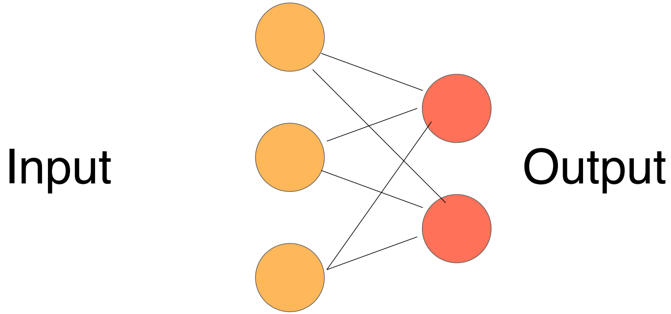
Deep Learning:
Convolutional Neural Networks (CNNs)

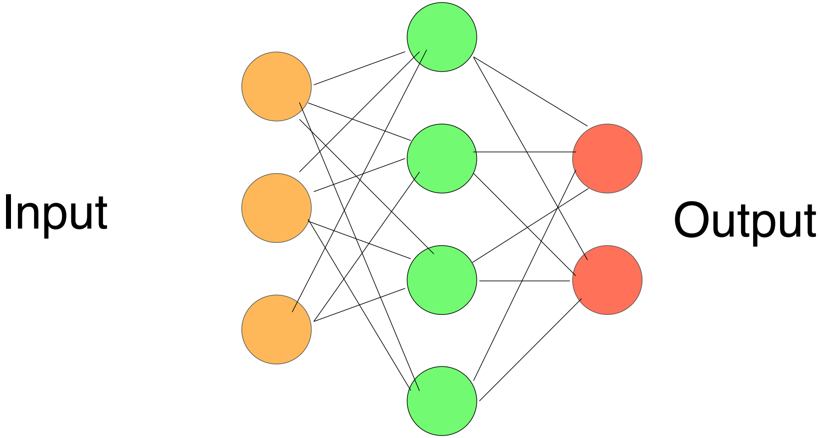
Manuel Sánchez-Montañés

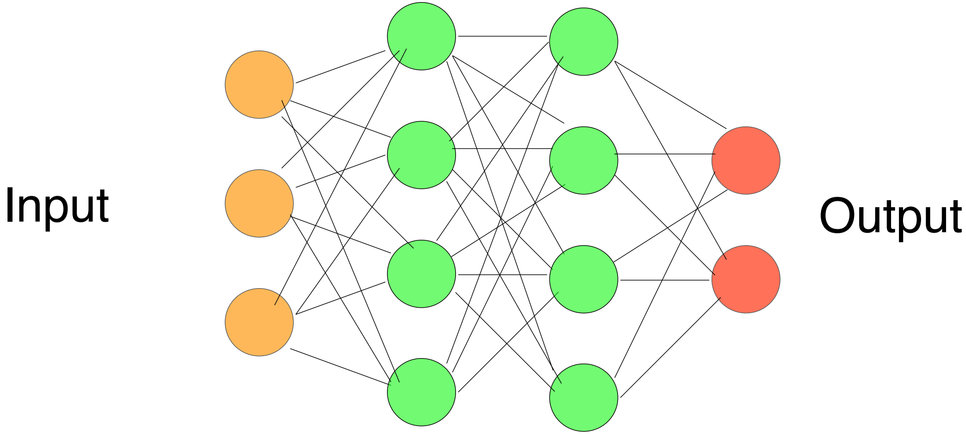
Escuela Politécnica Superior, Universidad Autónoma de Madrid

Limitations of shallow neural networks

Shallow neural networks







Shallow neural networks

Tabular data: one or two hidden layers are sufficient (with nonlinear activation function).
The number of hidden neurons must be adjusted

Shallow neural networks

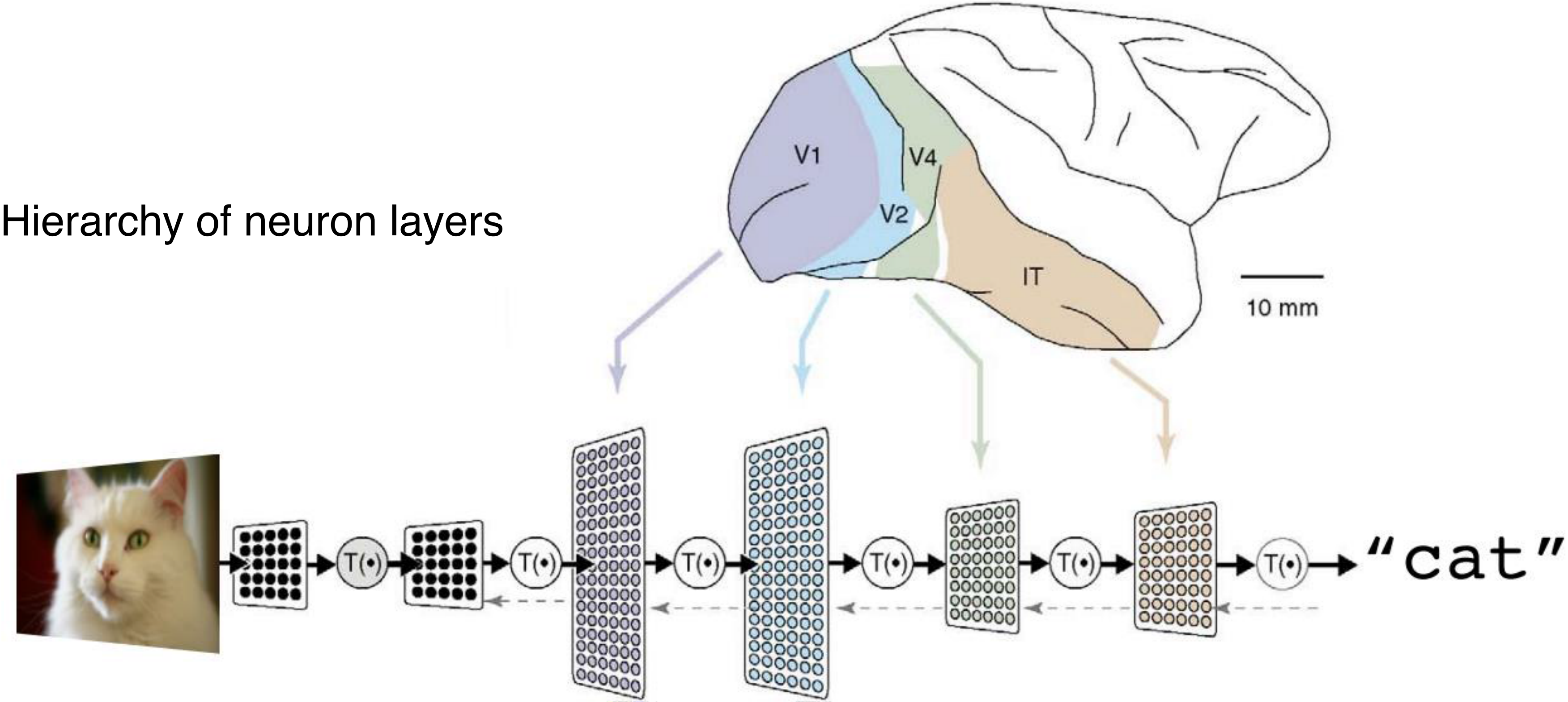
Data composed of a hierarchy of elements of the same nature (images, texts, audio, time series, genetic sequences, etc.):

Shallow neural networks do not generalize well on these problems.
An architecture that extracts these relationships in the data is needed.

Deep Convolutional Neural Networks (CNNs)

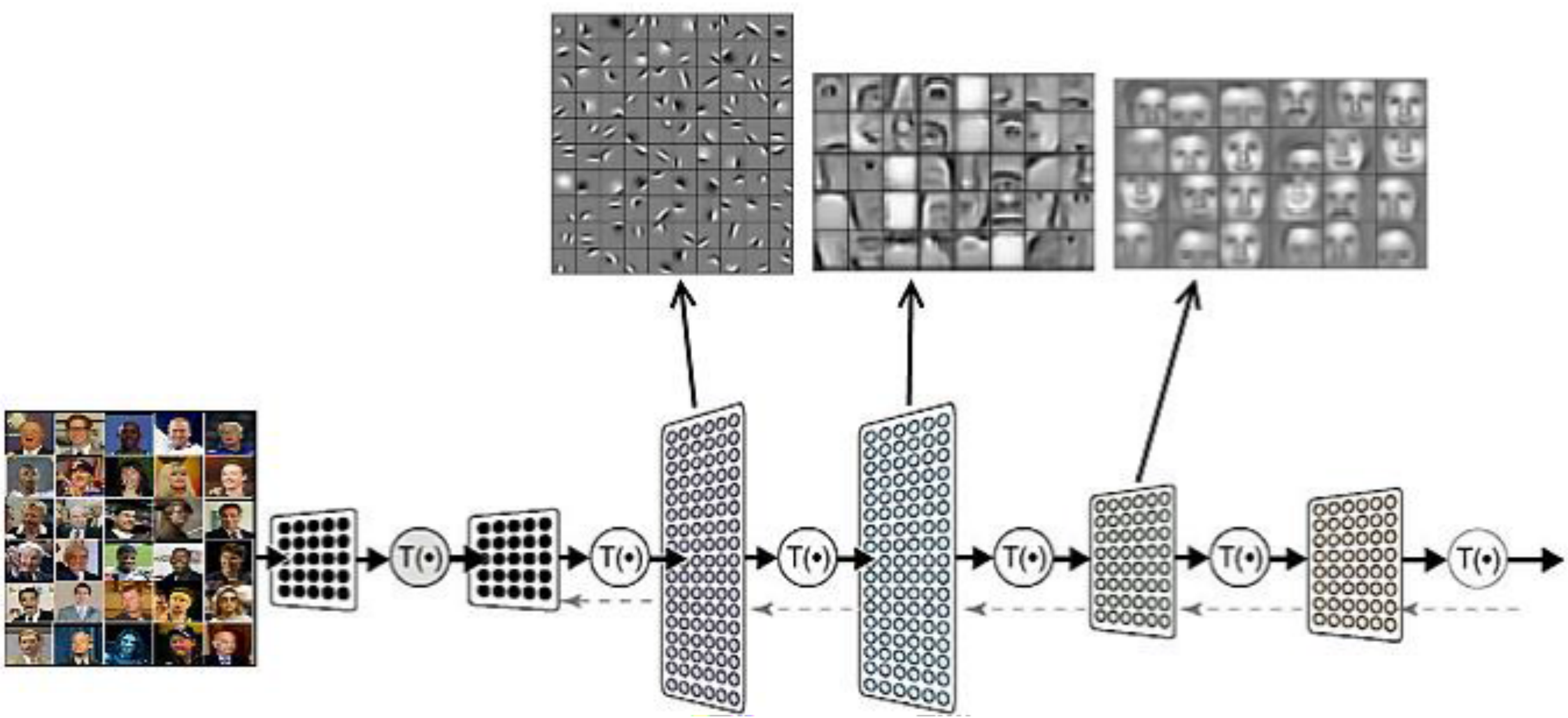
Convolutional networks

Hierarchy of neuron layers

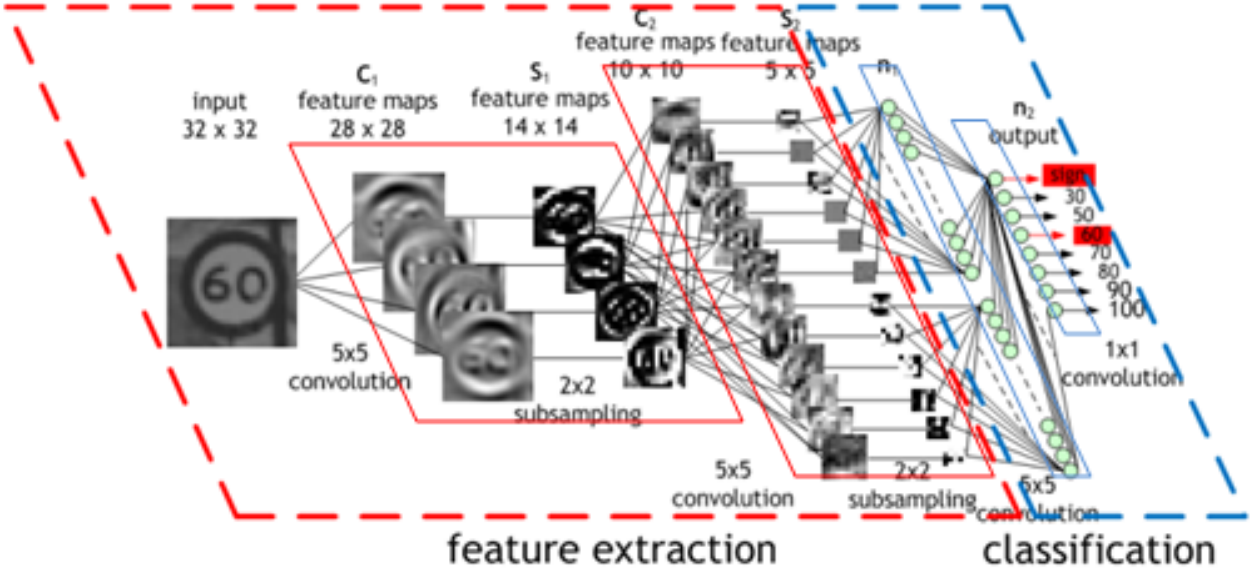


Google Talk by Jeff Dean at Seoul's Campus, 7/3/2016

Convolutional networks



Example of a convolutional network (CNN)



