Evaluating Lightweight and Deep CNN Architectures for Traffic Sign Recognition

Yoel Augustan
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 14240
yoel.augustan@binus.ac.id

Clariant Benedictus Tan
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11830
clariant.tan@binus.ac.id

Ivan Sebastian Edbert
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
ivan.edbert@binus.ac.id

Alvina Aulia
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia 11480
aaulia@binus.edu

***Abstract*—Traffic sign recognition (TSR) is a critical component of autonomous driving and Advanced Driver Assistance Systems (ADAS) that demands a careful balance between computational efficiency and classification accuracy. this paper evaluates two convolutional neural network (CNN) architectures: MobileNetV3-Large (lightweight CNN) and ResNet50 (deep CNN) on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Both models were trained making use of a two-phase transfer learning technique that involves fine-tuning followed the freezing of basic layers. Accuracy, F1-score, inference time, and model complexity were the metrics used to evaluate of performance. ResNet50 achieved superior classification performance with 94.16% test accuracy and a 94.29% weighted F1-score, compared to MobileNetV3's 92.19% accuracy and 92.02% F1-score. However, MobileNetV3 excelled in GPU inference speed, making it more appropriate for real-time applications. These findings reveal a crucial trade-off for ADAS implementation, ResNet50's depth provides higher reliability, especially for challenging sign classes, making it ideal for systems where maximum accuracy is paramount. In contrast, MobileNetV3's efficiency is essential for deployment on resource-constrained hardware. This analysis offers direct guidance for developers in selecting an optimal architecture that aligns with specific system constraints and safety requirements in autonomous technology.**

Keywords— ***Traffic Sign Recognition, MobileNetV3, ResNet-50, Real-Time Detection, Computational Efficiency, Sustainable Transportation***

I. Introduction

Traffic sign recognition (TSR) is an important component of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. Traffic signs convey various pieces of information such as road laws, traffic conditions, warning signs for hazards, driving guidelines for vehicle safety, and much more [1]. This information is essential for maintaining driver safety and making traffic processes overall efficient. As shown in figure 1, an estimated 1.19 million individuals lost their lives from traffic-related occurrences in 2021 according to World Health Organization (WHO) reports as shown in Figure 1 [2]. With traffic sign recognition, ADAS and autonomous vehicles can substantially lower these statistics.

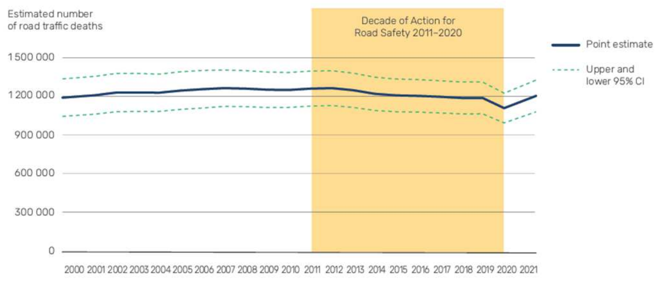

Fig. 1. *WHO estimated number of road traffic fatalities, 2000–2021*

To identify traffic signs, most early approaches of traffic sign recognition depended on machine learning and template matching. Early on, researchers applied machine learning techniques such as SVM (Support Vector Machines) and Random Forest [3]. These early traditional methods needed manual feature extraction, which was time-consuming, and struggled to achieve fine-grained classification and differentiate between traffic signs with similar characteristics. Here, manual features include shape-based segmentation and colour-based segmentation. Colour-based segmentation applied technologies like RGB (Red Green Blue), HSI (Hue Saturation Intensity), and HSV (Hue Saturation Value) to obtain the traffic sign ROI (Region of Interest). Performance for colour-based segmentation could decrease if it meets poor lighting and different traffic signs with similar colours [4]. Shape-based segmentation, which is also used in detecting traffic signs, uses techniques such as Hough Transform and Harris Detector to detect frequently used sign shapes, such as circles, triangles, and rectangles. However, this segmentation has its limitations when the sign is partially occluded or distorted [5]. Template matching was also commonly used for traffic sign recognition; it involved comparing the sign to a prepared template database. Like shape-based segmentation, it has difficulty recognizing signs with distorted shapes [6].

