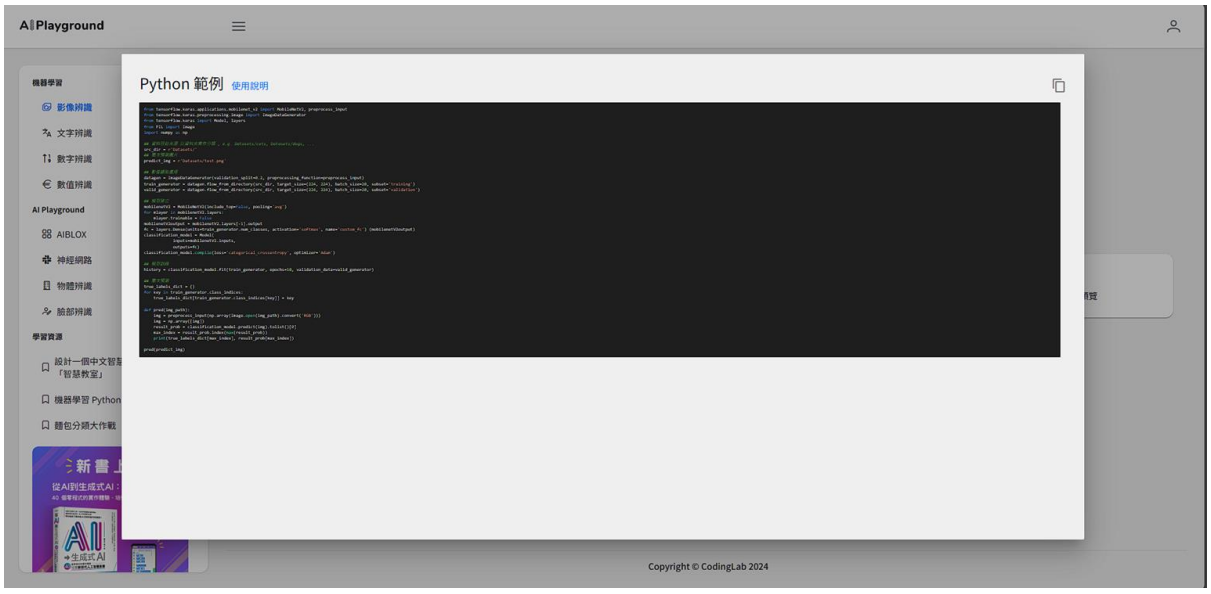
深度學習期末報告

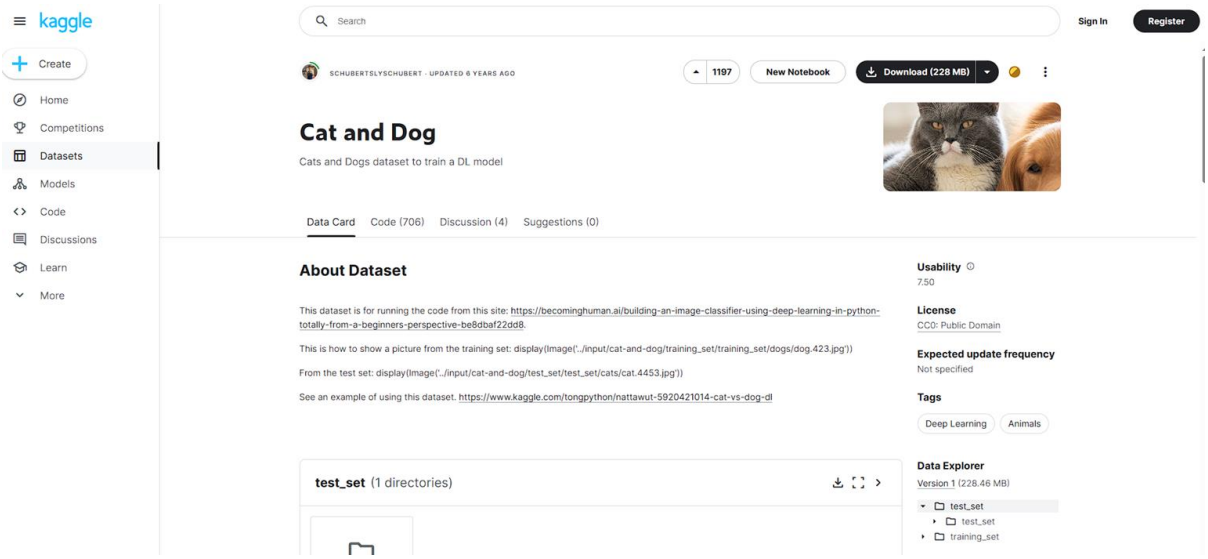
U0933023 資管四甲 郝佳倫

Project1 影像辨識

1. 前言

在本次深度學習期末報告中，我運用 vscode 以及 Python，並且採用了來自 AI Playground 平台中的影像辨識範例，來進行貓跟狗的影像辨識，而所使用的訓練集則是來自 Kaggle 中的 Cat and Dog，我將其下載並放入本地端的 Datasets 資料夾中進行訓練及驗證。