https://developer.nvidia.com/discover/convolutional-neural-network
Image: Maurice Peemen

Comparison of shallow versus convolutional networks

Shallow models versus deep models

Demo

Shallow dense network:
https://adamharley.com/nn_vis/mlp/3d.html

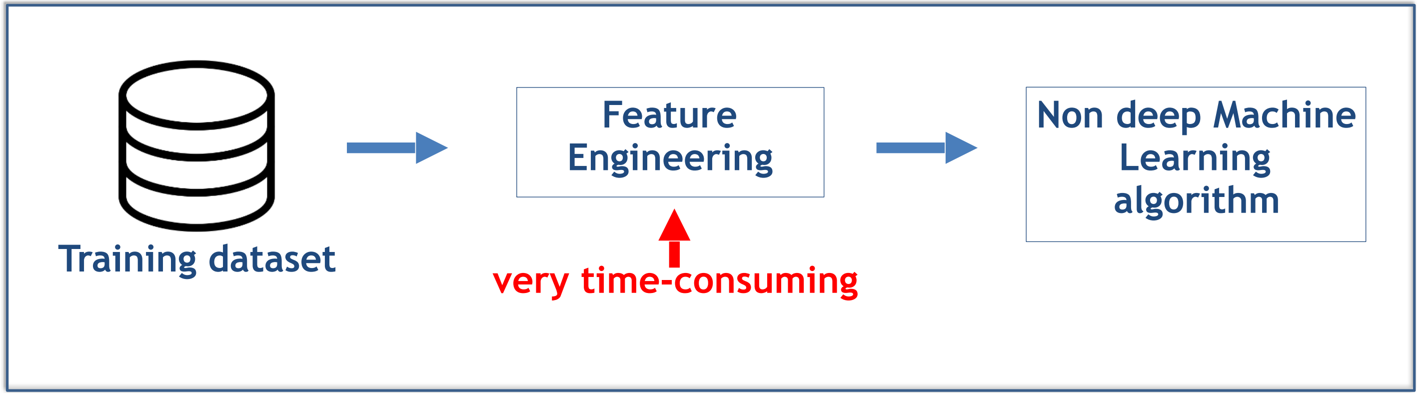
Shallow models versus deep models

Demo

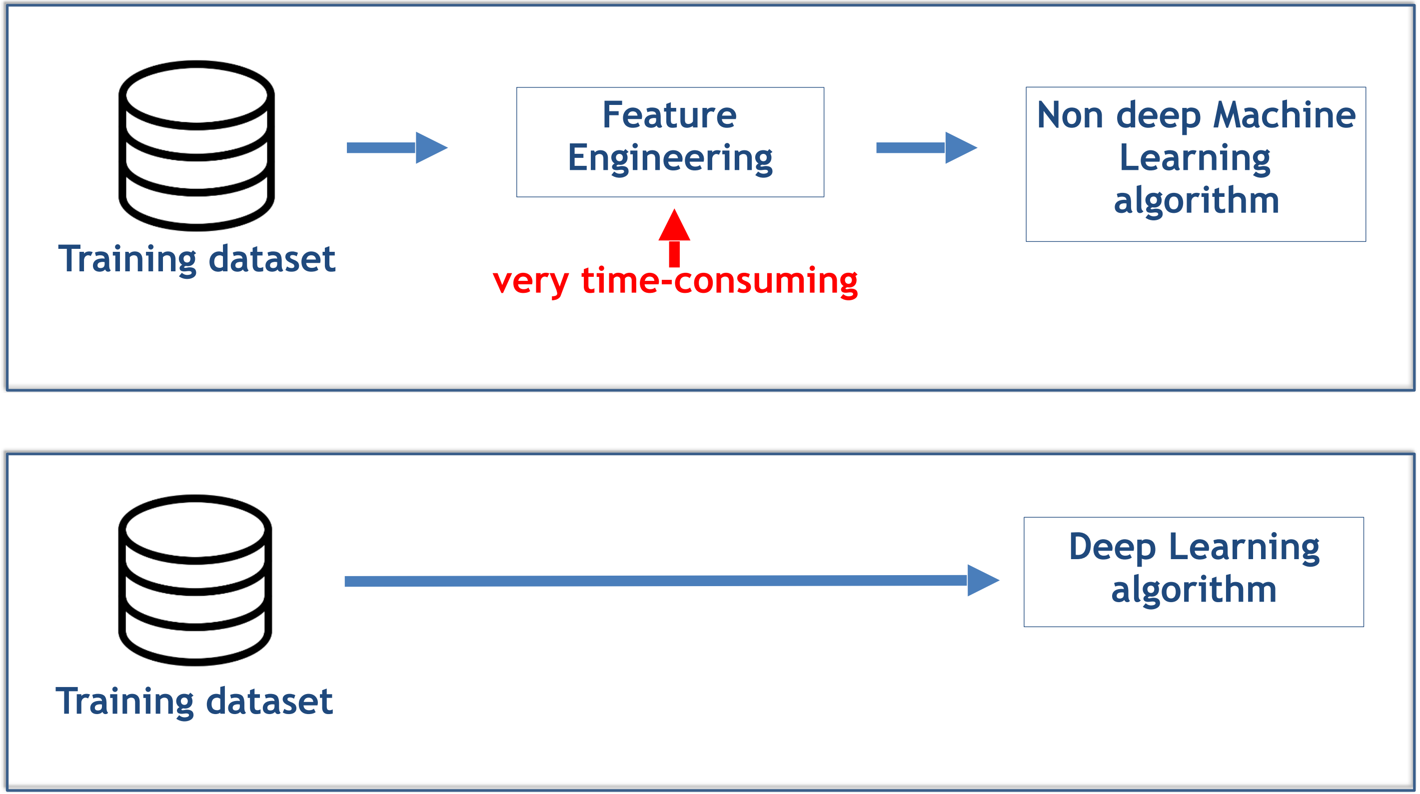
Shallow dense network:
https://adamharley.com/nn_vis/mlp/3d.html

