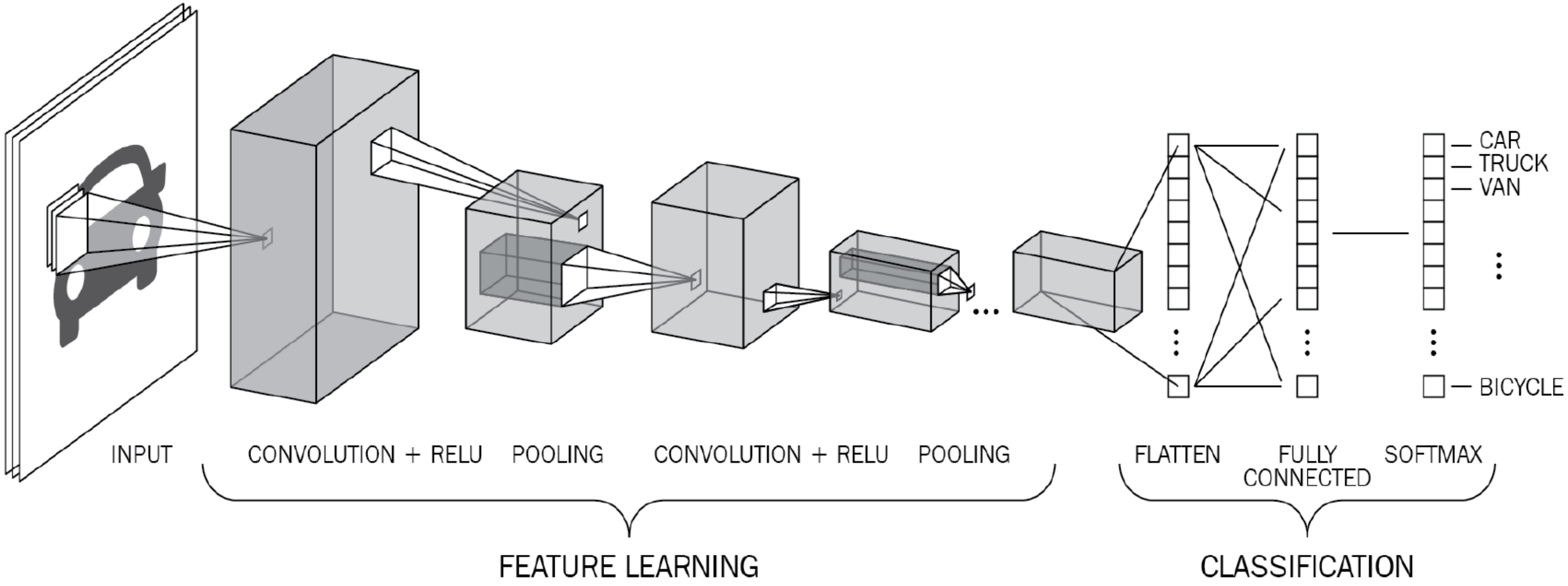


# Representation learning: where CNNs are great!




Adapted from Nicola Strisciuglio

# Representation learning: where CNNs are great!



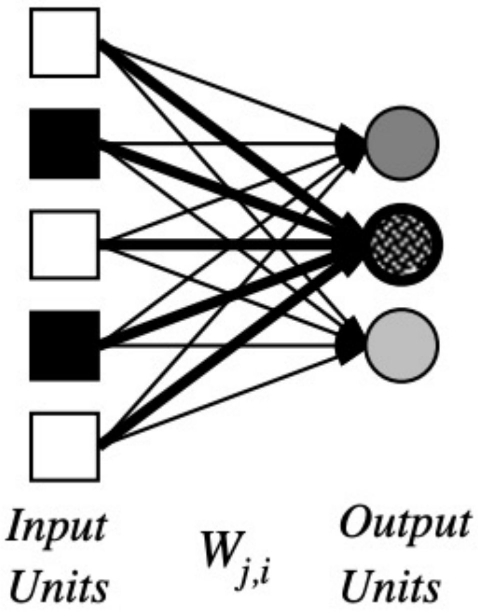

Modern Computer Vision with PyTorch

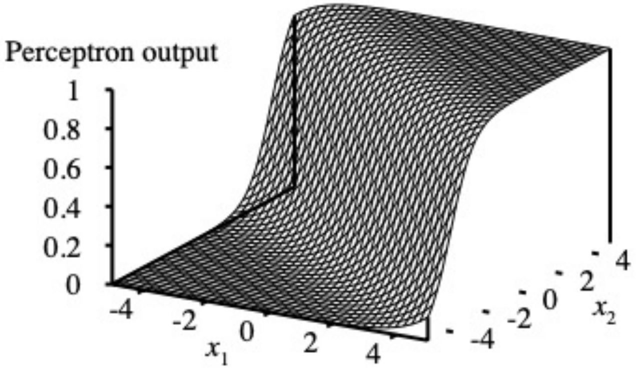
Adapted from Nicola Strisciuglio

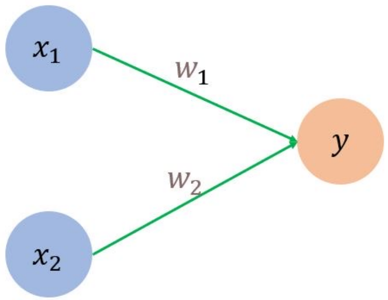
# Neural Networks

# Neural Networks (NN): single layer perceptrons

Weighted linear combination of feature values and weights can be illustrated as a network




Perceptron

bias    i th input

$$y = b + \sum_i x_i w_i$$

output    index over input connections    weight on i th input

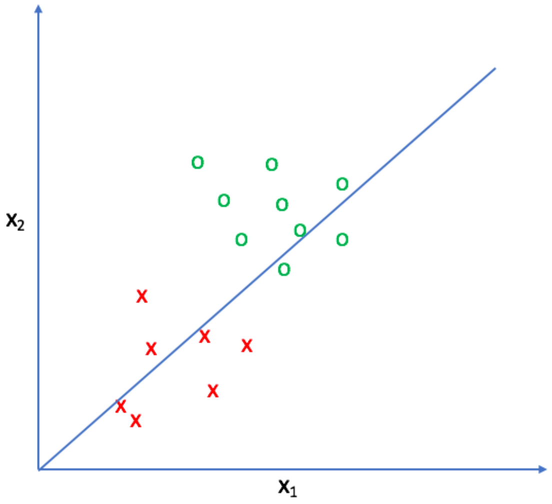
Adjusting weights moves the location, orientation, and steepness of cliff

Adapted from Stuart Russell

# Neural Networks (NN): single layer perceptrons

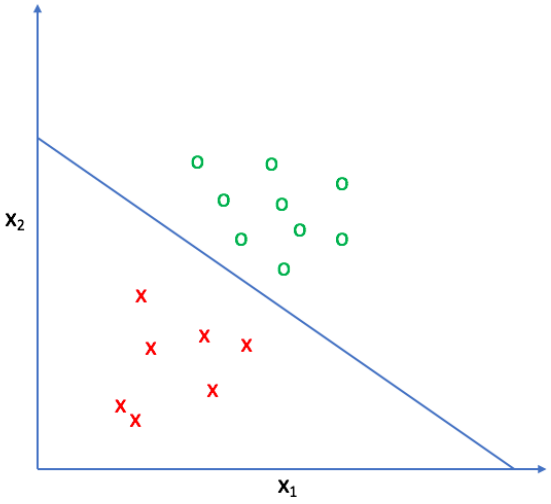
Bias

No bias (intercept) term




Our line is forced to pass through the origin.

With bias (intercept) term




Adding the intercept term allows for much better fit.

bias    ith input

$$y = b + \sum_i x_i w_i$$

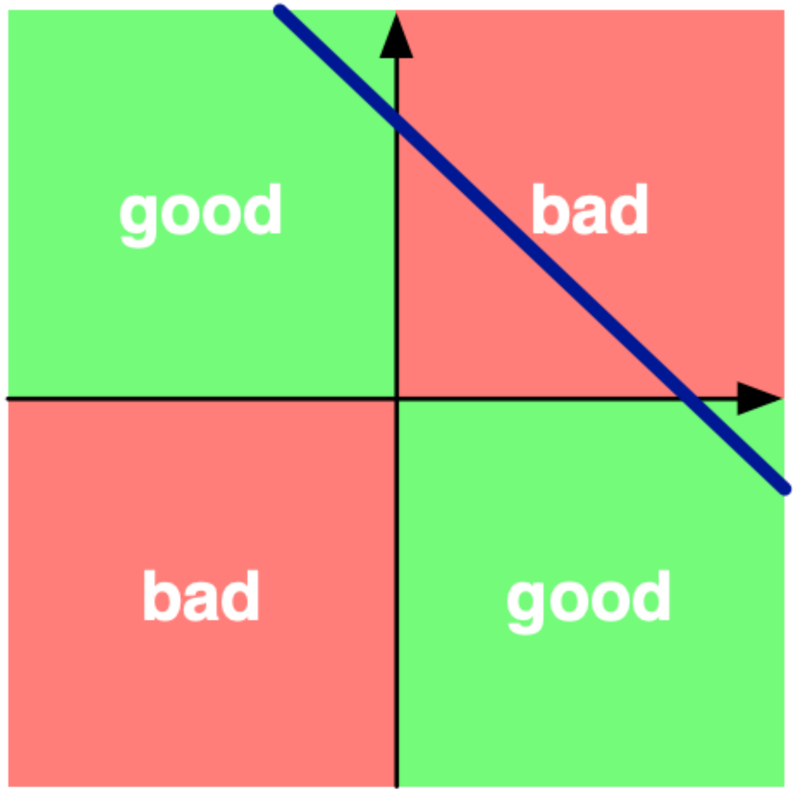
output    index over input connections    weight on ith input

https://www.jeremyjordan.me/intro-to-neural-networks/


universidad de león

# Neural Networks (NN): single layer perceptrons

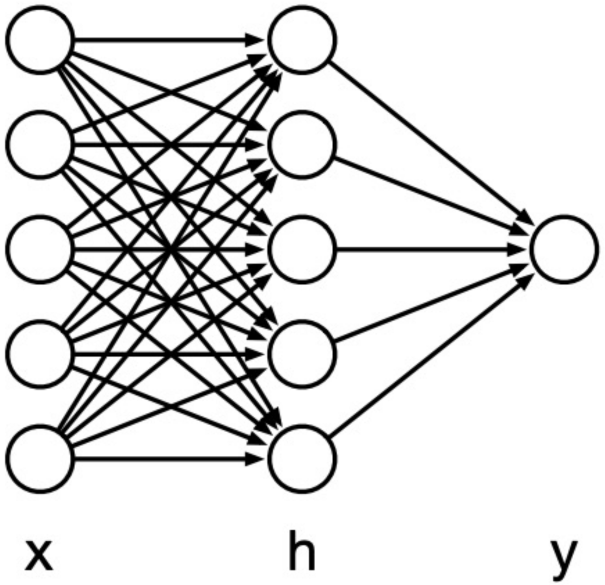
Linear models cannot model XOR




Adapted from Philipp Koehn

# Neural Networks (NN): multiple layers

Add an intermediate (“hidden”) layer of processing (each arrow is a weight)




Have we gained anything so far? The result of combining linear transformations is also a linear transformation

Adapted from Philipp Koehn

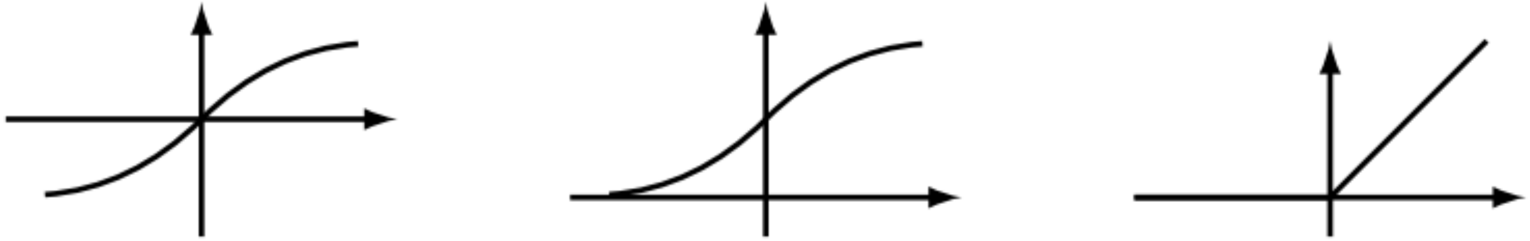

universidad de león

# Neural Networks (NN): Non-Linearity

Instead of computing a linear combination, add a non-linear function.

