Classfication For Food Dataset

Content

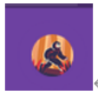
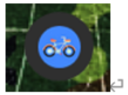
- 模型: 訓練過程/選擇/架構/Finetune
- 資料分析/處理
- 改optimizer
- 有無class weight 影響
- Top 1 Accuracy 最高的model探討好壞

中英對照(因為Kaggle檔案名稱需英文)

```
translation_dict = {
    '白米飯': 'White_Rice',
    '香蕉': 'Banana',
    '油菜': 'Rapeseed',
    '青江菜': 'Qingjiang_Cai',
    '高麗菜': 'Cabbage',
    '螞蟻上樹': 'Ants_Climbing_a_Tree',
    '麻婆豆腐': 'Mapo_Tofu',
    '有機小松菜': 'Organic_Baby_Kale',
    '三杯雞': 'Three_Cup_Chicken',
    '菠菜': 'Spinach',
    '有機青松菜': 'Organic_Qing_Pine_Cabbage',
    '鵝白菜': 'Goose_Cabbage',
    '洋蔥炒蛋': 'Onion_Scrambled_Eggs',
    '葡萄': 'Grape',
    '紅蘿蔔炒蛋': 'Carrot_Scrambled_Eggs',
    '番茄炒蛋': 'Tomato_Scrambled_Eggs',
    '玉米炒蛋': 'Corn_Scrambled_Eggs',
    '棗子': 'Jujube',
    '小番茄': 'Cherry_Tomato',
    '橘子': 'Orange',
    '客家小炒': 'Hakka_Stir-Fry',
    '白菜滷': 'Napa_Cabbage_Stew',
    '咖哩雞': 'Curry_Chicken',
    '柳丁': 'Mandarin_Orange',
    '關東煮': 'Oden',
    '蓮霧': 'Wax_Apple',
    '空心菜': 'Hollow_Heart_Vegetable',
    '大陸妹': 'Mainland_Sister',
    '蒜泥白肉': 'Garlic_Pork_Slices',
    '滷蛋': 'Marbled_Egg',
    '福山萵苣': 'Fushan_Lettuce',
    '滷雞腿': 'Braised_Chicken_Leg',
    '鳳梨': 'Pineapple',
    '木瓜': 'Papaya',
    '西瓜': 'Watermelon',
    '芥藍菜': 'Kale',
    '蒸蛋': 'Steamed_Egg',
    '麥克雞塊': 'McNuggets',
    '豆芽菜': 'Bean_Sprouts',
    '馬鈴薯燉肉': 'Potato_Stew_with_Meat',
    '沙茶肉片': 'Satay_Pork_Slices',
    '塔香海茸': 'Tower_Fragrance_Sea_Mushroom',
    '麻油雞': 'Sesame_Oil_Chicken',
    '鹽酥雞': 'Salt_and_Pepper_Chicken',
    '義大利麵': 'Spaghetti',
    '糖醋雞丁': 'Sweet_and_Sour_Chicken',
    '什錦炒麵': 'Assorted_Fried_Noodles',
    '黑胡椒豬柳': 'Black_Pepper_Pork_Fillet',
    '瓜仔肉': 'Melon_Pork',
    '香酥魚排': 'Crispy_Fish_Fillet',
```

模型訓練過程

- 4個帳號(四個路線)，綠色字體表示與前一版的的差異
- 一開始嘗試單一模型: 準確度Top1 Accuracy: 75%
- 嘗試雙模型: 一開始即上升至81%
- 改變模型搭配?
  - 嘗試過 Xception + EfficientNetV2L，或 EfficientNetV2L + Resnet50。但因為 Resnet50 + EfficientNet 在一開始epoch15 情況下準度少於另一版(82%)，因此沒有繼續嘗試。

Waywuwin: 	Adam_6e-4 X(116)_eff(964) 512 286 128 Flatten (81%)	Adam_9e-4 X(-4)_eff(994) 1024 512 286 Flatten Drop(0.2) (81) ep15	Adam_9e-4 X(-4)_eff(994) 1024 512 286 Flatten Drop(0.2) (84%) ep35	Adam_9e-4 X(-4)_eff(994) 1024 512 286 Flatten Drop(0.2) (84%) ep50		Model Structure Graph
Oneforkaggle: 	Adam_6e-4 X(116)_eff(964) 512 286 128 GlobalAvgPool (80%)	RMS_e-3 X(116)_eff(964) 1024 512 286 Flatten Dr0.2 (82%) ep15	RMS_e-3 X(116)_eff(964) 1024 512 286 Flatten Dr0.2 (84%) ep30	RMS_e-3 X(116)_eff(964) 1024 512 286 Flatten Dr0.2 (84%) ep50	RMS_e-3 X(116)_eff(964) 1024 512 256 128 Flatten Dr0.2 (84%) ep45	Nadam_e-3_Classweight_ep45_ 1024-512-256-128_Flatten_ Dr0.2 (85%)
Azonwu/n2612: 	RMS_6e-4 Resnet50(143) _eff(964) 512 286 128 Flatten , dr025 (81%)			(No classweight) RMS_e-3 X(116)_eff(964) 1024 512 256 Flatten Dr0.2 (85%) ep45		
C04: 	(Single Model) Adam_e-4 _1024_Dr0.4 (75%)				(No classweight) RMS_e-3 X(116)_eff(964) 1024 512 256 128 Flatten Dr0.2 (85%) ep45	Single Model_Confusion Matrix

Format:
Optmizer_learningRate_DenseLayer_Dropout_(Top1Accuracy)

Abbreviation:
- ep: epoch
- dr: dropout
- X(116): xception finetune from layer 116
- eff(964): EfficientNetv2L finetune starts from layer 964

