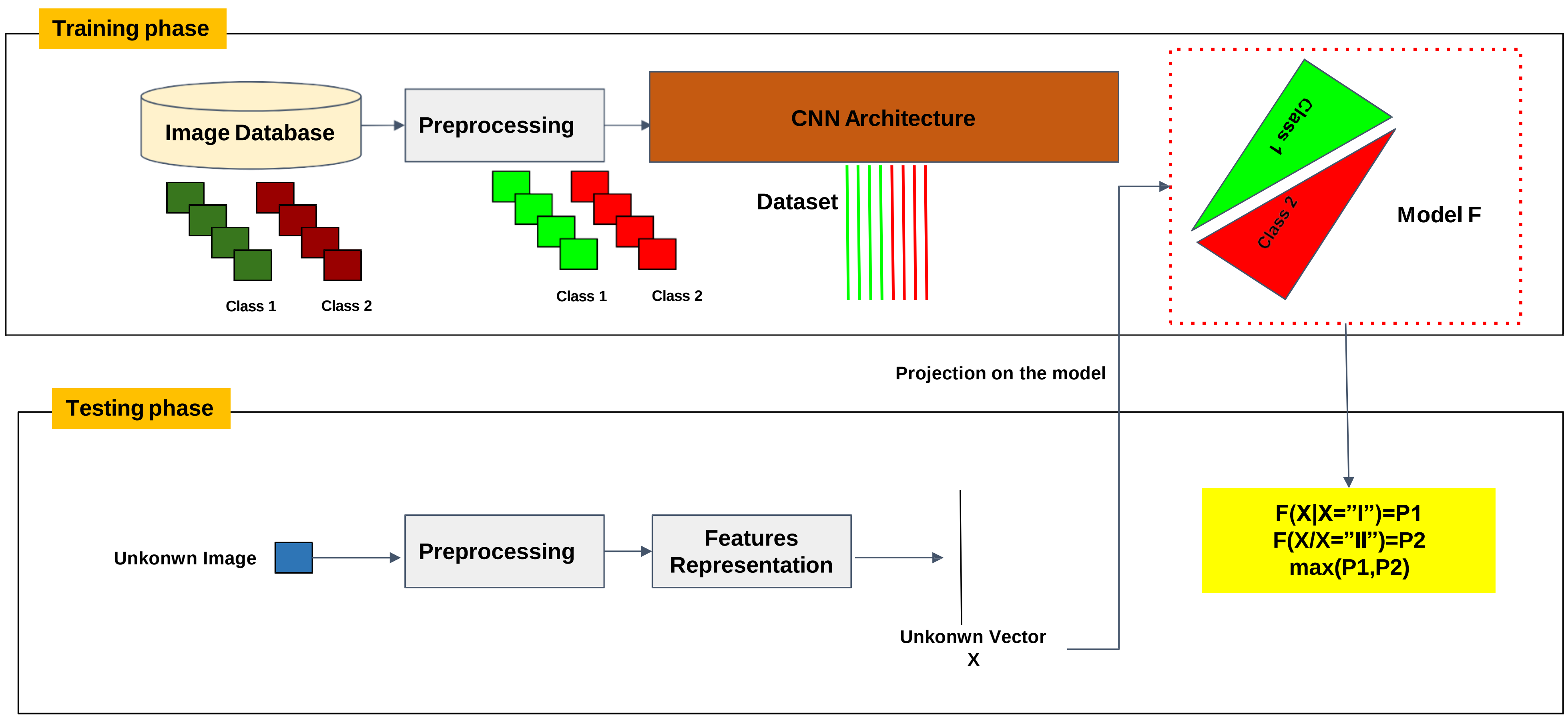
Deep Learning for Natural Language Processing

Convolutional Neural Network (CNN)

Long Short Term Memory (LSTM)

Kais Ameur

Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning



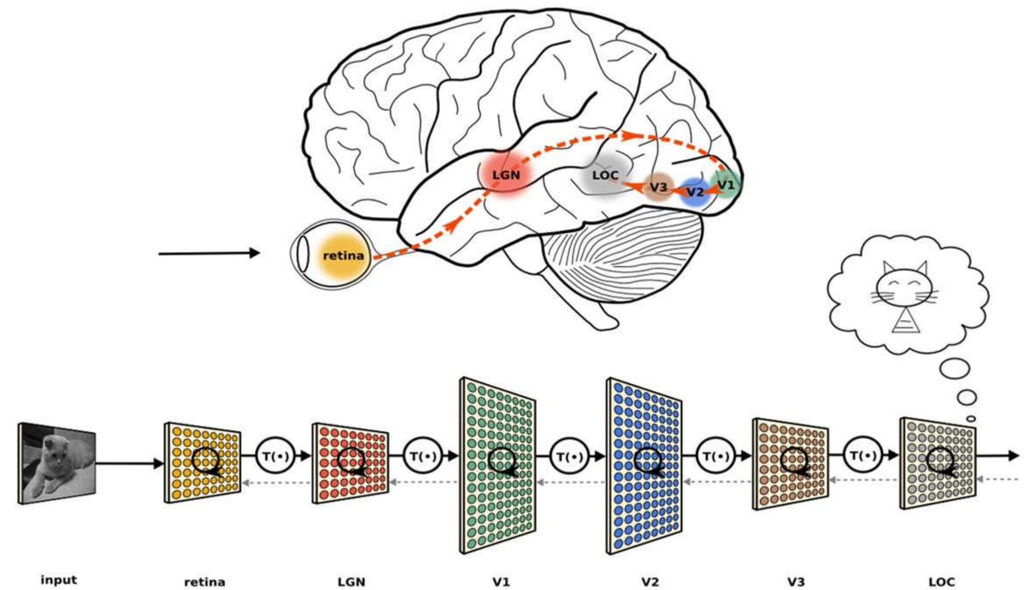
Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Hubel and Wiesel Experiments (1962)

- ➔ Insert electrodes into specific parts of the visual cortex of the cat;
- ➔ Measurement of activation when the cat saw some basic shapes;
- ➔ The visual Cortex is the responsible of Perception;
- ➔ A cell of Neurons which are organized in Column

LeCun, Bottou, Bengio and Haffner (1998)

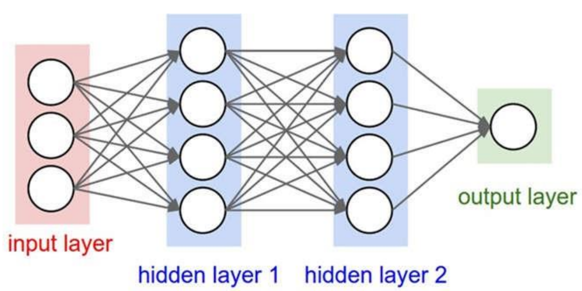
- ➔ Introduction of Convolutional Neural Network (CNN);
- ➔ Inspiration from Visual Cortex
- ➔ Organization in Column of each Layer of the Neural Network Architecture

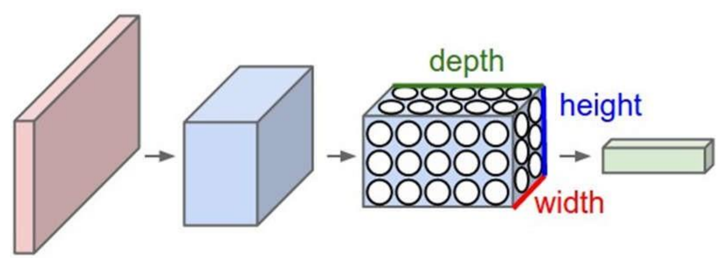


Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Convolutional Neural Network (CNN)

- **A Feedforward Neural Network;**
- **Emulate the Visual Cortex in the Visual Perception Task;**
- **CNN have specific Layers that encodes the properties of Images**
  - Low Level Features;
  - Middle Level Features;
  - High Level Features.
- **MLP Vs CNN**
  - **MLP: Organization of Neurons into sequence of Layers**
  - **CNN: Organization of Neurons into sequence of 3D-Layers (called *Depth*)**





