a1

Liheng Xiao

July 2025

1 Part 1

1.1

```
Test set: Average loss: 1.0089, Accuracy: 6958/10000 (70%)
 [  8.  60. 689.  26.  27.  22.  47.  38.  46.  37.]
 [  4.  36.  56. 759.  16.  58.  14.  18.  28.  11.]
 [ 60.  51.  80.  19. 624.  19.  32.  37.  21.  57.]
 [  8.  28. 125.  16.  20. 724.  27.   8.  33.  11.]
 [  5.  22. 145.   9.  25.  24. 723.  21.   9.  17.]
 [ 17.  30.  26.  12.  83.  16.  55. 623.  89.  49.]
 [ 11.  35.  92.  42.   8.  32.  46.   7. 705.  22.]
 [  8.  60. 689.  26.  27.  22.  47.  38.  46.  37.]
 [  4.  36.  56. 759.  16.  58.  14.  18.  28.  11.]
 [ 60.  51.  80.  19. 624.  19.  32.  37.  21.  57.]
 [  8.  28. 125.  16.  20. 724.  27.   8.  33.  11.]
 [  5.  22. 145.   9.  25.  24. 723.  21.   9.  17.]
 [ 17.  30.  26.  12.  83.  16.  55. 623.  89.  49.]
 [  8.  60. 689.  26.  27.  22.  47.  38.  46.  37.]
 [  4.  36.  56. 759.  16.  58.  14.  18.  28.  11.]
 [ 60.  51.  80.  19. 624.  19.  32.  37.  21.  57.]
 [  8.  28. 125.  16.  20. 724.  27.   8.  33.  11.]
 [  5.  22. 145.   9.  25.  24. 723.  21.   9.  17.]
 [  8.  60. 689.  26.  27.  22.  47.  38.  46.  37.]
 [  4.  36.  56. 759.  16.  58.  14.  18.  28.  11.]
 [  4.  36.  56. 759.  16.  58.  14.  18.  28.  11.]
 [ 60.  51.  80.  19. 624.  19.  32.  37.  21.  57.]
 [  8.  28. 125.  16.  20. 724.  27.   8.  33.  11.]
 [  8.  28. 125.  16.  20. 724.  27.   8.  33.  11.]
 [  5.  22. 145.   9.  25.  24. 723.  21.   9.  17.]
 [  5.  22. 145.   9.  25.  24. 723.  21.   9.  17.]
 [ 17.  30.  26.  12.  83.  16.  55. 623.  89.  49.]
 [ 17.  30.  26.  12.  83.  16.  55. 623.  89.  49.]
 [ 11.  35.  92.  42.   8.  32.  46.   7. 705.  22.]
 [ 11.  35.  92.  42.   8.  32.  46.   7. 705.  22.]
 [  9.  51.  87.   3.  55.  31.  18.  31.  39. 676.]]
 [  9.  51.  87.   3.  55.  31.  18.  31.  39. 676.]]
```

Figure 1: NetLin

$$\underbrace{784 \times 10}_{\text{weights}} + \underbrace{10}_{\text{biases}} = 7{,}850$$

1.2

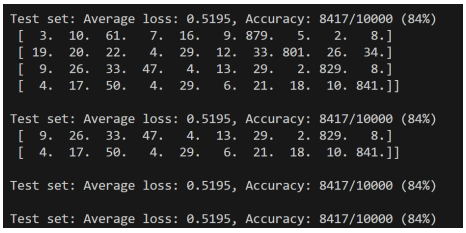

Figure 2: NetFull

$$\underbrace{784 \times 100 + 100}_{\text{first layer}} + \underbrace{100 \times 10 + 10}_{\text{second layer}} = 78400 + 100 + 1000 + 10 = 79510$$

1.3

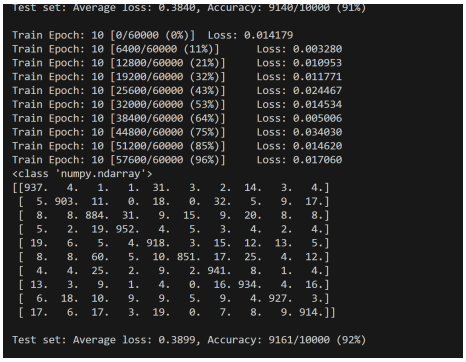

Figure 3: NetConv

$$\underbrace{32 \times 1 \times 3 \times 3 + 32}_{\text{Conv1}} + \underbrace{64 \times 32 \times 3 \times 3 + 64}_{\text{Conv2}} + \underbrace{64 \times 24 \times 24 \times 128 + 128}_{\text{FC1}} + \underbrace{128 \times 10 + 10}_{\text{FC2}}$$

which expands numerically as

$$(288{+}32){+}(18432{+}64){+}(36864{\times}128{+}128){+}(1280{+}10) = 320{+}18496{+}4718720{+}1290 = 4738826.$$

