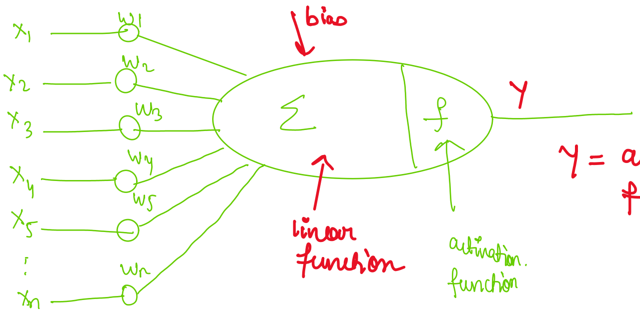
# Neural network

Saturday, 11 May 2024 5:31 PM

Basic unit of neural network is an artificial neuron that connect with other artificial neuron and create a network mimicking the brain



$$Y = \text{activation function}\left(\sum_{i=1}^{n} w_i x_i + \text{bias}\right)$$

## Activation function

The activation function decides whether a neuron should be activated or not by calculating the weighed sum and further adding bias to it.

## Choice of activation function

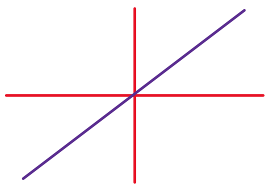
* The choice of activation function in the hidden layer how well the network model learns the training dataset.

* The choice of activation function in the output layer will define the type of predictions the model can make.

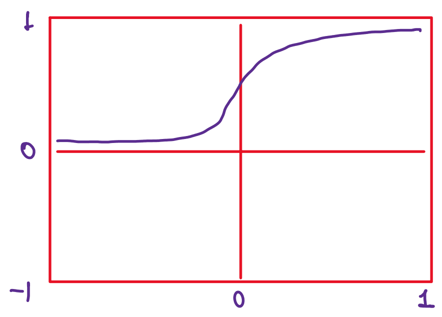
## Type of activation function

**linear activation function:** The linear activation function is also known as "identity" (multiplied by 1.0) or "no activation".

$Y = X$

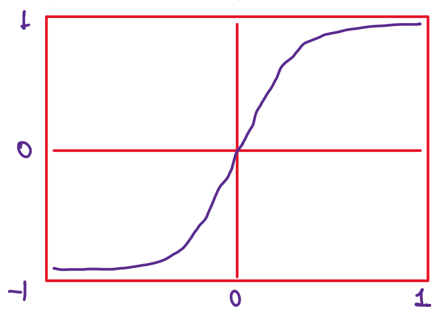


Sigmoid Activation function: The target labels used to train a model with a sigmoid activation function in the output layer will have the values 0 or 1



$$\boxed{Y = \frac{1}{1+e^{-x}}}$$

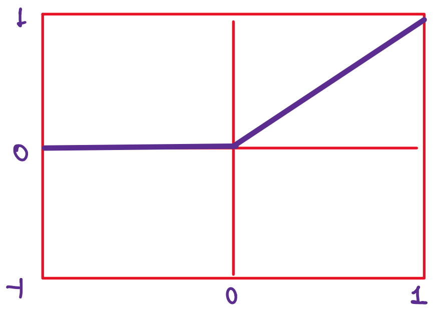
## Hyperbolic Tangent



$$Y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

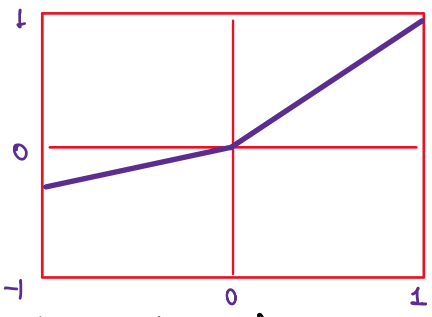
## Modern Activation function

### Rectified linear unit (ReLU)



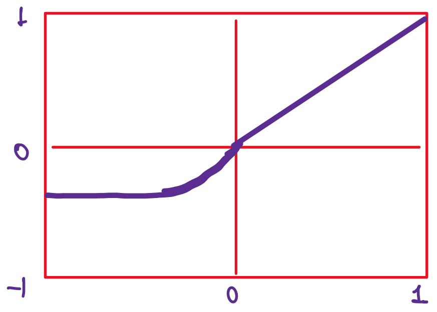
$Y = \max(0, x)$

### Leaky ReLU



$Y = \max(ax, x)$
$a$ = small constant (eg 0.1)

