HW3 109081009 李承叡

➢ Task A: Reduce Overfitting

若使用原本的 model(都使用 30 個 epoch)最後的結果會如下圖，會出現 overfitting 的狀況，training 的 accuracy 很高(100%左右)但 validation 的 accuracy 就低很多

Epoch 30

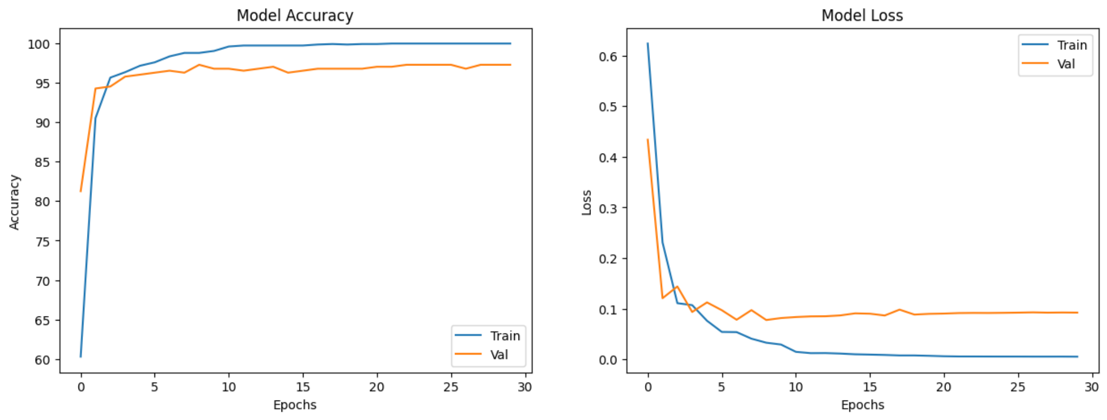
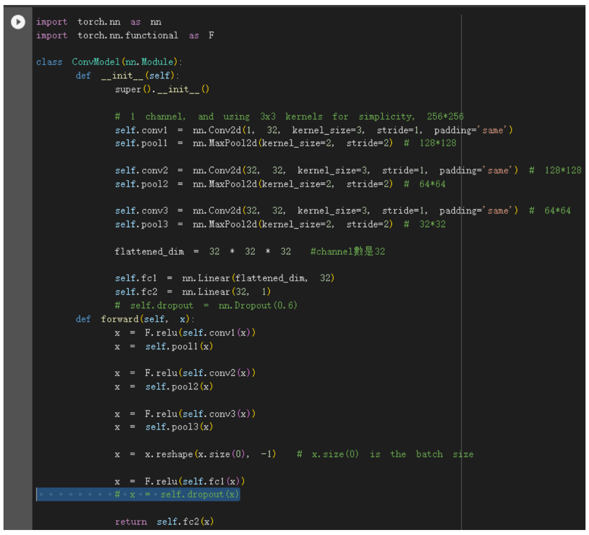


圖 1, 原本 model 的 accuracy 和 loss



```
import torch.nn as nn
import torch.nn.functional as F

class ConvModel(nn.Module):
    def __init__(self):
        super().__init__()

        # 1 channel, and using 3x3 kernels for simplicity, 256*256
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding='same')
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2) # 128*128

        self.conv2 = nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same') # 128*128
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2) # 64*64

        self.conv3 = nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same') # 64*64
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2) # 32*32

        flattened_dim = 32 * 32 * 32   #channel數是32

        self.fc1 = nn.Linear(flattened_dim, 32)
        self.fc2 = nn.Linear(32, 1)
        # self.dropout = nn.Dropout(0.6)
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool1(x)

        x = F.relu(self.conv2(x))
        x = self.pool2(x)

        x = F.relu(self.conv3(x))
        x = self.pool3(x)

        x = x.reshape(x.size(0), -1)   # x.size(0) is the batch size

        x = F.relu(self.fc1(x))
        # x = self.dropout(x)

        return self.fc2(x)
```

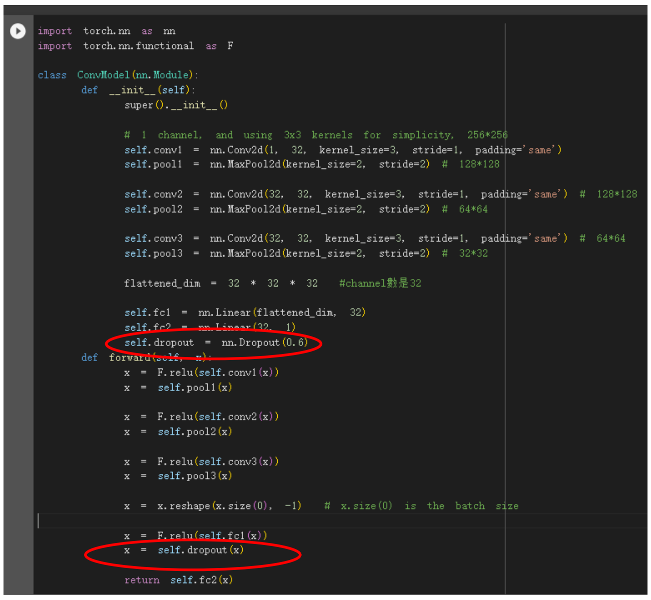
圖 2, 原本 model 的架構

所以為了防止他繼續 overfitting，我選擇使用了 dropout 的方式，因為感覺此 model 問題不大，accuracy 也蠻高的，現在的問題只有 overfitting，所以想說就直接藉由減少他對 training data 的 fitting 程度就可以了。

我增加了兩行程式碼

```
self.dropout = nn.Dropout(0.6)
x = self.dropout(x)
```

讓 model 把部分 train 出來的 neuron 丟掉，防止他過度學習 train data 的特色，而在這裡我讓 model 把 6 成的 neuron 丟掉。



```python
import torch.nn as nn
import torch.nn.functional as F

class ConvModel(nn.Module):
    def __init__(self):
        super().__init__()

        # 1 channel, and using 3x3 kernels for simplicity, 256*256
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding='same')
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2) # 128*128

        self.conv2 = nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same') # 128*128
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2) # 64*64

        self.conv3 = nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same') # 64*64
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2) # 32*32

        flattened_dim = 32 * 32 * 32   #channel數是32

        self.fc1 = nn.Linear(flattened_dim, 32)
        self.fc2 = nn.Linear(32, 1)
        self.dropout = nn.Dropout(0.6)
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool1(x)

        x = F.relu(self.conv2(x))
        x = self.pool2(x)

        x = F.relu(self.conv3(x))
        x = self.pool3(x)

        x = x.reshape(x.size(0), -1)   # x.size(0) is the batch size

        x = F.relu(self.fc1(x))
        x = self.dropout(x)

        return self.fc2(x)
```

