Road Traffic Sign Image Classification
Using Multi-Layer Perceptron Neural Networks (MLPs)
And Convolutional Neural Network (CNNs)

Student 1: Ngoc Bao Vy Le
s3828276
Student 2: Minh Hoang Nguyen
S3712611
May 2022

1 Introduction

There has been a growing research interest in traffic sign recognition (TSR), driven by the growing popularity of autonomous driving vehicles, advanced driver assistance, and mobile mapping in recent years [1]. The traffic sign recognition problem consists of traffic sign localization, to locate traffic signs within a video frame, traffic sign classification, to categorise traffic signs into predefined groups, and real world traffic sign detection [2]. This paper will focus on exploring traffic sign classification and the evaluation of certain neural network models, which could be used to correctly label traffic signs.
The paper will go through related state of the art works in this field in Sect. 2, providing information about the dataset being used in Sect. 3, and then present the neural network models that were experimented in Sect. 4. Results and the ultimate judgement are addressed in Sect. 5 and 6.
There are two traffic sign dataset being used in this research which are the modified Belgium Traffic Sign Recognition Benchmark (BTSRB) dataset, which were splitted into 60% training set, 20% validation set and a 20% test set, and a smaller Australian Traffic Sign (ATS) dataset, which was collected from the internet with the main purpose as to evaluate the trained model in a real world scenario. *An assumption that the trained models will perform better on the 20% extracted BTSRB test set than on the ATS dataset, due to the differences between Australian traffic signs and Belgium traffic signs, was made*.
The two main classification tasks that these trained models need to achieve are to classify images according to the sign-shape (For example: diamond, hex, rectangle, round, etc) and sign-type (stop, warning, parking, etc).

2 Related Work

There has been numerous research works being conducted on traffic sign image classification. The study in [3] describes an approach that has used CNNs/MLPs for traffic sign recognition that achieve a recognition performance rate of 98.98% on the German Traffic Dataset, which is better than human recognition, and by further training, up to 99.15% ; the study in [5] developed another traffic sign recognition algorithm which was a combination of colour transformation and CNNs; with an assumption that through incorporating data augmentation, the model would be able to achieve better performance; etc. Although these research works are exploring different problems in traffic sign classification, most of them implement their experiments based on either MLPs or CNNs, which is the motivation behind why this project is being done to experiment with these neural networks.

3 Traffic Sign Dataset

The trained models in this research were evaluated using two datasets, the modified Belgium Traffic Sign Recognition Benchmark (BTSRB) dataset and the Australian Traffic Sign (ATS) dataset. The specifications for these two datasets are provided in **Table 1** below.
It is crucial to notice that the data provided in the BTSRB dataset experiences severe imbalance, which could affect the performance of the model being trained on this dataset. The lowest distributed sign-shape is *hex*, which only corresponds to nearly 2% of the whole dataset, while *round* corresponds

to nearly 48%. The same imbalance is experienced with sign-type, where *warning* corresponds to nearly 19% of the dataset, while *crossing* only corresponds to 2.5%.
In addition, as mentioned earlier, the shape and type between these 2 datasets are also largely different (refer to **Appendix A** for details).

4 Experimental Setup

The experiment conducted in this paper is performed on Google Colab, with Intel(R) Xeon(R) CPU @ 2.20GHz. The experiment was set up to evaluate the performance of three based models, a multiple layer perceptrons neural network versus two popular convolutional neural network architectures which are the VGGnet architecture and a simplified LeNet architecture. The base architectures were trained using part of the BTSRB dataset (60% of the original data) and modified depending on its performance on the validation set (20% of original data from BTSRB dataset) and a summary and the reason behind these modifications are presented in this section (please refer to **Appendix B** for detailed model performances on training and validating).

Multiple Layer Perceptrons

The base MLP model is a classic 3 layers MLP consisting of input, output layers and one hidden layer. The hidden layer has 32 perceptrons with sigmoid as an activation function, whereas the output layer has softmax as an activation function. The model performs well on the training set with an accuracy of roughly 95%, however, only gets an accuracy of roughly 90% on the validation set. This means that this model experiences a slight overfit. In addition, the model only starts to converge after 100 epochs. The purposes of parameter tuning would be to find a balance between further improving the accuracy on the training set, as well as reducing overfit and improving convergent time.
The base model was saved as *based_MLP_model* to be evaluated on the test data.

Class weight, hidden layers and perceptrons

With the purpose of improving the accuracy of the training set, hidden layers and perceptrons were added to the base model. The numbers of hidden layers were increased to 2 and the numbers of perceptrons were increased to 128. In addition to adding layers and perceptrons, class weights were also considered in this tuned model since the training data experiences quite severe imbalance. Although the performance on the training set did improve as expected to nearly 100%, the model also takes longer to converge.
This is the reason why *Relu* activation function was experimented, since *Relu* is not subjected to the problem of vanishing gradient as *Sigmoid* activation function is, despite the fact that the base network is very shallow. Moreover the gap between the performance on the training set and validation set also increased. For this reason, different techniques to reduce overfitting were also employed.

