SN Computer Science (2026) 7:12
https://doi.org/10.1007/s42979-025-04634-6

ORIGINAL RESEARCH

Traffic Sign Classification Using Real-Time GPU-Embedded Systems

Miguel Lopez-Montiel[1] · Ulises Orozco-Rosas[2] · Moisés Sánchez-Adame[1] · Oscar Montiel[1] · Kenia Picos[2] · Juan J. Tapia[1]

Received: 26 December 2024 / Accepted: 8 December 2025

Abstract
Traffic Sign Classification (TSC) is crucial for autonomous driving and intelligent transportation systems. Desktop implementations of deep learning achieved state-of-the-art performance on TSC benchmarks; however, they are unsuitable for real-time embedded systems due to resource limitations. We propose an Efficient GPU-Embedded Network (EGENet) for embedded platforms, such as NVIDIA's Jetson, to overcome these drawbacks. When implemented on a desktop system with NVIDIA GeForce RTX 2080, EGENet can reduce the number of parameters by 24 million while speeding up by 2.59×. EGENet introduces a new concept called Asymmetric Depth Dilated Separable Convolution (ADDSC), which enables a reduction in parameters and inference time while maintaining the receptive window size. A novel evaluation metric is proposed, considering frames per second (FPS), accuracy, and deployment on embedded GPU devices with constrained resources, targeting at least 98.85% accuracy and a frame rate of more than 30 FPS. Thorough evaluations were performed on the NVIDIA Jetson Xavier AGX and Jetson Nano, utilizing limited resources, to validate EGENet's real-time performance. Evaluation of GTSRB and LISAC datasets demonstrates outperforming results, with an accuracy of 99.58% and 98.18% and a response time of 253 FPS and 90 FPS on Jetson Xavier AGX and Jetson Nano devices, respectively. Our work contributes to efficient TSC systems based on embedded GPUs and offers a comprehensive performance evaluation methodology for autonomous driving. We present exhaustive statistical comparative tests against state-of-the-art systems.

Keywords Autonomous vehicles · Deep learning · Traffic sign classification · Real-time · Embedded systems · Convolution

Introduction

Traffic Sign Recognition (TSR) is one of the most critical tasks for autonomous vehicles and intelligent transportation systems. Traffic Sign Recognition Systems (TSRS) can enhance safety, improve efficiency, aid in autonomous driving, contribute to intelligent transportation systems, enhance energy efficiency, and be cost-effective. TSRS perform the detection, localization, and classification of traffic signs in an image or video, typically utilizing a combination of Traffic Sign Detection (TSD) and Traffic Sign Classification (TSC) to recognize traffic signs in real time [1, 2]. Deep learning algorithms, especially those based on Convolutional Neural Networks (CNNs), are currently considered the state-of-the-art (SOTA) methods for TSC [3]. These approaches are widely used because they excel at performing feature extraction tasks and provide high accuracy for TSR [4]. However, the adequate choice of an algorithm depends on several factors, including the application's specific requirements for accuracy and computational efficiency, hardware availability, and dataset size [5].

Despite these advances, a clear research problem remains: how to achieve accurate and real-time TSC in resource-constrained embedded systems. Autonomous driving applications require models that simultaneously ensure high recognition accuracy and low latency. Yet, most high-performance solutions rely on desktop-class GPUs that

✉ Ulises Orozco-Rosas
ulises.orozco@cetys.mx

[1] Instituto Politécnico Nacional, CITEDI-IPN, Av. Instituto Politécnico Nacional No. 1310, Nueva Tijuana, 22435 Tijuana, Baja California, México

[2] CETYS Universidad, Av. CETYS Universidad No. 4, Fracc. El Lago, 22210 Tijuana, Baja California, México

Published online: 18 December 2025

cannot be deployed in embedded platforms. This gap creates a strong motivation to design efficient neural architectures that balance accuracy and inference speed, while also meeting the power and memory constraints of embedded devices.

In light of these challenges, the specific objectives of this research are threefold: (i) to design a compact and efficient CNN architecture capable of achieving real time TSC on embedded GPUs, (ii) to develop innovative convolutional operations that reduce parameters and latency without sacrificing accuracy, and (iii) to propose a comprehensive evaluation metric that jointly considers accuracy, frames per second (FPS), and implementation feasibility on low power devices.

Our previous research centered on the TSD task [6–8]. In this work, we focus on the TSC task, an essential component of a TSRS. The proposed architecture is inspired by the EESP (Extremely Efficient Spatial Pyramid) [9], incorporating Asymmetric Depth Separable Convolution operations [10] with a dilation rate configuration to improve both accuracy and efficiency in embedded systems. In summary, the main contributions of this paper are:

- Introduction of a novel network architecture named EGENet, which leverages modules from ESPNetv2, DABNet, and ADSCNet. EGENet outperforms most SOTA models [11–13] regarding parameters, inference time, and accuracy. Tested on Jetson Xavier, our architecture achieves remarkable results and retains its position as the fastest model across all GPU devices.
- Development of an innovative convolution block that integrates various efficient convolution operations [14, 15], resulting in a new layer called "ADDSC" (Asymmetric Depth Dilated Separable Convolution). This reduces parameters and inference time while maintaining receptive window size, highlighting the importance of efficient convolution operators in real-time embedded applications.
- Introduction of a novel evaluation metric for real-time TSC that considers FPS, accuracy, and implementation on low-power GPU embedded systems. This metric sets a minimum accuracy of 98.85% (exceeding human performance) and a frame rate above 30 FPS, providing a more comprehensive assessment of embedded TSC performance compared to previous works [15–18].

The paper is organized as follows: "Related Work" section provides an overview of the related work that addresses TSC for real-time applications. "Proposed Network" section outlines the proposed network for GPU embedded systems and explains the methodology applied in this work. "Experimental Setup and Results Analysis" section presents the experimental setup and the results obtained. "Limitations of the Study" section discusses the limitations of the study, highlighting the main constraints and directions for future work. Finally, "Conclusion" section presents the conclusions drawn from the statistical analysis of comparative experiments performed to demonstrate the contributions.

Related Work

In this section, we present a comprehensive review of related research on TSC, specifically emphasizing real-time approaches that utilize DL (Deep Learning) techniques. TSC has been widely studied in the literature, leading to the proposal of various methods for this task [17]. In recent years, DL methods based on CNNs have emerged as the SOTA for TSC [16]. These systems have demonstrated remarkable effectiveness in handling challenging conditions, such as variations in lighting and sign degradation, and have achieved top performance on the benchmark dataset known as the German Traffic Sign Recognition Benchmark (GTSRB) [19]. Nevertheless, significant challenges remain in the context of real-time embedded systems for TSC [20].

Real-time embedded systems for TSC face diverse challenges, such as limited computational resources and reduced memory capacity in mobile devices [21]. These challenges often need a trade-off between processing time and accuracy, involving a reduction in computational complexity and a decrease in TSC accuracy to mitigate high latency and memory requirements [22]. Several approaches have been proposed to address these challenges and improve the efficiency of TSC models, including compact models, tensor decomposition, quantization, and sparsification [23]. Nevertheless, while these techniques effectively enhance the efficiency of architectures, it is imperative to prioritize the development of an efficient architecture first before considering these optimization methods as a final step.

Recent research on TSC systems has focused on developing real-time embedded systems for TSC [11, 12]. These systems are designed to run on resource-constrained embedded platforms and can classify TSC in real-time. One of the first works focusing on real-time embedded systems for TSC is [22]. They introduced MicronNet, a deep convolutional neural network optimized for real-time TSR in embedded systems. They optimized the microarchitecture of each layer to reduce the number of parameters and improve the macroarchitecture and parameter precision. As a result, MicronNet has a model size of only 1MB and 510,000 parameters, which is 27 times fewer parameters than SOTA models. It requires only 10 million multiply-accumulate operations to perform inference and has a computation time of 32.19 ms on a Cortex-A53 high-efficiency processor. Despite its

compact size and efficient architecture, MicronNet achieves a top-1 accuracy of 98.9% on the GTSRB dataset. The experimental results demonstrate the need to create highly compact and optimized deep neural networks to enable the real-time implementation of TSR on embedded systems.

In [24], a novel network for TSC named ENet has been developed. ENet improves accuracy and generalization by incorporating data mining and multiscale operations while enhancing processing time through depth separable convolution (DSC). ENet has only 0.9 million parameters, which is significantly fewer than those of SOTA methods. Nonetheless, it still attains a remarkable accuracy of 98.6% on the GTSRB. This work confirms that the proposed method is a suitable solution for real-time embedded systems with limited computational resources and memory capacity. It also proves that it is possible to create an efficient neural network architecture for real-time embedded TSR that offers the right balance of accuracy, generalization, and processing time.

In [25], a real-time embedded system for recognizing road signs using the Raspberry Pi 3 platform was introduced. The system incorporates the MSERs (Maximally Stable Extremal Region) technique and color and shape information in the HSV (Hue, Saturation, Value) color space for the detection stage. In contrast, the recognition stage employs ORB (Oriented Fast and Rotated Brief) descriptors to match traffic signs with a database model. The system is shown to work well in various situations, including challenging weather conditions, rotation, similar object color, multiple signs, and partial occlusion. The system is highly accurate, achieving a 95.39% for recognition in the GTSRB with 13 FPS. A comparison with SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features) descriptors demonstrates the efficiency of the developed system, which is used to support TSR for real-time driving conditions.

In [26], a TSR system was developed using Python. The system incorporates pre-processing techniques and object detection methods in various color spaces, along with classification algorithms to optimize the trade-off between accuracy and processing time. The pre-processing step employs the Shadow and Highlight Invariant Method for color segmentation, while the recognition phase utilizes a CNN. This combination achieved the highest classification accuracy of 92.97% with a processing time of 7.81 ms, as demonstrated on their new dataset, the Napier University traffic dataset. The system is designed to run on the NVIDIA Jetson Nano and can adapt its frame rate, ultimately achieving high detection and classification accuracy.

In [27], the authors proposed a Deep Neural Network named MDEffNet, which is designed for TSR. The network incorporates batch normalization layers to reduce the internal covariant shift and utilizes data augmentation to achieve high efficiency across different datasets from various countries. The network achieved high test accuracies of 99.16% for GTSRB, 98.69% for Belgium Traffic Sign Classification Dataset (BTSC) [28], and 99.22% for Mapping and Assessing the State of Traffic InFrastructure (rMASTIF) [29]. MDEffNet has only 1.36 million parameters, much lower than SOTA models, and has a relatively low training time of 130 s per epoch. As a result, it is suitable for real-time embedded TSRS.

In [12], the authors introduced MicronNet-BF, which incorporates MicronNet [22], batch normalization, and factorization techniques. The addition of batch normalization improved recognition accuracy to 98.74%, surpassing MicronNet's performance by 1.05%. Further accuracy improvements were achieved by applying factorization, resulting in a recognition accuracy of 98.77%. When all three techniques were combined, MicronNet-BF achieved recognition accuracies of 99.383% in GTSRB and 82.122% in BTSC datasets. Remarkably, MicronNet-BF uses only 0.44 million parameters, making it an excellent option for implementation in embedded systems.

