IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 October 2024, accepted 24 October 2024, date of publication 31 October 2024, date of current version 12 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3489476

RESEARCH ARTICLE

Enhanced Classification System for Real-Time Embedded Vision Applications

RAMZI KHELIFI[1], BRAHIM NINI[1], AND MOHAMED BERKANE[2]
[1]Research Laboratory on Computer Science's Complex Systems (ReLa(CS)2), University of Oum El Bouaghi, Oum El Bouaghi 04000, Algeria
[2]Artificial Intelligence and Autonomous Things Laboratory, University of Oum El Bouaghi, Oum El Bouaghi 04000, Algeria

Corresponding author: Ramzi Khelifi (ramzi.khelifi@univ-oeb.dz)

This work was supported by the Research Laboratory on Computer Science's Complex Systems (ReLa(CS)2), University of Oum El Bouaghi, Oum El Bouaghi, Algeria.

ABSTRACT Embedded computer vision systems are increasingly being adopted across various domains, playing a pivotal role in enabling advanced technologies such as autonomous vehicles and industrial automation. Their cost-effectiveness, compact size, and portability make them particularly well-suited for diverse implementations and operations. In real-time scenarios, these systems must process visual data with minimal latency, which is crucial for immediate decision-making. However, these solutions continue to face significant challenges related to computational efficiency, memory usage, and accuracy. This research addresses these challenges by enhancing classification methodologies, specifically in Gray Level Co-occurrence Matrix (GLCM) feature extraction and Support Vector Machine (SVM) classifiers. To maintain a high level of accuracy while preserving performance, a smaller feature set is selected following a comprehensive complexity analysis and is further refined through Correlation-based Feature Selection (CFS). The proposed method achieves an overall classification accuracy of 84.76% with a feature set reduced by 79.2%, resulting in a 72.45% decrease in processing time, a 50% reduction in storage requirements, and up to a 77.8% decrease in memory demand during prediction. These improvements demonstrate the effectiveness of the proposed approach in improving the adaptability and capabilities of embedded vision systems (EVS), optimizing their performance under the constraints of real-time limited-resource environments.

INDEX TERMS Embedded computer vision, limited resource systems, machine learning, pattern classification, real-time image processing.

I. INTRODUCTION

Technological advancements that were once the realm of science fiction have now become reality, seamlessly integrating into daily life and transforming our work and living environments. A notable example is the development of autonomous vehicles, including self-driving cars, which can interpret their surroundings and navigate traffic with minimal human intervention [1], [2]. However, the high cost of these advanced technologies remains a significant barrier to widespread adoption, limiting access primarily to those with significant financial means [3]. A potential solution lies in the use of relatively low-cost microcontrollers and single-board computers, which are capable of being programmed to perform many functions that traditionally require numerous expensive components [4], [5]. This approach can significantly reduce the overall cost of implementing advanced technology [6], [7]. Although designing an embedded system can be more complex than traditional solutions [8], [9], due to the need for specialized software and, in some cases, extra expansion modules, the advantages are significant. The majority of the design work focuses on software development, as embedded systems rely heavily on software to manage hardware components, ensure efficient operation, and allow for customization. Emphasizing software development allows for greater flexibility and cost-effectiveness over time, as software is easier

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenhua Guo.

to update and enhance than hardware [10]. By leveraging the power of microcontrollers and embedded systems, advanced technologies can be democratized, making them more accessible and affordable for everyone. This shift not only promotes innovation but also ensures broader access to technological innovations that were once merely a figment of our imaginations [11]. As we continue to bridge the gap between science fiction and science fact by leveraging these widely available and cost-effective systems, this brings us closer to a future where cutting-edge technology is within reach for all.

The main contribution of the current research lies in the development and optimization of a deployable, compact trained Support Vector Machine (SVM) model. We reduced the model size by 50% using advanced feature selection method, which significantly decreased storage requirements and improved memory usage and processing speed. This downsizing makes the model particularly adaptable for deployment in limited-resource environment [12], [13].

Furthermore, the research introduces a lightweight classifier that maintains a satisfactory balance between computational time, memory usage, and storage. Although there is a slight decrease in accuracy, this trade-off is justified by the substantial gains in adaptability and performance. Balancing these factors is critical for practical applications requiring quick response times and high operational efficiency. This makes the classifier highly effective for real-time embedded vision applications (Real-time EVA), where rapid processing and minimal resource consumption are essential [14], [15], [16].

