Research Article

Real time traffic Sign detection and recognition for autonomous vehicle

Abstract

The advancement of technology has made it possible for modern cars to utilize an increasing number of processing systems. Many methods have been developed recently to detect traffic signs using image processing algorithms. This study deals with an experiment to build a CNN model which can classify traffic signs in real-time effectively using OpenCV. The experimentation method involves building a CNN model based on a modified LeNet architecture with four convolutional layers, two max-pooling layers and two dense layers. The model is trained and tested with the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Parameter tuning with different combinations of learning rate and epochs is done to improve the model's performance. Later, this model is used to classify the images introduced to the camera in real- time. The graphs depicting the accuracy and loss of the model before and after parameter tuning are presented. Also, an experiment was performed to classify the traffic sign image introduced to the camera by using the CNN model. A high probability score is achieved during the process.

Key words: convolutional neural network, image processing, deep learning algorithms, LeNet, OpenCV

Volume 8 Issue 3 - 2022

Ahmed J. Abougarair,[1] Mohammed Elmaryul,[2] Mohamed KI Aburakhis[3]

[1]Electrical & Electronic Engineering, University of Tripoli, Libya
Electrical & Electronic Engineering, University of Tripoli, Libya
[2]Clark State College, Springfield, USA

Correspondence: Ahmed J Abougarair, Electrical & Electronic Engineering, University of Tripoli, Tripoli Libya, Tel: +218916094184, Email: a.abougarair@uot.edu.ly

Received: August 27, 2022 | **Published:** September 15, 2022

Introduction

Traffic signs typically use symbols and graphics to provide important information to drivers. Traffic signs are classified based on the information they provide.[1]Traffic lights are indicators placed on the side of the road to provide information about directions and road conditions. There are three main types of traffic sign recognition methods: the shape method, the color method, and the learning method Figure 1.

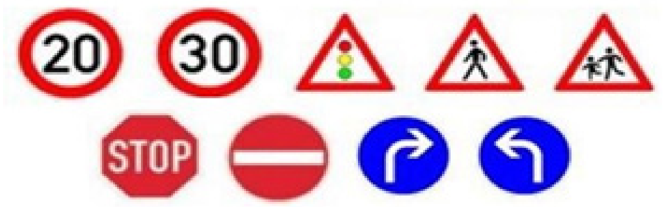

Figure 1 Different types of traffic signs.[2]

The most essential of these indications are warning signs. The presence of warning signs indicates the presence of a potential hazard, impediment or road condition. The warning signs may convey information about the condition of the road or road hazards that the driver may not notice right away. Pedestrian crossing signs, caution signs, crosswalk alerts, and traffic light alerts are all included in this category. Priority signs show the vehicle's course, or the direction in which it should pass to avoid a collision. Stop signs, intersection signs, one-way traffic signs, and other traffic signs fall within this category. Restrictive signs are used to limit specific sorts of maneuvers and vehicles from being driven on public highways. This could be, for example, no-entry sign, a no- motorcycle sign, or a no-pedestrian sign. Mandatory signs are the polar opposite of prohibition signs, in that they tell drivers exactly what they must do. This category includes permitted directions and vehicle allowances. Special signs and regulations indicate rules or provide information about how to use something. This category includes one-way signs and home zone indications. Directional signs indicate the various destinations that can be reached from a given site. This includes some orders such as turning right, turning left, and so on. Recently, several attempts have been made to recognize traffic signs and alert the motorist. The Driver Assistance System (DAS) can benefit from automatic traffic sign recognition.[1,2] DAS is based on automatic traffic sign recognition, which detects and classifies traffic signs in real time. In unmanned driving technology, traffic sign recognition is also a critical component. With the development of the modern automobile industry, the focus has been on the implementation of autonomous vehicles. The majority of traffic sign recognition systems include two phases: the first part entails applying image processing to detect the traffic sign in an image. The second phase entails applying an artificial neural network model to classify the detected image.[3,4] Traffic sign detection and identification has become increasingly challenging in recent years. According to the World Health Organization (WHO), enforcing traffic speed limits helps to decrease traffic accidents.[5] Drivers who exceed the speed restriction frequently disregard this speed limit. Road accidents can occur for a variety of reasons, including poor traffic sign quality, driver irresponsibility, and objects near the traffic sign that cause sight obstruction. Human-caused errors are among the causes highlighted.[6]

