

HAM10000

皮膚鏡影像分析

408170508 資數三 賴冠羽

目錄

壹、疾病內容說明	2
貳、資料集描述	4
參、訓練過程	5
一、 資料前處理	5
二、 資料增強(data augmentation).....	6
肆、模型選擇	8
伍、實驗結果	11
一、 confusion matrix.....	11
二、 Grad-CAM.....	12
陸、討論	14

壹、 疾病內容說明

在 HAM10000 資料中共有七種病徵如下表：

類別	名稱	英文全名	中文
0	nv	melanocytic nevi	黑素細胞痣
1	akiec	Actinic keratoses and intraepithelial carcinoma	鱗狀細胞癌
2	mel	melanoma	黑色素瘤
3	df	dermatofibroma	皮膚纖維瘤
4	bcc	basal cell carcinoma	基底細胞癌
5	vasc	vascular lesions [1]	血管病變
6	bkl	benign keratosis-like lesions [2]	良性角化病變

[1]：Kaggle 網站上對 vasc 類完整說明為 vascular lesions

(angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc)

(血管瘤、血管角質瘤、化膿性肉芽腫及出血)

[2]：Kaggle 網站上對 bkl 類完整說明為 benign keratosis-like lesions

(solar lentigines / seborrheic keratoses and lichen-planus like keratoses, bkl)

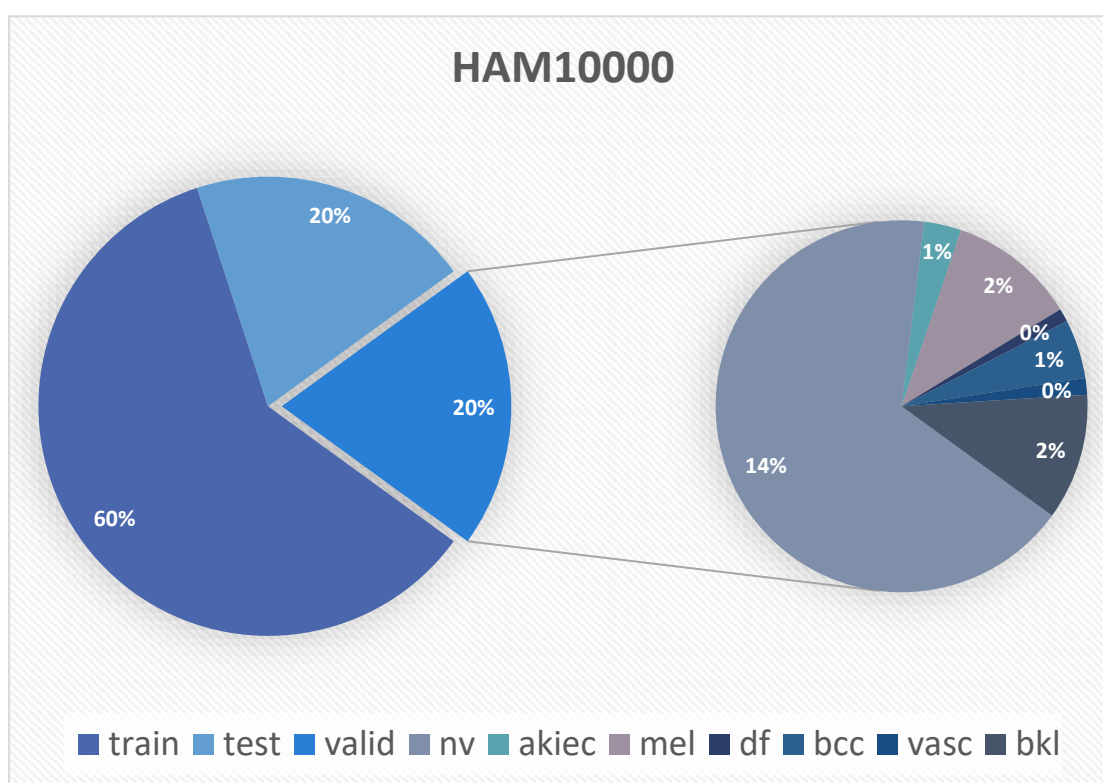
(曬斑、脂漏性角化症(老人斑)、扁平苔蘚)

