NEURAL NETWORKS AND CONVOLUTIONAL NEURAL NETWORKS WITH PRACTICAL EXAMPLE

Muhammad Numan Anwar
Department of Physics
Polytechnic University of Bari
INFN, Italy

Outline

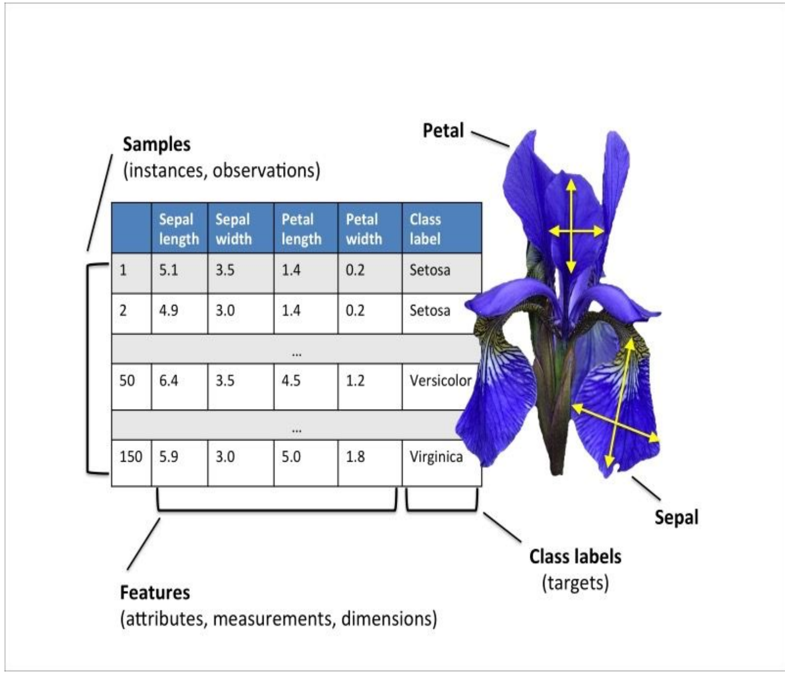
1. **Introduction of Neural Networks and Multi Layer Perceptron**
2. **Activation and Loss Functions**
3. **Validation Procedure**
4. **Hyperparameters in a NN**
5. **Overfitting**
6. **Evaluation Metrics**
7. **Introduction to Convolution in 1 and 2 Dimensions**
8. **An Overview on the Different Layers of a CNN**
9. **A Brief Explanation of the Data Augmentation**

ML Classification Problem

- **A Neural Network is a very powerful classification algorithm**
- **The simplest version of a Neural Network is the Perceptron**
- **Perceptron can be considered the building block of a NN**
- **GOAL: The discrimination between 2 classes or among more classes through a training done on a specific dataset**

ML Classification Problem

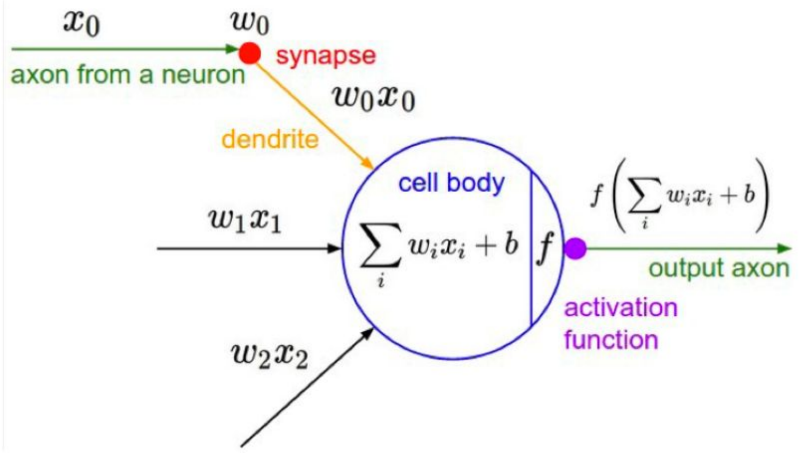
- FIRST STEP: **Selection of our dataset in which there are a set of features that describes what we want to discriminate (for example the discrimination of different types or irises)**
- SECOND STEP: **Defining classes and targets as shown in the diagram: Setosa, Versicolor and Virginica**



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

SINGLE-LAYER NEURAL NETWORK

- The simplest algorithm able to discriminate between two classes (binary classificator) is the perceptron
- The perceptron is the fundamental building-block of a Neural Network
- **In a single perceptron we have:**
- Input Variable (x) and weight (w)
- An activation function
- An output that is a linear function of inputs and weights

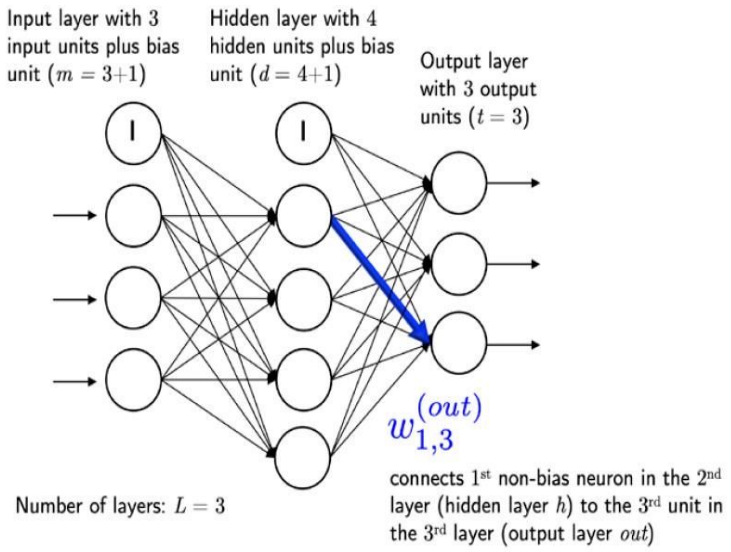


MULTI-LAYER PERCEPTRON

- First example of NN: **Multilayer perceptron**:

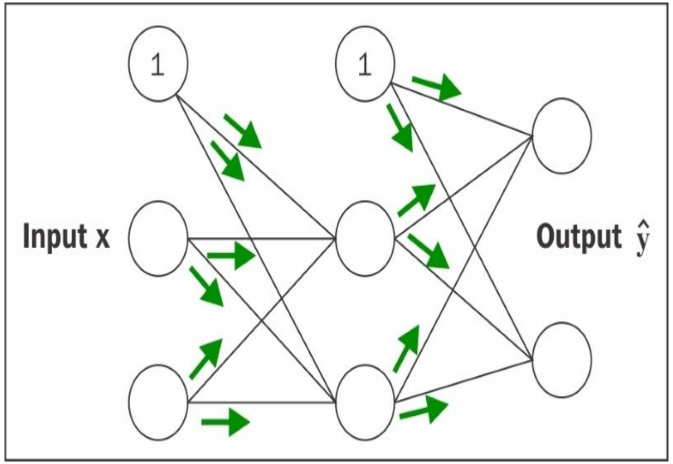
❖ **a net of fully connected perceptron**

