




Neural Network: ReLu()

🏠 Domain	AI
👤 Assign	 Lara Sousa
⚙️ Status	To Publish

Introduction

What is ReLU? Is ReLu a neural network itself? ReLU (Rectified Linear Unit) is not a neural network itself, but rather an activation function used within neural networks.

In this paper, we will explore briefly the key concepts beneath one of the most used activate functions in this "deep world" of neural networks (*deep learning** - AI's branch).

**Deep learning:*

Deep Learning is making machines think the way humans do. It's not about processing data anymore; it's about processing information the way a biological brain does. - Geoffrey Hinton, "Father of Deep Learning"

Deep Learning is a subset of machine learning that uses artificial neural networks with multiple layers (deep neural networks) to progressively learn from data.

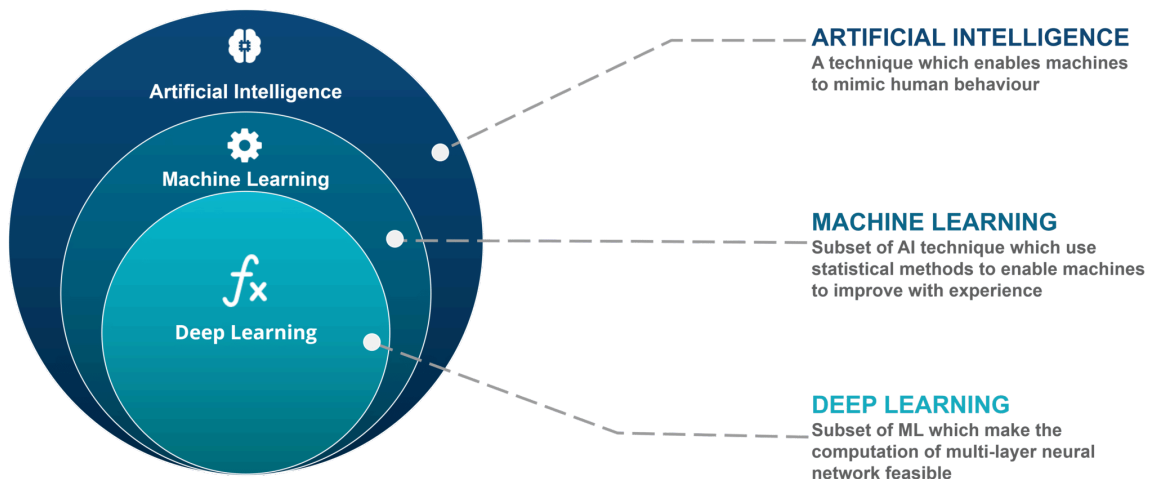


image from: <https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/>

Formal Research Explanation

Mathematical Definition:

$$f(x) = \max(0, x)$$

Key Characteristics

1. Mathematical Implementation

```
self.encoder = nn.Sequential(  
    nn.Linear(input_size, hidden_size),  
    nn.ReLU(), # ReLU activation function  
    nn.Dropout(0.2)  
)
```

2. Fundamental Properties

- **Non-linearity:** Introduces non-linearity into the model;
- **Computational simplicity:** Simple threshold operation;
- **Constant gradient:** For positive values, gradient = 1;
- **Sparsity:** Produces sparse activations (many zeros).

Advantages and Disadvantages

Advantages 😊

1. Computational Efficiency

- Simple calculation: $\max(0, x)$
- Efficient gradients during back-propagation

2. Solves Vanishing Gradient Problem

- No saturation for positive values
- Constant gradient for $x > 0$

3. Fast Convergence

- Speeds up training compared to sigmoid/tanh
- Less computationally intensive

Disadvantages 😞

1. Dying ReLU Problem

- Neurons can "die" during training
- Occurs when gradients become permanently zero

2. Not Differentiable at $x=0$

- Can cause numerical instabilities
- Solved in practice using subgradients

Types of Neural Networks Using ReLU

ReLU is widely used in various types of neural networks:

1. Feed-Forward Neural Networks

```
class neural_network(nn.Module):
    def __init__(self, input_size: int, hidden_size: int = 256):
        self.encoder = nn.Sequential(
            nn.Linear(input_size, hidden_size),
            nn.ReLU(), # ReLU after linear layer
            nn.Dropout(0.2),
            nn.Linear(hidden_size, hidden_size // 2),
            nn.ReLU() # ReLU after second layer
        )
```

2. Convolutional Neural Networks (CNNs)

```
class ConvNet(nn.Module):
    def __init__(self):
```

```
super().__init__()
self.conv1 = nn.Sequential(
    nn.Conv2d(3, 64, 3),
    nn.ReLU(),    # ReLU after convolution
    nn.MaxPool2d(2)
```

3. Recurrent Neural Networks (RNNs)

```
class RNNModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.rnn = nn.RNN(input_size, hidden_size)
        self.fc = nn.Linear(hidden_size, output_size)
        self.relu = nn.ReLU() # ReLU after recurrent layer
```

Specific Use Cases

1. Image Processing

- CNNs with ReLU for visual pattern recognition
- Maintains important spatial features

2. Text Processing

- RNNs and Transformers using ReLU in their feed-forward layers
- Helps model complex dependencies

3. Regression

```
self.dimension_line_head = nn.Sequential(
    nn.Linear(hidden_size // 2, hidden_size // 4),
    nn.ReLU(), # ReLU for coordinate prediction
    nn.Linear(hidden_size // 4, 4)
)
```

Why ReLU is Popular?

Computational Efficiency

- Simple operation: $\max(0, x)$
- Efficient gradients

Problem Solving

- Avoids vanishing gradient problem
- Enables training of deeper networks

Empirical Results

- Better performance in many tasks
 - Faster convergence
-

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

Books

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). "Deep Learning." MIT Press.
- Chollet, F. (2021). "Deep Learning with Python." Manning Publications.

PyTorch Implementation

- PyTorch Documentation: <https://pytorch.org/do>