Lesson 10: Convolutional Neural Networks (CNNs)

Overview: Convolutional Neural Networks (CNNs) are specialized types of neural networks designed for processing structured grid data, such as images. CNNs use convolution layers to apply filters to input data and extract important features, reducing the need for manual feature extraction.

Key Concepts:

Convolution Layer: The core layer of a CNN, where filters are applied to input data to generate feature maps.
Pooling Layer: Reduces the dimensionality of the feature maps, which helps in making the network more computationally efficient.
Fully Connected Layer: A standard neural network layer that connects every neuron to every neuron in the next layer.
Mathematical Formulation:

Convolution: The output of the convolution operation is computed by multiplying the filter with patches of the input data.
Activation: After the convolution, an activation function like ReLU is applied to introduce non-linearity.
Pooling: A pooling operation (like max pooling) reduces the spatial dimensions.
Code Example (Simple CNN in Keras):

Chapter 10: Recurrent Neural Networks (RNNs)
Overview: Recurrent Neural Networks (RNNs) are a class of neural networks designed for sequence prediction problems. Unlike traditional neural networks, RNNs have connections that loop back on themselves, allowing information to persist across time steps.

Key Concepts:

Hidden State: The internal memory of an RNN that keeps track of the information across time steps.
Vanishing Gradient Problem: A common problem in training RNNs, where gradients become too small to learn effectively as they propagate backward.
Long Short-Term Memory (LSTM): A type of RNN that mitigates the vanishing gradient problem by using gates to control the flow of information.
Mathematical Formulation:

Code example
python
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Define the model
model = Sequential()

# Add an LSTM layer with 50 units and input shape
```

```
model.add(LSTM(50, activation='relu', input_shape=(10, 1)))

# Add a fully connected layer
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Summary of the model architecture
model.summary()
```

These chapters cover foundational but complex topics in deep learning, providing theoretical explanations along with practical code examples to help in understanding how these concepts are implemented.