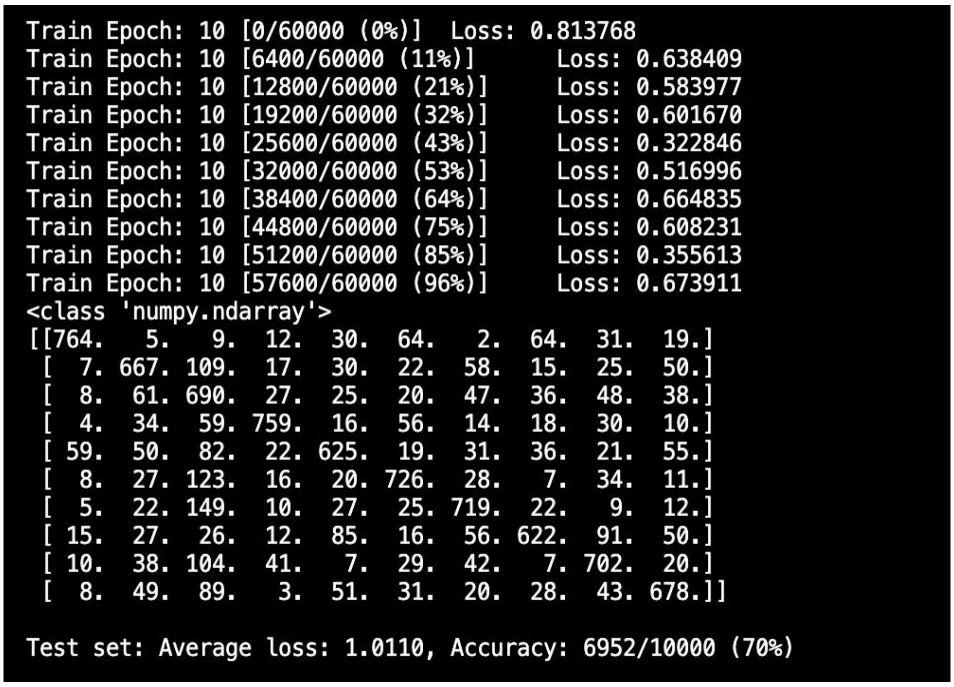
9444 ASS1

Xianjing LAI z5575173

Part 1

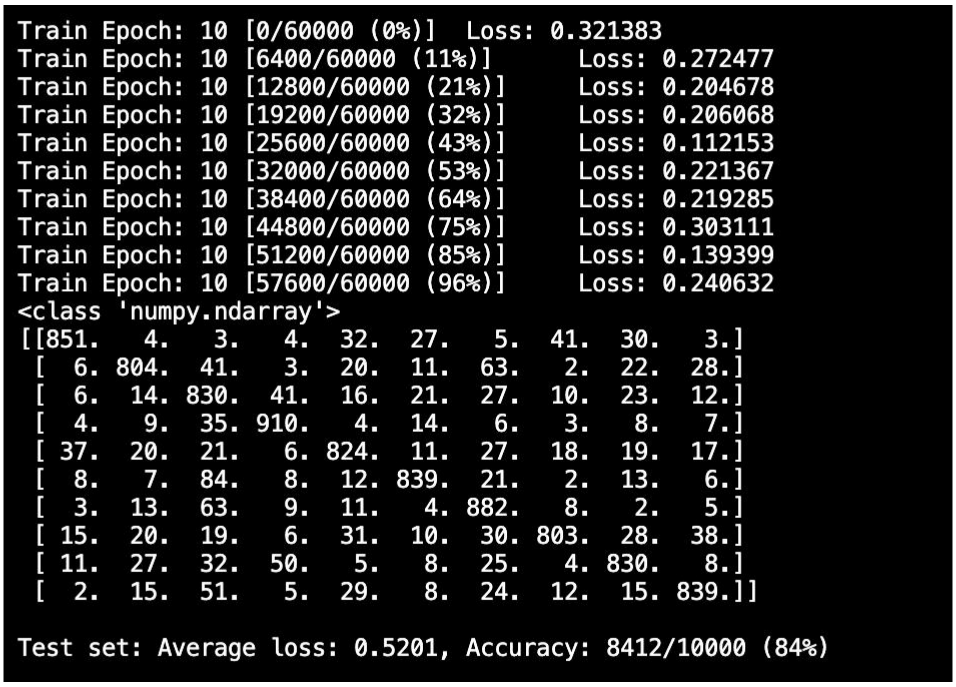
1.



```
Train Epoch: 10 [0/60000 (0%)]  Loss: 0.813768
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.638409
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.583977
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.601670
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.322846
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.516996
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.664835
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.608231
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.355613
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.673911
<class 'numpy.ndarray'>
[[764.   5.   9.  12.  30.  64.   2.  64.  31.  19.]
 [  7. 667. 109.  17.  30.  22.  58.  15.  25.  50.]
 [  8.  61. 690.  27.  25.  20.  47.  36.  48.  38.]
 [  4.  34.  59. 759.  16.  56.  14.  18.  30.  10.]
 [ 59.  50.  82.  22. 625.  19.  31.  36.  21.  55.]
 [  8.  27. 123.  16.  20. 726.  28.   7.  34.  11.]
 [  5.  22. 149.  10.  27.  25. 719.  22.   9.  12.]
 [ 15.  27.  26.  12.  85.  16.  56. 622.  91.  50.]
 [ 10.  38. 104.  41.   7.  29.  42.   7. 702.  20.]
 [  8.  49.  89.   3.  51.  31.  20.  28.  43. 678.]]

Test set: Average loss: 1.0110, Accuracy: 6952/10000 (70%)
```

