*I.J. Image, Graphics and Signal Processing,* **2022, 6, 70-83**
Published Online on December 8, 2022 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2022.06.06

# Deep Learning Based Autonomous Real-Time Traffic Sign Recognition System for Advanced Driver Assistance

**Sithmini Gunasekara**
Department of Electrical and Electronic Engineering, University of Peradeniya, Kandy, 20000, Sri Lanka
E-mail: e15112@eng.pdn.ac.lk
ORCID iD: https://orcid.org/0000-0003-1849-2309

**Dilshan Gunarathna**
Department of Electrical and Electronic Engineering, University of Peradeniya, Kandy, 20000, Sri Lanka
E-mail: e15111@eng.pdn.ac.lk
ORCID iD: https://orcid.org/0000-0002-1510-5513

**Maheshi B. Dissanayake***
Department of Electrical and Electronic Engineering, University of Peradeniya, Kandy, 20000, Sri Lanka
E-mail: maheshid@eng.pdn.ac.lk
ORCID iD: https://orcid.org/0000-0001-5209-5441
*Corresponding Author

**Supavadee Aramith**
Multimedia Data Analytics and Processing Research Unit, Department of Electrical Engineering, Chulalongkorn University, Bangkok, 10330, Thailand
E-mail: Supavadee.A@chula.ac.th
ORCID iD: https://orcid.org/0000-0001-9840-3171

**Wazir Muhammad**
Department of Electrical Engineering, Balochistan University of Engineering and Technology, Khuzdar, 207124, Pakistan
E-mail: wazir.laghari@gmail.com
ORCID iD: https://orcid.org/0000-0002-3860-2213

Received: 25 August, 2022; Revised: 28 September, 2022; Accepted: 02 November, 2022; Published: 08 December, 2022

**Abstract:** Deep learning (DL) architectures are becoming increasingly popular in modern traffic systems and self-driven vehicles owing to their high efficiency and accuracy. Emerging technological advancements and the availability of large databases have made a favorable impact on such improvements. In this study, we present a traffic sign recognition system based on novel DL architectures, trained and tested on a locally collected traffic sign database. Our approach includes two stages; traffic sign identification from live video feed, and classification of each sign. The sign identification model was implemented with YOLO architecture and the classification model was implemented with Xception architecture. The input video feed for these models were collected using dashboard camera recordings. The classification model has been trained with the German Traffic Sign Recognition Benchmark dataset as well for comparison. Final accuracy of classification for the local dataset was 96.05% while the standard dataset has given an accuracy of 92.11%. The final model is a combination of the detection and classification algorithms and it is able to successfully detect and classify traffic signs from an input video feed within an average detection time of 4.5fps

**Index Terms:** YOLO, Xception, preprocessing, data augmentation, template matching.

## 1. Introduction

Traffic signs provide valuable information to the driver, specially to prevent traffic accidents and to reduce congestion. With the emergence of driverless transportation concepts, automated systems have found applications in vehicular technology as well. Hence, traffic sign recognition is an essential part of the autonomous self-driving vehicles as well as in advanced driver assistance systems (ADAS).

The realization of a real-time traffic sign recognition system can be divided into three stages: detection, tracking and classification. These state-of-the-art systems are increasingly using artificial intelligence (AI); machine learning (ML), and deep learning (DL) architectures for building accurate and efficient systems. Most applicable literature on this topic discusses traffic sign classification systems and challenges faced at the classifications stage. Few applications present traffic sign detection systems. Yet, there is a significant lack of overall systems where challenges of both detection and classification in real-time feeds are addressed.

In this study, we are proposing a complete framework for real-time traffic sign recognition, which considers a live input from a dashboard mounted camera and outputs the classified output within a short time interval using live processing. The proposed framework is based on novel DL architectures, YOLO (You Only Look Once) v3 [1] and Xception models [2], with proven accuracy. Image preprocessing methods such as filters and Image Super Resolution (ISR) [3] were also used to improve model performance. Further, the system is trained and tested on a live video feed from a Sri Lankan driving scenario. Furthermore, the proposed model is offline tested on popular German Traffic Sign Recognition Benchmark (GTSRB) dataset [4] to validate its accuracy.

One of the key challenges faced by DL studies is the requirement of a huge dataset. This study investigates the possibility of adopting smaller datasets to customize the application domain while maintaining a good accuracy. The proposed model is based on Sri Lankan traffic signs. Using automotive traffic systems is rather new to Sri Lankan traffic systems. Hence, this study also attempts to introduce modern technological trends in the automobile industry into the Sri Lankan traffic systems.

## 2. Related Work

In the past few years, the concept of Convolutional Neural Network (CNN) has been subjected to vast modifications due to the increasing computational powers of modern processors. This has been a favorable inclination towards computer vision applications. Moreover, the availability of ample amounts of data has also been beneficial for these advancements. The first milestone of these modern CNN architectures is introduced by ref. [5] as the LeNet architecture. They give a gradient-based learning architecture using efficient back propagation algorithms, providing good solutions to the problems in pattern recognition applications.

With modifications, CNN architectures trained for very large datasets can be improved in terms of performance. Ref. [6] investigate why large CNN architectures perform really well and also possible methods for further improvements. They suggest that the layer size, i.e., depth of the network, plays an important role in improving the performance. According to ref. [7], the classification accuracy of large-scale image recognition systems based on conventional convolution networks can be improved with the increased depth of the network.

Another improvement to CNN architectures is presented by ref. [8]. They propose a method to prevent overfitting caused by using insufficient test data to train a large and complex neural network. Overfitting can be reduced by the proposed method called “dropout”, where units (or neurons) are randomly ignored at the training phase.

