2.1 Task A: Performance between BCE loss and BC loss (20 pts)

Compare and analyze the model's performance such as loss and accuracy in both training and testing phases when applying Binary Cross-Entropy (BCE) loss and Cross-Entropy (CE) loss. To ensure a fair comparison, maintain the same deep learning architecture and hyperparameters.

1. training phases BCE V.S. CE

```
model = nn.Sequential(
        nn.Flatten(),

        nn.Linear(256*256*1,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  1)
).cuda()

print(model)
```

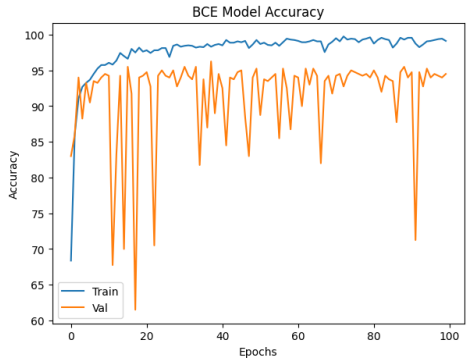
圖(a)BCE training 中使用的 model

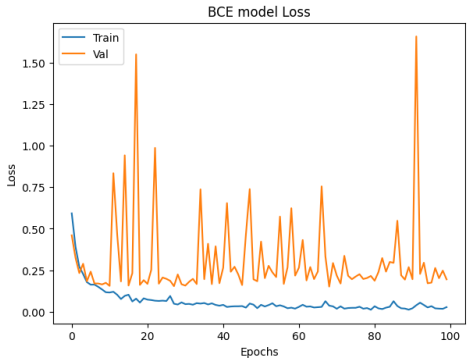
```
ce_model = nn.Sequential(
        nn.Flatten(),

        nn.Linear(256*256*1,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  2)
).cuda()

print(ce_model)
```

圖(b)CE training 中使用的 model

- BCE

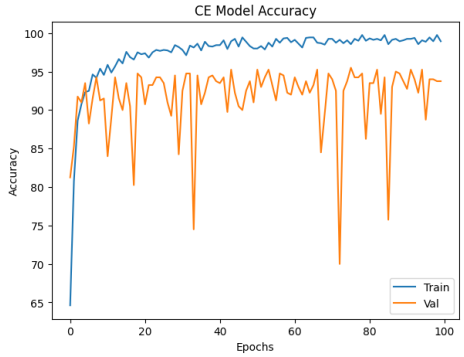
於 training 的時候使用 BCE 後，其 accuracy 以及 loss 如下(進行 100 個 epoch)