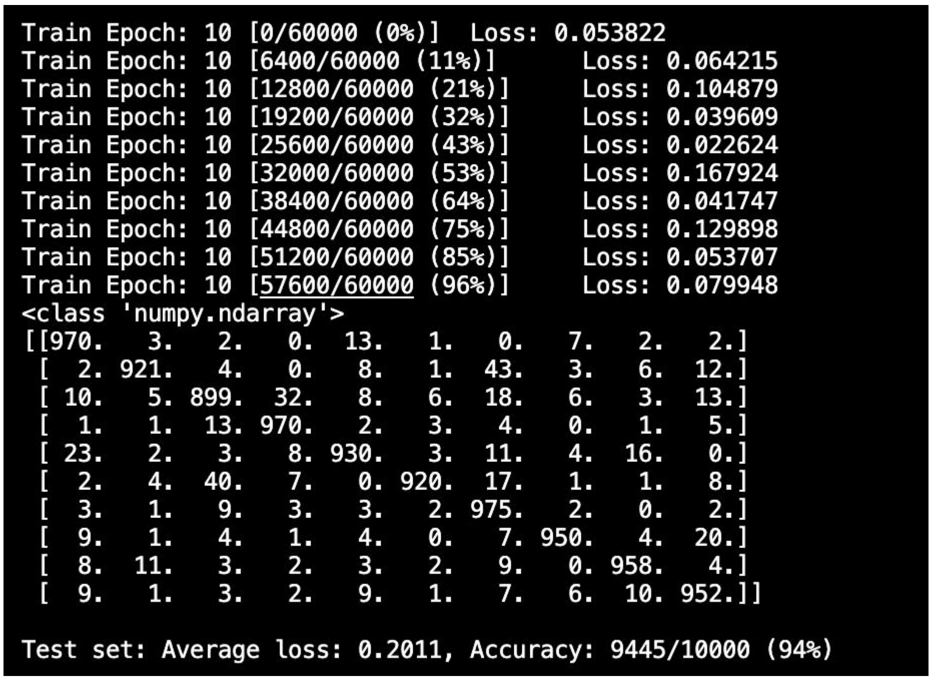
2.



```
Train Epoch: 10 [0/60000 (0%)]  Loss: 0.321383
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.272477
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.204678
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.206068
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.112153
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.221367
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.219285
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.303111
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.139399
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.240632
<class 'numpy.ndarray'>
[[851.   4.   3.   4.  32.  27.   5.  41.  30.   3.]
 [  6. 804.  41.   3.  20.  11.  63.   2.  22.  28.]
 [  6.  14. 830.  41.  16.  21.  27.  10.  23.  12.]
 [  4.   9.  35. 910.   4.  14.   6.   3.   8.   7.]
 [ 37.  20.  21.   6. 824.  11.  27.  18.  19.  17.]
 [  8.   7.  84.   8.  12. 839.  21.   2.  13.   6.]
 [  3.  13.  63.   9.  11.   4. 882.   8.   2.   5.]
 [ 15.  20.  19.   6.  31.  10.  30. 803.  28.  38.]
 [ 11.  27.  32.  50.   5.   8.  25.   4. 830.   8.]
 [  2.  15.  51.   5.  29.   8.  24.  12.  15. 839.]]

Test set: Average loss: 0.5201, Accuracy: 8412/10000 (84%)
```

The total number of parameters is calculated as [(28*28 input features) * 150+150] (first layer)+[150*10 +10] (second layer) =119260 parameters.

3. 