Convolucional neural network:
https://adamharley.com/nn_vis/cnn/3d.html

Shallow models versus deep models

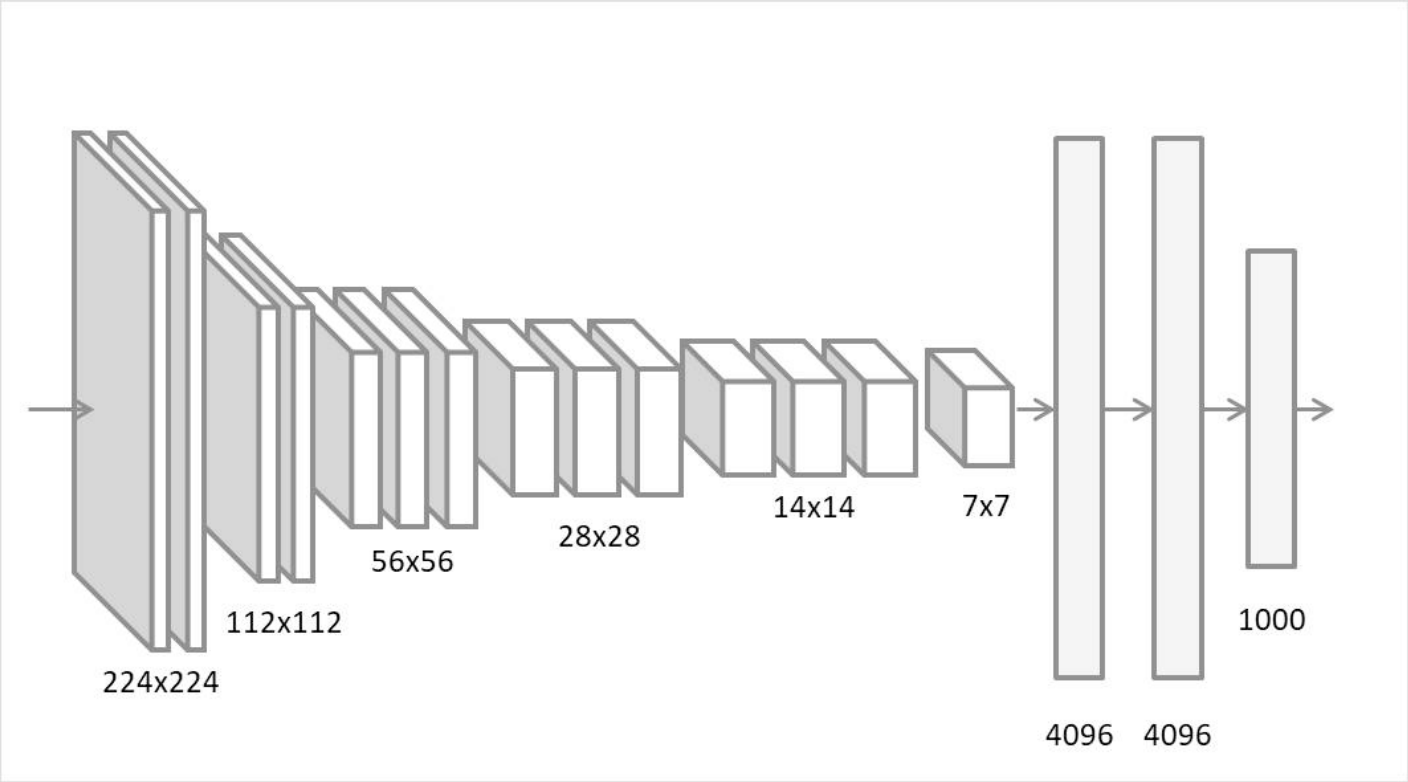


Shallow models versus deep models

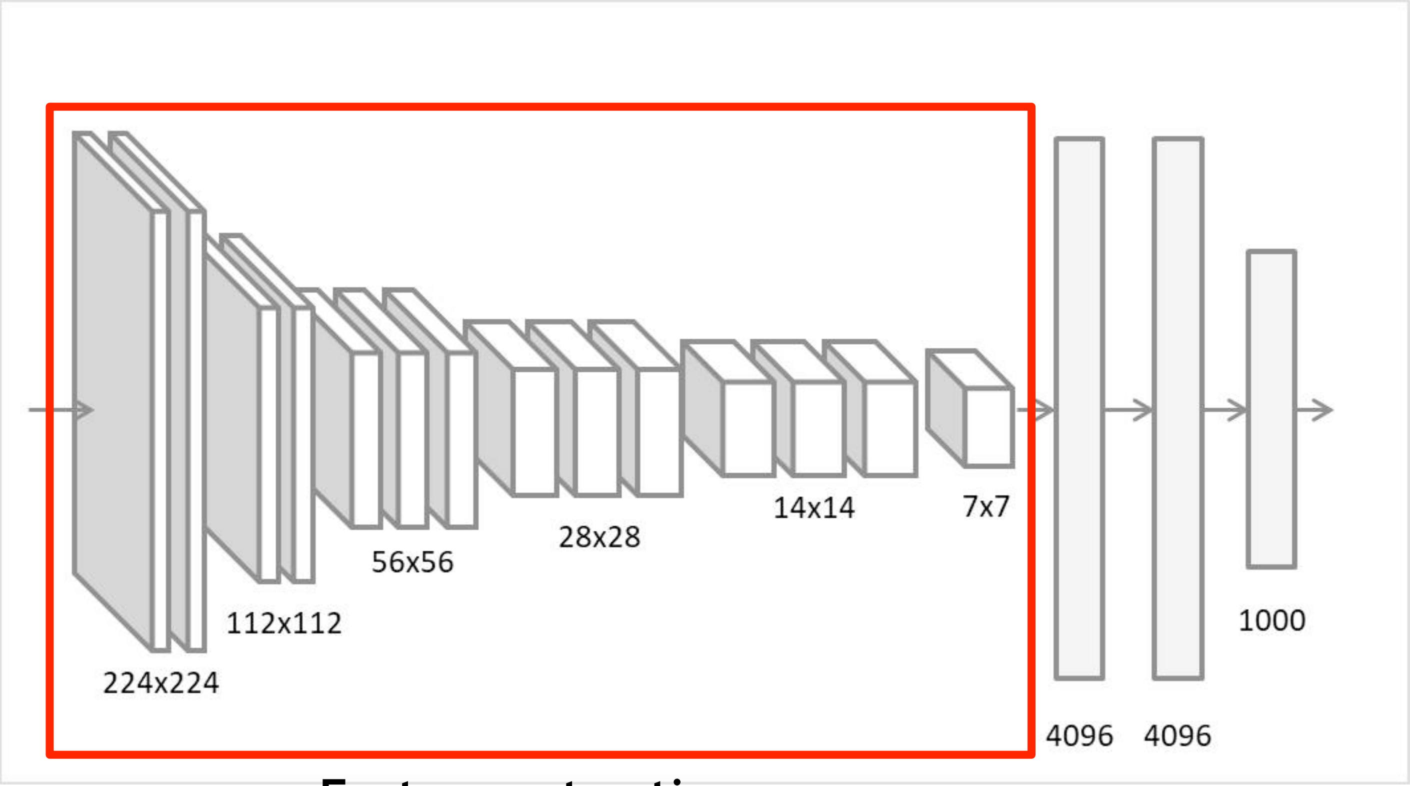


Convolutional Neural Networks: Arquitecture

Architecture of a CNN

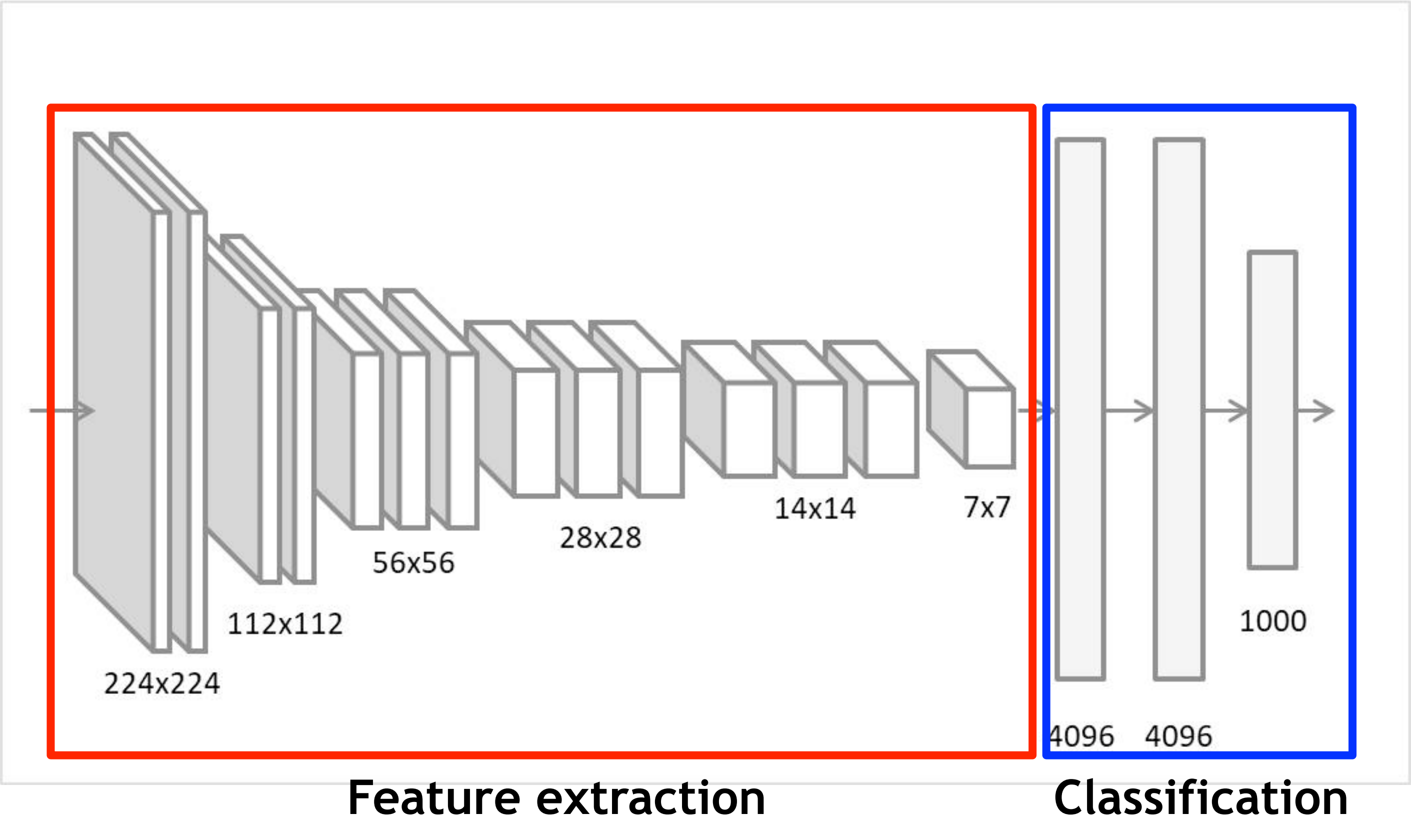


Architecture of a CNN

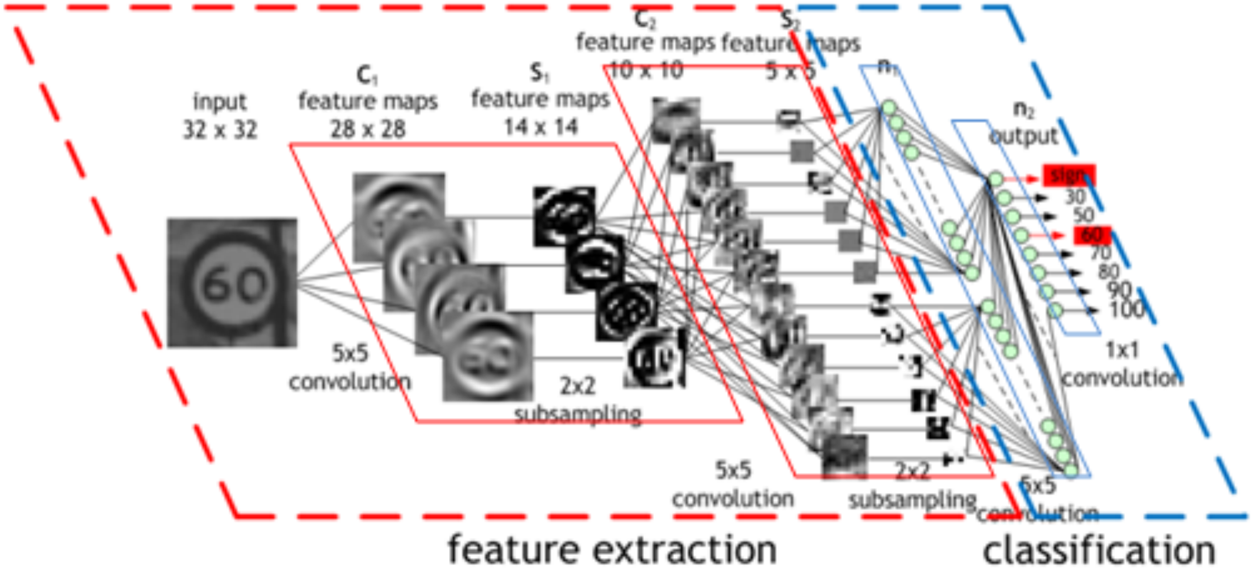


Feature extraction

Architecture of a CNN

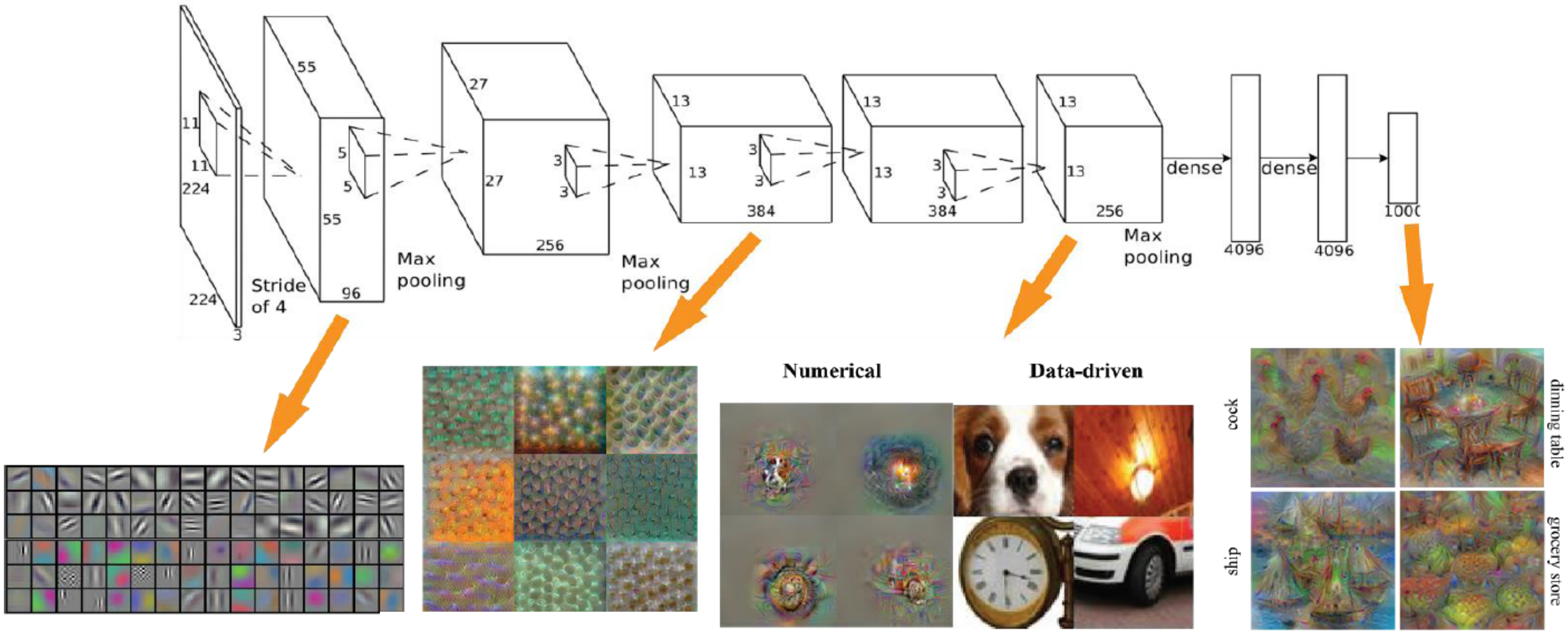


Architecture of a CNN



https://developer.nvidia.com/discover/convolutional-neural-network
Image: Maurice Peemen

