Convolutional Neural Network - CNN

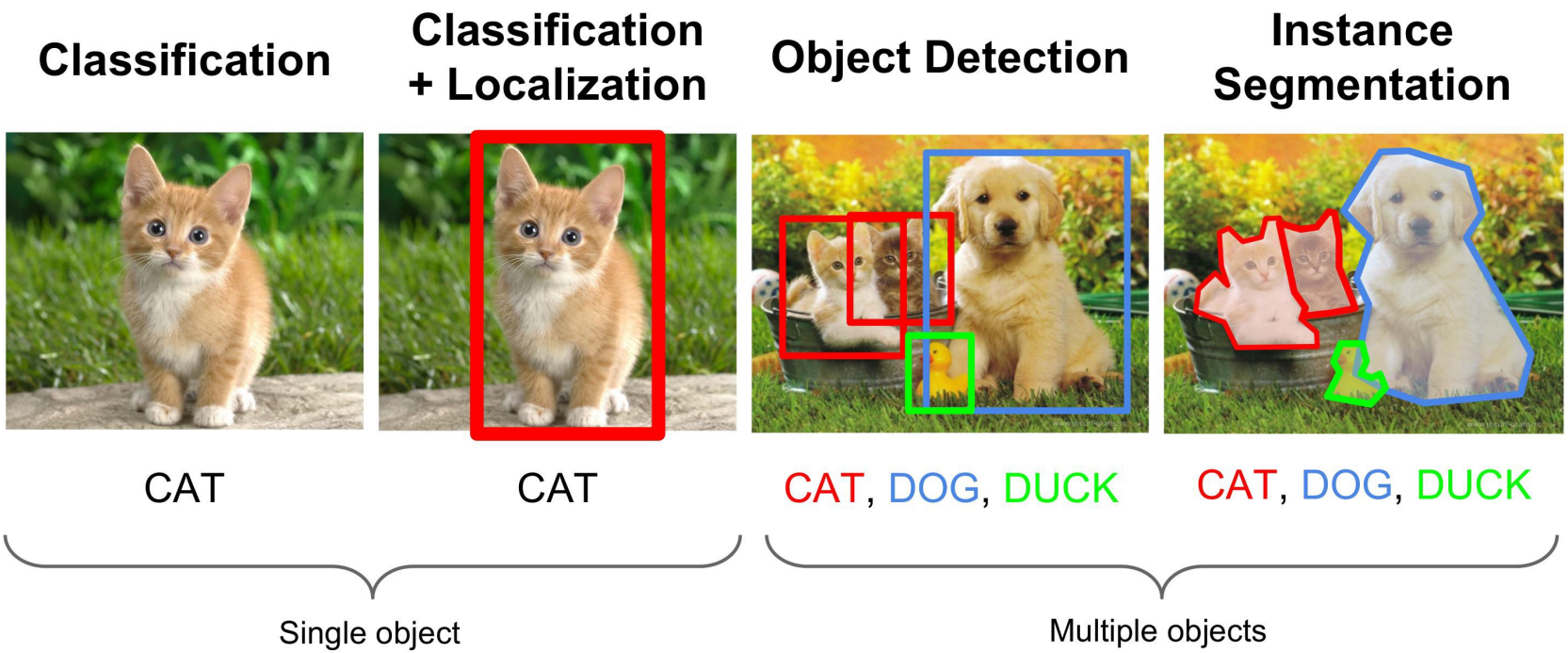
Eduardo Todt,
Bruno Alexandre Krinski

VRI Group - Vision Robotic and Images
Federal University of Paraná

November 30, 2019

Definition

Convolutional Neural Networks (CNNs) are Artificial Intelligence algorithms based on multi-layer neural networks that learns relevant features from images, being capable of performing several tasks like object classification, detection, and segmentation.



History

An initial idea of convolution was proposed by Kunihiko Fukushima in 1980 and initially was called Neocognitron.

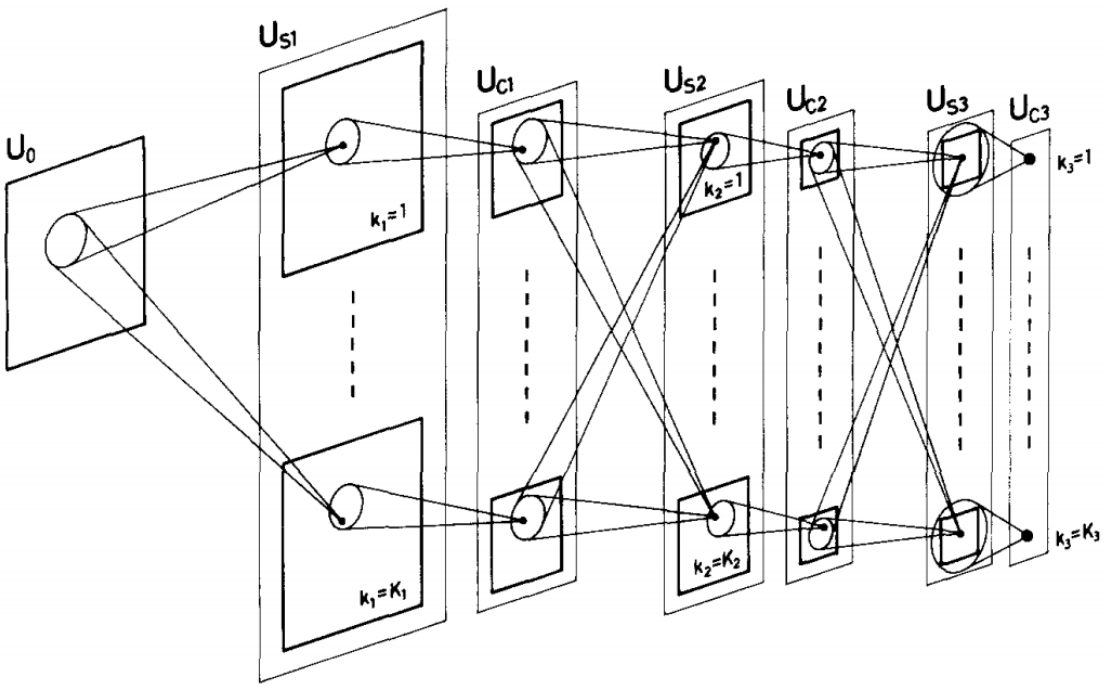


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

History

The modern concept of Convolutional Neural Networks comes from the work of Yann LeCun published in 1998. LeCun proposed a CNN called LeNet for hand-write recognition.

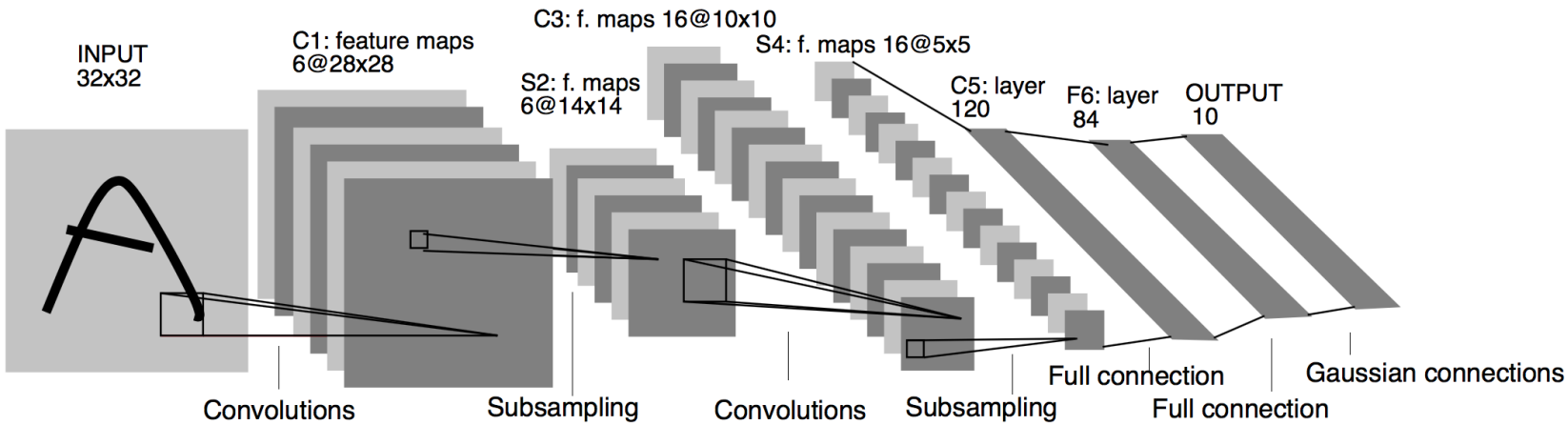


Figure: Gradient-Based Learning Applied to Document Recognition

History

In 2012, Alex Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge with a CNN model called AlexNet. Krizhevsky used GPUs to train the AlexNet, which enabled faster training of CNNs models and started a wave of interest and new works based on CNNs.

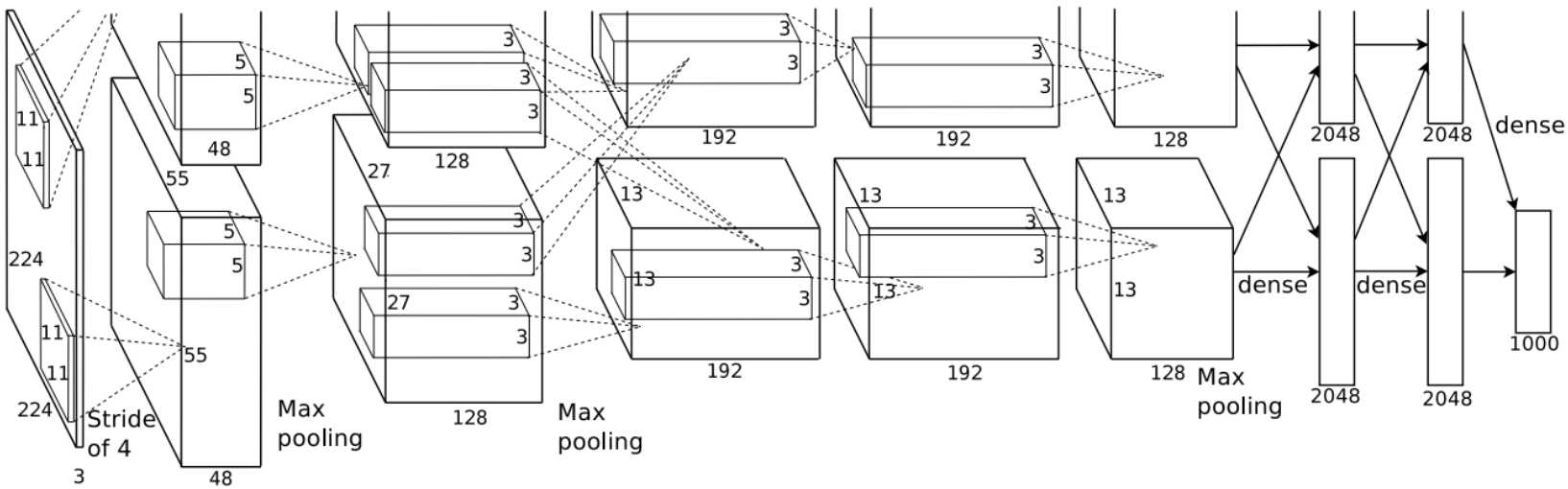


Figure: ImageNet Classification with Deep Convolutional Neural Networks

Why CNN is better?

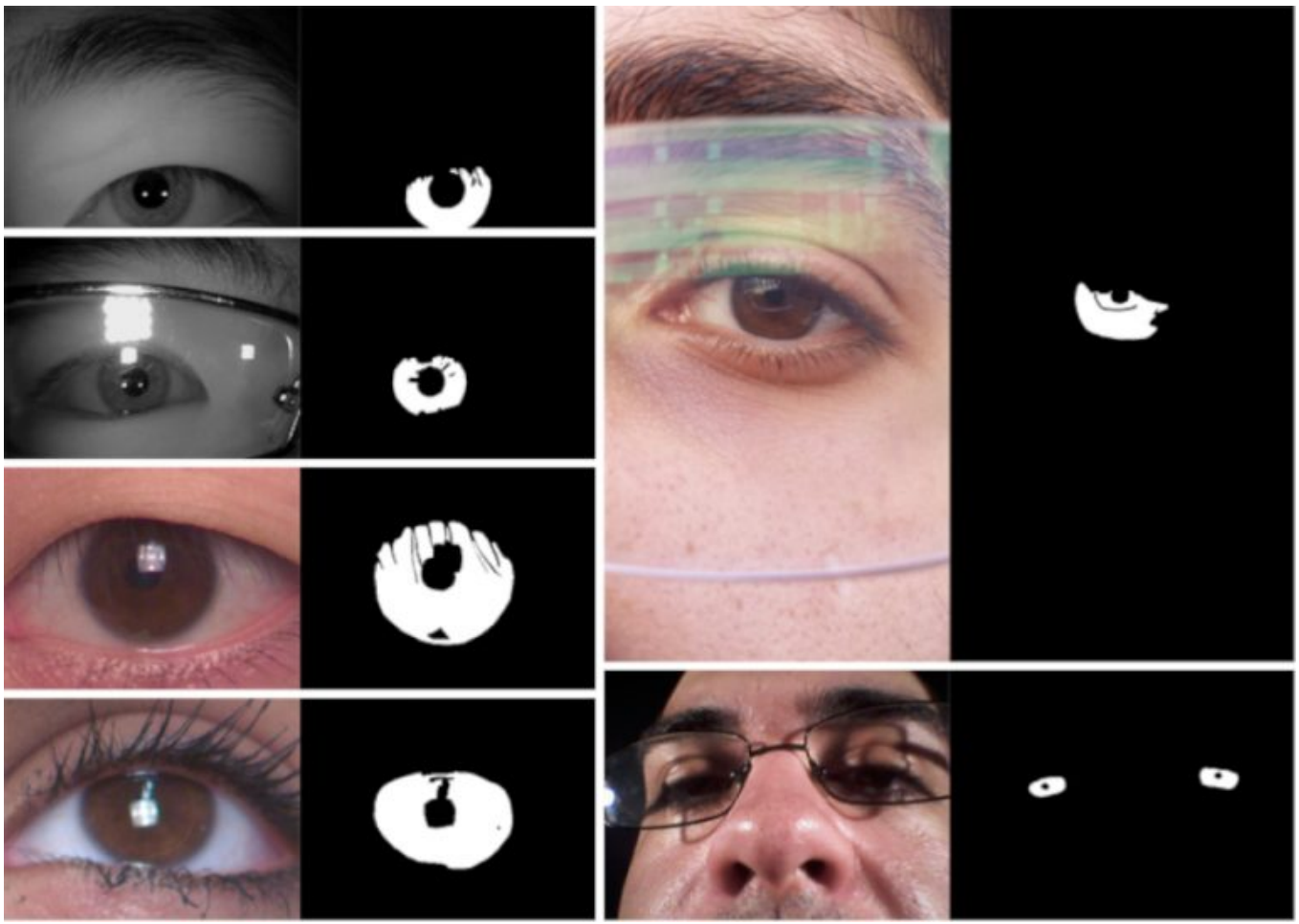
The advantage of CNNs over others classification algorithms (SVM, K-NN, Random-Forest, and others) is that the CNNs learns the best features to represent the objects in the images and has a high generalization capacity, being able to precisely classify new examples with just a few examples in the training set.

