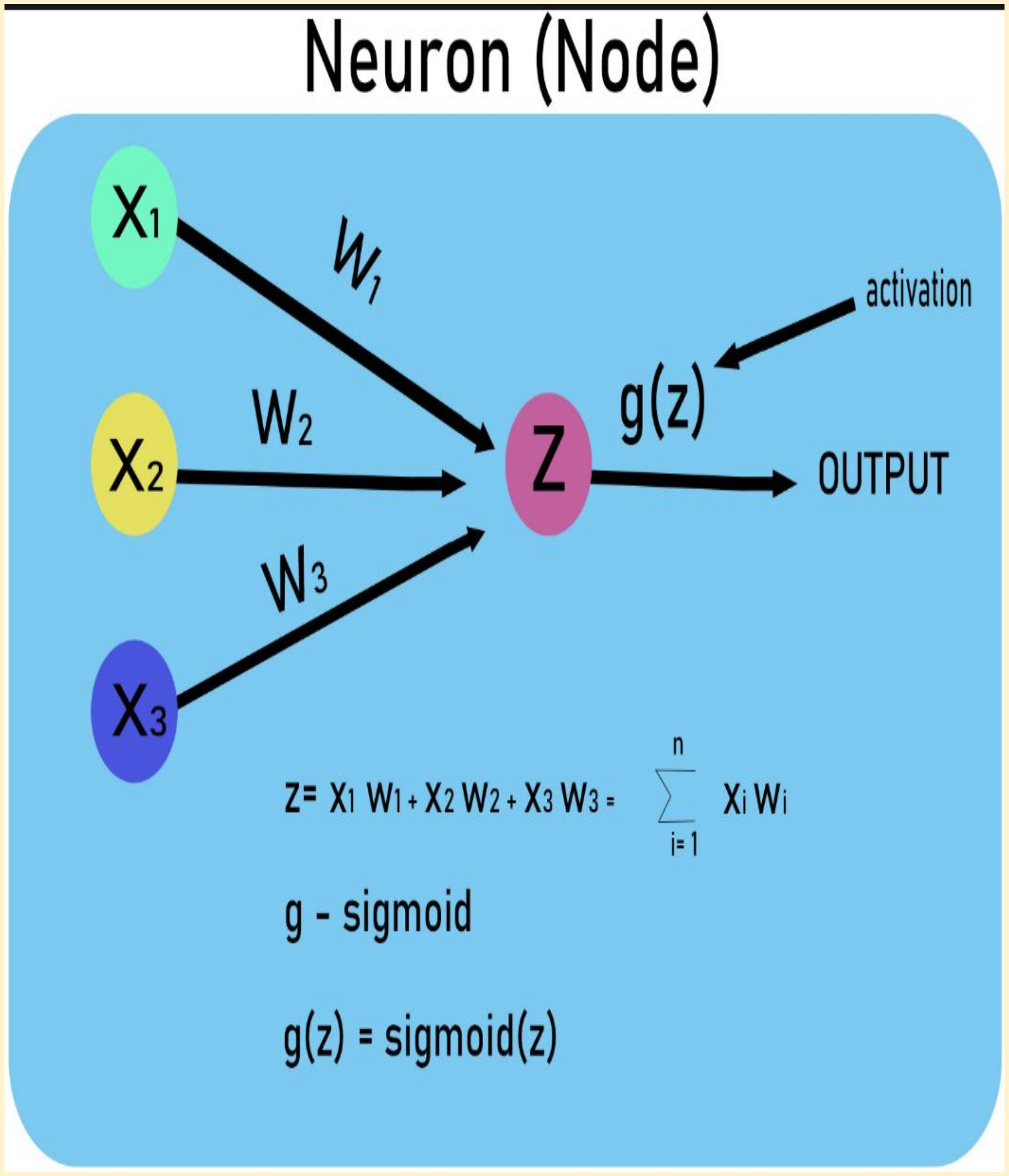
Norina Akhtar

Design XOR Gate Using Neural Network Primer

Table oF Content

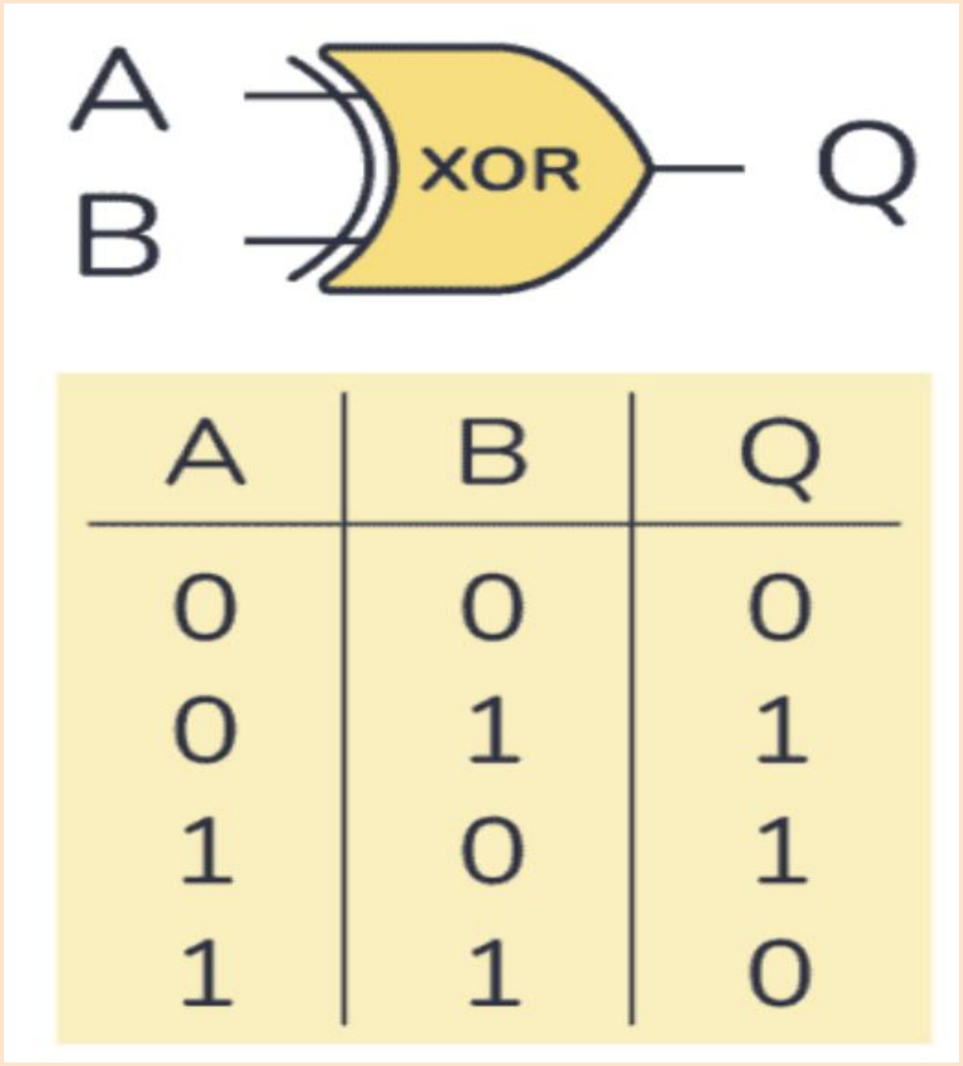
Introduction

A neural network primer is an introduction to the basics of artificial neural networks (ANNs), which are a type of machine learning model inspired by the structure and function of biological neural networks in the brain. ANNs are composed of interconnected nodes (neurons) that process and transmit information through weighted connections. Neural networks can be trained on a dataset to learn patterns and relationships in the data, and then make predictions or classifications on new data.



