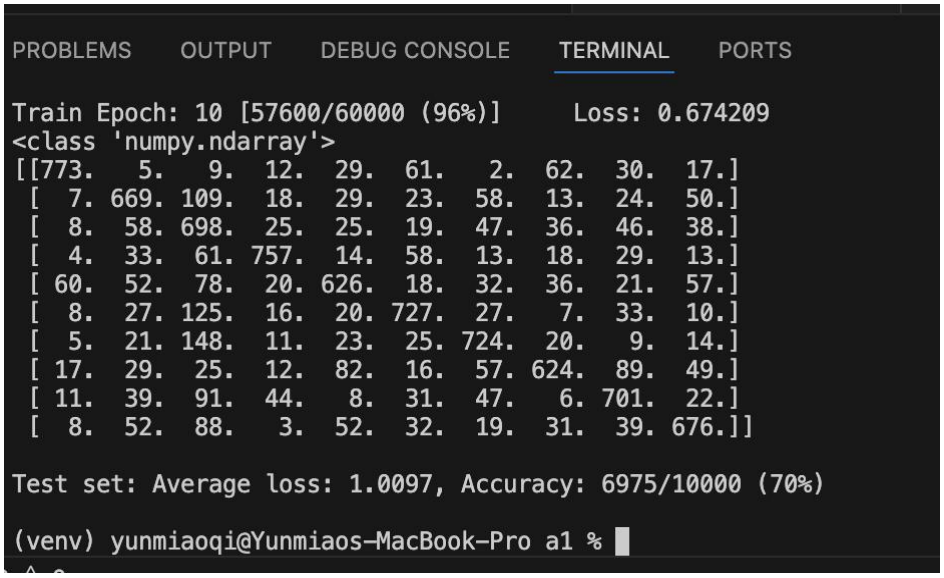
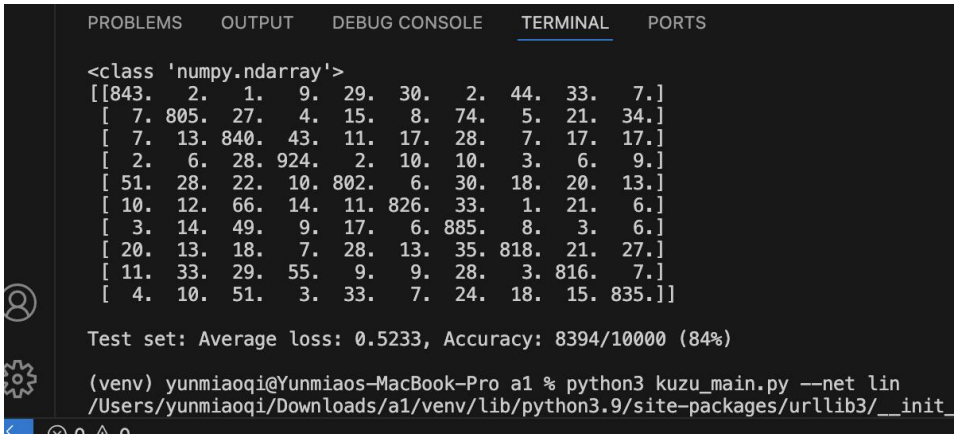
Assignment 1

Part 1: Japanese Character Recognition

1.

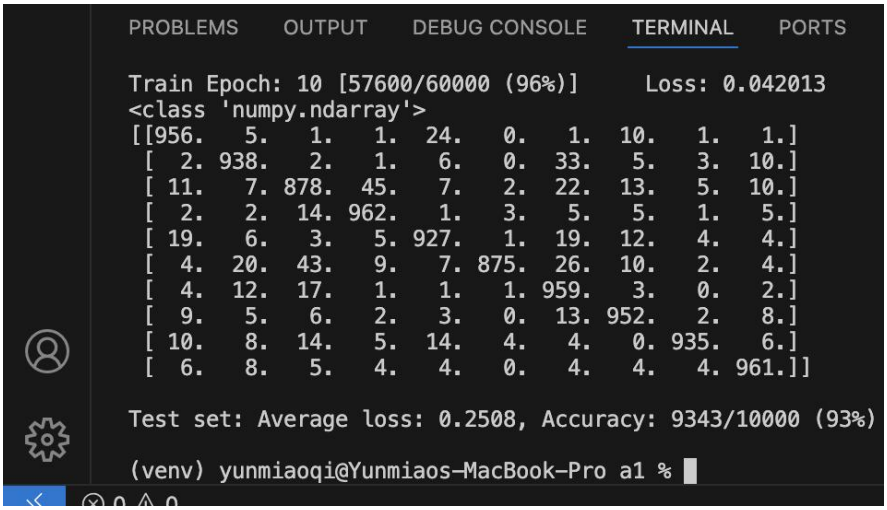


2.



fc1 weights: 78,400
fc1 bias: 100
fc2 weights: 1,000
fc2 bias: 10
Total parameters =100*784 +100+10*100+10 = 78,400 +100+1,000+10= **79,510**

3.



