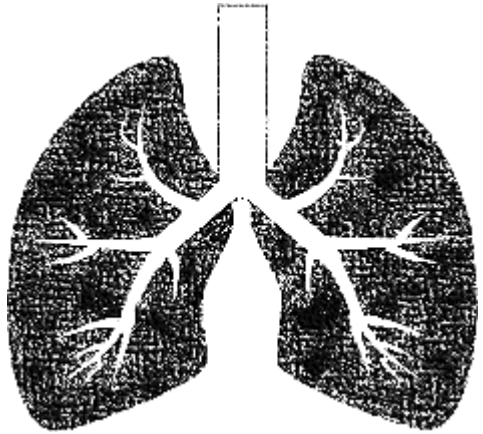
智慧化企業整合
Final Project

利用深度學習網路進行肺音異常分類



第6組
109034554 蔡丞洲
109034518 蕭詠仁
109034803 危佳容
109034515 江毓翔

Outline

背景介紹

方法介紹

個案研究

結論

Outline

背景介紹

方法介紹

個案研究

結論

背景說明

- 醫生可透過聽診器聽取肺部聲音(呼吸音)，並進行肺部疾病和異常診斷
- 臨床上一種簡單友好、非侵入式且對患者無害的方法，因此被廣泛使用
- 但因此種方法評估呼吸音，均仰賴醫生專業及經驗，較主觀，且無法進行定性評估。
- 而肺音信號中蘊藏著豐富的肺部病理及生理上的重要資訊

本研究將究利用深度學習方法進肺音分類
以客觀的方式判斷呼吸音異常與否，研究結果可作為臨床診斷的輔助

5W1H

Why	**診器聽取肺部聲音(呼吸音) 評估呼吸音，較主觀，無法進行定性評估**
What	**透過呼吸音辨識，進行肺部疾病診斷**
Where	**醫院或任何需要進行肺部疾病診斷的地點**
When	**需要判斷呼吸音是否異常時**
Who	**醫生或個人**
How	**利用深度學習網路來進行呼吸音辨識**

Outline

背景介紹

方法介紹

個案研究

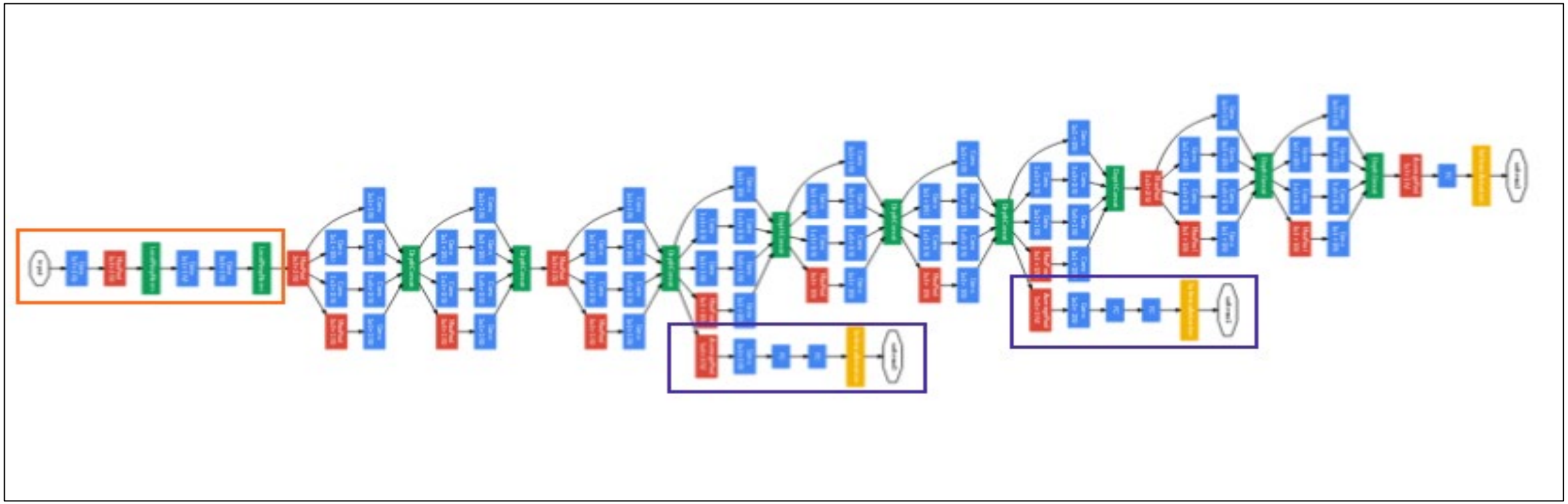
結論

Inception v3 (1/2)

- Inception
  - CNN 分類器發展史上一個重要的里程碑
  - Inception 出現之前，大部分流行 CNN 僅僅是把卷積層堆疊得越來越多，使網絡越來越深，以此希望能夠得到更好的性能
  - GoogLeNet第一次提出 Inception 結構
    - ✓ 目的是設計一種具有優良局部拓撲結構的網絡
    - ✓ 對輸入圖像並行地執行多個卷積運算或池化操作
    - ✓ 將所有輸出結果拼接為一個非常深的特徵圖

Inception v3 (2/2)