Popular applications

Car and Plate recognition

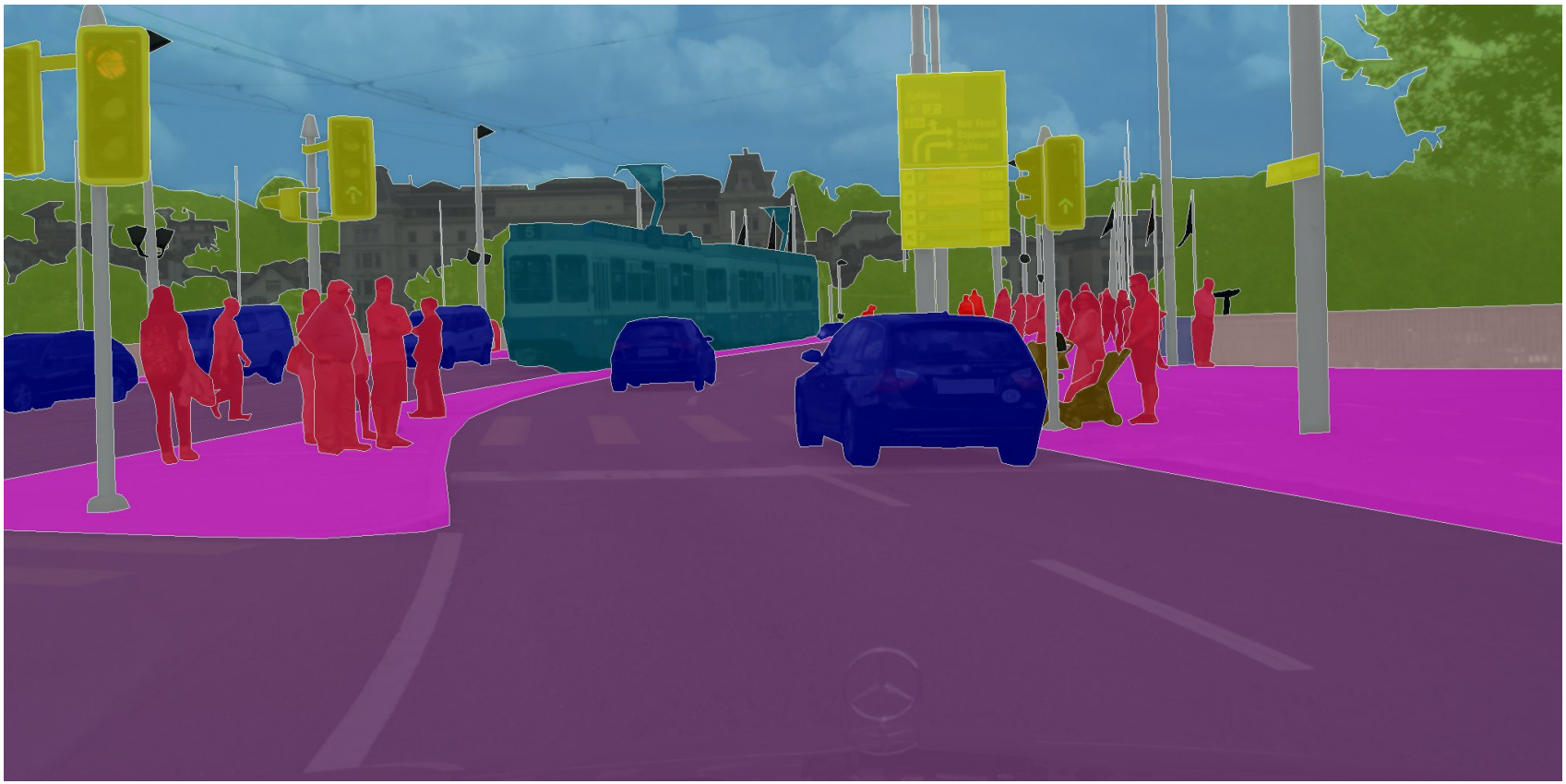
Popular applications

Biometry



Popular applications

Autonomous car

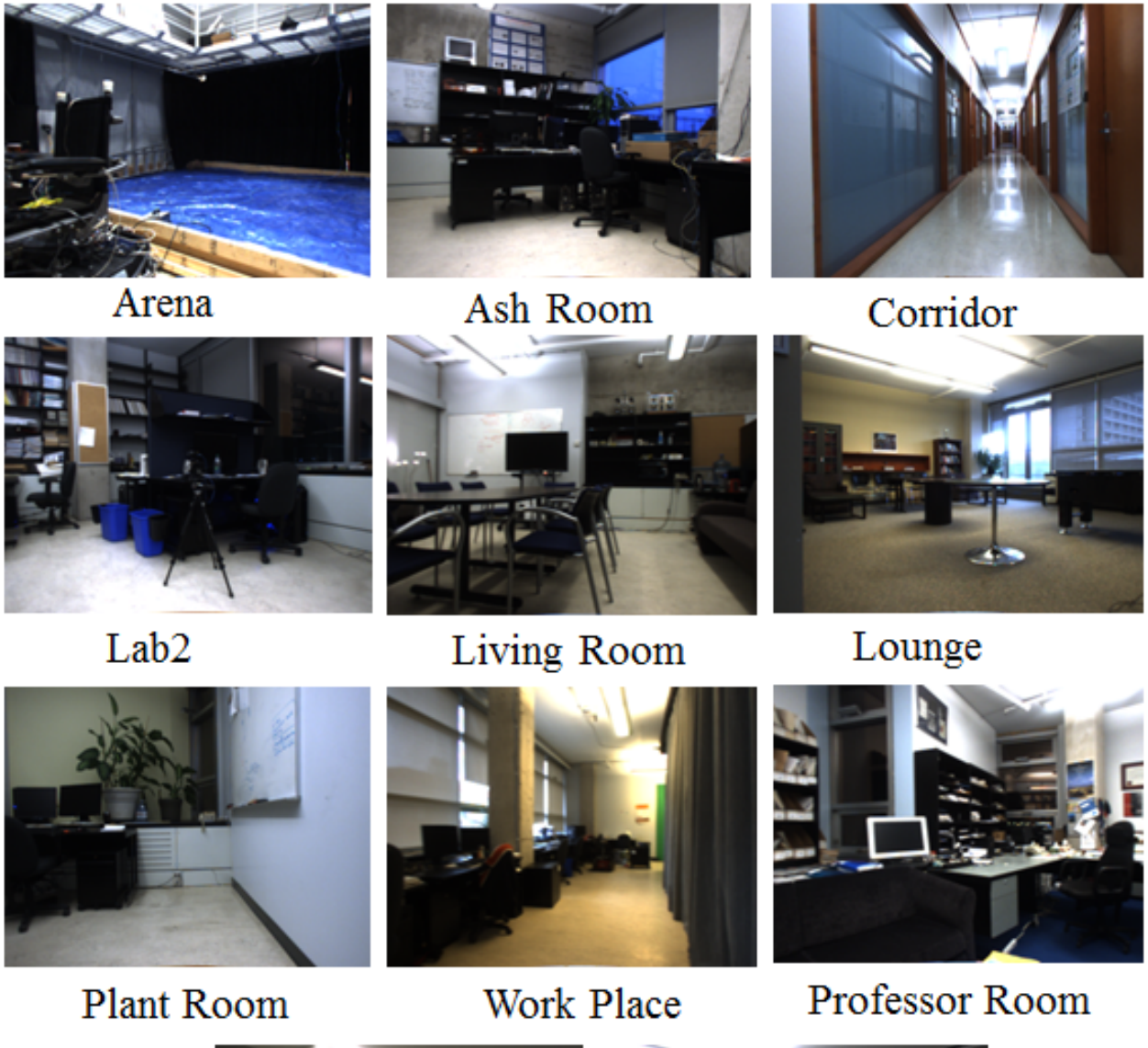


Popular applications

Pedestrian Detection

Popular applications

Indoor environments recognition



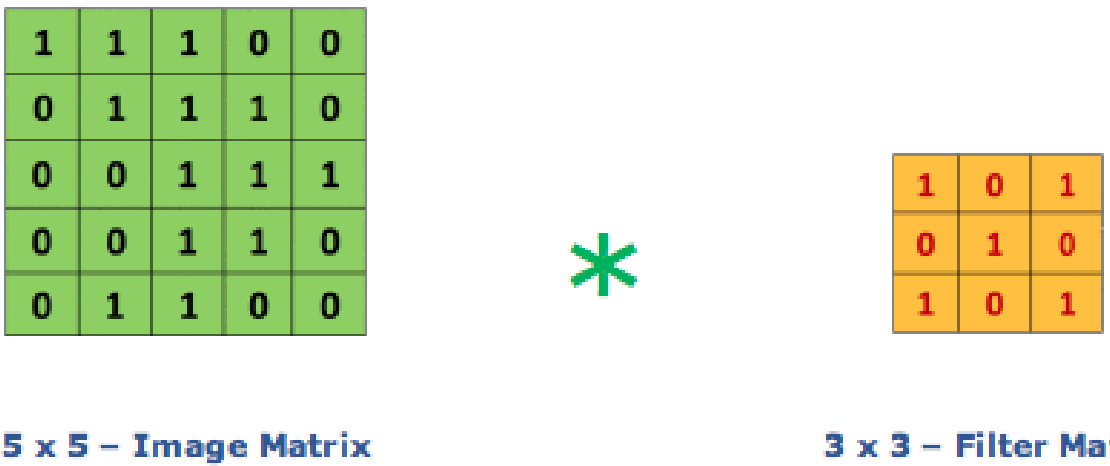
CNN Layers

A CNN is topically composed by four types of layers:

- Convolutional
- Pooling
- Relu
- Fully Connected

Convolutional Layer

- A convolutional layer is composed by a set of filters, also called kernels, that slides over the input data.
- Each kernel has a width, a height and width $\times$ height weights utilized to extract features from the input data.
- In the training step, the weights in the kernel starts with random values, and will be learning based on the training set.



Convolutional Layer

Example of convolution with 1 channel: https://miro.medium.com/max/658/1*GcI7G-JLAQiEoCON7xFbhg.gif

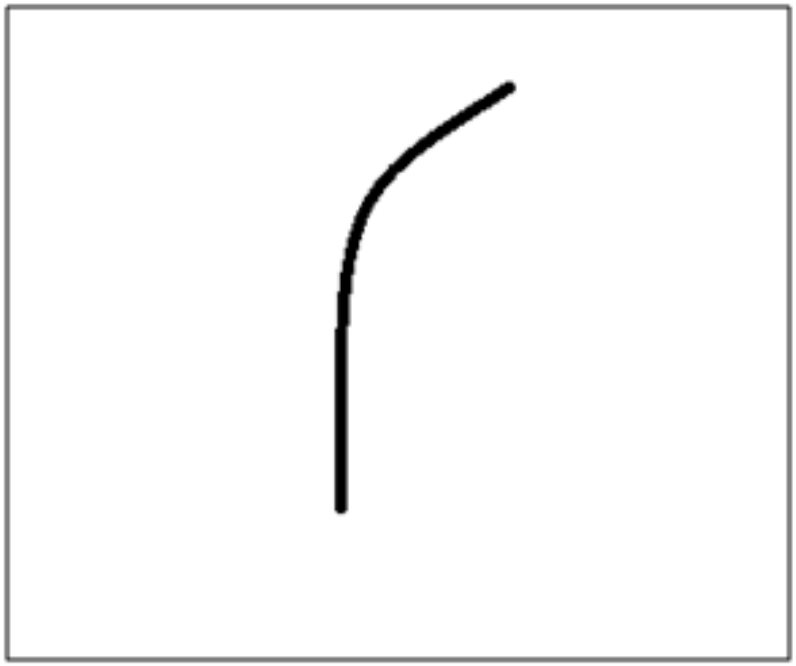
Example of convolution with 3 channels: https://miro.medium.com/max/1600/1*ciDgQEjViWLnCbmX-EeSrA.gif

Convolutional Layer

Each filter in the convolutional layer represents a feature.

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

Pixel representation of filter

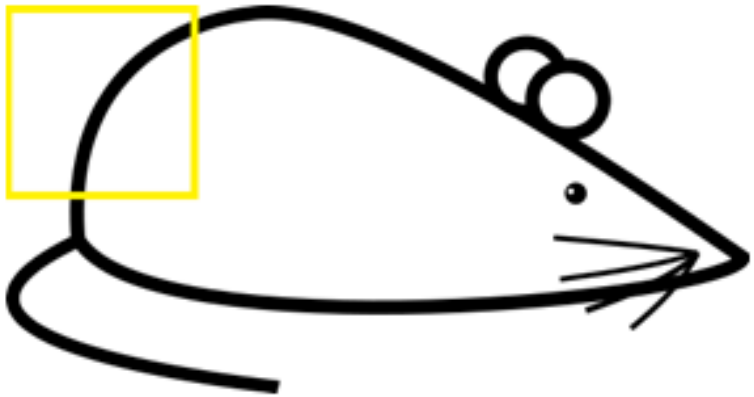


Visualization of a curve detector filter

Convolutional Layer

When the filter slides over the image and finds a match...

Original image



Visualization of the filter on the image

Convolutional Layer

The convolution operation generates a large number, activating the filter to that characteristic.

Visualization of the 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

Pixel representation of the receptive field

*

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

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Convolutional Layer

When the filter slides over the image and finds no match, the filter does not activate.
The CNN uses this process to learn the best filters to describe the objects.