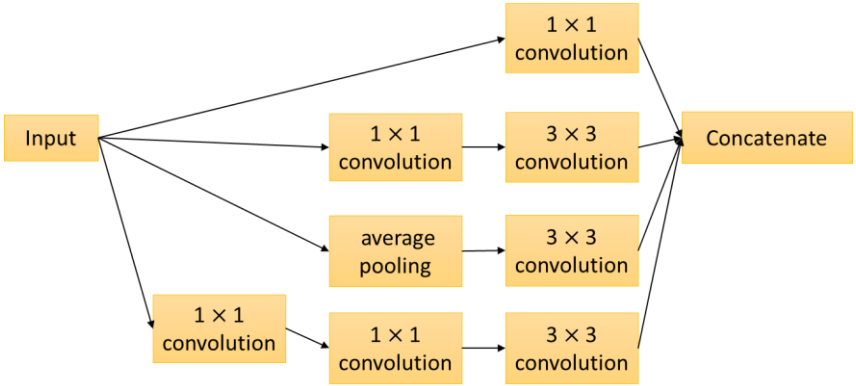


Fig. 1. Inception module [2]

To increase the representational power of neural networks, ref. [9] propose the Network-in-Network (NIN) approach. Instead of using linear filters along with a fully connected layer as in CNN, NIN uses micro neural networks instantiated by multilayer perceptrons along with a global average pooling layer to build the classification model. This method leads to easily interpreted and less overfitting DL models. NIN concept is majorly used in the Inception V1

architecture [10]. The additional 1×1 convolutional layers introduced by ref. [9] are used as dimensional reduction modules to prevent size limitations of the network. This results in increased depth and width of the network without significantly affecting the performance. With further modifications to Inception V1, Inception V2 [11], Inception V3 [12], Inception V4 and Inception-ResNet [13] were later introduced. Fig. 1 shows an Inception module, which is the fundamental building block of the Inception model. A number of such modules are stacked up to create the Inception model. This is a different architecture from the traditional CNN models where a series of simple convolutions are linearly stacked up.

Xception [2] is a network architecture inspired by the Inception model which significantly outperforms the Inception V3 model. Xception can be considered as an extreme version of the Inception model. Fig 2 shows an extreme version of the inception module which can be considered as identical to a depthwise separable convolution or simply a separable convolution [14]. This modified version is used in the Xception architecture. A depthwise separable convolution consists of a depthwise convolution followed by a pointwise convolution. The depthwise convolution performs spatial convolution independently over the input channels, and the pointwise convolution (i.e., 1×1 convolution) transfers the outputs of the depthwise convolution onto a new channel space. A slight difference between an extreme form of an Inception model and a depthwise separable convolution is that the first performs the 1×1 convolution at the beginning followed by the channel-wise spatial convolution whereas the latter does the reverse. Apart from the Inception model and depthwise separable convolution, Xception model strongly relies on the VGG-16 architecture [7] which shows schematic similarities to the Xception model.

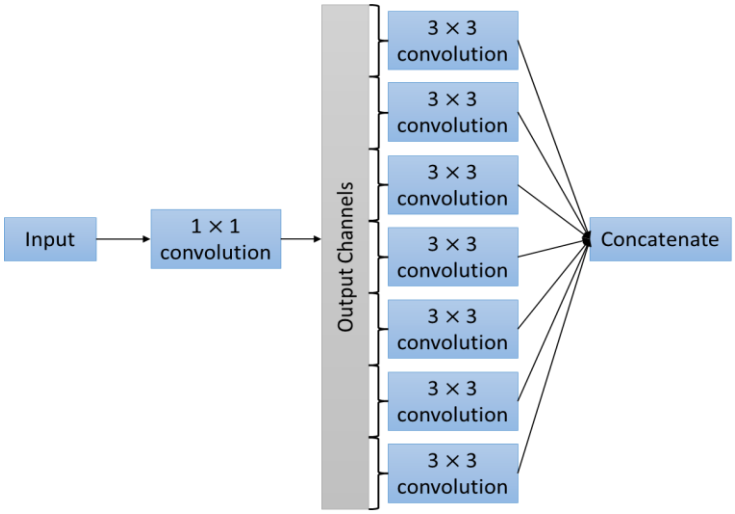


Fig. 2. An extreme version of Inception module [2]

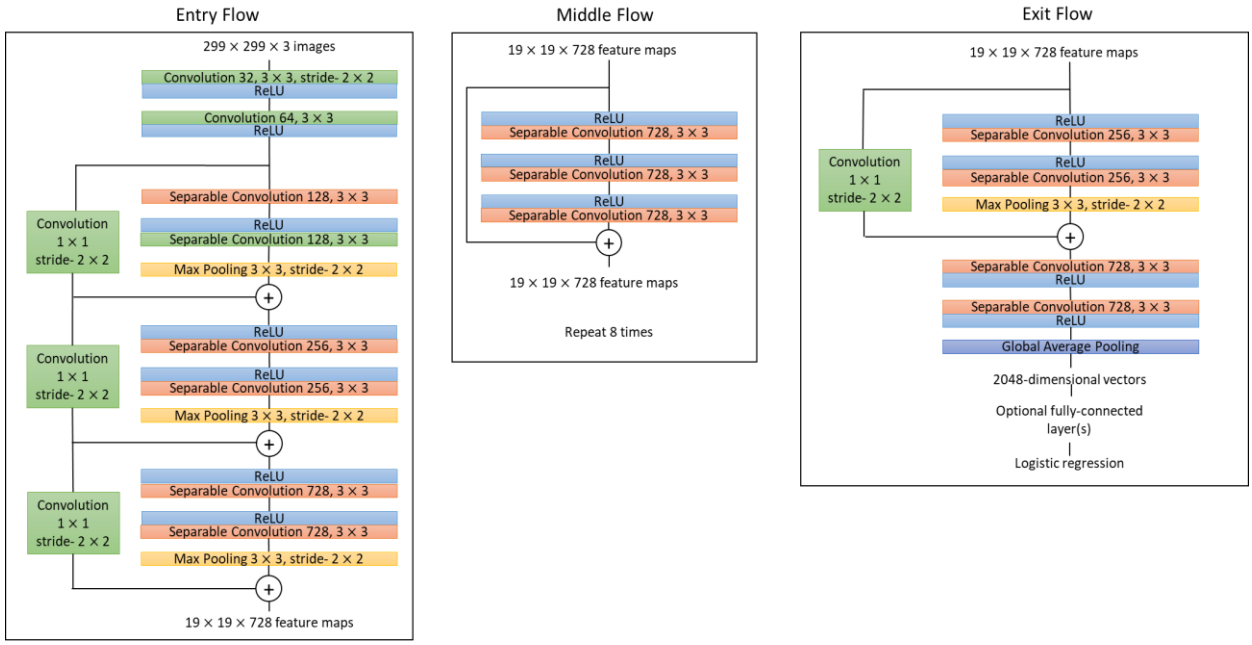


Fig. 3. Xception architecture [2]

