運用CNN進行胸腔X光辨識

110034566 黃采琳

Outline

1. 研究背景
2. 資料前處理
3. 模型建立與參數優化
4. 結論

1

研究背景

背景描述

疫情初期
民眾恐慌

一有相關呼吸道症狀
就前往醫院進行篩檢

對醫院造成
極大的負擔

放射科醫師一天約需看
數百張至數千張的X光片

5W1H

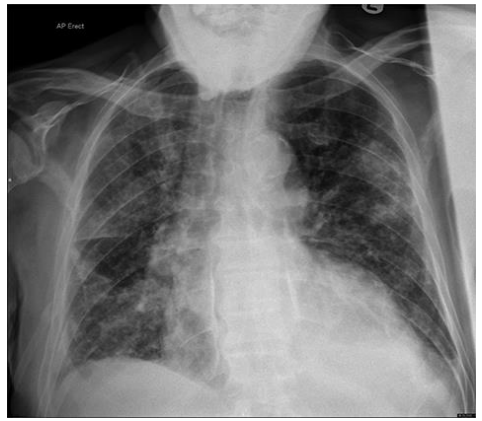
What	醫師判讀的胸腔X光片過多，可能會有判讀錯誤的情形
Who	臨床醫師、放射科醫師
When	欲檢測是否罹患COVID19
Where	各大醫院
Why	協助降低醫院人力負擔，能將資源放在嚴重患者身上
How	運用深度學習方法

2

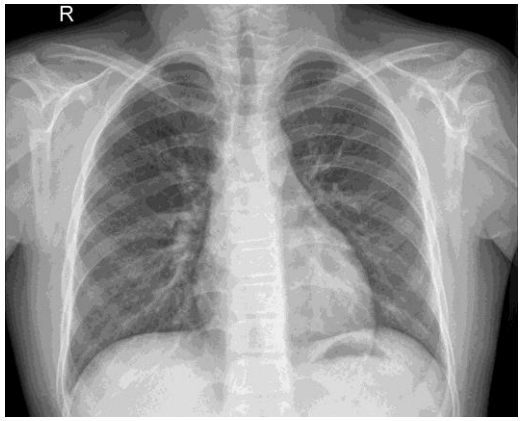
資料前處理

資料來源

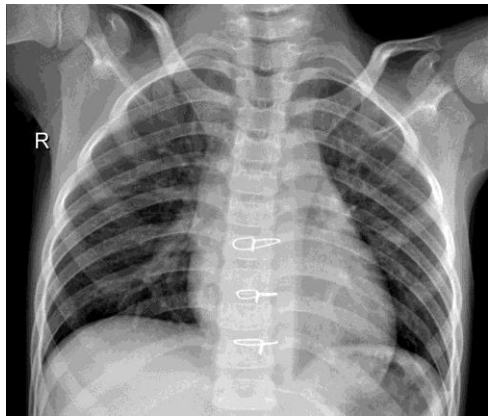
- From Kaggle : Chest X-Ray (Pneumonia,Covid-19,Tuberculosis)資料集



Covid19
共有558張



Normal
共有1530張



Pneumonia
共有1257張

Tuberculosis
共有 586張

資料前處理(1/4)

以8:1:1切分
訓練、驗證、測試集

```
#訓練、驗證、測試：8:1:1
#train
def  moveFile(fileDir):
    pathDir  =  os.listdir(fileDir)         #取圖片的原始路徑
    filenumber=len(pathDir)
    rate=0.8        #自定義抽取圖片的比例，比方說100張抽10張，那就是0.1
    picknumber=int(filenumber*rate)  #按照rate比例從資料夾中取一定數量圖片
    sample  =  random.sample(pathDir,  picknumber)     #隨機選取picknumber數量的樣本圖片
    print  (sample)
    for  name  in  sample:
        shutil.move(fileDir+name,  tarDir+name)
    return
NAME  =  ["COVID19","NORMAL","PNEUMONIA","TURBERCULOSIS"]
name  =  ["covid19","normal","pneumonia","turberculosis"]
if  __name__  ==  '__main__':
    for  i  in  range(4):
          fileDir  =  "/content/drive/MyDrive/chest_x_ray/{}/"  .  format(NAME[i])          #源圖片資料夾路徑
          tarDir  =  '/content/drive/MyDrive/chest_x_ray/train/{}/'.format(name[i])          #移動到新的資料夾路徑
          moveFile(fileDir)
```

