# 各類蔬果 CNN 辨識系統

109034531 黃怡菁

指導教授：邱銘傳教授

## 摘要

現今社會步調迅速，許多事情都講求效率，然而日常生活中時常面臨排隊結帳，浪費許多時間，越來越多商場或超市為了減少收銀員的人事成本，以及短顧客等待時間，而設立自助結帳機台，但是顧客從購物車內拿出選購商品並掃描條碼依舊需要花費不少時間，或是遇到條碼汙損的情況仍然須請門市人員確認價格。

因此本研究是使用 CNN 機器學習圖像辨識系統來自動辨識蔬果，未來可加入價格以自動加總金額，達到快速結帳的目的。由 Kaggle 圖像資料庫去得所需的圖片資料，建立一個類神經網路系統，判斷的準確率達到 98.10%。

**關鍵字**：類神經網路、CNN、水果分類

# 目錄

# 一、 緒論

## 1. 研究動機與目的

隨著科技進步，且現今是個講求效率的時代，許多店家推出自助結帳系統，減少人力支出，如果能夠透過類神經網路進行影像辨識各類蔬果，當消費者將購物車內的物品放置掃描機台下，便能迅速辨識蔬果種類及個數，並加總消費金額，即可大大縮短結帳時間，因此本專題建立一個卷積神經網路，幫助分辨各類水果，大幅降低結帳時間。

## 2. 問題定義 5W1H

- Who

  舉凡有在用實體商場或超市購買蔬果的人們

- What

  解決結帳時排隊或是條碼標示不清的困擾

- Why

  越來越多店家採用自助結帳機台，結帳時顧客仍須一一將購物車內的物品拿出來逐一掃瞄條碼，可能因為不熟悉機台操作方式而增加結帳時間

- When

  任何需要使用自助結帳機台的時候

- Where

  任何購物後須使用自助結帳機台的場所，例如：大型商場、超市

- How

  利用類神經網路訓練並建立一個可以識別水果種類的系統

# 二、 資料整理

## 1. 資料集來源

由 Kaggle 公開數據集中取得水果的圖像資料集，此次是使用「Fruit Recognition」，共分成 33 種類別，訓練集總共 16872 張照片，測試集總共 5641 張照片。

## 2. 資料前處理過程

每張圖片大小統一為 100x100，將 33 個資料類別加上標籤 0 至 32，並將圖片特徵標準化，全部除以 255，轉為 0 到 1 之間的數據，幫助資料訓練。

```
X = []
Y = [] #33
# enumerate 顯示序號
for i,labelname in enumerate(os.listdir("train/train")):
    print(i,labelname)
    for imgname in os.listdir("train/train/{}".format(labelname)):
        img = image.load_img('train/train/{}/{}'.format(labelname,imgname))
        X.append(image.img_to_array(img)/255) # append新增資料。將img array/255轉成0~1之間
        y = [0 for j in range(33)]
        y[i] = 1
        Y.append(y)
```

```
x_val = []
for imgname in os.listdir("test/test"):
    img = image.load_img('test/test/{}'.format(imgname))
    x_val.append(image.img_to_array(img)/255) #新增資料
x_val = np.array(x_val)
```

# 三、 模型設定

## 1. 模型一

(1) 模型架構

損失函數：categorical

卷積層：Conv2D

池化層：MaxPool2D

激活函數：relu

最後一層激活函數：sigmoid

優化器：Adam(lr = 0.01)

```
# 建一個CNN model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), input_shape=(100,100,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2))) # 池化
model.add(Dropout(0.3)) # 30%不修正，避免過擬合
model.add(Conv2D(64, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.3))
model.add(Flatten()) # 展開層
model.add(Dense(32,activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(33,activation="sigmoid"))


optim = optimizers.Adam(lr = 0.01)
model.compile(loss="categorical_crossentropy", optimizer=optim, metrics=["accuracy"])

model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 98, 98, 32)        896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 49, 49, 32)        0
_________________________________________________________________
dropout_18 (Dropout)         (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 47, 47, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 23, 23, 64)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 23, 23, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_12 (Dense)             (None, 32)                1083424
_________________________________________________________________
dropout_20 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_13 (Dense)             (None, 33)                1089
=================================================================
Total params: 1,103,905
Trainable params: 1,103,905
Non-trainable params: 0
```

