جامعة الجلالة
GALALA UNIVERSITY

Artificial Intelligence Science Program

Chapter 5: Neural Networks and Deep Learning

Gradient Descent

- Gradient Descent is a machine learning algorithm that operates iteratively to find the optimal values for its parameters. It takes into account, user-defined learning rate, and initial parameter values.
- Working: (Iterative)
  1. Start with initial values.
  2. Calculate cost.
  3. Update values using the update function.
  4. Returns minimized cost for our cost function

Update the value of weights

$$X = X - lr * \frac{d}{dX} f(X)$$

$Where,$

$X \quad = \; input$

$F(X) \; = \; output \;\; based \; on \; X$

$lr \quad = \; learning \; rate$

- Input for perceptron: $ino = w_1 * i_1 + w_2 * i_2 + w3 * b$
- Applying sigmoid function for predicted output : $outo = \frac{1}{1 + e^{-ino}}$
- Calculate the error: $Error = \sum \frac{1}{2}(target\ output - outo)^2$

- Changing the weight value based on gradient descent formula:

$$w = w - lr * \frac{\partial Error}{\partial w}$$

- **Calculating the derivative:** $$\frac{\partial Error}{\partial w} = \frac{\partial Error}{\partial outo} * \frac{\partial outo}{\partial ino} * \frac{\partial ino}{\partial w}$$

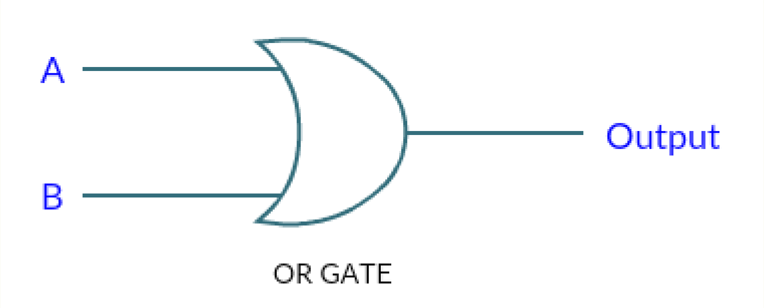
- **Individual derivatives:**

$$\frac{\partial Error}{\partial outo} = (outo - target\ output)$$

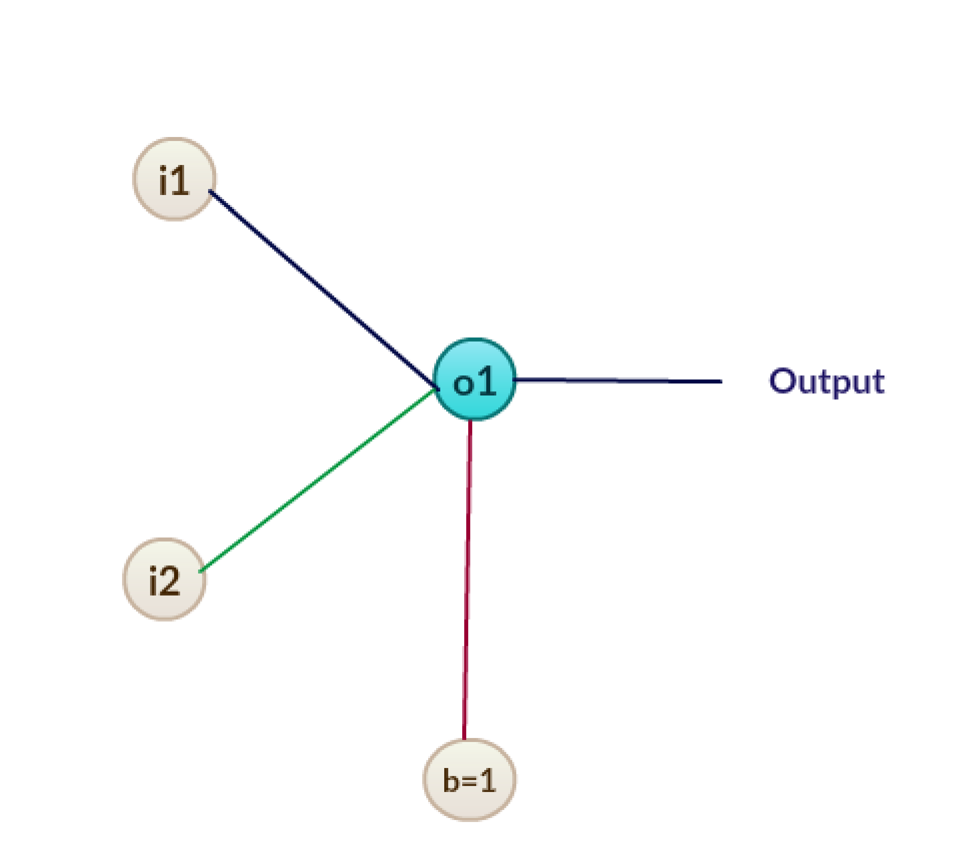
$$\frac{\partial outo}{\partial ino} = outo\,(1 - outo)$$