Architecture of a CNN



Conv 1: Edge+Blob

Conv 3: Texture

Conv 5: Object Parts

Fc8: Object Classes

Types of basic layers in a CNN

Types of basic layers

Convolutional layer

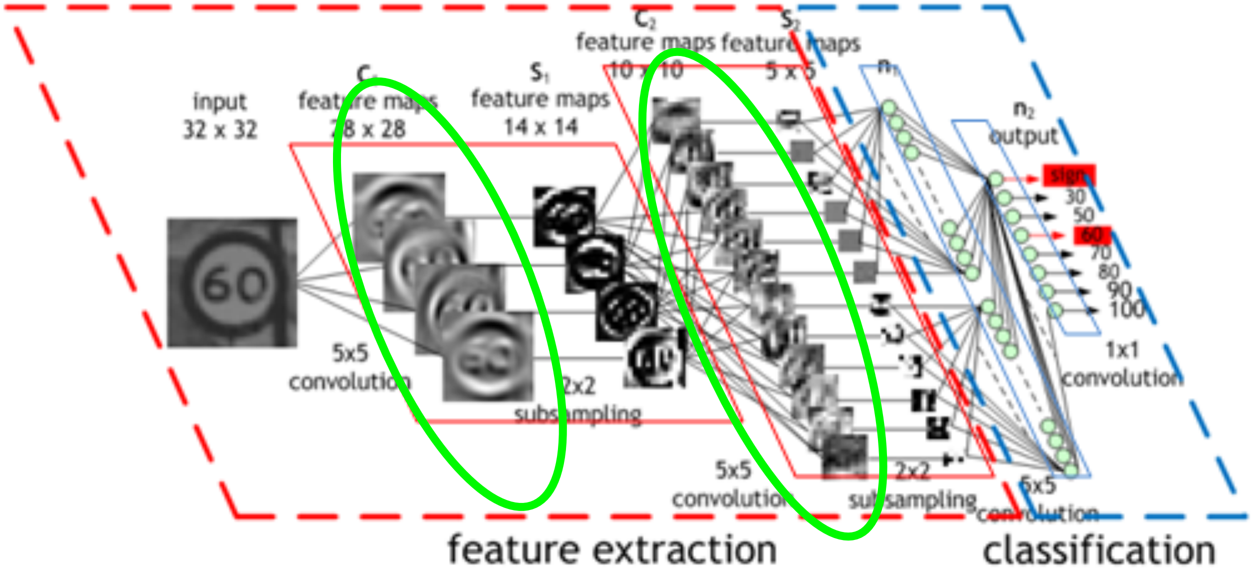
Pooling layer

Flattening layer

Dense layer

Convolutional layer

Convolutional layer



- They create "filtered" versions of the image that reaches them
- Each filter is focused on extracting a particular feature

Convolutional layer

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Kernel 3x3

0.	2	0
-1	2	0
-1	2	0

Filtered image

0	2	3	0	-1
0	2	5	-1	-1
0	0	6	-3	0
0	0	6	-3	0
0	0	4	-2	0

Pixels with higher value

0 if ReLU is applied

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0$

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0$

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0$

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

-1·0 + 2·0 + 0·0 +
-1·0

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$
$-1 \cdot 0 + 2 \cdot 0$

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$
$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 1$

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

-1·0 + 2·0 + 0·0 +
-1·0 + 2·0 + 0·1 +
-1·0

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$
$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 1 +$
$-1 \cdot 0 + 2 \cdot 0$

Filtered image

?				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$$\begin{aligned}&-1\cdot 0 + 2\cdot 0 + 0\cdot 0 + \\ &-1\cdot 0 + 2\cdot 0 + 0\cdot 1 + \\ &-1\cdot 0 + 2\cdot 0 + 0\cdot 0 \quad = \mathbf{0}\end{aligned}$$

Filtered image

0				

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$
$-1 \cdot 0$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1\cdot 0 + 2\cdot 0 + 0\cdot 0 +$
$-1\cdot 0 + 2\cdot 1$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$
$-1 \cdot 0 + 2 \cdot 1 + 0 \cdot 1$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$
$-1 \cdot 0 + 2 \cdot 1 + 0 \cdot 1 +$
$-1 \cdot 0$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$$-1 \cdot 0 + 2 \cdot 0 + 0 \cdot 0 +$$
$$-1 \cdot 0 + 2 \cdot 1 + 0 \cdot 1 +$$
$$-1 \cdot 0 + 2 \cdot 0$$

Filtered image

0	?			

Convolutional layer

Kernel 3x3

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

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Calculation:

$$
\begin{aligned}
&-1\cdot 0 + 2\cdot 0 + 0\cdot 0 + \\
&-1\cdot 0 + 2\cdot 1 + 0\cdot 1 + \\
&-1\cdot 0 + 2\cdot 0 + 0\cdot 1 \quad = \mathbf{2}
\end{aligned}
$$

Filtered image

0	2			

Convolutional layer

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Kernel 3x3

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

Filtered image

0	2	3	0	-1
0	2	5	-1	-1
0	0	6	-3	0
0	0	6	-3	0
0	0	4	-2	0

Convolutional layer

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Kernel 3x3

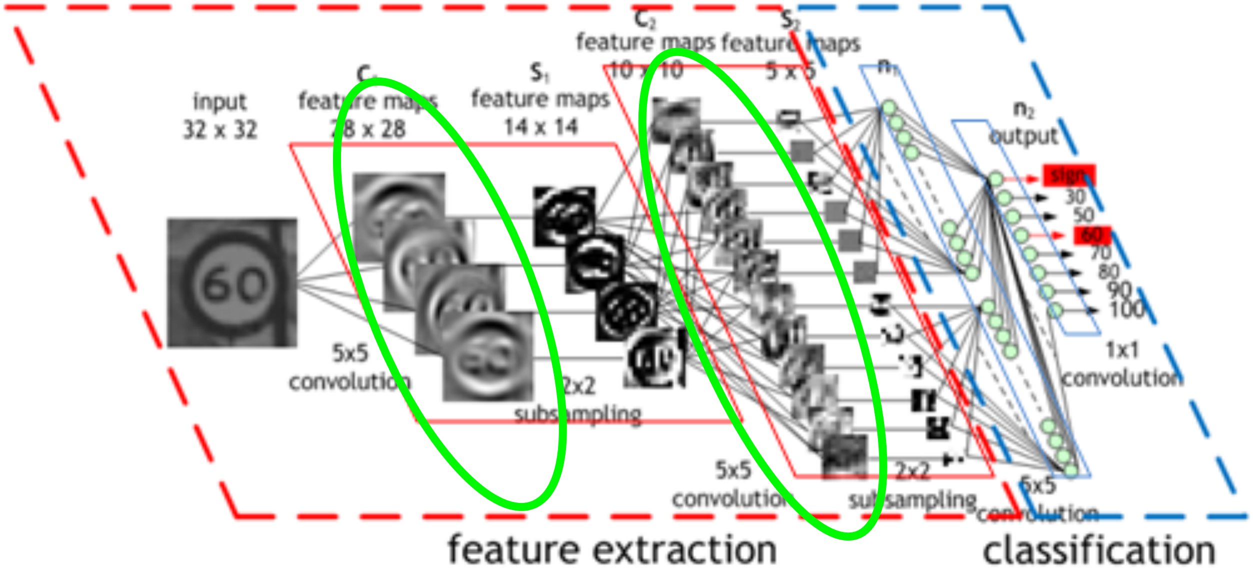
-1	2	0
-1	2	0
-1	2	0

Filtered image

0	2	3	0	-1
0	2	5	-1	-1
0	0	6	-3	0
0	0	6	-3	0
0	0	4	-2	0

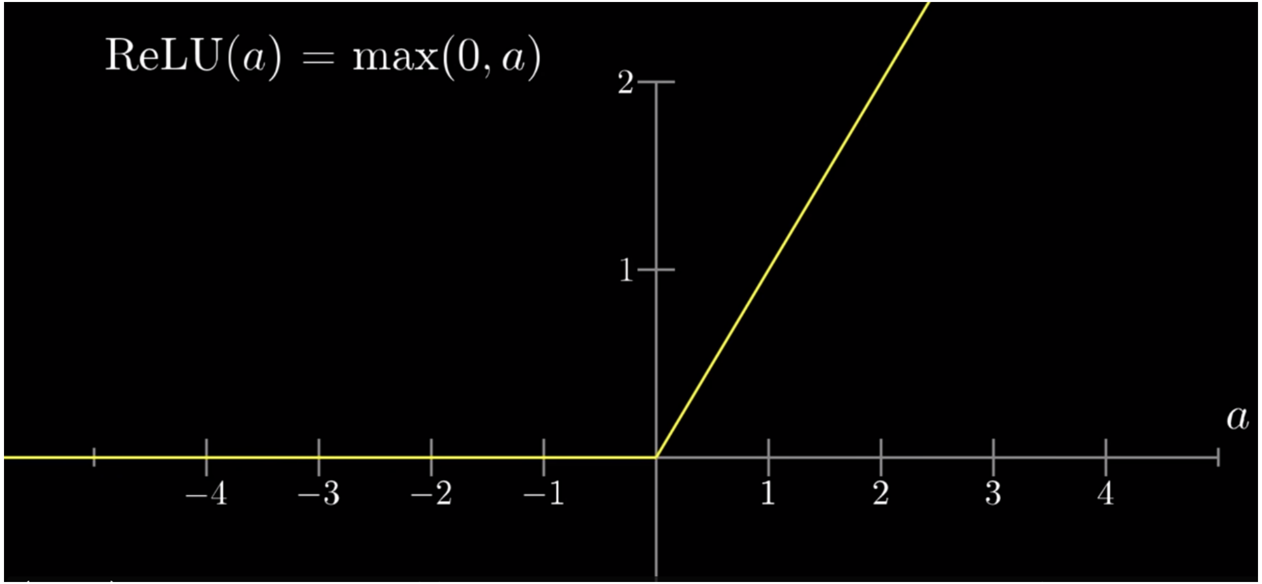
Note:
Each kernel has another parameter that is learned, the bias, which would be added to the final calculation. In these examples we assume for simplicity that this constant is 0

Convolutional layer



Convolutional layer

If ReLU is added:



Convolutional layer

If ReLU is added:

Input image

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Kernel 3x3

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

Filtered image (before ReLU)

