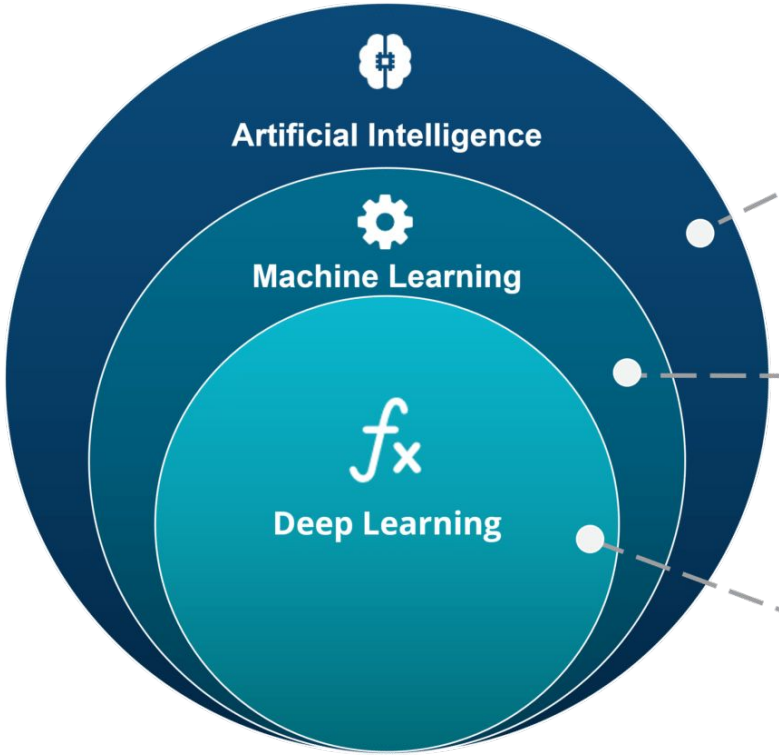
Presentation on Deep Learning

ES 612 : Artificial Intelligence

Aarchi Agrawal
20250017
MTech, CSE



ARTIFICIAL INTELLIGENCE

A technique which enables machines to mimic human behaviour

MACHINE LEARNING

Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING

Subset of ML which make the computation of multi-layer neural network feasible

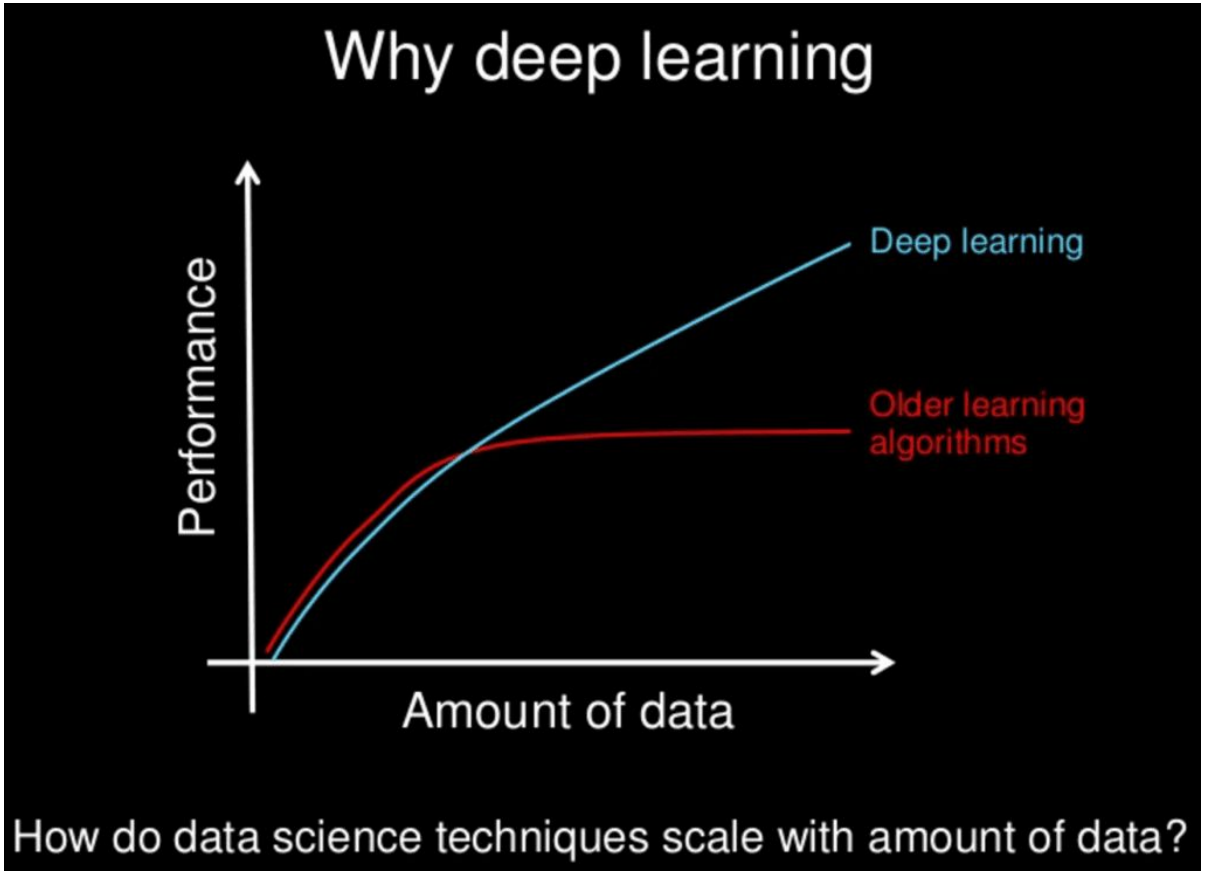
https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/

Content

- ➔ What is deep learning ?
- ➔ Why deep learning ?
- ➔ FeedForward Networks
- ➔ Activation Function
- ➔ Computation Graphs
- ➔ Gradient Descent
- ➔ Building computational graphs
- ➔ Applications

What is Deep Learning?