模型選擇

- 一大一小

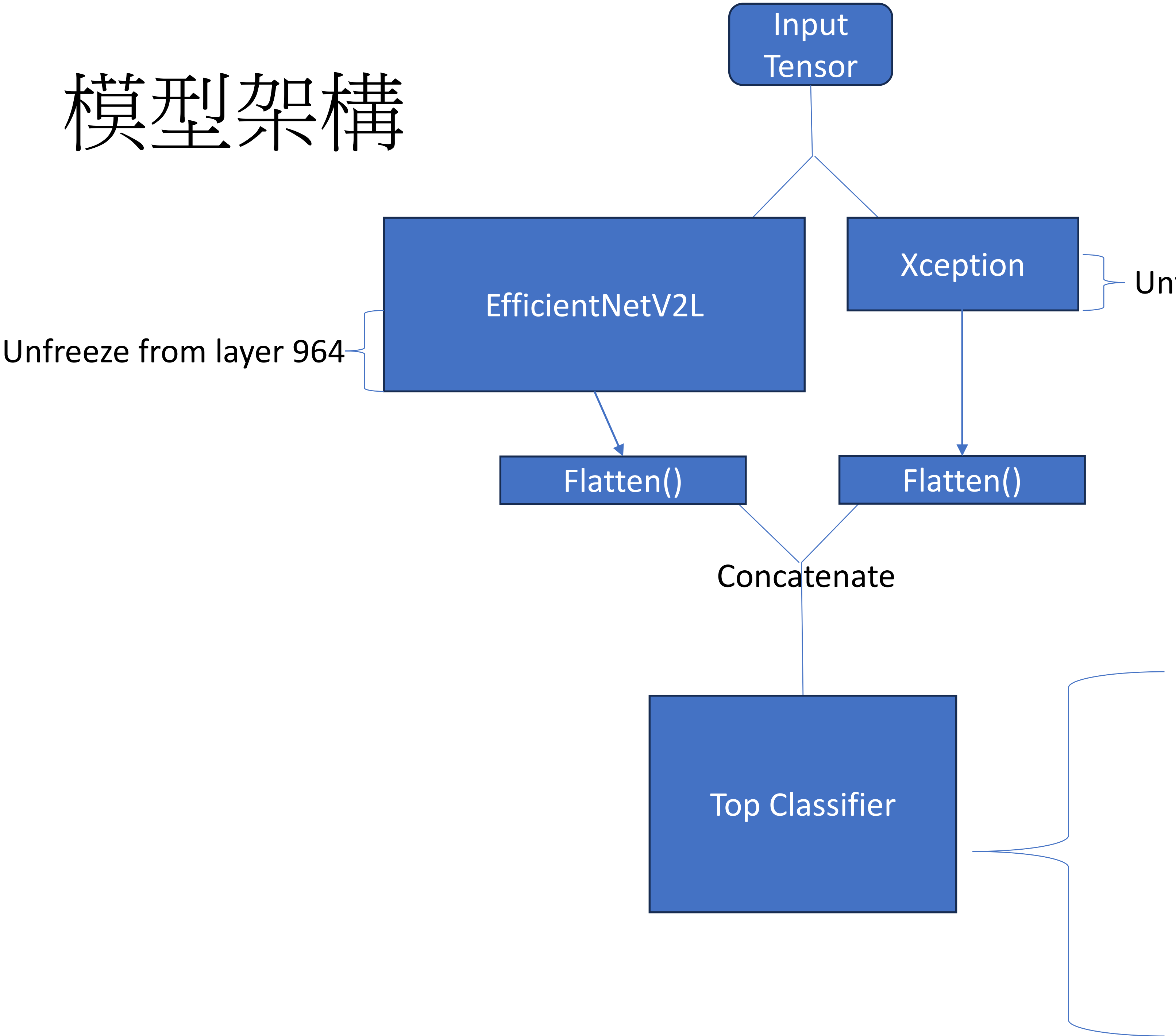
- Xception在相似大小模型中Top1, Top5表現次好的
- 原本使用EfficientNetV2S但因Layer名稱會與EfficientNetV2L重複，會出錯
- Resnet50的搭配不如Xception沒有在後續嘗試

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy
Xception	88	79.0%	94.5%
ResNet50	98	74.9%	92.1%
ResNet50V2	98	76.0%	93.0%
InceptionV3	92	77.9%	93.7%
DenseNet201	80	77.3%	93.6%
EfficientNetV2S	88	83.9%	96.7%

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	
ConvNeXtLarge	755.07	86.3%	-	197.7M		
ConvNeXtXLarge	1310	86.7%	-	350.1M		
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3
EfficientNetB5	118	83.6%	96.7%	30.6M	312	579.2
EfficientNetB6	166	84.0%	96.8%	43.3M	360	958.1
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9
EfficientNetV2B0	29	78.7%	94.3%	7.2M	-	-
EfficientNetV2B1	34	79.8%	95.0%	8.2M	-	-
EfficientNetV2B2	42	80.5%	95.1%	10.2M	-	-
EfficientNetV2B3	59	82.0%	95.8%	14.5M	-	-
EfficientNetV2S	88	83.9%	96.7%	21.6M	-	-
EfficientNetV2M	220	85.3%	97.4%	54.4M	-	-
EfficientNetV2L	479	85.7%	97.5%	119.0M	-	-

Top 1 Accuracy 第三高，但大小那麼ConvNeXtLarge大
Top 5 Accuracy 最高

模型架構




```python
# Efficient Net model
model_1 = EfficientNetV2L(
    include_top = False,
    weights = 'imagenet',
    input_tensor = input_tensor,
    classes = 50,
    include_preprocessing=False,
)
model_1.trainable = False  # set false trainable as default
unfreeze_model(model_1 , 964 ) #NC
output_1 = Flatten()(model_1.output)

# Xception model
model_2 = tf.keras.applications.Xception(
    weights="imagenet",
    include_top=False,
    input_tensor = input_tensor,
    classes = 50
)
model_2.trainable = False
unfreeze_model(model_2, 116) #NC
output_2 = Flatten()(model_2.output)


# Concatenate the outputs of Xception and EfficientNetV2-Large
concatenated_output = Concatenate()([output_1 , output_2 ])


# Top layer
x = Dropout(0.2)(concatenated_output)

x = BatchNormalization()(x)
x = Dense(1024, activation='relu')(x)

x = BatchNormalization()(x)
x = Dense(512, activation='relu')(x)

x = BatchNormalization()(x)
x = Dense(286, activation='relu')(x)

x = BatchNormalization()(x)
classifier_output = Dense(50, activation='softmax')(x)

# Create the combined model
combined_model = Model(inputs=input_tensor, outputs=classifier_output)
```

