CV HW2 report

姓名: 梁中瀚
學號: r09922a02
系級: 資工 人工智慧 碩一

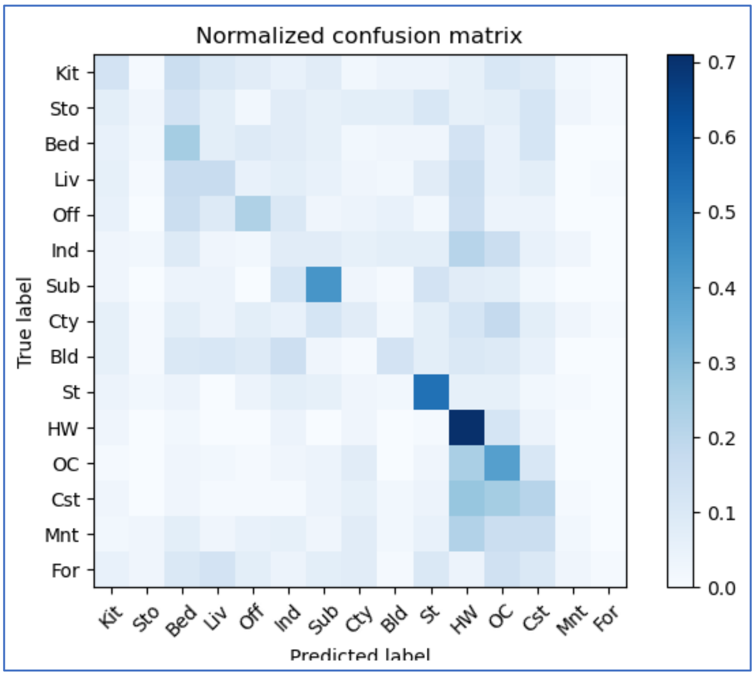
Part 1

I. Report accuracy of two settings

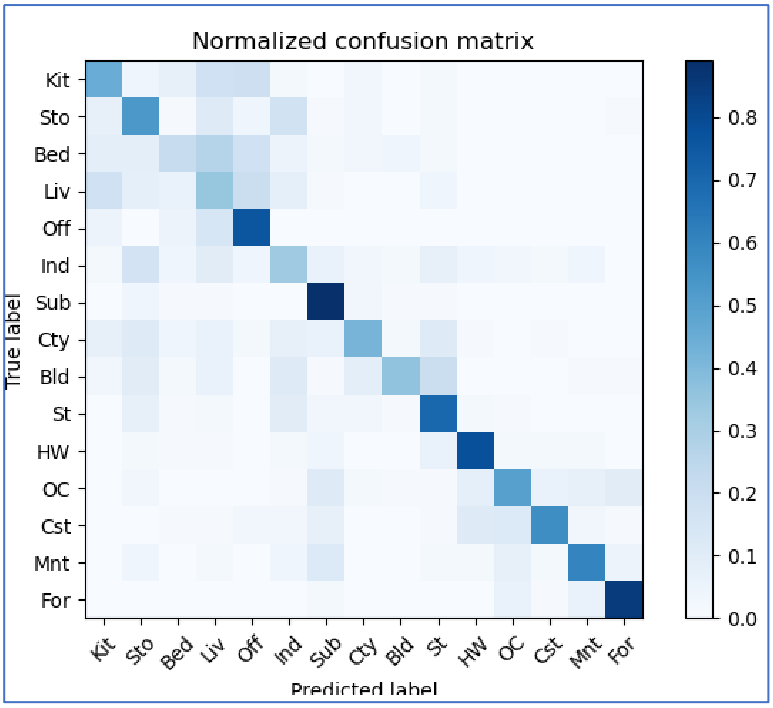
- Tiny images + KNN: 0.2273
- Bag of sift + KNN: 0.553

II. Plot confusion matrix of two settings

- Tiny images + KNN:



- Bag of sift + KNN:



III. Compare the results of both settings and explain the result

Tiny images 的做法僅僅是將原圖的 resolution 降低，再 pixel-wise 的計算兩張圖的 Euclidean distance，當作 KNN 當中的距離求出最接近的 training data。

Bag of sift 的做法會用 sift 求出多個 descriptors 再計算預先建好的 vocabulary 的出現次數當作 feature，相較於 Tiny image 中 pixel-wise 的計算方式，這些 sift descriptor 具有更好的比較意義。

