Indian Traffic Sign Classification Using Transfer Learning and Fine Tuning

Varun Remesh
Department of Mathematics
Amrita School of Physical Sciences
Amritapuri, Kollam, India
varunramesh7838@gmail.com

K. Namitha
Department of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Amritapuri, India
namithak@am.amrita.edu

Abstract—**The need for prompt and accurate detection of traffic signs has increased rapidly with the ever-increasing development of autonomous vehicles, driver assistance technologies, and advanced traffic monitoring systems. This study aims to evaluate the efficacy of four convolutional neural network architectures (VGG16, VGG19, ResNet50, and MobileNet) in the context of Indian Traffic Sign Recognition. The models employed transfer learning and fine-tuning to assimilate the distinctive features of Indian traffic signs, thereby reducing training duration and improving their capacity to generalize under challenging settings. All models were tested on the dataset according to their accuracy, loss, recall, and F1-score for recognizing traffic signs under various conditions. In this paper, we introduce a novel methodology to address sign classification in near real-time, enabling autonomous systems and drivers to make informed decisions on the road. This paper addresses the issues posed by Indian traffic signs and elucidates the regional applicability of these models, which may facilitate the advancement of intelligent transportation systems. The experimental findings demonstrated that the accuracies for the four models (VGG16, VGG19, ResNet50, and MobileNet) were 90.73%, 93.33%, 87.90%, and 83.33%, respectively.**

Index Terms—**CNN, Indian Traffic Sign Dataset, traffic sign classification**

I. Introduction

Traffic signs play a crucial role in contemporary road systems, offering vital guidance, warnings, and information to promote road safety and facilitate efficient traffic movement. These signs often require drivers to adjust their behavior, such as reducing speed, changing lanes, or preparing for road conditions ahead, thereby complying with road regulations. Without such vital indicators, drivers might face challenges like increased accidents and unsafe travel conditions. Every year, around 1.19 million fatalities occur on roads around the world; and represent the primary cause of mortality among children and young adults between the ages of 5 and 29 years [1], a number that would be significantly higher without the existence of traffic signs.

In urban environments, the rise of surveillance cameras presents an opportunity to enhance traffic management and safety [2], [3]. Video footage from these cameras can be leveraged not only to monitor traffic states and detect congestion, accidents, or infractions but also to assist in long-term mobility planning and improve citizens' security. This forms the basis of many smart city applications [4]. However, recognizing and classifying traffic signs from real-world images is a challenging task due to variations in lighting, angles, occlusions, and weather conditions [5]. Detecting traffic signs in Indian scenarios becomes complicated, where traffic signs differ in design, placement and maintainance quality as compared to international standards.

This study explores techniques for automatic Indian traffic sign classification by employing transfer learning and fine-tuning as a step toward addressing these challenges. Transfer learning [6], [7], [8] allows the reuse of pre-trained deep learning models, significantly reducing the training time required to achieve high accuracy on the Indian Traffic Sign Dataset. Fine-tuning [9] isolates and retrains certain layers on the dataset to further optimize the already well-trained model specific to the Indian traffic signs. In this report, the effectiveness of different deep learning architectures (VGG16, VGG19, ResNet50, and MobileNet) is compared on the Indian Traffic Sign Dataset [10] to find the most efficient one.

II. Related Work

Mannan et al. [11] proposed an original method to preprocess deteriorated traffic signs using a Gaussian mixture model featuring adaptable split and merge techniques. Additionally, they developed a multiscale CNN that includes a dimensionality reduction layer for the purpose of recognizing traffic signs from German Traffic Sign Detection Benchmark (GTSDB). Jency et al. [12] utilized the publicly available LISA dataset to fine-tune a convolutional neural network architecture that comprised convolutional layers, max-pooling layers, and fully connected layers. Subsequently, the sensitivity of the proposed detection model was assessed using the PASCAL metric.

Safavi et al. [13] trained a CNN model with the GTSDB dataset and then used the same model to classify the signs in Persian Traffic Sign Dataset which classified the images with an impressive accuracy of 97%.

