國立成功大學

自動駕駛實務 Project 2報告

German Traffic Signs Classifier

學生: 陳育政

學號: E24094198

指導教授: 傅志雄、莊智清

大綱

1. **前言**
2. **實作方法**
3. Load the Data
4. Dataset Summary & Exploration
5. Pre-process Dataset

* Convert to Grayscale
* Data Augmentation
* Normalize Dataset

1. Train the LeNet5 Model
2. Accuracy Curves, Confusion Matrix and Classification Report
3. Test LeNet5 model on New Images

**三、問題討論**

1. The Necessity of Data-Preprocessing

**四、結論**

**五、參考資料**

1. **前言**

Project 2的目的是試圖利用由Yann LeCun於1995年所提出的LeNet5模型架構去解決German Traffic Sign Recognition Benchmark(GTSRB)的問題，而GTSRB資料集共有43種不同情況的德國交通號誌RGB相片，例如:限速、右轉、禁止通行…等圖示。整體處理資料的pipeline包括資料前處理、建構訓練LeNet5模型以及測試模型的準確率，最後的目標就是訓練好的LeNet5模型可以針對不曾餵入訓練的德國交通號誌照片成功辨認種類，達成自動駕駛中不可或缺的辨識路標的功能。

1. **實作方法**
2. Load the Data

根據Fig. 1.的程式碼讀取p-code file，包括training、validation及test的dataset。

**A screen shot of a computer code

Description automatically generated**

Fig. 1. 讀取資料集

1. Dataset Summary & Exploration

印出有關dataset的基本資料，包括各資料集比數、圖片大小以及交通號誌種類數等數據，結果如Fig. 2.所示。

A screen shot of a computer

Description automatically generated

Fig. 2. 資料集的基本資料

除了印出dataset的基本資料，我還嘗試隨機挑選幾張資料集內的圖片顯示，讓我對於即將處理的資料有進一步的認知，結果如Fig. 3.所示，可以發現影像都是RGB，並且大小一致。

A collage of different road signs

Description automatically generated

Fig. 3. 隨機挑選8張traffic sign images觀察

接著，為了更好的了解資料集中的交通號誌種類分布，我將每個資料集的各個class的總數以直線圖呈現，training、validation、test dataset的直線圖依序由Fig. 4. (a)、(b)、(c)呈現。

A graph of blue and white bars

Description automatically generatedA graph of blue lines

Description automatically generatedA graph of blue lines

Description automatically generated

Fig. 4. (a)training dataset各class資料筆數 (b)validation dataset各class資料筆數 (c)test dataset各class資料筆數

1. Pre-process Dataset

* Convert to Grayscale

根據Fig. 5.的程式碼，我將原先資料集內的RGB影像皆轉成灰階圖，grayscale的效果如Fig. 6.呈現，而轉成灰階的原因不僅僅可以減少模型參數，讓訓練更快速，還可以讓模型專注在交通號誌圖示的輪廓差異，而非注意可能隨著光線而有顯著差異的RGB色階。

A computer screen shot of a program code

Description automatically generated

Fig. 5. 將RGB images轉成grayscale images的程式碼

A collage of different road signs

Description automatically generated

Fig. 6. Traffic sign images轉成灰階的效果

* Data Augmentation

藉由觀察Fig. 4.的資料集各class總數可以發現不同class之間的數量差異相當顯著，而這將會使訓練的效果因為資料的不平均性而不如預期，為了使資料筆數較少的class增加一些數據，我採用了常見的data augmentation，藉由平移、旋轉及縮放…等技術，在基於原先的資料下，產生更多筆訓練資料，達到各class的資料數平均的目標。

其中，我使用Affine Transformation、Perspective Transformation及Rotation共3種data augmentation的技術增加train dataset資料數，確保每個class都至少有3000筆資料。而3種data augmentation技術的code implementation可依序參考Fig. 7, Fig. 8及Fig. 9.。另外我也將data augmentation產生的圖片和原先的圖片顯示出來做比較，可參考Fig. 10.發現影像有輕微更動，但卻不失去本身原有的特徵。