- ➔ Deep learning is a broad family of techniques for machine learning in which hypotheses take the form of complex algebraic circuits.
- ➔ The word “deep” is used because the circuits are typically organized into many layers.
- ➔ Tried to model networks of neurons in the brain with computational models.
- ➔ Learning method is based on data representation or feature learning.
- ➔ Data goes through multiple number of non linear transformations to obtain the output

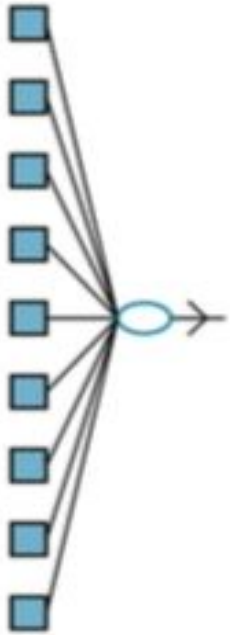


https://machinelearningmastery.com/what-is-deep-learning/

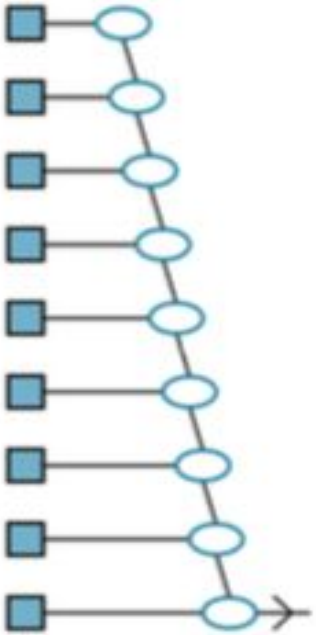
Reasons to use deep learning

- ➔ The expressive power of Linear and logistic regression models is limited.
- ➔ Can represent only linear functions
- ➔ Decision tree allow long paths for small fraction of input

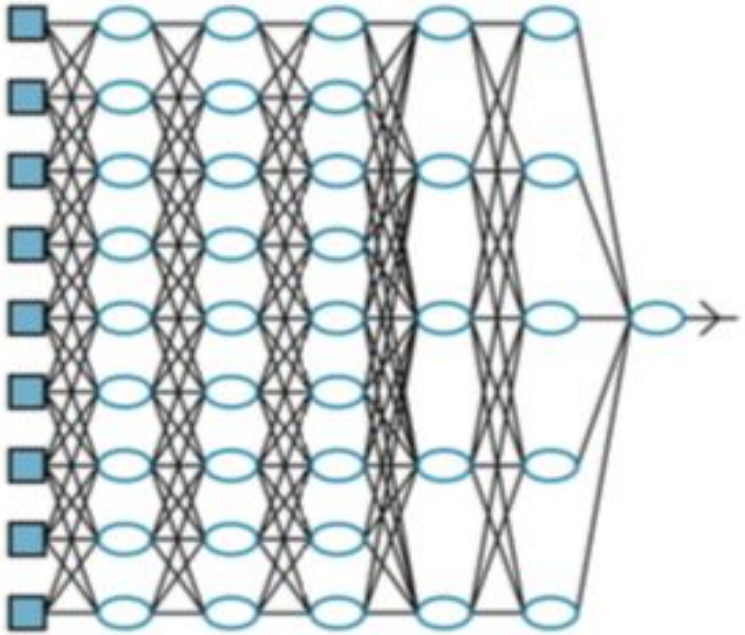
The basic idea of deep learning is to train circuits such that the computation paths are long, allowing all the input variables to interact in complex ways



