wyq52

MINI PROJECT

Task 1--Preprocess the images in order to produce satisfying datasets:

In [32]:
```
#import the packages:
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import sklearn
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from keras import initializers
```

In [33]:
```
image_path = "/Users/wangyingqi/Desktop/Tom and Jerry/tom_and_jerry/tom_and_jerry"

#create dataset of training
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    image_path ,
    labels = "inferred",
    label_mode = "int",
    batch_size =32,
    image_size =(128,128),
    shuffle = True,
    seed = 123,
    validation_split = 0.2,
    subset = "training",
)
#create dataset of valiging
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    image_path ,
    labels = "inferred",
    label_mode = "int",
    batch_size =32,
    image_size =(128,128),
    shuffle = True,
    seed = 123,
    validation_split = 0.2,
    subset = "validation",
)
```

Result:

```
Found 5478 files belonging to 4 classes.
Using 4383 files for training.
Found 5478 files belonging to 4 classes.
Using 1095 files for validation.
```

Task2(first)--Construct CNN using different layers:

CNN

```
#build the model of CNN:
nor_lay = tf.keras.layers.Rescaling(1./255,input_shape =(128,128,3))

model = Sequential()
model.add(nor_lay)
model.add(layers.Conv2D(filters =32,kernel_size = (5,5),strides =2,padding = 'same',activation ='relu'))
model.add(layers.MaxPooling2D(pool_size = (5,5),strides = 2,padding = 'same'))
model.add(layers.Conv2D(filters =32,kernel_size = (5,5),strides =2,padding = 'same',activation ='relu'))
model.add(layers.MaxPooling2D(pool_size = (5,5),strides = 2,padding = 'same'))
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())
model.add(layers.Dropout(0.2))
model.add(layers.Dense(units = 64,activation ='relu'))
model.add(layers.Dense(units =4,kernel_initializer =initializers.random_normal(stddev=0.01),activation ='softmax'))

model.compile(optimizer= tf.optimizers.Adam(learning_rate =0.001),
              loss =tf.keras.losses.SparseCategoricalCrossentropy(from_logits = False),
              metrics = 'accuracy',
              )

epoch_num = 30
history = model.fit(train_ds,validation_data = test_ds,epochs = epoch_num)
```

