

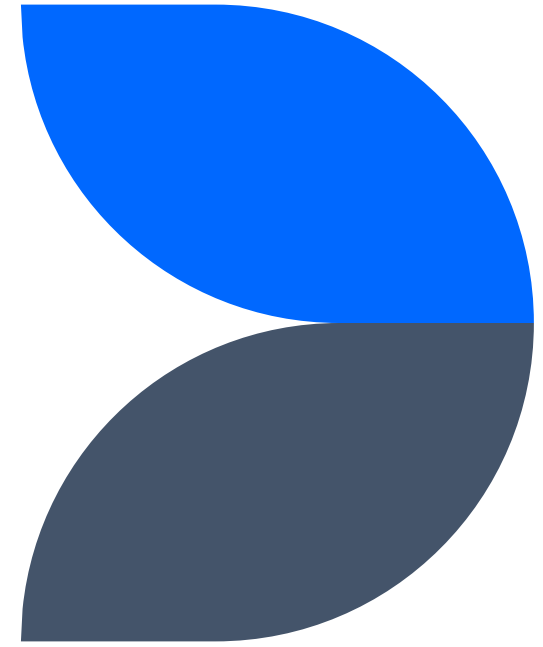


Deep Neural Networks (DNNs): Part 1

Hamed Araab

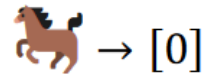


Common Tasks in Deep Learning




Binary Classification


- The primary objective is to assign each item to either Class 1 or Class 0.
- Consider the task of classifying images as either "cat" (Class 1) or "not cat" (Class 0):





Multi-Class Classification

- The primary objective is to categorize items across multiple classes rather than just two.
- Consider a scenario where we aim to classify images into distinct categories such as "cat," "dog," and "neither"

 $\rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$


 $\rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$


 $\rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$


 $\rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$


Multi-Label Classification


- In this scenario, the goal is to identify and assign multiple labels to each input.
- For instance, we may seek to find out whether each image contains a "cat," "dog," or "elephant":

 $\rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

 $\rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

 $\rightarrow \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

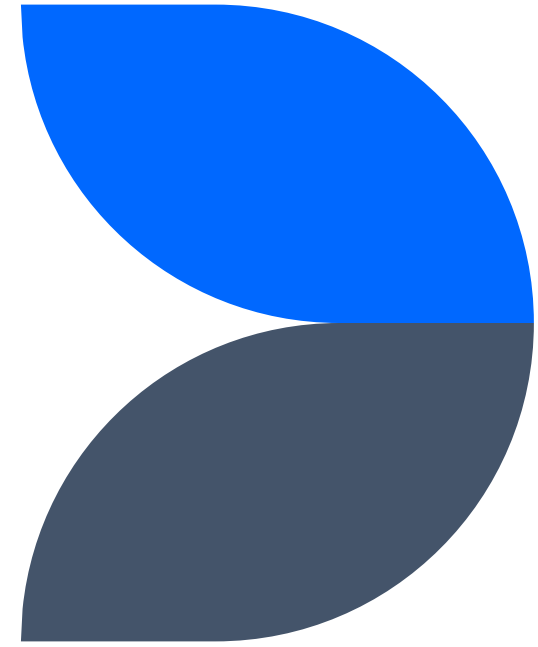
 $\rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

 $\rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Regression

- In this task, the goal is to predict a continuous numerical value rather than a class label.
- It involves learning a mapping function from input features to a continuous output variable.
- For example, we can predict house prices based on features such as square footage, number of bedrooms, and location.

