Part 1: Japanese Character Recognition

1.

```
[[766.   5.   8.  12.  30.  62.   2.  62.  32.  21.]
 [  7. 667. 107.  19.  29.  23.  59.  13.  26.  50.]
 [  9.  61. 690.  26.  28.  21.  48.  33.  47.  37.]
 [  3.  37.  56. 756.  15.  58.  15.  18.  30.  12.]
 [ 60.  52.  80.  20. 622.  20.  34.  36.  20.  56.]
 [  8.  29. 128.  17.  19. 724.  27.   7.  33.   8.]
 [  4.  22. 149.  10.  26.  24. 721.  20.  11.  13.]
 [ 17.  28.  27.  11.  81.  17.  55. 625.  89.  50.]
 [ 10.  37.  91.  42.   6.  32.  45.   7. 707.  23.]
 [  9.  52.  88.   3.  51.  30.  20.  30.  39. 678.]]
Test set: Average loss: 1.0087, Accuracy: 6956/10000 (70%)
```

2.

```
[[835.   7.   4.   7.  26.  33.   4.  43.  30.  11.]
 [  7. 784.  33.   5.  29.  17.  80.   3.  19.  23.]
 [  6.  18. 825.  35.  14.  20.  31.  11.  25.  15.]
 [  5.  13.  39. 888.   3.  17.   8.   8.  10.   9.]
 [ 58.  32.  30.   6. 754.  11.  41.  23.  19.  26.]
 [ 10.  10.  90.   8.  11. 823.  26.   3.  12.   7.]
 [  4.  15.  73.   8.  17.   8. 859.   5.   3.   8.]
 [ 18.  13.  16.   7.  38.  12.  49. 769.  45.  33.]
 [ 13.  23.  28.  43.   4.  14.  36.   5. 826.   8.]
 [  2.  27.  62.   6.  33.  13.  20.  18.  16. 803.]]
Test set: Average loss: 0.5899, Accuracy: 8166/10000 (82%)
```

3.

```
[[961.   5.   1.   1.  15.   1.   2.   6.   3.   5.]
 [  4. 931.   4.   1.   6.   0.  34.   4.   7.   9.]
 [ 10.  15. 862.  41.   9.   6.  21.  17.   7.  12.]
 [  2.   7.  14. 960.   2.   1.   3.   5.   1.   5.]
 [ 33.   5.   6.   5. 910.   0.  17.  11.   8.   5.]
 [  8.  14.  35.   6.   3. 899.  19.  11.   2.   3.]
 [  2.   8.  13.   3.   5.   1. 963.   4.   0.   1.]
 [  9.   7.   3.   1.   4.   2.  10. 943.   3.  18.]
 [  7.  15.   5.   8.   8.   3.   7.   3. 942.   2.]
 [  8.  11.   8.   5.   9.   1.   1.   5.   2. 950.]]
Test set: Average loss: 0.2375, Accuracy: 9321/10000 (93%)
```

4.

a.

The accuracy of NetLin is 70%.
The accuracy of NetFull is 82%.
The accuracy of NetConv is 93%.

b.

The number of independent parameters in of NetLin is 7850.
The number of independent parameters in of NetFull is 50890.
The number of independent parameters in of NetConv is 123234.

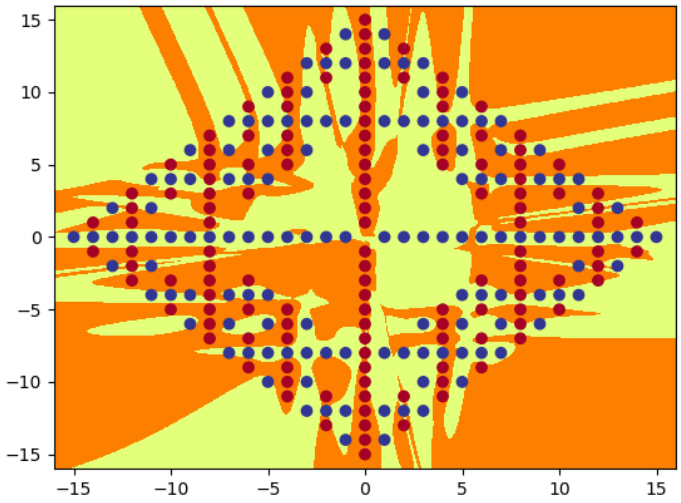
c.

