天氣與人工智慧 II　　HW2

大氣 4A 106601015 黃展皇

HW1 的 DNN test 最佳模型：(10epoch, 32*32*3 cifar10 資料集)

```
def create_dnn_model_v4():
    model = Sequential()

    model.add(tf.keras.Input(shape=(32, 32, 3), batch_size=None))
    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dense(256, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.summary()
    return model
```

準確率 50.09%

CNN test結果(無pre-trained model)：

```
cnn_v1_10epochs.h5 test acc: 0.73380, test loss: 0.78907
cnn_v1_20epochs.h5 test acc: 0.74030, test loss: 0.81207
cnn_v2_10epochs.h5 test acc: 0.71290, test loss: 0.82936
cnn_v2_20epochs.h5 test acc: 0.72770, test loss: 0.86576
cnn_v3_10epochs.h5 test acc: 0.70100, test loss: 0.86570
cnn_v3_20epochs.h5 test acc: 0.71400, test loss: 0.85117
cnn_v4_10epochs.h5 test acc: 0.73090, test loss: 0.77854
cnn_v4_20epochs.h5 test acc: 0.73930, test loss: 0.75445
cnn_v5_10epochs.h5 test acc: 0.76160, test loss: 0.69342
cnn_v5_20epochs.h5 test acc: 0.78120, test loss: 0.66755
cnn_v6_10epochs.h5 test acc: 0.76730, test loss: 0.91139
cnn_v6_20epochs.h5 test acc: 0.74830, test loss: 1.17145
cnn_v7_10epochs.h5 test acc: 0.10000, test loss: 2.30262
cnn_v7_20epochs.h5 test acc: 0.10000, test loss: 2.30274
```

佛系建構7種模型做各種模型結構上的差異，並訓練10、20epochs共14個case

10epochs最佳者為v6版本，準確率76.73%，架構如下：

```python
def create_cnn_model_v6():
    model = Sequential()
    model.add(Conv2D(filters=64, kernel_size=3, input_shape=(img_row, img_col, 3), activation='relu', padding='same'))
    model.add(Conv2D(filters=64, kernel_size=3, activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=128, kernel_size=3, activation='relu', padding='same'))
    model.add(Conv2D(filters=128, kernel_size=3, activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=128, kernel_size=3, activation='relu', padding='same'))
    model.add(Conv2D(filters=128, kernel_size=3, activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(rate=0.25))
    model.add(Dense(10, activation='softmax'))
    #model.summary()
    return model
```

20epochs最佳者為v5版本，準確率78.12%，架構如下：

```python
def create_cnn_model_v5():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(img_row, img_col, 3)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(10, activation='softmax'))
    #model.summary()
    return model
```

v5與v6模型主要差在前期v5多了Dropout層，而且整體網路較深，這使得v6可以較快訓練，但容易會有模型太簡單的問題，當訓練次數拉高時就容易產生overfitting，20epochs時測試準確度反而下降；反之v5則多了Dropout設計，讓整體網路的預測泛化能力可以隨著加深而拓展。

CNN網路除了過於複雜所以訓練到壞掉的情況(v7)以外，準確度隨便都是70%以上，比DNN最佳的50%左右好上不少。(以上都是採用32*32*3 cifar10資料集)

CNN test結果(pre-trained model==VGG16)：

```
VGG16_pretrain_10epochs.h5 test acc: 0.61660, test loss: 1.14414
VGG16_pretrain_20epochs.h5 test acc: 0.61070, test loss: 1.21924
VGG16_pretrain_finetuning_10epochs.h5 test acc: 0.72850, test loss: 1.01168
VGG16_pretrain_finetuning_20epochs.h5 test acc: 0.73340, test loss: 1.21259
VGG16_pretrain_64_10epochs.h5 test acc: 0.69550, test loss: 0.98513
VGG16_pretrain_64_20epochs.h5 test acc: 0.69490, test loss: 1.22168
VGG16_pretrain_finetuning_64_10epochs.h5 test acc: 0.79300, test loss: 0.90522
VGG16_pretrain_finetuning_64_20epochs.h5 test acc: 0.79510, test loss: 1.07846
```

pre-trained model 部分比較麻煩，模型採用VGG16，主要是分成不動主架構/ finetuning、10/20 epochs、(32*32)/(64*64)*3資料集共8種情況。