從 Confusion matrix 中可以看到，使用 tiny images 方法預測的結果，在 HW 的類別上有最高的準確度，應該是因為高速公路的圖片，有比較相近的結構，所以就算 resolution 降低 KNN 依然很有效。而其它類別的預測準確度都不高，尤其是 Mnt 與 For(森林與高山)的類別式最少被預測到的，可能是因為住兩個類別圖片間的結構相差很大，所以把 low resolution 的圖片當作 feature 會讓 KNN 無法判斷與其他類別的不同。

Bag of sift 相比於 tiny image 有較高的準確率。只有 Bed 與 Liv 兩個類別的預測準確度相對較低，這兩個類別有極大的機率預測為其它室內空間的類別(Bed, liv, Off)，猜測可能的原因是這些類別的 sift descriptor 都太相似，都有大量相似的方形特徵。

IV. Bag of sift parameters

(Best one, which have 0.55 accuracy)

1. Constructing vocabulary
   - Number of sift descriptors sample from each image: 250
   - Sift step: [10, 10]
   - Sift size: [3, 3]
   - Number of vocabulary: 600
2. Calculating histogram of each image
   - Number of sift descriptors sample from each image: 750
   - Sift step: [3, 3]
   - Sift size: [3, 3]
3. KNN
   - K: 250

V. Files

- p1.py
- get_tiny_images.py
- build_vocabulary.py
- get_bags_of_sifts.py
- nearest_neighbor_classify.py
- vocab.pkl
- train_image_feats.pkl
- test_image_feats.pkl

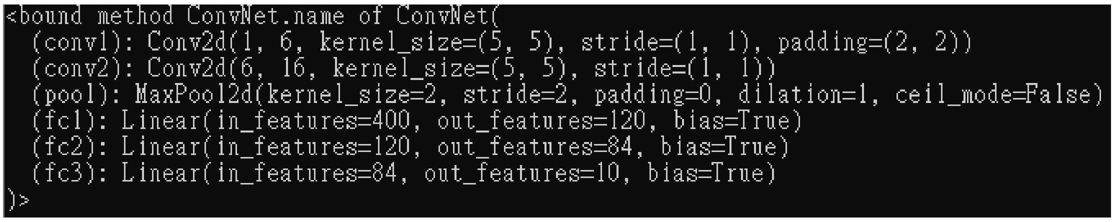
Part 2

I. Print the network architectures &

number of parameters of both models

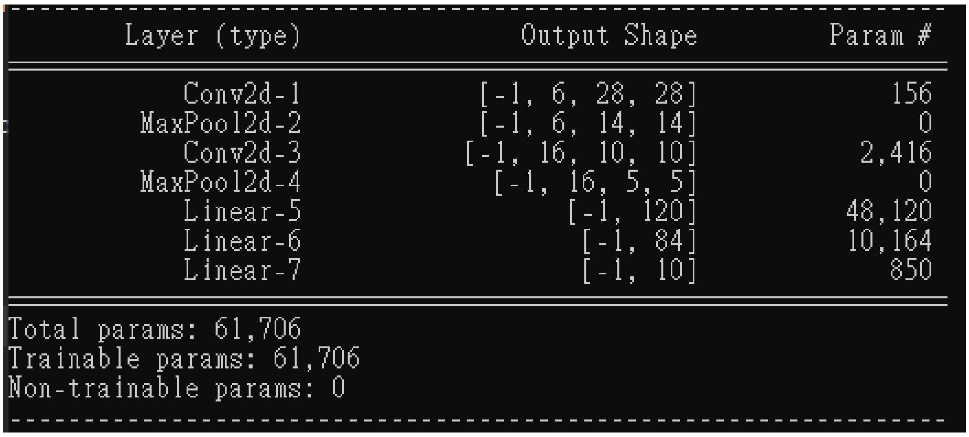
- Baseline model (LeNet-5):

- Network architecture



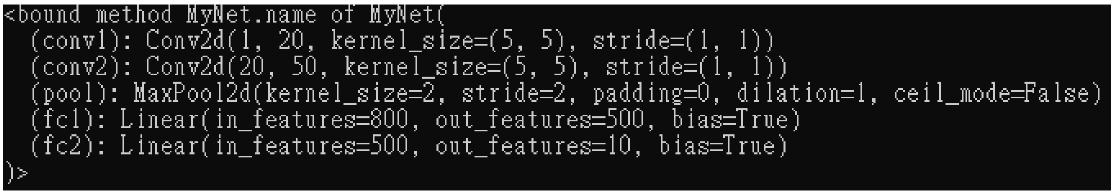
```
<bound method ConvNet.name of ConvNet(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)>
```

