Machine Learning - Assignment 3
COVID-19 30-day Mortality Prediction from CXR Report

106070038 杜葳葳

一、Paper Survey

- **Transfer Learning from Chest X-Ray Pre-trained Convolutional Neural Network for Learning Mammogram Data (2018)**
  - 此篇論文的目標是訓練一個可以用 X 光片判斷是否患有乳癌的模型，分為三個階段：Pre-Training、Transfer Learning、Comparison
  - Related Work：此類問題 Deep Learning 的準確性比 SVM 高，若資料集的資料量不夠，可以使用 Transfer Learning 來精進
  - Method：
    - Pre-Training Phase：使用 CheXNet 為預訓練模型，CheXNet 是一個 121 層的 DenseNet 模型，每層 layer 有四個 dense blocks。此篇論文是使用 Chest-X-Ray 資料集、北京大學實作的 CheXNet 模型
    - Transfer Learning Phase：因為乳房攝影的影像為四個一組，故將模型改為四個平行的輸入通道
    - Comparison Phase：比較不同數量的 dense blocks、不同 layer 數目、不同 Learning rate、不同 Dropout rate、不同 L2 regularization rate，每次訓練跑 20 個 epochs、比較 Accuracy，用 Adam 優化、Loss function 用 Categorical cross-entropy，Training set：Validation set：Testing set = 60：20：20

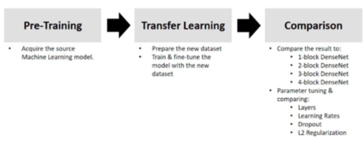


Fig. 1: Research Workflow

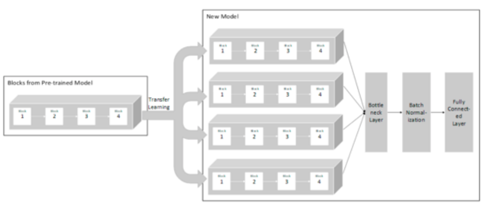


Fig. 2: Illustration of Transfer Learning Scheme

Table 1: The result of the experiment with blocks.

Blocks	Training		Validation		Running Time
	Loss	Accuracy	Loss	Accuracy	(20 Epochs)
4	0.3493	96.31 %	0.4380	87.31 %	2h 38m 58s
3	0.3407	97.20 %	0.4408	87.12 %	2h 31m 56s
2	**0.3960**	**91.68** %	**0.4324**	**88.27** %	**2h 19m 32s**
1	0.4625	85.76 %	0.4588	85.77 %	2h 6m 49s

Table 2: The result of the experiment with layers.

Layers	Training		Validation		Running Time
	Loss	Accuracy	Loss	Accuracy	(20 Epochs)
12	0.4013	91.39 %	0.4290	88.46 %	2h 11m 38s
11	0.3262	98.80 %	0.4232	88.65 %	2h 10m 41s
10	0.3293	98.47 %	0.4240	88.65 %	2h 6m 6s
9	0.3456	97.01 %	0.4250	88.46 %	2h 9m 2s
8	0.3436	97.12 %	0.4258	88.46 %	2h 6m 56s
7	0.3220	99.22 %	0.4279	88.65 %	2h 6m 7s
6	**0.3345**	**98.02 %**	**0.4107**	**90.38 %**	**2h 3m 57s**
5	0.3314	98.34 %	0.4258	88.85 %	2h 2m 19s
4	0.3366	97.76 %	0.4149	89.61 %	2h 2m 1s
3	0.3442	97.16 %	0.4155	89.62 %	1h 59m 48s
2	0.3392	97.60 %	0.4135	90.00 %	2h 0m 1s
1	0.3465	96.82 %	0.4263	88.85 %	1h 59m 21s

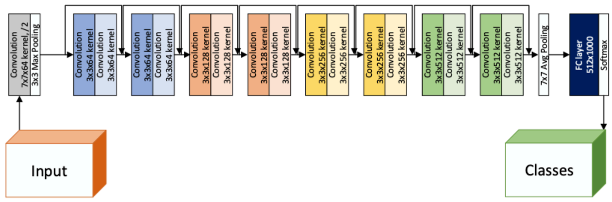
  - Conclusion：
    - 僅使用 CheXNet 的前兩個 Dense blocks，最後一個 block 使用 6 層 layers，最佳 Learning rate 為 1e-3
    - Validation accuracy：90.38%

- **Deep-COVID: Predicting COVID-19 From Chest X-Ray Images Using Deep Transfer Learning (2020)**
  - 此篇論文的目標是用 X 光影像判斷該病患是否感染 COVID-19 的二元分類問題，使用 end-to-end 的深度學習模型，同樣可分為三個階段：Pre-Training、Transfer Learning、Comparison
  - Method：
    - Dataset：5000 Chest X-ray images (2000 training, 3000 testing)

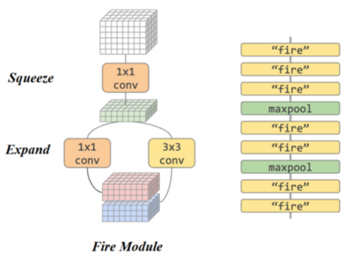
Table 1. Number of images per category in COVID-Xray-5k dataset.