1.4 Discussion

(a) **Relative accuracy of the three models:** NetLin achieves around 70% accuracy, reflecting its limited capacity to separate complex handwritten patterns. NetFull improves to about 84% accuracy thanks to its non-linear hidden layer, while NetConv achieves 92–93% accuracy by exploiting spatial feature extraction through convolutional layers.

(b) **Number of independent parameters:** NetLin has 7850 parameters, NetFull has 79510, and NetConv has approximately 4738826 parameters, reflecting the increasing complexity and representational power of each model.

(c) **Confusion matrix analysis:** The confusion matrices indicate frequent misclassification between characters with similar structures, such as “su” and “tsu”, or “o” and “wo”, which differ only by subtle strokes or diacritical marks. NetConv reduces these confusions significantly compared to simpler models, but still struggles with nearly identical handwritten shapes.

2 Part 2

2.1

Figures 4 and 5 show the trained network with 5 hidden nodes using sigmoid activations. These hidden units create flexible, non-linear decision regions that perfectly classify all 9 samples in the dataset.

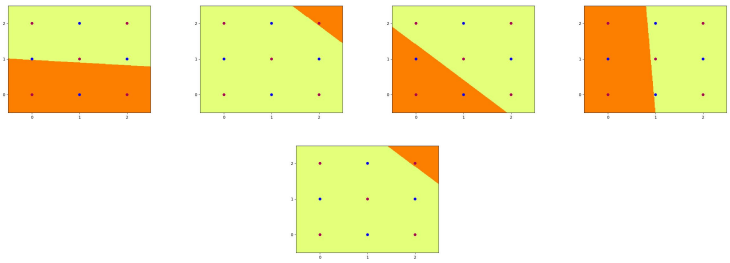

Figure 4: Decision boundaries of the 5 hidden units with sigmoid activation.

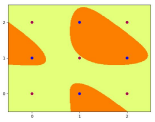

Figure 5: Overall output decision boundary of the trained sigmoid network.

2.2

Figures 6 and 7 show the manually designed 4-hidden-node network using step activations. Each hidden node produces a linear threshold to split the input space, and the output node combines them to form a cross-shaped class-1 prediction region in the center.

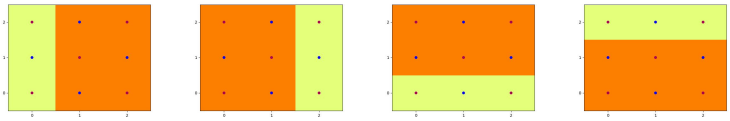

Figure 6: Decision boundaries of the 4 hidden units with Heaviside activation.

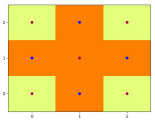

Figure 7: Overall output decision boundary of the manually designed step-activation network.

Table 1: Activations of the 4 hidden units and the output unit for each of the 9 training samples

x	y	h_1	h_2	h_3	h_4	o
0	0	0	1	0	1	0
1	0	1	1	0	1	0
2	0	1	0	0	1	0
0	1	0	1	1	1	0
1	1	1	1	1	1	1
2	1	1	0	1	1	0
0	2	0	1	1	0	0
1	2	1	1	1	0	0
2	2	1	0	1	0	0

2.3

To verify that a sigmoid network can approximate a step activation function, the manually designed weights and biases were multiplied by a factor of 10. This scaling causes the sigmoid activation to transition sharply, closely mimicking a hard threshold. Applying these scaled weights to the sigmoid network correctly classified all 9 training samples with 100% accuracy. Figures 8 and 9 illustrate the decision boundaries of each hidden node and the final output node under the rescaled weights.