Popular choices of activation functions:

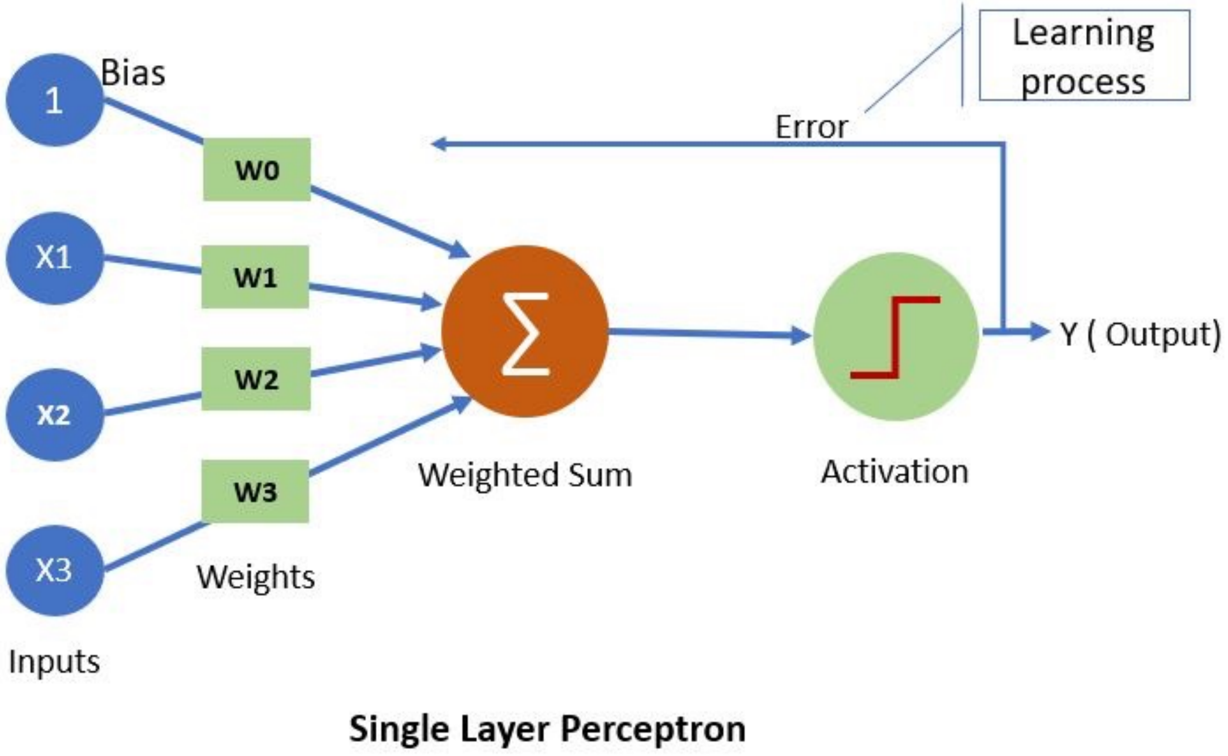
$\tanh(x)$ $\quad$ $\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}}$ $\quad$ $\mathrm{relu}(x) = \max(0,x)$



sigmoid is also called the “logistic function”

Adapted from Philipp Koehn

# Neural Networks (NN): single layer perceptrons



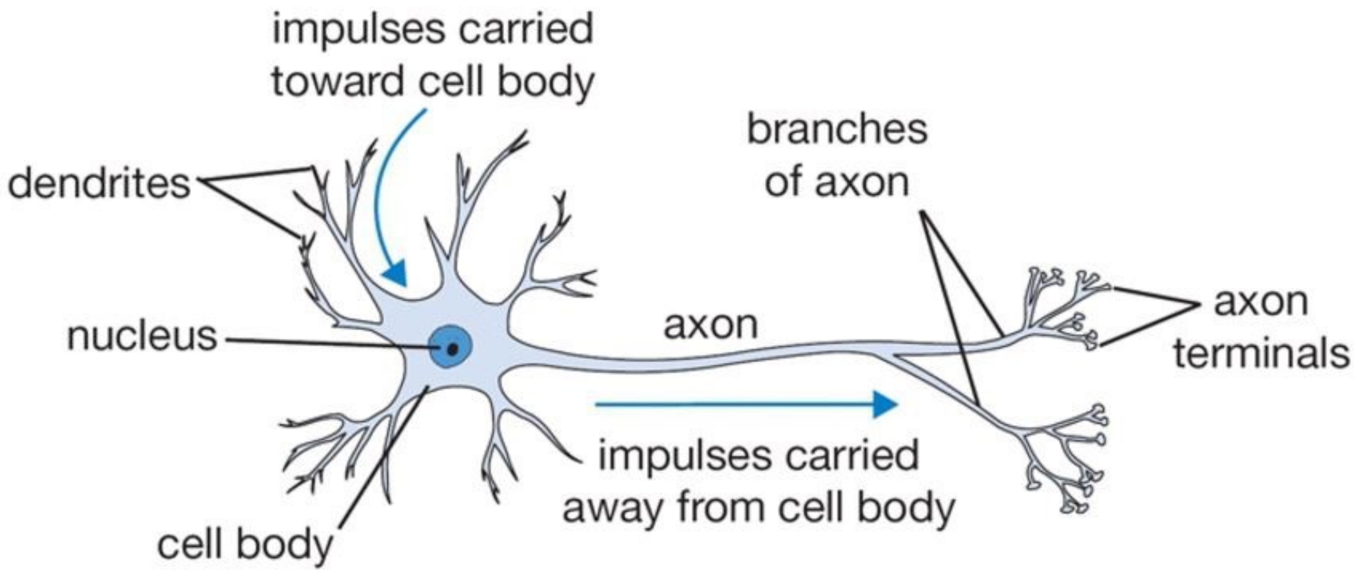

https://sumanthrb.com/ml/perceptron/

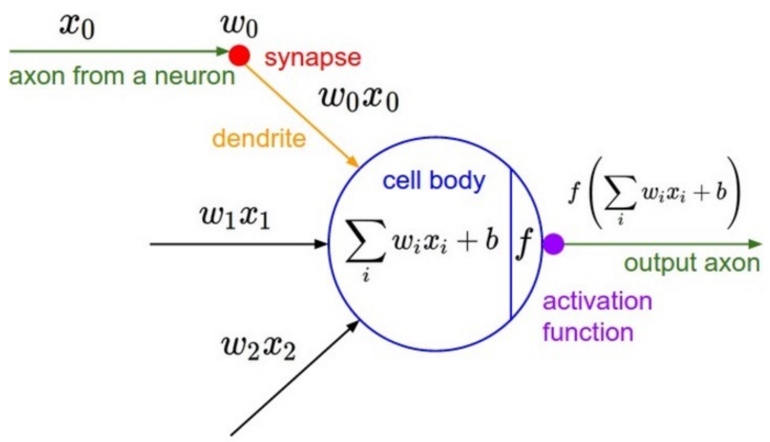


universidad de león

REGINNA 4.0

# Neural Networks (NN): Why “neural“ networks?




Taken from Andrea Palazzi

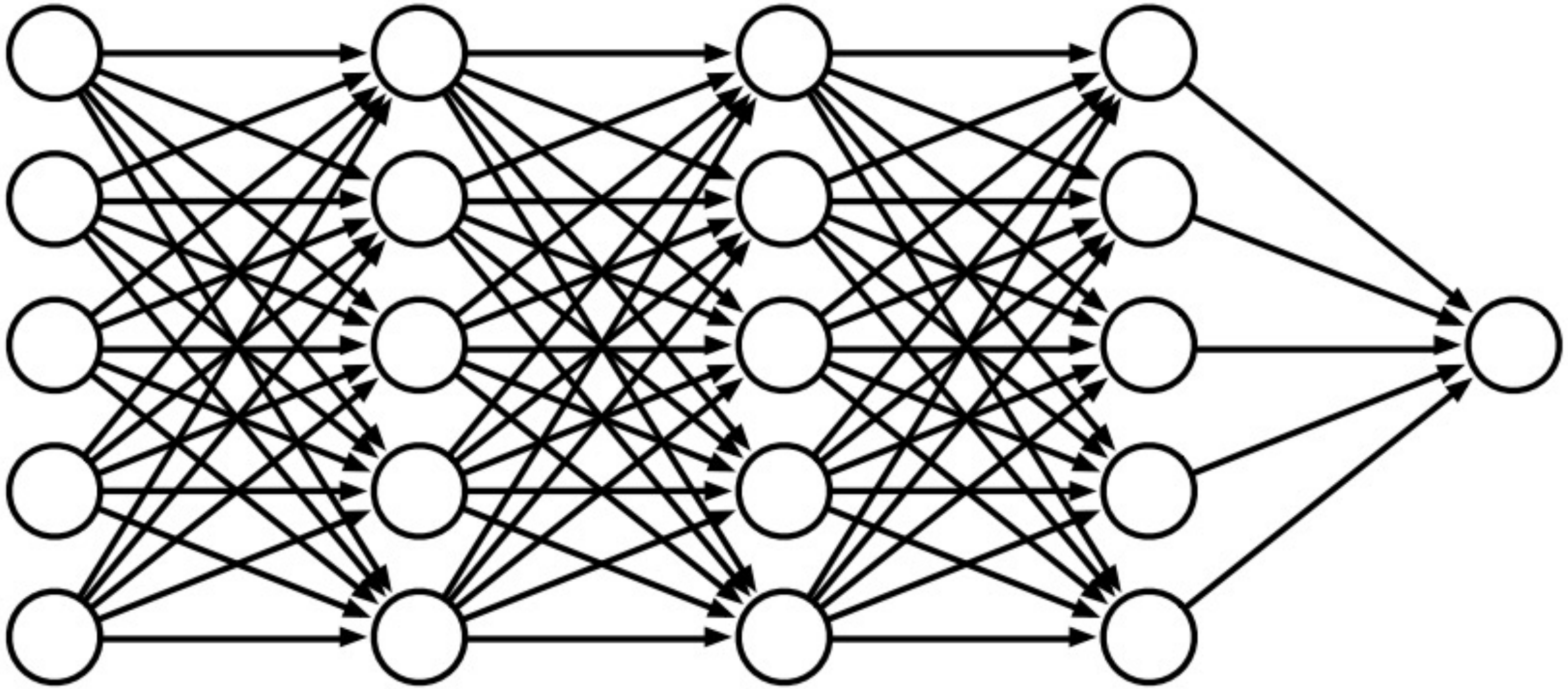


universidad de león

# Deep Neural Networks (DNN)

More layers = deep learning

Having multiple processing steps allows complex functions



Adapted from Philipp Koehn



universidad de león

REGINNA 4.0

# Convolutional Neural Networks

# Deep learning on large images

High number of input parameters -> high number of training parameters:

- Requires lot of data to avoid overfitting
- Requires high memory to train the parameters

$n_H$ x $n_W$ x 3 = 1000 x 2000 x 3

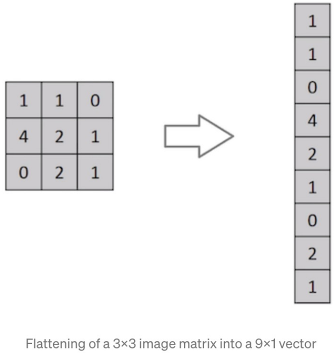
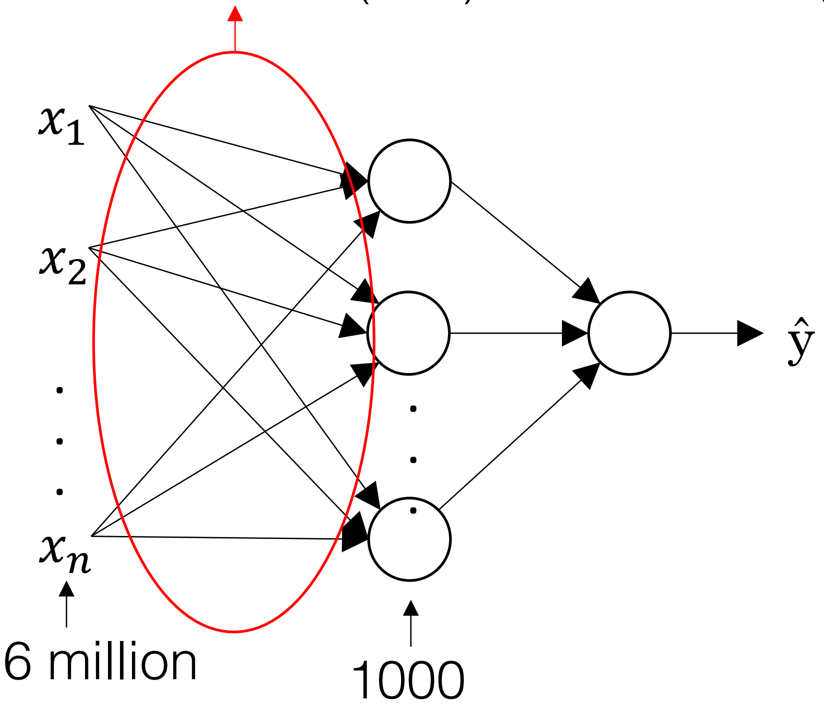



Image from Sumit Saha

1000 x 6 m + 1000 (bias) = >6 billion weights






universidad de león

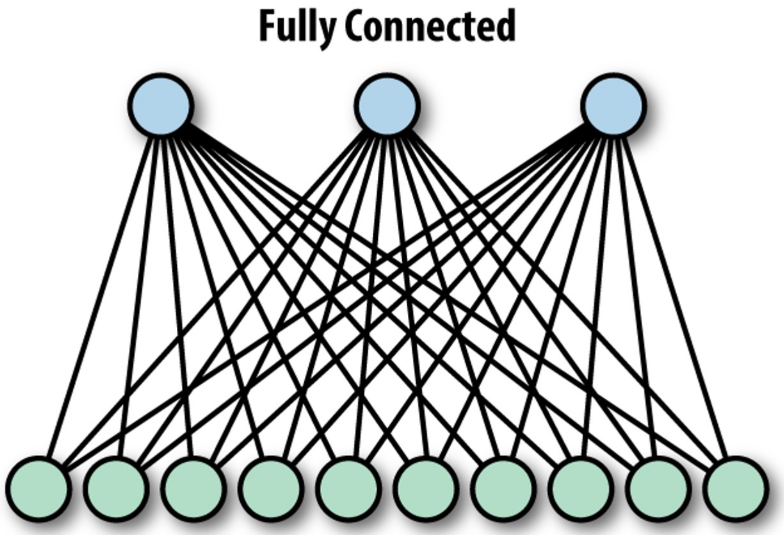
# Fully Connected vs Convolutional Networks

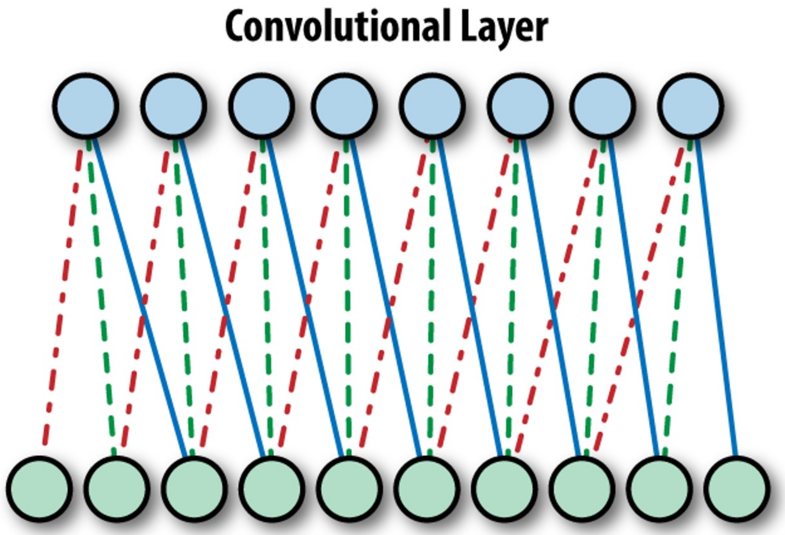
Subsequent units receive input from ALL units in the previous layer

10 inputs, 3 outputs = 10 x 3 + 3 (bias) = 34 weights

**Exploit locality of patterns**

- Each unit receives inputs from only few units (3 in this case) in the previous layer
- The pattern of weights slides on (convolves) the input




(Tom Hope, Yehezkel S. Resheff, Itay Lieder)