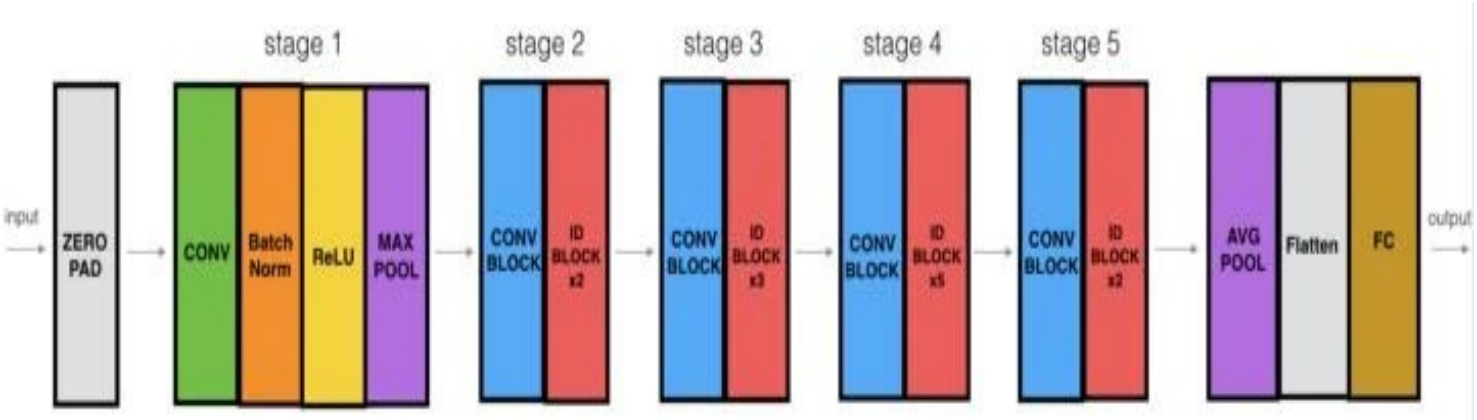
- 最重要的改進是卷積分解（Factorization），將7x7卷積分解成兩個一維的卷積串聯（1x7和7x1），3x3卷積分解爲兩個一維的卷積串聯（1x3和3x1），樣既可以加速計算，又可使網絡深度進一步增加
- 為每增加一層都要進行ReLU(增加了網絡的非線性)。
- 網絡輸入從224x224變爲299x299。
- 其增加了輔助分類器(BatchNorm)、標籤平滑和RMSProp優化器



ResNet-50

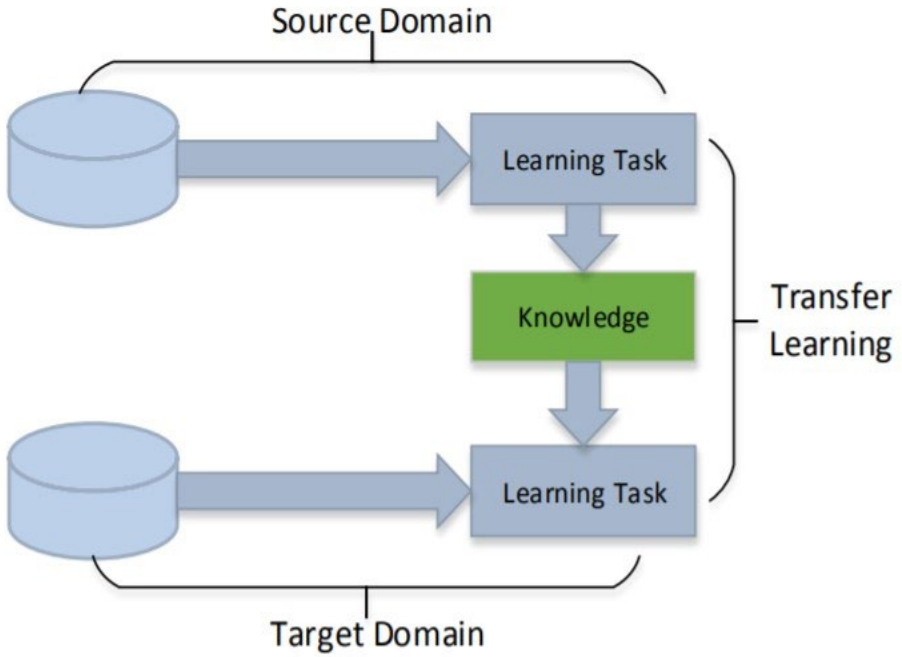
- ResNet在2015年被提出，在ImageNet比賽classification任務上獲得第一名
- 因爲它“簡單與實用”
- ResNet-50
  - 參考VGG19網路，在其基礎上進行修改
  - 通過短路機制加入了殘差單元
  - 用global average pool層替換了全連線層
  - 設計原則：當feature map大小降低一半時 feature map的數量增倍，保持網路層複雜度

layer name	output size	50-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2 [1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[1×1, 128 3×3, 128 1×1, 512] ×4
conv4_x	14×14	[1×1, 256 3×3, 256 1×1, 1024] ×6
conv5_x	7×7	[1×1, 512 3×3, 512 1×1, 2048] ×3



遷移學習(Transfer learning) (1/2)

- 指把訓練好的模型參數遷移到新的模型來幫助新模型訓練
- 從而加快並優化模型的學習效率
- 就是學習Source Domain並應用在Target Domain上
- 來使訓練更加快速



Transfer Learning - Overview

		Source Data (not directly related to the task)	
		labelled	unlabeled
Target Data	labelled	Fine-tuning Multitask Learning	Self-taught learning Rajat Raina , Alexis Battle , Honglak Lee , Benjamin Packer , Andrew Y. Ng, Self-taught learning: transfer learning from unlabeled data, ICML, 2007
	unlabeled	Domain-adversarial training Zero-shot learning	Self-taught Clustering Wenyuan Dai, Qiang Yang, Gui-Rong Xue, Yong Yu, "Self-taught clustering", ICML 2008