Finetune

- Xception: unfreeze from layer 116

```
111 block12_sepconv2_bn BatchNormalization
112 block12_sepconv3_act Activation
113 block12_sepconv3 SeparableConv2D
114 block12_sepconv3_bn BatchNormalization
115 add_10 Add
116 block13_sepconv1_act Activation
117 block13_sepconv1 SeparableConv2D
118 block13_sepconv1_bn BatchNormalization
119 block13_sepconv2_act Activation
120 block13_sepconv2 SeparableConv2D
121 block13_sepconv2_bn BatchNormalization
122 conv2d_3 Conv2D
123 block13_pool MaxPooling2D
124 batch_normalization_3 BatchNormalization
125 add_11 Add
126 block14_sepconv1 SeparableConv2D
127 block14_sepconv1_bn BatchNormalization
128 block14_sepconv1_act Activation
129 block14_sepconv2 SeparableConv2D
130 block14_sepconv2_bn BatchNormalization
131 block14_sepconv2_act Activation
```

- EfficientNetV2L: unfreeze from layer 964

```
961 block7c_project_bn BatchNormalization
962 block7c_drop Dropout
963 block7c_add Add
964 block7d_expand_conv Conv2D
965 block7d_expand_bn BatchNormalization
966 block7d_expand_activation Activation
967 block7d_dwconv2 DepthwiseConv2D
968 block7d_bn BatchNormalization
969 block7d_activation Activation
970 block7d_se_squeeze GlobalAveragePooling2D
971 block7d_se_reshape Reshape
972 block7d_se_reduce Conv2D
973 block7d_se_expand Conv2D
974 block7d_se_excite Multiply
975 block7d_project_conv Conv2D
976 block7d_project_bn BatchNormalization
977 block7d_drop Dropout
978 block7d_add Add
979 block7e_expand_conv Conv2D
980 block7e_expand_bn BatchNormalization
981 block7e_expand_activation Activation
982 block7e_dwconv2 DepthwiseConv2D
983 block7e_bn BatchNormalization
984 block7e_activation Activation
985 block7e_se_squeeze GlobalAveragePooling2D
986 block7e_se_reshape Reshape
987 block7e_se_reduce Conv2D
988 block7e_se_expand Conv2D
989 block7e_se_excite Multiply
990 block7e_project_conv Conv2D
991 block7e_project_bn BatchNormalization
992 block7e_drop Dropout
993 block7e_add Add
994 block7f_expand_conv Conv2D
995 block7f_expand_bn BatchNormalization
996 block7f_expand_activation Activation
997 block7f_dwconv2 DepthwiseConv2D
998 block7f_bn BatchNormalization
999 block7f_activation Activation
1000 block7f_se_squeeze GlobalAveragePooling2D
1001 block7f_se_reshape Reshape
1002 block7f_se_reduce Conv2D
1003 block7f_se_expand Conv2D
1004 block7f_se_excite Multiply
```

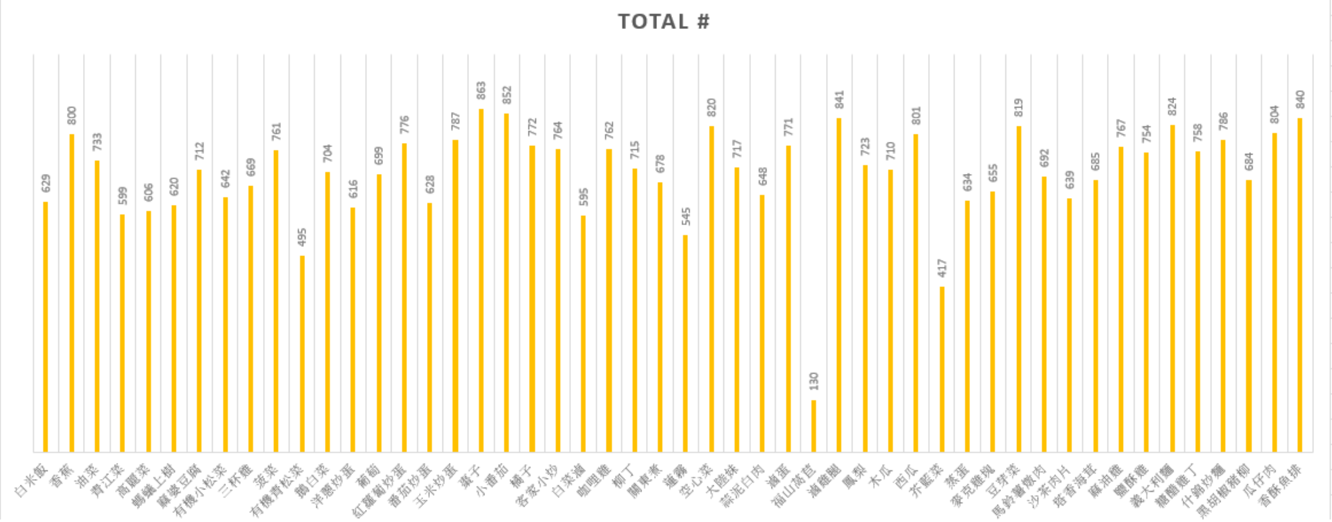
```python
# Unfreeze specific layers for fine-tuning (optional)

def unfreeze_model(model, unfreeze_start_at):
    model.trainable = True

    for layer in model.layers[:unfreeze_start_at]:
        layer.trainable = False
    for layer in model.layers[unfreeze_start_at:]:
        if not isinstance(layer, layers.BatchNormalization):
            layer.trainable = True
```

- 以block或block中block為單位來finetune
- BatchNormalization曾不finetune以免破壞pretrained mode學到的特徵

資料分析/處理



- 針對資料分布不均
  - ex.福山萵苣(class 12)資料少
  - 利用 Class_weight(處理imbalance)
- 針對尺寸不一: 在改為目標大小時，使用Bilinear Interpolation來降低distortion和loss of information
  - 有想要crop，但無從得知如何針對每一張都切到食物，所以沒有使用
  - 有想要pad到與target size的寬高比一致，使不至於有distortion，但多出來的空白處可能也會被學成特徵，
  - 並且覺得有一點的distortion對本來就不會很整齊一致擺放的食物可能就具有一點regularizaion的效果，因此最後選擇interpolation的方法
- 訓練集: 驗證集 = 8 : 2

```
trainBatch = trainDataGenerator.flow_from_directory(
    directory = dirpath+'train/train',
    target_size = imgSize,
    class_mode = 'categorical',
    shuffle = True,
    batch_size = batchSize,
    subset='training',
    seed = SEED_VAL,
    interpolation='bilinear'
)
```

