Lesson 2: Understanding Neural Networks

How Do Neural Networks Work?
A neural network processes data through layers of nodes (or neurons), with each layer learning to extract relevant features from the data. Each connection has an associated weight, which is adjusted through the learning process.

Perceptron Model:
The perceptron is the simplest form of a neural network. It consists of a single layer of output neurons. In practice, multi-layered perceptrons (MLP) are used for more complex tasks.

Multilayer Perceptron (MLP):
An MLP consists of an input layer, one or more hidden layers, and an output layer. The deeper the network, the more complex patterns it can learn.

Code Example: Building a Neural Network in PyTorch

```
python
import torch
import torch.nn as nn
import torch.optim as optim

 Define a simple neural network
class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(5, 10)   5 input features, 10 neurons in the hidden
layer
        self.relu = nn.ReLU()         ReLU activation function
        self.fc2 = nn.Linear(10, 1)   Output layer

    def forward(self, x):
        x = self.relu(self.fc1(x))   Forward propagation
        return torch.sigmoid(self.fc2(x))   Sigmoid for binary classification

 Create model instance
model = SimpleNN()

 Print the model architecture
print(model)
```

Key Topics to Explore:
- Forward and Backpropagation: Forward propagation is when data moves from the input to the output layer, while backpropagation adjusts the weights based on the loss.
- Training the Model:Learn how the optimizer and loss functions work together to update the network.