Background

A. Convolutional Neural Network

Deep learning has recently proven useful in a variety of fields, including voice recognition, image processing, and processing natural language.[7] Convolution networks are a type of neural network that can process input in a grid-like structure.[8,9] In the case of images, a 2D grid of pixels exists. A CNN adds relevance to an input image by applying biases and weights to distinct objects in the image.[10] Instead of matrix multiplication, CNN's layers use a mathematical operation called convolution. In a CNN, there are many convolutional, pooling, or fully connected layers between the input and output layers.[11-15]

• **Convolutional layer:** The convolutional layers in a CNN are utilized to extract features from the input Kernels are applied to the image's neighborhood pixels as the input image pixels move through the convolutional layer. The kernel matrix is cross-correlated before being applied to the picture pixels. The weights in the kernel are modified during training, but they remain constant during calculations.

• **Pooling** is a sliding window technique in which the values within a window are subjected to statistical functions. The maximum value within the window is taken with max pooling, which is a regularly used function. For down-sampling, pooling is Pooling is mostly used to reduce the number of parameters and improve the robustness of feature detection.

• **Fully connected layer:** The classification of the objects is done by a fully connected or dense layer using the output from the pooling layer. The dense layer's input should be reduced to a single N x 1 tensor.

• **The dropout layer:** The dropout layer mitigates overfitting by setting inputs to zero at random with a configurable probability. The goal behind dropout is to remove units and their connections during the neural network's training. Dropout enhances neural network performance.

Data overview

T The dataset used in this investigation is GTSRB. It's a multi-class benchmark dataset for TSR that's utilized for single-image classification. The dataset was utilized in the IJCNN competition, and both CNN models and hand classification were employed to obtain excellent classification accuracy. Over 35,000 images of various traffic signals are included in the dataset. It's further divided into 43 separate categories. These 43 class labels are traffic indicators that can be found in almost every country. Figure 2 depicts an example image from each class. The data is fairly diverse. Some of the classes feature a lot of images, while others don't.[16-18]

Figure 2 Sample images from dataset.[19,20]

Method

We used the experimentation technique in our paper to attain the paper's goal and to answer the research questions. The experiment was carried out using a CNN model based on a modified LeNet model. Figure 3&4 summarizes the steps performed in the traffic sign recognition experiment. The steps are specially categorized into phases.

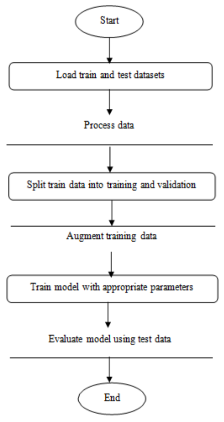
Start
Load train and test datasets
Process data
Split train data into training and validation
Augment training data
Train model with appropriate parameters
Evaluate model using test data
End

Figure 3 Flowchart of building our CNN.

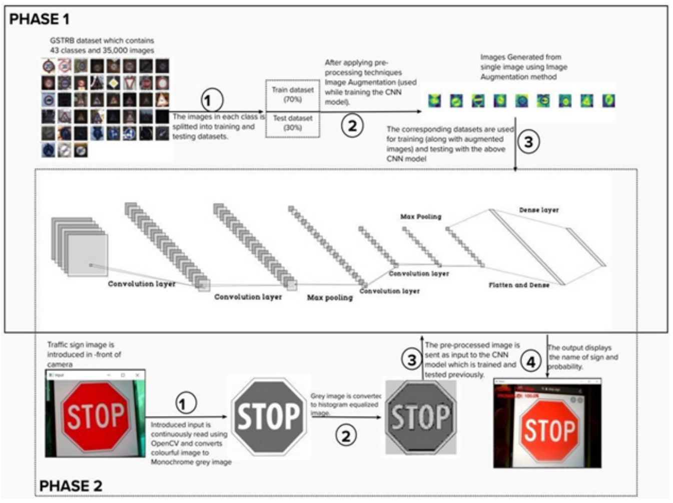
PHASE 1
GSTRB dataset which contains 43 classes and 35,000 images
1
The images in each class is splitted into training and testing datasets.
Train dataset (70%)
Test dataset (30%)
After applying pre-processing techniques Image Augmentation (used while training the CNN model).
2
Images Generated from single image using Image Augmentation method
The corresponding datasets are used for training (along with augmented images) and testing with the above CNN model
3
Convolution layer
Convolution layer
Max pooling
Convolution layer
Convolution layer
Max Pooling
Dense layer
Flatten and Dense
Traffic sign image is introduced in -front of camera
1
Introduced input is continuously read using OpenCV and converts colourful image to Monochrome grey image
Grey image is converted to histogram equalized image.
2
3
The pre-processed image is sent as input to the CNN model which is trained and tested previously.
4
The output displays the name of sign and probability.
PHASE 2