Total parameters=5*5*1*32+(5*5*32*64)+64+(1024*50)+50+(50*10)+10=103856

4.

a.The relative accuracy of the three models are 70%,84%,93%,respectively.

b.The number of independent parameters in each of the three models are 7840,79510,103856,respectively.

c.

From the confusion matrix of Netlin, “ki”(class1) is likely to be mistaken for “su”(class2); “ ha”(class5) is likely to be mistaken for “ya”(class7); “ma”(class6) is likely to be mistaken for “su”.

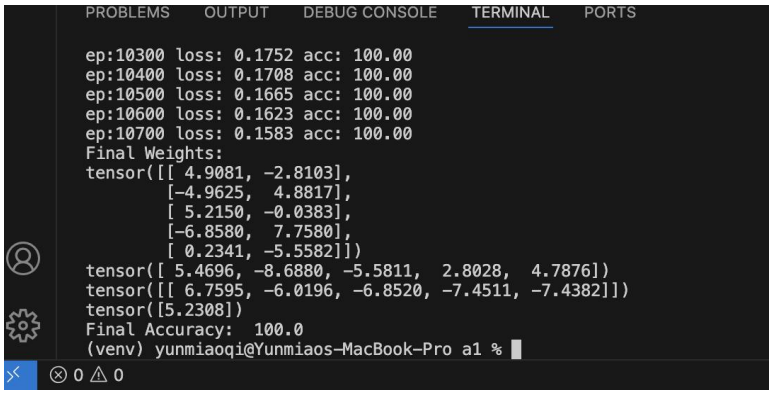
From the confusion matrix of Netfull, “su”(class2) is likely to be mistaken for “tsu”(class3).; “ha”(class5) is likely to be mistaken for “su”(class2).

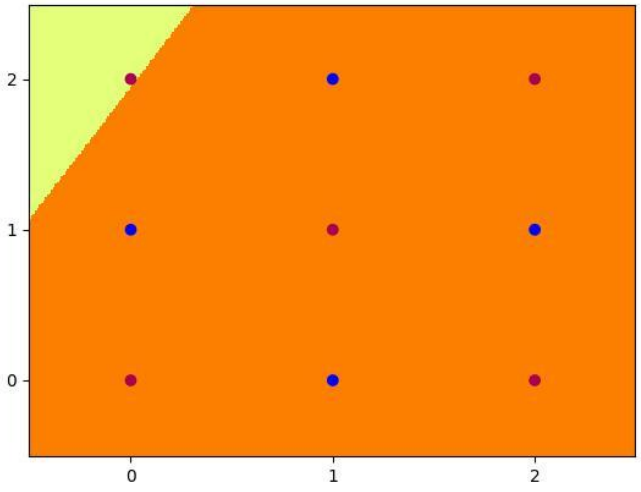
From the confusion matrix of Conv, “ha”(class5) is likely to be mistaken for “su”(class2); “su”(class2) is likely to be mistaken for “tsu”(class3).

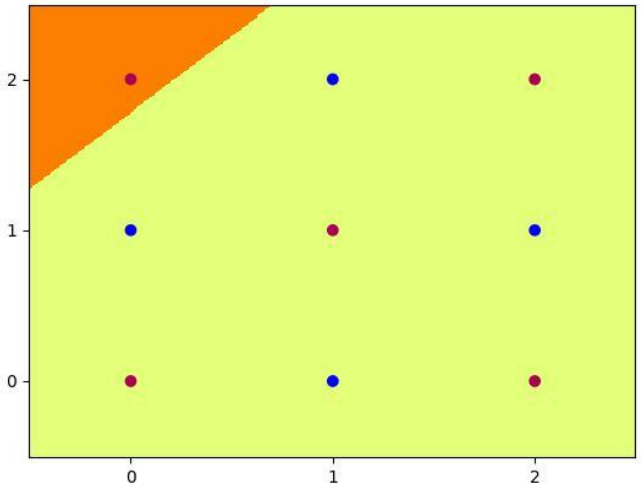
The reason might be those characters are visually similar or the handwriting are sloppy。