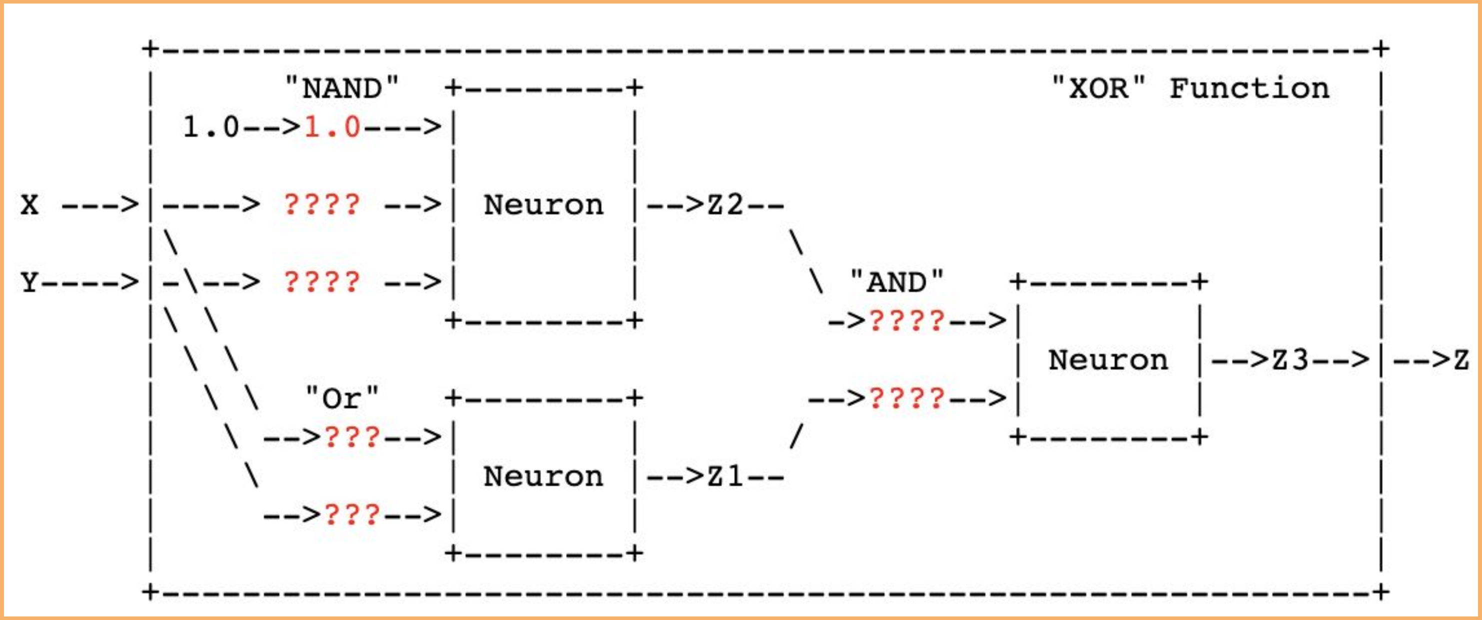
XOR -Gate

The XOR gate, or exclusive or gate, is a logical operator that takes two binary inputs and outputs 1 (true) if and only if the inputs are different. Otherwise, it outputs 0 (false). The XOR function is important in computer science and electronics, as it can be used to represent logical operations and is a fundamental building block of digital circuits.