可以發現幾件事情：

1. 隨著epochs提升，不動主架構的模型不一定會提升準確度，甚至會有準確度倒退可能為overfitting的情況；反之finetuning的部分則幾乎都有程度不一的提升。追究原因應該是不動主架構的模型只有最後的Dense在訓練，相較於動到VGG16 block5_conv1以及最後的Dense的finetuning模型更容易overfitting。
2. Finetuning普遍比不動主架構準確率更高(好蠻多的)。
3. 資料集從32*32->64*64對於VGG16 case有顯著幫助(準確率好很多)。

```python
def create_VGG16_model_complete():
    VGG16_model = VGG16(
        weights='imagenet',
        include_top=False,
        input_shape=(img_row, img_col, 3)
    )

    VGG16_model.trainable = False

    model = Sequential()
    model.add(VGG16_model)
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(10, activation='softmax'))

    #model.summary()
    return model
```

```python
def create_VGG16_model_finetuning():
    VGG16_model = VGG16(
        weights='imagenet',
        include_top=False,
        input_shape=(img_row, img_col, 3)
    )

    VGG16_model.trainable = True
    set_trainable = False
    for layer in VGG16_model.layers:
        if layer.name == 'block5_conv1':
            set_trainable = True
        if set_trainable:
            layer.trainable = True
        else:
            layer.trainable = False

    model = Sequential()
    model.add(VGG16_model)
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(10, activation='softmax'))

    #model.summary()
    return model
```

結論：

附上各模型的test準確率與loss

```
cnn_v1_10epochs.h5 test acc: 0.73380, test loss: 0.78907
cnn_v1_20epochs.h5 test acc: 0.74030, test loss: 0.81207
cnn_v2_10epochs.h5 test acc: 0.71290, test loss: 0.82936
cnn_v2_20epochs.h5 test acc: 0.72770, test loss: 0.86576
cnn_v3_10epochs.h5 test acc: 0.70100, test loss: 0.86570
cnn_v3_20epochs.h5 test acc: 0.71400, test loss: 0.85117
cnn_v4_10epochs.h5 test acc: 0.73090, test loss: 0.77854
cnn_v4_20epochs.h5 test acc: 0.73930, test loss: 0.75445
cnn_v5_10epochs.h5 test acc: 0.76160, test loss: 0.69342
cnn_v5_20epochs.h5 test acc: 0.78120, test loss: 0.66755
cnn_v6_10epochs.h5 test acc: 0.76730, test loss: 0.91139
cnn_v6_20epochs.h5 test acc: 0.74830, test loss: 1.17145
cnn_v7_10epochs.h5 test acc: 0.10000, test loss: 2.30262
cnn_v7_20epochs.h5 test acc: 0.10000, test loss: 2.30274
VGG16_pretrain_10epochs.h5 test acc: 0.61660, test loss: 1.14414
VGG16_pretrain_20epochs.h5 test acc: 0.61070, test loss: 1.21924
VGG16_pretrain_finetuning_10epochs.h5 test acc: 0.72850, test loss: 1.01168
VGG16_pretrain_finetuning_20epochs.h5 test acc: 0.73340, test loss: 1.21259
VGG16_pretrain_64_10epochs.h5 test acc: 0.69550, test loss: 0.98513
VGG16_pretrain_64_20epochs.h5 test acc: 0.69490, test loss: 1.22168
VGG16_pretrain_finetuning_64_10epochs.h5 test acc: 0.79300, test loss: 0.90522
VGG16_pretrain_finetuning_64_20epochs.h5 test acc: 0.79510, test loss: 1.07846
```

程式碼以及訓練曲線附錄於壓縮檔中。