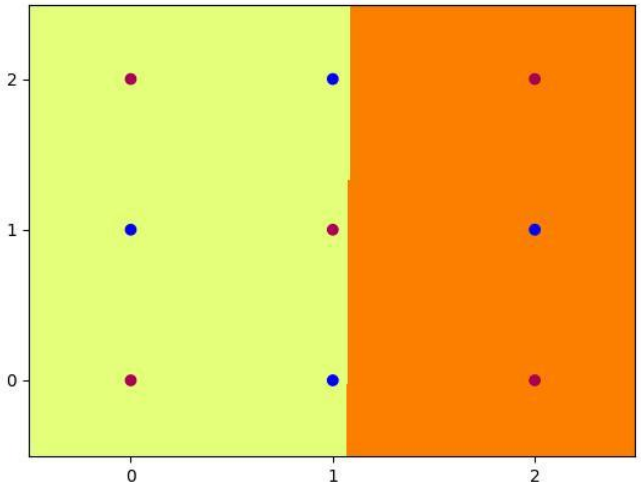
Part 2: Multi-Layer Perceptron

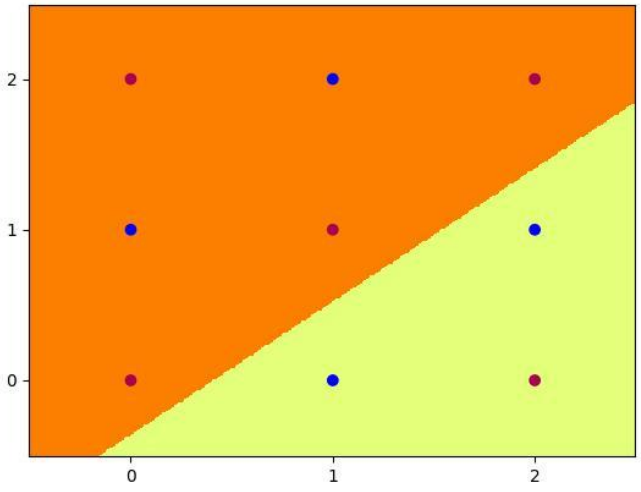
1.