(a)Shallow model
Ex- linear regression



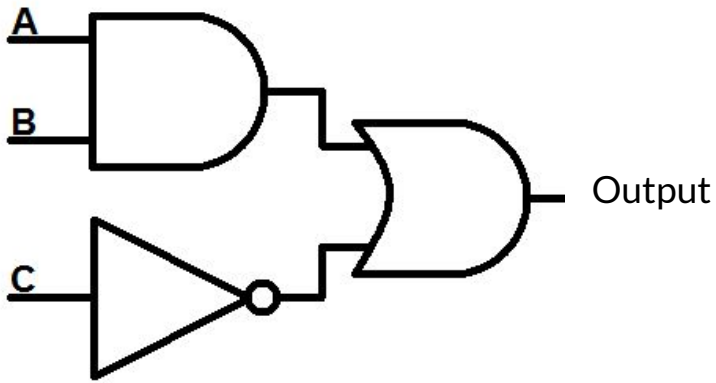
(b)Decision tree network

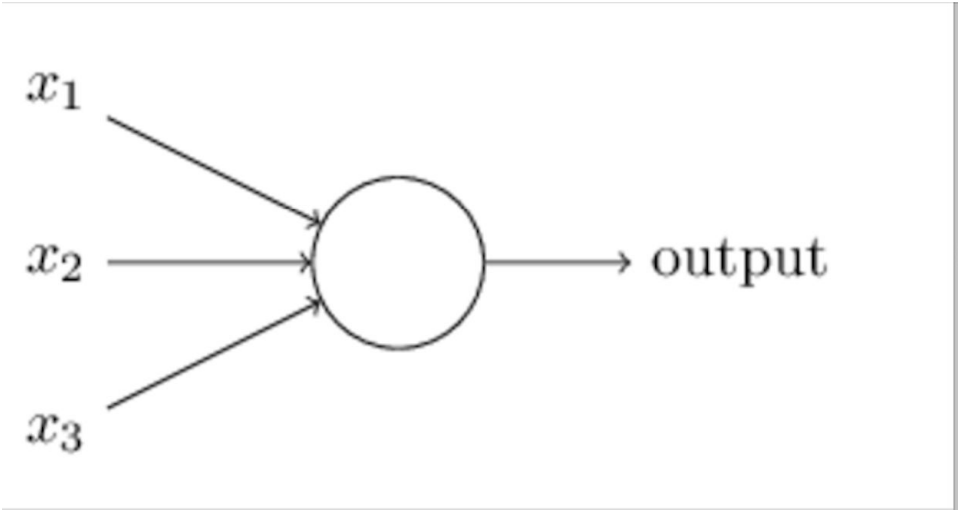


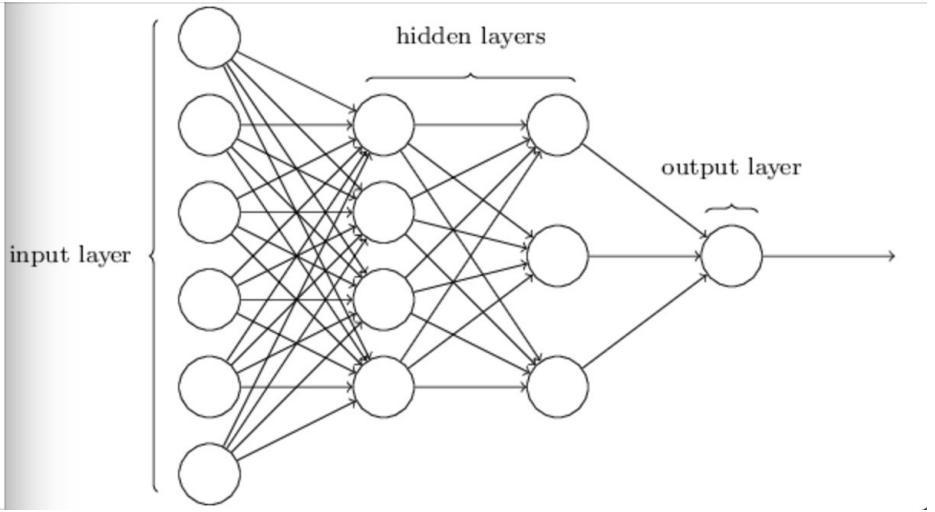
(c)Deep learning network

FeedForward Networks

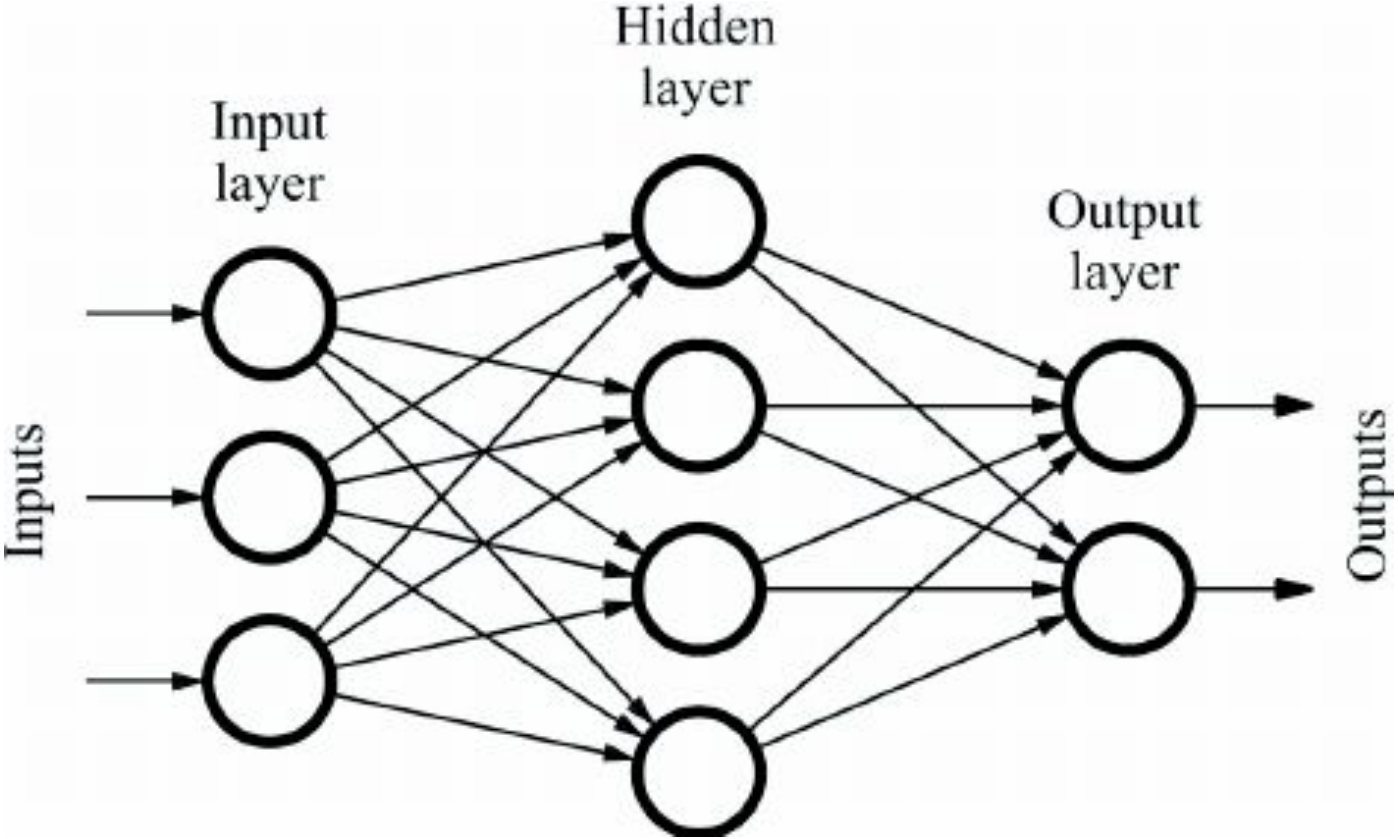
- ➔ Information flows through the function being evaluated in only one direction.
- ➔ Each node computes a function of its inputs and passes result to its successors in network.
- ➔ There are no cycles or loops present in feedforward network.
- ➔ Ex- Boolean circuit







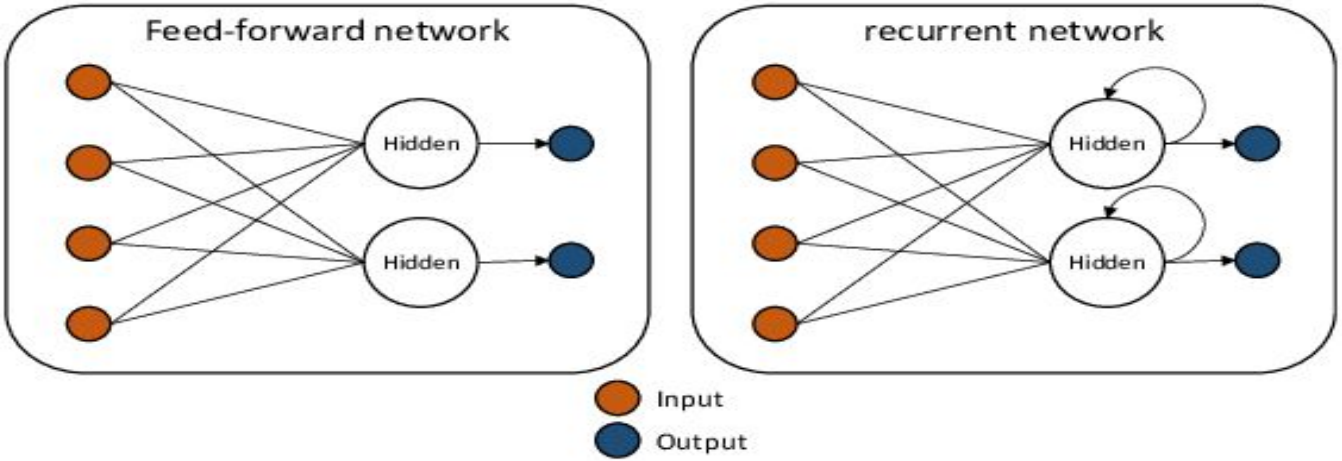
From a simple perceptron to deep feedforward network



https://medium.com/random-techpark/simple-feed-forward-neural-network-code-for-digital-handwritten-digit-recognition-a234955103d4

What if there is a loop in the network?