資料前處理(2/4)

Augmentor 是一個 Python Package，
用於幫助機器學習任務的影像增強與生成。

- 透視偏斜（Perspective Skewing）
- 彈性變形（Elastic Distortions）
- 旋轉（Rotating）
- 裁剪（Cropping）
- 鏡像（Mirroring）

```
p.rotate90(probability=0.5)

p.rotate(probability=0.5,max_left_rotation=25,
max_right_rotation=10)

p.sample(938)
```

Resource: https://cynthiachuang.github.io/Augmentor-Image-Augmentation-Library-in-Python

資料前處理(3/4)

新增前

```
#各類別訓練集數量
name = ["covid19","normal","pneumonia","turberculosis"]
for i in range(4):
    train_path = "/content/drive/MyDrive/chest_x_ray/train/{}" .format(name[i])
    read = os.listdir(train_path)
    num = len(read)
    print(num)
```

```
552
1512
1200
560
```

新增後

```
#新增後 #各類別訓練集數量
name = ["covid19","normal","pneumonia","turberculosis"]
for i in range(4):
    train_path = "/content/drive/MyDrive/chest_x_ray/train/{}" .format(name[i])
    read = os.listdir(train_path)
    num = len(read)
    print(num)
```

```
1500
1514
1500
1500
```

資料前處理(4/4)

```
data_gen = ImageDataGenerator(rescale=1./255)
training_set = data_gen.flow_from_directory(path_train,
                                            class_mode='categorical',
                                            shuffle=True,
                                            target_size=(224,224),
                                            batch_size=64,
                                            )
val_set = data_gen.flow_from_directory(path_val,
                                            class_mode='categorical',
                                            shuffle=True,
                                            target_size=(224,224),
                                            batch_size=64,
                                            )
testing_set = data_gen.flow_from_directory(path_test,
                                            class_mode='categorical',
                                            shuffle=True,
                                            target_size=(224,224),
                                            batch_size=64,
                                            )

training_set.class_indices
```

```
Found 6014 images belonging to 4 classes.
Found 752 images belonging to 4 classes.
Found 752 images belonging to 4 classes.
{'covid19': 0, 'normal': 1, 'pneumonia': 2, 'turberculosis': 3}
```

