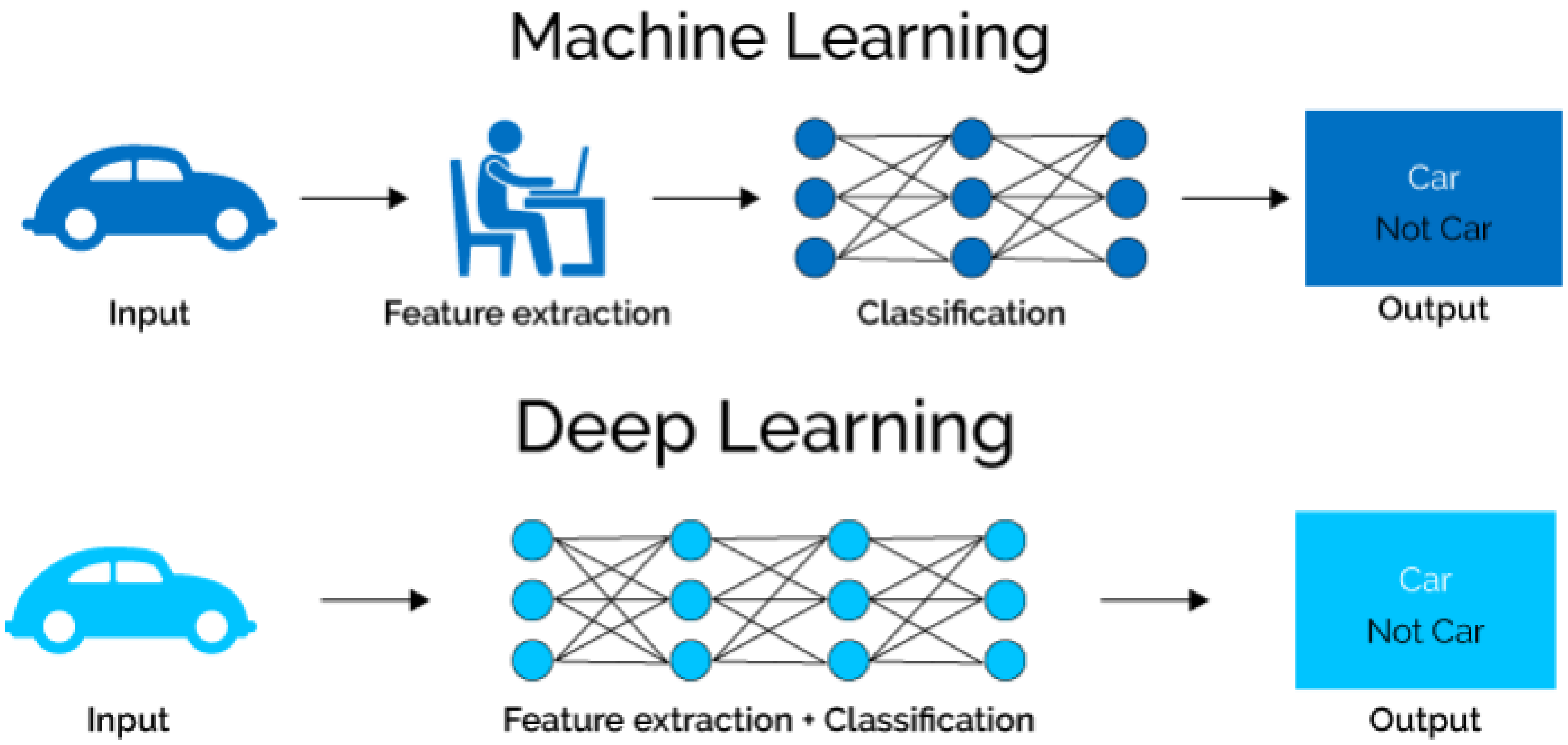
International University for Science & Technology (IUST)
Department of Computer &Informatics Engineering
Neural Networks unit (7)

Deep Learning

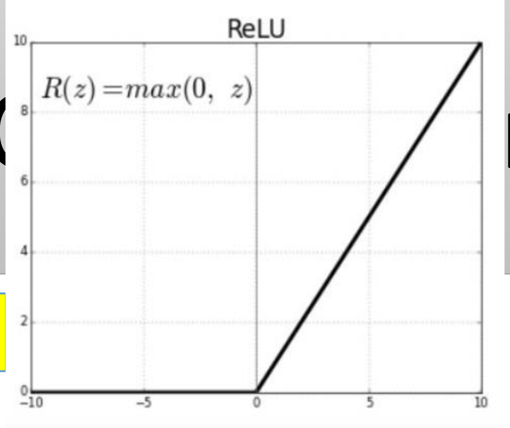
Neural Networks

Dr. Ali Mayya

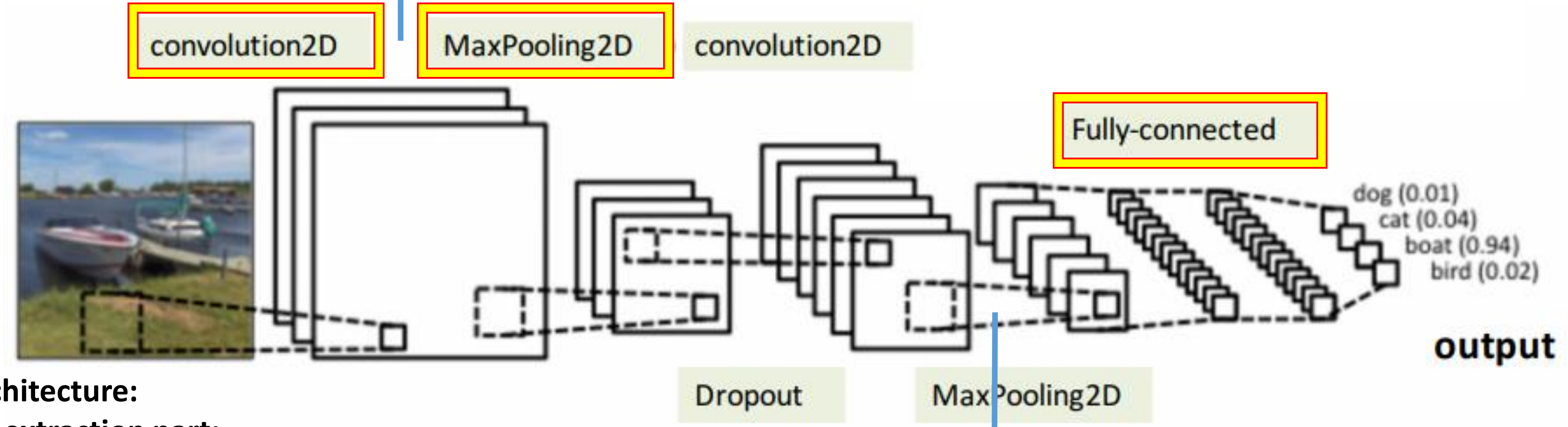
Machine Learning Vs. Deep Learning



Deep Learning- CNN (nal Neural Networks



•



CNN architecture:

Feature extraction part:

1- Convolution Layers

2- Pooling Layers

Classification part:

1- Fully-Connected layers:

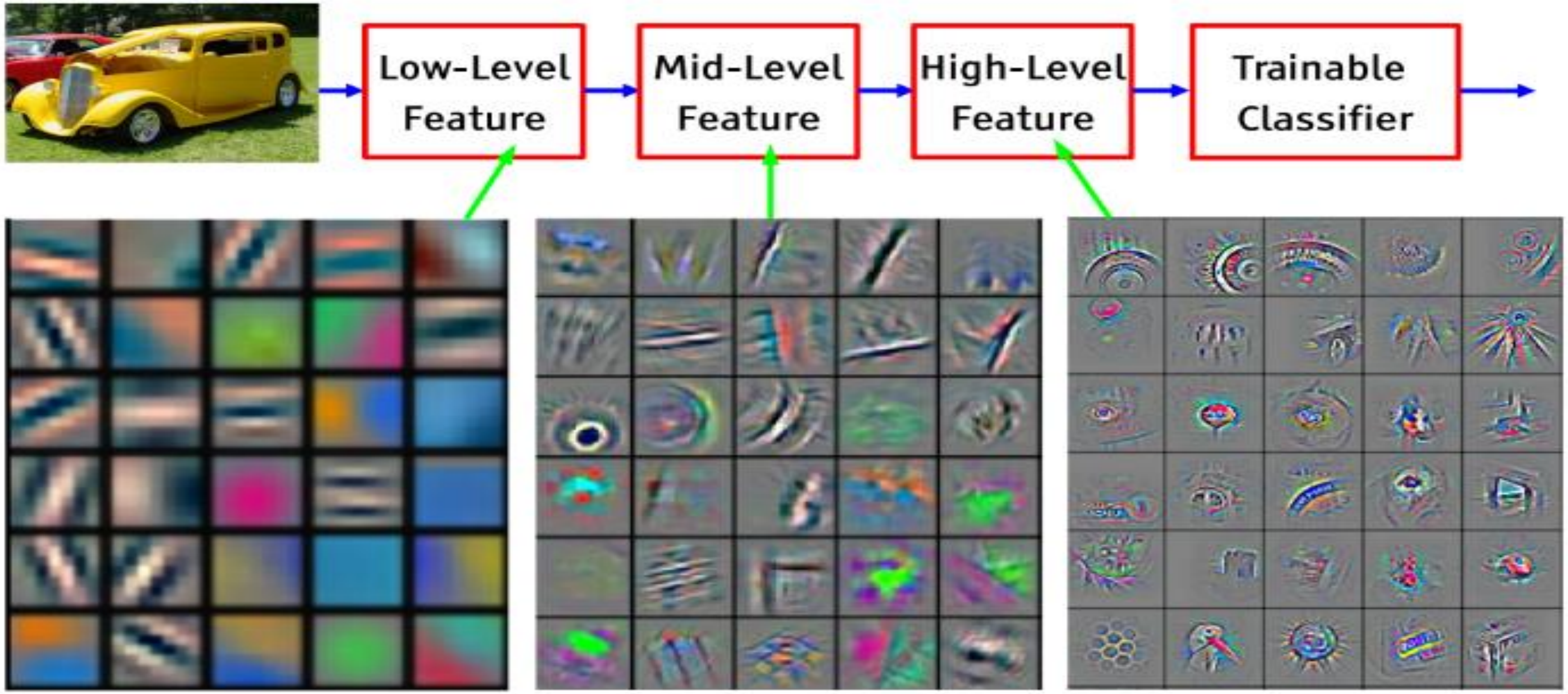
Some dense layers followed by

final layer is the classification layer with ***sigmoid*** or ***Softmax*** activation function.

$$\mathrm{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Deep Learning- CNN Convolutional Neural Networks

