智慧化企業整合

Project3

基於深度學習辨識肉品新鮮與否

110034541 李顥廷

指導教授：邱銘傳 教授

目錄

一、背景介紹

1-1. 情境描述

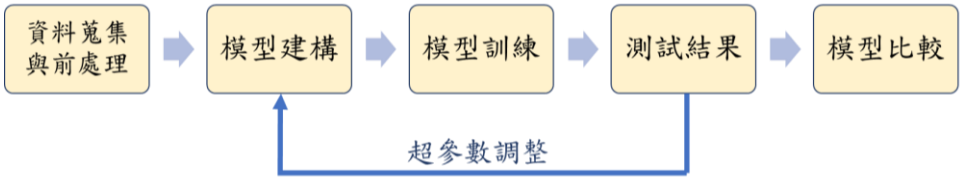
肉是我們日常生活中不可或缺的蛋白質來源，不論是對於我們的健康或是味蕾，有好的肉類主食對我們的正餐是非常重要的。然而，我們都會有過吃到的肉品不新鮮的時候，肉品的不新鮮可能造成烹調的口感不佳；不好聞的氣味；嚴重的可能進一步導致腸胃不適的症狀，因此如何挑選新鮮的肉品來避免產生負面的影響，讓我們能享受更美味的佳餚變成了需要探討的課題。但有過買菜經驗的人就知道，市面上的肉品看起來都大同小異，常常我們當下看覺得肉應該是新鮮的吧?到家拆封時卻發現其實不然，它其實有點腐壞的跡象，但我們肉眼卻無法分辨出來，當下一定會非常的失望，因為少了一道料理還花了冤枉錢成了冤大頭。為了避免類似的情況發生，若能藉由深度學習分辨圖形優異的能力來幫助我們辨認肉品是否新鮮來幫助我們選購肉品，會有很大的幫助。

1-2. 5W1H

Who	一般民眾、餐廳、食材供應商
What	辨認圖片肉品的新鮮與否
When	當看到肉品時不確定是否新鮮
Where	菜市場、餐廳、肉品工廠
Why	肉品的新鮮程度無法輕易由肉眼分辨
How	深度學習、機器學習、資料分析

1-3. 計畫流程

本研究的流程如下圖。首先蒐集肉品的圖片，並針對所蒐集的圖片進行前處理，接著建構模型，並對其訓練，接著測試模型的訓練結果，並針對本研究建立的模型進行模型比較。



二、資料蒐集與整理

2-1. 資料來源

資料源自於 kaggle 網站的 Meat Quality Assessment Dataset 資料集，其中含新鮮的肉品圖片 948 張及不新鮮的肉品圖片 948 張，共 1896 張圖片。所有的肉品圖片都是透過網絡攝影機拍攝，圖像解析度為 1280 x 720。本研究希望透過這個資料集來幫助建立一個肉品品質的分辨系統，幫助我們選購到好的肉品。

2-2. 資料前處理

2-2-1. 圖片生成

因為擔心資料數據僅有 1896 張無法有較佳的訓練結果，因此先利用圖片生成的方式，利用轉換角度及平移的方式隨機對每張照片進行處理，經過這樣的轉換對電腦而言會是一張全新的圖片，依此原理讓一張圖片以 5 倍的方式增加，達到圖片資料集擴增的效果，進而增加機器能學習的樣本數。

```
datagen = ImageDataGenerator(
        rotation_range = 30,
        width_shift_range = 0.2,
        height_shift_range = 0.2,
        horizontal_flip = True,
        vertical_flip = True,
        shear_range = 0.4
        )
```

```
for meat in range(1):
    fnames = [os.path.join(fresh_valid_dir, fname) for fname in os.listdir(fresh_valid_dir)]
    k = 0
    for j in fnames:
        img_path = j
        img = image.load_img(img_path, target_size = (240, 240))
        x = image.img_to_array(img)
        x = x.reshape((1, ) + x.shape)
        i = 0
        for batch in datagen.flow(x, batch_size = 1):
            plt.figure(i)
            implot = plt.imshow(image.array_to_img(batch[0]))
            implot = plt.axis('off')
            plt.savefig("/content/drive/MyDrive/meat1/Fresh/valid/copy_{}.jpg" . format(k))
            k += 1
            i += 1
            if i % 5 == 0:
                break
```