- In the schematic view on the right every circle is a perceptron with a fixed number of inputs and outputs
- In the example (on the right) we have an input layer, only one hidden layer and an output layer


Input layer with 3 input units plus bias unit ($m = 3+1$)
Hidden layer with 4 hidden units plus bias unit ($d = 4+1$)
Output layer with 3 output units ($t = 3$)
I
I
$w_{1,3}^{(out)}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)
Number of layers: $L = 3$

MULTI-LAYER PERCEPTRON

- **<u>In this case we just have:</u>**
- ❑ Two input variables
- ❑ An hidden layer with size 2 (i.e. with 2 neurons)
- ❑ Two outputs
- #neurons in the next layer · #variables + #neurons in the layer = 6 weights (between input layer and hidden layer)
- #neurons in the next layer · #hidden layers size + #neurons in the layer = 6 weights (between hidden layer and output layer



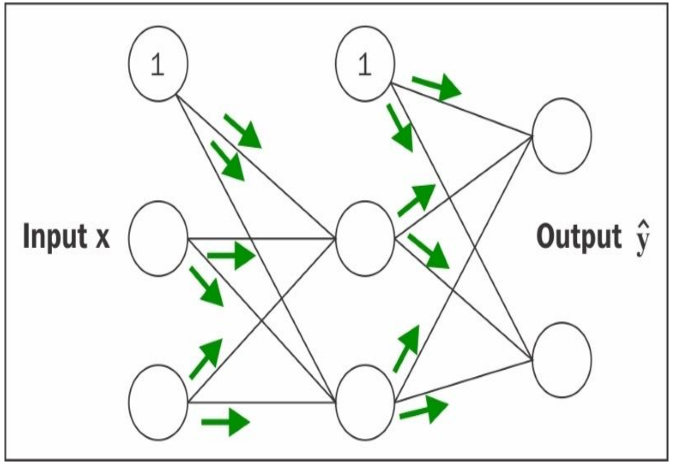
MULTI-LAYER PERCEPTRON

- Example: in this case there are two outputs
- The hidden layer output h is function of the input x:

$$h_1 = \sigma^h(w^i_{11}x_1 + w^i_{12}x_2 + b^i_1)$$
$$h_2 = \sigma^h(w^i_{21}x_1 + w^i_{22}x_2 + b^i_2)$$

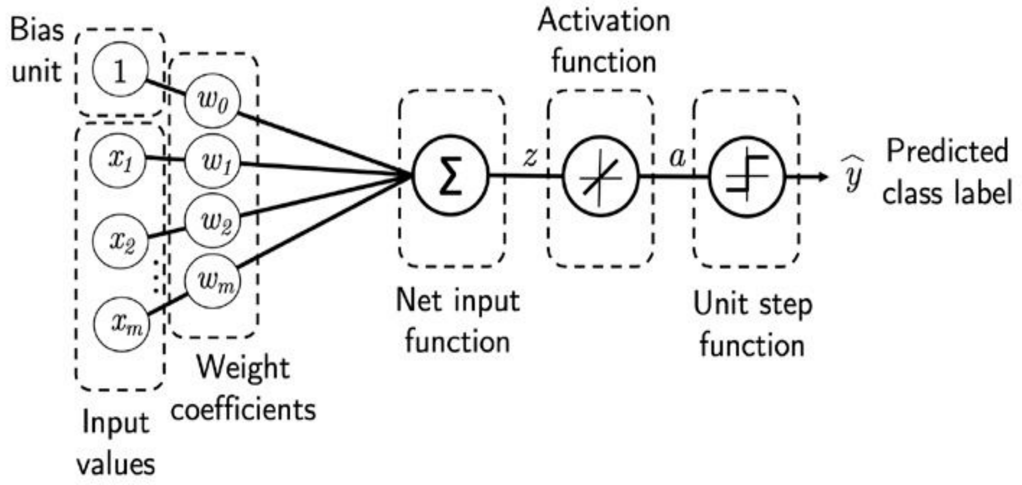
- The output o is a different function of its input, i.e. h:

$$o_1 = \sigma^o(w^h_{11}h_1 + w^h_{12}h_2 + b^h_1)$$
$$o_2 = \sigma^o(w^h_{21}h_1 + w^h_{22}h_2 + b^h_2)$$



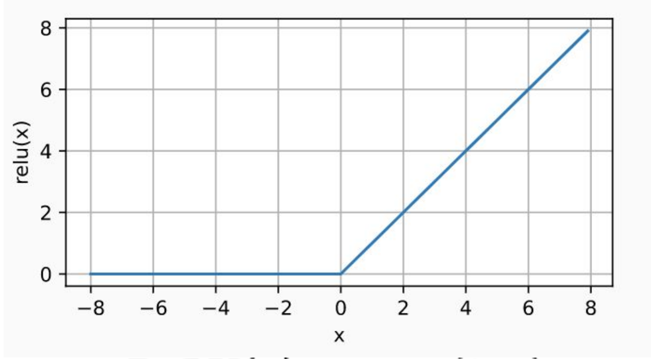
ACTIVATION FUNCTION

- Most of them provides to add non-linearity to the mode
- The activation function σ has as input the weighted sum of the input variables x, added with the bias b
- The functions are in general differentiable operators in order to transform the inputs to outputs

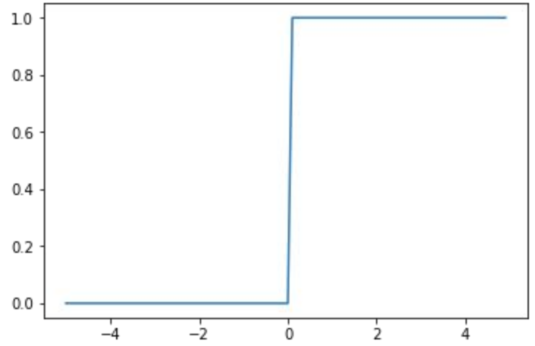


RECTIFIED LINEAR UNIT (RELU)

- One the most popular non-linear activation function is the REctified Linear Unit (ReLU)
- It provides a non-linear transformation and returns the max value between the input x (the argument) and 0
- The ReLU function is also differentiable in as given below:



$$ReLU(x) = max(0, x)$$



$$\frac{dReLU(x)}{dx} = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

SCALED EXPONENTIAL LINEAR UNIT (SELU)

