智慧化企業整合

資源回收分類辨識

指導教授： 邱銘傳 教授

學生：109034541 蘇詠心

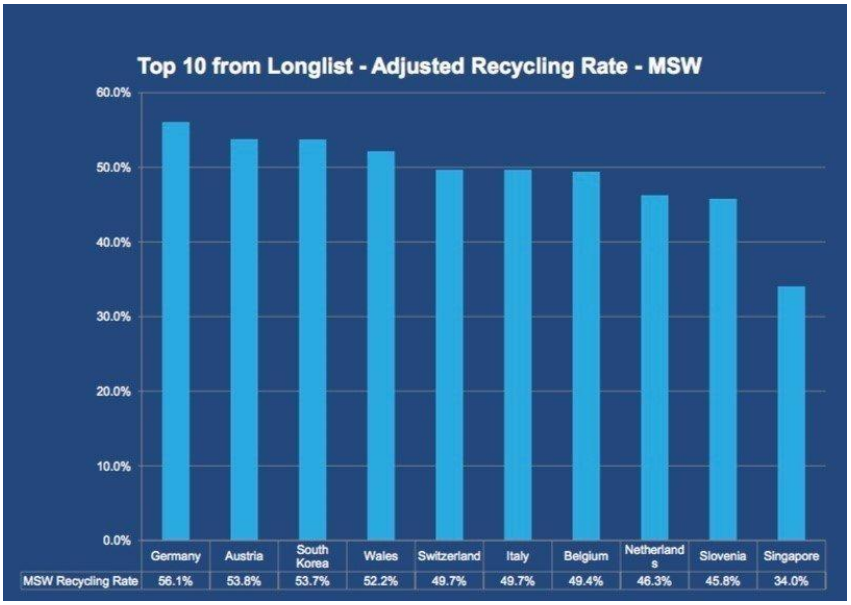
中 華 民 國 １１０ 年 一 月 七 日

目錄

1 研究背景

資源回收一直是全球專注的重要議題，隨著資本主義的發展，人們對於物質的需求與日俱增，再加上快時尚與網購需求的的影響，根據統計，全球每年平均製造出 21 億噸垃圾，足以填滿 82 萬座奧林匹克規格的游泳池，且此數值每年仍在增加中，預估到達 2025 年時，全球的垃圾產出量將會提高 70%。其中，垃圾製造的排行中，佔世界人口 16% 高收入國家人口，製造了全球 34% 的，然而，在如此奢侈的消費習慣與龐大的垃圾製造下，實際的資源回收能力相配卻無法與之相配。在未將垃圾做好分類並回收的情況下，往往浪費許多可回收再利用的物資，除了資源浪費外，垃圾在焚燒時不僅會造成空氣污染，填埋後還可能造成土壤和地下水源污染。

下圖為全球資源回收率最高之前 10 名國家，可以看出德國的資源回收率高達 56.1%，位居全球最高，而新加坡的資源回收率僅有 34%，位居第 10，故可從中推斷出，世界上仍有許多國家，其資源回收率極低，因此若能夠提升人民的回收意願，以及將垃圾正確分類，相信能夠提升資源回收率。



2 問題定義

卷積神經網絡 (Convolutional Neural Network, CNN) 是一種前饋神經網路，其模型在影像識別方面的表現非常出色，主要借助卷積層 (Convolution Layer) 的方法，將輸入從原始的圖，改為經過影像處理技術萃取的特徵，使機器更容易辨識。本專案以 Kaggle 資源回收資料集為建立資源回收種類分類模型，運用 CNN 建立模型以便我們將不同資源回收種類做分類，並期望透過不同之超參數調整，能夠提升模型的準確率，並協助我們在未來做使用。

由於過去資源回收垃圾收集後，需要以人工方式進行分類，然而使用人工的方式將垃圾分類，不僅準確度不高，效率不佳，還需耗費許多人力成本，但若能夠透過深度學習之方式，建立資源回收種類辨識模型去協助將垃圾分類，如此一來，便可大幅降低在垃圾分類上的時間，同時也會提升資源回收率，不論是對於物品的再利與地球的環保都可盡一份心力，故本專案之 5W1H 如下所示：