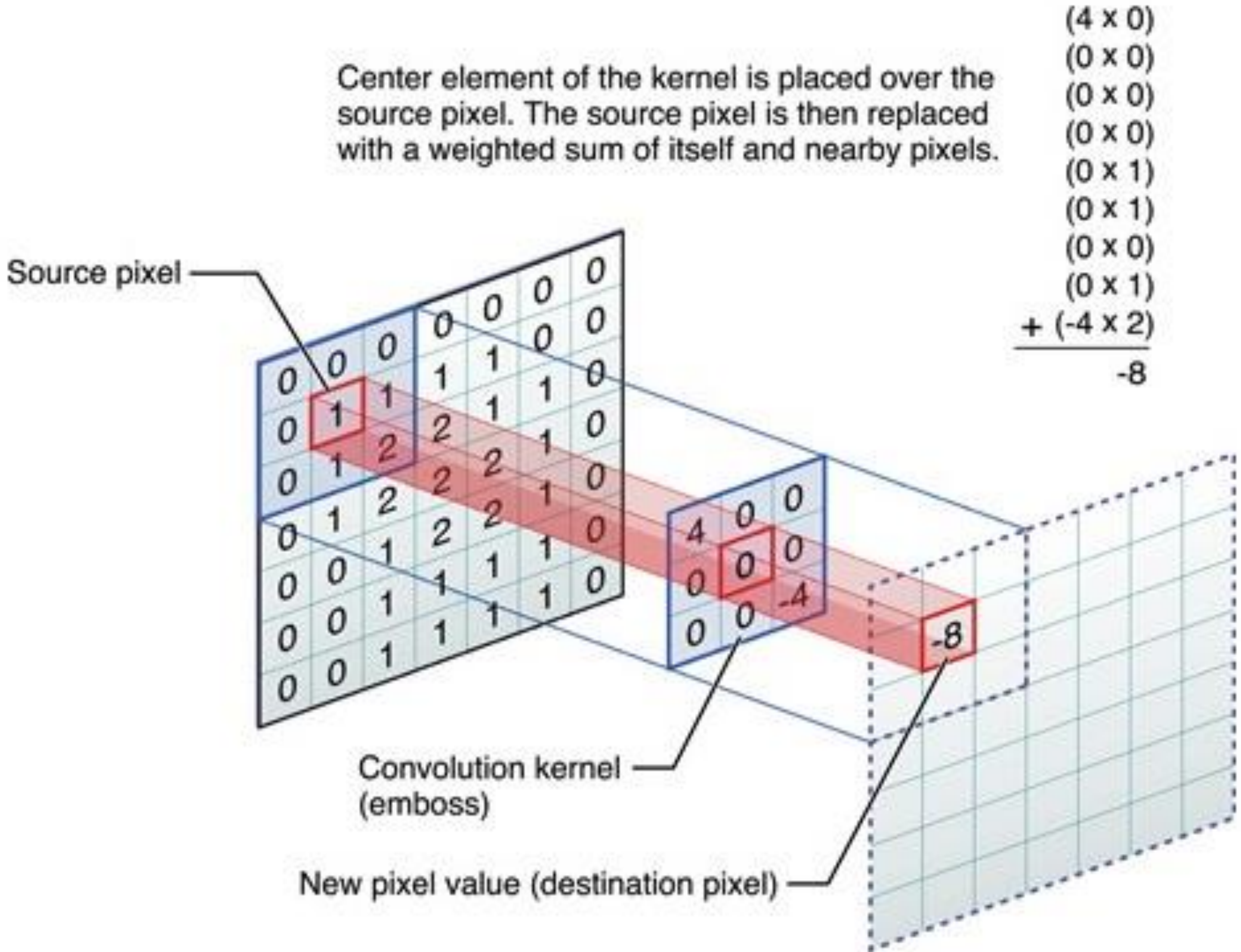
Hierarchical representation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Deep Learning- CNN Convolutional Neural Networks

Convolution



Convolution layer contains main part called filter (kernel) which is a k*k matrix applied to the input image in a specific neighborhood
What is convolution?
The weighted sum of multiplication between kernel and corresponding neighborhood of the image's pixel
Based on the values of the kernel, the output of the convolution is defined.

Deep Learning- CNN Convolutional Neural Networks

Convolution

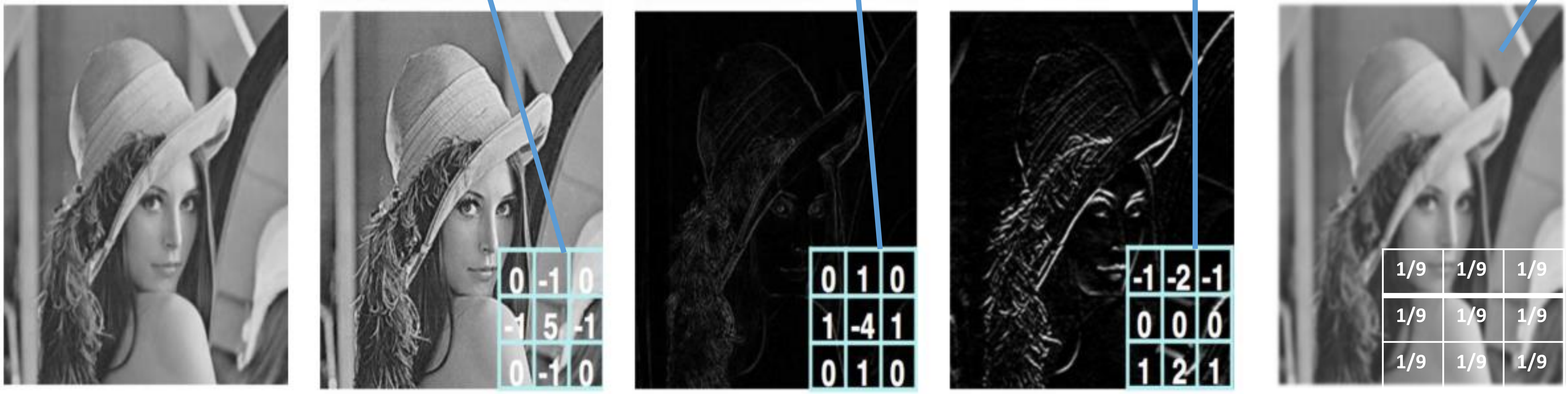
- **Based on the values of the kernel, the output of the convolution is defined.**

Derivation on both x and y (sum of kernel values=1) → Horizontal and vertical edge sharpening

Derivation on both x and y (sum of kernel values=0) → Horizontal and vertical edge detection → isolated points detection

Derivation on y axis (sum of values=0) and 4 neighborhood is multiplied by 2 → Strong Horizontal edges

Sum of values=1 (all positives) and apply the weighted sum → Smoothing

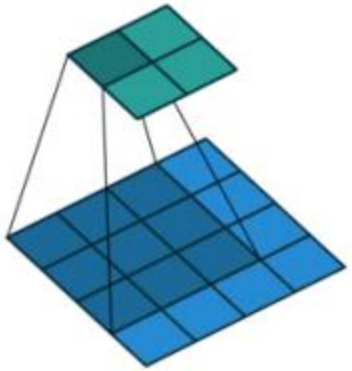


Original | Sharpen | Edge Detect | "Strong" Edge Detect

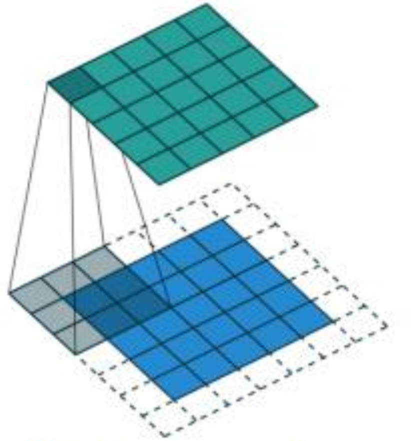
Deep Learning- CNN Convolutional Neural Networks

Convolution

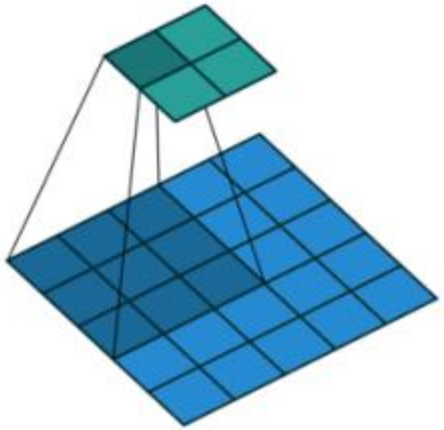
•



No padding, stride 1



Padding 1, stride 1



No padding, stride 2

Input image size: 4*4
Kernel (3*3),
stride=1, padding= No
Output size: 2*2

Input image size: 5*5
Kernel (3*3),
stride=1, padding= 1
Output size: 5*5

Input image size: 5*5
Kernel (3*3),
stride=2, padding= No
Output size: 2*2

Convolution has two parameters:
Stride: how many squares the kernel skip when moving across image, from left to right and from top to bottom.
Padding: number of added rows and columns to the boarders of the image to handle the neighborhood pixels.
The output of each convolution layer is called the ***activation maps***

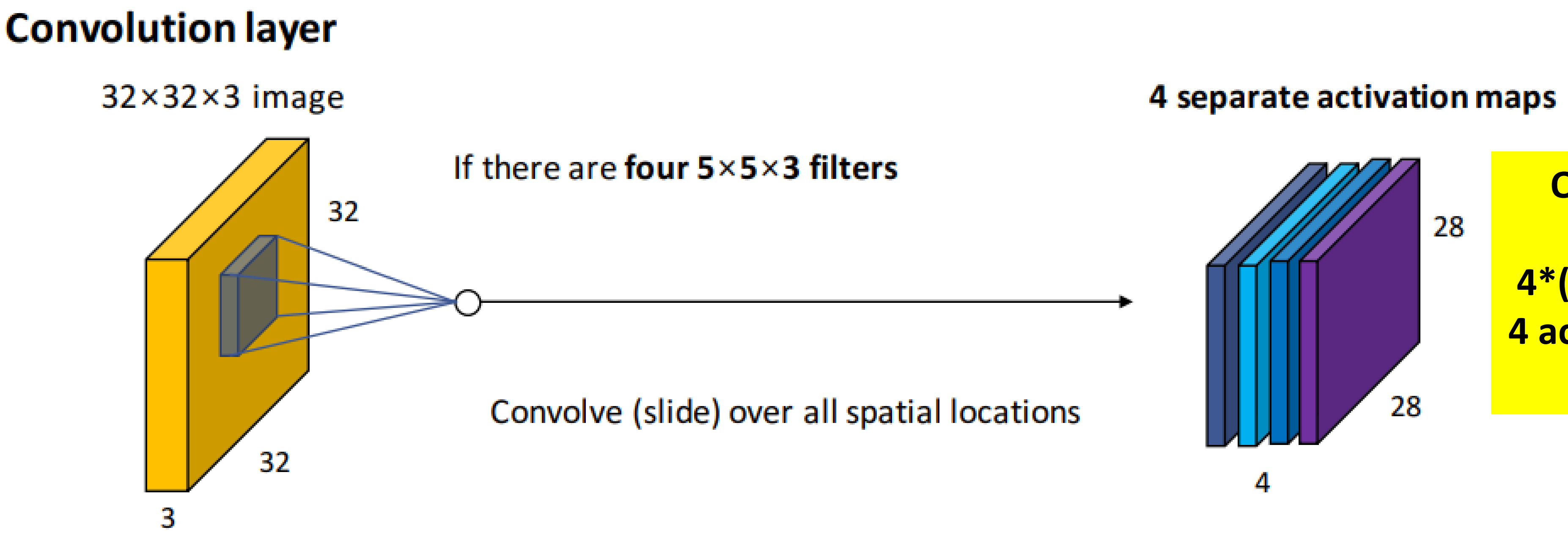
All activation maps's values are subjected to Relu activation function