Result:

```
Epoch 1/30
137/137 [==============================] - 10s 70ms/step - loss: 1.3372 -
accuracy: 0.3541 - val_loss: 1.2602 - val_accuracy: 0.4274
Epoch 2/30
137/137 [==============================] - 10s 70ms/step - loss: 1.1498 -
accuracy: 0.4914 - val_loss: 0.9832 - val_accuracy: 0.5653
Epoch 3/30
137/137 [==============================] - 10s 72ms/step - loss: 0.9582 -
accuracy: 0.5909 - val_loss: 0.9590 - val_accuracy: 0.5954
Epoch 4/30
137/137 [==============================] - 11s 76ms/step - loss: 0.8337 -
accuracy: 0.6523 - val_loss: 0.7792 - val_accuracy: 0.6849
Epoch 5/30
137/137 [==============================] - 11s 77ms/step - loss: 0.7463 -
accuracy: 0.6966 - val_loss: 0.7147 - val_accuracy: 0.7306
Epoch 6/30
137/137 [==============================] - 11s 82ms/step - loss: 0.6661 -
accuracy: 0.7317 - val_loss: 0.7416 - val_accuracy: 0.7105
Epoch 7/30
137/137 [==============================] - 12s 83ms/step - loss: 0.5786 -
accuracy: 0.7643 - val_loss: 0.5818 - val_accuracy: 0.7854
Epoch 8/30
137/137 [==============================] - 12s 83ms/step - loss: 0.5541 -
accuracy: 0.7823 - val_loss: 0.5741 - val_accuracy: 0.7817
Epoch 9/30
137/137 [==============================] - 12s 83ms/step - loss: 0.4878 -
accuracy: 0.8113 - val_loss: 0.5561 - val_accuracy: 0.7982
Epoch 10/30
137/137 [==============================] - 12s 84ms/step - loss: 0.4592 -
accuracy: 0.8202 - val_loss: 0.5450 - val_accuracy: 0.8183
Epoch 11/30
137/137 [==============================] - 12s 84ms/step - loss: 0.4176 -
accuracy: 0.8392 - val_loss: 0.4475 - val_accuracy: 0.8420
```

```
Epoch 20/30
137/137 [==============================] - 13s 93ms/step - loss: 0.2022 - accuracy: 0.9233 - val_loss: 0.4462 - val_a
ccuracy: 0.8630
Epoch 21/30
137/137 [==============================] - 12s 88ms/step - loss: 0.1848 - accuracy: 0.9327 - val_loss: 0.3988 - val_a
ccuracy: 0.8749
Epoch 22/30
137/137 [==============================] - 12s 88ms/step - loss: 0.1880 - accuracy: 0.9302 - val_loss: 0.5145 - val_a
ccuracy: 0.8447
Epoch 23/30
137/137 [==============================] - 13s 93ms/step - loss: 0.1873 - accuracy: 0.9279 - val_loss: 0.3930 - val_a
ccuracy: 0.8758
Epoch 24/30
137/137 [==============================] - 13s 93ms/step - loss: 0.1557 - accuracy: 0.9441 - val_loss: 0.4074 - val_a
ccuracy: 0.8721
Epoch 25/30
137/137 [==============================] - 13s 93ms/step - loss: 0.1535 - accuracy: 0.9443 - val_loss: 0.4322 - val_a
ccuracy: 0.8639
Epoch 26/30
137/137 [==============================] - 13s 93ms/step - loss: 0.1412 - accuracy: 0.9482 - val_loss: 0.4363 - val_a
ccuracy: 0.8712
Epoch 27/30
137/137 [==============================] - 12s 89ms/step - loss: 0.1485 - accuracy: 0.9482 - val_loss: 0.4461 - val_a
ccuracy: 0.8712
Epoch 28/30
137/137 [==============================] - 13s 90ms/step - loss: 0.1461 - accuracy: 0.9496 - val_loss: 0.4478 - val_a
ccuracy: 0.8475
Epoch 29/30
137/137 [==============================] - 13s 90ms/step - loss: 0.1310 - accuracy: 0.9525 - val_loss: 0.4202 - val_a
ccuracy: 0.8840
Epoch 30/30
137/137 [==============================] - 13s 92ms/step - loss: 0.1218 - accuracy: 0.9546 - val_loss: 0.4395 - val_a
ccuracy: 0.8740
```

Depthwise Separable 2D CNN

```
input_size=(128,128,3)
nor_lay = tf.keras.layeSrs.Rescaling(1./255,input_shape =input_size)

model = Sequential()
model.add(nor_lay)
model.add(layers. SeparableConv2D(filters =32,kernel_size = (3,3),strides =2,activation ='relu',padding = 'same',depth_
model.add(layers.MaxPooling2D(pool_size = (3,3),strides = 2,padding = 'same'))
model.add(layers. SeparableConv2D(filters =32,kernel_size = (3,3),strides =2,activation ='relu',padding = 'same',depth_
model.add(layers.MaxPooling2D(pool_size = (3,3),strides = 2,padding = 'same'))
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())
model.add(layers.Dropout(0.2))
model.add(layers.Dense(units = 64,activation ='relu'))
model.add(layers.Dense(units =4,kernel_initializer =initializers.random_normal(stddev=0.01),activation ='softmax'))

model.compile(optimizer= tf.optimizers.Adam(learning_rate =0.01),
              loss =tf.keras.losses.SparseCategoricalCrossentropy(from_logits = False),
              metrics = 'accuracy',
             )

epoch_num = 25
history = model.fit(train_ds,validation_data = test_ds,epochs = epoch_num)
```