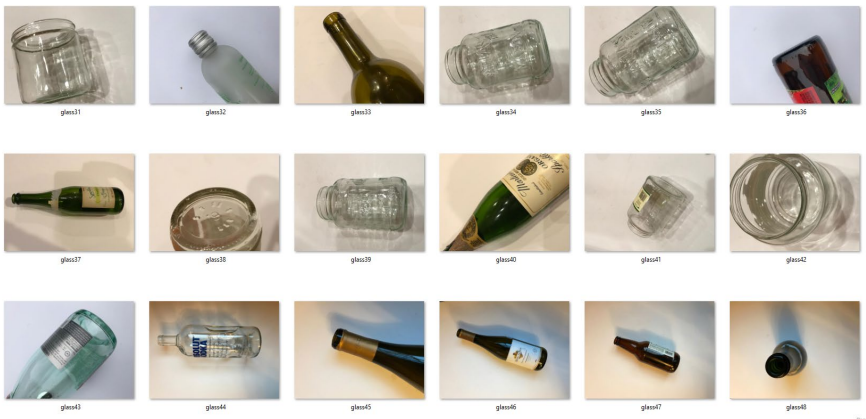
1. Why：由於目前資源回收率甚低，希望可透過機器協助垃圾分類辨識，以提升民眾資源回收的意願，另外也可協主資源回收處裡業主在垃圾分類上具有更高的正確度與效率。
2. Who：一般民眾、資源回收處理業主
3. What：協助其進行垃圾分類
4. Where：垃圾桶、資源回收場
5. When：進行垃圾分類前
6. How：運用 Python 與 CNN 架構建立資源回收種類辨識模型

3 資料分析與前處理

3.1 資料來源與結構分析

為建立合適模型，本實驗採用 Kaggle 內的數據作為資料來源。Kaggle 是一個數據建模和數據分析競賽平台。企業和研究者可在此發布數據，提供統計學家和數據挖掘專家以此資料為基礎，進行競賽以產生最好的模型。

本實驗以 Kaggle 資源回收資料集進行資源回收種類辨識，其中將資源回收種類分成五種種類，分別為紙板、玻璃、金屬、紙張、塑膠，共 2,000 張圖片，以玻璃罐照片為例，如下圖所示。並針對上述照片進行模型訓練與驗證。



3.2 資料前處理

3.2.1 資料視覺化

將資料以路徑的形式讀出來，並使用 Matplotlib 繪製出各種數據之資料量。

```
dir_path = 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification'
img_list = glob.glob(os.path.join(dir_path, '*/*.jpg'))
```

```
len(img_list)
```

```
2000
```

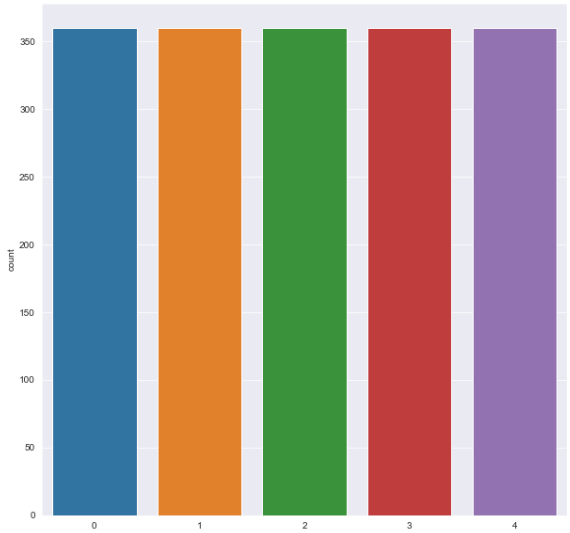
```
img_list
```

```
['C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard1.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard10.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard100.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard101.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard102.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard103.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard104.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard105.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard106.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard107.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard108.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard109.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard11.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard110.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard111.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard112.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard113.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard114.jpg',
 'C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification\\cardboard\\cardboard115.jpg',
```