Visualization of the filter on the image

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

Pixel representation of receptive field

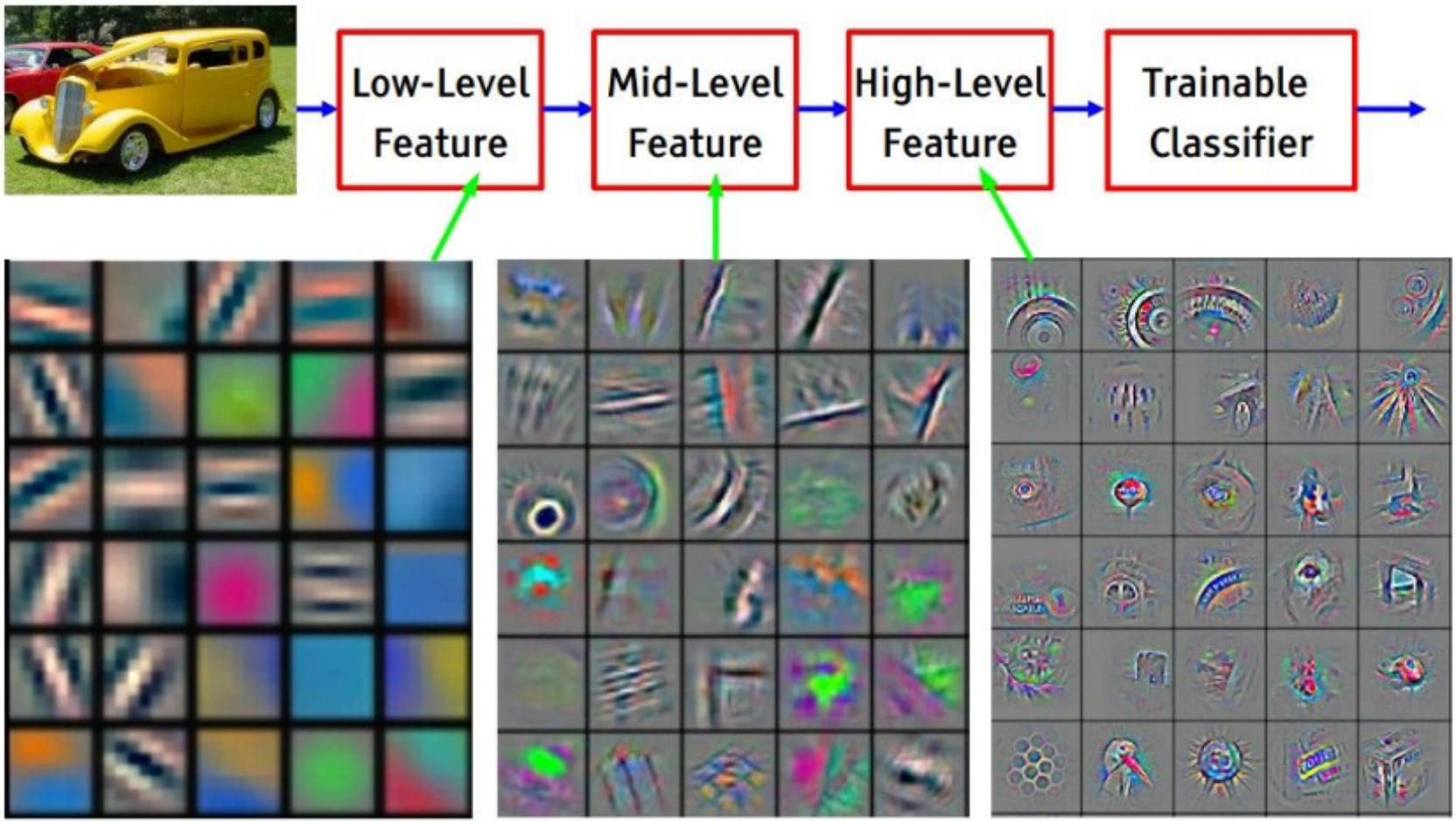
*

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

Pixel representation of filter

Multiplication and Summation = 0

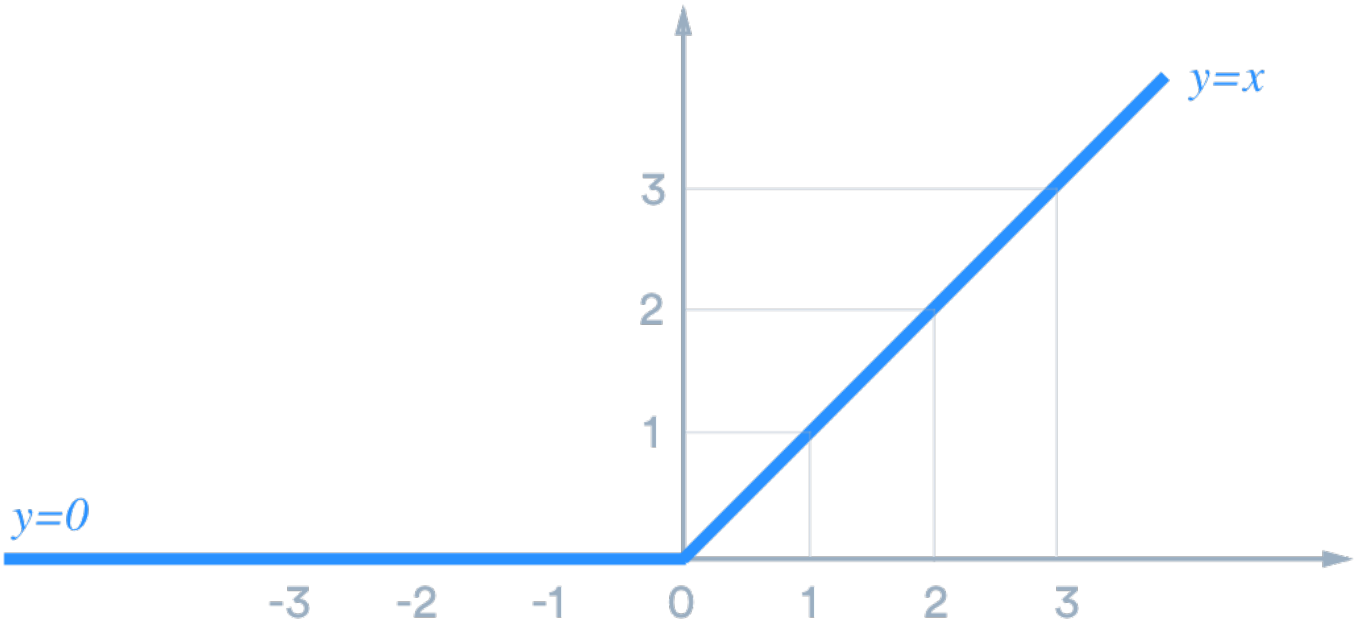
Convolutional Layer



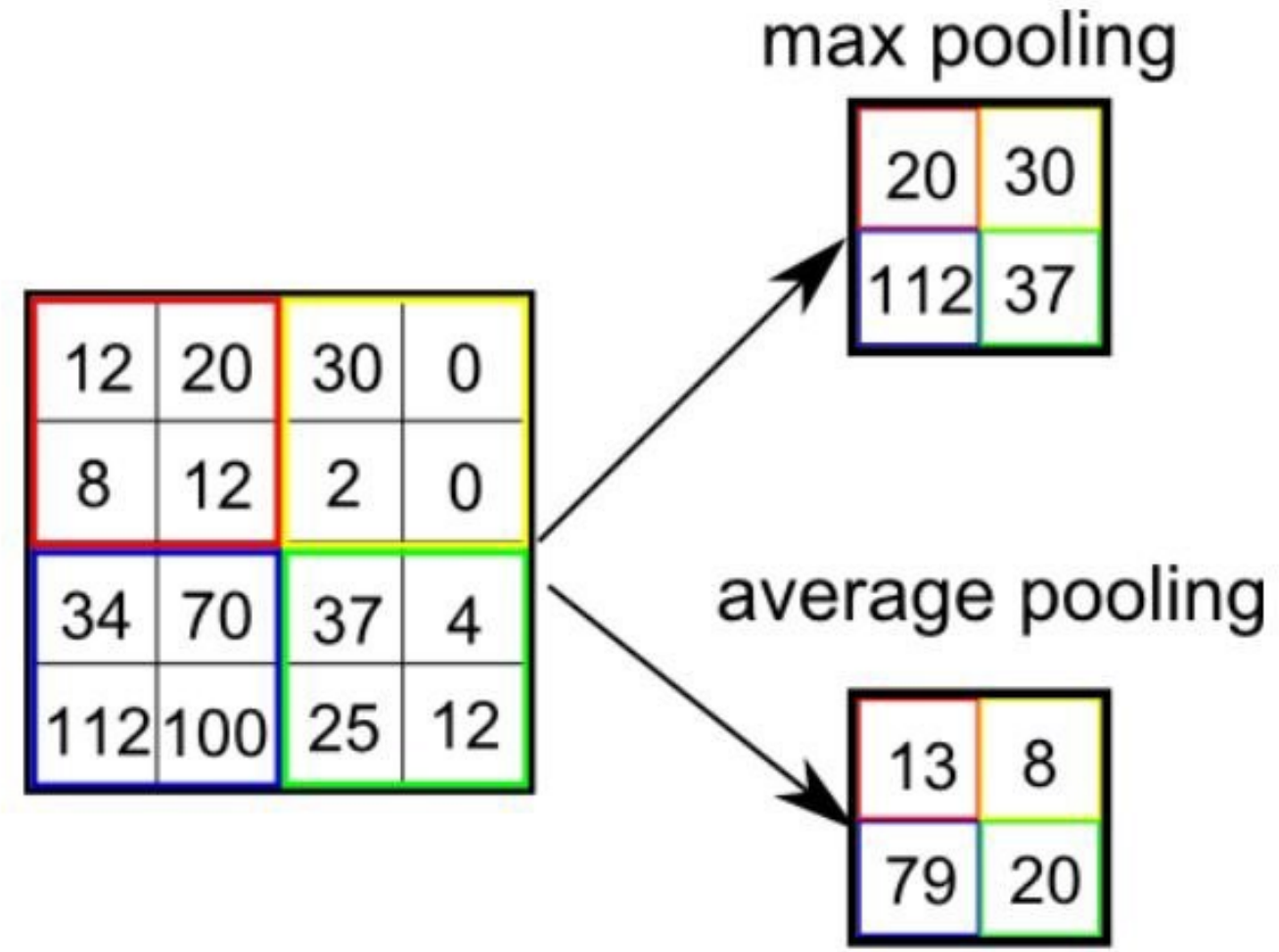
ReLu

- The ReLU (Rectified Linear Units) layers, is an activation layer linked after a convolutional layer to generate non-linearity in the network.
- The ReLu helps the network to learn harder decision functions and reduce the overfitting.
- The ReLu applies the function $y = max(x, 0)$

ReLu



Pooling

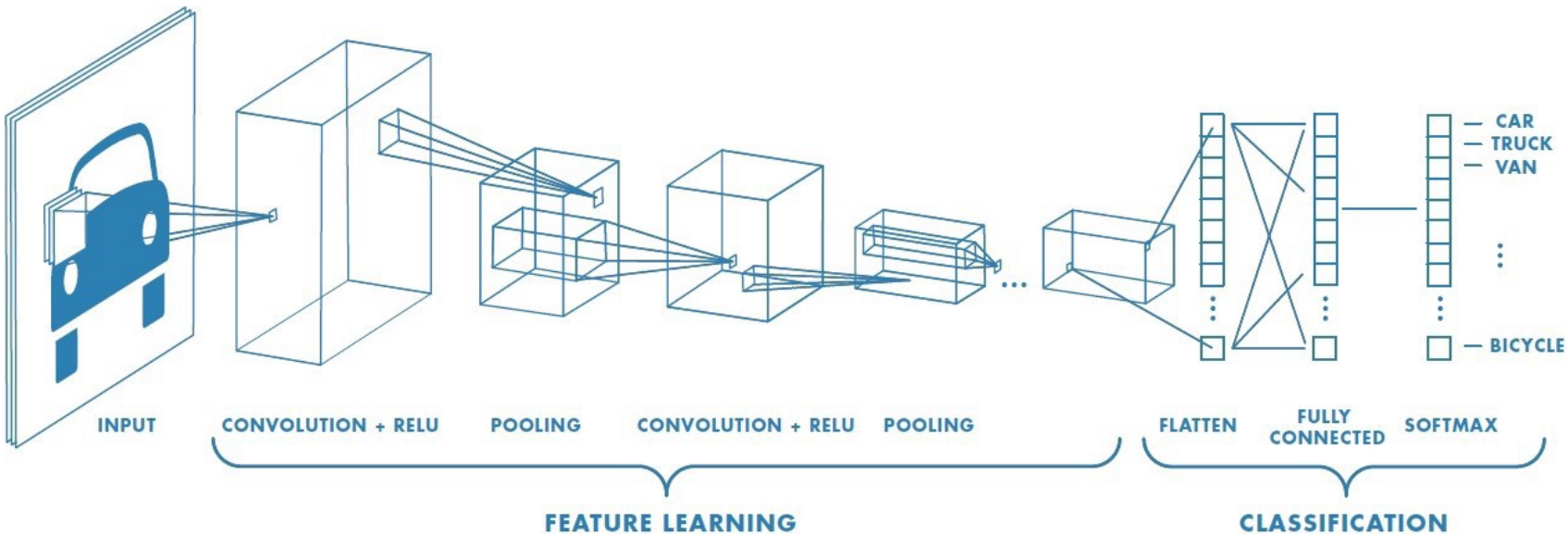

max pooling
20 30
112 37
12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
average pooling
13 8
79 20

Pooling

- The pooling layer, or down-sampling layer, is applied to reduce the dimensionality of the feature maps in a way to save the most relevant information from the feature maps.
- In the pooling layer, a filter slides over the input data and applies the pooling operation (max, min, avg).
- The max pooling is the most used in the literature.
- `https://developers.google.com/machine-learning/practica/image-classification/images/maxpool_animation.gif`

Fully Connected Layer

A CNN is topically divided into two parts: the convolutional and the dense steps. The former learns the best features to extract from the images and the latter learns how to classify the features in different categories.



Fully Connected Layer

The Fully Connected layer is a MultiLayer Perceptron (MLP), composed by three types of layers: input, hidden, and output layers.

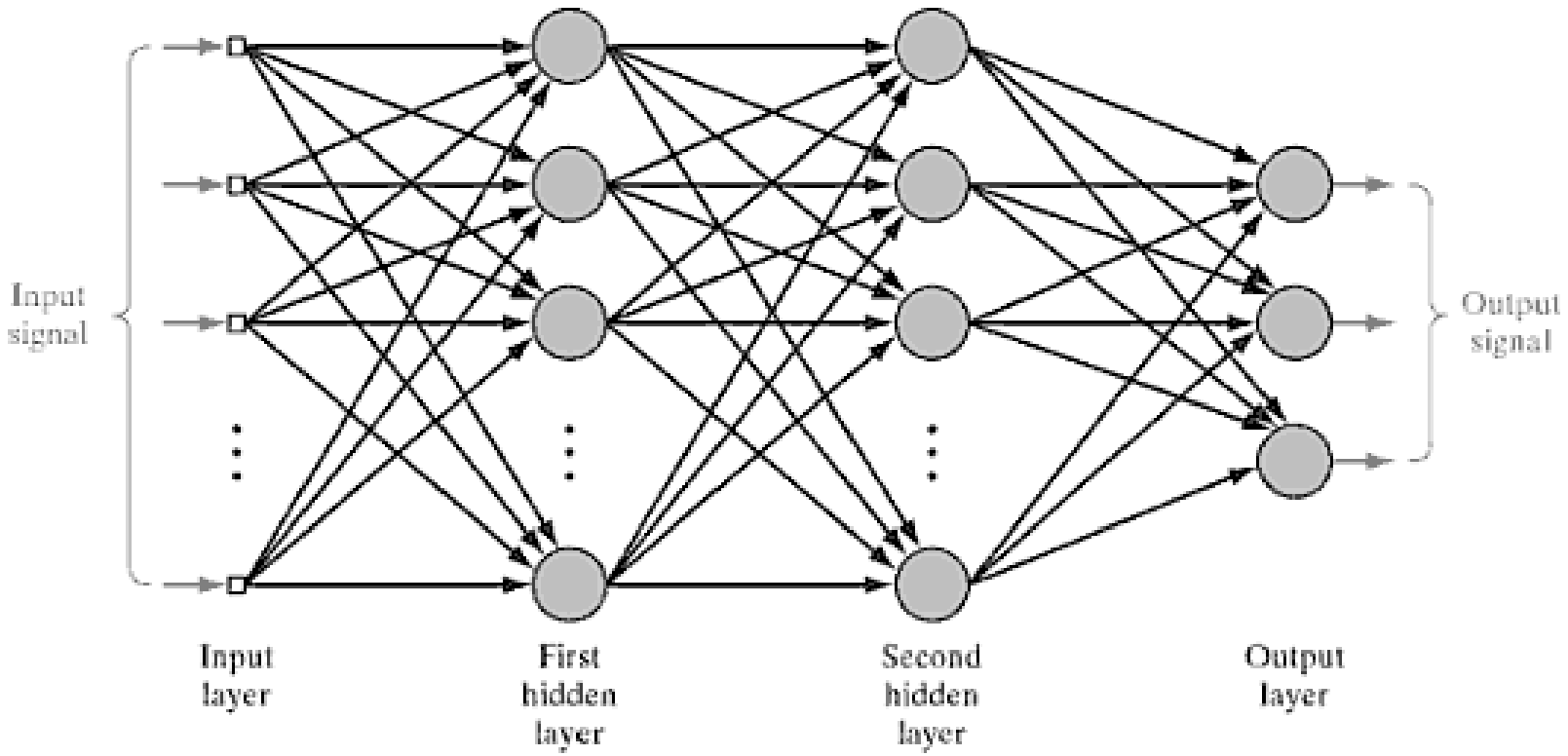


FIGURE 4.1 Architectural graph of a multilayer perceptron with two hidden layers.

