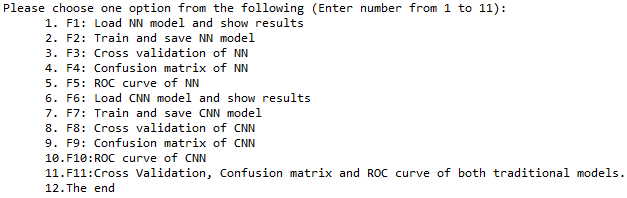
**Checklist:**

|  |  |  |
| --- | --- | --- |
| F1: | 1. Successfully build the structure of both models | √ |
|  | 1. Successfully save both models into file | √ |
| F2: | 1. Successfully do cross validation, confusion matrix, ROC curve and discussion on the discovery for deep learning models. | √ |
|  | 1. Successfully do cross validation, confusion matrix, ROC curve and discussion on the discovery for traditional models in the first assignment | √ |

***1. Detailing how to run your program, including the software dependencies***

**How to run my program：**



This is the user interface of my program. User should choose a number from 1 to 12, any other numbers or digits will not be accepted. The first option intends to load the deep neural network (without convolutional layer) and show its training and test accuracy. The second option means training the dataset and saving the NN model into file. The third, fourth and fifth option intend to do model evaluation on NN model. Option 6-10 aim to perform the same functions on CNN as above. The 11th option intends to do the 3 required model evaluation on both traditional models: sklearn KNN and my KNN. Finally, upon get the input ‘12’, the program will end.

**Software dependencies:**

The program is written by python. It can run in Spyder(python 3.7), sklearn 0.21 and tensorflow 1.14. It uses Scikit-learn algorithm from the machine learning libraries. Numpy, pickle, time, matplotlib and math are all imported in my program. Classification is chosen.

***2. Explaining how the functionalities and additional requirements are implemented and details of your implementation.***

**F1:** **Load NN model and show results**

The method loadNN() is called. nn = joblib.load(‘filename’) is used to load the neural network from file. Accuracy is presented by calling the method self.scoreNN() in class NeuralNetwork.

**F2: Train and save NN model**

1. call trainNN() to train NN model:

Use nn.fit(X\_train, labels\_train, learning\_rate=0.2, epochs=100) to fit the training dataset.

1. save NN model:

Use joblib.dump(nn, 'nnModel.m') to save model into file 'nnModel.m'.

**F3: Cross validation of NN**

1. Call CrossValidationNN() to do cross validation on the digits dataset.
2. Because of the fact that the digits dataset cannot be equally divided into 5 parts, I choose the 1st to 1795th digits to do the cross validation. The shuffle() function is called to mess up the order of this dataset. The vsplit() and hsplit() function are called to partition the data into five subsamples.
3. I iteratively leave one subsample out for the test set and train on the rest using the method nn.fit(x\_train).
4. This process takes around 45 seconds.

**F4: Confusion matrix of NN**

1. Call CMofNN():

Firstly, load the NN model from file.

Secondly, add the predicted test targets into a list.

Thirdly, call the method confusionmatrix(a, b). ‘a’ stands for the list of real targets of Y\_test and ‘b’ stands for the list of predicted test targets.

1. Call confusionmatrix(a, b):

Firstly, compare the target of a[i] and b[i].

Secondly, create a matrix. If the targets of a[i] and b[i] match, add one to the diagonal. If not, add one to (x,y), where x stands for the row that x= b[i].target, y stands for the column that y= a[i].target.

Thirdly, use mat() method to draw the matrix.

3. The error number for each class can be seen from the matrix.

**F5:** ROC curve of NN

1. Call plotROC\_NN()

Firstly, turn the given 9 classes into 2 classes: negative and positive classes. I make class 9 to be the negative class, using 0 to represent the target of them. All the other classes are changed to positive class, using 1 to represent their target. Then calculate both the number of positive instances and the number of negative instances in the test set.

Secondly, calculate the confidence after doing forward propagation on the model.

Thirdly, Sort the test set according to their confidence score.

1. Call the method: ROC(sorted\_pre,sorted\_y\_test\_pre, num\_neg, num\_pos)

The algorithm used in ROC() is from the ppt of model evaluation. (FPR,TPR) are calculated according to the algorithm. Finally, ROC curve will be drawn.

**F6:** **Load CNN model and show results**

The method loadCNN() is called. cnn = joblib.load(‘filename’) is used to load cnn from file. Training ccuracy is presented by calling the method model.accuarcy(). Testing accuracy is displayed by calling model.forward\_propagation().

**F7: Train and save CNN model**

1. call trainCNN() to train CNN model:

Firstly, create the model using the class CNN.

Secondly, add the convolutional layer, maxpool layer, another convolutional layer, another maxpool layer and fully connected layer to the model.

Thirdly, train the dataset using forward propagation and back propagation, iterating n times.

1. save CNN model:

Use joblib.dump(cnn, 'cnnModel.m') to save model into file 'cnnModel.m'.

**F8: Cross validation of CNN**

1. call CrossValidationCNN() to do cross validation on the dataset

However, it may take a long time (around 5 mins) to do cross validation for CNN.

\*The details of this implementation are the same as F3: Cross validation of NN

**F9: Confusion matrix of CNN**

1. Call CMOfCNN():
2. Call confusionmatrix(a, b):

\*The details of this implementation are the same as what I wrote in F4: Confusion matrix of NN

**F10: ROC curve of CNN**

1. Call plotROCCNN():

\*The details of this implementation are the same as F5: ROC curve of NN

**F11: Cross validation, Confusion matrix and ROC curve of traditional models: my knn and sklearn knn.**

\*The details of the implementation of Cross validation and Confusion matrix implementation are the same as F3 and F4.

In case of the ROC curve of traditional models, the confidence of sklearn knn is calculated by calling the method knn.predict\_proba(X\_test). The confidence of my knn is calculated by calling the self.confidence() method in the class KNNClassfier. When neighbor equals to 9, we calculate the number n of the instance which occur most frequently in the label. Then n/9 is the confidence.

**Additional Requirements:**

1. The marker can run my code directly. The training and testing accuracy of functionality f1 can be seen by loading the models directly.
2. Clear annotations and user interface are provided.
3. ***Discussion on the discovery:***
4. Traditional models:

The training and testing accuracy of both traditional models are all above 0.97 in corss validation part. It means the knn models are good when training the digits dataset.

In the confusion matrixs, the numbers are mainly distributed in the diagonal. It means the error number for test dataset is rather low.

The ROC curves are represented at a point (0,1) and then have a line that travels from the bottom left of the plot to the top left and then across the top to the top right. It means that the model is with perfect skill.

In sum, the traditional models is perfect for digits dataset.

1. Deep learning models:

The training and testing accuracy of nn model are around 0.95 in cross validation part. It means the nn model are good when training the digits dataset. The training and testing accuracy of cnn model are around 0.65 in cross validation part. I think it is because of the fact that the number of times I do forward propagation and back propagation on cnn model is low(iteration number=3). The testing accuracy will be increased to 0.8 if the iteration number is 5. However, it will be more time consuming if the time for iteration is increased.

In the confusion matrixs, the numbers are mainly distributed in the diagonal for both models. It means the error number for test dataset is rather low.

The ROC curves of both models are not so skillful. It is above the diagonal line but it does not travel to the top left of the plot.

In sum, the deep learning models need to be trained for more times in order to get a higher testing accuracy.