### Exponential LU

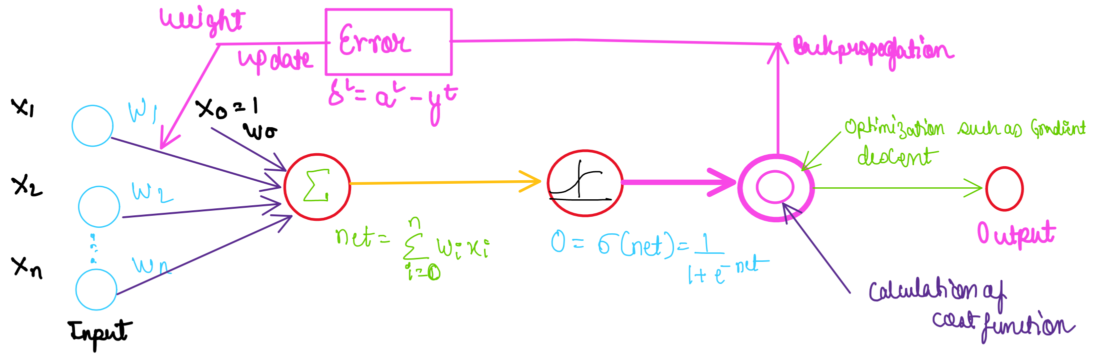


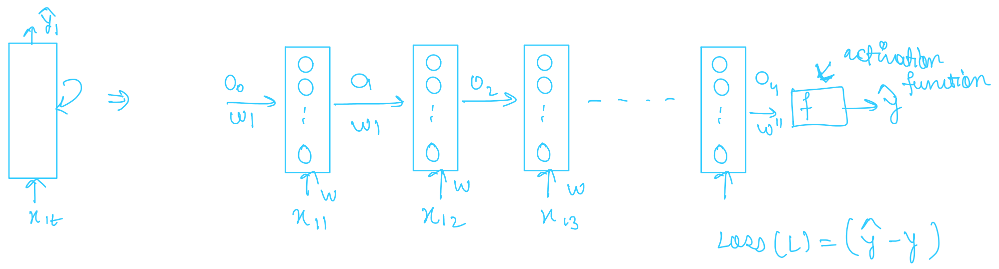
$$y = \begin{cases} x & x \geq 0 \\ a(e^{x} - 1), & x < 0 \end{cases}$$

★★★ V. Imp

### Vanishing Gradient problem

Deep network are hard to train because of the notorious vanishing gradient problem - as the gradient is back propogated to Earlier layers, the repeated multiplication may make the gradient infinitely small


weight
update
Error
$\delta^{L} = a^{L} - y^{t}$
Backpropagation
$x_1$
$x_2$
$x_n$
$w_1$
$w_2$
$w_n$
$x_0 = 1$
$w_0$
$\Sigma$
Input
$net = \sum_{i=0}^{n} w_i x_i$
$O = \sigma(net) = \frac{1}{1 + e^{-net}}$
Optimization such as Gradient descent
Output
Calculation of cost function



we get : $\frac{\partial L}{\partial \hat{y}}$

We need ① $w'_{new} = w'_{old} - \frac{\partial L}{\partial w'}$

$$\eta = 1$$

$$\boxed{W'_{new} = W'_{old} - \eta \frac{\partial L}{\partial W'_{old}}}$$

$$\frac{\partial L}{\partial W'_{old}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W'_{old}}$$

② $$\frac{\partial L}{\partial W'_{old}} = \underbrace{\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_4} \cdot \frac{\partial O_4}{\partial W'_{old}}}_{\text{Chain Rule}}$$

depends on activation function

Vanishing gradient problem is a problem caused by some of the activation functions in neural networks where the gradient of network's become increasingly small in the early layers thereby making it extremely difficult to understand the contribution of a particular node (attribute) to the output of the network.