Fully Connected Layer

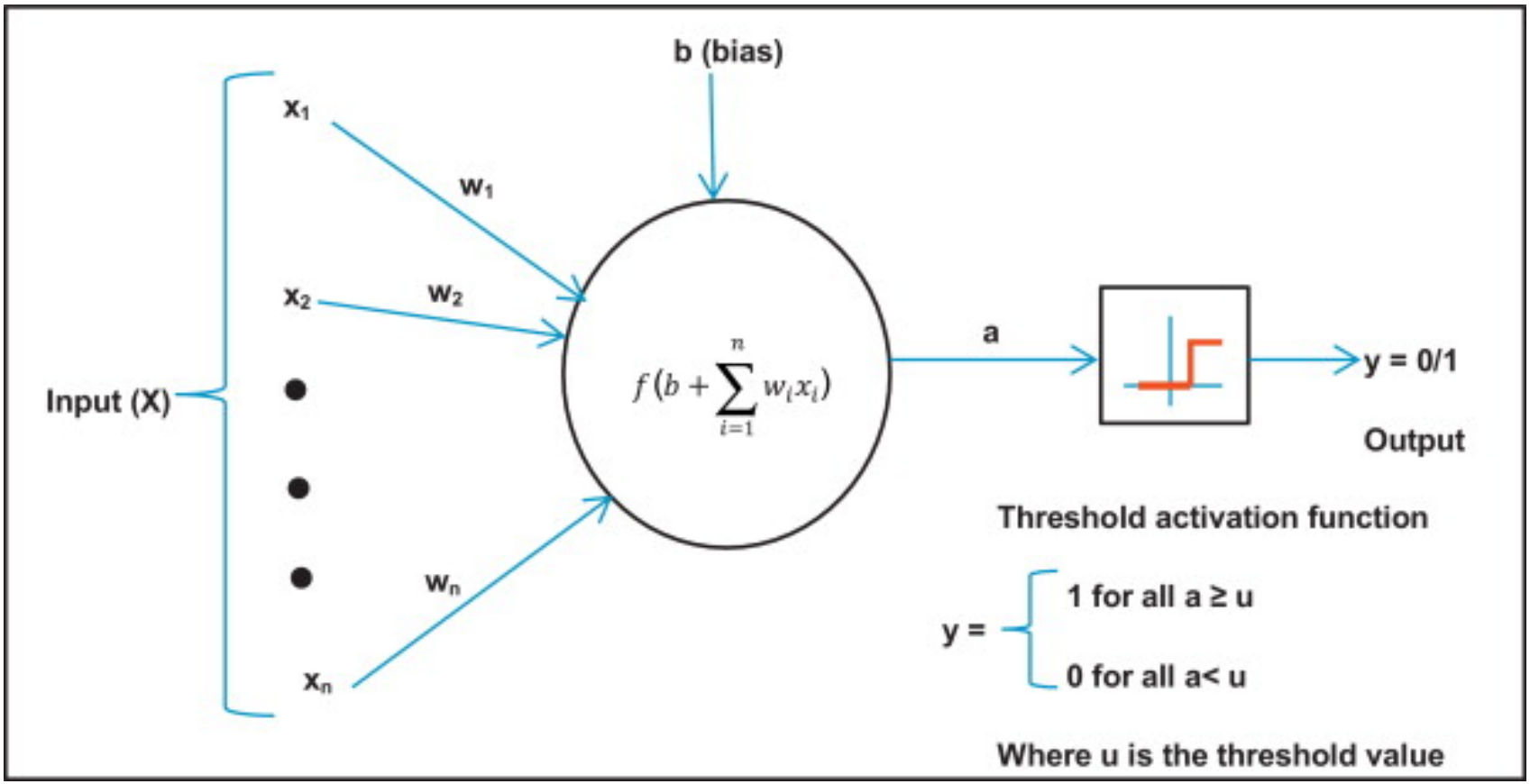
- The input layer receives the features generated by the CNN .
- The hidden layer is a sequence of neurons with weights that will be learned in the training step. A MLP is composed by one or more hidden layers.
- The output layer is also a sequence of neurons. However, it has a different activation function. Usually, the softmax function is used to generate the probabilities of each category in the problem scope.

Fully Connected Layer

Each neuron is composed by:

- A input vector $x_0, x_1, \ldots, x_n$, that represent the features
- A weight vector $w_0, w_1, \ldots, w_n$, that will be learned in the training step
- The bias
- An activation function
- The output

Fully Connected Layer



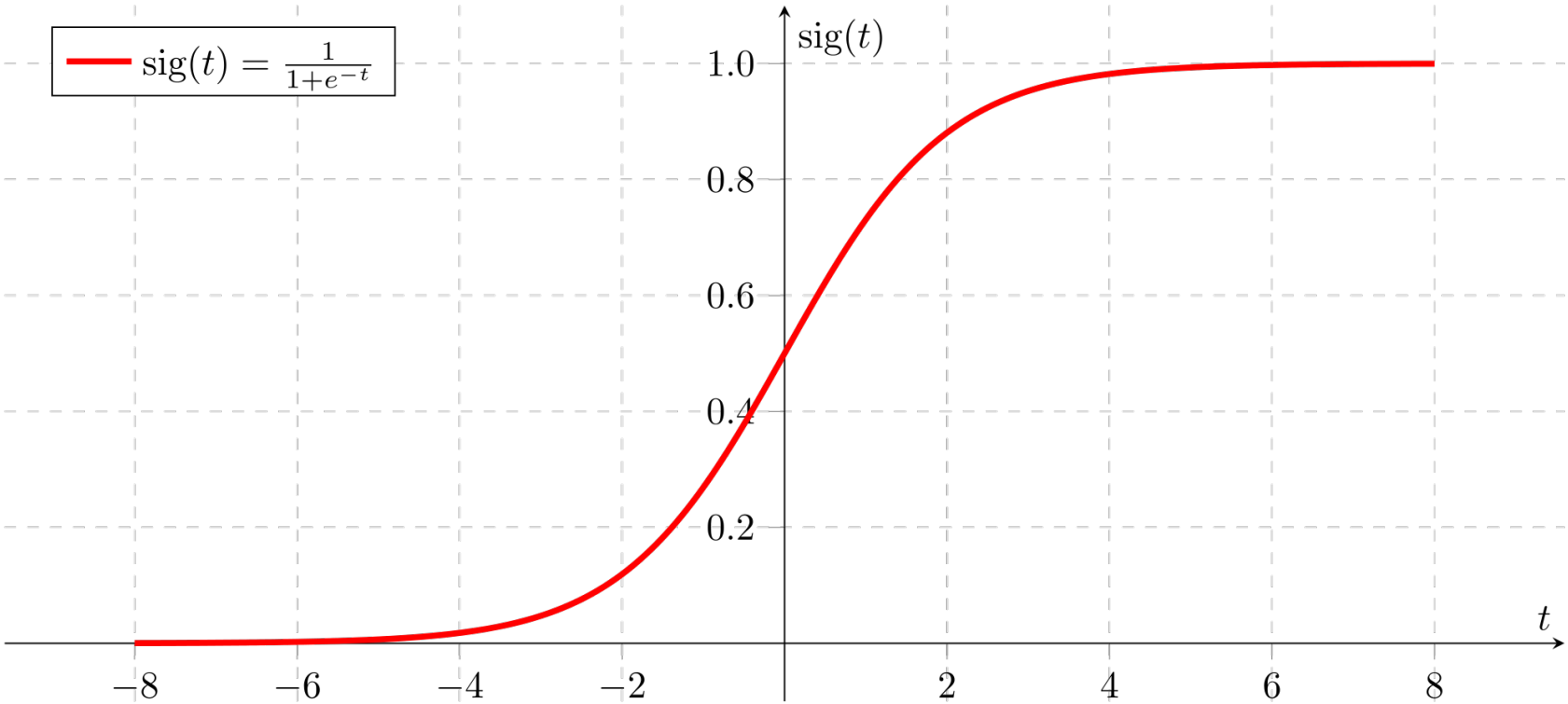
Fully Connected Layer

The Perceptron performs the following operation:

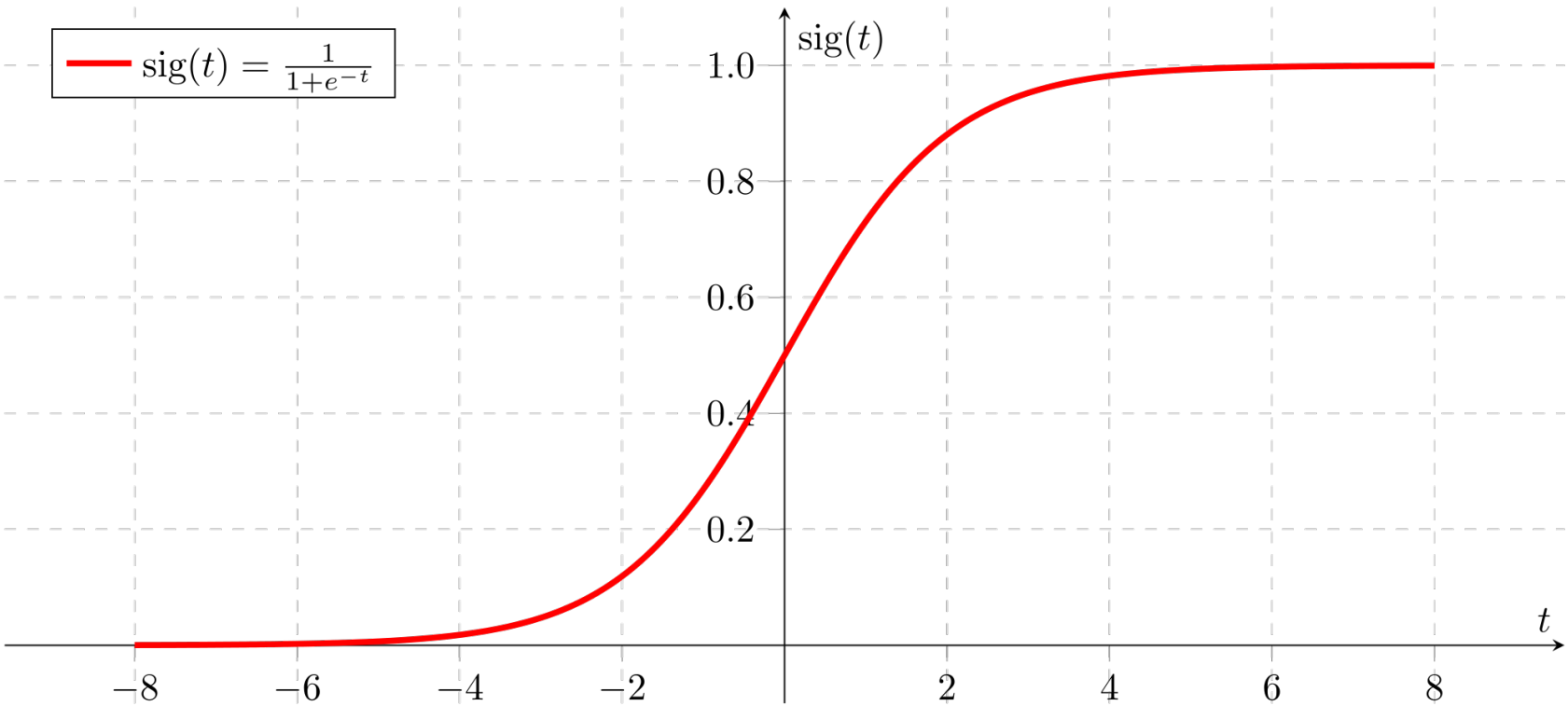
$$output = ActivationFunction(x_0 * w_0 + x_1 * w_1 + \ldots + x_n * w_n + bias) \quad (1)$$

The most common activation functions used in the literature are:
ReLu, Sigmoid, Softmax, Tanh, Hardlim

