基於深度學習辨識肉品新鮮與否

Group 9: 110034541 李顓廷

CONCENTS

1

背景介紹

1.1情境描述及目的

情境:

- 肉是我們日常生活中不可或缺的蛋白質來源
- 如何選擇新鮮的肉品對消費者或餐廳是非常重要的
- 有時無法有效辨別出差異

目的:

- 利用深度學習發展肉類新鮮程度判定系統

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

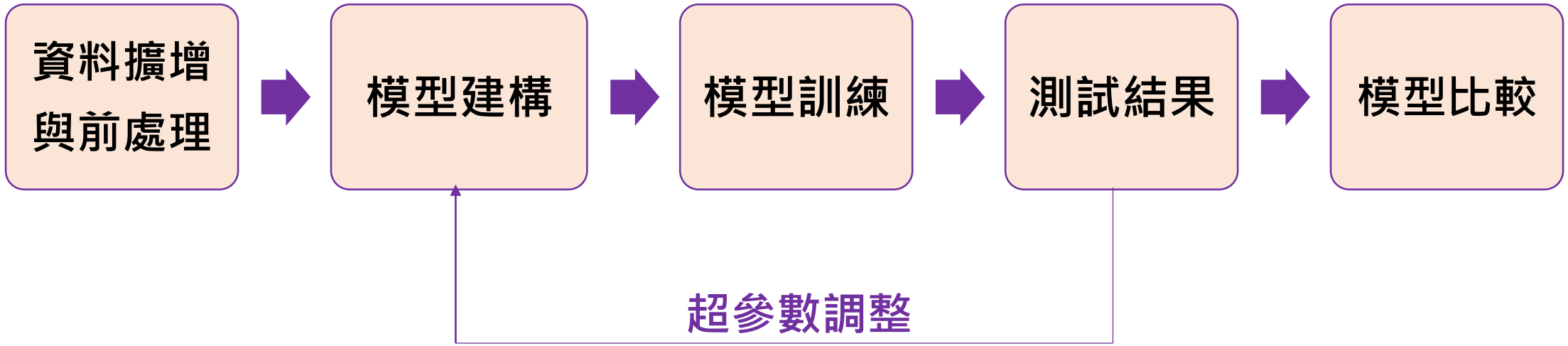
How
深度學習、資料分析

When
當看到肉品時不確定是否新鮮

Why
肉品新鮮程度無法輕易由肉眼分辨

Where
菜市場、餐廳、肉品工廠

1.3 計畫流程



2

資料擴增與前處理

2.1 資料收集

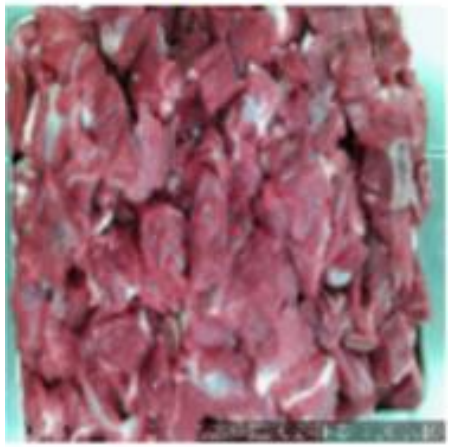
資料來源

Kaggle:
Meat Quality Assessment Dataset

原始資料集

1896張圖片
包含 Fresh :948 、Spoiled:948

Fresh



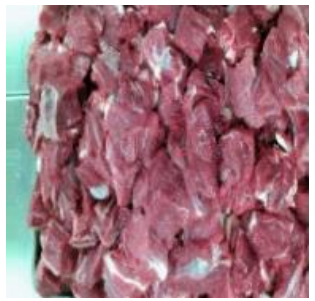
Spoiled

2.2 資料前處理

圖片生成:

- 擔心資料數據僅有1896張無法有較佳的訓練結果
- 利用轉換角度及平移的方式隨機對每張照片進行處理
- 讓一張圖片以5倍的方式增加為9480張
- 達到圖片資料集擴增的效果