Ferencz and Zöldy [14] trained a Deep ConvNet model on the GTSRB dataset aiming to improve the predictions of a Deep Convolutional Neural Networks based autonomous driving system, achieving a final recognition performance of 97.98%. Satti et al. [15] presented the Indian Cautionary Traffic Sign (ICTS) dataset containing 19,775 samples belonging to 40 different classes collected in different environmental conditions. Different models, including VGG16, LeNet, ResNet, and AlexNet, were used by the authors for sign classification. They assessed and contrasted the performance using various metrics, including specificity, F-measure, precision, recall, and accuracy.

Vennelakanti et al. [16] introduced a classification model for traffic signs that used a hybrid dataset, employing image processing techniques for sign detection alongside an ensemble of Convolutional Neural Networks (CNN). This model exhibited recognition accuracies exceeding 99% for circular signs within the Belgian and German datasets. Zhu et al. [17] created a new dataset containing 100,000 high-resolution images with 30,000 annotated traffic sign instances from Tencent Street View panoramas and also annotated them with bounding boxes, pixel masks, and class labels. An impressive convolutional neural network was designed to simultaneously identify and categorize traffic signs, effectively tackling challenges associated with small object recognition. This approach also enhanced the precision of both detection and classification in comparison to previously established techniques.

III. Methodology

Figure 1 illustrates the block diagram representing how the models were employed.

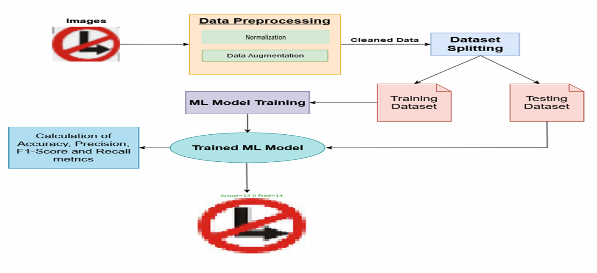

Fig. 1. Block diagram for the models used

A. Dataset

The research uses the "Indian Traffic Sign Dataset," which includes 52 different classes. Each class is represented by an image sourced from government websites, and the images have undergone prior scaling. The dimensions of the images within this dataset are 32x32x3.

Indian traffic signs are broadly classified into 3 categories [18]:

1) **Mandatory signs** [18] are those that users must adhere to without exception, as compliance is required by law. Ignoring these signs constitutes an illegal act and may result in penalties.
2) **Cautionary or warning signs** [18] serve to inform drivers of potential hazards, including reduced speed limits, construction activities, dangers, or accidents that may lie ahead.
3) **Information signs** [18] play a crucial role for road users by guiding them in the correct direction and ensuring they are on the appropriate route to their desired destination. Additionally, these signs offer valuable details regarding nearby services for travelers, such as dining options, accommodations, rest areas, restroom facilities, and fuel stations.

Table I presents the relevant information about the dataset being used in the study.

TABLE I
Dataset Information

Dataset	Total Images	Classes	Data Types	Training	Validation	Testing
ITSD	13,653	52	PNG	10,903	1334	1416

Table II presents a CSV file that enumerates class names alongside their corresponding class IDs.

TABLE II
Labeling Images with their respective Class IDs

ClassId	Name	ClassId	Name
0	Give way	26	Narrow road
1	No entry	27	Narrow bridge
2	One-way traffic	28	Unprotected quay
3	No vehicles in both directions	29	Road hump
4	No entry for cycles	30	Dip
5	No entry for goods vehicles	31	Loose gravel
6	No entry for pedestrians	32	Falling rocks
7	No entry for bullock carts	33	Cattle
8	No entry for hand carts	34	Crossroads
9	No entry for motor vehicles	35	Side road junction
10	Height limit	36	Oblique side road junction
11	Weight limit	37	T-junction
12	Axle weight limit	38	Y-junction
13	Length limit	39	Staggered side road junction
14	No left turn	40	Roundabout
15	No right turn	41	Guarded level crossing ahead
16	No overtaking	42	Unguarded level crossing ahead
17	Maximum speed limit (90 km/h)	43	Level crossing countdown marker
18	Maximum speed limit (110 km/h)	44	Parking
19	Horn prohibited	45	Bus stop
20	No parking	46	First aid post
21	No stopping	47	Telephone
22	Turn left	48	Filling station
23	Turn right	49	Hotel
24	Steep descent	50	Restaurant
25	Steep ascent	51	Refreshments

B. Data Preprocessing

1) Image Rescaling and Normalization: The dataset comprises images with a fixed aspect ratio of 32×32. To optimize model performance and maximize classification accuracy, it was essential to determine a suitable pixel ratio that would allow the models to effectively learn critical features.