Deep Learning- CNN Convolutional Neural Networks

Convolution

-

Output volume size= N*((W−K+2P)/S+1)
N: number of filters of the convolutional layer
W: image size
K: kernel size
P: padding, S: stride



Output volume size= 4*((32−5+0)/1+1) = 4 activation maps of size 28*28

Deep Learning- CNN Convolutional Neural Networks

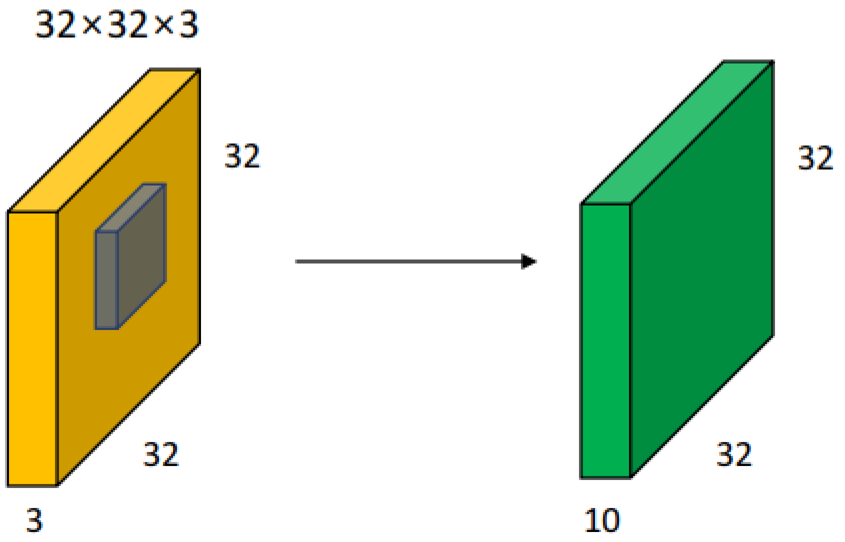
Convolution

-

Output volume size= N*((W−K+2P)/S+1)

Input size: 32*32, Kernel size: 5*5, padding=2, stride = 1

Output volume size=(32 + 2×2 - 5)/1 + 1 = 32 spatially



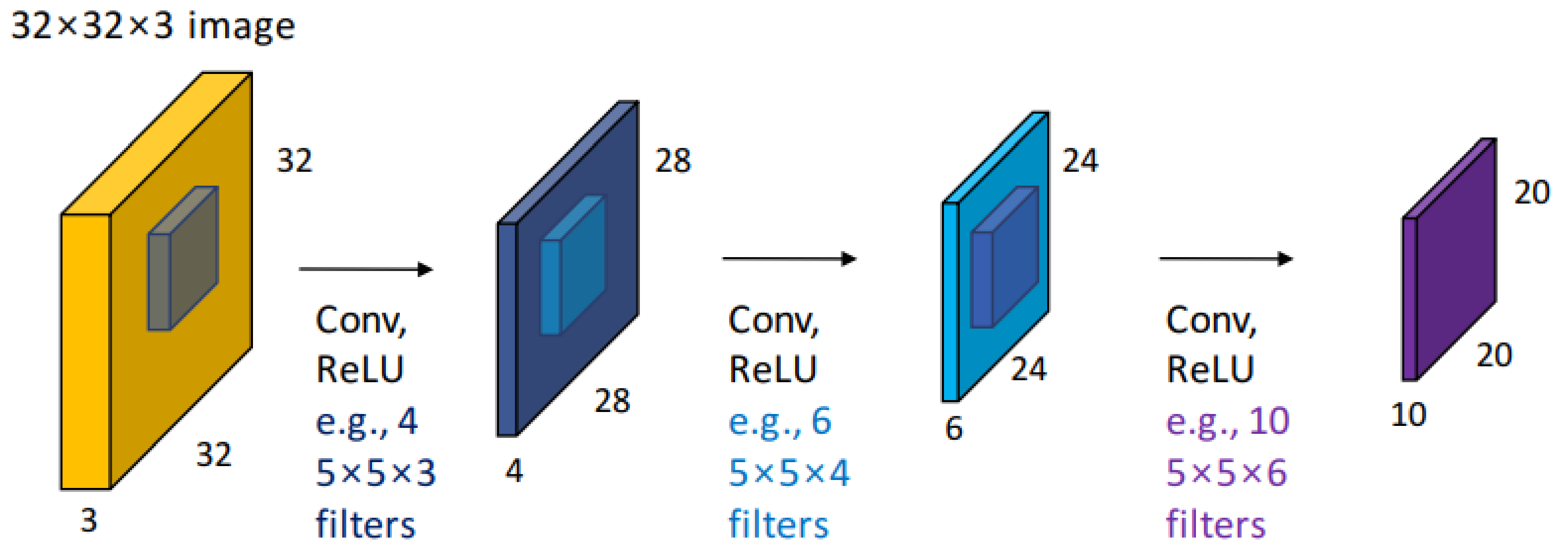
How to compute number of parameters of each layer?
For each filter k*k*3+1 (3: colored image, 1:Bias)
For all filters of the convolution layer: number of filters*number of parameters per layer
In this example:
Each filter has 5*5*3 + 1 (for bias) = 76 parameters
For all 10 filters → 760 parameters

Deep Learning- CNN Convolutional Neural Networks

Convolution

-

Output volume size= N*((W−K+2P)/S+1)

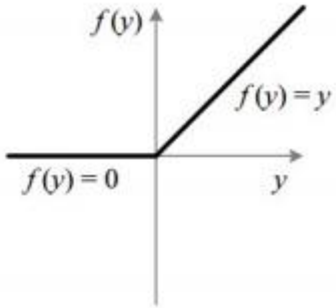


(32-4,32-4,4)

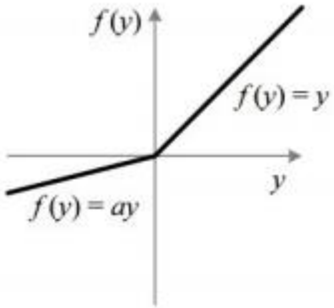
(28-4,28-4,6)

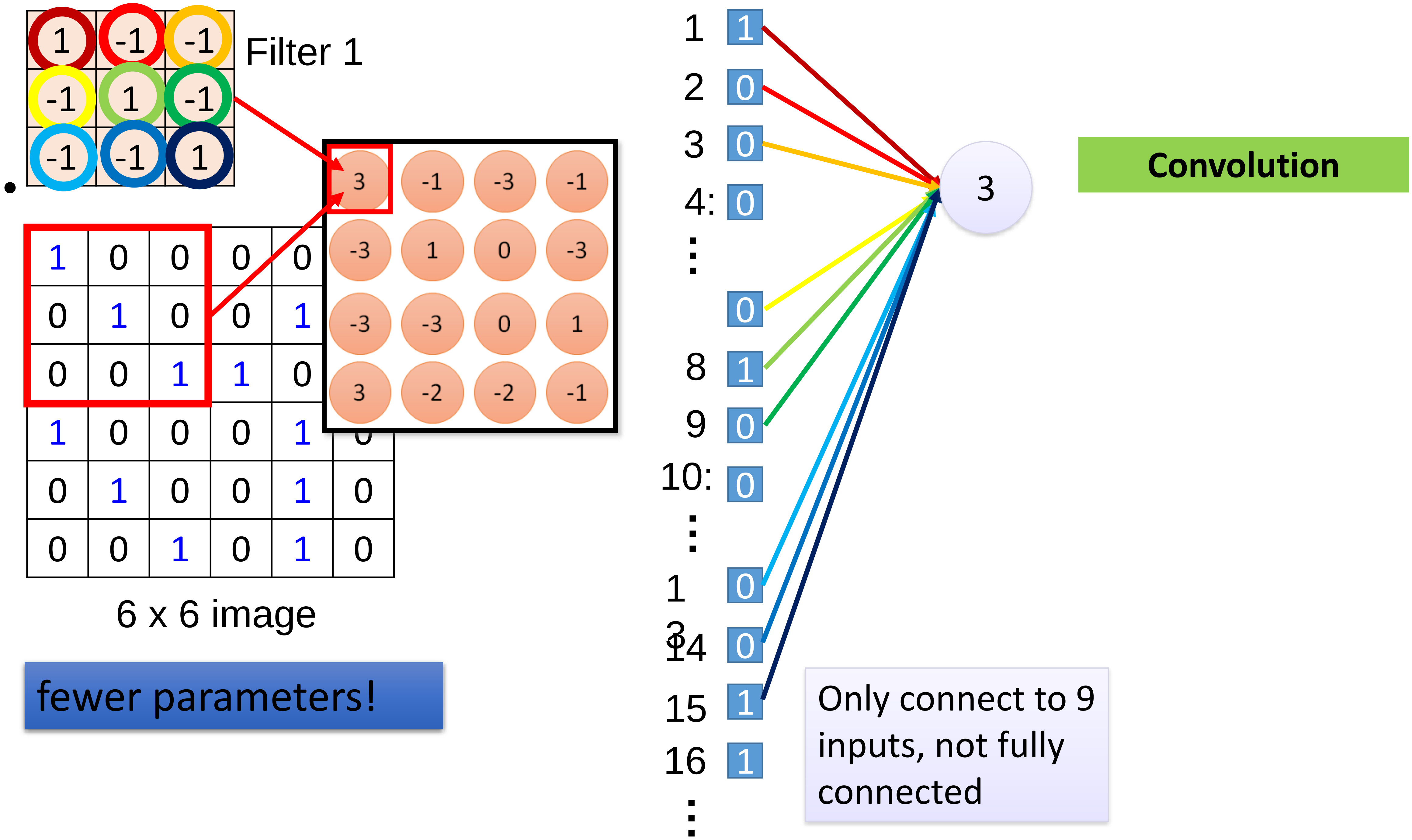
(24-4,24-4,10)