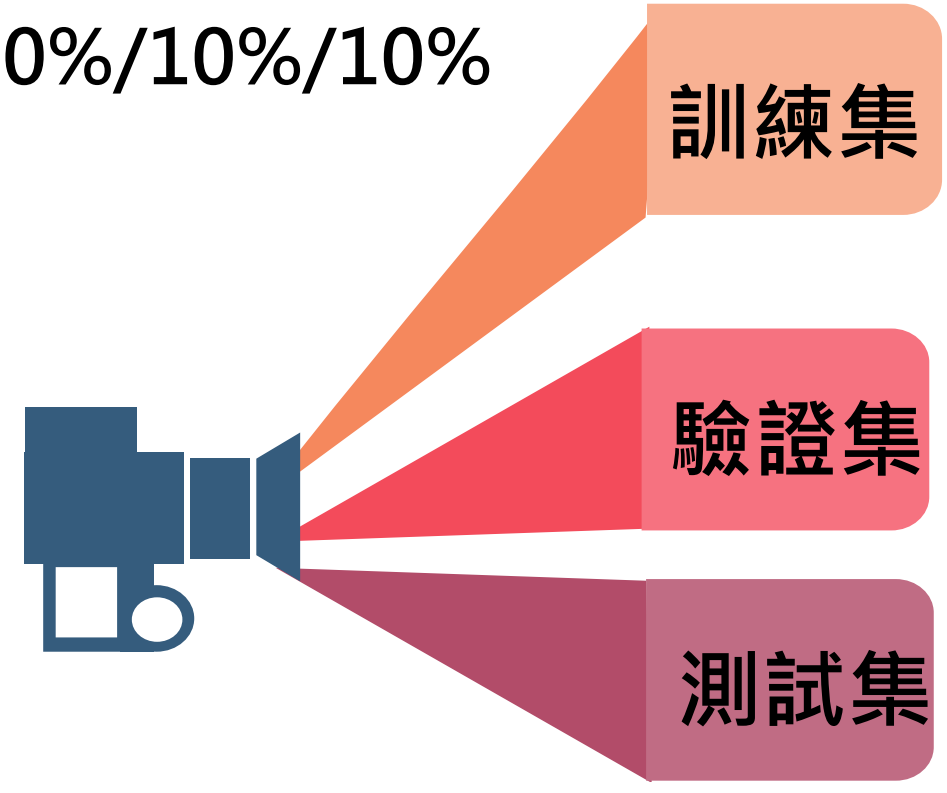



```
datagen = ImageDataGenerator(
        rotation_range = 30,
        width_shift_range = 0.2,
        height_shift_range = 0.2,
        horizontal_flip = True,
        vertical_flip = True,
        shear_range = 0.4
        )
```

2.2 資料前處理

切分資料集:

以80%/10%/10%



共7584張
分為Fresh:3792張 、Spoiled:3792張

共948張
分為Fresh:474張 、Spoiled:474張

共948張
分為Fresh:474張 、Spoiled:474張

```
# 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 資料前處理

圖像歸一化:

- 使用ImageDataGenerator來讓像素值從[0-255]壓縮至[0-1]
- Class mode設為 categorical
- 圖片大小target size設為(224,224)
- Shuffle設定打亂數據
- Batch size設為96

```
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)
```

3

模型建構與參數優化

3.1 模型介紹 - CNN

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

降低了各個特徵圖的維度，但可以保持大分重要的訊息。
池化層夾在連續的卷積層中間，壓縮數據和參數的量

3. 全連接層 Fully Connected Layer

將先前的結果拉直，配合輸出層進行分類

3.1 模型建立 - CNN

建立一卷積神經網路模型：

第一層layer的filters = 64

第二層layer的filters = 64

第三層layer的filters = 64

第四層layer的filters = 64

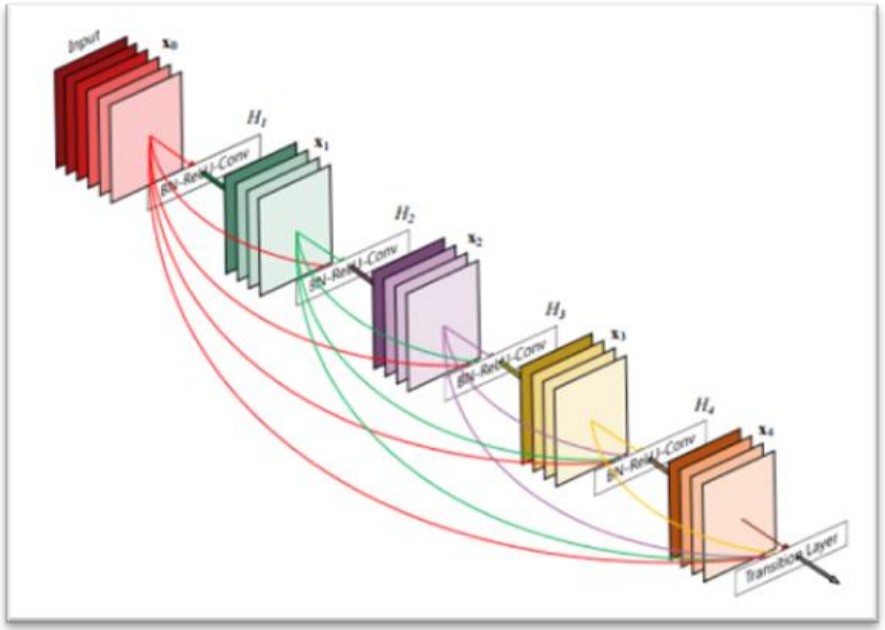
進行flatten拉直成1-D陣列