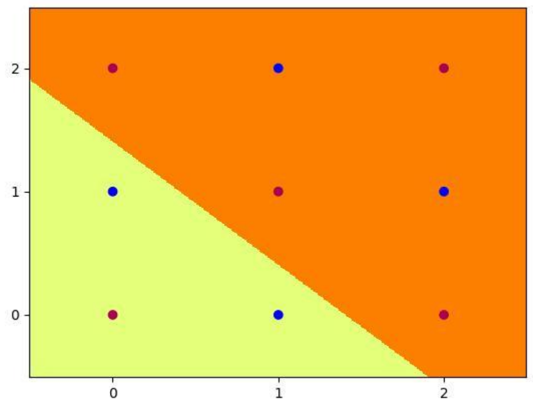
```
Train Epoch: 10 [0/60000 (0%)]  Loss: 0.053822
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.064215
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.104879
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.039609
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.022624
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.167924
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.041747
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.129898
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.053707
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.079948
<class 'numpy.ndarray'>
[[970.   3.   2.   0.  13.   1.   0.   7.   2.   2.]
 [  2. 921.   4.   0.   8.   1.  43.   3.   6.  12.]
 [ 10.   5. 899.  32.   8.   6.  18.   6.   3.  13.]
 [  1.   1.  13. 970.   2.   3.   4.   0.   1.   5.]
 [ 23.   2.   3.   8. 930.   3.  11.   4.  16.   0.]
 [  2.   4.  40.   7.   0. 920.  17.   1.   1.   8.]
 [  3.   1.   9.   3.   3.   2. 975.   2.   0.   2.]
 [  9.   1.   4.   1.   4.   0.   7. 950.   4.  20.]
 [  8.  11.   3.   2.   3.   2.   9.   0. 958.   4.]
 [  9.   1.   3.   2.   9.   1.   7.   6.  10. 952.]]

Test set: Average loss: 0.2011, Accuracy: 9445/10000 (94%)
```

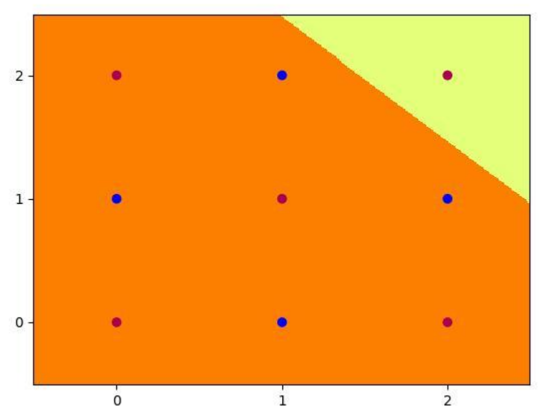
Total parameters are calculated as: Conv1 ((3*3 kernel * 1 input channel)*32 filters + 32 biases) + Conv2 ((3*3 kernel * 32 input channels) * 64 filters + 64 biases) + FC1 ((64 channels * 7*7 spatial dim) * 256 + 256 biases) + FC2 (256 * 10 + 10 biases) = 824458 parameters.

4. Comparing the three models: The linear model (70%) serves as a baseline, the two-layer network (84%) shows the benefit of non-linearity, and the CNN (94%) demonstrates the power of learning spatial hierarchies. Parameter counts grow significantly from NetLin (7860) to NetFull (119260) to NetConv (824458). The confusion matrices reveal that all models struggle most with visually similar characters, but the CNN makes the fewest such errors. The improvement comes from the CNN's ability to learn local features and spatial relationships rather than treating pixels independently.

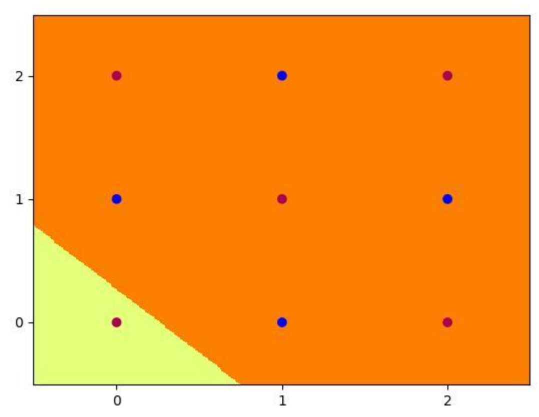
Part 2

1. 