II. RELATED WORKS

The integration of embedded systems into various industries represents a pivotal advancement in enhancing operational efficiency and precision [17]. These systems have revolutionized processes through real-time monitoring, data analysis, and automation [10], [18]. They are characterized by their affordability, compact size, low power consumption, and considerable processing capabilities, have been instrumental in addressing labor-intensive tasks [5], [6], [9]. Research demonstrates the potential of these solutions to significantly improve accuracy and speed, contributing to increased productivity [4], [7], [14], [19]. This literature review focuses on the recent advancements made over the past five years in this technology within distinct applications, examining their techniques, implementation, performance metrics, and implications for future practices.

By exploring these innovations, we aim to provide a comprehensive overview of the current state and future prospects of these cost-effective solutions across multiple domains. For instance, in the domain of assistance for visually impaired individuals [9], methods were evaluated to reduce the memory and computation time required in a classifier system, making it suitable for low-power microcontroller platforms. This approach maintains the accuracy obtained in workstations with much larger resources, adhering to the standards of a typical implementation. The target application, used for the experimental evaluation, is a crosswalk detector embedded on a wearable computer vision device used to help visually impaired persons.

The implementation combines Gray Level Co-occurrence Matrix (GLCM) and Support Vector Machine (SVM), enabling the embedding of a computer vision application based on a feature computation study that allowed authors to minimize the number of those features, sharing the least complexity level instead of GLCM's twenty four (24) statistical measures and the Correlation-based Feature Selection's (CFS) three (03) evaluated features as most performant, besides using resized images according to a proposed procedure to select the appropriate resolution. The authors reported that the experimental results indicate that it is not necessary to use the largest resolution available; a smaller resolution can still obtain good accuracy with low memory allocation. The feature selection criteria based on time complexity proposed proved to be efficient not only in execution time, but also in memory usage.

The main optimization effort focused on the feature extraction process, based on GLCM. In fact, the results showed that 94% of the classification time is spent in that process. The SVM classifier is complex, but the feature extraction is far more so. The SVM memory footprint depends on two aspects: the number of Support Vectors (SV) and the number of Decision Functions (DF), both defined by the training process. Previous works usually neglected the latter because SVM is a binary classifier. However, applications with more classes tend to use more data for decision functions.

Moving from healthcare to security, the application of embedded vision systems extends to large-scale surveillance as well. Reference [8] developed a scalable architecture to optimize image processing and object recognition in large-scale surveillance environments, reducing data transfer and bandwidth consumption while enhancing processing speed and scalability. This optimization is based on reducing the amount of data transferred and bandwidth consumption. The architecture is organized into three layers: the Perception Layer, the Network (Gateway) Layer, and the Application (Cloud) Layer.

The Perception Layer includes cameras and sensors that capture raw data. The Network (Gateway) Layer processes the raw data to reduce its size and complexity before transmitting it, handling tasks such as image enhancement and initial object detection. The Application (Cloud) Layer performs more complex processing tasks, such as object recognition and the coordination of the overall system.

The system divides image processing tasks between edge devices (gateways) and cloud servers. Lightweight preprocessing and object detection are conducted at the edge to reduce bandwidth consumption and minimize response delay. Only the recognized data, rather than entire image or video streams, is sent to the cloud for further processing and identification of specific targets.

The use of microservices allows the system to be modular. Each service, including preprocessing, object detection, and recognition, is developed, deployed, and scaled independently. This modularity enables the allocation and scaling of individual components as needed without impacting other parts of the system. According to the authors, the findings highlight significant improvements in large-scale surveillance operations by reducing bandwidth consumption and enhancing processing speed, scalability, and flexibility. Evaluations showed its convenience for applications in smart buildings and industrial parks, offering a more resource-efficient and responsive solution compared to traditional centralized systems. Nevertheless, this research faces several challenges, such as the strong dependency on internet connectivity and ambiguous allocation of tasks between the edge and the cloud, which is essential for maximizing system performance. Additionally, the experiment was conducted in a controlled lab environment rather than a real-world Internet of Things (IoT) scenario, which may limit the applicability of the results to larger, more complex environments.