To achieve this, experiments were conducted across various image sizes for each model to evaluate the trade-off between computational efficiency and classification accuracy. Based on these experiments, the optimal image size was assigned to each model, as detailed in Table III. Furthermore, all images were standardized to a range of [0, 1]..

2) Data Augmentation: A disparity was noticed in the distribution of data among the 52 classes. For example, some images for a particular category were very high, whereas others did not. There was also not much variability in some image categories.

So, to include much more diverse images, various augmentation techniques were applied such as translation, random rotation, flipping, scaling, and affine transformations.

Augmentation introduced variations in position, orientation and viewpoint, thus helping the models take in much more distinct traffic scenarios for training.

IV. Models Used

A. VGG16

Simonyan et al. [19] developed the VGG16 architecture, which is a deep convolutional neural network distinguished by its use of small (3,3) convolutional filters and comprises a total of 16 layers with trainable parameters. This architecture attained one of the highest classification accuracies on the ImageNet [20] dataset and has established a standard for image classification tasks. The use of 3×3 kernels throughout the network allows VGG16 to generalize well to other datasets and classification problems, even those it was not explicitly trained on.

The first two convolutional layers (CONV) of the VGG16 architecture are meant to learn 64 (3×3) filters, after which a ReLU activation takes place. Then a max-pooling layer (POOL) applies the pooling operation with a window of 2×2 and a stride of 2 Two more sets of CONV layers follow, raising the number of filters to 128, then 256, with pooling layers in between. This continues until the final CONV layers, which use 512 filters.

The network is comprised of three fully connected layers that follow the convolutional layers. The initial two fully connected layers consist of 4096 neurons each, while the last layer has a quantity of neurons that corresponds to the total number of classes. A SoftMax function is utilized at the end to produce class probabilities. To bolster generalization and counteract overfitting, regularization strategies, including batch normalization and dropout layers, are incorporated following the pooling and FC layers. The initial layers are designed with a limited number of filters (64), while the deeper layers are equipped with an increased number of filters (512) to capture more complex features. Pooling layers effectively down-sample the spatial dimensions, facilitating the network's ability to process high-resolution images.

B. VGG19

VGG19 was designed as an advancement within the Visual Geometry Group's convolutional neural network family [19], building on the VGG16 framework by increasing the network's depth to 19 layers of adjustable weights. VGG19 has been extensively utilized in a variety of computer vision applications. The architecture's reliance on small 3×3 convolutional filters allows for the extraction of detailed hierarchical features, promoting effective generalization across diverse datasets.

The VGG19 architecture begins with two CONV layers learning 64 filters, followed by a ReLU activation and a POOL layer. The subsequent layers increase the filter count to 128, 256, and finally 512 filters, with two or four CONV layers per block, each followed by ReLU activations and pooling. This structured stacking allows VGG19 to capture more nuanced features than its predecessor.

The network concludes with three fully connected layers. The initial two layers contain 4,096 neurons each, and the last layer corresponds to the number of output classes. A SoftMax function generates class probabilities. To mitigate overfitting, regularization techniques such as batch normalization and dropout layers are incorporated subsequent to the pooling and fully connected layers.

Like VGG16, the filter counts start at 64 for the initial layers and grow to 512 for the deeper layers. The POOL layers function on 2×2 windows with a stride of 2, gradually decreasing the spatial dimensions.