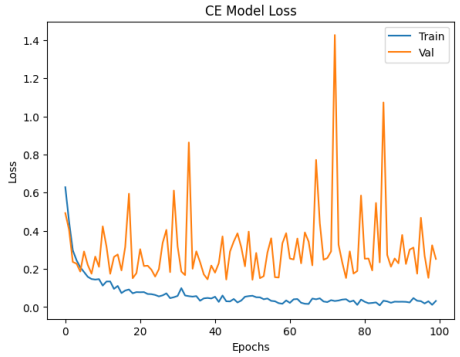




- CE

於 training 的時候使用 CE 後，其 accuracy 以及 loss 如下



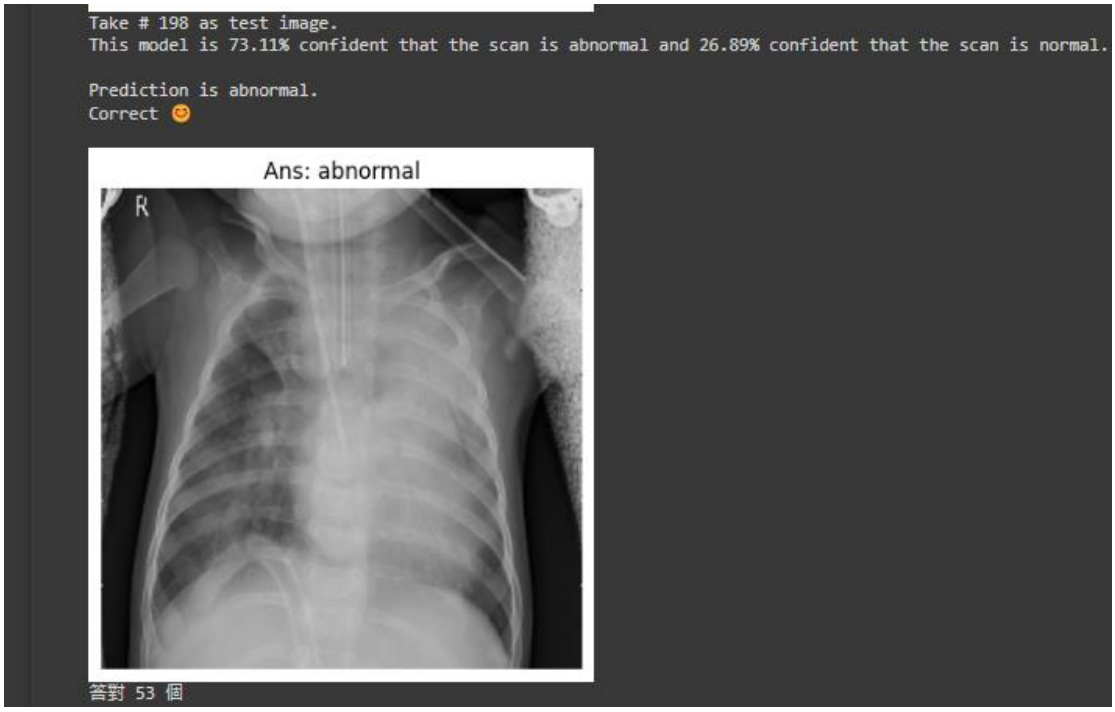


BCE 和 CE 兩者用來 training 後的 loss 和 accuracy 看起來基本上差不多，震盪都很大，只不過整體看下來，可能 CE 整體的 loss 較低，accuracy 好一點吧。

2. testing phases BCE V.S. CE

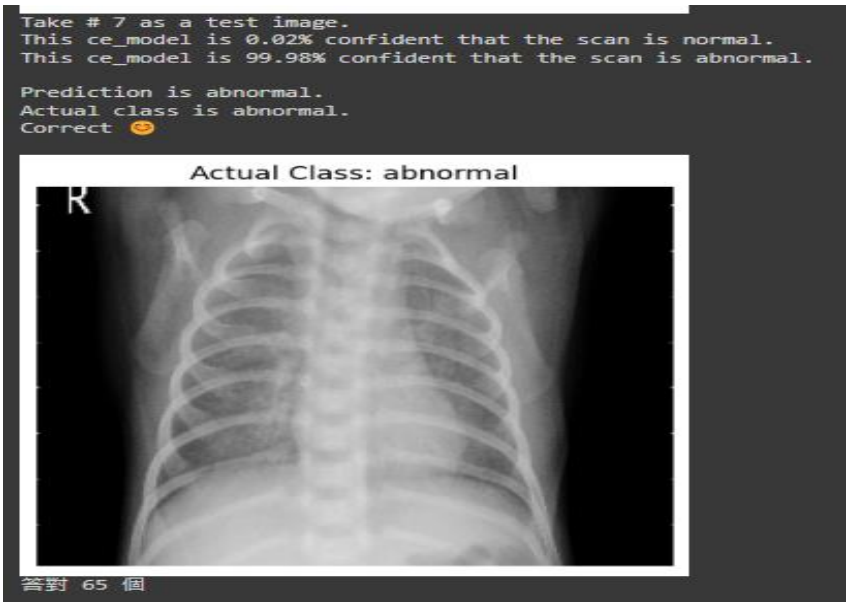
- **BCE**

  Load BCE training 時使用 weight 後，對 100 比 test data 進行測試後之準確度為下圖，約為 100 筆資料答對 53 個，**正確率為 53%。**(詳細在 hw2_report 檔案裡面)



- **CE**

  Load CE training 時使用 weight 後，對 100 比 test data 進行測試後之準確度為下圖，約為 100 筆資料答對 65 個，**正確率為 65%。**(詳細在 hw2_report 檔案裡面)



總結:　進行 test data 測試 100 個 data 後，整體來說用 CE 弄出來的 model 最終預測正確率較高，不過兩者的正確率都不高，所以實際上其 variance 很大，有可能在 training 的時候不小心 overfitting training data 了。

2.2 Task B: Performance between Different Hyperparameters (40 pts)

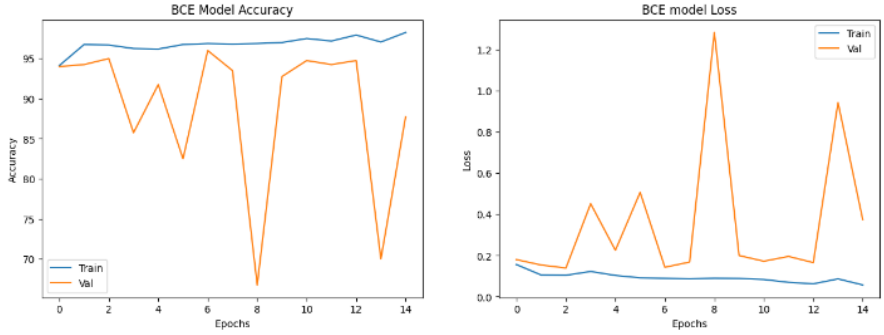
Choose two hyperparameters and experiment with three distinct values for each. Train and test your experiment with the provided chest X-ray dataset. You need to indicate what hyperparameters you choose and which values you use in your report