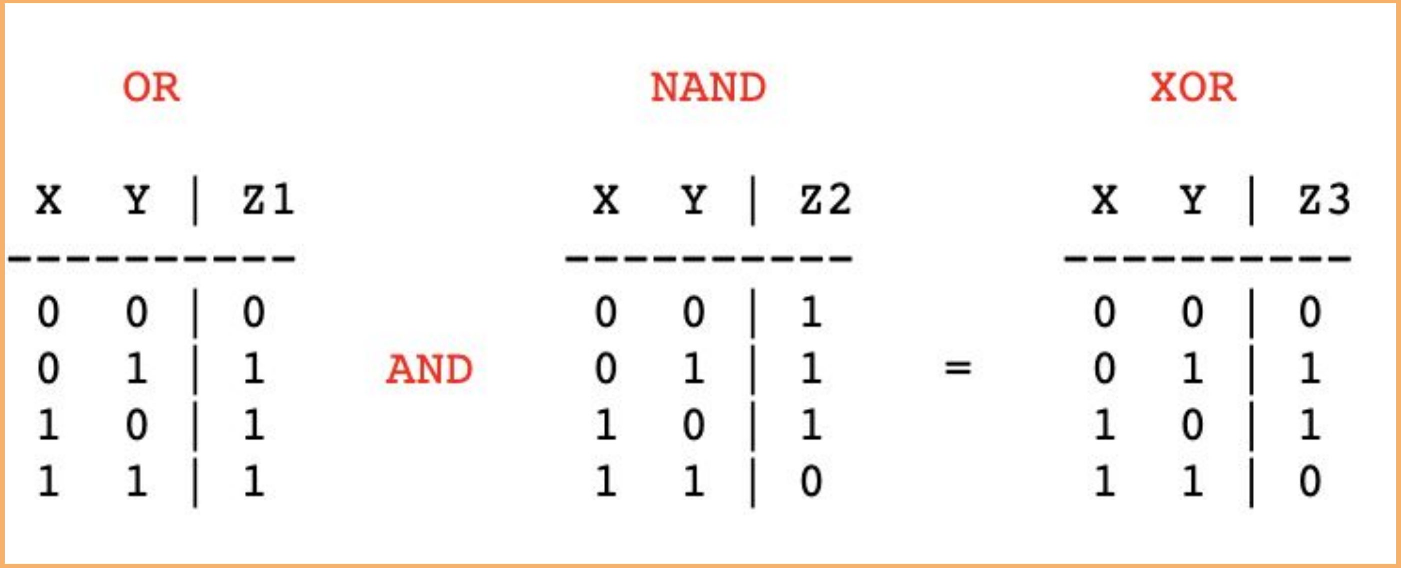


A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Design



```
      +-------------------------------------------------------------+
      |      "NAND"  +--------+                     "XOR" Function  |
      | 1.0-->1.0--->|        |                                     |
      |              |        |                                     |
X --->|----> ???? -->| Neuron |-->Z2--                              |
      |\             |        |        \                            |
Y---->|-\--> ???? -->|        |         \ "AND"   +--------+        |
      |\ \           +--------+          ->????-->|        |        |
      | \ \                                       | Neuron |-->Z3-->|-->Z
      |  \ \  "Or"   +--------+         -->????-->|        |        |
      |   \ -->???-->|        |        /          +--------+        |
      |    \         | Neuron |-->Z1--                              |
      |     -->???-->|        |                                     |
      |              +--------+                                     |
      +-------------------------------------------------------------+
```