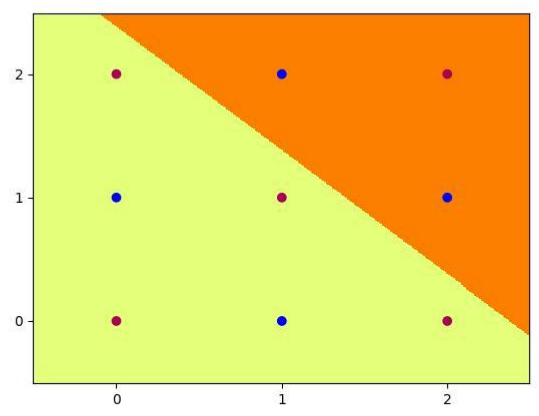
1 hid_5_0



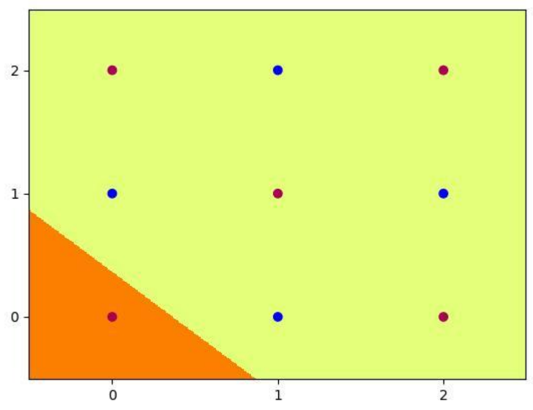
2 hid_5_1



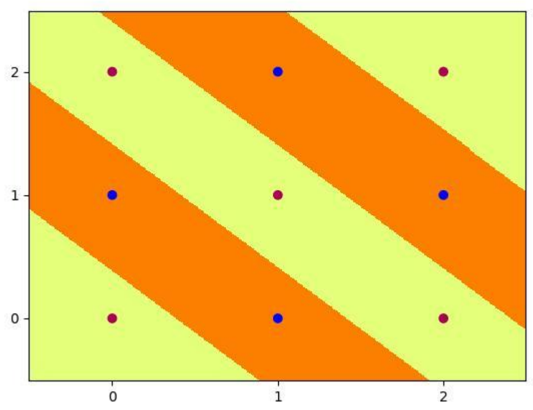
3 hid_5_2



4 hid_5_3

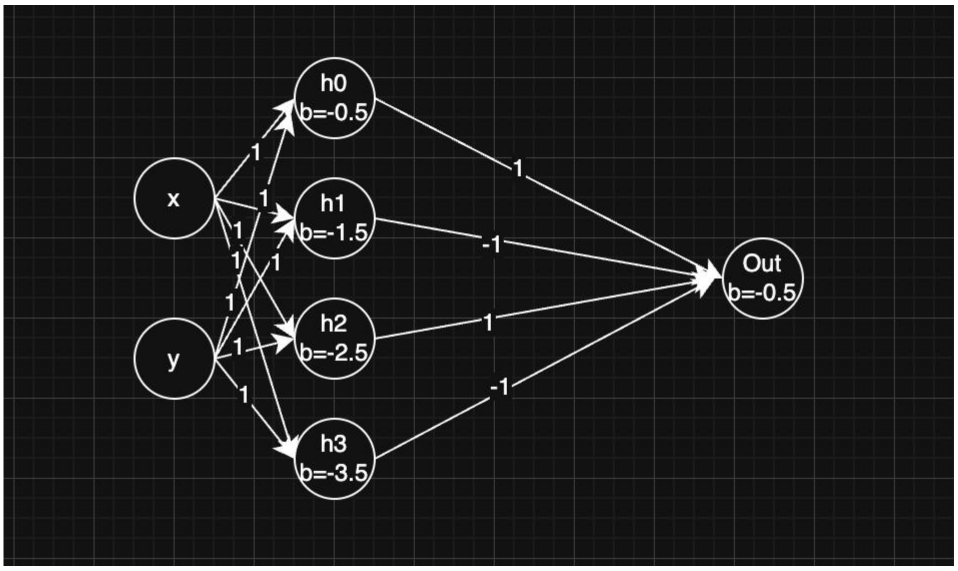


5 hid_5_4



6 out_5

2.

