Part1 linear function

```
Total number of parameters: 7850
```

```
[[762.   6.   9.  14.  30.  65.   2.  63.  31.  18.]
 [  7. 668. 106.  19.  29.  21.  58.  13.  26.  53.]
 [  7.  61. 689.  27.  26.  20.  47.  39.  46.  38.]
 [  5.  37.  58. 758.  14.  55.  14.  18.  29.  12.]
 [ 62.  53.  76.  19. 623.  21.  32.  35.  20.  59.]
 [  8.  28. 122.  17.  20. 728.  26.   8.  33.  10.]
 [  5.  21. 144.  10.  27.  24. 722.  21.  10.  16.]
 [ 18.  29.  27.  10.  81.  18.  53. 626.  89.  49.]
 [  9.  37.  95.  40.   7.  30.  45.   7. 710.  20.]
 [  7.  52.  81.   3.  54.  33.  19.  31.  39. 681.]]

Test set: Average loss: 1.0090, Accuracy: 6967/10000 (70%)
```

2 layer nerwork

```
Total trainable parameters in full network: 109386
[[864.   5.   1.   5.  29.  25.   1.  32.  29.   9.]
 [  4. 821.  40.   4.  16.   5.  57.   3.  19.  31.]
 [  9.  20. 851.  27.   5.  17.  19.  16.  22.  14.]
 [  2.   9.  34. 926.   1.   9.   3.   2.   5.   9.]
 [ 34.  24.  25.   5. 810.   5.  35.  21.  19.  22.]
 [  8.  11.  92.   7.   9. 836.  17.   2.   8.  10.]
 [  3.  10.  47.   5.  10.   2. 902.   8.   3.  10.]
 [ 21.   7.  31.   3.  12.   4.  30. 840.  16.  36.]
 [ 11.  31.  31.  31.   4.   7.  21.   3. 854.   7.]
 [  2.   9.  49.   4.  22.   1.  21.   7.  12. 873.]]

Test set: Average loss: 0.4660, Accuracy: 8577/10000 (86%)
```

Neueral network

```
MAIN FUNCTION STARTED
Total trainable parameters in conv network: 421834
[[972.   4.   2.   1.  13.   0.   0.   7.   0.   1.]
 [  0. 932.   5.   1.  12.   2.  30.   3.   3.  12.]
 [  9.   4. 906.  33.   7.  12.  15.   4.   2.   8.]
 [  0.   0.  10. 981.   0.   3.   3.   2.   0.   1.]
 [ 12.   4.   2.  14. 944.   4.   5.   1.  11.   3.]
 [  1.   4.  24.   9.   4. 946.   8.   0.   2.   2.]
 [  2.   2.   6.   1.   6.   2. 980.   1.   0.   0.]
 [  5.   4.   7.   1.   7.   2.   7. 941.  10.  16.]
 [  2.   9.   7.   8.   9.   2.   4.   0. 959.   0.]
 [  8.   3.   3.   2.   8.   3.   4.   8.   9. 952.]]

Test set: Average loss: 0.1808, Accuracy: 9513/10000 (95%)
```

1.4.a In the 1st model using logistic regression, the accuracy is about 70%,

In the 2nd model using 2 layer network, the accuracy is about 86 percent

The 3rd model is based on convolutional network, which is about 95% accuracy.

From the comparison we can see that For logistic regression, the accuracy is the lowest

Then 2 layer network the convolutional network share the highest amount of accuracy, which is 95 percent

1.4.b

The no of independent parameters increase as model become more complex.

In linear function. The model has single layer, leading smallest number of parameter which is 7850

Then 2 layer network,109836

And then neueral network which is 421834

1.4.c in the 1st method of linear function the word that misclassified the most is は (ha) and す (su) matched by matrix[5][2], which indicate the number 122

In the 2nd method of 2 layer network, this is about matrix[5][2] = 92 which is は (ha) and す (su) as well.

In the 3rd model of neueral network, this pic is about matrix[2][3] = 33, which means that the most difference is す (su) and つ (tsu).

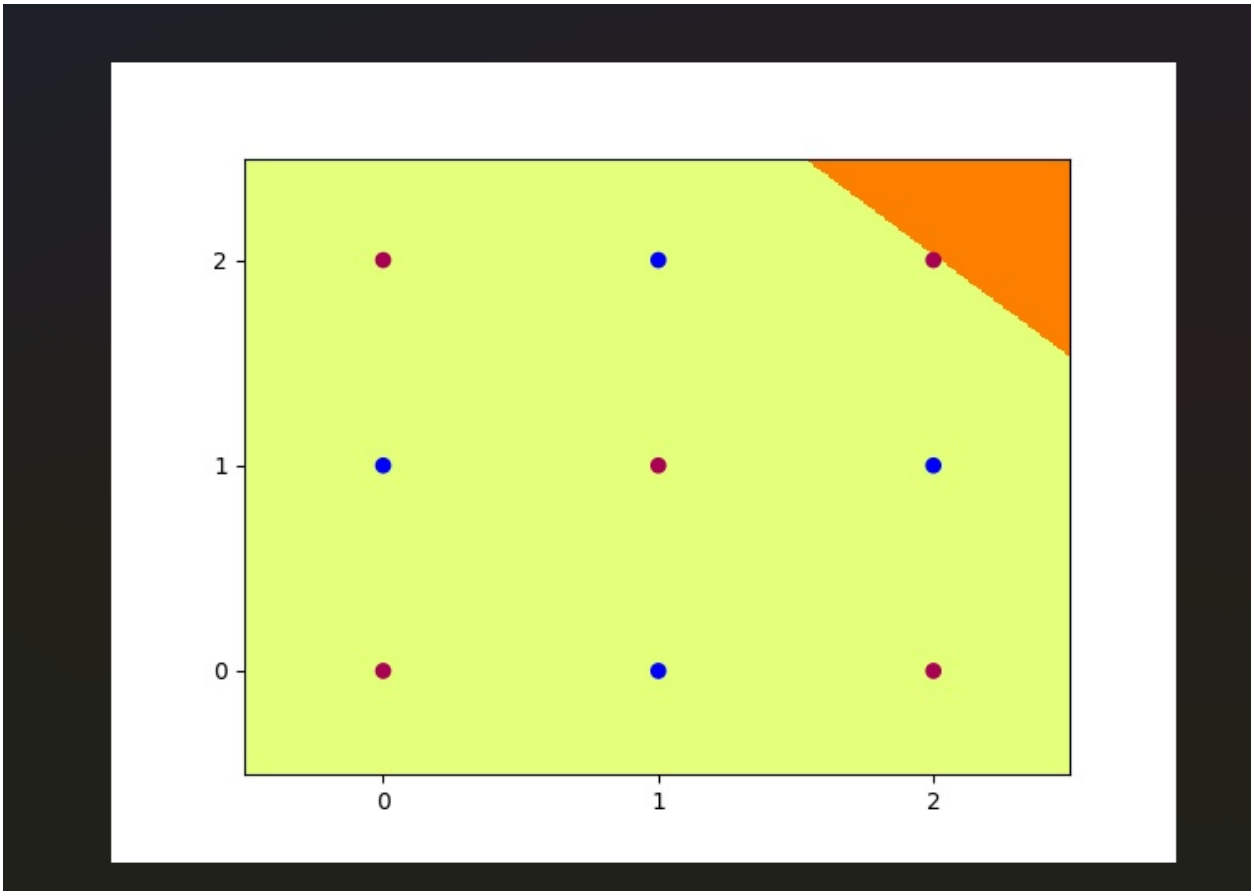
As the reason why it is happened, The primary reason behind this is that Both **す (su)** and **つ (tsu)** are curved characters and Have a main loop with additional small marks.

Also, the kmnist resolution is low, and have limited ability to distinguish very find-gained stroke differences at this resolution.