- 進行圖像歸一化
- 調整圖片大小

3

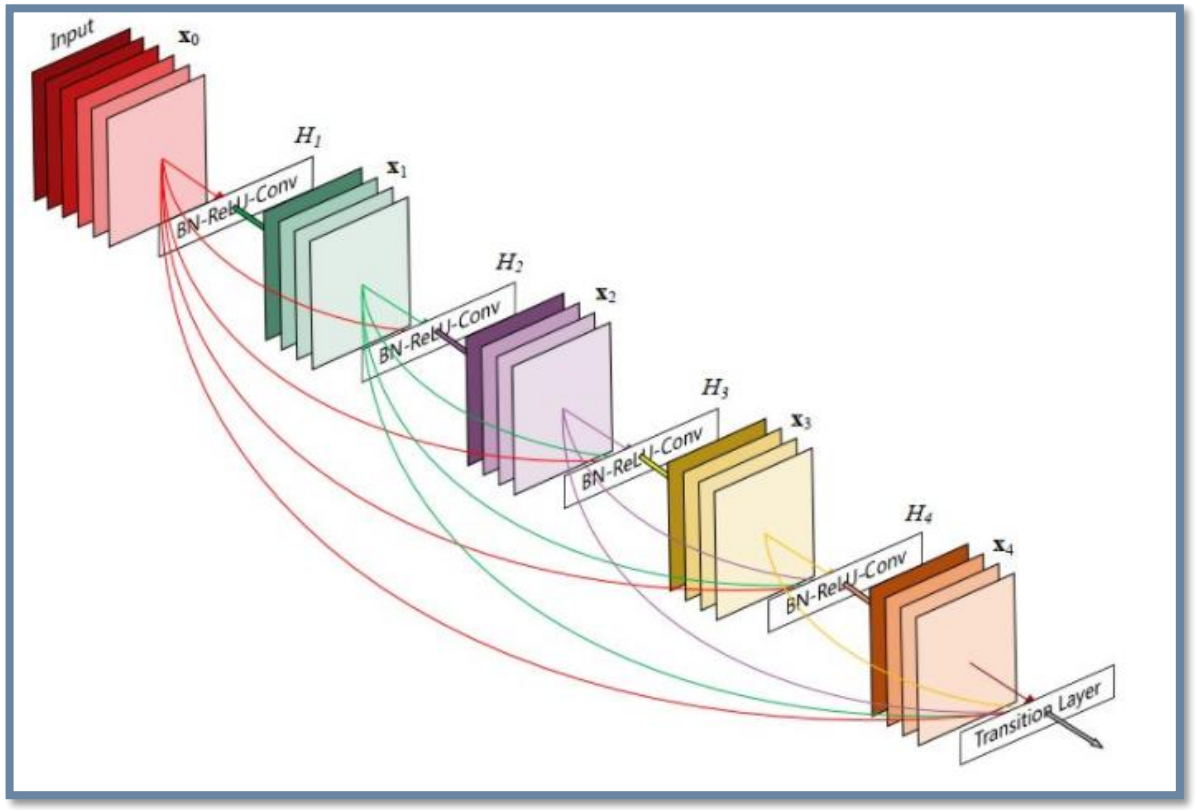
模型建立與參數優化

Densenet(1/3)

- DenseNet 全名為
  Densely Connected Convolutional Network

- 許多「層與層的連結」來達到特徵重用性。

- 比如我們有L層卷積神經網路，
  那就有L個(層與層之間的)連結，
  但DenseNet設計成有L(L+1)/2個連結



Densenet(2/3)

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			

Densenet(3/3)