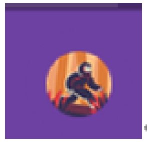
Data Augmentation

```
# train generator
trainDataGenerator = ImageDataGenerator(rescale = 1/255,
                      # augmentation
                      rotation_range = 90,
                      width_shift_range=0.3,
                      height_shift_range=0.3,
                      zoom_range = 0.3,
                      validation_split = 0.2)
valDataGenerator = ImageDataGenerator(rescale=1/255, validation_split=0.2)
```

- 只針對train data進行資料增強
- 隨機轉動正負90度: 認為食物應該那個方向看應該都可以分得出
- 左右，上下平移範圍: 30%
- 放大縮小 30%
- 切分Train : Val = 8 : 2

增加資料前後

- 藍色箭頭(增加資料)
- 發現準度沒有變化
- 但實際看其中加入資料的種類
  - 其recall 變高

Waywuwin: 	Adam_6e-4 X(116)_eff(964) 512 286 128 Flatten (81%)	Adam_9e-4 X(-4)_eff(994) 1024 512 286 Flatten Drop(0.2) (81) ep15

根據confusion Matrix: 分不清，所以加資料

Confusion Matrix中部分較深的種類: (根據下頁左邊Confusion Matrix)

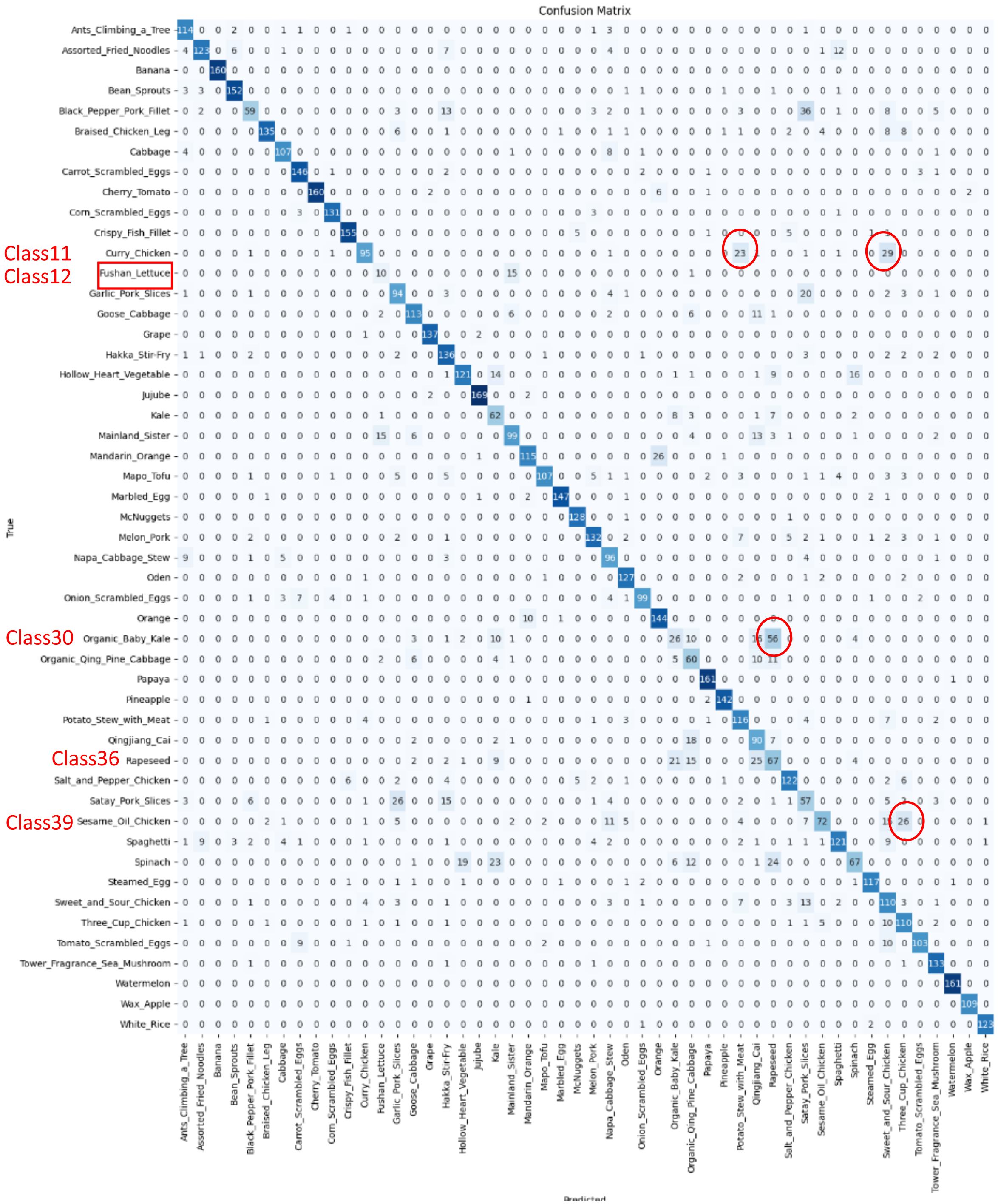
1. 黑胡椒豬柳/沙茶豬肉片
2. 油菜/有機小松菜
3. 柳丁/橘子
4. 麻油雞/三杯雞/咖哩雞