- Number of parameters: 61706



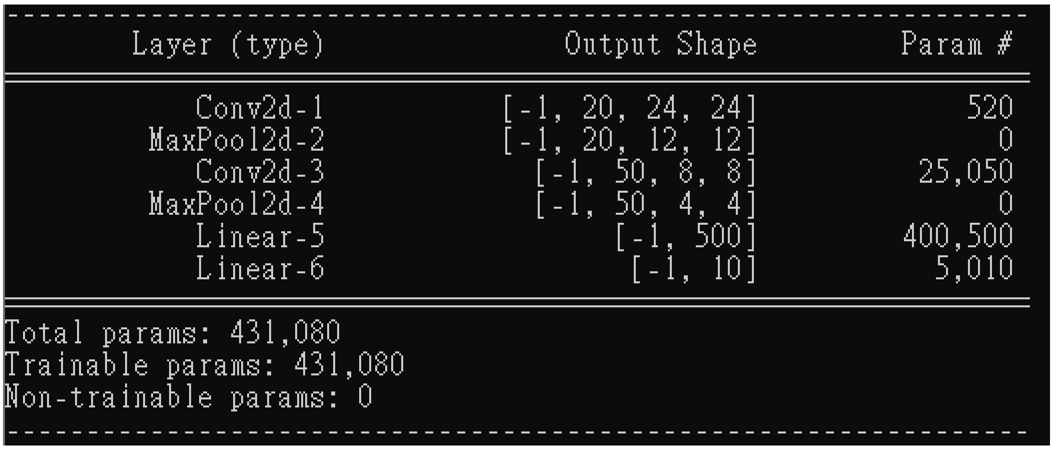
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 28, 28]             156
         MaxPool2d-2            [-1, 6, 14, 14]               0
            Conv2d-3           [-1, 16, 10, 10]           2,416
         MaxPool2d-4             [-1, 16, 5, 5]               0
            Linear-5                  [-1, 120]          48,120
            Linear-6                   [-1, 84]          10,164
            Linear-7                   [-1, 10]             850
================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
----------------------------------------------------------------
```

- Improved model (MyNet):
  - Network architecture



```
<bound method MyNet.name of MyNet(
  (conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=800, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=10, bias=True)
)>
```

  - Number of parameters 431080



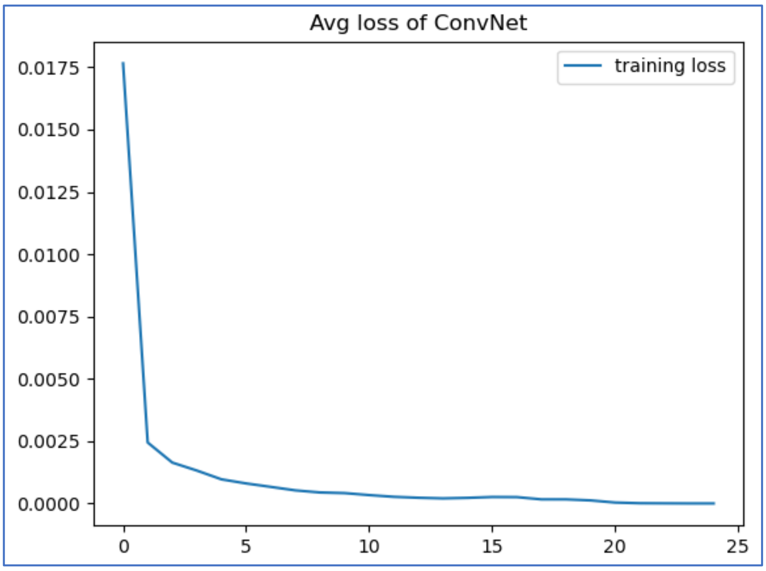
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 20, 24, 24]             520
         MaxPool2d-2           [-1, 20, 12, 12]               0
            Conv2d-3             [-1, 50, 8, 8]          25,050
         MaxPool2d-4             [-1, 50, 4, 4]               0
            Linear-5                  [-1, 500]         400,500
            Linear-6                   [-1, 10]           5,010
================================================================
Total params: 431,080
Trainable params: 431,080
Non-trainable params: 0
----------------------------------------------------------------
```