Traffic sign recognition presents more difficulties other than lighting, obstacles, and extreme weather. CNN is very accurate, but it is difficult for real-world applications since it requires significant computational capability. Lightweight CNN improves efficiency to tackle this, but it still lags behind in terms of speed compared to other models [7]. In this research, the researcher will investigate traffic sign recognition between lightweight CNN and deep CNN, both similarities and differences. We will evaluate the computational efficiency, processing speed, and recognition accuracy, as well as the time needed. This research will help us to determine which approach best fits real-life traffic sign detection.

II. Literature Review

Due to the limitations machine learning faces, researchers moved to a deep learning approach, which could automatically learn the sign features. One of the popular deep learning approaches is CNN. A model was developed that combines both traditional and deep learning methods, making use of Hough Transform and CNN. [1]. A small CNN model was proposed that uses multichannel convolutional kernels and reduces the number of parameters compared with traditional CNN [6]. These models reach high accuracy, but they struggle in real-time detection. A lightweight CNN is later introduced to combat real-time applications. By reducing the model size, it uses MicroNet for embedded systems to achieve a balance in efficiency and accuracy and applies multi-stage CNN-based classification to improve extraction. This model achieved a result of 98.41% accuracy on GTSRB, but this model uses augmented techniques to expand the dataset's size, which does not fully replicate real-world conditions [7].

Recent advancements have been made for lightweight CNNs, DeepThin was then proposed, which was a CNN optimized to run without a GPU. On the GTRSB dataset, this model achieved a result of 99.40% on the grayscale model, 99.42% on the RGB model, and 99.73% on the ensemble model. However, this model has its limitations; it confuses similar classes, and lighting conditions impact the model's performance [8]. Zaibi et al. [9] and An et al. [10] develop a lightweight CNN, leveraging the classic LeNet-5 framework. Zaibi et al. enhance the model by reducing the parameters while maintaining high accuracy. On the other hand, An et al. modify the model by balancing efficiency and resource requirements. Both of these models receive an accuracy of 99.84% and 97.53%, respectively, on the GTSRB dataset.

Khalifa et al. [11] proposed LE-CNN, a lightweight and efficient CNN that incorporates depth-wise separable convolutions and channel pruning. It achieves 96.5% accuracy in ArTS (Arabic Traffic Sign). However, this model ignores whole end-to-end latency, including capture, preprocessing, and collision avoidance, which is essential for real-world application. Xi et al. [12] introduced IECES-network, an efficient CNN-based encoder with a Siamese neural network, which enhances robustness to motion blur and occlusions. It achieves a result of 86.1% accuracy with motion blur and occlusions images.

With the increasing popularity of deep learning, researchers investigated several deep CNN architectures to make traffic sign recognition more accurate and robust, particularly in challenging real-world situations. Triki et al. [13] implemented a real-time system for traffic sign recognition using an attention-based deep CNN in combination with the Haar cascade detection. The model achieves an accuracy of 99.91% using the GTSRB dataset. However, the model has difficulty identifying rare or unseen traffic signs and might fail to generalize to various real-world settings. Benfaress et al. [14] proposed an explainable deep CNN architecture using the technique of Grad-CAM. They achieve an accuracy of 99.62% without data augmentation and 99.06% with data augmentation in TT100K dataset. Even with high accuracy, the model struggles with occlusions, environmental variation, and deployment in low-resource hardware.

Bi et al. [15] proposed a model by fine-tuning the VGG-16 architecture, reducing the number of convolutional layers and parameters, and using a global average pooling (GAP) layer and a batch normalization (BN). The improved model achieved 99.47% on GTSRB. On the other hand, Zhang et al. [16] improved the LeNet-5 by using Gabor filters, BN layer after each convolution, and replace the Sigmoid function with the ReLU. Combined with SVM for classification, it reached an accuracy of 96% on the expanded GTSRB dataset. At the same time, Huo et al. [17] presented a Deep Mutual Learning (DML) technique that used dual ResNet-20 networks that used Kullback-Leibler alignment to learn from each other simultaneously. When tested with GTSRB, the introduced model achieved an accuracy of 99.612%.