In [11], the authors introduced a novel approach for TSR called Real-Time Image Enhanced CNN (RIECNN). This approach can handle multiple diverse traffic sign datasets and outperforms existing SOTA methods in terms of recognition rate and execution time. The RIECNN approach achieved the highest rank in the GTSRB, BTSC, and rMASTIF datasets, surpassing all previous SOTA techniques with recognition accuracies of 99.75% in GTSRB, 99.25% in BTSC, and 99.55% in rMASTIF. Furthermore, the RIECNN approach meets the real-time constraint, as the pre-processing time and inference time combined do not exceed 1.3 ms per image.

In the related work, it has been observed that achieving a balance between accuracy and processing time remains a challenge for TSC in real-time applications, particularly on embedded devices. Recent works have attempted to address this challenge by finding a trade-off between accuracy and processing time, focusing solely on the number of parameters. However, the number of parameters is only occasionally directly related to real-time performance in systems with limited resources. Therefore, our proposal prioritizes the design and development of an efficient convolutional block and model, with an emphasis on minimizing processing time. This optimization is crucial when implementing our solution on embedded systems with limited resources. We aim to ensure efficient and effective performance on these devices by minimizing the impact on these resource-constrained systems.

Unlike previous works that primarily emphasize reducing parameter count or applying optimization techniques

such as quantization, decomposition, or sparsification, our approach introduces a new convolutional block (ADDSC) and an efficient CNN architecture (EGENet) conceived explicitly for real-time deployment on embedded GPUs. While models such as MicronNet, ENet, and MDEffNet achieved compactness and accuracy, they often prioritized parameter reduction as the primary efficiency criterion. In contrast, EGENet is designed not only to minimize parameters but also to explicitly address inference speed, receptive field preservation, and integration within resource-constrained GPU-embedded devices. Furthermore, we propose a novel evaluation metric that jointly considers accuracy, FPS, and deployment feasibility on embedded platforms (Jetson Nano and Xavier AGX), offering a more comprehensive and practical assessment of real-time TSC performance. This combination of architectural innovation, efficiency, and a new evaluation criterion distinguishes our work from existing TSC systems.

Proposed Network

This section introduces the EGENet (Efficient GPU-Embedded Network) as a CNN architecture designed specifically for real-time TSC on GPU-Embedded Systems. Our architecture stands out for incorporating a highly efficient convolution layer composed of advanced convolutional operations. This layer, called "ADDSC" (Asymmetric Depth Dilated Separable Convolution), has been designed to be lightweight and efficient, making it a suitable choice for implementation on integrated platforms with GPU support. Our approach builds on the foundation of efficient convolutional operators [15]. These operators were explicitly designed to decrease the number of parameters and computations needed by the convolution layers in CNNs.

Asymmetric Depth Dilated Separable Convolution

In practice, the computational cost of convolution is often dominated by the product of feature map size ($H \times W$), number of input channels (C_{in}), and number of output channels (C_{out}). While big O notation only provides an upper bound on Time Complexity (TC), both $\mathcal{O}(C^2K^2HW)$ and $\mathcal{O}(C^2K^2D^2)$ are considered to have quadratic complexity ($D^2 = H \times W$). Therefore, the choice of expression depends on the specific problem and notation convention, but both expressions represent quadratic TC. To maintain consistency and avoid confusion with different mathematical notations, we express the computational cost of the standard convolution as,

$$\mathcal{O}(K_W \times K_H \times H \times W \times C_{in} \times C_{out}) \quad (1)$$

this expression illustrates that the computational cost is directly proportional to the kernel size (width K_W and height K_H), the feature map size (width W and height H), and the number of input channels (C_{in}) and output channels (C_{out}).

The computational challenge of CNNs in mobile and embedded devices is addressed by techniques like MobileNetV1's Depthwise Separable Convolution (*DSC*) [30]. This method breaks down the standard convolution into depthwise and pointwise convolutions, significantly reducing computational cost. Following this, MobileNetV2 further optimizes the process with a bottleneck block called Depthwise Separable Bottleneck Convolution (*DSBC*) that combines different convolution types, thereby lowering the computational cost while maintaining accuracy [31]. ADSCNet advances this reduction by factorizing convolutions with the Asymmetric Depthwise Separable Bottleneck Convolution (*ADSBC*), offering even more significant computational savings [10]. Collectively, these approaches enhance the efficiency of CNNs for mobile and embedded device applications.

Recent studies in DL have aimed at reducing TC while balancing accuracy and computational cost. For instance, MobileNetv3 focuses on enhancing model accuracy without significantly increasing the computational load for mobile and embedded systems [31]. In contrast, the Dilated Asymmetric Bottleneck (DABNet) block introduces dilation rates in convolution kernels to expand the receptive field and improve accuracy, albeit at a higher computational cost [32]. The increased number of FLOPs in DABNet is a trade-off for better accuracy, highlighted by the computational formula for dilated convolution. This formula considers the dilation rate, kernel size, and input and output channel interactions, underscoring the complexity and computational demands of achieving high accuracy in DL models.

This study introduces a novel operation called Asymmetric Depth Dilated Separable Convolution (ADDSC). This method integrates various approaches to strike a balance between computational efficiency and model accuracy.

Figure 1 illustrates the steps involved in executing Asymmetric Depth Separable Convolution (ADSC). Our proposed ADDSC method is an improvement over ADSC that incorporates a dilated convolution technique to enhance its performance capabilities further. Our method aims to balance computational efficiency and model accuracy by integrating various approaches.

The computational cost of our proposed Asymmetric Depth Dilated Separable Convolution (ADDSC) with a dilation rate of R is calculated as follows,

$$O\big((C_{in}^2 + 2K_R \times C_{in} + C_{in} \times C_{out}) \times H \times W\big) \quad (2)$$

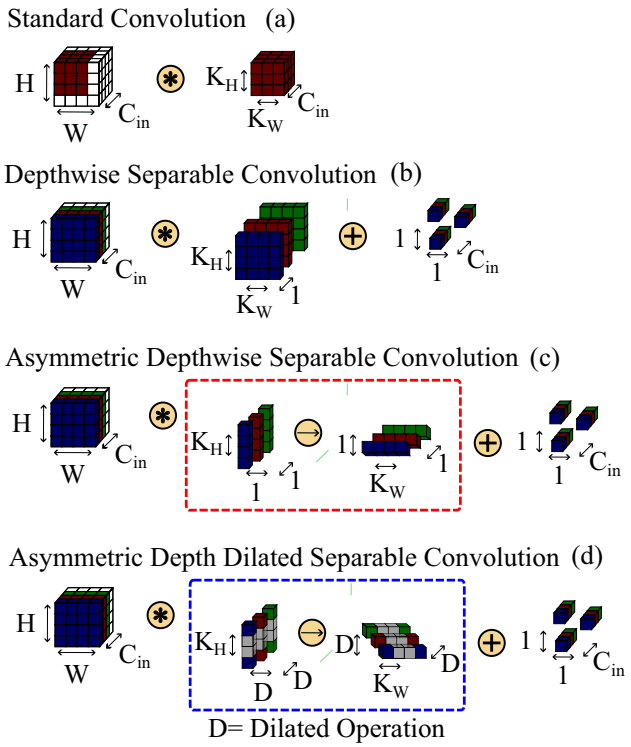


Fig. 1 Convolution operations: **a** Standard Convolution, **b** Depthwise Separable Convolution, **c** Asymmetric Depthwise Separable Convolution (ADSC), and **d** Asymmetric Depth Dilated Separable Convolution (ADDSC). In ADSC, $1 \times k$ and $k \times 1$ depthwise convolutions are applied sequentially, while ADDSC extends this by introducing dilated depthwise convolutions (gray gaps), enlarging the receptive field without increasing the number of parameters

C_{in} and C_{out} represent the number of input and output channels, respectively, and H and W represent the height and width of the input feature map. The Eq. (2) has three terms.

The first term (C_{in}^2) accounts for the computational cost resulting from the interactions between input channels (C_{in}) in the dilated convolution. It reflects the additional computations required when processing multiple input channels simultaneously.

The second term ($2K_R \times C_{in}$) represents the computational cost associated with the kernel size (K_R) and the number of input channels (C_{in}) in the convolutional operation. It arises from the fact that when the dilation rate (R) is set to 1, the dilated convolution simplifies into a regular convolution.

The third term ($C_{in} \times C_{out}$) captures the computational cost linked to the number of input channels (C_{in}) and the number of output channels (C_{out}) in the convolutional operation.

Table 1 provides a comparative analysis of convolution techniques in CNNs, highlighting their computational trade-offs. Standard Convolution (SC) is established in SOTA models but is often replaced by Depthwise Separable Convolution (DSC) in efficiency-oriented architectures. Asymmetric Convolution (AC) builds on this by employing non-square filters, optimizing parameter usage.

The Asymmetric Dilated Depthwise Separable Convolution (ADDSC) advances beyond the Asymmetric Depthwise Separable Bottleneck Convolution (ADSBC) through asymmetric kernels and dilation. This approach allows for enhanced feature extraction with reduced computational overhead. ADDSC's use of dilated asymmetric filters efficiently expands the receptive field, capturing extensive contextual information without increasing model complexity.

Consequently, ADDSC is preferable in scenarios that demand high computational efficiency and intricate feature extraction, thereby asserting its relevance in the design of lightweight, high-performance CNNs. The ADDSC provides several advantages over the ADSBC:

1. The ADDSC offers an increased receptive field compared to the ADSBC. While the ADSBC has a receptive field of $K_H \times K_W$ (where K_H and K_W are the kernel sizes used in the height and width dimensions), the asymmetric dilated version expands the receptive field further. By incorporating dilation, the receptive field expands to capture larger contextual information, allowing for a better understanding of spatial relationships between features.
2. This larger receptive field provided by the ADDSC enables the model to incorporate more global context while processing local features. This enhanced contextual understanding can improve performance in tasks such as object recognition, segmentation, and scene

Table 1 Analysis of convolution time complexity, [9, 33]

Convolution type	Effective receptive field	TC
SC	$H \times W$	$\mathcal{O}(C^2 \times K^2 \times D^2)$
DSC	$H \times W$	$\mathcal{O}(K^2 \times C_{in} \times D^2)$
DSBC	$H \times W$	$\mathcal{O}((C_{in}^2 + K^2 \times C_{in} + C_{in} \times C_{out}) \times D^2)$
ADSBC	$H \times W$	$\mathcal{O}((C_{in}^2 + 2 \times K \times C_{in} + C_{in} \times C_{out}) \times D^2)$
ADDSC*	$(H-1) \times (W-1) \times R + 1$	$\mathcal{O}((C_{in}^2 + 2 \times K_R \times C_{in} + C_{in} \times C_{out}) \times D^2)$

$H = W$. The $*$ indicates our proposal

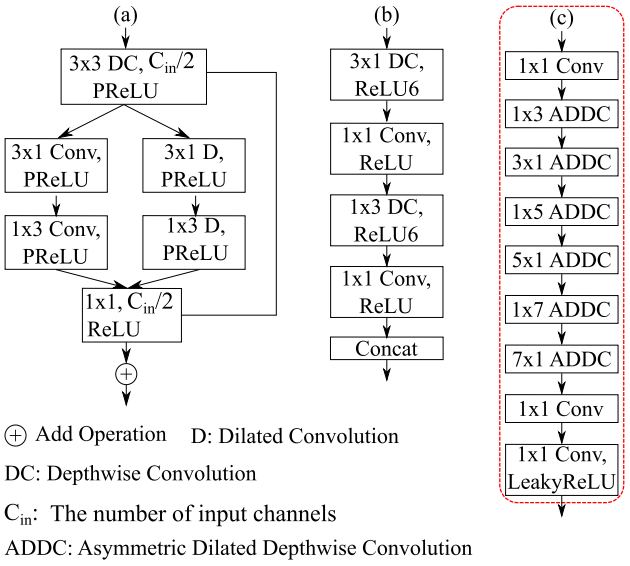


Fig. 2 Network block design. **a** DABNet, **b** ADSCNet, and **c** Our proposed EGENet

understanding, where capturing long-range dependencies is crucial.

