| BE(E&C) | Data Science & Visualization Lab | |
|---|---|---|
| Experiment No.:09 | **Implementation of AND/NAND gate using feed forward Neural Network.** | **Page:** /6 |

- **Aim:** Implementation of AND/NAND gate using feed forward Neural Network.
- **Software Used:** Python 3.12, Jupyter Notebook
- **Learning Objective:**

  By the end of this student should be able to:

  1. Understand the concepts of neural networks and their structure.
  2. Learn how to represent binary logic gates using neural networks.
  3. Implement a Feed Forward Neural Network to model AND and NAND gates.
  4. Train the model using binary input data and evaluate its performance.
  5. Visualize the training process and results.

- **Learning Outcomes:**

  **After performing the experiment students will be able to-**

  1. Ability to build and train a neural network using TensorFlow/Keras.
  2. Knowledge of how neural networks can be applied to logic functions.
  3. Skills in evaluating neural network performance and visualizing results.

- **Theory:**

  • **Neural Networks**:

  - A neural network is a computational model inspired by the way biological neural networks in the human brain work. It consists of interconnected nodes (neurons) organized in layers: input layer, hidden layers, and output layer.
  - Each connection has a weight, and neurons apply an activation function to the weighted sum of their inputs.

  • **Feed Forward Neural Network**:

  - A type of neural network where connections between nodes do not form cycles. Information flows in one direction—from input to output.
  - Activation functions such as Sigmoid or ReLU are commonly used to introduce non-linearity.

  • **Logic Gates**:

  - **AND Gate**: Outputs true (1) only if both inputs are true (1).

- **NAND Gate**: Outputs false (0) only if both inputs are true (1). This is the negation of the AND gate.

• **Training Neural Networks**:

- Neural networks are trained using a dataset that consists of input-output pairs. The training process involves adjusting the weights through backpropagation based on the error between predicted and actual outputs.

• **Activation Functions**:

- **Sigmoid**: f(x)=11+e−xf(x) = \frac{1}{1 + e^{-x}}f(x)=1+e−x1 maps input values to a range between 0 and 1, making it suitable for binary classification.
- **Binary Cross-Entropy Loss**: Used as a loss function for binary classification problems.

**Code:**

```
import numpy as np

import matplotlib.pyplot as plt

from keras.models import Sequential

from keras.layers import Dense


# Prepare the dataset for AND gate

X_and = np.array([[0, 0],

                  [0, 1],

                  [1, 0],

                  [1, 1]])

y_and = np.array([[0], [0], [0], [1]])  # AND gate outputs


# Prepare the dataset for NAND gate

X_nand = np.array([[0, 0],

                   [0, 1],

                   [1, 0],
```

```
            [1, 1]])
y_nand = np.array([[1], [1], [1], [0]])  # NAND gate outputs

# Function to create and train the model
def create_and_train_model(X, y):
    # Create the model
    model = Sequential()
    model.add(Dense(2, input_dim=2, activation='sigmoid'))  # Hidden layer
    model.add(Dense(1, activation='sigmoid'))  # Output layer

    # Compile the model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

    # Train the model
    model.fit(X, y, epochs=5000, verbose=0)

    return model

# Train the AND gate model
and_model = create_and_train_model(X_and, y_and)
print("AND Gate Predictions:")
print(and_model.predict(X_and))

# Train the NAND gate model
nand_model = create_and_train_model(X_nand, y_nand)
print("\nNAND Gate Predictions:")
```

```
print(nand_model.predict(X_nand))


# Visualize predictions
def plot_predictions(model, X, title):
    predictions = model.predict(X)
    plt.figure()
    plt.scatter(X[:, 0], X[:, 1], c=predictions.flatten(), cmap='coolwarm', s=100)
    plt.title(title)
    plt.xlabel('Input 1')
    plt.ylabel('Input 2')
    plt.colorbar(label='Output')
    plt.xlim(-0.5, 1.5)
    plt.ylim(-0.5, 1.5)
    plt.axhline(0.5, color='grey', lw=0.5, ls='--')
    plt.axvline(0.5, color='grey', lw=0.5, ls='--')
    plt.show()

# Plotting the predictions for both gates
plot_predictions(and_model, X_and, "AND Gate Predictions")
plot_predictions(nand_model, X_nand, "NAND Gate Predictions")
```

| BE(E&C) | Data Science & Visualization Lab | |
|---|---|---|
| Experiment No.:09 | **Implementation of AND/NAND gate using feed forward Neural Network.** | **Page:** /6 |