Different from semi-supervised learning

遷移學習(Transfer learning) (2/2)

Fine Tuning

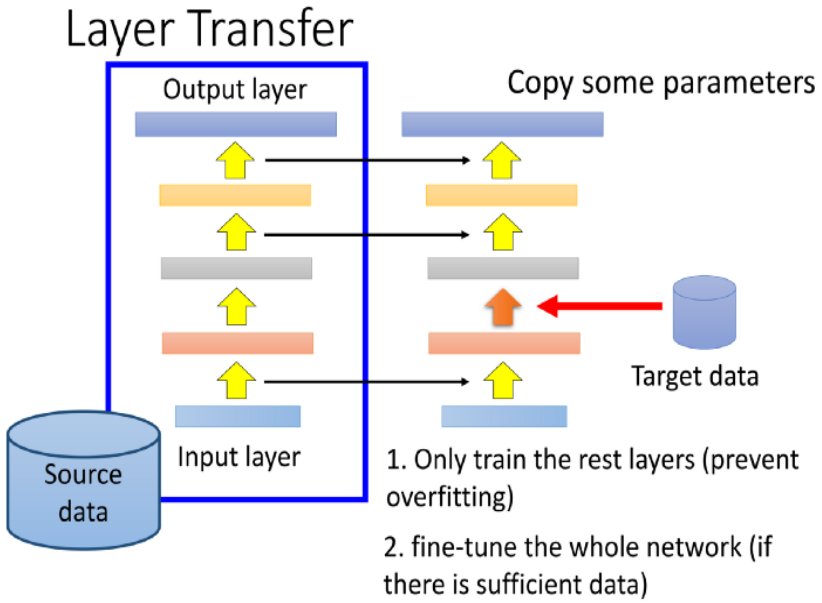
- 目前深度學習(Deep Learning)常見的使用方法
- 為使用已經訓練完成的資料集(Source data)作為目標資料(Target data)的pre-trained model，並且對參數做一些微調
- 但如果目標資料過少可能會導致overfitting的問題

Layer Transfer

- 是透過使用Source data訓練好的模型
- 萃取其中幾層的參數
- 其餘沒有萃取的參數再透過訓練Target data來取得

語音辨識任務

- transfer模型最後幾層的參數
- 原因是人類口腔結構、肺部組織會存在差異
- 後幾層才是去聽人類說了甚麼，且與發音者無關



Outline

PLACED LOGO
YOUR COMPANY NAME HERE

背景介紹

方法介紹

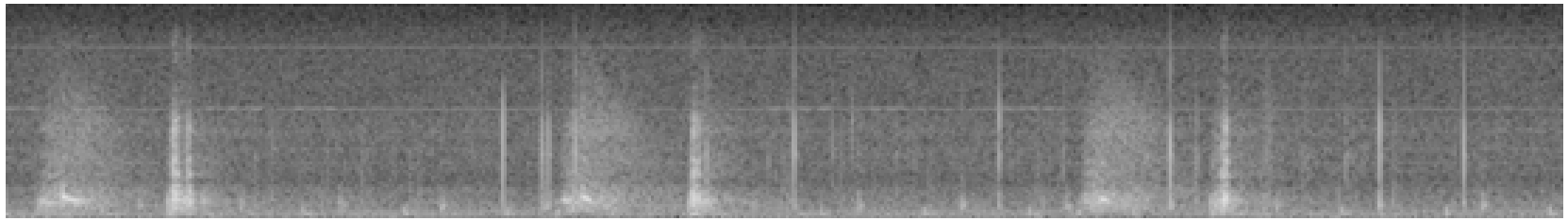
個案研究

結論

資料介紹

本次專題所使用的dataset是現有的資料
(透過短時距傅立葉轉換將收集到的音頻檔轉換為頻譜圖)

情緒	張數
正常(無CAS)	3221
異常(有CAS)	7521
合計	10742



可視化頻譜圖

資料前處理 (1/3)

Step1

- 圖片檔案轉換成jpg以方便訓練及測試
- 圖片檔案轉為nparray的形式存成npy檔
- 所有label的.txt檔整合存成.json檔以方便標籤

```python
import os
path=input('請輸入文件路徑：')+'\\'
#獲取該目錄下所有文件，存入列表中
f=os.listdir(path)
n=0
for i in f:
    #設置舊文件名（就是路徑+文件名）
    oldname=path+f[n]
    #設置新文件名
    newname=path+"train_"+str(n+1)+'.JPG'
    #用os模塊中的rename方法對文件改名
    os.rename(oldname,newname)
    print(oldname,'已經改名為：',newname)
    n+=1
```

tiff檔轉成.jpg檔

```python
print(files)


for file in files:
  img_path = path +"\\"+ file
  img = cv2.imread(img_path)
  print(img_path)
  img_re = np.reshape(img, (1, 129, 934, 3))
  concat = np.concatenate((concat, img_re), axis = 0)


print(concat.shape)


with open(r"C:\Users\rh\Desktop\train_img.npy", "wb") as fpw:
    np.save(fpw, concat)

with open(r"C:\Users\rh\Desktop\train_img.npy", "rb") as fpr:
    c = np.load(fpr)

print(c.shape)
```