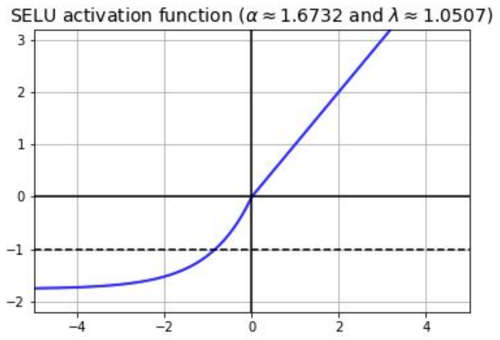
- Another choice is the Scaled Exponential Linear Unit (SELU)

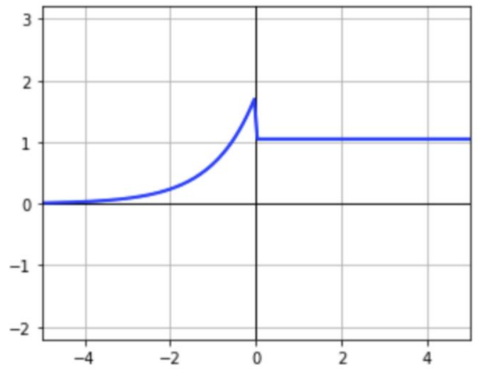
- The functions depends on two parameters and the equation is the following:

$$SELU(x) = \lambda \begin{cases} \alpha(e^x - 1) & x \leq 0 \\ x & x > 0 \end{cases}$$

- The function is not differentiable in zero

$$\frac{dSELU(x)}{dx} = \lambda \begin{cases} \alpha e^x & x \leq 0 \\ 1 & x > 0 \end{cases}$$





SIGMOID FUNCTION

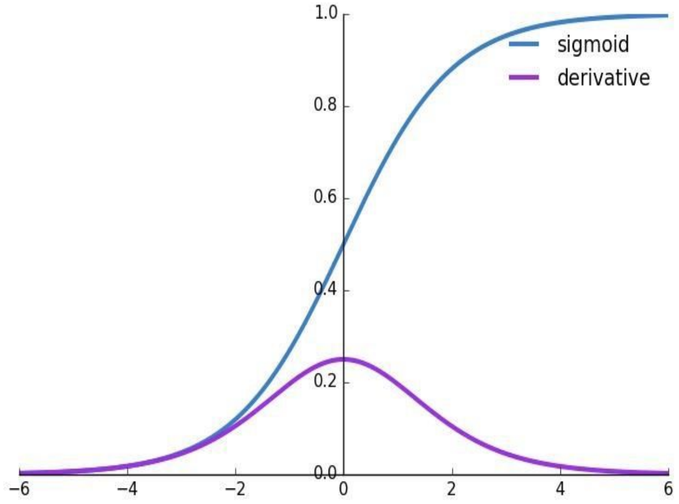
- **Sigmoid function:**

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

- **Derivative Sigmoid function:**

$$\frac{d\ sigmoid(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} =$$

$$= sigmoid(x)(1 - sigmoid(x))$$



LOSS FUNCTION

- To measure the quality of our predicted probabilities we need a loss function
- We will suppose that the entire dataset (or the batch we are considering) has n samples {X,Y}
- The predicted class can be compared with the real class by checking the probability associated to the actual class according to the model
- According to the maximum likelihood estimation, we want to maximize P(Y|X), or minimize the negative log-likelihood

$$\mathcal{P}(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{n} \mathcal{P}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)})$$

$$-log\mathcal{P}(\mathbf{Y}|\mathbf{X}) = \sum_{i=1}^{n} -log\mathcal{P}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)})$$

CROSS-ENTROPY LOSS FUNCTION

- The negative log-likelihood is equal to:

$$-log\mathcal{P}(\mathbf{Y}|\mathbf{X}) = \sum_{i=1}^{n} l(\mathbf{y}^{(i)}, \mathbf{y}_{pred}^{(i)})$$

- Where l(y,ypred) is the loss function, also called cross-entropy, defined as:

$$l(\mathbf{y}^{(i)}, \mathbf{y}_{pred}^{(i)}) = -\sum_{j=1}^{\#Classes} y_j^{(i)} \; log \, y_j^{(i)\,pred}$$

EXAMPLES OF OTHER LOSS FUNCTIONS

- Mean Squared Error(MSE)/ Quadratic Loss/ L2:

$$MSE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)^2}{n}$$

- Mean Absolute Error (MAE)/ L1 Loss:

$$MAE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left|y^{(i)} - y^{(i)}_{pred}\right|}{n}$$

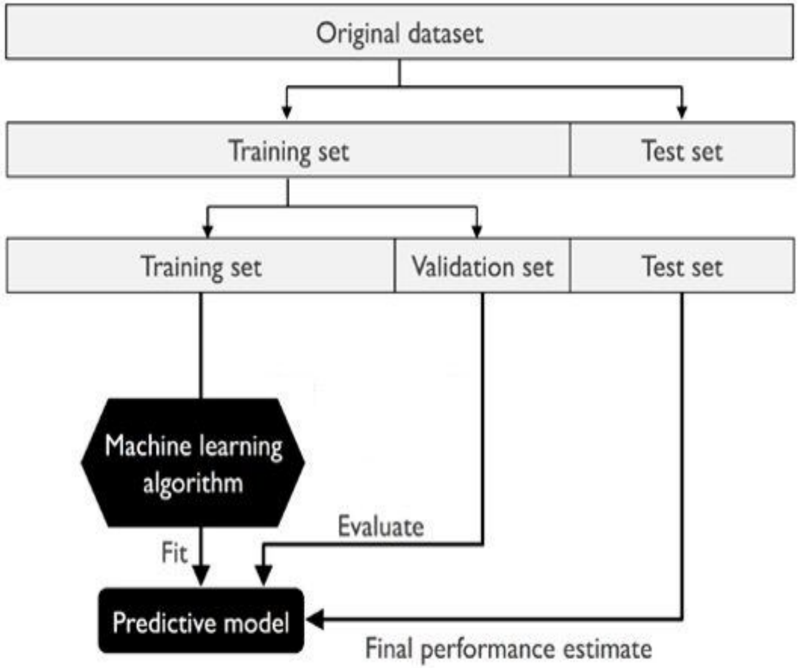
- Mean Bias Error (MBE):

$$MBE(y^{(i)}, y^{(i)}_{pred}) = \frac{\left(y^{(i)} - y^{(i)}_{pred}\right)}{n}$$