Rectified linear unit (Relu)

When *Relu* was experimented as an alteration to the *Sigmoid* function, the model tended to converge much faster. This observation is consistent with relevant studies on the performance of *Relu* versus *Sigmoid* [6]. An explanation for this observation would be due to the computational requirement to calculate *Sigmoid*, whereas this calculation is not required for *Relu.* Additionally, the model employing *Relu* also shows better performance on both training and validating sets in terms of accuracy, which leads to the decision to keep *Relu* as the activation function for later exploration.

Dataset	Size	Colour	Numbers of Image	Numbers of Shape	Numbers of Sign
BTSRB	28 * 28	Greyscale	7430	5	16
ATS	28 * 28	Greyscale	135	5	16

Table 1: Dataset Specifications

Ridge Regularisation, Dropout, and Early Stopping

The three techniques above were experimented to see whether the model can avoid the problem of overfitting.

Two regularisation penalty values of 0.001 and 0.0001 were experimented on the MLP model, however, the model still appears to be overfit, despite that the effects of these regularisation can be seen in several early epochs where there were accuracies dropped.

A dropout rate of 0.2 was then chosen, as this is a shallow network, to be applied to the hidden layer of the base model (not to the output layer of the model as the model might lose crucial information if dropout is applied this way). The idea behind dropout is that at every training step, some proportion of neurons will be ignored. Although the application of *dropout* has solved the problem of overfit, it also affects the performance of the model, which leads to the decision of employing early stopping technique instead.

Early stopping was applied with the better *Ridge regularisation* value which is 0.0001. When these two techniques were implemented together a model with an accuracy of roughly 90% on both the training and validating set was produced.

After this step, a model which was referred to as *fine_tuned_MLP_model* was saved to be evaluated on the testing dataset.

Data Augmentation

Acknowledging that the training dataset is imbalanced, data augmentation was used to create a new training dataset. Image processing has been applied to classes where there are not many data points. The best architecture, after overfitting the handling step, was trained using this new dataset and was saved for comparison as *model_MLP_augmented*.

Convolutional Neural Network

Experiments were conducted on two very popular convolutional neural network architectures which are the *VGG* and the *LeNet,* as the team believes that, despite the fact that these often worked well, there are still areas that have to be improved within these architectures to suit the purpose of this project (refer to source code for detailed model summaries). The two base models were saved as *LeNet_base_model* and *VGG_base_model* to be evaluated on the test set.

According to observations, the *LeNet* base model performs really well on both sign-shape prediction and sign-type prediction. On the other hand, the *VGG* base model, although having a high accuracy on the training data, has a slightly lower accuracy on the validating data. For this reason, two techniques to reduce overfit were experimented, which are regularisation and dropout.

Ridge Regularisation

Three values of *Ridge regularisation* {0.001, 0.01, 0.1} were experimented using hyperparameter tuning and the best parameter which is 0.01 was kept. Although the gap between training accuracy and validation accuracy has been reduced after this step, dropout was still applied in the next step, expecting that this would further reduce overfit.

Dropout

Two values of *dropout_rate* {0.5, 0.2} were manually experimented, incorporating the best regularisation parameter which is 0.01. The result loss curve shows that, unlike in the case of MLP model, a high value of *dropout_rate* works better. By implementing a *dropout_rate* of 0.5, the difference between training and validating accuracy was reduced to 1%. For this reason, this model was saved as *VGG_tuneL2_DropOut_model_05* to be evaluated on the testing set.

5 Experimental Result

In this section, the paper will discuss and evaluate the performance of models that have been saved on the two testing/unseen datasets. These models include *based_MLP_model, fine_tuned_MLP_model, model_MLP_augmented, LeNet_base_model, VGG_base_model,* and *VGG_tuneL2_DropOut_model_05*. The evaluation will be broken down into two main sections which is between MLPs model without augmented data versus MLPs model with augmented data, and the best performing MLPs model with CNNs model. The main metrics of evaluation that this project will consider is accuracy, macro f1-score and f1-score on low data points class (*hex* class for sign-shape classification and *crossing* class for sign-shape).
Due to the limitation, only certain figures will be included in the report, please refer to **Appendix B** or the jupyter notebook for detailed results.

