HW2 Report

學號:311512015　姓名:謝元碩

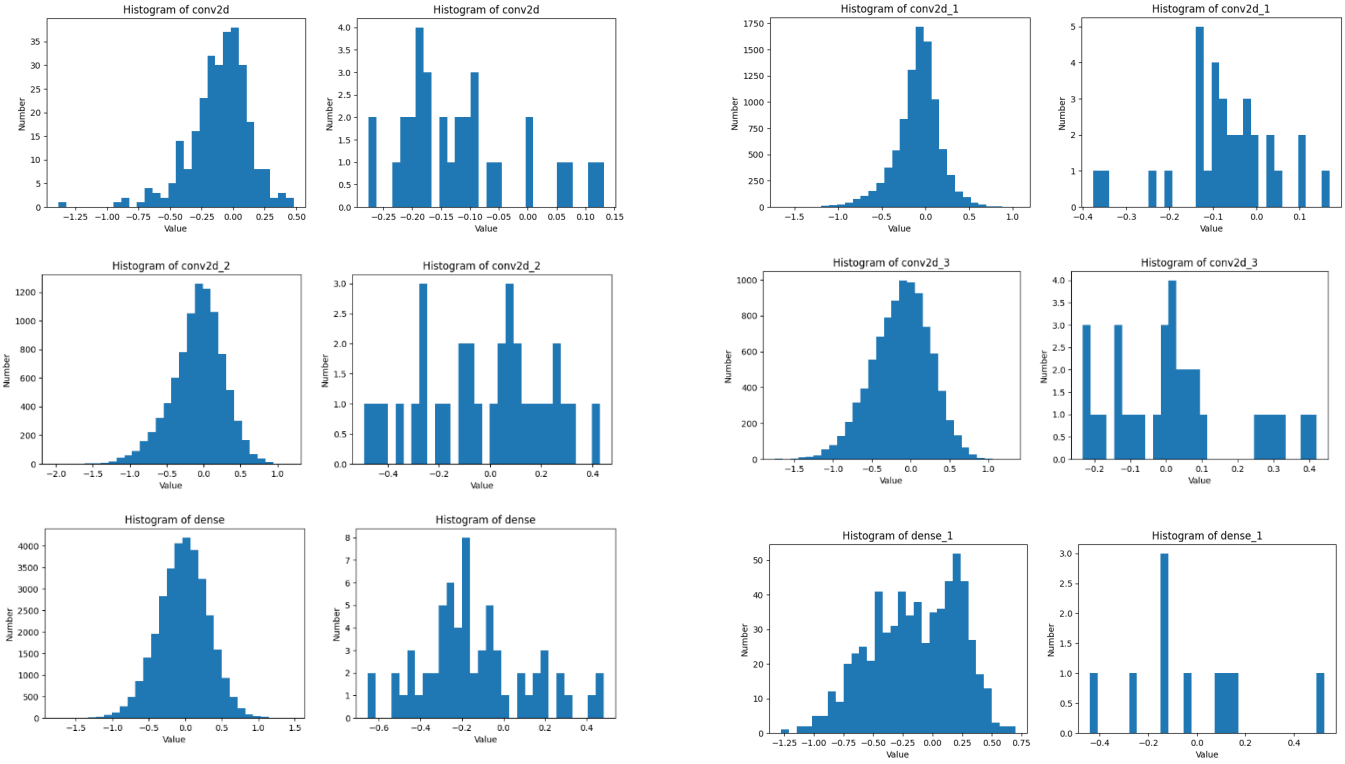
一、 Using Convolutional Neural Network for Image Recognition

1. 設計網路架構，分析 stride 大小和 filter 大小對訓練的影響。

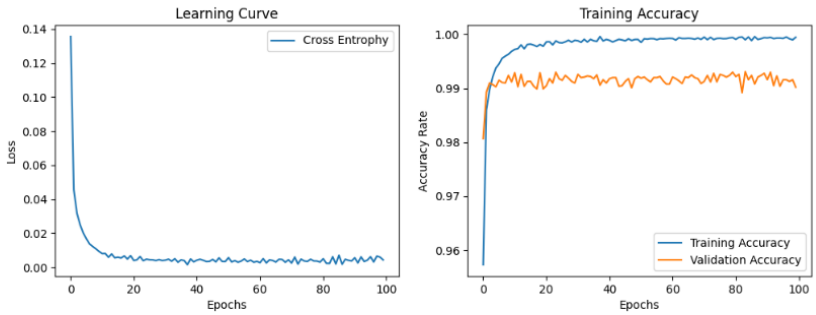
(1) 網路架構設計

```
[layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
layers.MaxPooling2D((2, 2)),
layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
layers.MaxPooling2D((2, 2), strides=2),
layers.Flatten(),
layers.Dense(64, activation='relu'),
layers.Dense(10, activation='softmax'),
```