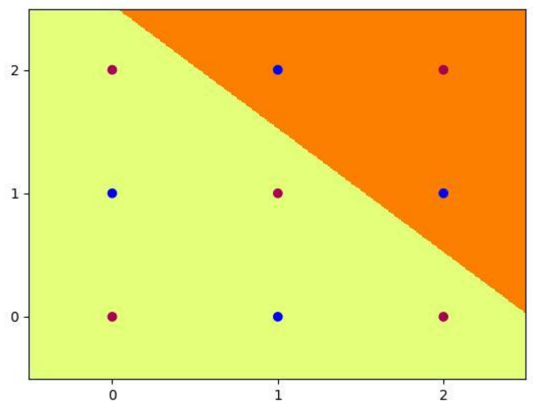


Hidden Node	Equation
h0	$x+y-0.5\geq 0$

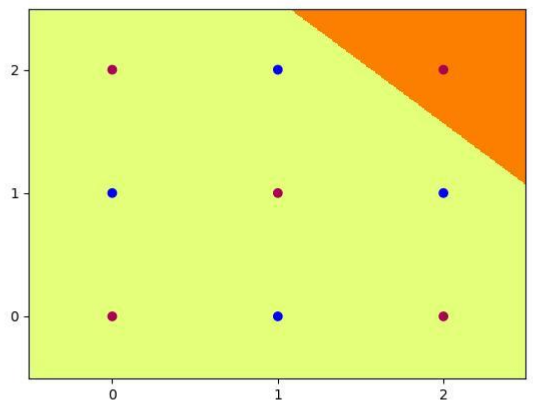
Hidden Node	Equation
h1	$x+y-1.5\geq 0$
h2	$x+y-2.5\geq 0$
h3	$x+y-3.5\geq 0$

X	Y	T	h0	h1	h2	h3	Output
0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1
0	2	0	1	1	0	0	0
1	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0
1	2	1	1	1	1	0	1
2	0	0	1	1	0	0	0
2	1	1	1	1	1	0	1
2	2	0	1	1	1	1	0

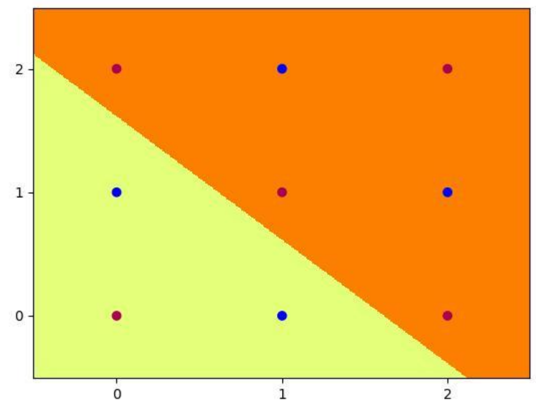
3.