3. The ADDSC achieves an increased receptive field without significantly increasing the number of parameters compared to the ADSBC. The model can capture more contextual information without additional convolutional filters by utilizing dilated convolutions, resulting in better parameter efficiency. Unlike some other types of convolutions, such as SC or DSC, the ADDSC preserves the spatial resolution of the input feature maps. This preservation of spatial resolution is essential for tasks that require accurate localization, such as object detection or instance segmentation.

Our ADDSC offers an expanded receptive field and improved contextual understanding, parameter efficiency, and spatial resolution preservation compared to the ADSBC. These advantages make it a powerful tool for various computer vision tasks, enabling models to capture more extensive spatial dependencies and achieve higher performance in tasks that require global context awareness.

Efficient GPU-Embedded Network

The EGENet model proposed (see Fig. 3) primarily consists of EBlocks (Efficient Blocks) illustrated in Fig. 2c. A detailed description of this block is found in Table 2. The EBlock unit is applied to all feature maps larger than 7×7, followed by a Depthwise Convolution operation with a kernel size of 3×3, Pointwise Convolution with a kernel size of 1×1, Batch Normalization, and a Leaky ReLU activation function. This consistent structure is utilized across all versions of EGENet. It is important to note that the ADDSC operation within the EBlock unit provides a computationally efficient solution. Additionally, the EBlocks unit is not designed as a residual block, aiming to minimize the TC in more extensive model versions (Fig. 3).

Table 2 presents a comprehensive overview of the EGENet body architecture in its small version, specifically designed for efficient TSC in real-time embedded systems. The table provides detailed insights into the architecture's key components and characteristics. The "Operation Number" column indicates the sequence of operations, facilitating a clear understanding of the architecture's flow. The "Layer Name" column provides identifiers for each layer, facilitating component identification and analysis. The "Input Size" and "Output Size" columns specify the dimensions of the input and output feature maps, respectively, providing information about the data flow through the architecture. The "Kernel Size" column indicates the convolutional kernel size utilized in each operation, influencing the receptive field and the range of information considered during convolution. The "Dilation Rate" column reveals the spacing between kernel elements, impacting the receptive

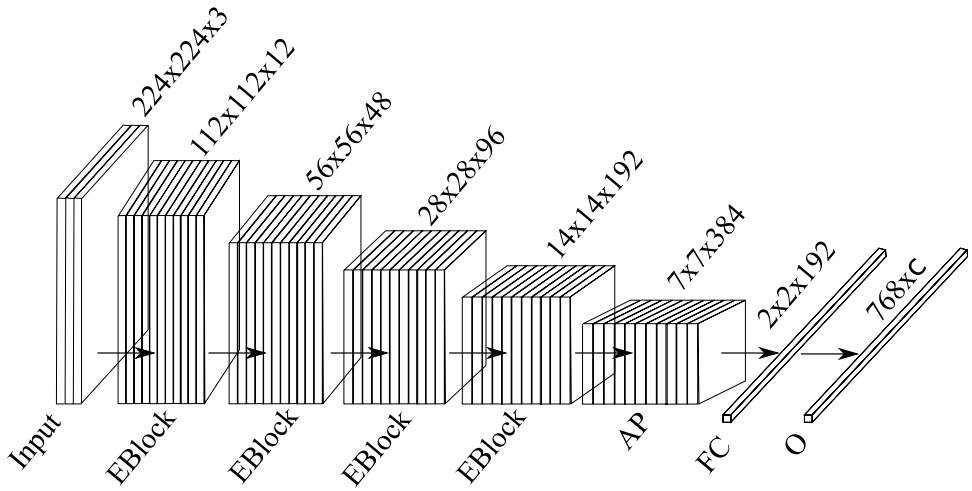


Fig. 3 Proposed EGENet architecture: medium version

Table 2 EGENet body architecture small version

Operation number	Layer name	Input size	Output size	Kernel size	Dilation rate	Groups	Stride
1	Conv	3×28^2	12×14^2	1×1	1×1	1	2
2	Conv	12×14^2	12×14^2	1×3	1×0	12	1
3	Conv	12×14^2	12×14^2	3×1	0×1	12	1
4	Conv	12×14^2	12×14^2	1×5	0×2	12	1
5	Conv	12×14^2	12×14^2	5×1	2×0	12	1
6	Conv	12×14^2	12×14^2	1×7	0×3	12	1
7	Conv	12×14^2	12×14^2	7×1	3×0	12	1
8	Conv	12×14^2	12×14^2	1×1	1×1	1	1
9	BN	12×14^2	12×14^2	–	–	–	–
10	LReLU	12×14^2	12×14^2	–	–	–	–
11	Conv	12×14^2	48×7^2	3×3	1×1	1	2
12	Conv	48×14^2	24×7^2	1×1	1×1	1	1
13	BN	24×14^2	24×7^2	–	–	–	–
14	LReLU	24×14^2	–	–	–	–	–
15	AvgPool	24×14^2	24×2^2	5×5	–	1	2
16	Linear	96	c	–	–	–	–

The use of – indicates that this operation does not include that parameter

Table 3 Performance evaluation of EGENet configurations on GeForce RTX 2080 with an output of 43 Classes

EGENet configuration	Input size	Layers	Params (M)	Forward time (ms)	Backward time (ms)	L (ms)
Tiny	3×28^2	16	0.0084	0.81	2.64	0.72
Small	3×112^2	36	0.1211	1.12	2.77	0.95
Medium	3×224^2	46	0.4649	1.32	3.23	1.12
Large	3×448^2	56	1.816	1.57	5.86	1.34

field size and the level of data sampling. The "Groups" column signifies the number of groups used in each operation, facilitating efficient computation by dividing channels into separate groups. Notably, operations 2 to 9 comprise the ADDSC operation, which remains consistent across different versions of EGENet. The body architecture remains the same for all versions, with differences in input size (ranging from 28×28 to 448×448) and the number of ADDSC operations repeated, which varies based on the input size (ranging from 1 to 5). This table provides valuable insights into the EGENet body architecture, allowing researchers and practitioners to understand its components and variations across different input sizes.

Table 3 provides a comprehensive analysis of the performance of various EGENet configurations. It presents important metrics for evaluating these configurations. The "EGENet Configuration" column lists the different configurations considered, which can vary in architecture, size, and other parameters. The "Input Size" column specifies the dimensions of the input data, giving insights into the resolution or scale of the input images or feature maps used. The "Layers" column indicates the number of layers, providing an understanding of the network's complexity and depth for each configuration. The "Params" column reveals the number of parameters, offering insights into model complexity. Higher parameter counts generally indicate more expressive models at the expense of increased computational resources. The "Forward Time" column measures the time for forward propagation, reflecting the inference speed of each EGENet configuration. The "Backward Time" column indicates the time for backward propagation, providing information on computational demands during training. The "Latency" column represents the total time delay, or latency, for processing each input, combining forward and backward times. This metric is useful for assessing real-time performance. Researchers and practitioners can gain a deeper understanding of the trade-offs between model complexity, inference speed, and computational requirements by evaluating and comparing these metrics. This knowledge empowers them to make informed decisions when selecting an EGENet configuration for specific applications. EGENet code is available at https://github.com/mmontielpz/egenet.

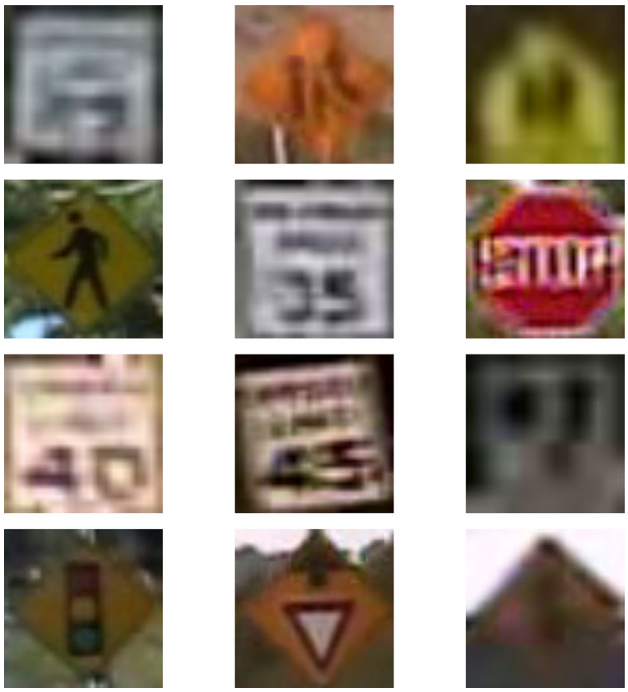


Fig. 4 Example batch classes from LISAC

Table 4 Traffic sign classes in LISAC dataset

Class id	Class name	Class samples
0	addedLane	76
1	keepRight	174
2	Merge	128
3	pedestrianCrossing	230
4	School	98
5	schoolSpeedLimit25	51
6	signalAhead	338
7	speedLimit25	80
8	speedLimit35	208
9	speedLimit40	73
10	speedLimit45	68
11	speedLimitUrdbl	113
12	Stop	929
13	stopAhead	76
14	yieldAhead	53

Traffic Sign Classification Datasets

Although the German Traffic Sign Recognition Benchmark (GTSRB) dataset is a reliable benchmark to assess TSR performance in real-world scenarios [34], it lacks sufficient challenging conditions to test the robustness of TSRS. Therefore, we chose to demonstrate the robustness of our proposed EGENet model using a different dataset. We selected the Laboratory for Intelligent & Safe Automobiles (LISA) dataset [35], which comprises 6,610 images with 47 unique traffic signs and several distinct characteristics. Unlike GTSRB, the LISA dataset presents challenging conditions such as illumination, occlusion, shadow, blur, reflection, codec error, dirty lens, decolorization, and overcast, making it more suitable to evaluate the robustness of TSRS. However, this dataset was initially designed for the detection task, and we extracted the traffic signs along with their coordinates, as provided by the dataset. As a result, the extracted traffic signs have a different resolution range than GTSRB, with smaller sizes ranging from 8×8 to 103×94 pixels.

