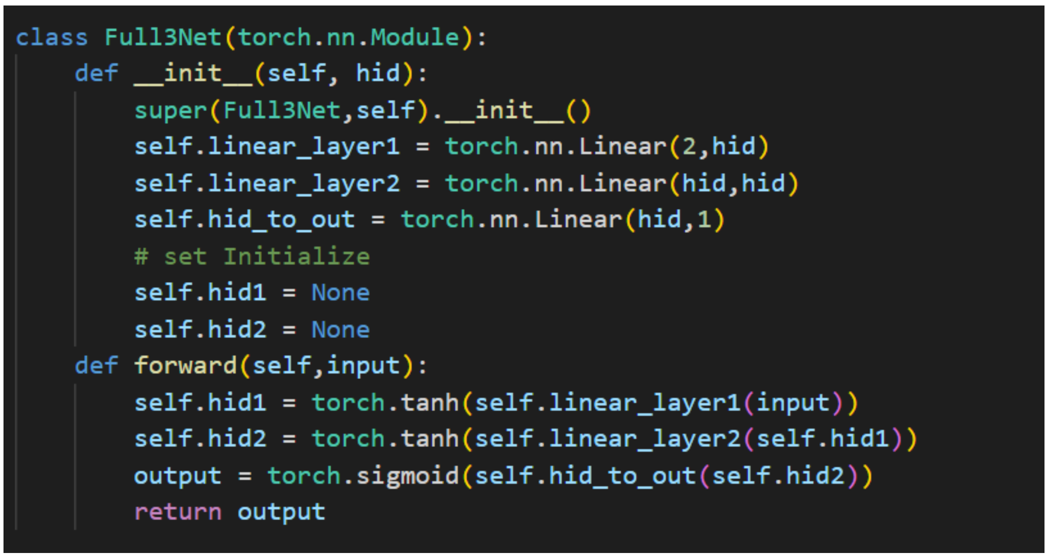
COMP9444 Neural Networks and Deep Learning

Term 2, 2022

Assignment 1 - Network Structures and Hidden Unit Dynamics

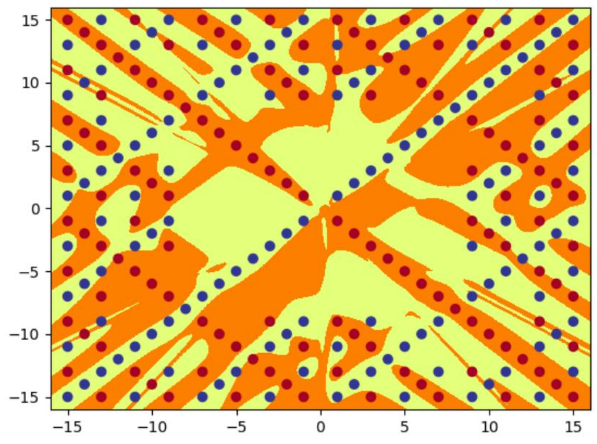
Part 1: Fractal Classification Task

1.

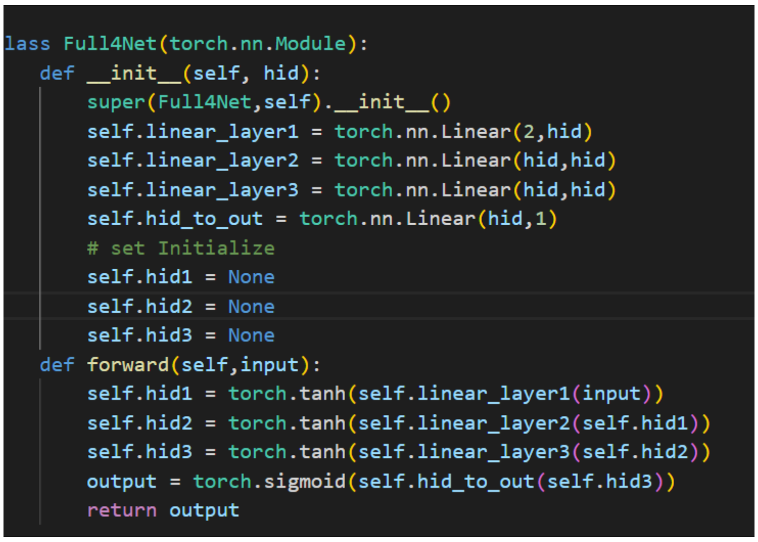


```python
class Full3Net(torch.nn.Module):
    def __init__(self, hid):
        super(Full3Net,self).__init__()
        self.linear_layer1 = torch.nn.Linear(2,hid)
        self.linear_layer2 = torch.nn.Linear(hid,hid)
        self.hid_to_out = torch.nn.Linear(hid,1)
        # set Initialize
        self.hid1 = None
        self.hid2 = None
    def forward(self,input):
        self.hid1 = torch.tanh(self.linear_layer1(input))
        self.hid2 = torch.tanh(self.linear_layer2(self.hid1))
        output = torch.sigmoid(self.hid_to_out(self.hid2))
        return output
```

2. Hid = 20. The total number of independent parameter is (2*20+20)+(20*20+20)+(20+1) = 501



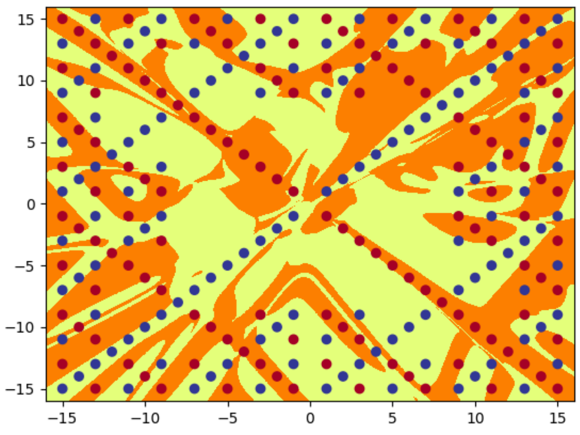
3.