** Since ReLU functions have a derivative of either 0 or 1, there is no question of vanishing derivatives. The derivatives can be propogated through the network easily in case of ReLU, therefore making it more favorable for deep learning problems.

**Bias:** Bias is the algorithm's tendency to consistently learn the wrong things by taking into account all the information in the data.

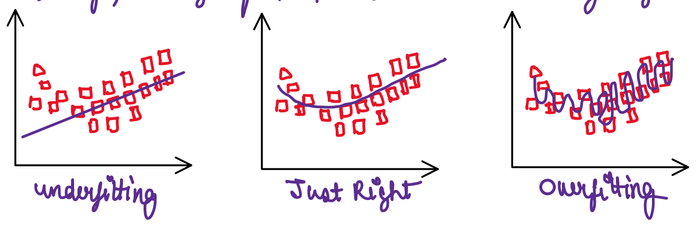
**Variance:** Variance is an error from sensitivity to small fluctuation in the training set.

Underfitting : High Bias and high variance

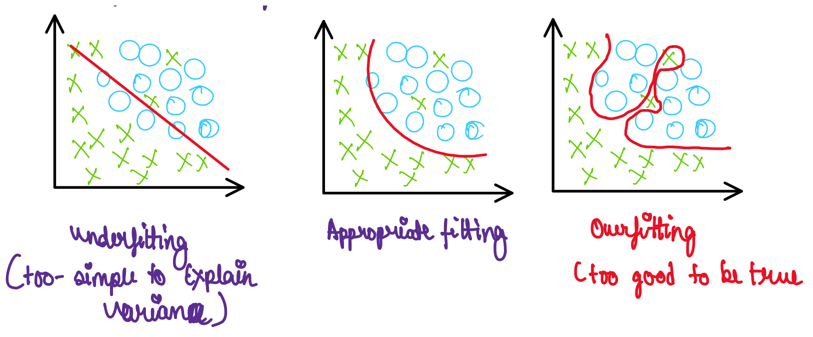
Overfitting : Low Bias and high variance

( Less training error and high test error)

Overfitting, underfitting & optimal model in case of Regression



Overfitting, underfitting & optimal model in case of Classification

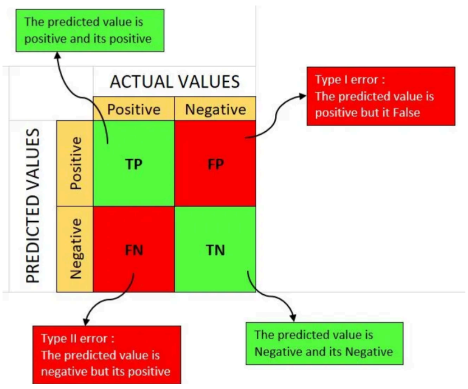


Underfitting (too- simple to explain variance)
Appropriate fitting
Overfitting (too good to be true

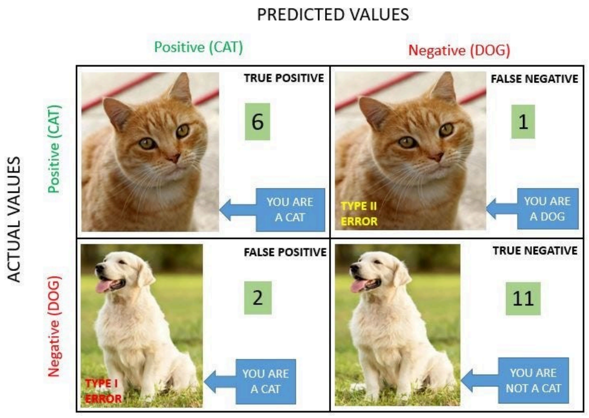
***V.V.Imp

Confusion Matrix : A confusion matrix is an N x N matrix used for Evaluating the performance of a classification model, where N is the number of target classes.

The matrix compares the actual target values with those predicted by the machine learning model.