架構設計如上圖，首先經過兩組 filter size 為 3、neuron 數量為 32 的卷積層後，使用一層最大池化層(size 為 2x2)降低 feature map 尺寸以減少參數數量和計算量，接著同樣再經過一組參數相同的 conv-conv-maxpooling。最後使用 FNN，以 RELU 將特徵資訊非線性化，再以 softmax 轉成機率分布表示方式作為輸出結果，完成網路架構之設計。



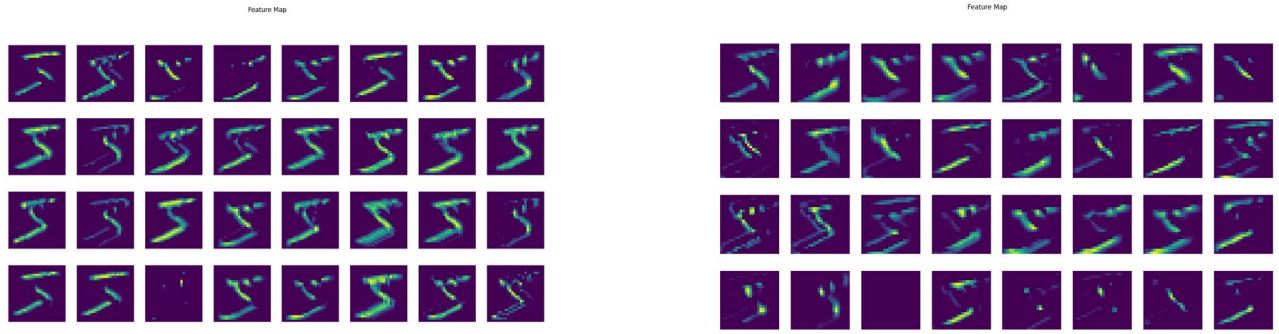
以前述模型作訓練後，每一層的 weight 和 bias 分布狀況如上圖(左為 weight、右為 bias)，而學習曲線以及訓練集和驗證集之正確率變化如下圖所示。最終訓練集正確率 99.95%、驗證集正確率 99.02%，使用測試集之正確率為 98.05%。



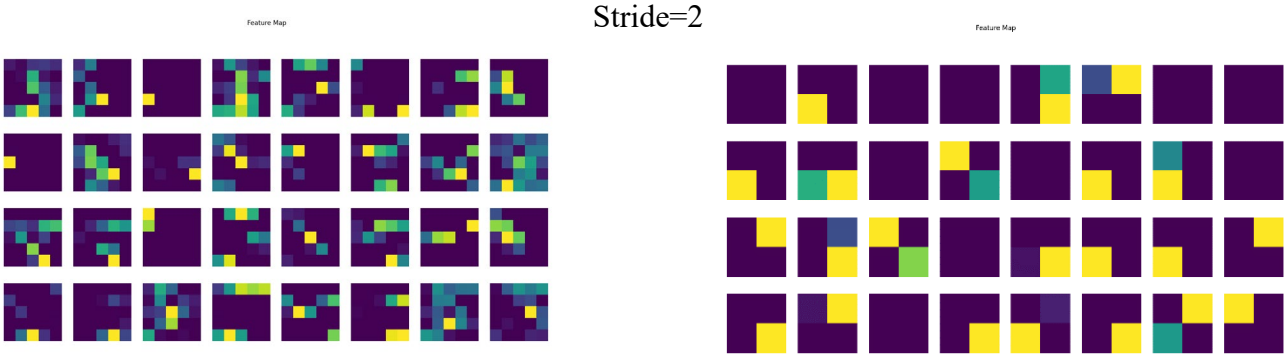
(2) Stride size 對訓練的影響

下圖為分別使用 stride=1 及 stride=2 的 feature map。Stride 代表卷積層中 filter 對輸入數據進行內積操作的間格移動距離，因此較高的 stride 代表作完一次內積後，會跳過幾個 pixel 才會再進行一次內積，如此會導致輸出 feature map 的解析度降低。下圖可見 stride=1 的特徵圖較清晰，且保留更多的細節特徵，而 stride=2 的解析度則逐漸降低，特徵相當不明顯，但能夠減少計算複雜度以及 overfitting 的情況。