VALIDATION PROCEDURE

VALIDATION

- **Task**: we want to optimize our model, without touching the test dataset and avoiding the risk of overfitting
- We are going to use our test dataset only after the training is finished in order to assess the very best model
- The best practice to address this problem is to split our dataset in three (instead of two) parts, incorporating a validation dataset (or validation set) in addition to the training and test datasets



HYPERPARAMETERS

- **With a DNN we can change a lot of parameters, most of which are:**

❑ **The loss function**
❑ **The activation function of every layer**
❑ **The learning rate**
❑ **The number of epochs**
❑ **The number of hidden layers and the number of cells in them**
❑ **Many others**

NUMBER OF EPOCHS

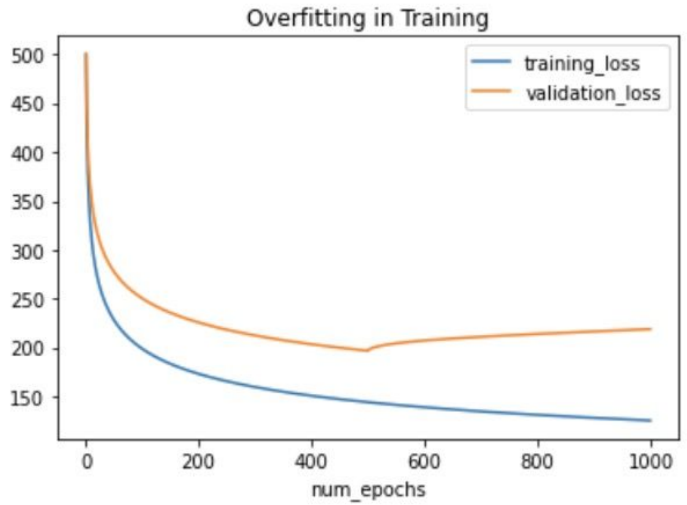
- **Epoch**: In terms of artificial neural networks, an epoch refers to one cycle through the full training dataset
- Number of epochs is a delicate choice:
  - ❑ A large number of epochs can induce our model to an overfitting problem
  - ❑ Too small number of epochs can lead to an under fitting problem
- To avoid a wrong choice we can use the ' EarlyStopping', also implemented by Keras:
  - ❑ It allows to stop the training when a monitor (set by us and tipically the loss function) has stopped improving.

HIDDEN LAYERS

- The number of hidden layers add complexity to our model
- Adding hidden layers makes our algorithm more performing, but at the same time we lead it to an overfitting problem
- Another crucial factor is the number of cells in the hidden layer, also in this case a lot of cells increase the complexity of the model and increase the risk to an overfitting problem
- This choice has to be done carefully, it is the most difficult one and only comparing the evaluation metrics between different approaches we can know which is the best one.

OVERFITTING PROBLEM

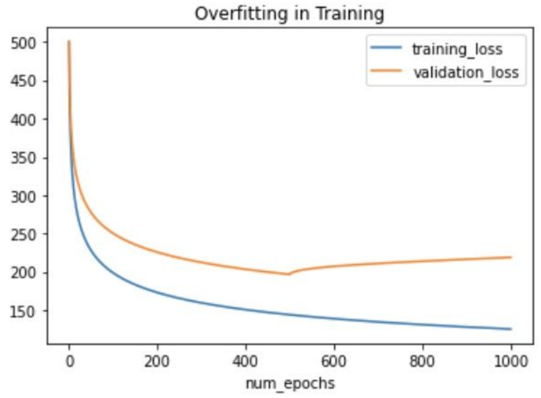
- The more complex the model is, the higher is the risk of overfitting
- Here a clear example of overfittig, the train loss keeps going down while the validation loss get worse. It is always important to split the training in train and validation set and to have a clear picture of the train history
- In order to avoid overfitting and make the training stable we have different approach

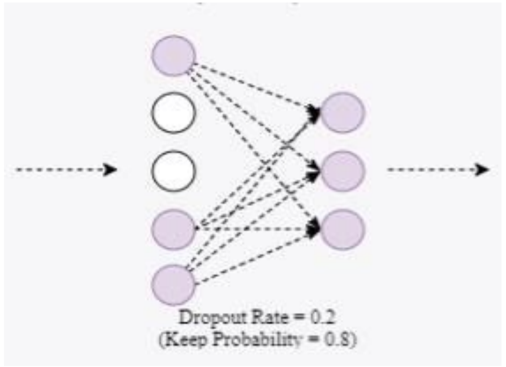


FACING OVERFITTING PROBLEM

- Introduce a callback function that stops the training if the validation loss get worse and restore the best parameters (**Early Stop function**).

- **<u>Dropout:</u>** it refers to the practice of disregarding certain nodes in a layer at random during training. A dropout is a regularization approach that prevents overfitting by ensuring that no units are co-dependent with one another





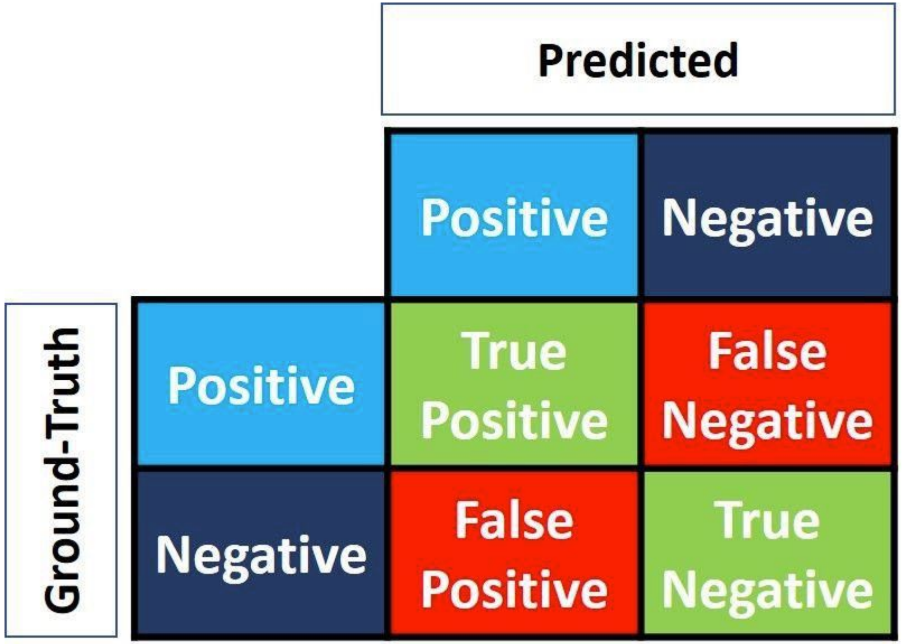
EVALUATION METRICS

- The idea of building machine learning models works on a constructive feedback principle:
  - ➢ building a model, getting a feedback from metrics, making improvements and continuing until you achieve the desired accuracy
- An important aspect of evaluation metrics is their capability to discriminate among model results
- The real goal is creating and selecting a model which gives high accuracy on sample data:

➢ It is crucial to check the accuracy of your model prior to computing predicted values.

