COMP 9444: Assignment 1 Report

Part 1: Japanese Character Recognition

1) The final confusion matrix and accuracy yielded by the model NetLin is:

```
[[767.   7.   6.   4.  60.   8.   4.  17.  11.   8.]
 [5. 670.  61.  33.  53.  29.  22.  29.  37.  52.]
 [7. 107. 697.  62.  81. 125. 148.  26.  99.  88.]
 [14.  17.  26. 758.  22.  17.  10.  12.  42.   4.]
 [31.  28.  25.  15. 624.  19.  25.  82.   6.  52.]
 [64.  24.  19.  57.  18. 726.  24.  17.  31.  31.]
 [2.  59.  47.  14.  32.  25. 724.  56.  45.  20.]
 [62.  13.  39.  18.  35.   8.  20. 622.   6.  31.]
 [30.  26.  44.  27.  21.  32.  10.  91. 702.  40.]
 [18.  49.  36.  12.  54.  11.  13.  48.  21. 674.]]
```

Test set: Average loss: 1.0099, Accuracy: 6964/10000 (70%)

2) The final confusion matrix and accuracy yielded by the model NetFull is:

```
[[826.   4.   6.   2.  79.   8.   3.  16.   9.   5.]
 [3. 773.  31.  18.  44.  20.  25.  17.  31.  34.]
 [3.  39. 781.  54.  32.  98.  95.  19.  28.  69.]
 [9.   6.  34. 865.  15.  16.   9.   7.  40.   7.]
 [28.  29.  14.   6. 712.  24.  21.  72.   3.  62.]
 [34.   6.  11.  15.   5. 745.   2.   7.  17.  10.]
 [5.  75.  53.  13.  31.  45. 816.  56.  41.  21.]
 [46.   8.  18.   6.  15.   3.  10. 698.   3.  16.]
 [38.  27.  29.   8.  35.  31.   5.  71. 820.  17.]
 [8.  33.  23.  13.  32.  10.  14.  37.   8. 759.]]
```

Test set: Average loss: 1.2091, Accuracy: 7795/10000 (78%)

3) The final confusion matrix and accuracy yielded by the model NetConv is: (The network is implemented using maxpooling and all the metaparameter values are set to default values for the sake of simplicity and ease of understanding.)

```
[[951.   2.   8.   3.  16.   4.   2.  10.   5.   6.]
 [2. 929.   9.   4.  10.  18.   9.   6.  19.   4.]
 [1.   6. 882.  16.   4.  56.  11.   5.   7.   8.]
 [1.   0.  26. 946.   1.   4.   2.   0.   4.   1.]
 [29.  15.  14.  10. 938.  11.   9.  11.  19.  13.]
 [0.   0.   4.   4.   3. 856.   1.   0.   1.   1.]
 [2.  32.  30.   6.   7.  35. 960.   8.  16.   3.]
 [8.   4.  10.   5.   9.   5.   3. 938.   4.   3.]
 [3.   1.   4.   2.   5.   3.   0.   2. 921.   2.]
 [3.  11.  13.   4.   7.   8.   3.  20.   4. 959.]]
```

Test set: Average loss: 0.2834, Accuracy: 9280/10000 (93%)

4) A: The relative accuracy of the three models can be calculated by first calculating the relative accuracies of each model and by computing the average of all.

The relative accuracy of NetLin: 0.7 (70%)
The relative accuracy of NetFull: 0.78 (78%)
The relative accuracy of NetConv: 0.93 (93%)

The net relative accuracy of all the models is: 80%

B: For Model NetLin:
The Confusion matrix indicates that the model gets confused between characters “su” (2) and “ma” (6) as seen in the matrix attached above. The rest of the characters exhibit little to no confusion.

For Model NetFull:
The confusion matrix indicates that the model is not confused while classifying values based on the target data set. This indicates that the model is more suited for this classification.

For Model NetConv:
The confusion matrix indicates that the model is not confused while classifying values based on the target data set, the final degree of accuracy achieved correlates with this inference.

C: The models in this file classify datasets in an increasing order of accuracy. This is achieved by increasing the number of hidden layers and by using Convolutional layers instead of linear layers. The model with two convolutional layers and one fully connected layer as used in the last model attains an accuracy of 93%, which helps to classify the characters from the dataset. Although the model has been implemented using default values for the metaparameters, which is 0.01 for learning rate, changing them based on following values lets us infer the following:

- `python3 kuzu_main.py --net conv --lr 0.1`

```
[[975.   3.  11.   1.  19.   2.   6.  13.  10.   7.]
 [  1. 952.  11.   1.  13.   9.  16.  13.  14.  13.]
 [  0.   4. 921.  10.   1.  12.   7.   2.   1.   8.]
 [  2.   1.  15. 978.   6.   7.   3.   2.   8.   2.]
 [  9.   4.   5.   1. 931.   2.   1.   3.   7.  12.]
 [  2.   4.   9.   2.   8. 942.   2.   5.   3.   5.]
 [  1.  17.  10.   3.   9.  18. 962.   5.   3.   1.]
 [  6.   2.   4.   1.   7.   1.   1. 952.   2.   1.]
 [  2.   4.   7.   2.   3.   4.   1.   3. 946.   3.]
 [  2.   9.   7.   1.   3.   3.   1.   2.   6. 948.]]
```

```
Test set: Average loss: 0.2862, Accuracy: 9507/10000 (95%)
```

- python3 kuzu_main.py --net conv --lr 0.5

```
[[   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [1000. 1000. 1000. 1000. 1000. 1000. 1000. 1000. 1000. 1000.]]
```

Test set: Average loss: 2.3090, Accuracy: 1000/10000 (10%)