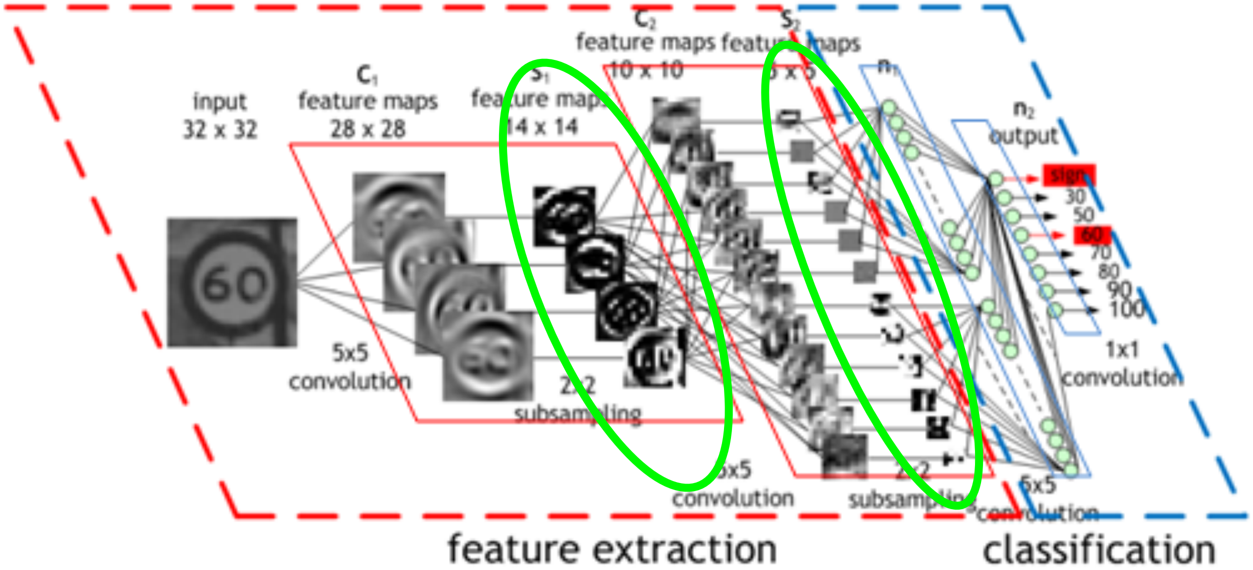
0	2	3	0	-1
0	2	5	-1	-1
0	0	6	-3	0
0	0	6	-3	0
0	0	4	-2	0

Filtered image (after ReLU)

0	2	3	0	0
0	2	5	0	0
0	0	6	0	0
0	0	6	0	0
0	0	4	0	0

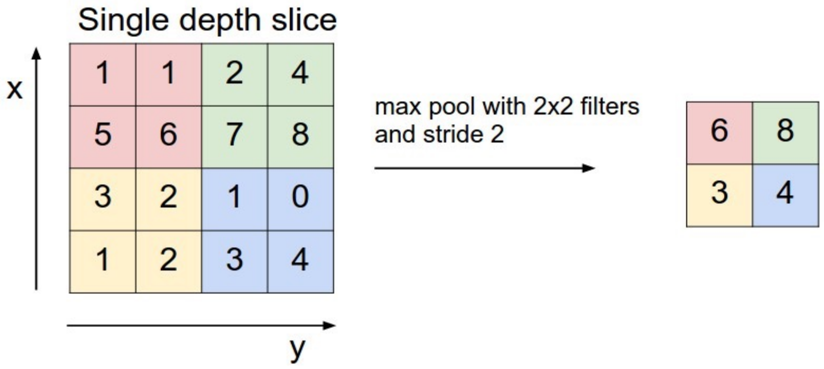
Pooling layer

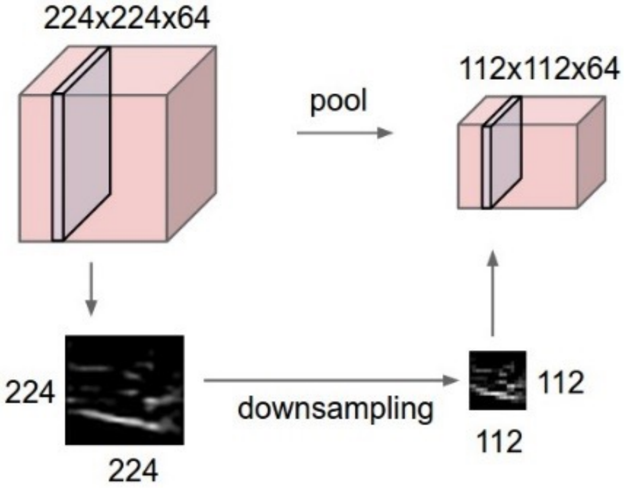
Pooling layer



- Pooling layer creates low-resolution versions of the images that reach it
- It forces the next layer to focus on extracting more global characteristics

Example: max-pooling layer

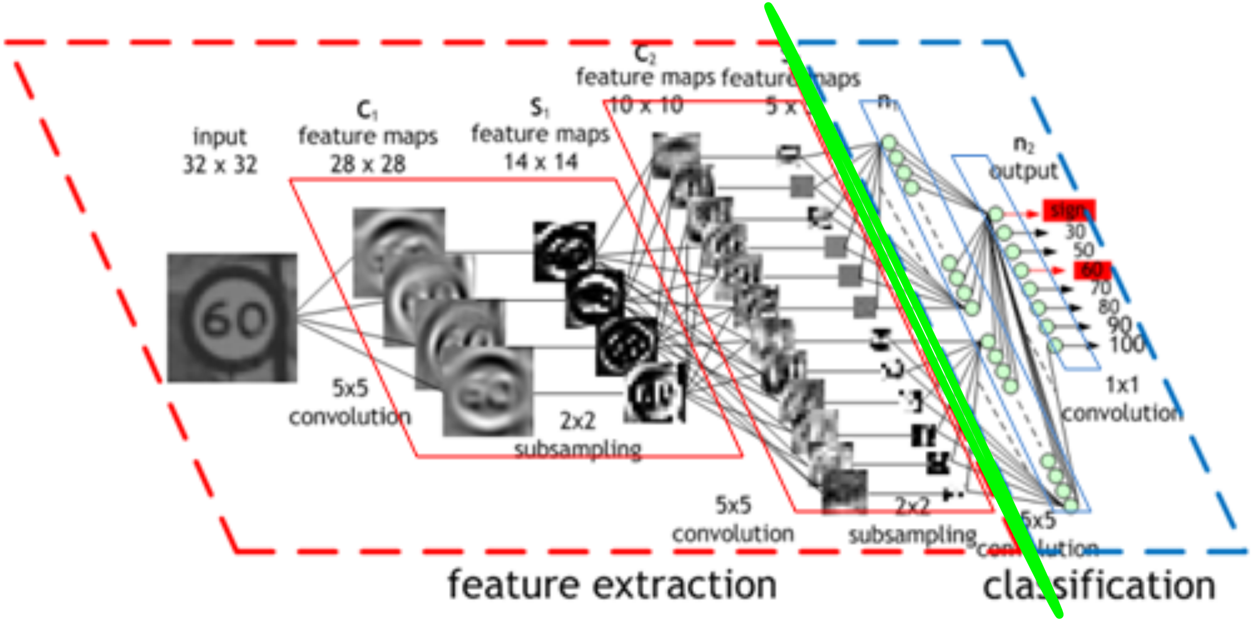




- Pooling layer creates low-resolution versions of the images that reach it
- It forces the next layer to focus on extracting more global characteristics
- It also adds robustness against image translations

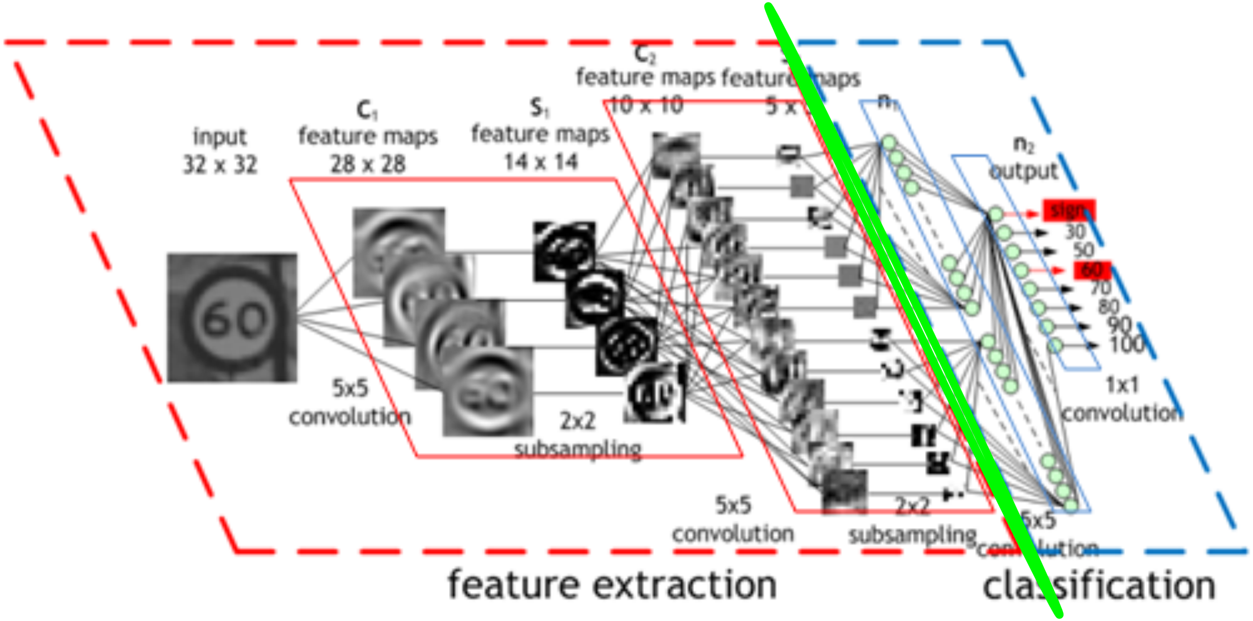
Flattening layer

Flattening layer



The flattening layer (sometimes not represented as a separate layer) converts a set of images into a single vector

Flattening layer



It is the **transition** between the **feature extraction** stage and the **classification** stage, which operates with dense layers

Flattening layer

Example: if we have these images from a previous layer:

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

5	2	4
2	1	5
3	4	1

3	0	6
4	2	0
2	1	9

The flattening layer would transform them into a single vector:

-1	0	3	2	4	-1	-1	2	1	5	2	4	2	1	5	3	4	1	3	0	6	4	2	0	2	1	9

The idea is that from that moment on, the processing will be performed by dense layers (typical layers of shallow networks)

Flattening layer

Flattening can also be performed by averaging each image:

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

5	2	4
2	1	5
3	4	1

3	0	6
4	2	0
2	1	9

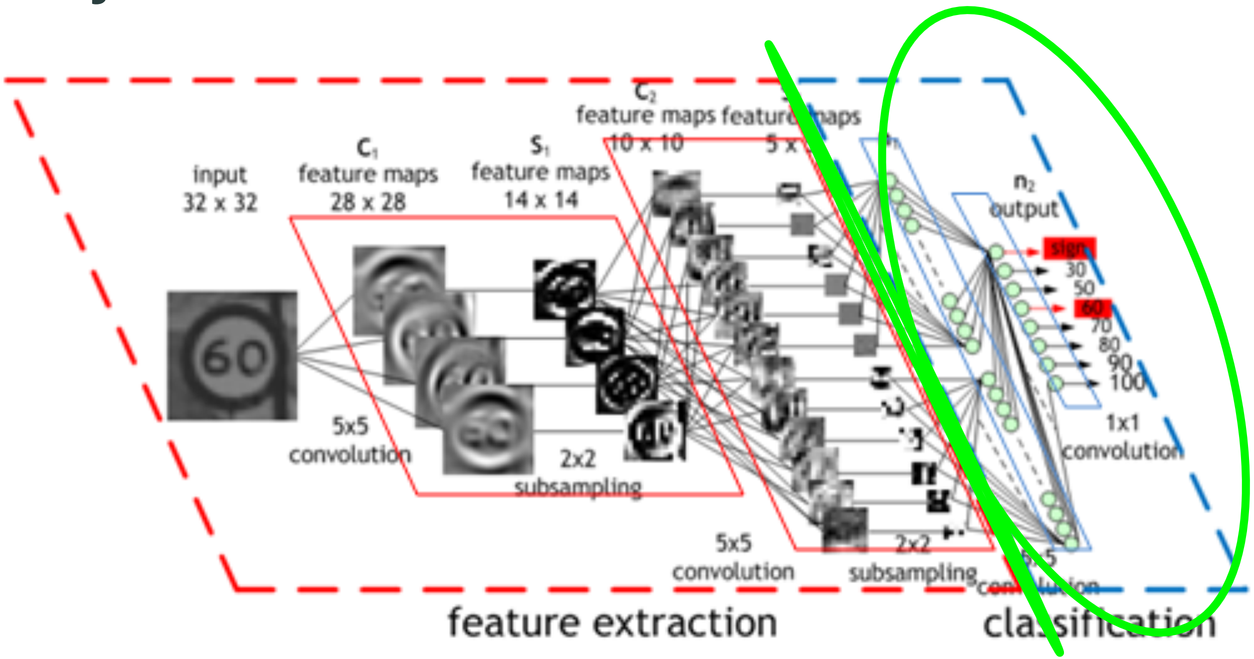
In this case the flattening layer would transform them into:

1	3	3

In this case the simplification is greater but relevant information may be lost

Dense layer

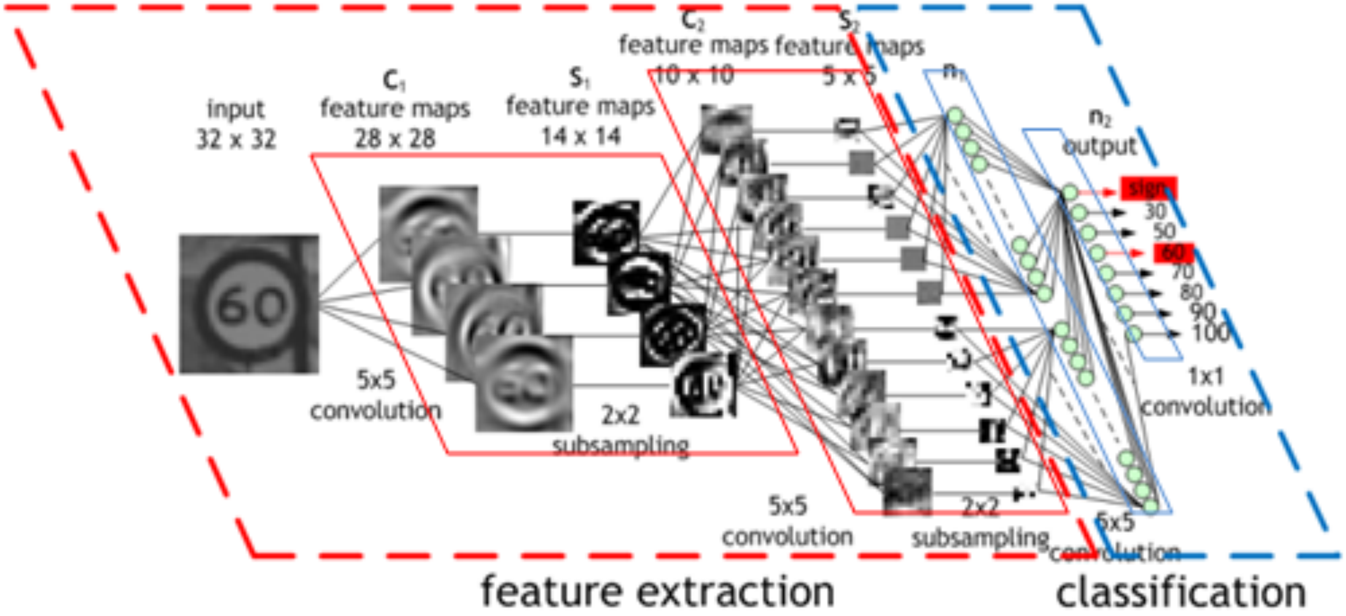
Dense layers



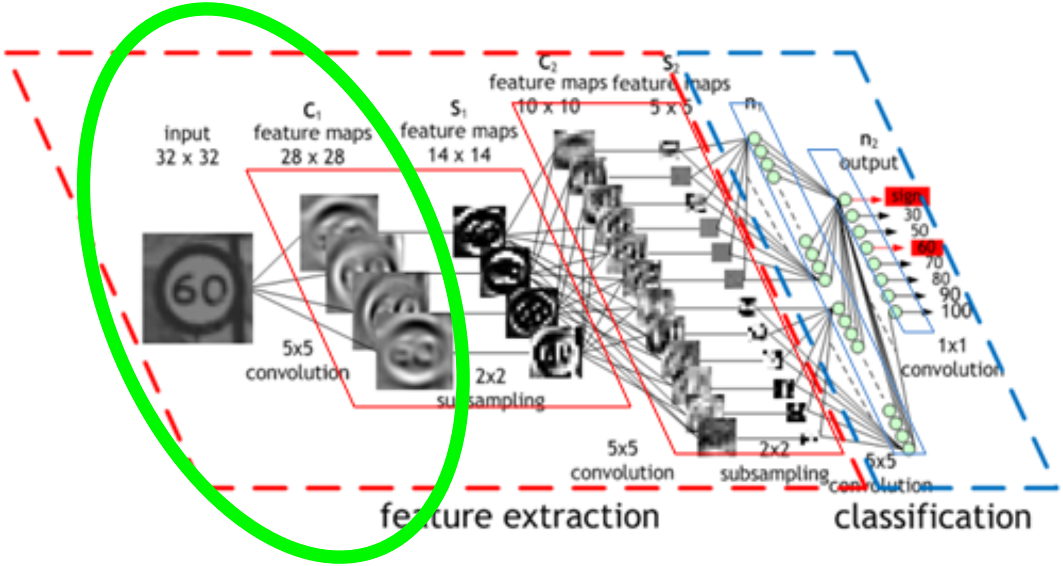
- These are the "typical" layers of non-deep (shallow) neural networks
- They take as input a vector and return a vector
- Each neuron processes all the outputs of the previous layer: many connections!

Creation of Convolutional Neural Networks (CNNs) in Keras

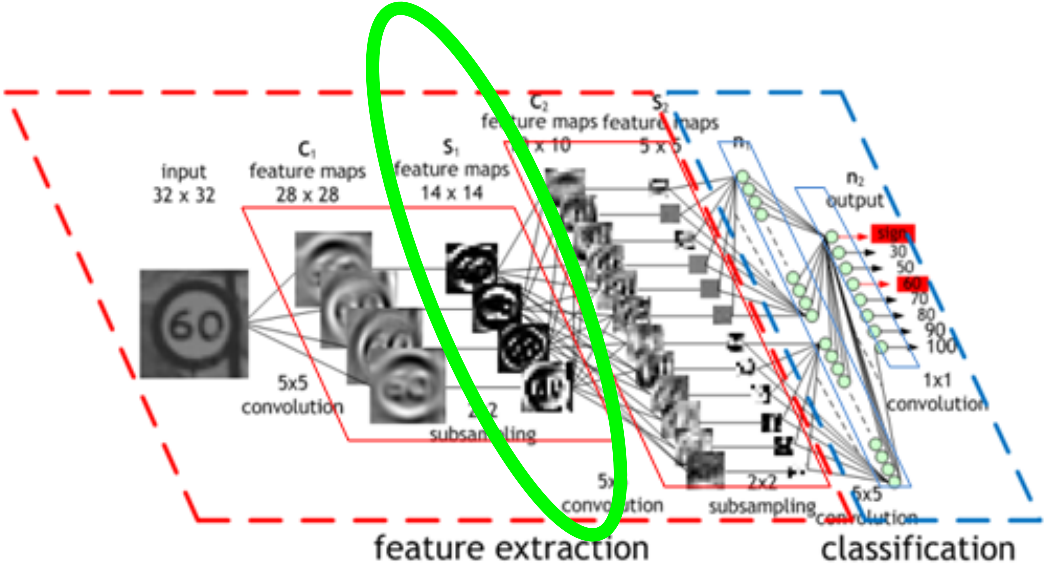
Deep Convolutional Neural Network (CNN)



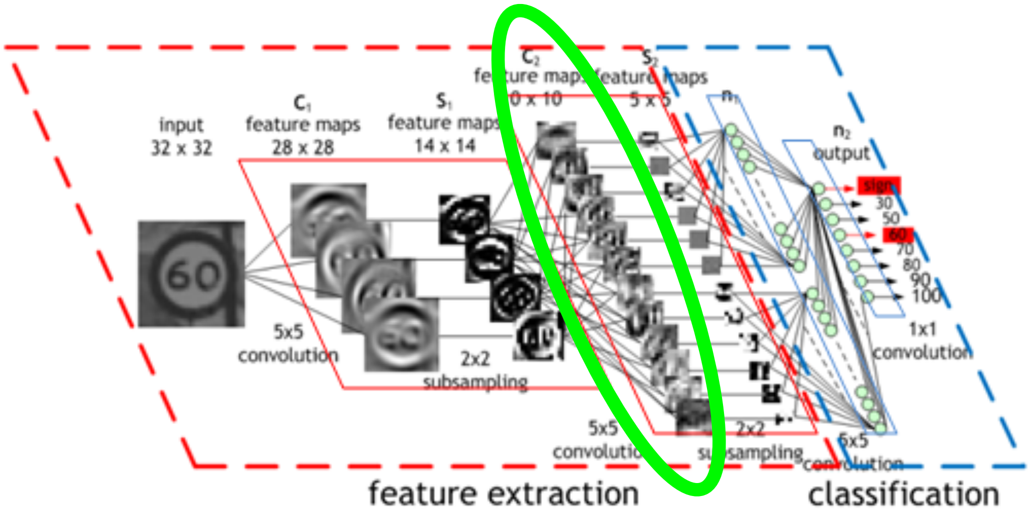
https://developer.nvidia.com/discover/convolutional-neural-network
Imagen: Maurice Peemen



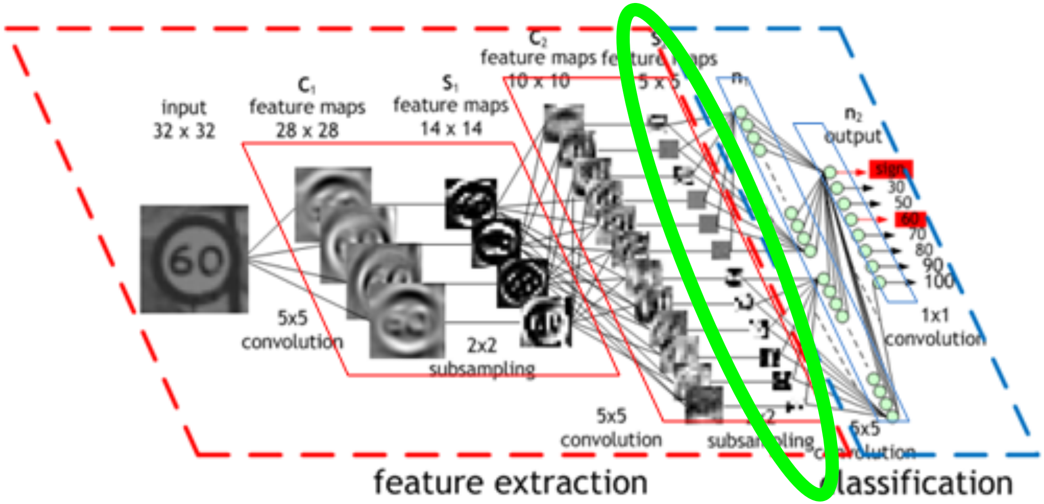

```
model = Sequential()
model.add(Conv2D(filters=4, input_shape=(32,32,1,),
          kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=10,kernel_size=(5,5),activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(8, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```



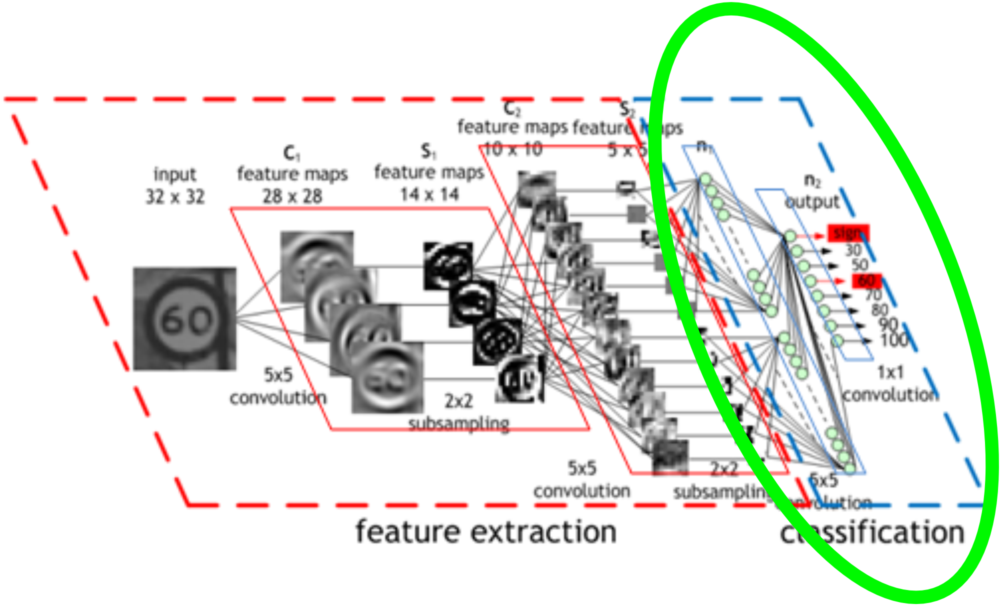