七類病徵分析：

類別	名稱	外觀	顏色	圖片
0	nv	扁平、突起、疣狀、顆粒狀	棕色、黑色、藍色	
1	akiec	頂端為鱗狀的硬塊，乾燥粗糙、黏附脫屑	皮膚色、黃棕色，帶有一些紅色色調	
2	mel	不對稱、邊界不均衡、破爛或缺口	黑色或棕褐色	
3	df	質地硬、圓形腫物	棕紅、黃褐、黑褐	
4	bcc	像蟲咬般，腫瘤有時出現潰瘍出血，周圍可見微血管擴張	紅色、肉色、黑色	
5	vasc	突出斑點、丘疹粗糙脫屑、圓形或略扁平的綠豆至櫻桃大小乳頭狀肉芽腫	紅色、暗紅至紫色，栓塞則轉成深紫至黑色	
6	bkl	約米粒大小、隆起、粗糙斑駁的外觀、苔蘚樣的乾燥波紋	黃棕到黑色、淡褐色到深黑色	

貳、 資料集描述

HAM10000 資料集共有 10015 張影像，其中 train, valid, test 資料集的比例分為 6:2:2，而每個資料集中有 {nv, akiec, mel, df, bcc, vasc, bkl} 共七類病徵。



各資料集詳細數據如下表：

類別	名稱	train	valid	test	合計
0	nv	4,023	1,341	1,341	6,705
1	akiec	196	65	66	327
2	mel	668	222	223	1,113
3	df	69	23	23	115
4	bcc	308	103	103	514
5	vasc	85	29	28	142
6	bkl	660	220	219	1,099
合計	-	6,009	2,003	2,003	10,015

參、 訓練過程

一、 資料前處理

因病徵共有七類，所以在資料前處理中 `class_mode` 設定為 `categorical`，其中 `valid` 和 `test` 上資料不打亂，因 `train` 資料較多，所以 `shuffle=True`，並且在 `img_shape` 上則是設定(224, 224)。

```
1 train_generator = train_datagen.flow_from_dataframe(  
2       
3       
4     x_col="img_path",  
5     y_col="label",  
6     target_size=img_shape,  
7     batch_size=batch_size,  
8     class_mode='categorical',  
9     shuffle=True)  
10 valid_generator = valid_datagen.flow_from_dataframe(  
11       
12       
13     x_col="img_path",  
14     y_col="label",  
15     target_size=img_shape,  
16     batch_size=batch_size,  
17     class_mode='categorical',  
18     shuffle=False)  
19 test_generator = test_datagen.flow_from_dataframe(  
20       
21       
22     x_col="img_path",  
23     y_col="label",  
24     target_size=img_shape,  
25     batch_size=batch_size,  
26     class_mode=None,  
27     shuffle=False)
```

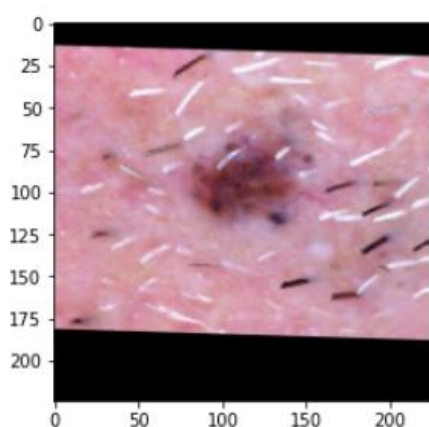
Found 6009 validated image filenames belonging to 7 classes.
Found 2003 validated image filenames belonging to 7 classes.
Found 2003 validated image filenames.