圖 3, 使用 dropout 的 model

最終結果如下，成功解決了 overfitting 的狀況，train 和 validation 的 accuracy 已經幾乎一樣了。

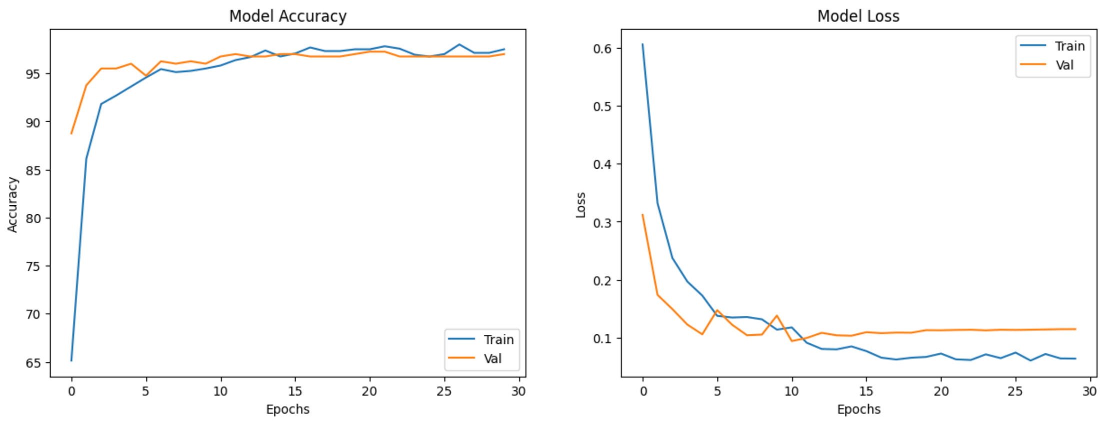
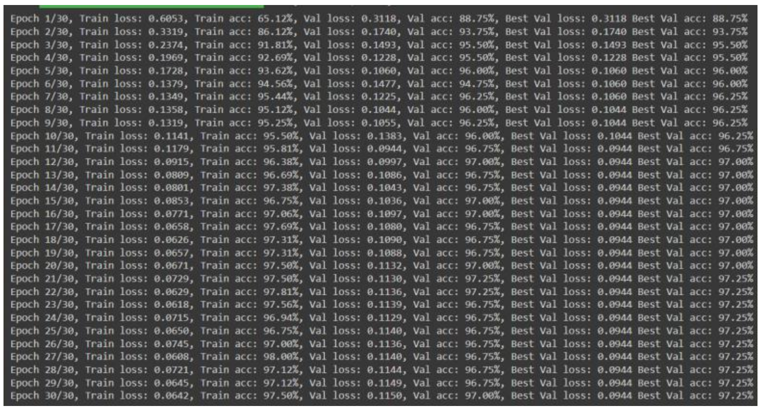


圖 3, 使用 drop out 的後的 accuracy 和 loss



```
Epoch 1/30, Train loss: 0.6053, Train acc: 65.12%, Val loss: 0.3118, Val acc: 88.75%, Best Val loss: 0.3118 Best Val acc: 88.75%
Epoch 2/30, Train loss: 0.3319, Train acc: 86.12%, Val loss: 0.1740, Val acc: 93.75%, Best Val loss: 0.1740 Best Val acc: 93.75%
Epoch 3/30, Train loss: 0.2374, Train acc: 91.81%, Val loss: 0.1493, Val acc: 95.50%, Best Val loss: 0.1493 Best Val acc: 95.50%
Epoch 4/30, Train loss: 0.1969, Train acc: 92.69%, Val loss: 0.1228, Val acc: 95.50%, Best Val loss: 0.1228 Best Val acc: 95.50%
Epoch 5/30, Train loss: 0.1728, Train acc: 93.62%, Val loss: 0.1060, Val acc: 96.00%, Best Val loss: 0.1060 Best Val acc: 96.00%
Epoch 6/30, Train loss: 0.1379, Train acc: 94.56%, Val loss: 0.1477, Val acc: 94.75%, Best Val loss: 0.1060 Best Val acc: 96.00%
Epoch 7/30, Train loss: 0.1349, Train acc: 95.44%, Val loss: 0.1225, Val acc: 96.25%, Best Val loss: 0.1060 Best Val acc: 96.25%
Epoch 8/30, Train loss: 0.1358, Train acc: 95.12%, Val loss: 0.1044, Val acc: 96.00%, Best Val loss: 0.1044 Best Val acc: 96.25%
Epoch 9/30, Train loss: 0.1319, Train acc: 95.25%, Val loss: 0.1055, Val acc: 96.25%, Best Val loss: 0.1044 Best Val acc: 96.25%
Epoch 10/30, Train loss: 0.1141, Train acc: 95.50%, Val loss: 0.1383, Val acc: 96.00%, Best Val loss: 0.1044 Best Val acc: 96.25%
Epoch 11/30, Train loss: 0.1179, Train acc: 95.81%, Val loss: 0.0944, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 96.75%
Epoch 12/30, Train loss: 0.0915, Train acc: 96.38%, Val loss: 0.0997, Val acc: 97.00%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 13/30, Train loss: 0.0809, Train acc: 96.69%, Val loss: 0.1086, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 14/30, Train loss: 0.0801, Train acc: 97.38%, Val loss: 0.1043, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 15/30, Train loss: 0.0853, Train acc: 96.75%, Val loss: 0.1036, Val acc: 97.00%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 16/30, Train loss: 0.0771, Train acc: 97.06%, Val loss: 0.1097, Val acc: 97.00%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 17/30, Train loss: 0.0658, Train acc: 97.69%, Val loss: 0.1080, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 18/30, Train loss: 0.0626, Train acc: 97.31%, Val loss: 0.1090, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 19/30, Train loss: 0.0657, Train acc: 97.31%, Val loss: 0.1088, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 20/30, Train loss: 0.0671, Train acc: 97.50%, Val loss: 0.1132, Val acc: 97.00%, Best Val loss: 0.0944 Best Val acc: 97.00%
Epoch 21/30, Train loss: 0.0729, Train acc: 97.50%, Val loss: 0.1130, Val acc: 97.25%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 22/30, Train loss: 0.0629, Train acc: 97.81%, Val loss: 0.1136, Val acc: 97.25%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 23/30, Train loss: 0.0618, Train acc: 97.56%, Val loss: 0.1139, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 24/30, Train loss: 0.0715, Train acc: 96.94%, Val loss: 0.1129, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 25/30, Train loss: 0.0650, Train acc: 96.75%, Val loss: 0.1140, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 26/30, Train loss: 0.0745, Train acc: 97.00%, Val loss: 0.1136, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 27/30, Train loss: 0.0608, Train acc: 98.00%, Val loss: 0.1140, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 28/30, Train loss: 0.0721, Train acc: 97.12%, Val loss: 0.1144, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 29/30, Train loss: 0.0645, Train acc: 97.12%, Val loss: 0.1149, Val acc: 96.75%, Best Val loss: 0.0944 Best Val acc: 97.25%
Epoch 30/30, Train loss: 0.0642, Train acc: 97.50%, Val loss: 0.1150, Val acc: 97.00%, Best Val loss: 0.0944 Best Val acc: 97.25%
```