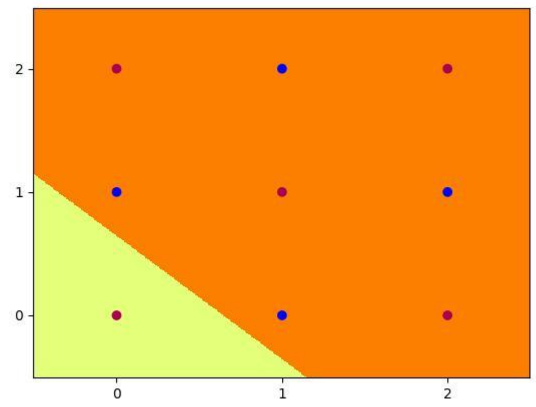
7 hid_4_0



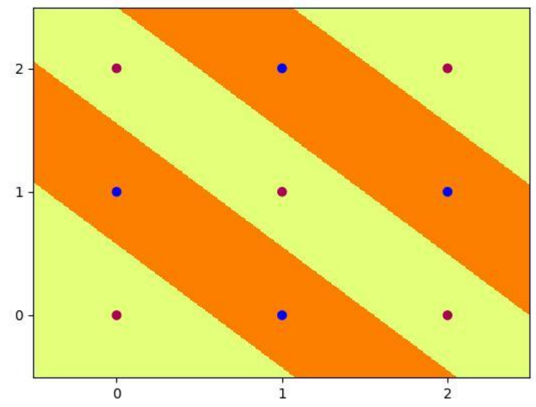
8 hid_4_1



9 hid_4_2

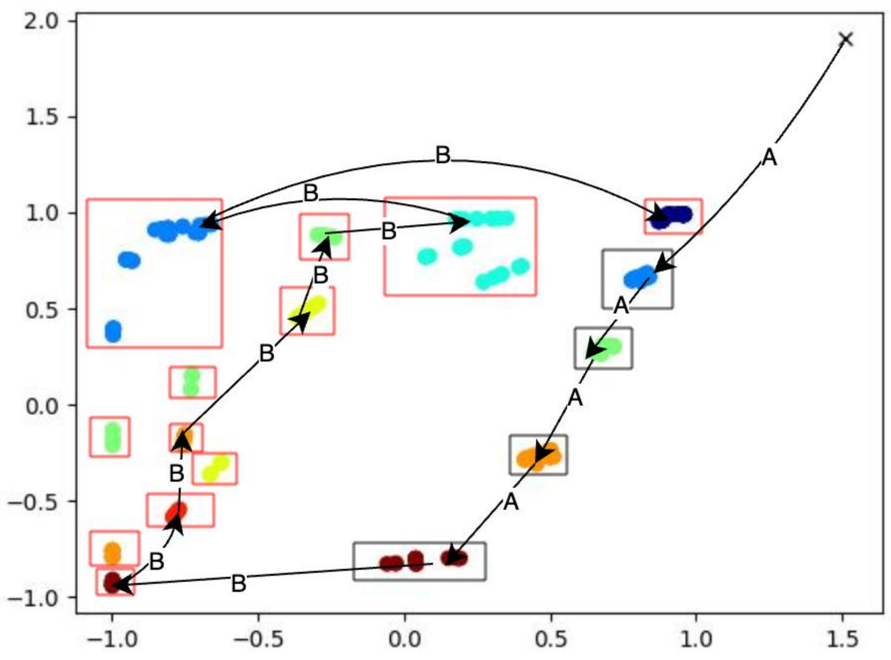


10 hid_4_3

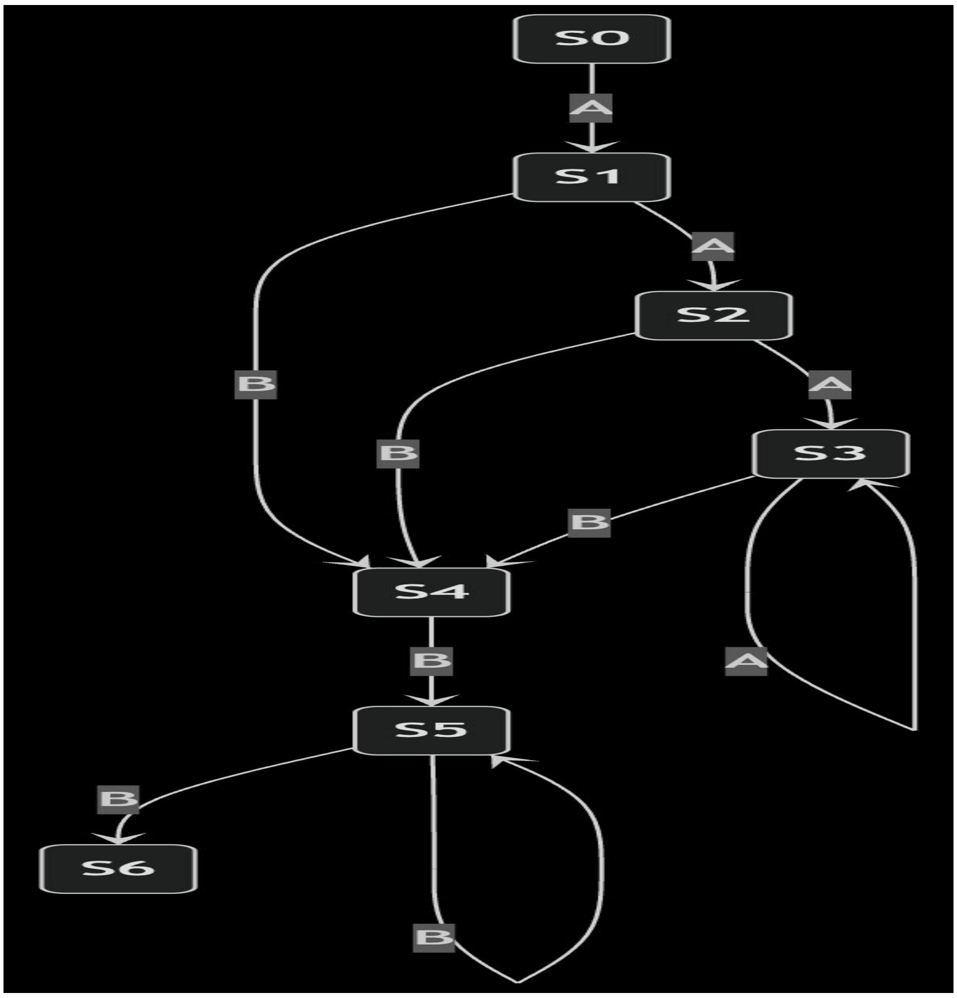


11 out_4

Part 3



1.