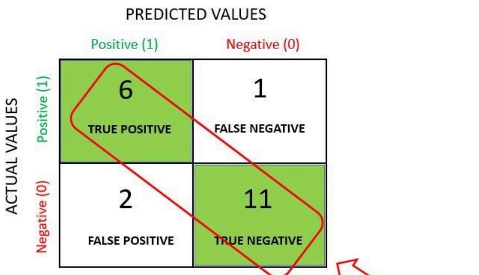


The predicted value is positive and its positive
ACTUAL VALUES
Positive
Negative
PREDICTED VALUES
Positive
Negative
TP
FP
FN
TN
Type I error : The predicted value is positive but it False
Type II error : The predicted value is negative but its positive
The predicted value is Negative and its Negative



- True Positives (TP): when the actual value is Positive and predicted is also Positive.
- True negatives (TN): when the actual value is Negative and prediction is also Negative.
- False positives (FP): When the actual is negative but prediction is Positive. Also known as the Type 1 error
- False negatives (FN): When the actual is Positive but the prediction is Negative. Also known as the Type 2 error

**Accuracy** simply measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions.

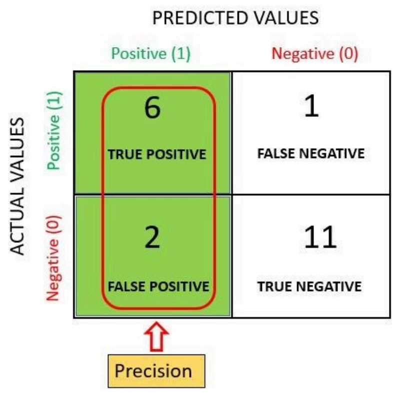


$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{6 + 11}{6 + 11 + 2 + 1} = 85\%$$

**Precision*:***

It is a measure of **correctness** that is achieved in **true prediction**. In simple words, it tells us how many predictions are ***actually positive*** out of all the ***total positive predicted***.

Precision is defined as the ratio of the total number of *correctly classified positive classes* divided l *itive classes*.

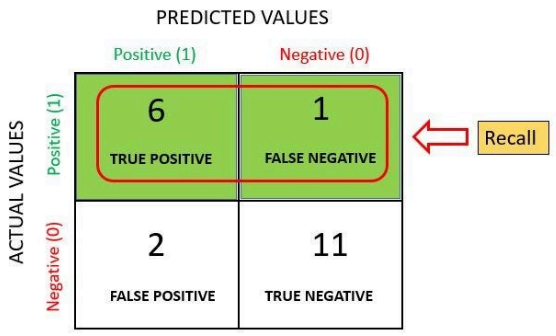


$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{\text{Predictions Actually Positive}}{\text{Total Predicted positive}} = \frac{6}{6+2} = 0.75$$

***Recall:***

It is a measure of **actual observations** which are predicted **correctly**, i.e. how many observations of positive class are actually predicted as positive. It is also known as **Sensitivity**. ***Recall*** is a valid choice of evaluation metric when we want to capture ***as many positives*** as possible.

Recall is defined as the ratio of the total number of *correctly classified positive classes* divide by the *total number of positive classes*.



$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}} = \frac{6}{6+1} = 0.85$$

***F1-Score***

The **F1 score** is a number between **0 and 1** and is the ***harmonic mean of precision and recall***. We use harmonic mean because it is not sensitive to extremely large values, unlike simple averages.

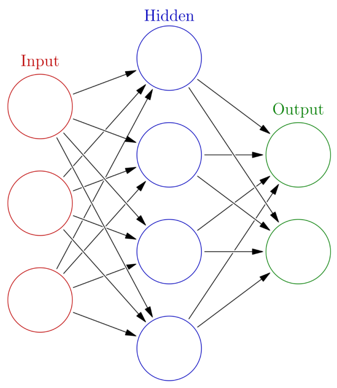
$$\text{F1-Score} = 2* \frac{(\text{Recall*Precision})}{(\text{Recall + Precision})} = 2* \frac{(0.85*0.75)}{(0.85+0.75)} = 0.79$$

## What is a neural network?

A **neural network** is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

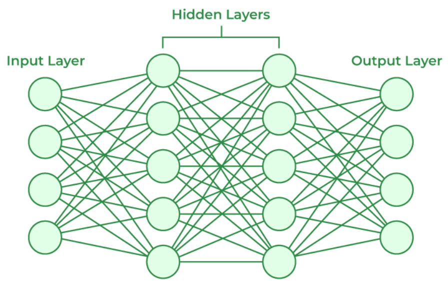
## Simple neural network architecture

A basic neural network has interconnected artificial neurons in three layers:
**Input Layer-** Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.
**Hidden Layer-** Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
**Output Layer -** The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