圖 4, 更細節的測試結果

● Discussion

總而言之在我使用了 Dropout 把一些 neuron 丟掉後，出現了以下的情況

1. **Accuracy:** 實施 Dropout 後，我 train 和 validation 的 accuracy 變得幾乎相同了，也就代表這次比較沒有出現 overfitting 的現象。

2. Loss : 在 loss 的部分反而沒有那麼明顯，雖然似乎 loss 有比原本還低一點，但實際上 loss 也一樣很快就沒辦法繼續下降了。可能 dropout 沒辦法用來降低 loss 吧。

3. Generalization: 從 accuracy 來看的話，理論上此 model 變得更 generalization，因為我把 model 改成這樣後，train 和 validation 的 accuracy 基本上是很接近一模一樣

的，都約 97%左右，換句話說如果用來 test 其他 data 的話因為比較不會 overfitting ，所以拿還 test 理論上會有較好的表現。

➢ Task B: Performance Comparison between CNN and ANN

Discussion

我分別使用了 class LinearModel and class ConvModel
最終 CNN 和 ANN 的差別如下

	feature extraction capabilities	training speed	model performance
CNN	較好(accuracy 較高)	2.35s/it	非常好
ANN	較差(accuracy 較低)	2.65s/it	不太好

1. **feature extraction capabilities：**

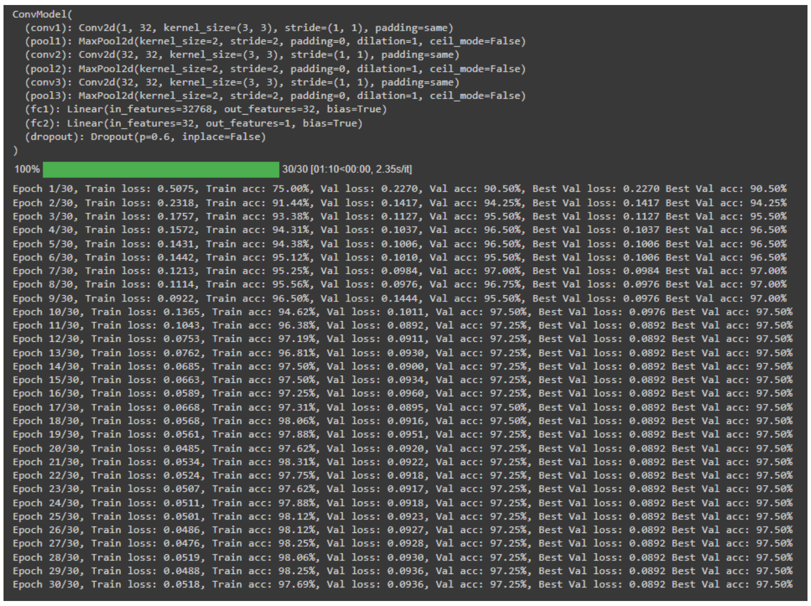
從結果中可以看出，CNN 的 accuracy 最終到了 97.25%，loss 則下降到 0.1052。這代表 CNN feature extraction capabilities 很好。
相比之下，ANN 模型的性能不如 CNN。ANN 的 accuracy 最終到了 94.50%，loss 則最終下降到 0.1694。雖然 ANN 也在一定程度上提高了性能，但它的性能仍然不及 CNN。代表 ANN 在圖像分類任務中的 feature extraction capabilities 不如 CNN。
此外本來 CNN model 的架構本來就是藉由提取 feature 進行簡化與優化的，所以 **feature extraction capabilities** 一定是 CNN 比較好。

2. **training speed：**

CNN 在每個周期中需要大約 2.35 秒，而 ANN 在每個周期中需要大約 2.73 秒。這代表出 CNN 在訓練速度上略微快於 ANN。這是因為 CNN 的卷積操作可以進行高效的並行計算，而 ANN 的全連接層需要更多的計算資源。