(2) 執行過程

訓練集 8 成，測試集 2 成。

```
# 訓練及八成，測試集兩成
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, shuffle=True)
```

設定 early_stopping，避免 overfitting，patience= 5。

```
early_stopping = EarlyStopping(monitor="val_loss",patience=5,verbose=1)
```

開始訓練、batch_size 設定為 1024，epochs= 100。

```
history = model.fit(x_train,y_train,
    epochs=100,batch_size=1024,verbose=1,
    callbacks=[early_stopping],
    validation_data=(x_test,y_test))
```

在 Epoch 35/100 時出現 early stopping，accuracy= 90.82%，val_accuracy= 99.70%。

```
Epoch 35/100
14/14 [==============================] - 155s 11s/step - loss: 0.2596 - accuracy: 0.9082 - val_loss: 0.0228 - val_accuracy: 0.9970
Epoch 00035: early stopping
```

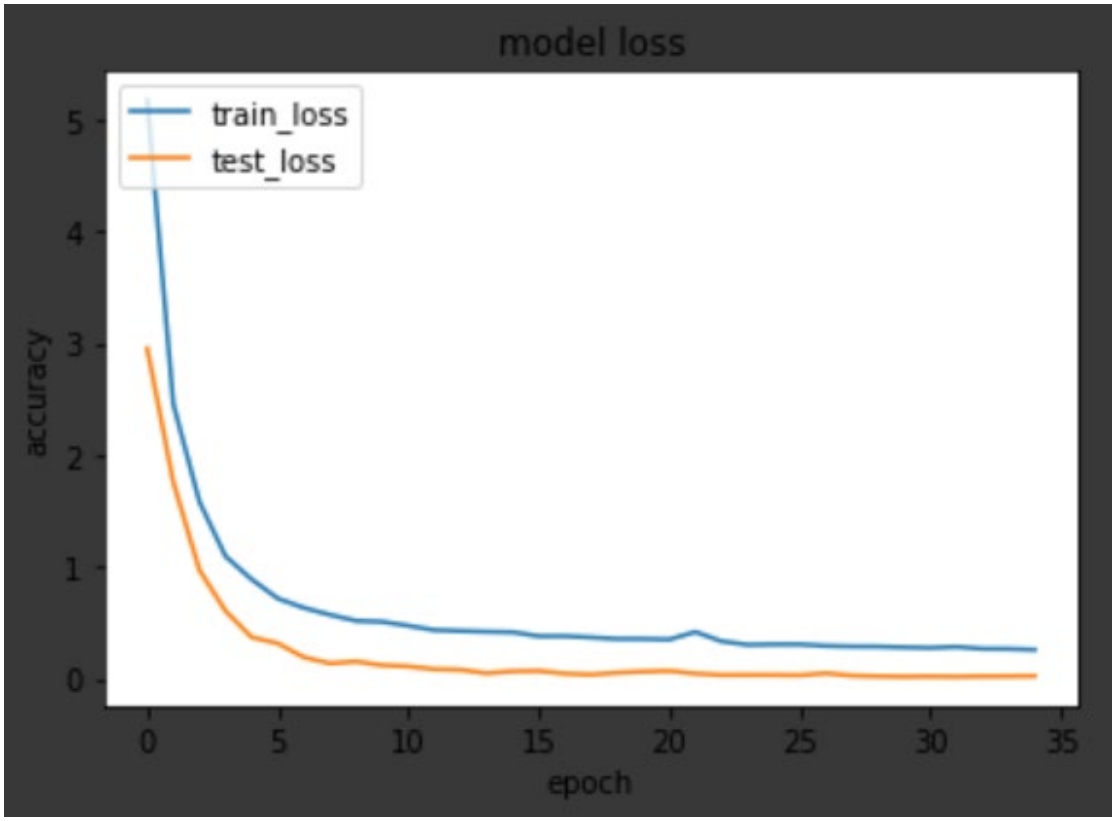
```
score  =  model.evaluate(x_test,y_test,verbose=0)
print(score)
y_val  =  model.predict(x_val)
print(y_val)
```

