COMP219 Assignment 2 Checklist:

functionality f1: 50% implemented successfully

functionality f2: 50% implemented successfully

Chosen dataset: Optical recognition of handwritten digits dataset

Dataset information:

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers Min:0 Max:16

- How to run my programme

Open the python file called 'sourcecode.py' in IDE and run it. A simple UI is designed for the testers to see the results of **f1** and **f2.** It will ask you to input integers as the commands.

Instructions:

command	Chosen model	Description
0		Quit
1	deep neural network without convolutional layer	1. Show the detail of trained model' s layer structure 2. Do cross validation 3. Plot the confusion matrix 4. Plot the ROC curve
2	deep neural network with convolutional layer	1. Show the detail of trained model' s layer structure 2. Do cross validation 3. Plot the confusion matrix 4. Plot the ROC curve
3	my KNN model	1. Do cross validation 2. Plot the confusion matrix
4	sklearn KNN model	1. Do cross validation 2. Plot the confusion matrix

- Software dependencies

```
from sklearn import datasets
from sklearn.metrics import roc_curve, auc
from sklearn.neighbors import KNeighborsClassifier as sklearnKNN
from tensorflow import keras
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
```

1. importing 'datasets' is used for loading the handwritten digits dataset

2. importing 'roc_curve' and 'auc' for calculating false positive rate, true positive rate and accuracy

3. importing the KNN learning algorithm from scikit-learn

4. importing ‘keras’ for constructing deep neural networks

5. importing ‘numpy’ for array-processing

6. importing ‘Counter’ is used to find out the most frequent element in an array

7. importing ‘matpltlib’ for plotting confusion matrix and ROC curve

8. importing ‘MultipleLocator’ for set the location of x, y axis

- How the functionalities and additional requirements are implemented

  1. functionality f1

  (Codes below cannot be found in ‘sourcecode. py’, models are constructed beforehand)

  The structure deep neural network without convolutional layer:

```
model=keras.Sequential()
model.add(keras.layers.Flatten(input_shape=(8, 8)))
model.add(keras.layers.Dense(64, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

Layer number	Layer type	Description
1	Flatten	Input layer: Reshape the input shape into one-dimension matrix
2	Dense	Hidden layer: 64 neurons fully connected
3	Dense	Output layer: 10 neurons stand for 10 digits classes

  The structure deep neural network with convolutional layer:

```
model=keras.Sequential()
model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(8,8,1), padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Flatten(input_shape=(4, 4)))
model.add(keras.layers.Dense(10, activation="softmax"))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

Layer number	Layer type	Description
1	Convolutional	Convolutional layer: kernel size is 3*3, output size is 8*8
2	Pooling	Pooling layer: pool size is 2*2, output size is 4*4
3	Flatten	Input layer: Reshape the input shape into one-dimension matrix
4	Dense	Output layer: 10 neurons stand for 10 digits classes

  2. functionality f2

  - Cross validation: Methods ‘DNN_cross_validation()’ and ‘KNN_cross_validation()’ are used for doing cross validation. Through there are some slight differences in these two methods, the concepts are the same:

Cross validation steps	Description
1	Load the chosen model
2	Split the dataset into 5 parts. Select one subset as test set, and the rest will be used as training set each time.
3	Normalize training and validation data samples to fit the input shape of different models. Also, all the features in the matrix are converted into decimals between 0 and 1.
4	Train the model with training samples.
5	Predict the validation samples and record the results
6	Add all the errors detected in each cross validation together and then accuracy can be calculated.

- Confusion matrix: Methods ‘confusion_matrix()’ and ‘plot_CM()’ are used for calculating and plotting confusion matrix. The data source of confusion matrix is based on the last cross validation over the same model.

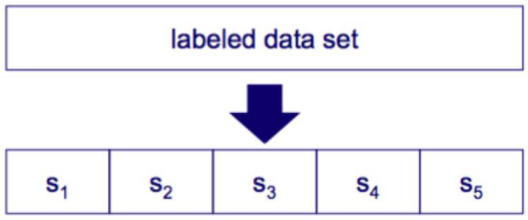
Plot confusion matrix steps	Description
1	Initialize a 10*10 matrix with zeros
2	According to the results of the last cross validation, update the confusion matrix with each pair of expectation and prediction. For example, if expectation=0 and prediction=0, then CM[0][0]+=1.
3	Plot the confusion matrix as a figure.

- ROC curve for one class vs. all other classes: Method ‘plot_ROC()’ is used for plotting ROC curve. Since classification with KNN model does not have confidence probability, this function only applies to DNN models. Also, the data source of ROC curve is based on the last cross validation over the same model.

Plot ROC curve steps	Description
1	Since ROC curve in this assignment is one class vs. all other classes, one class is selected as the positive while the others are regarded as the negative. Therefore, I rewrite the validation set with only 1s and 0s. 1 stands for the current positive class, 0s represents the negative.
2	Use method ‘roc_curve()’ to calculate true and false positive rate. The rewritten validation set will be used for distinguishing positive and negative classes in this method.
3	Plot the ROC curve for each digit class with corresponding true and false positive rate.

- Evaluation of the models

  1. Cross validation



5 subsets are divided equally, each size is 20% of the total sample size. Subset S_i is chosen as the validation set at time i. Additionally, ‘epochs’ for deep neural networks is 5 in this case.

models / errors / time	deep neural network without convolutional layer	deep neural network with convolutional layer	my KNN model	sklearn KNN model
1	44	29	17	18
2	41	34	14	13
3	41	38	12	13
4	18	12	8	7
5	45	18	16	13
Total errors	189	161	67	64
Accuracy	89.48%	91.04%	96.27%	96.44%

2. Confusion matrix
Confusion matrix will show up when running the program.

3. ROC Curve
ROC Curve will show up when running the program.

Discussion on the discovery:

In this case, recognizing handwritten digits, KNN model which is called from sklearn library performs the best among all the other models. Its accuracy reaches 96.44% because KNN is good at multi-modal. Additionally, since each handwritten digit is only 8*8 pixels, memory usage and time cost of KNN are both low. If input samples are high-quality images, KNN may not perform that well.
On the other side, deep neural network with convolutional layer perform better than deep neural network without convolutional layer. Obviously, the convolutional and pooling layer play a role in extracting the most important features and refining the image for less calculation. Since the deep neural networks in this case are constructed in the simplest way with only necessary layers and neurons, DNN still have potential to perform much better if additional layers are added to the structure.

- Details of implementation
  Comments are written in the 'sourcecode.py', they will explain the meaning of parameters, variables and methods' function.