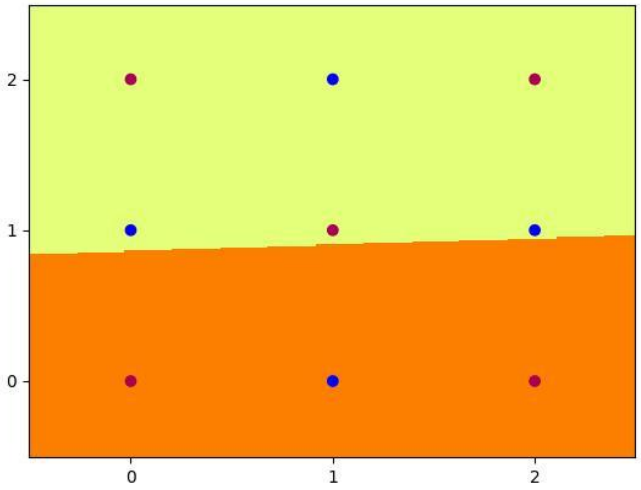


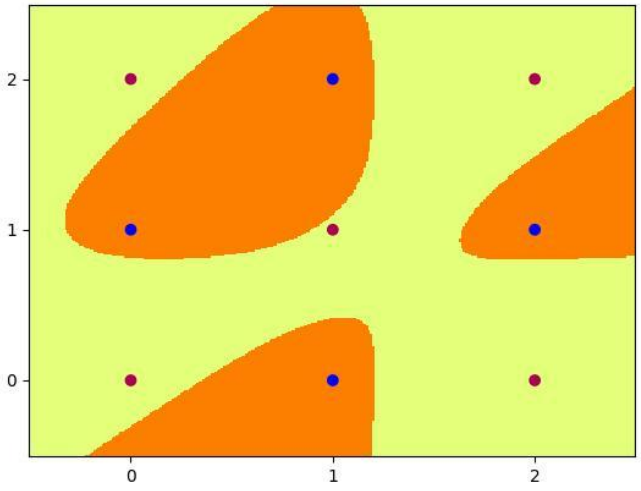

2
1
0
0
1
2


2
1
0
0
1
2


2
1
0
0
1
2

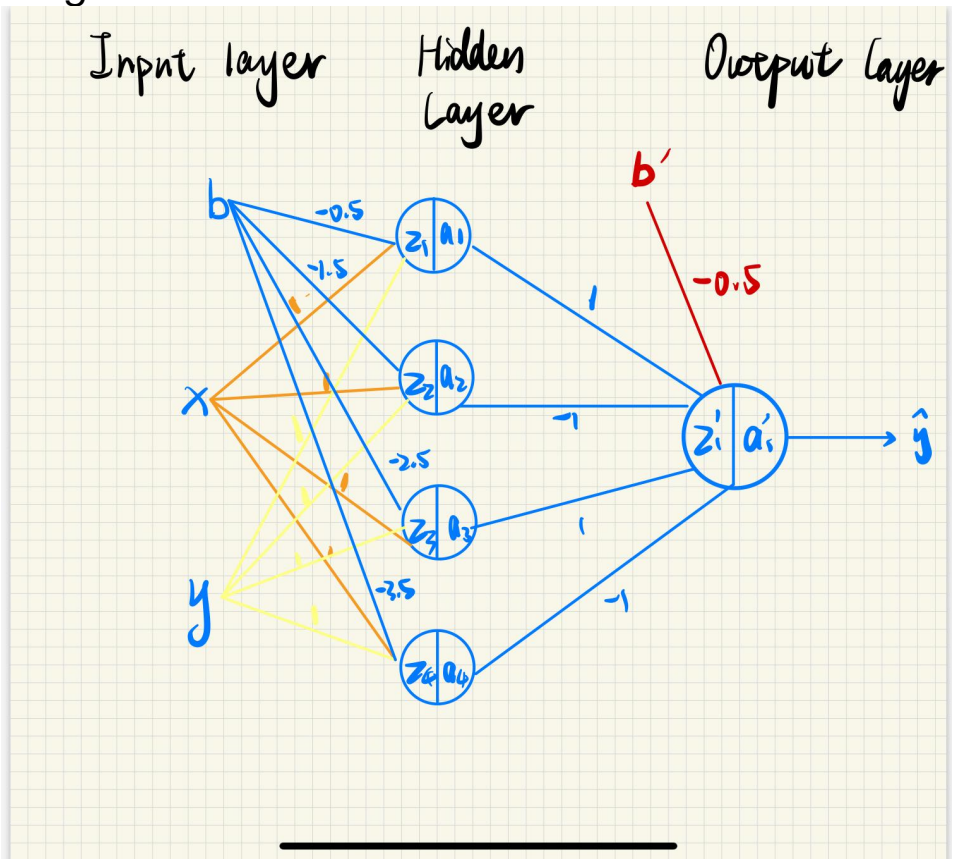






2.

Diagram of the network



Hidden layer:

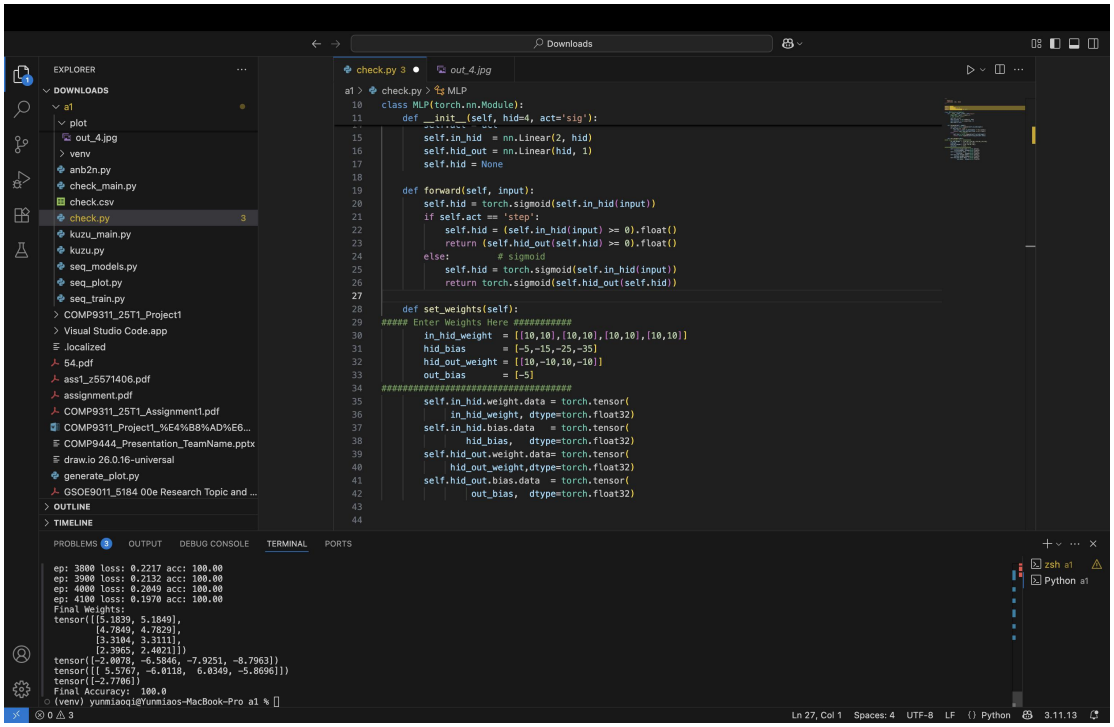
h1:if x+y-0.5>=0, then h1=1
h2:if x+y-1.5>=0, then h2=1
h3:if x+y-2.5>=0, then h3=1
h4:if x+y-3.5>=0, then h4=1
Output layer:
if h1 - h2 + h3 - h4 - 0.5 >= 0, then h0 =1