Split	COVID-19	Non-COVID
Training Set	84 (420 after augmentation)	2000
Test Set	100	3000

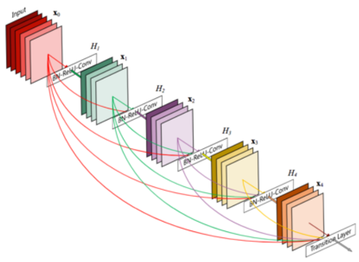
    - Pre-processing：因為資料不夠多，使用 Data augmentation 來產生更多圖片，包含將照片翻面、旋轉、加上雜訊，產生比原始資料多五倍的照片。使用四種用 ImageNet 預訓練好的卷積神經網路，分別為：ResNet-18、ResNet-50、SqueezeNet、DenseNet-121，再將最終層做小幅的修改，
    - Transfer Learning Phase：僅 fine-tune 最後一層 CNN、使用 Pre-trained model 作為特徵提取。
      - ResNet-18、ResNet-50：用 ImageNet 資料集訓練，以 shortcut connection 的方式可以跳過幾層 layer



      - SqueezeNet：可達到 AlexNet 的準確度、但小於 0.5MB，是個輕量的模型



      - DenseNet-121：每一層都將前面所有層的特徵圖譜作為輸入



- **Comparison Phase：**
  - 用 Sensitivity、Specificity、ROC curve、Area under the curve、Precision Recall curve、Histogram of the prediction scores
  - 由下表可知，SqueezeNet 和 ResNet 的表現最好

Table 6. Comparison of sensitivity and specificity of four state-of-the-art deep neural networks.

Model	Sensitivity	Specificity
ResNet18	98% ± 2.7%	90.7% ± 1.1%
ResNet50	98% ± 2.7%	89.6% ± 1.1%
SqueezeNet	98% ± 2.7%	92.9% ± 0.9%
Densenet-121	98% ± 2.7%	75.1% ± 1.5%

二、 Preprocessing Steps

- 用 shutil 函式庫，讀取每張照片對應到 csv 檔的 output，將 output=0 和 output=1 分別放入兩個資料夾

```python
# pre_processing.py
import pandas as pd
import os
from os import listdir
import shutil

output0_List = os.listdir('./split_data/output_0')
output1_List = os.listdir('./split_data/output_1')

for i in range(983):
    filename = output0_List[i]
    pace = "./split_data/output_0/" + str(filename)
    shutil.move(pace, "./data/train/non")

for i in range(984,1229):
    filename = output0_List[i]
    pace = "./split_data/output_0/" + str(filename)
    shutil.move(pace, "./data/val/non")

for i in range(131):
    filename = output1_List[i]
    pace = "./split_data/output_1/" + str(filename)
    shutil.move(pace, "./data/train/covid")

for i in range(132,164):
    filename = output1_List[i]
    pace = "./split_data/output_1/" + str(filename)
    shutil.move(pace, "./data/val/covid")
```

三、 Model and Features

- 實驗不同 Pre-trained model

(5 epochs, learning rate = 0.001, train:valid = 4:1, adam optimizer)

arch	ResNet-18	ResNet-34	ResNet-50	SqueezeNet1_0	SqueezeNet1_1
TP	4	9	16	15	6
FN	26	21	14	15	24
FP	23	34	98	44	46
TN	225	214	150	204	202
F1 score	**0.14**	**0.25**	**0.22**	**0.33**	**0.15**

- 實驗是否要 Normalize

arch	No	Yes
TP	15	13
FN	15	17
FP	44	38
TN	204	210
F1 score	**0.33**	**0.32**

- 實驗 F1 score average 參數

F1 score	Macro	Micro	Weighted	Binary
TP	15	19	15	16
FN	15	11	15	14
FP	44	45	40	50
TN	204	203	208	198
F1 score	**0.33**	**0.4**	**0.35**	**0.33**

- 實驗 cnn_learner 的參數 cut

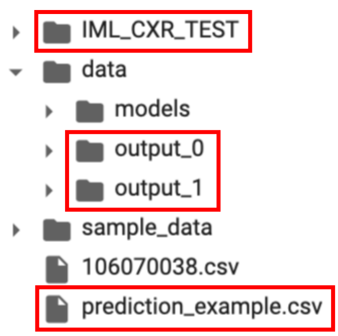
#epochs	-1	1	2	20	None
TP	12	14	8	9	19
FN	18	16	22	21	11
FP	47	43	29	29	45
TN	201	205	220	219	203
F1 score	0.27	0.32	0.24	0.26	**0.4**