使用以 seaborn 套件成現資料分布情形，發現資料沒有不平衡。

```
plt.figure(figsize=(10,10))
sns.set_style("darkgrid")
sns.countplot(x=a)
```

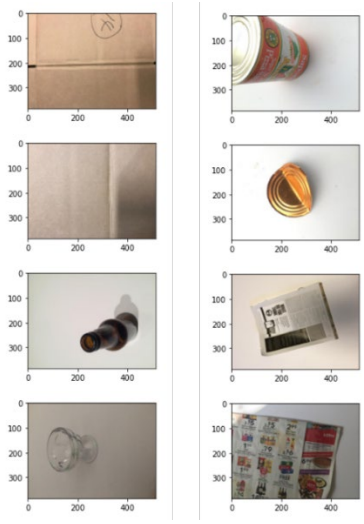
```
<matplotlib.axes._subplots.AxesSubplot at 0x2063f1e83d0>
```



使用 Matplotlib 將圖片顯示在 python 上，以確保匯入資料正確。

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
folder = os.listdir('C:/Users/fr318/data_garbage_01/Garbage classification/Garbage classification')

for i in range(5):
    for j in range(2):
        j+=1
        plt.figure(figsize = (3, 3))
        folder = os.path.join(str(dir_path+'/'+labels[i]+'/'+labels[i]+str(j)+'.jpg'))
        img = mpimg.imread(folder)
        imgplot = plt.imshow(img)
        plt.show()
```



3.2.2 資料標準化

由於色碼各個像素值範圍介於 0 ~ 255，因此將所有特徵值除上 255 ，標準化以縮減數據間的跨度，有利於模型的收斂。

3.2.3 統一圖片大小

將每張圖片的大小統一，並將每張圖片大小設定為 300 X 300 的圖像。

3.2.4 資料增強

在深度學習訓練時，需要大量資料以確保訓練時不會產生過擬合 (over-fitting) 的情況，然而由於本次使用之資料集圖片張數不多，故使用 Data augmentation 將 dataset 中既有的圖片予以修改變形，以創造出更多的圖片來讓機器學習，彌補資料量不足的困擾。本次使用 Keras 之 ImageDataGenerator 套件進行資料增強，並利用以下方式產生不同圖片：

- 隨機對圖片執行水平/垂直翻轉操作
- 把像素值放縮到 0 和 1 之間有利於模型的收斂
- 錯切變換

- 讓圖片在長或寬的方向進行放大
- 水平位置平移和上下位置平移

```
#Data Augmentation
train=ImageDataGenerator(horizontal_flip=True,
                         vertical_flip=True,
                         validation_split=0.1,
                         rescale=1./255,
                         shear_range = 0.1,
                         zoom_range = 0.1,
                         width_shift_range = 0.1,
                         height_shift_range = 0.1,)

test=ImageDataGenerator( validation_split=0.1,
                         rescale=1/255)


train_generator=train.flow_from_directory(dir_path,
                                          target_size=(300,300),
                                          batch_size=32,
                                          class_mode='categorical',
                                          subset='training')