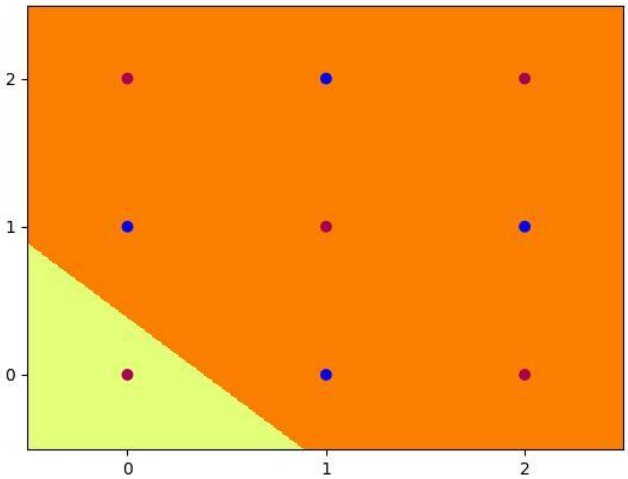
ACTIVATION TABLE:

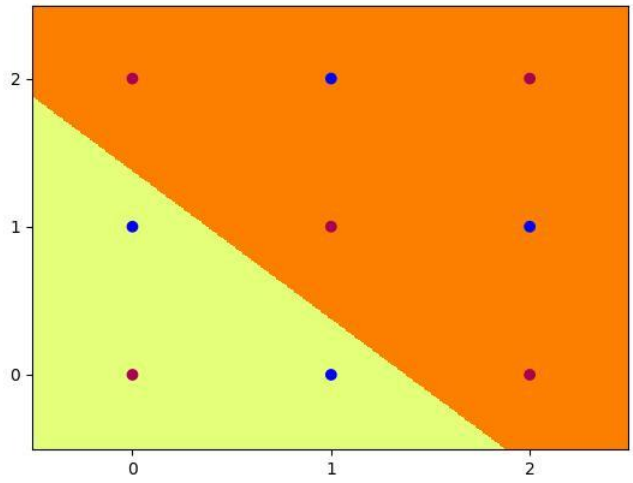
X	Y	T	h1	h2	h3	h4	output
0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1
0	2	0	1	1	0	0	0
1	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0
1	2	1	1	1	1	0	1
2	0	0	1	1	0	0	0

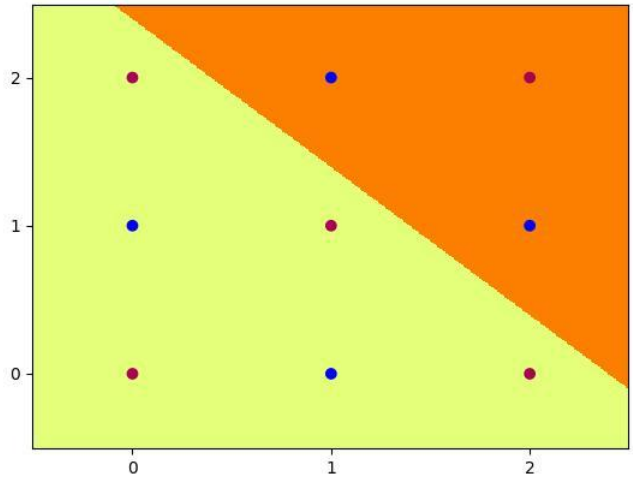
2	1	1	1	1	1	0	1
2	2	0	1	1	1	1	0

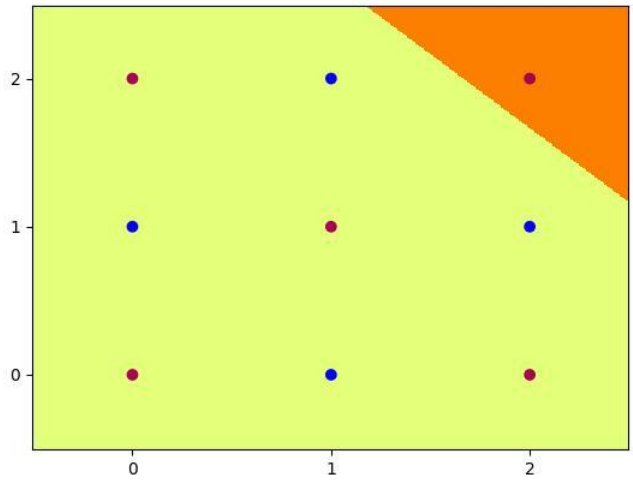
3.