OR

X	Y	Z1
0	0	0
0	1	1
1	0	1
1	1	1

AND

NAND

X	Y	Z2
0	0	1
0	1	1
1	0	1
1	1	0

=

XOR

X	Y	Z3
0	0	0
0	1	1
1	0	1
1	1	0

Implementation

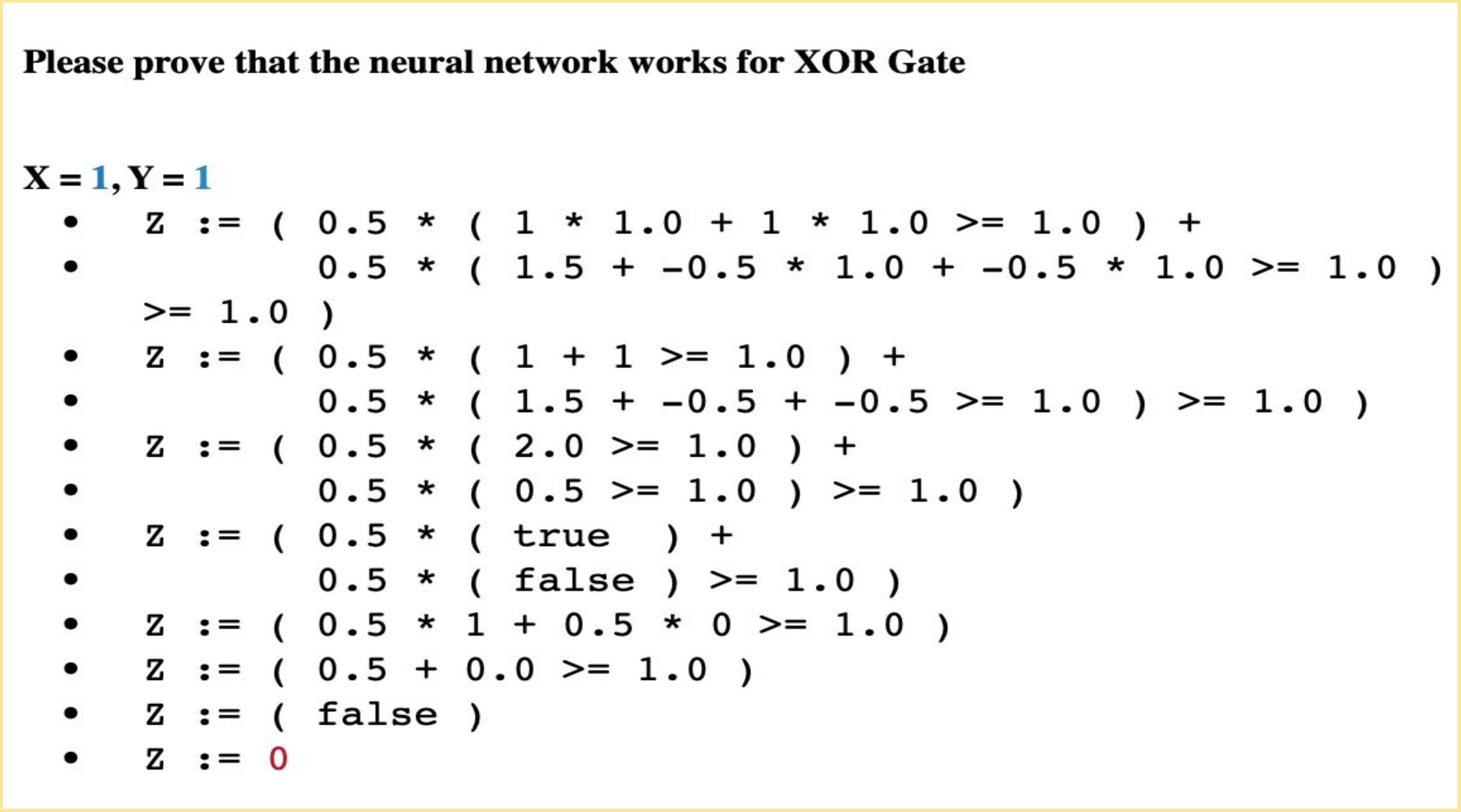
```
Z1  :=  X   "Or"    Y
Z2  :=  X   "NAND"  Y
Z  :=  Z3  :=  Z1  "AND"  Z2
Z  :=  (  X  "Or"  Y  )  "AND"  (  X  "NAND"  Y  )
Z  :=  (  1  *  X  +  1  *  Y  >=  1.0  )  "AND"
       (  1  *  1.5  +  -0.75  *  X  +  -0.75  *  Y  >=  1.0  )

Z  :=  (  0.5  *  (  1  *  X  +  1  *  Y  >=  1.0  )  +
          0.5  *  (  1.5  +  -0.5  *  X  +  -0.5  *  Y  >=  1.0  )  >=  1.0  )
```

Note: for AND

```
       AND
-----------
X   Y   |   Z
-----------
0   0   |   0
0   1   |   0
1   0   |   0
1   1   |   1
|
Z  :=  (  0.5  *  X  +  0.5  *  Y  >=  1.0  )
```

Test



Please prove that the neural network works for XOR Gate

X = 1, Y = 1

- `Z := ( 0.5 * ( 1 * 1.0 + 1 * 1.0 >= 1.0 ) +`
- `         0.5 * ( 1.5 + -0.5 * 1.0 + -0.5 * 1.0 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( 1 + 1 >= 1.0 ) +`
- `         0.5 * ( 1.5 + -0.5 + -0.5 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( 2.0 >= 1.0 ) +`
- `         0.5 * ( 0.5 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( true  ) +`
- `         0.5 * ( false ) >= 1.0 )`
- `Z := ( 0.5 * 1 + 0.5 * 0 >= 1.0 )`
- `Z := ( 0.5 + 0.0 >= 1.0 )`
- `Z := ( false )`
- `Z := 0`