In the field of traffic surveillance, On-board vision systems have also made contributions. Reference [10] introduced an embedded computer-vision system designed for multi-object detection in traffic environments. Their system addresses the limitations of existing systems, which are often expensive, complex to operate, and struggle with congestion, occlusion, and varying lighting conditions. The proposed system is designed to function accurately and swiftly under challenging conditions, such as illumination variations and different daytime scenarios. It focuses on detecting and categorizing traffic objects in diverse scenarios using a specialized framework based on an improved Sequential Monte Carlo (SMC) Faster R-CNN model.

The enhanced Modified Faster R-CNN (MF R-CNN) framework introduces a novel architecture to improve the detection of small objects and a new likelihood function that better selects target samples with correct labels. This improvement extends the likelihood function from the SMC framework based on background-subtraction spatial-temporal cue, favoring the selection of positive samples in specific scenes. A tracklet-based algorithm is used to assign weights to target samples according to their importance, prioritizing positive samples and reducing the inclusion of incorrectly labeled examples in the training dataset. The results indicate that the proposed likelihood function algorithm, enhances detector performance and accelerates the specialization process.

The findings demonstrate the effectiveness of the new likelihood function in accurately selecting positive samples and its superiority over the previously proposed SMC framework function. Future research should focus on overcoming the identified limitations, particularly by enhancing the robustness of the system to diverse traffic scenarios, optimizing computational performance, and ensuring high-quality, diverse training datasets.

Embedded computer vision has also impacted the medical diagnostics. In their study, Vora and Shrestha [11] presented an embedded vision algorithm designed to classify retinal images for the detection of diabetic retinopathy. Utilizing a convolutional neural network and k-fold cross-validation, the researchers trained their model on 88,000 high-resolution images from the Kaggle/EyePacs database. The algorithm achieved a detection accuracy of 76%, indicating potential for non-specialist diagnostic use in underserved areas. However, the current accuracy is insufficient for clinical deployment, necessitating further refinement.

The study underscores the promise of embedded computer vision in enhancing early detection of diabetic retinopathy and reducing vision loss among high-risk populations.

Microcontroller-based vision systems are also involved in environmental monitoring and wildlife management. Reference [4] investigated an IoT-based real-time image processing system integrated with a Deep Convolutional Neural Network (DCNN) to detect and classify animals, aiming to resolve human-wildlife conflicts through an early warning and monitoring module. Utilizing field cameras and RF networks, the system captures and transmits images to a base station, where a pre-trained AlexNet model processes them for feature extraction and classification. Alerts are then sent via Global System for Mobile Communications (GSM) to relevant authorities. Global Positioning System (GPS) information like latitude and longitude, animal type, gender classification in the case of elephants, pose recognition, age estimation (calf, aged), strength (group or individual), and log date and log time are provided and accessible through a secure web interface.

The research demonstrates effective animal detection and classification, even under challenging conditions, showcasing the practical application of IoT and deep learning in wildlife monitoring. It contributes significantly to wildlife management and conflict resolution, providing a foundation for further technological advancements in this field. However, potential limitations include dependency on data quality, environmental factors, scalability, and technical infrastructure.

In the context of waste management, incorporated vision systems have proven to be valuable as well. Myers and Secco [5] created an inexpensive computer vision system to help sort recyclables more efficiently in recycling centers. They built this system using a Raspberry Pi computer and used TensorFlow and OpenCV software to recognize different materials like glass, plastic, metal, paper, and cardboard. To train the system, they used a large dataset of trash images called TrashNet, along with additional images from the internet.

In tests, the system was able to accurately identify recyclables 90% of the time when working in a controlled, simulated environment. However, when they tried using it in real-time with a Raspberry Pi camera, the performance dropped. The accuracy went down to 70%, and the system

slowed to process just 1.4 images per second, compared to 10 images per second in simulations.

These issues were mainly due to the Raspberry Pi's limited computing power and the lower quality of the images from its camera. Despite these challenges, the study shows that with better hardware and clearer images, such a system could be very useful in automating the sorting of recyclables in recycling facilities. More work is needed to improve the system's speed and accuracy for it to be practical in real-world recycling operations.

Built-in vision systems are also enhancing safety in autonomous driving [6] developed a vehicle-mounted real-time pedestrian trajectory prediction system based on a small computer called Jetson Xavier. The system integrates pedestrian detection, pose estimation, and optical flow data to predict future pedestrian movements. Tested on the JAAD and PIE datasets, the system reduced displacement errors by 6.35% and 3.28% respectively, compared to the BiTraP model. These results underscore the value of multi-information fusion in trajectory prediction. However, the study's reliance on specific datasets and the need for broader real-world validation highlight areas for future research.