Convolutional Neural Network (CNN) Architecture:
Learning & Transfer Learning

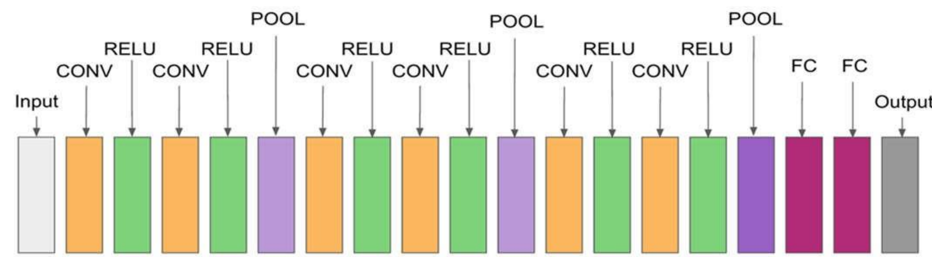
Convolutional Neural Network (CNN) defines specific layers:

➔ **Convolutional Layer (CL)**
- ◆ Convolution Operator
- ◆ Linear Representation (Sum of Product)

➔ **Pooling Layer (PL)**
- ◆ Downsampling
  - Average Operator
  - Max Operator

➔ **ReLU Layer (RL)**
- ◆ Help the optimization of the Gradient Descent
- ◆ To introduce the Non-Linear Representation

➔ **Fully Connected Layer (FC)**
- ◆ Emulate MLP Classifiers
- ◆ Classification Task



Example of a CNN Architecture

Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Convolutional Layer (CL)

- Convolutional Layer
  - based on Convolution Operator;
  - The most important concept in Signal Processing;
  - Construct the Output of any System knowing its inputs and its Impulse Response.

- From 1D to 2D Convolution
  - Convolution 1D for Signal Processing;
  - Convolution 2D for Image Processing.

- Convolutional Kernel
  - Used to be applied on Image;
  - examples of kernel are used in Image processing
    - Edge Detection;
    - Blurring Image;
    - Smoothing, etc.

Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Convolutional Layer (CL)

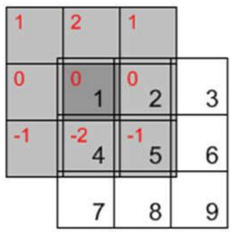
1	2	3
4	5	6
7	8	9

Input x

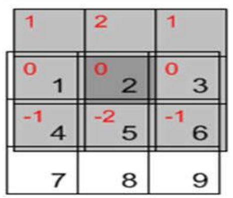
*

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

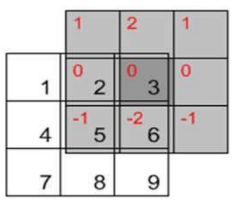
Kernel h



$$\begin{aligned} y[0,0] &= x[-1,-1]\cdot h[1,1] + x[0,-1]\cdot h[0,1] + x[1,-1]\cdot h[-1,1] \\ &+ x[-1,0]\cdot h[1,0] + x[0,0]\cdot h[0,0] + x[1,0]\cdot h[-1,0] \\ &+ x[-1,1]\cdot h[1,-1] + x[0,1]\cdot h[0,-1] + x[1,1]\cdot h[-1,-1] \\ &= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 0\cdot 0 + 1\cdot 0 + 2\cdot 0 + 0\cdot(-1) + 4\cdot(-2) + 5\cdot(-1) = -13 \end{aligned}$$



$$\begin{aligned} y[1,0] &= x[0,-1]\cdot h[1,1] + x[1,-1]\cdot h[0,1] + x[2,-1]\cdot h[-1,1] \\ &+ x[0,0]\cdot h[1,0] + x[1,0]\cdot h[0,0] + x[2,0]\cdot h[-1,0] \\ &+ x[0,1]\cdot h[1,-1] + x[1,1]\cdot h[0,-1] + x[2,1]\cdot h[-1,-1] \\ &= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 1\cdot 0 + 2\cdot 0 + 3\cdot 0 + 4\cdot(-1) + 5\cdot(-2) + 6\cdot(-1) = -20 \end{aligned}$$



$$\begin{aligned} y[2,0] &= x[1,-1]\cdot h[1,1] + x[2,-1]\cdot h[0,1] + x[3,-1]\cdot h[-1,1] \\ &+ x[1,0]\cdot h[1,0] + x[2,0]\cdot h[0,0] + x[3,0]\cdot h[-1,0] \\ &+ x[1,1]\cdot h[1,-1] + x[2,1]\cdot h[0,-1] + x[3,1]\cdot h[-1,-1] \\ &= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 2\cdot 0 + 3\cdot 0 + 0\cdot 0 + 5\cdot(-1) + 6\cdot(-2) + 0\cdot(-1) = -17 \end{aligned}$$

-13	-20	-17

Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Convolutional Layer (CL)

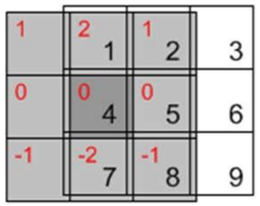
1	2	3
4	5	6
7	8	9

Input x

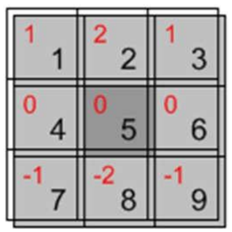
*

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

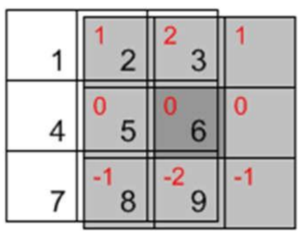
Kernel h



$$\begin{aligned} y[0,1] &= x[-1,0]\cdot h[1,1] + x[0,0]\cdot h[0,1] + x[1,0]\cdot h[-1,1] \\ &\quad + x[-1,1]\cdot h[1,0] + x[0,1]\cdot h[0,0] + x[1,1]\cdot h[-1,0] \\ &\quad + x[-1,2]\cdot h[1,-1] + x[0,2]\cdot h[0,-1] + x[1,2]\cdot h[-1,-1] \\ &= 0\cdot 1 + 1\cdot 2 + 2\cdot 1 + 0\cdot 0 + 4\cdot 0 + 5\cdot 0 + 0\cdot(-1) + 7\cdot(-2) + 8\cdot(-1) = -18 \end{aligned}$$



$$\begin{aligned} y[1,1] &= x[0,0]\cdot h[1,1] + x[1,0]\cdot h[0,1] + x[2,0]\cdot h[-1,1] \\ &\quad + x[0,1]\cdot h[1,0] + x[1,1]\cdot h[0,0] + x[2,1]\cdot h[-1,0] \\ &\quad + x[0,2]\cdot h[1,-1] + x[1,2]\cdot h[0,-1] + x[2,2]\cdot h[-1,-1] \\ &= 1\cdot 1 + 2\cdot 2 + 3\cdot 1 + 4\cdot 0 + 5\cdot 0 + 6\cdot 0 + 7\cdot(-1) + 8\cdot(-2) + 9\cdot(-1) = -24 \end{aligned}$$



$$\begin{aligned} y[2,1] &= x[1,0]\cdot h[1,1] + x[2,0]\cdot h[0,1] + x[3,0]\cdot h[-1,1] \\ &\quad + x[1,1]\cdot h[1,0] + x[2,1]\cdot h[0,0] + x[3,1]\cdot h[-1,0] \\ &\quad + x[1,2]\cdot h[1,-1] + x[2,2]\cdot h[0,-1] + x[3,2]\cdot h[-1,-1] \\ &= 2\cdot 1 + 3\cdot 2 + 0\cdot 1 + 5\cdot 0 + 6\cdot 0 + 0\cdot 0 + 8\cdot(-1) + 9\cdot(-2) + 0\cdot(-1) = -18 \end{aligned}$$

-13	-20	-17
-18	-24	-18

Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Convolutional Layer (CL)