The overall architecture of a Xception model is shown in fig. 3, where residual connections are used to maintain the model performance as the model goes deeper. Although the Xception model has the same number of parameters as the Inception V3 model, its increased performance is due to the efficient use of model parameters. A classification model based on Xception architecture is presented by ref. [15]. They propose a modified Xception model named LM2Xception for early identification of peach diseases based on imbalanced image data. A highest validation accuracy of 93.85% was obtained from the proposed model.

If the amount of training data is not large enough, DL models may not give the desired performance. Transfer learning can be used as a solution to deal with such limited datasets to achieve better performance. Ref. [16] explains the progress of using transfer learning for machine learning problems in modern applications. Ref. [17] proposes a transfer learning method known as Attentive Feature Distillation and Selection (AFDS) and investigates its performance on ResNet-101 [18]. Ref. [19] investigate the performance of the Inception V3 model in terms of accuracy and efficiency when it is used along with transfer learning.

Hyperparameter optimization helps to improve the performance of machine learning models. Ref. [20] study the impact of hyperparameter optimization on common machine learning models. They also discuss the challenges of hyperparameter optimization research. Ref. [21] provides a review on hyperparameter optimization and related topics. They discuss hyperparameters related to developing and training a model, useful toolkits, and provide a comparison between optimization algorithms.

Enhancing the useful features of image data prior to sending them into the machine learning model can improve the results of the model. This process is known as preprocessing. Ref. [22] provide an overview on image filtering operations done at the preprocessing level. They mainly focus on edge detection filters, smoothing filters, and greyscale operation on images to enhance images to fit for emerging applications in modern vision systems. Ref. [23] also provides an overview on widely used image filtering techniques and their applications.

A rather novel approach on image preprocessing known as Image Super Resolution is discussed by Ref. [3]. They use DL for ISR. The proposed algorithm can directly learn an end-to-end mapping between the low and high resolution images. This mapping provides a deep CNN network to increase the resolution of the input images. Ref. [24] use multiple state-of-the-art ISR methods to enhance medical images and provide a comparison between the experimented ISR architectures.

To address the issue of insufficient and imbalanced training data, ref. [25] focus on using data augmentation methods to improve the datasets. They compare multiple data augmentation methods for image classification models. The focused methods include classical transformations like rotations, cropping, etc., and modern implementations such as Style Transfer and General Adversarial Networks. Ref. [26] provide a survey on data augmentation methods to provide a solution to the overfitting problem due to limited data in medical image applications.

The detection of Region of Interest (ROI) plays an important role in traffic sign recognition applications. YOLO architecture [27] can be used to detect a certain ROI. YOLO approaches object detection as a regression problem instead of a classification problem. It uses a single convolutional network to predict multiple bounding boxes and their class probabilities simultaneously. Each bounding box is predicted using the features from the entire image. YOLO provides real time object detection with high average precision in an end-to-end training process. The network architecture is inspired by the GoogLeNet [10] image classification model. As shown in fig. 4, it has 24 convolutional layers for feature extraction and 2 fully connected layers to predict bounding box coordinates and class probabilities. Instead of the inception modules used by ref. [10], this architecture employs 1×1 reduction layers followed by 3×3 convolutional layers. The final output is a 7×7×30 tensor containing the predictions.

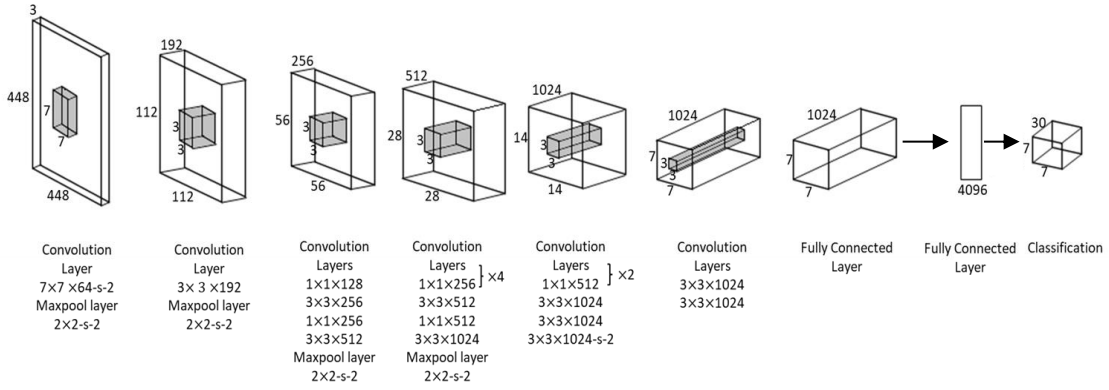


Fig. 4. YOLO network architecture [27]

In general YOLO can process images in real-time at 45 frames per second. A shortened version of the same network known as Fast YOLO has the capability to process images with a rate of 155 frames per second. A modified YOLO network named as YOLOv2 is presented by [28]. The proposed algorithm can detect over 9000 categories in real-time. This introduces a multi-scale training method which offers a tradeoff between detection speed and model

accuracy. A slightly bigger network named YOLOv3 has been presented by [28], which introduces further updates to the YOLOv2 algorithm. This has increased accuracy compared to the previous YOLO networks.

An ROI can be identified by measuring similarities between different objects. Template matching is an object detection process based on this idea. Ref. [29] reviews the basic concepts and applications of template matching. R-CNN introduced by ref. [30] is another state-of-the-art object detection method widely used in modern applications. The high performance of R-CNN is a result of combining classical tools of computer vision and deep learning.