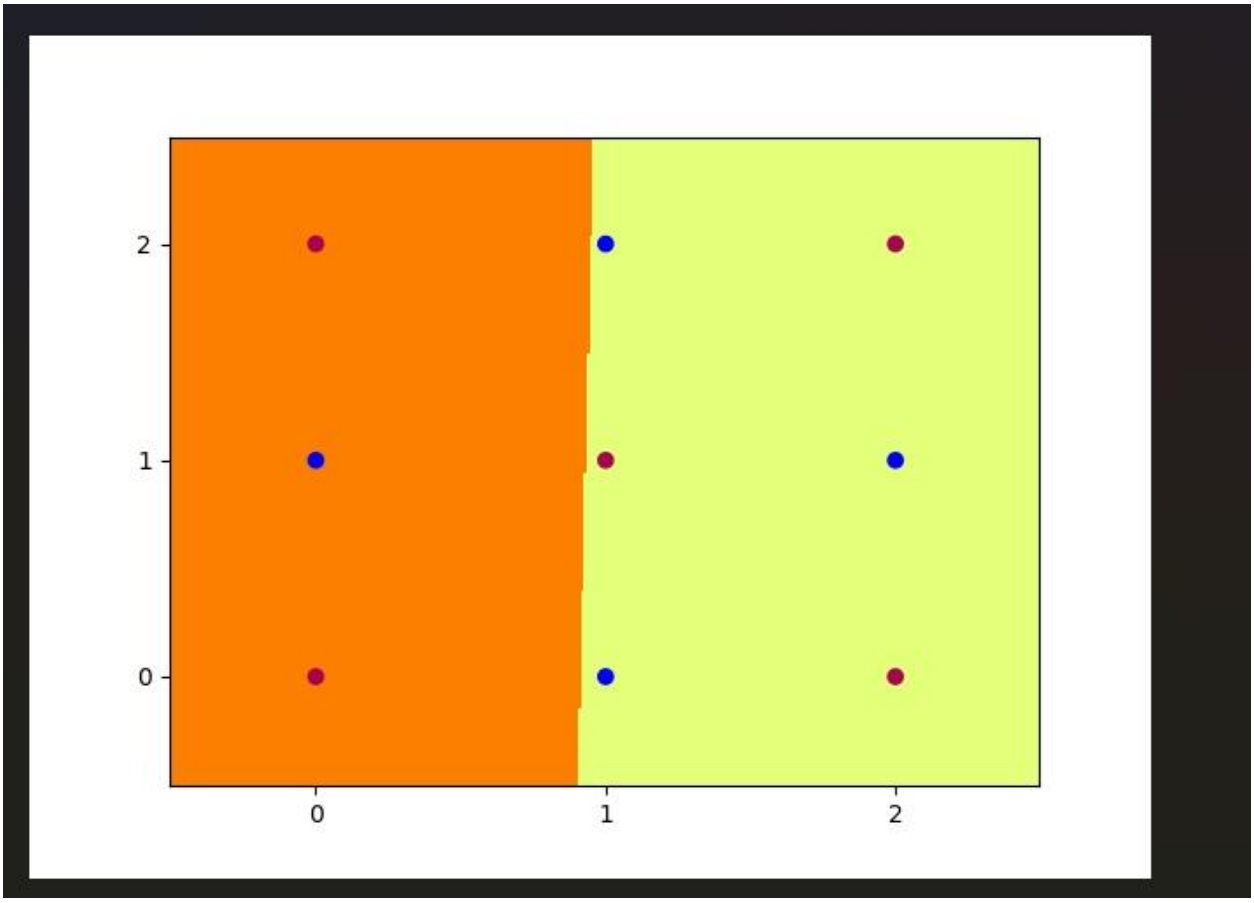
In addition, the handwriting style also influence the final result.