Figure 4 shows sample images from different classes in the resulting LISAC dataset (see Table 4), which we derived as a variant of the original LISA dataset. In our case, we performed a cropping process on the annotated traffic signs to adapt the dataset from its original detection-oriented design to a classification-oriented version. This adjustment constitutes our main contribution, as it enables focused evaluation of TSC models. The dataset still exhibits challenging conditions such as blur, poor resolution, and small object size. While small-sized object detection is a complex research topic [36, 37], requiring precise localization under occlusion, for TSC the problem is less complex, since only feature extraction and recognition are involved. Nonetheless, classification under varying sizes and degraded conditions remains non-trivial, which motivates the use of LISAC as a benchmark for robust TSC and, ultimately, for the development of more accurate TSD and TSRS.

To aid in the reproducibility of our research and support fellow researchers, we have made the datasets available on Kaggle.[1] While both datasets have official pages, our LISAC dataset has been modified for classification purposes. Regarding the GTSRB dataset, we have organized it in a more intuitive, accessible, clean, and straightforward manner to facilitate the development of experimental bases.

Embedded Devices

As we mentioned, a significant challenge in achieving real-time TSRS is enabling the efficient deployment of CNNs on embedded devices [22]. Currently, these models are designed to run on high-end computational platforms, and the implementation of TSRS on embedded systems has not been extensively researched [18].

Hardware acceleration for DL inference on edge devices is advantageous over cloud-based computing, as it reduces latency and allows for real-time use cases. The general-purpose CPU (Central Processing Unit) can run DL training and inference. However, it is not feasible for on-edge inference due to high power consumption, low throughput performance, and large silicon area. The GPU offers high-performance computing using parallel processing, but its

[1] https://www.kaggle.com/datasets/mmontiel/lisac-traffic-sign-classification-dataset

power consumption is not tolerable for most edge devices. Additionally, the limitation of GPUs is the I/O latency for transferring data from/to memory. Therefore, the usage of GPU for edge inference depends on the power and energy handling capacity of the target [18].

Embedded platforms offer slightly lower performance and accuracy than desktops. However, they are highly power-efficient and have built-in Tensor Cores that provide four times higher performance than CUDA cores. The hardware specifications of Jetson platforms vary depending on the model. The Jetson Nano is a small, powerful AI computer that can run DL neural networks in parallel. In contrast, the Jetson AGX Xavier is the highest-performing embedded system on the market designed for autonomous machines [38]. Table 5 compares the specifications of Jetson products, showing the better general hardware features based on our proposed work.

Performance Metrics

In this section, we describe the metrics utilized to evaluate the effectiveness of real-time embedded systems for TSC. To assess the classification performance, we used a range of metrics such as the confusion matrix, precision, recall, F1-score, and accuracy. Furthermore, we evaluated the implementation on embedded GPU devices with constrained resources (such as NVIDIA Jetson Nano and Xavier AGX) by considering factors such as inference time and frames per second (FPS).

Accuracy (*ACC*) is a commonly used metric in classification to measure the overall performance of a model. It indicates how many predictions are correct relative to the total number of predictions [39].

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

where *TP* represents true positives, *TN* true negatives, *FP* false positives and *FN* false negatives.

Table 5 NVIDIA GPU embedded systems specifications

Hardware accelerator	GPU	Memory	AI performance
Jetson AGX Xavier	512-core Volta™ 64 tensor cores	32 GB 256-bit LPDDR4x 136.5 GB/s	32 TOPs
Jetson Nano	128-core Maxwell™	4 GB 64-bit LPDDR4 25.6 GB/s	472 GFLOPs

Precision (*PREC*) measures the proportion of correctly classified positive instances out of the total number of instances classified as positive (*FP* + *TP*).

$$PREC = \frac{TP}{TP + FP} \tag{4}$$

Sensitivity (*REC*), also known as recall or true positive rate, measures the proportion of correctly classified positive instances out of the total number of actual positive instances.

$$REC = \frac{TP}{TP + FN} \tag{5}$$

F1-score (*F*1) merges precision and recall into a singular metric, offering a balanced assessment of a model's accuracy in identifying positive samples. This score encapsulates the trade-off between precision and recall, providing a comprehensive measure of performance [40].

$$F1 = 2\left(\frac{PREC \times REC}{PREC + REC}\right) \tag{6}$$

An alternative method to evaluate model performance is through the confusion matrix and the *AUROC* curve (Area Under the Receiver Operating Characteristic Curve). The confusion matrix clarifies class-specific classification accuracy, while the *AUROC* curve quantifies overall performance across different threshold settings [39].

$$TPR = REC = \frac{TP}{TP + FN} \tag{7}$$

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

$$ROC(\sigma) = (FPR(\sigma), TPR(\sigma)) \tag{9}$$

$$AUC(\sigma) = \int_a^b TPR(\sigma)\, d(FPR(\sigma)) \tag{10}$$

$$MAUROC = \frac{1}{c}\sum_{i=1}^{c} AUC_i \tag{11}$$

The Receiver Operating Characteristic (*ROC*) curve (9), constructed using the True Positive Rate (*TPR*) and False Positive Rate (*FPR*) see (7) and (8), visualizes a model's classification ability at various thresholds. The Area Under the Curve (*AUC*) is determined by integrating the *TPR* against the *FPR* in the range of [0, 1], as shown in (10). For multiclass classification, the Mean Area Under the Receiver

Operating Characteristic Curve (*MAUROC*) averages the *AUC* values for each class, offering a unified evaluation of the model's discriminative power (see (11)). These metrics, particularly useful in imbalanced datasets and multiclass scenarios, effectively gauge the model's capacity to distinguish between classes at different thresholds [41, 42].

The inference time, also known as latency (L), is a crucial factor for achieving a real-time response from a model [16, 43]. It measures the time required for a model to infer from one sample and is typically reported in milliseconds. A lower value of L indicates a faster response,

$$L = s - e \tag{12}$$

s denotes the start time of the model's inference, and e denotes the time when the model finished processing the sample. It is important to minimize the inference time to achieve real-time performance.

Frames Per Second (*FPS*) is essential in real-time computer vision applications [44, 45]. It represents the number of images or frames presented per second within a video or image sequence,

$$FPS = \frac{1}{L} \tag{13}$$

where L represents the time required to display/process a specific frame in the visual sequence (in this work, it refers to the model's inference time or latency). A higher number of *FPS* indicates smoother playback, which is crucial in applications that rely on real-time interaction [46, 47].

Experimental Setup and Results Analysis

This section delineates the experimental framework and results analysis for evaluating the EGENet model TSC on real-time GPU-embedded systems. The evaluation utilized two established datasets: GTSRB and the LISAC dataset (refer to "Traffic Sign Classification Datasets" section).

The research comprised two experimental phases. The initial phase aimed at surpassing human-level performance with a target accuracy rate above 98.85% in classification evaluation. The subsequent phase centered on deploying the EGENet model in real-time GPU-embedded systems, with an emphasis on efficient operation in resource-constrained environments. A key objective was to achieve a processing rate of 30 FPS, catering to time-critical applications such as autonomous driving.

The training utilized both GTSRB and LISAC datasets. For GTSRB, 90% of the data (46,655 samples) was allocated for training and 10% (5,184 samples) for testing, with a further split of the training set into 70% (32,641 samples) for training and 30% (14,014 samples) for validation. The LISAC dataset followed a similar distribution, with 90% (2,412 samples) for training and 10% (274 samples) for testing, and a 70%–30% split (2164 samples for training and 248 samples for validation) within the training set.

The hardware configuration for training and evaluation is comprised of an Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz, 32GiB system memory, and a GeForce RTX 2080 GPU. Two GPU-equipped embedded systems boards, the NVIDIA Jetson Nano and the NVIDIA Jetson AGX Xavier, were employed for evaluating the real-time embedded implementation (see "Embedded Devices" section). The experiments utilized PyTorch [48], with Docker containers [49] ensuring reproducibility across all hardware components, including the workstation and NVIDIA Jetsons. The workstation operated on `Python 3.8` and `Torch 1.11.0`.

Figure 5 provides an overview of the complete end-to-end workflow, from dataset preparation and training to deployment on embedded GPU devices. The training phase illustrates how the LISAC and GTSRB datasets are processed on a desktop workstation equipped with an RTX GPU to train the proposed EGENet + ADDSC model. The trained model is then transferred to the deployment phase, where it is executed on NVIDIA Jetson platforms (Nano and AGX Xavier) using PyTorch and Docker for reproducibility. Finally, the inference stage performs traffic sign classification in real time, demonstrating the integration of both hardware environments (desktop GPUs and embedded GPUs) within the proposed system.

Objective classification evaluation employed metrics including *MAUROC*, $F1$, *PREC*, and *REC*, applied to the test and validation sets across three categories: all classes, major (top five frequent classes), and minor (five least frequent classes). *ACC* was included to assess correct class identification and recognition. For real-time embedded systems, FPS was the key metric, vital for real-time operability. The EGENet model training involved hyperparameters: batch size of 32, 100 epochs with early-stopping at 25 epochs, and Adam optimizer [50] with an initial learning rate of 0.0001. The Class-Balanced Loss Based on Effective Number of Samples function [51] was used for optimization.

Data Analysis

Although good generalization requires a minimum of 1000 samples per class [52], both GTSRB and LISAC datasets do not meet this requirement, and a minimum of 50 Class samples is selected for these datasets in this work.

Both Figs. 6 and 7 demonstrate imbalanced class issues. Figure 6 represents the LISAC dataset and shows two main

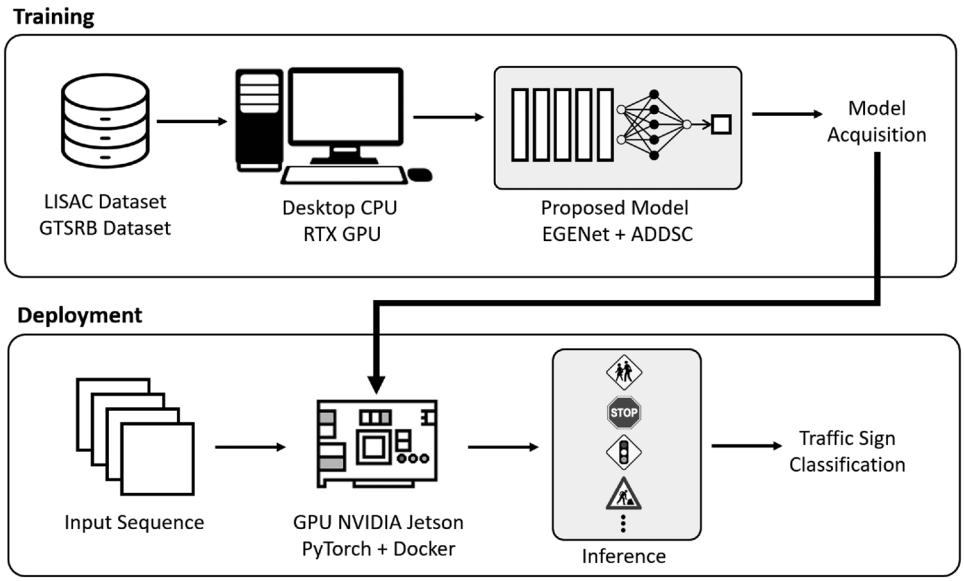


Fig. 5 End-to-end workflow of the proposed system. The pipeline starts with dataset preparation (GTSRB and LISAC), followed by training of the proposed EGENet model on a desktop GPU (RTX 2080). The trained model is then deployed on embedded GPU platforms (Jetson Nano and AGX Xavier) using PyTorch and Docker, enabling real-time inference for traffic sign classification