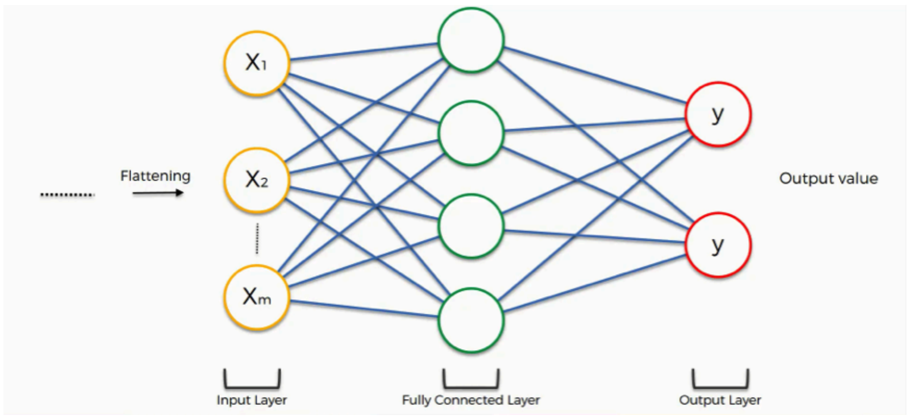


## Deep neural network architecture

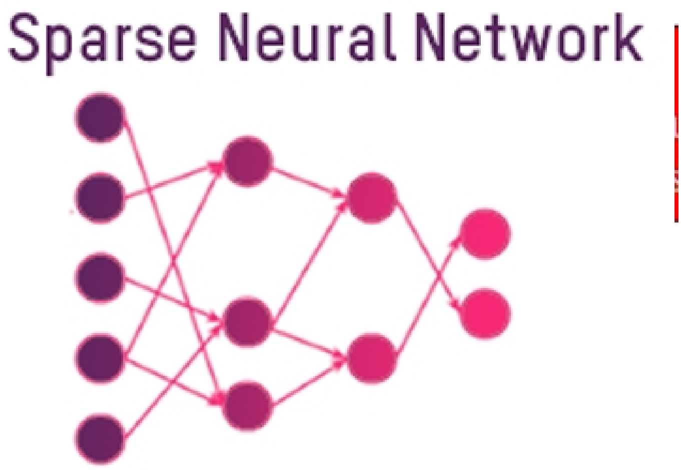
Deep neural networks, or deep learning networks, have several hidden layers with millions of artificial neurons linked together. A number, called weight, represents the connections between one node and another. The weight is a positive number if one node excites another, or negative if one node suppresses the other. Nodes with higher weight values have more influence on the other nodes.



A **fully connected neural network** (FCNN) is a type of artificial neural network where every node in one layer is connected to every node in the next layer. In a fully connected layer, each input node is connected to each output node, and each neuron applies a linear transformation to the input vector.



A **sparsely connected neural network**, or SCL, is a neural architecture where a large portion of the connections or neurons are removed, leaving only the most relevant components. In an SCL, each neuron is only connected to a limited number of other neurons.