When the intermediate or final outputs are looped back into the inputs then it is called as recurrent network.



https://www.researchgate.net/figure/An-illustration-of-feed-forward-and-recurrent-networks_fig1_327612400

Network as function

In the network, every node that is present is called as a unit. It is the unit that calculates the weighted sum of the inputs and then use a nonlinear function to generate the output.

$$a_j = g_j\left(\sum_i w_{i,j} a_i\right) \equiv g_j(in_j),$$

Here, a_j = output of unit j

$w_{i,j}$ = weight attached to the link from unit i to j

g_j = activation function related with unit j

in_j = weighted sum of the inputs to unit j

In vector form

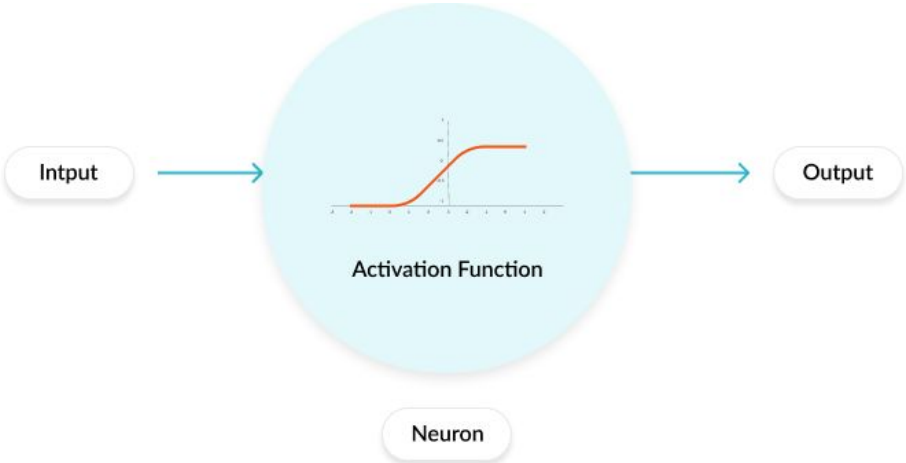
$$a_j = g_j(\mathbf{w}^\top \mathbf{x})$$

Here w = the vector of weights into unit j (including $w_{0,j}$)

X = vector of inputs (including the +1))

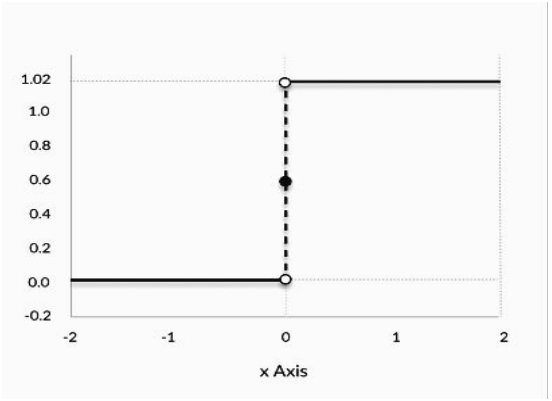
Activation function

The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.

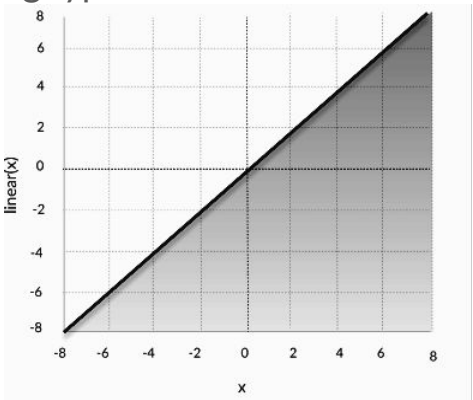


Activation function (contd)

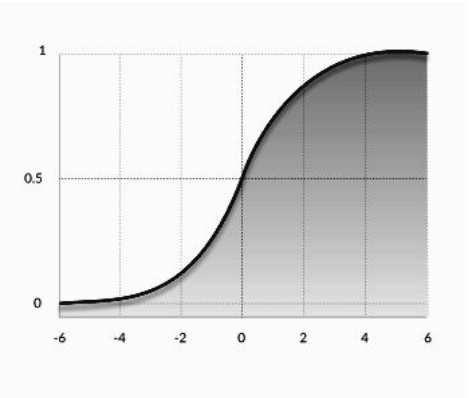
Activation function can be of the following types:



(a)Binary Step



(b)Linear Activation



(c)Non-linear

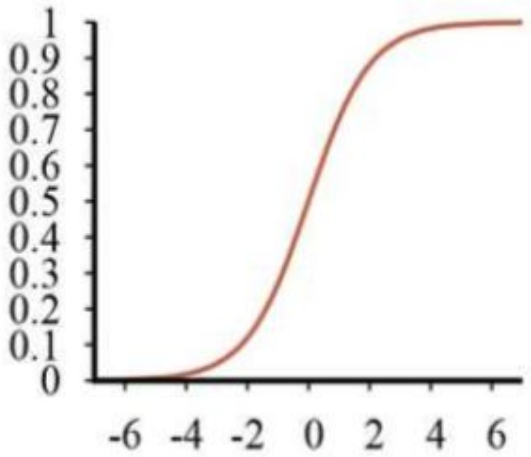
Why should we use non-linear activation function?

- ➔ To overcome the challenges of linear activation function
  - ◆ Backpropagation was not possible
  - ◆ All layers collapsed into one layer
- ➔ Learning and modelling complex data
- ➔ Universal approximation theorem