```
model = Sequential()
model.add(Conv2D(filters=4, input_shape=(32,32,1,),
          kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=10,kernel_size=(5,5),activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(8, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```




```
model = Sequential()
model.add(Conv2D(filters=4, input_shape=(32,32,1,),
          kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=10,kernel_size=(5,5),activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(8, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```




```
model = Sequential()
model.add(Conv2D(filters=4, input_shape=(32,32,1,),
          kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=10,kernel_size=(5,5),activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(8, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```




```
model = Sequential()
model.add(Conv2D(filters=4, input_shape=(32,32,1,),
          kernel_size=(5,5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=10,kernel_size=(5,5),activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(8, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

CNN: training

How does a neural network learn?

Basic algorithm

A subset (batch) of training examples is taken, and for each weight the following is checked:

How does a neural network learn?

Basic algorithm

A subset (batch) of training examples is taken, and for each weight the following is checked:

If we increase the weight a little, what would happen to the objective function (cost)?

It increases: positive sensitivity

It decreases: negative sensitivity

How does a neural network learn?

Basic algorithm

A subset (batch) of training examples is taken, and for each weight the following is checked:

↓

If we increase the weight a little, what would happen to the objective function (cost)?

It increases: **positive sensitivity**

It decreases: **negative sensitivity**

↓

Each weight is updated against sensitivity

How does a neural network learn?

Basic algorithm

A subset (batch) of training examples is taken, and for each weight the following is checked:

If we increase the weight a little, what would happen to the objective function (cost)?

It increases: **positive sensitivity**

It decreases: **negative sensitivity**

Each weight is updated against sensitivity

Complete algorithm

1. Divide the training set into parts of the same size: "batches"

2. Apply the basic algorithm once for each of the batches ("epoch")

3. Return to step 1 if stop criteria are not met

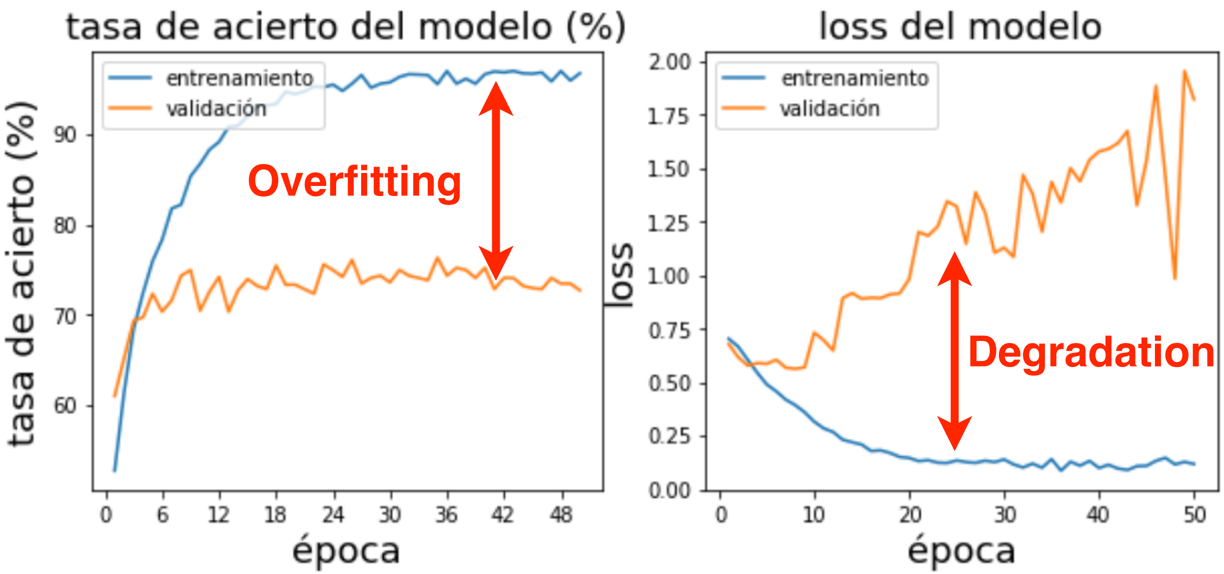
Other aspects to take into account

Data normalization

- Important: inputs to the model must be normalized, they should not exceed the interval [-1, 1]
- If the model is a regression model, the target should also be normalized and should not exceed the interval [-1, 1]

Training monitoring

Training monitoring gives us a lot of information



Techniques for controlling overfitting

Overfitting in CNNs

- Neural network "memorizes" training data, generalizes poorly
- This is because it has too many parameters for the volume of training data

Techniques to avoid overfitting

- Minimize network complexity
- Regularization of weights
- Monitoring of overfitting and early stopping
- Data Augmentation
- Dropout
- Transfer Learning

Techniques to avoid overfitting

- **Minimize network complexity**
    - Start with simple networks, with few parameters: few filters in convolutional layers, few neurons in dense layers, etc.
    - The "bottleneck" (large number of connections) is usually between the flattening and the first dense layer: try to minimize the size of the flattening

Techniques to avoid overfitting

-Regularization

- -The idea is to reward many weights close to or equal to zero ("pruning")
- -Typical mechanisms: introduction of regularization L1, L2 or a mixture of the two in each layer where "pruning" is desired.
- -The regularization factor must be adjusted (neither too large nor too small)
- -L1 is more aggressive than L2

https://playground.tensorflow.org/

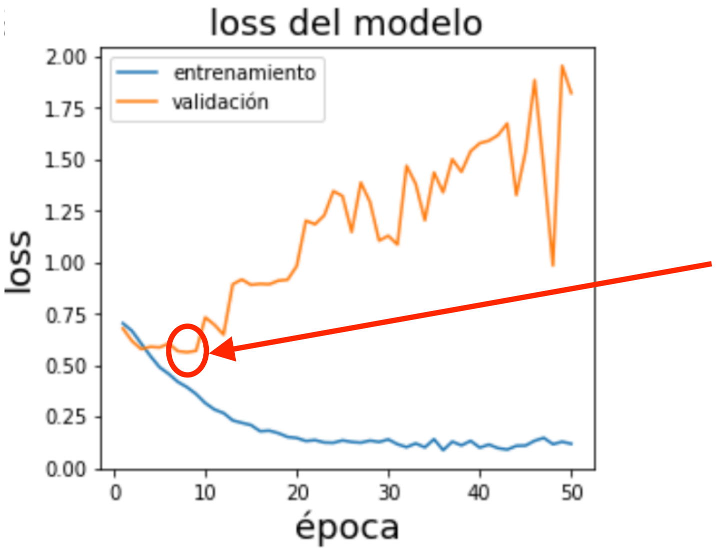
Techniques to avoid overfitting

-Regularization hiperparameters

- Regularization type (L1, L2, mixed, no regularization)
- Regularization strength
- In which layers to apply it

Techniques to avoid overfitting

- **Monitoring of overfitting and early stopping**



Idea: stop training when the error in validation stagnates or begins to increase

Another strategy: let the training run but save to file the network if in validation it improves. At the end of the training, load the network from file

Techniques to avoid overfitting

-**Data Augmentation**

-The idea is to create variants of the available data by means of manipulations

-Images: rotations, translations, zooms, changes in contrast, brightness etc.