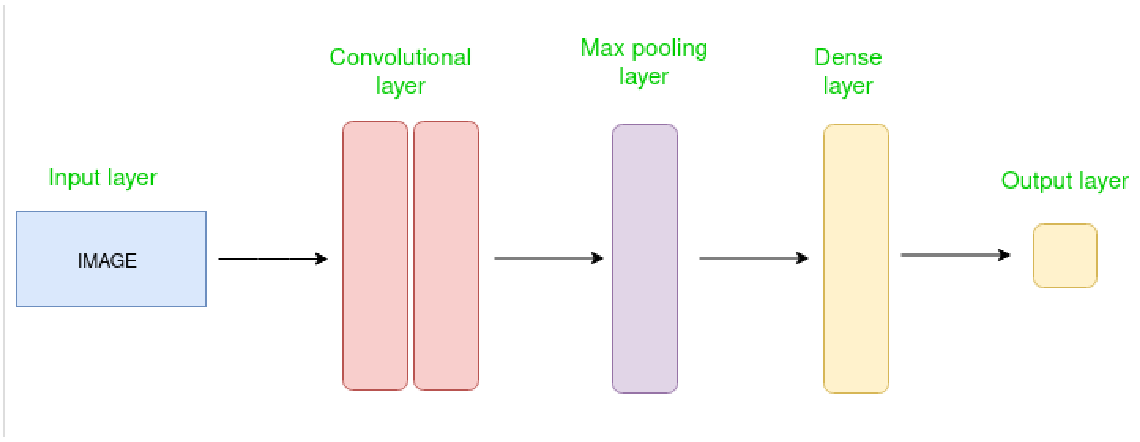


**Convolution Neural Network**

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

**CNN architecture**

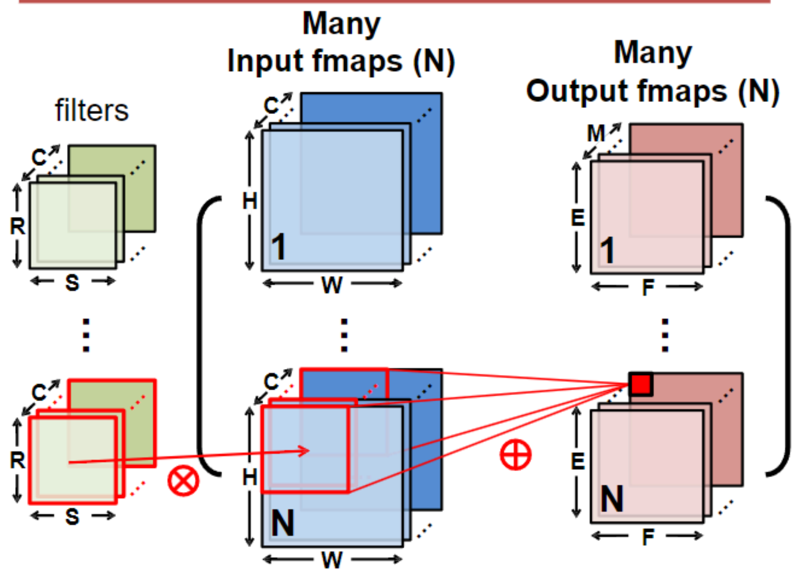
Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.



*Simple CNN architecture*

The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

## Convolution (CONV) Layer



- **N – Number of input fmaps/output fmaps (batch size)**
- **C – Number of 2-D input fmaps /filters (channels)**
- **H – Height of input fmap (activations)**
- **W – Width of input fmap (activations)**
- **R – Height of 2-D filter (weights)**
- **S – Width of 2-D filter (weights)**
- **M – Number of 2-D output fmaps (channels)**
- **E – Height of output fmap (activations)**
  **F – Width of output fmap (activations)**

## CONV Layer Tensor Computation

**Output fmaps (O)** **Biases (B)** **Input fmaps (I)** **Filter weights (W)**

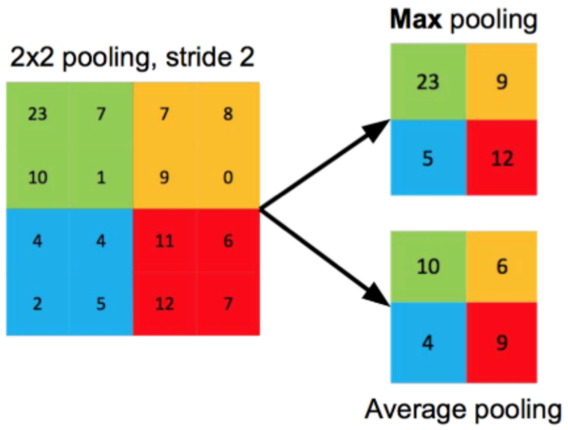
$$\mathbf{O}[n][m][x][y] = \text{Activation}(\mathbf{B}[m] + \sum_{i=0}^{R-1}\sum_{j=0}^{S-1}\sum_{k=0}^{C-1}\mathbf{I}[n][k][Ux+i][Uy+j] \times \mathbf{W}[m][k][i][j]),$$

$$0 \le n < N, 0 \le m < M, 0 \le y < E, 0 \le x < F,$$

$$E = (H - R + U)/U, F = (W - S + U)/U.$$

## Pooling (POOL) Layer

- Reduce resolution of each channel independently
- Overlapping or non-overlapping → depending on stride



| | | | |
|---|---|---|---|
| 23 | 7 | 7 | 8 |
| 10 | 1 | 9 | 0 |
| 4 | 4 | 11 | 6 |
| 2 | 5 | 12 | 7 |