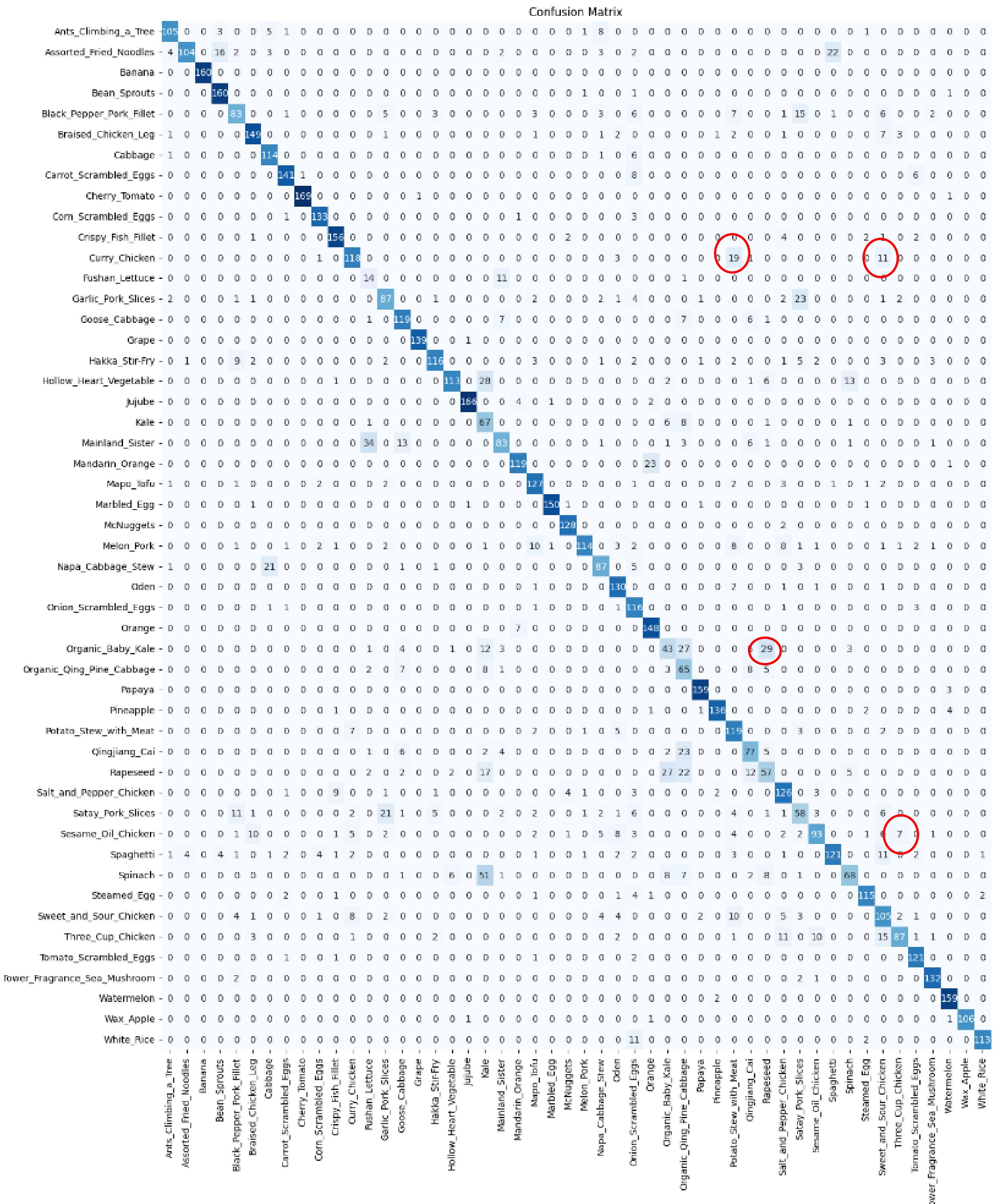
亂碼:為中文字(“複製”)

資料處理方法:

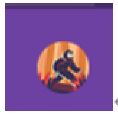
- 針對不易分辨的資料，先肉眼人工的判斷說，哪些我一看就知道是什麼
  - 比如說像三杯雞，常能輕易讓我便是是因為配9層塔
  - 麻油雞就會配的可能油油的湯，咖啡色的，配上米血
  - 咖喱雞的話，可能就會配上比較黃的湯，然後還有蘿蔔

- 那我就會主動把這些能輕易分辨的圖片們去複製貼上多份，使他們可以更多地被訓練到


Confusion Matrix
Class11
Class12
Class30
Class36
Class39
True
Predicted
Ants_Climbing_a_Tree
Assorted_Fried_Noodles
Banana
Bean_Sprouts
Black_Pepper_Pork_Fillet
Braised_Chicken_Leg
Cabbage
Carrot_Scrambled_Eggs
Cherry_Tomato
Corn_Scrambled_Eggs
Crispy_Fish_Fillet
Curry_Chicken
Fushan_Lettuce
Garlic_Pork_Slices
Goose_Cabbage
Grape
Hakka_Stir-Fry
Hollow_Heart_Vegetable
Jujube
Kale
Mainland_Sister
Mandarin_Orange
Mapo_Tofu
Marbled_Egg
McNuggets
Melon_Pork
Napa_Cabbage_Stew
Oden
Onion_Scrambled_Eggs
Orange
Organic_Baby_Kale
Organic_Qing_Pine_Cabbage
Papaya
Pineapple
Potato_Stew_with_Meat
Qingjiang_Cai
Rapeseed
Salt_and_Pepper_Chicken
Satay_Pork_Slices
Sesame_Oil_Chicken
Spaghetti
Spinach
Steamed_Egg
Sweet_and_Sour_Chicken
Three_Cup_Chicken
Tomato_Scrambled_Eggs
Tower_Fragrance_Sea_Mushroom
Watermelon
Wax_Apple
White_Rice


Confusion Matrix
Ants_Climbing_a_Tree
Assorted_Fried_Noodles
Banana
Bean_Sprouts
Black_Pepper_Pork_Fillet
Braised_Chicken_Leg
Cabbage
Carrot_Scrambled_Eggs
Cherry_Tomato
Corn_Scrambled_Eggs
Crispy_Fish_Fillet
Curry_Chicken
Fushan_Lettuce
Garlic_Pork_Slices
Goose_Cabbage
Grape
Hakka_Stir-Fry
Hollow_Heart_Vegetable
Jujube
Kale
Mainland_Sister
Mandarin_Orange
Mapo_Tofu
Marbled_Egg
McNuggets
Melon_Pork
Napa_Cabbage_Stew
Oden
Onion_Scrambled_Eggs
Orange
Organic_Baby_Kale
Organic_Qing_Pine_Cabbage
Papaya
Pineapple
Potato_Stew_with_Meat
Qingjiang_Cai
Rapeseed
Salt_and_Pepper_Chicken
Satay_Pork_Slices
Sesame_Oil_Chicken
Spaghetti
Spinach
Steamed_Egg
Sweet_and_Sour_Chicken
Three_Cup_Chicken
Tomato_Scrambled_Eggs
Tower_Fragrance_Sea_Mushroom
Watermelon
Wax_Apple
White_Rice