C. Resnet50

ResNet50, introduced by Kaiming He et al. [21], is part of the Residual Neural Network (ResNet) family that revolutionized deep learning by addressing the degradation problem associated with training very deep networks. ResNet50 comprises 50 layers of trainable weights and employs residual learning to ease optimization, making it significantly deeper yet efficient compared to earlier architectures like VGG19. ResNet50 features a modular architecture which begins with a 7×7 convolutional layer that utilizes 64 filters, succeeded by max pooling to reduce the dimensions of the input image. The next part includes residual blocks made up of bottleneck blocks, which have the following components:

1×1 Convolution: For reducing dimensionality.

3×3 Convolution: To capture spatial features.

1×1 Convolution: Aimed at restoring dimensionality.

These blocks are arranged in four stages, with progressively increasing filter counts of 64, 128, 256, and 512.

Shortcut connections perform element-wise addition between the input and output of the bottleneck block, ensuring gradient flow even in deep networks. After global average pooling, the network outputs predictions through a fully connected layer with a SoftMax activation.

D. MobileNet

Howard et al. [22] developed MobileNet, a convolutional neural network tailored for mobile and integrated vision applications. It incorporates depthwise separable convolutions, which facilitate the creation of a lightweight and high-performance model that operates with low latency and reduced computational demands, thereby making it well-suited for environments with constrained resources.

MobileNet replaces standard convolutions with depthwise separable convolutions, dividing the process into:

- Depthwise Convolution: Applies an individual filter to each separate input channel.
- Pointwise Convolution: Involves the combination of outputs from the depthwise layer utilizing 1×1 convolution techniques.

This factorization reduces the computational cost significantly while maintaining competitive accuracy.

It has two hyperparameters for customization; Width Multiplier (α) which scales the number of channels in each layer, allowing adjustment of the network's size and computational cost and Resolution Multiplier (ρ) which reduces input image size, decreasing the internal feature map resolution for faster inference.

V. Results and Discussion

A. System Configuration

The execution of the codes were done on Google Colab, leveraging an NVIDIA Tesla T4 GPU which utilized an NVIDIA Tesla T4 GPU with a memory of 16 GB. The environment provided was based on Python 3.7, facilitating the implementation of the study. The training of the pre tuned models was carried out utilizing the Keras framework, with TensorFlow-GPU operating as the backend. Furthermore, OpenCV3 was utilized for the tasks of image pre-processing and data augmentation.

The dataset contained 13,653 traffic sign samples for use in training and 10,903 samples reserved for testing. These samples were collected over a variety of environmental conditions, including daytime, low-light, and nighttime conditions. Various qualitative metrics were used to analyze the performance of the model (specifically recall, precision, accuracy and F-measure).

Table III lists the hyperparameters used in training the CNN models. Initial training and evaluation of models were performed using the proposed dataset. After this, they were fine-tuned on the ITSD dataset, and their performances were compared for validation.

TABLE III
HYPERPARAMETERS USED TO TRAIN THE MODELS

Model	Image Size	Learning Rate	Batch Size	Epoch	Verbose	Optimizer
VGG16	32×32	0.0001	32	20	1	Adam
VGG19	50×50	0.00001	32	30	1	Adam
Resnet50	64×64	0.00001	32	30	1	Adam
MobileNet	64×64	0.0001	32	20	1	Adam

B. Evaluation Parameters

The classification models were evaluated on several metrics specifically recall, precision, accuracy and F-measure. The following abbreviations were used for true positive, false positive, true negative, false negative: T_P, F_P, T_N, F_N.

a) **Precision**
Precision, defined in (1) as a ratio of true positives (T_P), which illustrates the exact positive predictions produced by the model in connection with the total positive predictions generated by the model. This measure gives information regarding how well the model is reducing false positives (F_P), thus verifying that the respective traffic sign is correctly recognized.

$$\text{Precision} = \frac{T_P}{T_P + F_P} \tag{1}$$

b) **Recall**
This metric represents the proportion of data points identified as positive in relation to the total actual positive instances, specifically the True Positives (T_P) divided by the overall positives. The corresponding mathematical expression is provided in (2).

$$\text{recall} = \frac{T_P}{F_N + T_P} \tag{2}$$

c) **F-Measure**
The F-measure harmonizes the relationship between recall and precision. The mathematical notation is presented in (3). This metric provides a more nuanced perspective on the model's performance by assessing both precision and recall. A high F score signifies that the model demonstrates robust performance, adeptly balancing precision (Pr) and recall (Rc).