## 3. Materials and Methods

### *3.1 Database Preparation*

One of our main objectives is to introduce autonomous traffic sign recognition systems in the context of Sri Lankan traffic systems. Therefore, the need for a database consisting of local data was essential. Hence, as the first output of this work, we have created a database of Sri Lankan traffic sign images, using dashboard mounted camera (dashcam) recordings of vehicles traveling at 40-50 kmph. Our local traffic sign database consists of both danger warning signs and regulatory signs.

The most commonly occurring traffic sign in Sri Lankan road systems has been recognized as the pedestrian crossing sign with a yellow background and diamond shape, shown in fig. 5 (a). Consequently, the presented prototype is designed to specifically recognize this traffic sign. Later we plan to extend this model to detect other signs through live video feeds.

Accordingly, the three classes for the system were selected as follows:

Class 0: Yellow, diamond shaped pedestrian crossing sign
Class 1: Yellow, diamond shaped other traffic signs
Class 2: Any other traffic sign

Fig. 5 (b) and (c) shows examples of class 1 and class 2 traffic signs from the local dataset. The proposed system is expected to identify signs with different quality levels under different environment variables such as sunlight, shadows and glare. Hence, the dataset is prepared to cover different quality aspects of the traffic images as well as different environment settings. The newly introduced local database consists of a total of 750 images, with 250 images from each class.

(a) (b) (c)

Fig. 5. Sample traffic signs from the local database (generated by the Authors) categorized as (a) Class 0 (b) Class 1 (c) Class 2

When designing a machine learning model, it is important to verify the generalization of its performance. To see if the model adopts correctly in a general setting, we trained and tested the same model using a benchmark standard database as well and compared the results with the local database. The standard database is an abbreviation of the GTSRB database [4].. The original GTSRB database has 43 classes of traffic sign images. We selected 11 of these 43 classes to train and test our model. Sample images from each of these classes are given in fig. 6. The standard database includes 9060 images of traffic signs.


Fig. 6. Sample traffic signs from the standard database, GTSRB [4]

### *3.2 Methodology*

The traffic sign recognition problem consists of two major tasks; detection of ROI, and classification of traffic signs. Since one of the key challenges of designing a real-time live decision making system is the processing time, we have utilized image analysis and detection models and techniques with faster response time. The first ROI detection method tested was template matching [29]. Yet, this approach lacks the automation capabilities and it requires constant user involvement for accurate detection. DL based object detection has high automation potential. The DL based object detectors can be broadly categorized as one-stage detectors and two-stage detectors. Although two-stage detectors

(such as Faster R-CNN, Mask RCNN) have high localization and detection accuracies, the one-stage detectors (such as YOLO, SSD) operate at high inference speeds. Since our proposed solution architecture demands time critical response rate, YOLO [27], a state of the art object detection technique is utilized for the ROI detection. Moreover, we have isolated the object detection and classification architectures in our model to further accelerate the overall processing time. As of the literature, Xception model exhibits superior performance in the classification task compared to the other state of the art CNN architectures such a ResNet, DenseNet and MobileNet [2,31]. Because of this observation, we have selected Xception as our classifier in the proposed architecture.

The proposed prototype system is graphically illustrated in fig. 7. The proposed prototype system includes four main stages: Input feed; detection of ROI; Recognition of the traffic sign and output of the label.

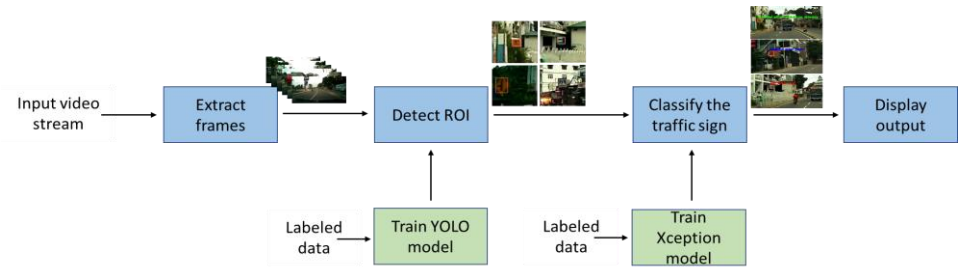


Fig. 7. Flow diagram of traffic sign recognition process

### *3.2.1 Input feed*

The proposed algorithm utilizes a direct video feed from a dashboard mounted camera on a moving vehicle as the input. The input feed is reformatted to 243×243 resolution to ease the complexity at processing. The motion of the vehicle has a tendency to create distortion in the input video. Hence, sharpening and smoothing filtering as shown in fig. 8 are used as pre-processing methods to correct artifacts due to the vehicle movements. For the model training purpose, a custom database was created using extracted frames of the dashcam recordings and was labeled using the labelImg tool.

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |

(a)

| | | |
|---|---|---|
| 0.1107 | 0.1113 | 0.1107 |
| 0.1113 | 0.1119 | 0.1113 |
| 0.1107 | 0.1113 | 0.1107 |

(b)

Fig. 8. (a) Laplacian sharpening filter (b) Gaussian smoothing filter

### *3.2.2 Detection of ROI*

The second stage of the proposed algorithm was the detection of the ROI. At the design stage two approaches were tested as possible solutions for the detection of ROI. Initially, template matching was tried, as it requires less resources at training and testing of the model. But later much advanced YOLO architecture was utilized, as it is of higher detection accuracy than template matching.

*Template Matching:* Template matching was implemented using the python OpenCV function cv.matchTemplate(). Initially, the grayscale traffic sign image shown in fig. 9 was used as the template image for the search. OpenCV library includes several comparison methods for the above mentioned function. They are namely; cv.TM_SQDIFF, cv.SQDIFF_NORMED, cv.TM_CCORR, cv.TM_CCORR_NORMED, cv.TM_CCOEFF, cv.TM_CCOEFF_NORMED. Each of these comparison methods are tested on a single image frame to find the most suitable method for the application. Although the algorithm worked perfectly for a single image frame, it failed considerably with the live video feed as it was unable to generalize the ROI detection for a new set of frames.