2-2-2. Colab 導入資料集並載入相關套件

將資料集上傳雲端於 colab:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

載入相關套件:

```
# import section for the notebook

import os
import pathlib
import tensorflow as tf
import numpy as np
import pandas as pd
import cv2
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.optimizers import RMSprop,SGD,Adam,Adagrad
from sklearn import metrics
from sklearn.model_selection import train_test_split

# use notebook's built in display
%matplotlib inline
```

2-2-3. 建立目錄及切分訓練集、驗證集、測試集

建立目錄:

```
# create image path
image_dir = "/content/drive/MyDrive/meat"
path_list = os.listdir(image_dir)

classes = []
file_paths = []
labels = []
```

切分訓練集、驗證集、測試集:

將資料以 80/10/10 的比例，依序分為訓練集、驗證集、測試集

```
# Split data into training, validation, and test datasets
train_split = .8 # 80% data for training
train_dataframe , rem_dataframe = train_test_split(meat_dataframe, train_size = train_split, shuffle = True, random_state = 16)

# Split remaining 20% evenly between test and validation
test_dataframe, val_dataframe = train_test_split(rem_dataframe, train_size = .5, shuffle = True, random_state = 16)
```

2-2-4. 將圖像歸一化

使用 ImageDataGenerator 來讓 train、validation、test 的像素值從[0-255]壓縮至[0-1]，class_mode 設為 categorical、圖片大小 target_size 設為(224,224)，shuffle 設定是否打亂數據，batch_size 設為 96。

程式結果呈現 train 共有 7584 筆、validation 共有 948 筆、test 共有 948 筆，分為兩個 classes。

```
batch_size = 96

# Scaling function for an image, scales to 0-1
def scalar(x):
    return x/255.0

# Training Generator: Vertical and Horizontal Flips, using scalar function
tgen = tf.keras.preprocessing.image.ImageDataGenerator(preprocessing_function=scalar,
                                                       horizontal_flip = True,
                                                       vertical_flip = True)

train_gen = tgen.flow_from_dataframe(train_dataframe, x_col='file_paths', y_col='label',
                                     target_size = (224, 224), class_mode = 'categorical',
                                     batch_size=batch_size, subset = 'training', shuffle = True)

# Validation Generator: Vertical and Horizontal Flips, using scalar function
vgen = tf.keras.preprocessing.image.ImageDataGenerator(preprocessing_function=scalar,
                                                       horizontal_flip = True,
                                                       vertical_flip = True)

val_gen = vgen.flow_from_dataframe(val_dataframe, x_col = 'file_paths', y_col = 'label',
                                   target_size = (224, 224), class_mode = 'categorical',
                                   batch_size = batch_size, shuffle = False)

# Test Generator: Vertical and Horizontal Flips, using scalar function
tsgen = tf.keras.preprocessing.image.ImageDataGenerator(preprocessing_function=scalar,
                                                        horizontal_flip = True,
                                                        vertical_flip = True)

test_gen = tsgen.flow_from_dataframe(test_dataframe, x_col = 'file_paths', y_col = 'label',
                                     target_size = (224, 224), class_mode = 'categorical',
                                     batch_size = batch_size, shuffle = False)
```