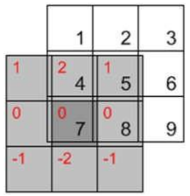
1	2	3
4	5	6
7	8	9

Input x

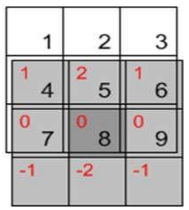
*

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

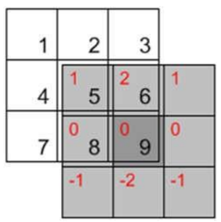
Kernel h



$$
\begin{aligned}
y[0,2] &= x[-1,1]\cdot h[1,1] + x[0,1]\cdot h[0,1] + x[1,1]\cdot h[-1,1] \\
&\quad + x[-1,2]\cdot h[1,0] + x[0,2]\cdot h[0,0] + x[1,2]\cdot h[-1,0] \\
&\quad + x[-1,3]\cdot h[1,-1] + x[0,3]\cdot h[0,-1] + x[1,3]\cdot h[-1,-1] \\
&= 0\cdot 1 + 4\cdot 2 + 5\cdot 1 + 0\cdot 0 + 7\cdot 0 + 8\cdot 0 + 0\cdot(-1) + 0\cdot(-2) + 0\cdot(-1) = 13
\end{aligned}
$$



$$
\begin{aligned}
y[1,2] &= x[0,1]\cdot h[1,1] + x[1,1]\cdot h[0,1] + x[2,1]\cdot h[-1,1] \\
&\quad + x[0,2]\cdot h[1,0] + x[1,2]\cdot h[0,0] + x[2,2]\cdot h[-1,0] \\
&\quad + x[0,3]\cdot h[1,-1] + x[1,3]\cdot h[0,-1] + x[2,3]\cdot h[-1,-1] \\
&= 4\cdot 1 + 5\cdot 2 + 6\cdot 1 + 7\cdot 0 + 8\cdot 0 + 9\cdot 0 + 0\cdot(-1) + 0\cdot(-2) + 0\cdot(-1) = 20
\end{aligned}
$$



$$
\begin{aligned}
y[2,2] &= x[1,1]\cdot h[1,1] + x[2,1]\cdot h[0,1] + x[3,1]\cdot h[-1,1] \\
&\quad + x[1,2]\cdot h[1,0] + x[2,2]\cdot h[0,0] + x[3,2]\cdot h[-1,0] \\
&\quad + x[1,3]\cdot h[1,-1] + x[2,3]\cdot h[0,-1] + x[3,3]\cdot h[-1,-1] \\
&= 5\cdot 1 + 6\cdot 2 + 0\cdot 1 + 8\cdot 0 + 9\cdot 0 + 0\cdot 0 + 0\cdot(-1) + 0\cdot(-2) + 0\cdot(-1) = 17
\end{aligned}
$$

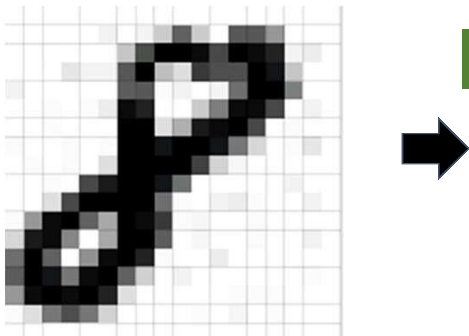
-13	-20	-17
-18	-24	-18
13	20	17

Convolutional Neural Network (CNN) Architecture:
Learning & Transfer Learning

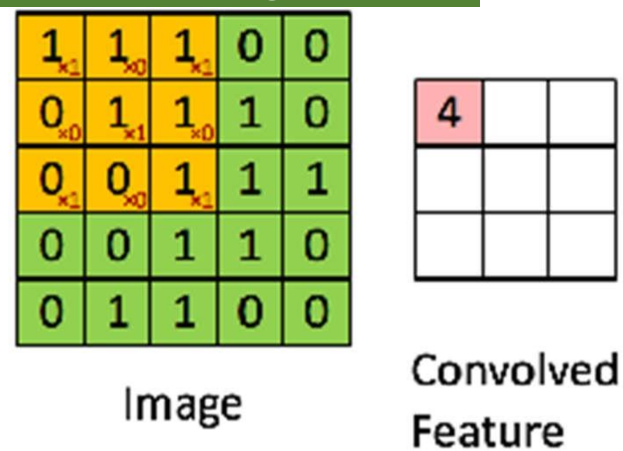
Convolutional Layer (CL)

Let fix our kernel of convolution h

1	0	1
0	1	0
1	0	1



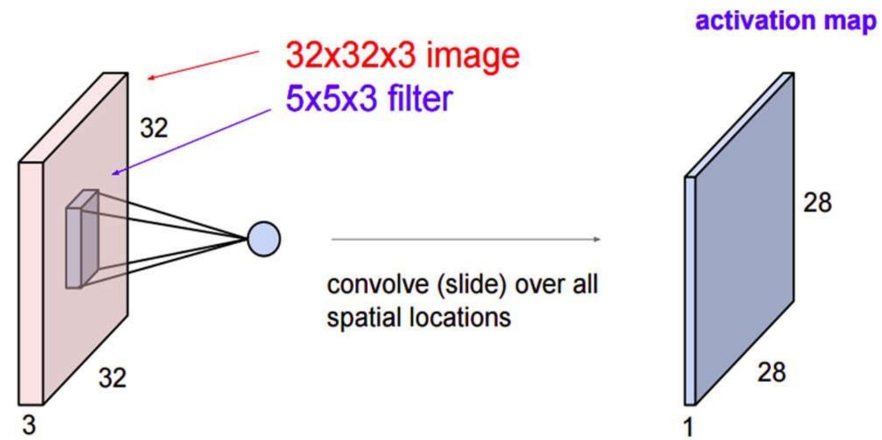
Zero Padding



Sliding Window (Stride)

Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

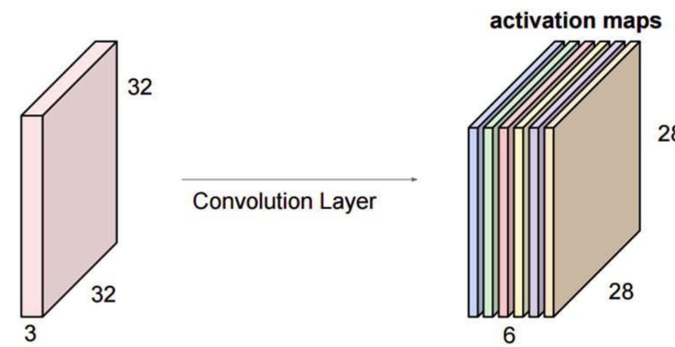
Convolutional Layer (CL)



Parameters of CL

- **Kernel size(K);**
- **Stride(S): Sliding Window (1 for CL and 1 for PL);**
- **Number of filters(F): Number of filters**
- **Zero Padding: Number of zeros to be add**

Activity



Given I an input image 32x32x3
6 Kernel Filters of 5x5x3
Pad=0
Slide=1
The result will be an activation map 28x28x6.

Explain it!

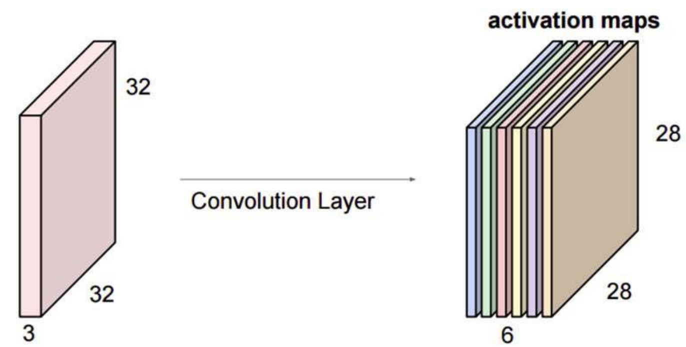
Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Convolutional Layer (CL)

Activity

Given I an input image 32x32x3
6 Kernel Filters of 5x5
Pad=0
Slide=1
The result will be an activation map 28x28x6.

