

Hyper-parameter	V1	V2
batch size	128	256
learning rate	0.001	0.0005
hidden size	128	256
epoch	10	15
optimizer	Adam	AdamW

### **Explain briefly the role of the forget gate, input gate, output gate, and candidate cell in an LSTM.**

In an LSTM,

- The forget gate decides what old stuff we should throw away.
- The input gate chooses what new stuff we should save.
- The candidate cell is like a new idea we might want to add.
- The output gate picks what parts we want to show to the next layer.

They all work together to help the model remember important things and forget useless things!

### **Describe what hyperparameters you tuned and how they affected your model's final accuracy.**

I made the hidden size bigger, from 128 to 256, so the model could learn more patterns. I made the learning rate a little smaller, from 0.001 to 0.0005, so it could train smoother without jumping around too much.

And I also used “AdamW” instead of Adam and added weight decay to help the model not overfit. Plus, I used a cosine scheduler to slowly lower the learning rate over time. After these changes, the training loss got way lower, and the test accuracy improved a little. It made the model more stable and better at learning!

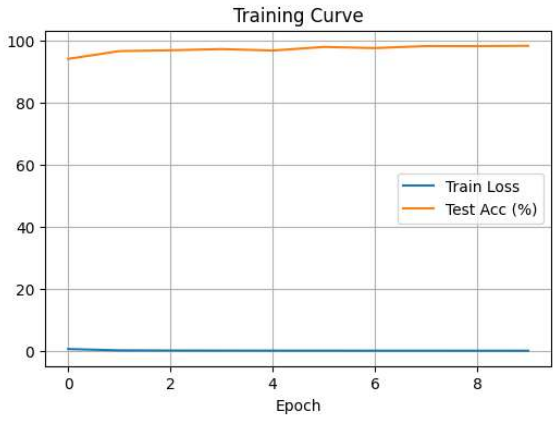
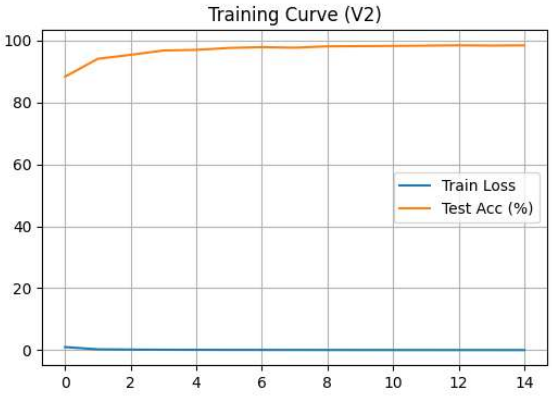
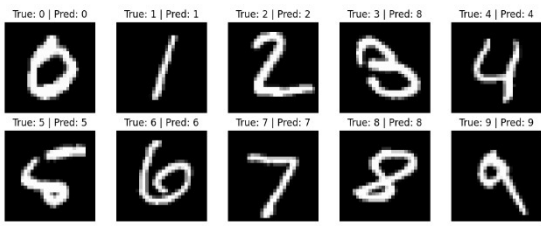
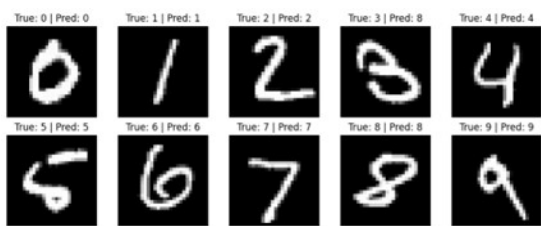
### **Between a simple RNN and an LSTM, which one is better for sequence learning tasks? Explain your reasoning, and discuss in which situations LSTM is more useful and in which situations a simple RNN might still be sufficient.**

LSTMs are better for most sequence learning tasks because they can remember things for a longer time. Simple RNNs forget stuff too fast, so they're not good when you need to connect things far apart, like in long sentences or videos.