3. **model performance：**

根據最終的 accuracy 和 loss，可以看出 CNN 模型不管在 training 速度還是準確度和 loss 上都比 ANN 模型還強。此外從圖 6 和圖 8 的表較中可以看出，甚至在 ANN 中還有點出現 overfitting 的現象故 CNN 的表現比 ANN 好非常多。



```
ConvModel(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=same)
  (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=32768, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=1, bias=True)
  (dropout): Dropout(p=0.6, inplace=False)
)
100% 30/30 [01:10<00:00, 2.35s/it]
Epoch 1/30, Train loss: 0.5075, Train acc: 75.00%, Val loss: 0.2270, Val acc: 90.50%, Best Val loss: 0.2270 Best Val acc: 90.50%
Epoch 2/30, Train loss: 0.2318, Train acc: 91.44%, Val loss: 0.1417, Val acc: 94.25%, Best Val loss: 0.1417 Best Val acc: 94.25%
Epoch 3/30, Train loss: 0.1757, Train acc: 93.38%, Val loss: 0.1127, Val acc: 95.50%, Best Val loss: 0.1127 Best Val acc: 95.50%
Epoch 4/30, Train loss: 0.1572, Train acc: 94.31%, Val loss: 0.1037, Val acc: 96.50%, Best Val loss: 0.1037 Best Val acc: 96.50%
Epoch 5/30, Train loss: 0.1431, Train acc: 94.38%, Val loss: 0.1006, Val acc: 96.50%, Best Val loss: 0.1006 Best Val acc: 96.50%
Epoch 6/30, Train loss: 0.1442, Train acc: 95.12%, Val loss: 0.1010, Val acc: 95.50%, Best Val loss: 0.1006 Best Val acc: 96.50%
Epoch 7/30, Train loss: 0.1213, Train acc: 95.25%, Val loss: 0.0984, Val acc: 97.00%, Best Val loss: 0.0984 Best Val acc: 97.00%
Epoch 8/30, Train loss: 0.1114, Train acc: 95.56%, Val loss: 0.0976, Val acc: 96.75%, Best Val loss: 0.0976 Best Val acc: 97.00%
Epoch 9/30, Train loss: 0.0922, Train acc: 96.50%, Val loss: 0.1444, Val acc: 95.50%, Best Val loss: 0.0976 Best Val acc: 97.00%
Epoch 10/30, Train loss: 0.1365, Train acc: 94.62%, Val loss: 0.1011, Val acc: 97.50%, Best Val loss: 0.0976 Best Val acc: 97.50%
Epoch 11/30, Train loss: 0.1043, Train acc: 96.38%, Val loss: 0.0892, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 12/30, Train loss: 0.0753, Train acc: 97.19%, Val loss: 0.0911, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 13/30, Train loss: 0.0762, Train acc: 96.81%, Val loss: 0.0930, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 14/30, Train loss: 0.0685, Train acc: 97.50%, Val loss: 0.0900, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 15/30, Train loss: 0.0663, Train acc: 97.50%, Val loss: 0.0934, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 16/30, Train loss: 0.0589, Train acc: 97.25%, Val loss: 0.0960, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 17/30, Train loss: 0.0668, Train acc: 97.31%, Val loss: 0.0895, Val acc: 97.50%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 18/30, Train loss: 0.0568, Train acc: 98.06%, Val loss: 0.0916, Val acc: 97.50%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 19/30, Train loss: 0.0561, Train acc: 97.88%, Val loss: 0.0951, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 20/30, Train loss: 0.0485, Train acc: 97.62%, Val loss: 0.0920, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 21/30, Train loss: 0.0534, Train acc: 98.31%, Val loss: 0.0922, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 22/30, Train loss: 0.0524, Train acc: 97.75%, Val loss: 0.0918, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 23/30, Train loss: 0.0507, Train acc: 97.62%, Val loss: 0.0917, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 24/30, Train loss: 0.0511, Train acc: 97.88%, Val loss: 0.0918, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 25/30, Train loss: 0.0501, Train acc: 98.12%, Val loss: 0.0923, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 26/30, Train loss: 0.0486, Train acc: 98.12%, Val loss: 0.0927, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 27/30, Train loss: 0.0476, Train acc: 98.25%, Val loss: 0.0928, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 28/30, Train loss: 0.0519, Train acc: 98.06%, Val loss: 0.0930, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 29/30, Train loss: 0.0488, Train acc: 98.25%, Val loss: 0.0936, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
Epoch 30/30, Train loss: 0.0518, Train acc: 97.69%, Val loss: 0.0936, Val acc: 97.25%, Best Val loss: 0.0892 Best Val acc: 97.50%
```

圖 5, CNN 花的時間、loss、accuracy 等等細項

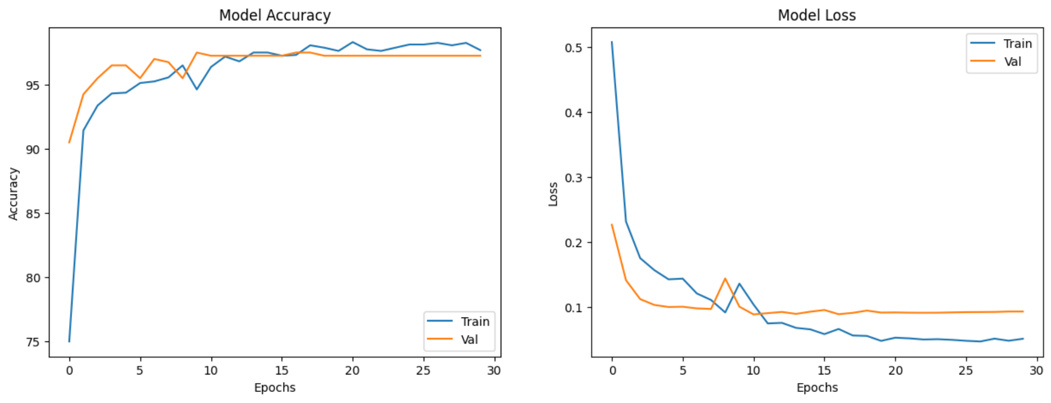
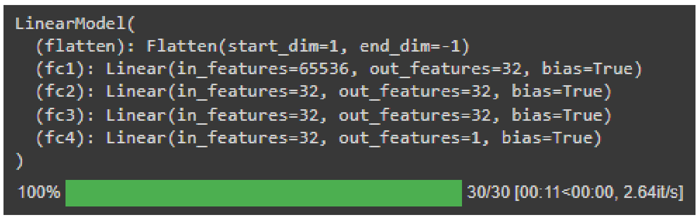
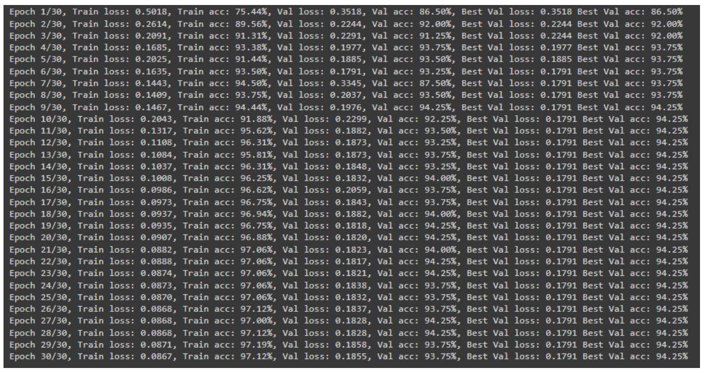


