Traffic Sign Classification

Tianyu Bell Pan, Jiacheng Liu, Shuhao Zhang, Chengming Yang
Electrical and Computer Engineering Department (ECE), University of Florida, Gainesville, FL, USA;
Tourism, Hospitality, and Event Management Department (THEM), University of Florida, Gainesville, FL, USA
Email: tpan1@ufl.edu; jiacheng.liu@ufl.edu; shuhaozhang@ufl.edu; yangc1@ufl.edu

***Abstract*—Traffic sign classification has been an area of intense study for many years. In this paper, we evaluate the performance of our proposed CNN model. Continuous advances in processing power, image analysis, and machine learning enable the development of fresh performance-enhancing approaches. This study will first detail data preprocessing, including resize, data augmentation, and greyscale. Further, we will determine the best learning rate, optimizer, and BatchSize. Moreover, we explored how kernel amount and dropout impact model performance. In the end, we constructed the final model with the learning rate μ = 0.0003, Nadam optimizer, with dropout, and BatchSize = 32. The final model reaches 91% average accuracy.**

I. Introduction

Driver Assistance Systems (DAS) rely on traffic sign recognition to provide drivers with safety and precaution information, and traffic sign recognition systems traditionally consist of two phases: detection and classification [1]. Traffic signs are constructed with specific shapes and colors for safe driving to present drivers with important information such as traffic rules, route direction, and various road conditions. The primary goal of building an advanced driver assistance system is to prevent road accidents and poor decisions. The development of intelligent automobiles that can recognize environmental traffic signs is one of the hottest subjects in traffic sign detection systems today [2]. Several techniques have been proposed to classify traffic signs, such as Support Vector Machines (SVM), Neural Networks (NN), and Convolutional Neural Networks (CNN). [1] used K-d trees and random forests, [3] applied hierarchical support vector machines, [4] adopted the MicronNet, [5] used NNs, and the majority of others adopted CNNs.

These algorithms are frequently problem-specific, with distinct advantages and disadvantages that depend greatly on the experimental design. No single expert classifier currently produces expert results throughout the whole issue domain. With the rapid advancement of computer vision, new classification techniques may properly categorize even complicated objects such as bicycles, airplanes, chairs, small animals, etc. Unlike these objects, traffic signs with fixed colors and shapes appear easy to identify. However, a complicated external environment makes it difficult to recognize traffic signs [3]. As depicted in Figure 1, the key challenges are poor lighting conditions, rotation, same background color, partial occlusion, low image quality, motion blur, etc.

To systematically evaluate the categorization performance of different architectures, we implemented CNN on Graphics Processing Units (GPUs). CNNs can reliably classify objects regardless of their orientation; in particular, CNNs are insensitive to translation, viewpoint, size, and lighting. The remainder of the paper is organized as follows. Section 2 gives a brief description of our CNN. Section 3 discusses the creation of the training and test sets as well as the data preprocessing, then presents experimental results and shows how our CNN trained on raw pixels and measured performance. Section 4 concludes.

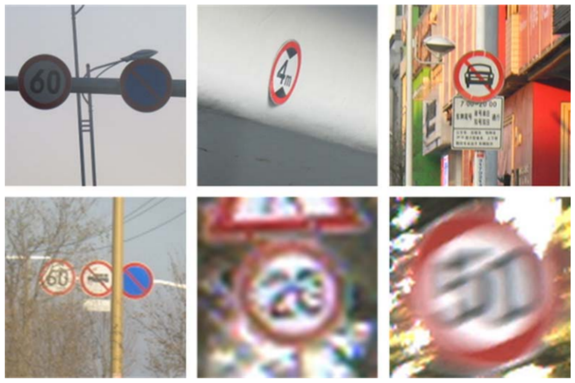


Fig. 1: Difficult cases for traffic sign classification [3]

II. Implementation

A convolutional neural network is a feed-forward neural network commonly employed for image-based classification, object identification, and recognition. The fundamental concept underlying CNN's operation is the use of convolution, which produces layered, filtered feature maps [2]. Fig. 2 provides the outline of a CNN, which includes convolution, pooling, and convoluted feature maps (output) [6].

Fig.3 shows the architecture of our CNN proposed for classifying traffic signs in this study.