綜合以上，最後使用 **SqueezeNet1_0** 作為 pre-trained model

以下為模型各個 layer 的參數：

```
Sequential (Input shape: 64)
================================================================
Layer (type)         Output Shape         Param #    Trainable
================================================================
                     64 x 96 x 157 x 157
Conv2d                                    14208      False
ReLU
MaxPool2d
________________________________________________________________
                     64 x 16 x 78 x 78
Conv2d                                    1552       False
ReLU
________________________________________________________________
                     64 x 64 x 78 x 78
Conv2d                                    1088       False
ReLU
________________________________________________________________
                     64 x 64 x 78 x 78
Conv2d                                    9280       False
ReLU
________________________________________________________________
                     64 x 16 x 78 x 78
Conv2d                                    2064       False
ReLU
________________________________________________________________
                     64 x 64 x 78 x 78
Conv2d                                    1088       False
ReLU
________________________________________________________________
                     64 x 64 x 78 x 78
Conv2d                                    9280       False
ReLU
________________________________________________________________
                     64 x 32 x 78 x 78
Conv2d                                    4128       False
ReLU
________________________________________________________________
                     64 x 128 x 78 x 78
Conv2d                                    4224       False
ReLU
________________________________________________________________
                     64 x 128 x 78 x 78
Conv2d                                    36992      False
ReLU
MaxPool2d
________________________________________________________________
                     64 x 32 x 39 x 39
Conv2d                                    8224       False
ReLU
________________________________________________________________
                     64 x 128 x 39 x 39
Conv2d                                    4224       False
ReLU
________________________________________________________________
                     64 x 128 x 39 x 39
Conv2d                                    36992      False
ReLU
________________________________________________________________
                     64 x 48 x 39 x 39
Conv2d                                    12336      False
ReLU
________________________________________________________________
                     64 x 192 x 39 x 39
Conv2d                                    9408       False
ReLU
________________________________________________________________
                     64 x 192 x 39 x 39
Conv2d                                    83136      False
ReLU
________________________________________________________________
                     64 x 48 x 39 x 39
Conv2d                                    18480      False
ReLU
________________________________________________________________
                     64 x 192 x 39 x 39
Conv2d                                    9408       False
ReLU
________________________________________________________________
                     64 x 192 x 39 x 39
Conv2d                                    83136      False
ReLU
________________________________________________________________
                     64 x 64 x 39 x 39
Conv2d                                    24640      False
ReLU
________________________________________________________________
                     64 x 256 x 39 x 39
Conv2d                                    16640      False
ReLU
________________________________________________________________
                     64 x 256 x 39 x 39
Conv2d                                    147712     False
ReLU
MaxPool2d
________________________________________________________________
                     64 x 64 x 19 x 19
Conv2d                                    32832      False
ReLU
________________________________________________________________
                     64 x 256 x 19 x 19
Conv2d                                    16640      False
ReLU
________________________________________________________________
                     64 x 256 x 19 x 19
Conv2d                                    147712     False
ReLU
AdaptiveAvgPool2d
AdaptiveMaxPool2d
Flatten
BatchNorm1d                               2048       True
Dropout
________________________________________________________________
                     64 x 512
Linear                                    524288     True
ReLU
BatchNorm1d                               1024       True
Dropout
________________________________________________________________
                     64 x 2
Linear                                    1024       True
________________________________________________________________

Total params: 1,263,808
Total trainable params: 528,384
Total non-trainable params: 735,424

Optimizer used: <function Adam at 0x7fc316a5ca60>
Loss function: FlattenedLoss of CrossEntropyLoss()

Model frozen up to parameter group #2

Callbacks:
  - TrainEvalCallback
  - Recorder
  - ProgressCallback
```

四、 How to use the model file



右圖為資料夾的結構，data 為 training data，在前處理的步驟中，分成 output_0、output_1 兩個資料夾(此兩個資料夾有上傳至 ilms)。

IML_CXR_TEST 為 testing data，

prediction_example.csv 為 output example，

output 檔的檔名為 106070038.csv

Step 1: 傳入 IML_CXR_TEST、output_0、output_1、prediction_example.csv

Step 2: 執行.ipynb 中 Model 1 的部分

Step 3: 得到 output 檔 (106070038.csv)

五、Summary

- 最初有嘗試自己搭建模型(如下圖所示)，是使用 ResNet-50 為 pre-trained model，然而效果皆不如預期

```
#### load model
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False

# Parameters of newly constructed modules have requires_grad=True by default
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)


model_conv = model_conv.to(device)
criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opoosed to before.
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr= args.learning_rate, momentum= args.momentum)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
```

- 於是最後使用 **fastai 套件**中的函式庫，**cnn_learner** 這個 function 來建模型，有嘗試不同的參數組合，亦有在 head 的地方自行搭建幾層 layer 再接到 SqueezeNet 上
- 前期花蠻多時間在閱讀論文和實驗不同模型的好壞，但由於受限於在本地端執行時電腦的運算資源有限，和用 colab 的 GPU 加速時有儲存空間的限制，故最終模型的 F1 僅落在 0.4 多左右，仍有進步空間。