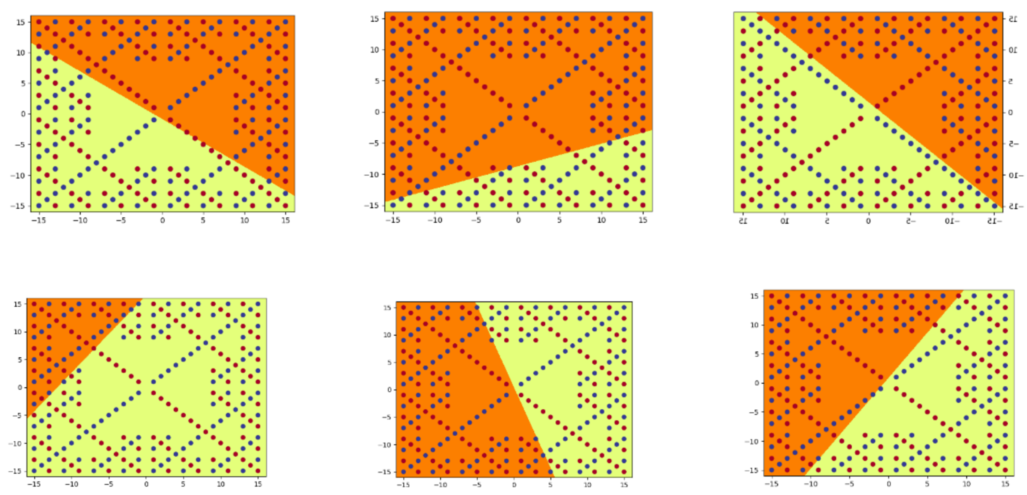
```
lass Full4Net(torch.nn.Module):
    def __init__(self, hid):
        super(Full4Net,self).__init__()
        self.linear_layer1 = torch.nn.Linear(2,hid)
        self.linear_layer2 = torch.nn.Linear(hid,hid)
        self.linear_layer3 = torch.nn.Linear(hid,hid)
        self.hid_to_out = torch.nn.Linear(hid,1)
        # set Initialize
        self.hid1 = None
        self.hid2 = None
        self.hid3 = None
    def forward(self,input):
        self.hid1 = torch.tanh(self.linear_layer1(input))
        self.hid2 = torch.tanh(self.linear_layer2(self.hid1))
        self.hid3 = torch.tanh(self.linear_layer3(self.hid2))
        output = torch.sigmoid(self.hid_to_out(self.hid3))
        return output
```

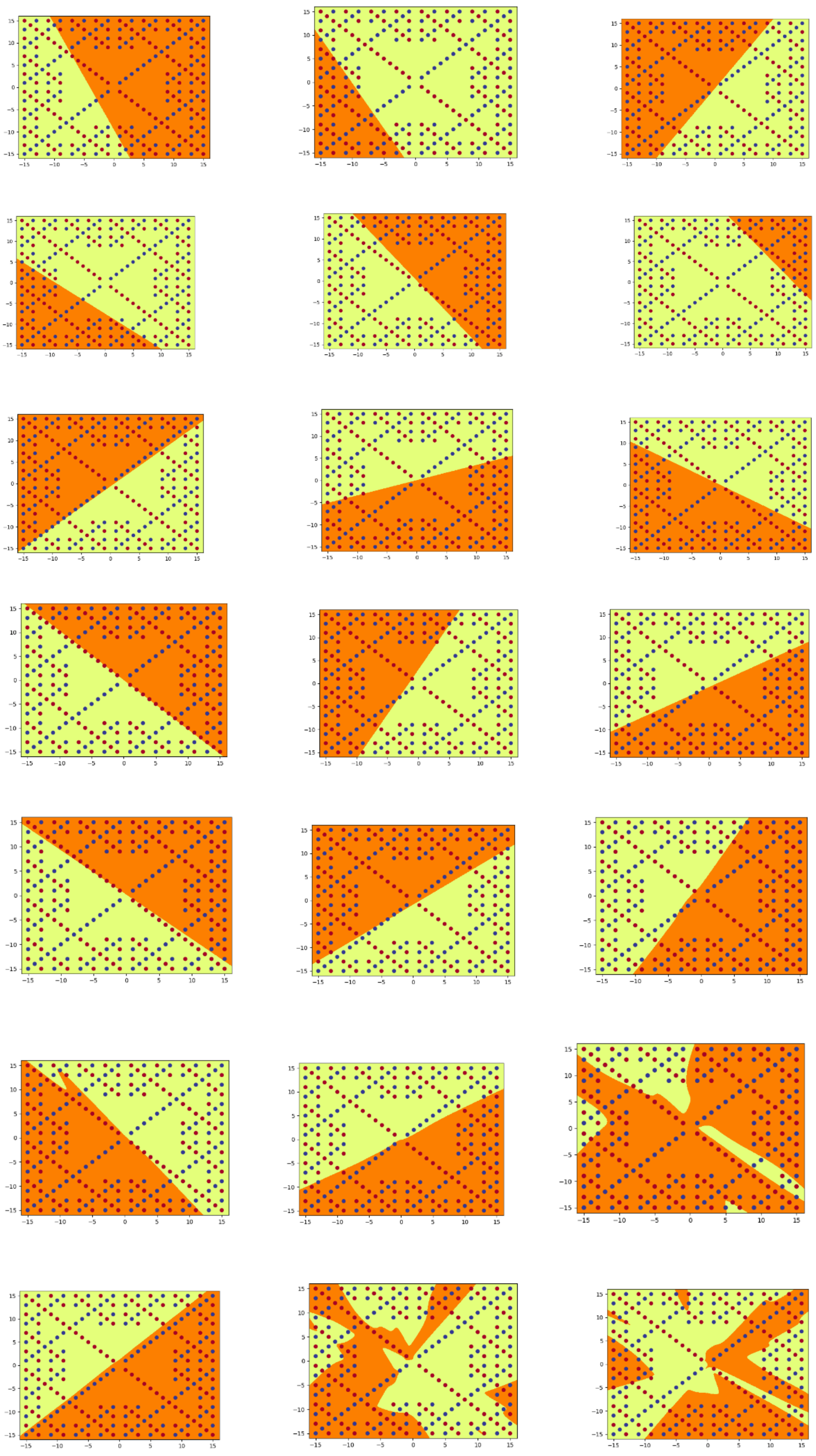
4. Hid = 20. The total number of independent parameters is (2*20+20)+(20*20+20)+(20*20+20)+(20+1) = 921

