自動駕駛實務 Autonomous-Driving
姓名 / 學號：何宇浩 VK6112026

辨識德國交通號誌。

使用資料集：Kaggle- GTSRB - German Traffic Sign Recognition Benchmark
原資料集分為兩個資料夾 Test 及 Train，其中 Test 有 12630 張照片，而 Train 有 39209 張照片，共 43 個類別。

資料前處理：

資料前處理主要分為資料讀取及資料分類兩大部分。

1. 資料讀取：

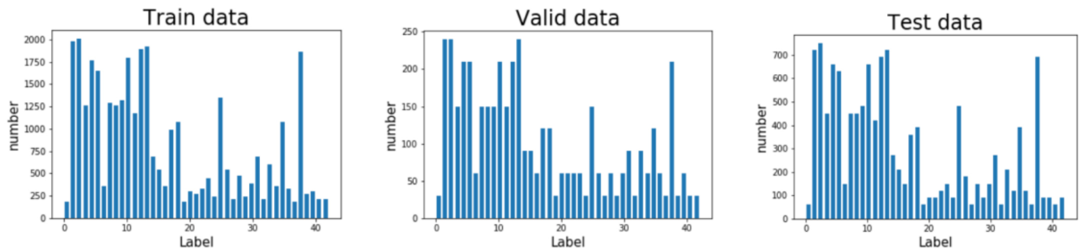
同時讀取影像及所屬類別，並調整影像大小，使得所有影像尺寸一致。

2. 資料分類：

總共分為訓練集、驗證集及測試集三類，其中訓練集及驗證集的資料為原 Train 資料夾中的照片隨機分配，訓練集的照片量佔 80% 而驗證集的照片佔 20%，最終三類之數據量如下所示：

```
Training data :  (31367, 32, 32, 3)
Validation data :  (7842, 32, 32, 3)
Testing data :  (12630, 32, 32, 3)
```

影像大小皆為 32*32，且皆為彩色圖像，因此有 RGB 共 3 層。
而此三個資料集的資料分布如下圖所示：



由上圖可知，各類別的數據量並不平均，但三個資料集數據的分布情形大致相同。

模型建立：

參考課堂簡報中 LeNet 5 模型，建立了兩種模型，以相同的數據對其進行訓練，探討各層的作用，並進行模型比較。
首先，介紹模型中各層的作用如下：
Convolution Layer (卷積層)：

將原始圖片的與特定的 Feature Detector(filter)做卷積運算，以提取圖像中的各種特徵，如形狀、顏色、輪廓、對比等特徵。

Max Pooling Layer (最大池化層)：

只提取矩陣當中的最大值，減少卷積特徵的空間大小，利用降維來降低處理數據所需的計算能力，有助於提取旋轉、位置不變等特徵，使模型有好的抗雜訊功能。

Batch Normalization：

對輸入數據進行正規化，用於規範輸入數據的分佈，有助於提高模型的穩定性和訓練速度。

Dropout：

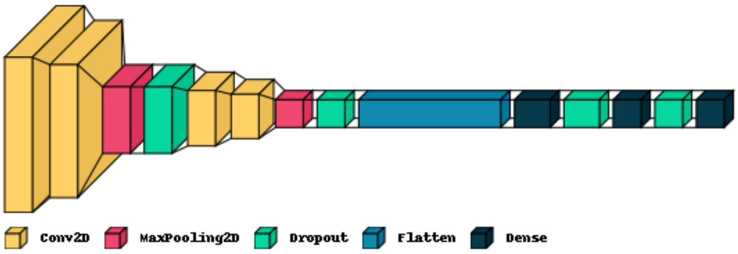
隨機丟棄部分比例的輸入單元，用於減少過擬合。

Fully Connected Layer (全連接層)：

將之前的結果平坦化後，接到最基本的神經網絡。

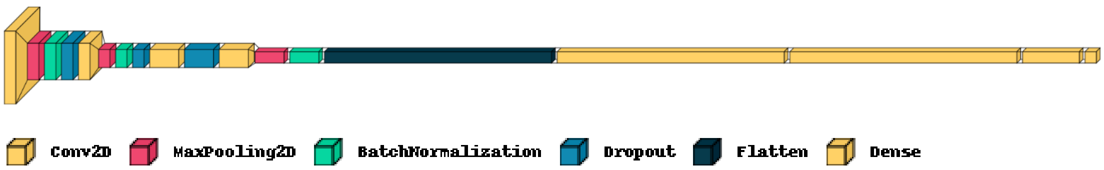
Model Structure :