將圖片檔轉為nparray

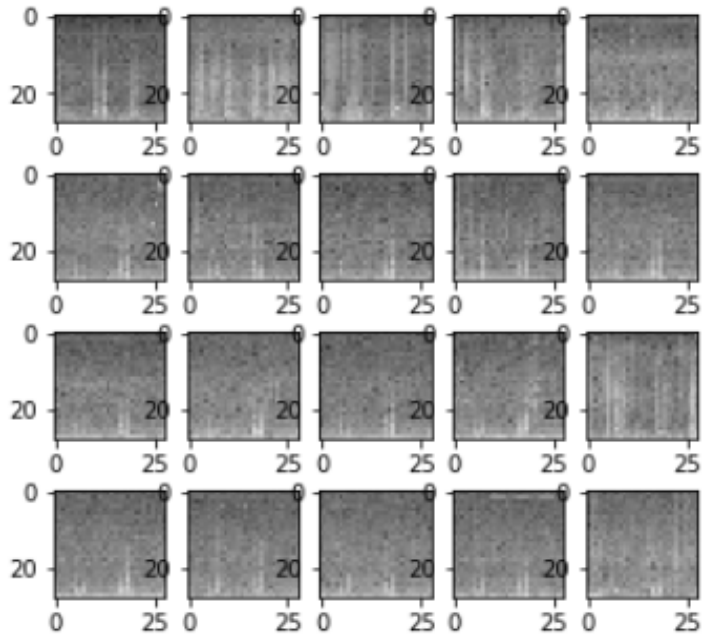
```python
import os
import json

path=r"C:\Users\rh\Desktop\VOICE\train_label"
files = os.listdir(path)

#print(len(files))

s = []

for file in files:
    fname = os.path.splitext(file)[0]
    with open(path+"//"+ file, "r") as test:
        content = test.readline()

        a=content[5]
        s.append(a)

print(s)
print(len(s))
path2=r"C:\Users\rh\Desktop\VOICE\train.json"

with open(path2,"w") as f:
    json.dump(s, f)

#print(content)
```

統整標籤檔案

資料前處理 (2/3)

Step2

- 利用resize來改變圖片的尺寸大小成(64×64)
- label利用one hot encoder的方式轉碼



可視化頻譜圖

```
#One hot encoder
from keras.utils import np_utils
y_train_label=np_utils.to_categorical(y)
Y_test_label=np_utils.to_categorical(Y_test)
```

executed in 52ms, finished 19:39:21 2020-11-28

```
Using TensorFlow backend.
```

```
x.shape, y_train_label.shape, X_test.shape, Y_test_label.shape
```

executed in 7ms, finished 19:39:21 2020-11-28

```
((10742, 64, 64, 3), (10742, 2), (3403, 64, 64, 3), (3403, 2))
```

轉碼程式與輸出圖片及label的shpae

資料前處理 (3/3)

Step3

- 進行資料的標準化，讓所有的特徵值介於0到1之間
- 使梯度運算時能夠更快收斂，降低運算速度
- 為了避免訓練張數太少導致Overfitting，透過Data augmentation的方式增加訓練及驗證張數

```
#One hot encoder
from keras.utils import np_utils
y_train_label=np_utils.to_categorical(y)
Y_test_label=np_utils.to_categorical(Y_test)
```

executed in 52ms, finished 19:39:21 2020-11-28

```
Using TensorFlow backend.
```

```
x.shape, y_train_label.shape, X_test.shape, Y_test_label.shape
```

executed in 7ms, finished 19:39:21 2020-11-28

```
((10742, 64, 64, 3), (10742, 2), (3403, 64, 64, 3), (3403, 2))
```

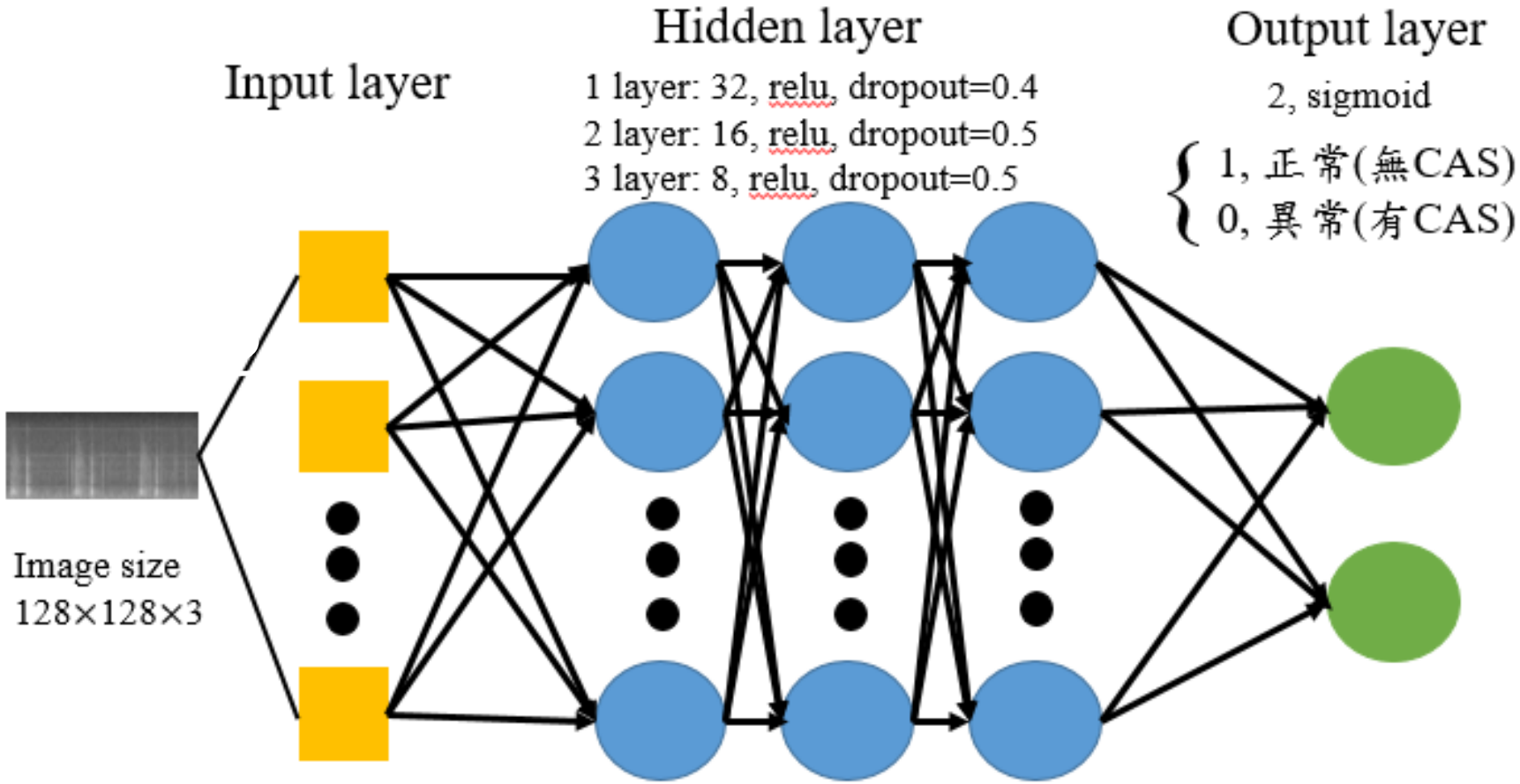
轉碼程式與輸出圖片及label的shape

```
#data augmentation
datagen = ImageDataGenerator(
        rotation_range=5,  # randomly rotate images in the range 5 degrees
        zoom_range = 0.1, # Randomly zoom image 10%
        width_shift_range=0.1,  # randomly shift images horizontally 10%
        height_shift_range=0.1,  # randomly shift images vertically 10%
        horizontal_flip=False,  # randomly flip images
        vertical_flip=False)  # randomly flip images
#datagen.fit(X_train)
```