圖 6, CNN 的 accuracy 和 loss



```
LinearModel(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=65536, out_features=32, bias=True)
  (fc2): Linear(in_features=32, out_features=32, bias=True)
  (fc3): Linear(in_features=32, out_features=32, bias=True)
  (fc4): Linear(in_features=32, out_features=1, bias=True)
)
100%  30/30 [00:11<00:00, 2.64it/s]
```



```
Epoch 1/30, Train loss: 0.5018, Train acc: 75.44%, Val loss: 0.3518, Val acc: 86.50%, Best Val loss: 0.3518 Best Val acc: 86.50%
Epoch 2/30, Train loss: 0.2614, Train acc: 89.56%, Val loss: 0.2244, Val acc: 92.00%, Best Val loss: 0.2244 Best Val acc: 92.00%
Epoch 3/30, Train loss: 0.2091, Train acc: 91.31%, Val loss: 0.2291, Val acc: 91.25%, Best Val loss: 0.2244 Best Val acc: 92.00%
Epoch 4/30, Train loss: 0.1685, Train acc: 93.38%, Val loss: 0.1977, Val acc: 93.75%, Best Val loss: 0.1977 Best Val acc: 93.75%
Epoch 5/30, Train loss: 0.2025, Train acc: 91.44%, Val loss: 0.1885, Val acc: 93.50%, Best Val loss: 0.1885 Best Val acc: 93.75%
Epoch 6/30, Train loss: 0.1635, Train acc: 93.50%, Val loss: 0.1791, Val acc: 93.25%, Best Val loss: 0.1791 Best Val acc: 93.75%
Epoch 7/30, Train loss: 0.1443, Train acc: 94.50%, Val loss: 0.3345, Val acc: 87.50%, Best Val loss: 0.1791 Best Val acc: 93.75%
Epoch 8/30, Train loss: 0.1409, Train acc: 93.75%, Val loss: 0.2037, Val acc: 93.50%, Best Val loss: 0.1791 Best Val acc: 93.75%
Epoch 9/30, Train loss: 0.1467, Train acc: 94.44%, Val loss: 0.1976, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 10/30, Train loss: 0.2043, Train acc: 91.88%, Val loss: 0.2299, Val acc: 92.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 11/30, Train loss: 0.1317, Train acc: 95.62%, Val loss: 0.1882, Val acc: 93.50%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 12/30, Train loss: 0.1108, Train acc: 96.31%, Val loss: 0.1873, Val acc: 93.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 13/30, Train loss: 0.1084, Train acc: 95.81%, Val loss: 0.1873, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 14/30, Train loss: 0.1037, Train acc: 96.31%, Val loss: 0.1848, Val acc: 93.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 15/30, Train loss: 0.1008, Train acc: 96.25%, Val loss: 0.1832, Val acc: 94.00%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 16/30, Train loss: 0.0986, Train acc: 96.62%, Val loss: 0.2059, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 17/30, Train loss: 0.0973, Train acc: 96.75%, Val loss: 0.1843, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 18/30, Train loss: 0.0937, Train acc: 96.94%, Val loss: 0.1882, Val acc: 94.00%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 19/30, Train loss: 0.0935, Train acc: 96.75%, Val loss: 0.1818, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 20/30, Train loss: 0.0907, Train acc: 96.88%, Val loss: 0.1820, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 21/30, Train loss: 0.0882, Train acc: 97.06%, Val loss: 0.1823, Val acc: 94.00%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 22/30, Train loss: 0.0888, Train acc: 97.06%, Val loss: 0.1817, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 23/30, Train loss: 0.0874, Train acc: 97.06%, Val loss: 0.1821, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 24/30, Train loss: 0.0873, Train acc: 97.06%, Val loss: 0.1838, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 25/30, Train loss: 0.0870, Train acc: 97.06%, Val loss: 0.1832, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 26/30, Train loss: 0.0868, Train acc: 97.12%, Val loss: 0.1837, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 27/30, Train loss: 0.0868, Train acc: 97.00%, Val loss: 0.1828, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 28/30, Train loss: 0.0868, Train acc: 97.12%, Val loss: 0.1828, Val acc: 94.25%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 29/30, Train loss: 0.0871, Train acc: 97.19%, Val loss: 0.1858, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
Epoch 30/30, Train loss: 0.0867, Train acc: 97.12%, Val loss: 0.1855, Val acc: 93.75%, Best Val loss: 0.1791 Best Val acc: 94.25%
```

圖 7, ANN 花的時間、loss、accuracy 等等細項

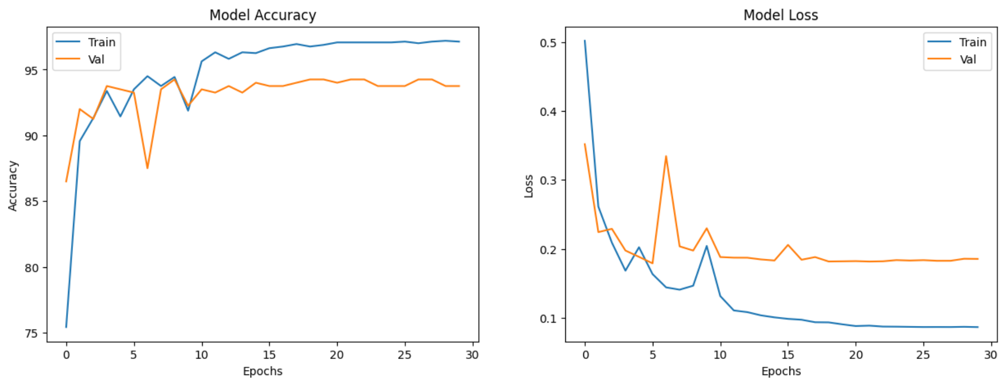
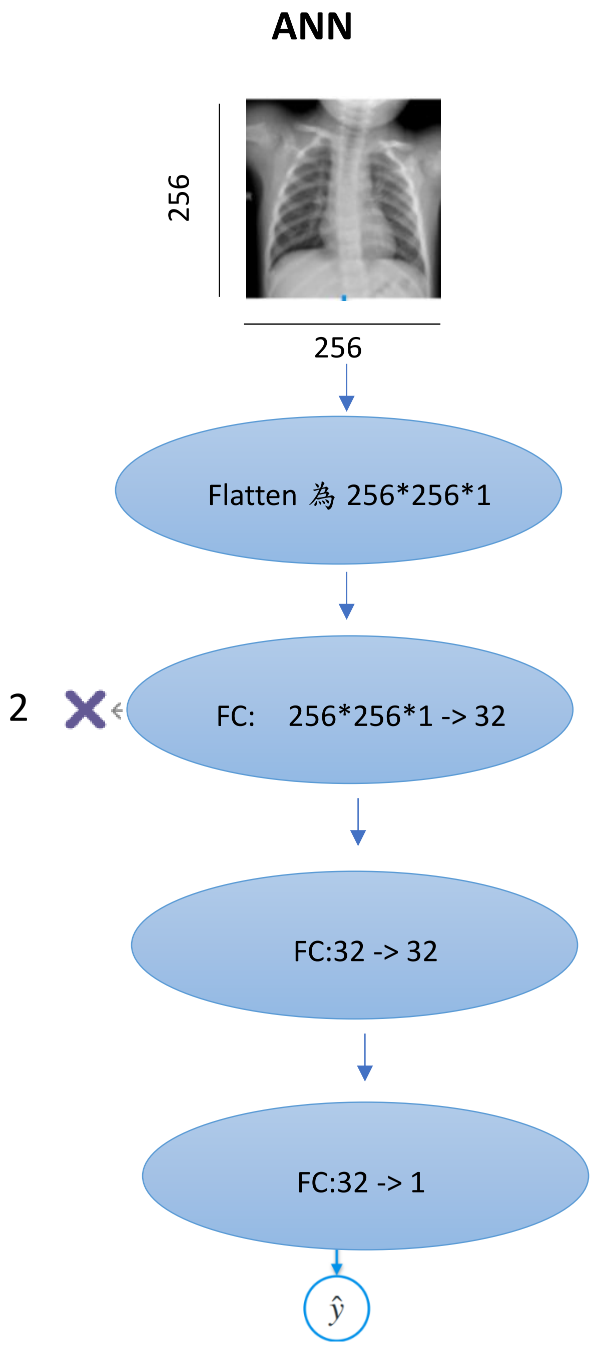
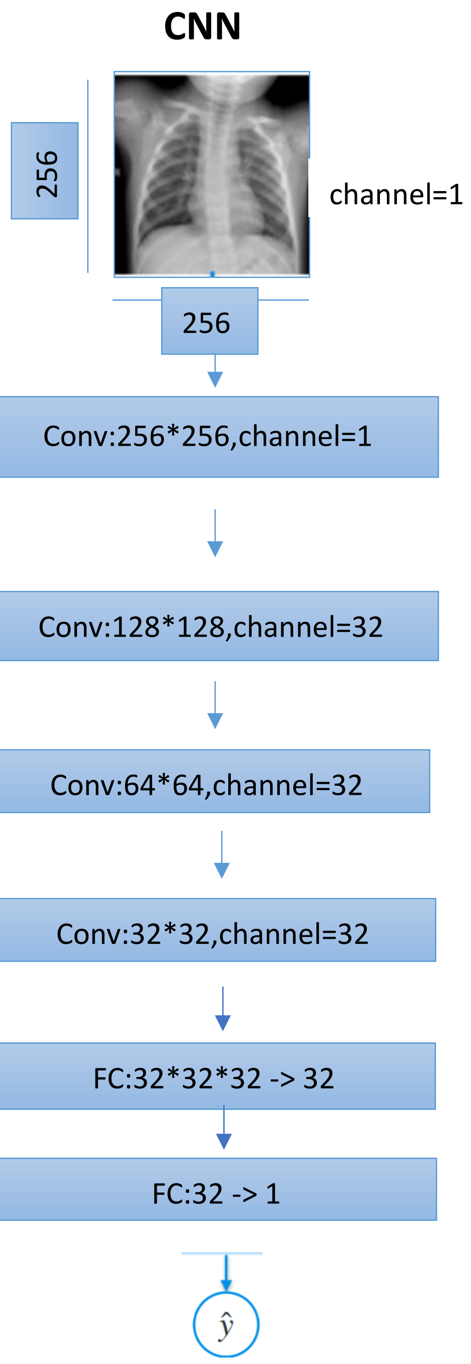