Model 1.




```
# defining model structure
model = models.Sequential()
model.add(layers.Conv2D(filters = 32, kernel_size = (5,5), activation ='relu', input_shape = (32,32,3)))
model.add(layers.Conv2D(filters = 32, kernel_size = (5,5), activation = 'relu'))
model.add(layers.MaxPool2D(pool_size = (2,2)))
model.add(layers.Dropout(rate = 0.25))

model.add(layers.Conv2D(filters = 64, kernel_size = (3,3), activation ='relu'))
model.add(layers.Conv2D(filters = 64, kernel_size = (3,3), activation ='relu'))
model.add(layers.MaxPool2D(pool_size = (2,2)))
model.add(layers.Dropout(rate = 0.25))

model.add(layers.Flatten())
model.add(layers.Dense(256, activation = 'relu'))
model.add(layers.Dropout(rate = 0.25))
model.add(layers.Dense(128, activation = 'relu'))
model.add(layers.Dropout(rate = 0.25))
model.add(layers.Dense(43, activation= 'softmax'))

# model compilation
model.compile(loss = 'categorical_crossentropy', optimizer ='adam', metrics =['accuracy'])
model.summary()
```

Model 2.



```
model = models.Sequential() #Sequential Model

#ConvLayer(64 filters) + MaxPooling + BatchNormalization + Dropout
model.add(layers.Conv2D(filters=32,kernel_size=3,activation=’relu’,padding=’same’,input_shape=X.shape[1:]))
model.add(layers.MaxPool2D(strides=2))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.3))
#ConvLayer(128 filters) + MaxPooling + BatchNormalization + Dropout
model.add(layers.Conv2D(filters=128,kernel_size=3,activation=’relu’,padding=’same’))
model.add(layers.MaxPool2D(strides=2))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.3))
#ConvLayer(512 filters) + Dropout + ConvLayer(512 filters) + MaxPooling + BatchNormalization
model.add(layers.Conv2D(filters=512,kernel_size=3,activation=’relu’,padding=’same’))
model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(filters=512,kernel_size=3,activation=’relu’,padding=’same’))
model.add(layers.MaxPool2D(strides=2))
model.add(layers.BatchNormalization())

#Flatten
model.add(layers.Flatten())
#2 Dense layers with 4000 hidden units
model.add(layers.Dense(4000,activation=’relu’))
model.add(layers.Dense(4000,activation=’relu’))
#Dense layer with 1000 hidden units
model.add(layers.Dense(1000,activation=’relu’))
#Softmax layer for output
model.add(layers.Dense(43,activation=’softmax’))

model.summary()
```

Model Summary

Model 1

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 28, 28, 32)	2432
conv2d_5 (Conv2D)	(None, 24, 24, 32)	25632
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 32)	0
dropout_3 (Dropout)	(None, 12, 12, 32)	0
conv2d_6 (Conv2D)	(None, 10, 10, 64)	18496
conv2d_7 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_3 (MaxPooling2D)	(None, 4, 4, 64)	0
dropout_4 (Dropout)	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 256)	262400
dropout_5 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_6 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 43)	5547

Total params: 384,331
Trainable params: 384,331
Non-trainable params: 0

Model 2

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
batch_normalization (BatchNormalization)	(None, 16, 16, 32)	128
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 128)	36992
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
batch_normalization_1 (BatchNormalization)	(None, 8, 8, 128)	512
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_2 (Conv2D)	(None, 8, 8, 512)	590336
dropout_2 (Dropout)	(None, 8, 8, 512)	0
conv2d_3 (Conv2D)	(None, 8, 8, 512)	2359808
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 512)	0
batch_normalization_2 (BatchNormalization)	(None, 4, 4, 512)	2048
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 4000)	32772000
dense_1 (Dense)	(None, 4000)	16004000
dense_2 (Dense)	(None, 1000)	4001000
dense_3 (Dense)	(None, 43)	43043

Total params: 55,810,763
Trainable params: 55,809,419
Non-trainable params: 1,344