Part 2

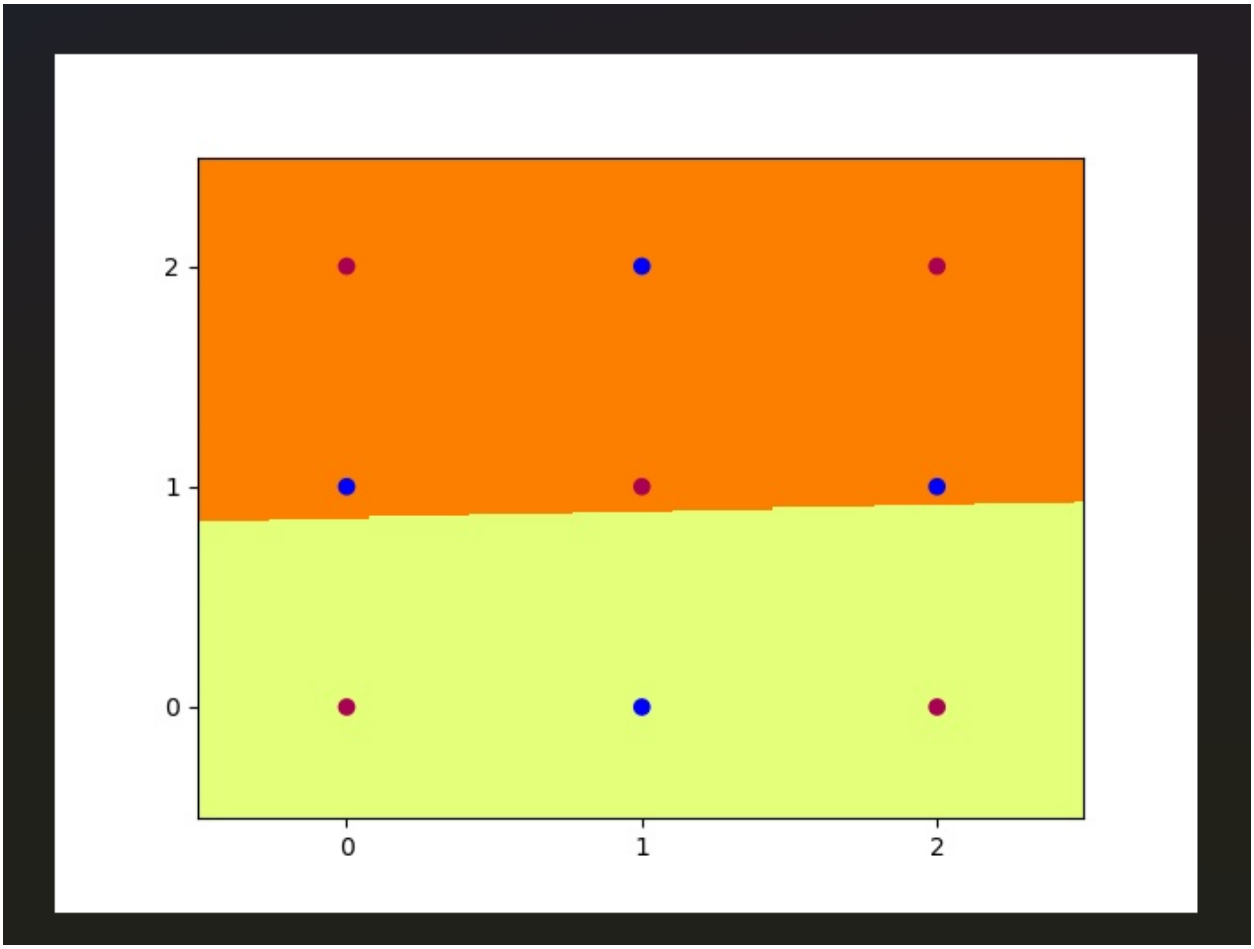
2.1



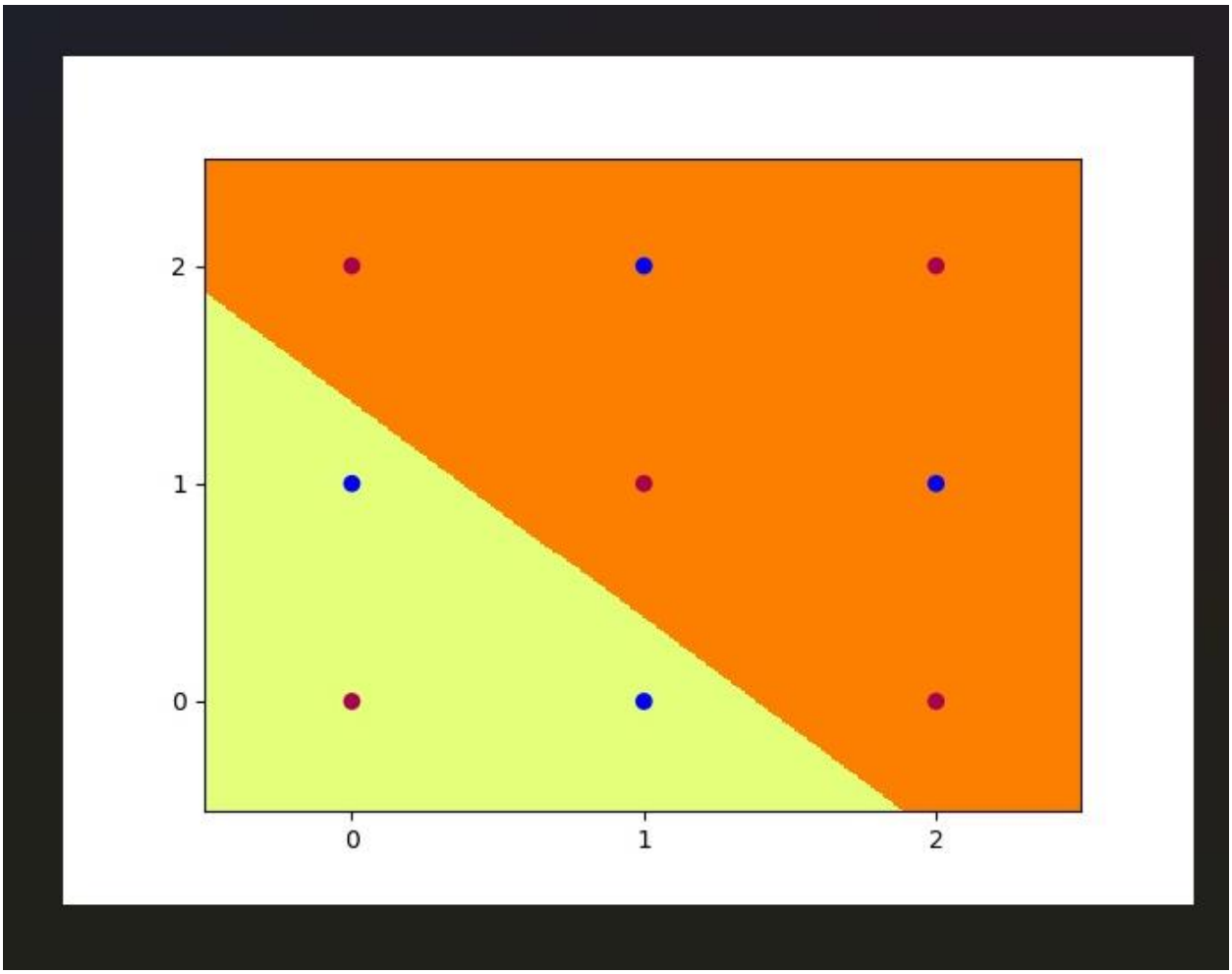
hid_5.0



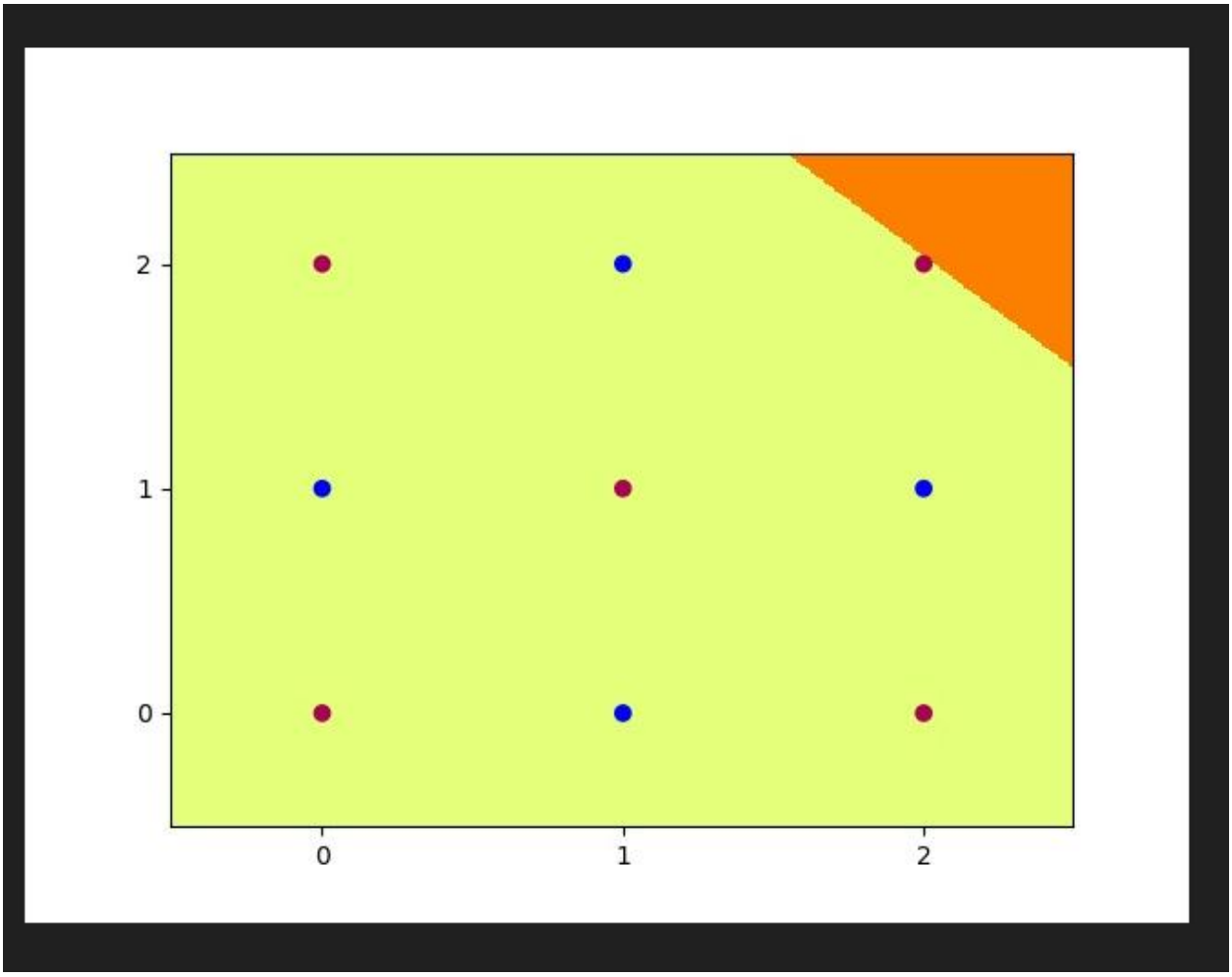
hid_5.1



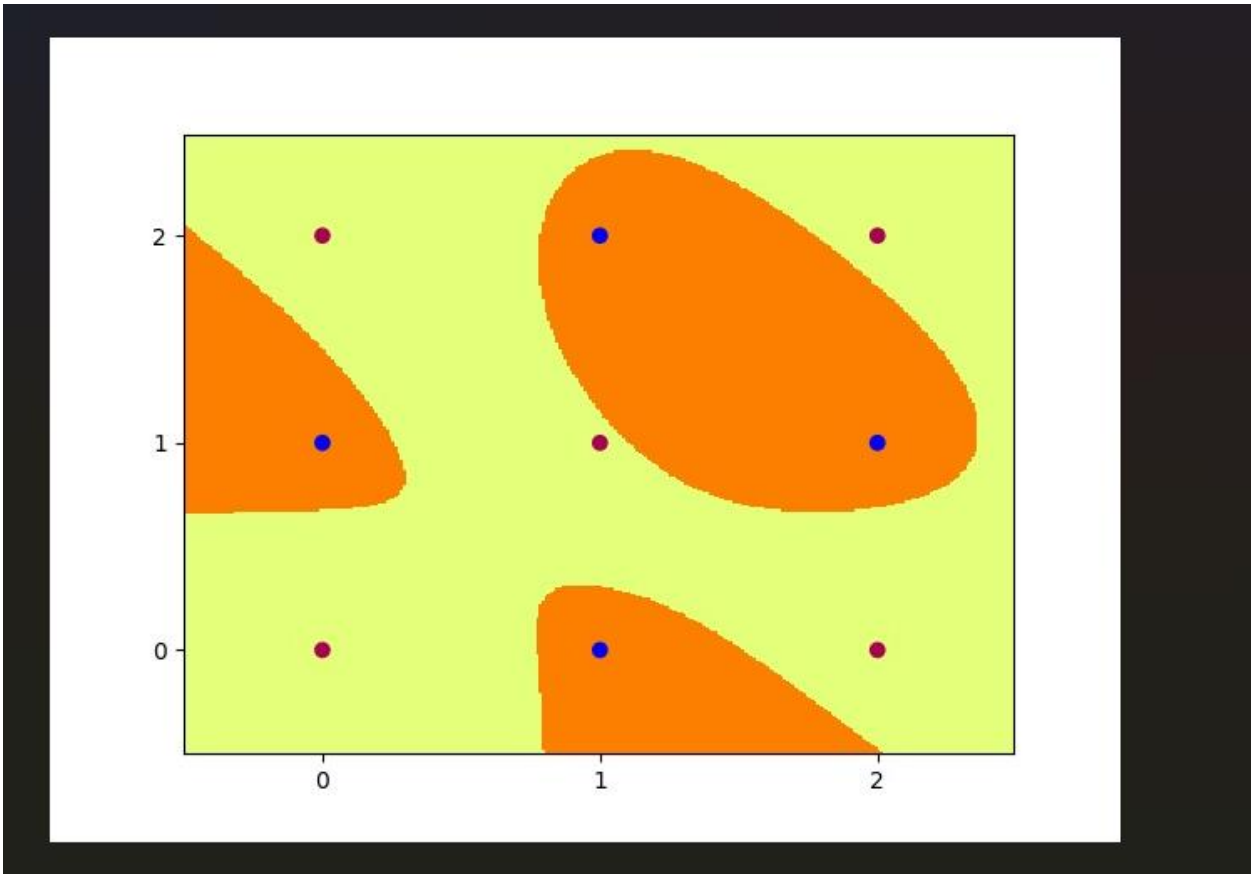
hid_5.2



hid_5.3



Hid_5.4



Out

2.2

According to the requirement, I am trying to linearly cut the point in order to find the area , in here we get 4 equations