CONFUSION MATRIX

- The confusion matrix helps us visualizing whether the model is "confused" in discriminating between two or more classes
- In the figure we have an example of binary model and the corresponding confusion matrix
- The 4 elements of the matrix represent the 4 metrics that count the number of correct and incorrect predictions the model made.



		Predicted	
		Positive	Negative
Ground-Truth	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

ACCURACY AND PRECISION

- The most famous metrics is the accuracy defined as the ratio between the number of correct predictions to the total number of predictions
- Accuracy values range between 0 and 1. Obviously an accuracy values near to 1 means that our model fits well the datasets

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

- The precision is calculated as the ratio between the number of Positive samples correctly classified to the total number of samples classified as Positive (either correctly or incorrectly). The precision measures the model's accuracy in classifying a sample as positive.
- **<u>The precision is high when</u>**:

  • The model makes many correct Positive classifications (maximize True Positive)

  • The model makes fewer incorrect Positive classifications (minimize False Positive)

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

RECALL

- The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples
- The recall cares only about how the positive samples are classified. This is independent of how the negative samples are classified, e.g. for the precision
- The decision of whether to use precision or recall depends on the type of problem to be solved:

➔ If the goal is to detect all the positive samples (without caring whether negative samples would be misclassified as positive) then we can use recall;

➔ if the problem is sensitive to classifying a sample as Positive in general, i.e. including Negative samples that were falsely classified as Positive we can use precision

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

AUC-ROC

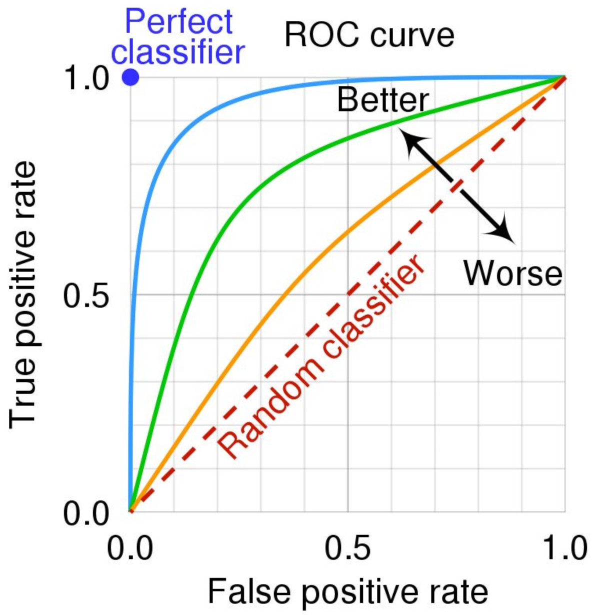
- A Receiver Operating Characteristic curve, or ROC curve, is a plot that illustrates the true positive rate against the false positive rate defined as follows:

$$TPR = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

$$FPR = \frac{False_{positive}}{False_{positive} + True_{negative}}$$

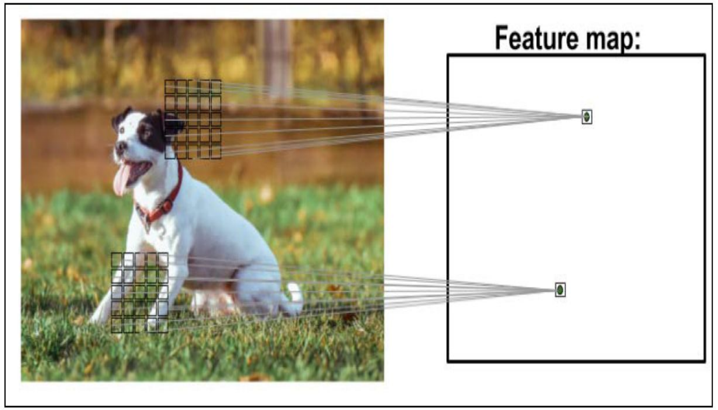
- The metric connected to the ROC curve is the area under the curve AUC

➢ An AUC near to 1 indicates a ROC curve near to the best result, an AUC near to 0 indicates a random classifier



Convolutional Neural Networks

- Convolutional Neural Networks are a powerful family of DNNs that are specifically designed for the Images Processing Task
- A Convolutional Neural Network (CNN) maintains the spatial structure of the data, and is better suited for finding spatial relationships in the image data
- The idea behind: to use filters that automatically learns the most Discriminants features in an image, such as edges, filled patterns, specific geometric forms and so on



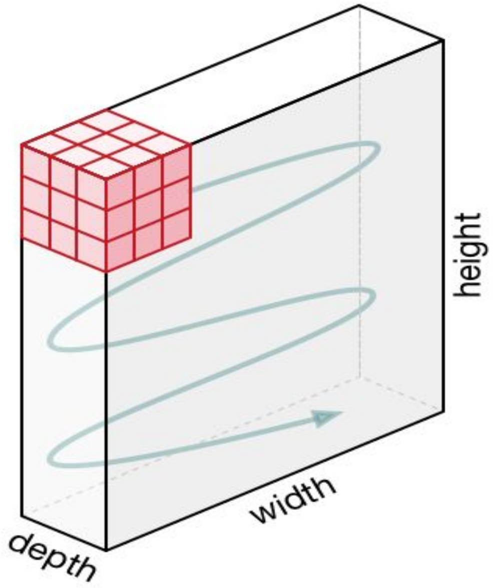
(Photo by Alexander Dummer on Unsplash)

The Convolutional Layer

- Let's start from the Convolutional Layer:
  - ➔ It is the core building block of a Convolutional Network that does most of the computational heavy lifting

Example:

- A typical filter on a first layer of a CNN has size 5x5x3
- Let's suppose we have a 28x28 pixel RGB image
- Convolution is the process of placing the filter 5x5x3 on the top left corner of the image, multiplying filter values by the pixel values and adding the results, moving the filter to the right one pixel at a time and repeating this process



The Convolutional Layer

THE MATHEMATICAL VIEW

- Now let's explain the convolution in mathematical details

  Discrete convolution in one dimension

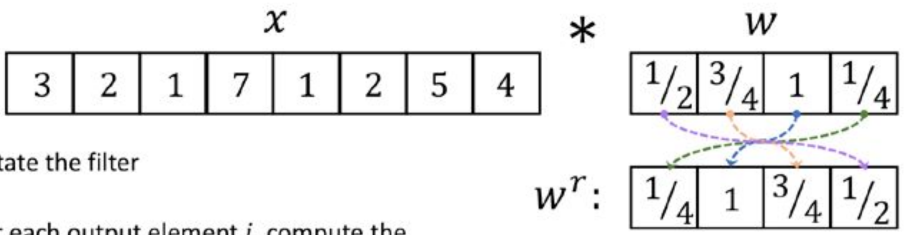
- A discrete convolution between two vectors with finite size, x and w, is mathematically defined as:

$$Y = X * W$$

Example:

x = [3 2 1 7 1 2 5 4], w = [1⁄2, 3⁄4, 1, 1⁄4]