MLPs model without augmented data versus MLPs model with augmented data

Shape

According to **Table 2** below, although all three models do not show much differences in term of accuracy, there are a clear differences in term of macro f1 - score with *model_MLP_augmented* being the best performing model (97%); the second best performing model is *find_tuned_MLP_model*, which only achieved 92%. This observation is consistent with expectation, as the training dataset is very imbalanced. This is also the reason why *hex* class f1 - score is observed. While *model_MLP_augmented* only performs slightly worse on *hex* class compared to the macro f1 - score, the other two models perform badly on this class compared to their macro f1 - score. As all of the classes are equally crucial in this classification task, *macro f1 - score is the better metrics to indicate a better model than overall accuracy*.
Additionally, one can observe a significant drop in both accuracy and f1-score when these models were being tested on the ATS dataset. This observation is also consistent with the assumption stated in the first section that the differences between Australian traffic signs and Belgium traffic signs could lead to a difference in performance. *model_MLP_augmented* is still the best performing model on this ATS dataset with an f1-score of 60%. A point to notice is that this model also performs better on *hex* class than on the macro f1-score.

Type

The macro f1-score performances of *based_MLP_model* and *fine_tuned_MLP_model* on sign-type classification, experimenting on the BTSRB dataset, are similar to that of *model_MLP_augmented* (all at roughly 95%), however, the performances of these two models on *crossing* f1-score are still slightly lower than that of *model_MLP_augmented.* This means that conclusion from the previous section about how data augmentation can help improve model f1-score performance is valid. Moreover, *model_MLP_augmented* is still the best model when testing on the Australian dataset with an f1-score of 26%, compared to 19% from *fine_tuned_MLP_model* and 17% from *based_MLP_model*. *model_MLP_augmented* will then be selected to compare with CNNs models in the next section since it is the best performing MLPs model.

MLPs model versus CNNs model

Shape

According to the data from **Table 3**, *LeNet_base_model* is the best performing model in terms of overall accuracy and macro f1-score in both dataset. In addition, bot *VGG_base_model* and *VGG_tunelL2_DropOut_model_05* performs better than *model_MLP_augmented* in both metrics

Model	Accuracy (BTSRB/ATS)	Macro f1 - score (BTSRB/ATS)	f1 - score of *hex* class (BTSRB/ATS)
based_MLP_model	95%/29%	92%/34%	82%/0%
fine_tuned_MLP_model	96%/28%	92%/40%	75$/12%
model_MLP_augmented	97%/54%	97%/60%	95%/74%

Table 2: MLPs Shape Classification

Model	Accuracy (BTSRB/ATS)	Macro f1 - score (BTSRB/ATS)	f1 - score of *hex* class (BTSRB/ATS)
model_MLP_augmented	97%/54%	97%/60%	95%/74%
LeNet_base_model	99%/71%	99%/65%	100%/64%
VGG_base_model	98%/55%	97%/58%	95%/33%
VGG_tuneL2_DropOut_model_05	98%/54%	98%/50%	100%/33%

Table 3: MLPs vs CNNs Shape Classification

when being tested on BTSRB; when being tested on ATS, *model_MLP_augmented* performs slightly better.
This has shown that CNNs is the better performing model for this shape-sign classification task. An explanation for this behaviour is that when images are sent into an MLPs model, the image is flattened into a 1D array, while in an CNNs model, image remains 2D. This allows CNNs to be able to create a spatial relation between pixels that are close to each other. On the other hand, *model_MLP_augmented*, having the highest f1-score in predicting *hex* class on ATS dataset, proves that data augmentation can improve the accuracy on a specific smaller class. If data augmentation was to be implemented in the CNNs model maybe a higher macro f1-score could have been achieved.

Type

Similar observations have been observed from sign-type classification as the *LeNet_base_model* is still the highest performing model in terms of f1-score on both dataset 98% for the BTSRB dataset and 30% for the ATS dataset. This means that, although the problem of overfitting has been resolved in the *VGG_tunelL2_DropOut_model_05* model, *LeNet_base_model* is still more robust in this case.
It is crucial, however, to acknowledge that these architectures were tested on a really small and specific dataset.

6 Ultimate Judgement

Based on all observations above, the best model for this image classification task in this project is the model using the modified version of LeNet architecture that was provided from the lecture. This architecture has proven to be the best performing model when f1-score metric was considered. In addition, it is also the best performing model to the independent dataset.

7 Conclusion

In this project, several neural networks models have been evaluated, performing on two datasets which are the modified Belgium Traffic Sign Dataset Benchmark (BTSDB) and the self - collected Australian Traffic Sign (ATS). A model that was constructed based on the LeNet architecture is the best model for this classification task due to the fact that it has the best performance, based on f1-score metric, on both dataset. The project also shows that choosing the right dataset for training is really important and depends on the region, as different traffic signs from different countries have different attributes. Data augmentation also has been proved to be very crucial to an imbalanced dataset and can improve predicting accuracy if implemented correctly. Furthermore, the training and testing dataset that have been used to complete this project is a small dataset, so it is recommended that any further study should consider using a bigger dataset to validate those results that were found in this project.