The plot of the output

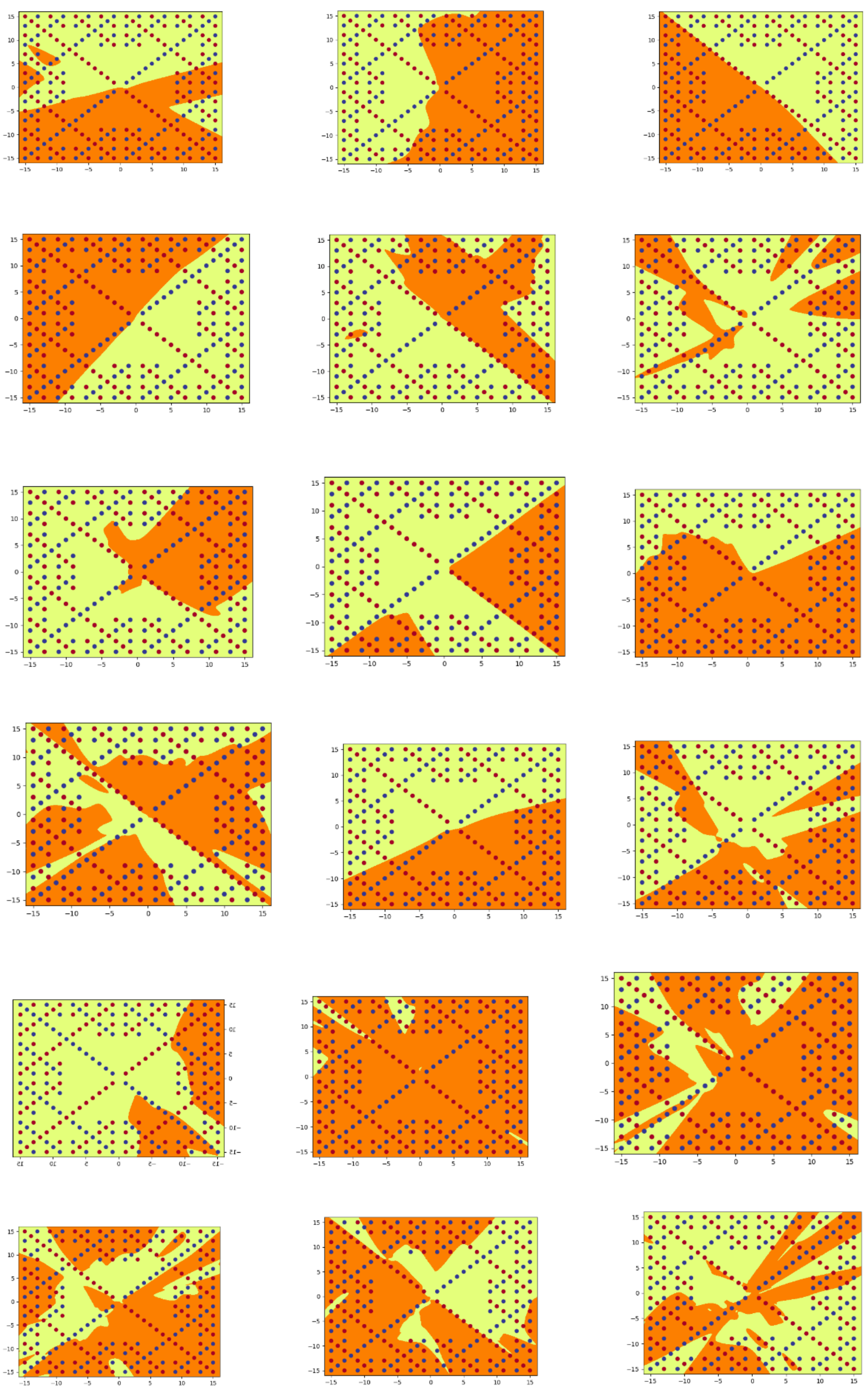


The plots of all the hidden units in all three layers

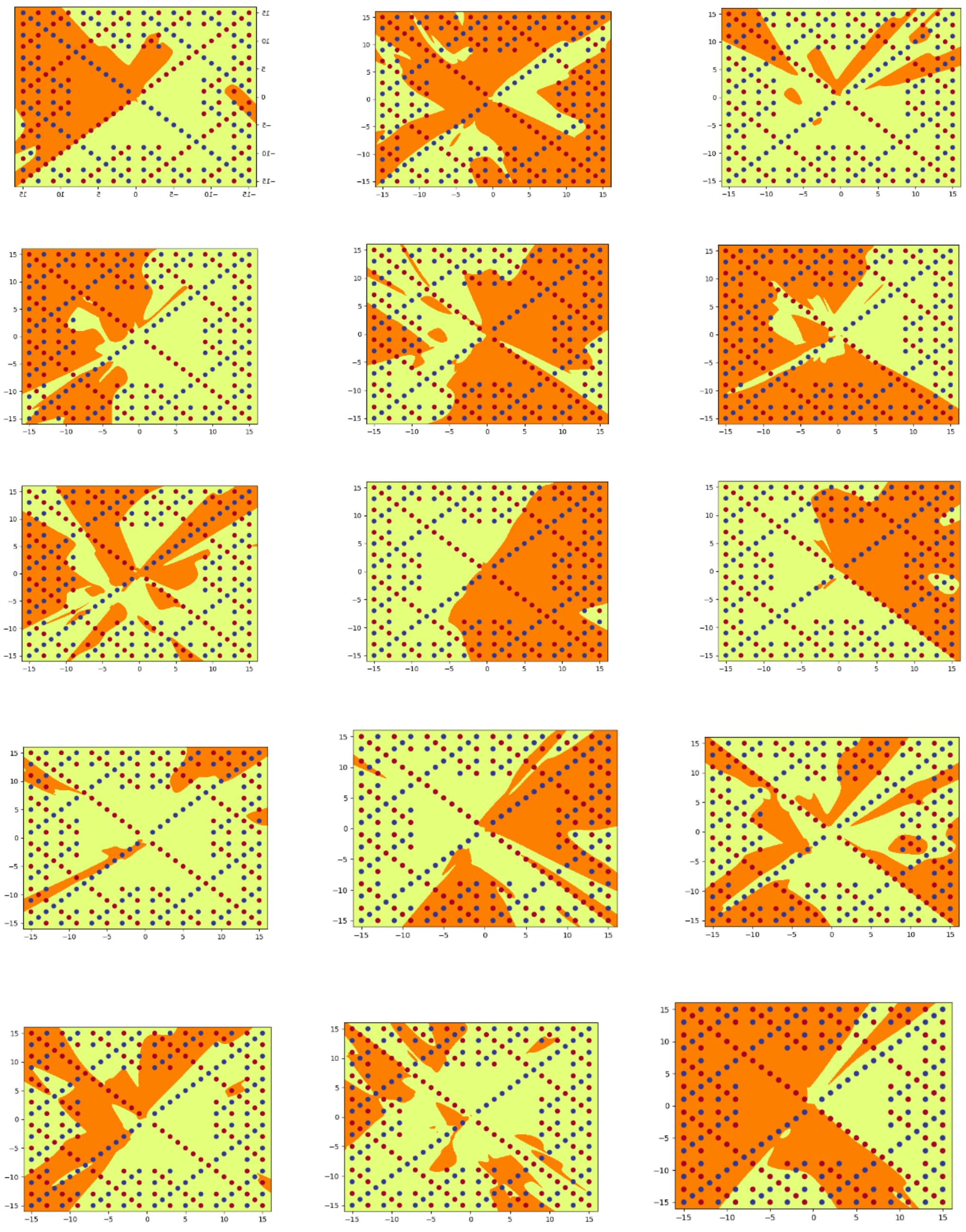