Particularly in embedded systems like ADAS, recent developments in CNN-based traffic sign recognition (TSR) place a strong emphasis on striking a balance between accuracy and real-time deployment. By introducing basic CNN-based architectures for classification using softmax, early works like those by Prasanna et al. [18] established benchmarks for constrained traffic datasets and laid the groundwork for future research.

Meng et al. [19] proposed SOS-CNN, a VGG-16-based model that slices large images into overlapping patches, in response to the need to detect smaller traffic signs. This model greatly increases the detection rates of small objects. In the meantime, a scale-aware CNN architecture tailored for traffic sign recognition was introduced by Yang et al. [20] In order to improve classification accuracy and robustness against different sign sizes, their model combined global and local contextual features to address scale variance in traffic signs.

Regarding architectural innovation, Xi et al. [21] introduced a Siamese network that improves recognition between visually similar sign classes by utilizing an effective CNN encoder. When tested on intricate datasets like GTSRB, this method proved to be successful. In a similar vein, Song et al. [22] presented E-MobileViT, a low-latency environment-specific lightweight vision transformer hybrid optimized with MobileNetV2 backbones. The goal of both models is to lower computational requirements without sacrificing detection reliability.

Wei et al. [23] combined multi-scale feature fusion and attention mechanisms in a CNN framework to further optimize feature representation. This demonstrated its potential for practical implementation by improving performance in dimly lit and obscured environments. Zhang et al. [24] achieved near-state-of-the-art results on lightweight devices by compressing deep models without sacrificing much accuracy through the use of techniques like quantization-aware training and knowledge distillation.

III. METHODOLOGY

3.1. Dataset

In this study, the research will make use of the widely used benchmark for traffic sign classification, the German Traffic Sign Recognition Benchmark (GTSRB) dataset [25]. There are over 50,000 images of different resolutions, lighting, and backgrounds, distributed across about 40 classes. However the dataset has its limitations, it only includes German sign standards, doesn't include heavy rain, snow, or night conditions, and doesn't include many heavily occluded signs. These controlled settings are good for our research goal since they make sure that the differences in performance we see between architectures are due to real model capabilities and not changes in the environment.

During the preprocessing stage, 80% of these photos were used to train and 20% were used to test. The images were resized from their original size to 224 by 224 pixels to fit the model output. To increase generalization, the researcher implements data augmentation techniques, including random rotation, zoom, brightness adjustment and horizontal flip. This guaranteed that the model acquired strong features in addition to preventing overfitting probability.

3.2. MobileNet

For the lightweight CNN, the researcher employed MobileNet, the MobileNetV3-Large, which is optimized for mobile as well as embedded applications, making the CNN appropriate for real-time applications [26]. Pretrained weights from ImageNet were used by the researcher for initializing the model, and then the researcher fine-tuned the model with the help of the GTSRB dataset. The original head of classification is replaced by a new dense layer of 43 output neurons, according to the number of classes of traffic signs, followed by a softmax function to produce probability distribution over the classes. In training, the base layers of the network were frozen so that the general features from ImageNet are retained, only the new classification head is trained. In the later stages, top layers of the base model are unfrozen and fine-tuned with a lower learning rate for better performance [27].

3.3. ResNet

Representing the deep CNN model, this study implements a ResNet-50 architecture following the approach by Antony et al. [28], who successfully applied ResNet for traffic sign recognition. ResNet-50, a deep residual network with 50 layers, utilizes skip connections to mitigate vanishing gradient problems, enabling more efficient and effective learning in deeper networks.

The model was adapted by replacing the final classification layer with a 43-unit dense layer followed by softmax activation to match the number of traffic sign classes. Input images were resized to 224×224 pixels and augmented similarly to MobileNet. Training was conducted using the Adam optimizer with a learning rate of 0.001 and a batch size of 64 over 50 epochs. This configuration balances high classification performance with computational feasibility, making it suitable for evaluating deep CNN capabilities on the GTSRB dataset.