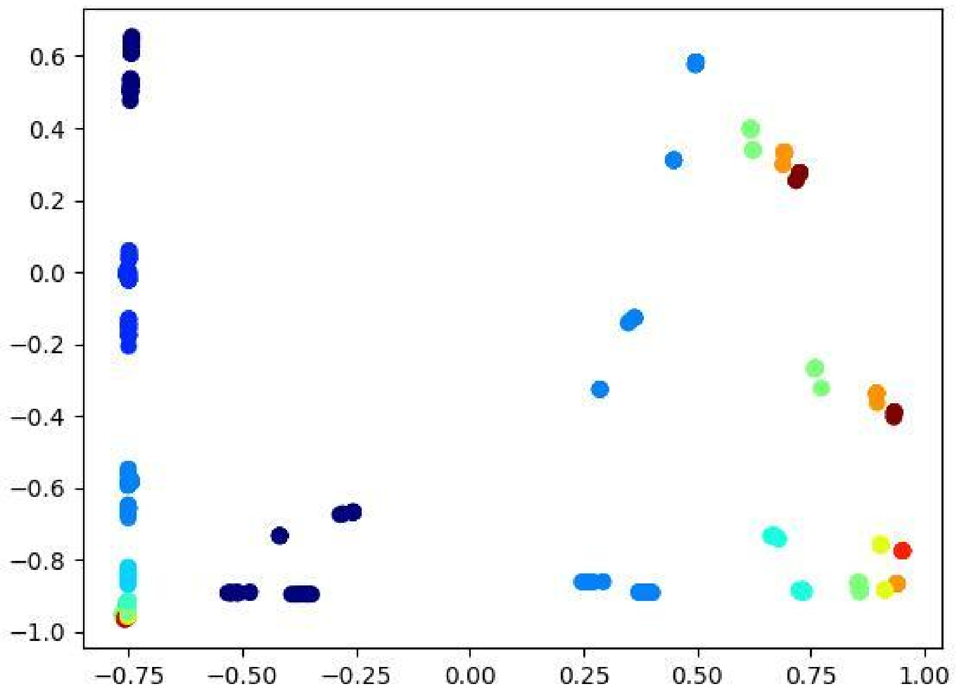


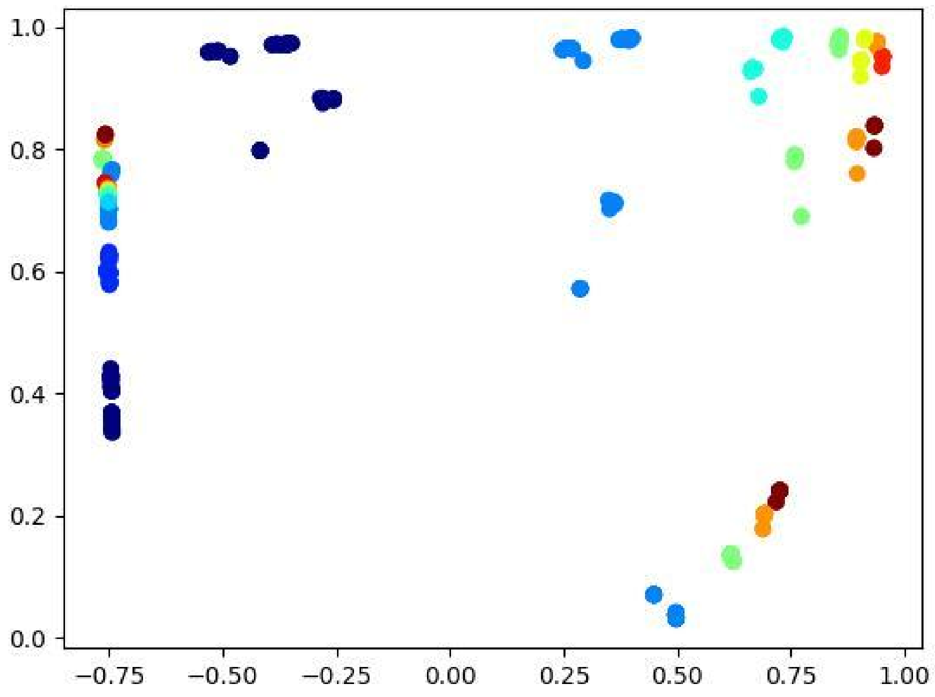
2.