$$\frac{\partial ino}{\partial w} = input\ values$$

Example (logical OR Gate)



Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1



Input for o1 = w1*x1 + w2*x2 + b*x3

= 0.2*0 + 0.3*1 + 0.5*1

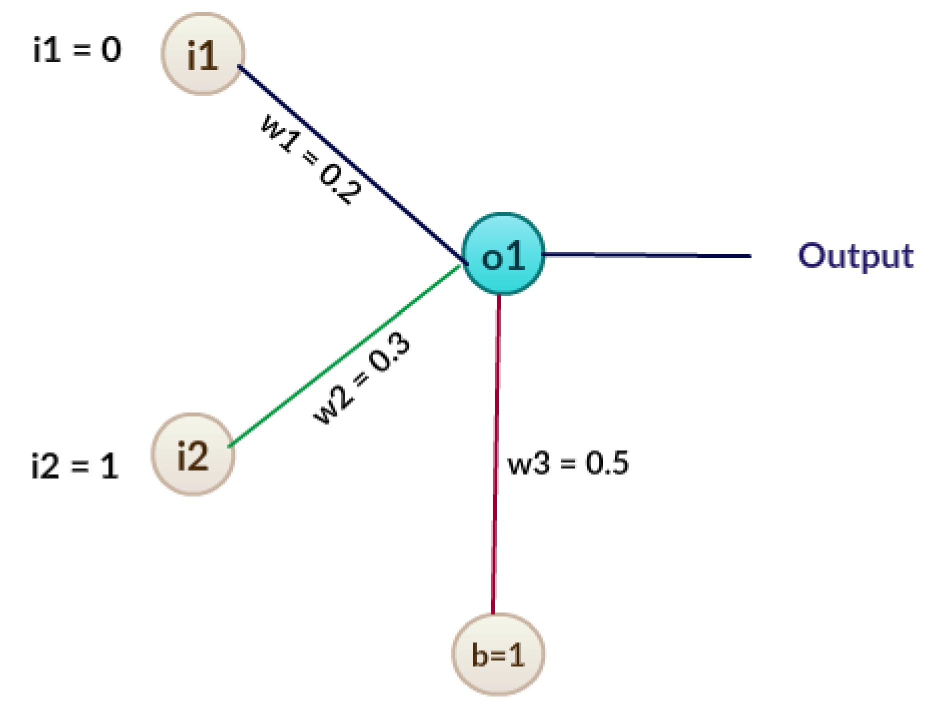
= 0 + 0.3 + 0.5

= 0.8

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

Output for 01 = $\frac{1}{1+e^{-0.8}}$

= 0.68997



$$MSE = \sum \frac{1}{2} * (target - output)^2$$

$$MSE = \frac{1}{2} * (1 - 0.68997)^2 = 0.048059$$

$$\frac{\partial Error}{\partial outo} = \frac{\partial}{\partial outo}\left(\frac{1}{2} * (target - output)^2\right)$$