TABLE I

Classification Head Architecture

Layers	Parameters
Base Model	Feature maps from backbone
GlobalAveragePooling2D	No parameters
Dense (Hidden)	128 units, ReLU activation
Dropout	Rate = 0.3
Dense (Output)	43 units, Softmax activation

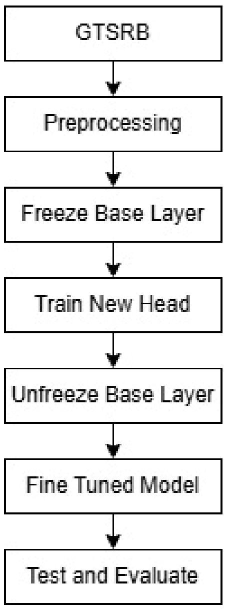

Fig. 2. *Training Workflow using Transfer Learning*

3.4. Training and Evaluation

All of the experiments were done on Kaggle's cloud computing platform using a Tesla P100 GPU with 16GB of VRAM and 13GB of RAM for training. For inference testing, both Tesla P100 GPUs and Intel Xeon processors were used to get a full picture of performance. The models in this study were trained using the Adam optimizer with default parameters, which is well-known for both its efficient convergence and adaptive learning rate. It is set to have the learning rate of 0.0001, so there are steady and stable updates of pretrained weight without overfitting of previously learned features. Sparse Categorical Cross-Entropy is also applied, since it is best suited for multi-class classification problems. The batch size is set to 64 to balance memory usage with efficiency. To avoid overfitting and ensure generalization, early stopping with a patience of 5 epoch was used to track the validation loss over the course of training. This approach stops the training once performance on the validation set no longer improves, thus avoiding overfitting as well as computational waste. Both models went through two training phase, initial training with frozen base layer followed by fine-tuning, both phases were set up to run for up to 40 epochs as shown in Figure 2 and Table I.

For the model evaluation, the researcher employed various performance measures, such as accuracy, precision, recall, and F1-score, to give the overall view of the classification performance. Besides accuracy-oriented measures, the inference time per image was also calculated, along with the total parameters of each model, to measure the efficiency of the model as well as its capability to be applied to real-time applications. Lastly, the models are evaluated to determine which model performs better and if it can be applied to real-time traffic sign recognition applications.

Based on past studies on related models, the researcher expects MobileNet to be faster and more efficient. On the other hand, ResNet's deep design and learning capacity should help it to attain more accuracy. However, it needed more computational power, which resulted in slower speed. As a result, the researcher thinks that MobileNet is more suitable in real-time applications, while ResNet would be preferred if it allows high computational power in contrast to speed.

IV. Result and Discussion

After training and testing with the GTSRB test set, the performance of the models was analyzed based on key metrics such as accuracy, F1-score, and time of inference. Table II provides a thorough comparison of the performance of the two models. This outcome provided an idea about the effectiveness and usability of both MobileNetV3 and ResNet50 models for traffic sign recognition. Comparison of the models shows their strength, weakness, and use in real-world systems.

Figure 3 and Figure 4 display the training progress for both MobileNet and ResNet. Both training progress experience a similar case where the validation accuracy rapidly increases and stabilizes above 90%, while the training accuracy increases gradually and always remains under the validation accuracy. This large gap between train accuracy and validation accuracy could be caused by the data augmentation that is applied to the training set, such as zoom, rotate, flip, and brightness adjustment, making it more challenging to classify in the training set but perform better in the validation set. Fine-tuning with a small learning rate and freezing the base model in the earlier phase may also limit the model's ability to adapt to the training data, contributing to the slow rise of the training accuracy