轉碼程式與輸出圖片及label的shpae

模型建立與訓練

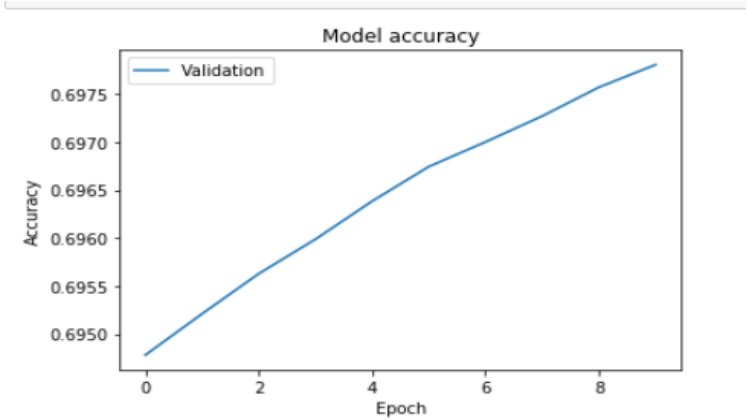
- 進行資料的標準化，讓所有的特徵值介於0到1之間
- 使梯度運算時能夠更快收斂，降低運算速度
- 為了避免訓練張數太少導致Overfitting，透過Data augmentation的方式增加訓練及驗證張數

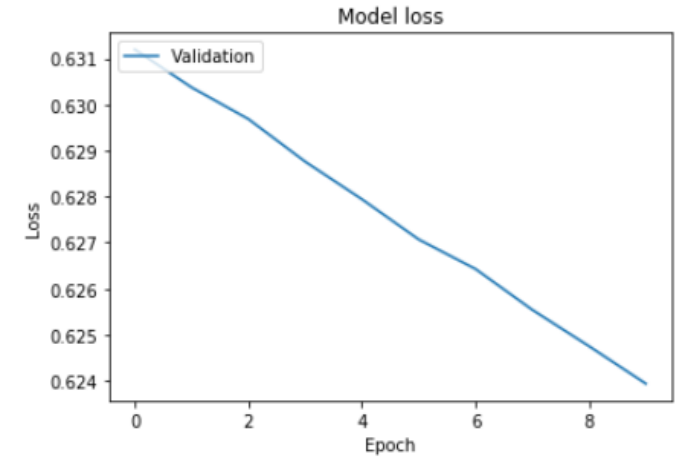


參數調整 (1/3)

初步參數設定

參數/函數	初步設定
圖片Resize大小	128,128
每個epoch資料增強張數	9661
資料分群	9:1
交叉驗證次數	4
Random stae	7
Epoch	10
Batch size	16
Activation function	Relu, Sigmoid
Optimizer	Adam
Loss	Categorical crossentropy
Dense neural	32-16-8-2
Dropout rate	0.4-0.5-0.5
Learning rate	0.000001