For NetLin, the largest value which is not in diagonal is 149 in 7th row (ma) is likely to be mistaken for 2 (su). Because their writing styles are similar, so it is more difficult to distinguish. For NetFull and NetConv, the reason is the same.

Part2: Fractal Classification Task

1.
See code in frac.py

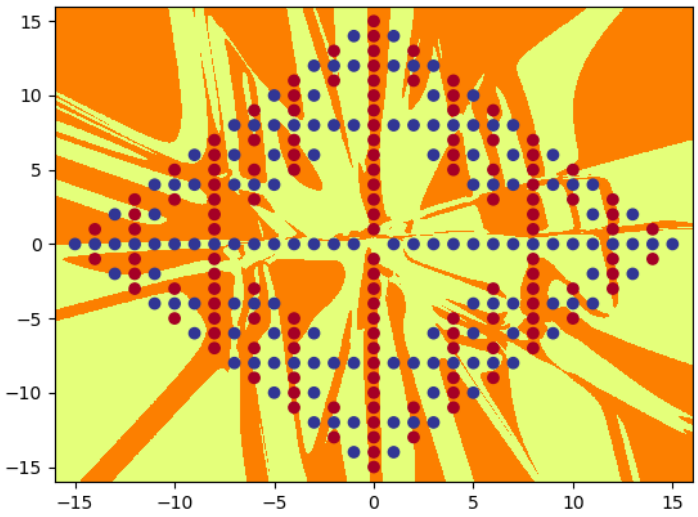
2.
The number of hidden nodes close to the mininum required for the network to be trained successfully is **16**.
The picture of the function is shown below:



The total number of independent parameters in my network is **337**.

3.
See code in frac.py

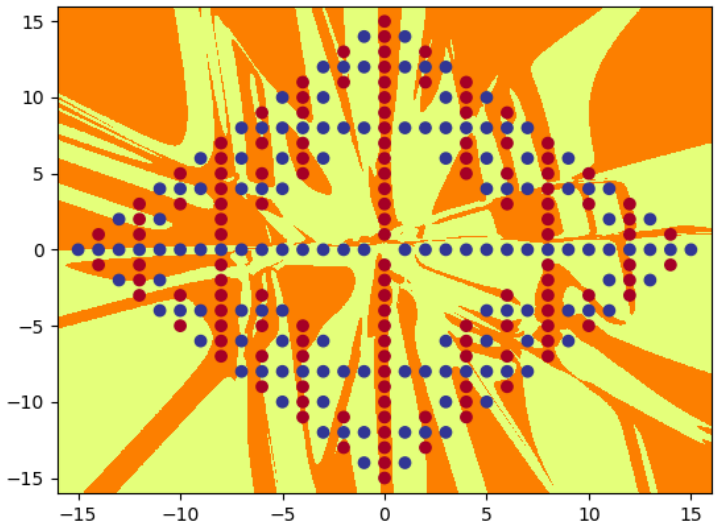
4.
The number of hidden nodes close to the mininum required for the network to be trained successfully is **15**.
The picture of the function is shown below:



The total number of independent parameters in my network is **541**.

5.
See code in frac.py

6.
The number of hidden nodes close to the mininum required for the network to be trained successfully is **14**.
The picture of the function is shown below:



The total number of independent parameters in my network is **311**.

7.
a.
The number of independent parameters in of Full2Net is 337.
The number of independent parameters in of Full3Net is 541.
The number of independent parameters in of DenseNet is 311.

b.
Full3Net only extracts features at one layer and transmits them to the next layer. In other words, features extracted from Hid1 of Full3Net cannot be transmitted to Hid3. In contrast, each feature of DenseNet can be transmitted to each subsequent layers.

c.
Through the comparison of the three pictures, I did not find any obvious differences between the overall function computed by the three networks, which all belong to the classification results of serious over-fitting.