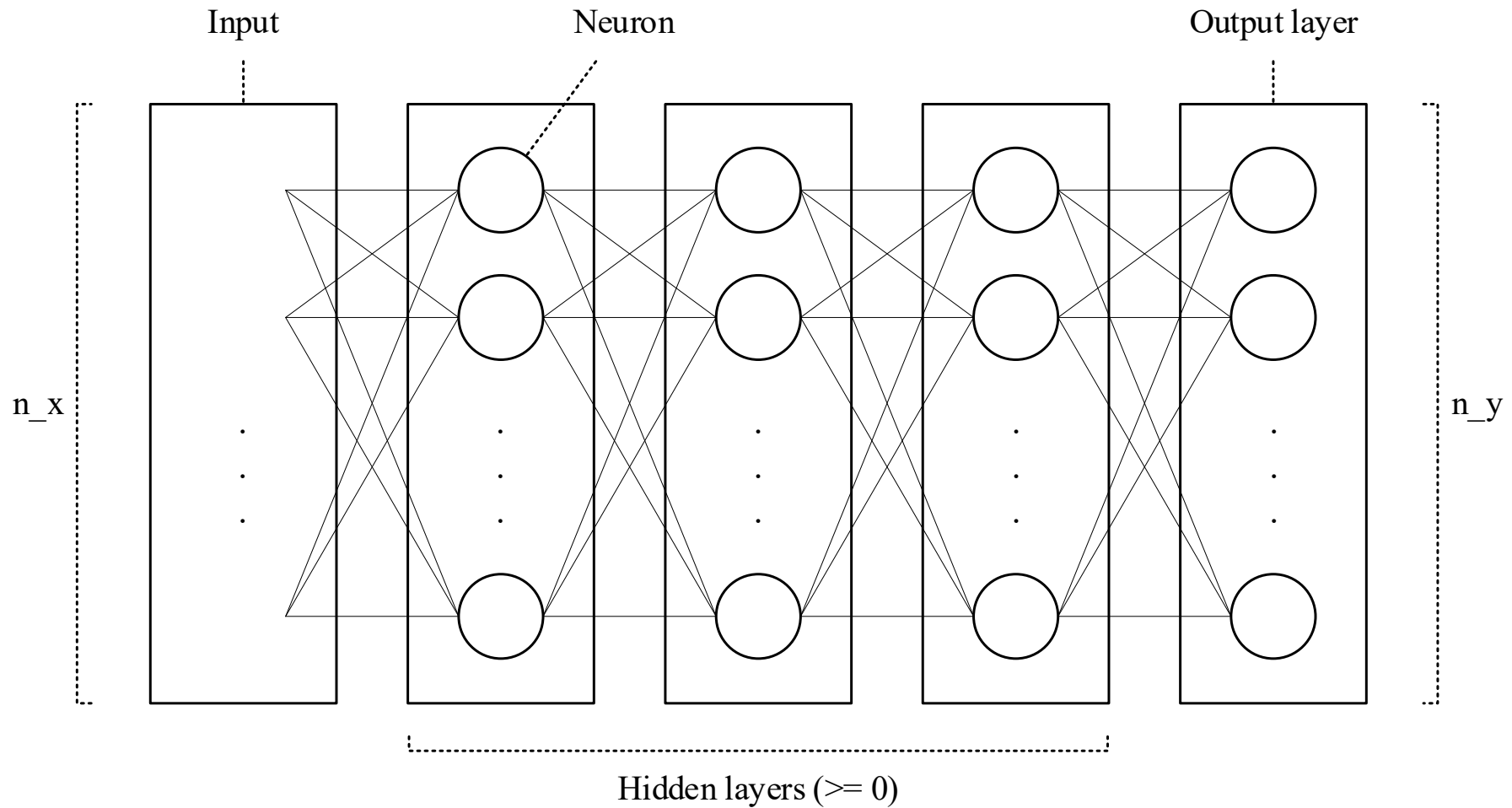
Deep Neural Networks (DNNs)



What is a DNN?

- An artificial neural network (ANN) with multiple layers.
- Can model complex relationships due to its multi-layer structure.
- In each layer, there are neurons connected to each other.
- These connections have weights and each neuron has a bias.

Forward Propagation: Predict the output for the given input.