增加資料 (也增加學習率)

Waywuwin:	Adam_6e-4	Adam_9e-4
	X(116)_eff(964)	X(-4)_eff(994)
	512 286 128	1024 512 286
	Flatten (81%)	Flatten
		Drop(0.2)
		(81) ep15

Classification Report:

		precision	recall	f1-score	support
	0	0.81	0.92	0.86	124
	1	0.89	0.78	0.83	158
	2	1.00	1.00	1.00	160
	3	0.93	0.93	0.93	163
	4	0.76	0.43	0.55	136
	5	0.96	0.80	0.87	169
	6	0.88	0.88	0.88	122
	7	0.87	0.94	0.90	156
	8	1.00	0.94	0.97	171
	9	0.95	0.95	0.95	138
	10	0.95	0.92	0.93	168
咖喱雞	11	0.86	0.62	0.72	153
福山萵苣	12	0.33	0.38	0.36	26
	13	0.63	0.72	0.67	130
有機小松菜	30	0.39	0.20	0.27	129
油菜	36	0.36	0.46	0.40	146
麻油雞	39	0.83	0.47	0.60	154
	accuracy			0.81	7002
	macro avg	0.80	0.79	0.79	7002
	weighted avg	0.82	0.81	0.81	7002

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.85	0.88	124
1	0.95	0.66	0.78	158
2	1.00	1.00	1.00	160
3	0.87	0.98	0.92	163
4	0.72	0.61	0.66	136
5	0.88	0.88	0.88	169
6	0.79	0.93	0.85	122
7	0.93	0.90	0.92	156
8	0.99	0.99	0.99	171
9	0.93	0.96	0.95	138
10	0.91	0.93	0.92	168
11	0.83	0.77	0.80	153
12	0.25	0.54	0.34	26
13	0.70	0.67	0.68	130
14	0.77	0.84	0.81	141
15	0.99	0.99	0.99	140
30	0.47	0.33	0.39	129
36	0.50	0.39	0.44	146
39	0.82	0.60	0.69	154
accuracy			0.81	7002
macro avg	0.80	0.80	0.80	7002
weighted avg	0.83	0.81	0.81	7002

改Optimizer

- 所有結參數相同，只改動優化器
- RMSprop 變 Nadam
- 使用Nadam的top one accuracy多了1%。
- Training time: Nadam > RMSprop

RMS_e-3 X(116)_eff(964) 1024 512 256 128 Flatten Dr0.2 (84%) ep45	Nadam_e- 3_Classweight _ep45_ 1024-512-256- 128_Flatten_ Dr0.2 (85%)