Test

X=1, Y=0

- `Z := ( 0.5 * ( 1 * 1.0 + 1 * 0 >= 1.0 ) + 0.5 * ( 1.5 + -0.5 * 1.0 + -0.5 * 0 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( 1 + 0 >= 1.0 ) + 0.5 * ( 1+ 0 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( 2.0 >= 1.0 ) +  0.5 * ( 1.0 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( true  ) + 0.5 * ( true) >= 1.0 )`
- `Z := ( 0.5 * 1 + 0.5 * 1 >= 1.0 )`
- `Z := ( 0.5 + 0.5 >= 1.0 )`
- `Z := ( true)`
- `Z := 1`

Test

```
X= 0, Y= 1
```

- `Z := ( 0.5 * ( 1 * 0 + 1 * 1 >= 1.0 ) + 0.5 * ( 1.5 + -0.5 * 0 + -0.5 * 1 >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( 1  >= 1.0 ) + 0.5 * ( 1.0  >= 1.0 ) >= 1.0 )`
- `Z := ( 0.5 * ( true  ) + 0.5 * ( true) >= 1.0 )`
- `Z := ( 0.5 * 1 + 0.5 * 1 >= 1.0 )`
- `Z := ( 0.5 + 0.5 >= 1.0 )`
- `Z := ( true)`
- `Z := 1`

Test

```
X = 0, Y = 0

•   Z := ( 0.5 * ( 1 * 0 + 1 * 0 >= 1.0 ) + 0.5 * ( 1.5 + -0.5
    * 0 + -0.5 * 0 >= 1.0 ) >= 1.0 )

•   Z := ( 0.5 * ( 0  >= 1.0 ) + 0.5 * ( 1.5  >= 1.0 ) >= 1.0 )

•   Z := ( 0.5 * ( false ) + 0.5 * ( true) >= 1.0 )

•   Z := ( 0.5 * 0 + 0.5 * 1 >= 1.0 )

•   Z := ( 0.0 + 0.5 >= 1.0 )

•   Z := (false)

•   Z := 0
```

Enhancement Ideas

There are several enhancement ideas that can be considered when training gates using neural networks:

1. **Using more advanced architectures:** Neural networks have evolved significantly in recent years, and there are many different architectures available that can improve the accuracy of gate training.