A. Data augmentation

Deep learning networks often require large volumes of data. Many datasets have fixed data, therefore augmentation must be accomplished by changing the original data [7]. Data augmentation has been applied to many projects. For example, image classification uses simple transformations such as scaling, flipping images, or sampling subwindows of the images; and handwriting recognition uses slight affine transformations.

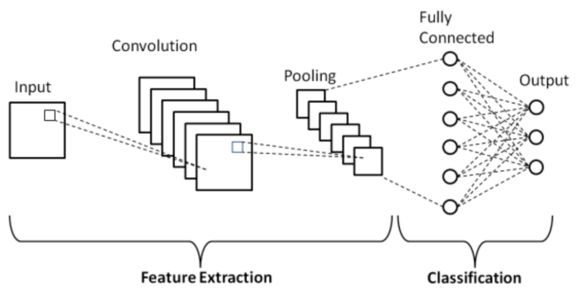
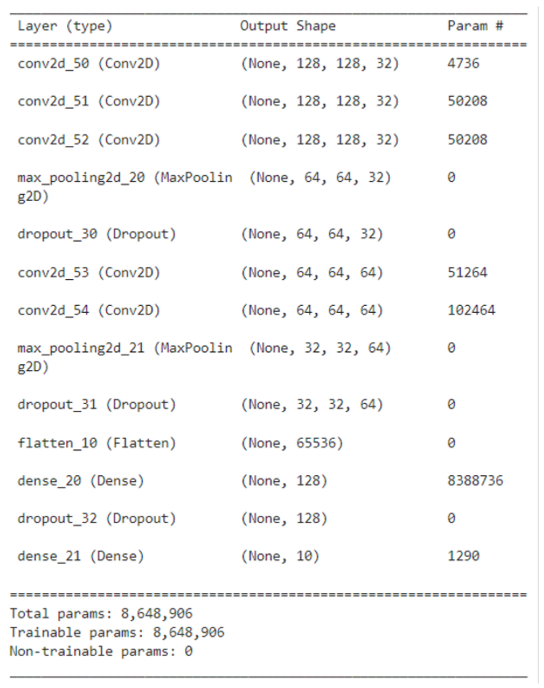


Fig. 2: The outline of a CNN [6]



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_50 (Conv2D)           (None, 128, 128, 32)      4736

conv2d_51 (Conv2D)           (None, 128, 128, 32)      50208

conv2d_52 (Conv2D)           (None, 128, 128, 32)      50208

max_pooling2d_20 (MaxPoolin  (None, 64, 64, 32)        0
g2D)

dropout_30 (Dropout)         (None, 64, 64, 32)        0

conv2d_53 (Conv2D)           (None, 64, 64, 64)        51264

conv2d_54 (Conv2D)           (None, 64, 64, 64)        102464

max_pooling2d_21 (MaxPoolin  (None, 32, 32, 64)        0
g2D)

dropout_31 (Dropout)         (None, 32, 32, 64)        0

flatten_10 (Flatten)         (None, 65536)             0

dense_20 (Dense)             (None, 128)               8388736

dropout_32 (Dropout)         (None, 128)               0

dense_21 (Dense)             (None, 10)                1290

=================================================================
Total params: 8,648,906
Trainable params: 8,648,906
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3: Our CNN architecture

B. Convolutional layer

A convolutional layer is parameterized by its number of maps, map sizes, kernel sizes, and skipping factors. Each layer has M equally-sized maps (M_x, M_y). A kernel of size (K_x, K_y) is shifted over the input image's valid region. The skipping factors S_x, S_y specify the number of pixels the filter/kernel misses in the x- and y-axes between successive convolutions. The size of the output map is therefore defined as:

$$M_x^n = \frac{M_x^{n-1} - K_x^n}{S_x^n + 1} + 1; \quad M_y^n = \frac{M_y^{n-1} - K_y^n}{S_y^n + 1} + 1$$

where index n represents the layer. Each map in layer L^n has a maximum of M^{n-1} connections to maps in layer L^{n-1}. A map's neurons have identical weights but distinct input fields.

C. Max-pooling layer

Max pooling is a pooling procedure that selects the greatest element from the feature map region covered by the filter. Therefore, the result of the max-pooling layer would be a feature map consisting of the most prominent features from the prior feature map [8]. Fig. 4 shows an example of Max Pooling in CNN.

