# Homework 1 report

## Part 1

The accuracy of the three models:

Net lin:

```
[[765.   7.   8.   4.  64.   8.   5.  15.  11.   8.]
 [  5. 669.  59.  36.  53.  27.  22.  30.  37.  54.]
 [  8. 106. 694.  59.  78. 125. 144.  26.  98.  87.]
 [ 15.  19.  27. 759.  20.  16.  10.  11.  39.   4.]
 [ 30.  28.  26.  14. 620.  19.  24.  80.   6.  54.]
 [ 61.  23.  20.  56.  20. 724.  23.  16.  31.  32.]
 [  2.  59.  46.  15.  34.  29. 726.  54.  44.  21.]
 [ 63.  13.  39.  19.  36.   8.  21. 626.   6.  31.]
 [ 31.  26.  44.  25.  20.  33.  10.  91. 704.  39.]
 [ 20.  50.  37.  13.  55.  11.  15.  51.  24. 670.]]

Test set: Average loss: 1.0094, Accuracy: 6957/10000 (70%)
```

Net full: When the number of hidden nodes is 130:

```
[[858.   5.   7.   5.  42.  10.   3.  21.  13.   2.]
 [  5. 820.  19.  12.  37.  13.  13.  16.  30.  17.]
 [  1.  30. 834.  29.  20.  73.  55.  16.  24.  48.]
 [  7.   3.  38. 911.   6.  12.   9.   4.  44.   7.]
 [ 22.  22.  11.   2. 808.  10.  17.  23.   2.  24.]
 [ 28.   9.  18.  14.   8. 825.   4.  10.  10.   6.]
 [  4.  53.  25.   8.  27.  27. 881.  34.  27.  23.]
 [ 39.   3.  11.   3.  16.   2.  10. 819.   3.  17.]
 [ 31.  22.  20.   5.  22.  19.   1.  25. 839.  14.]
 [  5.  33.  17.  11.  14.   9.   7.  32.   8. 842.]]

Test set: Average loss: 0.5186, Accuracy: 8437/10000 (84%)
```

When the number of hidden nodes is 180:

```
[[859.   5.   9.   4.  41.   8.   3.  16.  10.   2.]
 [  2. 826.  18.  10.  32.  16.  13.  19.  25.  19.]
 [  1.  33. 852.  32.  19.  82.  50.  19.  28.  44.]
 [  5.   4.  35. 913.   8.   9.  11.   5.  50.   9.]
 [ 25.  17.  10.   2. 810.  13.  13.  23.   5.  31.]
 [ 32.   7.  14.  17.  10. 828.   5.   9.   9.   5.]
 [  2.  56.  20.   6.  31.  19. 889.  34.  30.  23.]
 [ 37.   4.  12.   1.  17.   2.   6. 827.   4.  19.]
 [ 33.  20.  17.   6.  19.  17.   1.  18. 831.  15.]
 [  4.  28.  13.   9.  13.   6.   9.  30.   8. 833.]]

Test set: Average loss: 0.5061, Accuracy: 8468/10000 (85%)
```

Net conv:

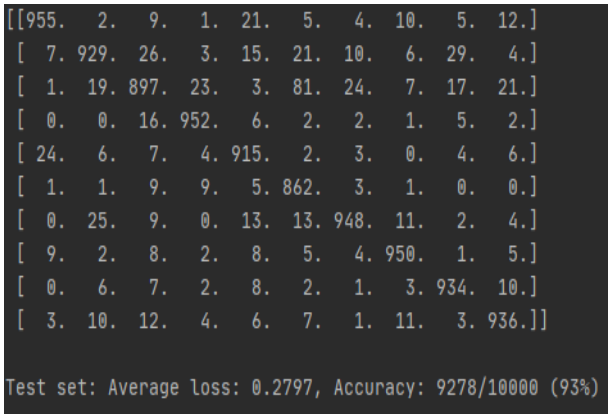
```
[[955.   2.   9.   1.  21.   5.   4.  10.   5.  12.]
 [  7. 929.  26.   3.  15.  21.  10.   6.  29.   4.]
 [  1.  19. 897.  23.   3.  81.  24.   7.  17.  21.]
 [  0.   0.  16. 952.   6.   2.   2.   1.   5.   2.]
 [ 24.   6.   7.   4. 915.   2.   3.   0.   4.   6.]
 [  1.   1.   9.   9.   5. 862.   3.   1.   0.   0.]
 [  0.  25.   9.   0.  13.  13. 948.  11.   2.   4.]
 [  9.   2.   8.   2.   8.   5.   4. 950.   1.   5.]
 [  0.   6.   7.   2.   8.   2.   1.   3. 934.  10.]
 [  3.  10.  12.   4.   6.   7.   1.  11.   3. 936.]]

Test set: Average loss: 0.2797, Accuracy: 9278/10000 (93%)
```