H1 = x + y ≥ 1.5

H2 = -x + y ≥ 0.5

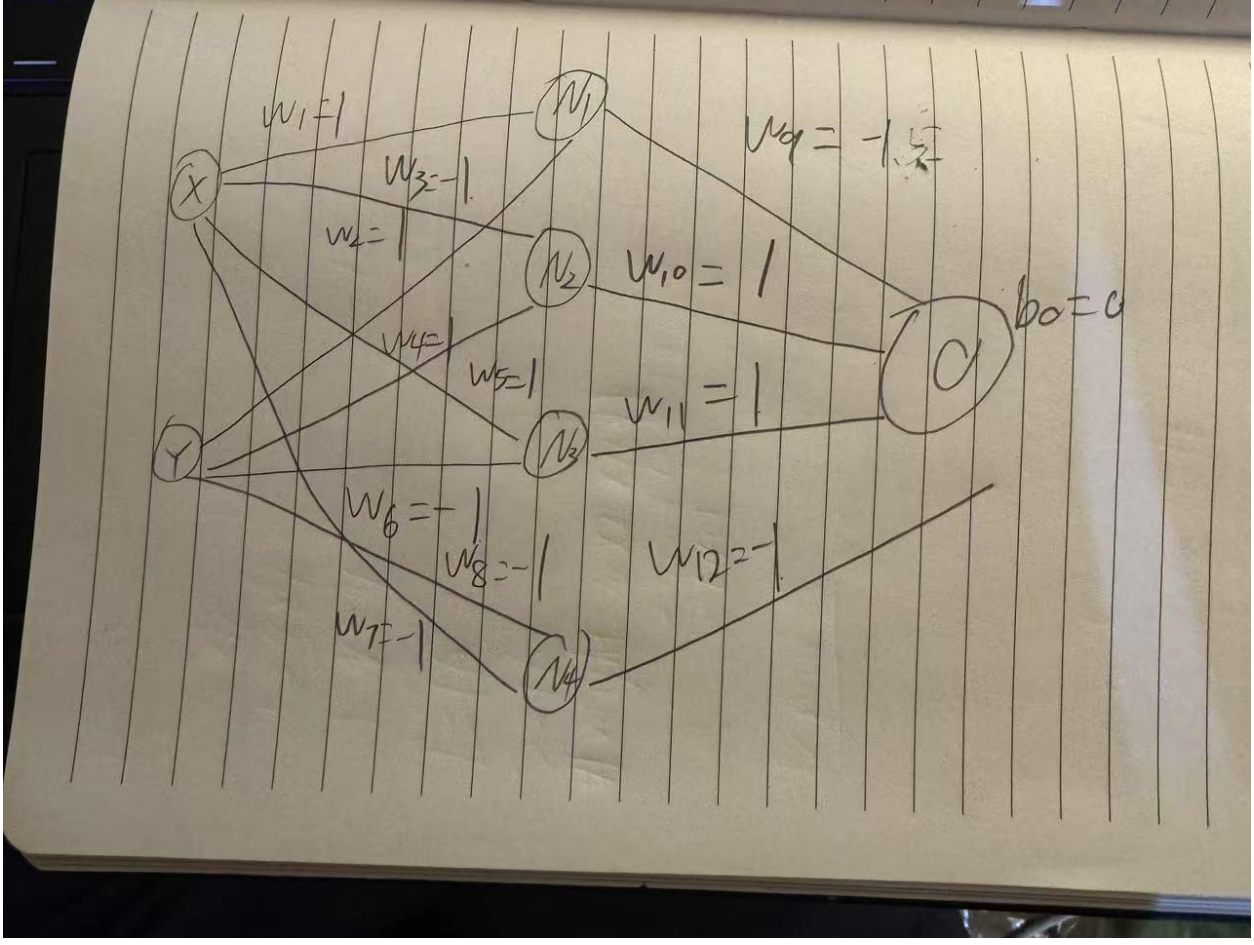
H3 = x - y ≥ 0.5

H4 = -x - y ≥ -2.5

In terms of the output, we can based on 4 function, output(-h1+h2+h3-h4)

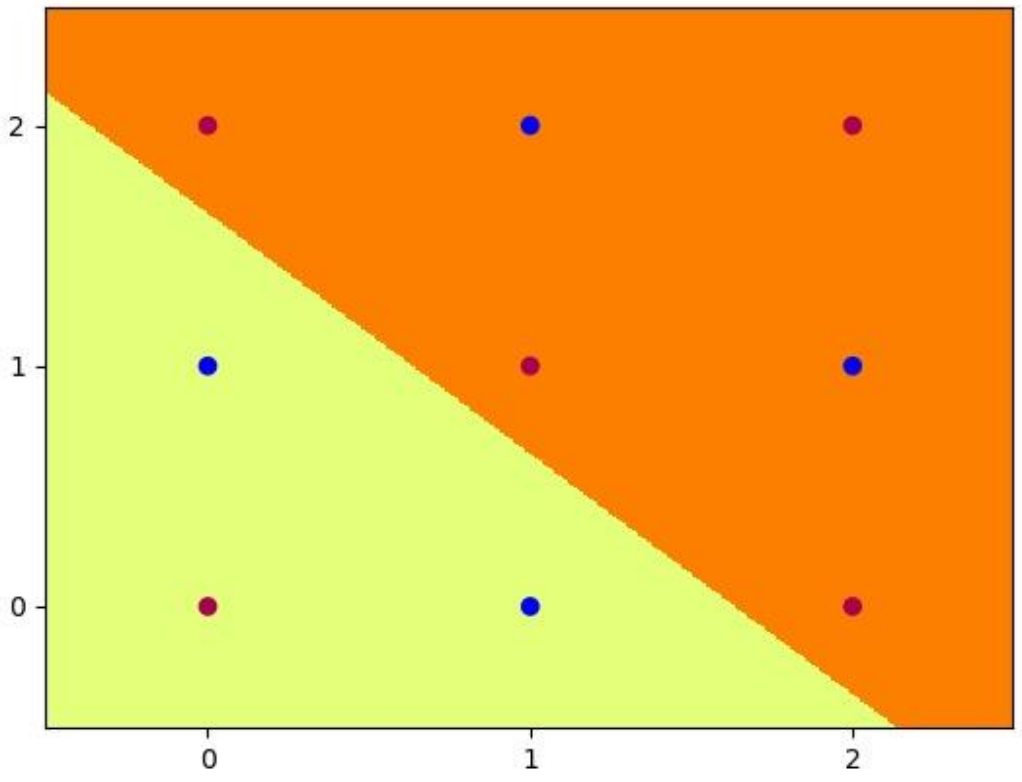
Based on what I got , We can form the table based on that