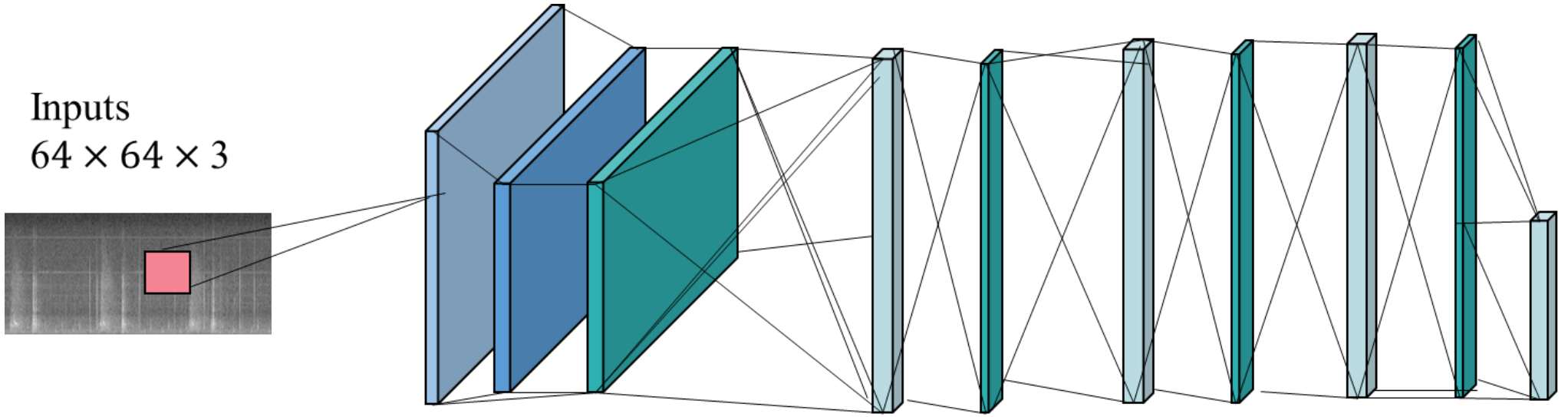




	ACCURACY	LOSS
TRAIN	0.6976	0.6247
VALIDATION	0.7002	0.6090

參數調整 (2/3)

初步模型調整



Depth = 16
2× 2 Conv(2)
MaxPool 2×2

Dense = 32
Activation = relu
Dropout = 0.4

Dense = 16
Activation = relu
Dropout = 0.5

Dense = 8
Activation = relu
Dropout = 0.5

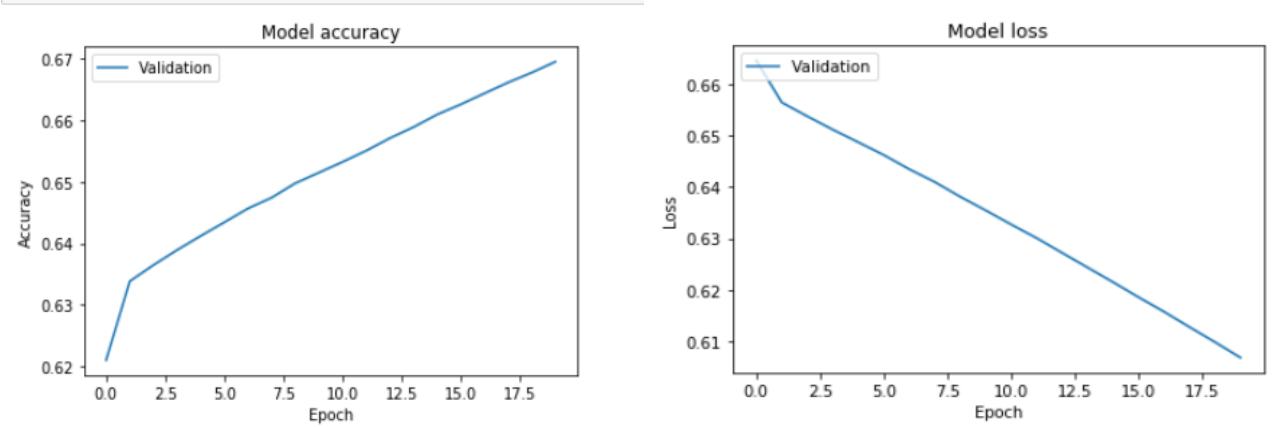
Dense = 2
Activation = sigmoid
Dropout = 0.5

參數調整(2/3)

第一次參數調整

參數/函數	設定
圖片Resize大小	64,64
每個epoch資料增強張數	9661
資料分群	9:1
交叉驗證次數	4
Random stae	42
Epoch	20
Batch size	16
Activation function	Relu, Sigmoid
Optimizer	Adam
Loss	Categorical crossentropy
Dense neural	32-16-8-2
Dropout rate	0.4-0.5-0.5
Learning rate	0.000001

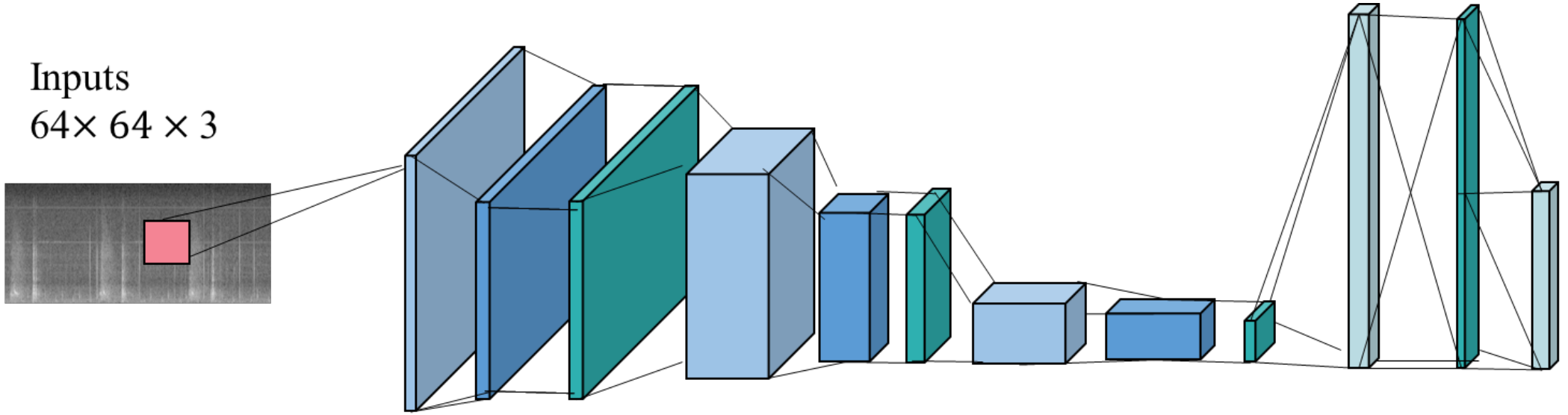
參數/函數	設定
Convolution layer	2,2
Pooling layer	Max pooling (2,2)
各層kernel之初始化	he_normal
各層CNN之正規化	BatchNormalization
自調整學習率	ReduceLROnPlateau