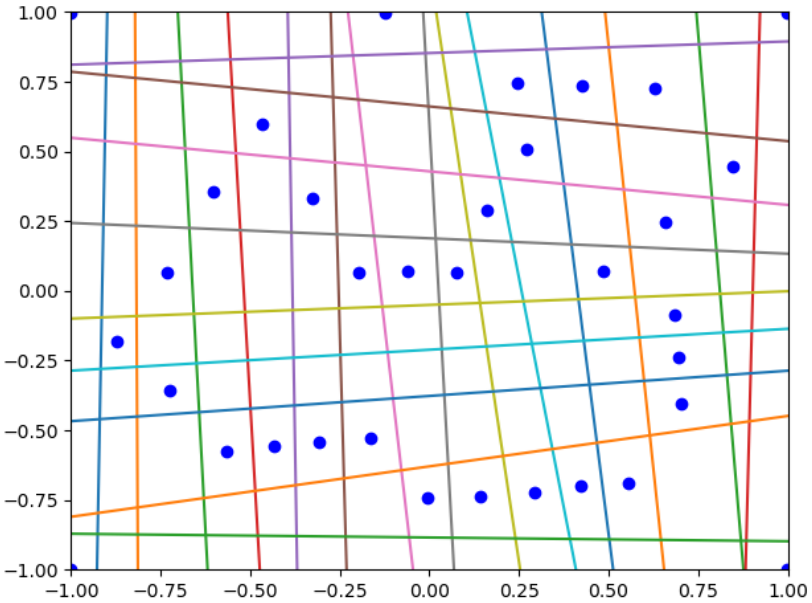
Part 3: Encoder Networks

1.