Adapted from Nicola Strisciuglio

# Convolution operation (cross-correlation, used in deep learning)

Convolution $y[m,n] = x[m,n] * h[m,n] = \sum_k \sum_l x[k,l]h[m-k,n-l]$

Cross-correlation $y[m,n] = x[m,n] * h[m,n] = \sum_k \sum_l x[k,l]h[m+k,n+l]$

Cross-correlation is always implemented, even if it is called convolution

Input image

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 74 | 66 | 62 | 59 | 63 | 71 | 75 | 72 |
| 72 | 65 | 61 | 58 | 62 | 70 | 75 | 72 |
| 71 | 68 | 63 | 59 | 63 | 69 | 73 | 69 |
| 70 | 67 | 62 | 59 | 62 | 69 | 73 | 69 |
| 70 | 64 | 59 | 57 | 61 | 70 | 74 | 72 |
| 68 | 66 | 64 | 63 | 65 | 68 | 69 | 68 |
| 70 | 69 | 67 | 66 | 67 | 69 | 69 | 67 |
| 69 | 68 | 66 | 64 | 65 | 67 | 66 | 63 |

Kernel or filter

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Output image

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | 43 | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

$$(-1) * 59 + 0 * 63 + 1 * 69 + (-2) * 59 + 0 * 62 + 2 * 69 + (-1) * 57 + 0 * 61 + 1 * 70 = 43$$

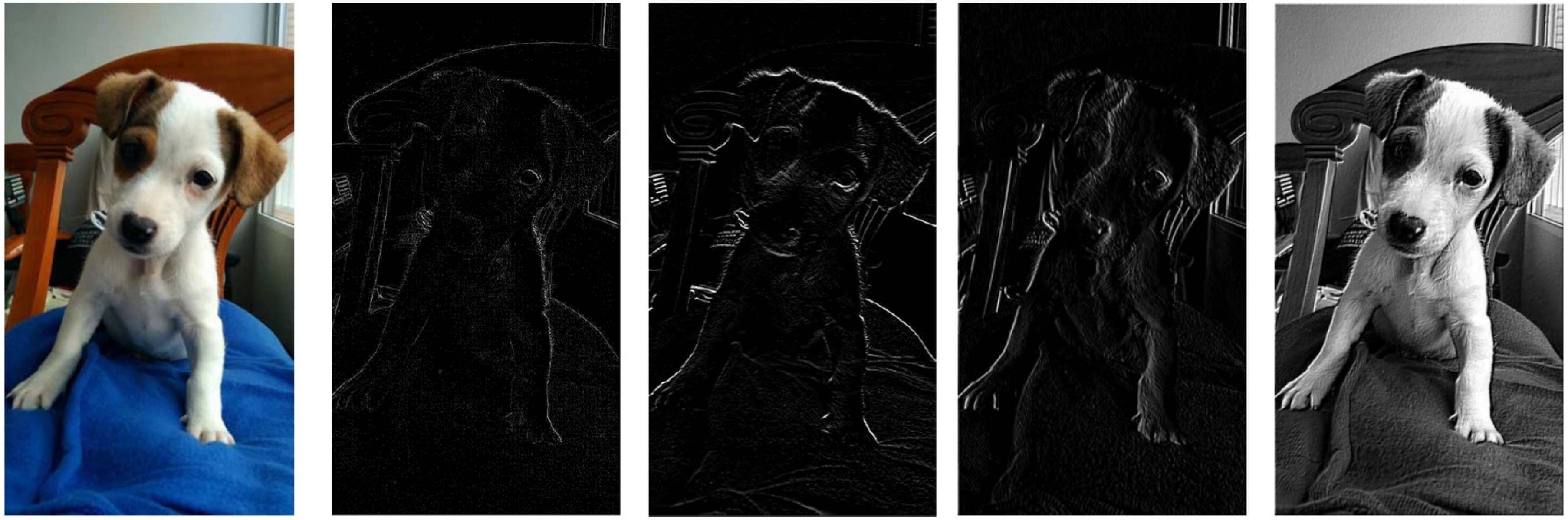
Adapted from CVBLAB



universidad de león

REGINNA 4.0

# Convolution operation



**Laplacian**

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

**Sobel H**

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

**Sobel V**

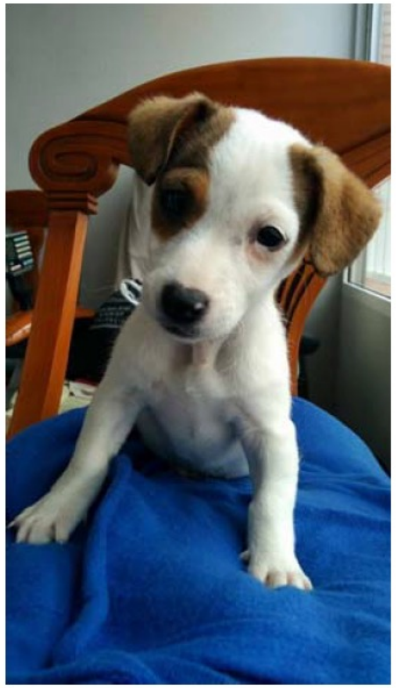
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

**Emboss**

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Adapted from CVBLAB

# Convolution operation



Blurr

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Severe blurr

Enhancement

$$\begin{bmatrix}-1 & -1 & -1\\ -1 & 9 & -1\\ -1 & -1 & -1\end{bmatrix}$$

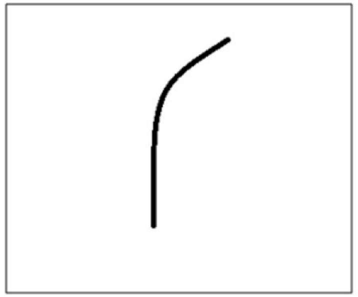
Adapted from CVBLAB

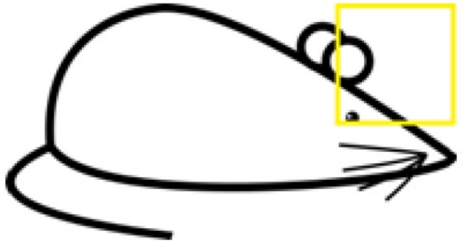


universidad de león

# Convolution operation: how low level features are detected

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



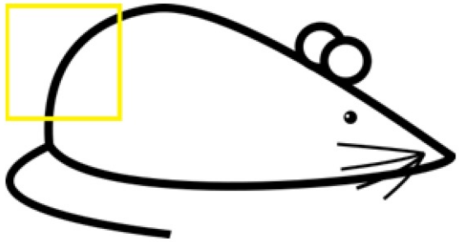


| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 40 | 0 | 0 | 0 | 0 | 0 |
| 40 | 0 | 40 | 0 | 0 | 0 | 0 |
| 40 | 20 | 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 50 | 0 | 0 | 0 | 0 |
| 25 | 25 | 0 | 50 | 0 | 0 | 0 |

*

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

No response in this receptive field



| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 30 |
| 0 | 0 | 0 | 0 | 50 | 50 | 50 |
| 0 | 0 | 0 | 20 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |

*

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

High response in this receptive field

Adapted from CVBLAB

# Convolution operation

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

Python commands for convolution operation:

- Python: `conv_forward`
- TensorFlow: `tf.nn.conv2d`
- Keras: `Conv2D`
- PyTorch: `torch.nn.functional.conv2d`

The kernel weights are learned!

Exploit locality of patterns (learn local relationships of pixels)

Sparcity of connections

Shared weights for all the image: translation invariance

Less parameters:

- Less prone to overfitting
- Less memory requirements



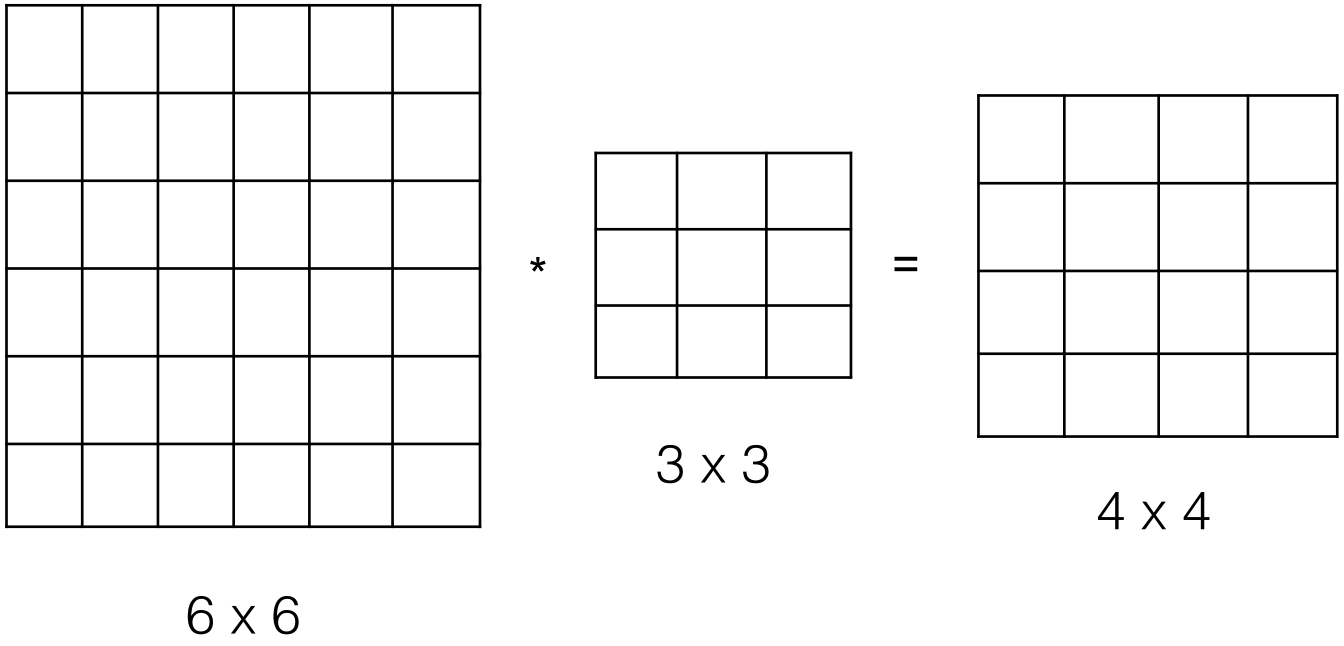
universidad de león

REGINNA 4.0

# Convolution operation

What are the dimensions of the activiation map if a 6x6 image is convolved with a 3x3 filter?

**n – f + 1** = 6 – 3 + 1 = 4



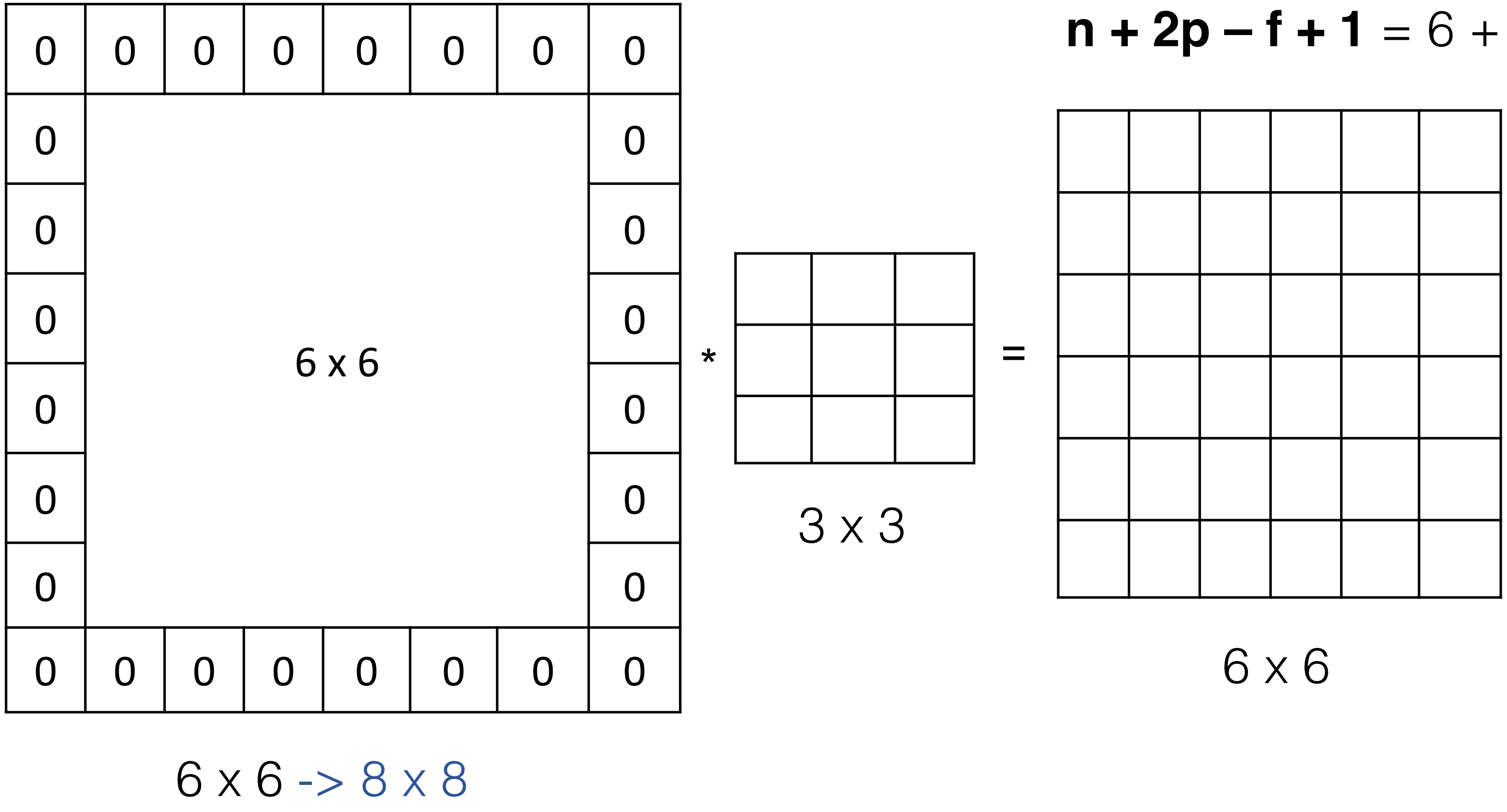

Issues:

- Shrinking output
- Throw away information from the edges

# Padding (Zero padding)

Hyperparameter: p (in this case p=1)

$\mathbf{n + 2p - f + 1} = 6 + 2 \cdot 1 - 3 + 1 = 6$




5x5x1 image is padded with 0s to create a 6x6x1 image (Sumit Saha, 2018)