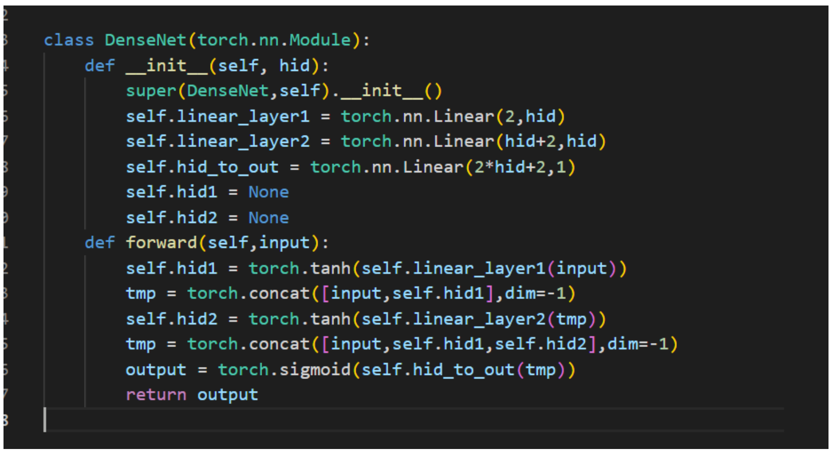
15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15



−15
−10
−5
0
5
10
15



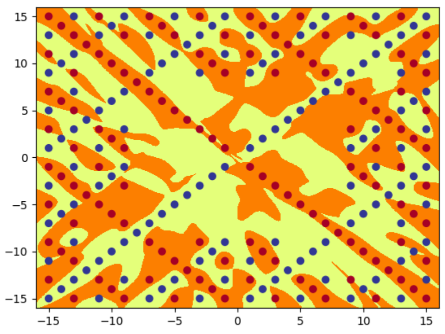
5.