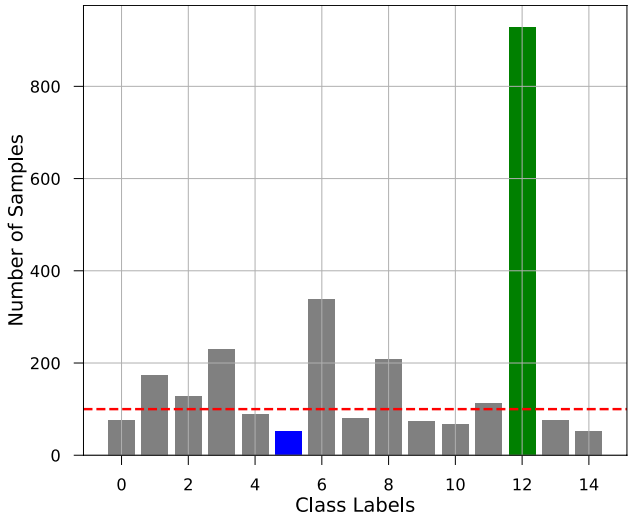


Fig. 6 Class distribution in the LISAC dataset (Table 4): This bar chart illustrates the distribution of classes in the LISAC dataset, where each bar corresponds to a specific class label. The tallest green bar, labeled as Class 12, represents the class with the highest number of samples, while the shortest blue bar, labeled as Class 5, indicates the class with the lowest number of samples. Notably, a dashed red line is included to highlight the required minimal number of samples, set at 100 samples

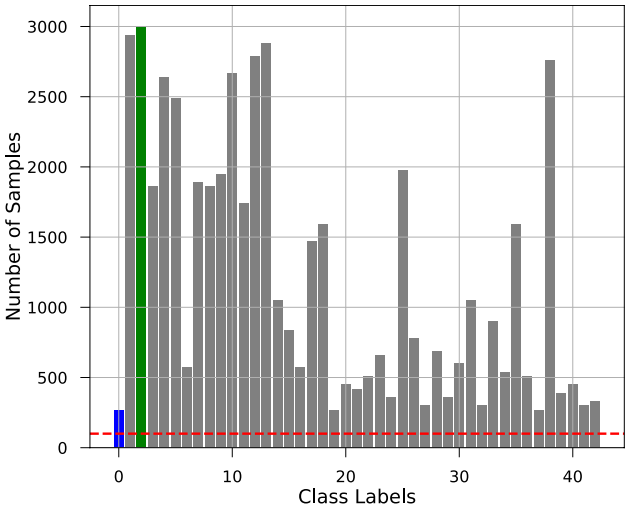


Fig. 7 Class distribution in the GTSRB dataset: This bar chart presents the distribution of classes within the GTSRB dataset, with each bar denoting a specific class label. The tallest green bar is labeled as Class 2, representing the class with the highest number of samples, while the shortest blue bar is labeled as Class 0, indicating the class with the lowest number of samples. Notably, a dashed red line is included to emphasize the required minimal number of samples, set at 100 samples

problems: a notable disparity in the number of samples across most classes and some classes having insufficient samples. The horizontal red dashed line indicates the minimum required number of samples per class (100 samples). In Fig. 7, corresponding to the GTSRB dataset, the primary concern is also a significant imbalance in the number of samples across most classes, similar to the LISAC dataset.

To demonstrate the robustness of our EGENet model, we deliberately do not address the data problems. While data is a crucial component of machine learning and DL, other key factors are also necessary to solve problems using these approaches. However, some previous works have solely focused on data, overlooking other crucial aspects such as loss functions and model complexity.

To overcome the challenge of imbalanced datasets in our work, we adopted a specialized loss function specifically developed to address this issue. We leveraged the Class-Balanced Focal Loss Based on the Effective Number of Samples function. Equation (14) represents the Class-Balanced Focal (*CBF*) Loss, which is an extension of the Focal Loss [53] The *CBF* Loss is defined as:

$$CBF(\mathrm{z}, y) = -\frac{1-\beta}{1-\beta^{n_y}} \sum_{i=1}^{\mathcal{K}} (1-p_i^t)^{\gamma} \log(p_i^t) \quad (14)$$

where $p_i^t = \text{sigmoid}(z_i^t) = 1/(1+\exp(-z_i^t))$. In this equation, z represents the predicted output from the model for all classes, where $\mathrm{z} = [z_1, z_2, \ldots, z_{\mathcal{K}}]$ and $\mathcal{K}$ is the total number of classes. The focusing parameter γ controls the down-weighting of the loss for well-classified examples. A higher value of γ emphasizes difficult examples more. When γ is set to 0, the *CBF* Loss reduces to the standard Cross Entropy Loss. The *CBF* Loss is further modified using the α-balanced variant, known as the class-balanced Focal Loss. Setting the parameter α_t in the Focal Loss to $(1-\beta)/(1-\beta^{n_y})$ makes it equivalent to the class-balanced Focal Loss. The class-balanced term allows for a direct and explicit way to set α_t in the Focal Loss based on the effective number of samples. In this work, we used $\gamma = 2$ and $\beta = 0.999$ based on the original paper [53].

Classification Evaluation

This section analyzes the results obtained during the training and testing stages of EGENet, highlighting its performance in TSC. The training process utilized a combination of 46,655 samples from the GTSRB dataset and 2412 samples from the LISAC dataset. Table 6 displays two key metrics: training time and the number of training steps per second. These metrics are crucial for determining the minimum workstation requirements needed to train EGENet or similar DL models within a reasonable timeframe.

A high $F1$ score generally indicates a model's ability to perform well on new data and effectively generalize. Achieving a balanced trade-off between training and validation losses is crucial when developing models with high $F1$ scores on new data. In our experiments, we observed a strong performance of the model on both datasets, with $F1$ scores approaching 1. For the GTSRB dataset, as shown in Fig. 8, both the training and validation $F1$ scores demonstrated excellent results, with a difference of less than 0.0029 between them. This indicates a minimal disparity, suggesting that both $F1$ scores were close to 1. Similarly, for the LISAC dataset illustrated in Fig. 9, we achieved favorable performance in both $F1$ scores. However, the validation $F1$ score exhibited a more significant difference than the training $F1$ score, with a difference of 0.0137, higher than the disparity observed in the GTSRB dataset. Based on our previous observations of the loss results in LISAC, this behavior was expected.

In order to thoroughly validate our model, we conducted a comprehensive analysis encompassing a wide range of commonly used metrics in classification tasks. We focused

Table 6 Training results on GeForce RTX 2080

Dataset	Model version	Epoch time (sec)	Step execution time (ms)	Throughput (samples/s)
GTSRB	Tiny	21	21	1,620,440
	Small	24	24	1,394,449
	Medium	77	75	436,023
	Large	134	132	249,075
LISAC	Tiny	1	14	157,380
	Small	2	20	110,773
	Medium	4	46	47,697
	Large	10	137	15,832

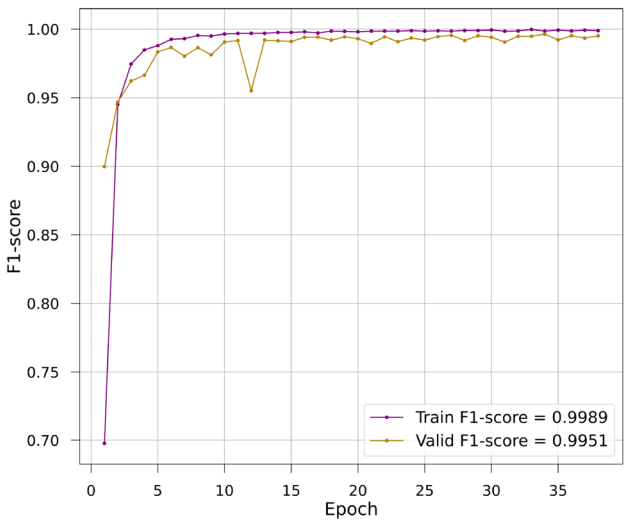


Fig. 8 Training and validation F1-Scores for the medium model on the GTSRB dataset, with early stopping at epoch 35 for best performance. Legend values indicate the maximum F1-scores reached in the last epoch

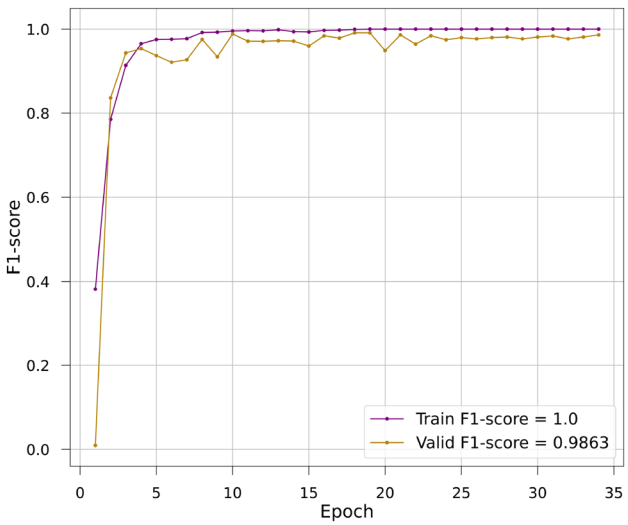


Fig. 9 Training and validation F1-Scores for the medium model on the LISAC dataset, with early stopping at epoch 35 for best performance. Legend values indicate the maximum F1-scores reached in the last epoch

on evaluating the performance of the EGENet model using four key metrics: F1 score (*F*1), accuracy (*ACC*), precision (*PREC*), recall (*REC*), and mean AUROC (*MAUROC*). We assessed the model's performance in three distinct scenarios: all classes, majority classes, and minority classes. This allowed us to identify the most relevant scenarios for applying our proposed model in this work. The results of our analysis are summarized in Table 5, providing a comprehensive overview of the model's performance in these different scenarios.

Table 7 provides compelling evidence of the notable performance of EGENet across all evaluation metrics (*F*1, *ACC*, *PREC*, *REC*, and *MAUROC*) for both datasets, demonstrating its robust generalization capability. Despite the diverse nature of the traffic sign signals in the datasets, with little similarity among most classes, EGENet consistently achieved remarkable performance, surpassing a threshold of 0.95 in all metrics across the three proposed scenarios.

In the first scenario, "All", which encompassed the 43 classes in the GTSRB dataset and the 15 classes in the LISAC dataset, EGENet displayed excellent results for both training and validation subsets. The validation subset in GTSRB consisted of 14,014 samples, while the LISAC validation subset contained 248 samples.