圖 8, ANN 的 accuracy 和 loss

Architecture Description





Task C: Global Average Pooling in CNNs

➢ **Explanation:**

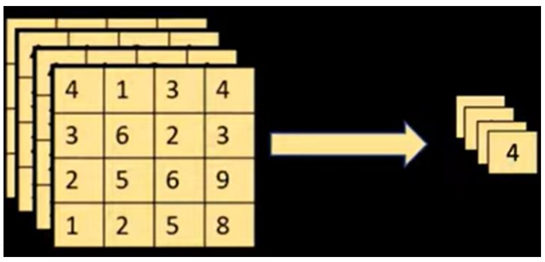
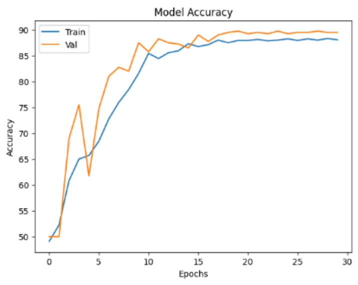


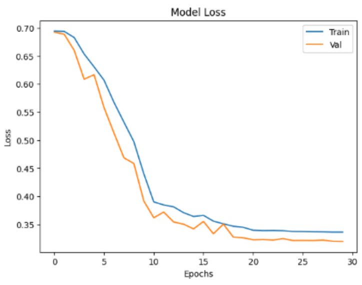
圖 9, GAP 示意圖, 每個 channel 取平均

Global Average Pooling（GAP）是一種在卷積神經網絡（CNN）中用於特徵提取的技術。它的主要優勢在於它可以自動消除對輸入特徵數目的需求，而不需要手動確定特徵數目。

GAP 的運作方式如下：
在 CNN 的最後一個卷積層時，GAP 會對每個 channel 的特徵圖進行平均化，讓每個通道的特徵都被壓縮成一個單一的數值。最終這些平均值的集合形成了一個特徵向量的輸出，該向量的每個元素對應於一個特徵通道。
這樣，GAP 產生了一個固定大小的特徵向量，該向量的大小獨立於輸入圖像的大小。因此最終並不需要使用到 flatten。直接輸出給 fully connected layer 即可。