Explain it!



$$Size_{Features\ Map} = 1 + \frac{(Size_{Input} + 2 * Padding) - Size_{Filer\ Kernel}}{Stride}$$

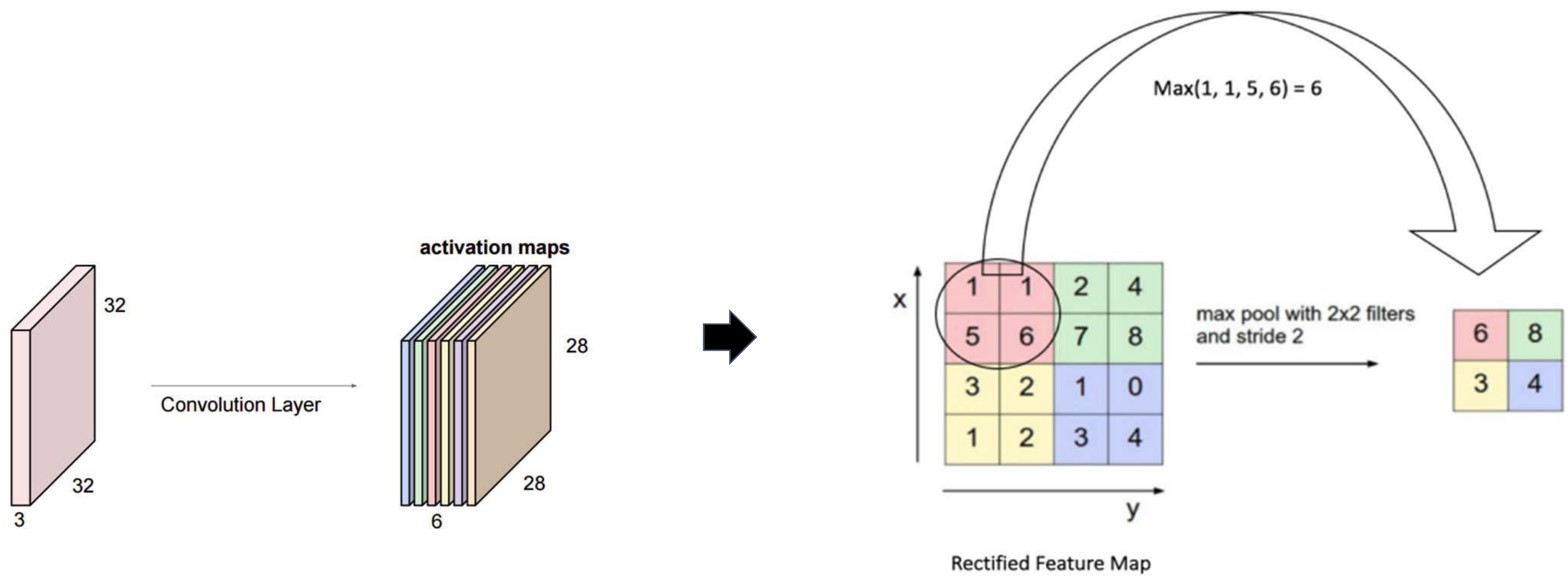
$$Size_{Features\ Map} = 1 + \frac{(32 + 2 * 0) - 5}{1}$$

$$Size_{Features\ Map} = 1 + 27 = 28$$

Convolutional Neural Network (CNN) Architecture:
Learning & Transfer Learning

Pooling Layer (PL)

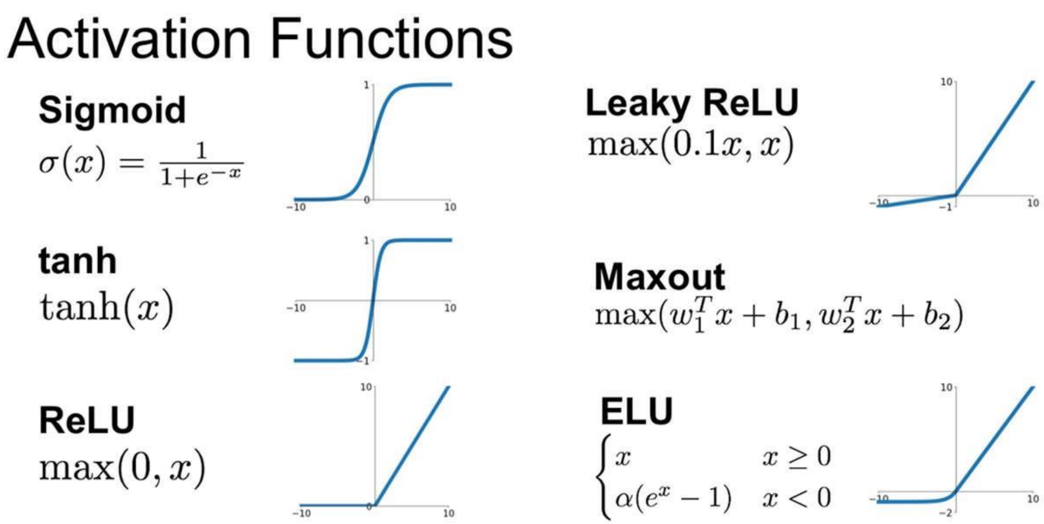
Spatial Pooling (subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum, etc.



Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

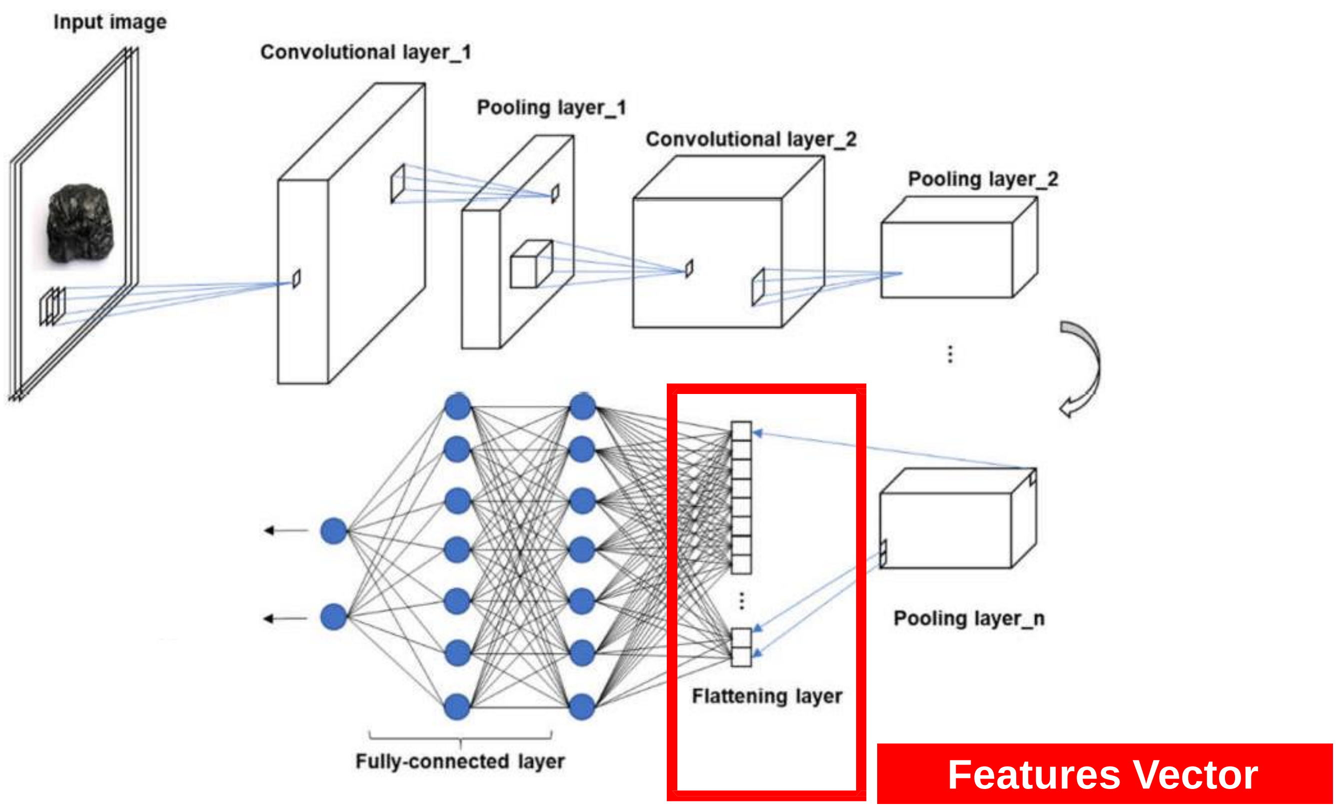
ReLU Layer (RL)