二、資料增強(data augmentation)

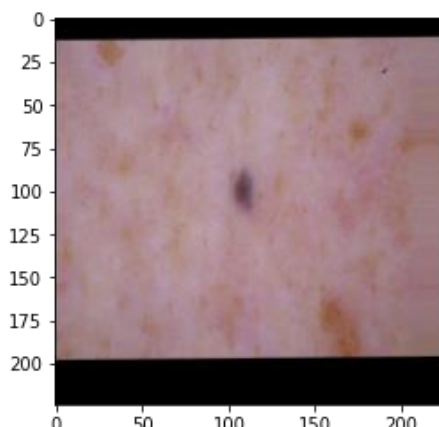
資料增強修正過程：

順序	程式碼	準確率
1	rotation_range=5, horizontal_flip=True, vertical_flip=False, width_shift_range=0.1, height_shift_range=0.1, preprocessing_function=preprocess_input	0.825
2	shear_range = 0.1(新增) zoom_range = 0.1(新增)	0.831+OOM
3	vertical_flip= True(修改)	0.725+OOM

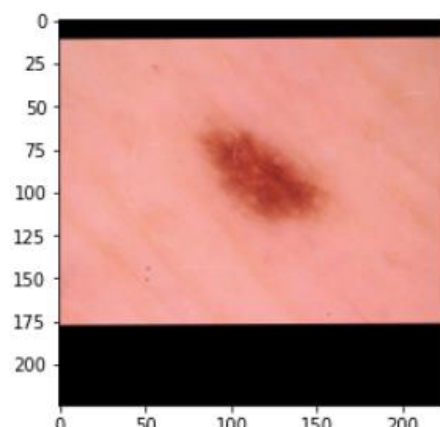
修正後圖片顯示：



1



2



3

針對皮膚鏡影像預測，若要使預測準確率越高，就需要讓病變處置於影像中間。

首先將影像隨機旋轉 5 度以內(rotation_range=5)，並設定影像水平隨機翻轉(horizontal_flip=True)，接著用水平、上下小幅度平移(width_shift_range=0.1、height_shift_range=0.1)，試著看能不能將病變處置於中央。

即使模型最後準確率僅小幅提升，但因為模型 OOM，代表準確率其實是有機會上升，於是最後決定稍微將影像放大(zoom_range=0.1)、比例平移(shear_range=0.1)，以利病變處更能被清楚預測。

```
1 # 注意皮膚要在正中間，位置要調整
2 train_datagen = ImageDataGenerator(
3     rotation_range=5,
4     horizontal_flip=True,
5     vertical_flip=False,
6     width_shift_range=0.1,
7     height_shift_range=0.1,
8     shear_range = 0.1,
9     zoom_range = 0.1,
10    preprocessing_function=preprocess_input
11
12 )
13 valid_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
14 test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
```

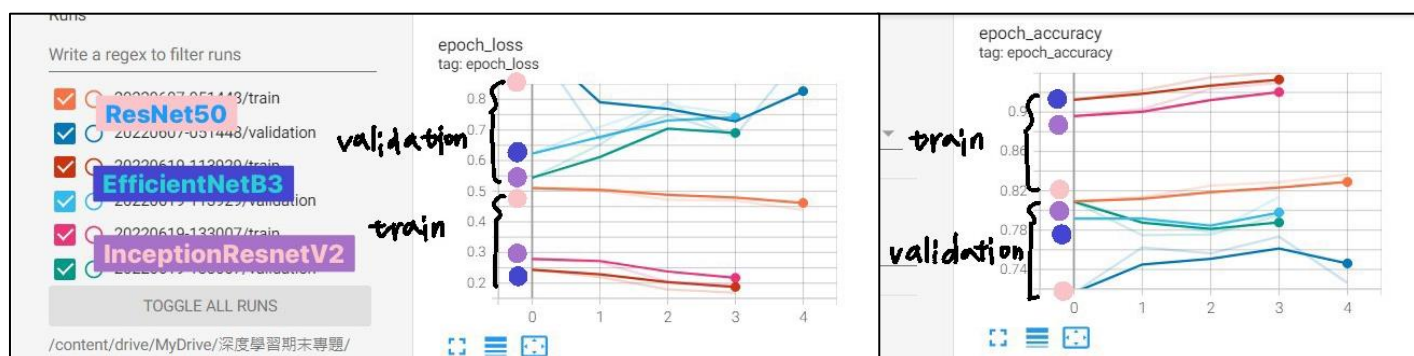

肆、模型選擇

首先根據 <https://keras.io/api/applications/>，

選擇官方準確率較高的模型，並使用相同的腳本進行預測：

{ResNet50, EfficientNetB3, InceptionResNetV2}							
Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0