	ACCURACY	LOSS
TRAIN	0.7101	0.5849
VALIDATION	0.6873	0.6158

參數調整(3/3)

第二次模型調整

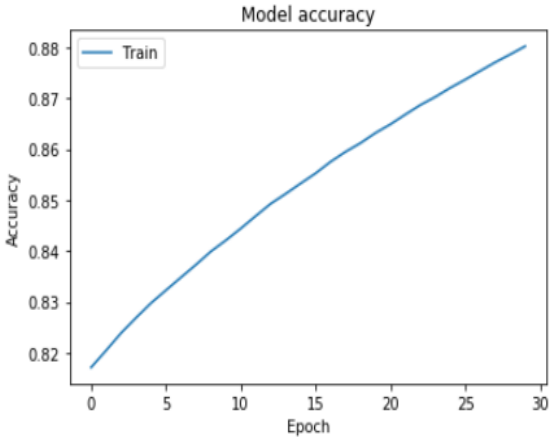


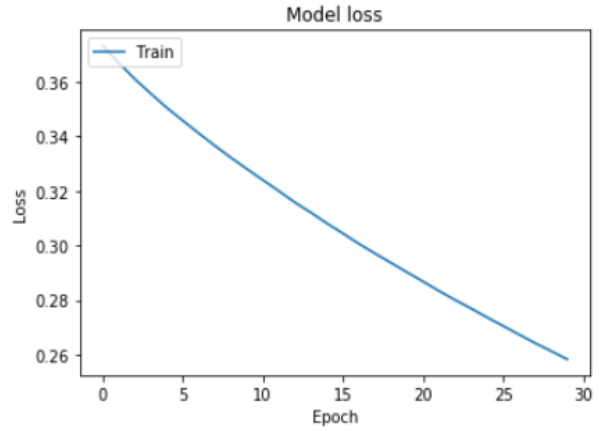
參數調整(3/3)

第二次參數調整

參數/函數	設定
圖片Resize大小	64,64
每個epoch資料增強張數	9661
資料分群	9:1
交叉驗證次數	4
Random stae	42
Epoch	30
Batch size	16
Activation function	Relu, Sigmoid
Optimizer	Adam
Loss	Categorical crossentropy
Dense neural	16-2
Dropout rate	0.5
Learning rate	0.000001

參數/函數	設定
Convolution layer	(3,3)-(2,2)-(2,2)
Pooling layer	Max pooling (2,2)
各層kernel之初始化	he_normal
各層CNN之正規化	BatchNormalization
自調整學習率	ReduceLROnPlateau





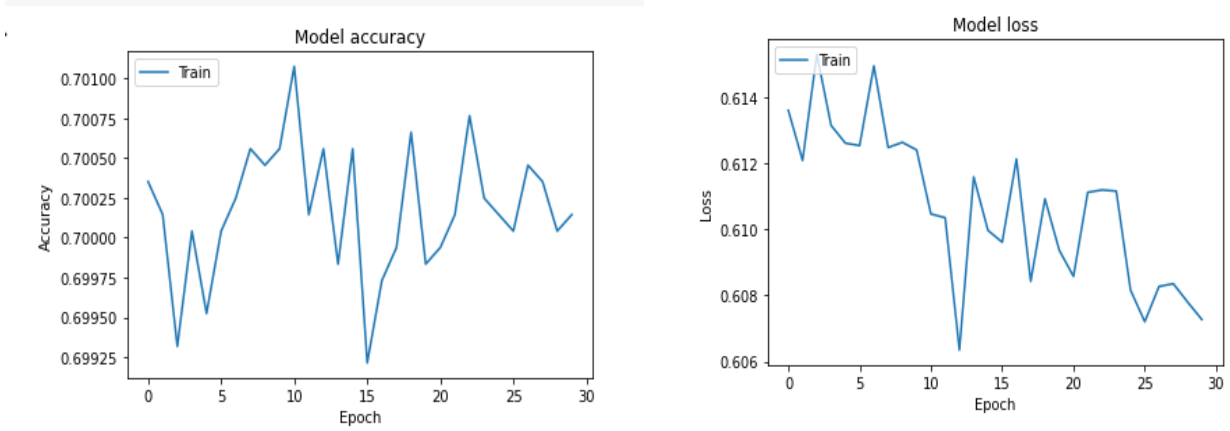
	ACCURACY	LOSS
TRAIN	0.8802	0.2582
VALIDATION	0.8212	0.3750

方法比較 (1/3)

Inception V3

參數/函數	設定
圖片Resize大小	75,75
每個epoch資料增強張數	9661
資料分群	9:1
交叉驗證次數	4
Random stae	42
Epoch	30
Batch size	16
Loss	Categorical crossentropy
Dense neural	1024-128-2
Dropout rate	0.5
Learning rate	0.000001