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

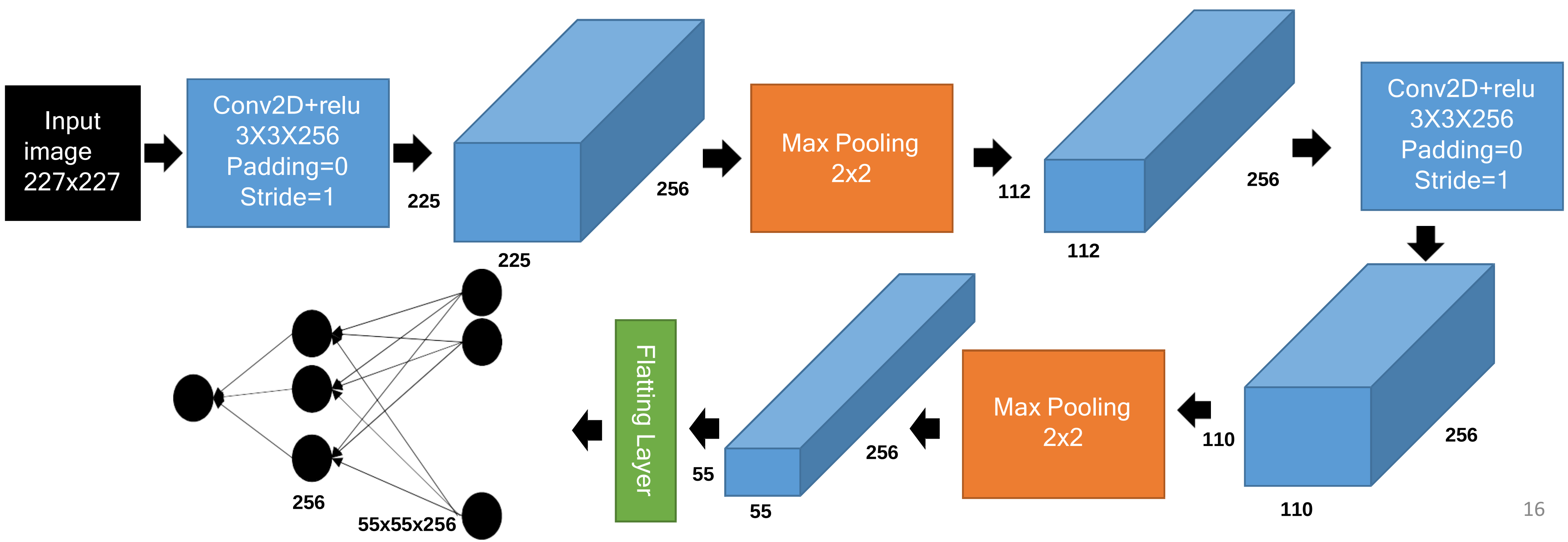


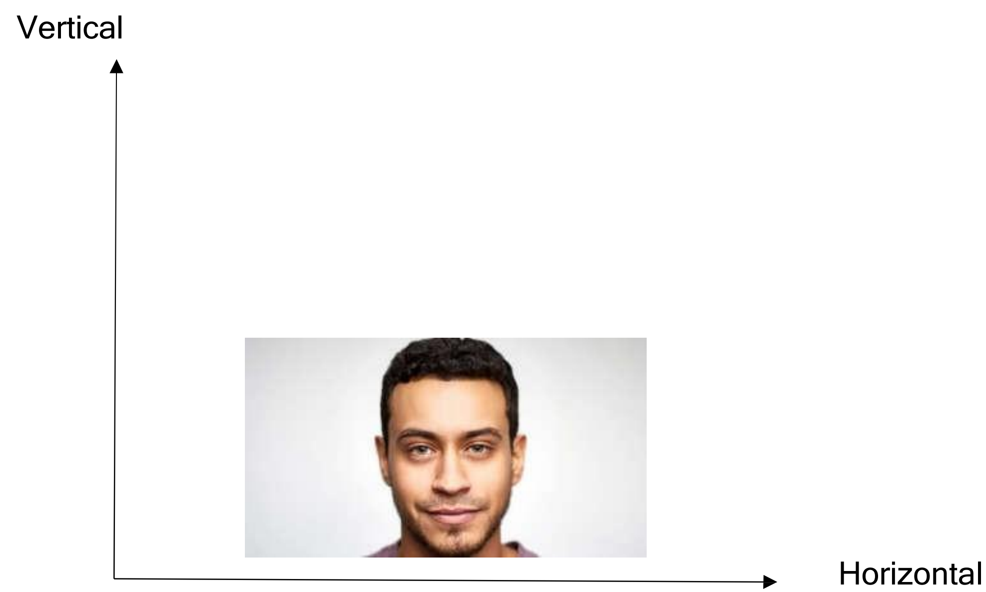
Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

FC Layer (FC)