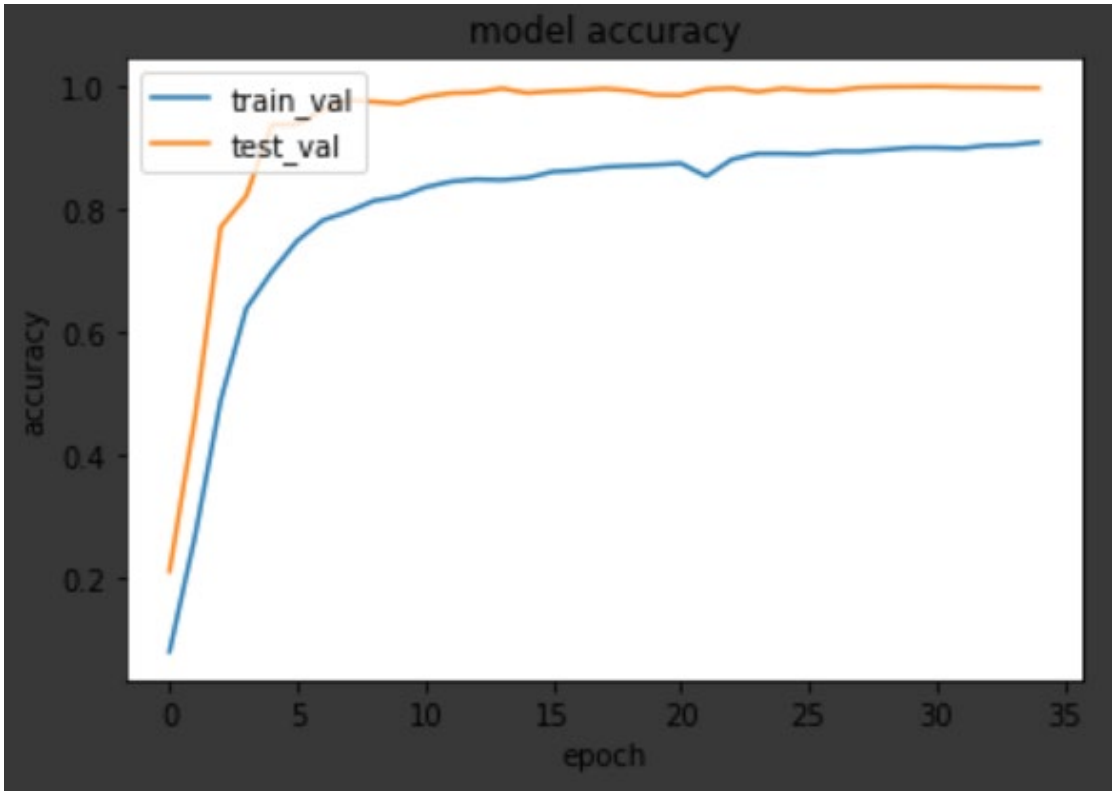
```
[0.02279523015022278, 0.9970335364341736]
[[2.2292754e-06 1.3314720e-12 9.2935413e-02 ... 8.0686164e-08
  1.7468527e-11 1.1778058e-06]
 [2.4881142e-09 9.8206270e-01 4.5281798e-02 ... 7.4178819e-10
  3.3337602e-07 1.2962500e-08]
 [1.9742656e-12 1.1144977e-12 9.9393368e-01 ... 2.1336529e-15
  2.6051459e-12 8.5137394e-16]
 ...
 [9.9950391e-01 0.0000000e+00 4.1431489e-10 ... 1.9596158e-23
  1.3028000e-21 6.4681713e-31]
 [4.3039918e-03 9.9999255e-01 1.3685717e-09 ... 1.1473060e-02
  2.7675837e-02 9.1987658e-01]
 [8.1216370e-12 1.8856898e-13 9.9066645e-01 ... 9.5604312e-16
  1.6805175e-12 3.2268440e-16]]
```