# Paddin: Valid convolutions and same convolutions

Valid convolution: no padding

Same convolution: Pad so that output size is the same as the input size

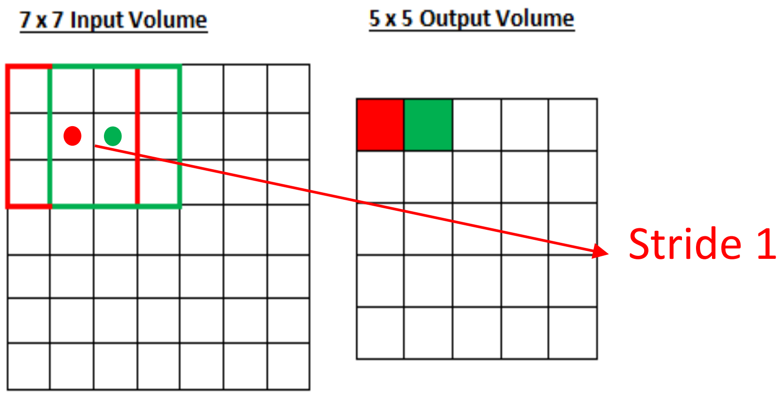
$$n + 2p - f + 1 = n$$

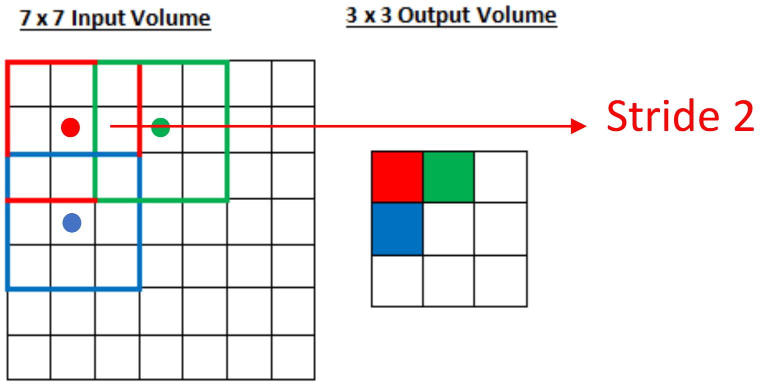
$$p = \frac{f - 1}{2}$$

Filter size **f** is *usually* odd:

- Natural padding region
- Central position

# Stride




Hyperparameter: s

Output size:

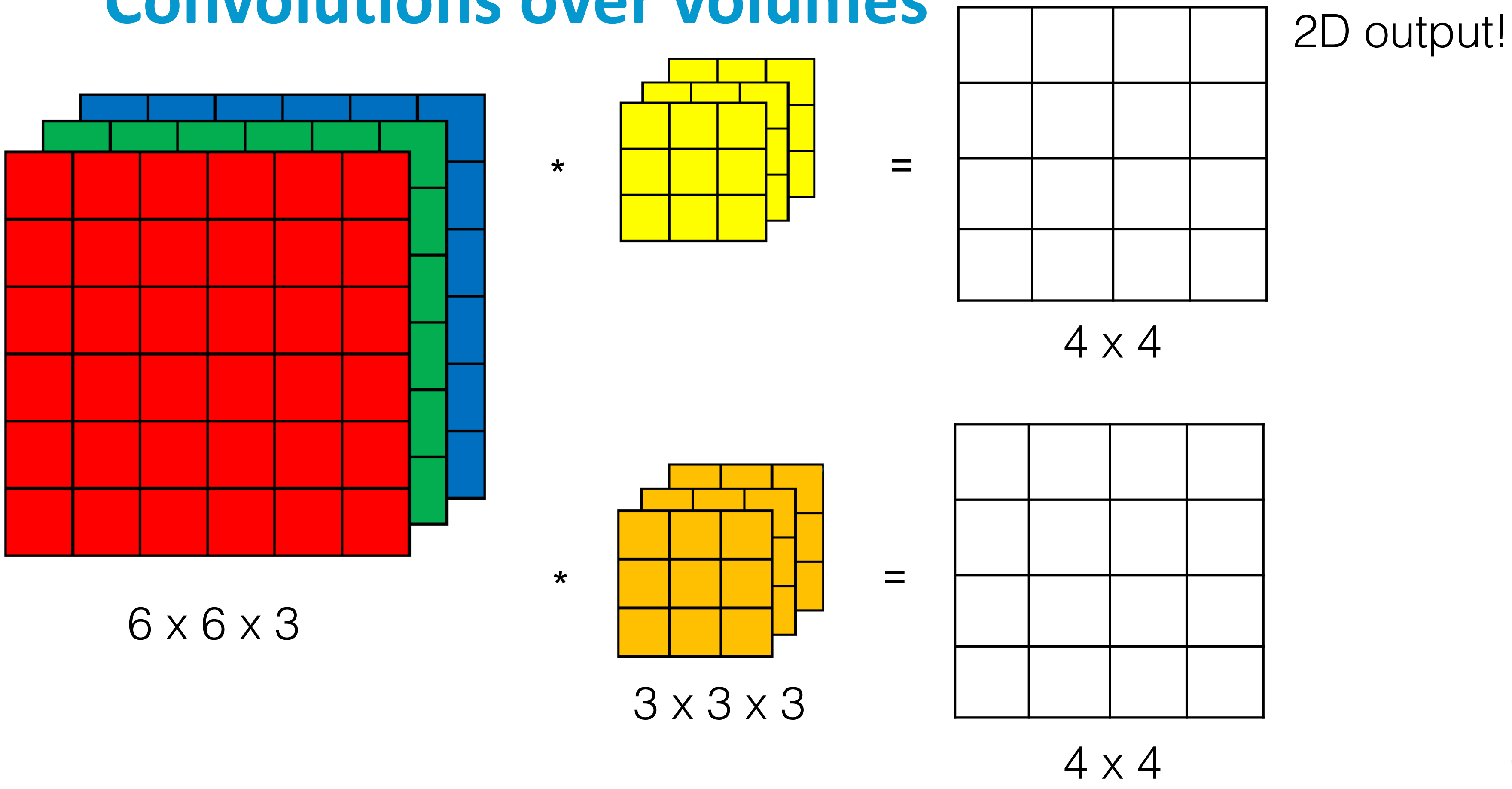
$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

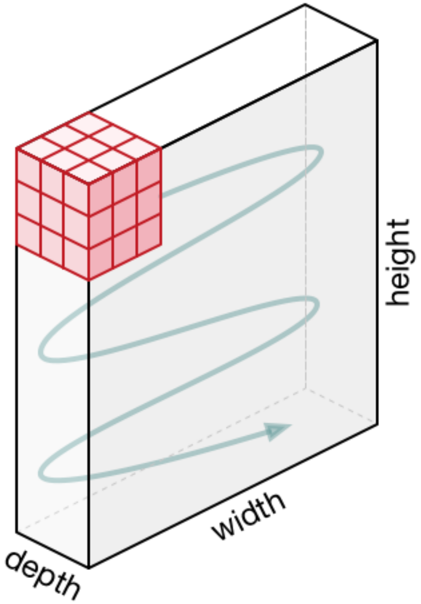
Convolution Operation with Stride Length = 2 (Sumit Saha, 2018)



