# 資料科學

## Data Science

**作業六 HW6**

電機所 R11921038 江讀晉

2022/12/22

## Problem. Image Classification with Convolutional Neural Networks

### 0. Problem

- Train CNN models to recognize handwritten digits (0~9).
- Dataset: MNIST-M, which contains 70k images of handwritten digits from "0" to "9".

### 1. Hyperparameter

| Parameter | | Value |
|---|---|---|
| Number of epochs | num_epochs | 50 |
| Learning rate | lr | 0.001 |
| Decay rate of learning rate | lr_gamma | 0.911 |
| Decay step of learning rate | lr_decay_step | 1 |
| Optimizer | Adam | |
| Momentum | momentum | 0.95 |
| Decay rate of weight | weight_decay | 0.0005 |
| Batch size of training dataloader | train_batch_size | 256 |

Table 1. Configure of hyperparameter

超參數的選擇，目標是以在 50 個 epoch 內得到可行的結果。將 learning rate 設計為由 $1x10^{-3}$ 降至 $1x10^{-5}$，並計算對應的 lr_gamma 和 lr_decay_step，使其可以達到合理的遞減效果。此外，momentum 和 weight_decay 皆使用預設的參數，並無特別改動。Optimizer，從原本的 SGD 改為同時使用 first 和 second momentum 的 Adam，一般來說，Adam 的效果會比 SGD 好。

### 2. Architecture of the model

```
ResNet18_modified(
  (model): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
```

```
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )

    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )

    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))

    (fc): Sequential(
      (0): Linear(in_features=128, out_features=64, bias=True)
      (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): GELU()
      (3): Linear(in_features=64, out_features=10, bias=True)
    )
  )
)
```

Figure 1. Architecture of the model

## 3. Explanation of the design of the model

整體而言，我是以已定義好的 ResNet18 [1] 為藍本，並去除原本的 layer3 和 layer4（可以參考 Figure 2），並重新定義最後的 fully-connected layer，詳見前段的 Figure 1。如此一來，在這次作業所使用的 MNIST-M 資料集上可以維持不錯的 accuracy 之外也能得到較快的 training latency。

而 data preprocessing 的部分，無論是 training 或是 validation step，都是將原本的影像 resize 至 64 x 64，並進行 normalization。

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Figure 2. layer3 和 layer4 為表格中 conv4_x 和 conv5_x [1]

(此部分可以對照 Figure 1 的 depth of feature map)

