นาย วรินทร์ สายปัญญา รหัสนิสิต 6630250435 sec. 870

การบ้าน Image Classification

นำเข้า library modules ที่ต้องใช้ในการสร้าง models

```
[38] from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten
     from tensorflow.keras.losses import SparseCategoricalCrossentropy
     from sklearn.metrics import accuracy_score, log_loss
     from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     import matplotlib.pyplot as plt
```

ประกาศตัวแปร list ไว้เก็บค่า accuracy และ ค่า loss ของ models ทั้งหมด

```
[39] acc_values = []
     loss_values = []
     model_names = ["Neural Network", "Decision Tree", "KNN"]
```

สร้างตัว model Neural Network

```
[40] # กำหนดตัว object ของ model
     model = Sequential()

     # เพิ่ม layers ให้ models
     model.add(Flatten(input_shape=(28, 28)))
     model.add(Dense(128, activation='relu'))
     model.add(Dense(10, activation='softmax'))
```

Compile ตัว model

```
[41] # compile model โดยเราใช้ ตัว optimizer เป็น adadelta
     model.compile(optimizer='adadelta',
                   loss=SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])
```

train ตัว model

```
[42] # กำหนดจำนวนรอบในการ train อยู่ที่ 30 รอบ
     model.fit(train_images, train_labels, epochs=30)
```

```
Epoch 1/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.1673 - loss: 2.2619
Epoch 2/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.3555 - loss: 1.9224
Epoch 3/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.4952 - loss: 1.6853
Epoch 4/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.5961 - loss: 1.5010
Epoch 5/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.6468 - loss: 1.3504
Epoch 6/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.6574 - loss: 1.2414
Epoch 7/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.6684 - loss: 1.1494
Epoch 8/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.6745 - loss: 1.0815
Epoch 9/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.6825 - loss: 1.0266
Epoch 10/30
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 7ms/step - accuracy: 0.6883 - loss: 0.9799
```

```
[42]  Epoch 19/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.7441 - loss: 0.7757
      Epoch 20/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.7489 - loss: 0.7632
      Epoch 21/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.7520 - loss: 0.7548
      Epoch 22/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7535 - loss: 0.7423
      Epoch 23/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7593 - loss: 0.7320
      Epoch 24/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.7635 - loss: 0.7233
      Epoch 25/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.7688 - loss: 0.7070
      Epoch 26/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7710 - loss: 0.7043
      Epoch 27/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.7737 - loss: 0.6975
      Epoch 28/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.7785 - loss: 0.6903
      Epoch 29/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.7785 - loss: 0.6849
      Epoch 30/30
      1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7801 - loss: 0.6840
      <keras.src.callbacks.history.History at 0x7bb3ccbdae10>
```

คำนวณค่า loss และ ค่า accuracy

```
[43]  # กำหนดค่าให้ 2 ทั้งแปรเพื่อรับค่า loss และ ค่า accuracy
      loss, acc = model.evaluate(test_images,  test_labels, verbose=2)

      # แสดงผลค่า
      print(f"Accuracy: {acc:.2f}")
      print(f"Loss: {loss:.2f}")

      # นำค่า loss และ ค่า accuracy ไปเก็บไว้ใน list เพื่อนำไป plot graph เส้นเปรียบเทียบ
      acc_values.append(acc)
      loss_values.append(loss)

      313/313 - 1s - 2ms/step - accuracy: 0.7696 - loss: 0.6894
      Accuracy: 0.77
      Loss: 0.69
```

เตรียมข้อมูล train และ test ให้กับ ML Models

```
[44]  x_train, x_test, y_train, y_test = train_test_split(train_images, train_labels, test_size=0.2)
```

เปลี่ยน shape ของข้อมูล train และ test ใหม่ให้เป็น array 2 มิติเพื่อนำข้อมูลไป train

```
[45]  # ไม่สามารถ train ข้อมูล array 3 มิติได้ต้องแปลงเป็น array 2 มิติ ก่อน
      x_train = x_train.reshape(x_train.shape[0], -1)
      x_test = x_test.reshape(x_test.shape[0], -1)
      y_train = y_train.reshape(y_train.shape[0], -1)
      y_test = y_test.reshape(y_test.shape[0], -1)
```

ML Model อันแรกที่ใช้คือ Decision Tree Model

train model สำหรับ model DecisionTreee

```
[46]  dc_model = DecisionTreeClassifier()
      dc_model.fit(x_train, y_train)

      DecisionTreeClassifier
      DecisionTreeClassifier()
```

ให้ model ทำนายว่าควรอยู่ class ไหน

```
[47] # model ทำนาย
     y_pred = dc_model.predict(x_test)

     # แสดงผลการทำนาย
     print(np.argmax(y_pred[0]))
     print(class_names[np.argmax(y_pred[0])])
```

```
0
T-shirt/top
```