A computer code with text

Description automatically generated with medium confidence

Fig. 7. Affine Transformation實作程式碼

A computer code on a black background

Description automatically generated

Fig. 8. Perspective Transformation實作程式碼

A computer code on a black background

Description automatically generated

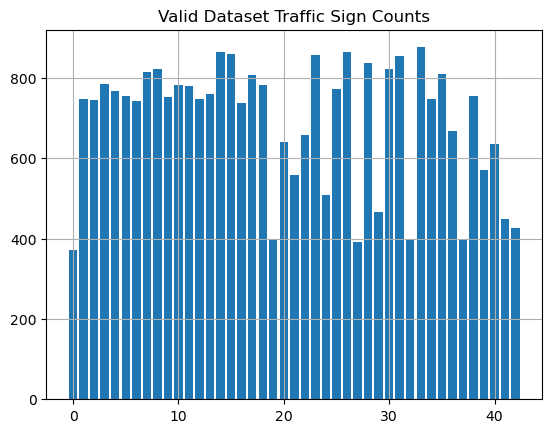
Fig. 9. Rotation實作程式碼

A close-up of a clock

Description automatically generated

Fig. 10. 右圖為原本資料集的圖片，而左側則為data augmentation產出的圖片

針對原有的資料做完data augmentation之後，我再次把dataset的各class資料數以直線圖呈現，更新過的training、validation dataset的直線圖依序為Fig. 11. (a)、(b)。我們可以發現training及validation的資料數相較於Fig. 4.因為使用data augmentation技術，各class的資料筆數都顯著提升，且class之間的數量平均。

A graph of blue lines

Description automatically generated

Fig. 11. (a)data augmentation後的training dataset各class資料筆數

(b) data augmentation後的validation dataset各class資料筆數

* Normalize Dataset

對原先的資料集images做完grayscale轉換和data augmentation之後，我又把資料做Normalization正規化的處理，將所有pixel value按比例轉換至[-1, 1]區間，詳細程式碼如Fig. 12.所示。

A computer screen with text and numbers

Description automatically generated

Fig. 12. 對資料集圖片做Normalization正規化

1. Train the LeNet5 Model

我先依據Yann LeCun所提出的LeNet5架構建構出一個模型，詳細的LeNet5參數及架構如Fig. 13.所示，並開始訓練這個LeNet5 model。我把batch size設為128，epochs設為200次，為了防止不必要的多餘訓練造成over-fitting的反效果，也有另外設定Early-Stopping的限制，詳細的訓練model的程式碼(包括Early-Stopping及optimizers的設置)可參考Fig. 14.。至於不同的optimizer一定也會對model的訓練效果有不同的影響，為了找出較適合的optimizer，我將嘗試RMSProp、Adam、Adagrad及SGD共4種optimizer，最後的訓練結果將呈現於Table 1.。

A screenshot of a computer program

Description automatically generated

Fig. 13. LeNet5詳細架構

A computer code on a black background

Description automatically generated

Fig. 14. Model training程式碼(包括Early-Stopping及optimizers的參數設定)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Different Optimizers’ Effect on GTSRB Dataset** | | | | |
| Optimizer Name | ***Adam*** | ***RMSProp*** | ***SGD*** | ***Adagrad*** |
| Training Epochs | 17 | 12 | 24 | 112 |
| Training Accuracy | 0.97 | 0.96 | 0.97 | 0.95 |
| Validation Accuracy | 0.96 | 0.96 | 0.97 | 0.95 |
| Prediction Accuracy | 0.93 | 0.93 | 0.93 | 0.91 |
| **Note: Adagrad的訓練所需epochs最多且accuracy最低，因此之後將排除該optimizer** | | | | |

Table 1. 考慮不同optimizers對於GTSRB辨識的表現差異

觀察Table 1.的結果可以發現Adagrad的training epochs遠比其他三個optimizers高，代表訓練所需花的時間相對多，accuracy也不如其他三者高，因此可以將其從最終的optimizer選擇名單中汰除。至於Adam、RMSProp及SGD的accuracy大致上沒有太大的差別，因此我接下來將再想其他方法嘗試是否能提高準確率，並由這三個optimizers中試圖找出最適合此次project的LeNet5 model。