Types of non-linear activation function

➔ Sigmoid / Logistic function

$$\sigma(x) = 1/(1 + e^{-x})$$

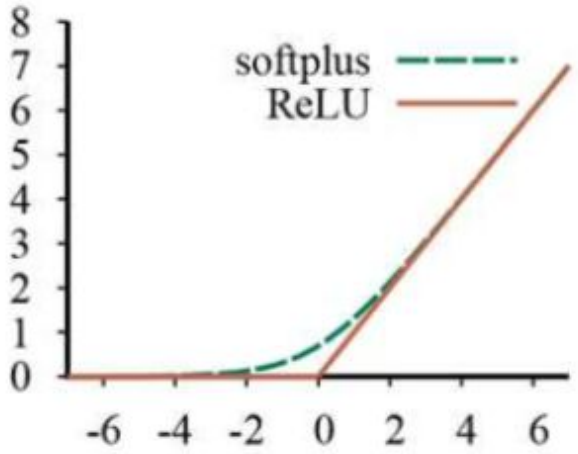


➔ ReLU or rectified linear unit

$$\mathrm{ReLU}(x) = \max(0,x)$$

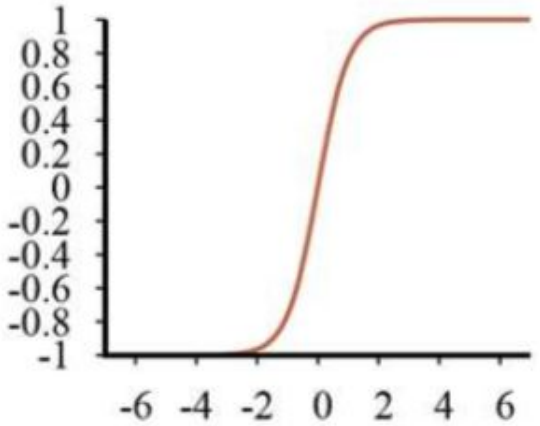
➔ Softplus or smooth version of Re

$$\mathrm{softplus}(x) = \log(1 + e^x)$$



➔ Tanh function

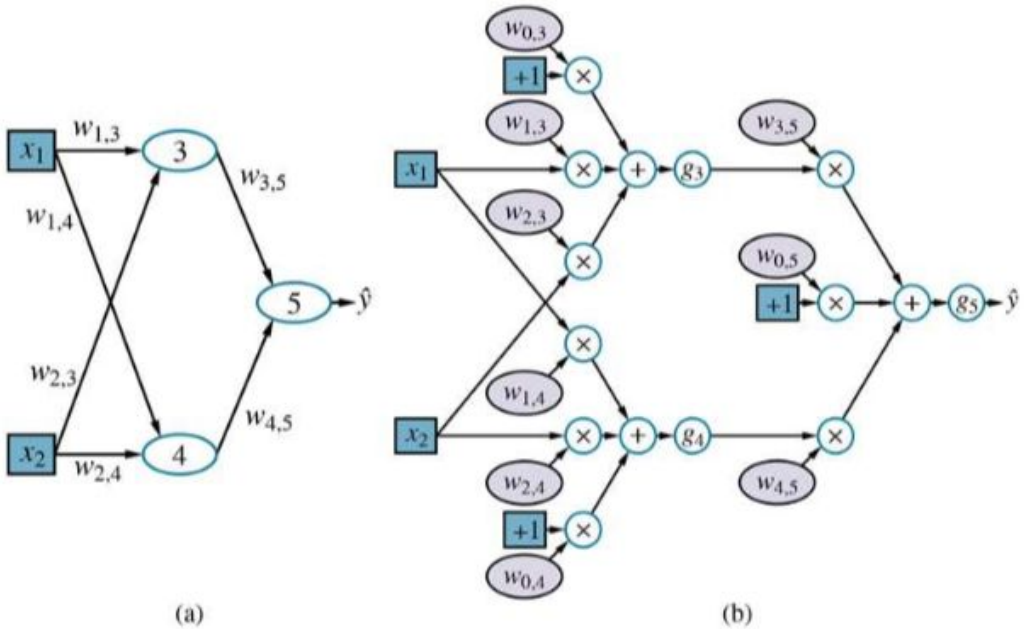
$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$



Computation Graph

- ➔ Computation graph is also called dataflow graph
- ➔ It is a circuit where very node is used to demonstrate an elementary computation.
- ➔ Using it we can explicitly represent each element of the overall computation
- ➔ It helps in better understanding.

Example



(a) A neural network with two inputs, one hidden layer of two units, and one output unit. Not shown are the dummy inputs and their associated weights. (b) The network in (a) unpacked into its full computation graph.

Output expression

$$
\begin{aligned}
\hat{y} &= g_5(in_5) = g_5(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4) \\
&= g_5(w_{0,5} + w_{3,5}g_3(in_3) + w_{4,5}g_4(in_4)) \\
&= g_5(w_{0,5} + w_{3,5}g_3(w_{0,3} + w_{1,3}x_1 + w_{2,3}x_2) \\
&\qquad + w_{4,5}g_4(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2))
\end{aligned}
$$

Output in vector form

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{g}^{(2)}(\mathbf{W}^{(2)}\mathbf{g}^{(1)}(\mathbf{W}^{(1)}\mathbf{x}))$$

$h_w(x)$ = output function of inputs and weights

W(1) = weights of first layer

W(2) = weights of second layer

g(1) = activation function in first layer

g(2) = activation function in second layer

Hence the graph is chain with weight matrices feeding into each layer

Gradient and learning

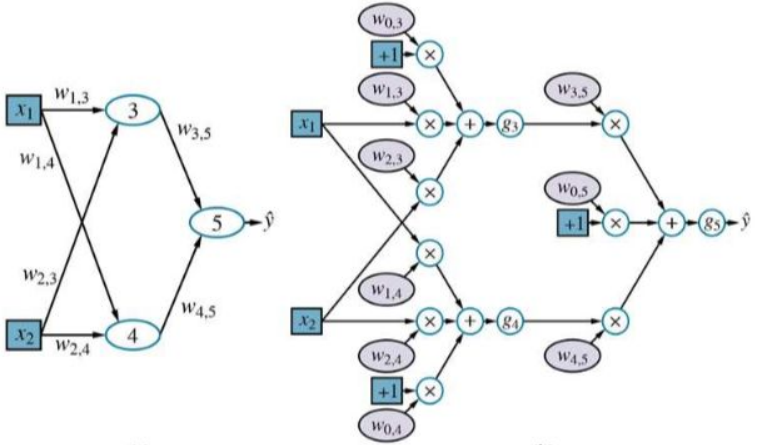
- ➔ It is an optimization technique
- ➔ We use gradient descent in order to learn the weights in computational graphs.
- ➔ Most popular means to optimise a neural network.

Loss equation

- ➔ We can use gradient descent in order to calculate the gradient of the loss function wrt the weights and then adjust the weights along the gradient direction to reduce the loss.
- ➔ For the network we saw earlier, we will use the squared loss function and calculate the gradient
- ➔ The true value is y and predicted value is y^ = $h_w(x)$, so

$$Loss(h_{\mathbf{w}}) = L_2(y, h_{\mathbf{w}}(\mathbf{x})) = \| y - h_{\mathbf{w}}(\mathbf{x}) \|^2 = (y - \hat{y})^2$$

Learning weight $w_{3,5}$



$$\frac{\partial}{\partial w_{3,5}} Loss(h_{\mathbf{w}}) = \frac{\partial}{\partial w_{3,5}} (y - \hat{y})^2 = -2(y - \hat{y}) \frac{\partial \hat{y}}{\partial w_{3,5}}$$

$$= -2(y - \hat{y}) \frac{\partial}{\partial w_{3,5}} g_5(in_5) = -2(y - \hat{y}) g_5'(in_5) \frac{\partial}{\partial w_{3,5}} in_5$$

$$= -2(y - \hat{y}) g_5'(in_5) \frac{\partial}{\partial w_{3,5}} \left(w_{0,5} + w_{3,5} a_3 + w_{4,5} a_4\right)$$

$$= -2(y - \hat{y}) g_5'(in_5) a_3.$$

$w_{3,5}$ is connected directly to the output, Similarly we can learn about the $w_{4,5}$

Learning weight $w_{1,3}$

$$\frac{\partial}{\partial w_{1,3}} Loss(h_{\mathbf{w}}) = -2(y-\hat{y})g_5'(in_5)\frac{\partial}{\partial w_{1,3}}\left(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4\right)$$