By the screenshots, we can find the accuracy of the first model is the lowest, which is 70%, because the neural networks only have one layer, the model is underfitting, it cannot fit such a complex image. The second model is better than the first one because it has two layers, although the accuracy of the second model is around 85%, the third model has two convolutional layers, compare to fully connected layer, the convolution layer has the ability to extract part information of pictures and it can process images with fewer parameters, so the third model has the highest accuracy 93%.

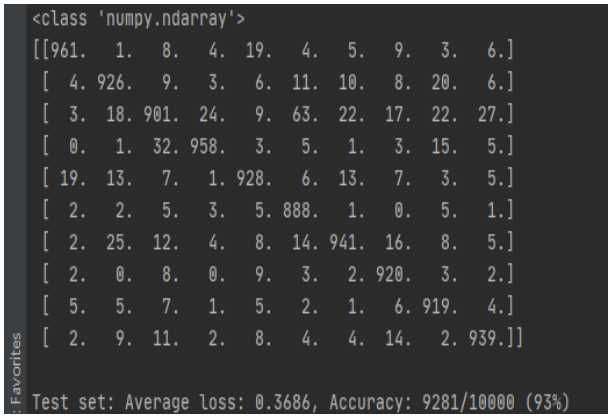
As we can find by screenshots, in the first model, target character ‘ma’ is most likely to be mistaken for character ‘su’, as well as in other two models, target character ‘ha’ is most likely to be mistaken for character ‘su’. In my opinion, the reason of that is characters ‘ma’ and ‘su’ look like ‘su’, there is no special feature (different type of strokes) to distinguish them.

When I tried the third model, I compared the results of whether to use max pooling:

Using max pooling:

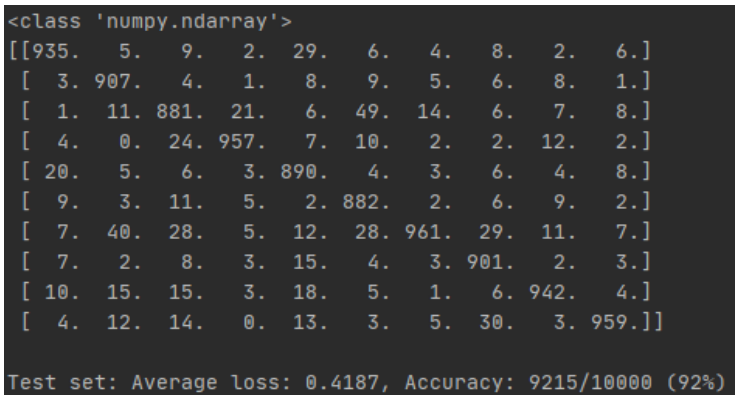



Without max pooling:




Although the accuracies of those are same, using max pooling not only can reduce the average loss but also compress features and reduce the computation.

Another experiment I tried is to add two more convolutional layers in the third model, here is the result:

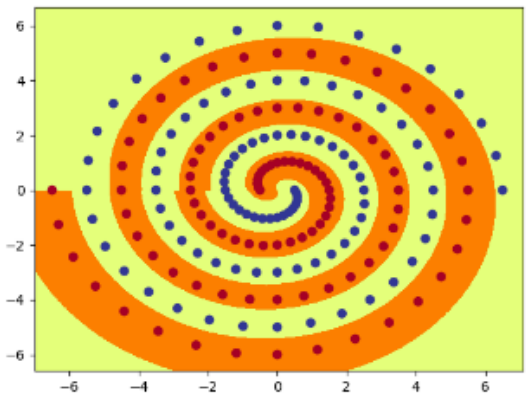



As we can see by the screenshots, the accuracy is lower than two convolution layers model. In my opinion, increasing the number of convolution layers means that the difficulty of training the network has increased, due to the limitations of the current training methods, more convolution layers may be easily to cause the overfitting.