The Convolutional Layer



Step 1: Rotate the filter

Step 2: For each output element i, compute the dot-product $x[i: i+4].w^r$

(move the filter two cells)

$$y[0] = 3\times{}^1/_4 + 2\times1 + 1\times{}^3/_4 + 7\times{}^1/_2$$

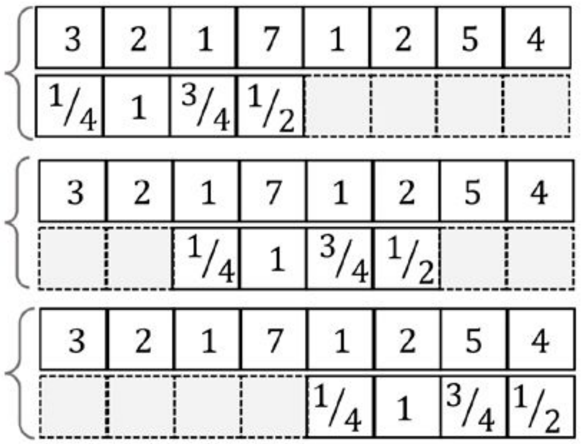
$\rightarrow y[0] = 7$

$$y[1] = 1\times{}^1/_4 + 7\times1 + 1\times{}^3/_4 + 2\times{}^1/_2$$

$\rightarrow y[1] = 9$

$$y[2] = 1\times{}^1/_4 + 2\times1 + 5\times{}^3/_4 + 4\times{}^1/_2$$

$\rightarrow y[2] = 8$



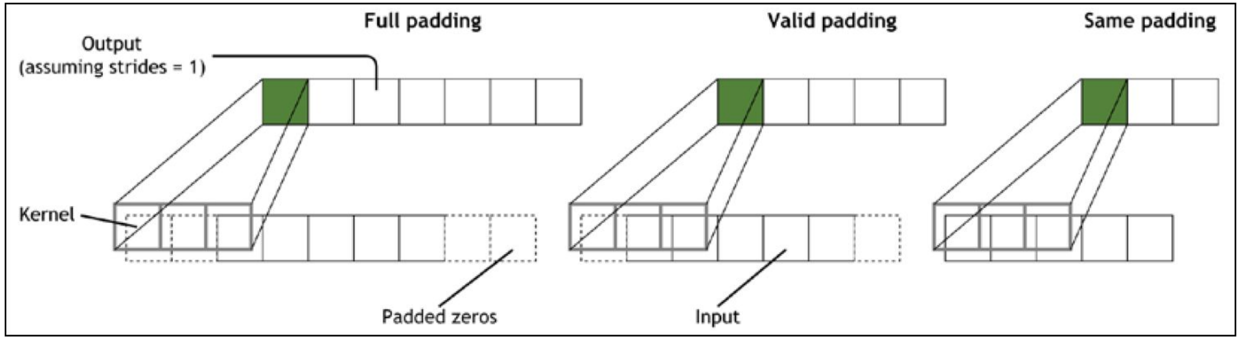
Padding Layer

- The result of this convolution is a tensor with a smaller shape than the input one
- To preserve/increase input shape we can use the so called padding procedure:

➔ It consists in **padding zero** pixels to input tensor

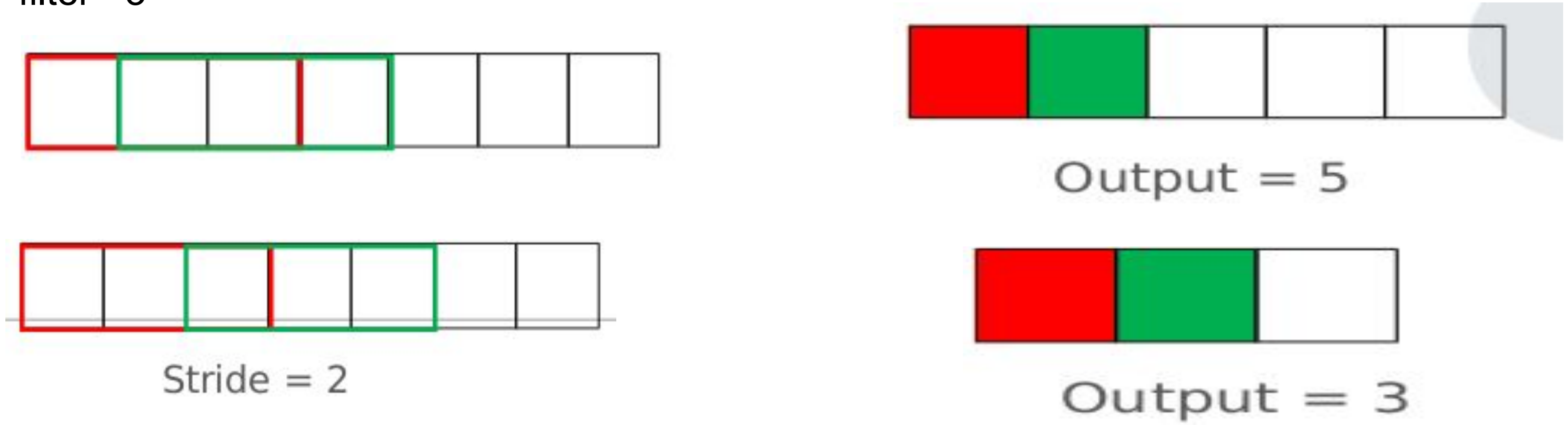
➔ Usually, a **same padding** procedure is used, meaning that the output vector has the same size as the input one

➔ **Valid padding**: we are not adding anything



Stride Parameter

- One concept introduced in the previous example is the number of cells the filter is moved when shifted across the vector x (to pass from a y index to another)
- It is called stride
- **Example:**
- N=7,
- filter =3



The Output Size

- The size of the vector obtained by a convolution can be calculated as follows:

$$o = \left\lceil \frac{n + 2p - m}{s} \right\rfloor + 1$$

- o = output dimension
- n = input dimension
- p = padding
- m = kernel size
- s = stride
- The bigger is the stride the smaller is the output dimension!