在參考網路上的文章及詢問ChatGPT的意見後得到了可以嘗試learning rate scheduler的概念，learning rate會依據epochs或是accuracy的變化動態更新learning rate的值，於是我試著增加名為ReduceLROnPlateau的callback，但效果並沒有像預期的一樣變好，training和validation的accuracy的確有增加，但是testing accuracy依舊卡在0.93，無法再提高，因此這個方法只能作罷。

接著，我又考慮了把原始images的對比度加強，因為我注意到原始images的解析度都不太好，甚至有些看不清楚交通號誌的詳細內容。於是我上網搜尋了加強解析度的方法，發現了有一個叫做Contrast Limited Adaptive Histogram Equalization(CLAHE)的演算法，可以加強灰階圖的對比度，而且OpenCV已經有內建好的function可以直接使用，因此我便針對dataset多增加CLAHE的前處理動作，程式碼呈現於Fig. 15.。

A computer screen shot of text

Description automatically generated

Fig. 15. CLAHE加強灰階圖對比度的程式碼

針對CLAHE的parameters調整，我分別嘗試(4,4)、(6,6)及(8,8)共3種不同大小的tileGridSize，optimizer先預設為Adam，詳細的accuracy比較可參考Table 2.。

|  |  |  |  |
| --- | --- | --- | --- |
| **Different CLAHE tileGridSize’s Effect on Accuracy** | | | |
| tileGridSize | ***(4,4)*** | ***(6,6)*** | ***(8,8)*** |
| Training Accuracy | 0.98 | 0.98 | 0.97 |
| Validation Accuracy | 0.97 | 0.97 | 0.96 |
| Prediction Accuracy | 0.95 | 0.94 | 0.94 |
| **Note: optimizer先預設為Adam，其餘程式碼皆無作調整，只有多增加CLAHE加強對比** | | | |

Table 2. 考慮不同tileGridSize對於CLAHE的效果表現

基於Table 2.的結果，最終我選擇了(4,4)大小的tileGridSize。目前的prediction accuracy已經由最初的0.93進步到0.95，我又想到之前用Data Augmentation增加更多的data，那如果我再讓其產生的資料更多應該會讓訓練效果更好。於是我把原先的每個class至少有3000筆資料改成至少有5000筆，期望可以提高準確率，optimizer一樣先預設Adam，而訓練出的結果也的確使prediction accuracy增加至0.96，後來我還有試著讓每個class有7000筆資料，但prediction accuracy依然保持在0.96，因此最終就決定讓每個class至少有5000筆資料。最後，我針對目前的狀態討論最後optimizer要從Adam、RMSProp和SGD選何者比較適合，結果呈現於Table 3.。

|  |  |  |  |
| --- | --- | --- | --- |
| **Final Decision of Optimizers** | | | |
| Optimizer name | ***Adam*** | ***RMSProp*** | ***SGD*** |
| Training Accuracy | 0.98 | 0.97 | 0.97 |
| Validation Accuracy | 0.97 | 0.97 | 0.97 |
| Prediction Accuracy | 0.96 | 0.95 | 0.94 |
| **Note: 最終選擇Adam作為optimizer(learning rate固定為0.001)** | | | |

Table 3. 在使用CLAHE和增加更多資料後，不同optimizer的準確率

1. Accuracy Curves, Confusion Matrix, and Classification Report

最後，我的model訓練模式及參數設定已經確定，所有相關資

將列在Table 4.，而最後訓練結果的Accuracy/Loss Curve(Fig. 16.(a)、(b))、Confusion Matrix(Fig. 17.)及Classification Report(Fig. 18.)將依序列在下方。

|  |  |
| --- | --- |
| **Final Model and Other Parameters Configure** | |
| **Preprocessing Techniques** | Grayscale Conversion  Data Augmentation  CLAHE Contrast Enhancement  Dataset Normalize |
| **NN Structure** | LeNet5 |
| **Callbacks** | Early Stopping |
| **Optimizer** | Adam(lr=0.001) |
| **Batch Size** | 128 |
| **Training Epochs** | 18(大約4min) |
| **Prediction Accuracy** | 0.96 |