$$\frac{\partial Error}{\partial outo} = \left(\frac{1}{2} * 2 * (target - output\ )\right) * \frac{\partial}{\partial outo}(target - output)$$

$$\frac{\partial Error}{\partial outo} = (target - output) * (-1)$$

$$\frac{\partial Error}{\partial outo} = output - target$$

- In our case:
- Output = 0.68997, Target = 1

$$\frac{\partial Error}{\partial out\,o_1} = (0.68997 - 1) = -0.31003$$

Finding the second part of the derivative: $\frac{\partial outo}{\partial ino} = outo\,(1 - outo)$

- **Value of outo1:** $outo_1 = \left(\frac{1}{1 + e^{-ino_1}}\right)$
- **Finding the derivative with respect to ino1:** $\frac{\partial outo_1}{\partial ino_1} = \frac{\partial}{\partial ino_1}\left(\frac{1}{1 + e^{-ino_1}}\right)$
- **Simplifying it a bit to find the derivative easily:** $\frac{\partial outo_1}{\partial ino_1} = \frac{\partial}{\partial ino_1}\,(1 + e^{-ino_1})^{-1}$
- **Applying chain rule and power rule:** $\frac{\partial outo_1}{\partial ino_1} = -1(1 + e^{-ino_1})^{-2} * \frac{\partial}{\partial ino_1}(1 + e^{-ino_1})$

- **Applying sum rule:** $$\frac{\partial outo_1}{\partial ino_1} = -1(1 + e^{-ino_1})^{-2} * \left(\frac{\partial}{\partial ino_1}(1) + \frac{\partial}{\partial ino_1}(e^{-ino_1})\right)$$
- **The derivative of constant is zero:** $$\frac{\partial outo_1}{\partial ino_1} = -1(1 + e^{-ino_1})^{-2} * \left(0 + \frac{\partial}{\partial ino_1}(e^{-ino_1})\right)$$
- **Applying exponential rule and chain rule:**

$$\frac{\partial outo_1}{\partial ino_1} = -1(1 + e^{-ino_1})^{-2} * \left(e^{-ino_1} * \frac{\partial}{\partial ino_1}[-ino_1]\right)$$

- **Simplifying it a bit:** $$\frac{\partial outo_1}{\partial ino_1} = -1(1 + e^{-ino_1})^{-2} * (e^{-ino_1} * -1)$$
- **Multiplying both negative signs:** $$\frac{\partial outo_1}{\partial ino_1} = (1 + e^{-ino_1})^{-2} * (e^{-ino_1})$$

- **Put the negative power in the denominator:** $$\frac{\partial outo_1}{\partial ino_1} = \frac{(e^{-ino_1})}{(1 + e^{-ino_1})^2}$$
- **Simplifying it:** $$\frac{\partial outo_1}{\partial ino_1} = \frac{1 * (e^{-ino_1})}{(1 + e^{-ino_1}) * (1 + e^{-ino_1})}$$
- **Further simplification:** $$\frac{\partial outo_1}{\partial ino_1} = \frac{1}{(1 + e^{-ino_1})} * \frac{(e^{-ino_1})}{(1 + e^{-ino_1})}$$
- **Adding +1–1:** $$\frac{\partial outo_1}{\partial ino_1} = \frac{1}{(1 + e^{-ino_1})} * \frac{(e^{-ino_1}) + 1 - 1}{(1 + e^{-ino_1})}$$
- **Separate the parts:** $$\frac{\partial outo_1}{\partial ino_1} = \frac{1}{(1+e^{-ino_1})} * \left(\frac{(1+e^{-ino_1})}{(1+e^{-ino_1})} - \frac{1}{(1+e^{-ino_1})}\right)$$