Techniques to avoid overfitting

-Data Augmentation

Original image

Techniques to avoid overfitting

-**Data Augmentation**

-Audio: shorten, lengthen, change pitch, introduce noise, etc.

Techniques to avoid overfitting

- **Data Augmentation: hyperparameters**
    - Types of transformations to be applied
    - Magnitude of these transformations

Techniques to avoid overfitting

-**Dropout**

- -The idea is to introduce noise at those points in the network where there is an excess of information.
- -In this way we force the network to focus not on small details but on global properties.

Techniques to avoid overfitting

-Dropout

No dropout

rate=0.1

rate=0.2

rate=0.5

rate=0.7

Techniques to avoid overfitting

- Dropout: hyperparameters

- Dropout quantity (rate): minimum: 0 (no dropout); maximum: 1

- Parts of the network where this effect can be introduced

- It is usually introduced before layers that process large amounts of data

Techniques to avoid overfitting

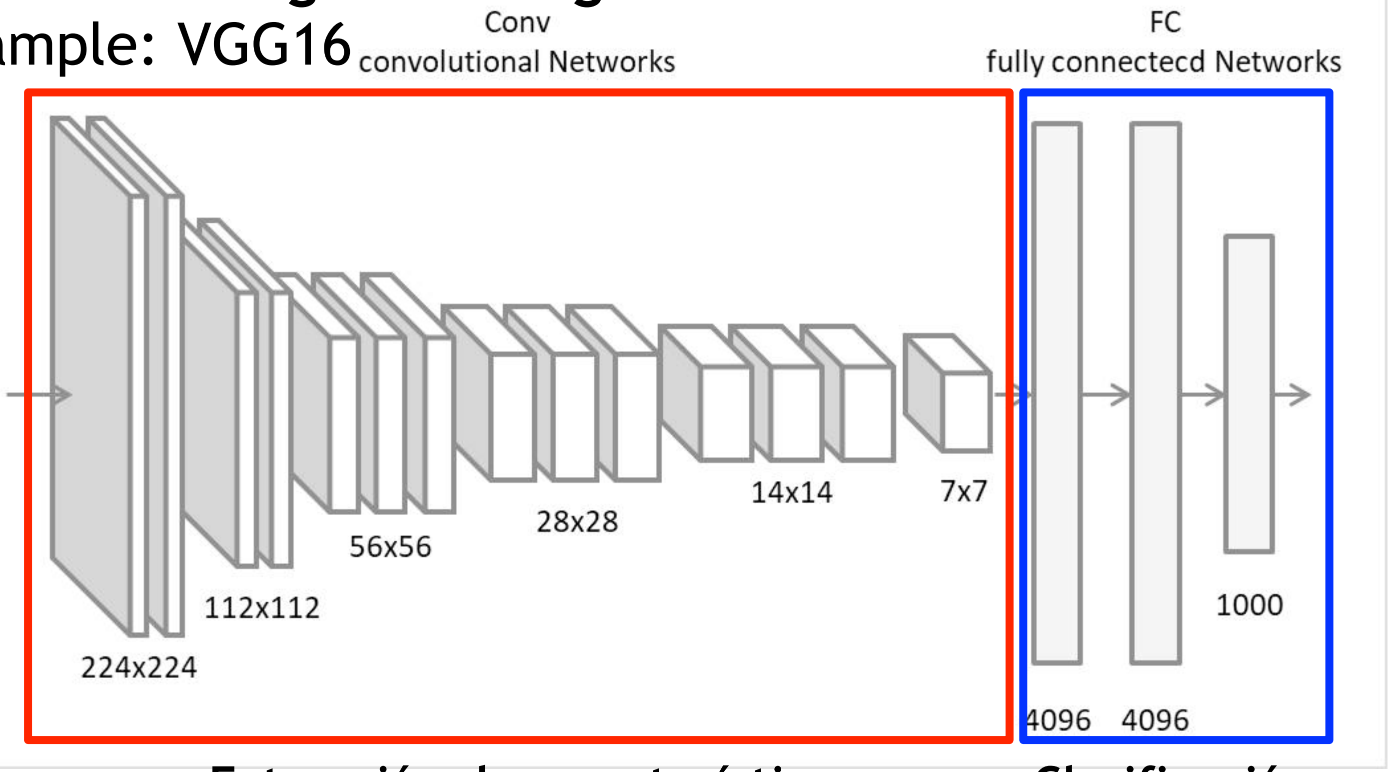
-**Transfer Learning:** take another system trained on another dataset and use parts of it in the network.

-Images: typically, we download a network trained on a similar domain, keep the feature extraction part and add our classification layers

Techniques to avoid overfitting

-Transfer Learning with images

-Example: VGG16



Extracción de características **Clasificación**

Techniques to avoid overfitting

- Some pre-trained CNNs that can be found on the Internet:

- Xception
- VGG16, VGG19
- ResNet, ResNetV2
- InceptionV3
- InceptionResNetV2
- MobileNet
- MobileNetV2
- DenseNet
- NASNet

Techniques to avoid overfitting

-**Transfer Learning**

- In texts: e.g., download embeddings (word representations) trained in other similar domains.

- For example:

  -word2vec (implemented in Python Gensim library)
  -GloVe: https://nlp.stanford.edu/projects/glove/

Interpretability in CNNs

Heatmaps: first idea

How to calculate a heatmap? (sensitivity to image zones)
One option is to analyze how the prediction changes as individual pixels change

CNN

How does altering that pixel change the prediction?

Heatmaps: first idea

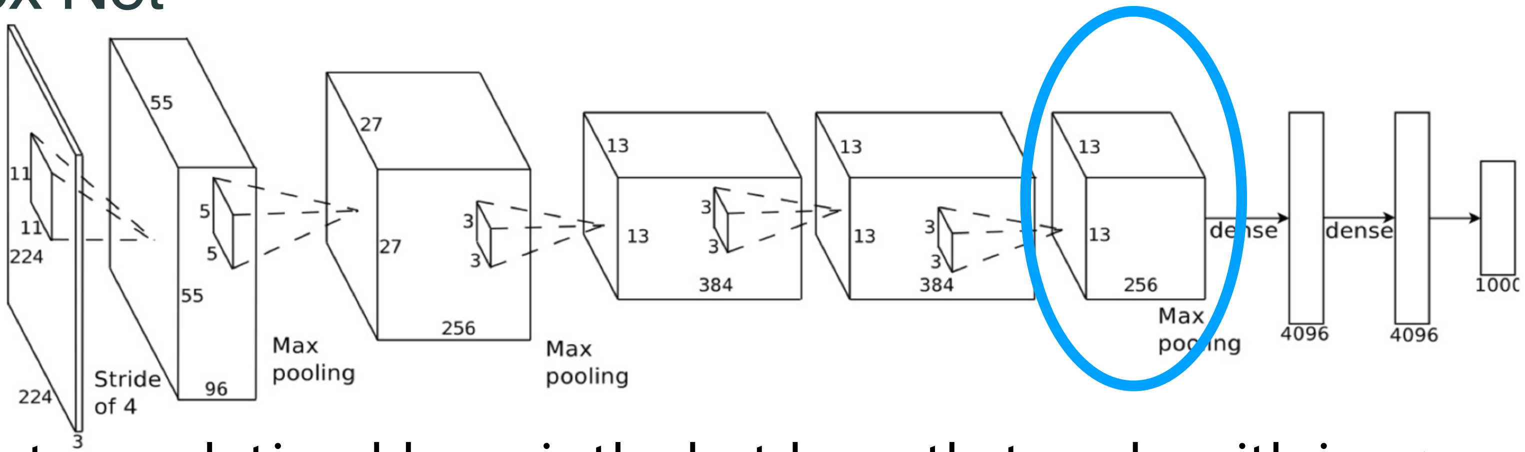
How to calculate a heatmap? (sensitivity to image zones)
One option is to analyze how the prediction changes as individual pixels change

Problem:

It makes no sense to change individual pixels in the image, it is not natural

Heatmaps: second idea

Alex Net



- The last convolutional layer is the last layer that works with images
- In AlexNet this layer extracts 256 images of 13x13 pixels. That is, 256 filtered versions of the original image (224x224 pixels)
- Each of these 13x13 pixels represents information extracted from at least 224/13x224/13 = 17x17 input pixels

Heatmaps: Gradcam

1- Passing the image over the network

2- Finding the most active output neuron N (most likely class)

3- Calculate how the output of N changes if there are small changes in the different output pixels of the last convolutional layer C

3- Averaging and weighing with the output pixels of C

4- Normalize and draw

Heatmaps: Gradcam

Class: CAT

Heatmaps: Gradcam

Class: DOG

Class: CHILD

Heatmaps: applications

Heatmaps: Detection of biases in the dataset

Predicted class: DOG

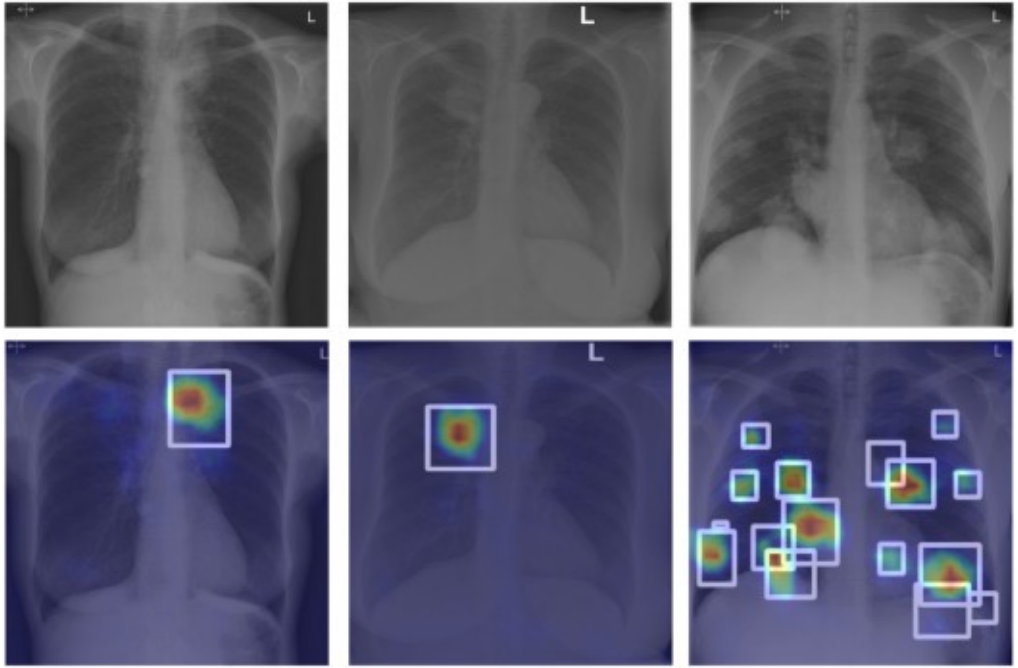
Predicted class: DOG

The network has learned "if there is weed -> DOG" !!!

Utility in medical decision support systems

Objective: TO ASSIST (not replace) medical personnel

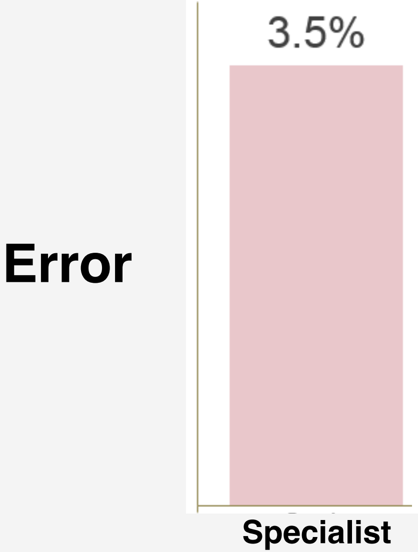
Example: Lung pathology detection



Utility in medical decision support systems

Objective: TO ASSIST (not replace) medical personnel

Example: Breast cancer screening



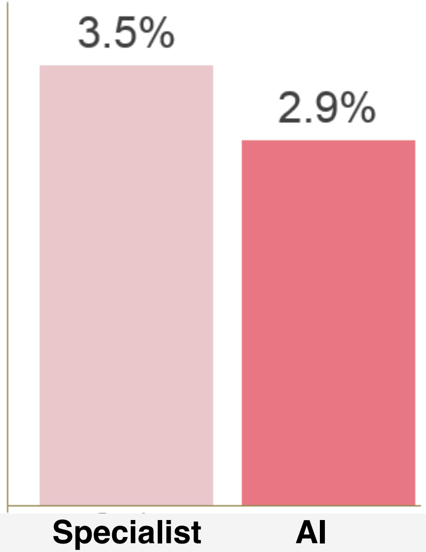
- Staff from Harvard Medical School's Beth Israel Deaconess Medical Center (BIDMC)

Utility in medical decision support systems

Objective: TO ASSIST (not replace) medical personnel

Example: Breast cancer screening

Error



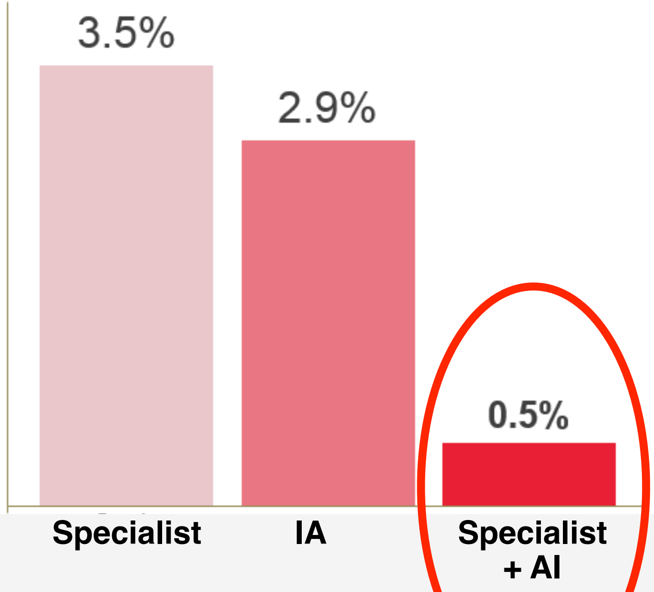
- Staff from Harvard Medical School's Beth Israel Deaconess Medical Center (BIDMC)
- The neural network was trained with millions of labeled images
- The network assigns to each part of the image the probability that it contains evidence of cancer

Utility in medical decision support systems

Objective: TO ASSIST (not replace) medical personnel

Example: Breast cancer screening

Error



- Staff from Harvard Medical School's Beth Israel Deaconess Medical Center (BIDMC)
- The neural network was trained with millions of labeled images
- The network assigns to each part of the image the probability that it contains evidence of cancer
- Probability maps are created that can be interpreted by medical staff

Deep Learning: Convolutional Neural Networks (CNNs)

Manuel Sánchez-Montañés

Escuela Politécnica Superior, Universidad Autónoma de Madrid