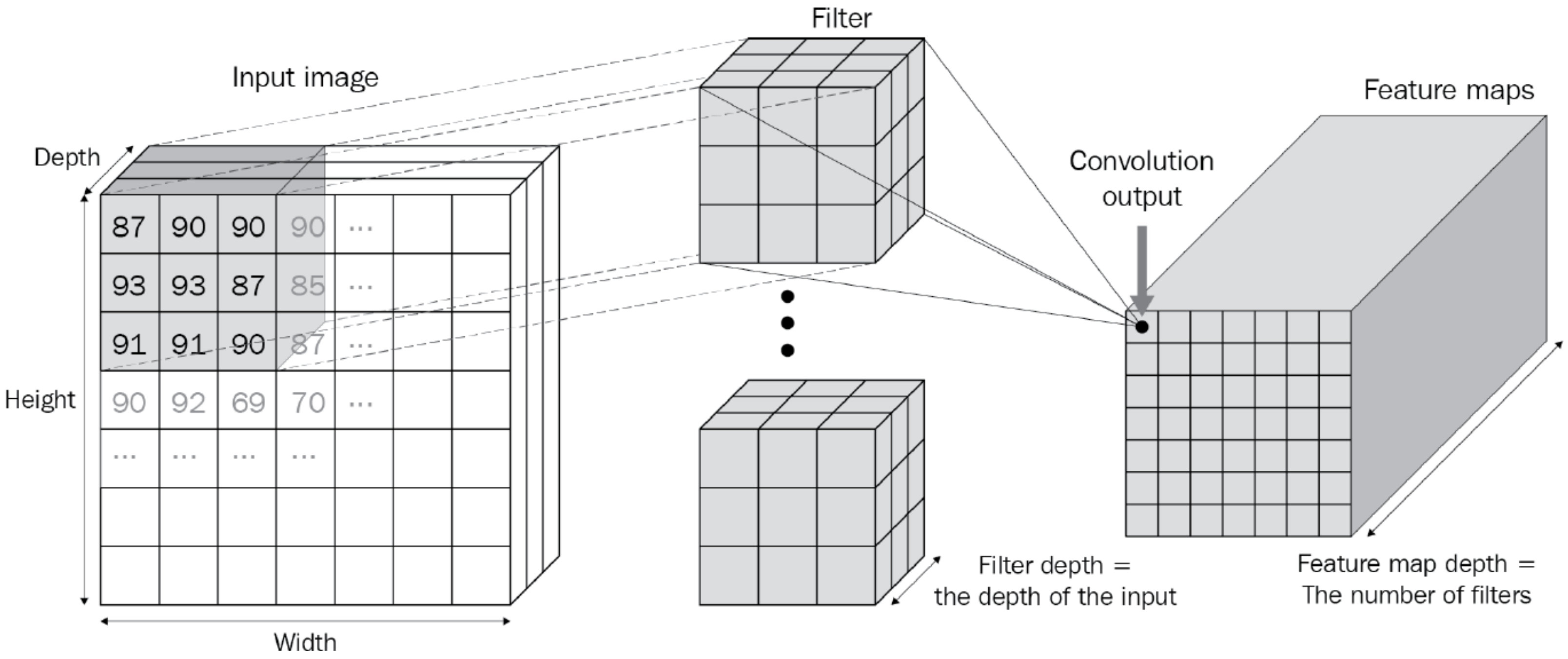
Addapted from Eduardo Fidalgo

# Convolutions over volumes



# Convolutions over volumes

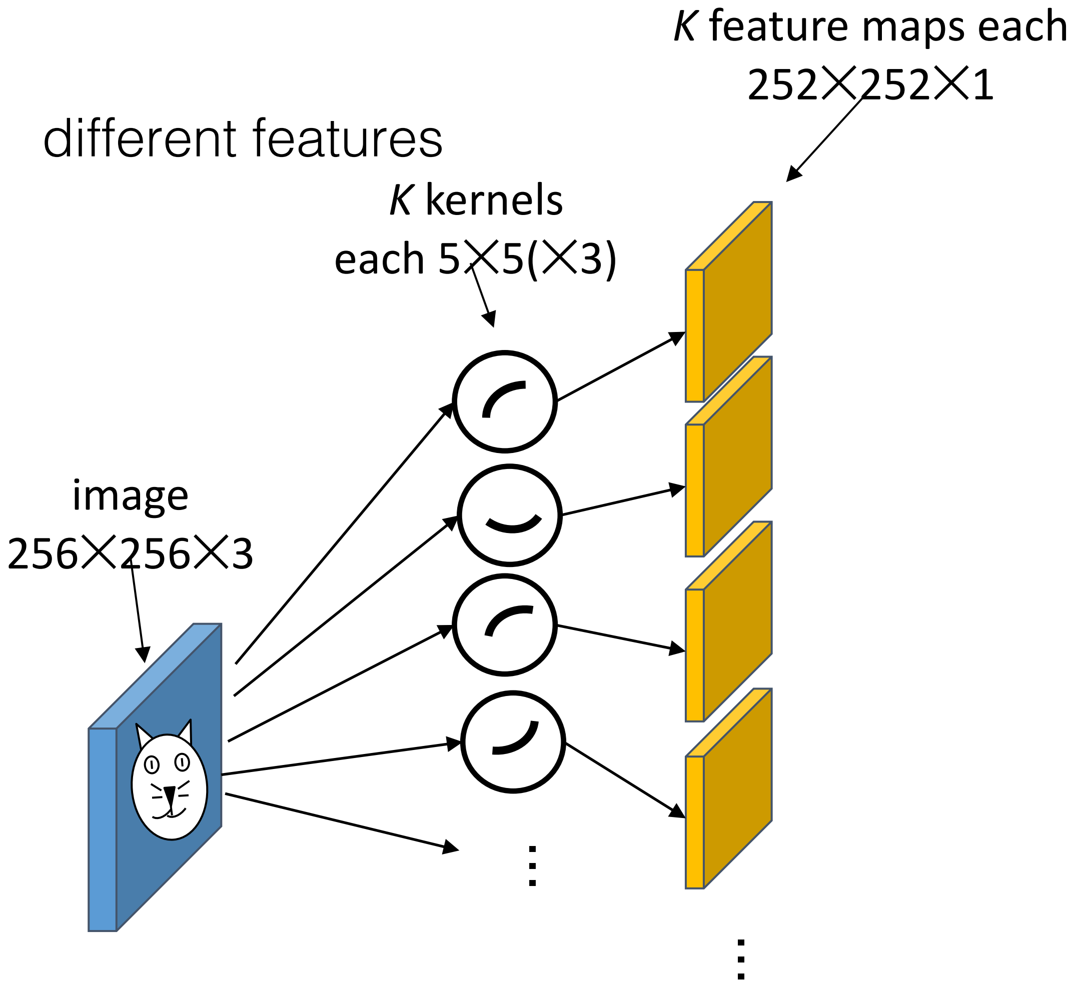


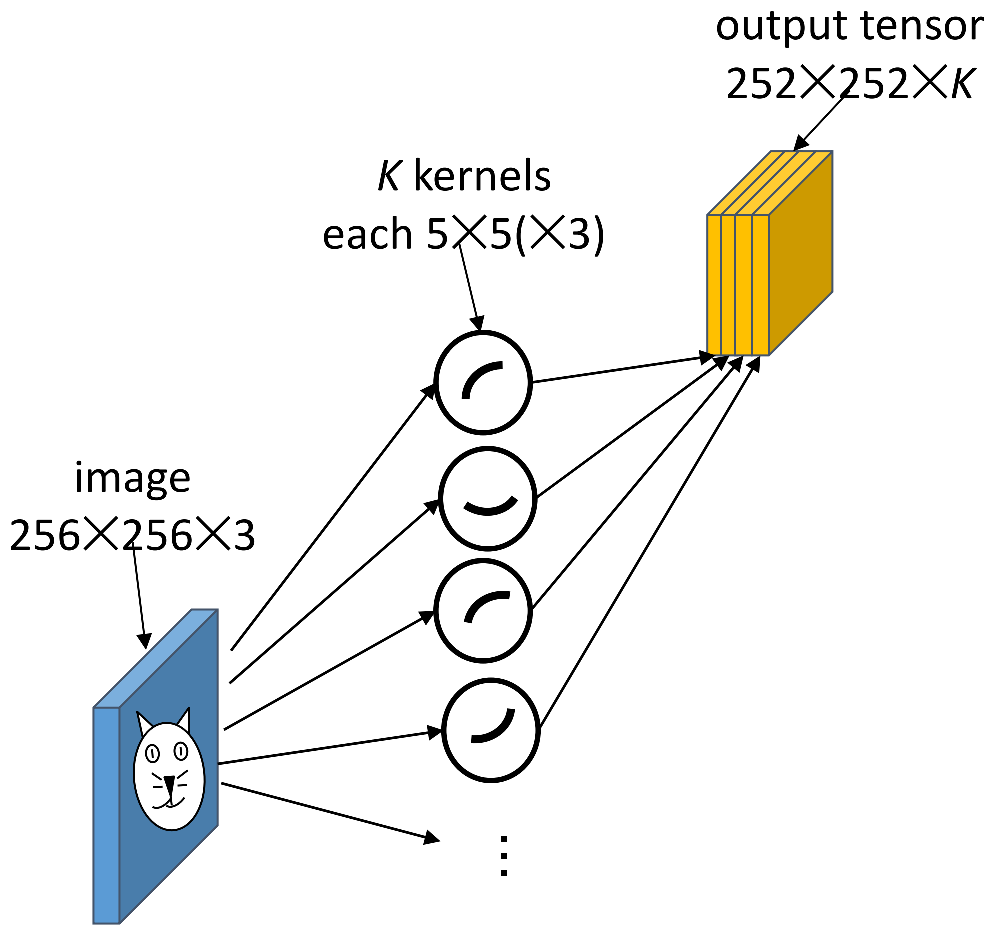

Output dimension: $\boxed{\left(\left\lfloor\frac{n_H+2p-f}{s}+1\right\rfloor, \left\lfloor\frac{n_W+2p-f}{s}+1\right\rfloor, \#filters\right)}$

# filters = # channels (depth) of the next layer

Figure from Modern Computer Vision with PyTorch

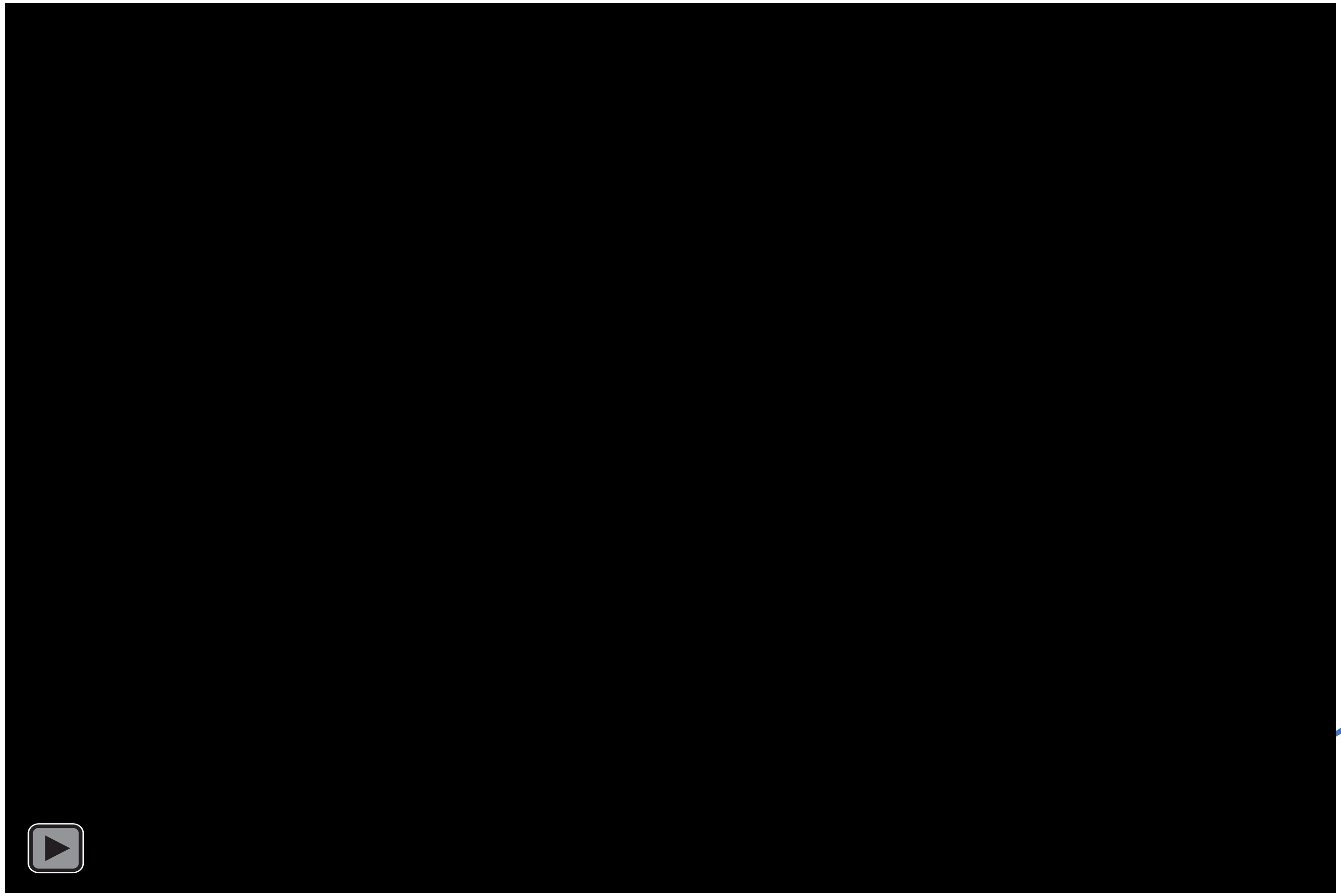
# Convolutions over volumes



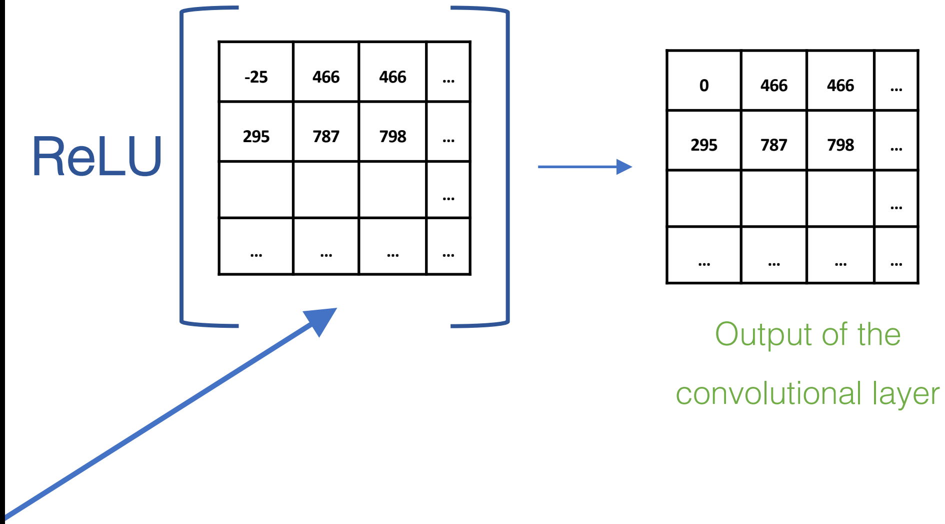




Adapted from CSC Traning for Brilliant Minds

# One layer of a CNN



**Bias** is a real number. The same bias to all elements of the output.




Other example: https://cs231n.github.io/assets/conv-demo/index.html

Figure by Sumit Saha.

# One layer of a CNN: number of parameters

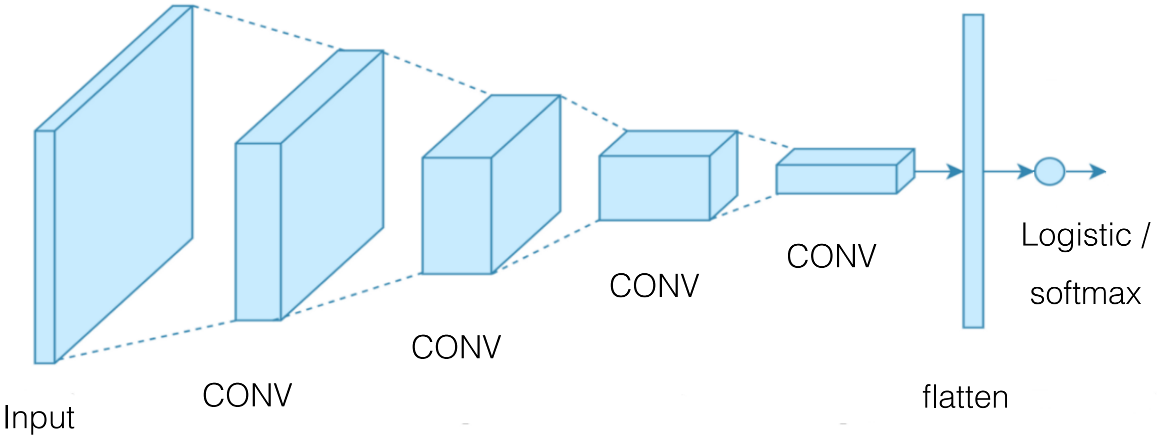
*If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?*

3 x 3 x 3 +1 (bias) = 28 parameters in each filter

28 x 10 = 280 parameters in total

No matter how big the input image is, the number of parameters remains fixed as 280!

# Simple (and incorrectly structured!) CNN example and types of layers in a CNN




```
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 28, 28, 1)]       0

conv2d_3 (Conv2D)            (None, 26, 26, 32)        320

conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496

conv2d_5 (Conv2D)            (None, 22, 22, 128)       73856

flatten_1 (Flatten)          (None, 61952)             0

dense_1 (Dense)              (None, 10)                619530
=================================================================
Total params: 712,202
Trainable params: 712,202
Non-trainable params: 0
```

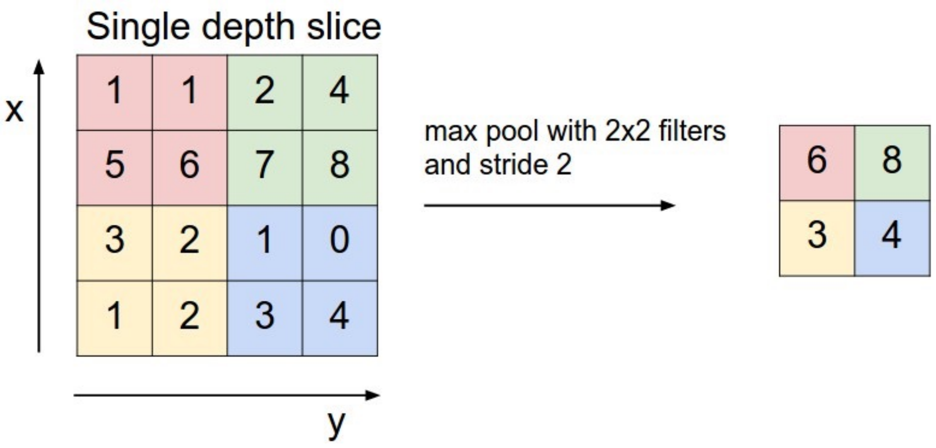
- It isn't conducive to learning a spatial hierarchy of features
- The final feature map is huge -> intense overfitting

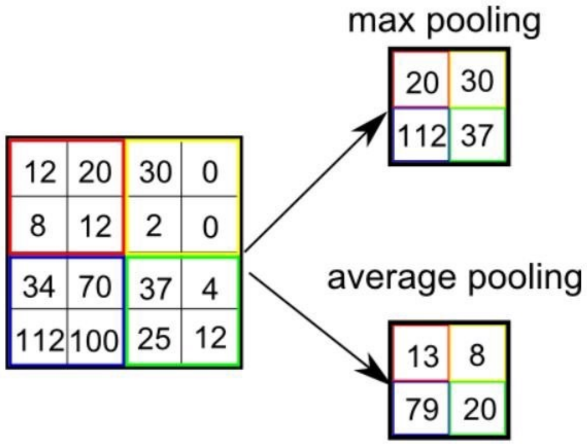
**As you go deeper, typically height and width decrease gradually and the number of channels (depth) increase**

Types of layers:

- Convolution (CONV)
- Pooling (POOL)
- Fully connected (FC)

# Pooling layers




Average pooling is not used very often (to collapse a vector dimensión)

Down sampling of feature maps:

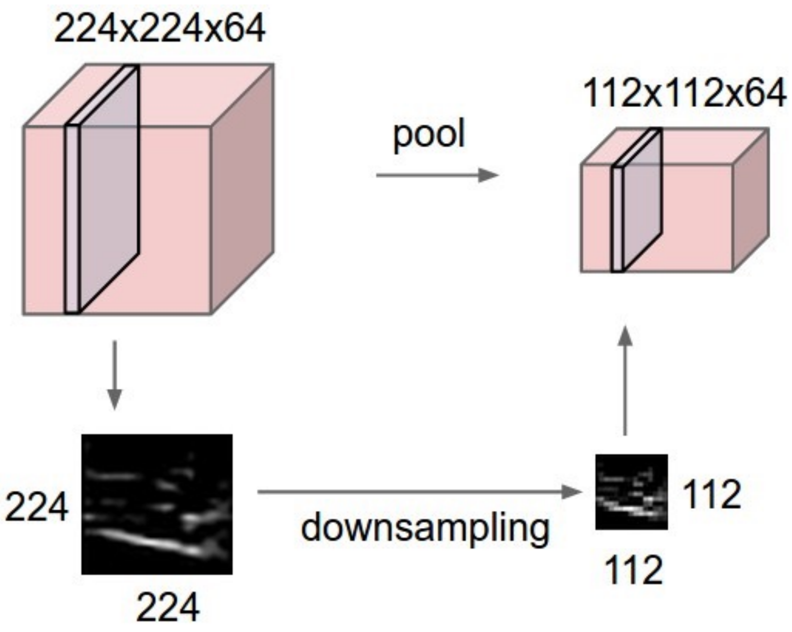
- Select one value for a f x f window
- If a feature exists in a region it is preserved after maxpooling

Hyperparameters:

- f: filter size
- s: stride
- Max, average, L2-norm,… pooling
- (Usually there is no padding)

No parameters to learn!