ReLU

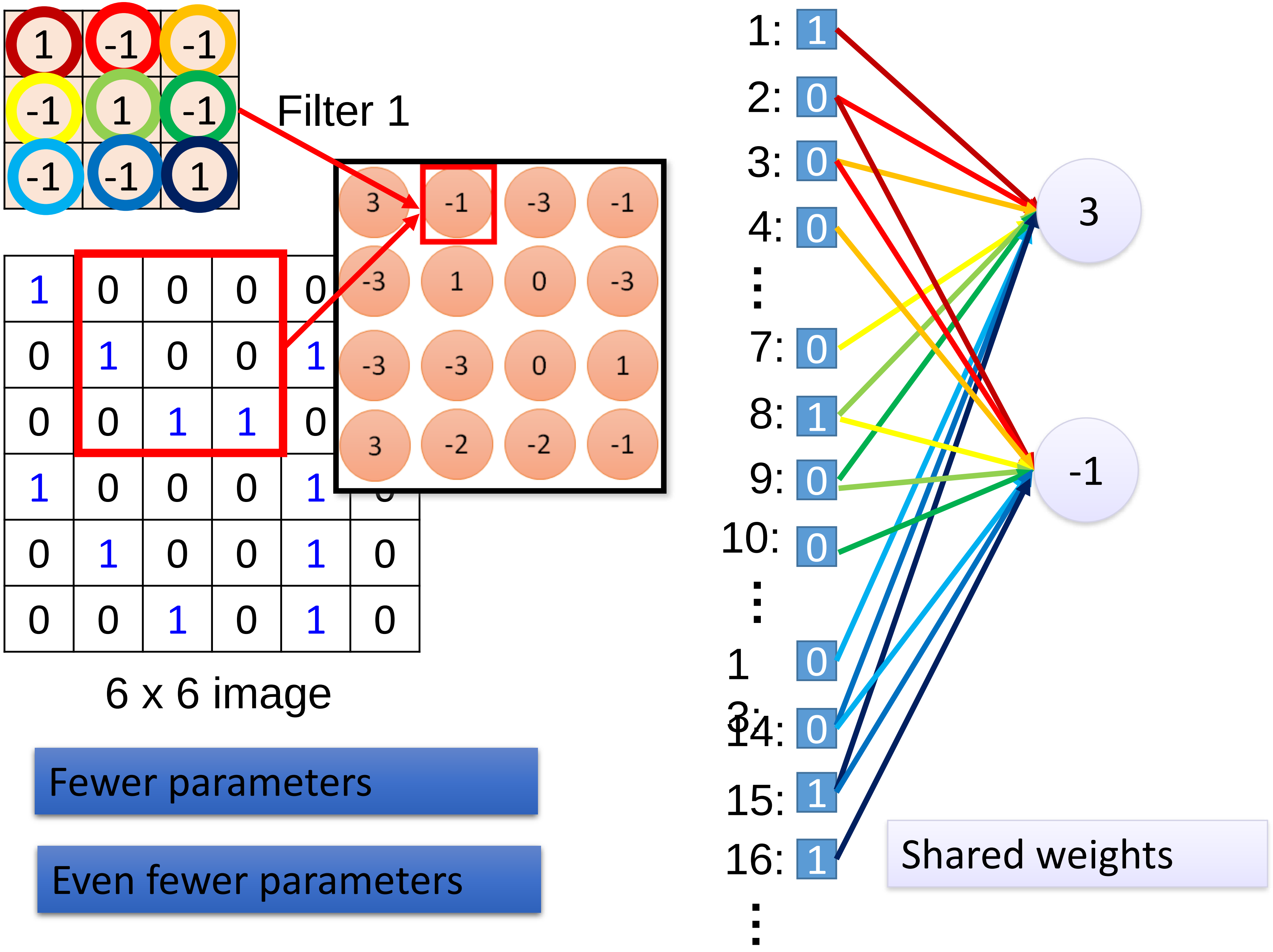


LeakyReLU





Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
fewer parameters!
1 2 3 4: 8 9 10: 13 14 15 16
3
Convolution
Only connect to 9 inputs, not fully connected



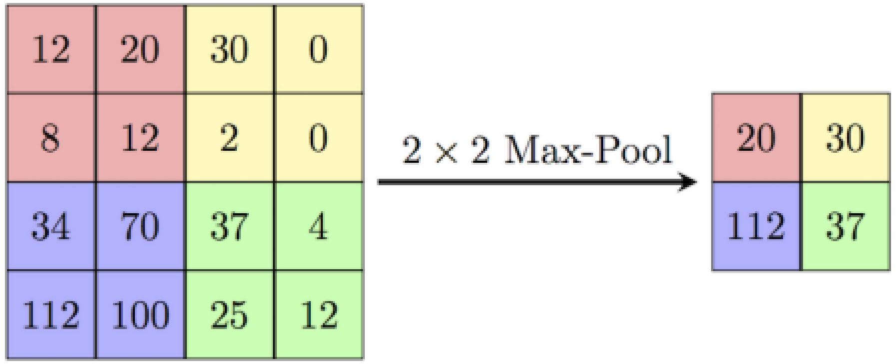
Fewer parameters

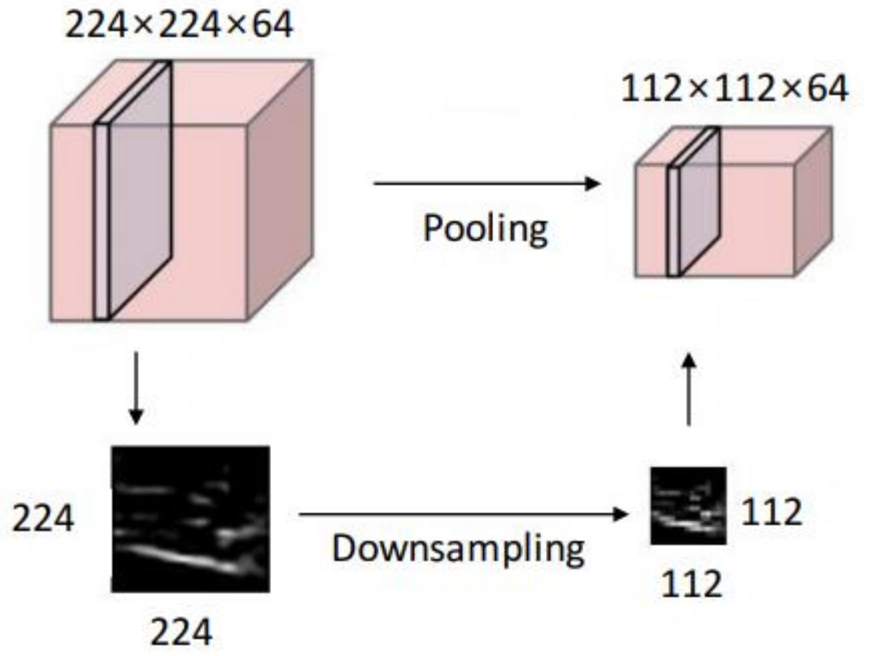
Even fewer parameters

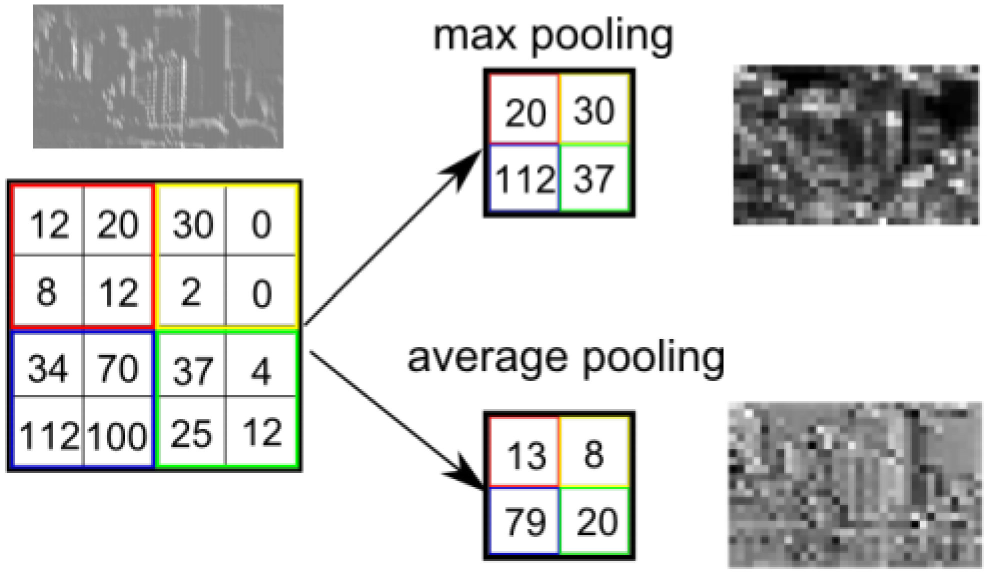
Deep Learning- CNN Convolutional Neural Networks

Pooling

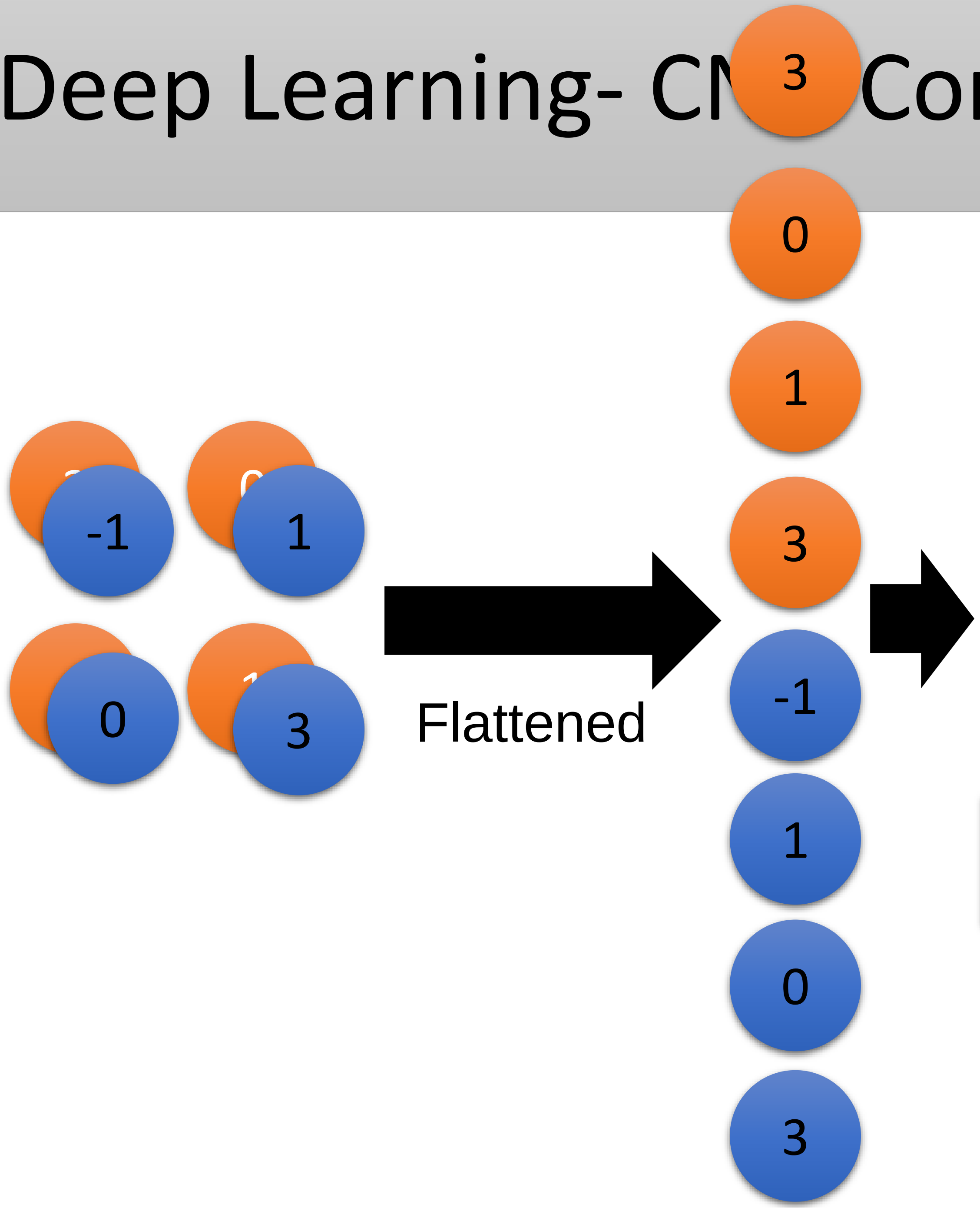
Minimize the activation maps size (lower computations) and maintain the best feature information