```
# 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'))
```

3.1 模型介紹 - Densenet

Densenet

- 為CNN所延伸發展出的稠密卷積網路
- 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，它是以前饋方式，將每層連結到每個其它層
- 每個層與其後一個層之間，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訊息，獲得更容易訓練、參數效率更高的稠密模型。



優點:

緩解梯度消失問題、加強特徵傳播、
大幅減少參數數量

3.1 模型建立 - Densenet

利用 Densenet 201 作為架構

```
pretrained_model3 = tf.keras.applications.DenseNet201(input_shape=(100,100,3),include_top=False,weights='imagenet',pooling='avg')
pretrained_model3.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201_weights_tf_dim_ordering_tf_kernels_notop.h5
74842112/74836368 [==============================] - 1s 0us/step
74850304/74836368 [==============================] - 1s 0us/step
```

```
inputs3 = pretrained_model3.input
x3 = tf.keras.layers.Dense(128, activation='sigmoid')(pretrained_model3.output)
outputs3 = tf.keras.layers.Dense(2, activation='softmax')(x3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
model.compile(optimizer= Adagrad(lr=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
model.summary()
```

模型架構圖:

```
conv5_block32_1_conv (Conv2D)   (None, 3, 3, 128)    241664      ['conv5_block32_0_relu[0][0]']

conv5_block32_1_bn (BatchNorma  (None, 3, 3, 128)    512         ['conv5_block32_1_conv[0][0]']
lization)

conv5_block32_1_relu (Activati  (None, 3, 3, 128)    0           ['conv5_block32_1_bn[0][0]']
on)

conv5_block32_2_conv (Conv2D)   (None, 3, 3, 32)     36864       ['conv5_block32_1_relu[0][0]']

conv5_block32_concat (Concaten  (None, 3, 3, 1920)   0           ['conv5_block31_concat[0][0]',
ate)                                                             'conv5_block32_2_conv[0][0]']

bn (BatchNormalization)         (None, 3, 3, 1920)   7680        ['conv5_block32_concat[0][0]']

relu (Activation)               (None, 3, 3, 1920)   0           ['bn[0][0]']

avg_pool (GlobalAveragePooling  (None, 1920)         0           ['relu[0][0]']
2D)

dense (Dense)                   (None, 128)          245888      ['avg_pool[0][0]']

dense_1 (Dense)                 (None, 2)            258         ['dense[0][0]']