# Pooling layers




Reduce spatial dimensions (not depth)

- Increase computation efficiency
- Tolerance to small translations/noise
- Less risk to overfit

# Fully connected layer

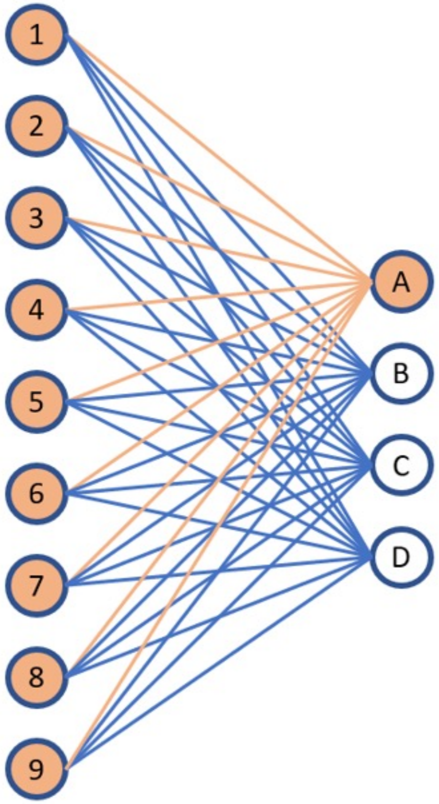



Figure by Diego Unzueta

Subsequent units receive input from ALL units in the previous layer

FC layers have high number of parameters

Softmax function or logistic regression is applied to the last FC layer

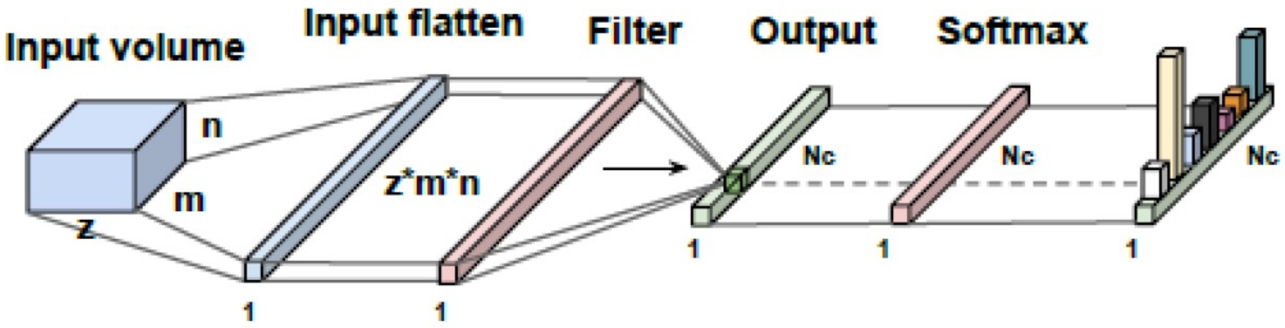
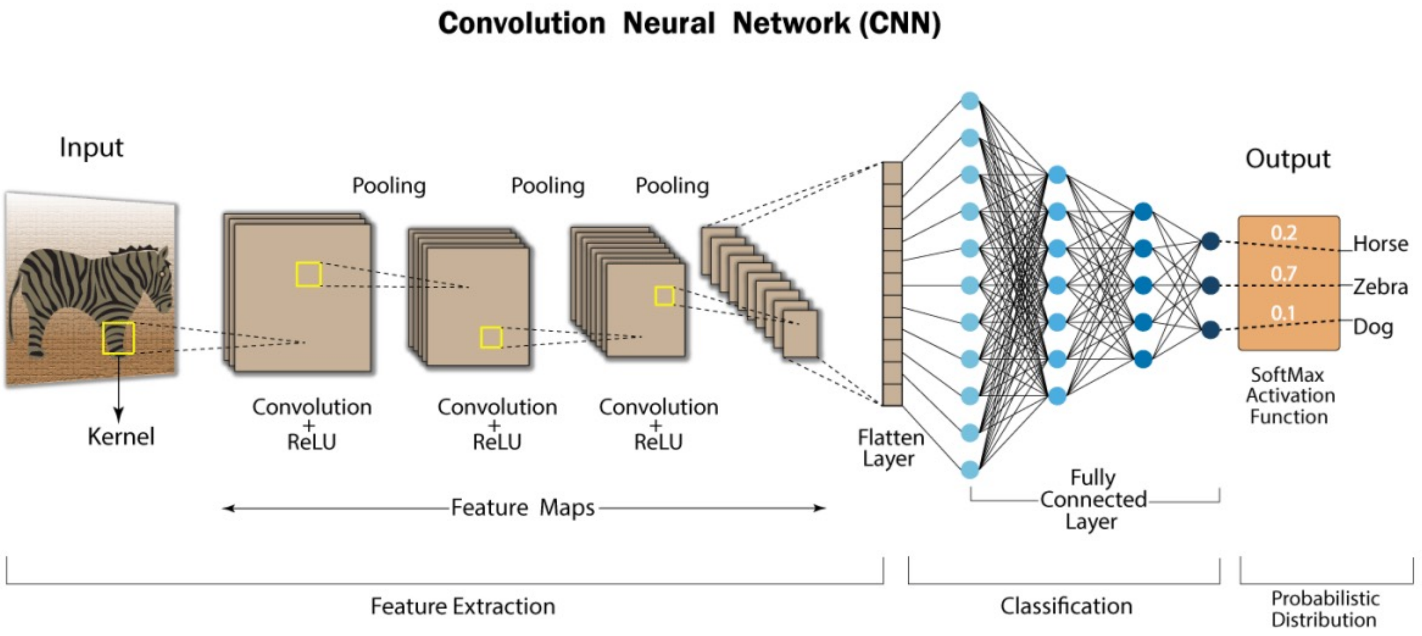



Figure from CVBLAB

# CNN example



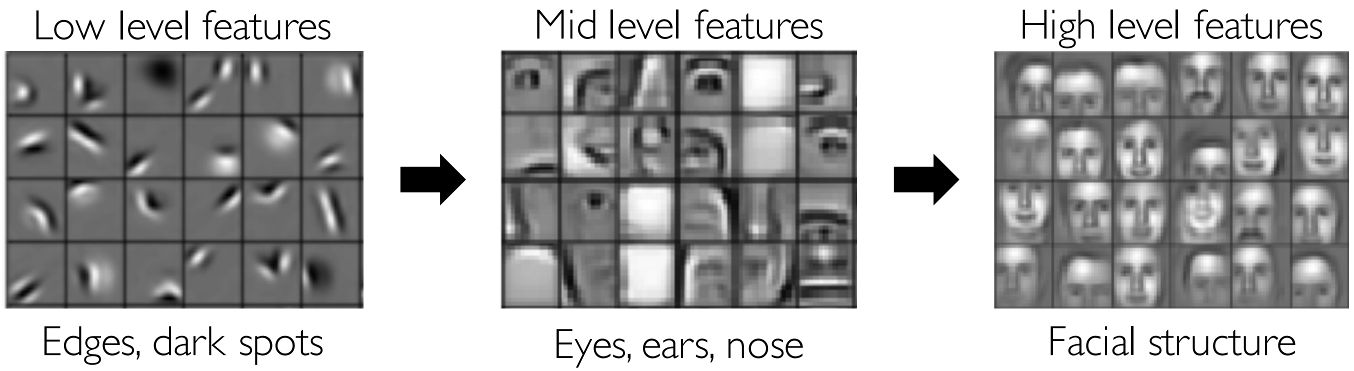

Typical architecture:

1. Input layer = image pixels
2. Convolution
3. ReLU
4. Pooling

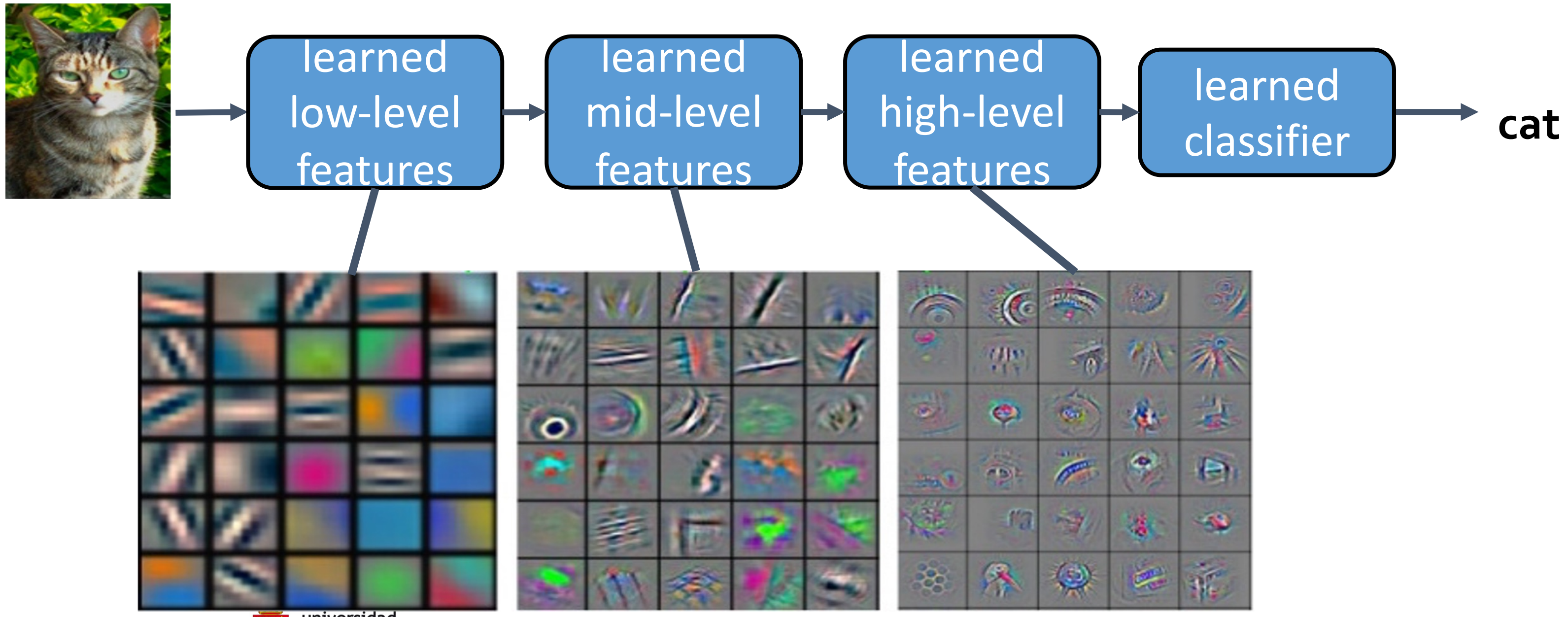
Repeat one or more times

5. One or more fully connected layers (+ReLU)
6. Final fully connected layer to get to the number of classes we want
7. Softmax to get probability distribution over classes

# Intuition of convolution in layers of a CNN




**Spatial filter hierarchy**: successive convolution filters look at increasingly larger windows



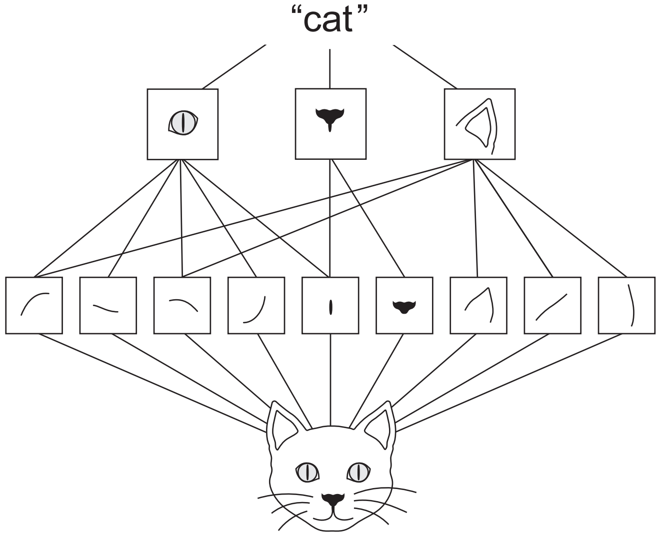




Image from Krizhevsky et al. NIPS (2012)

# Resources

CNN Explainer. Learn Convolutional Neural Network (CNN) in your browser!
https://poloclub.github.io/cnn-explainer/

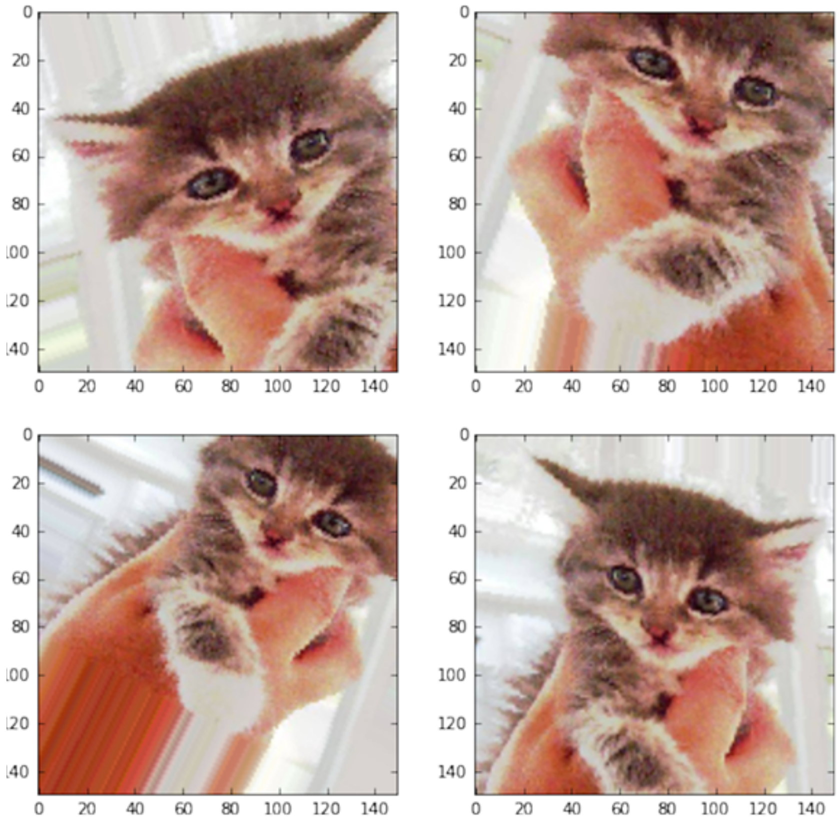
# Issues with CNNs

# Data augmentation

Overfitting is caused by having too few samples to learn from.