$$F \text{ measure} = \frac{2 \times Rc \times Pr}{Pr + Rc} \tag{3}$$

d) **Accuracy**
The accuracy metric measures how many predictions were correctly compared to the total number of examples that were evaluated, as shown in (4). This metric provides a holistic view of the model's accuracy, acknowledging the total proportion of well-categorized instances

$$\text{Accuracy} = \frac{T_N + T_P}{F_P + T_P + F_N + T_N} \tag{4}$$

C. Results

TABLE IV
COMPARITIVE PERFORMANCE ANALYSIS WITHOUT FINE TUNING

Model	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
VGG19	82.74	83.35	82.26	82.80
MobileNet	67.40	68.82	69.27	69.04
VGG16	78.70	79.51	79.66	78.58
ResNet50	72.91	74.02	73.73	73.88

Initially, the necessary packages for training the convolutional neural network models were imported. The parameters used in the model preparation are detailed in Table III. Following this, The models were trained and assessed with the help of the Indian Traffic Sign Dataset. The data was input into the models without any tuning, and the metrics of precision, accuracy, f-measure and recall were documented in Table IV.

TABLE V
COMPARITIVE PERFORMANCE ANALYSIS USING FINE TUNING

Model	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
VGG19	93.33	94.13	94.49	94.11
MobileNet	83.33	84.65	83.33	83.02
VGG16	90.73	92.82	91.52	90.87
ResNet50	87.90	89.43	89.05	89.05

Then each of the models were fine tuned using trial and error before finally arriving at a configuration that gave the highest performance. The graphs depicting the training and validation accuracy for the fine-tuned VGG19, MobileNet, VGG16, and ResNet50 are shown in the figures 2, 3, 4, and 5, respectively.

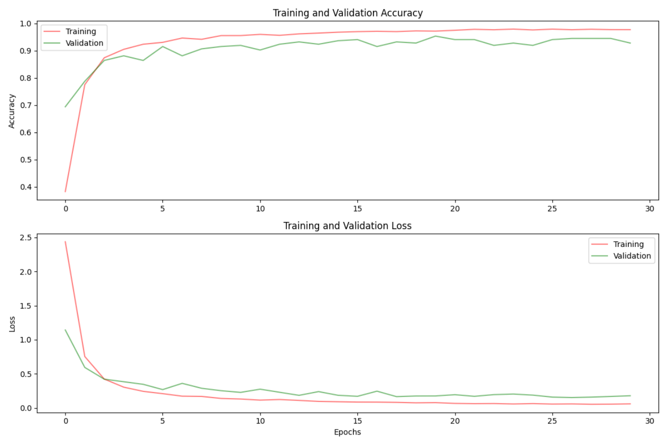

Fig. 2. Training and Validation Accuracy and Training and Validation Loss for VGG19

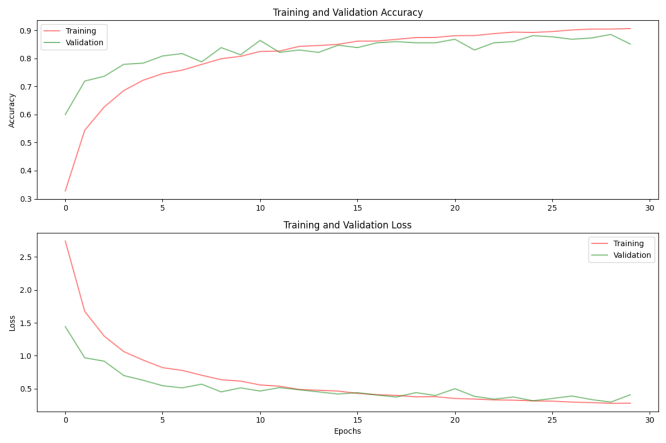

Fig. 3. Training and Validation Accuracy and Training and Validation Loss for MobileNet