ให้ model ทำนายค่าความน่าจะเป็นของการทำนาย

```
[48] # model ทำนายค่าความน่าจะเป็น
     y_pred_prob = dc_model.predict_proba(x_test)
     # แสดงผลค่าที่ได้มา
     print(np.argmax(y_pred_prob))
```

```
4
```

คำนวณค่า accuracy และ ค่า loss ของ decision tree

```
[49] # คำนวณค่า accuracy และค่า loss ของตัว model
     acc = accuracy_score(y_test, y_pred)
     loss = log_loss(y_test, y_pred_prob)

     # แสดงผลค่า accuracy และค่า loss
     print(f"Accuracy: {acc:.2f}")
     print(f"Loss: {loss:.2f}")

     # เก็บค่าที่คำนวณมาได้เพิ่มเข้าไปใน list
     acc_values.append(acc)
     loss_values.append(loss)
```

```
Accuracy: 0.80
Loss: 7.26
```

ML Model อันที่สองที่ใช้คือ KNN

train model สำหรับ model KNN

```
[50] knn_model = KNeighborsClassifier(n_neighbors=5, p=2)
     knn_model.fit(x_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/neighbors/_classification.py:239: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y
  return self._fit(X, y)
```

KNeighborsClassifier

KNeighborsClassifier()

ให้ model ทำนายว่าอยู่ class ไหน

```
[51] # model ทำนาย
     y_pred = knn_model.predict(x_test)

     # แสดงผลการทำนาย
     print(np.argmax(y_pred[0]))
     print(class_names[np.argmax(y_pred[0])])
```

```
0
T-shirt/top
```

ให้ model ทำนายค่าความน่าจะเป็นของการทำนาย

```
[52] # model ทำนายค่าความน่าจะเป็น
     y_pred_prob = knn_model.predict_proba(x_test)
     # แสดงผลค่าที่ได้มา
     print(np.argmax(y_pred_prob))
```

```
2
```

ให้ model ทำนายว่าอยู่ class ไหน

```
[51] # model ทำนาย
     y_pred = knn_model.predict(x_test)

     # แสดงผลการทำนาย
     print(np.argmax(y_pred[0]))
     print(class_names[np.argmax(y_pred[0])])
```

```
0
T-shirt/top
```

ให้ model ทำนายค่าความน่าจะเป็นของการทำนาย

```
[52] # model ทำนายค่าความน่าจะเป็น
     y_pred_prob = knn_model.predict_proba(x_test)
     # แสดงผลค่าที่ได้มา
     print(np.argmax(y_pred_prob))
```

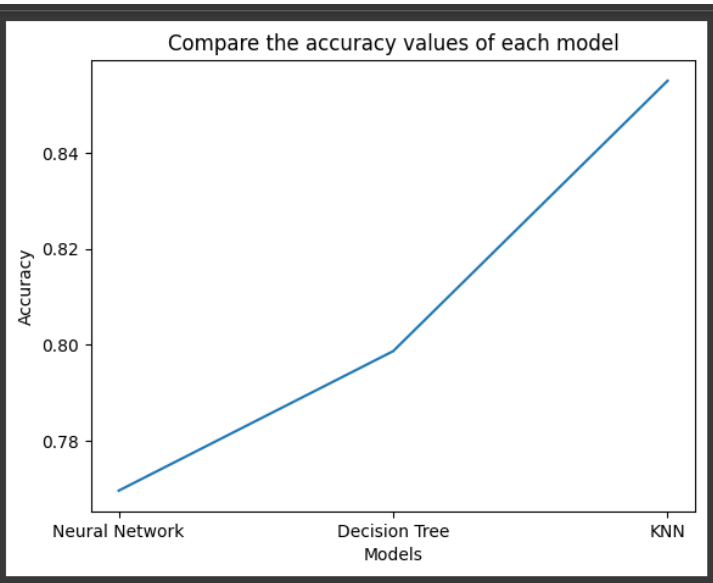
```
2
```

plot กราฟเส้นเปรียบเทียบวัดค่า accuracy และ ค่า loss ของทั้ง 3 models

กราฟเปรียบเทียบค่า accuracy

```
[54] plt.title("Compare the accuracy values of each model")
     plt.plot(model_names, acc_values)
     plt.xlabel("Models")
     plt.ylabel("Accuracy")
     plt.show()
```

[54]



กราฟเปรียบเทียบค่า loss

```
plt.title("Compare the loss values of each model")
plt.plot(model_names, loss_values)
plt.xlabel("Models")
plt.ylabel("Loss")
plt.show()
```

Compare the loss values of each model