Stride=1



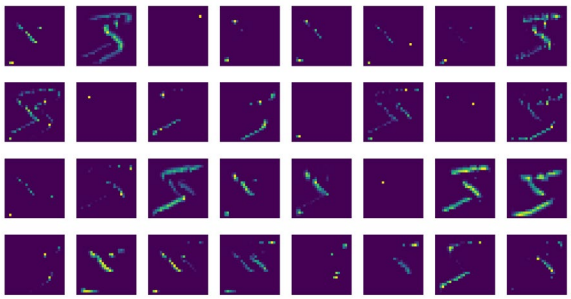
Stride=2



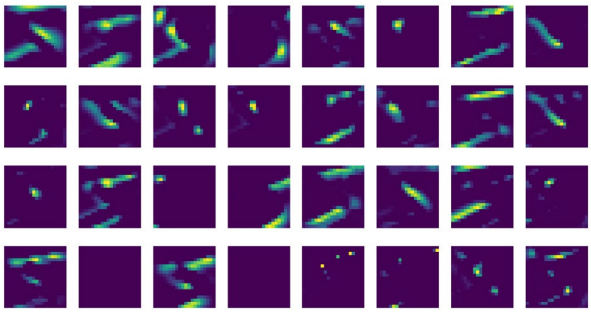
(3) Filter size 對訓練的影響

下圖針對 filter size 為 3、5 做比較。Filter 為卷積層進行內積計算時的矩陣大小，較小的 filter 通常用於觀察較小尺度的高級特徵，如邊緣；較大的 filter 則用於觀察較大尺度低級特徵，如紋理、形狀。從 feature map 可以明顯觀察到差別，size=5 的特徵圖大多為圖片中的線條、文字本身的形狀，size=3 則大多為轉角、特徵點等，較少有大尺度特徵出現。