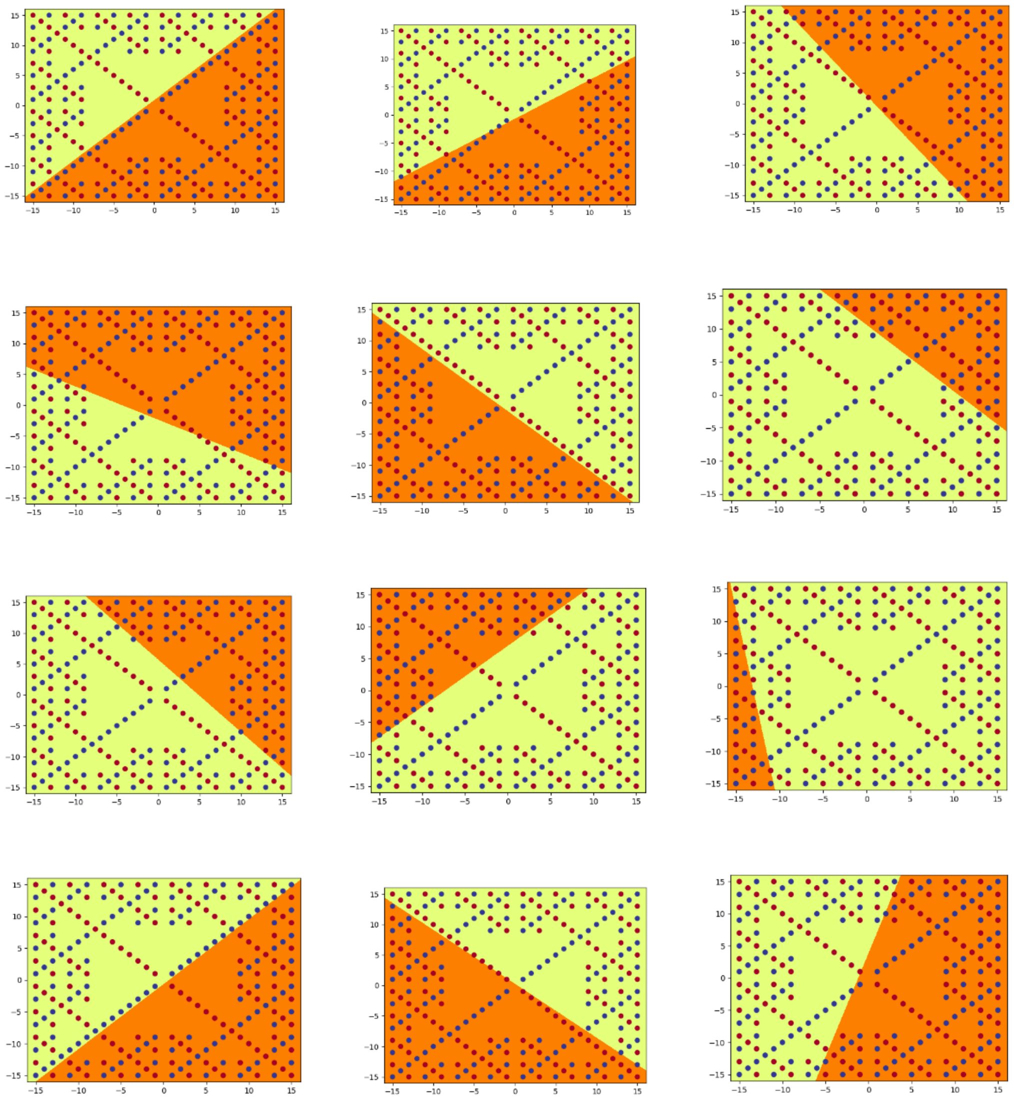
```python
class DenseNet(torch.nn.Module):
    def __init__(self, hid):
        super(DenseNet,self).__init__()
        self.linear_layer1 = torch.nn.Linear(2,hid)
        self.linear_layer2 = torch.nn.Linear(hid+2,hid)
        self.hid_to_out = torch.nn.Linear(2*hid+2,1)
        self.hid1 = None
        self.hid2 = None
    def forward(self,input):
        self.hid1 = torch.tanh(self.linear_layer1(input))
        tmp = torch.concat([input,self.hid1],dim=-1)
        self.hid2 = torch.tanh(self.linear_layer2(tmp))
        tmp = torch.concat([input,self.hid1,self.hid2],dim=-1)
        output = torch.sigmoid(self.hid_to_out(tmp))
        return output
```