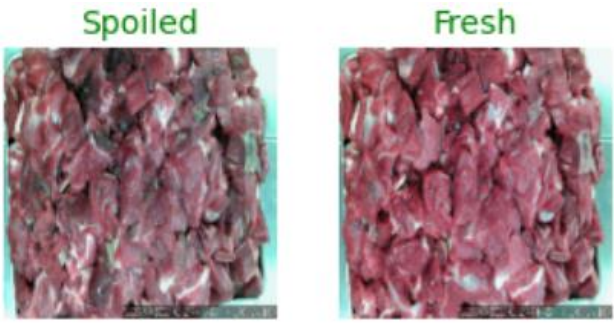
2-2-6. 視覺化呈現肉品圖片及新鮮與否標籤

```
# show a sample of the images from the provided generator
def show_generated_sample(generator):
    class_dict = generator.class_indices
    label_dict = {}

    # reverse value and keys: i.e. from 'Fresh':0 to 0:'Fresh'
    for key, value in class_dict.items():
        label_dict[value]=key

    # get a sample batch of images and labels
    imgs,labs = next(generator)

    plt.figure(figsize = (10, 10)) # create plot figure of size (10, 10)
    length = len(labels)
```



三、模型建構

3-1. 模型介紹

3-1-1. CNN

卷積神經網絡(Convolutional Neural Network)簡稱 CNN，其在影像識別上具有非常良好的成效，許多影像辨識模型也是以 CNN 為架構進行延伸，另外 CNN 也是少數參考人類大腦視覺組織來建立的深度學習模型，其架構會包含：

1. 卷積層 Convolution Layer

將原始圖片與特定的 Feature Detector(filter)做卷積運算

2. 池化層 Pooling Layer

主要採用 Max pooling， Max pooling 的好處是當整張圖片平移幾個 Pixel 對判斷上完全不會造成影響，且具有很好的抗雜訊功能。

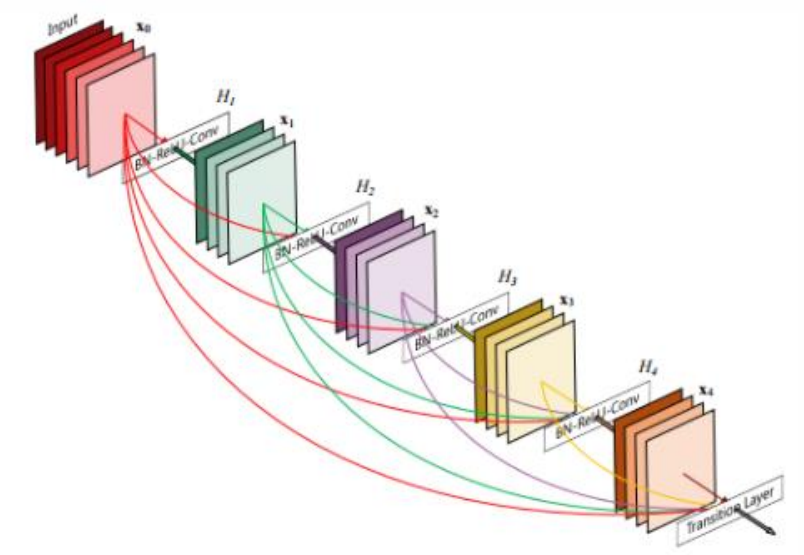
3. 全連接層 Fully Connected Layer

將先前的結果拉直，就可以接到基本的神經網路，之後即可進行分類。

3-1-2. Densenet

在本小組研究的過程中，發現由 CNN 所延伸發展出的稠密卷積網路(Densenet)，此網路不是用很深或很寬的網路來獲得呈現圖像辨識的能力，他是以前饋方式，將每層連結到每個其它層，不像傳統具有 L 層的卷積網絡有 L 個連接，而是每個層與其後一個層之間，又有 L(L+1)/2 個直接連接，並透過特徵的重複使用來得到網絡的隱含信息，獲得更容易訓練、參數效率更高的稠密模型。

Dense block 的結構圖：



每一層的輸入來自前面所有層的輸出。就是說 x_0 是 input，H_1 的輸入是 x_0（input），H_2 的輸入是 x_0 和 x_1 （x_1 是 H1 的輸出）。