將訓練與測試結果繪製成圖。

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model  loss')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train_loss','test_loss'],  loc='upper  left')
plt.show()

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model  accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train_val','test_val'],  loc='upper  left')
plt.show()
```




發現 train_loss 高於 test_loss，出現 underfitting 的狀況，因此決定進行第二次訓練，將最後一層激活函數調整為 softmax，並提高 patience 至 10，以加快訓練速度。

## 模型二

(1) 模型架構

損失函數：categorical

卷積層：Conv2D

池化層：MaxPool2D

激活函數：relu

最後一層激活函數：softmax

優化器：Adam(lr = 0.001)

由於水果分類是 Multi-class Classification 問題，因此改用 softmax。

```
input = Input(shape=(100,100,3))

conv1_1 = Conv2D(32, kernel_size=(3,3), activation="relu")(input)
conv1_2 = Conv2D(64, kernel_size=(3,3), activation="relu")(conv1_1)
pool1_3 = MaxPool2D(pool_size=(2,2))(conv1_2)
drop1_4 = Dropout(0.3)(pool1_3)
flat1_5 = Flatten()(drop1_4)

conv2_1 = Conv2D(32, kernel_size=(5,5), activation="relu")(input)
conv2_2 = Conv2D(64, kernel_size=(5,5), activation="relu")(conv2_1)
pool2_3 = MaxPool2D(pool_size=(2,2))(conv2_2)
drop2_4 = Dropout(0.3)(pool2_3)
flat2_5 = Flatten()(drop2_4)

conv3_1 = Conv2D(32, kernel_size=(7,7), activation="relu")(input)
conv3_2 = Conv2D(64, kernel_size=(7,7), activation="relu")(conv3_1)
pool3_3 = MaxPool2D(pool_size=(2,2))(conv3_2)
drop3_4 = Dropout(0.3)(pool3_3)
flat3_5 = Flatten()(drop3_4)

merge = Concatenate()([flat1_5,flat2_5,flat3_5])
hidden = Dense(128,activation="relu")(merge)
output = Dense(33,activation="softmax")(hidden)
model = Model(inputs=input, outputs=output)
optim = optimizers.Adam(lr = 0.001)

model.compile(loss="categorical_crossentropy", optimizer=optim, metrics=["accuracy"])

model.summary()
```

設定 early_stopping，避免 overfitting，patience= 10。

```
early_stopping = EarlyStopping(monitor="val_loss",patience=10,verbose=1)
```

(2) 執行過程

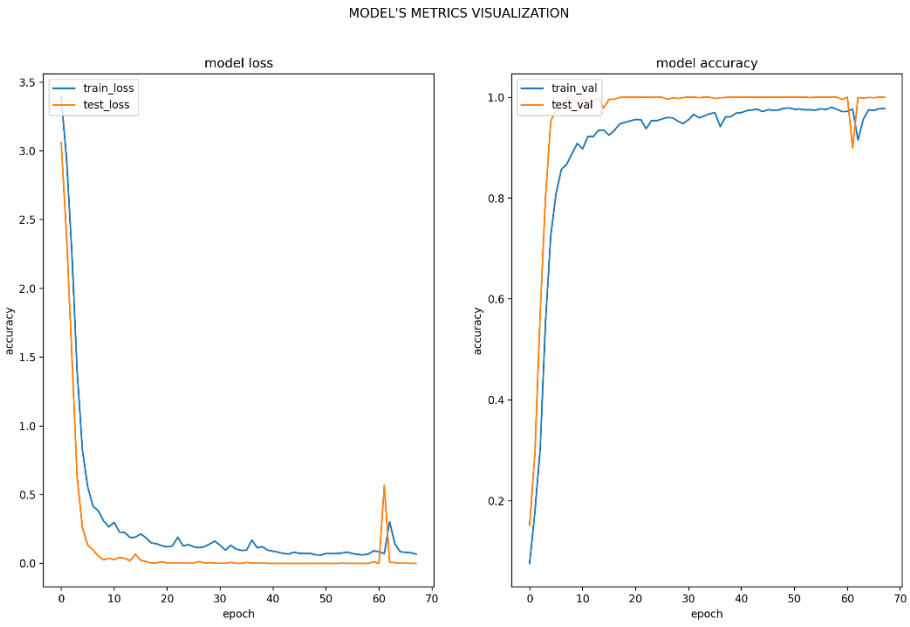
開始訓練、batch_size 設定為 512，epochs= 1000。