➢ **Increase Performance:**





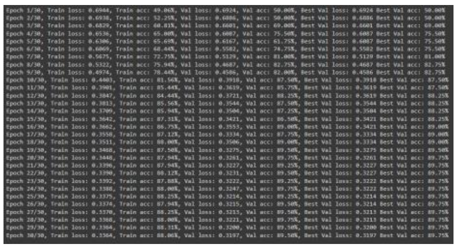
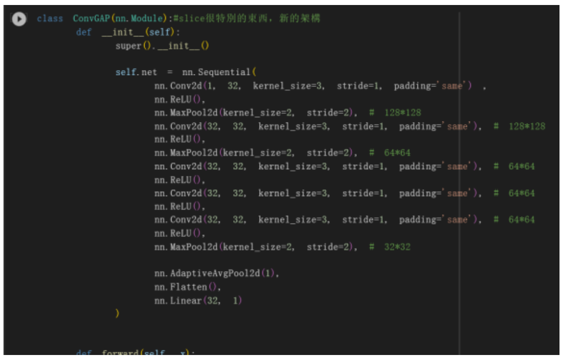

```
Epoch 1/30, Train loss: 0.6944, Train acc: 49.06%, Val loss: 0.6924, Val acc: 50.00%, Best Val loss: 0.6924 Best Val acc: 50.00%
Epoch 2/30, Train loss: 0.6938, Train acc: 52.25%, Val loss: 0.6886, Val acc: 50.00%, Best Val loss: 0.6886 Best Val acc: 50.00%
Epoch 3/30, Train loss: 0.6829, Train acc: 60.81%, Val loss: 0.6601, Val acc: 69.00%, Best Val loss: 0.6601 Best Val acc: 69.00%
Epoch 4/30, Train loss: 0.6536, Train acc: 65.00%, Val loss: 0.6087, Val acc: 75.50%, Best Val loss: 0.6087 Best Val acc: 75.50%
Epoch 5/30, Train loss: 0.6306, Train acc: 65.69%, Val loss: 0.6167, Val acc: 61.75%, Best Val loss: 0.6087 Best Val acc: 75.50%
Epoch 6/30, Train loss: 0.6069, Train acc: 68.44%, Val loss: 0.5582, Val acc: 74.75%, Best Val loss: 0.5582 Best Val acc: 75.50%
Epoch 7/30, Train loss: 0.5675, Train acc: 72.75%, Val loss: 0.5129, Val acc: 81.00%, Best Val loss: 0.5129 Best Val acc: 81.00%
Epoch 8/30, Train loss: 0.5322, Train acc: 75.94%, Val loss: 0.4687, Val acc: 82.75%, Best Val loss: 0.4687 Best Val acc: 82.75%
Epoch 9/30, Train loss: 0.4974, Train acc: 78.44%, Val loss: 0.4586, Val acc: 82.00%, Best Val loss: 0.4586 Best Val acc: 82.75%
Epoch 10/30, Train loss: 0.4403, Train acc: 81.56%, Val loss: 0.3918, Val acc: 87.50%, Best Val loss: 0.3918 Best Val acc: 87.50%
Epoch 11/30, Train loss: 0.3901, Train acc: 85.44%, Val loss: 0.3619, Val acc: 85.75%, Best Val loss: 0.3619 Best Val acc: 87.50%
Epoch 12/30, Train loss: 0.3847, Train acc: 84.44%, Val loss: 0.3721, Val acc: 88.25%, Best Val loss: 0.3619 Best Val acc: 88.25%
Epoch 13/30, Train loss: 0.3813, Train acc: 85.56%, Val loss: 0.3544, Val acc: 87.50%, Best Val loss: 0.3544 Best Val acc: 88.25%
Epoch 14/30, Train loss: 0.3709, Train acc: 85.94%, Val loss: 0.3504, Val acc: 87.25%, Best Val loss: 0.3504 Best Val acc: 88.25%
Epoch 15/30, Train loss: 0.3642, Train acc: 87.31%, Val loss: 0.3421, Val acc: 86.50%, Best Val loss: 0.3421 Best Val acc: 88.25%
Epoch 16/30, Train loss: 0.3662, Train acc: 86.75%, Val loss: 0.3553, Val acc: 89.00%, Best Val loss: 0.3421 Best Val acc: 89.00%
Epoch 17/30, Train loss: 0.3558, Train acc: 87.12%, Val loss: 0.3334, Val acc: 87.75%, Best Val loss: 0.3334 Best Val acc: 89.00%
Epoch 18/30, Train loss: 0.3511, Train acc: 88.00%, Val loss: 0.3506, Val acc: 89.00%, Best Val loss: 0.3334 Best Val acc: 89.00%
Epoch 19/30, Train loss: 0.3468, Train acc: 87.50%, Val loss: 0.3275, Val acc: 89.50%, Best Val loss: 0.3275 Best Val acc: 89.50%
Epoch 20/30, Train loss: 0.3448, Train acc: 87.94%, Val loss: 0.3261, Val acc: 89.75%, Best Val loss: 0.3261 Best Val acc: 89.75%
Epoch 21/30, Train loss: 0.3396, Train acc: 87.94%, Val loss: 0.3227, Val acc: 89.25%, Best Val loss: 0.3227 Best Val acc: 89.75%
Epoch 22/30, Train loss: 0.3390, Train acc: 88.12%, Val loss: 0.3231, Val acc: 89.50%, Best Val loss: 0.3227 Best Val acc: 89.75%
Epoch 23/30, Train loss: 0.3392, Train acc: 87.88%, Val loss: 0.3222, Val acc: 89.25%, Best Val loss: 0.3222 Best Val acc: 89.75%
Epoch 24/30, Train loss: 0.3388, Train acc: 88.00%, Val loss: 0.3247, Val acc: 89.75%, Best Val loss: 0.3222 Best Val acc: 89.75%
Epoch 25/30, Train loss: 0.3375, Train acc: 88.25%, Val loss: 0.3214, Val acc: 89.25%, Best Val loss: 0.3214 Best Val acc: 89.75%
Epoch 26/30, Train loss: 0.3374, Train acc: 87.94%, Val loss: 0.3215, Val acc: 89.50%, Best Val loss: 0.3214 Best Val acc: 89.75%
Epoch 27/30, Train loss: 0.3370, Train acc: 88.25%, Val loss: 0.3213, Val acc: 89.50%, Best Val loss: 0.3213 Best Val acc: 89.75%
Epoch 28/30, Train loss: 0.3368, Train acc: 88.00%, Val loss: 0.3221, Val acc: 89.75%, Best Val loss: 0.3213 Best Val acc: 89.75%
Epoch 29/30, Train loss: 0.3364, Train acc: 88.31%, Val loss: 0.3200, Val acc: 89.50%, Best Val loss: 0.3200 Best Val acc: 89.75%
Epoch 30/30, Train loss: 0.3364, Train acc: 88.06%, Val loss: 0.3197, Val acc: 89.50%, Best Val loss: 0.3197 Best Val acc: 89.75%
```

圖 10, 上圖為改 model 前的表現情況，accuracy 卡在 88-89%之間就上不去了

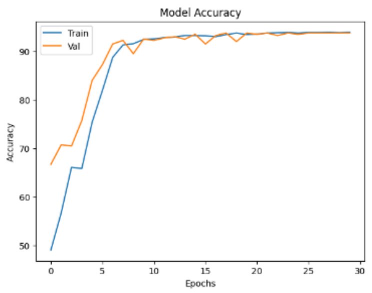
因為觀察圖 10 可以發現，在後期 accuracy 基本上都徘徊在相同的地方，所以我選擇不調整 epoch 次數，因為調整了也只會徘徊在相同的 accuracy。
所以我更改了 layer 的層數，在 model 中多增加了兩層的 layer，讓 model 更能夠抓住特徵值，其結果如下圖

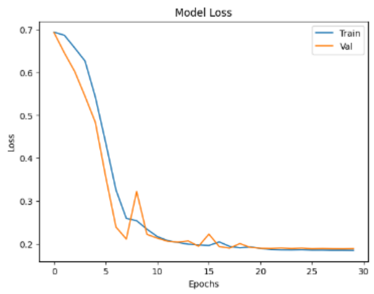

```
class ConvGAP(nn.Module):#slice很特別的東西，新的架構
    def __init__(self):
        super().__init__()

        self.net = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, stride=1, padding='same') ,
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2), # 128*128
            nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same'), # 128*128
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2), # 64*64
            nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same'), # 64*64
            nn.ReLU(),
            nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same'), # 64*64
            nn.ReLU(),
            nn.Conv2d(32, 32, kernel_size=3, stride=1, padding='same'), # 64*64
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2), # 32*32

            nn.AdaptiveAvgPool2d(1),
            nn.Flatten(),
            nn.Linear(32, 1)
        )

    def forward(self, x):
```
圖 11,更改後的 model