模型訓練：

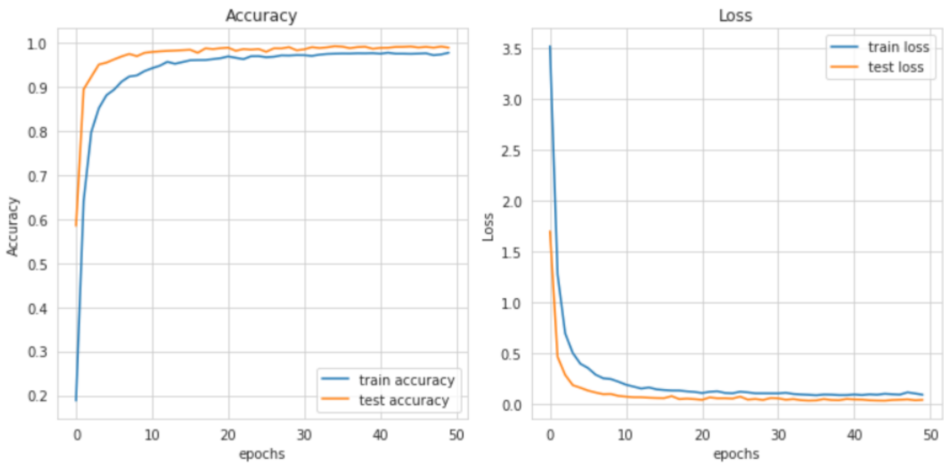
```
model.compile(optimizer='adam',
                           loss='categorical_crossentropy',
                           metrics=['accuracy'])

history=  model.fit(X_train,Y_train,
                                    epochs=15,
                                    batch_size=64,
                                    validation_data=(X_val,Y_val))
```

使用 compile 函式來編譯模型，並設定模型的優化器(optimizer)為 Adam，損失函數(loss function)為 categorical_crossentropy，並指定度量指標(metrics)為準確率(accuracy)。編譯模型是為了準備模型進行訓練前的配置。

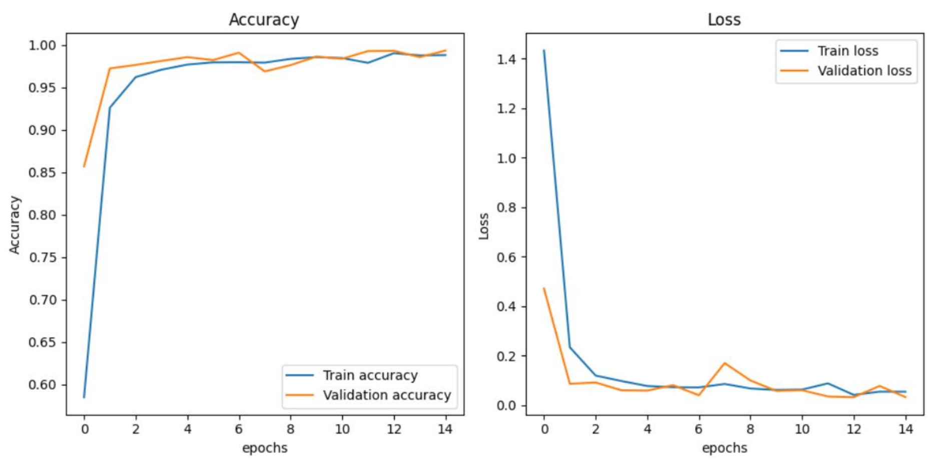
訓練過程中，模型將根據訓練數據進行反向傳播和權重更新，同時計算訓練損失和準確率。驗證數據用於在每個訓練迭代(epoch)結束後評估模型的性能。訓練的過程將返回一個 history 物件，其中包含了每個迭代的損失和準確率的記錄。

Training results :

Model 1.



Model 2.



模型測試結果討論：

將 testing data 丟入訓練好的模型中進行辨識並獲得準確率。

```
pred = np.argmax(model.predict(X_test),axis =1)
print("Test accuracy: ", accuracy_score(labels_test, pred) * 100 )
```

Model 1 :

```
Test accuracy:  95.37608867775138
```

Model 2 :

```
Test accuracy:  96.22327790973871
```

結果討論：

model 2 較 model 1 增加了更多層數，提取較多特徵，使得 model 2 模型所需訓練的參數量遠大於 model 1，但增添了 Batch Normalization 以提高模型的穩定性和訓練速度，因此在訓練模型時的收斂速度較 model 1 快。

以 testing data 的測試結果來看，model 2 的準確率略大於 model 1，但其差異並不大。

隨機列印圖像及辨識結果：