Figure 4 Steps involved in training, testing the proposed model (Phase 1) and testing the model in real-time (Phase 2).

Adaptive Moment Estimation (Adam) is a gradient-based approach that accumulates past squared gradients, such as Adadelta and RMSprop, and stores the decaying mean of the past gradients.[9] This optimization method is simple to implement and takes less memory. This approach is a good choice for optimizing stochastic objective functions using gradients.

Adam combines the AdaGrad and RMSProp algorithms' properties to provide an optimization algorithm that can manage sparse gradients on noisy issues. Adam combines the benefits of two other stochastic gradient descent extensions and the adaptive gradient algorithm (AdaGrad), which retains a learning speed per-parameter that improves performance on sparse gradient issues (e.g., natural language issues and computer vision issues). The Root Mean Square Propagation (RMSProp) also preserves per-parameter learning rates adjusted to the weight based on the average of recent magnitudes. This algorithm does well for offline and non-stationary problems. This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima at a faster pace. The ADAM algorithm has been briefly mentioned below.

$$w_{t+1} = w_t - \alpha m_t \tag{1}$$

$$m_t = \beta n_{t-1} + (1+\beta)\frac{\partial L}{\partial Wt} \tag{2}$$

mt = aggregate of gradients at time t [current] (initially, mt = 0)

mt-1 = aggregate of gradients at time t-1 [previous]

Wt = weights at time t Wt+1 = weights at time t+1

αt = learning rate at time t

∂L = derivative of Loss Function

∂Wt = derivative of weights at time t

β = Moving average parameter (const, 0.9)

Building the CNN model

We used the LeNet model[12] and tweaked it a bit. The LeNet-5 CNN architecture consists of 3 convolutional layers, 2 subsampling layers, and 2 fully linked layers that make up the layer composition. To the LeNet model, we added one extra convolutional layer and two dropout layers. We employed 4 convolution layers, 2 max-pooling layers, 2 dropout layers, 2 fully connected layers, and 1 flattening approach in our experiment, which is shown in Figure 5.

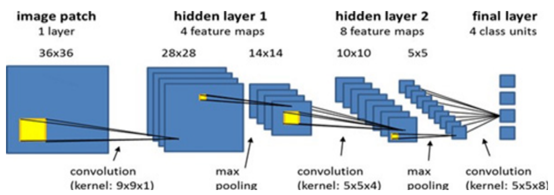

Figure 5 CNN model Architecture.

Training and testing the model

The Convolutional Neural Network CNN works by taking an image, assigning it a weight based on the various objects in the image, and then distinguishing them from one another. CNN is a particularly fast deep learning algorithm, requiring comparatively little pre-processing data compared to other algorithms. Figure 6 explains the training. Graphs for loss and accuracy are plotted for training and validation datasets with increasing epochs. The testing dataset is used once the training dataset has been finished. These are the photographs that aren't used in training, such as traffic images that aren't visible. The model is then saved with the (.h5) extension using the save function.

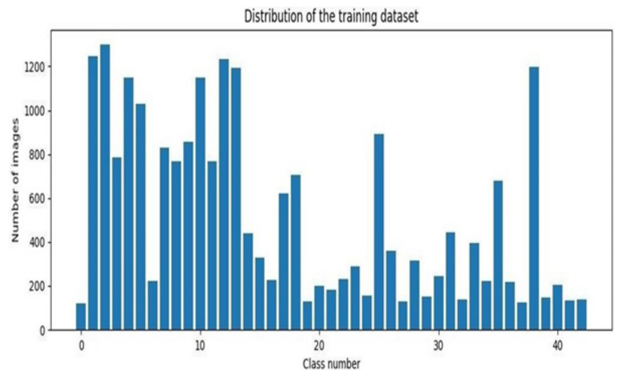

Figure 6 Distribution of the training dataset.