Similar to previous domains, agriculture has also experienced innovations driven by Integrated vision systems. Reference [7] developed a real-time embedded vision system (real-time EVS) for the online monitoring and sorting of citrus fruits, aimed at improving productivity and accuracy in agricultural operations. The system employs a Field Programmable Gate Array (FPGA) for hardware implementation, which performs fruit segmentation and classification based on color and size. The decision tree (DT) classifier was trained using a balanced dataset of reference images, ensuring robust pixel classification.

The system demonstrated high performance metrics, achieving 97% accuracy in fruit segmentation, 94% in color classification, and 90% in size classification. Operating at a rate of 60 frames per second (fps), the system ensures real-time processing capability. However, the current FPGA configuration is limited to handling two video inputs, which indicates a need for higher-capacity FPGAs to expand processing capabilities. Future work should focus on expanding the system's capabilities to handle a greater variety of fruits and increasing the number of video inputs to enhance processing power and throughput.

Aeronautics is also an active domain for computer vision applications. The study by [20] presents a vision-based in-flight collision avoidance system for Unmanned Aerial Vehicles (UAVs) using an embedded computing platform. The system utilizes a series of algorithms, including dynamic background subtraction, denoising, clustering, and tracking, to detect and avoid collisions with moving objects. Specifically, the implementation involves removing the background to highlight moving objects, applying morphological operations and binarization to reduce noise, clustering moving objects and eliminating noise blobs through Euclidean clustering, followed by object tracking using the Kalman filter. Additionally, stereo cameras are employed to estimate the three-dimensional trajectory of objects, and the system executes collision avoidance maneuvers based on the estimated trajectories. The system was tested on a low-cost embedded platform and demonstrated the capability to detect and track small moving objects, such as other drones, in real-time. The results highlight the system's effectiveness in performing evasive maneuvers, thereby showcasing the feasibility of deploying advanced vision-based collision avoidance techniques on cost-effective embedded systems. However, challenges such as dependency on lighting conditions and residual noise in background subtraction suggest areas for future improvement to address these limitations and ensure robust performance in diverse operational scenarios.

The conducted research by [19] in the field of robotics introduces an integrated computer vision system for enhancing the Long-fiber Embedded FRESH (LFE-FRESH) hydrogel 3D printing process. The system uses low-cost components to monitor and control the fiber embedding process in real-time, preventing fiber buckling and correcting over-extrusion. Tested with single-ply polyester and electrochemically aligned collagen (ELAC) fibers, the system demonstrated significant improvements in the structural integrity and accuracy of 3D printed hydrogel components. Despite its effectiveness, the study highlights limitations such as dependency on real-time response, suggesting areas for further optimization. This research underscores the potential of integrated computer vision systems in advancing hydrogel 3D printing applications, particularly in tissue engineering and biohybrid robotics.

A review of the literature reveals that embedded computer vision systems have achieved significant advancements across various domains, substantially enhancing operational efficiency, accuracy, and productivity [5], [8], [10], [20]. From healthcare [11] to environmental monitoring [7], waste management [5], and autonomous driving [6], these systems have revolutionized traditional processes through the implementation of innovative methods, techniques, and architectures [17]. However, despite the considerable progress made, there remains a pressing need to further refine computer vision algorithms and system designs for better adaptability to limited-resource environments.

A critical challenge within the field lies in addressing the high computational and memory demands of embedded computer vision systems [5], [6], [8], [9], [20]. This research specifically tackles these limitations by optimizing both the feature extraction process and the Support Vector Machine (SVM) classifier. Through our approach, we have successfully achieved a 79.2% reduction in the feature set, which significantly decreases the computational load without substantially compromising precision. The system maintains a high accuracy level of 84.76% while achieving a 72.45% reduction in processing time, which is particularly vital for

improving efficiency in environments where resources are limited.

In addition to computational efficiency, our research also addresses the issue of memory optimization [6], [7]. We demonstrate that by refining the prediction process, a 77.8% reduction in memory usage can be achieved, which is essential for the deployment of machine learning models in settings with constrained memory resources. Furthermore, we have developed a compact SVM model that reduces storage requirements by 50%, making it more suitable for environments where storage capacity is a limiting factor.