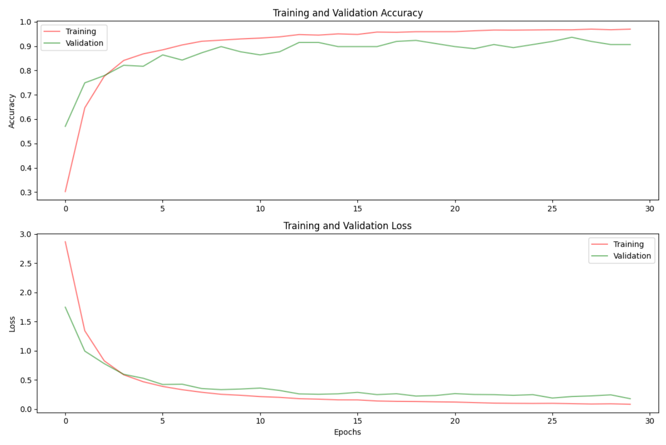

Fig. 4. Training and Validation Accuracy and Training and Validation Loss for VGG16

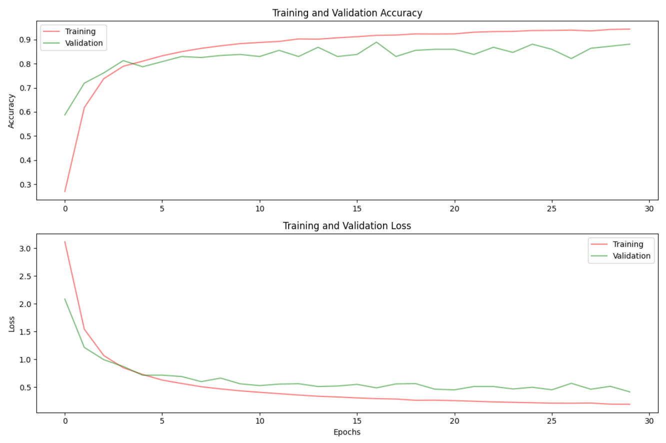

Fig. 5. Training and Validation Accuracy and Training and Validation Loss for Resnet

The data presented in Table V indicates that the VGG 19 model outperforms all other models in terms of F-measure, recall, precision and accuracy. Additionally, Figure 6 illustrates several example output predictions generated by the fine-tuned VGG19 model. VGG19 performing better can be attributed to several factors. Firstly, VGG19 contains a deeper architecture compared to VGG16, with 19 layers consisting trainable weights, enabling it to learn more complex and hierarchical features that are critical for distinguishing between the fine-grained variations present in traffic sign images. The consistent use of small (3,3) convolutional filters throughout the network allows the VGG19 model to effectively capture local textures and edge details. Secondly, compared to ResNet50, although ResNet benefits from residual connections, its greater depth and architectural complexity may lead to overfitting or convergence challenges on smaller datasets like ITSD. VGG19's straightforward and uniform architecture is thus better for effective fine-tuning in this context. Thirdly, while MobileNet offers computational efficiency, its reduced parameter count results in a lower capacity for feature extraction, which limits its performance relative to VGG19.

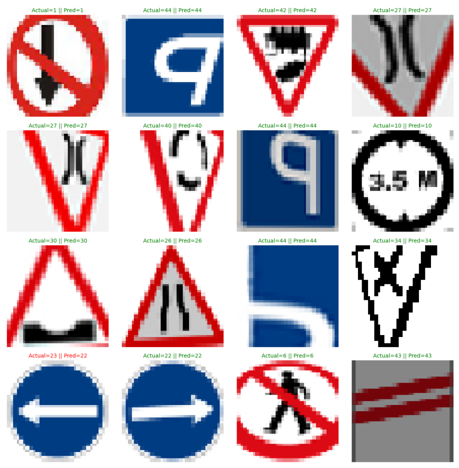

Fig. 6. Sample Outputs for VGG19 Model

VI. Conclusion

This study addressed the application of deep learning models for Indian Traffic Sign Recognition, employing transfer learning and fine-tuning methods on four CNN architectures: VGG16, VGG19, ResNet50, and MobileNet. The models were evaluated on a heterogeneous dataset of Indian traffic signs across different situations. VGG19 surpassed the other models across all evaluation metrics, establishing it as an optimal choice for practical application. The results underscore the capability of deep learning systems to improve road safety and traffic management, particularly for autonomous vehicles and driver assistance systems in areas with distinctive traffic signage, such as India. The models can be potentially used for real-world applications and for semi-autonomous vehicles. Integration with smart city traffic monitoring can also be done, where surveillance footage is assessed to monitor sign visibility. Future works may concentrate on enhancing model robustness, utilizing more extensive datasets, and incorporating detection and recognition into a comprehensive system, thereby facilitating more intelligent and secure transportation solutions in India.