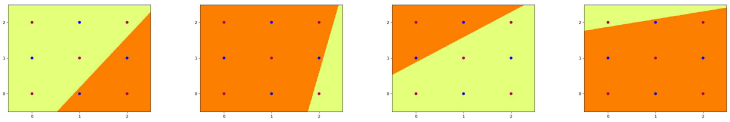

Figure 8: Decision boundaries of the 4 hidden units with sigmoid activations after rescaling.

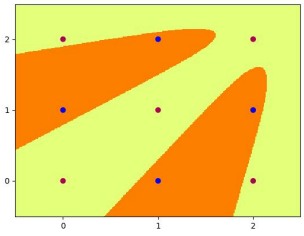

Figure 9: Overall output decision boundary of the sigmoid network with rescaled weights, closely mimicking the step activation.

3 Part 3

3.1

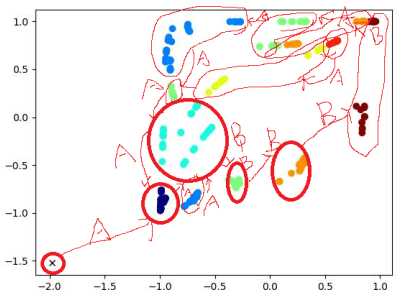

Figure 10: finite state machine

3.2

The SRN uses its hidden activations to count how many A symbols it has seen, moving between clusters that represent different counting states. Once the first B is observed, the hidden state switches to a region representing the B sequence, where it can deterministically predict all remaining Bs. After finishing the Bs, the network resets to predict the initial A of the next sequence. This mechanism allows correct predictions after the first ambiguous positions.

3.3

Figures have showed the learned hidden state trajectories of the LSTM with 3 hidden units on the $a^n b^{2n} c^{3n}$ prediction task, visualized from different angles. These plots illustrate how the LSTM clusters hidden activations to reliably distinguish between different segments of the input sequence (A, B, and C phases), enabling it to correctly predict the sequence after observing the first uncertain symbols.

3.4

The LSTM successfully solves the $a^n b^{2n} c^{3n}$ prediction task by separating memory into its cell state and local transitions into its hidden activations. As As are processed, the LSTM's cell state c_t is incrementally updated to store n, while the hidden units h_t remain in a cluster representing the "A phase". When the first B arrives, the hidden activations shift to a new cluster (the "B phase"), but the cell retains n to guide prediction of $2n$ Bs. This continues through the C symbols, where the same n stored in the cell supports predicting $3n$ Cs. Finally, after the last C, the hidden activations return to the initial cluster, and the cell

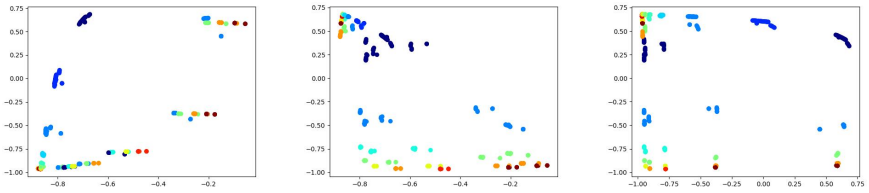

Figure 11: Visualisation of LSTM hidden activations on the $a^n b^{2n} c^{3n}$ task, from three different viewpoints.

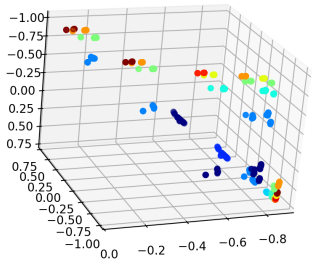

Figure 12: 3d view

state resets, ready for a new sequence. This division of labor allows the LSTM to handle the long-distance dependencies robustly.

To inspect these dynamics more precisely, the code can be modified so that the LSTM model returns both the hidden and cell state in its forward pass, for example:

```
hidden, output, cell = net(input)
```

and then plot or print the cell values alongside the hidden values in `seq_plot.py`.

The overall state transitions can be summarized as shown below:

Table 2: Illustrative state transitions in the LSTM for $a^n b^{2n} c^{3n}$

State	Input	Next State	Cell / Hidden meaning
S0	A	S1	Cell increments n counter
S1	A	S1	Keep incrementing n
S1	B	S2	Switch hidden, retain n in cell
S2	B	S2	Count B's, guided by n in cell
S2	C	S3	Switch hidden, retain n in cell
S3	C	S3	Count C's, guided by n in cell
S3	A	S0	Reset hidden and cell