Result:

```
Epoch 1/25
137/137 [==============================] - 7s 50ms/step - loss: 1.3443 -
accuracy: 0.3415 - val_loss: 1.3155 - val_accuracy: 0.3863
Epoch 2/25
137/137 [==============================] - 7s 52ms/step - loss: 1.2118 -
accuracy: 0.4565 - val_loss: 1.0869 - val_accuracy: 0.4941
Epoch 3/25
137/137 [==============================] - 8s 53ms/step - loss: 1.0469 -
accuracy: 0.5558 - val_loss: 1.0455 - val_accuracy: 0.5753
Epoch 4/25
137/137 [==============================] - 8s 55ms/step - loss: 0.9503 -
accuracy: 0.5991 - val_loss: 0.9613 - val_accuracy: 0.6155
Epoch 5/25
137/137 [==============================] - 8s 57ms/step - loss: 0.8836 -
accuracy: 0.6407 - val_loss: 0.9258 - val_accuracy: 0.6311
Epoch 6/25
137/137 [==============================] - 8s 55ms/step - loss: 0.7875 -
accuracy: 0.6742 - val_loss: 0.8803 - val_accuracy: 0.6630
Epoch 7/25
137/137 [==============================] - 8s 54ms/step - loss: 0.7349 -
accuracy: 0.7086 - val_loss: 0.8767 - val_accuracy: 0.6548
Epoch 8/25
137/137 [==============================] - 8s 54ms/step - loss: 0.6901 -
accuracy: 0.7317 - val_loss: 0.7865 - val_accuracy: 0.7068
Epoch 9/25
137/137 [==============================] - 8s 55ms/step - loss: 0.6493 -
accuracy: 0.7390 - val_loss: 0.7878 - val_accuracy: 0.6932
Epoch 10/25
137/137 [==============================] - 8s 55ms/step - loss: 0.6086 -
accuracy: 0.7582 - val_loss: 0.8611 - val_accuracy: 0.6740
Epoch 11/25
137/137 [==============================] - 8s 56ms/step - loss: 0.5966 -
accuracy: 0.7725 - val_loss: 0.8141 - val_accuracy: 0.7005
```

```
Epoch 15/25
137/137 [==============================] - 8s 56ms/step - loss: 0.5058 - accuracy: 0.7983 - val_loss: 0.7710 - val_ac
curacy: 0.7461
Epoch 16/25
137/137 [==============================] - 8s 56ms/step - loss: 0.4917 - accuracy: 0.8074 - val_loss: 0.7662 - val_ac
curacy: 0.7379
Epoch 17/25
137/137 [==============================] - 8s 57ms/step - loss: 0.4637 - accuracy: 0.8216 - val_loss: 0.8241 - val_ac
curacy: 0.7196
Epoch 18/25
137/137 [==============================] - 8s 57ms/step - loss: 0.4459 - accuracy: 0.8284 - val_loss: 0.9563 - val_ac
curacy: 0.6950
Epoch 19/25
137/137 [==============================] - 8s 58ms/step - loss: 0.4446 - accuracy: 0.8348 - val_loss: 0.7835 - val_ac
curacy: 0.7507
Epoch 20/25
137/137 [==============================] - 8s 57ms/step - loss: 0.4114 - accuracy: 0.8437 - val_loss: 0.8138 - val_ac
curacy: 0.7406
Epoch 21/25
137/137 [==============================] - 8s 57ms/step - loss: 0.4157 - accuracy: 0.8444 - val_loss: 0.8076 - val_ac
curacy: 0.7553
Epoch 22/25
137/137 [==============================] - 8s 60ms/step - loss: 0.4009 - accuracy: 0.8496 - val_loss: 0.9526 - val_ac
curacy: 0.7151
Epoch 23/25
137/137 [==============================] - 8s 60ms/step - loss: 0.4453 - accuracy: 0.8382 - val_loss: 0.7576 - val_ac
curacy: 0.7562
Epoch 24/25
137/137 [==============================] - 8s 60ms/step - loss: 0.3873 - accuracy: 0.8535 - val_loss: 0.8897 - val_ac
curacy: 0.7443
Epoch 25/25
137/137 [==============================] - 8s 60ms/step - loss: 0.3844 - accuracy: 0.8556 - val_loss: 0.8647 - val_ac
curacy: 0.7452
```