The tensor is

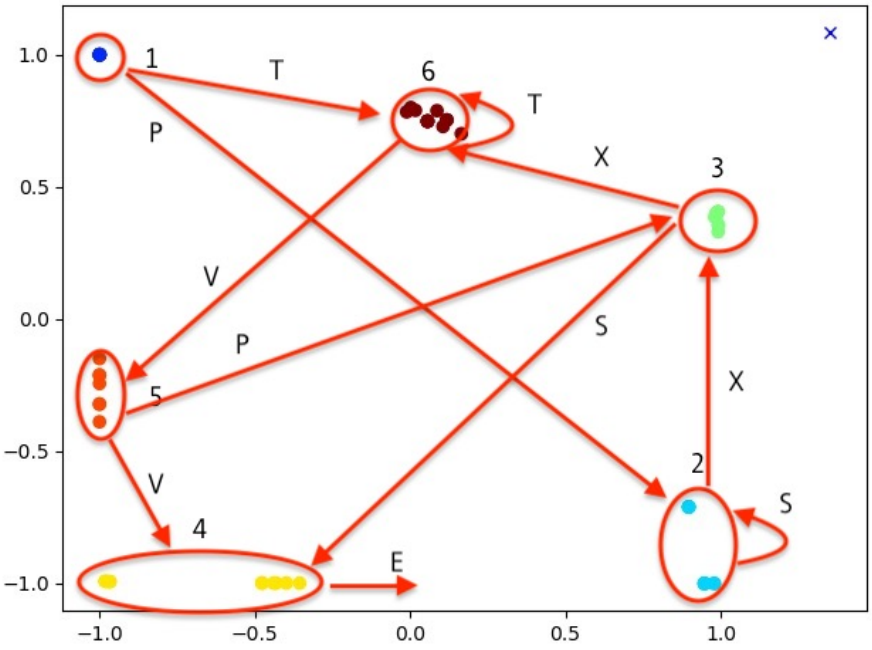
```
ch34 = torch.Tensor(
    [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1],
     [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1],
     [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1],
     [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

The final image is shown below:

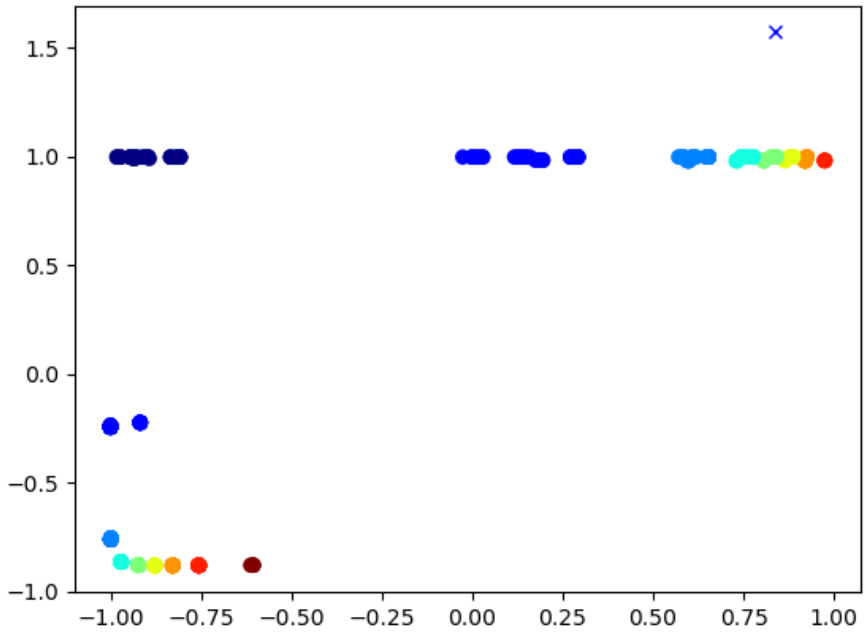


Part 4: Hidden Unit Dynamics for Recurrent Networks

1.
The hidden unit activations figure is shown below:



2.
The hidden unit activations figure is shown below:

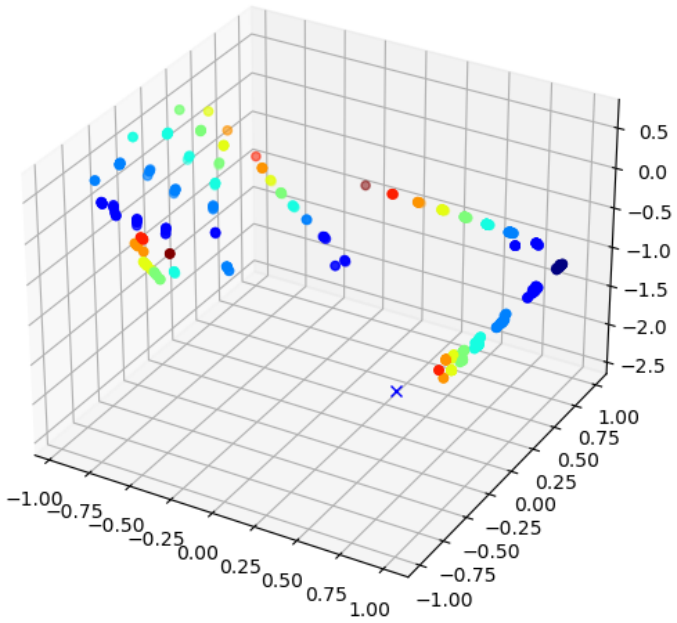


3.

The hidden unit activations start from $[-1,1]$, and then x gradually increases, and there is probability that it will change to $[x,-1]$. After changing to $[x,-1]$, x becomes smaller gradually, and then enters the hidden unit activation $[-1,0.3]$ (blue points), and then enters $[-1,1]$. Over and over again.

4.

The hidden unit activations figure is shown below:



5.

Similar to anbn, the last B and the last C will have a special hidden unit activations (blue points) corresponding to thems. The hidden unit activations start from $[1,1-1]$, and then x gradually decreases, and there is probability that it will change to $[1,y,-1]$. After changing to $[1,y,-1]$ y becomes bigger gradually, and then enters the hidden unit activation $[0.5,-0.5,1]$, and then x and z becomes bigger gradually, then enters the blue points, finally return $[1,1-1]$. Over and over again.