(1) 減緩梯度消失的問題

由於Densely connected，反向梯度傳播十分容易，模型收斂效果佳。

(2) 特徵重用性

從低階特徵到高階特徵都會被直連到最後一層卷積層，
這讓下一層接受到更全面的圖像資訊。

(3) 減少參數量

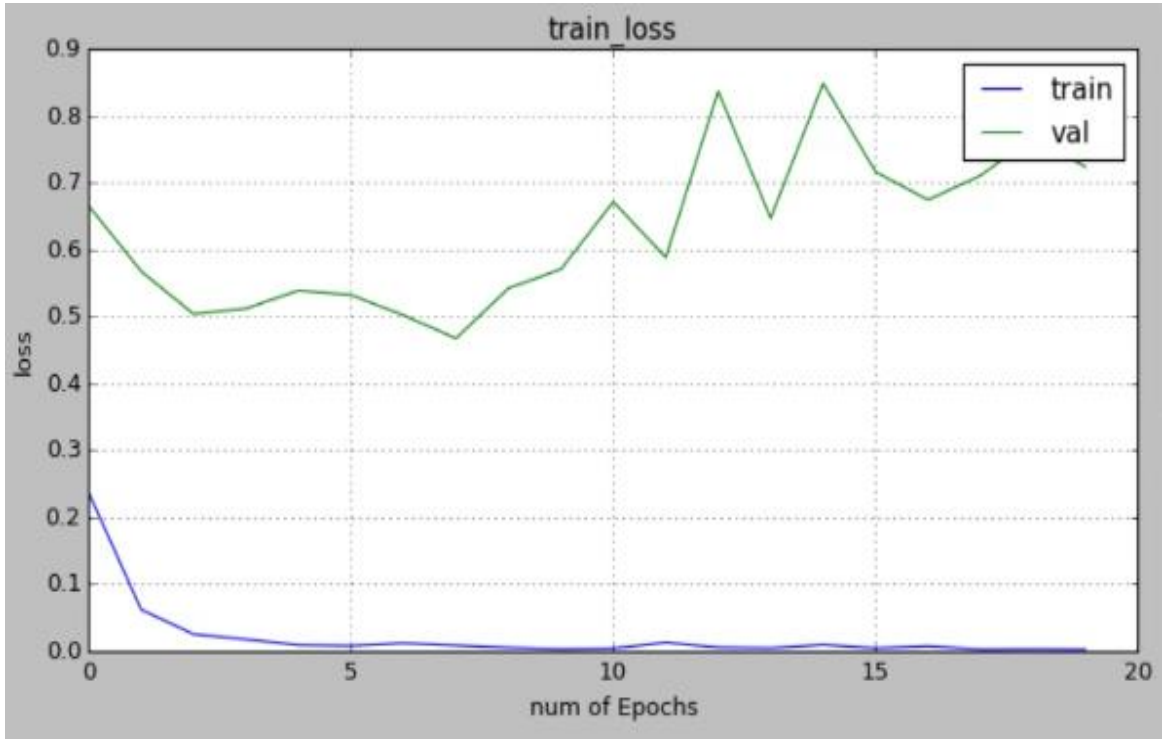
對於舊的特徵圖(feature-map)是不需要再去重新學習，能減少許多參數。

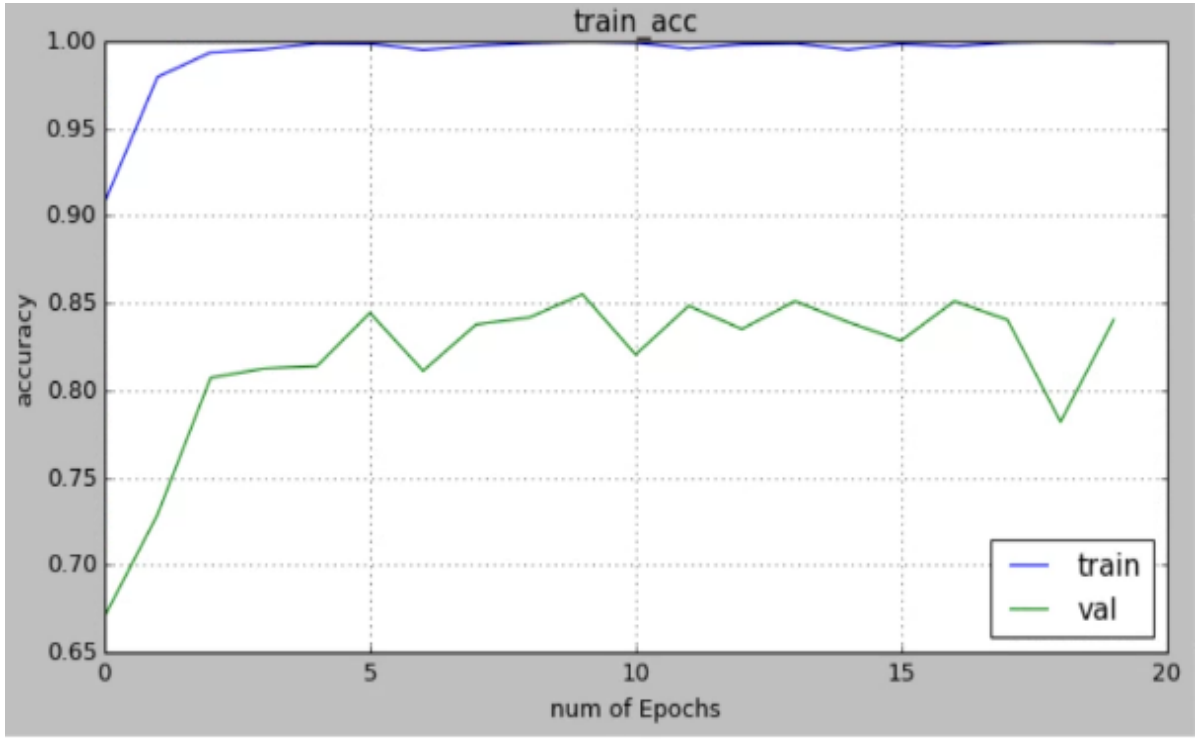
模型建立

```
#Densenet121
from  tensorflow.keras.applications  import  DenseNet121
def  get_model():
        densenet  =  DenseNet121(weights='imagenet',
                                  include_top=False,
                                  input_shape=(224,224,3)
                                                      )
        model  =  tf.keras.models.Sequential([densenet,
                                              GlobalAveragePooling2D(),
                                              Dense(512,  activation='relu'),
                                              BatchNormalization(),
                                              Dropout(0.3),
                                              Dense(4,  activation='sigmoid')
                                                                          ])
        model.compile(optimizer=Adam(lr=0.001),
                                  loss='binary_crossentropy',
                                  metrics=['accuracy']
                                  )

        return  model

model  =  get_model()
model.summary()
```