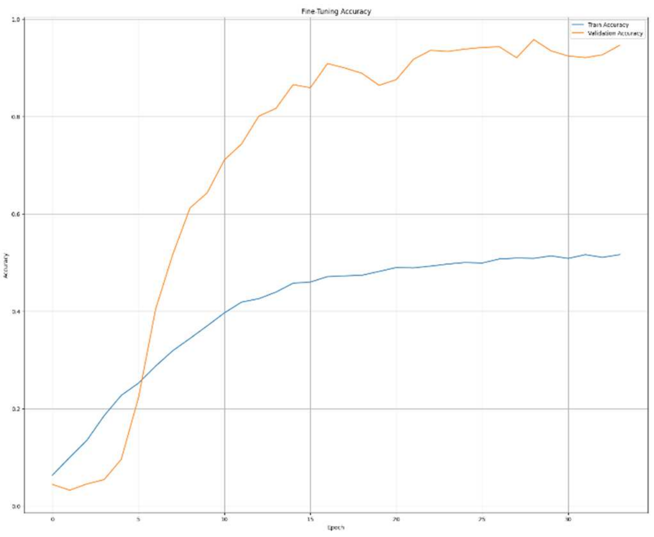

Fig. 3. Training Accuracy and Validation Accuracy during MobileNet Fine-Tuning Phase

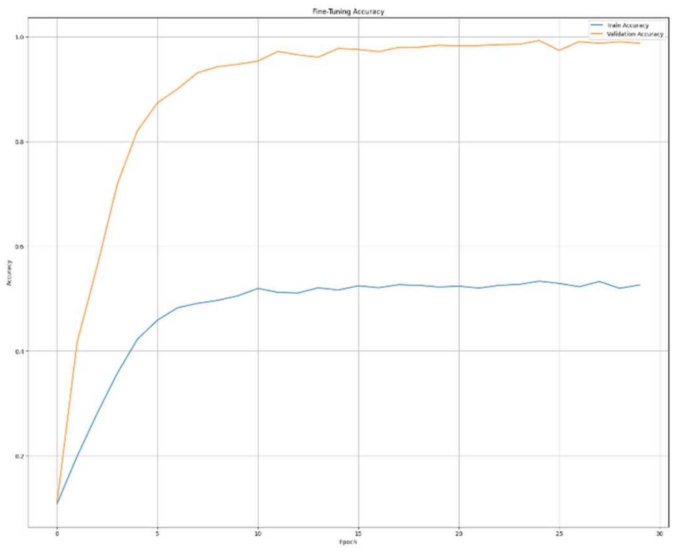

Fig. 4. Training Accuracy and Validation Accuracy during ResNet Fine-Tuning Phase

TABLE II

Comparison of MobileNet and ResNet Performance

Metrics	MobileNet	ResNet
Validation Accuracy	95.78%	99.29%
Test Accuracy	92.19%	94.16%
Weighted F1 Score	92.02%	94.29%
Inference Time (with Kaggle's GPU)	0.00048s	0.00227s
Inference Time (with CPU)	0.14692s	0.08530s

Throughout the training process, MobileNetV3 achieves a training accuracy of 96.34%, a test accuracy of 92.19%, and a weighted F1 Score of 92.02%, indicating a strong generalization across the 43 classes. The model also reached an average inference time of 0.00048 seconds per image on GPU and and 0.14692 seconds on CPU which is highly efficient and a great fit for real-time applications. Based on the classification report, shown in Table III, the researcher found that the model achieves F1-scores above 99% on most popular and visually recognizable classes, including class 14 ('stop sign)' and class 17 ('no entry sign'). On the other hand, the model faces difficulties in recognizing signs with similar shapes and colours (arrows, blue circles, or diagonal lines). For example, class 37 ('go straight or left sign') and class 39 ('keep left sign') achieved low scores with 7.75% and 23.53% respectively. The differences in performance are mostly due to the class imbalance in the GTSRB dataset. For example, Stop Signs, which have over 2,000 training images, get almost perfect F1-scores, while directional signs, which only have about 200 samples, have trouble recognizing them correctly because they don't have enough training data.