2. 程式碼說明

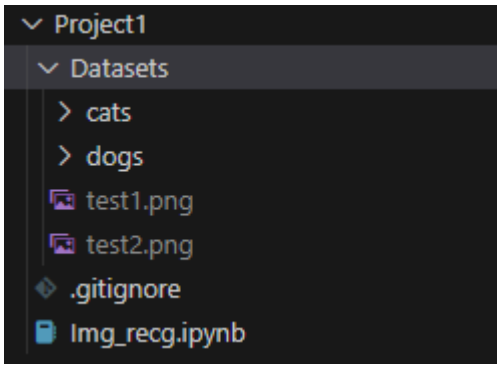
範例的程式碼主要分為四個部分，前三個部分為範例程式碼的說明，分別是收集資料、進行訓練、以及預測評估。第四部份為自行修改的程式碼與參數。以下將分別來做詳細地說明。

(1) 收集資料

下圖為專案資料夾的結構，其中 Datasets 對應 src.dir 所指向的位置，下一層的 cats 跟 dogs 則是預計訓練機器所能識別的類別名稱，同時會將從 Kaggle 取得的資料集中的圖片分別放置於兩個資料夾內，讓機器去學習圖片特徵與類別名稱當中的關聯性。如果想要辨識不同種類的動物，例如大象，則可以新增大象的訓練集去做應用。

而影像處理方式則是使用了 TensorFlow 的 ImageDataGenerator 來設定和自動處理圖像資料。它首先創建一個數據生成器 datagen，設定了將 20% 的數據作為驗證集並應用了適合 MobileNetV2 模型的預處理函數。然

後通過 flow_from_directory 方法從指定的目錄 src_dir 加載圖像，自動將圖像調整至 224x224 像素，並以 20 張圖像為一組批次生成訓練和驗證數據。這樣的設定除了可以保證數據格式與模型輸入要求相匹配，也方便了模型的訓練和驗證過程。




```
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import Model, layers
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from PIL import Image
import numpy as np
##驗證是否安裝tensorflow
import tensorflow as tf
print(tf.__version__)
from PIL import Image
print(Image.__version__)

## 資料目錄來源 以資料夾當作分類 , e.g. Datasets/cats, Datasets/dogs, ...
src_dir = r'Datasets/'
## 單次預測圖片
predict_img = r'Datasets/test1.png'

## 影像讀取處理
datagen = ImageDataGenerator(validation_split=0.2, preprocessing_function=preprocess_input)
train_generator = datagen.flow_from_directory(src_dir, target_size=(224, 224), batch_size=20, subset='training')
valid_generator = datagen.flow_from_directory(src_dir, target_size=(224, 224), batch_size=20, subset='validation')
```

(2) 進行訓練

當資料都準備齊全之後，便可以開始訓練。下圖為訓練階段的程式碼，範例程式預設進行10個訓練時期(epoch)，epoch 更改可以直接於程式碼內修改。

```python
# 初始化 MobileNetV2，明確指定輸入尺寸
mobilenetV2 = MobileNetV2(include_top=False, pooling='avg', input_shape=(224, 224, 3))

## 模型建立
mobilenetV2 = MobileNetV2(include_top=False, pooling='avg')
for mlayer in mobilenetV2.layers:
    mlayer.trainable = False
mobilenetV2output = mobilenetV2.layers[-1].output
fc = layers.Dense(units=train_generator.num_classes, activation='softmax', name='custom_fc') (mobilenetV2output)
classification_model = Model(
            inputs=mobilenetV2.inputs,
            outputs=fc)
classification_model.compile(loss='categorical_crossentropy', optimizer='Adam')

## 模型訓練
history = classification_model.fit(train_generator, epochs=10, validation_data=valid_generator)
```

(3) 預測評估

下圖為評估此次訓練的程式碼，訓練過程中的每個 epoch 報告了兩個損失值。loss 是模型在訓練數據上計算出的損失值。這個數值代表模型對訓練數據的擬合程度，數值越低表示模型對訓練數據的預測越準確；val_loss 是模型在驗證數據上計算出的損失值。這個數值用於評估模型對未見數據的泛化能力，同樣地，數值越低表示模型對驗證數據的預測越準確。

```python
## 單次預測
true_labels_dict = {}
for key in train_generator.class_indices:
    true_labels_dict[train_generator.class_indices[key]] = key

def pred(img_path):
    img = preprocess_input(np.array(Image.open(img_path).convert('RGB')))
    img = np.array([img])
    result_prob = classification_model.predict(img).tolist()[0]
    max_index = result_prob.index(max(result_prob))
    print(true_labels_dict[max_index], result_prob[max_index])

pred(predict_img)
```