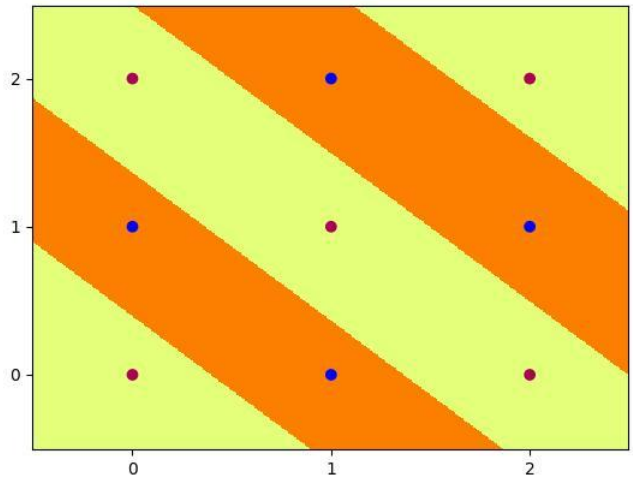





2
1
0
0
1
2

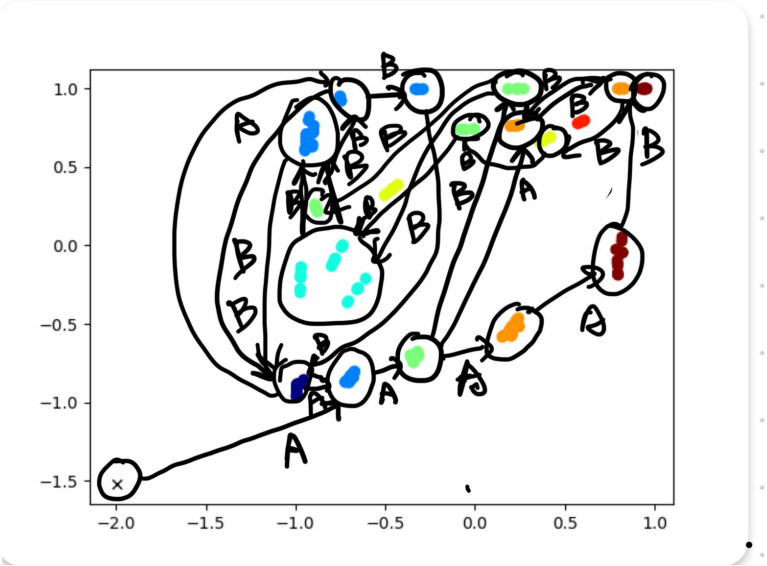

2
1
0
0
1
2


2
1
0
0
1
2

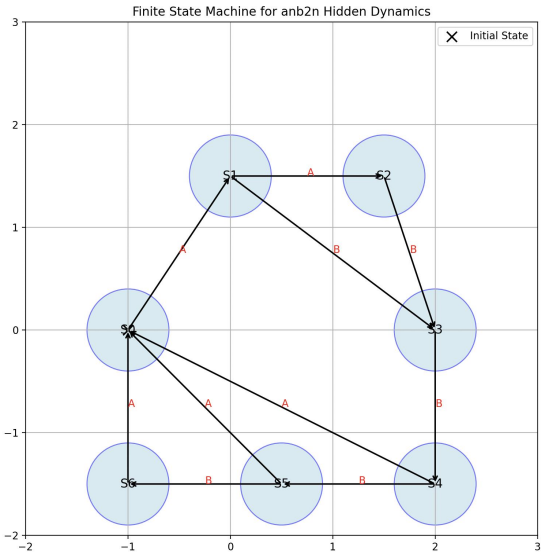


Part 3: Hidden Unit Dynamics for Recurrent Networks

1.



2.



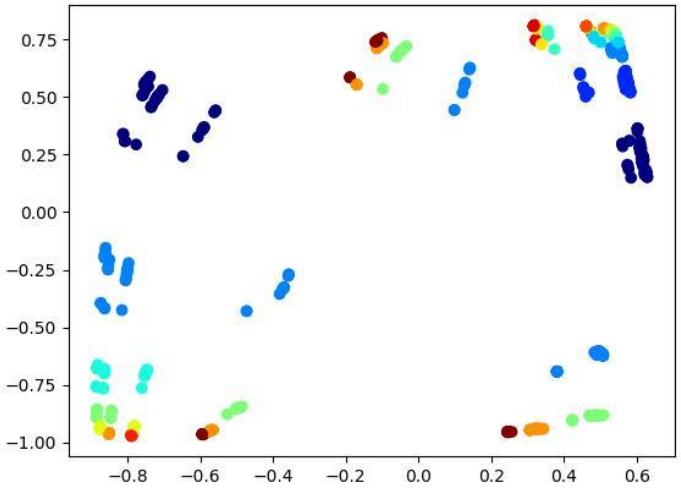
3.