II. Plot the learning curve (loss, accuracy) of the training process(train/validation)

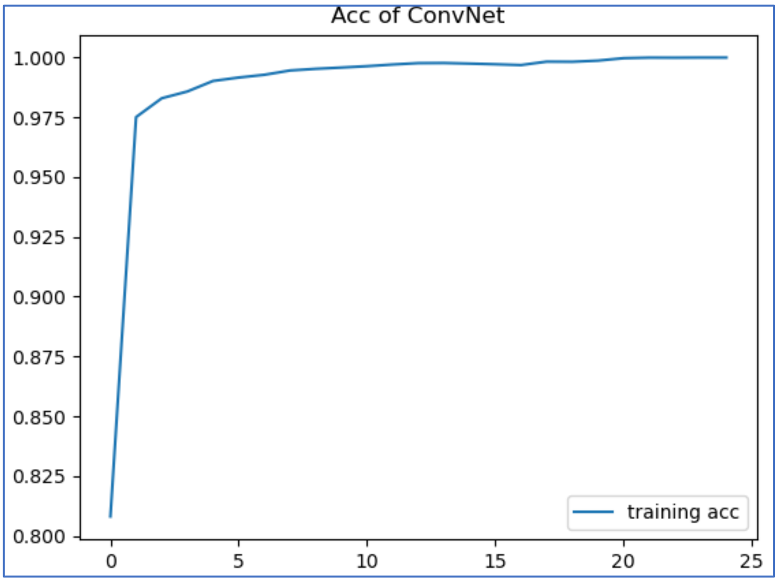
兩個 model 的訓練過程皆使用 25 個 epoch

- Baseline model (LeNet-5):