test_generator=test.flow_from_directory(dir_path,
                                        target_size=(300,300),
                                        batch_size=32,
                                        class_mode='categorical',
                                        subset='validation')
```

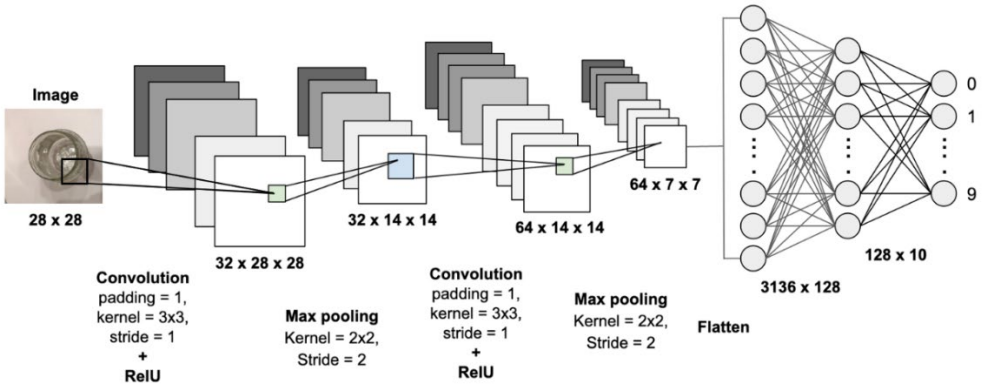
3.2.5 資料切割

將訓練資料集與測試資料集以 9:1 的比例做切割，故在 2,000 筆資料中，共分為 1,800 筆訓練資料集與 200 筆驗證資料集。

4 模型建立與訓練

4.1 CNN 模型介紹

本實驗使用深度學習中的 CNN 模型，相較於傳統的 DNN 多了卷積層 (Convolutional Layer) 以及池化層 (Pooling Layer)，主要用來維持形狀資訊並且避免參數大幅增加。接著利用全連接層 (Fully Connected Layer) 將之前的結果平坦化後，接到最基本的神經網絡，最後使用 Softmax activation function 來輸出分類結果，示意圖如下圖所示。



4.2 CNN 模型建立

本實驗之初始設定如下：

- Convolution Layer (# of neuron=32)
- Max-pooling Layer
- Convolution Layer (# of neuron=64)
- Max-pooling Layer
- Convolution Layer (# of neuron=32)
- Max-pooling
- Flatten
- Dense (# of neuron=64)
- Dropout = 0.2
- Dense (# of neuron=32)
- Dropout = 0.2
- Output Layer
- Activation function：ReLU, Softmax (in output layer)
- Loss function：:categorical_crossentrophy

```
model=Sequential()
#Convolution blocks

model.add(Conv2D(32,(3,3), padding='same',input_shape=(300,300,3),activation='relu'))
model.add(MaxPooling2D(pool_size=2))
#model.add(SpatialDropout2D(0.5)) # No accuracy

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

model.add(Conv2D(32,(3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))

#Classification Layers
model.add(Flatten())

model.add(Dense(64,activation='relu'))
#model.add(SpatialDropout2D(0.5))
model.add(Dropout(0.2))

model.add(Dense(32,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(5,activation='softmax'))

filepath="trained_model22.h5"
checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint1]