1. **changing epoch**

   我使用原本的 BCE model 並改變其 training 時的 epoch 分別為 15、30、100，而 test 的時候則用 100 個 data 測試並統計最後預測正確的數量。其餘參數皆相同不改變。

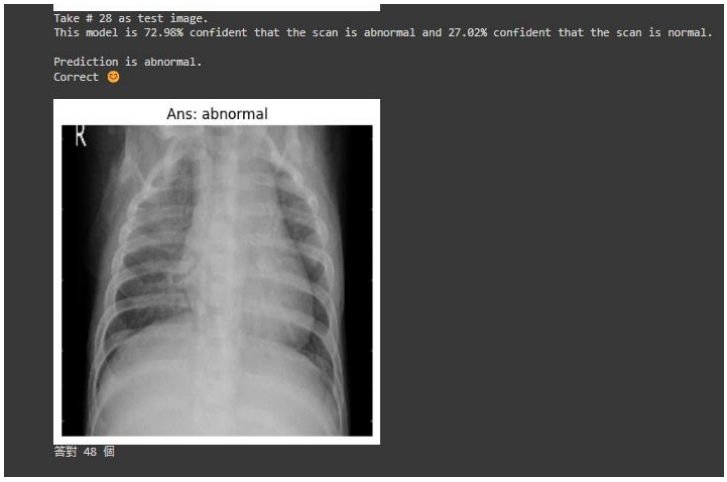
   - 15 epoch

     圖(a)為 15 epoch training 後的 accuracy 與 loss



圖(a)

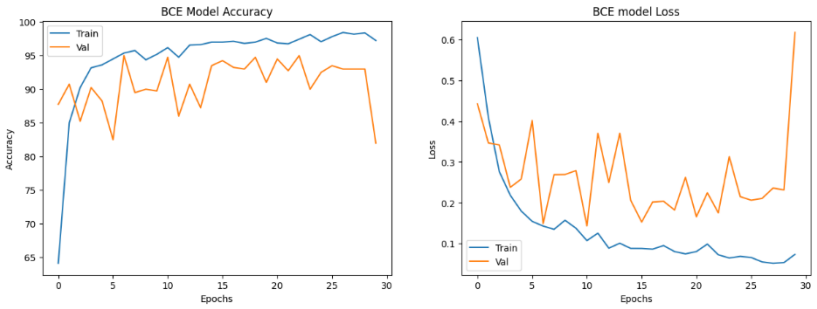
圖(b)為 15 epoch 最後一個 test 的機率以及 100 個測試中答對幾個。(72.98%且答對 48 個)



圖(b)

   - 30 epoch

     圖(c)為 30 epoch training 後的 accuracy 與 loss



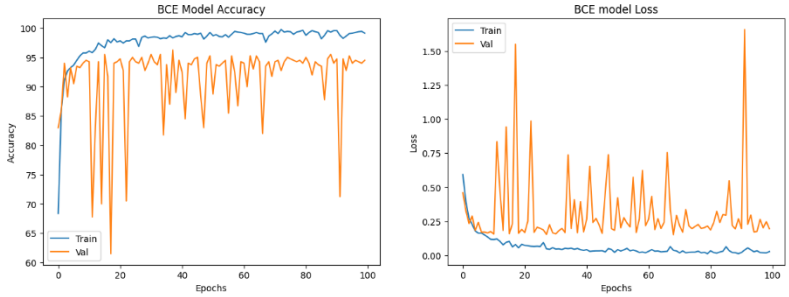
圖(c)

圖(d)為 30 epoch 最後一個 test 的機率以及 100 個測試中答對幾個。
(73.03%且答對 52 個)

圖(d)

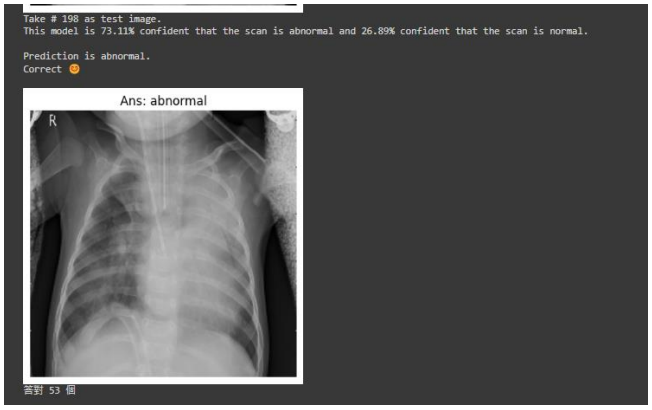
- 100 epoch

圖(e)為 100 epoch training 後的 accuracy 與 loss



圖(e)

