Fundamental Operations in Artificial Neurons: Dot Product, Matrix Multiplication, Linear Layers, and Activation Functions

July 24, 2025

Artificial neurons form the backbone of modern deep learning systems. This chapter explores the foundational mathematical operations that underpin artificial neurons: the dot product, matrix multiplication, linear layers, and activation functions. We extend the analysis with illustrative diagrams and a section on the computational flow of neural layers. These operations, while simple individually, enable the construction of powerful neural architectures capable of modeling highly complex functions.

1 Introduction

Artificial Neural Networks (ANNs) are inspired by the structure and functioning of the biological brain. Their core processing unit is the **artificial neuron**, which computes a weighted sum of its inputs, adds a bias term, and applies a non-linear transformation to produce an output.

This chapter presents the key operations that define the behavior of artificial neurons, leading to the composition of multi-layer neural networks.

2 Dot Product

The dot product of two vectors $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$ is defined as:

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

In neural computation, this operation computes the weighted sum of input features. It serves as the linear core of the neuron's function.

3 Matrix Multiplication

In practice, multiple inputs and multiple neurons are processed simultaneously using matrix multiplication.

Let:

- $\mathbf{X} \in \mathbb{R}^{m \times n}$ be a matrix of m input samples (rows) each with n features,
- $\mathbf{W} \in \mathbb{R}^{n \times p}$ be the weight matrix for p neurons.

Then the output of the layer is:

$$\mathbf{Z} = \mathbf{X} \cdot \mathbf{W}$$

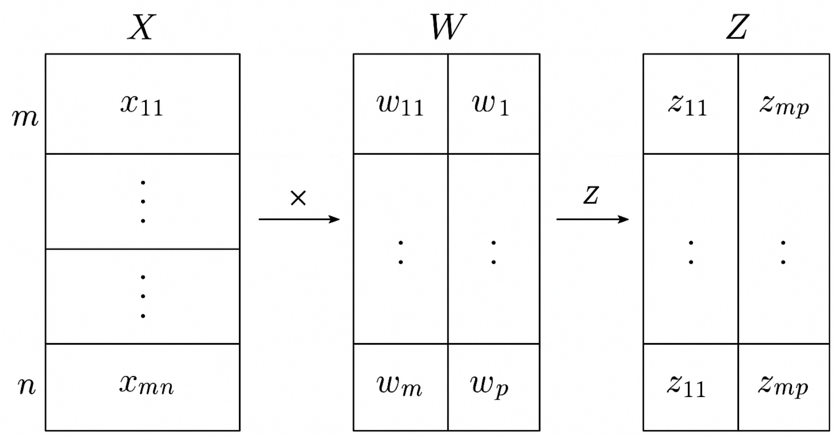

Illustration of matrix multiplication in a neural layer. Each output neuron receives a dot product between weights and input features.

Figure 1: Illustration of matrix multiplication in a neural layer. Each output neuron receives a dot product between weights and input features.

4 Linear Layer (Affine Transformation)

A **linear layer** computes:

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

where:

- $\mathbf{W}$: weights,
- $\mathbf{x}$: input vector,
- $\mathbf{b}$: bias vector.

This operation projects the input into a new feature space. The bias allows each output neuron to have an offset.

5 Activation Functions

After linear transformation, non-linear activation functions are applied to introduce expressiveness and prevent the network from collapsing into a purely linear system.

Common Activation Functions

- **Sigmoid:**
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
- **Tanh:**
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$
- **ReLU:**
$$\text{ReLU}(z) = \max(0, z)$$

Each function affects the optimization landscape and gradient behavior differently.

6 Computational Flow of an Artificial Neuron

Putting it all together, the output of a neuron is:

$$a = f(\mathbf{w} \cdot \mathbf{x} + b)$$

Or, in matrix form for a batch of inputs:

$$\mathbf{A} = f(\mathbf{X} \cdot \mathbf{W} + \mathbf{b})$$

7 Example Workflow

Assume a single-layer network with 3 inputs and 2 neurons:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 & b_2 \end{bmatrix}$$

Then the output is:

$$\mathbf{z} = \mathbf{x} \cdot \mathbf{W} + \mathbf{b} \quad \Rightarrow \quad \mathbf{a} = f(\mathbf{z})$$

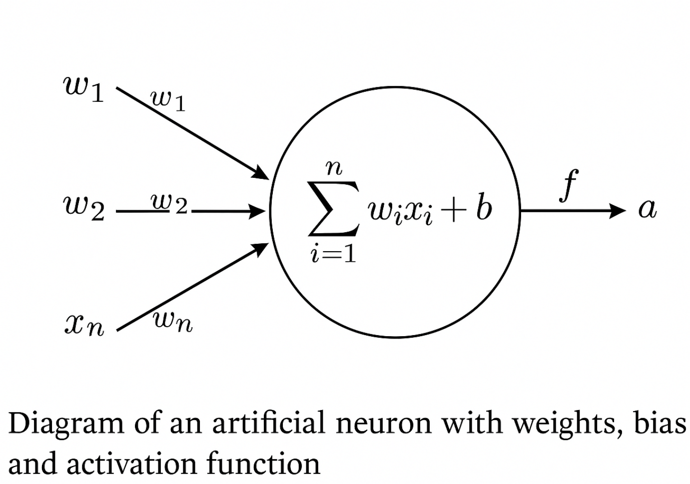

Figure 2: A basic artificial neuron takes a vector input, computes a dot product with weights, adds a bias, and passes the result through an activation function.