==================================================================================================
Total params: 18,568,130
Trainable params: 246,146
Non-trainable params: 18,321,984
__________________________________________________________________________________________________
```

3.2 實驗設計 - CNN

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

3.2 實驗設計 - CNN

CNN模型參數組合:

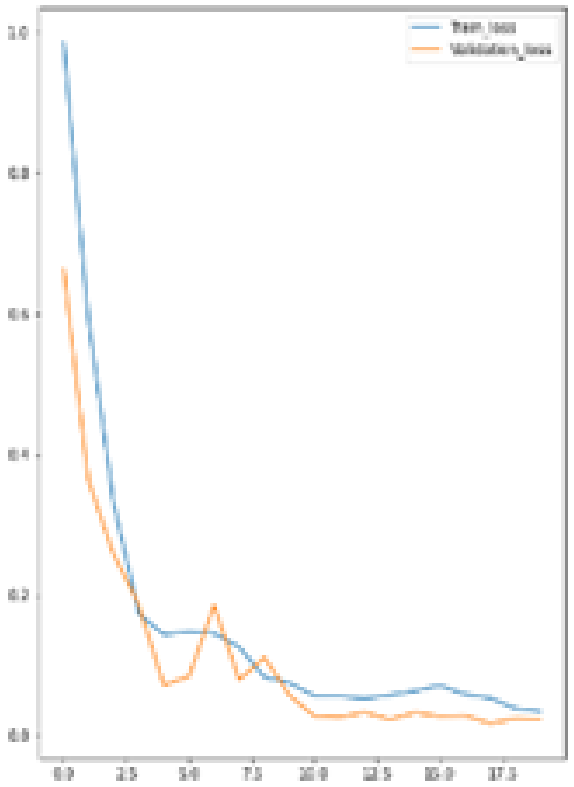
	Dropout	Optimizer	Activate function	train acc	test acc
1	0.2	adam	relu	0.9842	0.9789
2	0.2	SGD	tanh	0.9565	0.9474
3	0.2	Adagrad	sigmoid	0.498	0.5737
4	0.3	adam	relu	0.9835	0.9842
5	0.3	SGD	tanh	0.9532	0.9521