TABLE III

Top Three & Bottom Three Performing Classes with MobileNetV3

Class ID	Sign Name	F1-Score
14	Stop Sign	99.63%
17	No Entry Sign	99.30%
31	Wild animals crossing Sign	99.26%
36	Go straight or right Sign	50.63%
37	Go straight or left Sign	7.75%
39	Keep Left	23.53%

On the other hand, ResNet50 demonstrated superior overall performance, achieving 98.87% training accuracy, 94.16% test accuracy, and a weighted F1 score of 94.29%. This improvement was particularly noticeable in some of the sign classes where MobileNetV3 previously performed poorly. For example, the F1-score for Class 39 ('Keep left') saw a dramatic improvement, increasing by over 42 percentage points from 23.53% to 65.92%. However, the issue of class imbalance in the GTSRB dataset still had a significant impact on other classes. The F1-score for Class 37 ('Go straight or left') only improved by a modest 4 percentage points to 11.76%, while performance on Class 36 ('Go straight or right') slightly decreased. This indicates that while ResNet50's deeper architecture can be highly effective, the small amount of training data for certain underrepresented classes remains a challenge. This confirms that a robust model architecture is not entirely immune to the limitations of the training data.

These improvements suggest that the deeper architecture and residual connections in ResNet50 enable it to learn more discriminative features, particularly helpful for complex or visually similar traffic signs. The model showed improved precision and recall in previously low-performing classes such as class 37 and class 39, indicating better generalization. However, this gain in accuracy came with increased computational cost. ResNet50's inference time was 0.00227 seconds per image on GPU and 0.08530 seconds on CPU, shockingly beating MobileNet's inference time on CPU.

Though created as a light model, MobileNetV3 did not surpass ResNet-50 when it comes to CPU-based inference. However, both models proved to excel in different aspects. While ResNet-50 performed more accurately and better handled complex or similarly visible traffic scenes, MobileNetV3 provided significantly improved inference times when run on GPU and remained competitive in general performance. These insights indicate that both models excel in their own ways and the use should be based on the needs of the application, whether maximum accuracy or increased processing speeds for real-time applications.

Regarding how evaluation is done, it causes a number of difficulties that make it difficult to implement in real-world scenarios. The models were only tested in the Kaggle computational environment, so it's hard to say how well they'll operate in the real world on mobile devices, embedded systems, or edge computing platforms that are often utilized in automotive applications. Because of this computational limit, we don't know how long it will actually take to make inferences, how much memory it will use, or how much power it will use in real-world situations. This could make these methods less useful for self-driving cars or driver assistance systems.

These deployment limits are made worse by dataset-specific limits, the GTSRB dataset only has German traffic signs that were taken in controlled conditions, which restrict it to only German traffic management systems. Because the researcher didn't deploy on mobile devices, edge computing, or embedded systems, the models could not be tested in bad weather scenarios because the dataset didn't include any of those situations, such as heavy rain, snow, fog, or nighttime lighting. This limited our evaluation to controlled test settings alone. The dataset also has only a few examples of badly obscured signs that are typical in cities, and there is a big class imbalance, with some classes having more than 2,000 training samples and others having fewer than 200 photos. However, the comparison study gives us useful information about how different types of architecture perform in controlled settings.

V. CONCLUSION

This study compared the performance of the lightweight CNN (MobileNetV3-Large) with the deep CNN (ResNet-50) for traffic sign classification using the GTSRB dataset. Both models were trained with the two-stage training approach and validated in terms of accuracy, F1-score, and the inference time. ResNet-50 recorded slightly greater accuracy (94.16%) and weighted F1-score (94.29%) in comparison with MobileNetV3 (92.19% accuracy and 92.02% F1-score). Surprisingly, MobileNet did not outperform ResNet-50 in inference time completely. Although MobileNet achieved an inference time of 0.14692 seconds on GPU, it performed worse in CPU inference time, also reaching 0.14692 seconds, while ResNet achieved an inference time of 0.08530 seconds on CPU and 0.00227 seconds on GPU. These results point to a nuanced accuracy vs computational efficiency trade-off. While MobileNetV3 is still suitable for real-time use in GPU, it performs poorly on general-purpose CPUs. ResNet-50 excels in terms of classification accuracy and offers constant performance over several kinds of hardware as well.