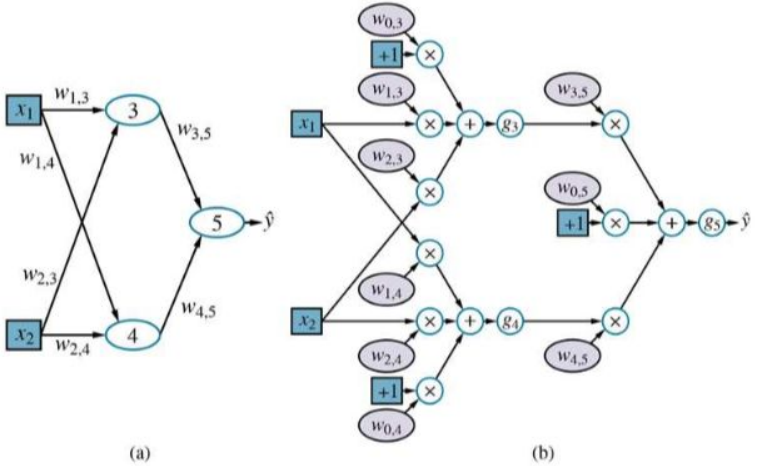
$$= -2(y-\hat{y})g_5'(in_5)w_{3,5}\frac{\partial}{\partial w_{1,3}}a_3$$

$$= -2(y-\hat{y})g_5'(in_5)w_{3,5}\frac{\partial}{\partial w_{1,3}}g_3(in_3)$$

$$= -2(y-\hat{y})g_5'(in_5)w_{3,5}g_3'(in_3)\frac{\partial}{\partial w_{1,3}}in_3$$

$$= -2(y-\hat{y})g_5'(in_5)w_{3,5}g_3'(in_3)\frac{\partial}{\partial w_{1,3}}\left(w_{0,3} + w_{1,3}x_1 + w_{2,3}x_2\right)$$

$$= -2(y-\hat{y})g_5'(in_5)w_{3,5}g_3'(in_3)x_1.$$



Since $w_{1,3}$ is not directly connected to output we have to apply chain rule more number of times, Similarly we can do for weight $w_{2,4}$

Relationships between weights learnt

Suppose $\Delta_5 = 2(\hat{y} - y)g_5'(in_5)$

then gradient with respect to $w_{3,5}$ is just $\Delta_5 a_3$

Now if , $\Delta_3 = \Delta_5 w_{3,5} g_3'(in_3)$

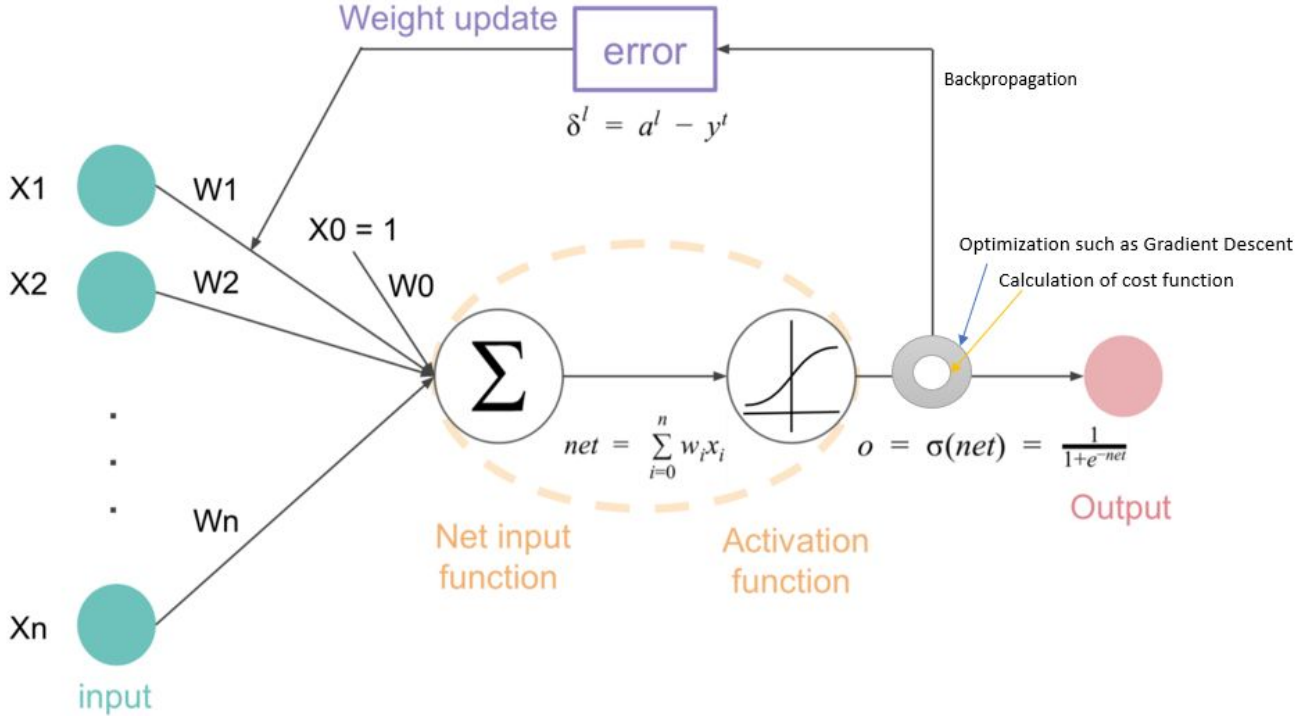
Then the gradient for $w_{1,3}$ becomes Δ_3X1

So the error at input to unit 3 is related to the error that we have at unit 5.

This is call **back-propagation**, because gradient calculation proceeds backwards through the network.

Backpropagation

- "Backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent.
- Calculates the gradient of the error function with respect to the neural network weights.
- First calculates the gradient of the final layer of weights and then calculation of gradient of the first layer of weights.
- Computation of the gradient for the previous layer uses partial computations of the gradient of the layers present ahead.
- Useful for efficient computation of the gradient at each layer.



https://datascience.stackexchange.com/questions/44703/how-does-gradient-descent-and-backpropagation-work-together

Vanishing gradient

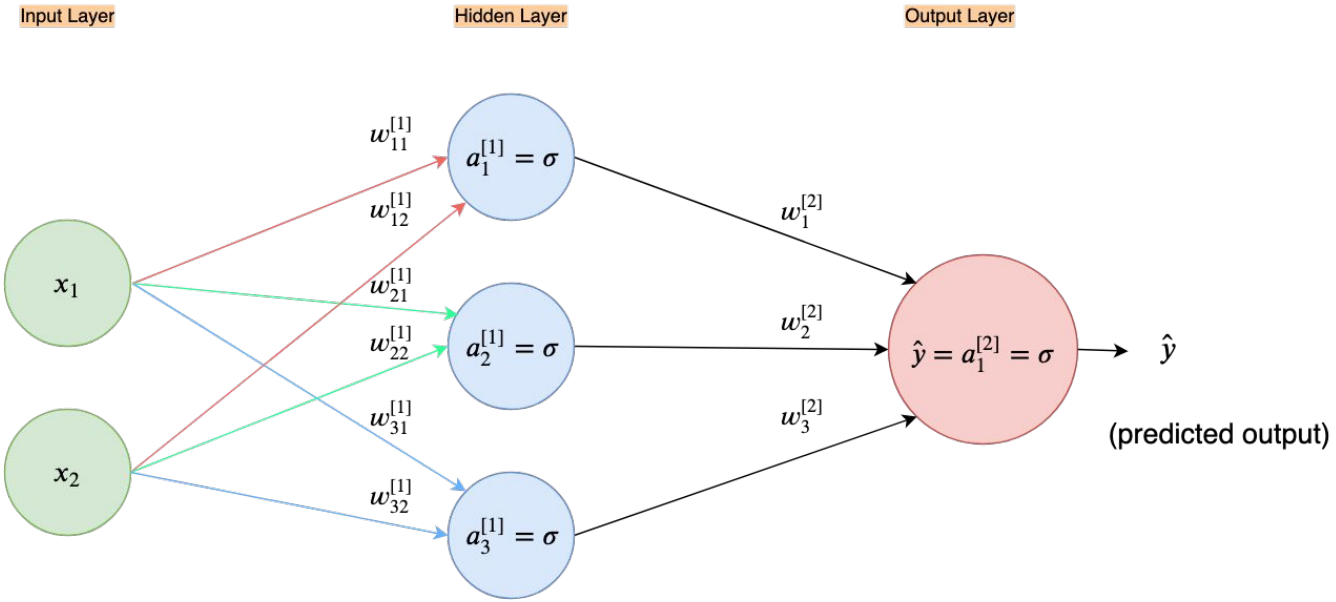
- ➔ Gradient expressions have factors of local derivatives
- ➔ These derivatives are non-negative but can be very small value close to zero.
- ➔ Which implies that derivative is zero or close to zero.
- ➔ Ex- sigmoid, softplus, tanh functions
- ➔ Due to this when the weights associated with that unit are changed it will cause a negligible effect on the output.
- ➔ Leading to fading the error signals that are propagated backwards in the network.