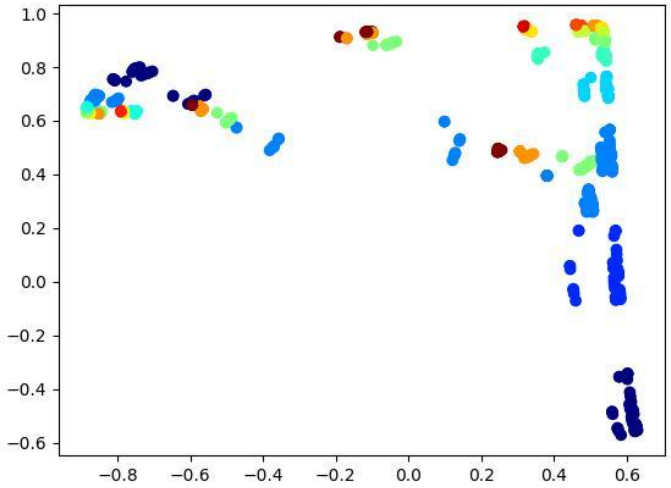
When the network starts reading A's, its hidden units change gradually to keep track of how many A's there are. Once it sees the first B, it "knows" how many B's should follow. From that point on, the hidden state stays in a region that signals it's in the B-part of the sequence, so it can confidently predict all the remaining B's.

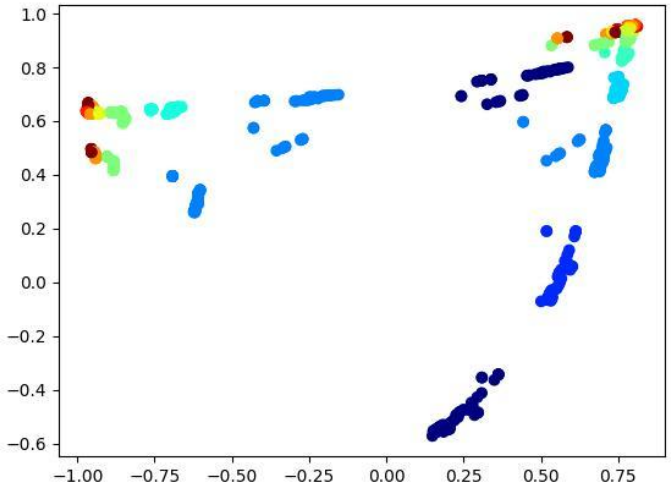
After finishing the B's, the hidden units shift into a state that signals the sequence has ended. This helps the network correctly expect a new A to start the next sequence.

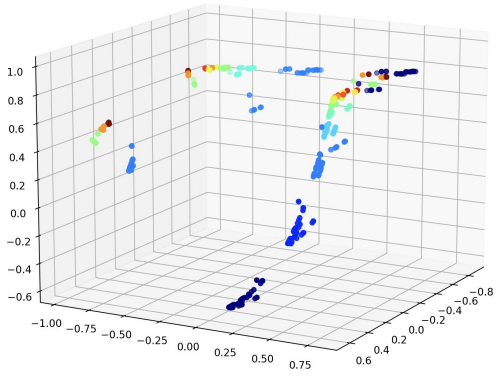
So the hidden activations act like a memory, counting A's and making sure the right number of B's — and then the next A , appear in the right order.

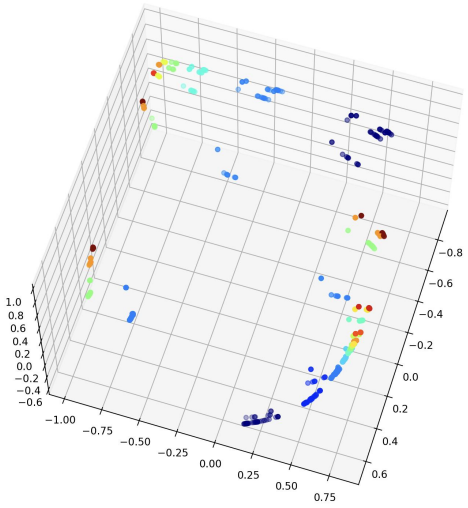
4.