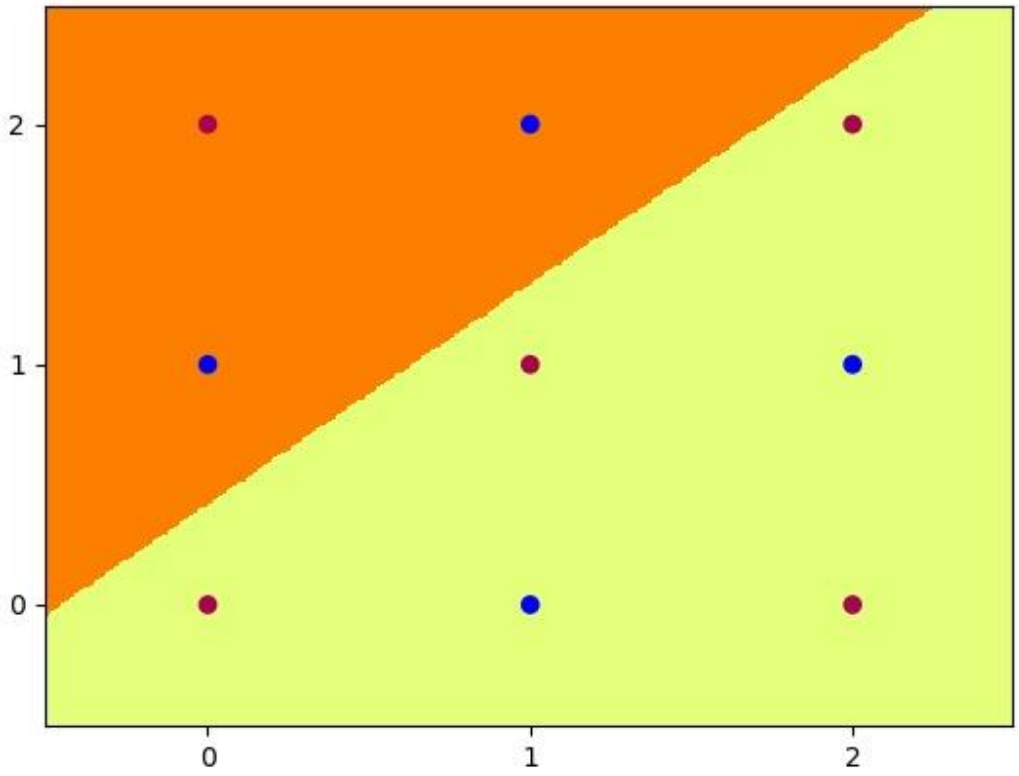
	x	y	H1	H2	H3	H4	output
1	0	0	0	0	0	1	0
2	2	0	1	1	0	1	0
3	0	2	1	0	1	1	0
4	2	2	1	0	0	0	0
5	1	1	1	0	0	1	0
6	1	0	0	1	0	1	1
7	0	1	0	0	1	1	1
8	1	2	1	0	1	0	1
9	2	1	1	1	0	0	1

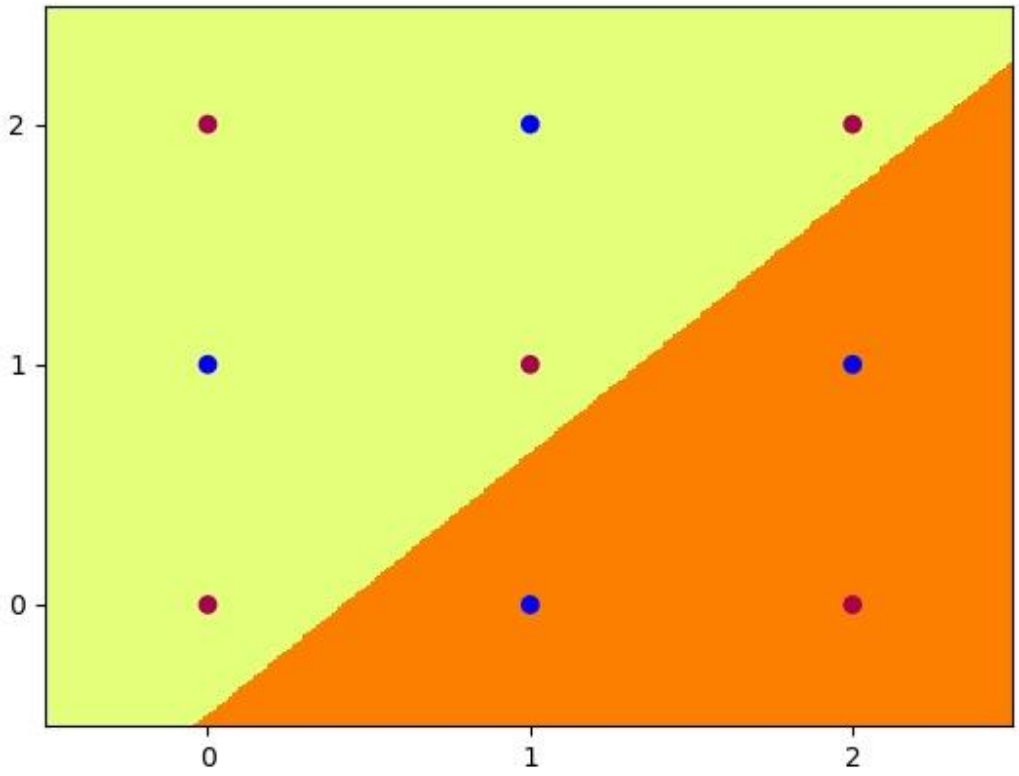

X
Y
N_1
N_2
N_3
N_4
C
$w_1=1$
$w_2=1$
$w_3=-1$
$w_4=1$
$w_5=1$
$w_6=-1$
$w_7=-1$
$w_8=-1$
$w_9=-1.5$
$w_{10}=1$
$w_{11}=1$
$w_{12}=-1$
$b_0=$