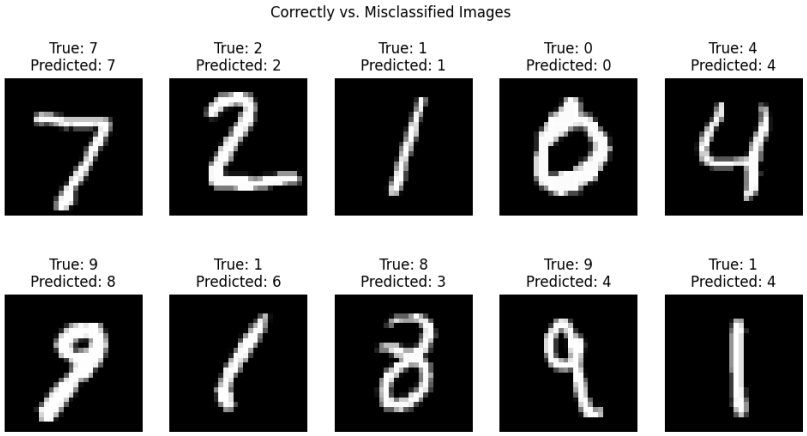
Filter size=3



Filter size=5

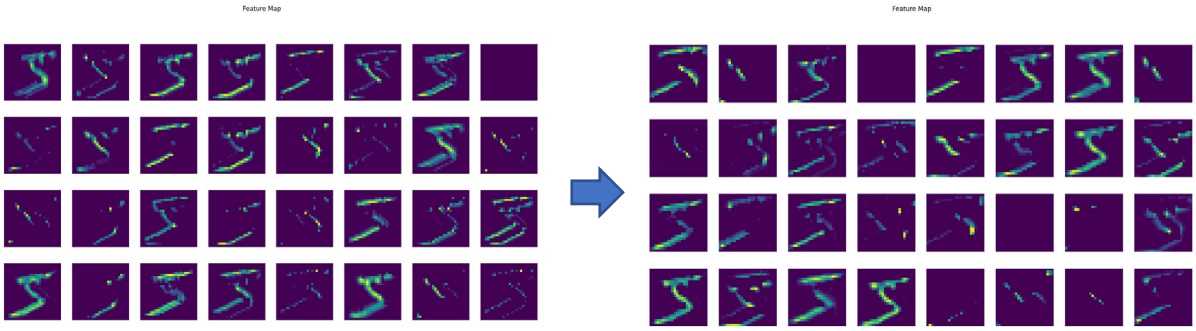


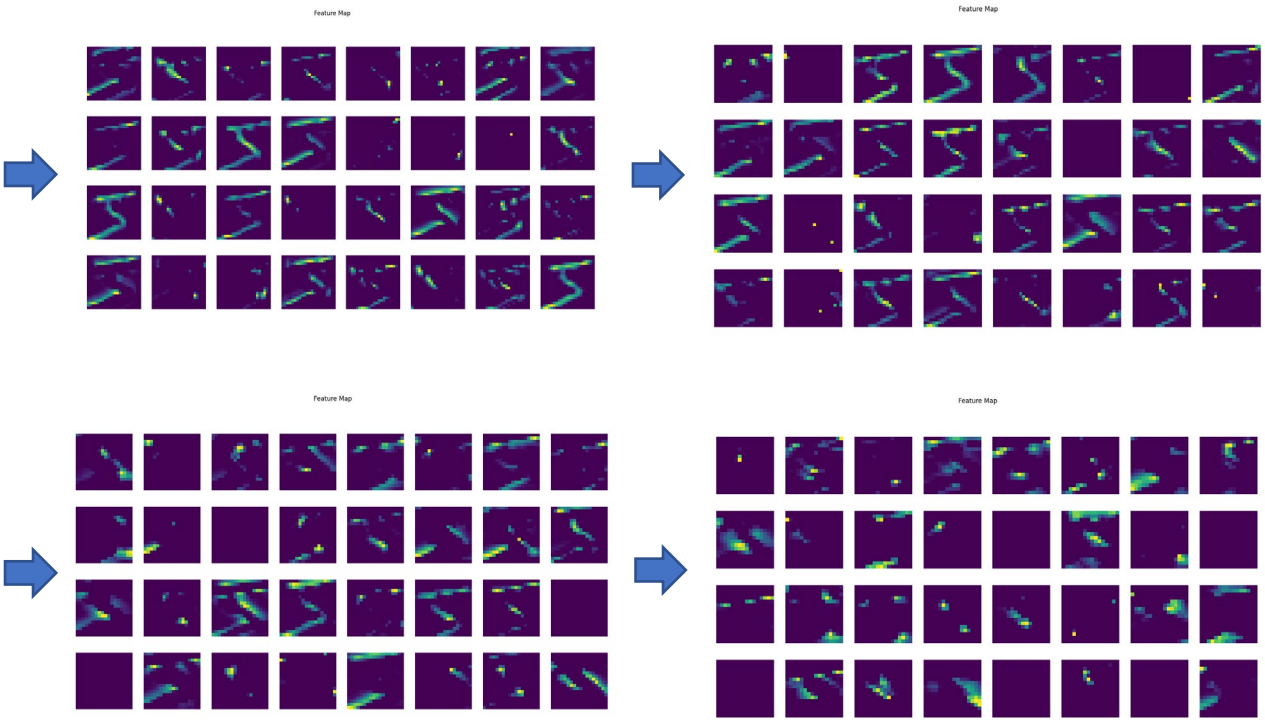
2. **Some examples of correctly classified and miss-classified images**



測試集經過所訓練之模型預測後，將預測正確與失敗的情況輸出如上圖。在預測失敗的情況中，往往是因為不同數字間擁有類似的特徵導致，如 8 預測成 3，8 與 3 皆擁有弧狀特徵；9 預測成 8，9 與 8 皆擁有 o 的形狀等，若網路架構設計太簡略，這些相似的特徵將難以透過層層 filter 去區分出來，然而這已經是極少發生的情況，因我們訓練出的模型在測試集之預測正確率為 98.05%。

3. **observe the feature maps from different convolutional layers and describe how a feature map changes with increasing depth.**