- Optimizer: Adam
- Loss : binary_crossentropy
- Dropout : 0.3
- Learning rate : 0.001

初始實驗結果


train_loss
loss
num of Epochs
train
val
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
5
10
15
20


train_acc
accuracy
num of Epochs
train
val
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
0
5
10
15
20

參數優化(1/3)

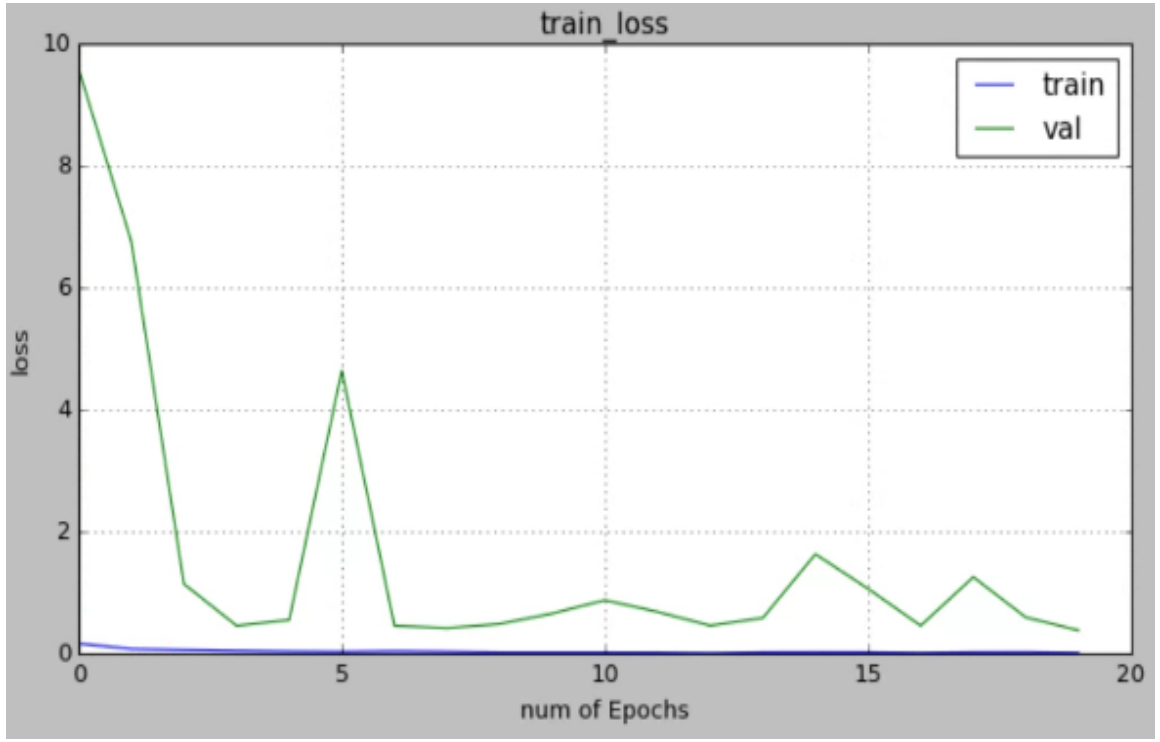
	optimizer	Learning rate	dropout	Activation function
水準一	Adam	0.001	0.3	relu
水準二	SGD	0.0001	0.4	tanh
水準三	Adagrad	0.005	0.5	sigmoid