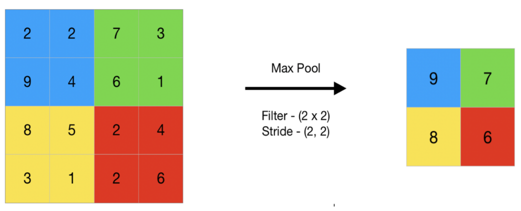


Fig. 4: Example of Max Pooling [8]

D. Classification layer

A classification layer computes the cross-entropy loss for mutually exclusive classes in classification and weighted classification problems. Within this layer, we commonly use softmax as the activation function, and it utilizes the results from the softmax function and the cross-entropy function for a 1-of-K coding scheme to assign each input to one of the K mutually exclusive classes [9]. The math could be defined as the following:

$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} w_i \mathrm{t}_{ni} \ln y_{ni}$$

where N is the number of samples, K is the number of classes, w_i is the weight for class i, t_{ni} is the indicator that the nth sample belongs to the ith class, and y_{ni} is the output for a sample n for class i, which is the value from the softmax function in this study.

The kernel sizes of convolutional filters, max-pooling rectangles, and skipping factors can be selected so that the output maps of the final convolutional layer are downsampled to a resolution of 1 pixel per map. A fully connected layer, instead, integrates the outputs of the last convolutional layer into a 1D feature vector. In recognition tasks, the final layer is always fully connected with one output unit per class. We utilize softmax as the activation function for the last layer, thus the output of each neuron indicates the class probability. Also, we used ReLU as the activation function for the other layers.

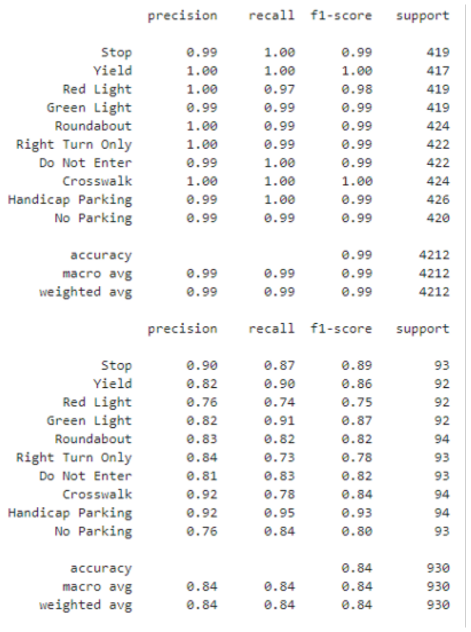
III. Experiment

The training dataset of this study contains a total of 5635 samples, including 10 traffic signs (10 classes). Classes were assigned labels from 0 to 9 (0: stop, 1: yield, 2: red light, 3: green light, 4:roundabout, 5: right turn only, 6: do not enter, 7: crosswalk, 8: handicap parking, 9: no parking).

A. Data Preprocessing

In data preprocessing, we firstly resized each picture, then applied data augmentation to the dataset. The size of each picture was processed from 300 x 300 to 128 x 128. The training time was 5 times longer while using the original size, and the training performance was much worse than the resized training dataset (test loss = 0.468; test accuracy = 0.898), which could cause the curse of dimensionality.

Secondly, we augmented the dataset by flipping and rotating images. The training performance showed that overfitting occurs when we train the data without data augmentation. Fig.5 presents the results of training data without data augmentation (overall loss = 0.336; overall accuracy = 0.970; validation loss = 5.501; validation accuracy = 0.846; test loss = 5.128; test accuracy = 0.837).



```
                  precision    recall  f1-score   support

           Stop       0.99      1.00      0.99       419
          Yield       1.00      1.00      1.00       417
      Red Light       1.00      0.97      0.98       419
    Green Light       0.99      0.99      0.99       419
     Roundabout       1.00      0.99      0.99       424
Right Turn Only       1.00      0.99      0.99       422
   Do Not Enter       0.99      1.00      0.99       422
      Crosswalk       1.00      1.00      1.00       424
Handicap Parking      0.99      1.00      0.99       426
     No Parking       0.99      0.99      0.99       420

       accuracy                           0.99      4212
      macro avg       0.99      0.99      0.99      4212
   weighted avg       0.99      0.99      0.99      4212

                  precision    recall  f1-score   support

           Stop       0.90      0.87      0.89        93
          Yield       0.82      0.90      0.86        92
      Red Light       0.76      0.74      0.75        92
    Green Light       0.82      0.91      0.87        92
     Roundabout       0.83      0.82      0.82        94
Right Turn Only       0.84      0.73      0.78        93
   Do Not Enter       0.81      0.83      0.82        93
      Crosswalk       0.92      0.78      0.84        94
Handicap Parking      0.92      0.95      0.93        94
     No Parking       0.76      0.84      0.80        93

       accuracy                           0.84       930
      macro avg       0.84      0.84      0.84       930
   weighted avg       0.84      0.84      0.84       930
```