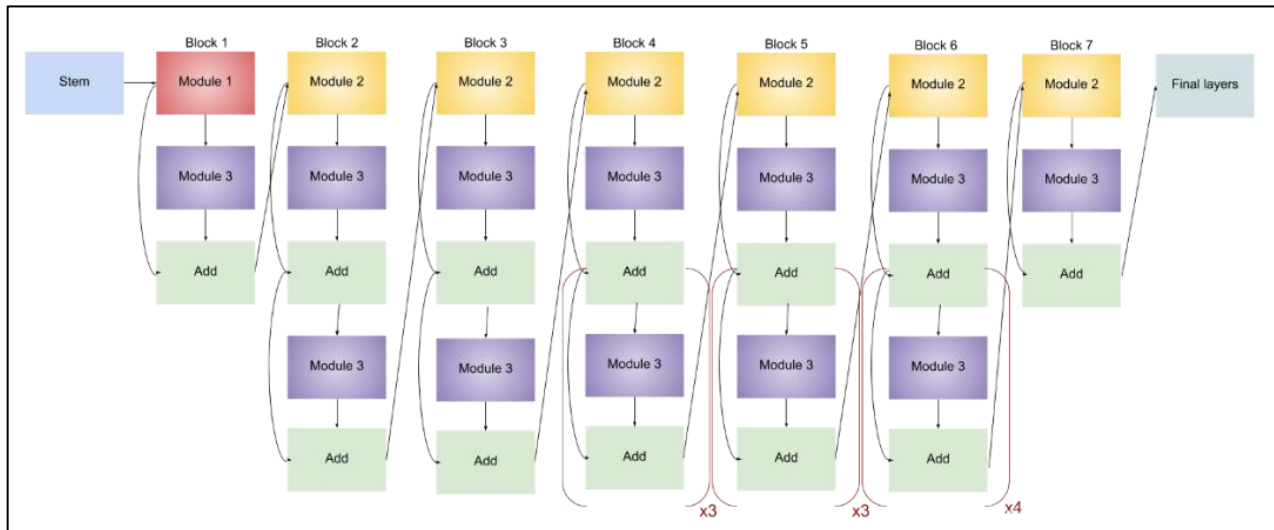
執行後 Tensorboard 成果如下圖：



由上圖透過 train_loss(橘線)可知 ResNet50 下降幅度相較平緩，在訓練集沒有學習到什麼成果。並透過 val_acc(淺藍色、綠線)，比較 EfficientNetB3, InceptionResNetV2，最後得到 EfficientNetB3 成果較好。而最後三個模型準確率分別為{0.767, 0.825, 0.796}。

故最終使用 **EfficientNetB3** 以及搭配上上述所修正的資料前處理、資料增強，進行最後的模型預測。

EfficientNetB3 結構圖



(x3, x4 代表重複三、四次)

編譯模型上，使用 GlobalAveragePooling2D 進行平均池化，降低參數：

```
1 pre_model = EfficientNetB3(weights='imagenet', input_shape=(img_shape[0], img_shape[1], 3), include_top=False)
2
3 x = layers.GlobalAveragePooling2D()(pre_model.output)
4
5 outputs = layers.Dense(num_classes, activation="softmax")(x)
```

Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb3_notop.h5

43941888/43941136 [=====] - 0s 0us/step

43950080/43941136 [=====] - 0s 0us/step

模型參數：

```
=====
Total params: 10,794,294
Trainable params: 10,706,991
Non-trainable params: 87,303
=====
```