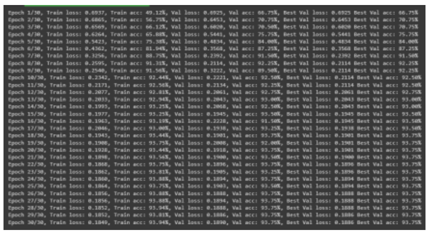

```
Epoch 1/30, Train loss: 0.6937, Train acc: 49.12%, Val loss: 0.6925, Val acc: 66.75%, Best Val loss: 0.6925 Best Val acc: 66.75%
Epoch 2/30, Train loss: 0.6865, Train acc: 56.75%, Val loss: 0.6453, Val acc: 70.75%, Best Val loss: 0.6453 Best Val acc: 70.75%
Epoch 3/30, Train loss: 0.6569, Train acc: 66.12%, Val loss: 0.6020, Val acc: 70.50%, Best Val loss: 0.6020 Best Val acc: 70.75%
Epoch 4/30, Train loss: 0.6264, Train acc: 65.88%, Val loss: 0.5441, Val acc: 75.75%, Best Val loss: 0.5441 Best Val acc: 75.75%
Epoch 5/30, Train loss: 0.5423, Train acc: 75.38%, Val loss: 0.4834, Val acc: 84.00%, Best Val loss: 0.4834 Best Val acc: 84.00%
Epoch 6/30, Train loss: 0.4362, Train acc: 81.94%, Val loss: 0.3568, Val acc: 87.25%, Best Val loss: 0.3568 Best Val acc: 87.25%
Epoch 7/30, Train loss: 0.3256, Train acc: 88.75%, Val loss: 0.2392, Val acc: 91.50%, Best Val loss: 0.2392 Best Val acc: 91.50%
Epoch 8/30, Train loss: 0.2595, Train acc: 91.31%, Val loss: 0.2114, Val acc: 92.25%, Best Val loss: 0.2114 Best Val acc: 92.25%
Epoch 9/30, Train loss: 0.2540, Train acc: 91.56%, Val loss: 0.3222, Val acc: 89.50%, Best Val loss: 0.2114 Best Val acc: 92.25%
Epoch 10/30, Train loss: 0.2342, Train acc: 92.44%, Val loss: 0.2221, Val acc: 92.50%, Best Val loss: 0.2114 Best Val acc: 92.50%
Epoch 11/30, Train loss: 0.2171, Train acc: 92.56%, Val loss: 0.2134, Val acc: 92.25%, Best Val loss: 0.2114 Best Val acc: 92.50%
Epoch 12/30, Train loss: 0.2077, Train acc: 92.81%, Val loss: 0.2061, Val acc: 92.75%, Best Val loss: 0.2061 Best Val acc: 92.75%
Epoch 13/30, Train loss: 0.2033, Train acc: 92.94%, Val loss: 0.2043, Val acc: 93.00%, Best Val loss: 0.2043 Best Val acc: 93.00%
Epoch 14/30, Train loss: 0.1995, Train acc: 93.25%, Val loss: 0.2068, Val acc: 92.50%, Best Val loss: 0.2043 Best Val acc: 93.00%
Epoch 15/30, Train loss: 0.1977, Train acc: 93.25%, Val loss: 0.1945, Val acc: 93.50%, Best Val loss: 0.1945 Best Val acc: 93.50%
Epoch 16/30, Train loss: 0.1963, Train acc: 93.19%, Val loss: 0.2228, Val acc: 91.50%, Best Val loss: 0.1945 Best Val acc: 93.50%
Epoch 17/30, Train loss: 0.2046, Train acc: 93.00%, Val loss: 0.1938, Val acc: 93.25%, Best Val loss: 0.1938 Best Val acc: 93.50%
Epoch 18/30, Train loss: 0.1943, Train acc: 93.44%, Val loss: 0.1901, Val acc: 93.75%, Best Val loss: 0.1901 Best Val acc: 93.75%
Epoch 19/30, Train loss: 0.1908, Train acc: 93.75%, Val loss: 0.2008, Val acc: 92.00%, Best Val loss: 0.1901 Best Val acc: 93.75%
Epoch 20/30, Train loss: 0.1928, Train acc: 93.44%, Val loss: 0.1918, Val acc: 93.75%, Best Val loss: 0.1901 Best Val acc: 93.75%
Epoch 21/30, Train loss: 0.1898, Train acc: 93.56%, Val loss: 0.1900, Val acc: 93.50%, Best Val loss: 0.1900 Best Val acc: 93.75%
Epoch 22/30, Train loss: 0.1868, Train acc: 93.75%, Val loss: 0.1896, Val acc: 93.75%, Best Val loss: 0.1896 Best Val acc: 93.75%
Epoch 23/30, Train loss: 0.1862, Train acc: 93.81%, Val loss: 0.1905, Val acc: 93.25%, Best Val loss: 0.1896 Best Val acc: 93.75%
Epoch 24/30, Train loss: 0.1860, Train acc: 93.88%, Val loss: 0.1894, Val acc: 93.75%, Best Val loss: 0.1894 Best Val acc: 93.75%
Epoch 25/30, Train loss: 0.1864, Train acc: 93.75%, Val loss: 0.1903, Val acc: 93.50%, Best Val loss: 0.1894 Best Val acc: 93.75%
Epoch 26/30, Train loss: 0.1856, Train acc: 93.88%, Val loss: 0.1888, Val acc: 93.75%, Best Val loss: 0.1888 Best Val acc: 93.75%
Epoch 27/30, Train loss: 0.1856, Train acc: 93.88%, Val loss: 0.1894, Val acc: 93.75%, Best Val loss: 0.1888 Best Val acc: 93.75%
Epoch 28/30, Train loss: 0.1852, Train acc: 93.94%, Val loss: 0.1888, Val acc: 93.75%, Best Val loss: 0.1888 Best Val acc: 93.75%
Epoch 29/30, Train loss: 0.1852, Train acc: 93.81%, Val loss: 0.1886, Val acc: 93.75%, Best Val loss: 0.1886 Best Val acc: 93.75%
Epoch 30/30, Train loss: 0.1849, Train acc: 93.94%, Val loss: 0.1890, Val acc: 93.75%, Best Val loss: 0.1886 Best Val acc: 93.75%
```

圖 12, train 和 validation 的 accuracy 都上升到了 93%左右。
最終結果如預期所想，accuracy 上升，並且 train 和 validation 的 loss 和 accuracy 之間的差距也變小了，理論上此 model 應該更能夠穩定的用在測試未知資料的預測上了