Reference Lists

[1] Mathias, M., Timofte, R., Benenson, R., & Van Gool, L. (2013, August). Traffic sign recognition—How far are we from the solution?. In *The 2013 international joint conference on Neural networks (IJCNN)* (pp. 1-8). IEEE.

[2] Jurišić, F., Filković, I., & Kalafatić, Z. (2015, November). Multiple-dataset traffic sign classification with OneCNN. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 614-618). IEEE.

[3] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. InProceedings of International Joint Conference on Neural Networks (IJCNN). 2011

[4] D. Li, D. Zhao, Y. Chen and Q. Zhang, "DeepSign: Deep Learning based Traffic Sign Recognition," *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1-6, doi: 10.1109/IJCNN.2018.8489623.

[5] Y. Wu, Y. Liu, J. Li, H. Liu and X. Hu, "Traffic sign detection based on convolutional neural networks," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1-7, doi: 10.1109/IJCNN.2013.6706811.

[6] Waoo, A. A., & Soni, B. K. (2021). Performance Analysis of Sigmoid and Relu Activation Functions in Deep Neural Network. In *Intelligent Systems* (pp. 39-52). Springer, Singapore.

Appendix A: Differences between Belgium Sign and Australian Sign

Shape	BTSRB (Sign)	ATS (Sign)
Diamond	rightofway	rightofway, warning, laneend
Hex	stop	stop
Round	bicycle, litmitedtraffic, noentry, speed, noparking, trafficdirective, roundabout, traveldirection	bicycle, continue, litmitedtraffic, noentry, noparking, speed, trafficdirective, traveldirection
Square	continue, crossing, laneend, parking	crossing, parking
Triangle	Giveaway, warning	giveaway, roundabout

Appendix B: Detailed Model Performance

Dataset 1: BTSRB

Dataset 2: ATS

Shape

	Model	f1	precision	recall	accuracy	Dataset
3	Shape LeNet Base	0.990803	0.991231	0.990380	0.990667	Dataset 1
5	Shape VGG tuned L2 and DropOut	0.977488	0.977153	0.978130	0.978667	Dataset 1
4	Shape VGG Base	0.971373	0.960050	0.983772	0.977333	Dataset 1
2	Shape MLP Tuned with Data Augmented	0.965807	0.969674	0.963723	0.974667	Dataset 1
0	Shape MLP Base	0.924452	0.965772	0.894213	0.950667	Dataset 1
1	Shape MLP Tuned	0.917442	0.973092	0.882462	0.958667	Dataset 1
9	Shape LeNet Base	0.657010	0.769048	0.662965	0.711111	Dataset 2
8	Shape MLP Tuned with Data Augmented	0.580962	0.601861	0.700016	0.540741	Dataset 2
10	Shape VGG Base	0.501301	0.603915	0.528446	0.555556	Dataset 2
11	Shape VGG tuned L2 and DropOut	0.496844	0.607775	0.499487	0.503704	Dataset 2
7	Shape MLP Tuned	0.409953	0.549249	0.450016	0.414815	Dataset 2
6	Shape MLP Base	0.347990	0.307524	0.425673	0.385185	Dataset 2

Sign

	Model	f1	precision	recall	accuracy	Dataset
3	Sign LeNet Base	0.981531	0.981590	0.982468	0.985333	Dataset 1
5	Sign VGG Tune L2 and DropOut	0.970597	0.975600	0.967630	0.977333	Dataset 1
1	Sign MLP Tuned	0.952965	0.954741	0.953990	0.956000	Dataset 1
4	Sign VGG Base	0.951661	0.952201	0.953175	0.958667	Dataset 1
2	Sign MLP Tuned with Data Augmented	0.950419	0.952189	0.951150	0.953333	Dataset 1
0	Sign MLP Base	0.943613	0.951606	0.941579	0.952000	Dataset 1
9	Sign LeNet Base	0.303499	0.291373	0.336603	0.459259	Dataset 2
8	Sign MLP Tuned with Data Augmented	0.256244	0.248039	0.282982	0.407407	Dataset 2
10	Sign VGG Base	0.237560	0.234207	0.264031	0.385185	Dataset 2
11	Sign VGG Tune L2 and DropOut	0.234405	0.255576	0.270851	0.355556	Dataset 2
7	Sign MLP Tuned	0.198688	0.293655	0.222671	0.288889	Dataset 2
6	Sign MLP Base	0.176099	0.250951	0.216489	0.296296	Dataset 2