But if the sequence is really short and simple, like just a few steps, a basic RNN can work fine and is faster to train.

So, LSTM is great for long stuff like text, music, and speech. RNNs are okay for tiny sequences where you don't need long memory.



V1	V2 (after fine-tuning)
	
	
<p>Epoch [1/10]   Loss: 0.5761   Test Acc: 94.16%</p> <p>Epoch [2/10]   Loss: 0.1521   Test Acc: 96.66%</p> <p>Epoch [3/10]   Loss: 0.1011   Test Acc: 96.93%</p> <p>Epoch [4/10]   Loss: 0.0799   Test Acc: 97.32%</p> <p>Epoch [5/10]   Loss: 0.0632   Test Acc: 96.85%</p> <p>Epoch [6/10]   Loss: 0.0550   Test Acc: 98.01%</p> <p>Epoch [7/10]   Loss: 0.0451   Test Acc: 97.64%</p> <p>Epoch [8/10]   Loss: 0.0425   Test Acc: 98.29%</p> <p>Epoch [9/10]   Loss: 0.0348   Test Acc: 98.27%</p> <p>Epoch [10/10]   Loss: 0.0322   Test Acc: 98.38%</p>	<p>Epoch 1/15   Loss 0.9864   Test Acc 88.31%   lr 4.95e-04</p> <p>Epoch 2/15   Loss 0.2649   Test Acc 94.10%   lr 4.78e-04</p> <p>Epoch 3/15   Loss 0.1723   Test Acc 95.37%   lr 4.52e-04</p> <p>Epoch 4/15   Loss 0.1226   Test Acc 96.77%   lr 4.17e-04</p> <p>Epoch 5/15   Loss 0.0969   Test Acc 96.97%   lr 3.75e-04</p> <p>Epoch 6/15   Loss 0.0767   Test Acc 97.59%   lr 3.27e-04</p> <p>Epoch 7/15   Loss 0.0655   Test Acc 97.86%   lr 2.76e-04</p> <p>Epoch 8/15   Loss 0.0546   Test Acc 97.67%   lr 2.24e-04</p> <p>Epoch 9/15   Loss 0.0449   Test Acc 98.12%   lr 1.73e-04</p> <p>Epoch 10/15   Loss 0.0370   Test Acc 98.18%   lr 1.25e-04</p> <p>Epoch 11/15   Loss 0.0307   Test Acc 98.25%   lr 8.27e-05</p> <p>Epoch 12/15   Loss 0.0260   Test Acc 98.33%   lr 4.77e-05</p> <p>Epoch 13/15   Loss 0.0217   Test Acc 98.43%   lr 2.16e-05</p> <p>Epoch 14/15   Loss 0.0196   Test Acc 98.35%   lr 5.46e-06</p> <p>Epoch 15/15   Loss 0.0182   Test Acc 98.42%   lr 0.00e+00</p>
<p>In the beginning, I tried the simple version (V1). The accuracy got up to 98.38% after 10 epochs, which was already pretty good! But I noticed the loss stopped going down much after like 6 or 7 epochs. It kinda stayed the same.</p> <p>So I tried tuning the model (V2). I made the hidden size bigger (256 instead of 128), lowered the learning rate, and added some other stuff like weight decay, dropout, and a learning rate scheduler. At first, V2 started slower, but after a while, it caught up and got even better. In the end, it reached 98.42% accuracy, and the loss was much smaller than V1. Also, the predictions were more stable and all 10 images (0-9) were correct, while V1 had one mistake.</p> <p>I think the bigger hidden size helped the model learn better patterns. Also, weight decay and dropout made it not overfit, which is super useful when the model gets big. The scheduler helped the learning go slow and steady near the end, so it didn't miss the tiny details. Overall, tuning really helped. (Even though the accuracy only went up a little but it works, right?!) </p>	