In the second scenario, "Majority", we focused on the five classes that appeared most frequently in the datasets. These classes in GTSRB were "Keep Right" (276 samples), "Priority Road" (279 samples), "Yield" (288 samples), "Speed Limit (30 km/h)" (294 samples), and "Speed Limit (50 km/h)" (300 samples). In the LISAC dataset, the majority of classes were "Keep Right" (128 samples), "Speed Limit 35 km/h" (208 samples), "Pedestrian Crossing" (230 samples), "Signal Ahead" (338 samples), and "Stop" (929 samples).

Lastly, the third scenario, "Minority", comprised the five classes with the lowest frequency in the datasets. These classes in GTSRB were "Pedestrian" (30 samples), "Go Straight or Left" (27 samples), "Speed Limit (20 km/h)" (27 samples), "Dangerous Curve to the Left" (27 samples), and "End of All Speed and Passing Limits" (30 samples).

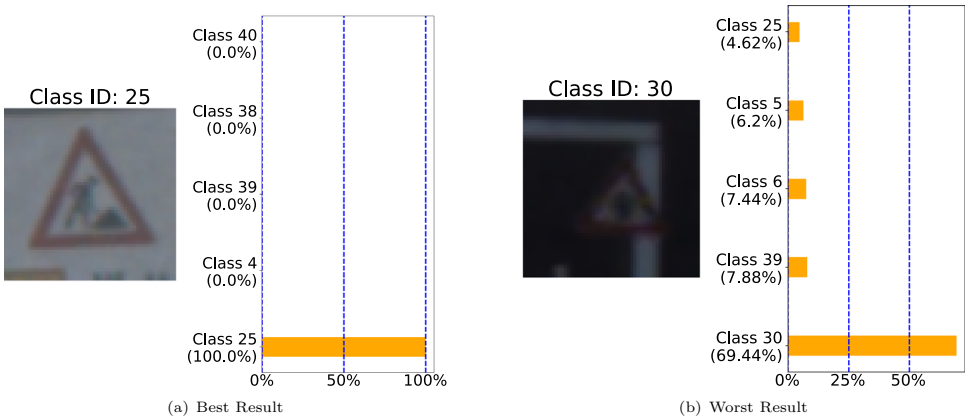
EGENet consistently demonstrated high performance across different evaluation scenarios, showcasing its effectiveness in handling the complexities of the datasets and its ability to achieve impressive results even for less frequent classes. At the same time, data quantity plays a critical role in DL model performance [54], it is worth noting that the GTSRB dataset provides a substantial amount of data, although the validation subset is imbalanced with fewer than 800 samples per class. On the other hand, the LISAC dataset presents a greater challenge, with fewer than 85 samples per class and even some classes with only five samples. Despite these data limitations and challenges such as undersampling and class imbalance, our model has consistently delivered excellent performance, surpassing the 0.95 threshold for all evaluation metrics and datasets presented in Table 7. This underscores the effectiveness of our model in overcoming data limitations, extracting meaningful patterns, and making accurate predictions even when data availability is limited.

Figures 10 and 11 present examples of the best and worst results obtained on the GTSRB and LISAC datasets, respectively. In Fig. 10a, we showcase the best result achieved on the GTSRB dataset using the proposed "medium" model version. This result includes the top five predictions with the highest confidence. The model successfully predicts the correct class with a prediction probability of 100%, demonstrating its robustness in this dataset and for the specific class (Class id: "2", Class name: "SpeedLimit(30 km/h)"). Furthermore, the prediction probability for the remaining four classes is 0%, further validating the model's accuracy.

In Fig. 10b, we present the worst result obtained on the GTSRB dataset using the same "medium" version of the model. This result also includes the top five predictions with the highest confidence. In this case, the model predicts the correct class with a probability of 69.44%, resulting in an error rate of over 30%. It is important to note that the model always selects the class with the highest probability.

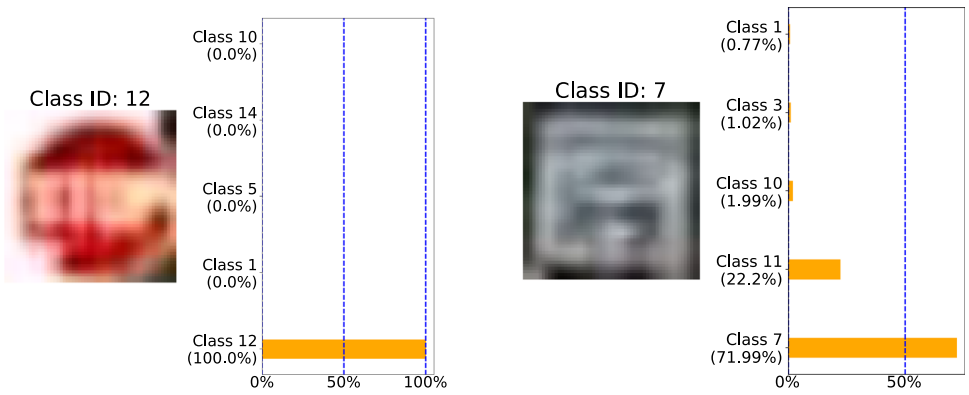
Table 7 Classification results obtained using the medium model, in bold, are the best results achieved

Dataset	Subset	Classes	*MAUROC*	*F*1	*PREC*	*REC*	*ACC*
GTSRB	Validation	All	0.9999	**0.9964**	0.9971	**1.0**	99.57
		Majority		0.9958	0.994	0.9914	**99.75**
		Minority		0.9934	**1.0**	0.9753	98.7
	Test	All	1.0	0.9953	**0.9942**	**1.0**	99.26
		Majority		**0.9945**	0.9941	0.9833	**99.5**
		Minority		0.9728	0.9827	0.9667	96.39
LISAC	Validation	All	0.9991	0.9912	0.9937	**1.0**	98.95
		Majority		0.9944	0.9971	**1.0**	99.19
		Minority		**1.0**	**1.0**	**1.0**	**100.0**
	Test	All	0.9996	0.9729	**0.9759**	**1.0**	97.39
		Majority		**0.98**	0.9686	**1.0**	99.26
		Minority		0.9686	0.941	**1.0**	**100.0**



(a) Best Result (b) Worst Result

Fig. 10 Best and worst results in GTSRB using the "medium" model



(a) Best Result (b) Worst Result

Fig. 11 Best and worst results in LISAC using the "medium" model

In controlled experiments with the specific GTSRB dataset, approximately 70% certainty or reliability is achieved for a specific class (Class id: "30", Class name: "Beware of ice/snow"). However, this percentage may vary in uncontrolled environments. Additionally, the prediction probability for the remaining four classes individually is below 10%. This is advantageous as it avoids class confusion, ensuring the model's performance and prediction accuracy. These experiments highlight the notable accuracy of the proposed model on the GTSRB dataset, both in its best and worst results.

In Fig. 11b, the worst result obtained on the LISAC dataset using the proposed "medium" version of the model is presented. This result showcases the top five predictions with the highest confidence. In this case, the model predicts the correct class with a prediction probability of 71.99%, resulting in a 28% error rate. It is important to note that the model always selects the class with the highest probability. In controlled experiments with the specific LISAC dataset, approximately 71.99% certainty or reliability is achieved for a specific class (Class id: "7", Class name: "SpeedLimit25"). However, this percentage may be lower in an uncontrolled environment. Additionally, the prediction probability for the remaining four classes individually is below 25%, although higher than what was obtained in GTSRB. This is due to the number of available samples, as mentioned in the Data Analysis Section (see "Data Analysis" section). It is worth noting that there is no other class with a high probability that could cause confusion between classes. Avoiding confusion between classes is crucial for the accurate functioning of the model and precise predictions. These experiments demonstrate the high level of accuracy achieved by the proposed model on the LISAC dataset, both in its best and worst results.

Comparison with SOTA

This section presents a comparative analysis of EGENet against previous works in the field. Table 8 displays the results of our three proposed versions of EGENet ("tiny", "small", and "medium"), and we compare them with the outcomes of previous studies addressing the same problem. accuracy (*ACC*), the most commonly employed metric in SOTA, was used to evaluate the classification performance. Regarding processing time, not all works directly address frames per second (*FPS*). Some studies estimate the model's latency (*L*), while others calculate processing time during training or omit it in real-time evaluation. Nevertheless, measuring *FPS* to assess real-time operation is essential, as emphasized in the literature cited in this work. To account for this, we calculate the *FPS* based on the reported results,

Table 8 SOTA results on GTSRB based on CNNs

Work	Hardware	Input (RGB)	Parameters (M)	*ACC* (%)	*FPS*
EGENet	RTX 2080	28^2	**0.01**	89.87	**1897**
Proposed model	Xavier				**594**
(tiny version)	Nano				**262**
EGENet	RTX 2080	112^2	0.13	98.9	948
Proposed model	Xavier				295
(small version)	Nano				126
EGENet	RTX 2080	224^2	0.49	99.58	766
Proposed model	Xavier				253
(medium version)	Nano				90
Human [4] (Average Performance) (2012)	N/A	N/A	N/A	98.84	N/A
Human [4] (Best Individual) (2012)	N/A	N/A	N/A	99.2	N/A
ENet-v1 [24] (2019)	GTX 2080	32^2	0.9	98.6	1613
MDEffNet [27] (2020)	Tesla K80	48^2	1.36	99.16	366*
RIECNN [11] (2022)	GTX 1070	60^2	3.2	99.75	952*
MicronNet-BF [12] (2022)	GTX 1080 Ti	48^2	0.44	99.383	0.09
TSC_MA [13]	Google Colab	32^2	0.9	**99.91**	6
(2023)	RPi 4B				

The $*$ indicates estimation based on their results. N/A indicates that the metric does not apply. The best results are highlighted in bold

as direct *FPS* values are not always provided. Considering the focus on real-time performance for embedded devices, we include recent works related to real-time TSC, regardless of whether they were implemented on an embedded device or not. Given the limited number of works addressing real-time performance on embedded devices, we extend our evaluation to encompass relevant studies that demonstrate competitive results for real-time TSC systems.

Table 8 presents the SOTA results in real-time TSC. Approximate results achieved by humans have been known since 2012 [4], demonstrating an average accuracy (*ACC*) close to 99%. This level of accuracy posed a significant challenge for the proposed models in this research area. Furthermore, two additional challenges arise: hardware implementation and response time. In the case of hardware utilized for implementation, specific hardware is required for embedded systems with limited resources. However, DL models, particularly those based on CNNs, demand highly specialized hardware such as GPUs. Nevertheless, these devices present their own set of challenges. Limited memory is one of them, stemming from their compact design and emphasis on energy efficiency. Another challenge is the response time required for real-time applications, which is demanding due to the limited computing resources available on these devices.

In the work conducted by Sanyal et al. in 2020 [27], promising results are obtained in terms of accuracy (*ACC*) and frames per second (*FPS*). Regarding accuracy, their proposed MDEffNet model surpasses the human average by 0.32. Furthermore, it achieves an adequate response time for real-time applications, with approximately 366 *FPS*, which is considered *super real-time* ($\geq$ 90 *FPS*), as mentioned by Bianco in [55]. On the other hand, the work by Abdel et al. in 2022 [11] achieves even better accuracy results (*ACC*) than those achieved by the best human individual ($ACC >$ 99.2), surpassing it by 0.55. Additionally, it achieves a notable high response time for real-time applications, with 952 *FPS*, also considered *super real-time*.