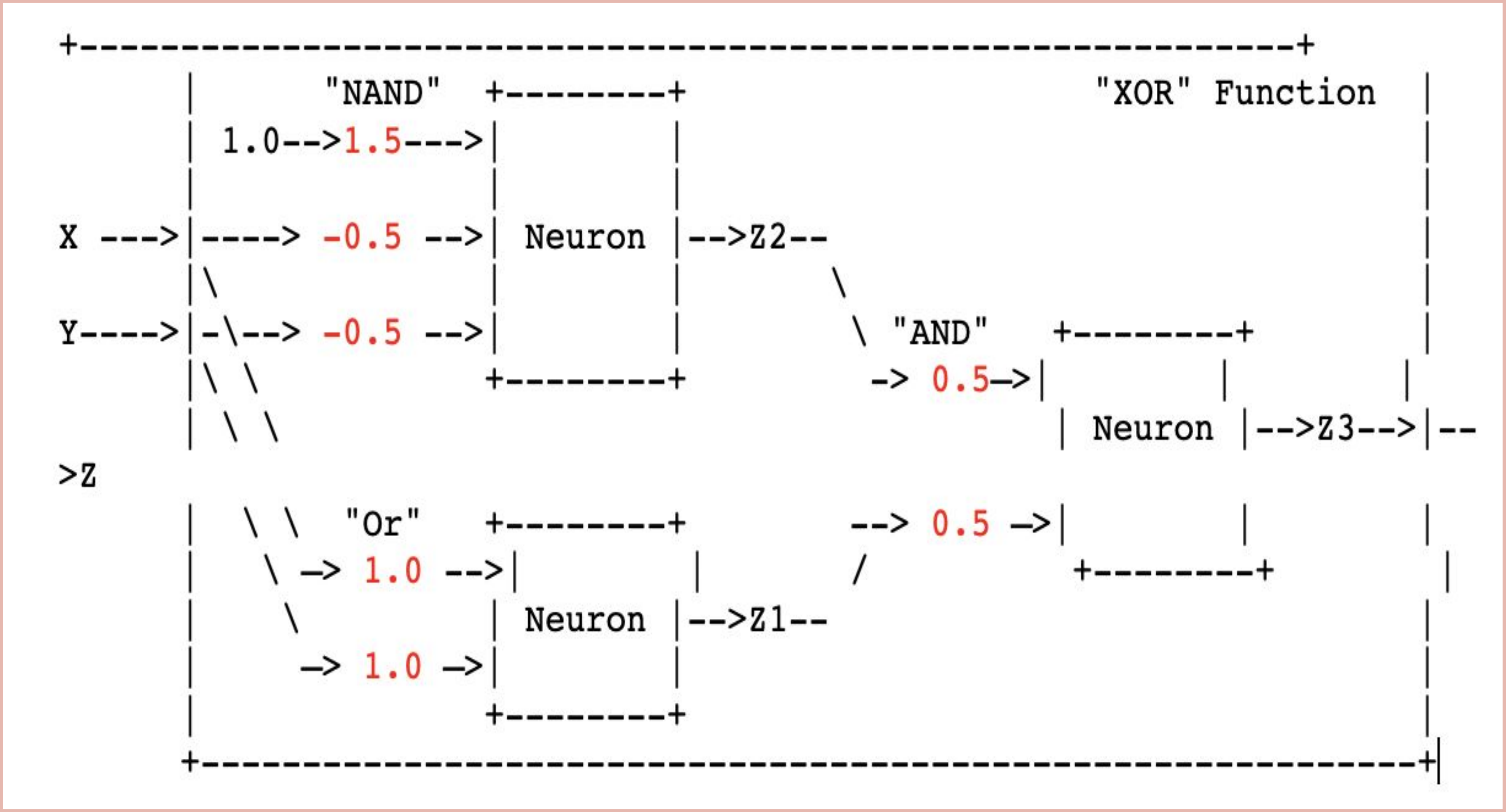
2. **Optimizing hyperparameters:** The performance of a neural network can be greatly influenced by its hyperparameters, such as learning rate, batch size, and regularization.

3. **Data augmentation**: Augmenting the training data with additional variations or transformations can help improve the generalization ability of the neural network.

4. **Transfer learning**: Pre-trained neural networks can be used as a starting point for gate training, especially when there is limited data available. Transfer learning involves using the weights and biases of a pre-trained network as a starting point for training a new network on a related task.

5. **Ensembling**: Ensemble learning involves combining multiple neural networks to improve the accuracy of gate training. This can include techniques such as bagging, boosting, or stacking.

Conclusion



```
+-------------------------------------------------------------+
      |      "NAND"   +--------+                 "XOR" Function  |
      | 1.0-->1.5--->|        |                                 |
      |              |        |                                 |
X --->|----> -0.5 -->| Neuron |-->Z2--                          |
      |\             |        |      \                          |
Y---->|-\--> -0.5 -->|        |       \ "AND"   +--------+      |
      |\ \           +--------+        -> 0.5->|        |      |
      | \ \                                     | Neuron |-->Z3-->|--
>Z
      |  \ \  "Or"   +--------+       --> 0.5 ->|        |      |
      |   \ -> 1.0 -->|        |       /         +--------+      |
      |    \          | Neuron |-->Z1--                         |
      |     -> 1.0 ->|        |                                 |
      |              +--------+                                 |
      +---------------------------------------------------------+
```

References

https://hc.labnet.sfbu.edu/~henry/sfbu/course/machine_learning/deep_learning/slide/exercise_deep_learning.html#xor

https://hc.labnet.sfbu.edu/~henry/sfbu/course/machine_learning/neural_network/slide/ann.html

Github link:

https://github.com/norinakhtar/Machine-Learning/tree/main/ChatGPT/Use%20ChatGPT%20to%20create%20customer%20support%20website