Densenet 模型架構：

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

總結：

Densenet 的優點包含緩解梯度消失問題、加強特徵傳播、大幅減少參數數量；缺點為訓練的過程很佔記憶體，在本研究利用此方法訓練的過程中也有明顯的感受到訓練時間冗長的狀況。

3-2. 模型建立與訓練

3-2-1. CNN

建立一卷積神經網路模型：

(1) 建立卷積層(filter = 64)

(2) 建立池化層

(3) 建立卷積層(filter = 64)

(4) 建立池化層

(5) 建立卷積層(filter = 64)

(6) 建立池化層

(7) 建立卷積層(filter = 64)

(8) 建立池化層

(9) 進行 flatten 拉直成 1-D 陣列

```
# Hyperparameters for the Neural Networks
kernel_size = 3
pool_size = 2
latent_dim = 16
filters = 64
layer_filters = [32, 64]
dropout = 0.40
Epochs = 20
img_shape = np.shape(imgs)[1:4]


classifier = tf.keras.models.Sequential()
# First Layer
classifier.add(tf.keras.layers.Conv2D(filters = filters, kernel_size=kernel_size,
                                      activation = 'tanh', input_shape = img_shape))
classifier.add(tf.keras.layers.MaxPooling2D(pool_size))
# Second Layer
classifier.add(tf.keras.layers.Conv2D(filters = filters, kernel_size = kernel_size,
                                      activation = 'tanh'))
classifier.add(tf.keras.layers.MaxPooling2D(pool_size))
# 3rd Layer
classifier.add(tf.keras.layers.Conv2D(filters = filters, kernel_size = kernel_size,
                                      activation = 'tanh'))
classifier.add(tf.keras.layers.MaxPooling2D(pool_size))
# 4th Layer
classifier.add(tf.keras.layers.Conv2D(filters = filters, kernel_size = kernel_size,
                                      activation = 'tanh'))
classifier.add(tf.keras.layers.MaxPooling2D(pool_size))

classifier.add(tf.keras.layers.Flatten())
classifier.add(tf.keras.layers.Dropout(dropout))

classifier.add(tf.keras.layers.Dense(500))
classifier.add(tf.keras.layers.Dense(2))
classifier.add(tf.keras.layers.Activation('softmax'))
```

3-2-2. Densenet

建立一稠密卷積神經網路(Densenet)模型:

```
pretrained_model13 = tf.keras.applications.DenseNet201(input_shape=(100,100,3),include_top=False,weights='imagenet',pooling='avg')
pretrained_model13.trainable = False
```

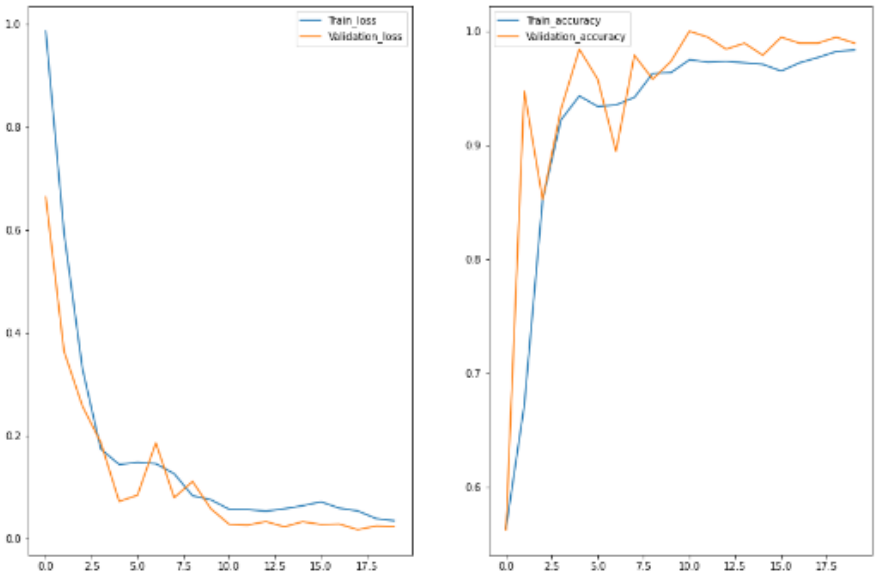
```
inputs3 = pretrained_model13.input
x3 = tf.keras.layers.Dense(128, activation='relu')(pretrained_model13.output)
outputs3 = tf.keras.layers.Dense(2, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.summary()
```