Building computational graphs

A computational graph is defined as a directed graph where the nodes correspond to mathematical operations. Computational graphs are a way of expressing and evaluating a mathematical expression.

Major parts include:

1. Input layer
2. Output layer
3. Hidden Layer



https://towardsai.net/p/machine-learning/nothing-but-numpy-understanding-creating-neural-networks-with-computational-graphs-from-scratch-6299901091b0

Input Layer

- ➔ Here the training or test set is encoded as values that can be given to the nodes.
- ➔ Factored data are encoded for the n input attributes, if these attributes are boolean than true can be assigned 1 and false can be assigned 0 or -1. If the values are integer or real then are they are scaled or used as it is.
- ➔ Images can be encoded as array-like structure so as to keep the adjacency property safe.
- ➔ One hot encoding can be used for categorical data.

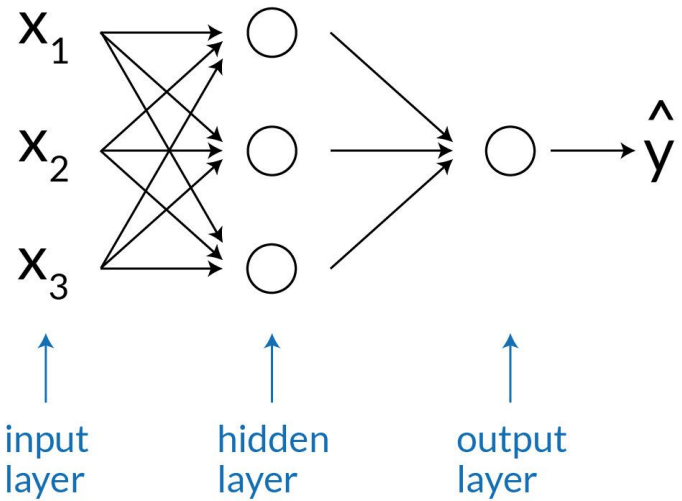
Output Layer

- ➔ The data values can be encoded in the same way as we did in the input layer.
- ➔ For the predicted values, since the loss is non-zero we need to measure the loss.
- ➔ Mostly the predicted values are interpreted as probabilities.
- ➔ Use negative log likelihood or cross entropy as loss function .
- ➔ We need to minimize this loss function.

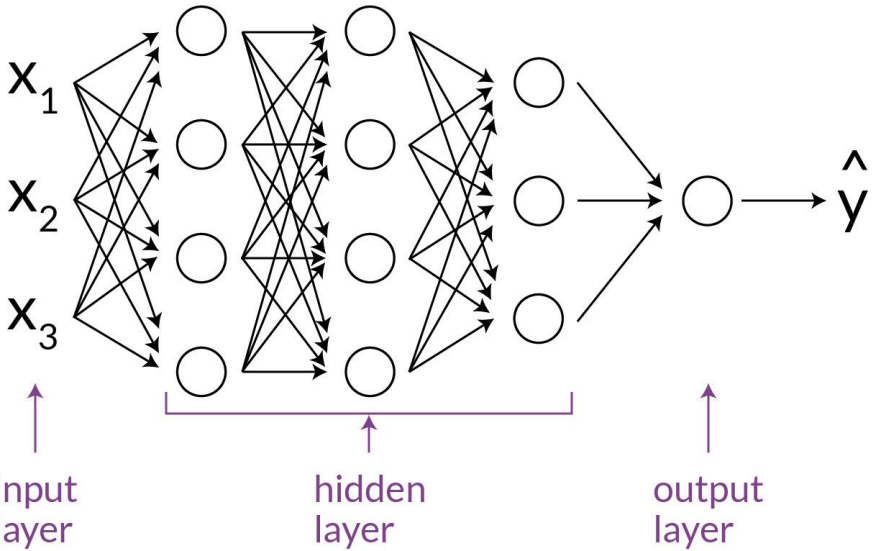
Hidden Layer

- ➔ Several intermediate computations are performed before outputting a result from the input
- ➔ Hidden layer are used to store these different representations of input .
- ➔ These intermediate representations are meaningful.
- ➔ Better learning are obtained from deep and narrow network instead of networks that are shallow and wide.