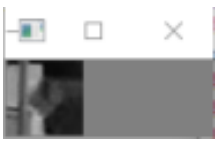

Fig. 9. Template image for template matching

As a secondary approach, the same template matching algorithm was tested for color image. At this instance, separate R, G, B channels of the image as shown in fig. 10 were utilized instead of the greyscale channel. Still the template matching algorithm failed to deliver the desired performance.

(a) (b) (c) (d)

Fig. 10. (a) Original template (b) R channel template (c) G channel template (d) B channel template

*YOLO*: YOLO is a well-known series of end-to-end deep learning models designed for fast object detection with high accuracy. YOLO is pretrained with a large database, which makes it easier to generalize for different applications. In this study, we used the pretrained weights of YOLOv3 architecture trained on COCO dataset [32] with darknet framework [1]. The parameter optimization for subdivision, batch size, number of iterations was carried out at the model initiation stage. The training was done using Google Colaboratory with Graphics Processing Unit (GPU) enabled. The trained model was applied to a dashcam recording clip for testing, which displayed a considerable improvement in the ROI detection.

YOLO can perform both detection and classification. However, it requires a large amount of data and powerful processors to obtain better performance when used for both detection and classification tasks. Considering the low amount of data in the training dataset and the limited resources, authors decided to use a separate machine learning model which can be easily implemented and can be successfully trained with a lesser amount of data for classification.

### *3.2.3 Classification: Recognition of the traffic sign*

When it comes to the task of classification CNN is a popular approach as of literature. CNN has come a long way from the simple LeNet implementation proposed by ref. [5], which adopts a stack layering to architectures which use skip connections with multiple layers. Recently Google has proposed Xception architecture which is an extreme version of Inception with a lesser number of tunable parameters compared to Inception V3. Xception showed improved accuracy in classification with larger datasets and slightly outperformed Inception with imagenet dataset [2]. Also, Xception finds applications in other DL based solutions, such as in image super resolution [33]. Considering all these merits, we have adopted the CNN architecture Xception, in our classification stage.

The experiments have adopted a pre-trained Xception model with the trained weights and have fine-tuned these weights to match the applications. In general, the performance of any machine learning model can be optimized by fine tuning the hyper parameters. We have conducted a manual hyperparameter optimization for the Xception transfer learning model, with a special focus on hyper-parameters, batch size, and the type of the optimizer. The final selections are tabulated in tables 1 and 2.

Two separate instances of the proposed Xception model based classifier was trained using both a customized local dataset and a standard dataset [4] with 50:50 train:test split. The performance of the proposed system was evaluated for each dataset whereas that of the standard dataset is used as a benchmark to support the comparison with the existing models in the literature. Data augmentation and preprocessing techniques were used to further improve the performance of the model.

We have adopted popular data augmentation methods such as random rotation, random contrast changes, random translation and random zoom on our datasets in the training process in order to increase the performance. However, for tasks such as traffic sign classification, data augmentation should be carefully used since the unwanted changes in data may result in false true positives. For example, if a traffic sign image of a left turn was rotated in 180 degrees, it would result in an image of right turn. Hence, the data augmentation techniques were used with care.

Preprocessing is also a popular method used to improve the performance of machine learning models. In this study we have adopted Laplacian sharpening filter (fig. 8 (a)), Gaussian smoothing filter (fig. 8 (b), and ISR as preprocessing techniques. ISR is used to upscale and improve the input resolution of the ROI of the images. We have tested four ISR methods, namely: 'RRDN:gans', 'RDN:psnr-large', 'RDN:psnr-small', and 'RDN:noise-cancel' [34]. As shown in fig. 10, applying ISR has improved the image resolution by 16-fold as well as improved the quality of the input patch. After experimenting with these different preprocessing methods, the method that best contributed to improved performance was incorporated to the final model. The overall training process can be summarized as in the flow diagram given in fig. 11.

### *3.2.4 Overall Traffic Sign Recognition Model*

Once suitable deep learning architectures were developed for the detection and classification tasks independently, particular focus was given to combine these two models with the best overall performance for the complete traffic sign recognition system with low delay. In the final presented system, the two models were installed in series, where input to the overall system is a live video feed from dashboard camera. The first model of the overall system recognizes the ROI and then extracts image patches with possible traffic signs. The second model in the overall system, classifies the extracted ROI image patch into one of the three classes in the database. Next the recognized class label is displayed along with a bounding box around the target area on the output screen. The flow diagram of the overall recognition process is given in fig. 7.

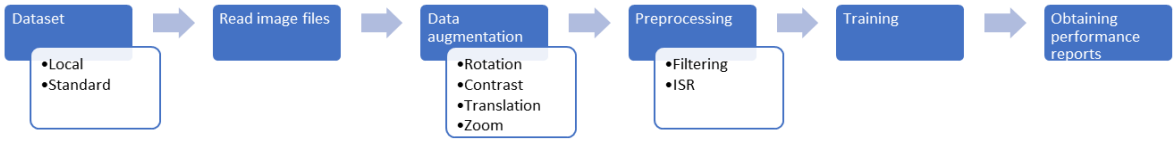


Fig. 11. Training process of the classification model

## 4. Results

### *4.1 Detection of Traffic sign- patches*

#### *4.1.1 Template matching*

At the initial stage of the design, template matching was utilized to extract traffic sign patches from the input stream. The performance accuracy and efficiency were observed for both grayscale images as well as RGB images. A sampled output for template matching for grayscale input image is shown in fig. 12, while those for the RGB color images, are shown in figures 13, 14, and 15. From these results it is evident that the template matching scheme is not highly suitable for automated traffic sign detection purposes.


Fig. 12. Result of template matching with greyscale channel analysis


Fig. 13. Result of template matching with R channel analysis


Fig. 14. Result of template matching with G channel analysis


Fig. 15. Result of template matching with B channel analysis

### *4.1.2 YOLO*

The YOLO model was trained using a manually labeled local dataset, extracted from dashcam footage. At the end of the training process, the average loss was at 0.1569. The related loss diagram is shown in fig. 16. The custom trained YOLO model was able to successfully identify the ROIs from a live RGB video feed accurately with low user interaction. A sample output for the YOLO based traffic sign patch detection is shown in fig. 17. It can be clearly seen that the YOLO model enables multi-object detection with a higher visual accuracy. Owing to the above observations, YOLO was selected to extract ROI, i.e., traffic sign patches, from the input video.