此模型採用 Densenet 201 架構，包含 4 層的 Dense Block 及 3 層的 Transition Layer，並利用 relu 及 softmax 作為激活函數，來幫助本研究分辨肉品新鮮與否。

四、參數優化

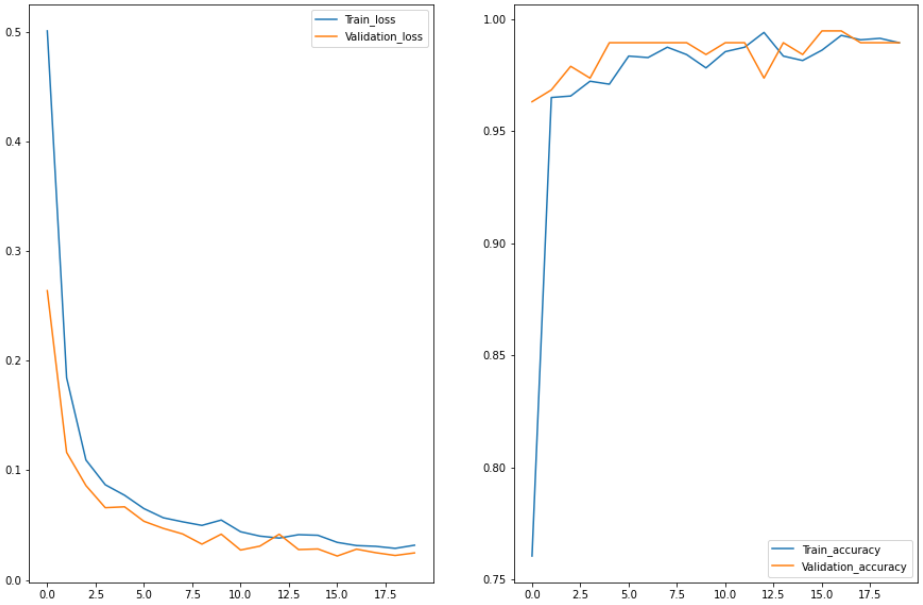
4-1. 模型訓練

以 3-2-1 的 CNN 模型，嘗試先跑 epoch＝20 來看訓練結果：



由上圖結果可以發現，在 epoch＝20 時，validation 的 loss 和 train 的 loss 有逐漸平穩，因此本研究後續進行模型提升、超參數優化，都將 epoch 設為 20 來訓練。

以 3-2-2 的 Densenet 201 架構，一樣以 epoch＝20 來看訓練結果：



由上圖結果可以發現，在 epoch＝20 時，validation 的 loss 和 train 的 loss 有

逐漸平穩，因此本研究後續進行模型提升、超參數優化，都將 epoch 設為 20 來訓練。

4-2. 實驗設計方法介紹

實驗設計是統計學的一種方法，透過合理地挑選試驗條件，並通過對試驗資料的分析，從而建立水準與因數之間的函數關係，找出總體最優的改進方案。實驗設計最大的目的就是能減少試驗次數，本研究欲利用此優勢來對模型的超參數調整，藉此能在有科學依據下，更有效率地找出最適合的模型參數組合。

下圖為 CNN 模型所考慮的因子及水準:

Factor \ Level	Level 1	Level 2	Level 3
Dropout	0.2	0.3	0.4
Optimizer	Adam	SGD	Adagrad
Activate function	relu	Tanh	sigmoid

下圖為 Densenet 201 模型所考慮的因子及水準:

Factor \ Level	Level 1	Level 2	Level 3
Learning rate	0.001	0.0005	0.0001
Optimizer	Adam	SGD	Adagrad
Activate function	relu	Tanh	sigmoid

4-3. 參數優化結果

因本研究考慮 3 因子及 3 水準，因此利用 L9 直交表來做為超參數優化的依據，幫助我們在減少試驗次數的同時找出適當的參數組合。

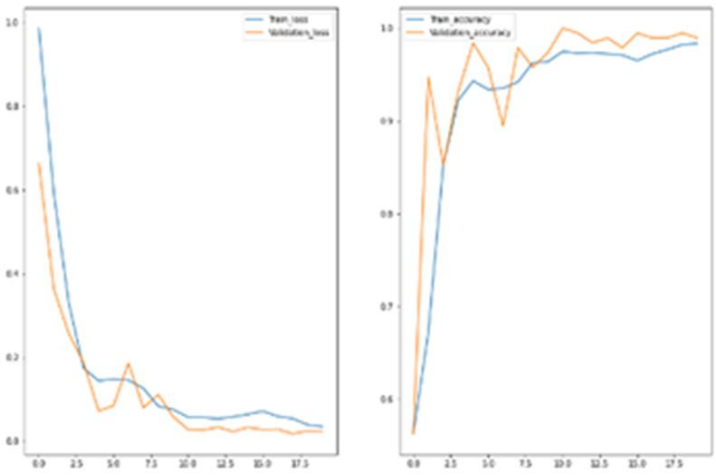
4-3-1. CNN 模型直交表:

	Dropout	Optimizer	Activate function	train acc	test acc
1	0.2	adam	relu	0.9842	0.9789
2	0.2	SGD	tanh	0.9565	0.9474

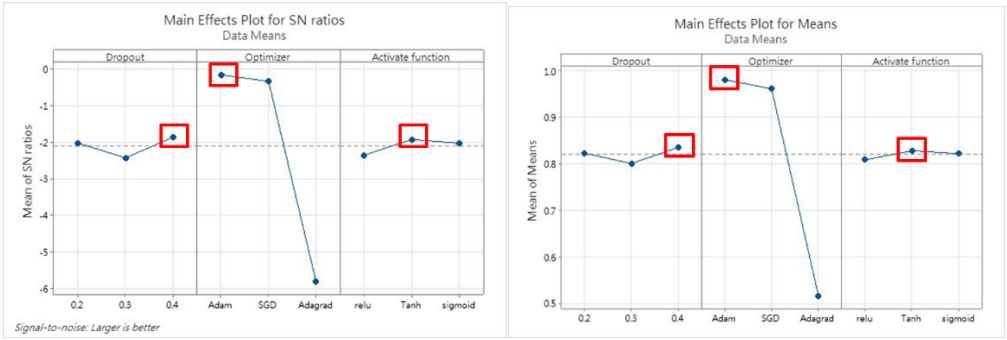
3	0.2	Adagrad	sigmoid	0.498	0.5737
4	0.3	adam	relu	0.9835	0.9842
5	0.3	SGD	tanh	0.9532	0.9521
6	0.3	Adagrad	sigmoid	0.5053	0.4263
7	0.4	adam	relu	0.9822	0.9737
8	0.4	SGD	tanh	0.9842	0.9737
9	0.4	Adagrad	sigmoid	0.5251	0.5737

從直交表的結果可看出最佳的模型為模型 4，其最佳參數組合為 Dropout = 0.3、Optimizer = adam、Activate function = relu、Train accuracy = 0.9835、Test accuracy = 0.9842。

下圖為利用 CNN 模型跑出的 model 4 結果:

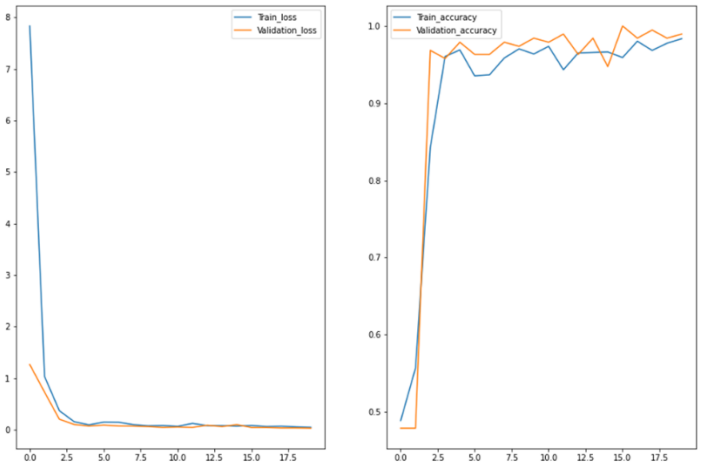


利用 Minitab 尋找最佳參數組合:



從上表的主效應圖可看出最佳參數組合為 Dropout = 0.4、Optimizer = adam、Activate function = Tanh，因此再利用這組參數做一次模型訓練觀看其結果。

進一步調整參數結果如下圖:



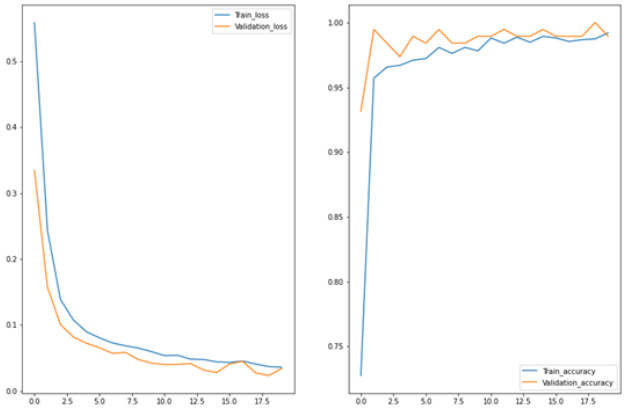
其訓練準確率為 0.9835、測試準確率為 0.9737，雖然訓練準確率和模型 4 一樣好，可惜的是測試準確率並沒有比較高，但也有不錯的表現。

4-3-2. Densenet 模型直交表:

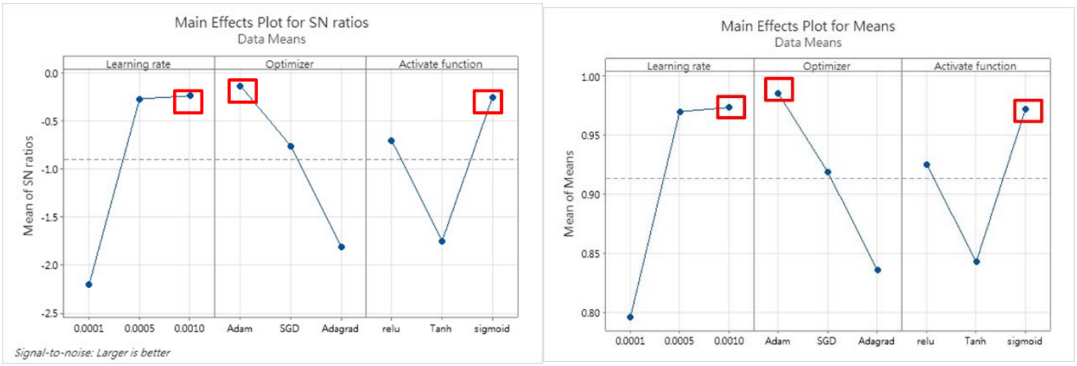
	Learning rate	Optimizer	Activate function	train acc	test acc
1	0.001	Adam	relu	0.9894	0.9789
2	0.001	SGD	tanh	0.9697	0.9562
3	0.001	Adagrad	sigmoid	0.9683	0.9789
4	0.0005	Adam	relu	0.9921	0.9895
5	0.0005	SGD	tanh	0.9631	0.9579
6	0.0005	Adagrad	sigmoid	0.965	0.9526
7	0.0001	Adam	relu	0.9835	0.9789
8	0.0001	SGD	tanh	0.8391	0.8263
9	0.0001	Adagrad	sigmoid	0.6016	0.5474

