Introduction to Deep Learning

# Assignment no. 2

Due Data: 04-06-25

**Part A: Model Training on Simple Time Series Data (25 Marks)**

**Q1. Predicting Future Values with a Simple RNN**

Download the classic international airline passengers dataset from this link:  
[**International Airline Passengers Dataset - CSV**](https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv)

This dataset contains monthly totals of international airline passengers from 1949 to 1960.

**Tasks:**

1. **Data Preparation**
   * Load the dataset and plot the number of passengers versus time.
   * Normalize the data to the range [0, 1].
   * Prepare the data for a simple sequence-to-one prediction: use the previous 5 months to predict the passenger number for the next month (i.e., use a sliding window of 5 time steps as input and 1 time step as output).
   * Split the dataset into training (first 80%) and testing (last 20%).
2. **Model Implementation and Training**
   * Implement a simple RNN model in **PyTorch** or **TensorFlow/Keras** (your choice), using a single RNN layer (e.g., nn.RNN or tf.keras.layers.SimpleRNN) followed by a dense output layer.
   * Train the model for at least **30 epochs**. Plot the training and validation loss curves.
3. **Evaluation and Interpretation**
   * Plot the predicted versus actual values for the test set.
   * Briefly describe what you observe about the model’s ability to capture patterns in the data. What are its limitations?
   * What would you try to improve the performance (mention at least one idea: e.g., deeper RNN, LSTM/GRU, more data, etc.)?

**Submission:**

* Submit your complete code (Jupyter notebook or Python script).
* Include all plots and a brief discussion (2-3 paragraphs) in your notebook/script.

**Part B: Theory ( 20 Marks)**

**Q2.**

(a) Explain how sequential learning differs from traditional supervised learning. Give a real-world example where sequential learning provides significant advantages.

(b) Summarize the main differences between CNNs and RNNs in terms of input structure, weight sharing, and typical application domains.

(c) Why is the tanh or sigmoid function commonly used as the activation function in RNNs? What could go wrong if you used the ReLU activation instead?

**Part C: Manual RNN Training (Hand Calculation, 15 Marks)**

**Q3. Manual RNN Training on a Simple Dataset**

Consider the following toy sequential dataset:

|  |  |
| --- | --- |
| **Input (x)** | **Target (y)** |
| 1 | 0 |
| 0 | 1 |
| 1 | 1 |

We will use a **vanilla RNN cell** (one input, one hidden unit, one output) defined as:

* **Hidden State Update:**
* **Output Calculation:**  
  where σ is the sigmoid function.

**Initial Parameters:**

* ,

**Task:**

1. **Forward Pass:** For the input sequence [1, 0, 1], compute the hidden states and outputs for all time steps.
2. **Loss Calculation:** Use **binary cross-entropy loss** for each step:  
   (where yt is the true target and y^t predicted is the output).
3. **Backpropagation Through Time (BPTT):** Manually compute gradients for **one epoch** (all three steps) and update all weights using a learning rate η=0.1\eta = 0.1.  
   (Show step-by-step calculations for the first epoch only.)
4. **Second Epoch:** Repeat the forward pass using the **updated weights** (from Step 3) and report the new outputs for each time step.

**Show all calculations clearly. Partial credit will be awarded for correct steps, even if the final answer is incorrect.**

**Submission Guidelines**

* Submit your code and notebook/scripts for coding parts.
* Typewritten PDF for theory and manual calculation parts (diagrams may be hand-drawn but must be clear).
* Include all plots and explanations as instructed.