- **Simplify:** $$\frac{\partial outo_1}{\partial ino_1} = \frac{1}{(1 + e^{-ino_1})} * (1 - \frac{1}{(1 + e^{-ino_1})})$$

- **Now we all know the value of outo1 from equation 1:** $$outo_1 = \left(\frac{1}{1 + e^{-ino_1}}\right)$$
- **From that we can derive the following final derivative:**

$$\frac{\partial outo_1}{\partial ino_1} = outo_1 * (1 - outo_1)$$

- **Calculating the value of our input:** $$\frac{\partial outo_1}{\partial ino_1} = 0.68997 * (1 - 0.68997) = 0.21391$$

-

Finding the third part of the derivative : $\frac{\partial ino}{\partial w} = input\ values$

- **Value of ino:** $ino_1 = w1 * i1 + w2 * i2 + w3 * 1$
- **Finding derivative:** All the other values except w2 will be considered constant here.

$$\frac{\partial ino_1}{\partial w_2} = 0 + i_2 + 0 = i_2$$

- **Calculating both values for our input:** $\frac{\partial ino_1}{\partial w_2} = i_2 = 1$
- **Putting it all together:** $\frac{\partial Error}{\partial outo_1} * \frac{\partial outo_1}{\partial ino_1} * \frac{\partial ino_1}{\partial w_2} = -0.31003 * 0.21391 * 1$

$$= -0.06631$$

- **Putting it in our main equation:** $w = w - lr * \frac{\partial Error}{\partial w_2}$

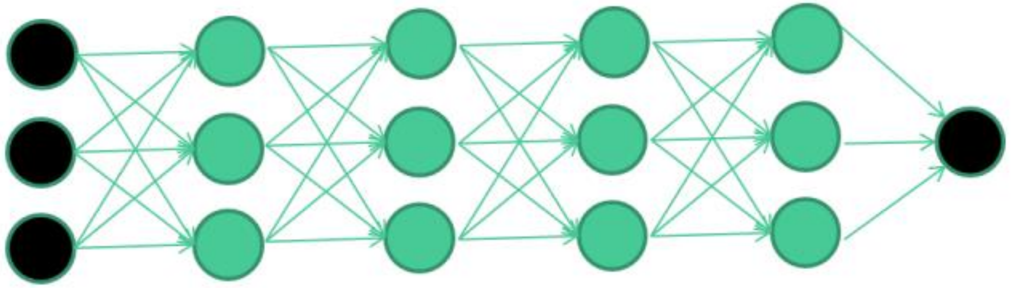
- **We can calculate:** $w2 = 1 - (0.05) * (-0.06631) = 1.0033155$