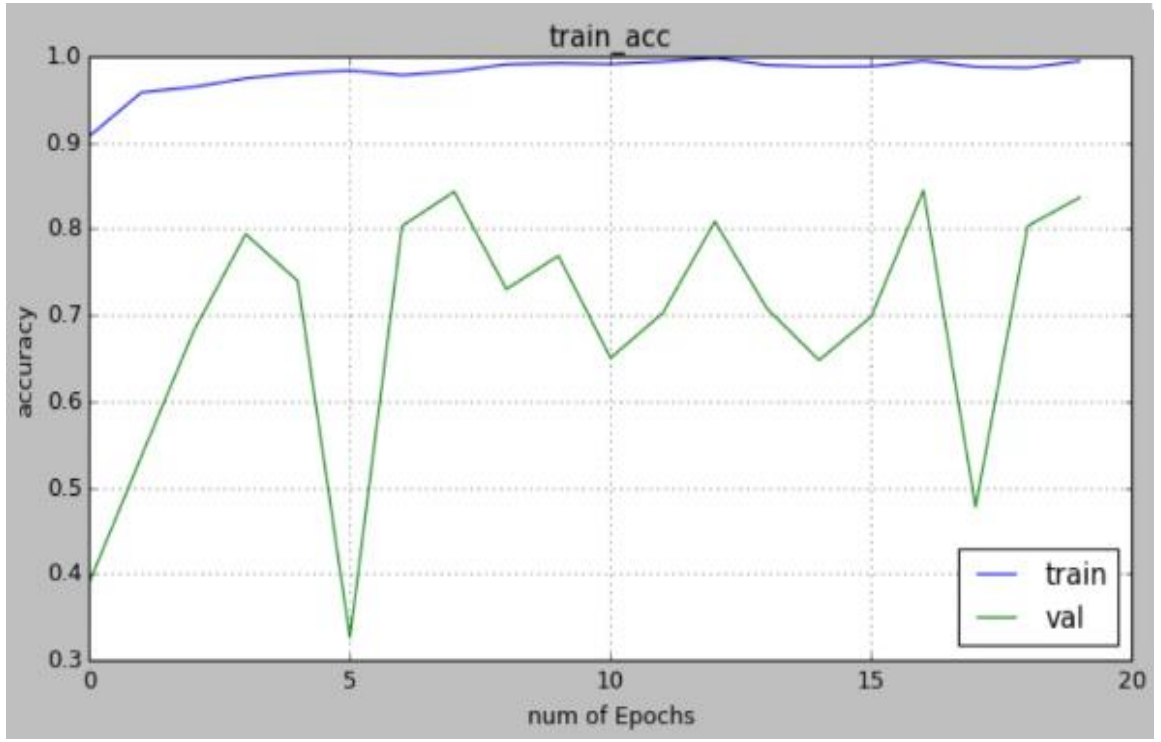
參數優化(2/3)