針對模型一共訓練 2 次。

第一次訓練如下圖：(lr=1e-3、batch_size=64)

```
1 pre_model.trainable = False

1 history = model.fit_generator(train_generator,
2                               steps_per_epoch=train_steps,
3                               epochs=num_epochs,
4                               validation_data=valid_generator,
5                               validation_steps=valid_steps,
6                               callbacks=callbacks_list)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version.

Epoch 1/20
94/94 [=====] - 2434s 26s/step - loss: 0.7528 - accuracy: 0.7379 - val_loss: 0.9167 - val_accuracy: 0.7544
Epoch 2/20
94/94 [=====] - 167s 2s/step - loss: 0.5223 - accuracy: 0.8065 - val_loss: 0.7370 - val_accuracy: 0.7958
Epoch 3/20
94/94 [=====] - 166s 2s/step - loss: 0.4191 - accuracy: 0.8486 - val_loss: 0.6074 - val_accuracy: 0.8223
Epoch 4/20
94/94 [=====] - 164s 2s/step - loss: 0.3608 - accuracy: 0.8700 - val_loss: 0.5411 - val_accuracy: 0.8158
Epoch 5/20
94/94 [=====] - 164s 2s/step - loss: 0.2920 - accuracy: 0.8963 - val_loss: 0.5976 - val_accuracy: 0.8218
Epoch 6/20
94/94 [=====] - 162s 2s/step - loss: 0.2550 - accuracy: 0.9106 - val_loss: 0.9512 - val_accuracy: 0.7838
Epoch 7/20
94/94 [=====] - 165s 2s/step - loss: 0.2194 - accuracy: 0.9190 - val_loss: 0.5744 - val_accuracy: 0.8308
Epoch 7: early stopping
```

第一次訓練後 val_accuracy 已上升至 0.8308。

第二次訓練取上次訓練末五層繼續訓練，使準確率提升，並將 lr 調降至 1e-5。

因為在資料集裡第 0 類 nv 的影像較其他類多很多，因此在第二次訓練上，將 batch_size 改為 32，讓模型在訓練過程中比較不容易被混淆。

```
1 pre_model.trainable = True
2 for each_layer in pre_model.layers[:-5]: #倒數五層
3     each_layer.trainable = False
4
5 lr = 1e-5
6 batch_size = 32
7 num_epochs = 10

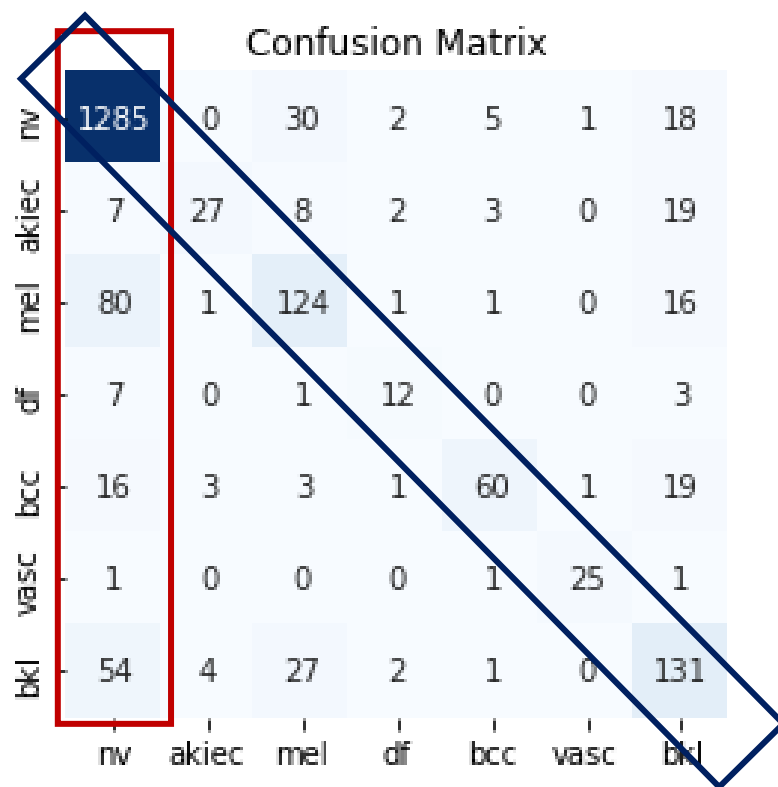
1 history = model.fit_generator(train_generator,
2                               steps_per_epoch=train_steps,
3                               epochs=num_epochs,
4                               validation_data=valid_generator,
5                               validation_steps=valid_steps,
6                               callbacks=callbacks_list)
```