Shallow Neural Network



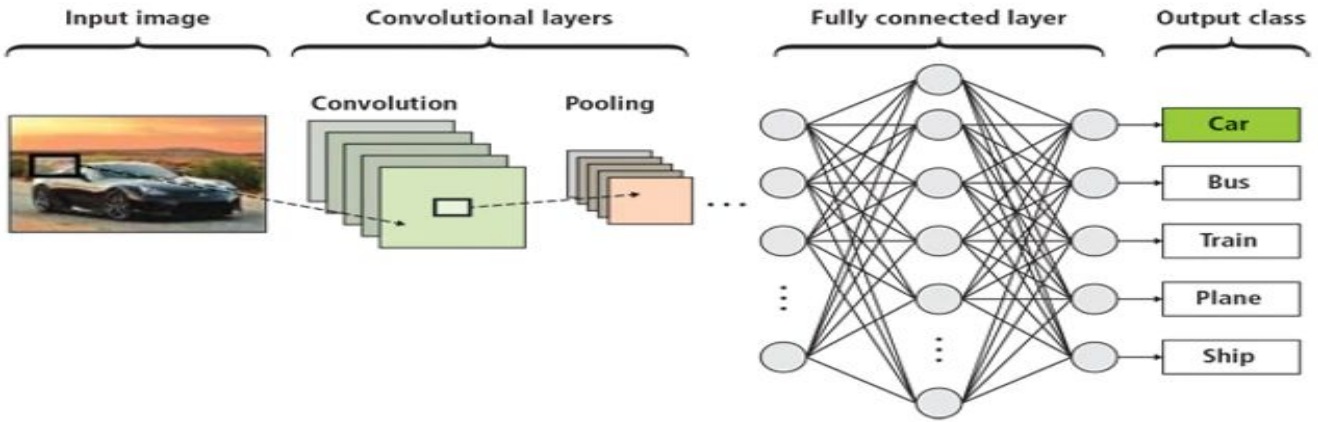
Deep Neural Network



https://www.druva.com/blog/understanding-neural-networks-through-visualization/

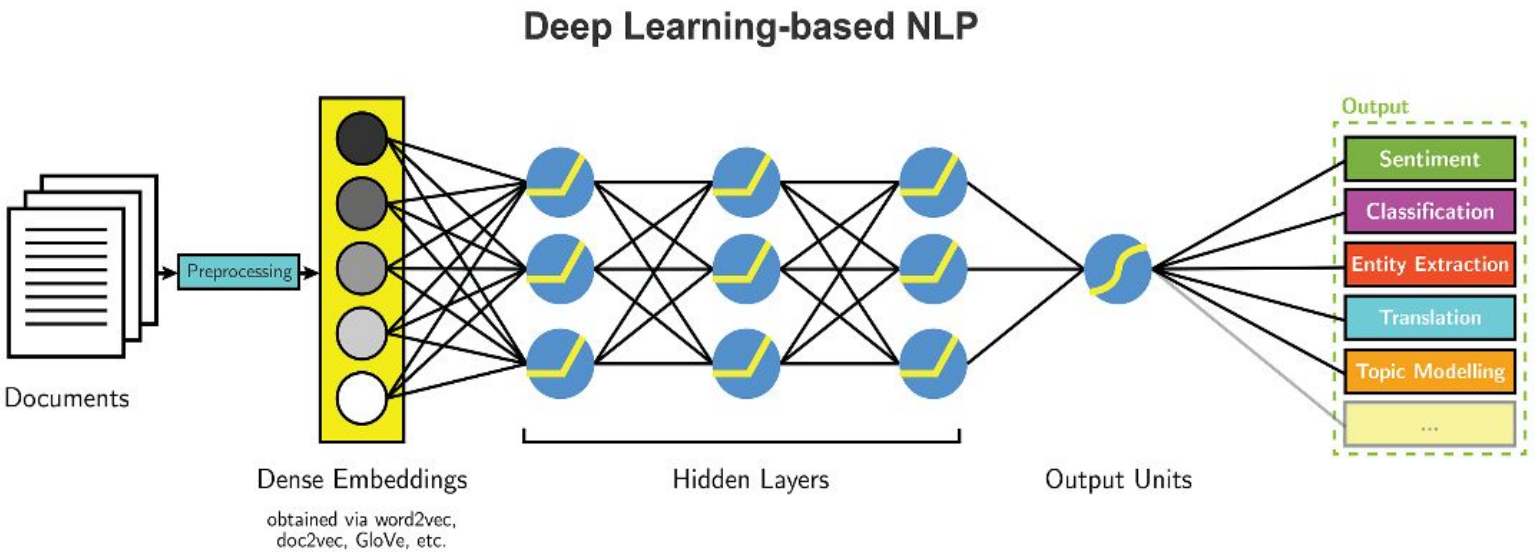
Applications

➔ Computer vision



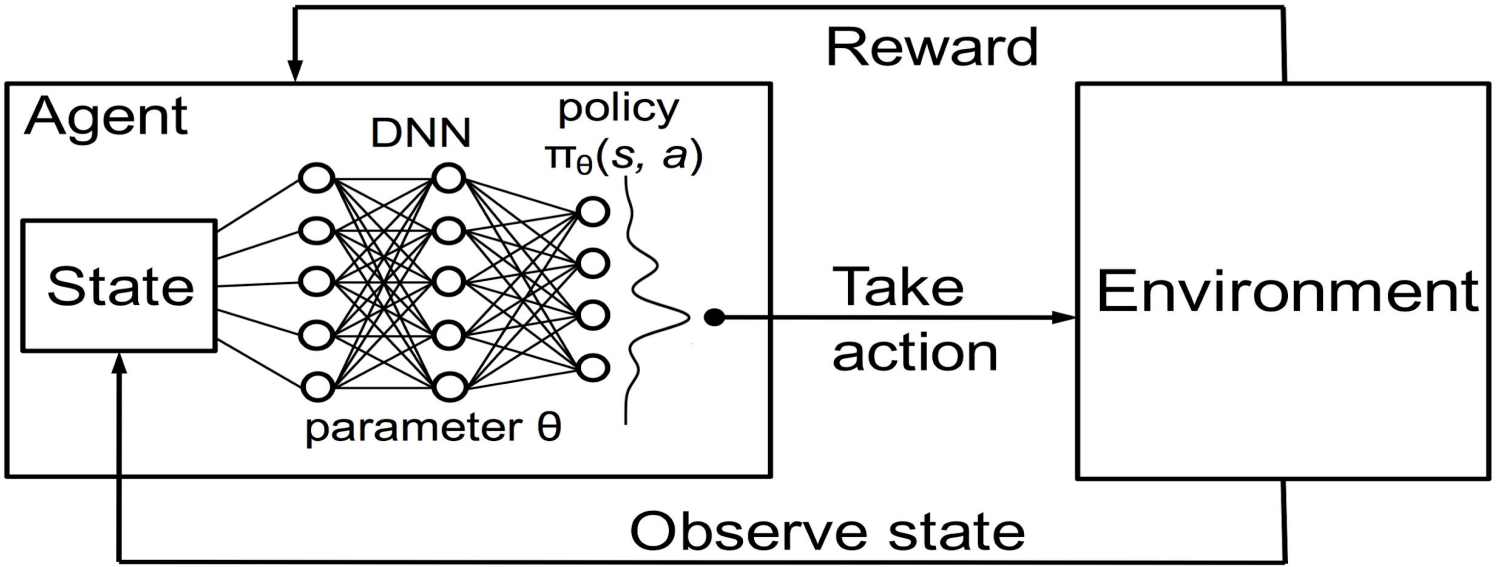
https://www.vision-systems.com/home/article/16736100/deep-learning-brings-a-new-dimension-to-machine-vision

➔ NLP (Natural Language Processing)



https://s3.amazonaws.com/aylien-main/misc/blog/images/nlp-language-dependence-small.png

➔ **Reinforcement Learning**



https://aaaminds.com/20180410/lessons-learned-reproducing-a-deep-reinforcement-learning-paper/

References

- Artificial Intelligence A Modern Approach-4th Edition, Stuart Russell and Peter Norvig
- https://towardsdatascience.com/a-beginners-guide-to-neural-networks-b6be0d442fa4
- https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
- https://brilliant.org/wiki/backpropagation/#:~:text=Backpropagation%2C%20short%20for%20%22backward%20propagation,to%20the%20neural%20network's%20weights.
- https://builtin.com/data-science/recurrent-neural-networks-and-lstm
- https://towardsdatascience.com/a-beginners-guide-to-neural-networks-b6be0d442fa4
- https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/
- https://www.upgrad.com/blog/types-of-activation-function-in-neural-networks/

Thank you