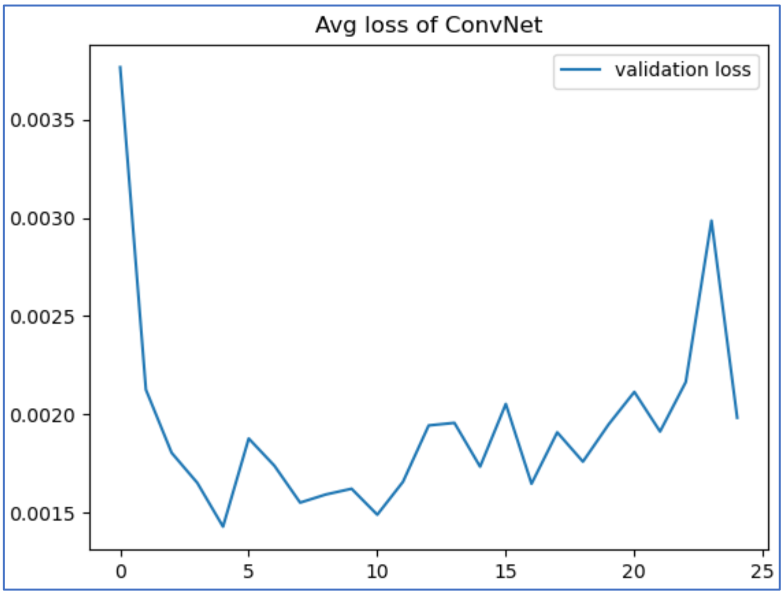
1. Learning curve of training loss



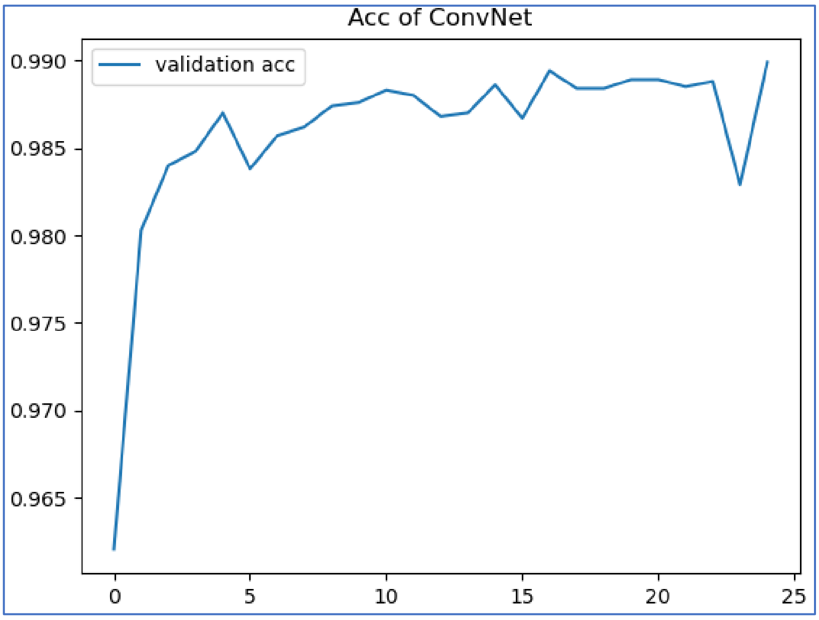
2. Learning curve of training accuracy



3. Learning curve of validation loss

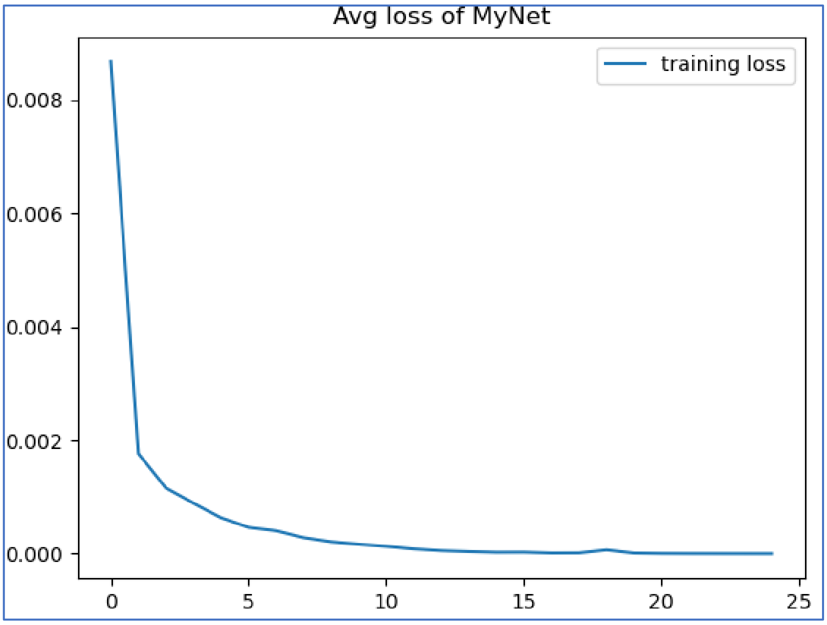


4. Learning curve of validation accuracy

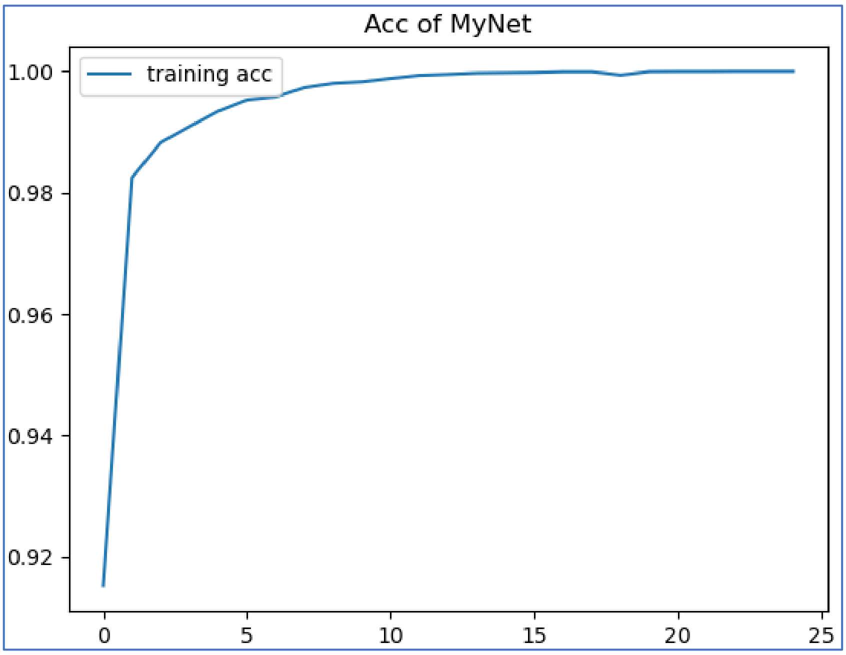


- Improved model (MyNet):