Convolution in 2D

THE MATHEMATICAL VIEW

- The concepts we have now discussed are easily extendible to 2D case:

Y = X * W

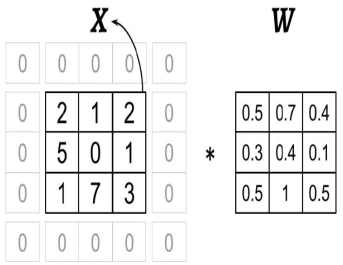
Example:

Input matrix $X_{3\times3}$

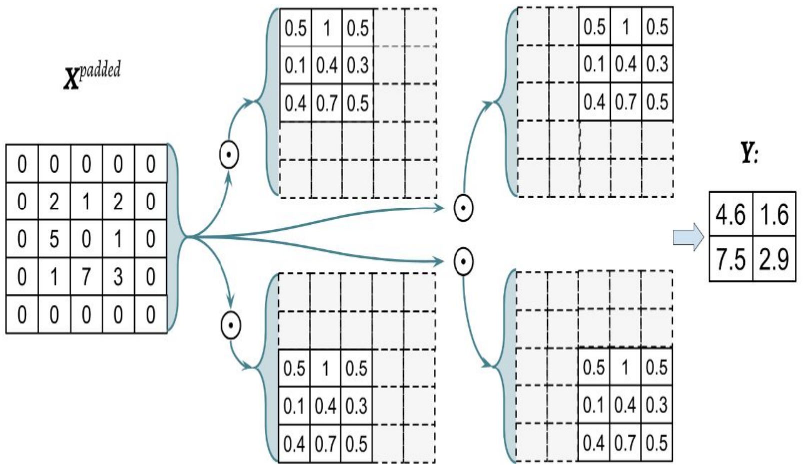
kernel matrix $W_{3\times3}$

P = (1, 1)

Stride s = (2, 2)



$$W^r = \begin{bmatrix} 0.5 & 1 & 0.5 \\ 0.1 & 0.4 & 0.3 \\ 0.4 & 0.7 & 0.5 \end{bmatrix}$$



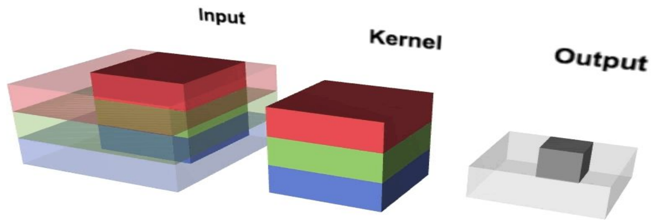
Convolution in 2D

- How does a convolutional layer work on a RGB image?

1. For each channel color there is a different filter

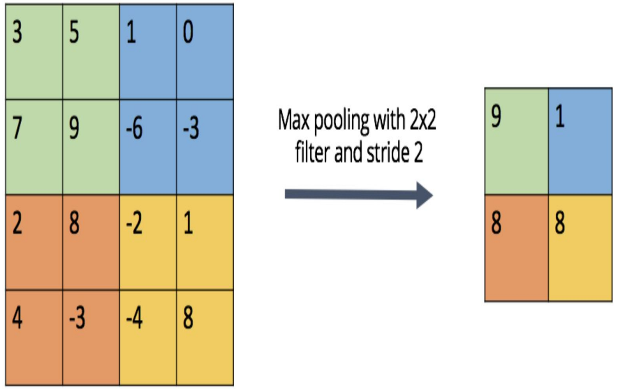
2. The three outputs are added together

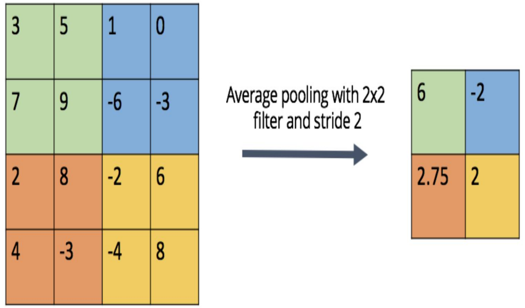
3. The output of a convolutional layer with a multi-layer input is a single layer



The Pooling Layer

- Tuning zero-padding and strides it is possible to change (usually reduce) the output dimension w.r.t. input dimension
- This task can be also performed with a "Pooling" layer
- **TWO KINDS OF POOLING:**
- **Maximum Pooling**: Calculate the maximum value for each patch of the feature map
- **Average Pooling**: Calculate the average value for each patch on the feature map

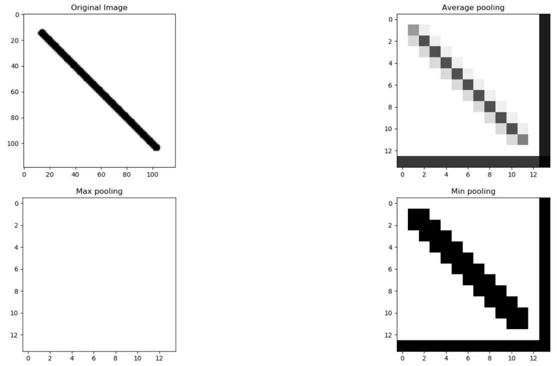




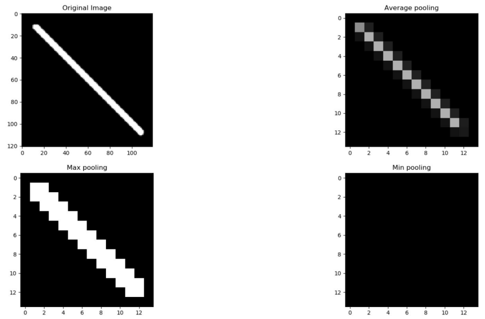
Different Pooling Layers

- Which pooling do I have to choose????
  - It depends!

❖ In this case min pooling let us preserving better the original information!