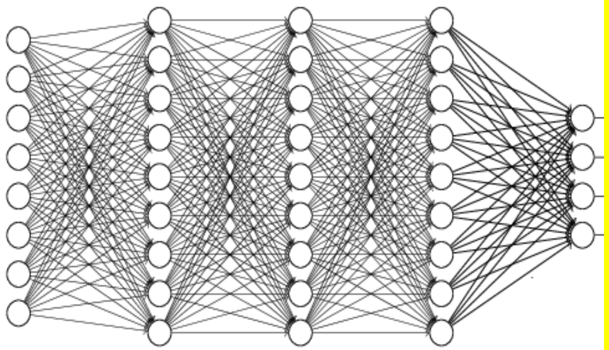






Deep Learning- CN Convolutional Neural Networks



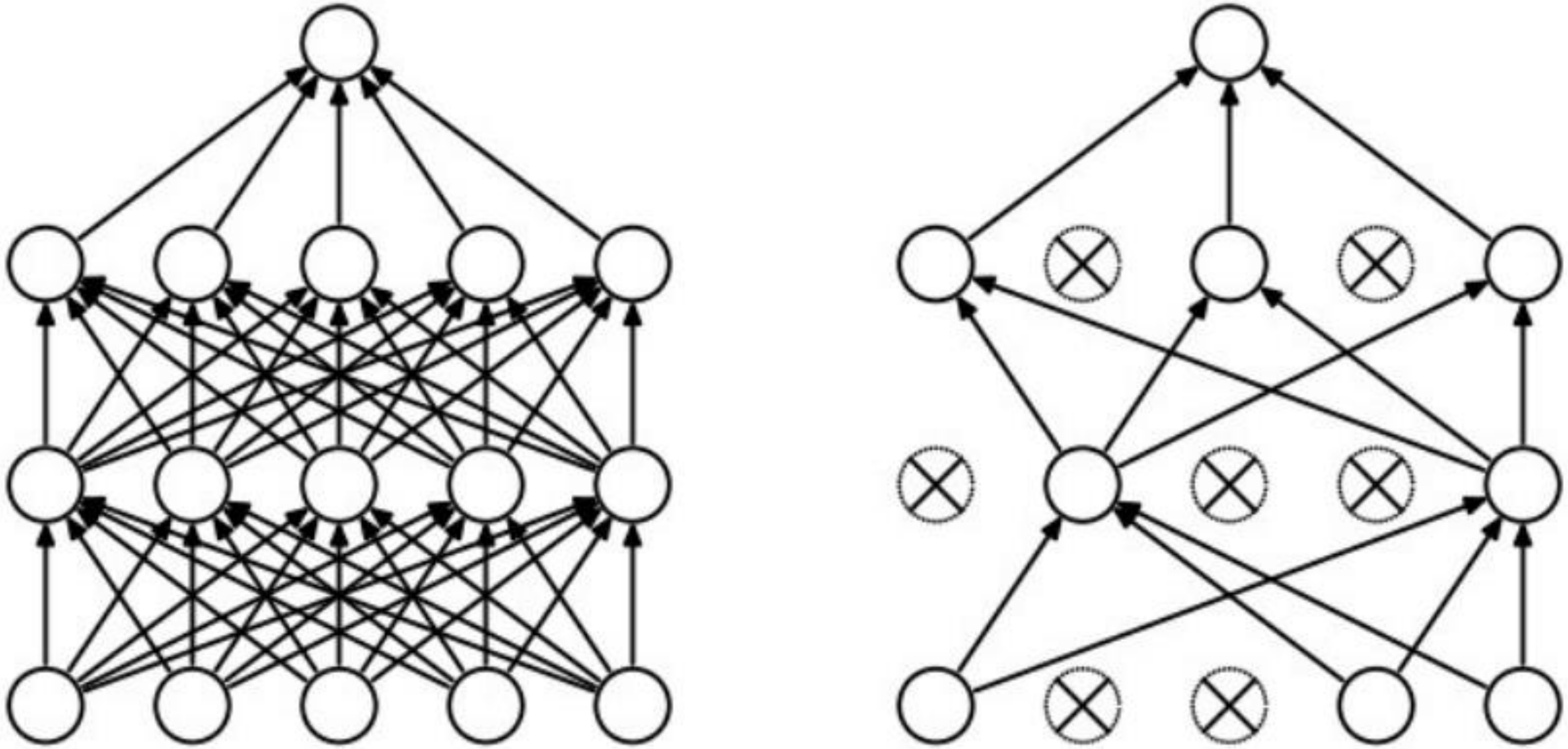


Fully Connected Feedforward network

Flattening and FC

Convert the last activation map of the feature extraction part into 1D vector
And then add fully-connected layers as needed
The final fully-connected layer is the output (classification layer)
Number of neurons in the classification layers equals to the number of required outputs.
Activation function: Sigmoid (binary classification)
Softmax (multi-class classification)

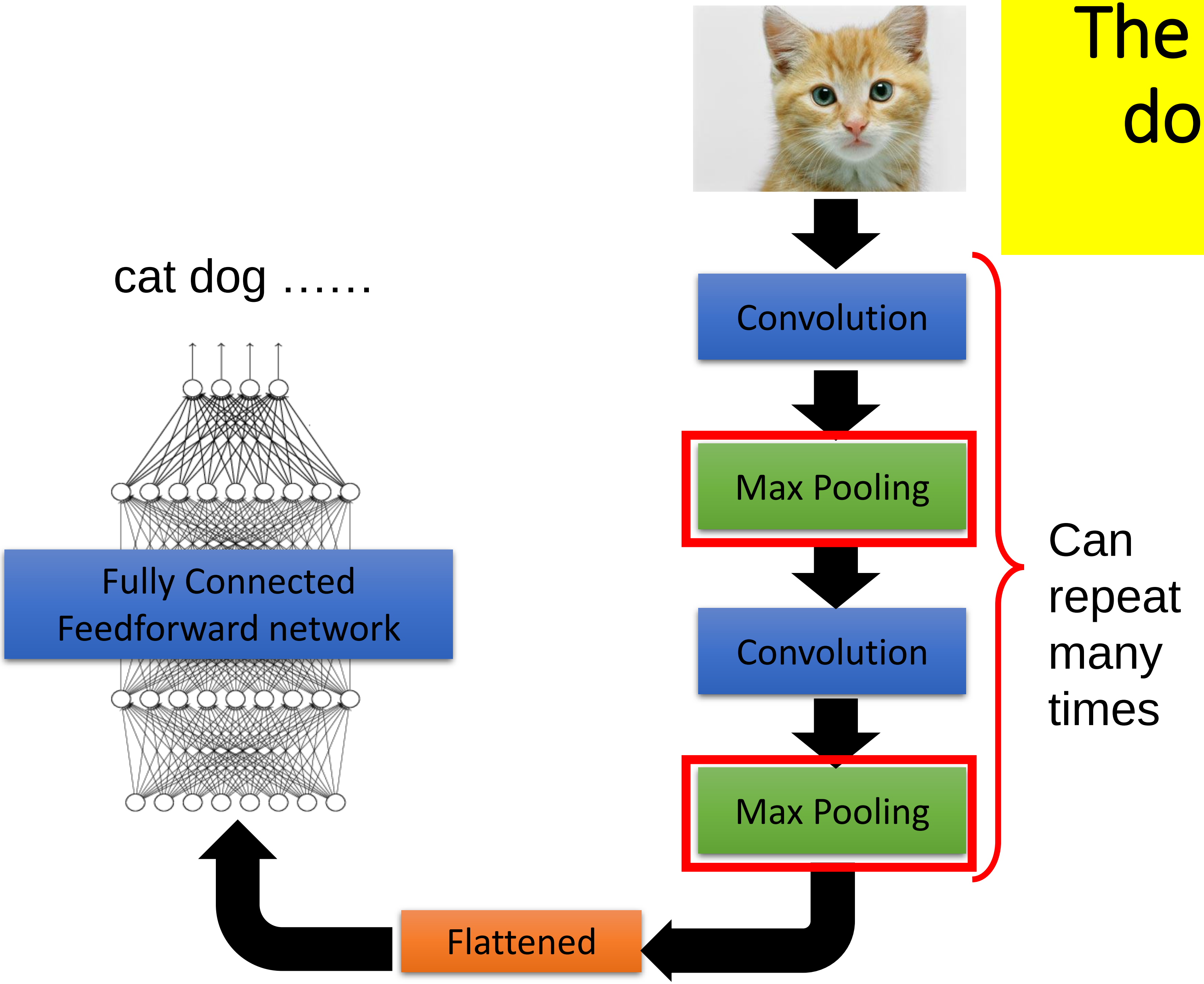
Deep Learning- CNN Convolutional Neural Networks



Prevent overfitting

We can reduce the effect of overfitting by adding nonlinearity to the architecture of the deep network.
In the classification part, we can add the dropout layers
Dropout layers can be inserted between two adjacent dense layers to remove some neurons with a specific rate (for example 25%)
Dropout 25% means dropping out 25% of the neurons of the current layer randomly (set their values to 0).