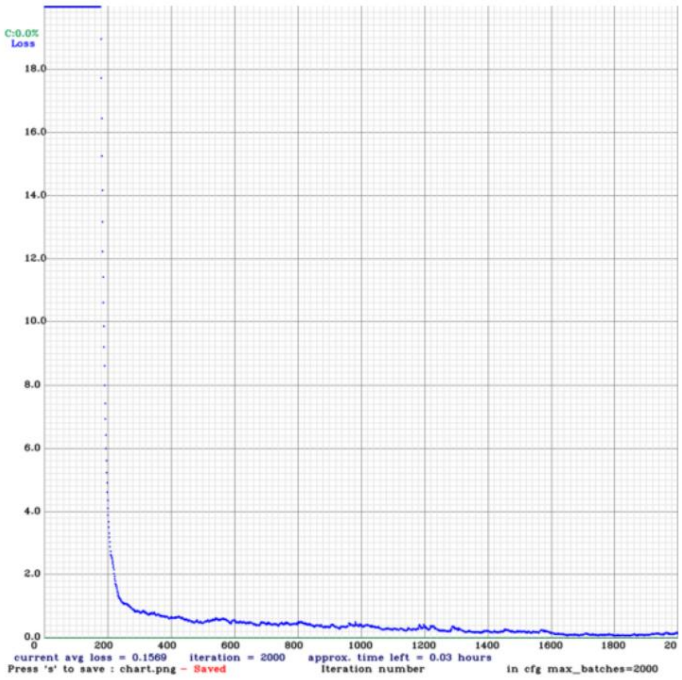


Fig. 16. Loss diagram of YOLO model training


Fig. 17. Detection of ROIs from the custom trained YOLO model

### *4.1.3 Classification of Traffic sign- patches: Xception model*

As stated in section 2, Xception model shows superior performance in classification tasks. Hence in the proposed design we have utilized the Xception model as the backbone architecture for the traffic sign classification. The results of hyperparameter optimization of the Xception transfer learning model for the local dataset are tabulated in table 1.

Table 1. Results of Hyperparameter Optimization for Local Dataset

| Optimizer | Adam | | | RMSprop | | |
|---|---|---|---|---|---|---|
| Batch size | 16 | 32 | 64 | 16 | 32 | 64 |
| Training accuracy | 1.0000 | 0.9996 | 0.9899 | 1.0000 | 0.9997 | 0.9990 |
| Validation Accuracy | 0.9618 | 0.9594 | 0.9570 | 0.9594 | 0.9666 | 0.9618 |
| Training loss | 0.0211 | 0.0399 | 0.0720 | 0.0116 | 0.0217 | 0.0375 |
| Validation loss | 0.1039 | 0.1177 | 0.1383 | 0.0989 | 0.0979 | 0.1127 |

Table 2. Results of hyperparameter optimization for standard dataset

| Optimizer | Adam | | | RMSprop | | |
|---|---|---|---|---|---|---|
| Batch size | 16 | 32 | 64 | 16 | 32 | 64 |
| Training accuracy | 1.0000 | 0.9999 | 0.9989 | 0.9985 | 0.9999 | 0.9994 |
| Validation Accuracy | 0.8102 | 0.7942 | 0.7936 | 0.8030 | 0.7986 | 0.8057 |
| Training loss | 0.0044 | 0.0133 | 0.0294 | 0.0077 | 0.0074 | 0.0126 |
| Validation loss | 0.1817 | 0.8056 | 0.7573 | 1.1382 | 1.0384 | 0.8733 |

The results of hyperparameter optimization of the same model for the standard dataset are given in table 2. At the transfer learning stage, the number of trainable parameters of the model was at 22,539. The training was carried out for 50 epochs with the learning rate fixed at 0.0003.

Table 3. Results of Classification Performance with preprocessing for local dataset

| | | | Training accuracy | Validation accuracy | Training loss | Validation loss |
|---|---|---|---|---|---|---|
| Without preprocessing | | | 0.9305 | 0.8758 | 0.1666 | 0.3622 |
| With preprocessing | Filters | Sharpening | 0.9298 | 0.9038 | 0.1794 | 0.2711 |
| | | Smoothing | 0.9341 | 0.8685 | 0.1622 | 0.3838 |
| | ISR | RRDN:gans | 0.9282 | 0.8465 | 0.1800 | 0.4233 |
| | | RDN:psnr-large | 0.9605 | 0.8733 | 0.1375 | 0.2956 |
| | | RDN:psnr-small | 0.9554 | 0.8258 | 0.1279 | 0.4609 |
| | | RDN:noise-cancel | 0.9569 | 0.8770 | 0.1285 | 0.3206 |

The model was tested with various preprocessing methods for the augmented dataset as presented at Section 3.2.3. The resulting performance for the local dataset is given in table 3, whereas that for the standard dataset is given in table 4.

Table 4. Results of Classification Performance with preprocessing for standard dataset

| | | | Training accuracy | Validation accuracy | Training loss | Validation loss |
|---|---|---|---|---|---|---|
| Without preprocessing | | | 0.9509 | 0.8769 | 0.1993 | 0.4109 |
| With preprocessing | Filters | Sharpening | 0.9537 | 0.8962 | 0.1915 | 0.3411 |
| | | Smoothing | 0.9582 | 0.8907 | 0.1884 | 0.3672 |
| | ISR | RRDN:gans | 0.9463 | 0.8962 | 0.1910 | 0.3660 |
| | | RDN:psnr-large | 0.9859 | 0.9205 | 0.0750 | 0.3003 |
| | | RDN:psnr-small | 0.9848 | 0.9178 | 0.0806 | 0.2934 |
| | | RDN:noise-cancel | 0.9905 | 0.9211 | 0.0723 | 0.2853 |

The model was trained for 50 epochs with a learning rate of 0.0003, and a batch size of 64, with Adam optimizer. With these improvements, the final Xception model has given an accuracy of 96.05% for the local dataset and an accuracy of 92.11% for the standard dataset. In an initial work [35], the authors have shown that a fine tuned support vector machine (SVM) model is able to classify the local dataset in Fig. 5, with an accuracy of 90.54%. From the results presented, it is evident that the proposed Xception model exhibits a superior performance with local dataset than SVM.