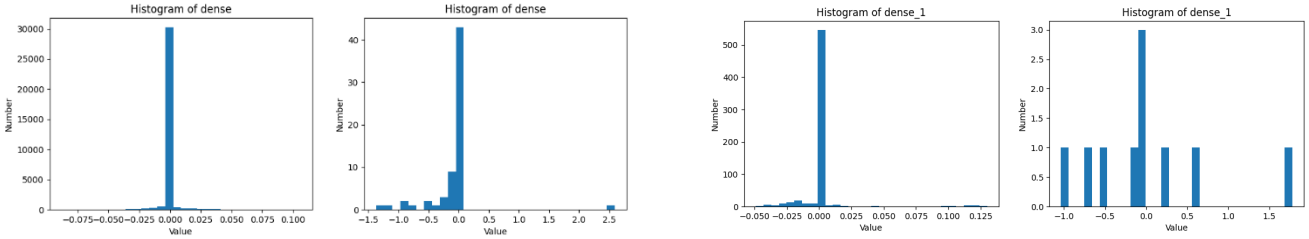




上圖是從第一層 CNN 卷積層至第 6 層卷積層的各層 feature map，可以觀察到在較前面的 layer(淺層深度)所取得的特徵大多為較簡單的特徵，如文字的紋理和形狀(可以明顯看到很多 5)；愈往後面的 layer(深層深度)，以形狀為主的特徵圖數量變少，轉而變成以小範圍的細節特徵為主，如數字 5 的轉角，甚至是我們看不出來的抽象特徵，通常用於辨識目標物的整體複雜特徵。

4. Please add L2 regularization to the CNN implemented in 1-1 and discuss its effect

將 L2 正規化加入損失函數中可以限制模型參數的大小，避免過大參數值造成訓練結果 overfitting 的情況發生、提升模型強健性及泛化性。此次作業中，我在模型最後面的 FNN 加入 L2 正規化，將參數降低為原本的 0.01 倍，以限制 weight 和 bias 的最大與最小範圍。



加入 L2 正規化後的全連接層，可以看到 weight 和 bias 的訓練結果，數值大多變為集中(更集中在 0)，然而在特徵圖沒辦法看到明顯變化，與未使用正規化的特徵圖相似。正確率的部分，訓練集和驗證集發生些微降低的情形，訓練集正確率 99.77%、驗證集正確率 98.91%，但測試集之正確率卻從從 98.05%提升為 98.78%，可見加入正規化項目後，此模型的泛化性可以達到提升的效果。

二、 Preprocessing Before Using Convolutional Neural Net- work for Image Recognition

1. 資料前處理

對於 CIFAR-10 資料集，以下是我做的幾項前處理：

(1) 正規化

將圖像 pixel 值限制在 0~1 以內，這可以讓所有輸入圖片特徵都在相似尺度上，讓訓練更加問穩定。

```
# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

(2) 均值減法

此方法減少光照和色彩偏差對模型的影響。

```
# decrease light
mean = np.mean(train_images, axis=(0, 1, 2))
train_images -= mean
test_images -= mean
```

(3) 合成數據集(數據增強)

對訓練圖像作隨機變換來增加數據多樣性，如翻轉、旋轉、飽和度、亮度、縮放等，此方法可提升模型泛化性，但訓練難度也會有所增加。

```
# image augmented
datagen = ImageDataGenerator(
    rotation_range=90,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

(4) 動態調整 learning rate

當 epoch 愈大，學習率逐漸下降，提升收斂、減少震盪

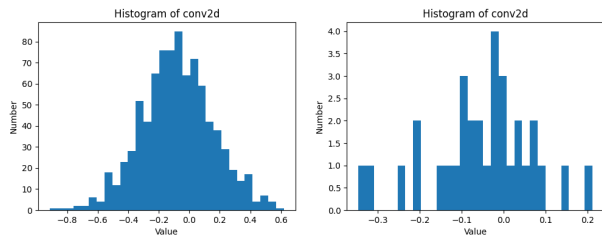
```
def lr_schedule(epoch):
    lr = 1e-3
    if epoch > 25:
        lr *= 0.5
    elif epoch > 50:
        lr *= 0.5
    elif epoch > 75:
        lr *= 0.5
    return lr
```