The whole CNN- Cat dog classification example



Deep Learning- CNN applications

•

Self-driving cars

Detection [Ren et al., 2015]

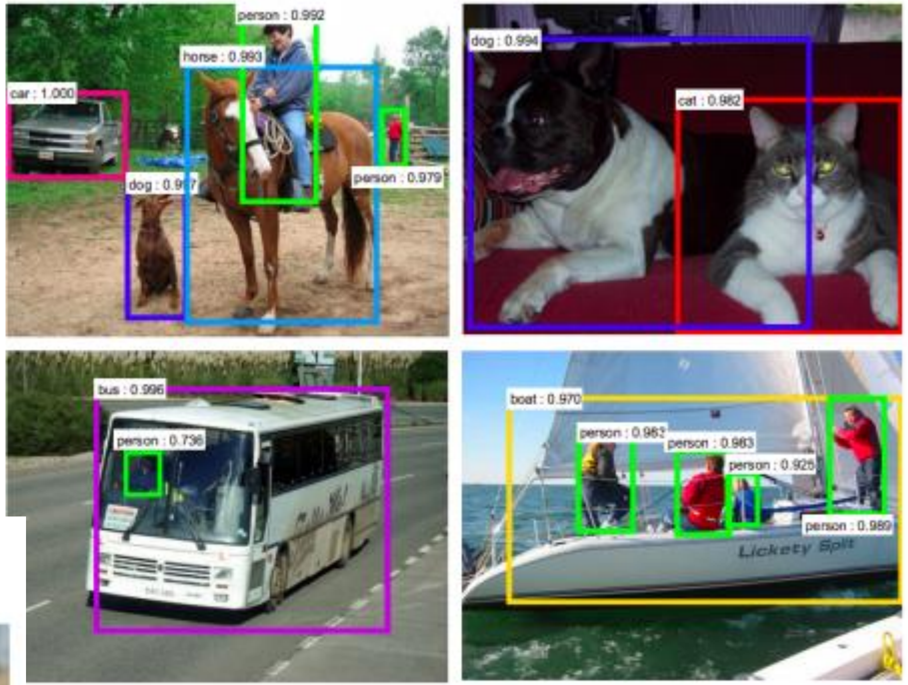


Image Captioning

Image Recognition

No errors

A white teddy bear sitting in the grass

Minor errors

A man in a baseball uniform throwing a ball

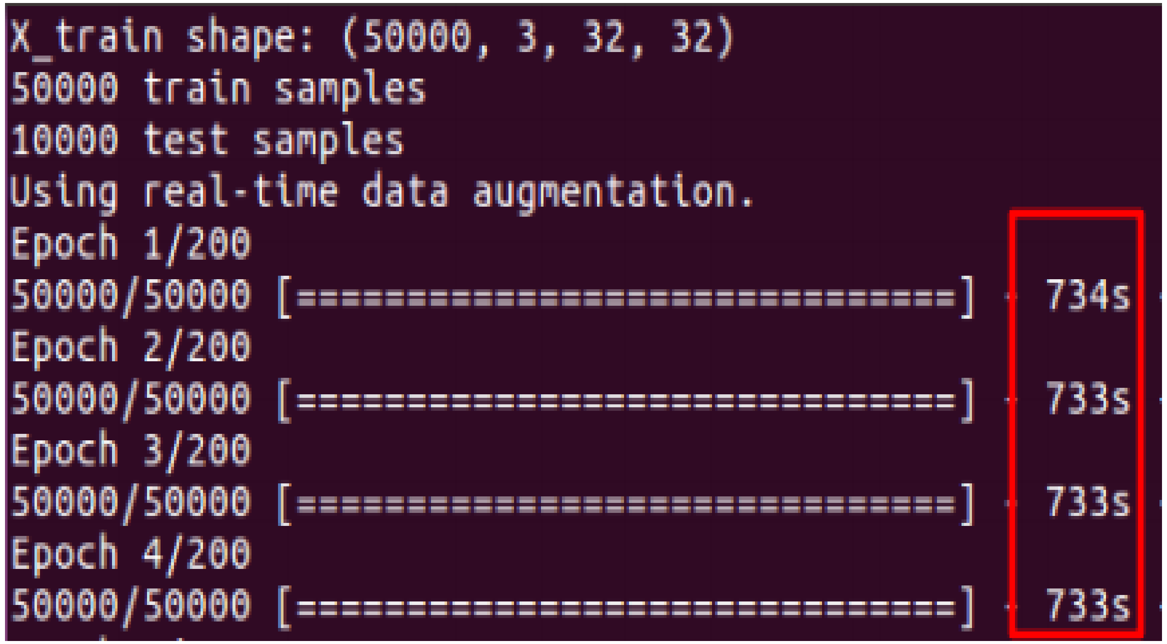
Somewhat related

A woman is holding a cat in her hand

Deep Learning- Why GPU?

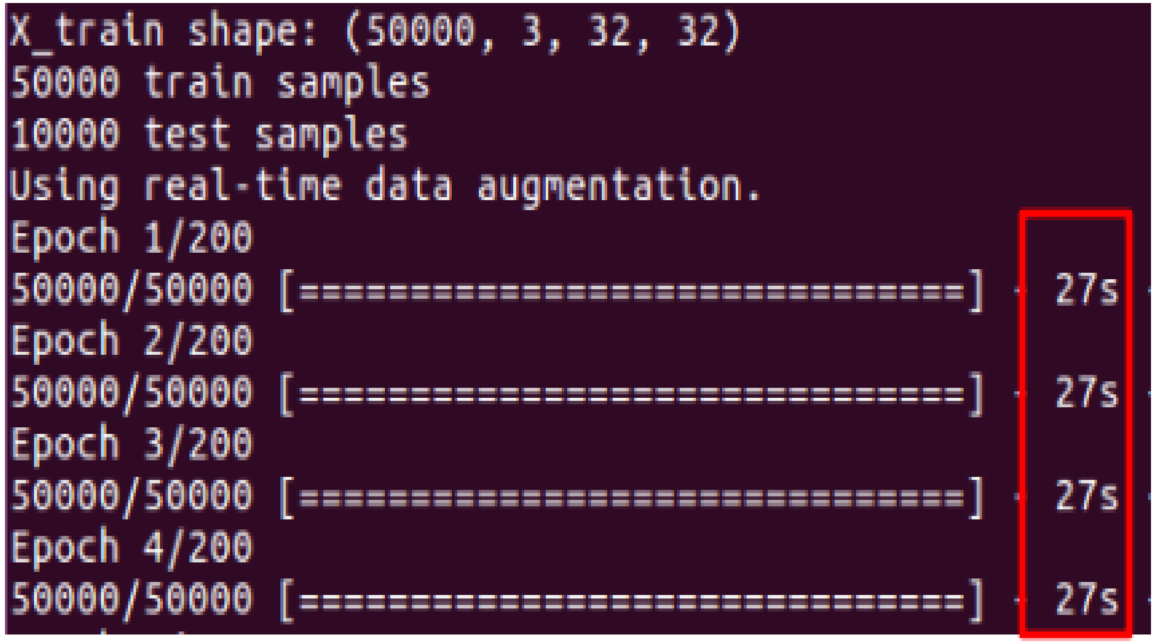
-

GPU has the power of parallel processing allowing to handle batch of images at the same time reducing the training time



Running time **without** GPU

VS

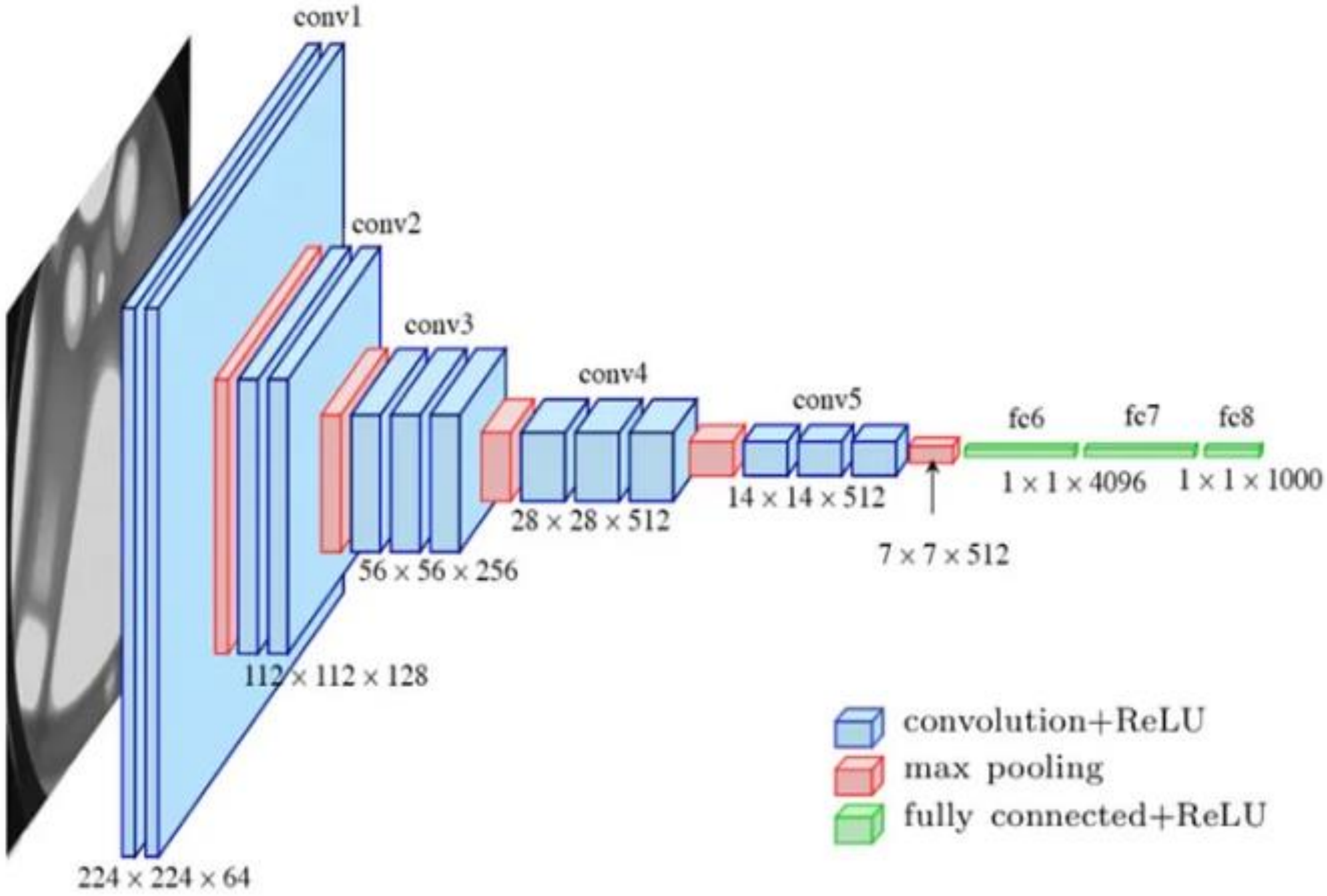


Running time **with** GPU

Some CNN famous examples

VGG16 model

5 Groups of Conv+Maxpool layers
Flattened layer (7*7*515,1) size layer
Fully-connected layer 1*1*4096
Fully-connected layer 1*1*4096
Fully-connected layer 1*1*1000 (1000 classes of ImageNet dataset)