6	0.3	Adagrad	sigmoid	0.5053	0.4263
7	0.4	adam	relu	0.9822	0.9737
8	0.4	SGD	tanh	0.9842	0.9737
9	0.4	Adagrad	sigmoid	0.5251	0.5737

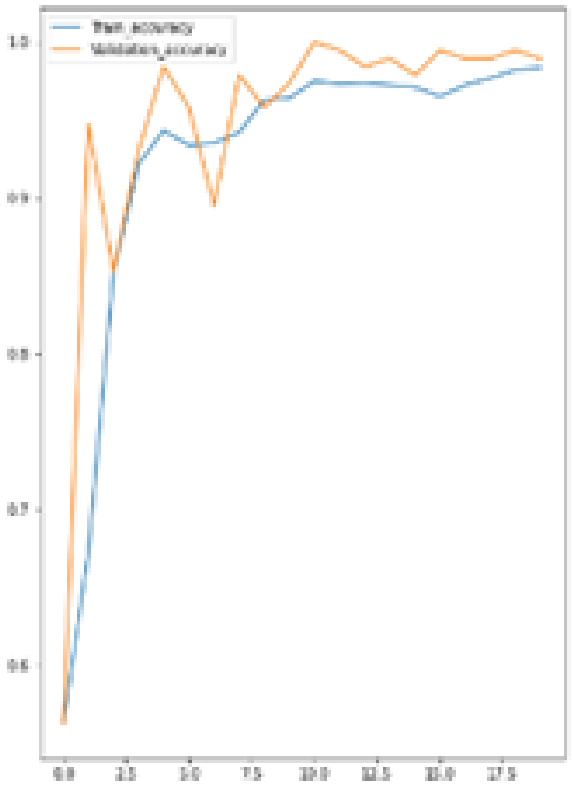
3.2 實驗設計 - CNN

CNN模型 model 4 結果:

參數組合: Dropout = 0.3、Optimizer = adam、Activate function = relu

Train accuracy = 0.9835、 Test accuracy = 0.9842





3.2 Minitab結果 -CNN

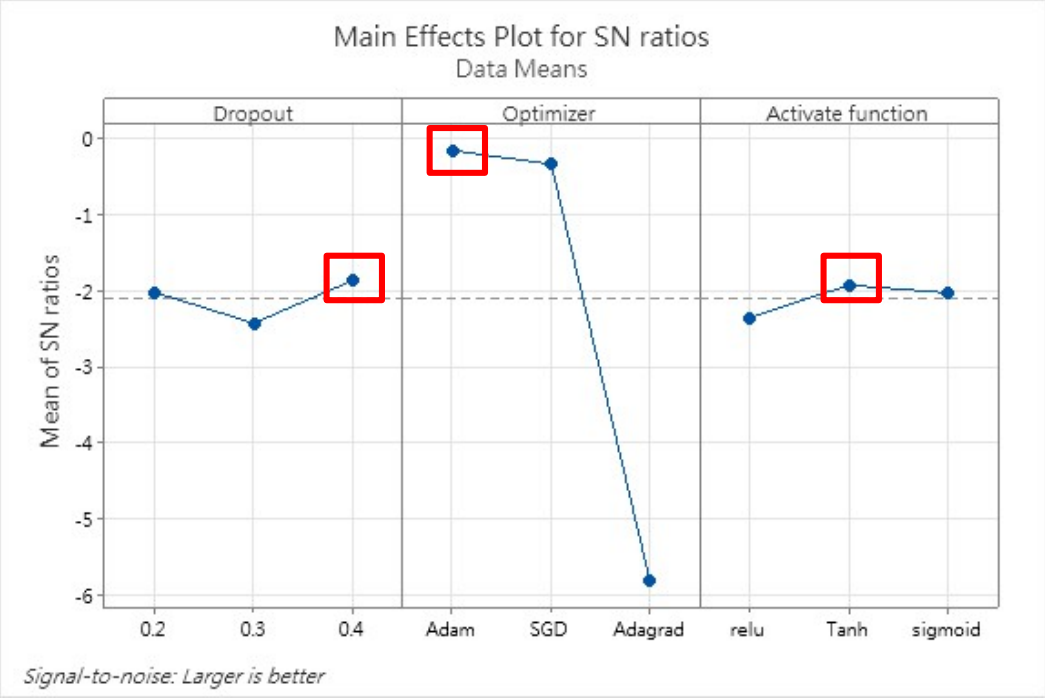
Response Table for Signal to Noise Ratios

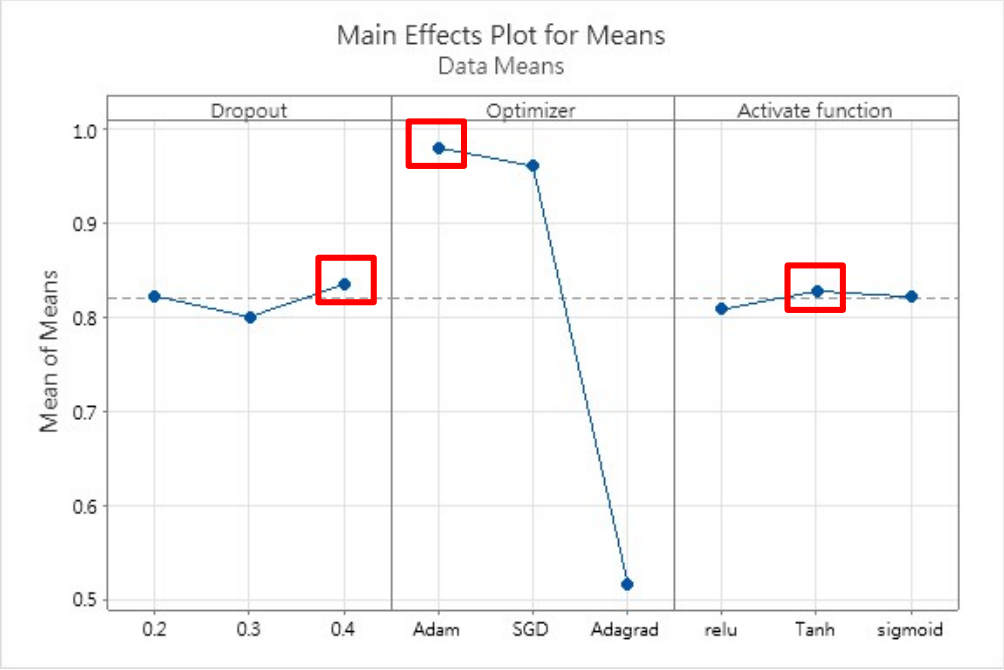
Larger is better

Level	Dropout	Optimizer	Activate function
1	-2.0247	-0.1657	-2.3589
2	-2.4309	-0.3448	-1.9324
3	-1.8689	-5.8139	-2.0331
Delta	0.5619	5.6482	0.4266
Rank	2	1	3