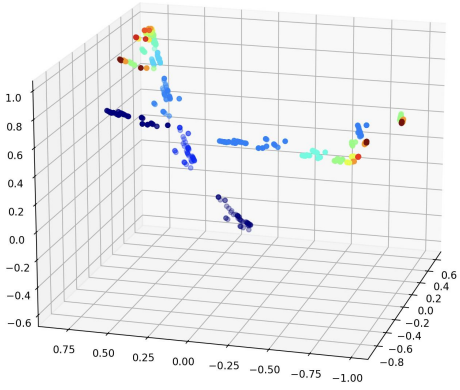



1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

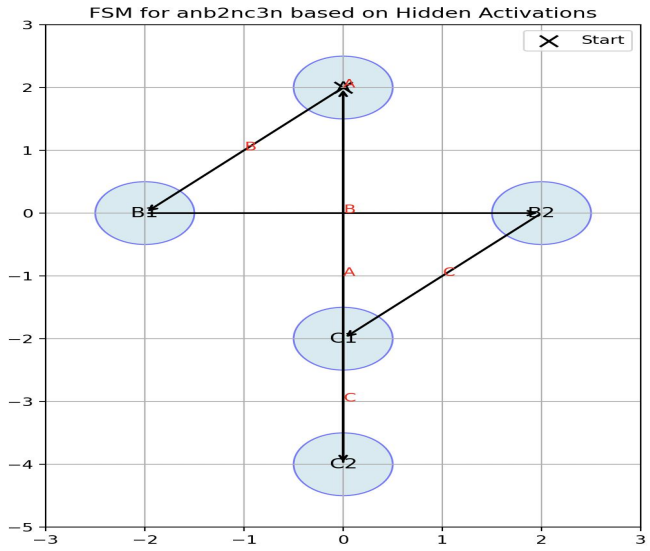

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75


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


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

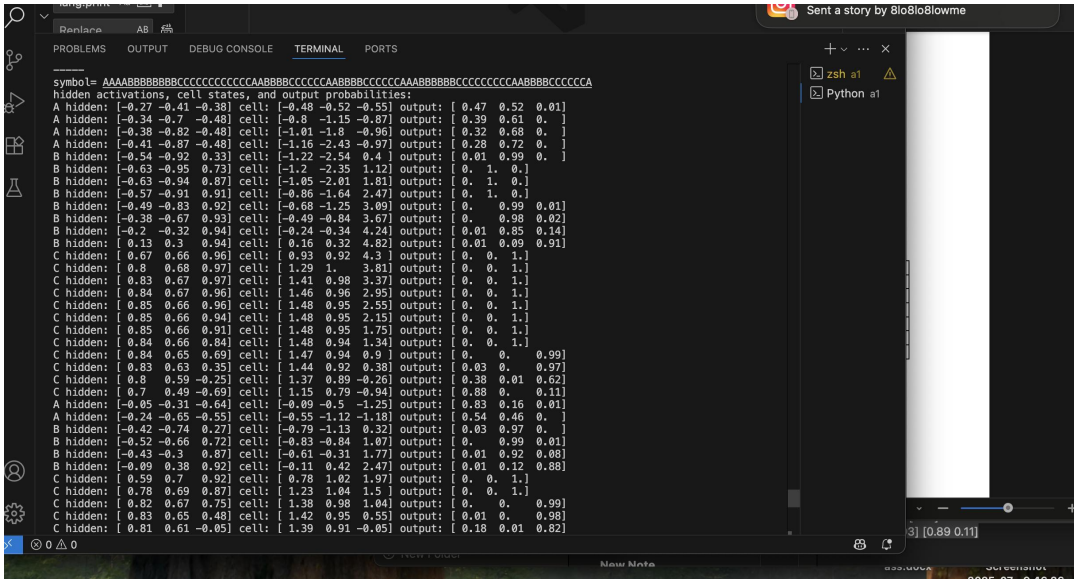

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

5.



stage	forget gate	input gate	output gate
Reading A	mostly open	open	open
First B appears	open	partially open	partially open
Counting B's	open	low	open
Switching to C	open	open	open
Counting c	open	low	open
End	may close	open	open

In the anb2nc3n task, the LSTM use its cell state to keep track of how many A's it has seen. As it reads each A, the input gate lets new information flow in, and the cell state grows, effectively counting. When it switches to B's, the cell state holds onto that count instead of forgetting it. This helps the LSTM know exactly how many B's to produce. After finishing the B's, the network transitions to predicting C's, and again relies on the stored count while scaling it to produce three times as many C's as A's. The hidden state shifts into different regions for A's, B's, and C's, acting like separate "modes" the network uses for each part of the sequence. Overall, the gates work together to decide when to save, update, or reveal this memory, letting the LSTM handle the nested structure of the language smoothly and predict the correct sequence lengths.



According to the result shown in the image,the network clearly differentiates among A, B, and C regions. For example, while reading A's, the hidden units remain negative, like [-0.27, -0.41, -0.38], and the cell values accumulate increasingly negative numbers, from [-0.48, -0.52, -0.55] up to [-1.16, -2.43, -0.97]. This shows the LSTM is counting how many A's have been seen.

When the sequence shifts to B's, the hidden state changes significantly. The third hidden value turns positive, e.g. [-0.54, -0.92, 0.33], and the cell's third dimension grows quickly from 0.40 to values like 1.81 and 2.47, reflecting the increasing count of B's being tracked. The network's output shifts decisively toward B with probabilities like [0.01, 0.99, 0.0].

During the C section, the hidden state becomes strongly positive, for example [0.67, 0.66, 0.96], and the cell state initially retains high values, such as [0.93, 0.92, 4.30]. As the sequence continues, these cell values gradually decrease (down to [1.15, 0.79, -0.94]), indicating the network counting down how many C's to produce. Meanwhile, the output probabilities shift solidly to C, with values like [0.0, 0.0, 1.0].

Overall, the LSTM uses the cell state as a long-term memory to keep track of how many A's it has read, helping to control the production of the correct number of B's and C's. The hidden state transitions between regions to signal which symbol type is currently being processed.