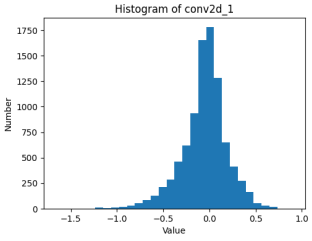
2. 訓練模型架構與結果

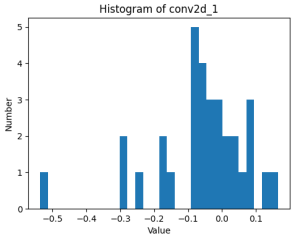
網路模型設計如下，與 MNIST 的架構不同之處，在於每一層的 neuron 數量有所增加，由於此數據集為彩色圖片，資訊較複雜，因此希望取得更多的特徵來提升辨識精準度，且在 max pooling 的 stride 為 2，目的是為了降低模型參數量，輕量化該模型。

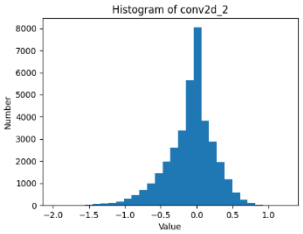
```
[layers.Conv2D(128, (3, 3), padding="same", activation='relu', input_shape=(32, 32, 3)),
layers.Conv2D(128, (3, 3), padding="same", activation='relu'),
layers.MaxPooling2D((2, 2), strides=2, padding='valid'),
layers.Conv2D(64, (3, 3), padding="same", activation='relu'),
layers.Conv2D(64, (3, 3), padding="same", activation='relu'),
layers.MaxPooling2D((2, 2), strides=2, padding='valid'),
layers.Flatten(),
layers.Dense(64, activation='relu'),
layers.Dense(10, activation='softmax'),
```

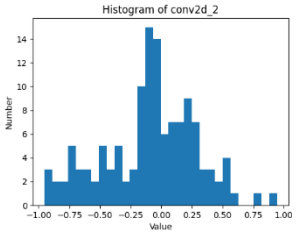
每一層的 weight 和 bias 分布狀況如下圖(左為 weight、右為 bias)，而學習曲線以及訓練集和驗證集之正確率變化如下圖所示。訓練集正確率 98.9%、驗證集正確率 74.95%，使用測試集之正確率則較低為 55.48%。