- python3 kuzu_main.py --net conv --lr 1.0

```
[[   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
 [1000. 1000. 1000. 1000. 1000. 1000. 1000. 1000. 1000. 1000.]]
```

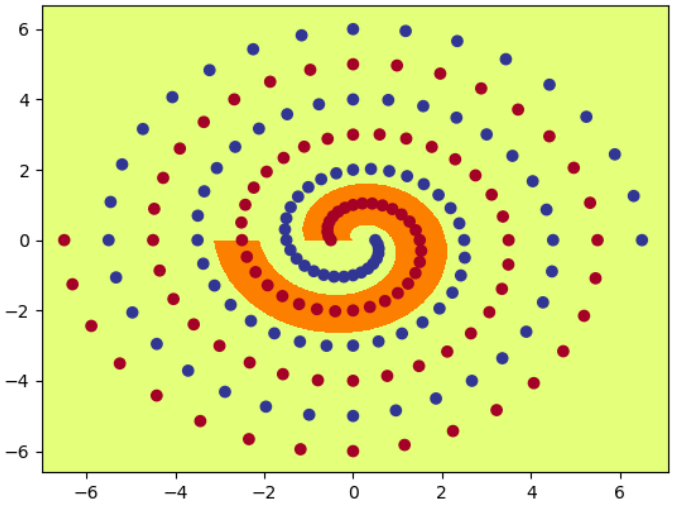
Test set: Average loss: 2.3123, Accuracy: 1000/10000 (10%)

As we can observe, for increasing order of learning rates, the accuracy of the model degrades significantly. In order to increase the accuracy under such cases, the remaining metaparameters such as momentum should be adjusted accordingly. Hence, it can be inferred that the model provides more accurate results when the learning rate is between 0.01 to 0.1

Part 2: Twin Spirals Task

1. PolarNet Pytorch Module:
The (x,y) coordinates are converted into polar coordinates (r,a) in the forward method of the NetPolar Module The output yielded upon execution of this code is a graph plotted in a 'polar_out.png' image file. The same is attached below:

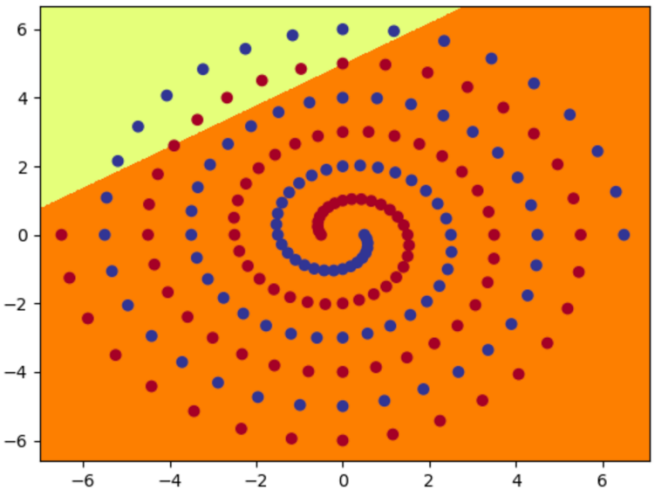
```
ep:99900 loss: 0.3655 acc: 66.49
```



2. The NetPolar module

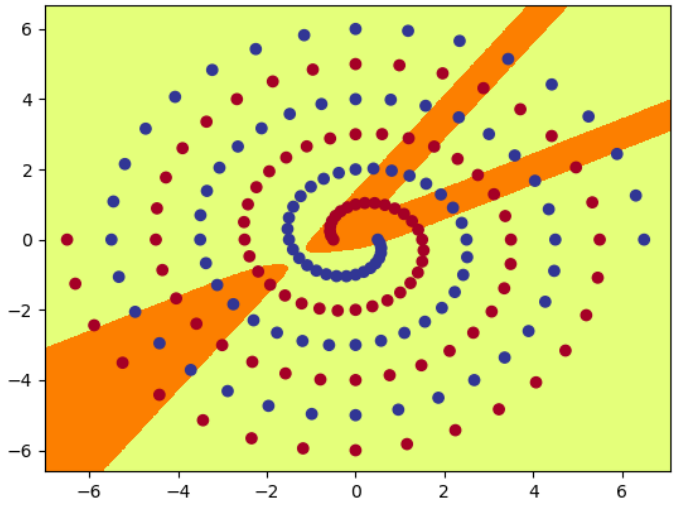
3. RawNet Pytorch Module:
The Network Module has been implemented using individual components as specified in the Assignment specification. This module includes two fully connected input layers wit tanh activation along with an output layer. This module executes using default values for the metaparameters. The output yielded upon execution of this code is a graph plotted in a 'raw_out.png' image file. The same is attached below:

```
ep:99900 loss: 0.6932 acc: 51.55
```



4. The module implemented doesn't include any specialised metaparameters and epochs. Once executed, the module runs the code based on pre-set default values for metaparameters, number of hidden nodes and number of epochs. These specialised values for the same can be mentioned at the time of executing the module through the terminal. For a specified value of the number of hidden nodes as 10 and number of epochs as 20000; upon incrementing the init values, a significantly higher accuracy can be noticed when the value is set at 2.1. The output generated following the execution of this command is as follows:

```
ep:19900 loss: 0.6060 acc: 61.34
```



8. a) For the given problem: Twin Spirals Task, the model needs to contain more than two hidden nodes in order to achieve higher accuracy. The Modules PolarNet and RawNet both contain two hidden nodes hence their relative accuracy is significantly less (around 60%)

b) The initial weight (init) for the module RawNet needs to be increased for a given 10 Hidden Nodes.