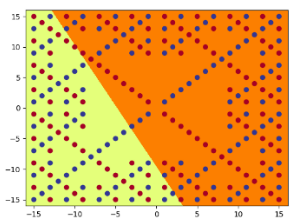
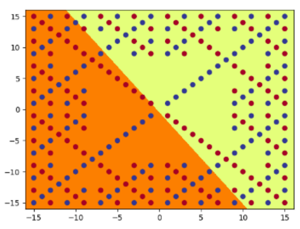
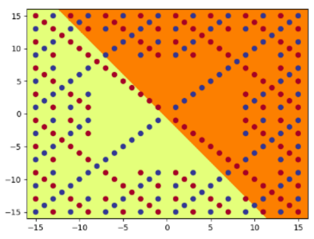
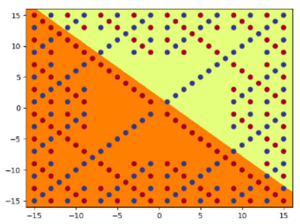
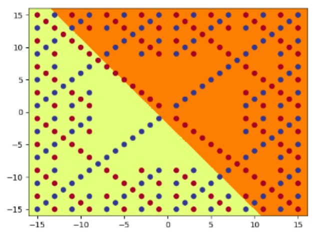
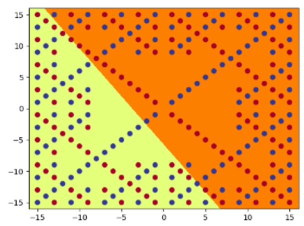
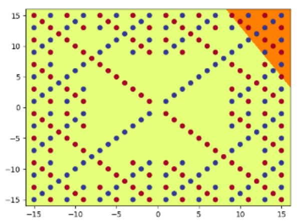
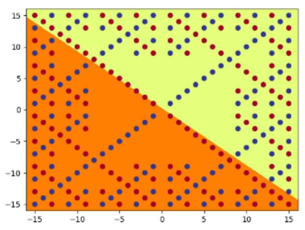
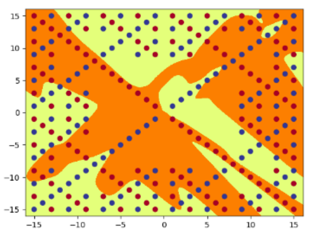
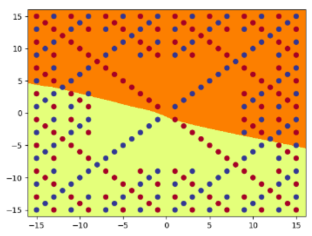
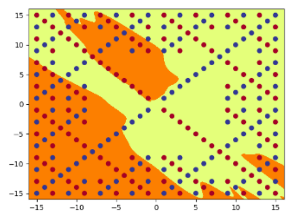
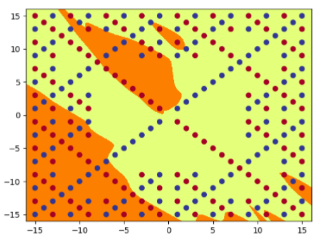
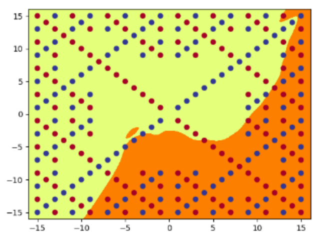
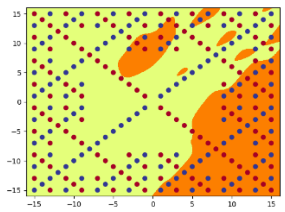
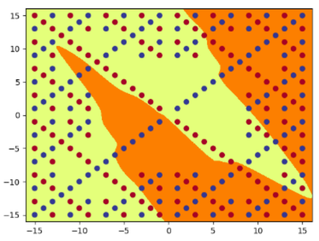
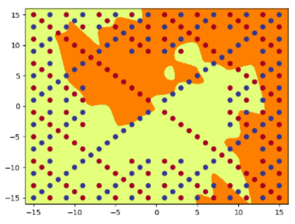
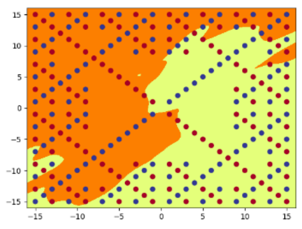
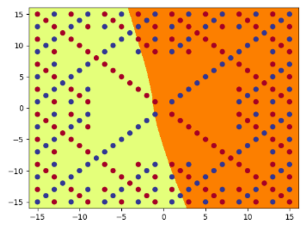
6. Hid = 20. The total number of independent parameter is (2*20+20)+(22*20+20)+(42+1) = 563