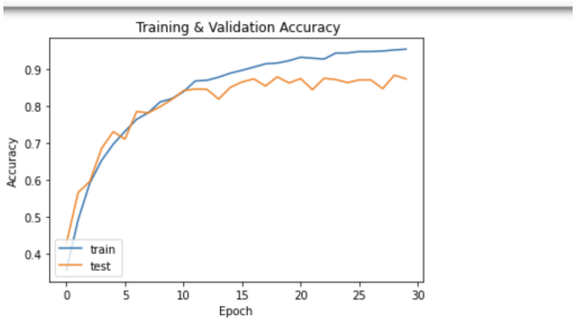
Task3(first)–Show the result of accuracy and loss of training and loss by graphs:

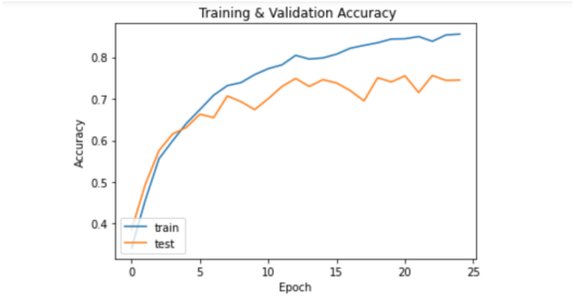
CNN

```
#make the image of the accuracy & loss of training and validation
#model compile first

plt.plot(range(epoch_num),history.history['accuracy'])
plt.plot(range(epoch_num),history.history['val_accuracy'])
plt.title(' Training & Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='lower left')
plt.show()


plt.plot(range(epoch_num),history.history['loss'])
plt.plot(range(epoch_num),history.history['val_loss'])
plt.title('Training & Validation Loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Depthwise Separable 2D CNN

```
#make the image of the accuracy & loss of training and validation
#model compile first

plt.plot(range(epoch_num),history.history['accuracy'])
plt.plot(range(epoch_num),history.history['val_accuracy'])
plt.title(' Training & Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='lower left')
plt.show()


plt.plot(range(epoch_num),history.history['loss'])
plt.plot(range(epoch_num),history.history['val_loss'])
plt.title('Training & Validation Loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Task2(second)**--Construct CNN using different layers**:

CNN

```
In [*]: nor_lay = tf.keras.layers.Rescaling(1./255,input_shape =(128,128,3))

        model = Sequential()
        model.add(nor_lay)
        model.add(layers.Conv2D(filters =32,kernel_size = (5,5),strides =2,padding = 'same',activation ='relu'))
        model.add(layers.MaxPooling2D(pool_size = (5,5),strides = 2,padding = 'same'))
        model.add(layers.Conv2D(filters =32,kernel_size = (5,5),strides =2,padding = 'same',activation ='relu'))
        model.add(layers.MaxPooling2D(pool_size = (5,5),strides = 2,padding = 'same'))
        model.add(layers.Dropout(0.2))
        model.add(layers.Flatten())
        model.add(layers.Dropout(0.2))
        model.add(layers.Dense(units = 128,activation ='relu'))
        model.add(layers.Dense(units =4,kernel_initializer =initializers.random_normal(stddev=0.01),activation ='softmax'))

        model.compile(optimizer= tf.optimizers.Adam(learning_rate =0.001),
                      loss =tf.keras.losses.SparseCategoricalCrossentropy(from_logits = False),
                      metrics = 'accuracy',
                      )

        epoch_num = 100
        history = model.fit(train_ds,validation_data = test_ds,epochs = epoch_num)
```

```
Epoch 1/100
137/137 [==============================] - 11s 80ms/step - loss: 1.3312 - accuracy: 0.3557 - val_loss: 1.2390 - val_a
ccuracy: 0.4384
Epoch 2/100
137/137 [==============================] - 11s 82ms/step - loss: 1.0984 - accuracy: 0.5108 - val_loss: 0.8915 - val_a
ccuracy: 0.6292
Epoch 3/100
137/137 [==============================] - 12s 83ms/step - loss: 0.8784 - accuracy: 0.6308 - val_loss: 0.7725 - val_a
ccuracy: 0.6831
Epoch 4/100
137/137 [==============================] - 12s 84ms/step - loss: 0.7401 - accuracy: 0.6961 - val_loss: 0.7335 - val_a
ccuracy: 0.7123
Epoch 5/100
137/137 [==============================] - 12s 87ms/step - loss: 0.6199 - accuracy: 0.7614 - val_loss: 0.7220 - val_a
ccuracy: 0.7251
Epoch 6/100
137/137 [==============================] - 12s 88ms/step - loss: 0.5390 - accuracy: 0.7890 - val_loss: 0.5923 - val_a
ccuracy: 0.7699
Epoch 7/100
137/137 [==============================] - 12s 89ms/step - loss: 0.5016 - accuracy: 0.8022 - val_loss: 0.5637 - val_a
ccuracy: 0.7954
Epoch 8/100
137/137 [==============================] - 13s 89ms/step - loss: 0.4463 - accuracy: 0.8266 - val_loss: 0.5612 - val_a
ccuracy: 0.7927
Epoch 9/100
137/137 [==============================] - 13s 90ms/step - loss: 0.3990 - accuracy: 0.8471 - val_loss: 0.4954 - val_a
ccuracy: 0.8174
Epoch 10/100
137/137 [==============================] - 13s 90ms/step - loss: 0.3613 - accuracy: 0.8631 - val_loss: 0.4995 - val_a
ccuracy: 0.8311
Epoch 11/100
137/137 [==============================] - 13s 91ms/step - loss: 0.3339 - accuracy: 0.8684 - val_loss: 0.5527 - val_a
ccuracy: 0.8128
```

```
accuracy: 0.8831
Epoch 95/100
137/137 [==============================] - 15s 109ms/step - loss: 0.0221 - accuracy: 0.9920 - val_loss: 0.7138 - val_
accuracy: 0.8831
Epoch 96/100
137/137 [==============================] - 14s 104ms/step - loss: 0.0183 - accuracy: 0.9941 - val_loss: 0.6926 - val_
accuracy: 0.8932
Epoch 97/100
137/137 [==============================] - 14s 102ms/step - loss: 0.0451 - accuracy: 0.9856 - val_loss: 0.6741 - val_
accuracy: 0.8895
Epoch 98/100
137/137 [==============================] - 14s 102ms/step - loss: 0.0226 - accuracy: 0.9925 - val_loss: 0.6032 - val_
accuracy: 0.8904
Epoch 99/100
137/137 [==============================] - 14s 102ms/step - loss: 0.0433 - accuracy: 0.9872 - val_loss: 0.5957 - val_
accuracy: 0.8858
Epoch 100/100
137/137 [==============================] - 14s 102ms/step - loss: 0.0306 - accuracy: 0.9888 - val_loss: 0.6388 - val_
accuracy: 0.8840
```