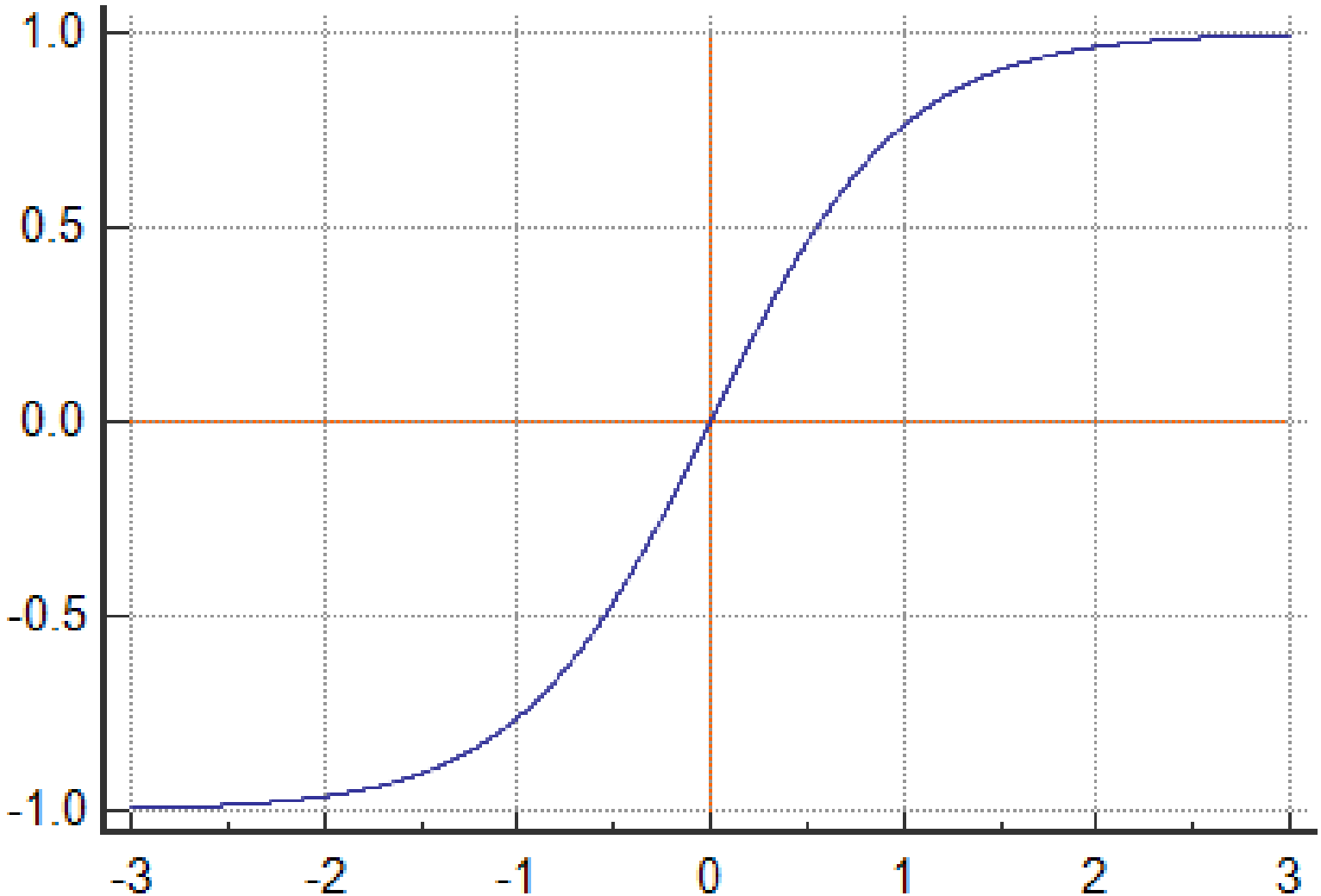
Sigmoid



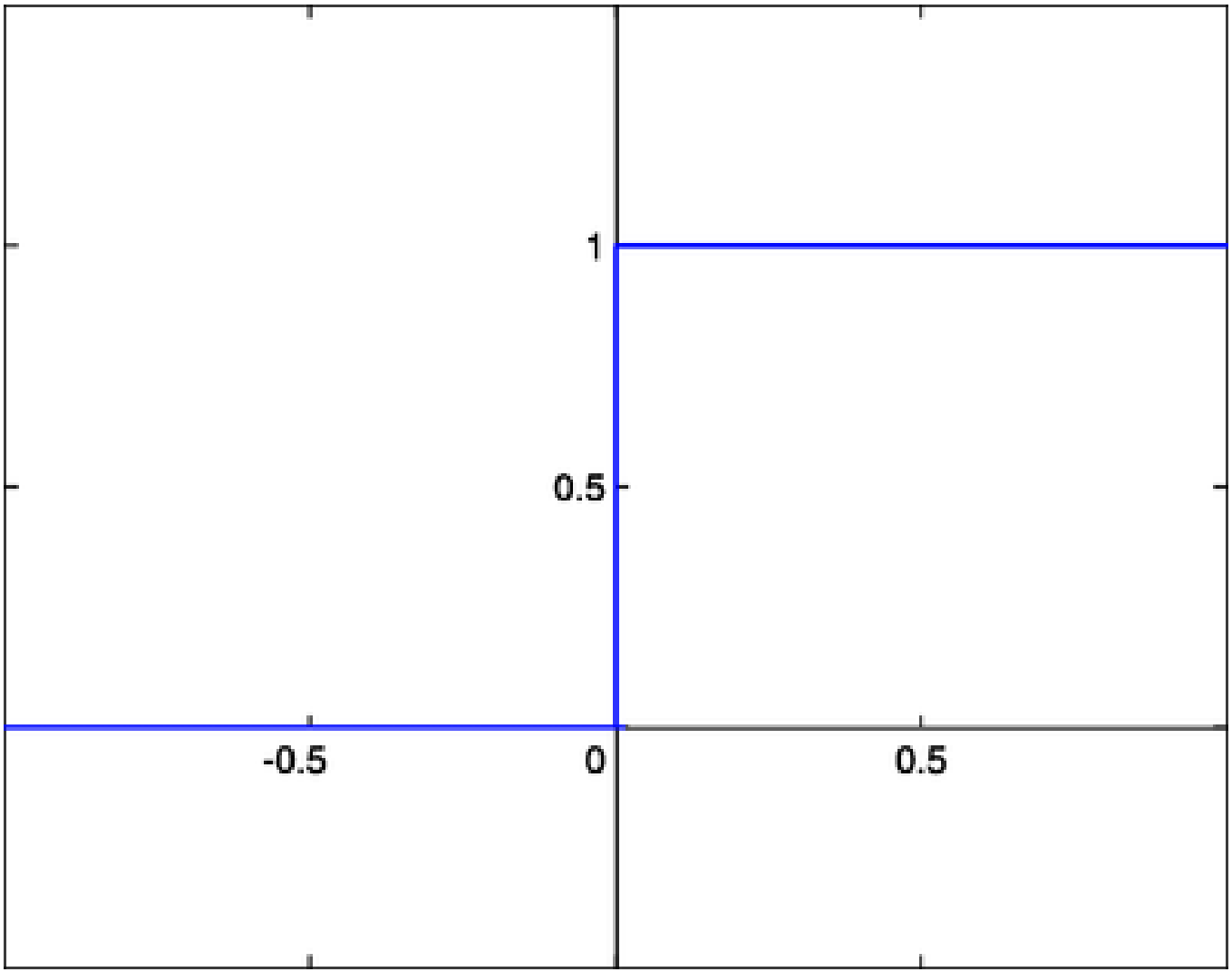
Softmax



Tahn



Hardlim



Training

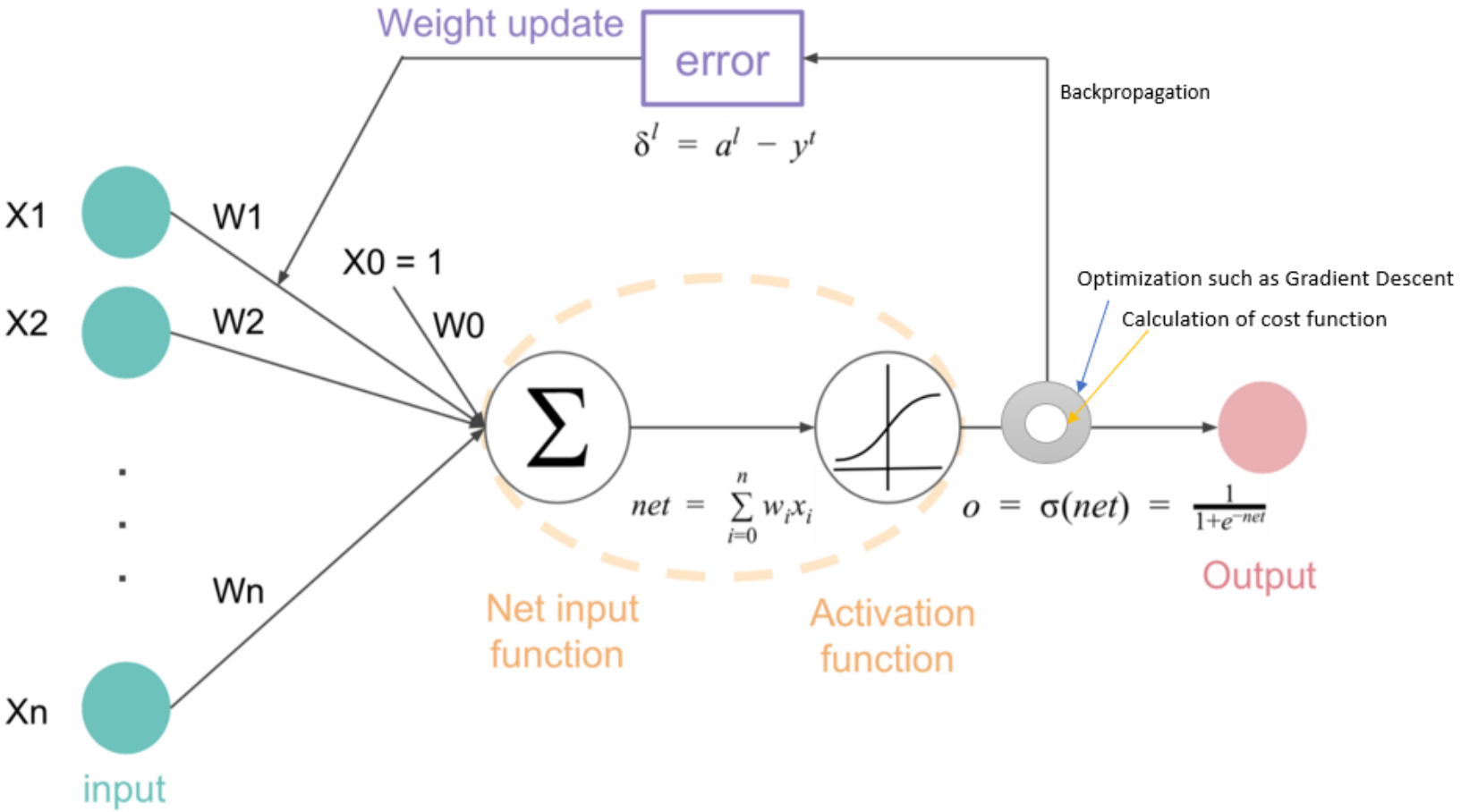
- In order to train a CNN model, a training dataset composed by a set of images and labels (classes, bounding boxes, masks) is used.
- The algorithm used to train a CNN is called back-propagation, that uses the output value of the last layer to measure an error value. This error value is used to update the weights of each neuron in that layer.
- The new weights are used to measure an error value and update the weights of the previous.
- The algorithm repeats the process until it reaches the first layer.

Training

A 3D CNN visualization:

https://scs.ryerson.ca/~aharley/vis/conv/

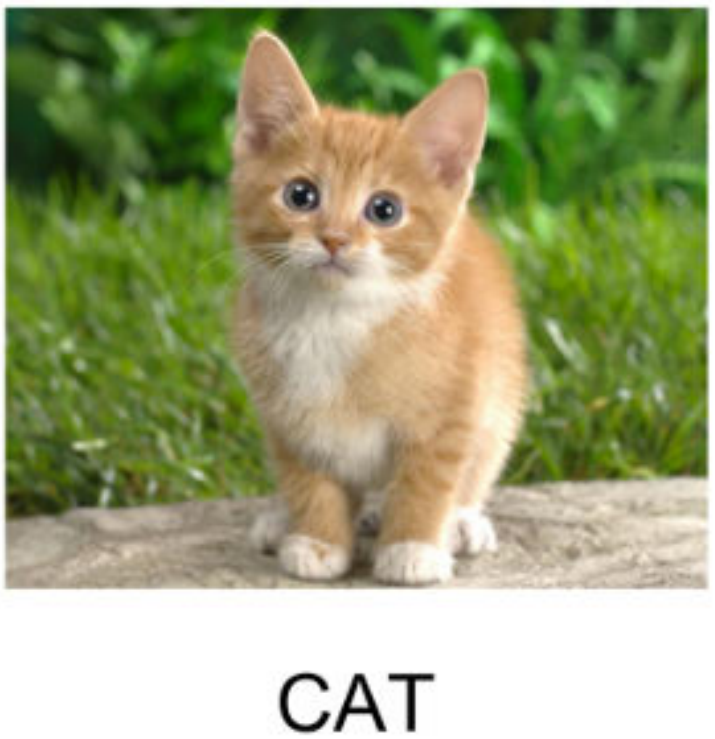
Training


Weight update
error
$\delta^l = a^l - y^t$
Backpropagation
X1
X2
Xn
W1
W2
Wn
X0 = 1
W0
Optimization such as Gradient Descent
Calculation of cost function
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$
Output
Net input function
Activation function
input

Classification

Popular CNNs for classification tasks

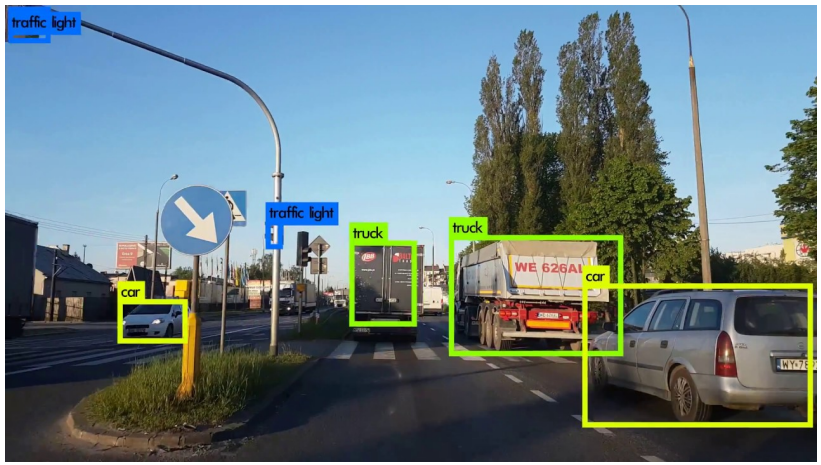
- VGG-16
- ResNets
- Inception



Detection

Popular CNNs for detection tasks

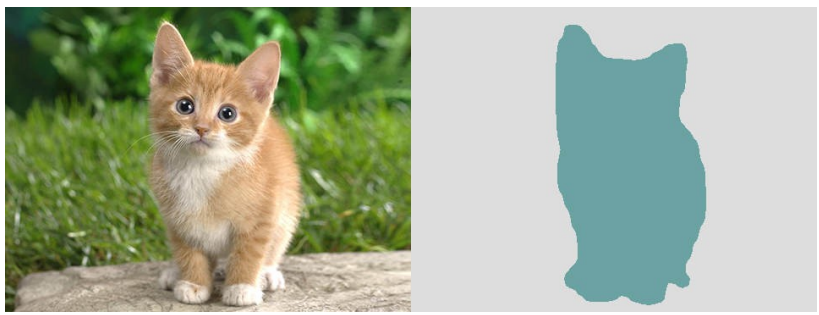
- Faster R-CNN
- YOLO



Segmentation

Popular CNNs for segmentation tasks

- FCN
- U-Net
- Mask R-CNN



VGG-16

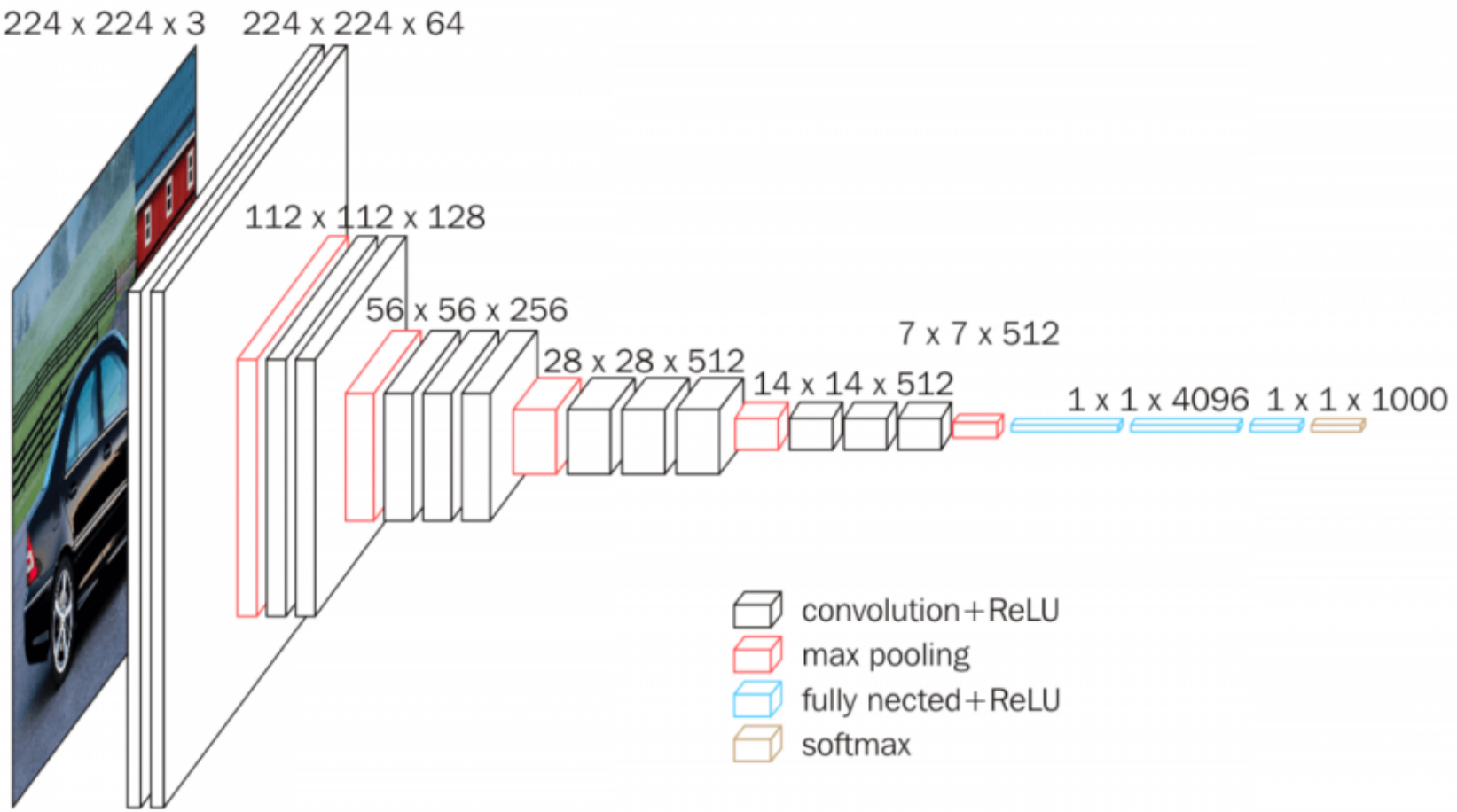


Figure: Very Deep Convolutional Networks for Large-Scale Image Recognition

VGG-16

- The VGG-16 was molded with stacks of convolutional layers with small kernels of size 3×3 instead of one convolutional layer with large kernels of size 7×7 and 11×11 in LeNet and AlexNet.
- A stack of convolutional layers with small filters sizes generates a decision function more discriminatory by increasing the number of non-linear rectification layers.
- VGG16 layers: `https://miro.medium.com/max/480/1*SPHdUMphNbJ9khi5Tg6fIw.png`