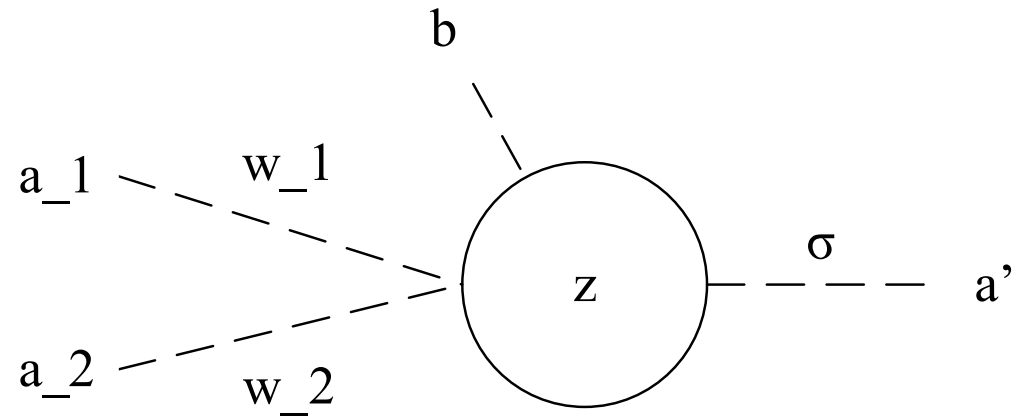
Backward Propagation: Updating the network's parameters by comparing the predicted output with the actual output

Notes

- The input is not usually considered to be a “layer”.
- Commonly, total layers = hidden layers + output layer

How a Neuron Works in a DNN

- a_i : the activation value of the i^{th} neuron in the previous layer
- w_j : weight of the connection to the j^{th} neuron in the previous layer
- b : bias
- z : pre-activation value (logit)
- σ : activation function
- a' : activation value



How a Neuron Works in a DNN

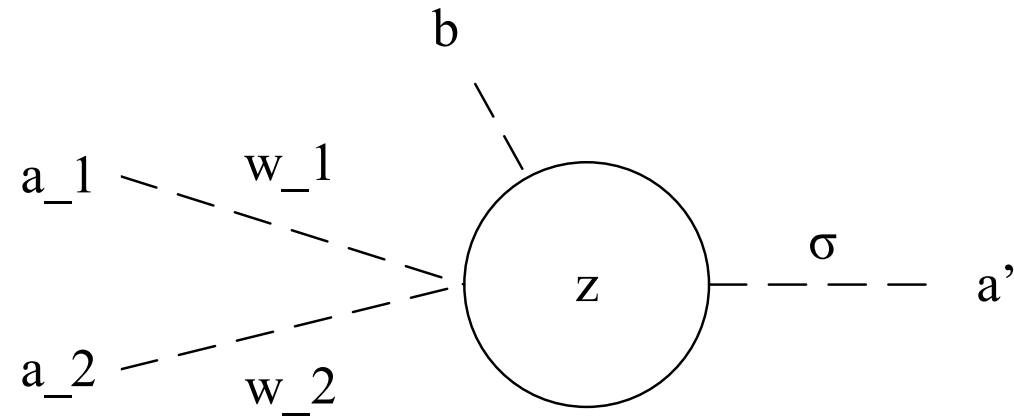
- $z = w_1 a_1 + w_2 a_2 + b$

This can be vectorized as:

- $\vec{w} = [w_1 \quad w_2] \quad \vec{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$

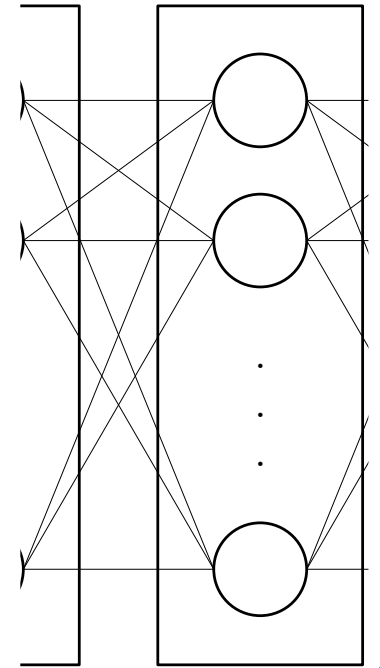
- $z = \vec{w} \cdot \vec{a} + b$

- $a' = \sigma(z)$



How a Layer Works in a DNN

- $\vec{a}^{[l-1]}$: The activation vector of the previous layer
- $\vec{w}_i^{[l]}$: The weight vector of the i^{th} neuron in the current layer
- b_i : The bias of the i^{th} neuron in the current layer
- z_i : The pre-activation value of the i^{th} neuron in the current layer
- $\sigma^{[l]}$: The activation function of the l^{th} layer
- $\vec{a}^{[l]}$: The activation vector of the current layer



How a Layer Works in a DNN

- $z_1^{[l]} = \vec{w}_1^{[l]} \cdot \vec{a}^{[l-1]} + b_1^{[l]}$
- $z_2^{[l]} = \vec{w}_2^{[l]} \cdot \vec{a}^{[l-1]} + b_2^{[l]}$
- ...