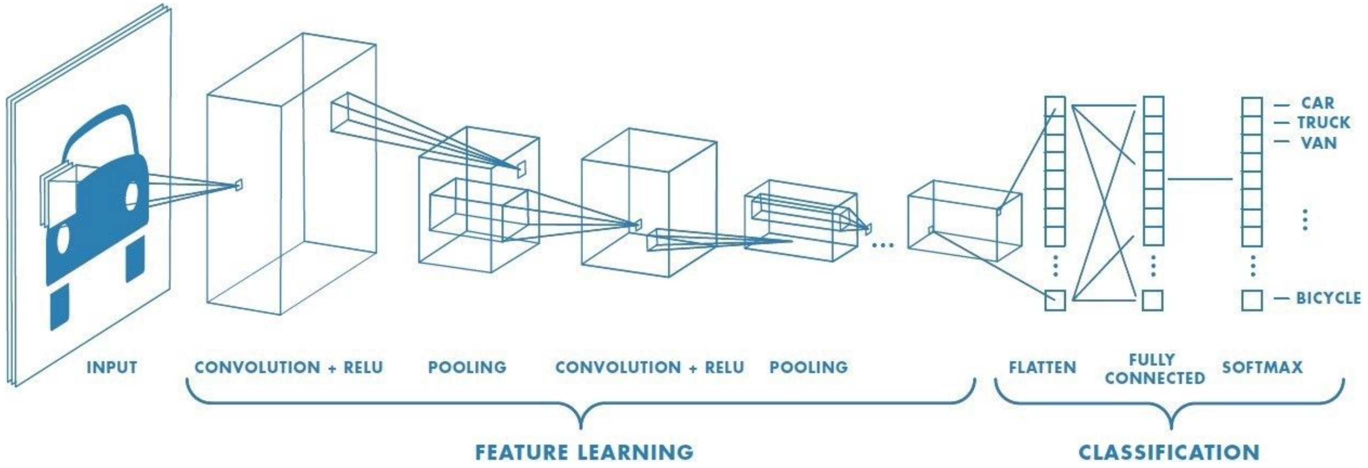


❖ **In this case max pooling let us preserving better the original information!**



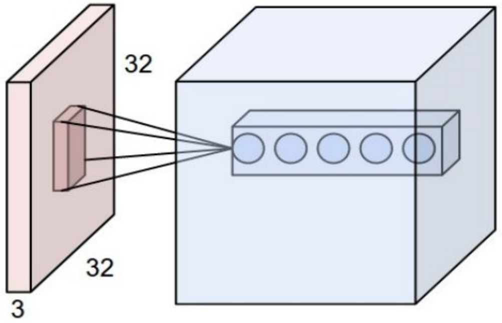
Full Diagram of CNN

- A convolutional neural network is a sequence of the following layers ordered in different ways:
- **Convolutional**
- **Pooling**
- **Dense layer**

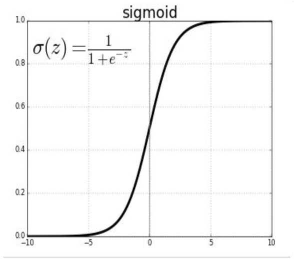


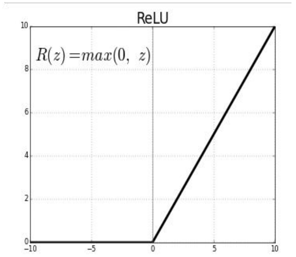
Full Diagram of CNN

- **In a convolution layer, usually, not only one but several filters are stacked together**
- **Each filter learns some different information from the same image**
- **Typically, a convolutional layer is composed of:**
- **N Filter + Activation Function**
- **Pooling Layer**

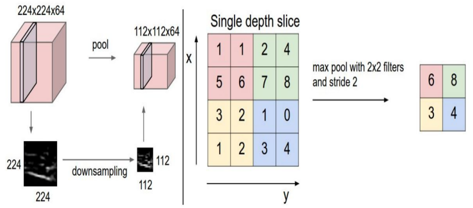


+





+

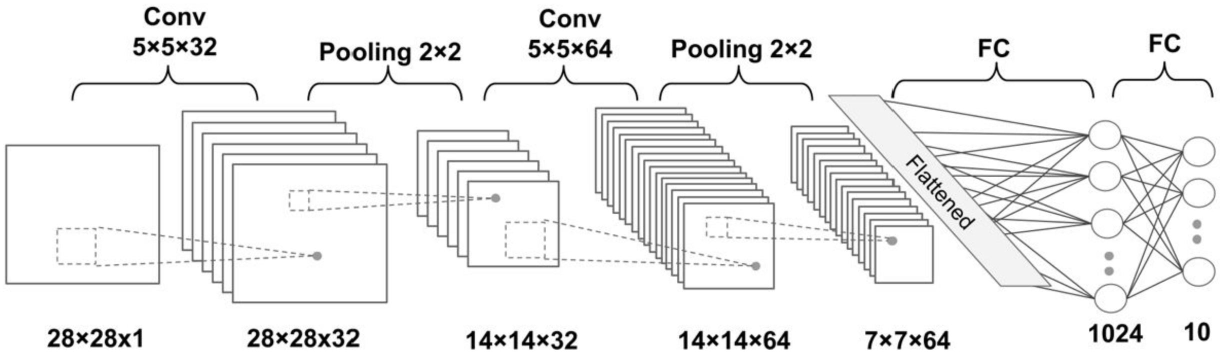


Dense Layers For Classification

- Now we must flatten the final output and feed it to a regular Neural Network for classification purposes
- Adding a Fully-Connected layer is a way of learning non-linear combinations of the high-level features (from filters)

1. The image is flattened into a column vector

2. then fed to a fully-connected neural network

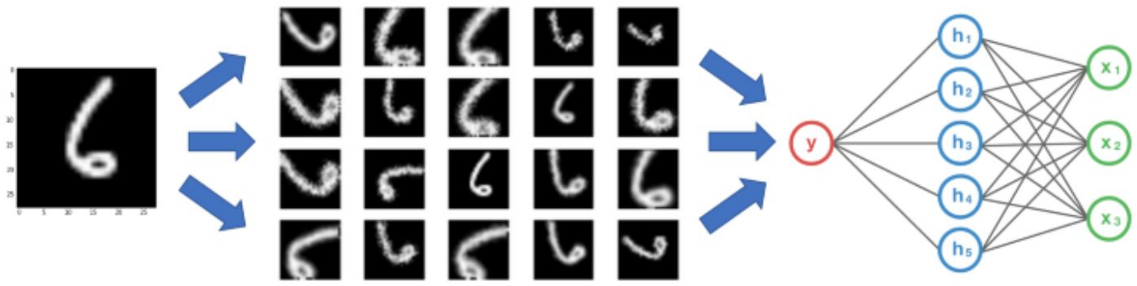


Data Augmentation

- When the size of training dataset is small, it is a good practice to increase the fit performances applying Data Augmentation:
- It consists in replicating existent images by applying small changes to it (rotation, translation, resizing, flip..)
- In this way not only the number of training samples increases but each picture is fed to the network with different prospections. **THE NETWORK GENERALIZES BETTER!!!**
- **EXAMPLE:**
- A poorly trained neural network would think that these three tennis balls shown below are distinct images, instead they are not

Data Augmentation

- In the real-world scenario, we may have a dataset of images taken in a limited set of conditions, but our target application may exist in a variety of conditions, such as different orientation, location, scale, brightness etc.
- A convolutional neural network that can robustly classify objects even if it is placed in different orientations is said to have the property called invariance to translation, viewpoint, size or illumination
- We account for these situations by training our neural network with additional synthetically modified data



Thank you