圖(f)為 30 epoch 最後一個 test 的機率以及 100 個測試中答對幾個。
(73.11%且答對 53 個)

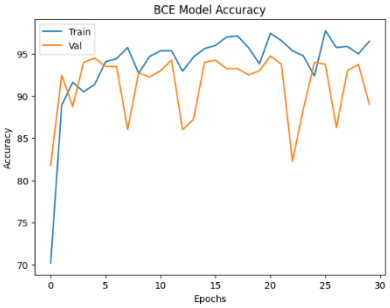


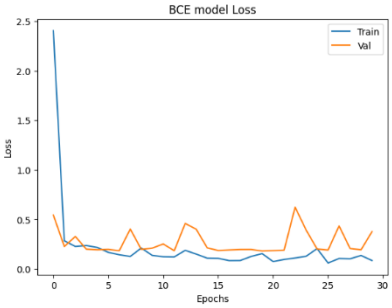
圖(e)

2. **changing layer numbers**

分別用 1、2、3 層的 model 進行 training 與 100 個 test data 測試(epoch 固定為 30 個)

➢ **1 layer**





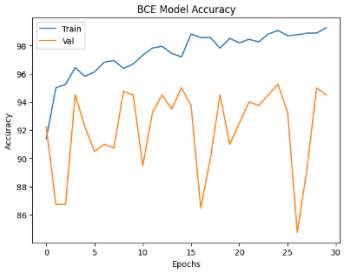

```
import torch.nn as nn
#  torch.nn.functional
#  Model definition
model = nn.Sequential(
        nn.Flatten(),
        nn.Linear(256*256*1,  256),
        nn.ReLU(),
        nn.Linear(256,  1)#現在用的是
).cuda()#cuda放進GPU裡面

print(model)
```

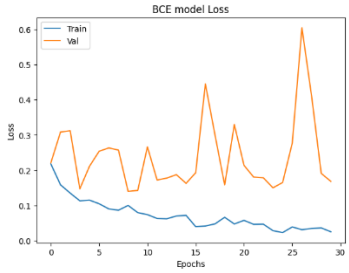
✧ 1 layer 100 epoch training 後的 accuracy 與 loss　　1 layer model

✧ 最終 100 個 test data 答對了 55 個

```
答對 55 個
```

➢ **2 layer**





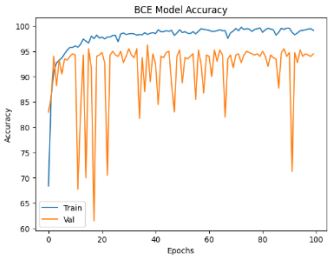

```
model = nn.Sequential(
        nn.Flatten(),

        nn.Linear(256*256*1,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

#       nn.Linear(64,  64),
#       nn.BatchNorm1d(64),
#       nn.ReLU(),
#       nn.Dropout(0.5),

        nn.Linear(64,  1)
).cuda()
```

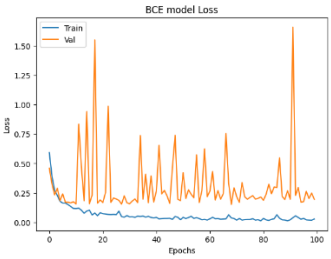
✧ 2 layer 100 epoch training 後的 accuracy 與 loss　　2 layer model

✧ 最終 100 個 test data 答對了 50 個

```
答對 50 個
```

➢ **3 layer**






```
model = nn.Sequential(
        nn.Flatten(),

        nn.Linear(256*256*1,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  64),
        nn.BatchNorm1d(64),
        nn.ReLU(),
        nn.Dropout(0.5),

        nn.Linear(64,  1)
).cuda()

print(model)
```

✧ 2 layer 100 epoch training 後的 accuracy 與 loss　　2 layer model

✧ 最終 100 個 test data 答對了 55 個

```
答對 53 個
```

總結:

1. 調整 epoch，似乎準確率有隨著 epoch 增加而變大，不過實際上的數值並沒有差到很多，搞不好多測幾次的結果會不同，或許 epoch 數對這個 model 的影響並沒有那麼大。
2. 調整 layer，目前將 layer 增加並沒有甚麼明顯趨勢產生，甚至這樣觀察起來 layer 層數少一點，似乎震盪的還比較小，accuracy 還比較大。