**Data augmentation** generates more training data from existing training samples.

Data is augmented via random transformations that yield believable-looking images. It helps expose the model to more aspects of the data and generalize better.






universidad de león

# Data augmentation: dropout

Still the inputs for the model would be heavily **intercorrelated**.

**Solution:** Add a **dropout** layer right before the fully connected layer.

Randomly sets to 0 a fraction of the inputs to the fully connected layer **during training time**, to prevent overfitting.

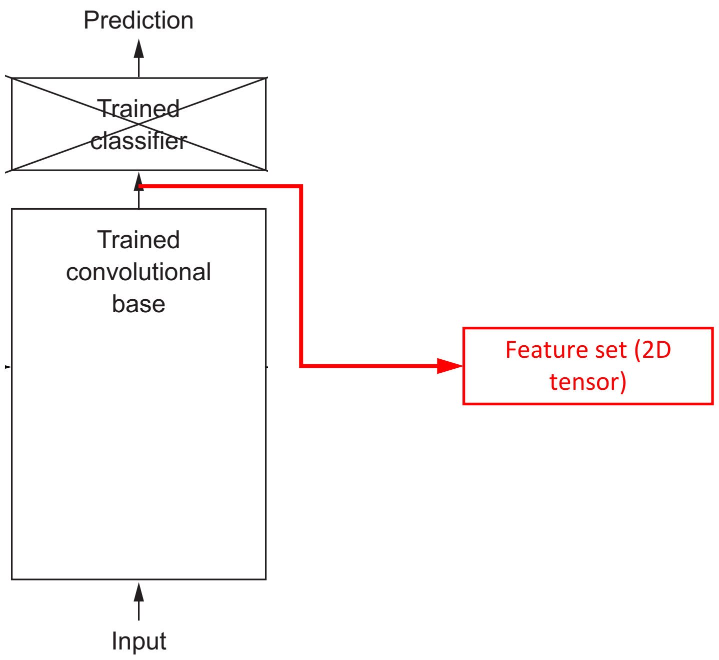
# Using pretrained CNNs

- **Pretrained network**: Saved network that was previously trained on a large dataset.
- Pretrained networks publicly available (e.g. in the module `keras.applications`).
- Examples:
  - Xception.
  - VGG19
  - VGG19
  - ResNet50
  - InceptionV3
  - …


universidad de león

# Using pretrained CNNs: Feature extraction (I)

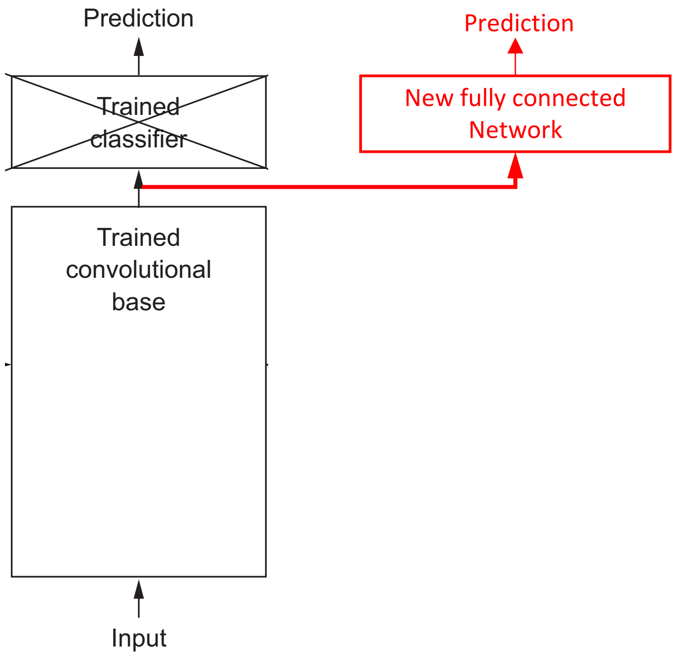
- Consists of using a pretrained network to extract features from new images using its *convolutional base*.
- These features are then run through a new classifier, which is trained from scratch.



# Using pretrained CNNs: Feature extraction (II)

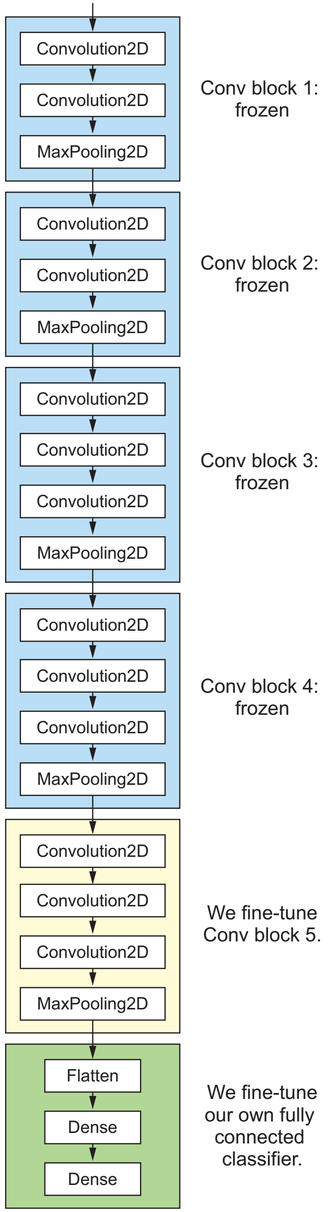
**Alternative**:

- Add fully connected layers on top of the convolutional base.
- Freeze the convolutional base (i.e. prevent its weights to be changed).
- Train the model with the frozen convolutional base using data augmentation.



# Using pretrained CNNs: fine tuning

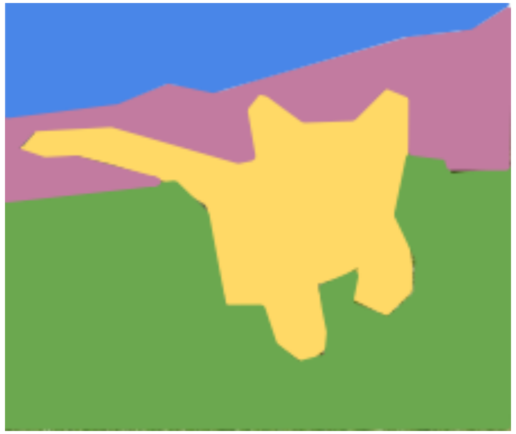
- Consists on freezing the convolutional base **except a few layers on top** of it, and jointly training this non-frozen part and the fully connected layers added on top of it.
- Only the weights on the top layers of the convolutional base will get adapted (fine-tuned) to this problem).



# Some applications

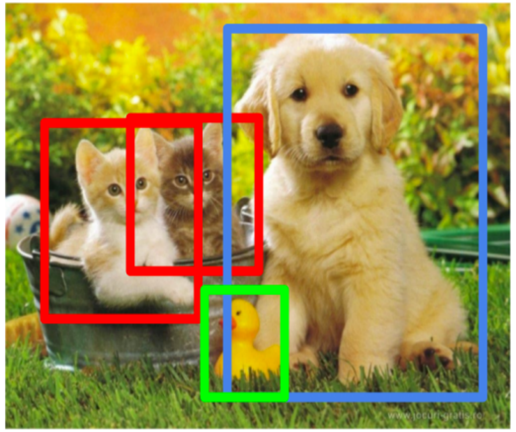
# Beyond classification

Semantic Segmentation



CAT

Object Detection



CAT, DOG, DUCK

Image Captioning

The cat is in the grass.

Instance Segmentation

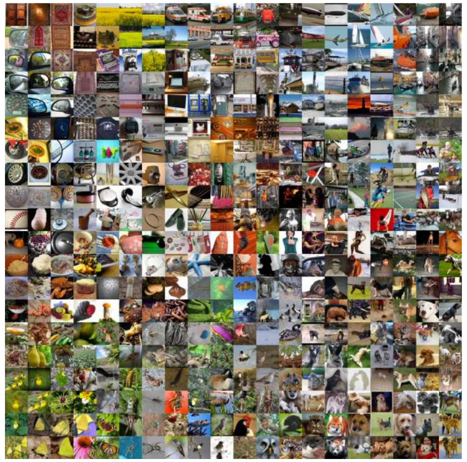
CAT, DOG, DUCK

# Data, data, data!

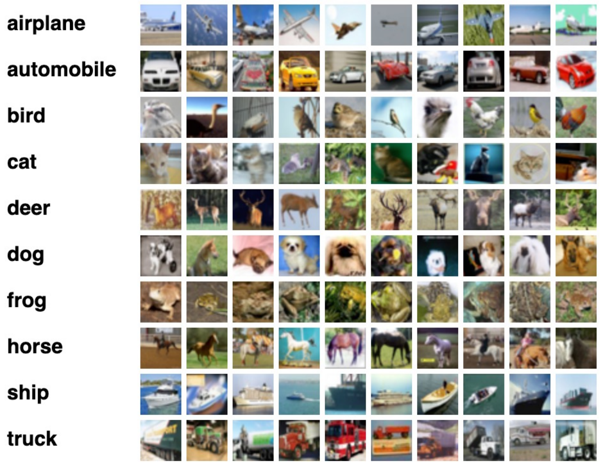
MNIST (Handwritten digits)

places
THE SCENE RECOGNITION DATABASE

Places (Natural Scenes)