Depthwise Separable 2D CNN

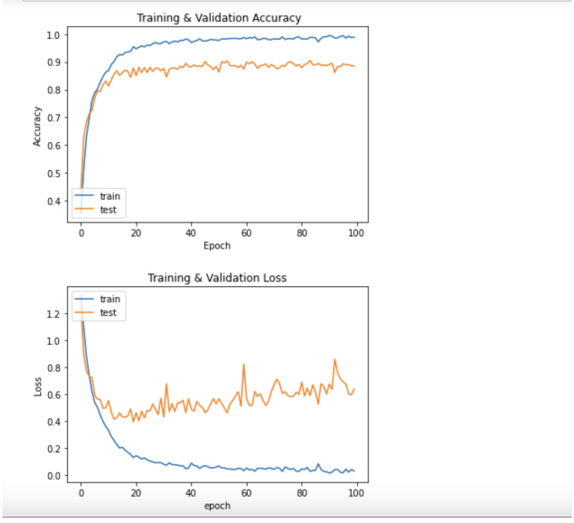
```
In [*]: input_size=(128,128,3)
        nor_lay = tf.keras.layers.Rescaling(1./255,input_shape =input_size)

        model = Sequential()
        model.add(nor_lay)
        model.add(layers. SeparableConv2D(filters =32,kernel_size = (3,3),strides =
        model.add(layers.MaxPooling2D(pool_size = (3,3),strides = 2,padding = 'same
        model.add(layers.Dropout(0.2))
        model.add(layers. SeparableConv2D(filters =32,kernel_size = (3,3),strides =
        model.add(layers.MaxPooling2D(pool_size = (3,3),strides = 2,padding = 'same
        model.add(layers.Dropout(0.2))
        model.add(layers.Flatten())
        model.add(layers.Dropout(0.2))
        model.add(layers.Dense(units = 64,activation ='relu'))
        model.add(layers.Dense(units =4,kernel_initializer =initializers.random_nor

        model.compile(optimizer= tf.optimizers.Adam(learning_rate =0.001),
                      loss =tf.keras.losses.SparseCategoricalCrossentropy(from_log
                      metrics = 'accuracy',
                      )

        epoch_num = 100
        history = model.fit(train_ds,validation_data = test_ds,epochs = epoch_num)
```

```
137/137 [==============================] - 9s 64ms/step - loss: 0.1763 -
accuracy: 0.9338 - val_loss: 0.4580 - val_accuracy: 0.8621
Epoch 53/100
137/137 [==============================] - 9s 63ms/step - loss: 0.1732 -
accuracy: 0.9361 - val_loss: 0.4878 - val_accuracy: 0.8493
Epoch 54/100
137/137 [==============================] - 9s 63ms/step - loss: 0.1785 -
accuracy: 0.9316 - val_loss: 0.4850 - val_accuracy: 0.8557
Epoch 55/100
137/137 [==============================] - 9s 63ms/step - loss: 0.1723 -
accuracy: 0.9352 - val_loss: 0.4705 - val_accuracy: 0.8575
Epoch 56/100
137/137 [==============================] - 9s 64ms/step - loss: 0.1847 -
accuracy: 0.9316 - val_loss: 0.4710 - val_accuracy: 0.8639
Epoch 57/100

137/137 [==============================] - 9s 63ms/step - loss: 0.1686 -
accuracy: 0.9391 - val_loss: 0.5042 - val_accuracy: 0.8548
```

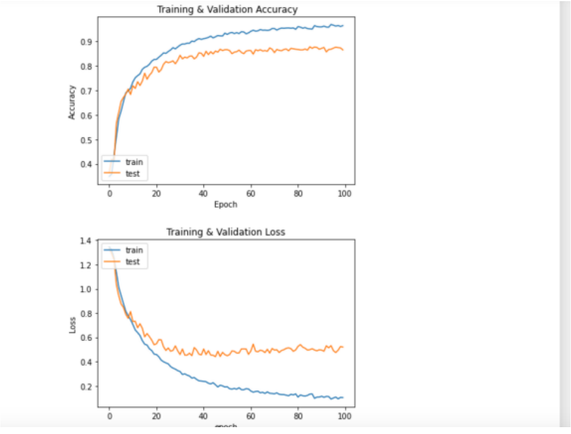
```
accuracy: 0.9576 - val_loss: 0.5130 - val_accuracy: 0.8667
Epoch 95/100
137/137 [==============================] - 10s 68ms/step - loss: 0.0909 -
accuracy: 0.9687 - val_loss: 0.5315 - val_accuracy: 0.8676
Epoch 96/100
137/137 [==============================] - 10s 68ms/step - loss: 0.0998 -
accuracy: 0.9655 - val_loss: 0.4901 - val_accuracy: 0.8721
Epoch 97/100
137/137 [==============================] - 10s 71ms/step - loss: 0.1077 -
accuracy: 0.9619 - val_loss: 0.4743 - val_accuracy: 0.8758
Epoch 98/100
137/137 [==============================] - 10s 73ms/step - loss: 0.0925 -
accuracy: 0.9644 - val_loss: 0.4939 - val_accuracy: 0.8740
Epoch 99/100
137/137 [==============================] - 10s 70ms/step - loss: 0.1051 -
accuracy: 0.9605 - val_loss: 0.5241 - val_accuracy: 0.8731
Epoch 100/100
137/137 [==============================] - 10s 69ms/step - loss: 0.1033 -
accuracy: 0.9640 - val_loss: 0.5196 - val_accuracy: 0.8648
```