Fig. 5: The Results of training data without data augmentation

Last, we ran an experiment on changing greyscale in the dataset. When we used grey images to train, the performance of red and green lights classification significantly dropped (test loss = 6.026; test accuracy = 0.769). Fig.7 shows the confusion matrix.

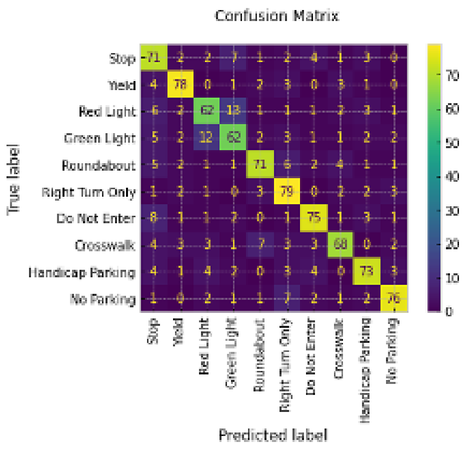


Fig. 6: The Results of training data with greyscaled images

B. Classification Results

After data preprocessing, we ran experiments to determine the best learning rate, optimizer, and BatchSize. Fig.7, 8, and 9 presents the performance comparisons for learning rate, optimizer, and BatchSize, respectively.

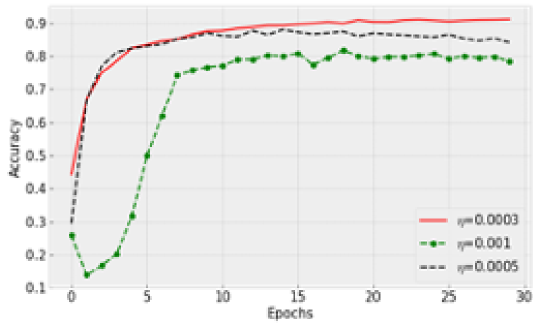


Fig. 7: Learning rate and learning curve

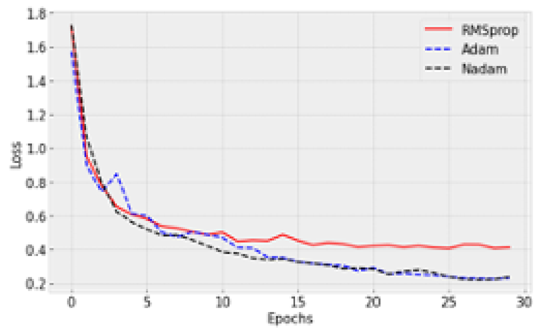


Fig. 8: Optimizer

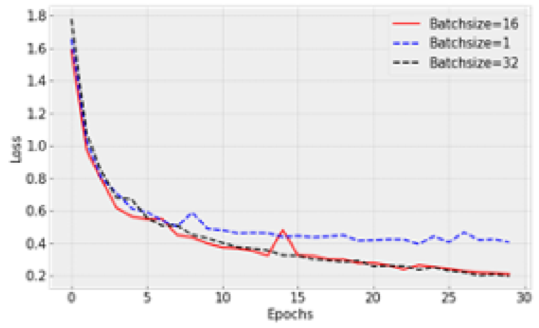


Fig. 9: BatchSize

Further, we explored how kernel amount impacts model performance. Fig. 10 shows the results of the experiment. Based on the results, we can see that the model performance is more stable when we doubled the kernel amount. We then investigated whether using dropout can improve model performance or not. Fig. 11 shows the results of training with and without dropout, and we found that using dropout within the model can prevent overfitting.