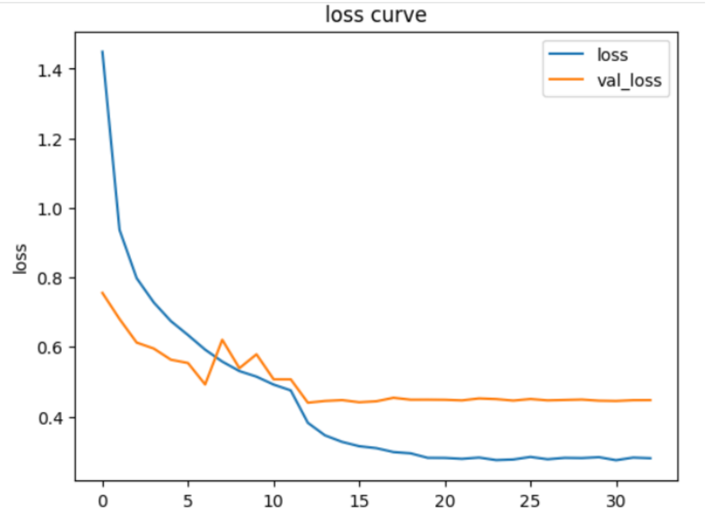
Run

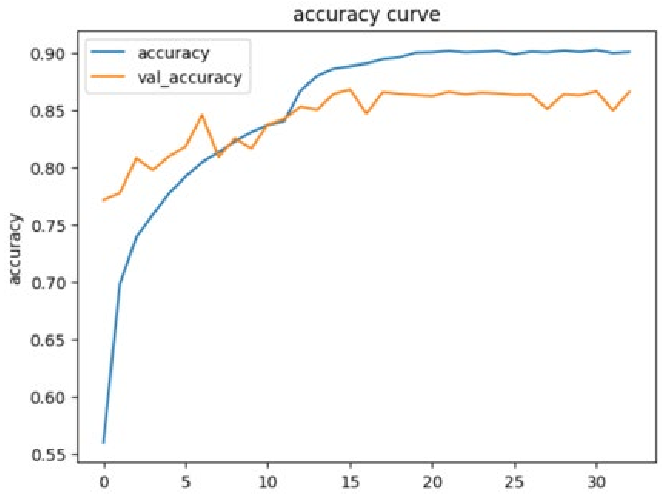
28447.1s - GPU P100

Run

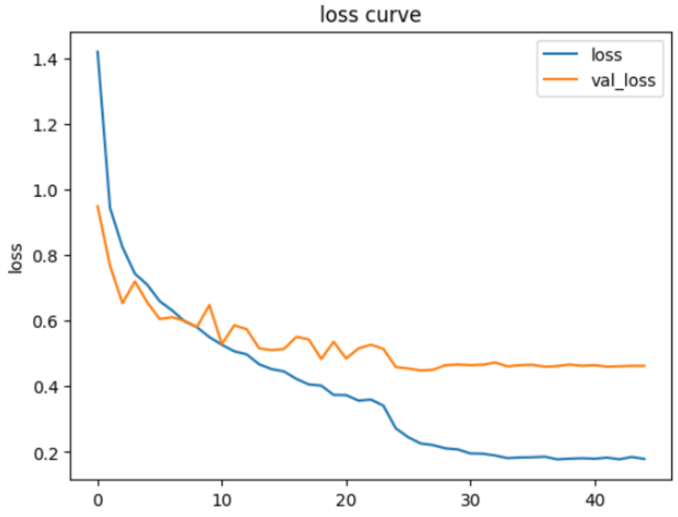
21898.7s - GPU P100

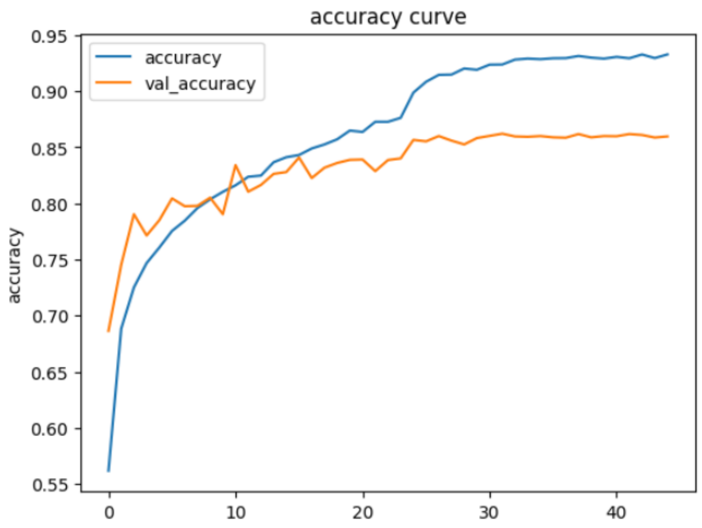
RMSprop





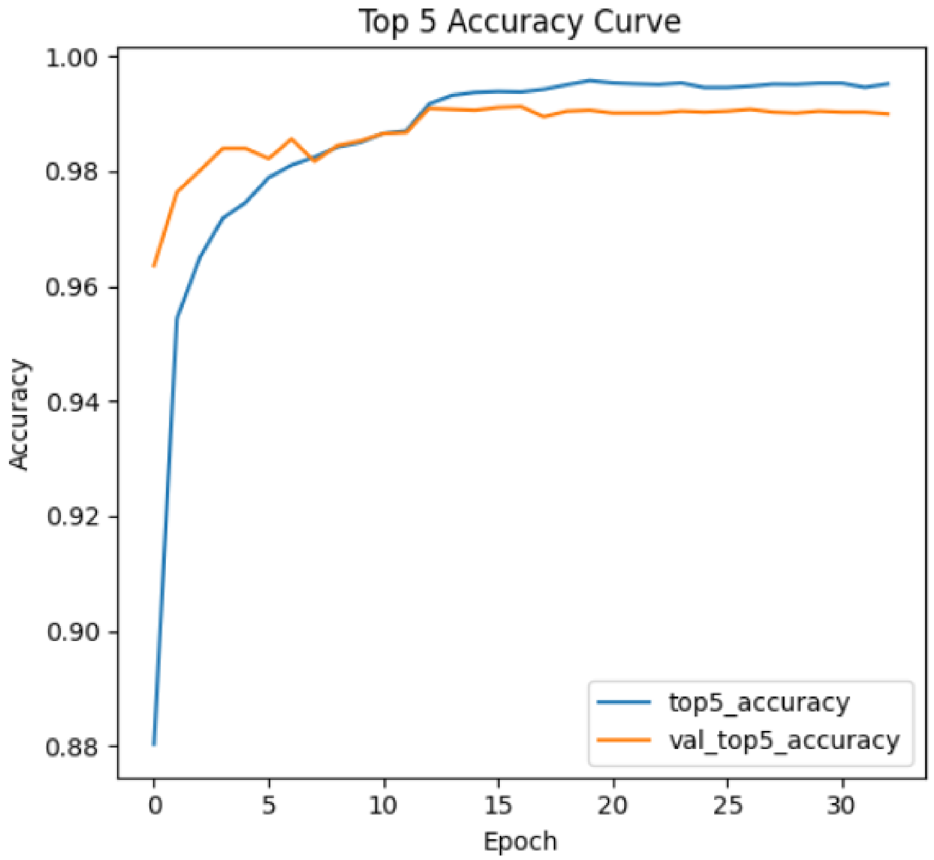
Nadam



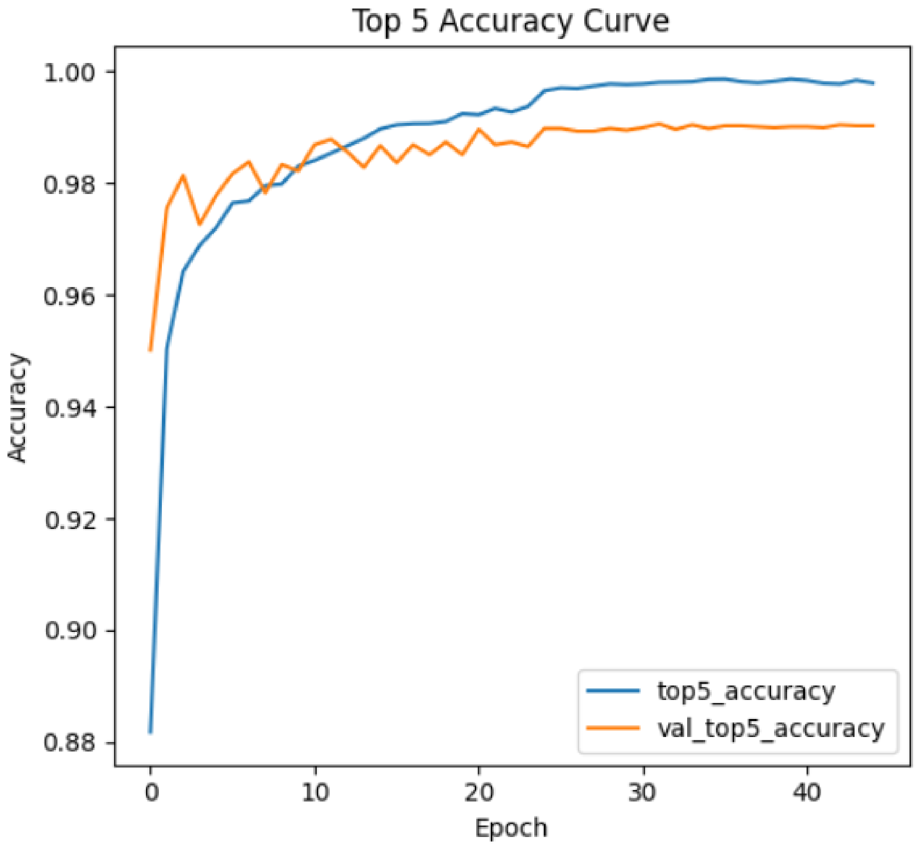


- Nadam收斂較慢，但train_accuracy收斂到近95%較，RMSprop的90%高。
- Val_accuracy兩者差異不大

RMSprop



Nadam



兩者的Top 5 accuracy(Train and Val)收的位置基本一樣，只是Nadam收斂比較慢。

RMSprop

Classification Report:

		precision	recall	f1-score	support
	0	0.90	0.92	0.91	124
	1	0.88	0.87	0.88	158
	2	0.99	1.00	0.99	160
	3	0.96	0.94	0.95	163
	4	0.67	0.76	0.71	136
	5	0.94	0.89	0.92	169
	6	0.93	0.93	0.93	122
	7	0.95	0.92	0.93	156
	8	1.00	0.98	0.99	171
	9	0.97	0.95	0.96	138
	10	0.89	0.95	0.92	168
咖喱雞	11	0.88	0.77	0.82	153
福山萵苣	12	0.31	0.58	0.41	26
	13	0.76	0.67	0.71	130
有機小松菜	30	0.41	0.43	0.42	129
油菜	36	0.49	0.43	0.46	146
麻油雞	39	0.80	0.66	0.73	154
accuracy				0.84	7002
macro avg		0.83	0.84	0.83	7002
weighted avg		0.85	0.84	0.85	7002

Nadam

F1 Score大致有提升

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.89	0.91	124
1	0.90	0.85	0.87	158
2	0.99	1.00	0.99	160
3	0.96	0.93	0.95	163
4	0.66	0.80	0.72	136
5	0.91	0.89	0.90	169
6	0.94	0.93	0.94	122
7	0.95	0.93	0.94	156
8	1.00	0.97	0.99	171
9	0.97	0.98	0.97	138
10	0.95	0.93	0.94	168
11	0.92	0.71	0.80	153
12	0.31	0.62	0.42	26
13	0.71	0.73	0.72	130
30	0.42	0.49	0.45	129
36	0.47	0.40	0.43	146
39	0.77	0.66	0.71	154
accuracy			0.85	7002
macro avg	0.84	0.84	0.84	7002
weighted avg	0.85	0.85	0.85	7002

class_weight拿掉後呢?

- 準度竟然上升?

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.90	0.89	124
1	0.89	0.89	0.89	158
2	0.98	1.00	0.99	160
3	0.97	0.93	0.95	163
4	0.68	0.73	0.70	136
5	0.90	0.90	0.90	169
6	0.93	0.91	0.92	122
7	0.96	0.91	0.93	156
8	0.99	0.99	0.99	171
9	0.95	0.94	0.95	138
10	0.90	0.93	0.92	168
11	0.88	0.74	0.80	153
12	0.48	0.50	0.49	26
13	0.68	0.72	0.70	130
19	0.59	0.56	0.58	84
31	0.50	0.64	0.56	99
accuracy			0.84	7002
macro avg	0.83	0.83	0.83	7002
weighted avg	0.84	0.84	0.84	7002

Without Classweight

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.90	0.89	124
1	0.89	0.89	0.89	158
2	0.99	1.00	1.00	160
3	0.96	0.96	0.96	163
4	0.70	0.68	0.69	136
5	0.94	0.91	0.93	169
6	0.91	0.93	0.92	122
7	0.95	0.93	0.94	156
8	1.00	0.99	1.00	171
9	0.97	0.96	0.96	138
10	0.90	0.92	0.91	168
11	0.91	0.81	0.86	153
12	0.39	0.62	0.48	26
13	0.70	0.77	0.73	130
19	0.67	0.67	0.67	84
31	0.57	0.60	0.58	99
accuracy			0.85	7002
macro avg	0.84	0.84	0.84	7002
weighted avg	0.86	0.85	0.85	7002

RMS_e-3 X(116)_eff(964) 1024 512 286 Flatten Dr0.2 (84%) ep50
(No classweight) RMS_e-3 X(116)_eff(964) 1024 512 256 Flatten Dr0.2 (85%) ep45

- 可能選擇的模型本身處理資料不均的問題能力就不錯
- 從增加的macro/ weighted avg之間可以觀察到一些class_weight的影響

取Top1 Accuracy最高(85%)的Model探討

Nadam_e-3_Classweight_ep45_1024-512-256-128_Flatten_Dr0.2 (85%)

- **資料量最少的福山萵苣發生了什麼變化? 仍然不容易辨識**
  - **precision: 31%**

		precision	recall	f1-score	support
福山萵苣	12	0.31	0.58	0.41	26

	precision	recall	f1-score	support
12	0.31	0.62	0.42	26

- 油菜 / 有機小松菜很難分的問題有解決嗎？
  - 準確度高，有變好，但是在有機小松菜/油菜還是常分不出來

Adam_9e-4 X(-4)_eff(994) 1024 512 286 Flatten Drop(0.2) (84%) ep35

		precision	recall	f1-score	support
有機小松菜	30	0.47	0.33	0.39	129
	31	0.40	0.66	0.50	99
	32	0.96	0.98	0.97	162
	33	0.96	0.94	0.95	145
	34	0.65	0.86	0.74	139
	35	0.65	0.64	0.64	120
油菜	36	0.50	0.39	0.44	146
	37	0.74	0.83	0.79	151
	38	0.50	0.45	0.48	128
麻油雞	39	0.82	0.60	0.69	154

	precision	recall	f1-score	support
30	0.42	0.49	0.45	129
31	0.51	0.62	0.56	99
32	0.99	0.98	0.98	162
33	0.97	0.95	0.96	145
34	0.65	0.90	0.75	139
35	0.67	0.70	0.68	120
36	0.47	0.40	0.43	146
37	0.87	0.82	0.85	151
38	0.55	0.55	0.55	128
39	0.77	0.66	0.71	154

Nadam_e-3_Classweight_ep45_1024-512-256-128_Flatten_Dr0.2 (85%)

Youtube影片 連結:

- https://youtu.be/wdH4tk-wVxQ