執行結果為 OOM，最後模型的 accuracy_score 為 0.831：

```
Node: 'Adam/gradients/sub_72'
failed to allocate memory
[[[{'node Adam/gradients/sub_72'}]]]
Hint: If you want to see a list of allocated tensors when OOM happens, add report_tensor_allocations_upon_oom to RunOptions for current allocation info. This isn't available when running in Eager mode.
[Op:__inference_train_function_28355]
```

伍、 實驗結果

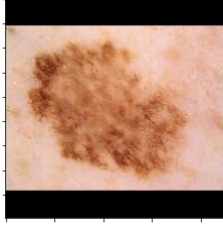
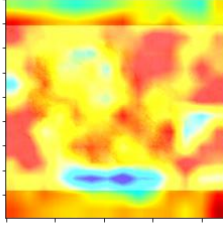
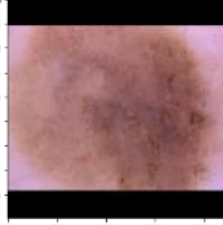
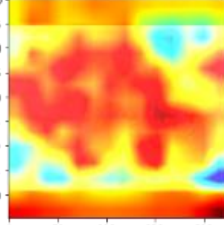
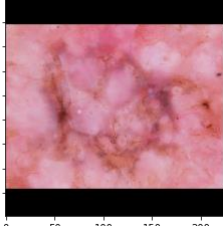
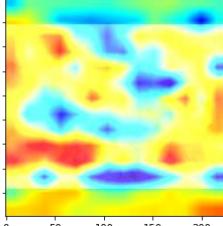
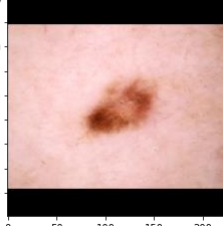
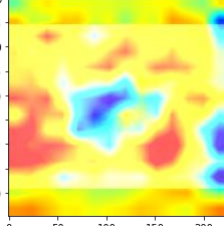
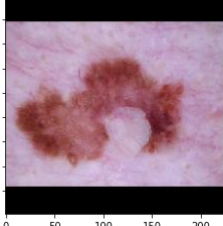
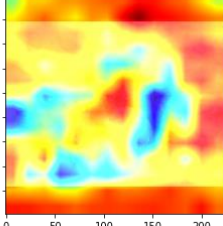
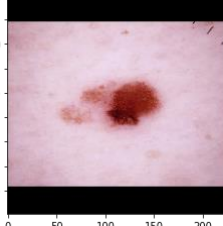
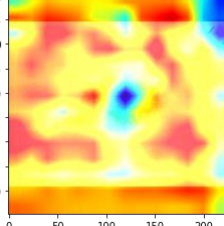
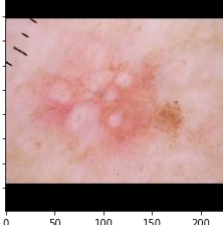
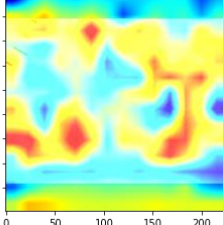
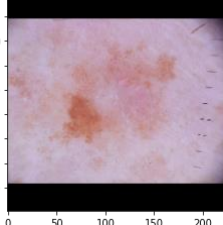
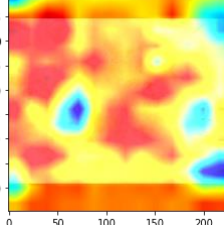
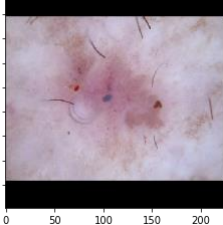
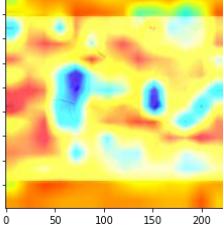
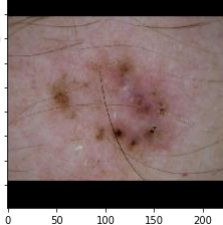
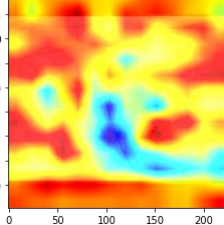
一、 confusion martix 分析