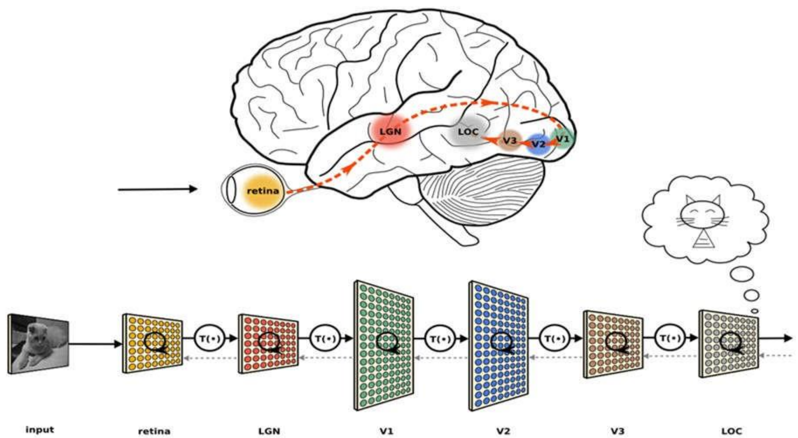
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,Flatten,MaxPooling2D
model = Sequential()
model.add(layers.Conv2D(256, (3, 3), activation='relu',input_shape=(227, 227, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='sigmoid'))
model.add(layers.Dense(1, activation='sigmoid'))
```

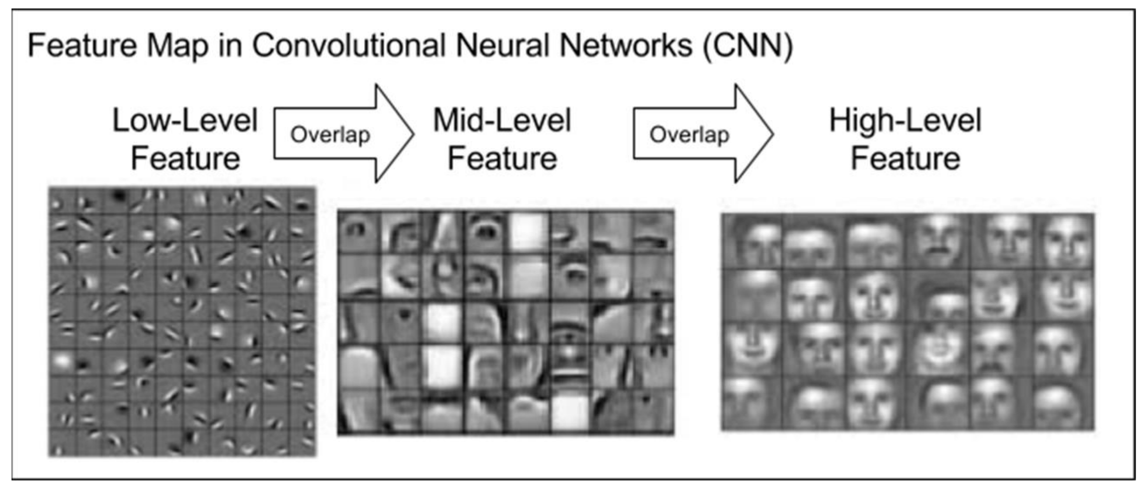



Vertical
Horizontal

Convolutional Neural Network (CNN) Architecture:
Learning & Transfer Learning

How to Design your Own CNN Architecture?





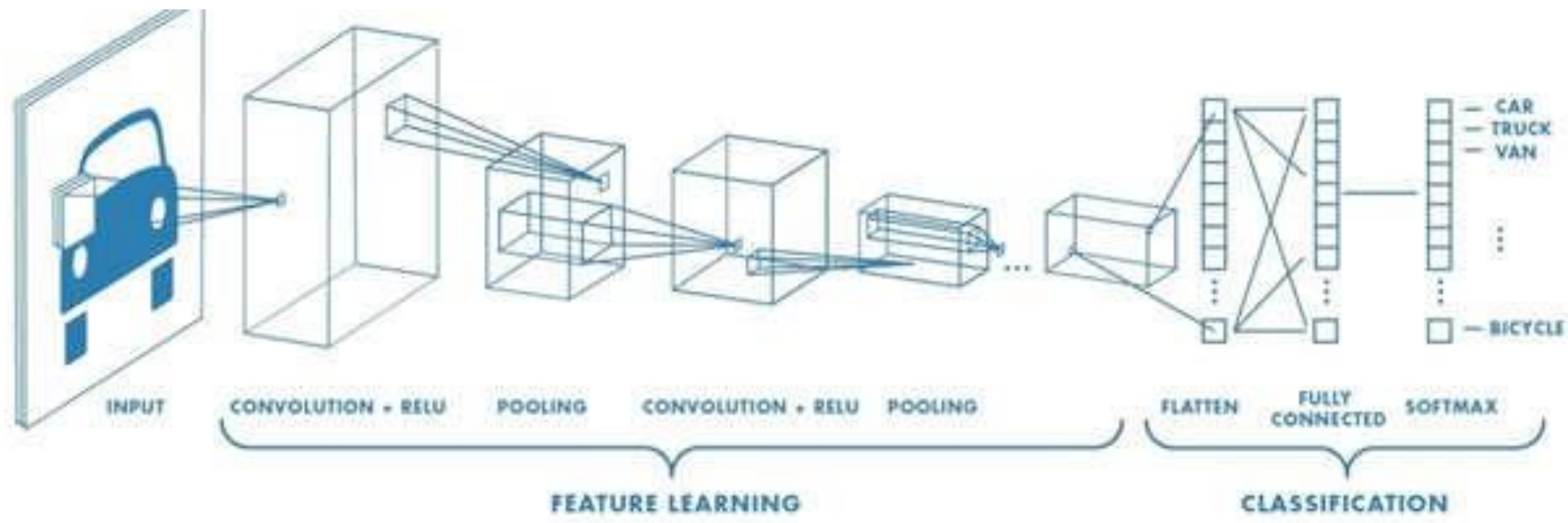
Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

ImageNet Large Scale Visual Recognition Competition

	#conv. layers	#MACCs [millions]	#params [millions]	#activations [millions]	ImageNet top-5 error
ZynqNet CNN	**18**	**530**	**2.5**	**8.8**	**15.4%**
AlexNet	5	1 140	62.4	2.4	19.7%
Network-in-Network	12	1 100	7.6	4.0	~19.0%
VGG-16	16	15 470	138.3	29.0	8.1%
GoogLeNet	22	1 600	7.0	10.4	9.2%
ResNet-50	50	3 870	25.6	46.9	7.0%
Inception v3	48	5 710	23.8	32.6	5.6%
Inception-ResNet-v2	96	9 210	31.6	74.5	4.9%
SqueezeNet	18	860	1.2	12.7	19.7%
SqueezeNet v1.1	18	390	1.2	7.8	19.7%

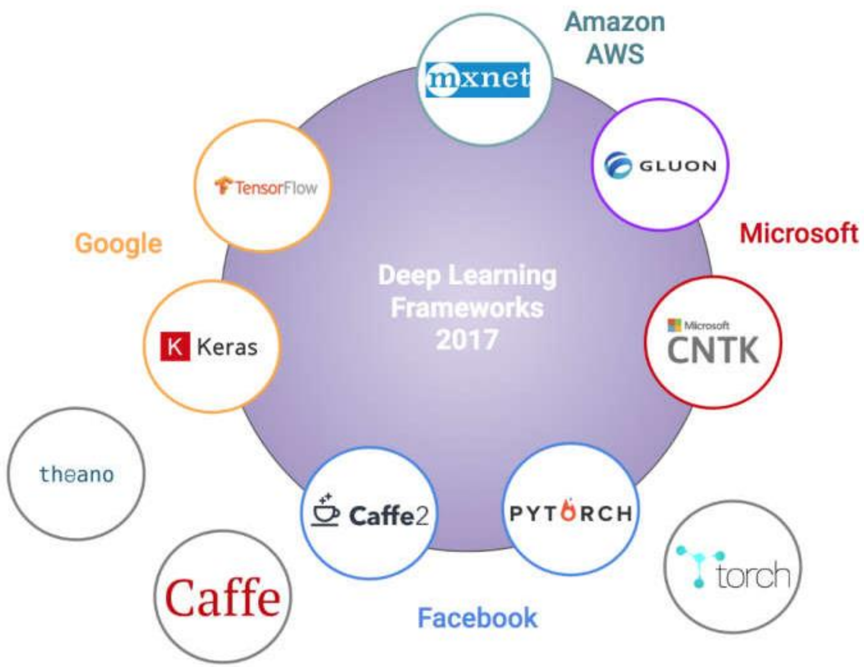
Convolutional Neural Network (CNN) Architecture:
Learning & Transfer Learning

Transfer Learning as Solution for lack of data!



6. Convolutional Neural Network (CNN) Architecture: Learning & Transfer Learning

Deep Learning Frameworks



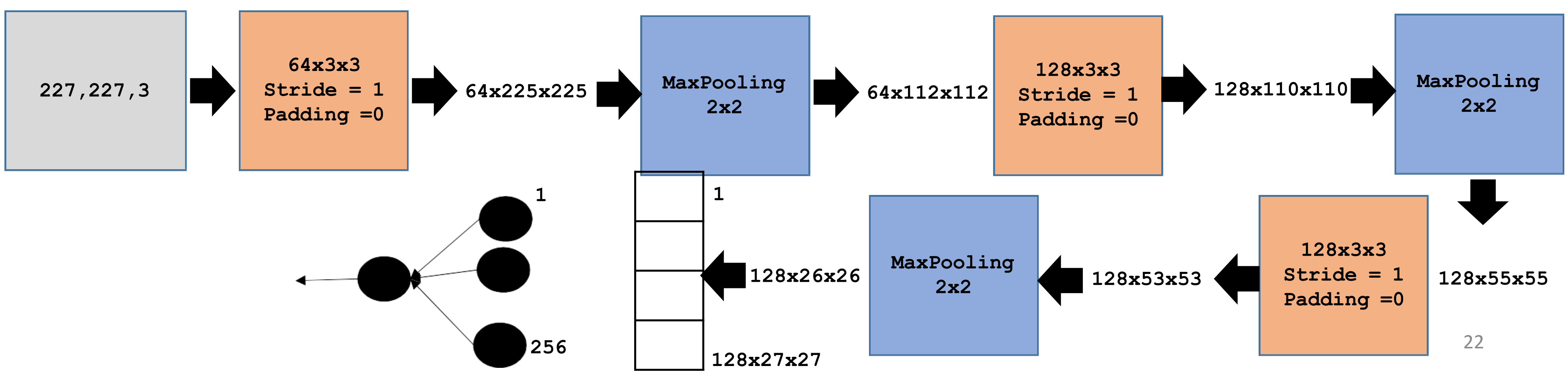
MathWorks
Deep Learning

jupyter

CO
the collaboratory