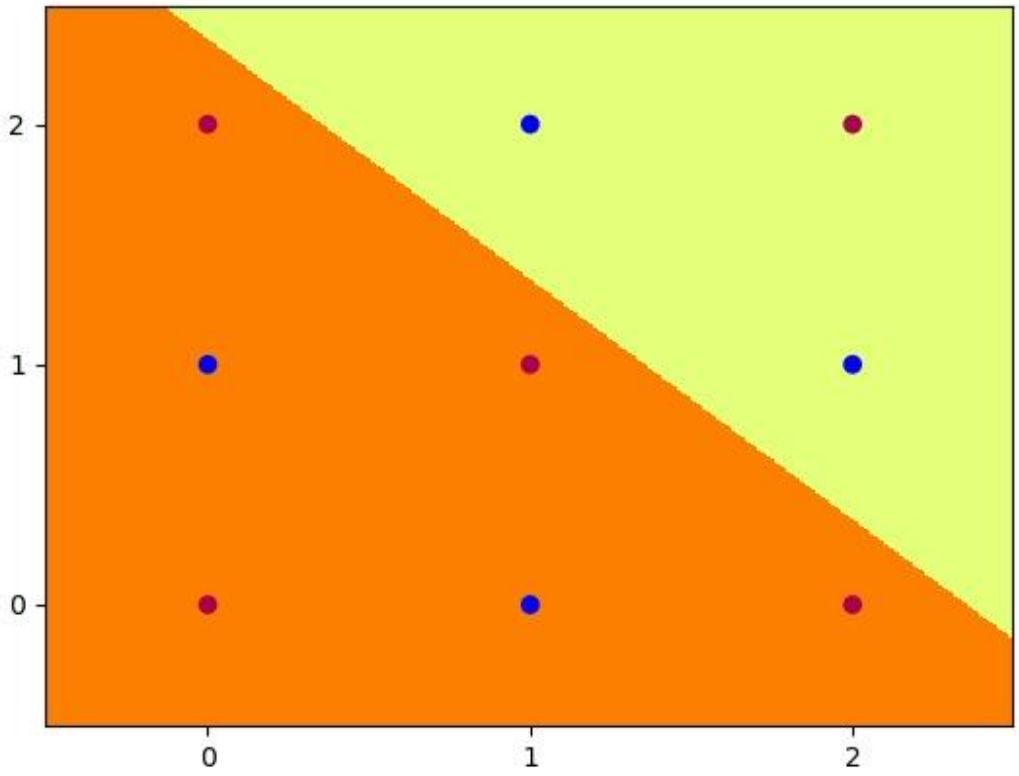
```python
def set_weights(self):
    ##### Enter Weights Here ############
    in_hid_weight = [
        [1, 1],     # h1: x + y ≥ 1.5
        [-1, 1],    # h2: -x + y ≥ 0.5
        [1, -1],    # h3: x - y ≥ 0.5
        [-1, -1]    # h4: -x - y ≥ -2.5
    ]
    hid_bias = [-1.5, -0.5, -0.5, 2.5]

    hid_out_weight = [[-1, 1, 1, -1]]  # Combine activations to select blue region
    out_bias = [0]                  # Clean threshold                    # Output layer bias

#########################################
```

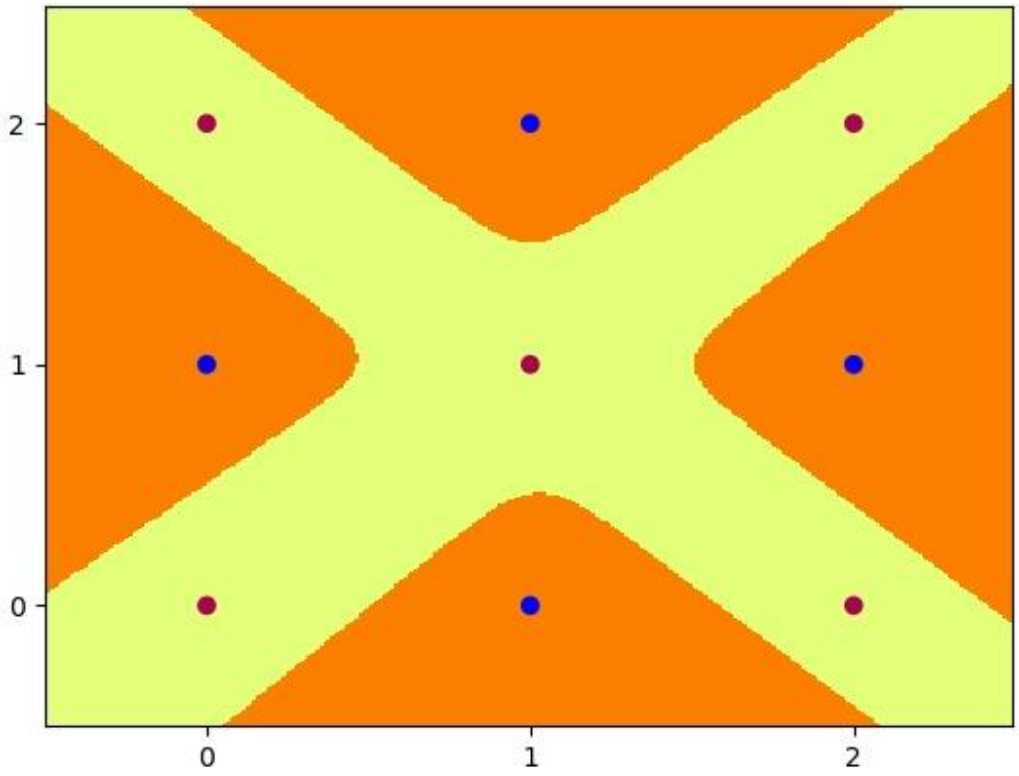
2.3




2
1
0
0
1
2


2
1
0
0
1
2


2
1
0
0
1
2



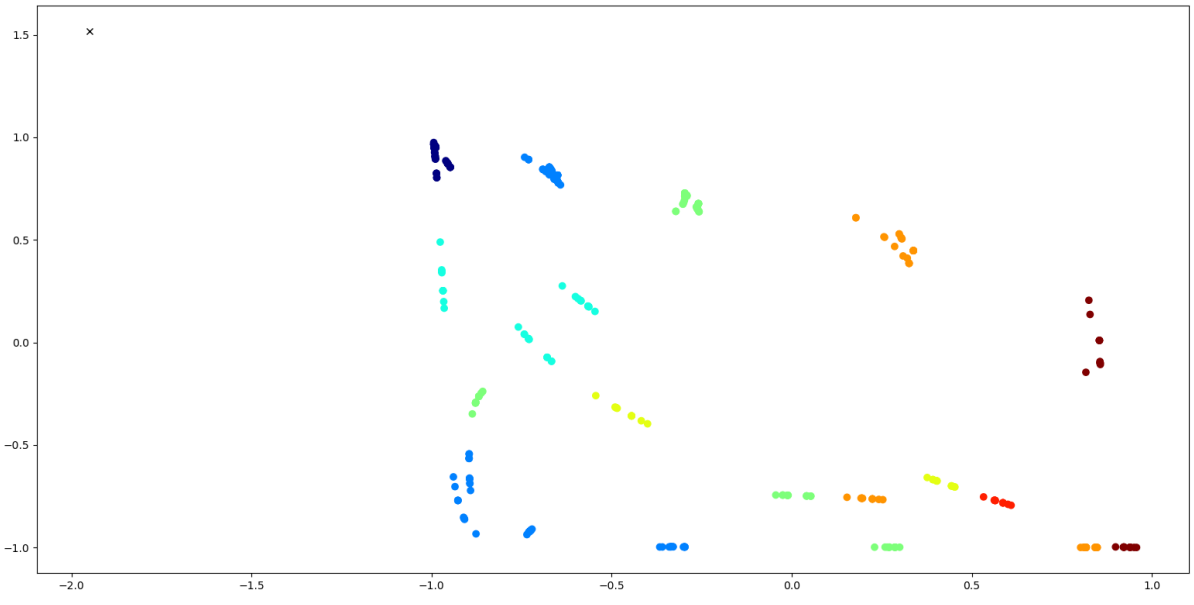

```
Final Weights:
tensor([[  9.4292,   9.4292],
        [ -9.6026,  10.4612],
        [ 10.4612,  -9.6026],
        [-10.3750, -10.3750]])
tensor([-15.4449,  -4.3828,  -4.3828,  24.3832])
tensor([[-8.9572, 11.0553, 11.0553, -8.9495]])
tensor([1.0540])
Final Accuracy:  100.0
```