ResNet

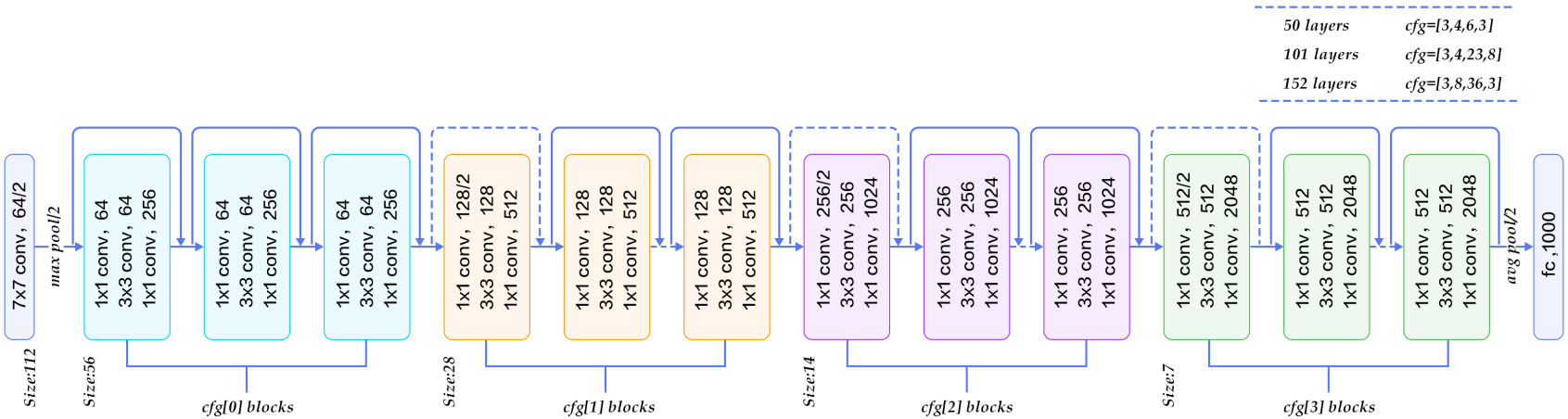


Figure: ResNet

ResNet Variations

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure: Deep Residual Learning for Image Recognition

ResNet

Aiming to build deeper neural networks, the ResNet was proposed with skip connections.

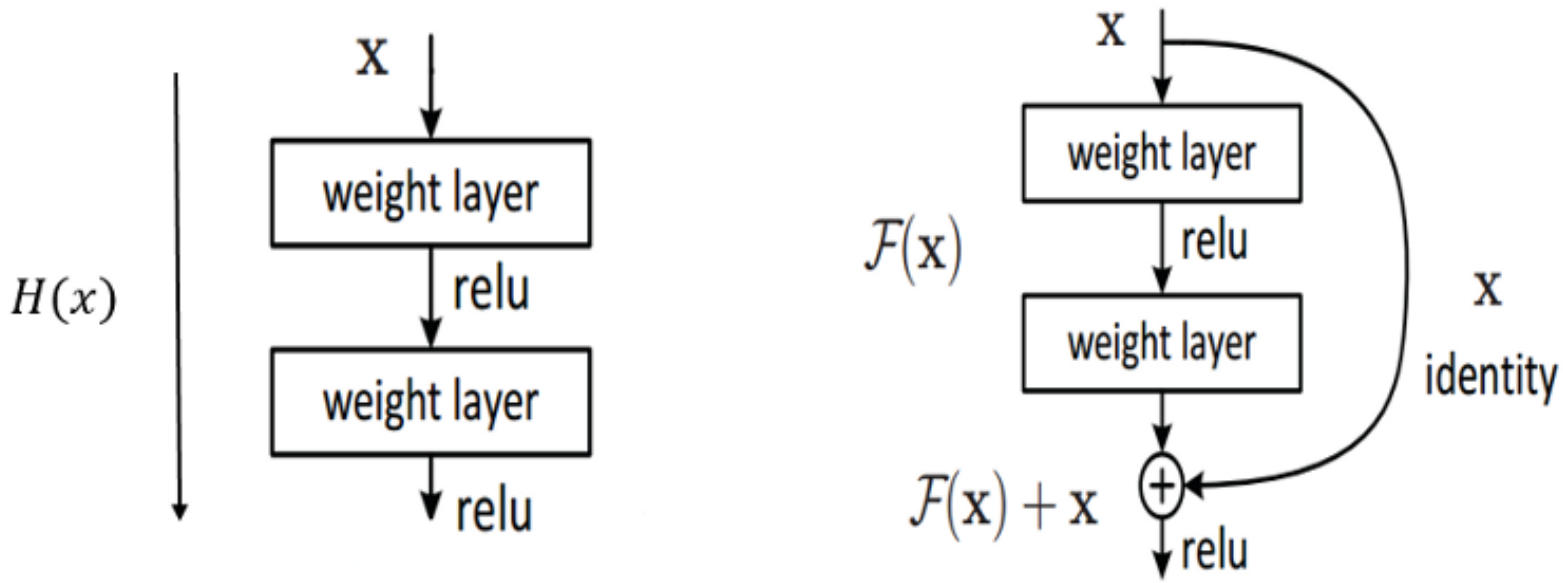
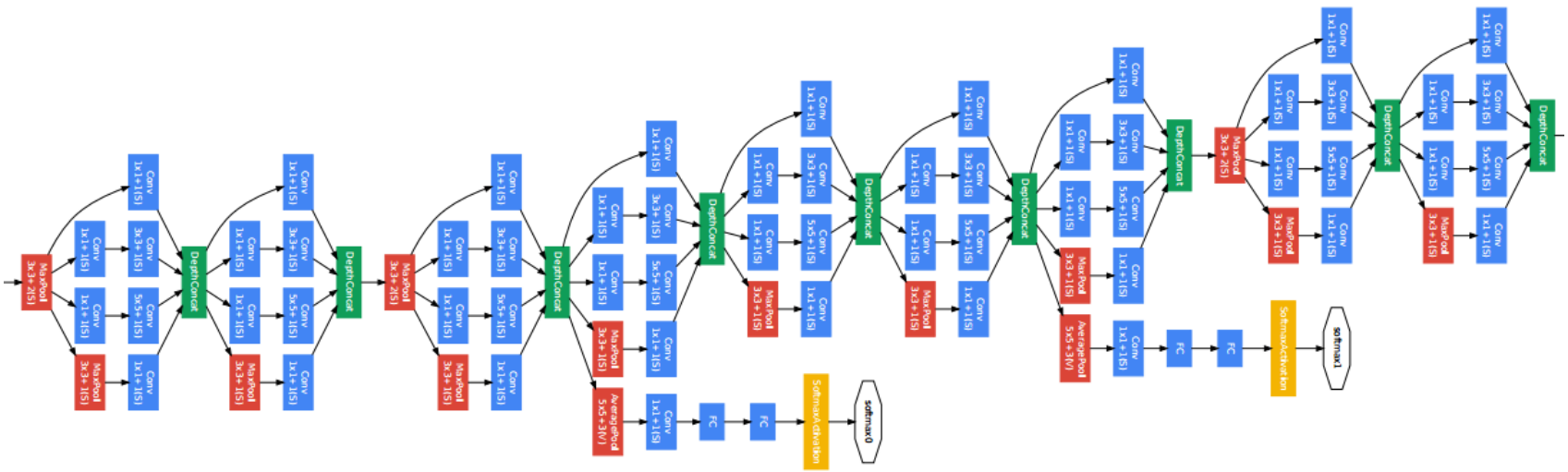


Figure: Difference between a convolution block without skip and with skip connection.

ResNet

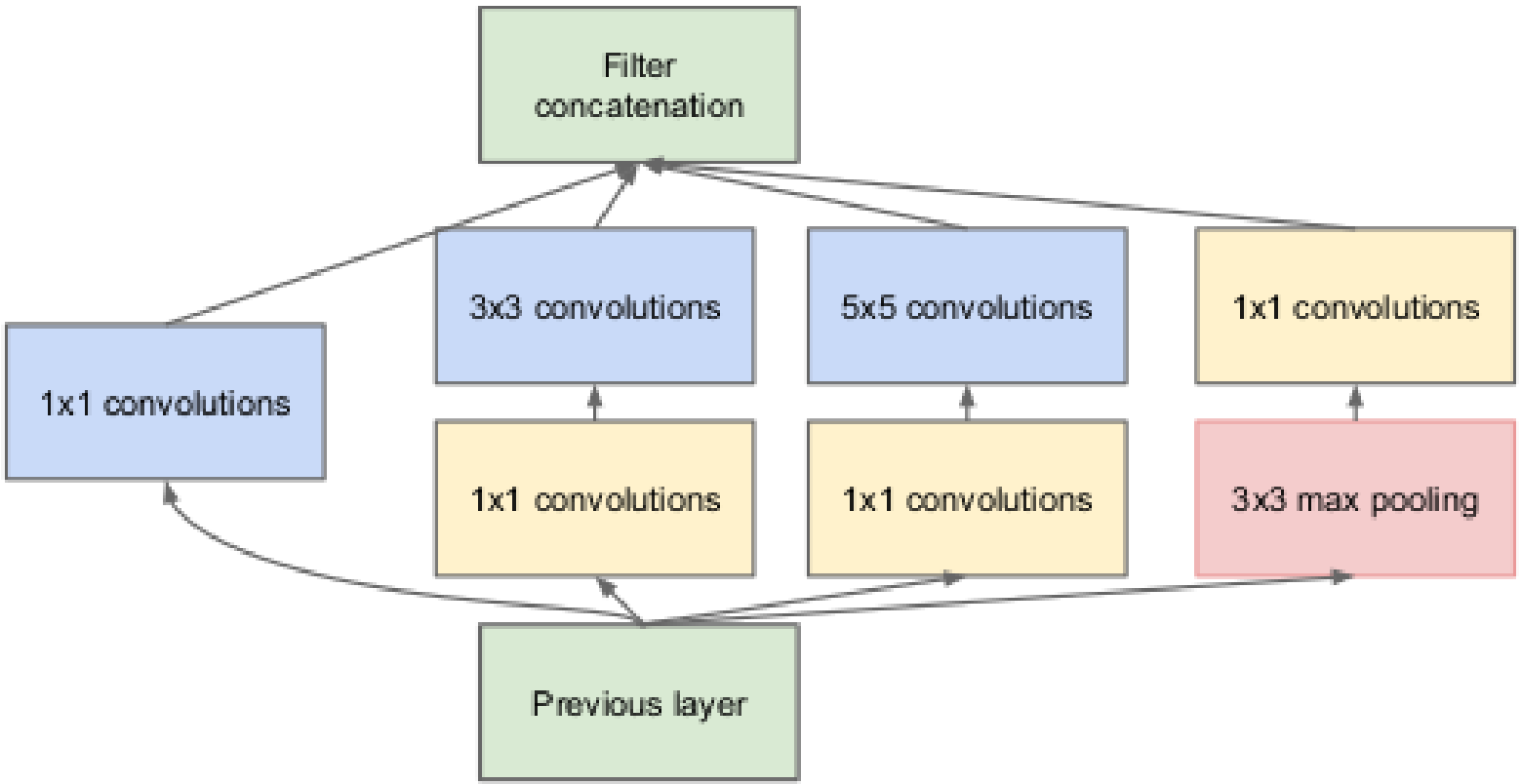
ResNet: https://adeshpande3.github.io/assets/ResNet.gif

Inception - GoogleLeNet



Inception - GoogleLeNet

Use a sequence of parallel convolutions of different sizes to learn objects of different size and position in the figure.



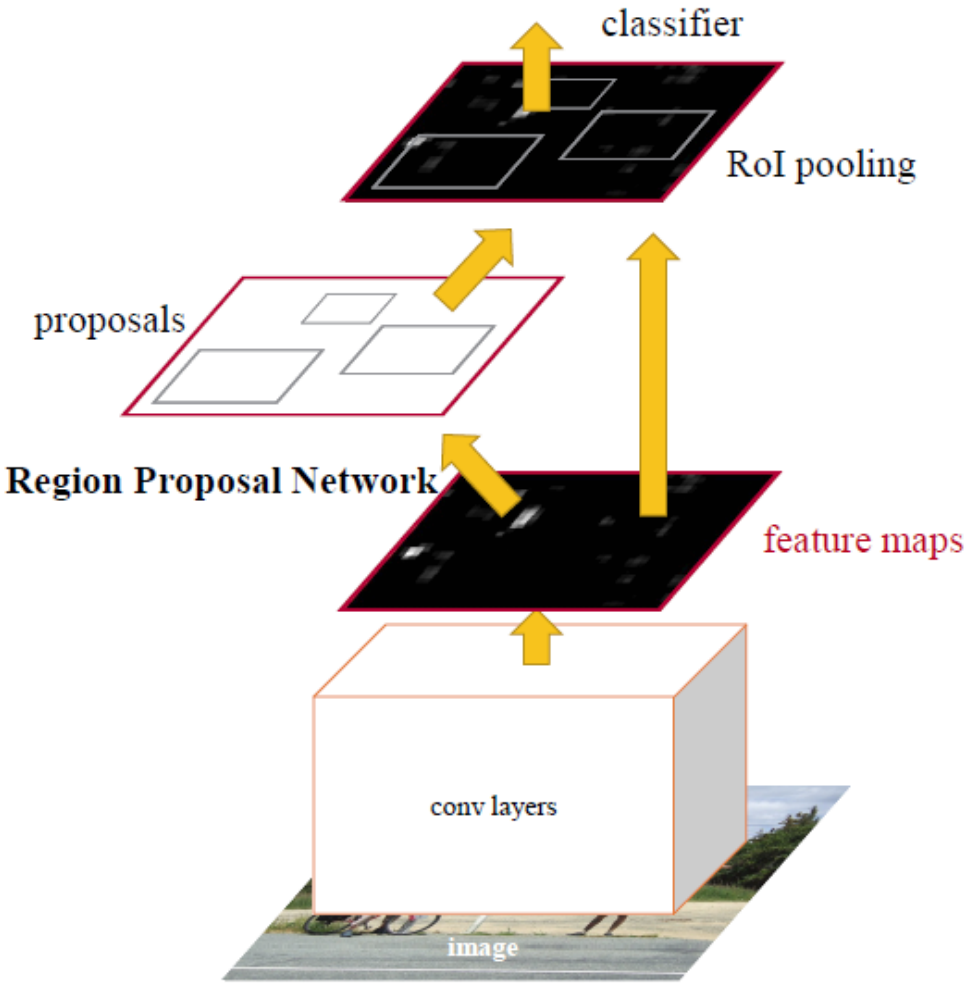
Inception - GoogleLeNet

GoogleLenet:

https://adeshpande3.github.io/assets/GoogleNet.gif

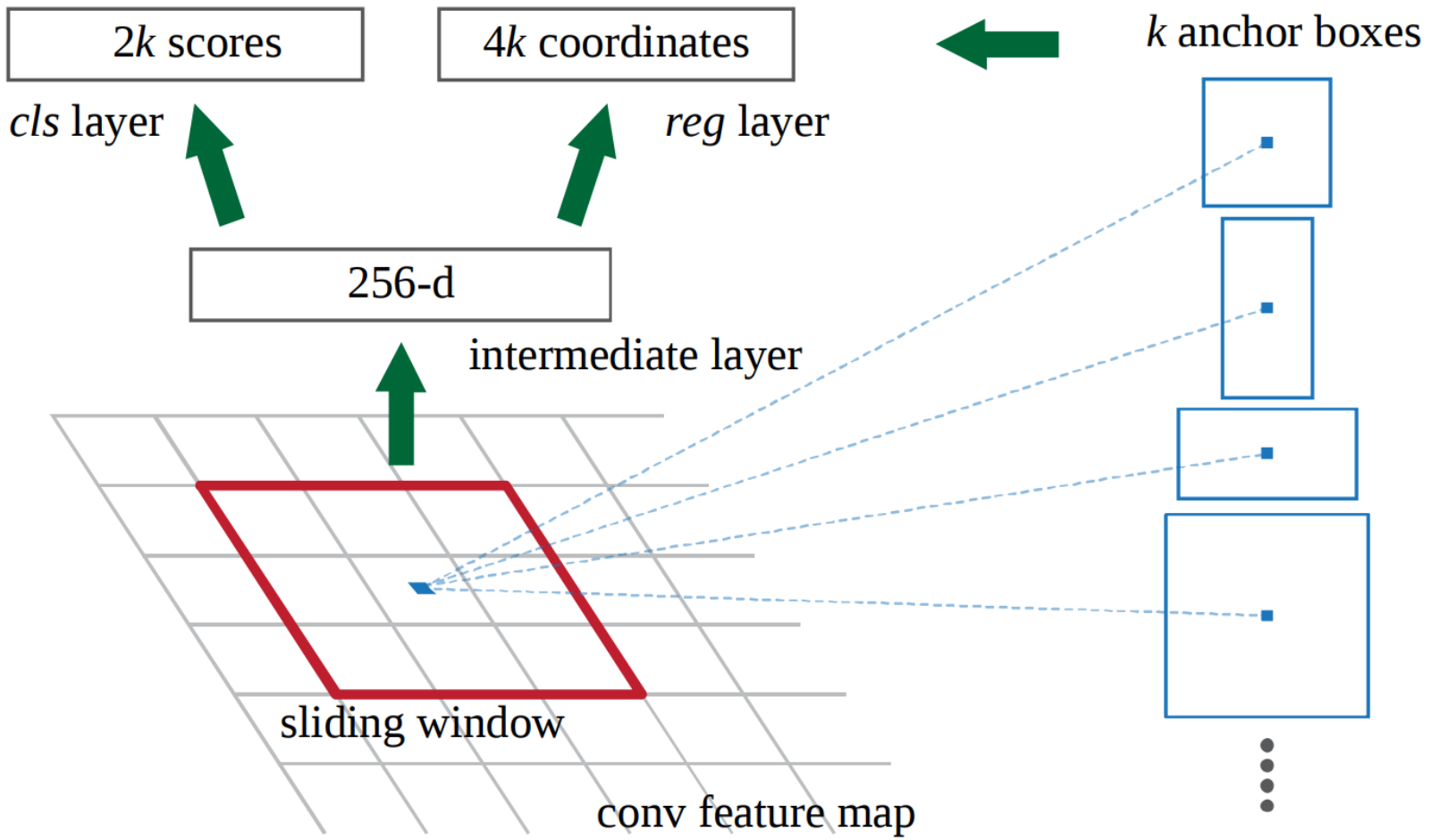
Faster R-CNN

The Faster R-CNN has the Region Proposal Network (RPN) to select the regions in the image with the highest probability of being objects.



Faster R-CNN

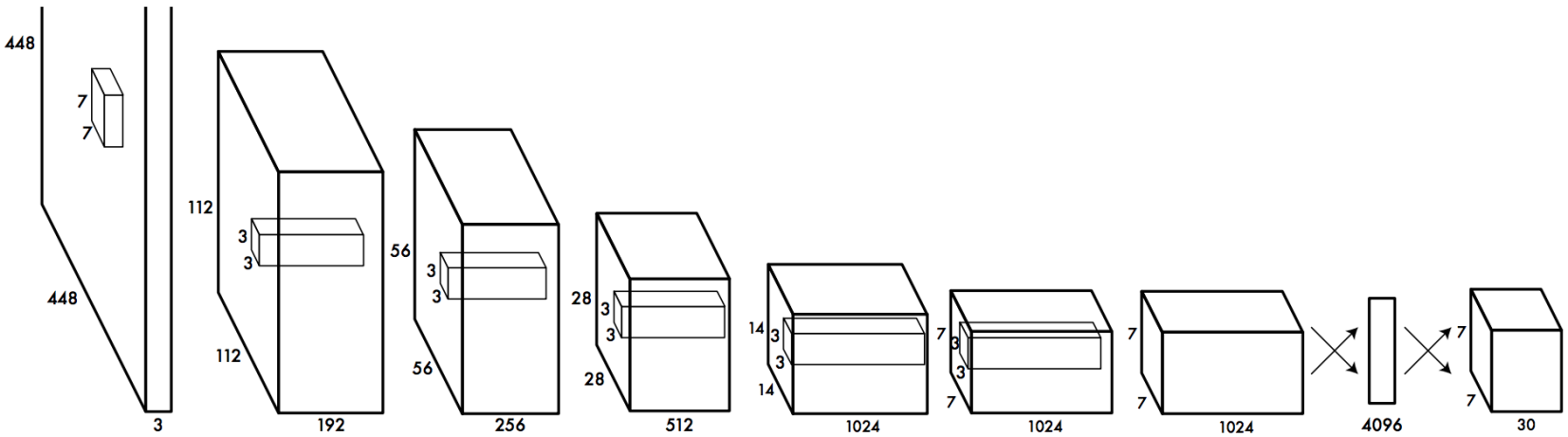
In the RPN, the Faster R-CNN generate a set of anchor boxes. The anchor boxes with the highest probability of being objects are used to generate the bounding boxes of each object in the image.



Faster R-CNN

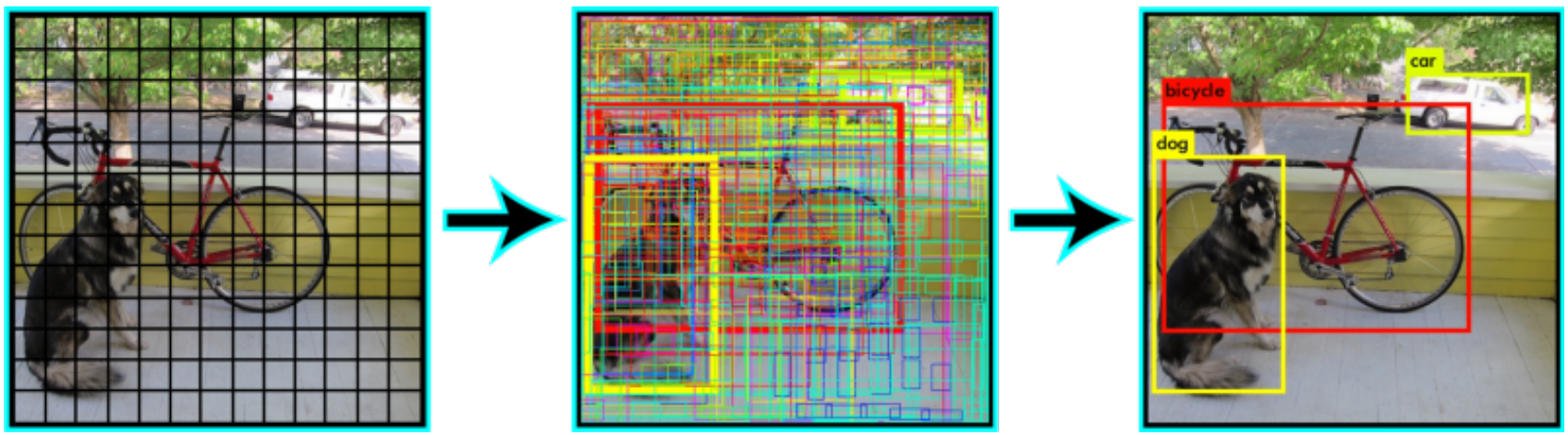
RoiPooling is applied to convert all proposed regions to a same size feature map. https://upload.wikimedia.org/wikipedia/commons/d/dc/RoI_pooling_animated.gif

You only look once (YOLO)



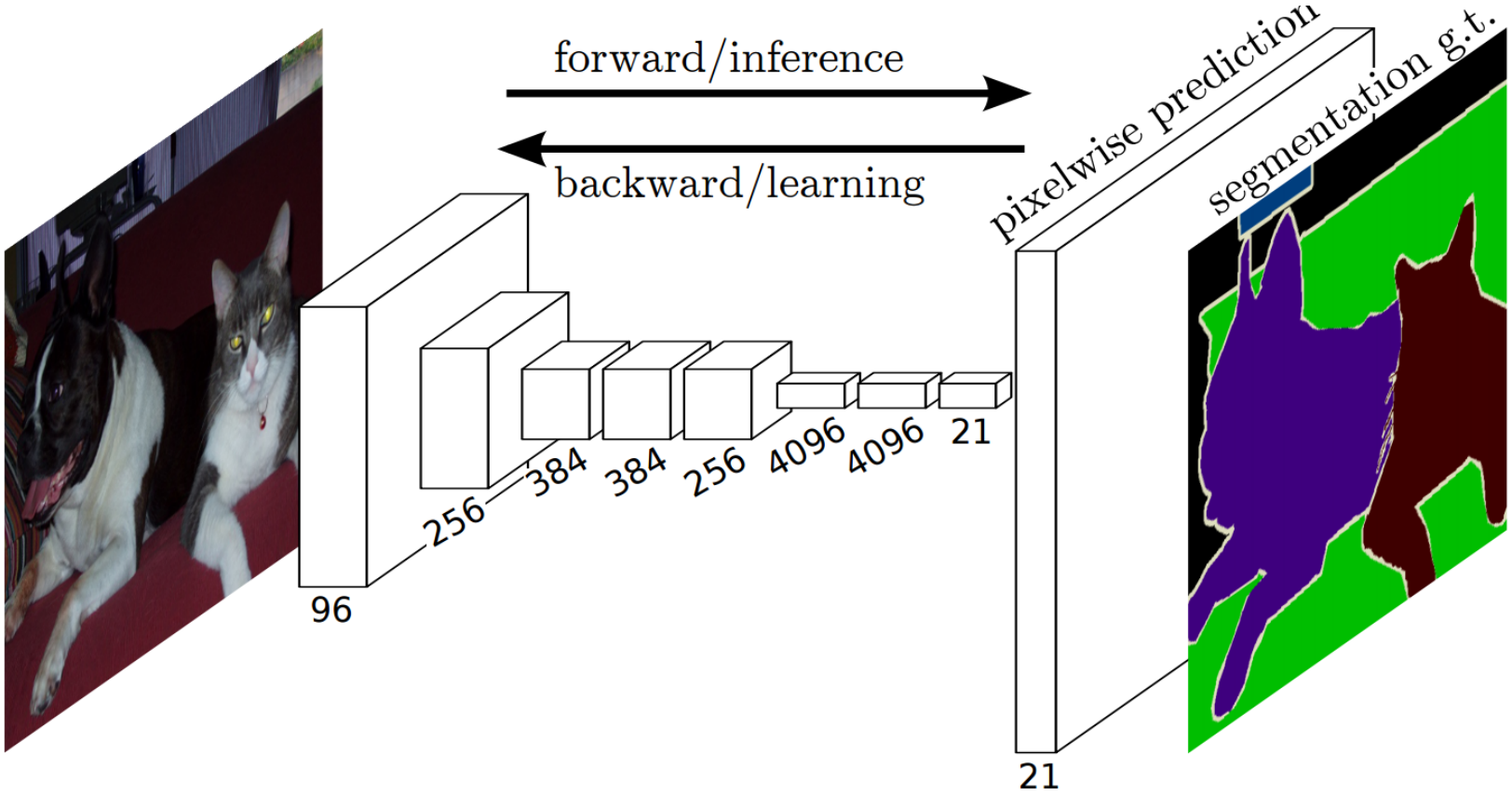
You only look once (YOLO)

In the YOLO, the image is divided into a $S \times S$ grid, with each cell in the grid predicting N bounding boxes. The bounding boxes with a very low probability of being an object are discarded and non-max suppression are utilized to remove multiple bounding box to a same object.

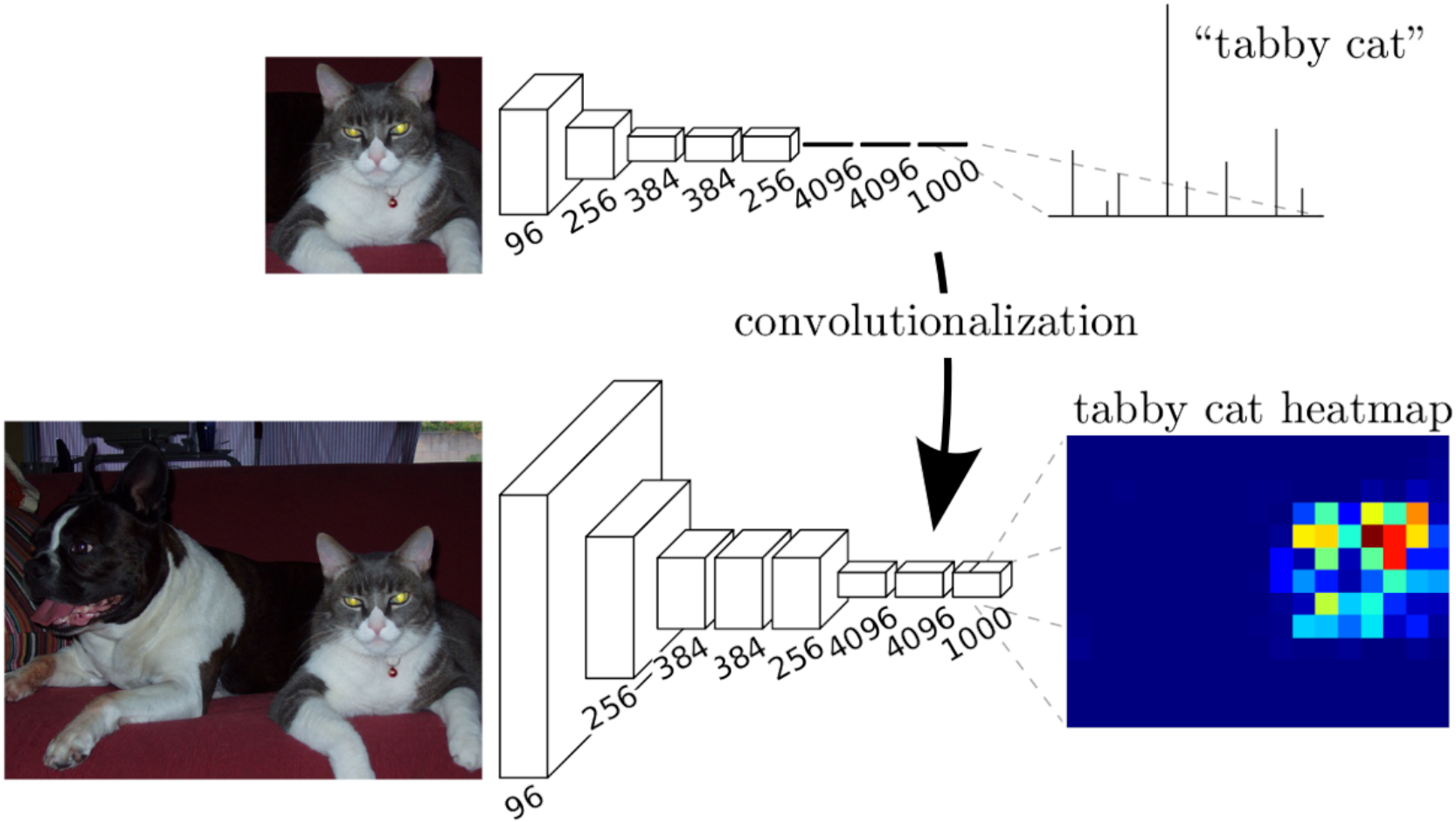


Fully Convolutional Network (FCN)

In the FCN, the fully connected layers are replaced by convolutional ones to generate a pixel level classification mask.