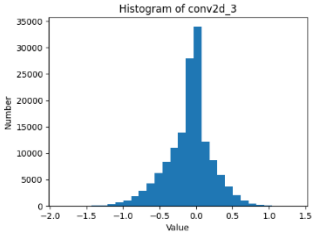


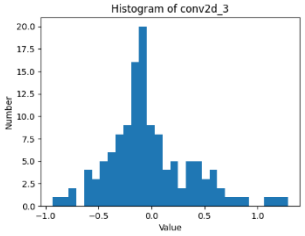


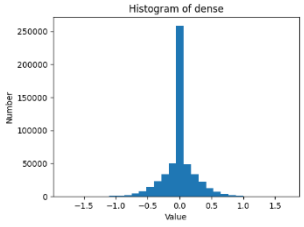


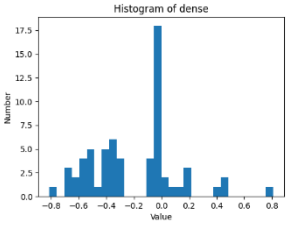


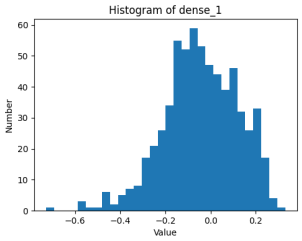


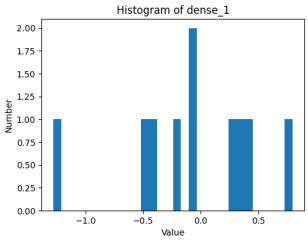


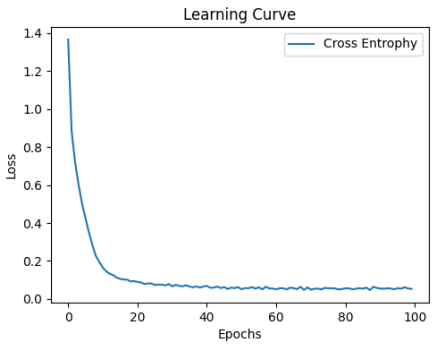


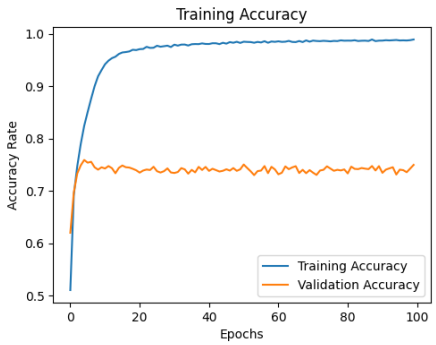




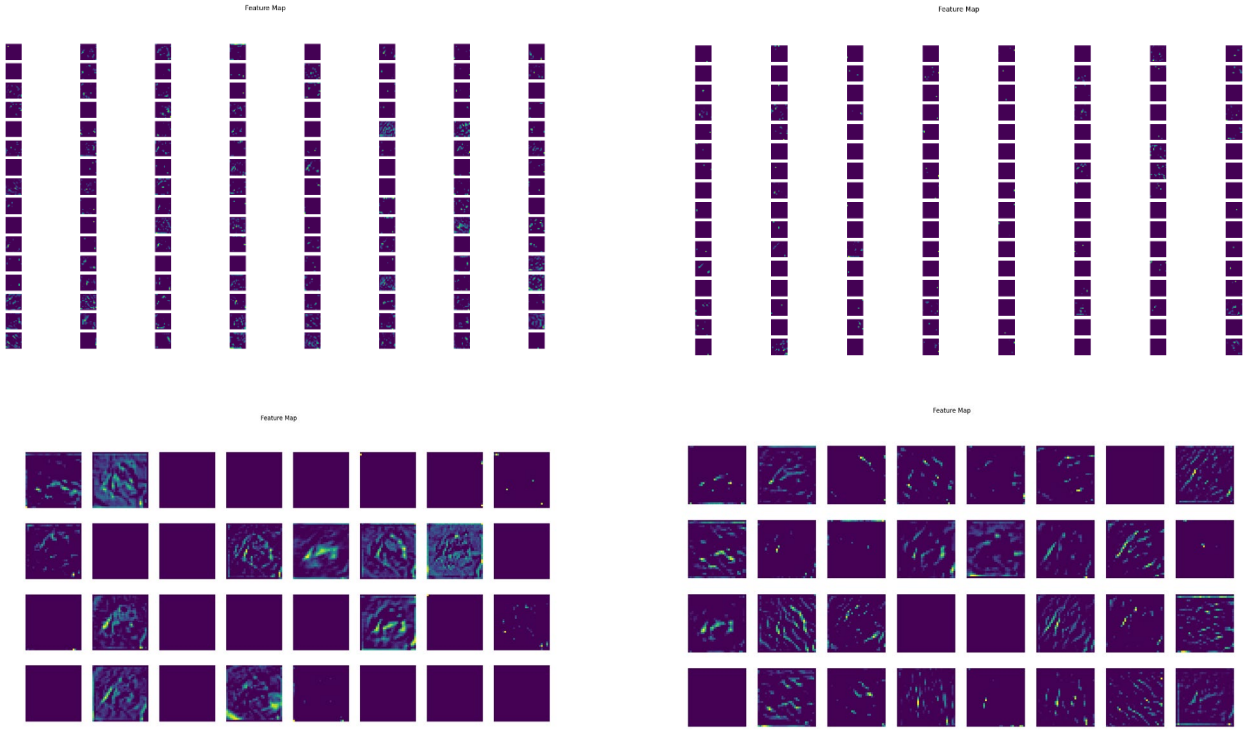








測試集經過所訓練之模型預測後，將預測正確與失敗的情況輸出如上圖。



訓練完成後之特徵圖則如上圖所示，filter size 皆為 3x3，同樣可以觀察到，較前面層數的特徵圖取得如線條或紋理的低級特徵，到深層深度的特徵圖，較無法看到物理上的實際特徵。

此外，在全連接層使用正規化之結果如第一題之文字辨識，訓練集正確率 97.94%、驗證集正確率 75.83%，但測試集之正確率卻從從 55.48%提升為 55.97%，泛化性達到提升效果。