In future studies, the researcher should look at the constraints that have been found by testing models on real mobile devices and embedded automotive systems to see if they can be used in real-world scenarios, including looking at how much memory and power they need. Furthermore, it could be improved if the evaluation included more than just German traffic signs and controlled environments, such as international standards signs, and challenging environmental conditions like poor weather, dark situations, and signs that are difficult to see. More research into CPU optimization methods, model compression methods, and hybrid architectures that can change based on the specifications of the target hardware, as well as advanced strategies for dealing with class imbalance through synthetic data generation, would help make traffic sign recognition systems more reliable and could be used in a wider range of automotive applications.

Acknowledgement

The Credit author's statement as follows, Yoel Augustan and Clariant Benedictus Tan contributed equally to conceptualization, data curation, investigation, formal analysis and writing, review and editing of the paper. Ivan Sebastian Edbert and Alvina Aulia equally contributed as supervisors, validators and academic advisors, offering direction and support during the research and the paper creation.

References

[1] Y. Sun, P. Ge, and Dequan Liu, *Traffic Sign Detection and Recognition Based on Convolutional Neural Network*. IEEE, 2019.

[2] World Health Organization, "Global status report on road safety 2023."

[3] J. Greenhalgh and M. Mirmehdi, "Traffic sign recognition using MSER and Random Forests," 2012. [Online]. Available: https://www.researchgate.net/publication/261331878

[4] K. Horak, P. Cip, and D. Davidek, "Automatic Traffic Sign Detection and Recognition Using Colour Segmentation and Shape Identification," in *MATEC Web of Conferences*, EDP Sciences, Aug. 2016. doi: 10.1051/matecconf/20166817002.

[5] A. Hechri and A. Mtibaa, "Two-stage traffic sign detection and recognition based on SVM and convolutional neural networks," *IET Image Process*, vol. 14, no. 5, pp. 939–946, Apr. 2020, doi: 10.1049/iet-ipr.2019.0634.

[6] W. Li, D. Li, and S. Zeng, "Traffic Sign Recognition with a small convolutional neural network," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Dec. 2019. doi: 10.1088/1757-899X/688/4/044034.

[7] M. A. Khan, H. Park, and J. Chae, "A Lightweight Convolutional Neural Network (CNN) Architecture for Traffic Sign Recognition in Urban Road Networks," *Electronics (Switzerland)*, vol. 12, no. 8, Apr. 2023, doi: 10.3390/electronics12081802.

[8] W. A. Haque, S. Arefin, A. S. M. Shihavuddin, and M. A. Hasan, "DeepThin: A novel lightweight CNN architecture for traffic sign recognition without GPU requirements," *Expert Syst Appl*, vol. 168, Apr. 2021, doi: 10.1016/j.eswa.2020.114481.

[9] A. Zaibi, A. Ladgham, and A. Sakly, "A Lightweight Model for Traffic Sign Classification Based on Enhanced LeNet-5 Network," *J Sens*, vol. 2021, 2021, doi: 10.1155/2021/8870529.

[10] Y. An, C. Yang, and S. Zhang, "A lightweight network architecture for traffic sign recognition based on enhanced LeNet-5 network," *Front Neurosci*, vol. 18, 2024, doi: 10.3389/fnins.2024.1431033.

[11] A. A. Khalifa, W. M. Alayed, H. M. Elbadawy, and R. A. Sadek, "Real-Time Navigation Roads: Lightweight and Efficient Convolutional Neural Network (LE-CNN) for Arabic Traffic Sign Recognition in Intelligent Transportation Systems (ITS)," *Applied Sciences (Switzerland)*, vol. 14, no. 9, May 2024, doi: 10.3390/app14093903.

[12] Z. Xi, Y. Shao, Y. Zheng, X. Liu, Y. Liu, and Y. Cai, "Road Traffic Sign Recognition Method Using Siamese Network Combining Efficient-CNN-Based Encoder," *IEEE Transactions on Intelligent Transportation Systems*, 2025, doi: 10.1109/TITS.2025.3530940.