Response Table for Means

Level	Dropout	Optimizer	Activate function
1	0.8231	0.9811	0.8088
2	0.8008	0.9612	0.8284
3	0.8354	0.5170	0.8222
Delta	0.0347	0.4641	0.0196
Rank	2	1	3



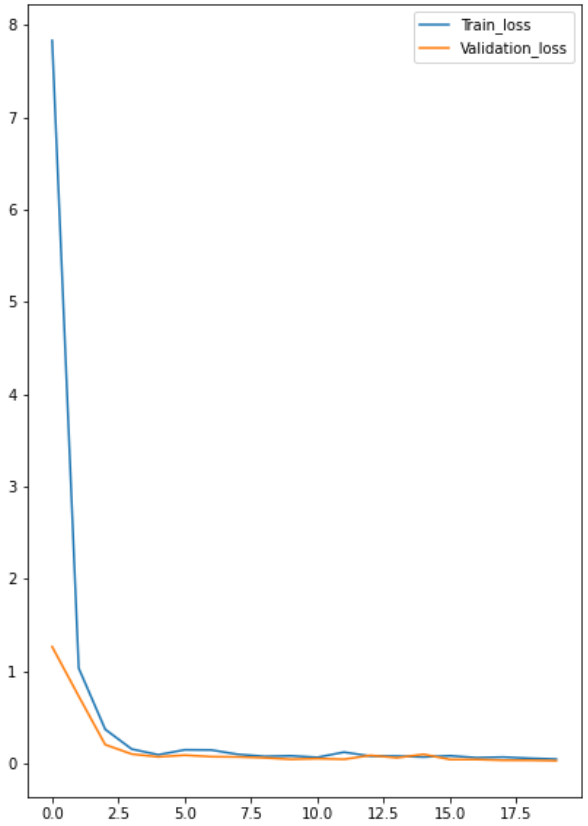


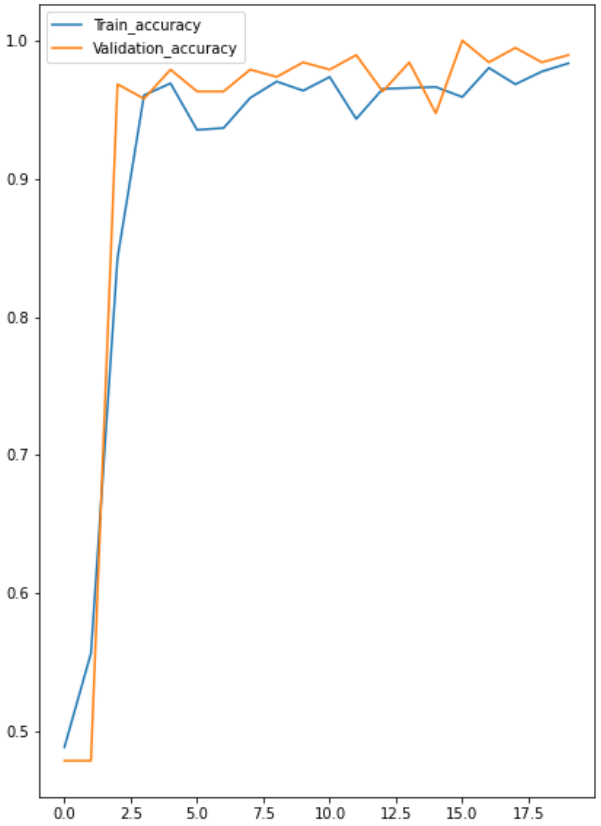
3.2 進一步調參 -CNN

參數組合:

Dropout = 0.4、Optimizer = adam、Activate function = Tanh

Train accuracy = 0.9835、 Test accuracy = 0.9737





3.2 實驗設計 -Densenet 201

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

3.2 實驗設計 - Densenet 201

Densenet 201模型參數組合:

	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

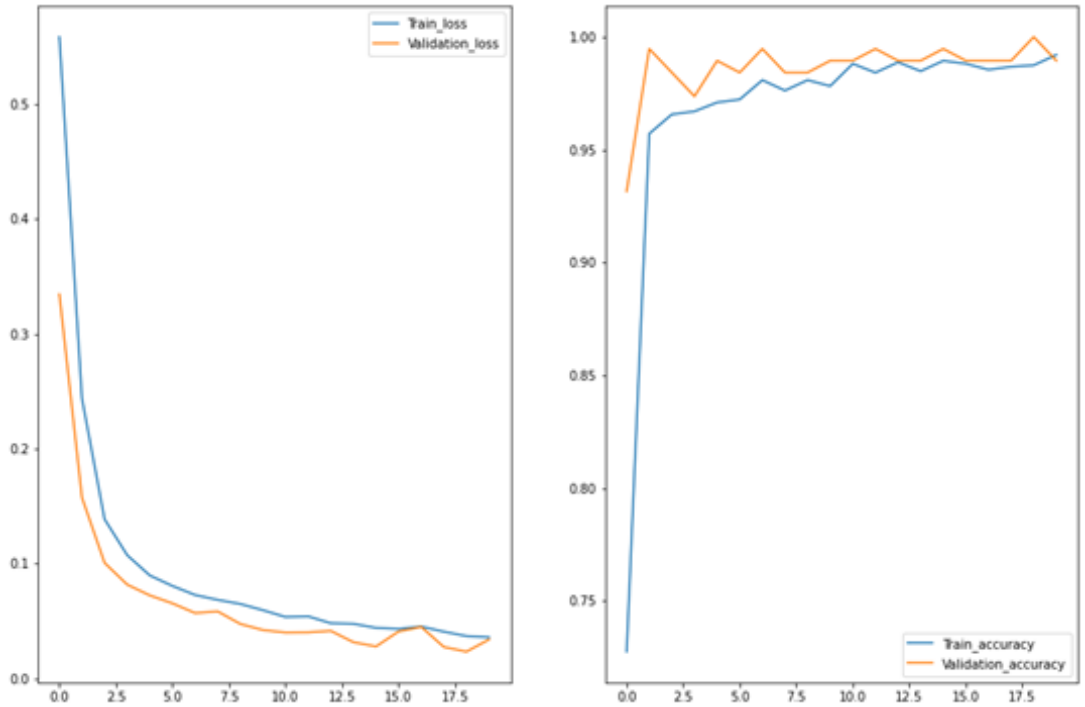
3.2 實驗設計 - Densenet 201