L9直交表

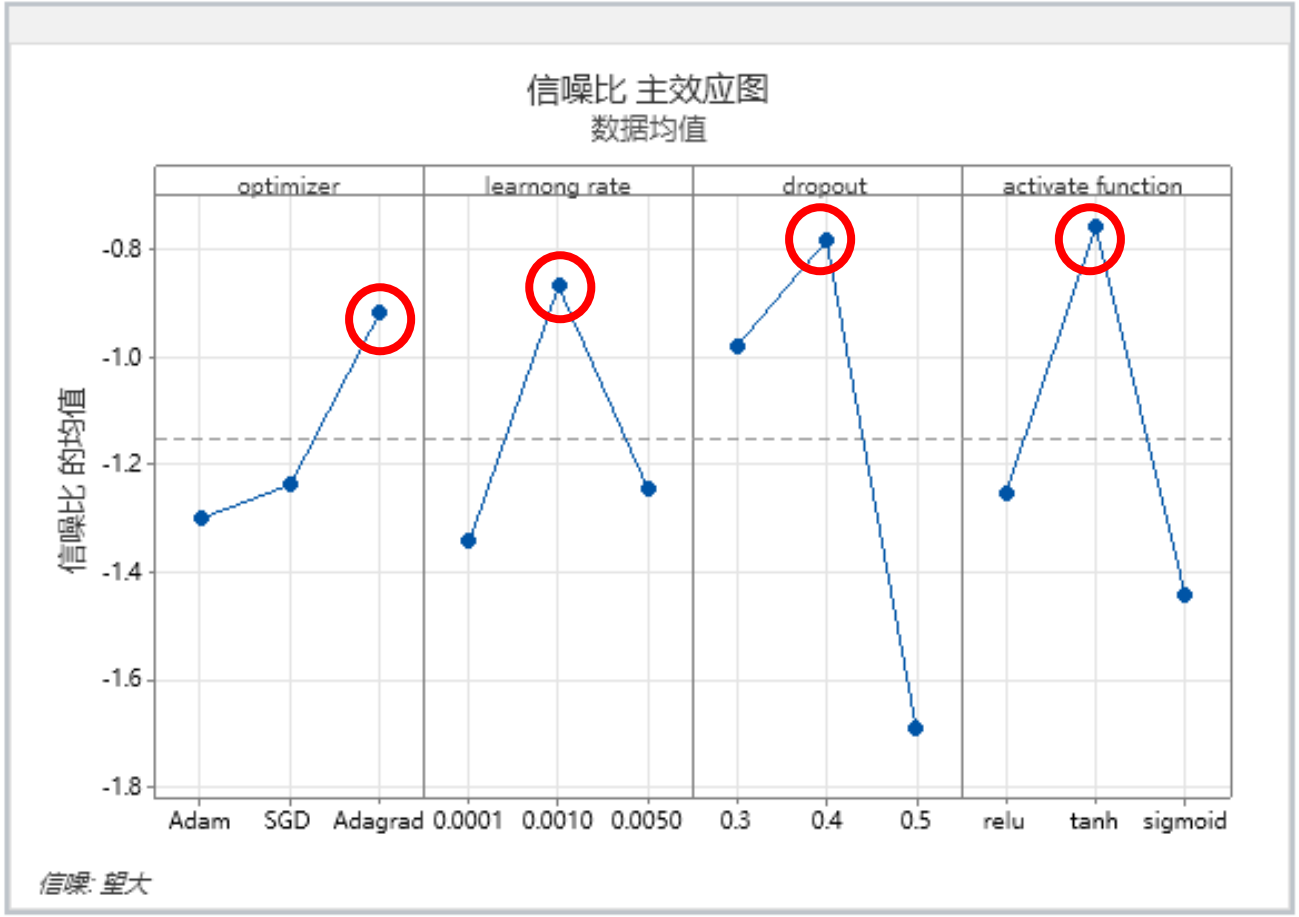
	optimizer	Learning rate	dropout	Activate function	Train acc	Test acc
1	Adam	0.001	0.3	relu	0.9943	0.8231
2	Adam	0.0001	0.4	tanh	0.9992	0.8564
3	Adam	0.005	0.5	sigmoid	0.9764	0.6609
4	SGD	0.001	0.4	sigmoid	0.9757	0.8457
5	SGD	0.0001	0.5	relu	0.7970	0.7793
6	SGD	0.005	0.3	tanh	0.9980	0.8511
7	Adagrad	0.001	0.5	tanh	0.9940	0.8511
8	Adagrad	0.0001	0.3	sigmoid	0.9214	0.8231
9	Adagrad	0.005	0.4	relu	0.9995	0.8540

實驗2


train_loss
loss
num of Epochs
train
val
0
2
4
6
8
10
0
5
10
15
20


train_acc
accuracy
num of Epochs
train
val
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0
5
10
15
20

參數優化(3/3)



Optimizer: Adagrad
Learning rate: 0.001
Dropout: 0.4
Activate function: tanh

Train_acc : 0.9940
Test_acc : 0.8497

4

結論

研究貢獻

- 本研究透過建立CNN模型來協助醫生快速判讀患者的疾病，
  特別是對新手醫師來說，可能會因為經驗不足而導致對某些變化視而不見，
  此模型可以降低醫師的出錯率。

- 另外為了讓大型醫院能專注治療病情嚴重的個案，
  會將篩檢量能分配到第二、三線的醫院，
  透過人工智慧的方法能提供醫院具有良好效率的工具，減輕醫院的人力負擔。

未來展望

- 可以嘗試增加可辨識疾病的數量，或建立一系統除了辨識疾病之外，還能標記出異常位置，或是有疑點的部位，降低對個人經驗值的依賴，協助醫生進行後續相關醫療處理。

Thanks for Listening