In [168…

```python
import numpy as np
```

class Perceptron

In [169…

```python
class Perceptron:
  def __init__(self, num_inputs, weights=None, bias=None):
    if weights is None:
      self.weights = np.random.rand(num_inputs)
    else:
      self.weights = np.array(weights)

    if bias is None:
      self.bias = np.random.rand()
    else:
      self.bias = bias

  def predict(self, inputs, debug=True):
    activation = self.calculate_activation(inputs)
    thresholded_activation = self.threshold_activation(activation)
    if debug:
      print(f"{activation=}, {thresholded_activation=}")
    return thresholded_activation

  def calculate_activation(self, inputs):
    return np.dot(inputs, self.weights) + self.bias

  def threshold_activation(self, activation):
    return 1 if activation >= 0 else 0

  def train(self, training_inputs, labels, learning_rate, epochs):
    training_inputs = np.array(training_inputs)

    for _ in range(epochs):
      for inputs, label in zip(training_inputs, labels):
        prediction = self.predict(inputs, debug=False)
        delta = label - prediction
        self.update_weights(inputs, delta, learning_rate)
        self.update_bias(delta, learning_rate)

  def update_weights(self, inputs, delta, learning_rate):
    self.weights += learning_rate * delta * inputs

  def update_bias(self, delta, learning_rate):
    self.bias += learning_rate * delta
```

In [170…

```python
def test_perceptron(perceptron, test_name, input_combinations):
  """
  Test a perceptron and print its predictions for all input combinations.

  Args:
  perceptron (Perceptron): Perceptron instance to test.
```

```
    test_name (str): Name of the test (e.g., "AND", "OR", "NOT", "NAND").
    input_combinations (list of tuples): List of input combinations to test.

    Returns:
    None
    """
    print(f"{test_name} Gate Truth Table:")
    print()
    for inputs in input_combinations:
        prediction = perceptron.predict(inputs)
        print(f"Inputs: {inputs}, Prediction: {prediction}")
        print()
```

input_combinations

In [171…
```
input_combinations = [(0, 0), (0, 1), (1, 0), (1, 1)]
```

AND Gate

In [172…
```
and_weights = [1, 1]
and_bias = -1.5
and_perceptron = Perceptron(num_inputs=2, weights=and_weights, bias=and_bias)
test_perceptron(and_perceptron, "AND", input_combinations)
```

```
AND Gate Truth Table:

activation=-1.5, thresholded_activation=0
Inputs: (0, 0), Prediction: 0

activation=-0.5, thresholded_activation=0
Inputs: (0, 1), Prediction: 0

activation=-0.5, thresholded_activation=0
Inputs: (1, 0), Prediction: 0

activation=0.5, thresholded_activation=1
Inputs: (1, 1), Prediction: 1
```

OR Gate

In [173…
```
or_weights = [1, 1]
or_bias = -0.5
or_perceptron = Perceptron(num_inputs=2, weights=or_weights, bias=or_bias)
test_perceptron(or_perceptron, "OR", input_combinations)
```

```
OR Gate Truth Table:

activation=-0.5, thresholded_activation=0
Inputs: (0, 0), Prediction: 0

activation=0.5, thresholded_activation=1
Inputs: (0, 1), Prediction: 1

activation=0.5, thresholded_activation=1
Inputs: (1, 0), Prediction: 1

activation=1.5, thresholded_activation=1
Inputs: (1, 1), Prediction: 1
```

NOT Gate

In [174…
```
not_weights = [-1]
not_bias = 0.5
not_perceptron = Perceptron(num_inputs=1, weights=not_weights, bias=not_bias)
test_perceptron(not_perceptron, "NOT", [(0,), (1,)])
```

```
NOT Gate Truth Table:

activation=0.5, thresholded_activation=1
Inputs: (0,), Prediction: 1

activation=-0.5, thresholded_activation=0
Inputs: (1,), Prediction: 0
```

NAND Gate

In [175…
```
nand_weights = [-1, -1]
nand_bias = 1.5
nand_perceptron = Perceptron(
    num_inputs=2, weights=nand_weights, bias=nand_bias)
test_perceptron(nand_perceptron, "NAND", input_combinations)
```

```
NAND Gate Truth Table:

activation=1.5, thresholded_activation=1
Inputs: (0, 0), Prediction: 1

activation=0.5, thresholded_activation=1
Inputs: (0, 1), Prediction: 1

activation=0.5, thresholded_activation=1
Inputs: (1, 0), Prediction: 1

activation=-0.5, thresholded_activation=0
Inputs: (1, 1), Prediction: 0
```

train_test_perceptron

In [176…

```python
def train_test_perceptron(perceptron, test_name, input_combinations, labels, learni
  """
  Train a perceptron for a given logic function, print its final weights and bias,
  and test its predictions for all input combinations.

  Args:
  perceptron (Perceptron): Perceptron instance to train and test.
  test_name (str): Name of the test (e.g., "AND", "OR", "NOT", "NAND").
  input_combinations (list of tuples): List of input combinations for training and
  labels (list): List of labels corresponding to the input combinations.
  learning_rate (float): Learning rate for training.
  epochs (int): Number of training epochs.

  Returns:
  None
  """
  print(f"Training and Testing Perceptron for {test_name} logic:")
  print()

  # Train the perceptron
  perceptron.train(input_combinations, labels, learning_rate, epochs)

  # Print the final weights and bias
  print("Final Weights:", perceptron.weights)
  print("Final Bias:", perceptron.bias)
  print()

  # Test the perceptron
  print(f"{test_name} Gate Truth Table:")
  print()
  for inputs, label in zip(input_combinations, labels):
    prediction = perceptron.predict(inputs, debug=False)
    print(f"Inputs: {inputs}, Label: {label}, Prediction: {prediction}")
```

TRAINING AND TESTING

In [177…
```
input_combinations = [(0, 0), (0, 1), (1, 0), (1, 1)]
and_labels = [0, 0, 0, 1]
perceptron = Perceptron(num_inputs=2)
train_test_perceptron(perceptron, "AND", input_combinations,
                      and_labels, learning_rate=0.1, epochs=100)
```

```
Training and Testing Perceptron for AND logic:

Final Weights: [0.22308991 0.17469778]
Final Bias: -0.3342066071121244

AND Gate Truth Table:

Inputs: (0, 0), Label: 0, Prediction: 0
Inputs: (0, 1), Label: 0, Prediction: 0
Inputs: (1, 0), Label: 0, Prediction: 0
Inputs: (1, 1), Label: 1, Prediction: 1
```

In [178…
```
input_combinations = [(0, 0), (0, 1), (1, 0), (1, 1)]
or_labels = [0, 1, 1, 1]
perceptron = Perceptron(num_inputs=2)
train_test_perceptron(perceptron, "OR", input_combinations,
                      or_labels, learning_rate=0.1, epochs=100)
```

```
Training and Testing Perceptron for OR logic:

Final Weights: [0.67805395 0.24452752]
Final Bias: -0.01447056038141556

OR Gate Truth Table:

Inputs: (0, 0), Label: 0, Prediction: 0
Inputs: (0, 1), Label: 1, Prediction: 1
Inputs: (1, 0), Label: 1, Prediction: 1
Inputs: (1, 1), Label: 1, Prediction: 1
```

In [179…
```
not_input_combinations = [(0,), (1,)]
not_labels = [1, 0]
perceptron = Perceptron(num_inputs=1)
train_test_perceptron(perceptron, "NOT", not_input_combinations,
                      not_labels, learning_rate=0.1, epochs=100)
```

```
Training and Testing Perceptron for NOT logic:

Final Weights: [-0.240034]
Final Bias: 0.07345579639606631

NOT Gate Truth Table:

Inputs: (0,), Label: 1, Prediction: 1
Inputs: (1,), Label: 0, Prediction: 0
```

In [180…
```
input_combinations = [(0, 0), (0, 1), (1, 0), (1, 1)]
nand_labels = [1, 1, 1, 0]
perceptron = Perceptron(num_inputs=2)
```

```
train_test_perceptron(perceptron, "NAND", input_combinations,
                      nand_labels, learning_rate=0.1, epochs=100)
```

```
Training and Testing Perceptron for NAND logic:

Final Weights: [-0.24765401 -0.15792179]
Final Bias: 0.3432164645210437

NAND Gate Truth Table:

Inputs: (0, 0), Label: 1, Prediction: 1
Inputs: (0, 1), Label: 1, Prediction: 1
Inputs: (1, 0), Label: 1, Prediction: 1
Inputs: (1, 1), Label: 0, Prediction: 0
```