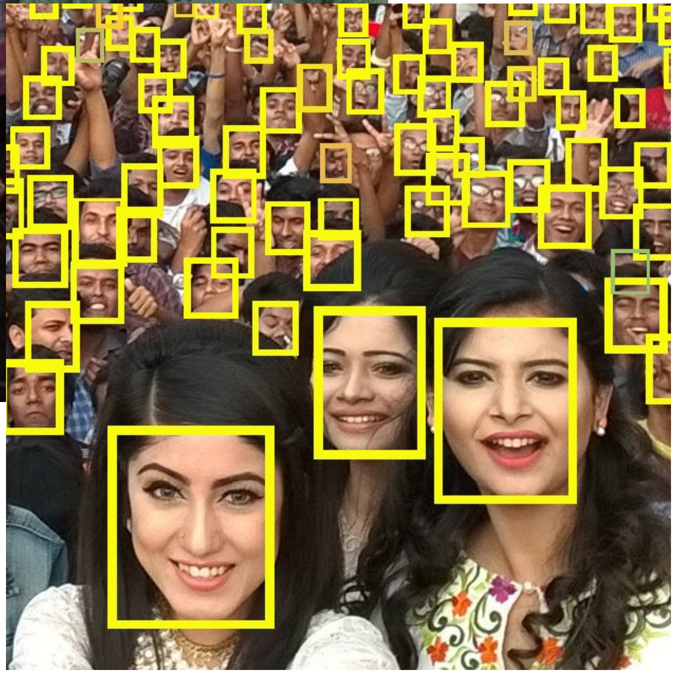


ImageNet
22K categories, 14M images

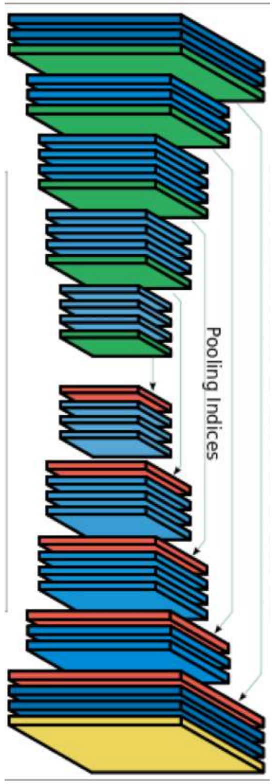



CIFAR-10 and CIFAR-100
10 or 100 categories, 60K images

# Face detection

# Self driving cars



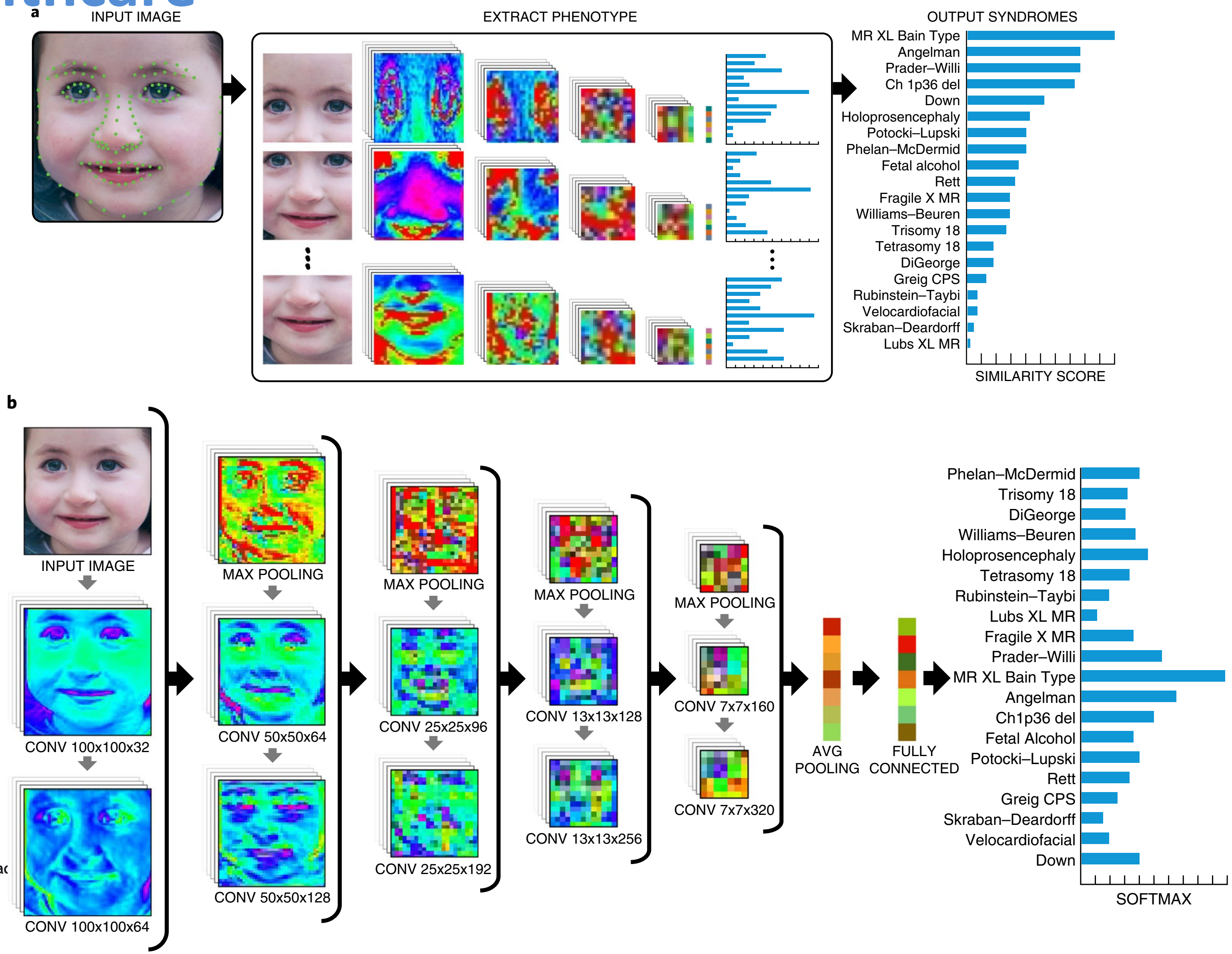




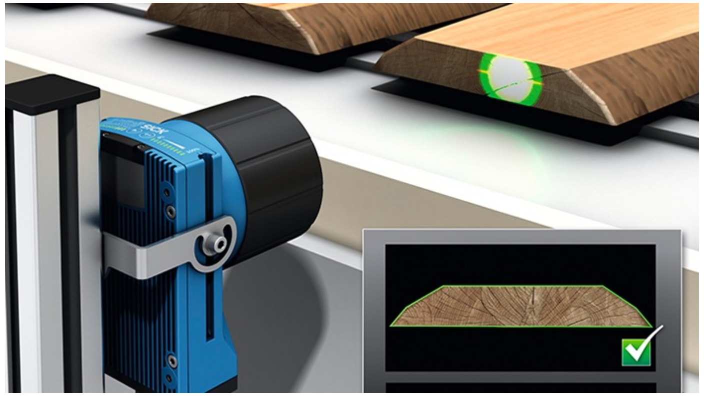
universidad de león

REGINNA 4.0

# Healthcare

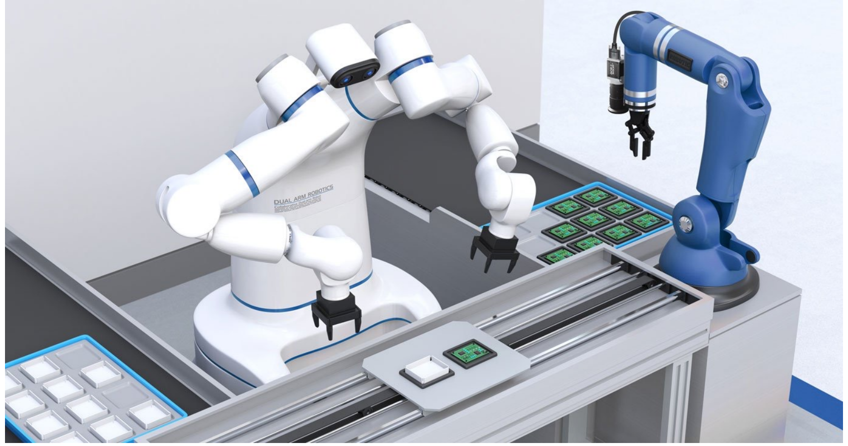

a
INPUT IMAGE
EXTRACT PHENOTYPE
OUTPUT SYNDROMES
MR XL Bain Type
Angelman
Prader–Willi
Ch 1p36 del
Down
Holoprosencephaly
Potocki–Lupski
Phelan–McDermid
Fetal alcohol
Rett
Fragile X MR
Williams–Beuren
Trisomy 18
Tetrasomy 18
DiGeorge
Greig CPS
Rubinstein–Taybi
Velocardiofacial
Skraban–Deardorff
Lubs XL MR
SIMILARITY SCORE
b
INPUT IMAGE
CONV 100x100x32
CONV 100x100x64
MAX POOLING
CONV 50x50x64
CONV 50x50x128
MAX POOLING
CONV 25x25x96
CONV 25x25x192
MAX POOLING
CONV 13x13x128
CONV 13x13x256
MAX POOLING
CONV 7x7x160
CONV 7x7x320
AVG POOLING
FULLY CONNECTED
Phelan–McDermid
Trisomy 18
DiGeorge
Williams–Beuren
Holoprosencephaly
Tetrasomy 18
Rubinstein–Taybi
Lubs XL MR
Fragile X MR
Prader–Willi
MR XL Bain Type
Angelman
Ch1p36 del
Fetal Alcohol
Potocki–Lupski
Rett
Greig CPS
Skraban–Deardorff
Velocardiofacial
Down
SOFTMAX

# Industry 4.0



**Inspection**

**Indoor augmented reality**

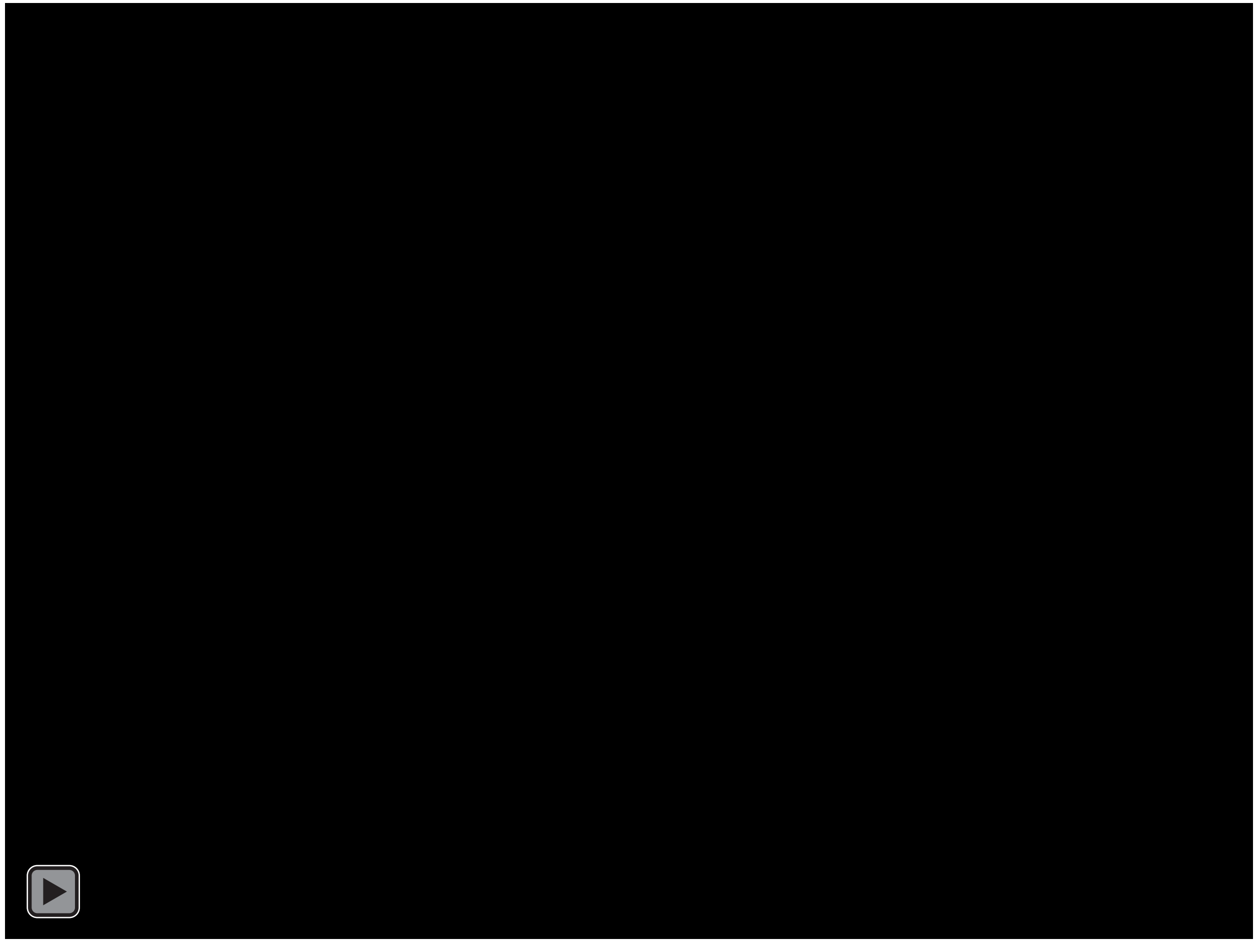


**Robotics**

Figure from https://www.sick.com/at/en/deep-learning-as-motor-for-industry-40/w/blog-deep-learning/
https://insidernavigation.com/ar-indoor-navigation/
https://tanhungha.com.vn/ung-dung-cua-camera-vision-cho-vision-guided-robotics-n290.html
https://www.anybotics.com/computer-vision-and-synthetic-data-are-key-to-training-autonomous-robots/

# Hands on: Automatic classification of inserts using image classification with CNN

## Hands on: Automatic classification of inserts using image classification with CNN



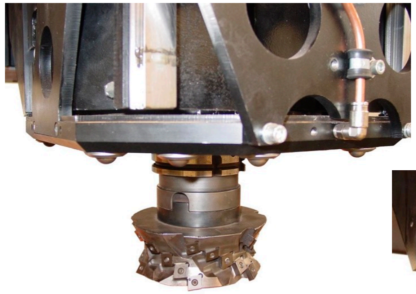
# Hands on: Automatic classification of inserts using image classification with CNN

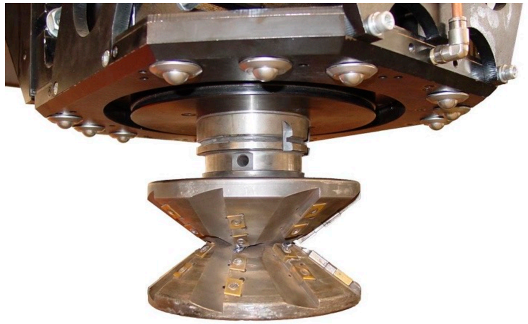
Prediction of wear of cutting tools operating in a single pass across thick plates

**Cutting machine**

Cutting using plasma or oxy-fuel

Milling the edge of the plate in order to leave the weld profiled



Wear monitoring of cutting tools operating in a **single pass** across **thick plates**

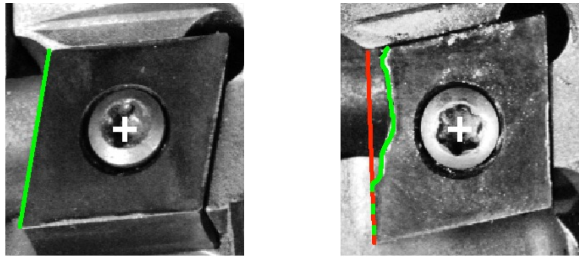
**Very aggressive**

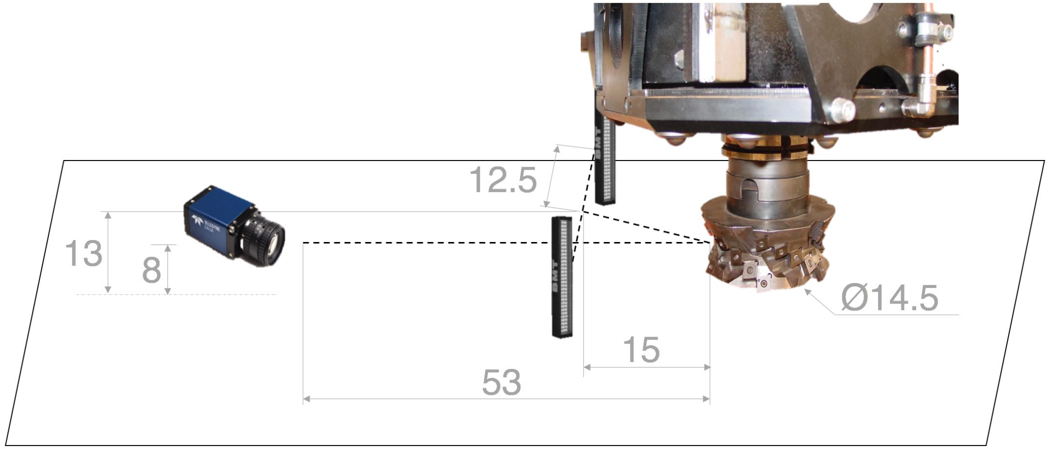
**After every pass**

**Of all inserts**

# Hands on: Automatic classification of inserts using image classification with CNN

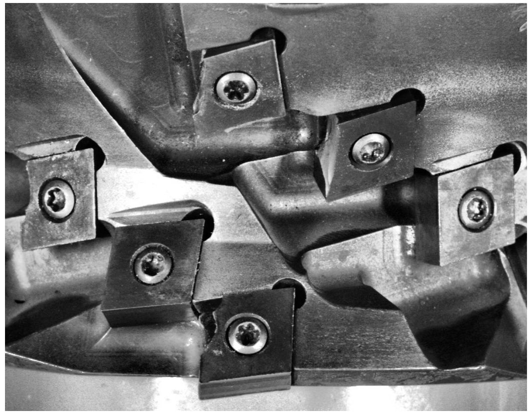
Unbroken Broken

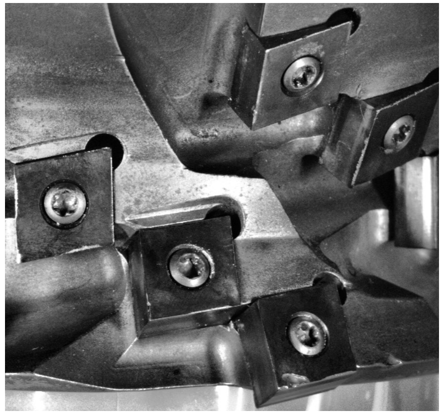





Breakage evaluation

Wear evaluation: Shape description
Texture description







universidad de león

REGINNA 4.0

REGINNA 4.0

Laura Fernández Robles
Universidad de León (Spain)
l.fernandez@unileon.es

Supported by
eit Manufacturing
Funded by the European Union
INNOVATION CAPACITY BUILDING FOR HIGHER EDUCATION
PART OF EIT HEI Initiative
eit-hei.eu