從直交表的結果可看出最佳的模型為模型 4，其最佳參數組合為 Learning rate = 0.0005、Optimizer = adam、Activate function = relu、Train accuracy = 0.9921、Test accuracy = 0.9895

下圖為利用 CNN 模型跑出的 model 4 結果:

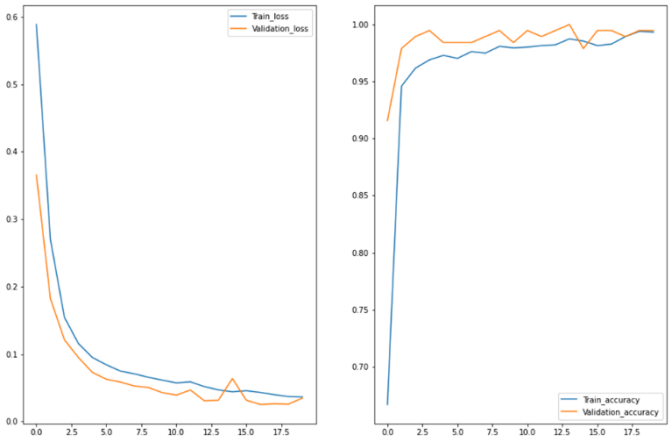


利用 Minitab 尋找最佳參數組合:



從上表的主效應圖可看出最佳參數組合為 Learning rate = 0.001、Optimizer = adam、Activate function = sigmoid，因此再利用這組參數做一次模型訓練觀看其結果。

進一步調整參數結果如下圖:



其訓練準確率為 0.9934、測試準確率為 0.9895，整體準確率為最高。

五、結論

5-1. 研究結果

本研究探討了 CNN 模型和 Densenet 模型在訓練上的差異，並透過實驗設計的方法確認了調整超參數對訓練集正確率、測試集正確率的影響，結果表明在 CNN 模型中，將 Dropout 設為 0.3、Optimizer 設為 adam、Activate function 設為 relu 能得到最好的結果。在 Densenet 模型中，將 Learning rate 設為 0.001、Optimizer 設為 adam、Activate function 設為 sigmoid 能得到最好的結果。下表為本研究所使用的兩種模型比較表，包含兩種模型的特點、差異、訓練過程的時間以及訓練結果準確率的情況。

	CNN	Densenet
全名	卷積神經網路	稠密卷積神經網路
特點	透過卷積和池化層，增進細節辨識能力	加強特徵傳播，特徵重用、減緩梯度消失問題
連接數(L 層)	L 個連接	L（L＋1）/ 2 個連接
訓練時間	較短	較長
測試結果	準確率略低(0.9842)	準確率略高(0.9895)

5-2. 研究限制

本研究的肉品新鮮與否的照片較侷限在同樣的肉品種類(豬肉)，若能增加不同的肉品種類，幫助我們能同時分辨出不同的肉品種類和其是否新鮮，能提供更好的泛化能力，能運用的範圍也能更廣泛。

5-3. 未來研究方向

未來能將本研究與網站、chatbot 做結合，在採購肉品時，民眾或廠商若對新鮮程度有疑慮時，即可將圖像上傳到所搭建的網站或 chatbot，即時確認圖像的真偽。亦可結合肉品工廠在原料進、出貨時，用於快速檢驗肉品是否新鮮。