```python
from keras import layers
from keras import models
from keras.layers import Dropout,MaxPooling2D,Dense,Flatten,Conv2D
model = models.Sequential()
model.add(layers.Conv2D(64, (3, 3), activation='relu',
                        input_shape=(227, 227, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='sigmoid'))
model.add(layers.Dense(1, activation='sigmoid'))
```

$$Size_{Features\ Map} = 1 + \frac{(Size_{Input} + 2 * Padding) - Size_{Filer\ Kernel}}{Stride}$$



Thank you for your attention ☺

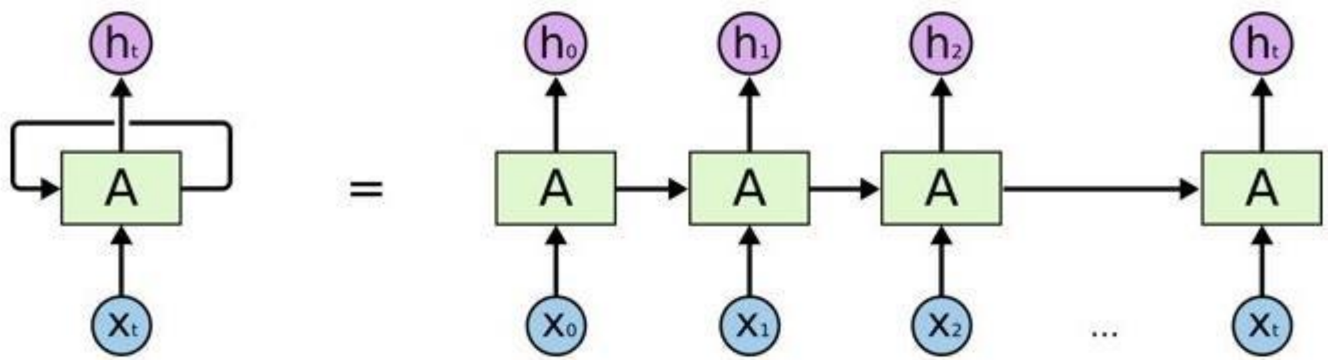
Dr. Eng. Wael Ouarda
E-mail: wael.Ouarda@ieee.org
Phone: +216 21 23 69 36
Web: http://www.crns.rnrt.tn/research-team/brain4ict

Centre de Recherche
en Numérique de Sfax
مركز البحث في الرقميات بصفاقس

Brain4ICT

Long Short Term Memory (LSTM)

Recurrent Neural Network



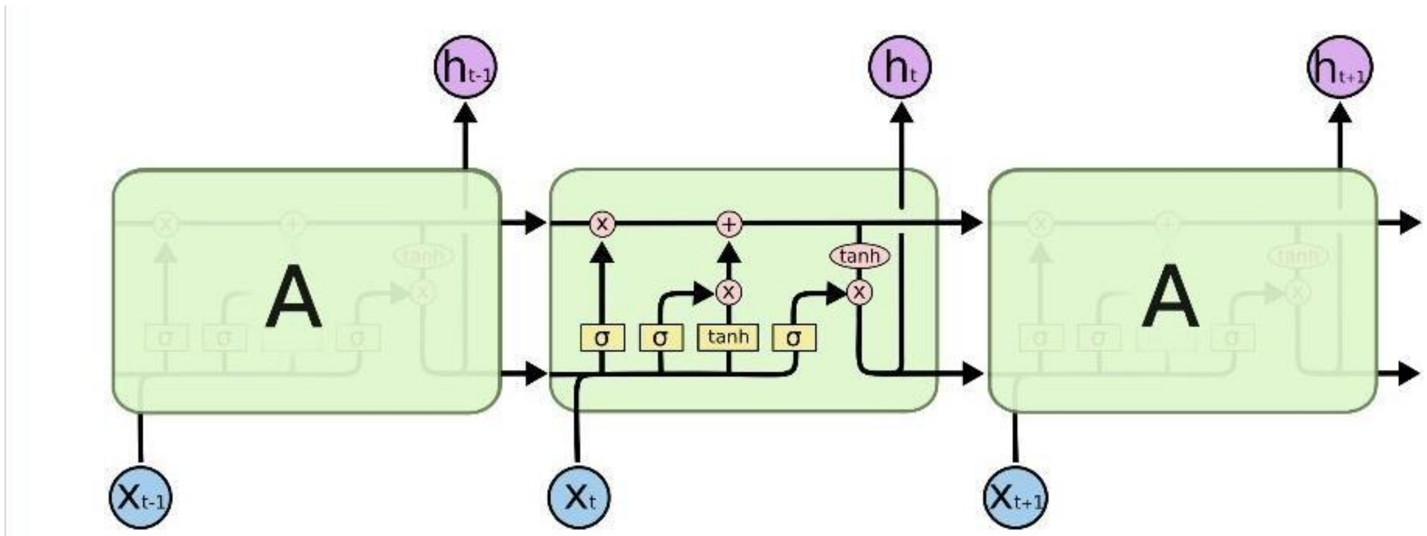
An unrolled recurrent neural network.

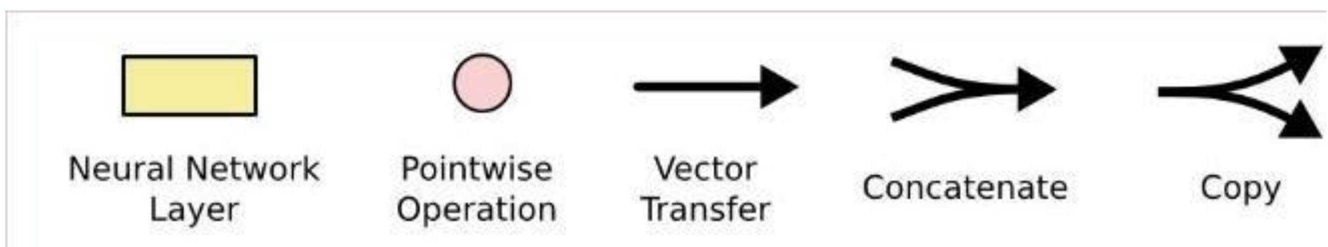
A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

Problem of Long Term Dependecy

Long Short Term Memory (LSTM)

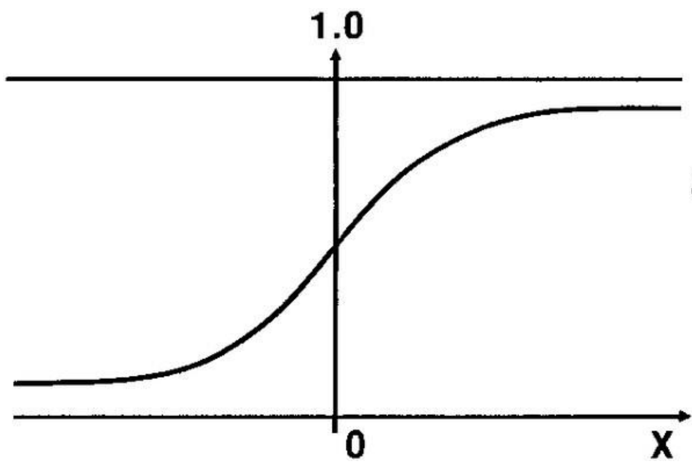
Long Short Term Memory





Long Short Term Memory (LSTM)

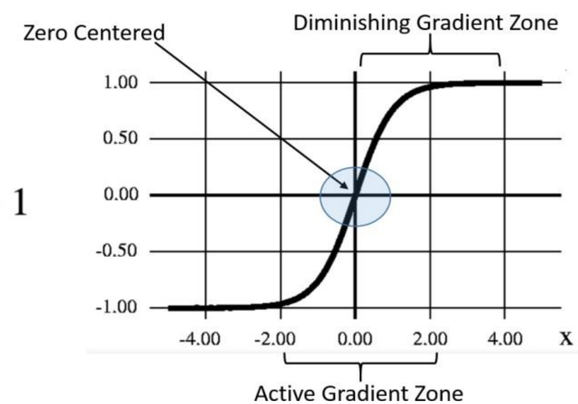
Activation Functions



Sigmoid

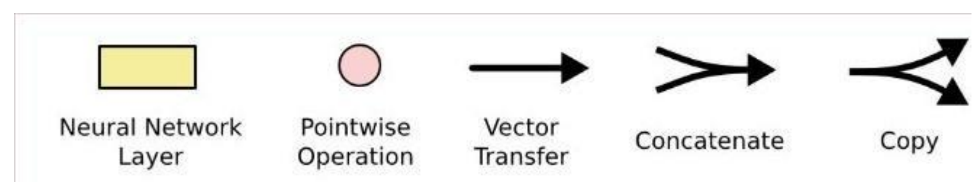
$$f(x) = \frac{1}{1+e^{-x}}$$

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



especially used for models where we have to predict the probability as an output.
Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice

mainly used classification between two classes

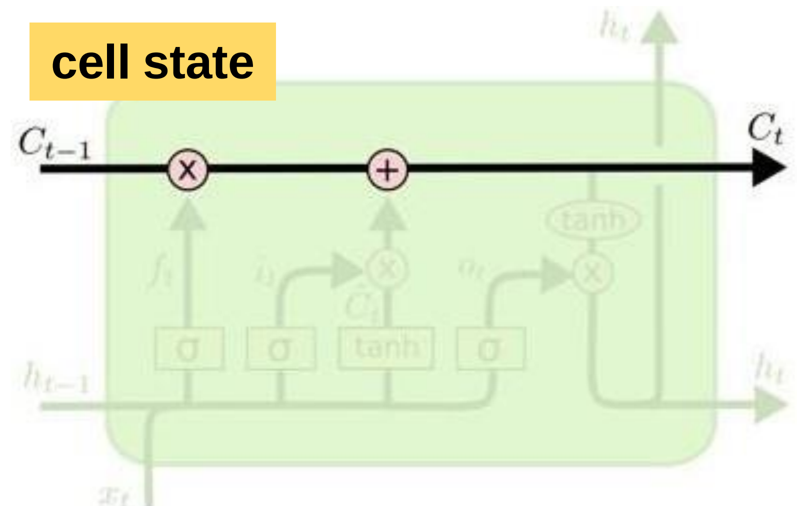


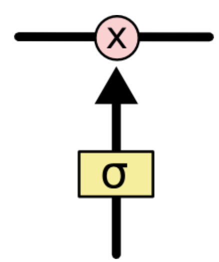