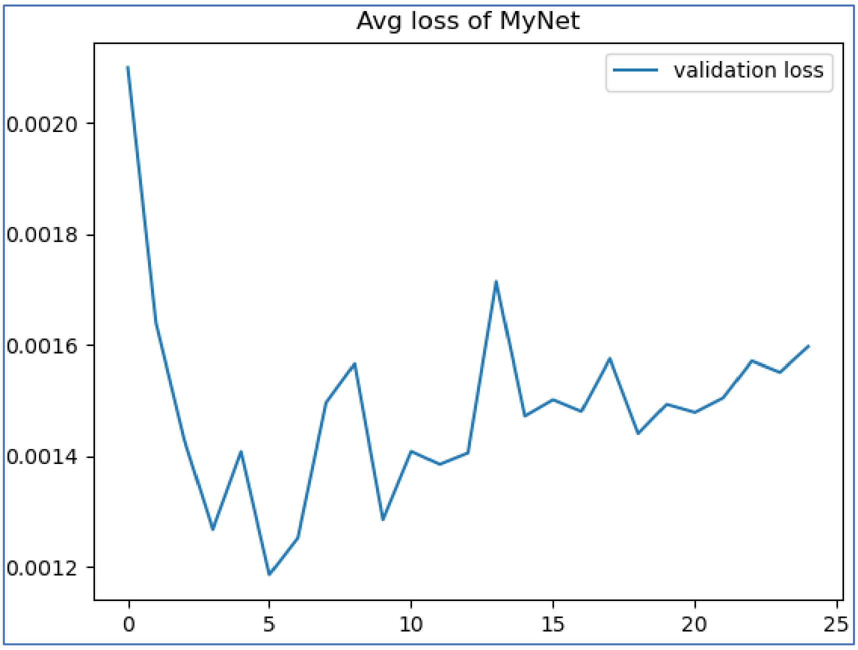
1. Learning curve of training loss



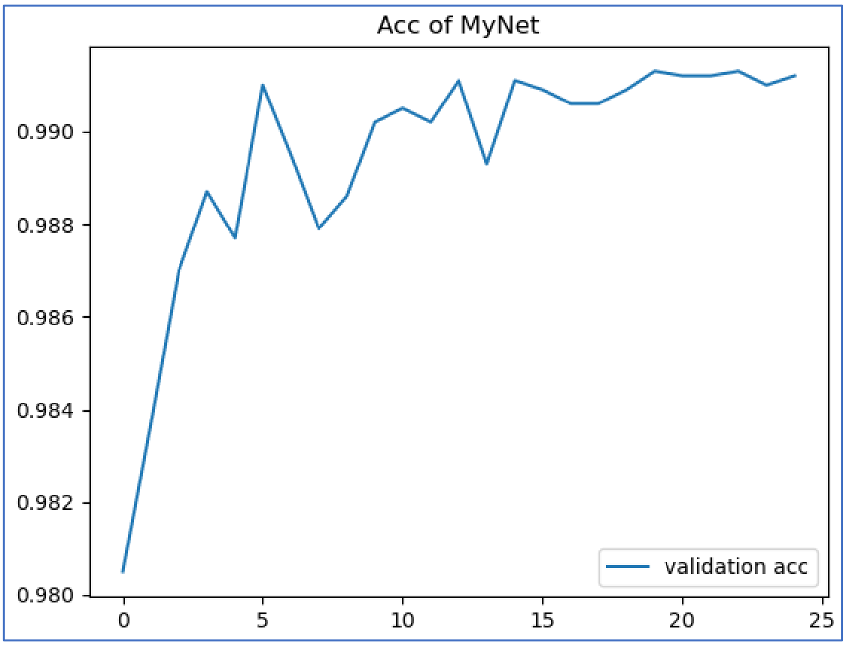
2. Learning curve of training accuracy



3. Learning curve of validation loss



4. Learning curve of validation accuracy



III. Compare the results of both model and explain the result

LeNet 在 validation set 上最好的預測結果為: 98.99%的準確率。網路的架構非常單純，只有三種 layer，convolution、pooling、fully connected。MyNet 在 validation set 上最好的預測結果為: 99.12%。MyNet 與 baseline model 不同的地方在於，增加捲積層的 kernel 數量，所以參數數量才會增加，並且在最後一層加入了 log_softmax，讓所有輸出的總合為 1。

IV. Files

- model.py
- eval.py
- train.py
- ConvNet.pth: baseline model
- MyNet.pth: improved model