參數/函數	設定
Activation function	Relu, Sigmoid
各層kernel之初始化	he_normal
各層CNN之正規化	BatchNormalization
自調整學習率	ReduceLROnPlateau



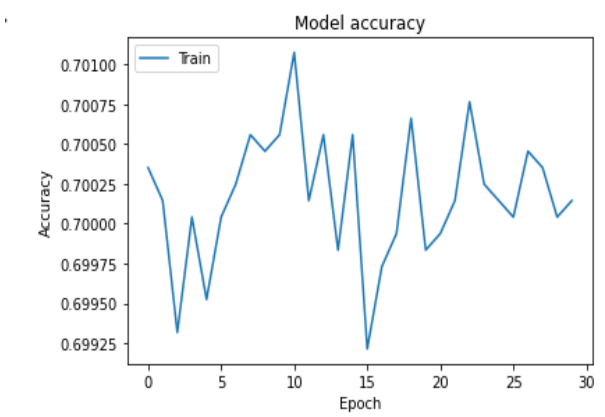
	ACCURACY	LOSS
TRAIN	0.7001	0.6073
VALIDATION	0.7002	0.5959

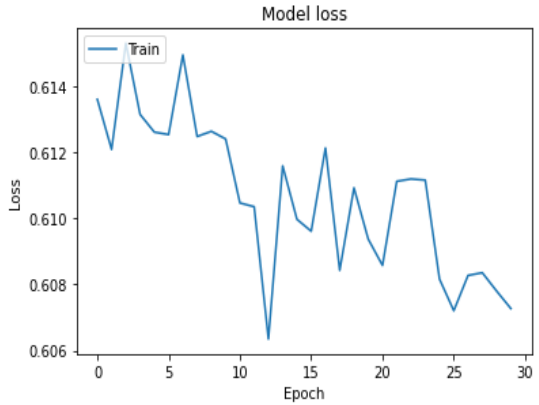
方法比較(2/3)

ResNet50

參數/函數	設定
圖片Resize大小	64,64
每個epoch資料增強張數	9661
資料分群	9:1
交叉驗證次數	4
Random stae	42
Epoch	30
Batch size	16
Loss	Categorical crossentropy
Dense neural	1024-128-2
Dropout rate	0.5
Learning rate	0.000001

參數/函數	設定
Activation function	Relu, Sigmoid
各層kernel之初始化	he_normal
各層CNN之正規化	BatchNormalization
自調整學習率	ReduceLROnPlateau





	ACCURACY	LOSS
TRAIN	0.7001	0.6191
VALIDATION	0.7002	0.6109

方法比較(3/3)

- 使用的模型是自己所建立的，在多次調整後，均無法達到更好的改善\
- 利用Transfer learning方式來與我們的結果進行比較
- 選擇Resnet50與Google Inception V3為Transfer learning的model

Model	Training	Validation	Test
	Accuracy/Loss	Accuracy/Loss	Accuracy/Loss
Proposed method	**0.8882/0.2582**	**0.8212/0.3750**	0.6844/**0.6115**
Inception V3	0.7001/0.5959	0.7002/0.6073	0.6873/0.6170
ResNet50	0.7001/0.6109	0.7002/0.6205	0.6873/0.6213

- Transfer Learning的結果沒有比想像中的好，表現結果甚至不如我們自己所建立的model
- 可能因其他超參數設定並不是這兩種model最合適的設定
- 從loss的值看出我們所訓練的model在相同情況下表現最佳

Outline

PLACED LOGO
YOUR COMPANY NAME HERE

背景介紹

方法介紹

個案研究

結論

結論 (1/2)

貢獻

- 雖然在測試集上面沒有顯著的差異，但是訓練過程中的表現很好，對於肺音異常分類在這項結果當中
- 提出的建置模型方法可針對不同的資料及進行設計，透過參數調整使模型更加擬合資料
- 利用Dropout等等的手法保持模型在訓練當中的彈性

侷限性

- 本專題的資料集在過去未被分析過，因此在結果的表現還有待改善。
- 若導入其他不同儀器蒐集的肺音異常資料可能會使辨識率更低
- 這個架構固然可行，但可能對於當中的參數要自己進行更改
- 若要用這個模型在現實中實行，需要參考原始資料集蒐集的規格，利用相同的設備進行才可以重現本次專題的實驗結果

結論 (2/2)

適用性

- 可以朝向臨床實施
- 將資料丟入模型當中提供醫生建議
- 協同別的醫生進行判斷
- 再次收集資料，使模型更加完善

未來改善

- 測試集的準確率可以透過增加資料集來提升
  （例如一年一次肺音檢查）
- 本次資料與前次資料（已由醫生正確判別完成）做比對
- 透過深度學習並聯的方式來降低錯誤率

參考文獻

1. Chamberlain, D., Kodgule, R., Ganelin, D., Miglani, V., & Fletcher, R. R. (2016, August). Application of semi-supervised deep learning to lung sound analysis. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 804-807). IEEE.
2. Li, L., Xu, W., Hong, Q., Tong, F., & Wu, J. (2016, October). Classification between normal and adventitious lung sounds using deep neural network. In *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)* (pp. 1-5). IEEE.
3. https://jkjahuang.pixnet.net/blog/post/9028106-%E3%80%90chest%E3%80%91%E5%91%BC%E5%90%B8%E9%9F%B3%E8%81%BD%E8%A8%BA
4. https://www.itread01.com/content/1550265688.html
5. https://blog.gtwang.org/programming/keras-resnet-50-pre-trained-model-build-dogs-cats-image-classification-system/
6. https://hackmd.io/@allen108108/H1MFrV9WH

Thank you