Long Short Term Memory (LSTM)

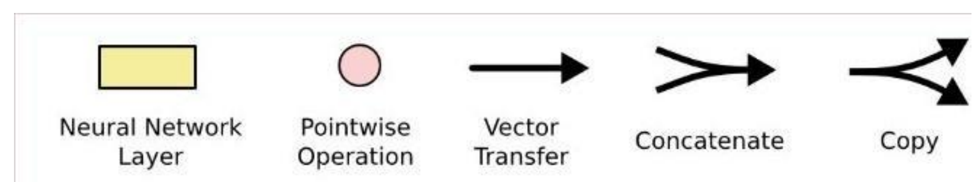
LSTM Cell state & Gates

Cell State is the key of the LSTM

- ❑ Horizontal Line over the Cell
- ❑ Ability to remove or add information to the cell state, regulated by structures called gates
- ❑ Gates = a sigmoid neural net layer and a pointwise multiplication operation
- ❑ Sigmoid is giving values within 0 and 1 to determine the quantity of information to add to the state cell

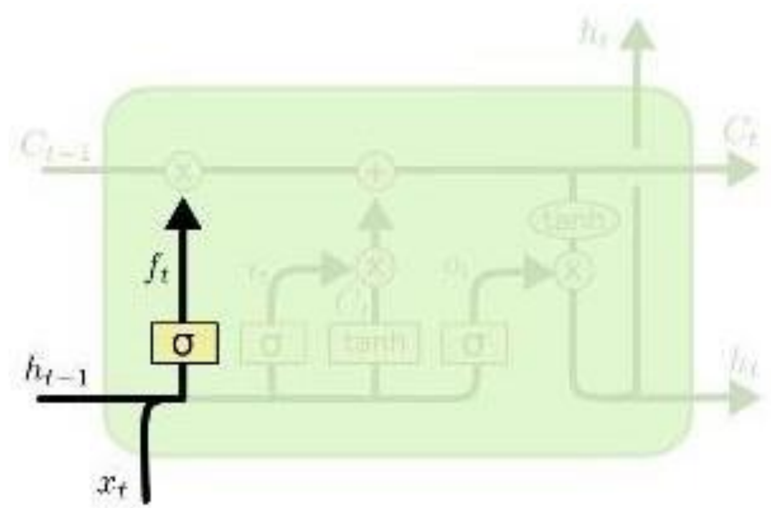






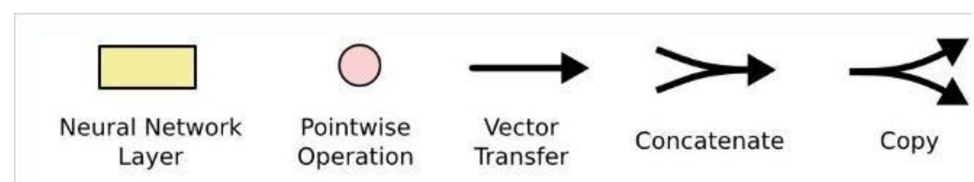
Long Short Term Memory (LSTM)

Step One – Forget Gate Layer To decide what information we're going to throw away from the cell state



1 represents "completely keep this"

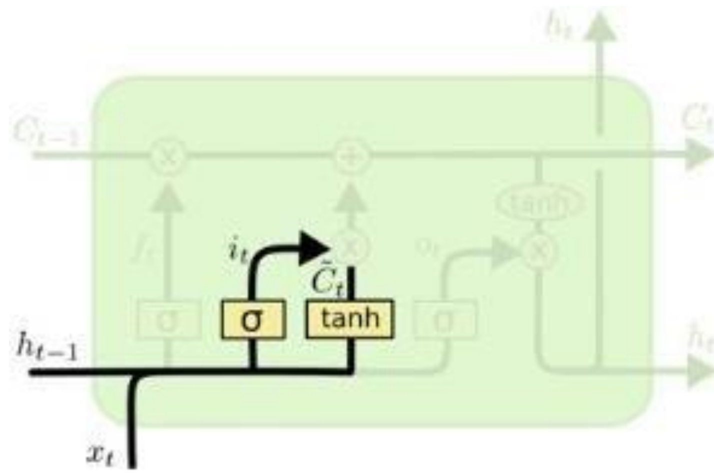
0 represents "completely get rid of this."



Long Short Term Memory (LSTM)

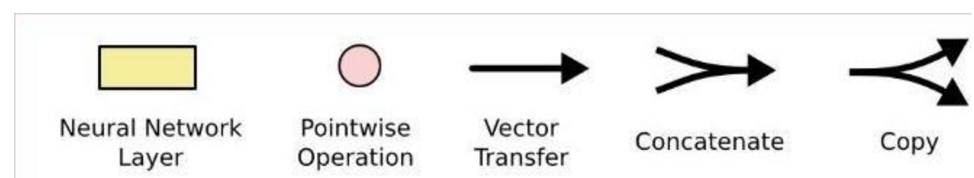
Step Two – Inpu Gate Layer

Decide what new information we're going to store in the cell state



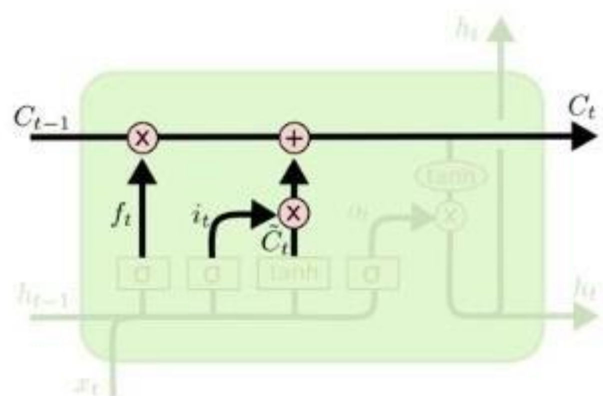
Part 1: a sigmoid layer called the "input gate layer" decides which values we'll update

Part 2: a tanh layer creates a vector of new candidate values that could be added to the state.



Long Short Term Memory (LSTM)

Step Three – Update Cell State



1. Multiply the old state by the forget Gate to forget what we don’t need from the previous state
2. Adding the new cell state new candidate Ct*input gates to construct a new state based on the new input

Long Short Term Memory (LSTM)

Step four – Output Decide what we are going to output

output will be based on our cell state, but will be a filtered version