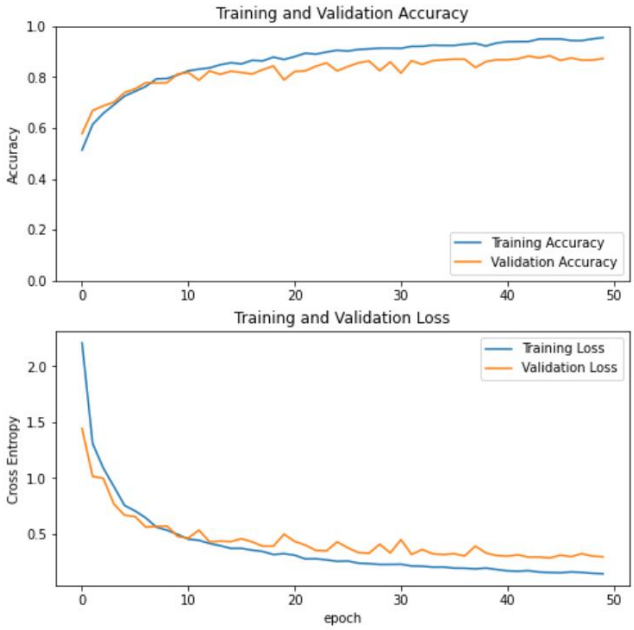


Fig. 18. Accuracy and loss diagrams for the standard

### *4.2 Overall traffic sign recognition system*

The proposed overall traffic sign recognition system presented in fig. 7, is tested with live video feed from dashcam of a moving vehicle with a speed of 50 kmph. The experiment results showed that the combined model for traffic sign recognition was able to successfully detect ROIs and classify them into the three classes in the local database. Successfully recognized traffic signs from Class 0, 1 and 2 from a random input video feed are presented in fig. 19, fig. 20 and fig. 21 respectively to demonstrate the overall performance of the proposed architecture.

Fig. 19. Recognizing traffic signs from Class 0

Fig. 20. Recognizing traffic signs from Class 1

The model was tested in a device with an Intel(R) Core(TM) i7 processor running at 1.80 – 2.30 GHz using 8.00 GB of RAM, running Windows version 10. For each frame with a traffic sign image an average time of 0.2235s was spent for detecting ROI by YOLOv3. The average time for classification by Xception was 0.0456s per frame.


Fig. 21. Recognizing traffic signs from Class 2

The processed frame rate of the output was maintained at 4.5 fps in the tested device, to accommodate the processing time. Although this is somewhat lower than real-time video feed which ranges from 15 fps-30 fps, the processing time can be increased simply by utilizing higher processing devices with GPU capabilities. To accommodate for the drop in the frame rate at processing, we have introduced manual frame skip to the system. That is, in an input feed of 15 fps, we have dropped 2 frames within every 3 consecutive frames. Yet, due to the high correlation present among neighboring frames, we do not lose significant information due to frame dropping. As future scope authors are working on methods to improve the response rate of the prototype system as well as to analyze the input in video context. The source code and the local dataset are available at https://github.com/maheshi81/Traffic_sign_recognition.git .

## 5. Conclusions

In this study, a fully autonomous traffic sign recognition system has been successfully developed using the state-of-the-art machine learning architectures and necessary image pre-processing techniques. Although YOLO can be used for both detection and classification tasks, it requires a significantly larger dataset along with powerful processing capacity to train for both tasks. Hence, the proposed system considers the detection and classification tasks separately. Furthermore, this research presents a new traffic sign dataset built using dashcam recording of a vehicle moving at 40 kmph. The trained YOLO model was able to successfully detect the ROIs from the input video feed. The detected ROI patches were extracted and fed into the Xception model trained using the local dataset for classification. To further improve the model accuracy, preprocessing methods such as smoothing and sharpening filters and ISR were employed. The proposed model was able to successfully recognize traffic signs from a given video feed. Yet, the output frame rate was 4.5 fps. Hence, in future work, we will further explore mechanisms for improved processing time, while retaining the recognition performance accuracy of the model. For example, as of the literature, the latest version of YOLO (i.e., YOLO v7) has the potential to improve the performance of the ROI detection time, which will eventually improve the overall system reaction time of the proposed architecture.

## References

[1] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv*, 2018.
[2] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1800–1807, 2017, doi: 10.1109/CVPR.2017.195.
[3] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, 2016, doi: 10.1109/TPAMI.2015.2439281.
[4] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2011, pp. 1453–1460.
[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Ha, "LeNet," *Proc. IEEE*, no. November, pp. 1–46, 1998.
[6] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.
[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.
[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," pp. 1–18, 2012, [Online]. Available: http://arxiv.org/abs/1207.0580.
[9] M. Lin, Q. Chen, and S. Yan, "Network in network," *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, pp. 1–10, 2014.