Based on the above, this work is competitive and surpasses the results obtained by the previous works. It is important to highlight that this work distinguishes itself by implementing the model on an *Embedded Device with Limited Resources*, which sets it apart from the previous works. When comparing EGENet with these previous works, we selected the version that achieves the best results in *ACC* ("medium" Version). It is worth noting that EGENet has a parameter count that is 2.77 times smaller than MDEffNet and 6.53 times smaller than RIECNN, making it, on average, 4.65 times smaller than both. The number of parameters directly impacts the inference time, i.e., the *FPS*.

Additionally, this enables us to estimate the required hardware and the time needed to train the CNN model. In terms of *ACC*, EGENet outperforms MDEffNet by 0.42. However, it falls short by 0.17 compared to RIECNN. Nevertheless, it is important to highlight that EGENet's ACC is 0.38 higher than that of the best human performer. In terms of *FPS*, EGENet is 2.09 times faster than MDEffNet. However, it is 1.24 times slower than RIECNN. Nonetheless, it is crucial to note that the "medium" version of EGENet meets the required response times to be considered as *super real-time*.

These results demonstrate that EGENet is not only a competitive model but also one of the pioneering models implemented on *GPU-Embedded Systems with Limited Resources*. It successfully surpasses the accuracy (*ACC*) of the best human performer and provides a real-time response (90 *FPS*), which is even considered as *super real-time*.

Table 9 provides details on the confidence intervals, mean, standard deviation, critical value t, and confidence level for the *ACC* and *FPS* metrics. Regarding the *ACC* metric, we observe a narrow confidence interval (0.0095) and a low standard deviation (0.0154), indicating high precision

Table 9 Results of confidence intervals. CI indicates confidence interval

Metric	Upper CI	Lower CI	Sample mean	Standard deviation	Critical value t	Confidence level
ACC	0.9974	0.9879	0.9926	0.0154	2.0181	0.95
FPS	0.849	0.7752	0.8121	0.1861	1.9842	0.95

Table 10 Hypothesis test results for *ACC*

Dataset	Test statistic (H)	Significance level (p-value)	Critical level (α)	Null hypothesis (H_0)
GTSRB	6.5893	$1.0259e^{-2}$	0.05	False
			0.01	True
LISAC	$1.4105e^{-2}$	$9.0546e^{-1}$	0.05	True
			0.01	True

Table 11 Hypothesis test results for *FPS*

Test statistic (H)	Significance level (p-value)	Critical level (α)	Null hypothesis (H_0)
$1.2570e^{2}$	$3.5776e^{-29}$	0.05	False
		0.01	False

in the estimation and minimal variation among values. These results suggest consistent and reliable outcomes. The critical value t of 2.0181 supports the accuracy of the confidence interval and provides further confidence in the results. On the other hand, when analyzing the *FPS* metric, we find a wider confidence interval (0.0738) and a higher standard deviation (0.1861) compared to *ACC*. This indicates greater variability in the results and a less precise estimation of the parameter of interest. There may be higher fluctuation in the *FPS* values, which could be attributed to various factors. In this case, the higher critical value t suggests increased uncertainty associated with the *FPS* estimation.

In this study, we propose the following alternative hypothesis (H_1): *Incorporating efficient convolutional layers into the design of a Convolutional Neural Network (CNN) markedly improves the balance between accuracy (ACC) and frames per second (FPS) in Traffic Sign Classification (TSC). This enhancement is expected to lead to more precise recognition capabilities and faster processing speeds, which are crucial for real-time applications.*

To validate this hypothesis, we employ the Kruskal-Wallis Test [56]. Table 10 focuses on *ACC* and Table 11 details the *FPS* metrics. The *ACC* metric is evaluated on the LISAC and GTSRB datasets, while the *FPS* metric undergoes 100 executions to ensure consistency. Both hypothesis tests are conducted using the two top-performing models proposed in this work: the "small" and the "medium" versions. These tables provide the decisive outcome of the work research, enabling us to determine the validity of the hypothesis put forth in this work.

Table 10 presents the Hypothesis Test results for the GTSRB dataset. Two scenarios were examined with significance levels of 0.05 and 0.01, respectively. In the case of 0.05, the null hypothesis (H_0) was rejected, indicating a statistically significant difference between the compared groups. Conversely, in the case of 0.01, the null hypothesis (H_0) was accepted, suggesting insufficient evidence to support such a difference. These findings suggest a statistically significant difference between the compared groups at a 0.05 significance level, while insufficient evidence was found at a stricter 0.01 level. The analysis was also conducted for the LISAC dataset, where the null hypothesis (H_0) was accepted at both significance levels (0.05 and 0.01), indicating no statistically significant difference between the compared groups. Therefore, these results do not provide sufficient evidence to claim a statistically significant difference between the compared groups in either case of the LISAC dataset.

Table 11 presents the Hypothesis Test results for the *FPS* metric. The analyses were conducted in two scenarios with significance levels of 0.05 and 0.01, respectively. In both cases, the null hypothesis (H_0) was rejected, indicating a statistically significant difference between the compared groups. These results prove that the observed differences are not due to random chance. Moreover, rejecting the null hypothesis even at the stricter significance level of 0.01 strengthens the robustness of the evidence for a statistically significant difference between the groups.

Limitations of the Study

Although the proposed EGENet architecture demonstrates competitive results in terms of accuracy and real-time performance on GPU-embedded devices, some limitations must be acknowledged. First, both GTSRB and LISAC datasets present class imbalance and limited samples for minority classes, which may restrict the model's ability to generalize to underrepresented categories in real-world conditions. Second, the LISAC dataset was initially designed for detection tasks; therefore, the extracted signs present variability in resolution and quality, which could influence classification robustness. Third, the evaluation focused exclusively on NVIDIA Jetson platforms, and although they are representative of current embedded GPU systems, additional testing in a broader variety of hardware platforms would further validate the scalability and portability of the

proposed approach. Finally, while our proposed evaluation metric integrates accuracy and FPS, future work should consider additional factors such as energy consumption and long-term deployment stability to ensure applicability in large-scale autonomous driving scenarios.

Conclusion

This work presented a novel architecture of CNN called EGENet for TSC in autonomous vehicles. It is a hand-optimized depth convolutional neural network that introduces an innovative convolution block, new concepts, and metrics that can outperform existing SOTA for real-time TSC applications. The deployed CNN demonstrated high accuracy and efficiency, surpassing human-level performance and competing with SOTA models on the GTSRB database. Hypothesis tests validated the superior performance of the model for both *ACC* and *FPS* metrics. The model's real-time responsiveness and suitability for resource-constrained embedded systems make it a promising solution for TSC in autonomous vehicles, enhancing road safety and autonomy. The EGENet CNN architecture strikes a balance between efficiency and accuracy, marking a significant advancement in the field.

The first contribution of this research is precisely the EGENet, a novel deep convolutional neural network architecture specifically designed for real TSC in GPU-Embedded Systems. EGENet demonstrates remarkable compactness with a significantly smaller parameter count, averaging up to 6.53× smaller than SOTA models. This efficient utilization of computational resources makes it highly suitable for deployment on embedded and mobile devices, achieving reduced processing time and improved real-time performance. EGENet addresses resource constraints without compromising classification accuracy, showcasing notable processing time superiority over compared models, being on average 2.09× faster than SOTA models. Its notable real-time performance makes it ideal for time-sensitive applications, particularly in autonomous driving and intelligent transportation systems. The comprehensive analysis confirms EGENet's superiority in terms of model parameters and processing time, establishing it as an ideal solution for resource-constrained embedded systems. The EGENet compactness and real-time capabilities significantly contribute to efficient TSC systems, especially in autonomous driving and intelligent transportation applications.

The second contribution focuses on enhancing the proposed architecture, EGENet, by integrating the "ADDSC" (Asymmetric Depth Dilated Separable Convolution) block. This integration brings efficient convolution operations from SOTA models, reducing model parameters and inference time while maintaining the receptive window size. By incorporating the ADDSC block, the study emphasizes the importance of efficient convolution operators in designing blocks optimized for real-time embedded applications, resulting in improved efficiency and performance of EGENet. The evaluation provides insights into different efficient convolution types, comparing parameters and computational time to understand the trade-offs and their impact on the computational efficiency of convolutional neural network models. The results demonstrate that the ADDSC method enhances feature representation, expands the receptive field, and improves classification accuracy. Leveraging advanced convolution techniques, EGENet achieves improved feature representation and classification accuracy, making it a valuable and significant contribution to real-time embedded systems.

The third contribution introduces a novel evaluation metric that combines *ACC*, *FPS*, and implementation on embedded GPU devices with constrained resources (such as NVIDIA Jetson Nano and Xavier AGX) to assess real-time TSC models. This comprehensive metric addresses the limitations of existing criteria, emphasizing efficient performance in real-time applications. Using this metric, the study demonstrates the superiority of EGENet, outperforming previous models, such as MDEffNet and RIECNN, in terms of higher accuracy and a significantly smaller parameter count. EGENet's implementation on embedded GPU devices further proves its viability for real-time TSC applications. Its impressive accuracy, compact parameter count, and *super real-time* response position it as a promising solution for TSC in resource-constrained environments. EGENet stands as a pioneering model for practical TSC applications, showcasing superior performance in terms of accuracy, real-time response, and efficient resource utilization.

EGENet stands out as a competitive and efficient architecture of depth convolutional neural networks, contributing to the field of autonomous driving. Although this research work has achieved solid findings, validated results, and a robust proposal, it is important to acknowledge and address certain limitations and restrictions:

Efficient deep learning requires substantial computational resources, particularly for training CNNs. To enhance efficiency, strategies such as hardware and software optimization, parallel training algorithms, and model compression are implemented, often leveraging platforms such as TensorFlow or PyTorch for distributed computing and GPU acceleration. Despite the resource-intensive nature, contemporary technological advancements have rendered this approach more achievable and cost-effective. Effective deep learning, on the other hand, links abundant, labeled datasets to train CNNs efficiently. Acquiring datasets for traffic signs, however, presents challenges due to real-world variabilities,

demanding significant time, resources, and expert input to ensure accuracy. The process requires diverse datasets encompassing various signs, angles, and lighting conditions, enabling the model to generalize and classify signs across scenarios adeptly. These extensive, labeled datasets play a pivotal role in successful CNN training, enabling precise real-time traffic sign classification.

This work presents valuable contributions both in theory and practice. Theoretical contributions include the proposal of a novel architecture for efficient deep learning-based TSC, supported by statistical results of comparative tests against the SOTA. Additionally, novel strategies for efficient deep learning-based TSC are introduced. On the practical side, the research presents an efficient architecture that enhances real-time classification in autonomous vehicles. These advancements directly impact road safety and drive progress in the field of autonomous driving.