Task3(second)--**Show the result of accuracy and loss of training and loss by graphs:**

CNN:



Depthwise Separable 2D CNN



Task 4--Evaluate the test data

CNN

```
# Q4:evaluate the test data:
print('    Evaluate on test data:   ')
result = model.evaluate(test_ds, verbose=1)
print("test loss & test acc :",result)
```

```
    Evaluate on test data:
35/35 [==============================] - 1s 29ms/step - loss: 0.6388 - ac
curacy: 0.8840
test loss & test acc : [0.638752818107605, 0.8840182423591614]
```

Depthwise Separable 2D CNN

```
[103]: # Q4:evaluate the test data:
print('    Evaluate on test data:   ')
result = model.evaluate(test_ds, verbose=1)
print("test loss & test acc :",result)
```

```
    Evaluate on test data:
35/35 [==============================] - 1s 23ms/step - loss: 0.5196 - ac
curacy: 0.8648
test loss & test acc : [0.5195709466934204, 0.8648402094841003]
```

Task5--Make predictions based on the test data and show the differences between them

CNN

```
#Q5:predict the test data:
print("Predict on test data")
prediction = model.predict(test_ds,verbose =1)
l = len(prediction)
print(l)
print(prediction)
```

```
Predict on test data
35/35 [==============================] - 1s 30ms/step
1095
[[2.2494394e-04 9.8435294e-06 9.9976522e-01 2.1537059e-09]
 [9.9976069e-01 1.0757259e-14 2.3934753e-04 1.7938131e-12]
 [1.0123175e-06 2.7153696e-06 9.9999630e-01 4.4905564e-14]
 ...
 [8.8705063e-02 7.1082432e-03 6.8044406e-01 2.2374269e-01]
 [7.9069395e-07 5.1162505e-09 9.9999917e-01 2.1302496e-13]
 [1.0000000e+00 2.6187127e-11 2.6928609e-08 9.7359308e-12]]
```

Depthwise Separable 2D CNN

```
[104]: #Q5:predict the test data:
print("Predict on test data")
prediction = model.predict(test_ds,verbose =1)
l = len(prediction)
print(l)
print(prediction)
```

```
Predict on test data
35/35 [==============================] - 1s 23ms/step
1095
[[9.4212782e-01 4.7846473e-04 5.1616821e-02 5.7768626e-03]
 [9.8008686e-06 9.9998522e-01 3.0615247e-06 1.8585951e-06]
 [1.4719586e-01 2.6710596e-04 8.5239762e-01 1.3934681e-04]
 ...
 [1.1376528e-07 9.9895787e-01 9.6456148e-04 7.7445213e-05]
 [2.2560569e-02 4.4160625e-01 1.7993765e-02 5.1783937e-01]
 [9.9771857e-01 6.4105721e-09 4.5359274e-04 1.8277861e-03]]
```

```
prediction_list=[]
for i in prediction:
    t = np.array(i)
    mid = np.argmax(t)
    prediction_list.append(mid)
print(len(prediction_list))
```

```
1095
```