8 Conclusion

This paper reviewed the basic mathematical operations behind artificial neurons:

- The dot product for weighted input summation,
- Matrix multiplication for parallel computation,
- Linear layers for affine transformations,
- Activation functions for non-linear modeling.

Together, these components form the backbone of deep learning models, enabling them to represent and learn complex functions.

References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.
2. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.
3. Nielsen, M. (2015). *Neural Networks and Deep Learning.* Determination Press.

9 Practice by Hand

This section includes beginner-friendly exercises that reinforce the theoretical concepts presented earlier. You are encouraged to solve them manually to gain intuitive understanding.

9.1 Dot Product Examples

Example 1: Let $\mathbf{a} = \begin{bmatrix}3\end{bmatrix}, \mathbf{b} = \begin{bmatrix}5\end{bmatrix}$

$$\mathbf{a} \cdot \mathbf{b} = 3 \times 5 = 15$$

Example 2: Let $\mathbf{a} = \begin{bmatrix}1 & 3\end{bmatrix}, \mathbf{b} = \begin{bmatrix}4 \\ 1\end{bmatrix}$

$$\mathbf{a} \cdot \mathbf{b} = (1 \times 4) + (3 \times 1) = 4 + 3 = 7$$

9.2 Matrix Multiplication Examples

Example 1: Let

$$\mathbf{A} = \begin{bmatrix}2 & 4 & -6 \\ 3 & 4 & 4\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}4 \\ -1\end{bmatrix}$$

Since dimensions are incompatible, this example may be meant for vector-matrix dot application or has a typo. Let's move to the next valid example.

Example 2: Let

$$\mathbf{A} = \begin{bmatrix}2 & -3 \\ 1 & 5\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}-1 & 2 \\ 1 & 1\end{bmatrix}$$

Then,

$$\mathbf{A} \cdot \mathbf{B} = \begin{bmatrix}(2)(-1) + (-3)(1) & (2)(2) + (-3)(1) \\ (1)(-1) + (5)(1) & (1)(2) + (5)(1)\end{bmatrix} = \begin{bmatrix}-2 - 3 & 4 - 3 \\ -1 + 5 & 2 + 5\end{bmatrix} = \begin{bmatrix}-5 & 1 \\ 4 & 7\end{bmatrix}$$

9.3 Linear Layer Example

Let

$$\mathbf{X} = \begin{bmatrix}3 \\ 1\end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix}2 & 3 & 4 \\ 1 & -1 & 3\end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix}2 & 2 & 2\end{bmatrix}$$

Then the output $\mathbf{Y}$ is:

$$\mathbf{Y} = \mathbf{W}^T \cdot \mathbf{X} + \mathbf{b} \Rightarrow \begin{aligned} Y_1 &= (2)(3) + (1)(1) + 2 = 6 + 1 + 2 = 9 \\ Y_2 &= (3)(3) + (-1)(1) + 2 = 9 - 1 + 2 = 10 \\ Y_3 &= (4)(3) + (3)(1) + 2 = 12 + 3 + 2 = 17 \end{aligned}$$

9.4 Activation Function: ReLU Practice

Apply ReLU to the following inputs:

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

Input (x)	Output (ReLU(x))
-5	0
-4	0
-3	0
-2	0
-1	0
0	0
1	1
2	2
3	3
4	4
5	5

9.5 Artificial Neuron Example

Given:

$$\mathbf{X} = \begin{bmatrix} 5 & 2 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 2 & 1 \\ 1 & -2 \\ -1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 & -1 & -1 \end{bmatrix}$$

Compute each neuron's linear and ReLU-activated output:

$$\begin{aligned} Y_1 &= \text{ReLU}(2 \cdot 5 + 1 \cdot 2 + 1) = \text{ReLU}(10 + 2 + 1) = \text{ReLU}(13) = 13 \\ Y_2 &= \text{ReLU}(1 \cdot 5 + (-2) \cdot 2 - 1) = \text{ReLU}(5 - 4 - 1) = \text{ReLU}(0) = 0 \\ Y_3 &= \text{ReLU}((-1) \cdot 5 + 0 \cdot 2 - 1) = \text{ReLU}(-5 - 1) = \text{ReLU}(-6) = 0 \end{aligned}$$

$$\Rightarrow \mathbf{Y} = \begin{bmatrix} 13 & 0 & 0 \end{bmatrix}$$

These examples help in grasping the fundamentals of neural operations and provide a basis for building more complex models.