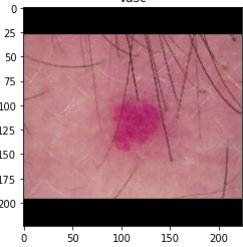
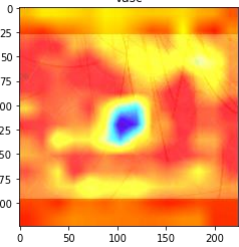
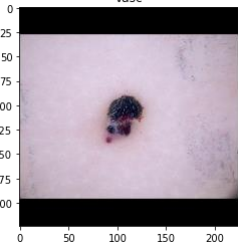
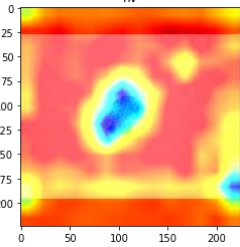
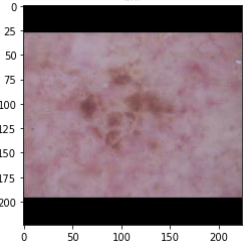
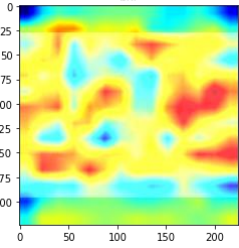
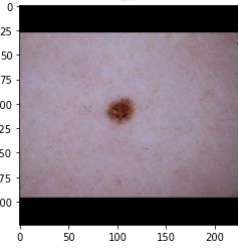
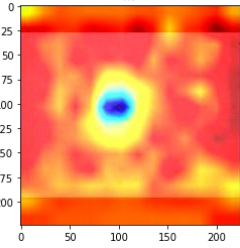


由 confusion martix 可發現，因為 nv 病例過多，所以有著很深的藍色，雖然其他類資料筆數不多，但對角線顏色都有比較深，代表著預測大致上是正確的。(如藍框)

而透過 confusion martix 也可以發現，多數病例在模型預測上都會被預測錯誤成 nv。(如紅框)

二、 Grad-CAM

類別	原始圖片	預測正確	原始圖片	預測錯誤
0 nv				
1 akiec				
2 mel				
3 df				
4 bcc				

類別	原圖	預測正確	原圖	預測錯誤
5 vasc				
6 bkl				

透過 confusion martix，我們了解多數病例在模型預測上都會被預測錯誤成 nv，而透過分析七類病徵，我們可以瞭解 nv 病徵為扁平、突起、疣狀、顆粒狀，顏色為棕色、黑色、藍色，通俗上常被稱作「痣」。

也因此 in Grad-CAM 上我們可以發現預測錯誤的原始圖片，大多都因為顏色、病徵扁平被誤判成 nv。

陸、 討論

增加皮膚鏡影像

透過最後的實驗結果，我們可以了解數據太過參差不齊，nv 提供太多，而 df、vasc 提供過少，數量相差高達 60 倍。所以如果要再繼續提高辨識皮膚鏡影像準確率，我認為可以跟醫院進行合作，讓醫院端提供皮膚鏡影像，增加各病徵數據量。

模型要求輸入的大小

再來是在模型訓練的過程中，因為每個模型都有其要求要輸入的大小，例如：Resnet50 需要 `img_shape=224`、EfficientNetB3 需要 `img_shape=300`、InceptionResNetV2 需要 `img_shape=299`，但後期嘗試的時候都因為 OOM 而不能繼續嘗試，實屬可惜。

了解病徵、了解為何預測錯誤

藉由這次研究皮膚鏡影像，我認為去了解每個病徵是很重要的一個動作，這樣在測試時也可以得知相似的點在哪裡，進而去加強資料增強部分，讓模型預測時能夠避免再次混淆。