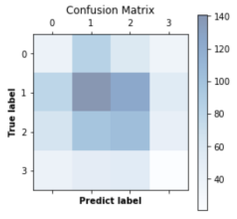
```
prediction_list=[]
for i in prediction:
    t = np.array(i)
    mid = np.argmax(t)
    prediction_list.append(mid)
print(len(prediction_list))
```

```
1095
```

```
test_ds_list = []
test_labels =  np.array([])
for x, y in test_ds:
    test_labels = np.concatenate([test_labels, y.numpy()])
test_labels.shape
print(len(test_labels))
for i in test_labels:
    test_ds_list.append(i)


#make the confusion matrix:
print("0-->jerry")
print("1-->tom")
print("2-->tom_jerry_0")
print("3-->tom_jerry_1")
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(test_ds_list,prediction_list)
plt.matshow(matrix,cmap=plt.cm.Blues,alpha = 0.9)
plt.colorbar()
plt.ylabel("True label",fontweight='bold')
plt.xlabel("Predict label",fontweight='bold')
plt.title("Confusion Matrix")
plt.show()
```

Result

```
1095
0-->jerry
1-->tom
2-->tom_jerry_0
3-->tom_jerry_1
```

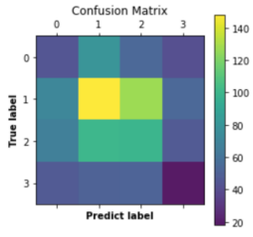



```
test_ds_list = []
test_labels =  np.array([])
for x, y in test_ds:
    test_labels = np.concatenate([test_labels, y.numpy()])
test_labels.shape
print(len(test_labels))
for i in test_labels:
    test_ds_list.append(i)


#make the confusion matrix:
print("0-->jerry")
print("1-->tom")
print("2-->tom_jerry_0")
print("3-->tom_jerry_1")
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(test_ds_list,prediction_list)
plt.matshow(matrix,alpha = 0.9)
plt.colorbar()
plt.ylabel("True label",fontweight='bold')
plt.xlabel("Predict label",fontweight='bold')
plt.title("Confusion Matrix")
plt.show()
```

```
1095
0-->jerry
1-->tom
2-->tom_jerry_0
3-->tom_jerry_1
```

Result:


```
1095
0-->jerry
1-->tom
2-->tom_jerry_0
3-->tom_jerry_1
```


```
]:
```

---model.summary—

```
02]: #model.summary:
     model.save('m.h5')
     model.summary()
```

CNN	Depthwise Separable 2D CNN

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_15 (Rescaling)    (None, 128, 128, 3)       0

 conv2d_30 (Conv2D)          (None, 63, 63, 32)        896

 max_pooling2d_30 (MaxPoolin  (None, 31, 31, 32)       0
 g2D)

 dropout_41 (Dropout)        (None, 31, 31, 32)        0

 conv2d_31 (Conv2D)          (None, 15, 15, 64)        18496

 max_pooling2d_31 (MaxPoolin  (None, 7, 7, 64)         0
 g2D)

 dropout_42 (Dropout)        (None, 7, 7, 64)          0

 flatten_15 (Flatten)        (None, 3136)              0

 dropout_43 (Dropout)        (None, 3136)              0

 dense_30 (Dense)            (None, 64)                200768

 dense_31 (Dense)            (None, 4)                 260

=================================================================
Total params: 220,420
Trainable params: 220,420
Non-trainable params: 0
```

```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_24 (Rescaling)    (None, 128, 128, 3)       0

 separable_conv2d_6 (Separab  (None, 64, 64, 32)       155
 leConv2D)

 max_pooling2d_42 (MaxPoolin  (None, 32, 32, 32)       0
 g2D)

 dropout_56 (Dropout)        (None, 32, 32, 32)        0

 separable_conv2d_7 (Separab  (None, 16, 16, 32)       1344
 leConv2D)

 max_pooling2d_43 (MaxPoolin  (None, 8, 8, 32)         0
 g2D)

 dropout_57 (Dropout)        (None, 8, 8, 32)          0

 flatten_21 (Flatten)        (None, 2048)              0

 dropout_58 (Dropout)        (None, 2048)              0

 dense_42 (Dense)            (None, 64)                131136

 dense_43 (Dense)            (None, 4)                 260

=================================================================
Total params: 132,895
Trainable params: 132,895
Non-trainable params: 0
```