[10] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.
[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, 2015.
[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.
[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," *31st AAAI Conf. Artif. Intell. AAAI 2017*, pp. 4278–4284, 2017.
[14] C. Garcia, "Simplifying ConvNets for Fast Learning Simplifying ConvNets for Fast Learning," no. January, pp. 1–8, 2015.
[15] N. Yao *et al.*, "L2MXception: an improved Xception network for classification of peach diseases," *Plant Methods*, vol. 17, no. 1, pp. 1–14, 2021, doi: 10.1186/s13007-021-00736-3.
[16] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.
[17] K. Wang, X. Gao, Y. Zhao, X. Li, D. Dou, and C.-Z. Xu, "Pay Attention to Features, Transfer Learn Faster CNNs," *Iclr*, pp. 1–14, 2019.
[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
[19] M. Hussain, J. J. Bird, and D. R. Faria, "A study on CNN transfer learning for image classification," *Adv. Intell. Syst. Comput.*, vol. 840, no. June, pp. 191–202, 2019, doi: 10.1007/978-3-319-97982-3_16.
[20] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020, doi: 10.1016/j.neucom.2020.07.061.
[21] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv*, pp. 1–56, 2020.
[22] J. Coady, A. O'Riordan, G. Dooly, T. Newe, and D. Toal, "An overview of popular digital image processing filtering operations," in *Proceedings of the International Conference on Sensing Technology, ICST*, 2019, vol. 2019-Decem, no. December, doi: 10.1109/ICST46873.2019.9047683.
[23] B. Desai, U. Kushwaha, and S. Jha, "Image Filtering -Techniques , Algorithm and Applications ISSN NO : 1869-9391 Image Filtering - Techniques , Algorithm and Applications," no. December, 2020.
[24] K. Yamashita and K. Markov, *Medical image enhancement using super resolution methods*, vol. 12141 LNCS. Springer International Publishing, 2020.
[25] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop, IIPhDW 2018*, 2018, no. August 2019, pp. 117–122, doi: 10.1109/IIPHDW.2018.8388338.
[26] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.
[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.
[28] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.
[29] N. S. Hashemi, R. B. Aghdam, A. S. B. Ghiasi, and P. Fatemi, "Template Matching Advances and Applications in Image Analysis," 2016, [Online]. Available: http://arxiv.org/abs/1610.07231.
[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.
[31] M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, "Advances in Deep Learning," vol. 57, pp. 13–29, 2019, doi: 10.1007/978-981-13-6794-6.
[32] T. Y. Lin et al., "Microsoft COCO: Common objects in context," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014, doi: 10.1007/978-3-319-10602-1_48.
[33] Muhammad, Wazir, Supavadee Aramvith, and Takao Onoye. "Multi-scale Xception based depthwise separable convolution for single image super-resolution." Plos one 16, no. 8 (2021): e0249278.
[34] F. Cardinale and D. Tran, "ISR," 2018. https://github.com/idealo/image-super-resolution.
[35] S. Gunasekara, D. Gunarathna, and M. Dissanayake. "Advanced Driver-Assistance System with Traffic Sign Recognition for Safe and Efficient Driving". International Journal on Recent and Innovation Trends in Computing and Communication, vol. 9, no. 9, Sept. 2021, pp. 11-15.

## Authors' Profiles

**Sithmini Gunasekara** received her bachelor's degree with Second Class Honors (Upper Division) in Electrical and Electronic Engineering from the University of Peradeniya, Sri Lanka in 2021. In the same year she joined the Department of Electrical and Electrical Engineering, University of Peradeniya as an instructor. Her research interests are machine learning and communication engineering.

**Dilshan Gunarathna** received his bachelor's degree with Second Class Honors in Electrical and Electronic Engineering from the University of Peradeniya, Sri Lanka in 2021. He currently works at Dialog Axiata PLC, Sri Lanka. His research interests are machine learning and communication engineering.

**Maheshi B. Dissanayake** received the B.Sc. Engineering degree with First Class Honors in Electrical and Electronic Engineering from the University of Peradeniya, Sri Lanka, in 2006, and the Ph.D. in electronic engineering from the University of Surrey, U.K., in 2010. She has been a senior lecturer from 2013 -2021 and a professor from 2021 to date with the Department of Electrical and Electronic Engineering, Faculty of Engineering, University of Peradeniya. She has been a visiting research fellow at King's College London from 2015-2017. Her research interests include error correction codes, robust video communication, molecular communication, machine learning, and biomedical image analysis. She has co-authored nearly 75 conference and journal articles and has a citation record of more than 300. Dr. Dissanayake is a Senior Member of Institute of Electrical and Electronics Engineers (IEEE) and Associate member of Institution of Engineers, Sri Lanka (IESL). She has served as an organizing committee member and TPC Member of many IEEE conferences, and as a reviewer in IEEE journals in the area of molecular communication and image processing.

**Supavadee Aramvith** (Senior Member, IEEE) received the B.S. degree (Hons.) in computer science from Mahidol University, in 1993, and the M.S. and Ph.D. degrees in electrical engineering from the University of Washington, Seattle, USA, in 1996 and 2001, respectively. In June 2001, she joined Chulalongkorn University, where she is currently an Associate Professor at the Department of Electrical Engineering, with a specialization in video technology. She has successfully advised 32 bachelor's, 27 master's, and nine Ph.D. graduates. She published over 130 articles in international conference proceedings and journals with four international book chapters. She has rich project management experiences as a project leader and a former technical committee chair to the Thailand Government bodies in Telecommunications and ICT. She is very active in the international arena with the leadership positions in the international network, such as JICA Project for AUN/SEED-Net, and the professional organizations, such as the IEEE, IEICE, APSIPA, and ITU. She is currently a member of the IEEE Educational Activities Board (EAB) and the Chair of the IEEE EAB Pre-University Education Coordination Committee. She is also a member of the Board of Governors of the IEEE Consumer Electronics Society, from 2019 to 2021. She formerly led Educational Activities and Women in Engineering for the IEEE Asia Pacific (Region 10), from 2011 to 2016.

**Wazir Muhammad** is a lecturer in the Electrical Engineering Department, Balochistan University of Engineering and Technology Khuzdar. He received his doctoral degree from the Department of Electrical Engineering, Chulalongkorn University, Bangkok, Thailand in 2019. He obtained his ME degree in the field of Communication Systems and Networks from Mehran University of Engineering and Technology, Jamshoro, Sindh, Pakistan. His research interests lie in the areas of Electrical Engineering, Communication Systems, Neural Networks, and Machine Learning, specifically in Deep Learning Image Super-Resolution.

**How to cite this paper:** Sithmini Gunasekara, Dilshan Gunarathna, Maheshi B. Dissanayake, Supavadee Aramith, Wazir Muhammad, "Deep Learning Based Autonomous Real-Time Traffic Sign Recognition System for Advanced Driver Assistance", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.14, No.6, pp. 70-83, 2022. DOI:10.5815/ijigsp.2022.06.06