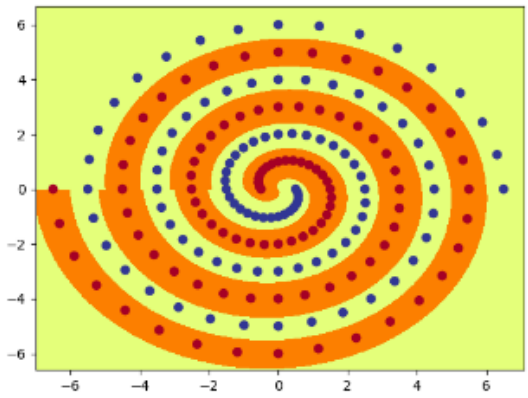
# Part 2

## Polar net:

Hidden nodes = 10:

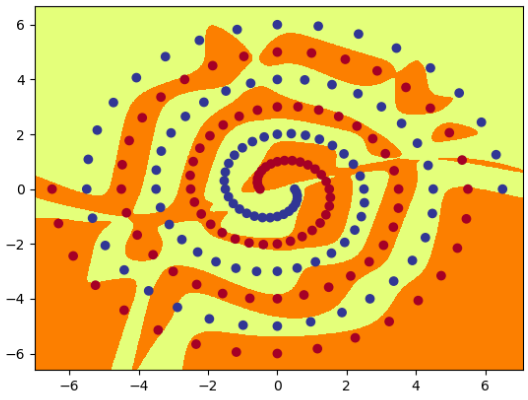


hidden nodes = 7(the minimum number):

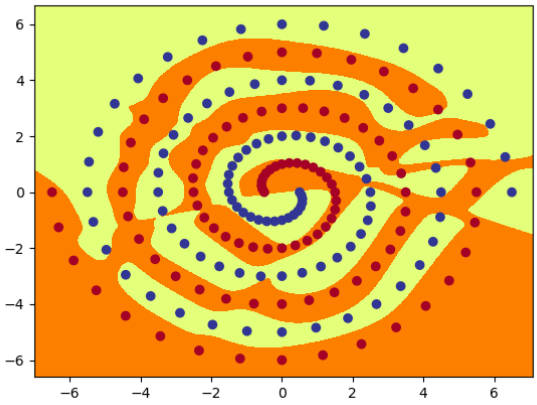


## Raw net:

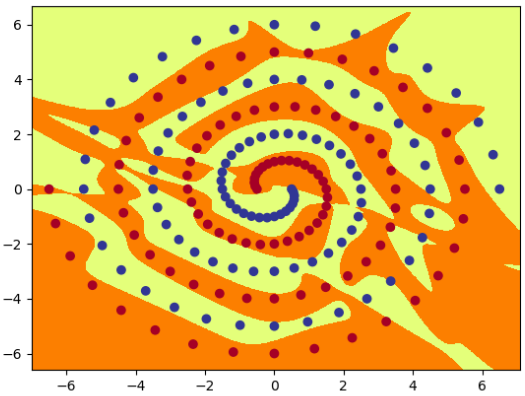
Initial weight = 0.2



Initial weight = 0.3:

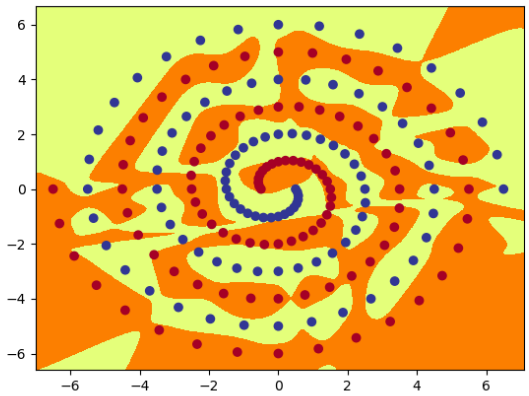


Initial weight = 0.16:

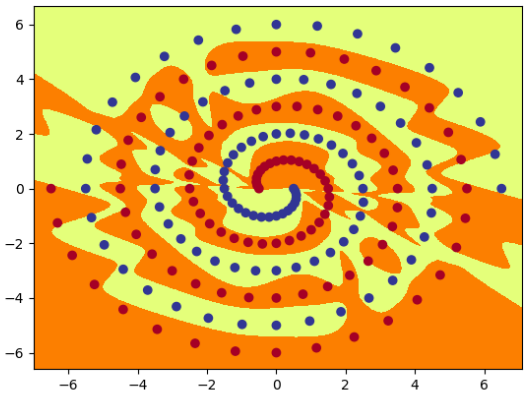


## Short net:

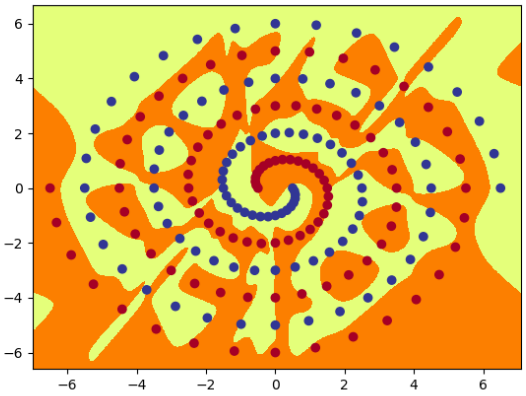
Initial weight = 0.16, hidden nodes = 8:



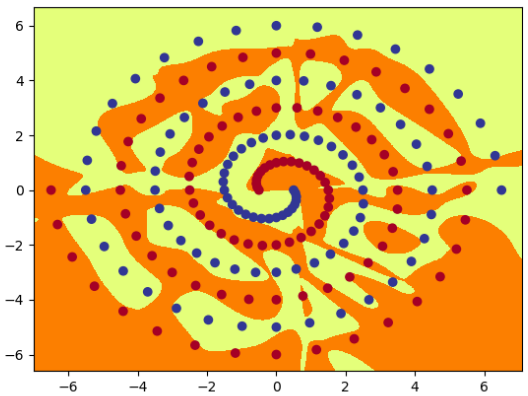
Initial weight = 0.16, hidden nodes = 7:



Initial weight = 0.2, hidden nodes = 7:

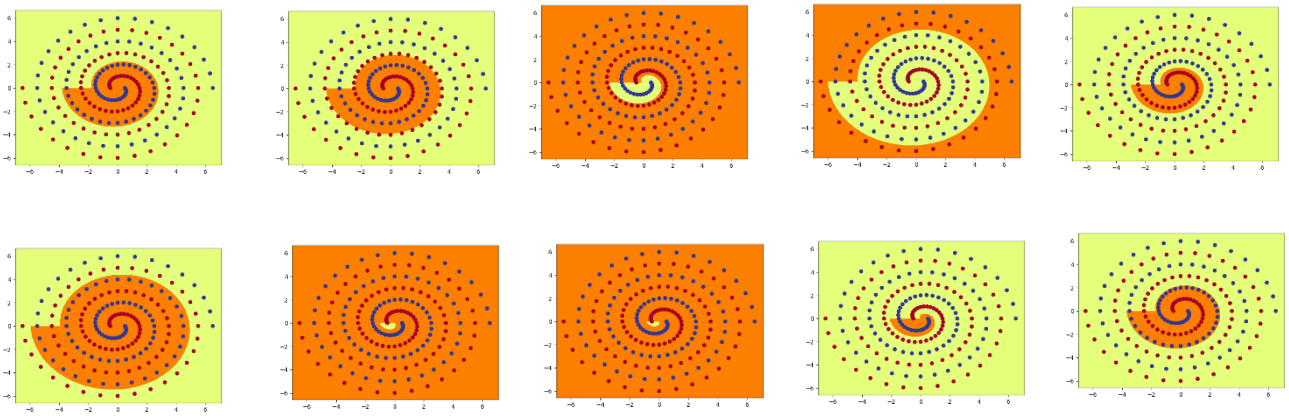


Initial weight = 0.2, hidden nodes = 9:

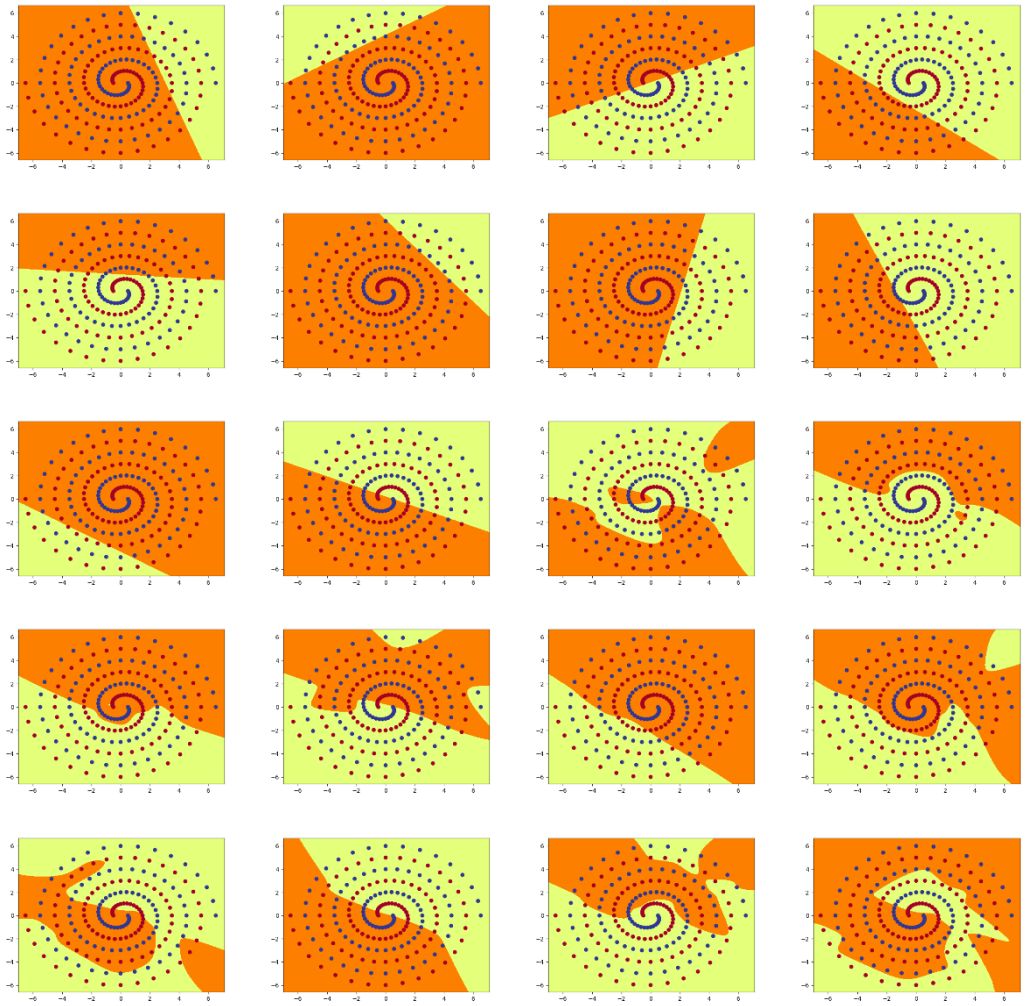


## Plots:

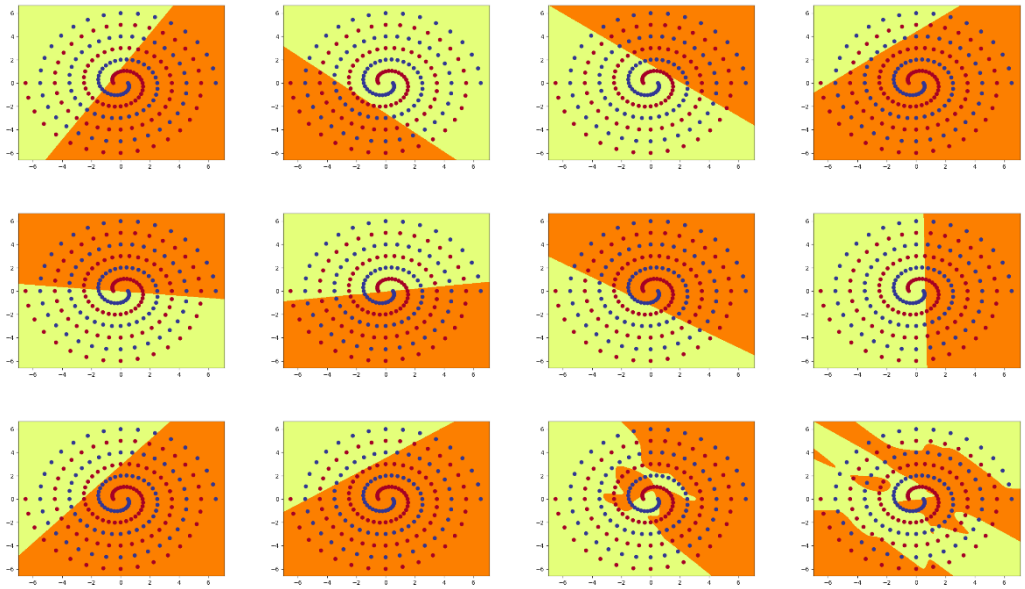
Polar net:

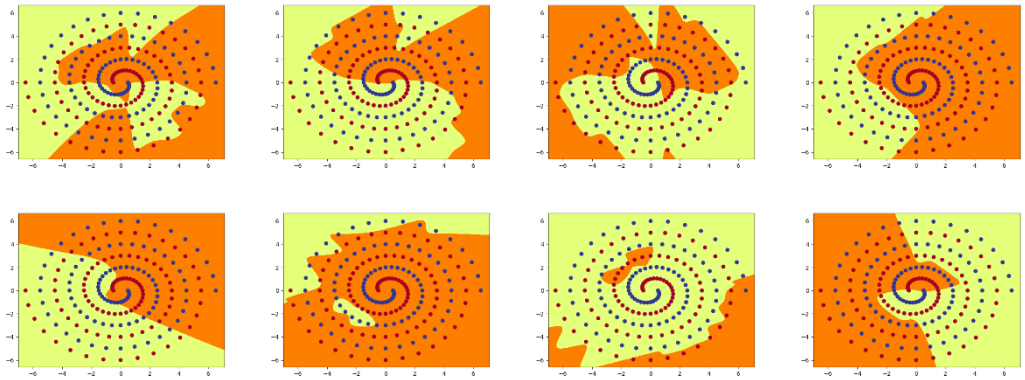


Raw net:



Short net:





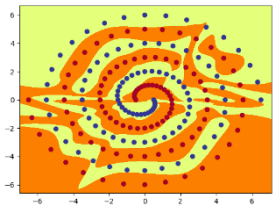
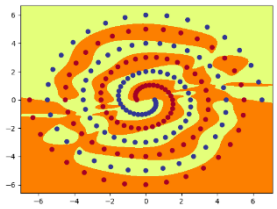
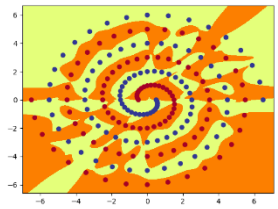
The first model only has one hidden layer, another two models have two hidden layers, the first layer must generate straight line, the second layer must not generate straight line. The first model using polar coordinates as input, points closer to the center have greater weight.

For both rawnet model and shortnet model, when the initial weight is between 0.11 to 0.3, machine can get success of learning, but the speed of learning is not stable, when the initial weight less than 0.2, the speed is decreased, and when the initial weight less than 0.1 or greater than 0.3, I cannot use my computer to get successful machine learning, and when the initial weight is 0.16, the learning speed is the fastest(around 4500 epochs).

As we can find by the screenshots, although the first model only has one fully connected neural network with one hidden layer, converting the input(x, y) to polar co-ordinates(r, a) can make the output more ‘natural’ than other two models, this means that the number of

layers of neural network is not necessarily the more the better, changing the form of input may leads to better results.

After changing batch size from 97 to 194, the three pictures generated by the three models:

by the result pictures as well as the speed and success of learning, we can know: increasing the batch size within a certain range, it can improve the utilization of memory, speed up the data processing with the same amount of data, and reduce the time of generating results, but it will affect the accuracy of the model.

For another experiment, I changing tanh to relu in three models, but I cannot get successful outputs, because the loss is too much (around 0.2-0.7). The computing speed of relu function is very fast, it can save CPU resources, and the convergence speed of relu is quicker than the tanh function, but relu function blocks too much features, which makes the model unable to learn effective features.

Using relu function cannot get 100% accuracy output:

```
ep:21100 loss: 0.2216 acc: 78.35
ep:21200 loss: 0.2213 acc: 78.87
```

```
ep:11200 loss: 0.2325 acc: 72.16
ep:11300 loss: 0.2321 acc: 73.20
```

```
p: 3500 loss: 0.7128 acc: 50.52
p: 3600 loss: 0.7128 acc: 50.52
p: 3700 loss: 0.7128 acc: 50.52
p: 3800 loss: 0.7128 acc: 50.52
p: 3900 loss: 0.7128 acc: 50.52
p: 4000 loss: 0.7128 acc: 50.52
p: 4100 loss: 0.7128 acc: 50.52
```