As future work, we plan to extend the evaluation of EGENet beyond controlled datasets and embedded platforms. Specifically, we aim to test the model in real-world driving scenarios using on-board cameras to assess its robustness under dynamic environments, varying illumination, and diverse traffic conditions. In addition, we will conduct detailed energy profiling on embedded GPU devices during these experiments. This comprehensive evaluation will complement accuracy and inference speed results with power consumption analysis, which is critical for large-scale deployment in embedded and mobile systems and will further validate EGENet's applicability to autonomous driving and intelligent transportation systems.

Acknowledgements This research was made possible thanks to the Instituto Politécnico Nacional, through its institutional project SIP-20230104, and the generous funding provided by the Mexican National Council of Humanities, Sciences, and Technologies (CONAHCYT, Mexico). Additionally, we extend our gratitude to the Institutional Research Coordination of CETYS University for their valuable collaboration and assistance throughout the project. We are truly grateful for the contributions and support of these institutions, which have played a crucial role in successfully completing this endeavor.

Data availability The raw data supporting the findings of this study are available from the corresponding authors [19, 35] and is publicly available at GTSRB and LISA. Processed datasets generated during the current study are publicly accessible and can be downloaded from LISAC.

Code availability The code developed for this study is available at the GitHub repository: https://github.com/mmontielpz/egenet. All scripts, functions, and necessary documentation for reproducing the analyses presented in this paper are included in the repository.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Babic D, Babic D, Fiolic M, Šaric Ž. Analysis of market-ready traffic sign recognition systems in cars a test field study. Energies. 2021;14:3697.
2. Megalingam RK, Thanigundala K, Musani SR, Nidamanuru H, Gadde L. Indian traffic sign detection and recognition using deep learning. Int J Transp Sci Technol. 2023;12(3):683–99. https://doi.org/10.1016/j.ijtst.2022.06.002.
3. Kim C-i, Park J, Park Y, Jung W, Lim Y-s. Deep learning-based real-time traffic sign recognition system for urban environments. Infrastructures. 2023;8(2):20. https://doi.org/10.3390/infrastructures8020020.
4. Stallkamp J, Schlipsing M, Salmen J, Igel C. Benchmarking machine learning algorithms for traffic sign recognition. Man Comput Neural Netw. 2012;32:323–32.
5. Gudigar A, Chokkadi S, Raghavendra U. A review on automatic detection and recognition of traffic sign. Multimed Tools Appl. 2016;75:333–64.
6. Lopez-Montiel M, Rubio Y, Sánchez-Adame M, Orozco-Rosas U. Evaluation of algorithms for traffic sign detection. Opt Photon Inform Process. 2019;11136:135–51. https://doi.org/10.1117/12.2529709.
7. Lopez-Montiel M, Orozco-Rosas U, Sánchez-Adame M, Picos K, Montiel O. Evaluation of deep learning algorithms for traffic sign detection to implement on embedded systems. Stud Comput Intell. 2021;915:95–115.
8. Lopez-Montiel M, Orozco-Rosas U, Sanchez-Adame M, Picos K, Ross OHM. Evaluation method of deep learning-based embedded systems for traffic sign detection. IEEE Access. 2021;9:101217–38.
9. Mehta S, Rastegari M, Shapiro L, Hajishirzi H. Espnetv2: a light-weight, power efficient, and general purpose convolutional neural network. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, 2019-June, 2018:9182–92.
10. Wang J, Xiong H, Wang H, Nian X. Adscnet: asymmetric depthwise separable convolution for semantic segmentation in real-time. Appl Intell. 2020;50:1045–56.
11. Abdel-Salam R, Mostafa R, Abdel-Gawad AH. Riecnn: real-time image enhanced cnn for traffic sign recognition. Neural Comput Appl. 2022;34:6085–96.
12. Fang H-F, Cao J, Li Z-Y. A small network micronnet-bf of traffic sign classification. Comput Intell Neurosci. 2022;2022:3995209.
13. Triki N, Karray M, Ksantini M. A real-time traffic sign recognition method using a new attention-based deep convolutional neural network for smart vehicles. Appl Sci. 2023;13:4793.
14. Mi JX, Feng J, Huang KY. Designing efficient convolutional neural network structure: a survey. Neurocomputing. 2022;489:139–56.
15. Holder CJ, Shafique M. On efficient real-time semantic segmentation: a survey (2022). arXiv:2206.08605.
16. Liu J, Liu J, Du W, Li D. Performance analysis and characterization of training deep learning models on mobile devices. In: Proceedings of the international conference on parallel and distributed systems-ICPADS,2019-December, 2019;506–15.
17. Saadna Y, Behloul A. An overview of traffic sign detection and classification methods. Int J Multimed Inf Retr. 2017;6:193–210.
18. Shuvo MMH, Islam SK, Cheng J, Morshed BI. Efficient acceleration of deep learning inference on resource-constrained edge devices: a review. In: Proceedings of the IEEE 2022.
19. Stallkamp J, Schlipsing M, Salmen J, Igel C. The german traffic sign recognition benchmark: a multi-class classification competition. In: Proceedings of the international joint conference on neural networks 2011;1453–60.

20. Wali SB, et al. Vision-based traffic sign detection and recognition systems: current trends and challenges. Sensors. 2019;19:2093.
21. Han Y, Virupakshappa K, Pinto EVS, Oruklu E. Hardware/software co-design of a traffic sign recognition system using zynq fpgas. Electronics. 2015;4:1062–89.
22. Wong A, Shafiee MJ, Jules MS. Micronnet: a highly compact deep convolutional neural network architecture for real-time embedded traffic sign classification. IEEE Access. 2018;6:59803–10.
23. Benmeziane H. et al. A comprehensive survey on hardware-aware neural architecture search 2021. https://uphf.hal.science/hal-03269441.
24. Bangquan X, Xiong WX. Real-time embedded traffic sign recognition using efficient convolutional neural network. IEEE Access. 2019;7:53330–46.
25. Faiedh H, Farhat W, Hamdi S, Souani C. Embedded real-time system for traffic sign recognition on arm processor. Int J Appl Metaheuristic Comput (IJAMC). 2020;11:77–98.
26. Santos A, Abu PAR, Oppus C, Reyes RS. Real-Time traffic sign detection and recognition system for assistive driving. Adv Sci Technol Eng Syst J. 2020;5:600–11.
27. Sanyal B, Mohapatra RK, Dash R. Real-time embedded tsr using mdeffnet: an efficient compact dnn on varied multi-resolution images across multiple databases. Int J Wavel Multi Res Inform Process. 2020. https://doi.org/10.1142/S0219691320500824.
28. Mathias M, Timofte R, Benenson R. Gool LV. Traffic sign recognition: How far are we from the solution? In: Proceedings of the International Joint Conference on Neural Networks (2013).
29. Jurisic F, Filkovic I, Kalafatic Z. Multiple-dataset traffic sign classification with onecnn. In: Proceedings-3rd IAPR asian conference on pattern recognition, ACPR 2015;614–18 (2016).
30. Howard AG, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications (2017). arXiv:1704.04861.
31. Howard A, et al. Searching for mobilenetv3. In: Proceedings of the IEEE international conference on computer vision, 2019-October, 2019;1314–24.
32. Li G, Yun I, Kim J, Kim J. Dabnet: depth-wise asymmetric bottleneck for real-time semantic segmentation. In: 30th British machine vision conference 2019, BMVC 2019 (2019).
33. Wei T, Tian Y, Wang Y, Liang Y, Chen CW. Optimized separable convolution: yet another efficient convolution operator. AI Open. 2022;3:162–71.
34. Sermanet P, LeCun Y. Traffic sign recognition with multi-scale convolutional networks. In: Proceedings of the international joint conference on neural networks 2011;2809–13.
35. Mogelmose A, Trivedi MM, Moeslund TB. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey. IEEE Trans Intell Transp Syst. 2012;13:1484–97.
36. Tong K, Wu Y, Zhou F. Recent advances in small object detection based on deep learning: a review. Image Vis Comput. 2020;97:103910.
37. Liu Y, Sun P, Wergeles N, Shang Y. A survey and performance evaluation of deep learning methods for small object detection. Expert Syst Appl. 2021;172:114602.
38. Cho S, Seo C, Shin D-J, Kim J-J. A deep learning framework performance evaluation to use yolo in nvidia jetson platform. Appl Sci. 2022;12:3734.
39. Davis J, Goadrich M. The relationship between precision-recall and roc curves. ACM Int Conf Proc Ser. 2006;148:233–40.
40. Grandini M, Bagli E, Visani G. Metrics for multi-class classification: an overview (2020). arXiv:2008.05756.
41. Villardon G, Zhao X, Le PB, Nguyen ZT. Roc curves, loss functions, and distorted probabilities in binary classification. Mathematics. 2022;10:1410.
42. Yang Z, Xu Q, Bao S, Cao X, Huang Q. Learning with multiclass auc: theory and algorithms. IEEE Trans Pattern Anal Mach Intell. 2022;44:7747–63.
43. Hanhirova J, et al. Latency and throughput characterization of convolutional neural networks for mobile computer vision. In: Proceedings of the 9th ACM multimedia systems conference, MMSys 2018. 2018;18:204–15.
44. Galoogahi HK, Fagg A, Huang C, Ramanan D, Lucey S. Need for speed: a benchmark for higher frame rate object tracking. In: Proceedings of the IEEE international conference on computer vision, 2017-October, 2017;1134–43.
45. Ren Z, Fang F, Yan N, Wu Y. State of the art in defect detection based on machine vision. Int J Precision Eng Manuf-Green Technol. 2021;9(2):661–91.
46. Kyrkou C, Plastiras G, Theocharides T, Venieris SI, Bouganis CS. Dronet: efficient convolutional neural network detector for real-time uav applications. In: Proceedings of the 2018 design, automation and test in Europe conference and exhibition, DATE 2018-January, 2018;967–72.
47. Wang J, et al. Effect of frame rate on user experience, performance, and simulator sickness in virtual reality. IEEE Trans Vis Comput Gr. 2023;29(5):2478–88.
48. Paszke A, et al. Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 2019;32.
49. MerkelDirk. Docker. Linux Journal (2014).
50. Kingma DP, Ba JL. Adam: a method for stochastic optimization. In: 3rd international conference on learning representations, ICLR 2015-conference track proceedings (2014).
51. Cui Y, Jia M, Lin TY, Song Y, Belongie S. Class-balanced loss based on effective number of samples. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, 2019-June, 2019;9260–69.
52. Dodge S, Karam L. Understanding how image quality affects deep neural networks. In: 2016 8th international conference on quality of multimedia experience, QoMEX 2016 (2016).
53. Lin TY, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object detection. IEEE Trans Pattern Anal Mach Intell. 2020;42:318–27.
54. Johnson JM, Khoshgoftaar TM. Survey on deep learning with class imbalance. J Big Data. 2019;6:1–54.
55. Bianco S, Cadene R, Celona L, Napoletano P. Benchmark analysis of representative deep neural network architectures. IEEE Access. 2018;6:64270–7.
56. Ostertagová E, Ostertag O, Kováč J. Methodology and application of the Kruskal-Wallis test. Appl Mech Mater. 2014;611:115–20.