Parametr tunning

The parameter adjustment was made after evaluating the model, which included a systematic test with numerous combinations of epochs and learning rate. The deep learning model has a lot of hyper-parameters. Epochs and learning rate are two of these characteristics that have a substantial impact on the model's accuracy. When the model's weights are changed, the learning rate determines how much the model should change. One of the main hyper-parameters that impacts the model's accuracy is the learning rate.[13-17] A full pass of the dataset is defined as an epoch. Epochs can have a significant impact on accuracy. As a result, both of these factors are used to tune the system. Table 1 summarizes the various combinations.

Table 1 Various combinations of parameter tuning

Learning rate	epochs
0.001	10
0.001	20
0.002	10
0.002	20
0.001	30
0.0005	20
0.001	5

Testing the model in real-time

The image is extracted using OpenCV and converted to an array of pixels. The array is then resized using the resize function to a 32*32-dimension array. The transformed array will be preprocessed with the grayscale method and histogram equalization method before being supplied as input to the previously trained and tested CNN model. Here, we use real-time video collected using a camera or any other external source to evaluate our model. After we gain access to the web camera, we process every frame of the live video. We use our saved model to predict the likelihood of the processed image. The image is recognized as a traffic sign if the likelihood of the prediction is greater than the threshold value of 0.75. The prediction probability value and the related class name are displayed Figure 7.

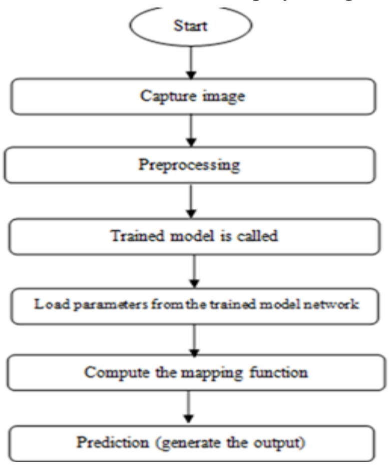
Start
Capture image
Preprocessing
Trained model is called
Load parameters from the trained model network
Compute the mapping function
Prediction (generate the output)

Figure 7 Flowchart of our application using webcam.

Results and analysis

Training the model

In this experiment, the parameters of learning rate and epochs are modified. It results in a significant improvement in the model's accuracy. To attain significant model accuracy, as shown in Figures 8–11, we used various combinations of these parameters. The graphs below show the model's accuracy and loss after the parameters were tuned. The train data is represented by the blue line, while the test data is represented by the orange line.

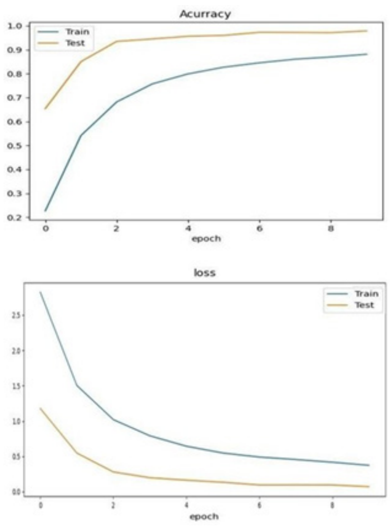

Figure 8 Accuracy & loss for learning rate = 0.001 and epochs = 20.

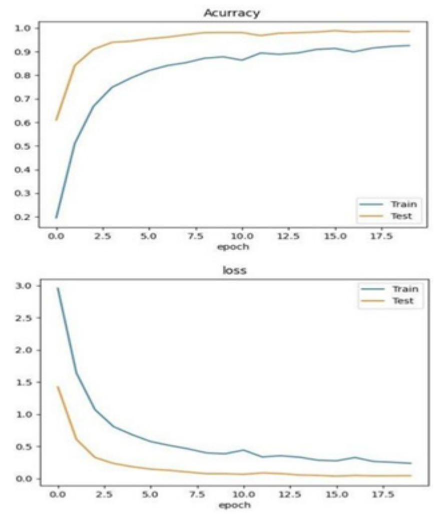

Figure 9 Accuracy & loss for learning rate = 0.0005 and epochs = 20.