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0

etc …

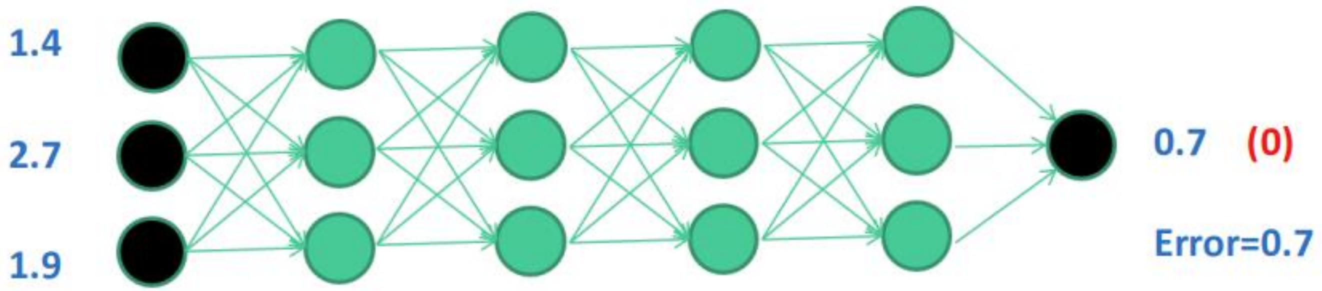
Train the deep neural network



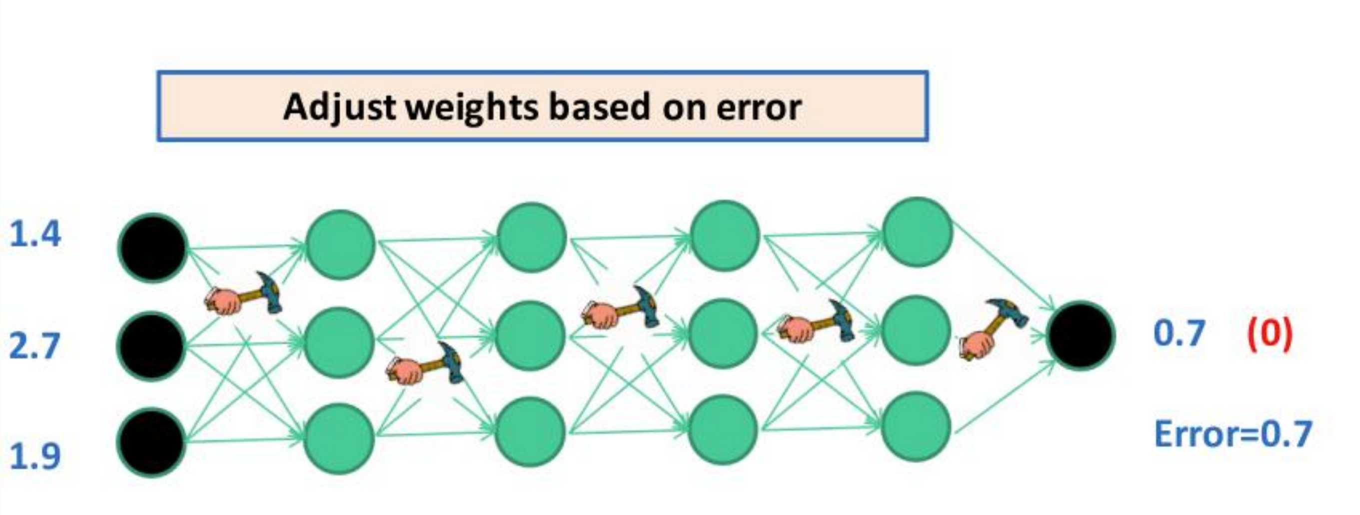
A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Initialize with random weights