Table 4. 最終選用的model、相關參數及前處理技術的總整理

A graph of training and validation

Description automatically generatedA graph of a line

Description automatically generated with medium confidence

Fig. 16. (a) Accuracy Curves (b) Loss Curves

A graph of a graph showing a line

Description automatically generated with medium confidence

Fig. 17. Confusion Matrix of LeNet5 prediction result

A black background with numbers and numbers

Description automatically generated

Fig. 18. Classification Report

1. Test LeNet5 model on New Images

以下將針對訓練完的model用於預測沒看過的10張照片做測試，

檢視這個model的廣泛性是否足夠。我先使用Fig. 19.的程式碼把testImages/內的照片讀出，並同樣做前處理(grayscale conversion、CLAHE及normalize)，最後處理完的影像如Fig. 20.所示。

A screen shot of a computer program

Description automatically generated

Fig. 19. 讀出testImages內影像，並做前處理

A group of road signs

Description automatically generated

Fig. 20. 前處理後的影像

接著，我根據已知的43類traffic signs，手動把testImages的影像對應的label輸入至test\_labels中，如Fig. 21所示，最後用evaluate function評估的表現呈現於Fig. 22.，準確率為0.9。仔細比對後，可以發現是最後一個應該是第25種的road work，而反而被誤判為30的Beware of ice/snow。我經過觀察發現應該是因為解析度太低，而兩者的圖案輪廓在模糊的狀態下看起來又十分接近，因而有誤判的情況產生。

A computer screen shot of text

Description automatically generated  
Fig. 21. 手動輸入label，並用evaluate評估模型表現

A screen shot of a computer

Description automatically generated

Fig. 22. 預測準確率為0.9，誤判的traffic sign為編號25的road work

1. **問題討論**
   1. The Necessity of Data-Preprocessing

從這次的project實作中，我體認到data-preprocessing對於

machine learning的訓練是不可或缺的，因為這次使用到的前處理技術

對於model training的成果提升有顯著的幫助，data augmentation可以讓我在資料不足或不平均的狀況下，產生足夠的資料訓練model，避免過低的accuracy或是偏向特定class的預測結果。Normalization也使model在訓練過程的收斂速度較快，至於CLAHE則是在GTSRB dataset的影像大多模糊不清的狀況下，最大限度地加強影像對比度，使原本看起來輪廓類似的影像可以被正確區分開來。

1. **結論**

在這次的Project我使用了LeNet5搭配一些資料前處理的技巧去

訓練並預測GTSRB資料集共分為43類的交通號誌影像，最終達到的

預測準確率可以達到96%。至於未來若是要提高GTSRB的辨識率，

我認為可以改變不同的NN架構，例如常見的ResNet、VGG16及

GoogLeNet…等。而我認為提升準確率也可以從另一個方向下手，就是著重在影像處理，試圖利用一些演算法，針對模糊不清的影像加強輪廓及對比度，像這次我使用的CLAHE就是其中一個方法。

最後，以下將附上我該次Project的GitHub Link，裡面有附上原始的Jupyter Notebook程式碼檔案。

GTSRB Project Repo: [Project\_2\_GitHub\_Link](https://github.com/urbao/autonomous-driving/tree/master/HW3)

**五、參考資料**

1. [https://github.com/Goddard/udacity-traffic-sign-classifier/](https://github.com/Goddard/udacity-traffic-sign-classifier/tree/master)
2. [data-augmentation-explanation](https://medium.com/mlait/affine-transformation-image-processing-in-tensorflow-part-1-df96256928a)
3. [perspective-transformation-explanation](https://rumn.medium.com/yolo-data-augmentation-explained-turbocharge-your-object-detection-model-94c33278303a)
4. [learning-rate-scheduler-idea](https://ithelp.ithome.com.tw/articles/10298468)
5. <https://en.wikipedia.org/wiki/Adaptive_histogram_equalization>
6. [CLAHE-OpenCV-docs](https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html)