[13] N. Triki, M. Karray, and M. Ksantini, "A Real-Time Traffic Sign Recognition Method Using a New Attention-Based Deep Convolutional Neural Network for Smart Vehicles," *Applied Sciences (Switzerland)*, vol. 13, no. 8, Apr. 2023, doi: 10.3390/app13084793.

[14] I. Benfaress, A. Bouhoute, and A. Zinedine, "Advancing Traffic Sign Recognition: Explainable Deep CNN for Enhanced Robustness in Adverse Environments," *Computers*, vol. 14, no. 3, Mar. 2025, doi: 10.3390/computers14030088.

[15] Z. Bi, L. Yu, H. Gao, P. Zhou, and H. Yao, "Improved VGG model-based efficient traffic sign recognition for safe driving in 5G scenarios," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 11, pp. 3069–3080, Nov. 2021, doi: 10.1007/s13042-020-01185-5.

[16] C. Zhang, X. Yue, R. Wang, N. Li, and Y. Ding, "Study on Traffic Sign Recognition by Optimized Lenet-5 Algorithm," *Intern J Pattern Recognit Artif Intell*, vol. 34, no. 1, Jan. 2020, doi: 10.1142/S0218001420550034.

[17] T. Huo, J. Fan, X. Li, H. Chen, B. Gao, and X. Li, "Traffic Sign Recognition Based on ResNet-20 and Deep Mutual Learning," in *Proceedings - 2020 Chinese Automation Congress, CAC 2020*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020, pp. 4770–4774. doi: 10.1109/CAC51589.2020.9327282.

[18] P. D. Prasanna and C. Dushyanth, "Traffic Sign Recognition Using Convolutional Neural Network," 2024. [Online]. Available: www.ijcrt.org

[19] Z. Meng, X. Fan, X. Chen, M. Chen, and Y. Tong, "Detecting Small Signs from Large Images."

[20] Y. Yang, S. Liu, W. Ma, Q. Wang, and Z. Liu, "Efficient Traffic-Sign Recognition with Scale-aware CNN."

[21] Z. Xi, Y. Shao, Y. Zheng, X. Liu, Y. Liu, and Y. Cai, "Road Traffic Sign Recognition Method Using Siamese Network Combining Efficient-CNN-Based Encoder," *IEEE Transactions on Intelligent Transportation Systems*, 2025, doi: 10.1109/TITS.2025.3530940.

[22] S. Song, X. Ye, and S. Manoharan, "E-MobileViT: a lightweight model for traffic sign recognition," *Industrial Artificial Intelligence*, vol. 3, no. 1, p. 3, Mar. 2025, doi: 10.1007/s44244-025-00024-2.

[23] W. Wei *et al.*, "A lightweight network for traffic sign recognition based on multi-scale feature and attention mechanism," *Heliyon*, vol. 10, no. 4, Feb. 2024, doi: 10.1016/j.heliyon.2024.e26182.

[24] G. Zhang, Z. Li, D. Huang, W. Luo, Z. Lu, and Y. Hu, "A Traffic Sign Recognition System Based on Lightweight Network Learning," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 110, no. 4, Dec. 2024, doi: 10.1007/s10846-024-02173-5.

[25] S. Johannes, Marc Schlipsing, Jan Salmen, and Chirstian Igel, *The German Traffic Sign Recognition Benchmark: A multi-class classification competition*. IEEE, 2011.

[26] A. Howard *et al.*, "Searching for MobileNetV3," May 2019, [Online]. Available: http://arxiv.org/abs/1905.02244

[27] N. Gupta, "A Pre-Trained Vs Fine-Tuning Methodology in Transfer Learning," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Aug. 2021. doi: 10.1088/1742-6596/1947/1/012028.

[28] J. C. Antony, G. M. Karpura Dheepan, K. Veena, V. Vikas, and V. Satyamitra, "Traffic sign recognition using CNN and Res-Net," *EAI Endorsed Transactions on Internet of Things*, vol. 10, pp. 1–7, Nov. 2024, doi: 10.4108/eetiot.5098.