Compare with the target output



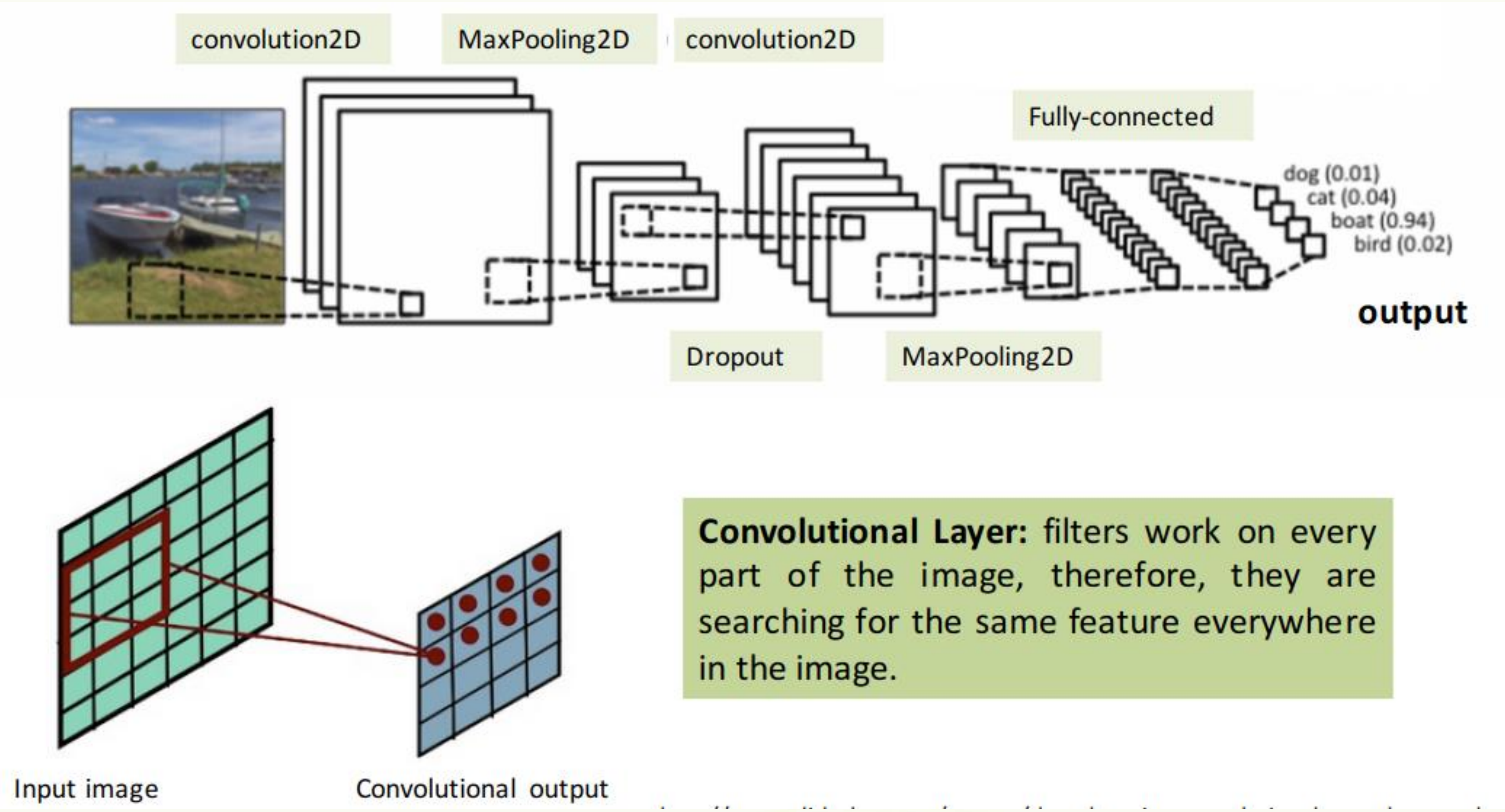
Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

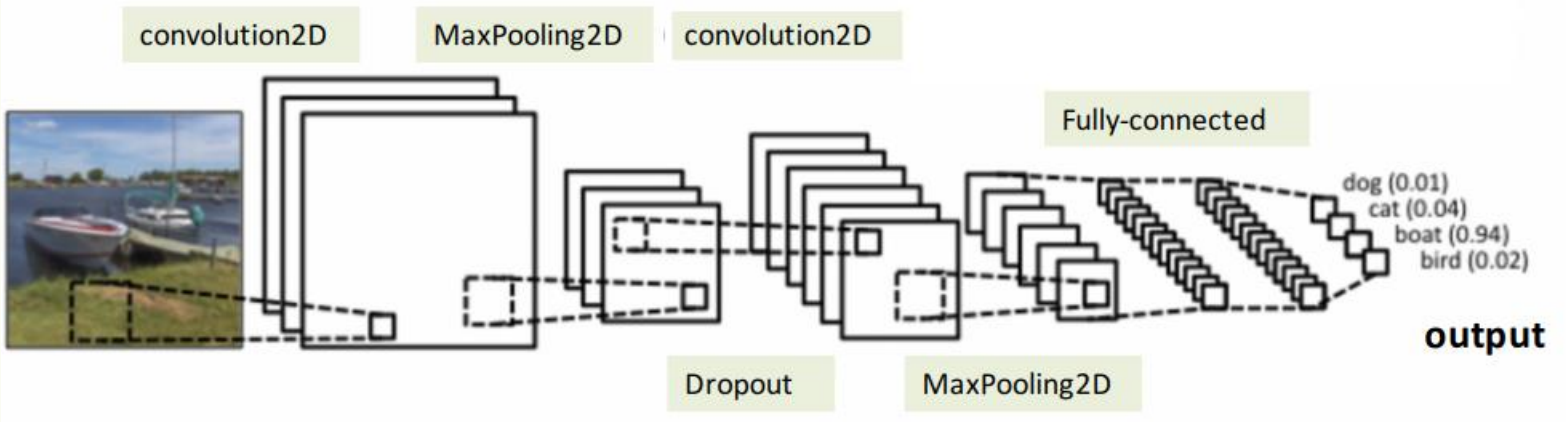
Algorithms for weight adjustment are designed to make changes that will reduce the error