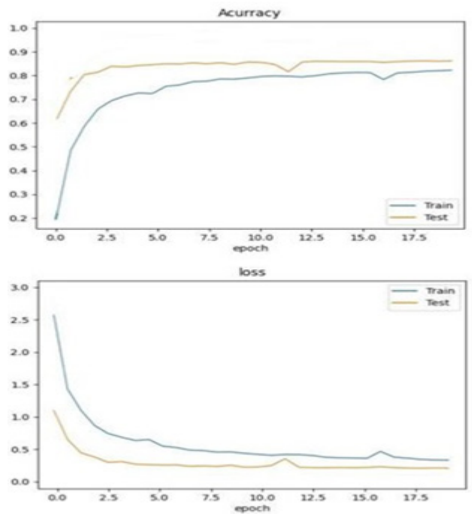

Figure 10 Accuracy & loss for learning rate = 0.001 and epochs = 5.

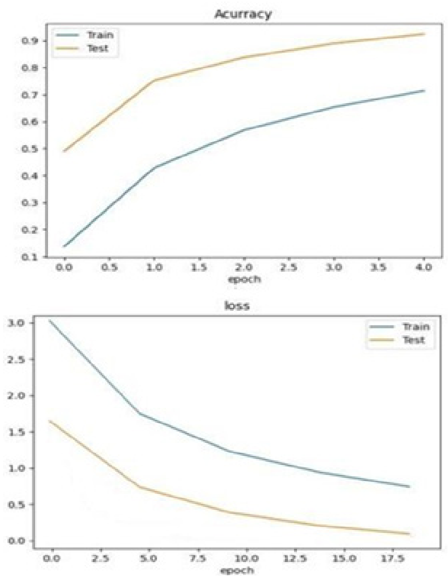

Figure 11 Accuracy & loss for learning rate = 0.001 and epochs = 5.

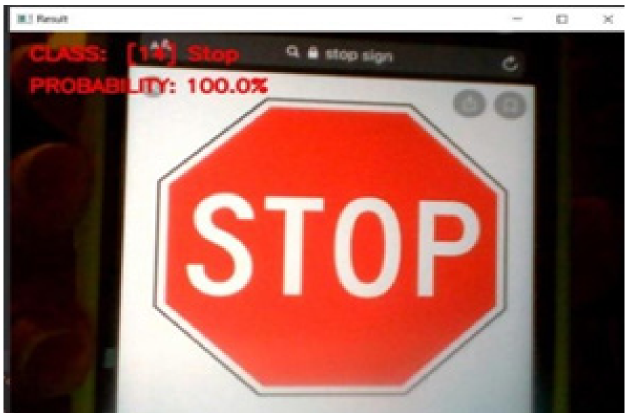

Figure 12 Traffic sign classification in real time using webcam.

Using the model in real-time

In this experiment, the model is in a real-time video utilizing a camera or any external live stream. The photos of traffic signs that have been introduced to the camera are brand new and have not been derived from training or testing data. For testing, new photos from each class are utilized. Every frame of the video is examined for a traffic sign. If the prediction's probability value is greater than the threshold, the class label will be presented alongside the likelihood. The traffic stop sign picture is placed in front of the webcam as shown in Figure 12. The likelihood of a prediction is presented alongside the class name. Every traffic sign in each of the 43 classifications is put to the test in real time. While some have probability scores of more than 90%, others have scores that fluctuate. The probability score fluctuates when the image of the traffic sign 'Speed limit (20 km/h)' is placed in front of the camera, for example. We put the model to the test with 43 different traffic signals, one for each class, and recorded the results. Table 2 shows the probabilities for each class designation. We suggested a CNN model with a LeNet architecture that can categorize traffic signs using the GTSRB dataset. The main questions are addressed by deducing specific conclusions from the findings. Considering the initial parameter setups of the proposed model, to what extent can parameter tuning improve the model's accuracy. The accuracy and loss of train and test data are depicted in Figures 8 to 11. This information is collected by experimenting with different combinations of epochs and learning rates. Every parameter adjustment combination resulted in a significant change in the model's accuracy. When the learning rate is tweaked, certain intriguing trends in the model's accuracy emerge. The number of weights that are updated during training the model is referred to as the learning rate or step size. The learning rate is a tunable hyper parameter that is utilized during model training and has a range of 0.0 to 1.0. Table 3 shows that increasing the learning rate beyond the default set optimum value of 0.001 reduces the model's accuracy. This is because the learning rate determines how quickly a model adapts to a challenge. As a result, a high learning rate makes the model learn faster but produces an unsatisfactory set of weights. Loss on the training dataset occurs due to the divergent weight decreasing the accuracy. A lower learning rate allows the model to learn an optimal set of weights, but may take a lot of time. The model may be based on a suboptimal solution. Hence, changing the learning rate from the default value decreases the model accuracy.