## 4. Plot the learning curve during training (CrossEntropy Loss)

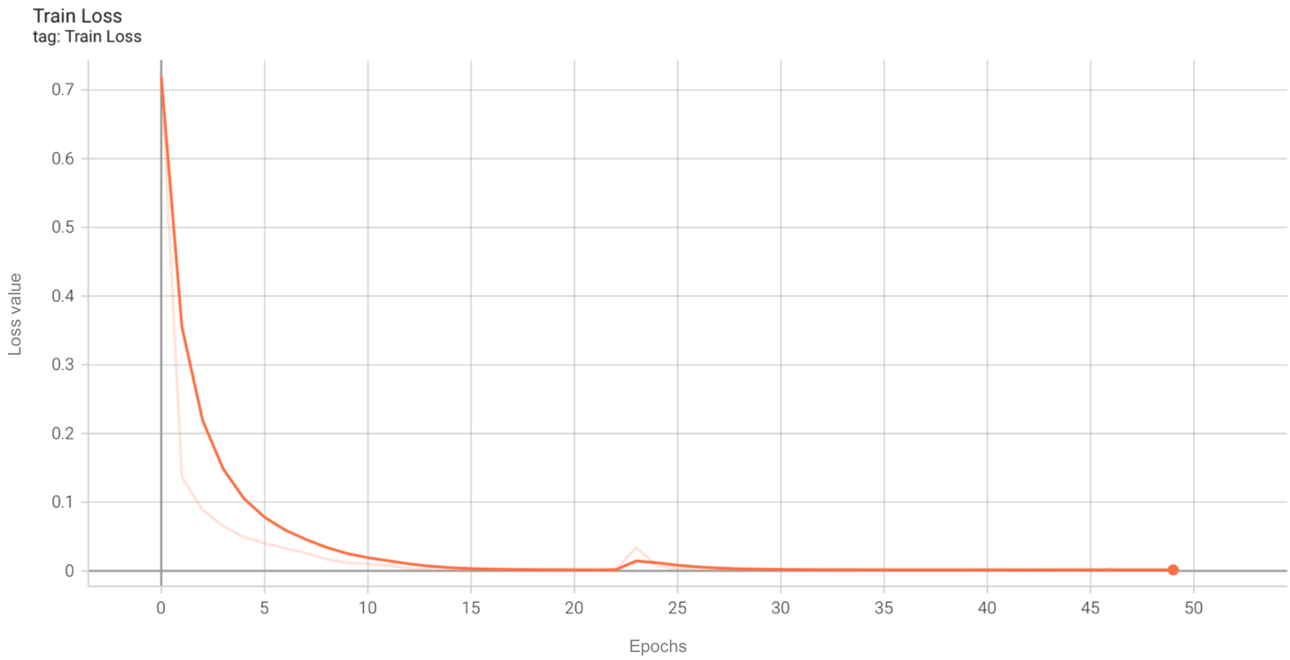


Figure 3. The curve of binary cross-entropy training loss

### 5. Plot the confusion matrix for validation set, and briefly explain the observation

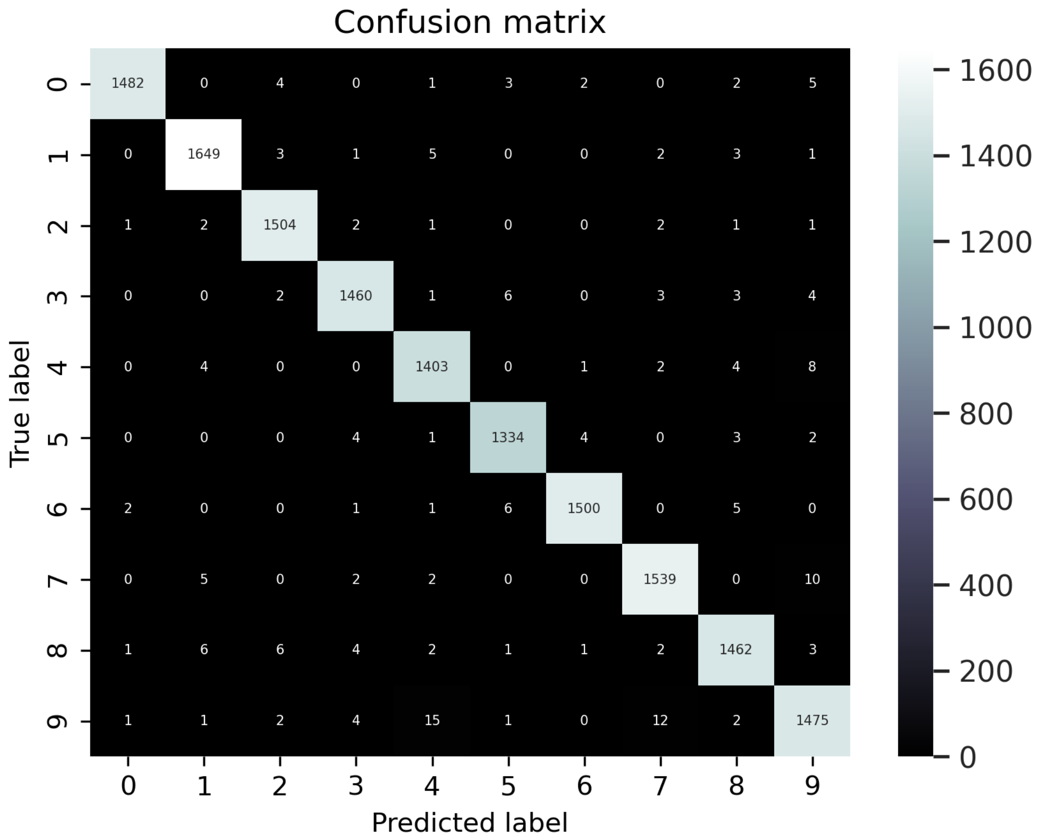


Figure 4. The confusion matrix of the validation dataset (15000 instances in total)

從 Figure 3 confusion matrix 的 diagonal 項可以看出，在扣除掉預測錯誤的部分之後，label 為 1 的 test image 佔最多數，不過整體 10 個類別的資料量仍分佈平均。

至於深色部分，也就是 predicted label 和 true label 不一致的部分，單筆最大值不超過整體資料量的 0.1%，結果還算理想，但仍有進步的空間。比如，將 9 預測錯誤的資料數就達 30 筆以上，其中預測成 4（15 筆）和 7（12 筆）佔最多數，而將 7 預測為 9 的資料數也有 10 筆，這裡顯然是可以著手的地方。或許，改變 data augmentation 能夠改善這樣的問題。

## Reference

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.