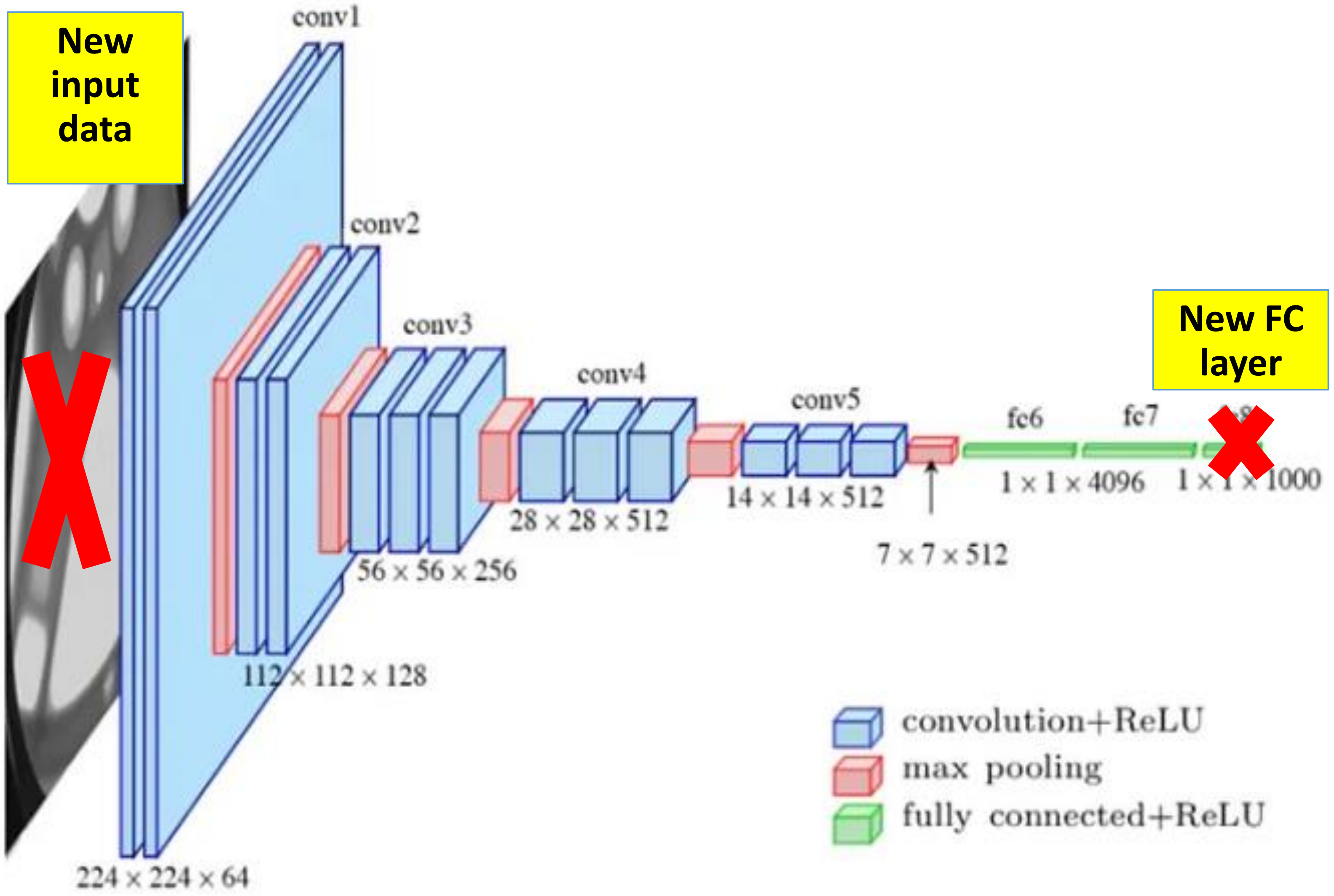


Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition (2014). Source : http://www.cs.toronto.edu/ frossard/post/vgg16/vgg16.png

Some CNN famous examples

VGG16 model

Transfer Learning:
Re-use of the VGG pretrained model (trained on ImageNet dataset) and re-train it on our specific dataset (cancer classification for example)
Two options:
- Maintain architecture with modification of the input data and the classification layer and retrain all layers.
- Maintain architecture with modification of the input data and the classification layer and retrain only the classification part (FC layers) (faster)



Numerical Example

- Using a CNN with input size of 32*32 to be classified into **four** classes (COVID-19 ,Pneumonia, Influenza, Normal) and using a **padding=1, stride=2**, two convolutional layers. Number of filters of the first layer=**8**, and for second layer=**16**. Suppose all filters of size **3*3**, and each conv layer is followed by a **2*2 max pooling** layer and has a **Relu** activation function. The classification part consists of Flatten, Dense layer of **16** neurons, and final classification dense layer. (Note: use floor for your calculations).
- What is the activation maps’ size on the output of each layer (conv and max pool layers).
- What is activation function of the output layer and what are number of neurons?
- To avoid overfitting, what is the required modification of the architecture.
- Suppose we have this kernel and the activation map on the output of the second pooling layer. Apply the convolution to the image. You need to make padding by repeating rows and columns before make the convolution.
- Compute the size of the output of the final max pool, then dense layer then classification layer.

2	0	0
0	0	0
0	0	-2

Filter المرشح

2	2	2	2
0	0	0	0
1	1	1	1
0	0	0	0

الصورة على خرج طبقة Max pooling الثانية

4	4	4	4

الصورة على خرج طبقة Convolution الثالثة

Numerical Example

- **1-**First layer output (W−K+2P)/S+1 = (32-3+2)/2+1= 8*16*16
- Max pooling: 8*8*8
- Second layer output (W−K+2P)/S+1 = (8-3+2)/1+1= 16*4*4
- Max pooling: 16*2*2
- Flatten → 64 (4 samples per kernel)
- Dense → 16
- **2-** Activation function is softmax, number of neurons: 4 since the problem is multi-class classification.
- **3-** Avoid overfitting by adding dropout layers between the dense layers of the classification part to add non-linearity of the model.
- **4-**

2	2	2	2
0	0	0	0
1	1	1	1
0	0	0	0

→ **Padding** →

2	2	2	2	2	2
2	2	2	2	2	2
0	0	0	0	0	0
1	1	1	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0

Activation map after padding

2	0	0
0	0	0
0	0	-2

Filter (kernel)

4	4	4	4
2	2	2	2
0	0	0	0
2	2	2	2

Final convolution (new activation map)

Numerical Example

- **5- output on the max pooling**

4	4	4	4
2	2	2	2
0	0	0	0
2	2	2	2

Final convolution (new activation map)

4	4
2	2

Output of the third Max pool layer

4	4	2	2

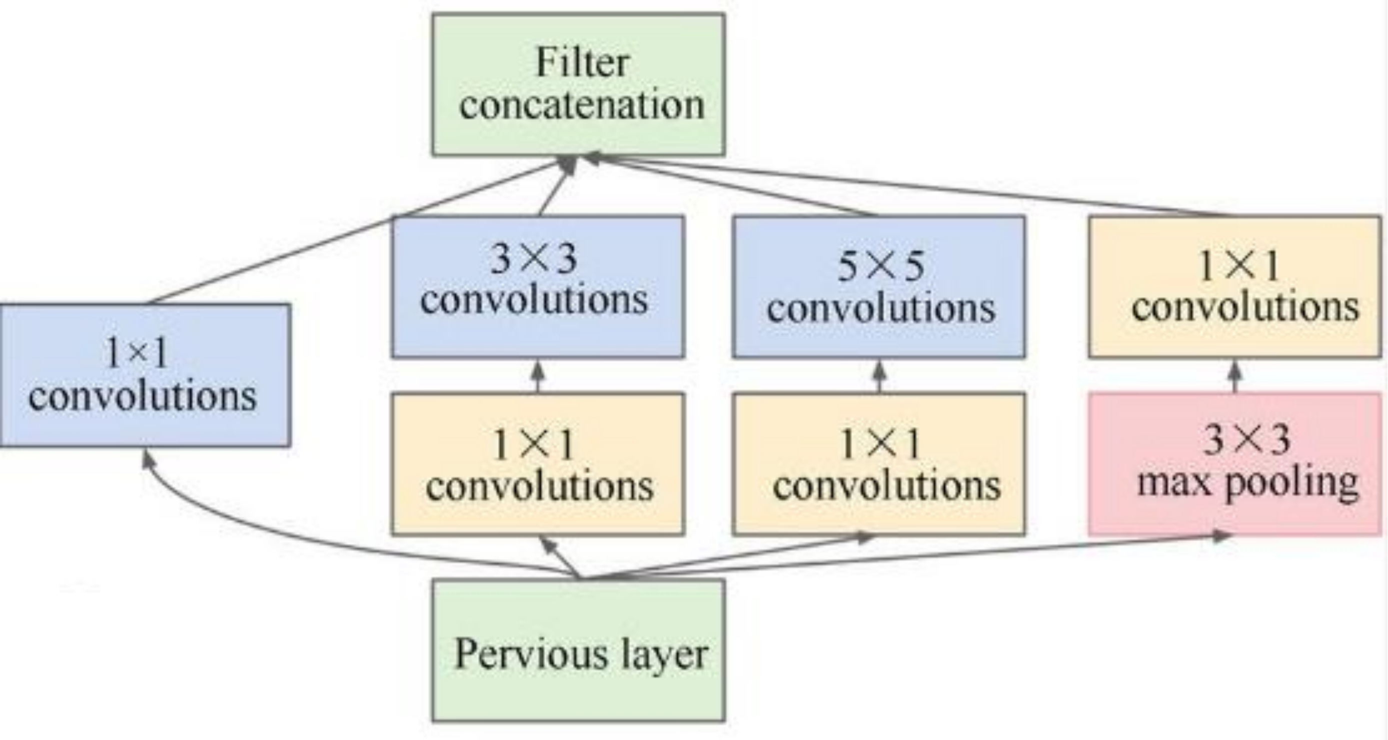
Flatten

Other CNN famous examples

Inception

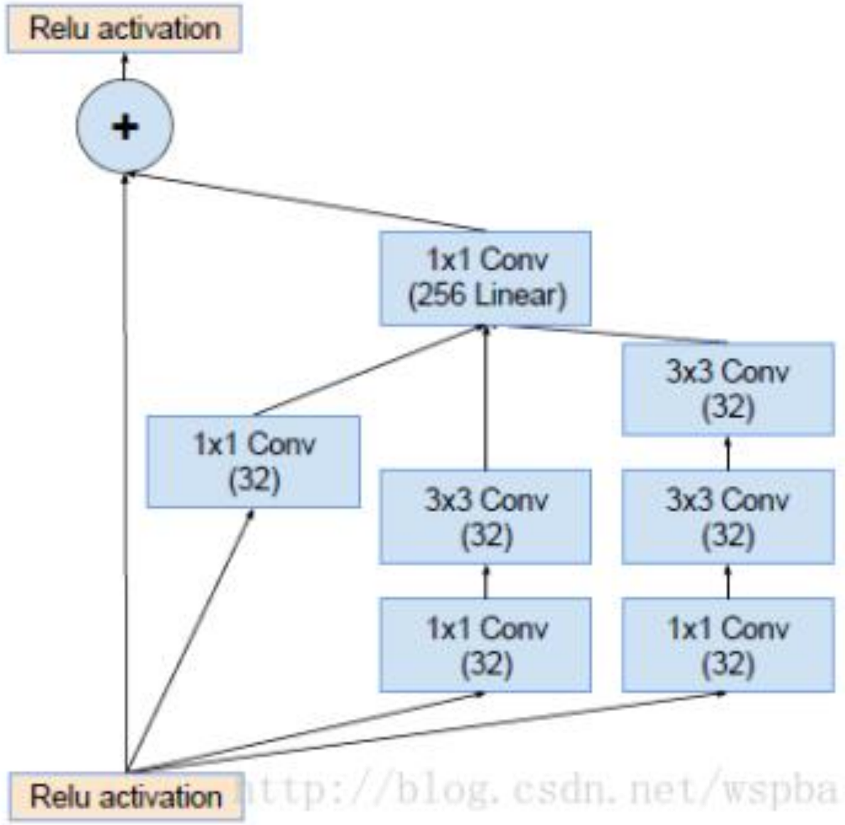
Inception modules, Szegedy et a

Parallel architecture contains different convolution sizes (1*1, 3*3, 5*5) and then they are concatenated to get different scaling features of the same input image.



