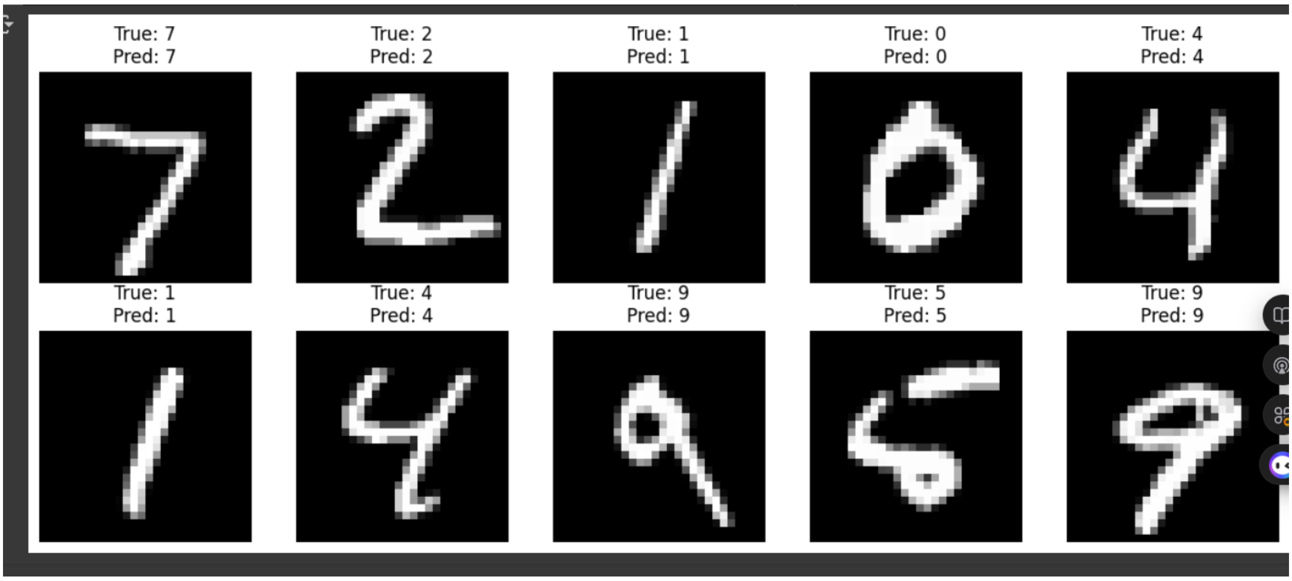
Lab7

Results:



QA:

5. Role of Forget Gate, Input Gate, Output Gate, and Candidate Cell in an LSTM

In an LSTM cell:

- **Forget gate (`f_t`)**:
  Determines how much of the previous cell state (`c_{t-1}`) to retain. It controls memory removal by applying a sigmoid activation, outputting values between 0 and 1.
- **Input gate (`i_t`)**:
  Controls how much new information from the current input (`x_t`) should be written into the cell state.
- **Candidate cell (`c̃_t`)**:
  Generates potential new memory content using a `tanh` activation, which is scaled by the input gate to update the cell state.
- **Output gate (`o_t`)**:
  Decides how much of the cell state should be exposed to the next layer or time step via the hidden state (`h_t`), using `tanh(c_t)` and a sigmoid gate.

These components together enable LSTM to preserve long-term dependencies while selectively updating or forgetting information.

6. Hyperparameters Tuned and Their Effect on Accuracy

I tuned the following hyperparameters:

- **Hidden size (128)**:
  Increased model capacity, allowing it to better capture patterns across time steps, which improved accuracy.
- **Learning rate (0.0005)**:
  A slightly higher learning rate accelerated convergence while remaining stable, helping the model reach a better solution within fewer epochs.
- **Number of epochs (10)**:
  Training longer allowed the model to better fit the data, boosting test accuracy to ~97.7%.

Each change contributed to performance gains, with the number of epochs and hidden size having the most noticeable impact.

---

7. Between a simple RNN and an LSTM, which one is better for sequence learning tasks? Explain your reasoning, and discuss in which situations LSTM is more useful and in which situations a simple RNN might still be sufficient.

LSTM is generally **better than simple RNNs** for most sequence learning tasks, especially when:

- **Long-term dependencies** are present (e.g., language modeling, time series prediction).
- RNNs struggle with **vanishing gradients**, causing them to "forget" information after a few steps.

LSTM is more useful when:

- Tasks require memory of events far apart in a sequence.
- The input data has complex temporal dynamics (e.g., speech recognition, video analysis).

Simple RNNs might be sufficient when:

- Sequences are short and the task is relatively simple (e.g., digit recognition from sequences like MNIST rows).
- Computational resources are limited, and model simplicity is preferred.