Densenet模型 model 4 結果:

參數組合:

Learning rate = 0.0005、Optimizer = adam、Activate function = relu

Train accuracy = 0.9921 、 Test accuracy = 0.9895



3.2 Minitab結果 - Densenet 201

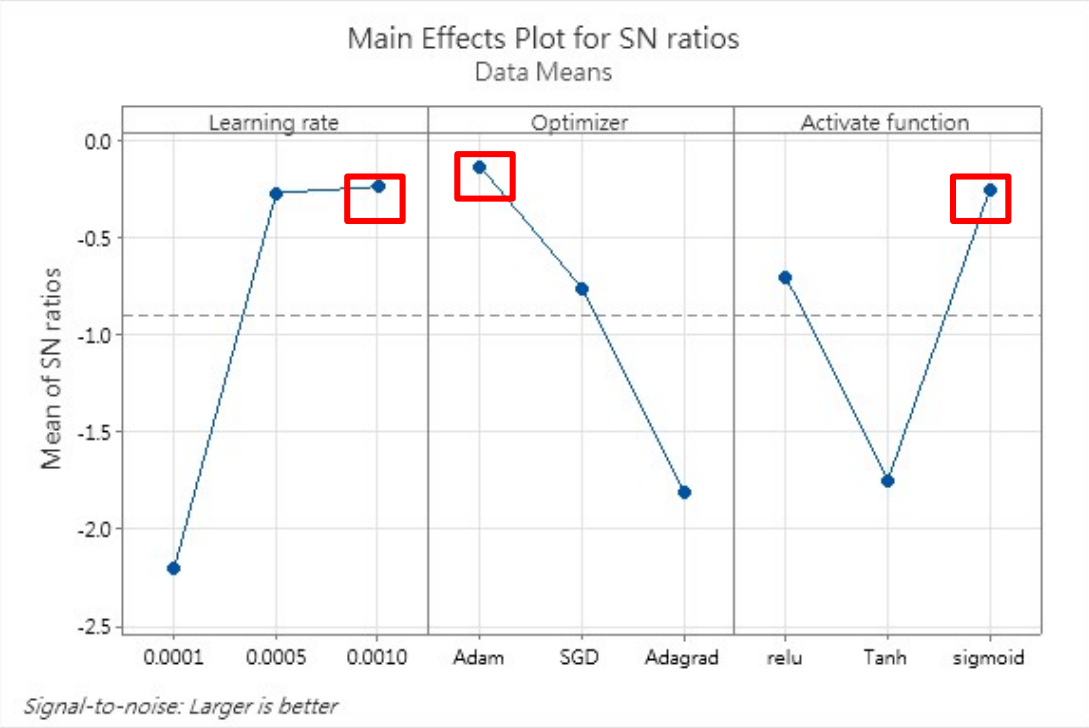
Response Table for Signal to Noise Ratios

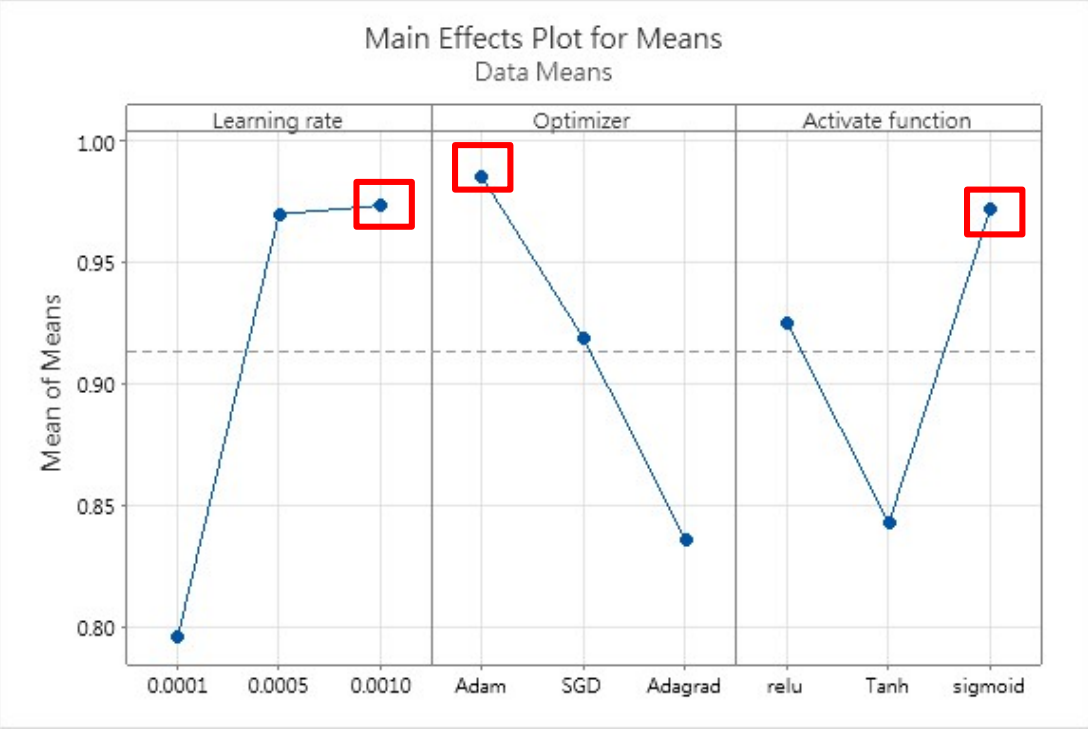
Larger is better

Level	Learning rate	Optimizer	Activate function
1	-2.1997	-0.1281	-0.6987
2	-0.2655	-0.7566	-1.7507
3	-0.2335	-1.8140	-0.2493
Delta	1.9662	1.6859	1.5014
Rank	1	2	3

Response Table for Means

Level	Learning rate	Optimizer	Activate function
1	0.7961	0.9854	0.9252
2	0.9700	0.9187	0.8427
3	0.9736	0.8356	0.9718