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 300, 300, 32)      896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 150, 150, 32)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 150, 150, 64)      18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 75, 75, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 75, 75, 32)        18464
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 43808)             0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                2803776
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 165
=================================================================
Total params: 2,843,877
Trainable params: 2,843,877
Non-trainable params: 0
_________________________________________________________________
```

預設模型正確度 (Test Accuracy)：76.5%

5 超參數調整

針對上述模型進行超參數的調整，並針對 optimizer, batch size, dropout rate 以及不同層數之 convolution, max-pooling and dense 來進行調整。

5.1 針對 optimizer 進行調整

調整 optimizer，本實驗比較兩種不同之 optimizer，adam 和 adagrad。可以發現此兩者結果相距甚大，optimizer 使用 adam 的時候模型表現最好，因此保留其參數設定。下表為 optimizer 評估比較表。

Convolution	Max-pooling	Dense	droupout rate	activation function	loss function	optimizer	metrics	epochs	batch size	accuracy
3	3	3	0.2	relu	categorical_crossentropy	**adam**	acc	100	32	76.50%
3	3	3	0.2	relu	categorical_crossentropy	**adagrad**	acc	100	32	49%

5.2 針對 batch size 進行調整

接著調整 batch size，本實驗比較三種 batch size，32、50、100 可以得知當 batch size 為 32 的時候模型表現最好，因此保留其參數設定。下表為 batch size 評估比較表。

Convolution	Max-pooling	Dense	droupout rate	activation function	loss function	optimizer	metrics	epochs	**batch size**	accuracy
4	4	3	0.2	relu	categorical_crossentropy	adam	acc	100	**32**	80%
4	4	3	0.2	relu	categorical_crossentropy	adam	acc	100	**50**	73.50%
4	4	3	0.2	relu	categorical_crossentropy	adam	acc	100	**100**	77.50%

5.3 使用不同層數之 convolution, max-pooling and dense 進行調整

調整模型的層數，使用不同層數之 convolution, max-pooling and dense 進行調整，並由下表可以得知當 convolution layer 為 4、max-pooling layer 為 4 、dense 為 3 的時候模型表現最好，因此保留其參數設定。下表為不同層數之比較表。

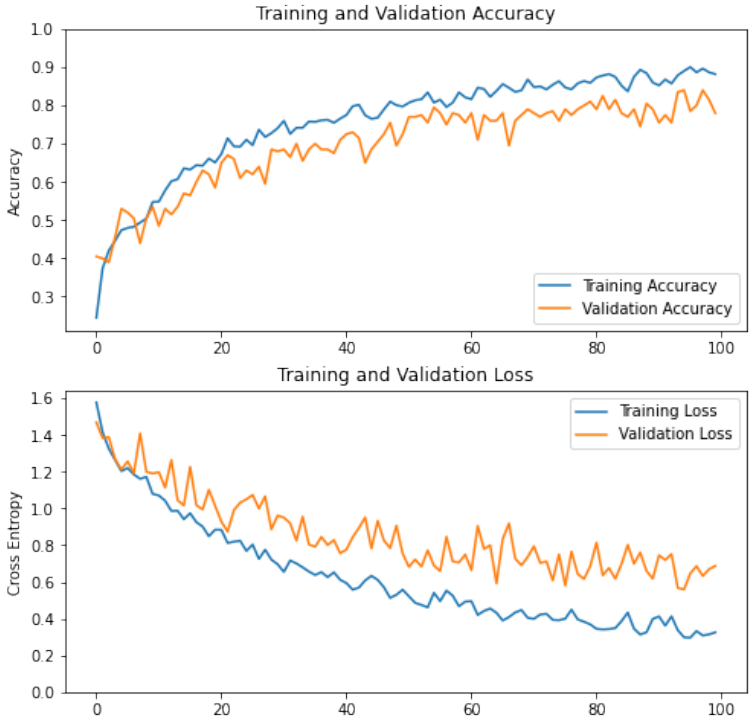
Convolution	**Max-pooling**	**Dense**	droupout rate	activation function	loss function	optimizer	metrics	epochs	batch size	accuracy
2	**2**	**3**	0.2	relu	categorical_crossentropy	adam	acc	100	32	71%
3	**3**	**3**	0.2	relu	categorical_crossentropy	adam	acc	100	32	76.50%
4	**4**	**3**	0.2	relu	categorical_crossentropy	adam	acc	100	32	80%
5	**5**	**3**	0.2	relu	categorical_crossentropy	adam	acc	100	32	73%
5	**5**	**4**	0.2	relu	categorical_crossentropy	adam	acc	100	32	77%

5.4 針對 dropout rate 進行調整

接著調整 dropout rate，本實驗比較五種 dropout rate，0.01、0.1、0.2、0.3、0.5，可以得知當 dropout rate 為 0.2 的時候模型表現最好，因此保留其參數設定。下表為 dropout rate 評估比較表。

Convolution	Max-pooling	Dense	**droupout rate**	activation function	loss function	optimizer	metrics	epochs	batch size	accuracy
4	4	3	**0.01**	relu	categorical_crossentropy	adam	acc	100	32	74%
4	4	3	**0.1**	relu	categorical_crossentropy	adam	acc	100	32	78.50%
4	4	3	**0.2**	relu	categorical_crossentropy	adam	acc	100	32	80%
4	4	3	**0.3**	relu	categorical_crossentropy	adam	acc	100	32	77.50%
4	4	3	**0.5**	relu	categorical_crossentropy	adam	acc	100	32	68.50%

從超參數調整的過程中，可以發現模型的測試資料準確度已經從原始的 76.50%、77.50%、78% 提升至 80%。另外，由下圖可看出訓練資料集與測試資料集的準確度分布以及其 loss function 的值。



6.結論

此次實驗利用 CNN 模型架構設計資源回收種類辨識，並經由模型調整訓練的結果可觀察到，模型的準確度提高至 80%，然而若要運用於實際生活中，仍會因為正確率不夠高，而達不到預期效果，因此未來若要提高模型的準確度，可以對模型進行更多的訓練，增加資料集圖片的數量，或是調整其他超參數來達到更高的正確率。