It clearly shows that increasing the number of epochs leads to higher accuracy. However, increasing the number of epochs beyond what is required may result in an over fitting problem. When the number of epochs used to train the model is rapidly increased, the model learns the sample data patterns to a large extent. On test data, this makes the model less efficient. The model has great accuracy on training data but fails to do soon test data due to over fitting. Under fitting occurs when there are fewer epochs. As a result, accuracy suffers as well. The model learning patterns with varying numbers of epochs are represented in the figure epoch.

Table 2 Probability scores achieved when testing the model with new images in real time

Class Id	Traffic sign	Probability
0	Speed Limit (20 km/h)	80-90%
1	Speed Limit (30 km/h)	100%
2	Speed Limit (50 km/h)	98%
3	Speed Limit (60 km/h)	99%
4	Speed Limit (70 km/h)	100%
5	Speed Limit (80km/h)	100%
6	End of speed limit (80km/h)	95-98%
7	Speed Limit (100km/h)	99%
8	Speed Limit (120km/h)	98%
9	No passing	100%
10	No passing for vehicles over 3.5 metric tons	99%
11	Right-of-way at the next intersection	100%
12	Priority road	97-99%
13	Yield	99%
14	Stop	92-95%
15	No vehicles	91-95%
16	Vehicles over 3.5 metric tons prohibited	93-96%
17	No entry	98-99%
18	General caution	99%
19	Dangerous curve to the left	80-90%
20	Dangerous curve to the right	85-90%
21	Double curve	89-93%
22	Bumpy road	86-91%
23	Slippery road	87-90%
24	Road narrows on the right	90-93%
25	Road work	98-99%
26	Traffic signals	94-96%
27	Pedestrians	86-90%
28	Children crosiing	89-94%
29	Bicycles crossing	85-90%
30	Beware of ice/snow	91-95%
31	Wila animals crossing	96-98%
32	End of all speed and passing limits	89-93%
33	Turn right ahead	95-97%
34	Turn left ahead	88-91%
35	Ahead only	99-100%
36	Go straight or right	89-92%
37	Go straight or left	85-88%
38	Keep right	100%
39	Keep left	85-88%
40	Round about mandatory	86-89%
41	End of no passing	90-92%
42	End of no passing by vehicles over 3.5 metric tons	87-91%

Table 3 Accuracy and loss of model for train and test data by tuning various combinations of learning rate and epochs

Learning rate	Epochs	Train accuracy	Train loss	Test accuracy	Test loss
0.001	10	0.8883	0.3659	0.9843	0.0669
0.002	10	0.8549	0.4684	0.9697	0.0996
0.001	20	0.9262	0.2352	0.9857	0.0477
0.002	20	0.9052	0.3155	0.9883	0.0469
0.0005	20	0.9222	0.2508	0.987	0.0407
0.001	5	0.7026	0.9544	0.9274	0.258

Conclusions

Activity sign acknowledgment could be a challenging computer vision assignment of tall industrial relevance. Within the case of self-driving vehicles, Activity Sign Acknowledgment assists inexperienced drivers through the Driver Help Framework and gives secure, proficient route. The unbalanced distribution of the data lead to some classes being underrepresented and ultimately achieved evaluation metric values lower than that of other classes. Batch normalization and the dropout layer integrated into the model helped increase training speed and overcome over fitting problems respectively. While Adam optimization was utilized for faster adaptation. increasing the epochs significantly increase the model's accuracy up to a certain extent. With optimum learning rate set to 0.001 and epochs

set to 30, 99% accuracy is achieved. A minimum loss of 0.1664 is achieved. Within the future, more variables, such as group estimate and dropout rate, can be changed to improve the model's precision essentially. It would be interesting to find the parameters that, when adjusted, can move forward the model's exactness. Amazon Web Server (AWS) Profound Learning Amazon Machine Pictures (AMI) can be utilized to construct auto-scaled GPU clusters for expansive scale preparing to overcome the problem of long preparing times. To prepare the show, one can effortlessly dispatch an Amazon EC2 instance that comes pre-installed with common profound learning systems and APIs.