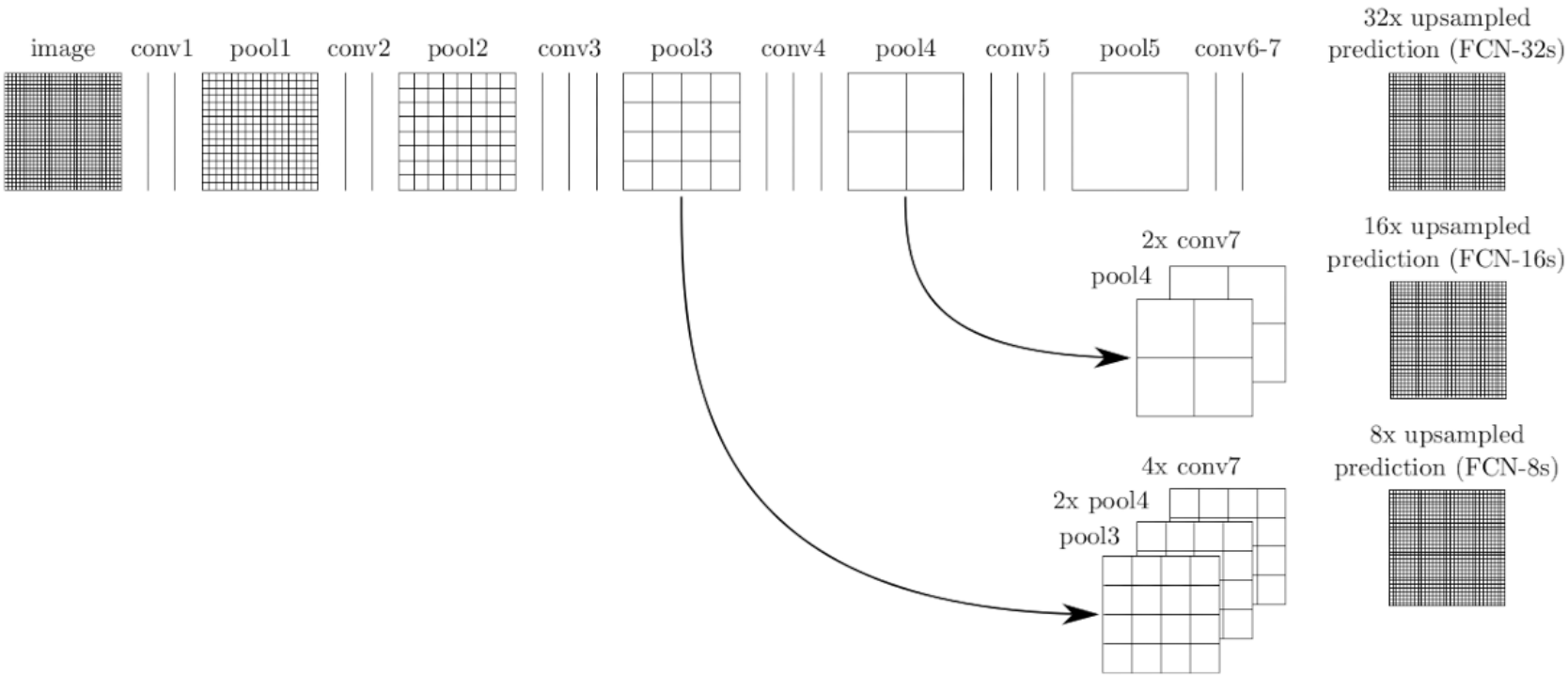


Fully Convolutional Network (FCN)



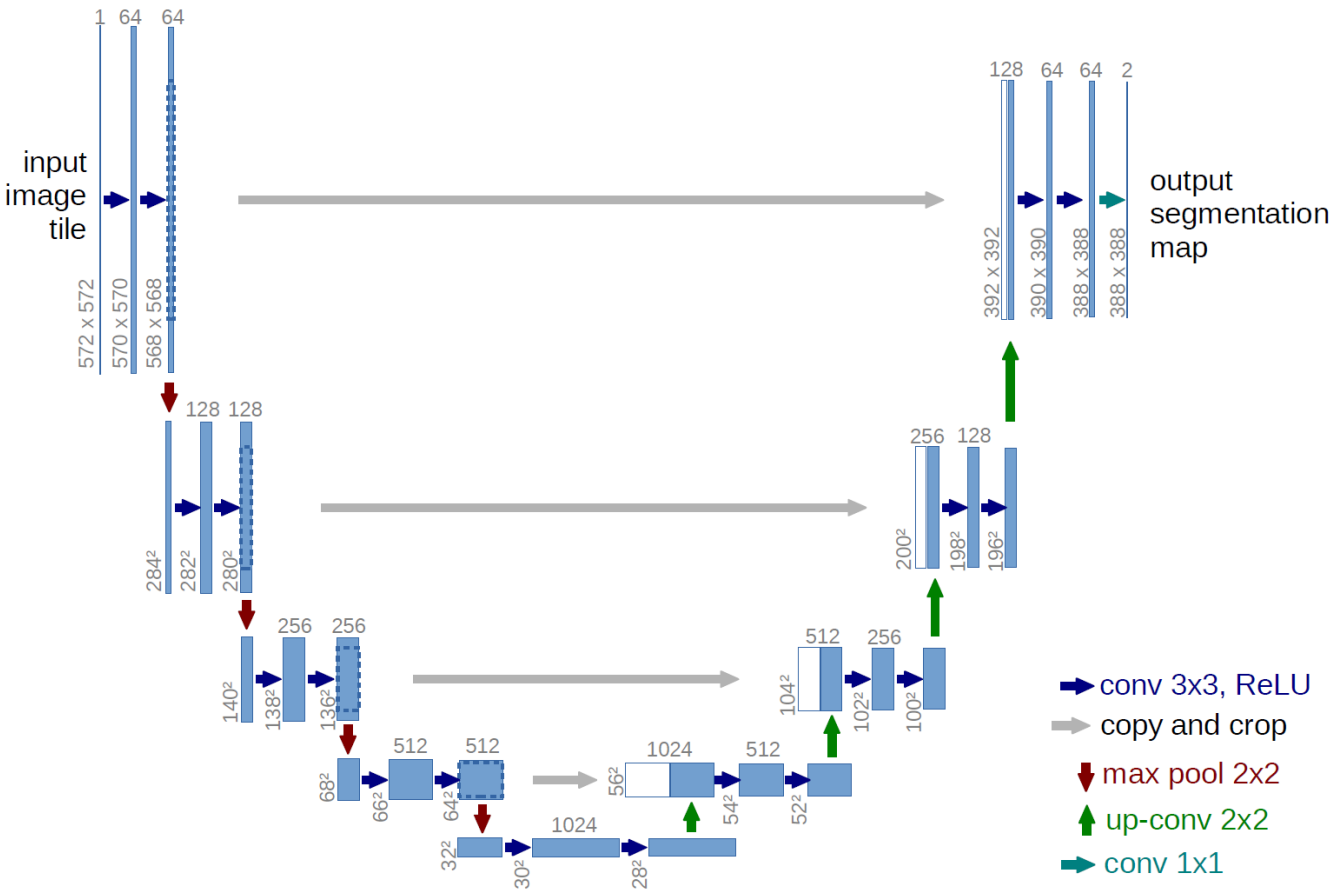
Fully Convolutional Network (FCN)

Three versions of the FCN was proposed.



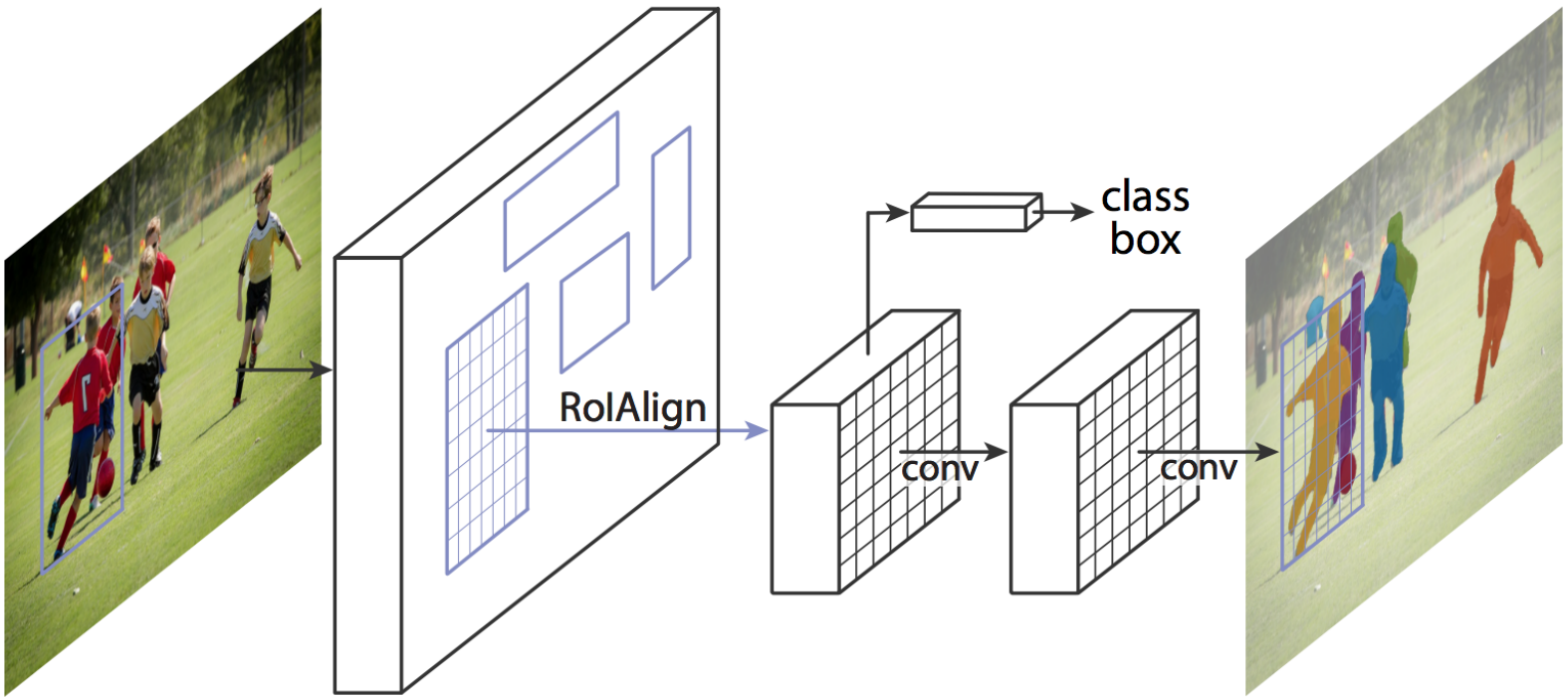
U-Net

Similar with the FCN. However, each convolutional block has a deconvolutional block.



Mask R-CNN

The Mask R-CNN is an evolution of the Faster R-CNN with a segmentation modulo. So, the Mask R-CNN performs the three tasks: classification, detection and segmentation.



Tensorflow

- Developed by Google
- Support for many programming languages: Python, C++, JavaScript, Java, Go, C# and Julia
- Runs on mobile platforms
- Works with static computation graphs (first defines a graph, than performs the training).

PyTorch

- Developed by Facebook
- Support for: Python and C++
- Runs on mobile platforms
- Works with dynamically updated graphs (the graph can be altered in the training process).

Keras

- It is a high level API
- Works on top of: Tensorflow, Theano, and CNTK
- Very easy of use

Model Zoo

There are high number of online repositories called Model Zoo's with a wide range of frameworks and implementations in many different frameworks. `https://modelzoo.co/`

Papers

- Deep Residual Learning for Image Recognition: https://arxiv.org/abs/1512.03385
- Fully Convolutional Networks for Semantic Segmentation: https://www.researchgate.net/publication/303505510_Fully_Convolutional_Networks_for_Semantic_Segmentation
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks: https://arxiv.org/abs/1506.01497
- Going Deeper with Convolutions: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf
- Gradient-Based Learning Applied to Document Recognition: https://ieeexplore.ieee.org/document/726791
- ImageNet Classification with Deep Convolutional Neural Networks: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Papers

- You Only Look Once: Unified, Real-Time Object Detection: https://arxiv.org/abs/1506.02640
- Mask R-CNN: https://arxiv.org/abs/1703.06870
- U-Net: Convolutional Networks for Biomedical Image Segmentation: https://arxiv.org/abs/1505.04597
- A Framework for Object Detection, Tracking and Classification in Urban Traffic Scenarios Using Stereovision: https://www.researchgate.net/publication/224602865_A_Framework_for_Object_Detection_Classification_in_Urban_Traffic_Scenarios_Using_Stereovision
- Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position: https://link.springer.com/article/10.1007/BF00344251

References

- https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852
- https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
- https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
- https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection
- https://elogeel.wordpress.com/2010/05/10/multilayer-perceptron-2/
- Network-level accident-mapping: Distance based pattern matching using artificial neural network

References

- https://datascience.stackexchange.com/questions/44703/how-does-gradient-descent-and-backpropagation-work-together
- https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-4-resnet/
- https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
- https://medium.com/@enriqueav/object-detection-with-yolo-implementations-and-how-to-use-them-5da928356035
- https://pjreddie.com/darknet/
- https://towardsdatascience.com/mask-r-cnn-for-ship-detection-segmentation-a1108b5a083

References

- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f
- https://web.inf.ufpr.br/vri/databases/
- https://medium.com/intro-to-artificial-intelligence/semantic-segmentation-udaitys-self-driving-car-engineer-nanodegree-c01eb6eaf9d
- https://www.raghavendersahdev.com/place-recognition.html
- https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

References

- https://www.tinymind.com/learn/terms/relu
- https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e
- https://www.medcalc.org/manual/tanh_function.php
- https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions