NNDL Assignment 2

1. Explain the architecture of a **Recurrent Neural Network (RNN)**. Derive the forward pass equations and discuss how weight sharing works across time steps.
2. What is the **vanishing and exploding gradient problem** in RNNs? Explain mathematically why it occurs and how it affects learning long-term dependencies.
3. Describe the **Bidirectional RNN (Bi-RNN)** architecture. How does it improve performance compared to standard RNNs? Give suitable applications.
4. Explain the **Encoder–Decoder (Sequence-to-Sequence) architecture** with a neat diagram. How is it used in machine translation?
5. Discuss the structure of **Deep Recurrent Neural Networks**. How are they different from shallow RNNs? What are their advantages and challenges?
6. Explain **Long Short-Term Memory (LSTM)** networks. Describe the role of input, forget, and output gates with equations.
7. Compare **LSTM and Gated Recurrent Unit (GRU)**. Highlight their similarities, differences, and use cases.
8. What are **Recursive Neural Networks**? How are they different from recurrent neural networks? Explain with an example (e.g., tree structures in NLP).
9. Define an **Autoencoder**. Explain its architecture, objective function, and how it performs feature learning.
10. Discuss different **types of Autoencoders**:

- Vanilla Autoencoder
- Denoising Autoencoder
- Convolutional Autoencoder
- Deep Autoencoder

Explain their working and applications.

11. What is **regularization in Autoencoders**? Explain techniques like sparse autoencoders and dropout. Why is regularization important?

Answer any two of the following questions:

1. **Implement a Convolutional Neural Network (CNN) for Image Recognition**
   Develop and implement a CNN model for an image recognition task such as handwritten digit recognition, plant disease detection, or face detection. Train and test the model using an appropriate dataset. Include code screenshots, model architecture, and sample output results in your submission.
2. **Implement a Recurrent Neural Network (RNN) for Sequence Prediction**
   Develop and implement an RNN model for a sequence-based application such as text generation, sentiment analysis, or time-series prediction. Train and evaluate the model using a suitable dataset. Include code screenshots, model architecture, and output predictions in your submission.
3. **Implement an Autoencoder for Data Compression or Image Denoising**
   Develop and implement an Autoencoder model for tasks such as dimensionality reduction, data compression, or image denoising. Train and test the model on an appropriate dataset. Include code screenshots, encoder-decoder architecture, and sample reconstructed outputs in your submission.