Max pooling

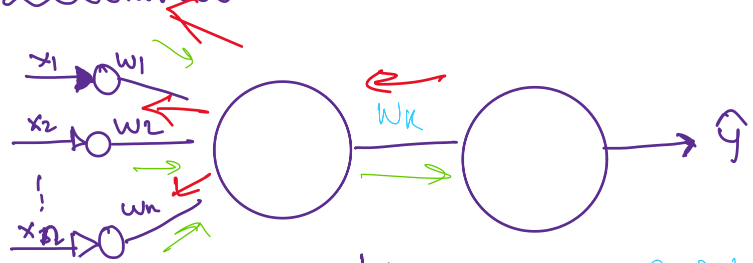
| | |
|---|---|
| 23 | 9 |
| 5 | 12 |

Average pooling

| | |
|---|---|
| 10 | 6 |
| 4 | 9 |

Increases translation-invariance and noise-resilience

Back Propagation



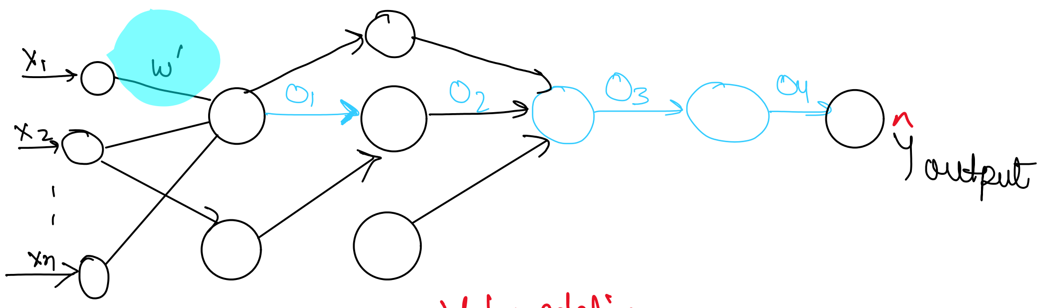
loss $= (y - \hat{y})^2$

$$w_{k_{new}} = w_{k_{old}} - \eta \frac{\partial L}{\partial w_k}$$

learning Rate

cost function $\Rightarrow \sum_{i=1}^{n} (y - \hat{y}$

$$w_{1_{new}} = w_{1_{old}} - \eta \frac{\partial L}{\partial w_1}$$



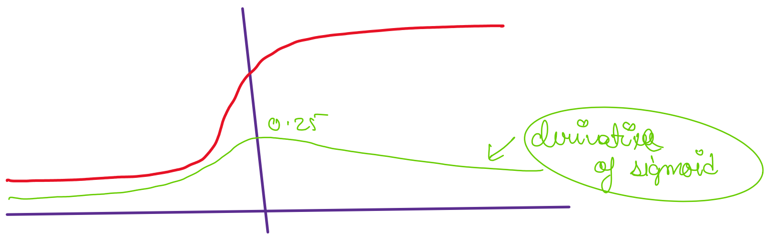
weight updation

$$w'_{new} = w'_{old} - \eta \overbrace{\frac{\partial L}{\partial w'_{old}}}^{\text{gradient}}$$

$$\frac{\partial L}{\partial w'_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_4} \cdot \frac{\partial O_4}{\partial w'_{old}}$$

$= \partial L \cdot \partial \hat{y} \cdot \partial O_4 \cdot \partial O_3$

$\partial \hat{y} \quad \partial O_4 \quad \partial O_3 \quad \partial w'_{old}$



Derivative of tanh $(0 \rightarrow 1)$

Vanishing Gradient $\rightarrow$ $\boxed{w_{11new} \approx w_{11old}}$