Part 3

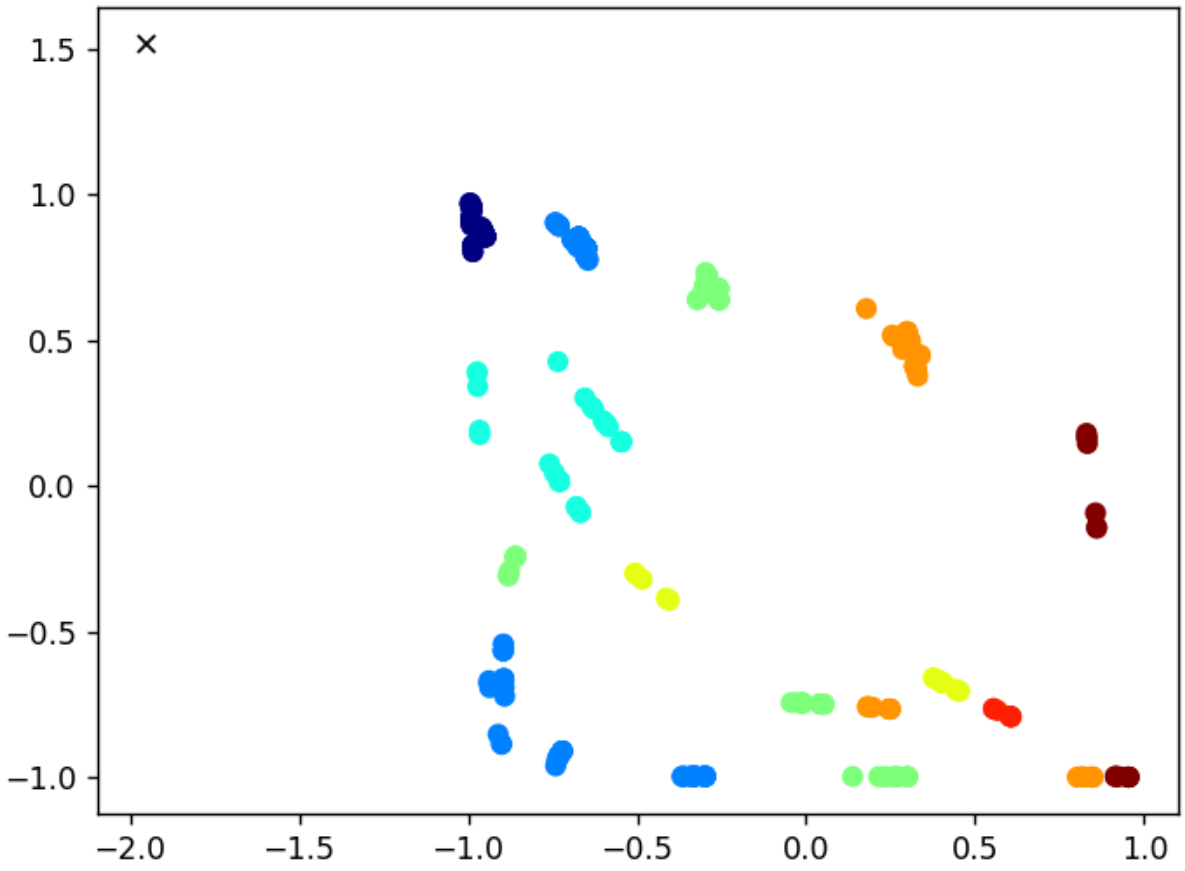
```
hidden activations and output probabilities:
A [-0.65  0.82] [ 0.8  0.2]
A [-0.26  0.68] [ 0.65  0.35]
B [ 0.3 -1. ] [ 0.  1.]
B [-0.55  0.15] [ 0.04  0.96]
B [-0.72 -0.91] [ 0.  1.]
B [-0.99  0.95] [ 0.9  0.1]
A [-0.65  0.79] [ 0.78  0.22]
A [-0.29  0.71] [ 0.7  0.3]
A [ 0.32  0.41] [ 0.27  0.73]
B [ 0.82 -1.  ] [ 0.  1.]
B [ 0.4  -0.68] [ 0.  1.]
B [-0.01 -0.75] [ 0.  1.]
B [-0.73  0.01] [ 0.01  0.99]
B [-0.89 -0.66] [ 0.  1.]
B [-0.99  0.89] [ 0.86  0.14]
A [-0.69  0.84] [ 0.83  0.17]
A [-0.3   0.67] [ 0.64  0.36]
A [ 0.25  0.52] [ 0.43  0.57]
B [ 0.82 -1.  ] [ 0.  1.]
B [ 0.4  -0.67] [ 0.  1.]
B [-0.01 -0.75] [ 0.  1.]
B [-0.73  0.02] [ 0.01  0.99]
B [-0.89 -0.67] [ 0.  1.]
B [-0.99  0.9 ] [ 0.86  0.14]
A [-0.69  0.84] [ 0.83  0.17]
A [-0.3   0.68] [ 0.64  0.36]
A [ 0.26  0.51] [ 0.42  0.58]
A [ 0.83 -0.02] [ 0.02  0.98]
B [ 0.93 -1.  ] [ 0.  1.]
B [ 0.57 -0.77] [ 0.  1.]
B [ 0.2  -0.76] [ 0.  1.]
B [-0.47 -0.33] [ 0.  1.]
B [-0.87 -0.28] [ 0.  1.]
B [-0.97  0.32] [ 0.09  0.91]
B [-0.92 -0.83] [ 0.  1.]
B [-0.99  0.96] [ 0.91  0.09]
```

1.

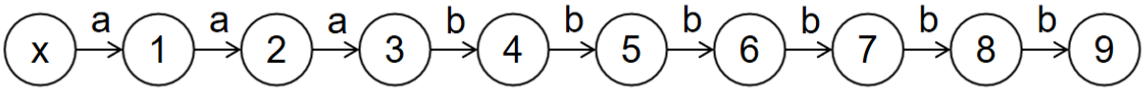
```
A [-0.69  0.84] [ 0.83  0.17]
A [-0.3   0.68] [ 0.64  0.36]
A [ 0.26  0.51] [ 0.42  0.58]
A [ 0.83 -0.02] [ 0.02  0.98]
B [ 0.93 -1.  ] [ 0.  1.]
B [ 0.57 -0.77] [ 0.  1.]
B [ 0.2  -0.76] [ 0.  1.]
B [-0.47 -0.33] [ 0.  1.]
B [-0.87 -0.28] [ 0.  1.]
B [-0.97  0.32] [ 0.09  0.91]
B [-0.92 -0.83] [ 0.  1.]
B [-0.99  0.96] [ 0.91  0.09]
A [-0.65  0.78] [ 0.76  0.24]
B [-0.33 -1.  ] [ 0.  1.]
B [-0.96  0.87] [ 0.84  0.16]
```