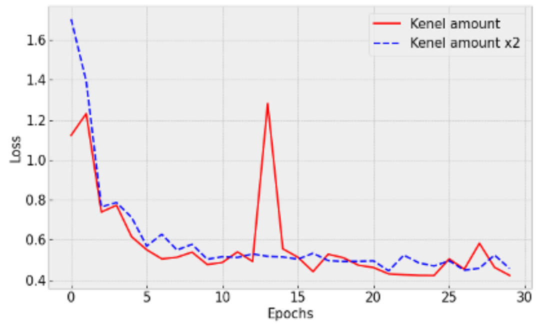


Fig. 10: Model performance by changing kernel amount

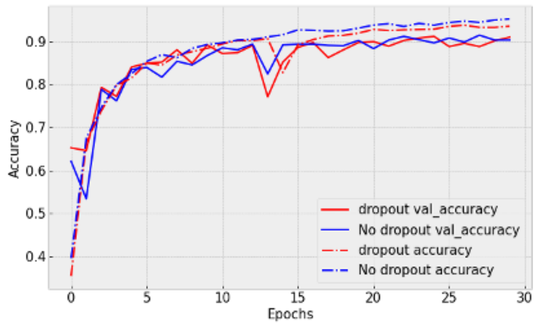


Fig. 11: Model performance with and without dropout

In the end, we constructed the final model with the learning rate μ = 0.0003, Nadam optimizer, with dropout, and Batch-Size = 32. Table 1 presents the final model performance.

TABLE I: Classification Report - Final Model

TRAINING SET				
	Precision	Recall	F1-Score	Support
Stop	0.95	0.95	0.95	419
Yield	0.92	0.95	0.94	417
RedLight	0.97	0.88	0.92	419
GreenLight	0.99	0.90	0.94	419
Roundabout	0.81	0.79	0.80	424
RightTurnOnly	0.82	0.87	0.84	422
DoNotEnter	0.96	0.90	0.93	422
Crosswalk	0.81	0.95	0.87	424
HandicapParking	0.89	0.98	0.93	426
NoParking	0.96	0.90	0.93	420
accuracy			0.90	4212
macroavg	0.91	0.90	0.91	4212
weightedavg	0.91	0.90	0.91	4212
VALIDATION SET				
	Precision	Recall	F1-Score	Support
Stop	0.92	0.95	0.93	93
Yield	0.85	0.92	0.89	92
RedLight	0.95	0.84	0.89	92
GreenLight	1.00	0.91	0.95	92
Roundabout	0.89	0.80	0.84	94
RightTurnOnly	0.86	0.87	0.87	93
DoNotEnter	0.99	0.87	0.93	93
Crosswalk	0.80	0.95	0.87	94
Handicap Parking	0.90	1.00	0.94	94
No Parking	0.91	0.91	0.91	93
accuracy			0.90	930
macroavg	0.91	0.90	0.90	930
Weighted avg	0.91	0.90	0.90	930

IV. Conclusion

In this paper, we presented a CNN classification algorithm for traffic signs. In constructing our final model, we ran several experiments to reach the highest model performance. In the experiments, the proposed CNN got a classification accuracy of 90% on the training dataset. By observing the failure cases, we find that there is still room for improvement. In the future, we plan to add extra preprocessing steps and discriminative features to further improve the classification accuracy.

References

[1] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using kd trees and random forests," in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 2151–2155.

[2] S. Mehta, C. Paunwala, and B. Vaidya, "Cnn based traffic sign classification using adam optimizer," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. IEEE, 2019, pp. 1293–1298.

[3] G. Wang, G. Ren, Z. Wu, Y. Zhao, and L. Jiang, "A hierarchical method for traffic sign classification with support vector machines," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–6.

[4] A. Wong, M. J. Shafiee, and M. S. Jules, "Micronnet: a highly compact deep convolutional neural network architecture for real-time embedded traffic sign classification," *IEEE Access*, vol. 6, pp. 59 803–59 810, 2018.

[5] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 1918–1921.

[6] S. Balaji, "Binary image classifier cnn using tensorflow," 2020, https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697. [Online]. Available: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697

[7] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a cnn-lstm network," in *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 639–645.

[8] S. Khosla, "Cnn: Introduction to pooling layer," 2021, https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/. [Online]. Available: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer

[9] MathWorks, "Classification layer," 2022, https://www.mathworks.com/help/deeplearning.html. [Online]. Available: https://www.mathworks.com/help/deeplearning.html