Acknowledgements

None

Conflicts of interests

Author declares that there is no conflict of interest.

References

1. Shangzheng L. *A traffic sign image recognition and classification approach based on convolutional neural network*, in the 11th International Conference on Measuring Technology and Mechatronics Automation;2019.

2. Mehta S, Paunwala C. CNN based traffic sign classification using Adam Optimizer, in the International Conference on Intelligent Computing and Control Systems;2019.

3. Abougarair AJ. Model reference adaptive control and fuzzy optimal controller for mobile robot. *Journal of Multidisciplinary Engineering Science and Technology*. 2019;6(3):9722–9728.

4. Luo L, Xiong Y. *Adaptive gradient methods with dynamic bound of learning rate,* in Proceedings of the 7th International Conference on Learning Representations, New Orleans, Louisiana;May 2019.

5. Abougarair AJ. Real time classification for robotic arm control based electromyographic signal. *2022 IEEE 2st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering* (MI- STA2022);2022:23–25.
6. Reddi S, Zaheer M, Sachan D, et al. *Adaptive methods for nonconvex optimization*, in Proceeding of 32nd Conference on Neural Information Processing Systems;2018.
7. Abougarair AJ. Electroencephalography based control approaches to wheelchair and robot arm control. *Al academia journal for Basic and Applied Sciences (AJBAS)*.2022;4(1).
8. Islam MT. *Traffic sign detection and recognition based on convolutional neural networks*, in 2019 International Conference on Advances in Computing, Communication and Control (ICAC3).2019:1–6p.
9. https://www.erda.dk/public/archives/YXJjaGl2ZS1McldVYkU=/published-archive.html
10. Bengio Y, Goodfellow I, Courville A. Deep learning. *MIT press Massachusetts*, USA;2017:1.
11. Albawi S, Mohammed TA, Al-Zawi S. *Understanding of a convolutional neural network*, in 2017 International Conference on Engineering and Technology (ICET).2017:1–6p.
12. Abougarair AJ. *Neural networks identification and control of mobile robot using adaptive neuro fuzzy inference system*, ICEMIS'20: Proceedings of the 6th International Conference on Engineering & MIS 2020;2020.
13. Shangzheng L. *A traffic sign image recognition and classification approach based on convolutional neural network,* in 2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). IEEE;2019:408–411p.
14. Shabarinath BB, Muralidhar P. Convolutional neural network based traffic-sign classifier optimized for edge inference, in 2020 IEEE REGION 10 CONFERENCE (TENCON);2020:420– 425p.
15. Tsoi TS, Wheelus C. Traffic signal classification with cost- sensitive deep learning models, in 2020 IEEE International conference on Knowledge Graph (ICKG);2020:586–592p.
16. Abogarair AJ. Robust control and optimized parallel control double loop design for mobile robot. *IAES International Journal of Robotics and Automation (IJRA)*.2020;9(3).
17. Abougarair AJ, Gnan H, Oun A, et al. Implementation of a brain-computer interface for robotic arm control, 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2021);2021:25–27.
18. Shabarinath BB and Muralidhar P. Convolutional neural network based traffic-sign classifier optimized for edge inference, in 2020 IEEE REGION 10 CONFERENCE(TENCON);2020:420–425p.
19. Nasar Aldian AS, Abougarair AJ. Fault detection based on validated model of data filtering based recursive least squares algorithm For box--jenkins systems. *2021 Global Congress on Electrical Engineering (GC--ElecEng 2021)*;2021:10–11,Valencia, Spain.
20. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. *Journal of Big Data*. 2019;6(1):1–48.
21. Smith LN. A disciplined approach to neural network hyper- parameters: Part 1–learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820;2018.
22. Abougarair AJ, Mohamed K, Aburakhis. Design and implementation of smart voice assistant and recognizing academic words. *Int Rob Auto J*. 2022;8(1):27–32.