(4) 修改程式碼

由於我好奇若把訓練時期拉長，會得到何種結果，所以便將 epoch 調高至50，並且新增了程式碼去監控其效能的變化。我先從 tensorflow.keras.callbacks 函式庫中 import 了 EarlyStopping，接著設定了早期停止的監控器，以監控驗證集上的損失。若在3個訓練時期內效能沒有提升，則停止訓練，並且恢復到最佳模型權重。以此來確保達到最佳效能的同時，又不會過度擬合。

```
from tensorflow.keras.callbacks import EarlyStopping
```

```
# 設定早期停止監控器
early_stopping_monitor = EarlyStopping(
    monitor='val_loss',  # 監控驗證集上的損失
    patience=3,          # 在3個epoch內如果性能沒有提升，則停止
    verbose=1,
    restore_best_weights=True  # 恢復到最佳模型權重
)
```

```
## 模型訓練
history = classification_model.fit(
    train_generator,
    epochs=50,
    validation_data=valid_generator,
    callbacks = [early_stopping_monitor] #新增早期停止callbacks
)
```

3. 結論

此次訓練的成果如圖所示，其中，模型預測正確出 test1(貓咪照)屬於 cats 類別，而模型對這個預測結果的信心程度達到0.9999536275863647，也就是模型認為 test1為「cats」類別的概率。這個數字非常接近1，表明模型非常確信這個預測是正確的，而結果也確實是正確的。

接著我們設定了早期停止監控器，並且把訓練時期增加到50，結果如第二張圖所示，令人驚訝的是，模型非但沒有訓練超過10次，反而第6次就停止了，也就是說，自第3個訓練時期後，接連3個訓練時期的效能都沒有增加，反而還下降，因此模型判定第3次訓練的結果為最佳。但最終結果的信心水準是比較低的，達到0.9997395873069763。

```
Found 6404 images belonging to 2 classes.
Found 1600 images belonging to 2 classes.
C:\Users\owner\AppData\Local\Temp\ipykernel_11368\2884025764.py:27: UserWarning
  mobilenetV2 = MobileNetV2(include_top=False, pooling='avg')
Epoch 1/10
C:\Users\owner\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5
  self._warn_if_super_not_called()
321/321 ━━━━━━━━━━━━━━━━━━━━ 84s 246ms/step - loss: 0.1310 - val_loss: 0.0470
Epoch 2/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 76s 237ms/step - loss: 0.0364 - val_loss: 0.0410
Epoch 3/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 77s 241ms/step - loss: 0.0271 - val_loss: 0.0432
Epoch 4/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 79s 247ms/step - loss: 0.0228 - val_loss: 0.0453
Epoch 5/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 76s 237ms/step - loss: 0.0187 - val_loss: 0.0645
Epoch 6/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 76s 237ms/step - loss: 0.0150 - val_loss: 0.0529
Epoch 7/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 77s 239ms/step - loss: 0.0111 - val_loss: 0.0498
Epoch 8/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 76s 236ms/step - loss: 0.0076 - val_loss: 0.0538
Epoch 9/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 76s 236ms/step - loss: 0.0089 - val_loss: 0.0524
Epoch 10/10
321/321 ━━━━━━━━━━━━━━━━━━━━ 76s 238ms/step - loss: 0.0072 - val_loss: 0.0526
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
cats 0.9999536275863647
```

```
Found 6404 images belonging to 2 classes.
Found 1600 images belonging to 2 classes.
C:\Users\owner\AppData\Local\Temp\ipykernel_19460\2344927676.py:37: UserWarning: `
  mobilenetV2 = MobileNetV2(include_top=False, pooling='avg')
Epoch 1/50
C:\Users\owner\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2k
  self._warn_if_super_not_called()
321/321 ━━━━━━━━━━━━━━━━━━━━ 88s 260ms/step - loss: 0.1182 - val_loss: 0.0466
Epoch 2/50
321/321 ━━━━━━━━━━━━━━━━━━━━ 77s 241ms/step - loss: 0.0384 - val_loss: 0.0525
Epoch 3/50
321/321 ━━━━━━━━━━━━━━━━━━━━ 77s 241ms/step - loss: 0.0324 - val_loss: 0.0451
Epoch 4/50
321/321 ━━━━━━━━━━━━━━━━━━━━ 78s 244ms/step - loss: 0.0207 - val_loss: 0.0454
Epoch 5/50
321/321 ━━━━━━━━━━━━━━━━━━━━ 77s 240ms/step - loss: 0.0160 - val_loss: 0.0460
Epoch 6/50
321/321 ━━━━━━━━━━━━━━━━━━━━ 78s 244ms/step - loss: 0.0145 - val_loss: 0.0484
Epoch 6: early stopping
Restoring model weights from the end of the best epoch: 3.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 987ms/step
cats 0.9997395873069763
```