3. The SRN solves the anb2n task by encoding the structure of the sequence in the trajectory of its hidden state activations. Initially, as the network processes a sequence of

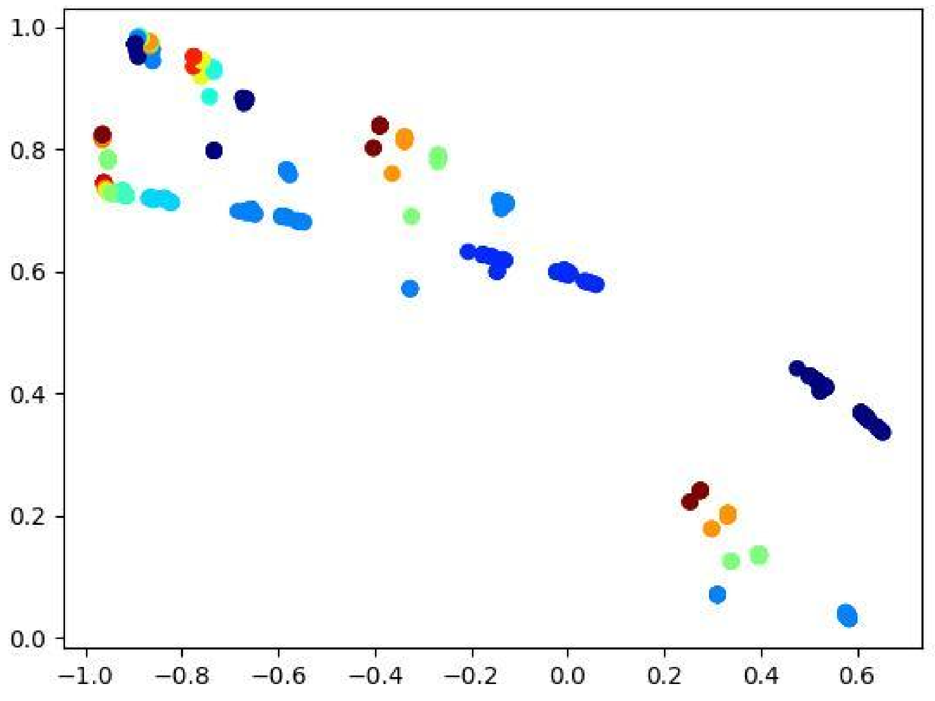
A’s, the hidden state follows a smooth and continuous path in one region of hidden space, gradually shifting with each A, this trajectory effectively acts as a memory, storing the number of A’ s seen so far. When the first B appears, the network transitions sharply to a different region of hidden space, signaling the start of the B-phase. This change enables the network to distinguish between A and B segments. Once in the B-phase, the network follows a deterministic path through hidden space that encodes a countdown, allowing it to correctly predict exactly twice the number of B’s that followed the A’s. After the final B, the hidden state returns close to the initial state, enabling the network to correctly anticipate the next A if a new sequence starts. The network's ability to output high-probability predictions for all non-initial B’s and the initial A shows that it has learned both the structure and transitions in the anb2n pattern.

4.

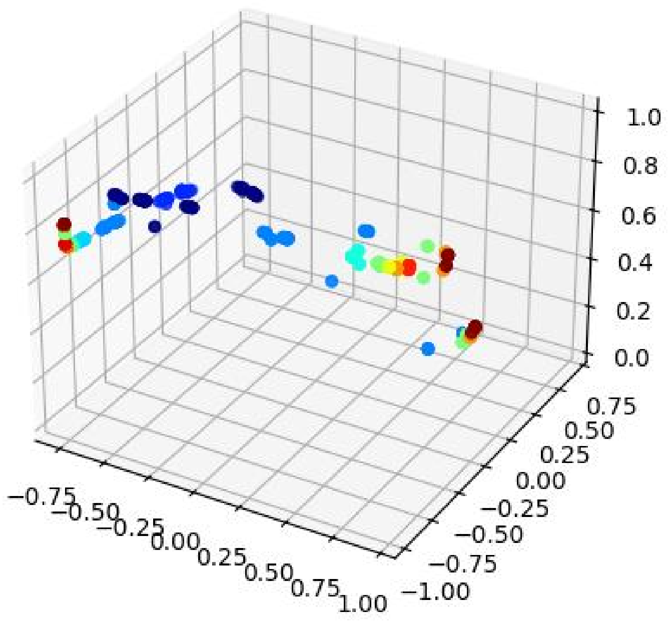




1.0
0.8
0.6
0.4
0.2
0.0
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6



5.The LSTM solves the anb2nc3n task by using its memory cells and hidden states to count how many A's, B's, and C's it has seen. During the A phase, the hidden state increases steadily, storing the number of A's. When the first B appears, the LSTM switches to a new pattern and begins a second phase. In this phase, it uses the memory from the A's to know when it has seen 2n B's. It then changes again to a third phase for the C's, where it must process 3n C's before returning to the start.

The LSTM's gates help control what information to keep or forget between phases. It keeps track of how far it is in each phase, and this allows it to confidently predict all the B's after the first one, all the C's, and the next A. Its internal states move through different regions depending on whether it's processing A's, B's, or C's. These changes help it know exactly when to switch phases and what to expect next.