Deep Learning

- The fundamental data structure in neural networks is the layer,
- layer is a data-processing module that takes as input one or more tensors and that outputs one or more tensors.

Deep Learning





Convolutional output

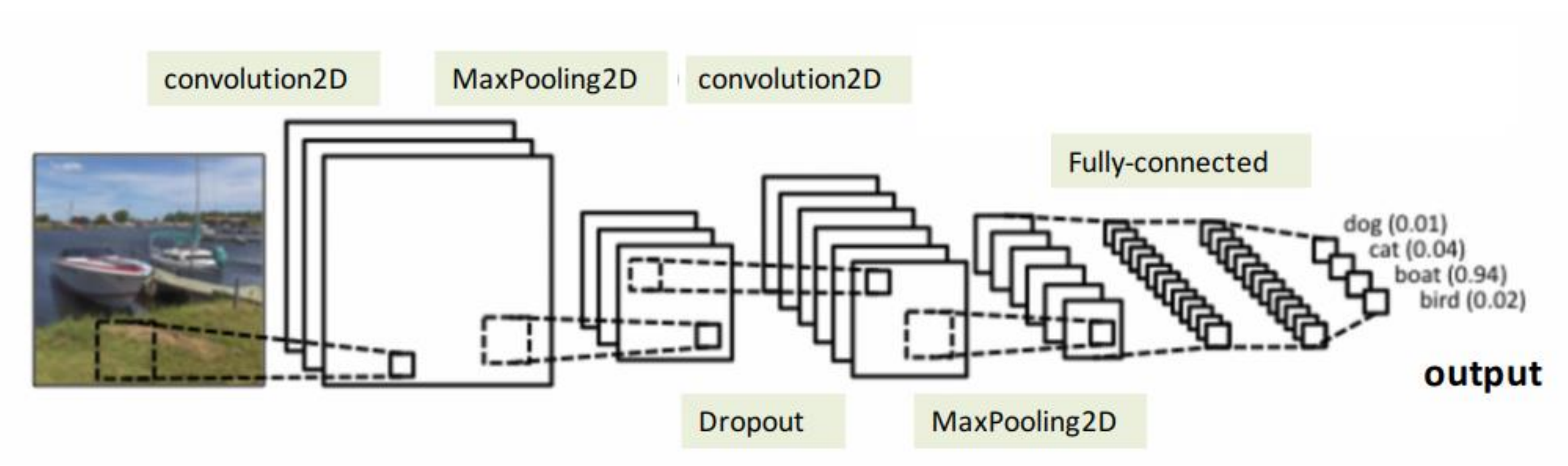
1	0	2	3
4	6	6	8
3	1	1	0
1	2	2	4

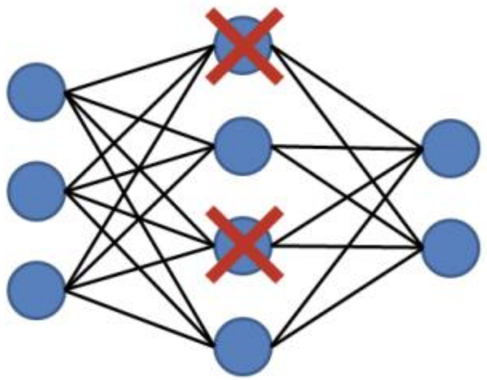
(2,2) →

MaxPooling

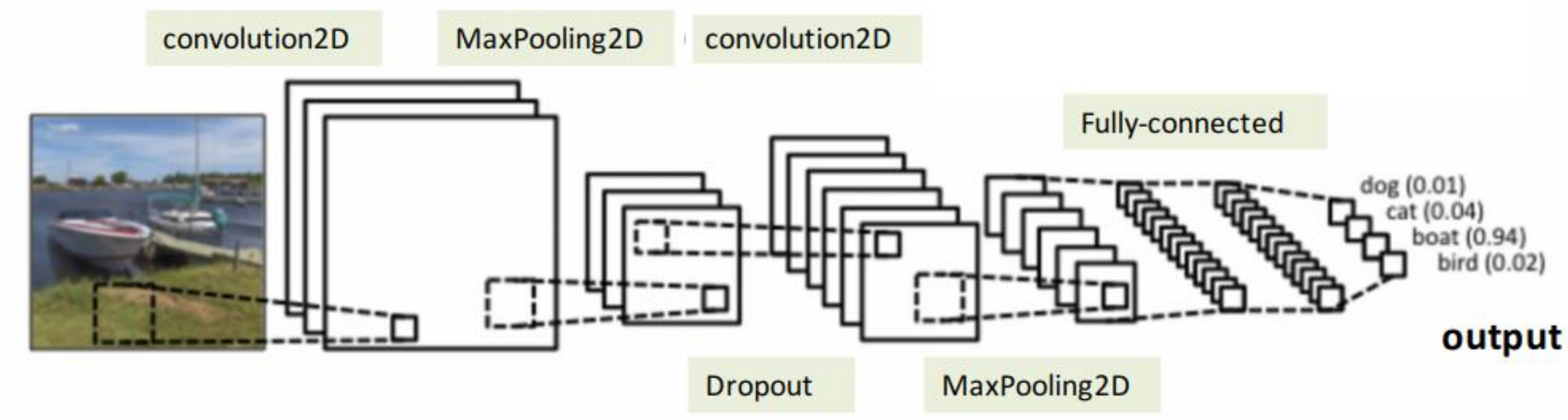
6	8
3	4

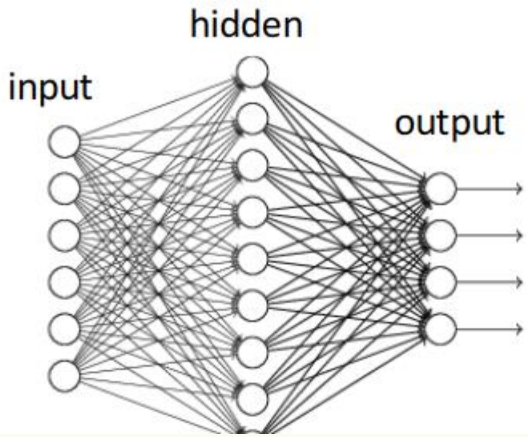
MaxPooling: usually present after the convolutional layer. It provides a down-sampling of the convolutional output





Dropout: randomly drop units along with their connections during training. It helps to learn more robust features by reducing complex co-adaptations of units and alleviate overfitting issue as well.





Fully-connected layer (dense): each node is fully connected to all input nodes, each node computes weighted sum of all input nodes. It has one-dimensional structure. It helps to classify input pattern with high-level features extracted by previous layers.

- http://alexlenail.me/NN-SVG/AlexNet.html
- https://github.com/ashishpatel26/Tools-to-Design-or-Visualize-Architecture-of-Neural-Network

- from keras import layers
- layer = layers.Dense(32, input_shape=(784,))