```
history = model.fit(x_train,y_train,
    epochs=1000,batch_size=512,verbose=1,
    callbacks=[early_stopping],
    validation_data=(x_test,y_test))
```

在 Epoch 59/1000 時出現 early stopping，test loss : 0.00013364345068112016，test accuracy : 1。

```
Epoch 59/1000
24/24 [==============================] - 276s 12s/step - loss: 0.0688 - accuracy: 0.9739 - val_loss: 1.2249e-04 - val_accuracy: 1.0000
Epoch 00059: early stopping
```

將訓練與測試結果繪製成圖。




模型的 loss 沒有再繼續掉下去，後面幾乎持平，因此判定沒有 under fitting。

# 四、 結論與未來展望

## 1. 結論

利用 Kaggle 上的數據集來進一步作分析和訓練，對資料的合理性和正確性做過濾，再將 RGB 圖像轉換至 0 到 1 之間，提升訓練效率，可以發現 CNN 模型的結果都不錯，透過調整激活函數(activation function)的種類、優化器(optimizer)的層數等，來比較模型在該參數組合下的準確率。在調整後(調整過程如表)，發現模型沒有 under fitting，且準確率相較於第一個參數組合之模型高，所以以下表之模型結構作為最終模型，並求得最終的準確率為 100%。

| Parameter | Model 1 | Model 2 |
|---|---|---|
| Convolution layer | Conv2D | Conv2D |
| Pooling layer | MaxPool2D | MaxPool2D |
| Activation | Relu | Relu |
| Last-layer activation | Sigmoid | Softmax |
| Optimizer | Adam(lr= 0.01) | Adam(lr= 0.001) |
| Loss | categorical_crossentropy | categorical_crossentropy |
| Patience | 5 | 10 |
| Batch-size | 1024 | 512 |
| Epoch | 100 | 1000 |
| Val_accuracy | 99.70% (under fitting) | 100.00% |

## 2. 未來展望

本次專題只有做水果分類，未來可以擴大至店家所有商品項目進行訓練，並連結結帳系統，達到影像辨識自助結帳的目標，降低顧客排隊時間，以及減少店家雇用收銀員的人事成本。

# 五、 參考資料


1. 剖析深度學習 (4)：Sigmoid, Softmax 怎麼來？為什麼要用 MSE 和 Cross Entropy？談廣義線性模型 from https://www.ycc.idv.tw/deep-dl_4.html

2. 使用 TensorFlow 學習 Softmax 回歸(Softmax Regression) from https://medium.com/%E6%89%8B%E5%AF%AB%E7%AD%86%E8%A8%98/%E4%BD%BF%E7%94%A8-tensorflow-%E5%AD%B8%E7%BF%92-softmax-%E5%9B%9E%E6%AD%B8-softmax-regression-41a12b619f04

3. [資料分析&機器學習] 第 5.1 講: 卷積神經網絡介紹 (Convolutional Neural Network) from https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC5-1%E8%AC%9B-%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1%E4%BB%8B%E7%B4%B9-convolutional-neural-network-4f8249d65d4f