2,



2,

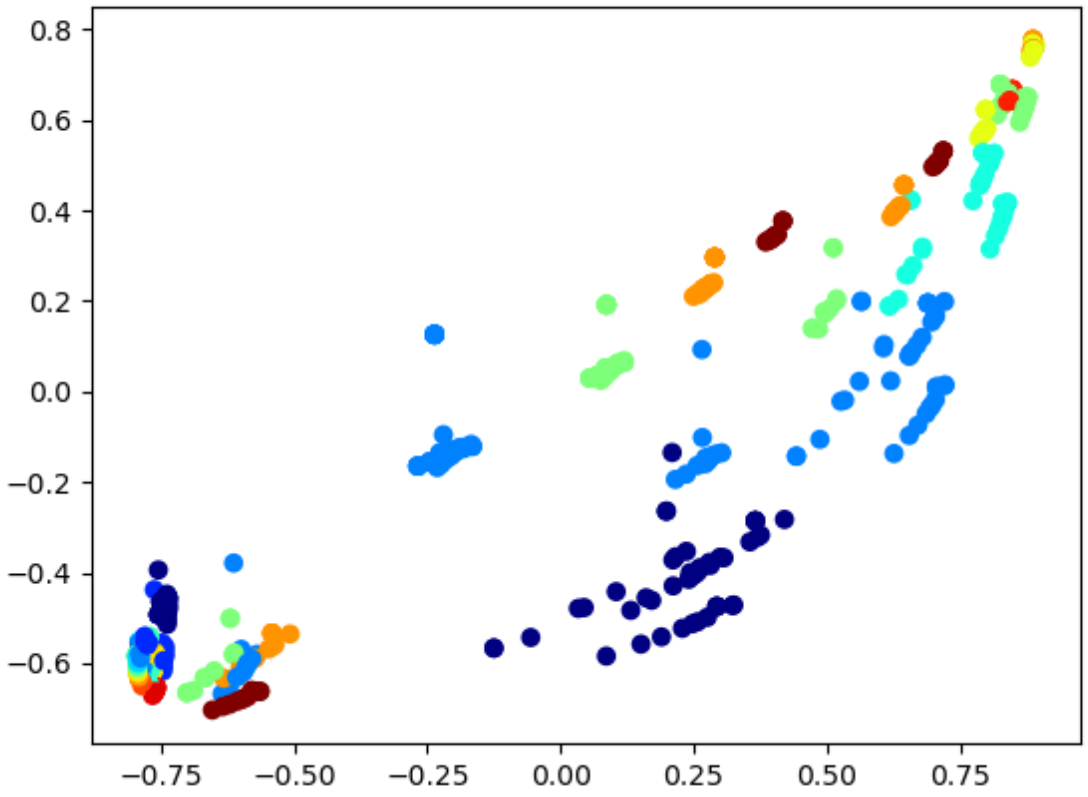


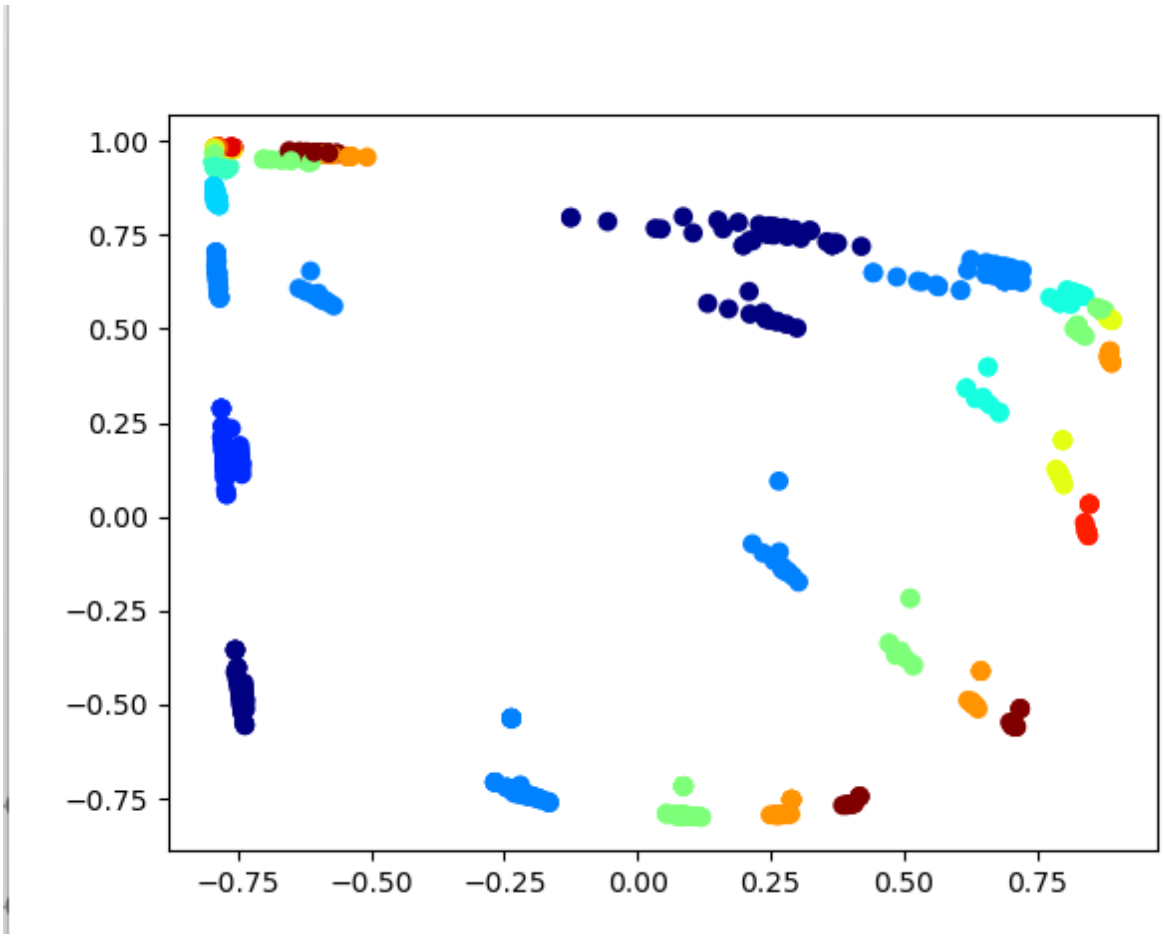
A finite state machine

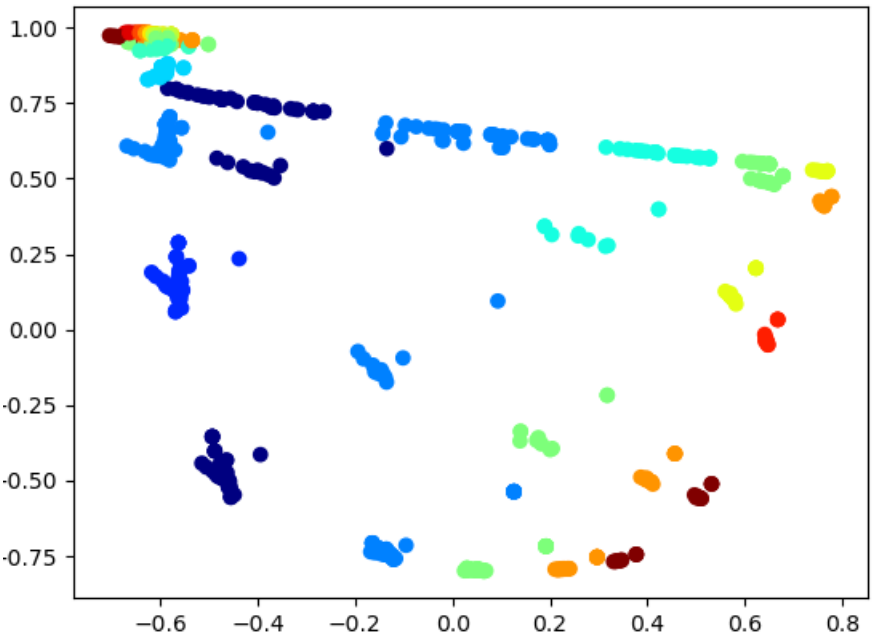
3,

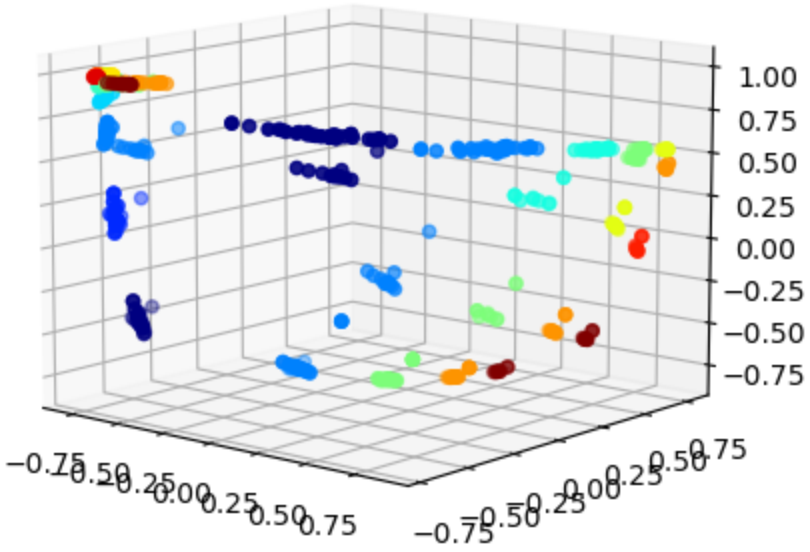
When the network successfully predict the 1st B then calculate the number of preceding As in the following process. In this task, the question require us to make sure B is exactly twice as the number of A.the network can correctly predict the entire B sequence. Until reash the last B, the the network detects that the B sequence is finished. Then the network knows to transition to the next sequence, then begin with a new A sequence.

4.




1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75


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



5,

LSTM use cell state (c_t) to count the occurrence of a b c over here. When processing A, LSTM imprement the internal state then swiching to b and c, it will compare the inpur pattern with the expected pattern using its memory.

In terms of the hidden states, they are reflecting how many a b and c have been proceed and the clustering reflect, it reflect distinct internal states for different stages if the input.