Delta	0.1774	0.1498	0.1290
Rank	1	2	3



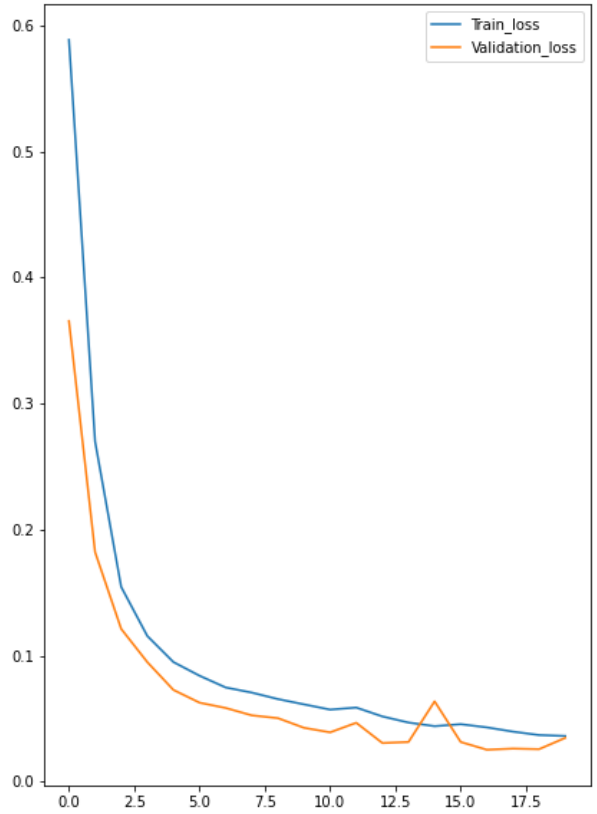


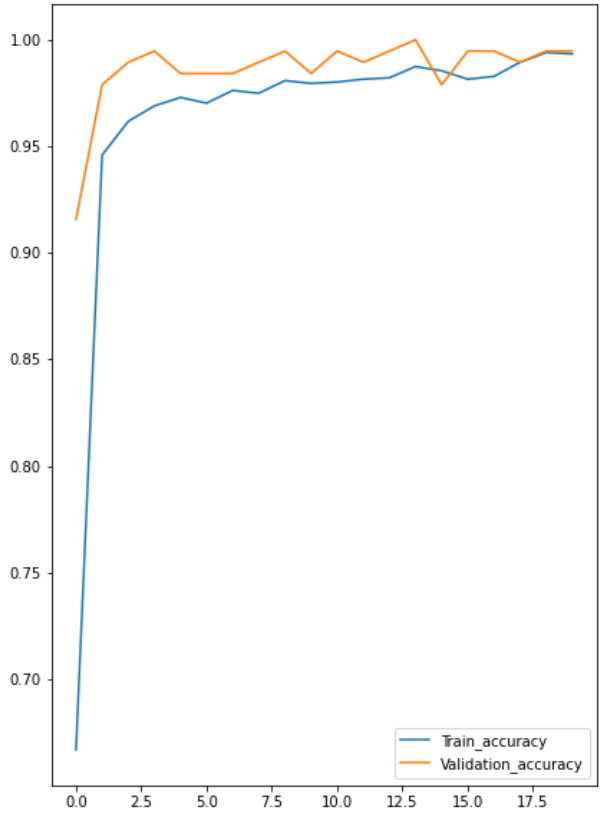
3.2 進一步調參 - Densenet 201

參數組合:

Learning rate = 0.001、Optimizer = adam、Activate function = sigmoid

Train accuracy = 0.9934 、 Test accuracy = 0.9895





4

結論
與
未來展望

4.1 研究結果

最佳參數組合:

CNN	**Dropout=0.3、Optimizer=adam、Activate function=relu**
Densenet	**Learning rate= 0.001、Optimizer=adam、Activate function=sigmoid**

模型比較:

	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)

4.2 研究限制與未來展望

研究限制:

- 照片較侷限在同樣的肉品種類(豬肉)
- 僅從資料集的類別分出是否新鮮，無較一致的準則判斷

未來展望:

- 辨別不同的肉品及其新鮮程度
- 與網站、chatbot做結合
- 與現場應用結合 ex.肉品工廠

THANK YOU