References

[1] W. H. Organization, *Global status report on road safety 2018*. World Health Organization, 2019.

[2] K. Namitha, M. Geetha, and N. Athi, "An improved interaction estimation and optimization method for surveillance video synopsis," *IEEE MultiMedia*, vol. 30, no. 3, pp. 25–36, 2022.

[3] V. Sood, S. K. Ray, K. K. Mahato, A. Sidharth, and K. Namitha, "A graph-based approach for estimating object interactions in surveillance video synopsis," in *2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE)*. IEEE, 2024, pp. 1–6.

[4] Y. Revanth and K. Namitha, "License plate recognition-based automatic gate opening system," in *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*. IEEE, 2024, pp. 1–6.

[5] W. Wang, X. Wu, X. Yuan, and Z. Gao, "An experiment-based review of low-light image enhancement methods," *Ieee Access*, vol. 8, pp. 87 884–87 917, 2020.

[6] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," *Journal of Big Data*, vol. 9, no. 1, p. 102, 2022.

[7] A. Amruth, R. Ramanan, C. Vimal, and B. Beena, "Deep learning solutions for real-world traffic sign recognition: A transfer learning approach," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2024, pp. 1–10.

[8] K. M. Mithra, P. A. George, S. J. Abit Sai, S. Abhishek, and A. T, "Transfer learning and custom cnns to advance traffic sign detection," in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, 2023, pp. 1–7.

[9] S. Academy, "Fine-tuning an image classification model in keras," 2025, accessed: 4 October 2024. [Online]. Available: https://www.scaler.com/topics/fine-tuning-an-image-classification-model-in-keras/

[10] R. K. Megalingam, K. Thanigundala, S. R. Musani, H. Nidamanuru, and L. Gadde, "Indian traffic sign detection and recognition using deep learning," *International journal of transportation science and technology*, vol. 12, no. 3, pp. 683–699, 2023.

[11] A. Mannan, K. Javed, A. Ur Rehman, H. A. Babri, and S. K. Noon, "Classification of degraded traffic signs using flexible mixture model and transfer learning," *IEEE Access*, vol. 7, pp. 148 800–148 813, 2019.

[12] S. Jency, S. Karthika, J. Ajaykumar, R. Selvaraj, and A. Aarthi, "Traffic sign recognition system for autonomous vehicles using deep learning," in *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2023, pp. 1116–1121.

[13] S. M. Safavi, H. Seyedarabi, and R. Afrouzian, "Persian traffic sign classification using convolutional neural network and transfer learning," *Arabian Journal for Science and Engineering*, vol. 50, no. 2, pp. 775–784, 2025.

[14] C. Ferencz and M. Zöldy, "Neural network-based multi-class traffic-sign classification with the german traffic sign recognition benchmark," *Acta Polytechnica Hungarica*, vol. 21, no. 7, 2024.

[15] S. K. Satti, K. S. Devi, K. Sekar, P. Dhar, and P. Srinivasan, "Icts: Indian cautionary traffic sign classification using deep learning," in *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*. IEEE, 2022, pp. 1–7.

[16] A. Vennelakanti, S. Shreya, R. Rajendran, D. Sarkar, D. Muddegowda, and P. Hanagal, "Traffic sign detection and recognition using a cnn ensemble," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, 2019, pp. 1–4.

[17] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2110–2118.

[18] Kerala Motor Vehicle Department, "Traffic signs," 2024, accessed: 12 December 2024. [Online]. Available: https://mvd.kerala.gov.in/en/traffic-signs

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861