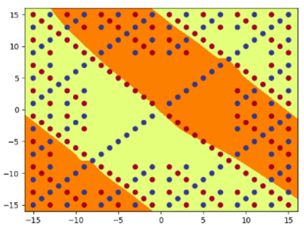
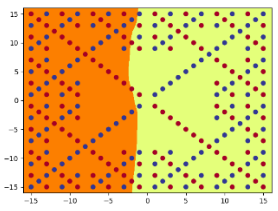
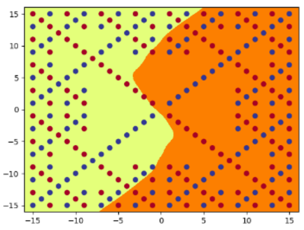
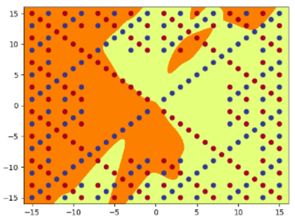
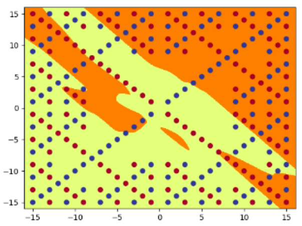
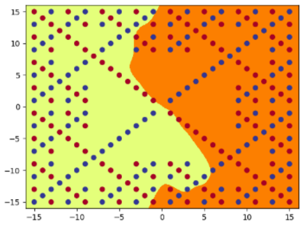
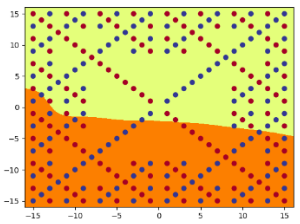
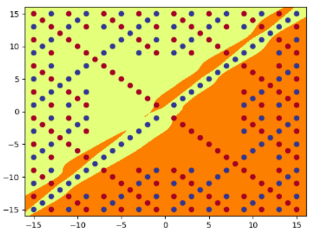
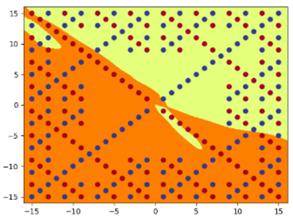
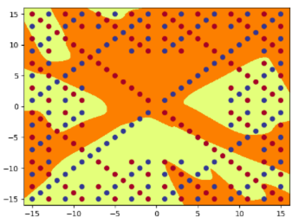
   The plot of the output



The plots of all the hidden units in all two layers



7. a)

   Full3Net: 501 independents parameters, 71300 epochs required

   Full4Net: 921 independents parameters, 119800 epochs required

   DenseNet: 563 independents parameters, 147800 epochs required

   b)

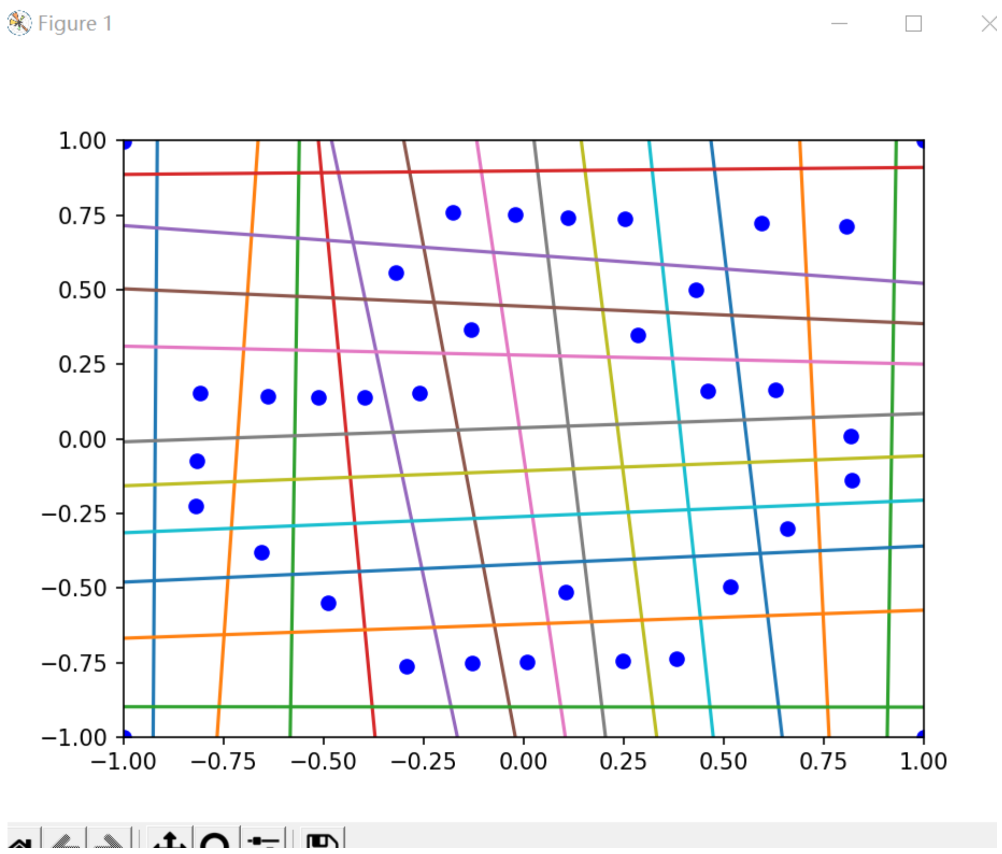
   Full4Net is a structure that the input of the latter layer is only from the output of the previous layer. The output of the first hidden layer is the input of the second layer. The output of the second hidden layer is the input of the third layer.