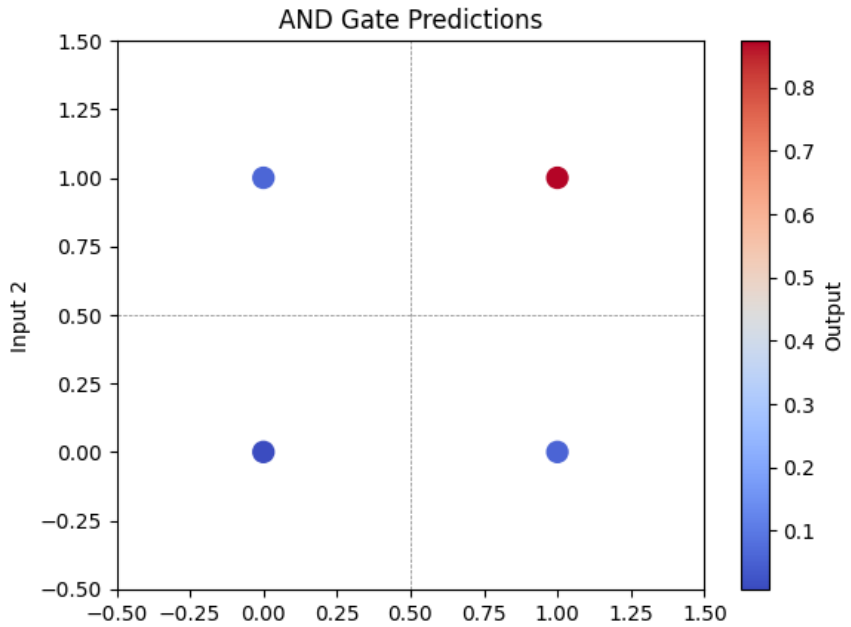
## ❖ Output

The output will include:

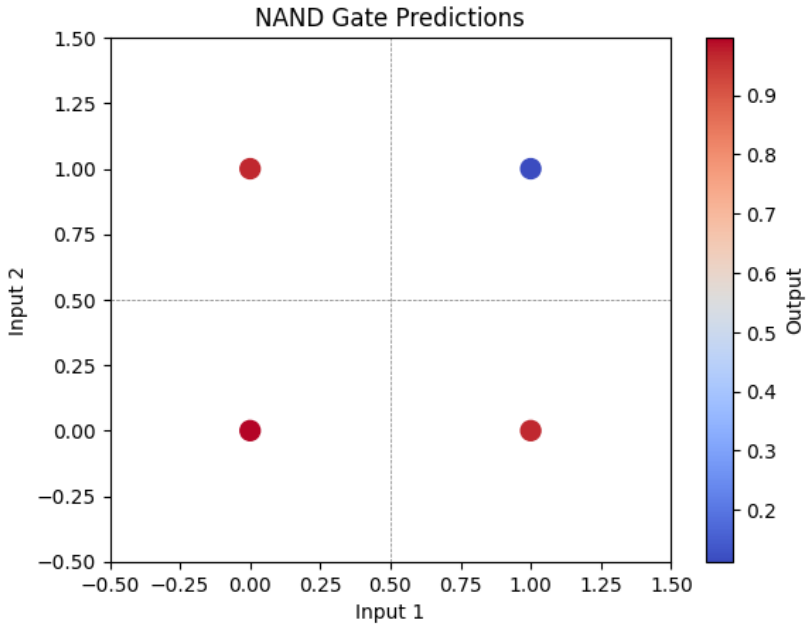
- Predictions of the AND gate for all input combinations.
- Predictions of the NAND gate for all input combinations.
- Visual representations of the outputs for both gates.

```
AND Gate Predictions:
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
[[0.0078743 ]
 [0.05962712]
 [0.05721872]
 [0.8741023 ]]

NAND Gate Predictions:
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
[[0.9968361 ]
 [0.96912295]
 [0.96753514]
 [0.11274968]]
```



| BE(E&C) | Data Science & Visualization Lab | |
|---|---|---|
| Experiment No.:09 | **Implementation of AND/NAND gate using feed forward Neural Network.** | **Page:** /6 |




## ❖ Conclusion:

_____

_____

_____

_____

## ❖ Questions:

1. Explain how a Feed Forward Neural Network is structured and its purpose in classification tasks.
2. What are the differences between the AND and NAND logic gates in terms of their output behavior?
3. How does the choice of activation function influence the performance of a neural network?
4. Discuss the role of loss functions in training neural networks and why binary cross-entropy is used for binary classification.
5. What are some common challenges when training neural networks, and how can they be addressed?