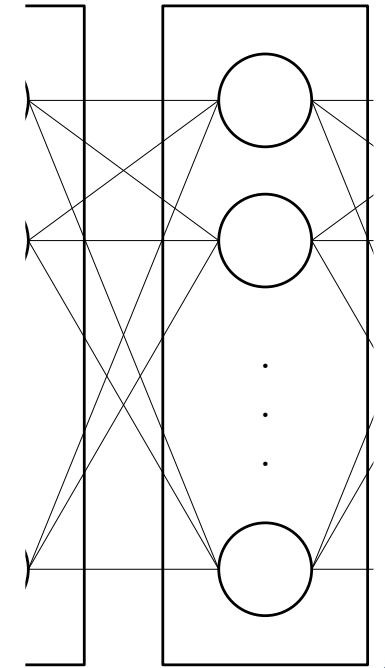
These equations can be vectorized as:

- $W^{[l]} = \begin{bmatrix} -\vec{w}_1^{[l]} & - \\ -\vec{w}_2^{[l]} & - \\ - & \dots & - \end{bmatrix}$

$$\vec{b}^{[l]} = \begin{bmatrix} b_1^{[l]} \\ b_2^{[l]} \\ \dots \end{bmatrix}$$

- $\vec{z}^{[l]} = W^{[l]} \times \vec{a}^{[l-1]} + \vec{b}^{[l]}$

$$\vec{a}^{[l]} = \sigma^{[l]}(\vec{z}^{[l]})$$



Input and Output

- \vec{x}_j : The input vector of the j^{th} item in the dataset
- \vec{y}_j : The actual output vector of the j^{th} item in the dataset
- $\hat{\vec{y}}_j$: The predicted output vector for the j^{th} item in the dataset

Input and Output

- $\vec{a}_j^{[0]} = \vec{x}_j \quad \forall j = 1, 2, \dots, m$
- $\vec{z}_j^{[l]} = W^{[l]} \times \vec{a}_j^{[l-1]} + \vec{b}^{[l]} \quad \forall l = 1, 2, \dots, L \quad \forall j = 1, 2, \dots, m$
- $\vec{a}_j^{[l]} = \sigma^{[l]} \left(\vec{z}_j^{[l]} \right) \quad \forall l = 1, 2, \dots, L \quad \forall j = 1, 2, \dots, m$
- $\vec{y}_j = \vec{a}_j^{[L]} \quad \forall j = 1, 2, \dots, m$

Final Form of Forward Propagation

$$\bullet \quad X = \begin{bmatrix} | & | & | \\ \vec{x}_1^{[l]} & \vec{x}_2^{[l]} & \vdots \\ | & | & | \end{bmatrix} \quad Y = \begin{bmatrix} | & | & | \\ \vec{y}_1^{[l]} & \vec{y}_2^{[l]} & \vdots \\ | & | & | \end{bmatrix} \quad \hat{Y} = \begin{bmatrix} | & | & | \\ \vec{\hat{y}}_1 & \vec{\hat{y}}_2 & \vdots \\ | & | & | \end{bmatrix}$$

$$\bullet \quad Z^{[l]} = \begin{bmatrix} | & | & | \\ \vec{z}_1^{[l]} & \vec{z}_2^{[l]} & \vdots \\ | & | & | \end{bmatrix} \quad A^{[l]} = \begin{bmatrix} | & | & | \\ \vec{a}_1^{[l]} & \vec{a}_2^{[l]} & \vdots \\ | & | & | \end{bmatrix}$$

Final Form of Forward Propagation

- $A^{[0]} = X$
- $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]} \quad \forall l = 1, 2, \dots, L$
- $A^{[l]} = \sigma^{[l]}(Z^{[l]}) \quad \forall l = 1, 2, \dots, L$
- $\hat{Y} = A^{[L]}$