   DenseNet is a structure that the input to the latter layer is a vector that concatenates the outputs of all the previous layers. The input of second layer is from the input and the output of the first layer. The input of the out layer is from the input layer, first hidden layer and the second hidden layer.

   c)

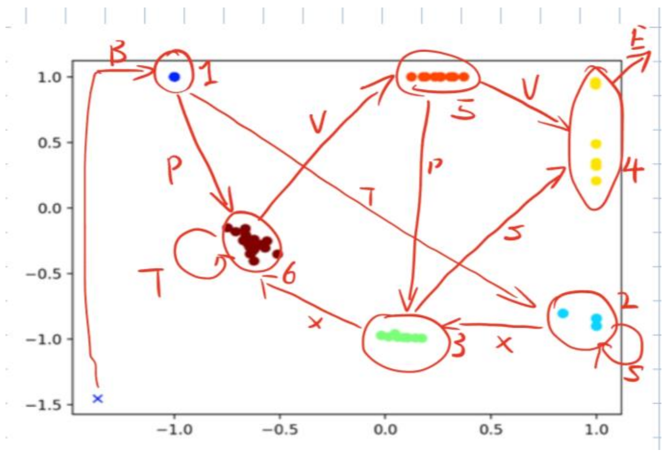
   I don't think there's that much difference between the overall function. Full3Net needs the least amount of computation.

Part 2: Encoder Networks

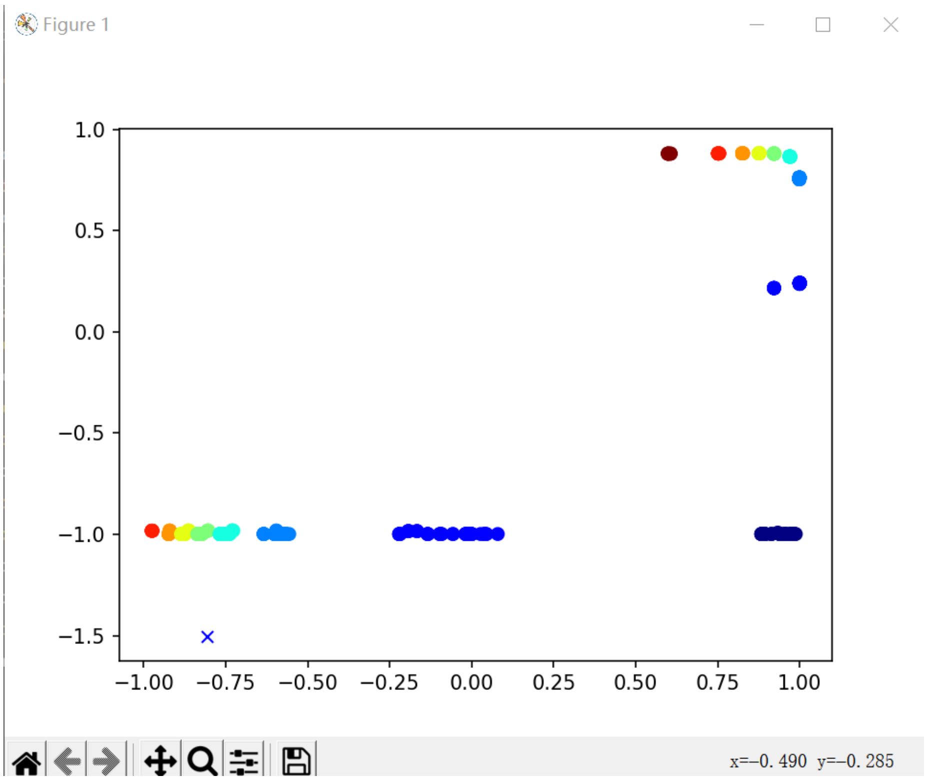


Part 3: Hidden Unit Dynamics for Recurrent Networks

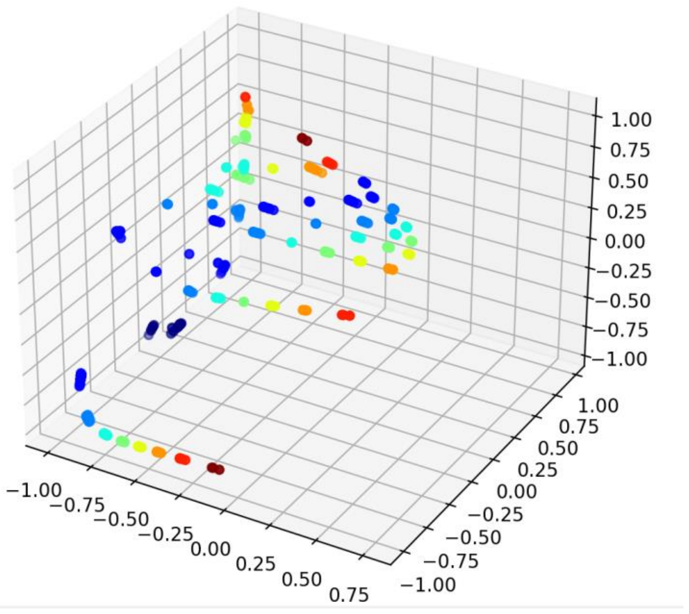
1.



2.



3. The Simple recurrent network can be augmented with connections from the output to the hidden layer. The network knows the pervious information. When the first B comes, the network knows the number of A in the network, so the network will know how many B it should has and when the next A will come. A and B should have the same number because of A^nB^n. The top-right of the image represents A, the bottom of the image represent represents B. The hidden unit activations jump from the top-right of the image to the bottom for each color except the dark blue. It is hard to predict A except the A following B.
4.



5. It is similar to question 3. The network knows the pervious information. In each hidden

node, it can get all the information from the pervious. When the first B comes, the network knows how many of A there will be in the network, so it can predict the number of B and C and when C and next A will come. The bottom-left of the image represents A.

6. The Long short-term memory is able to learn long range dependencies using a combination of forget, input and output gates. The forget gate can decide which information can be reserve.