InceptionResNet

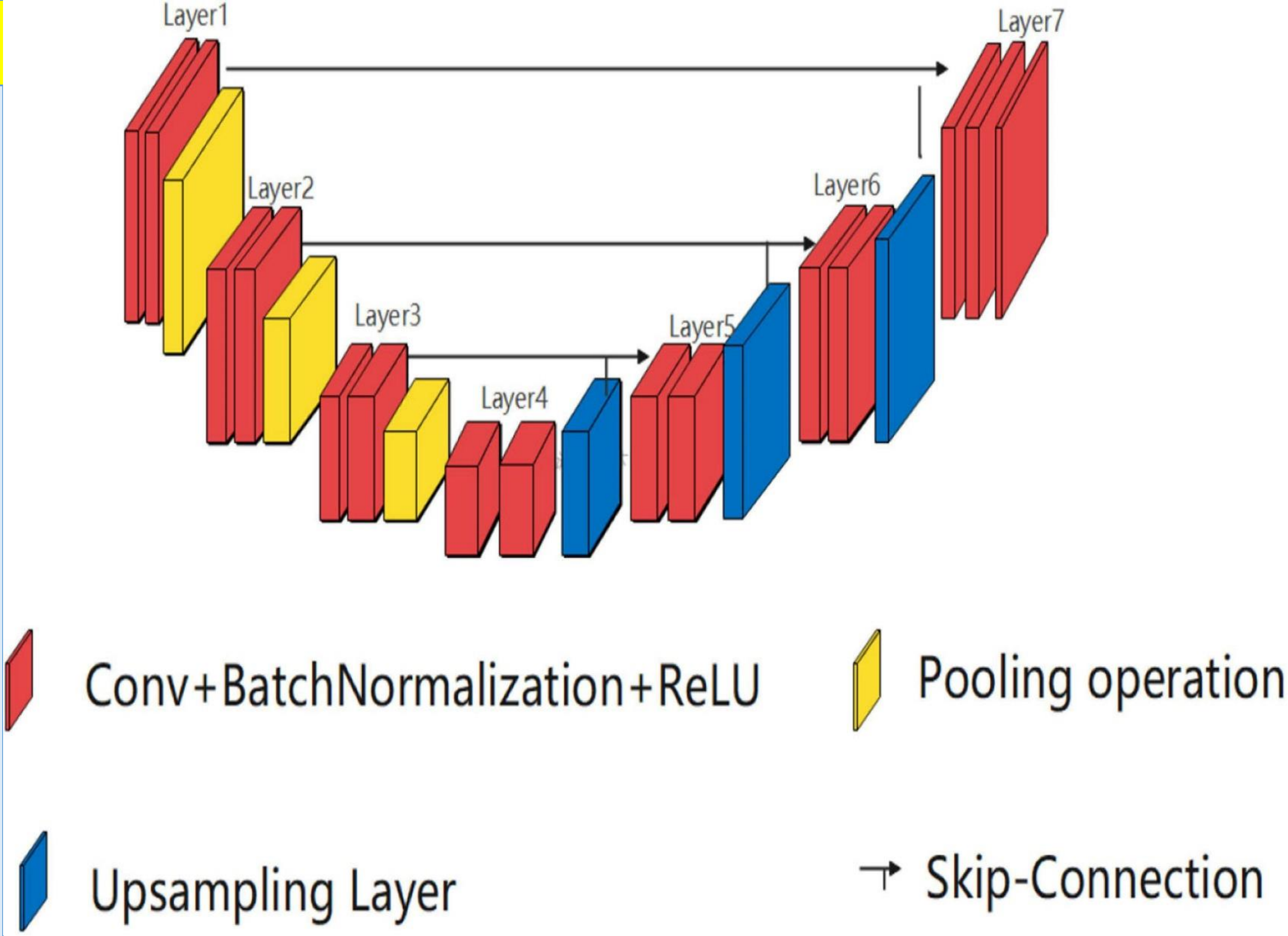
Add identity connections (called residual connections) that escape one or more convolution layers and this can allow more deep (more hidden layers) without problem of gradient vanishing.



Other CNN famous examples

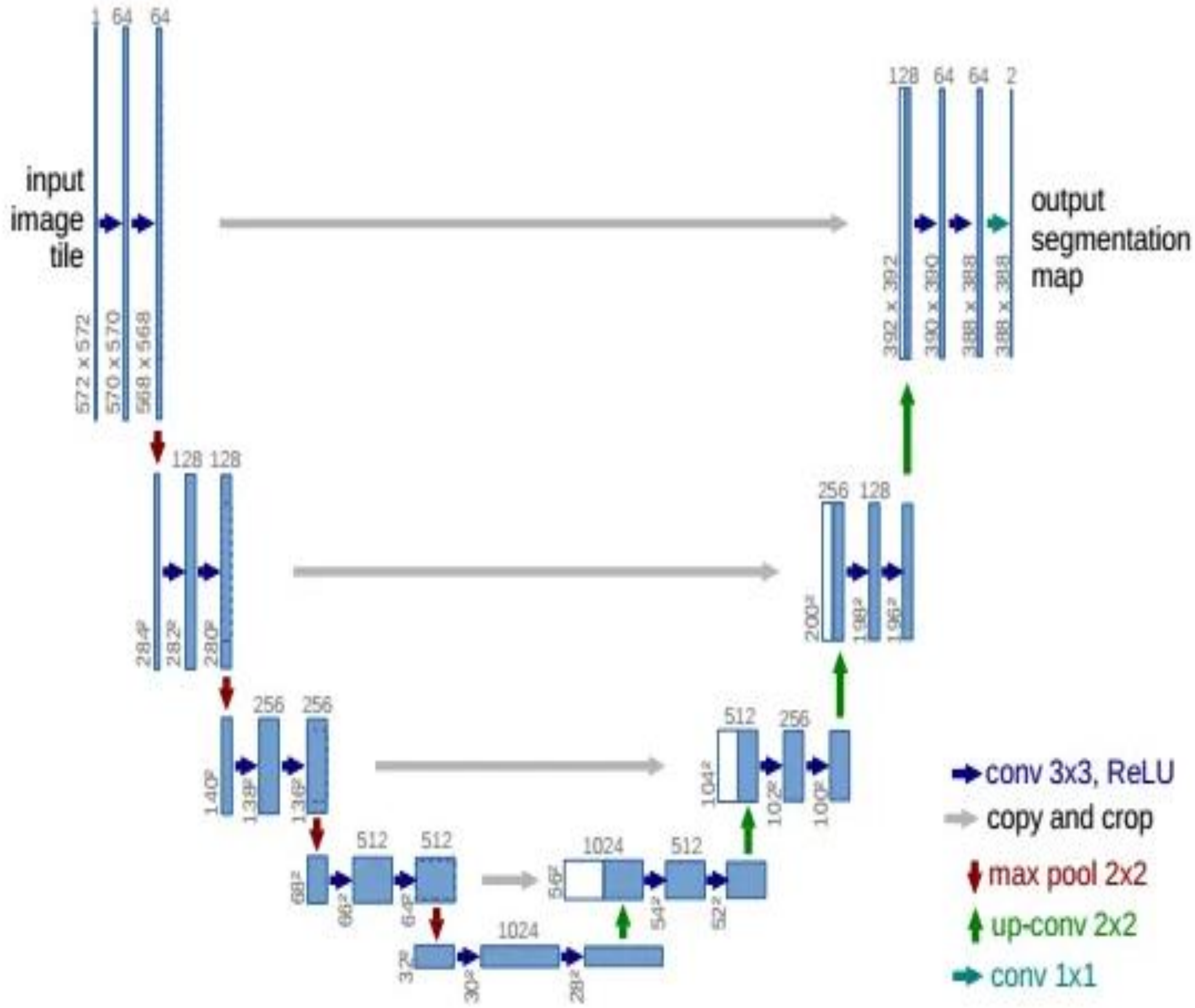
UNet

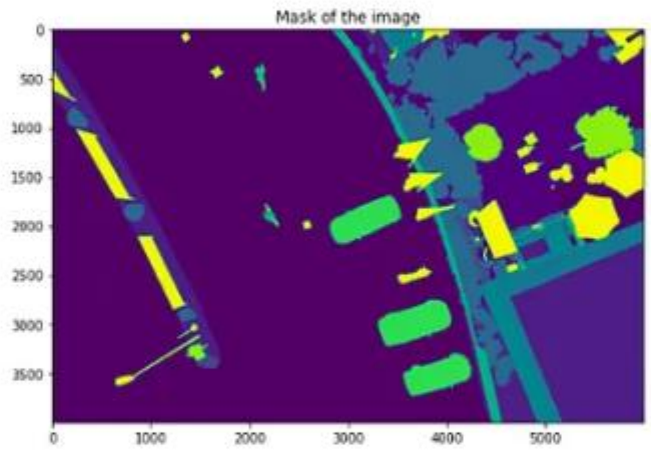
- The U-Net architecture, introduced in **2015**, is a popular model for image segmentation tasks. It has a **U-shaped design** with two main parts:
- **Encoder (Contraction Path):**
  - This part captures context and extracts features.
  - Consists of repeated **convolutional layers** followed by **max-pooling** for downsampling.
- **Decoder (Expansion Path):**
  - This reconstructs the spatial resolution to generate the segmentation map.
  - Includes **upsampling layers** (transpose convolutions) and **skip connections** from the encoder, which provide fine-grained details.



Other CNN famous examples

Better illustration of UNet





Some terms in DL field

Optimizer

An **optimizer** is an algorithm or method used to adjust the model's parameters (e.g., weights and biases) to minimize the **loss function (error)** during training. It determines how the model learns from the data by updating these parameters iteratively based on the gradients calculated during learning algorithm.
The most famous optimizer is "Adam" which uses the gradient descent algorithm

What is learning algorithm of DL models?

Backpropagation (used the gradient descent learning)

What is loss function?

- A **loss function** quantifies how well (or poorly) the model's predictions match the actual target values. The optimizer uses the gradients of the loss function to update the model parameters.

Binary classification

Binary Cross-Entropy:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Cross-Entropy Loss:

Multi-class classification

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log(\hat{y}_{ij})$$