Despite a minor trade-off in accuracy, the lightweight classifier introduced in this research, which builds upon and improves the study by Silva et al. [9] on embedding computer vision applications into wearable devices, brings substantial improvements in overall operational efficiency. These achieved results represent a significant step forward in the state of the art, making EVS more effective for real-time applications. This strategic balance between efficiency and exactness is vital for the success real-time EVA, where both rapid processing and minimal resource consumption are critical. Our methods specifically enhance the performance of EVS in real-world applications, such as autonomous vehicles and industrial automation.

Following the review of related works, the paper is structured as follows: Section III outlines the proposed methodology, explaining the techniques and algorithms employed and presenting the experimental setup. Section IV discusses the evaluation metrics used to assess the model's performance, followed by a comprehensive analysis of the results, including the training, testing, and prediction phases. Section V concludes the paper with a summary of the contributions, insights gained, and suggestions for future research directions.

III. PROPOSED METHODOLOGY

Building on the foundation laid by previous research, specifically the methodology outlined in [9], this section introduces a method designed to optimize the existing technique discussed therein. Our approach, as illustrated by the flowchart in Fig. 1, directly addresses the limitations identified in [9], particularly the challenges related to performance in constrained environments, enabling deployment for real-time purposes. The enhancements we introduce include a 79.2% reduction in the feature set, leading to a 72.45% reduction in processing time while maintaining a high accuracy level of 84.76%. In the context of embedded vision systems, where real-time decision-making is critical, achieving an optimal balance between accuracy and efficiency is essential. The following methodology outlines these enhancements in detail, highlighting the efforts made to reduce the features vector length by 50% compared to the results achieved in [9], thus advancing the capabilities of embedded computer vision systems toward more effective real-time vision systems.

GLCM generates 24 features [21], [22], [23] measurable from a single image as input, the obtained characteristics are listed in Table 1. In the study conducted by Silva et al. [9] the researchers extracted the 24 features from 600 images of the crosswalks in their original size (1280 × 720), then they calculated the accuracy of the set labeled FS1, after a prediction test run on 50 samples from the dataset. The study showed that the use of the total 24 features led to a good average precision; approximately 91%. However, they aimed to reduce the size of the target embedded system, thus the approach based on feature complexity study was applied.

TABLE 1. GLCM'S features presented by reference.

Reference	Haralick et al. (1973) 14 Features	Soh and Tsatsoulis (1999) 6 Features	Wang et al. (2010) 4 Features
	[21]	[22]	[23]
Features	Angular Second Moment (ASM), Contrast, Correlation, Sum of Squares, Sum Average, Inverse Difference Moment (IDM), Sum Variance, Sum Entropy, Entropy, Difference Variance, Difference Entropy, Information Measures of Correlation I (IMC I), Information Measures of Correlation II (IMC II), Maximal Correlation Coefficient (MCC)	Homogeneity, Autocorrelation, Dissimilarity, Cluster Shade, Cluster Prominence, Maximum Probability	Sum Mean, Cluster Tendency, Difference Mean, Inertia
		Total	**24 Features**

The complexity study focuses on how the running time of an algorithm increases with the size of the input n [24]. In this case, the value of the input is the size of the Co-occurrence Matrix. This involves analyzing the algorithm's behavior as the input size grows, which helps in understanding its efficiency and scalability. All the features provided by GLCM are presented in Table 2 according to their time complexity [9], grouped into five levels and listed in ascending order. Taking in consideration the lowest complexity, the 10 features of the first group were chosen by Silva et al. [9] to build a new classifier and accuracy analysis labeled FS3, which was close to the features set labeled FS1, the accuracy of the total 24 GLCM's features.

Table 2 [9] shows that theta notation Θ(n) is employed, this notation indicates that the algorithm's time complexity is both upper and lower bounded by the input n, providing a precise and symmetric bound on the running time. This means that the algorithm's performance consistently grows linearly with the input size [25], making it a reliable measure of the algorithm's efficiency in all scenarios.

It is important to distinguish theta notation Θ(n) from Big O notation. While Θ(n) provides an exact bound, indicating that the algorithm's growth rate is tightly bound both above and below by n, Big O notation serves a different purpose, it is used to describe an upper bound on the time complexity meaning that the algorithm will not exceed a certain growth rate, but it could be faster in some cases. Big O is

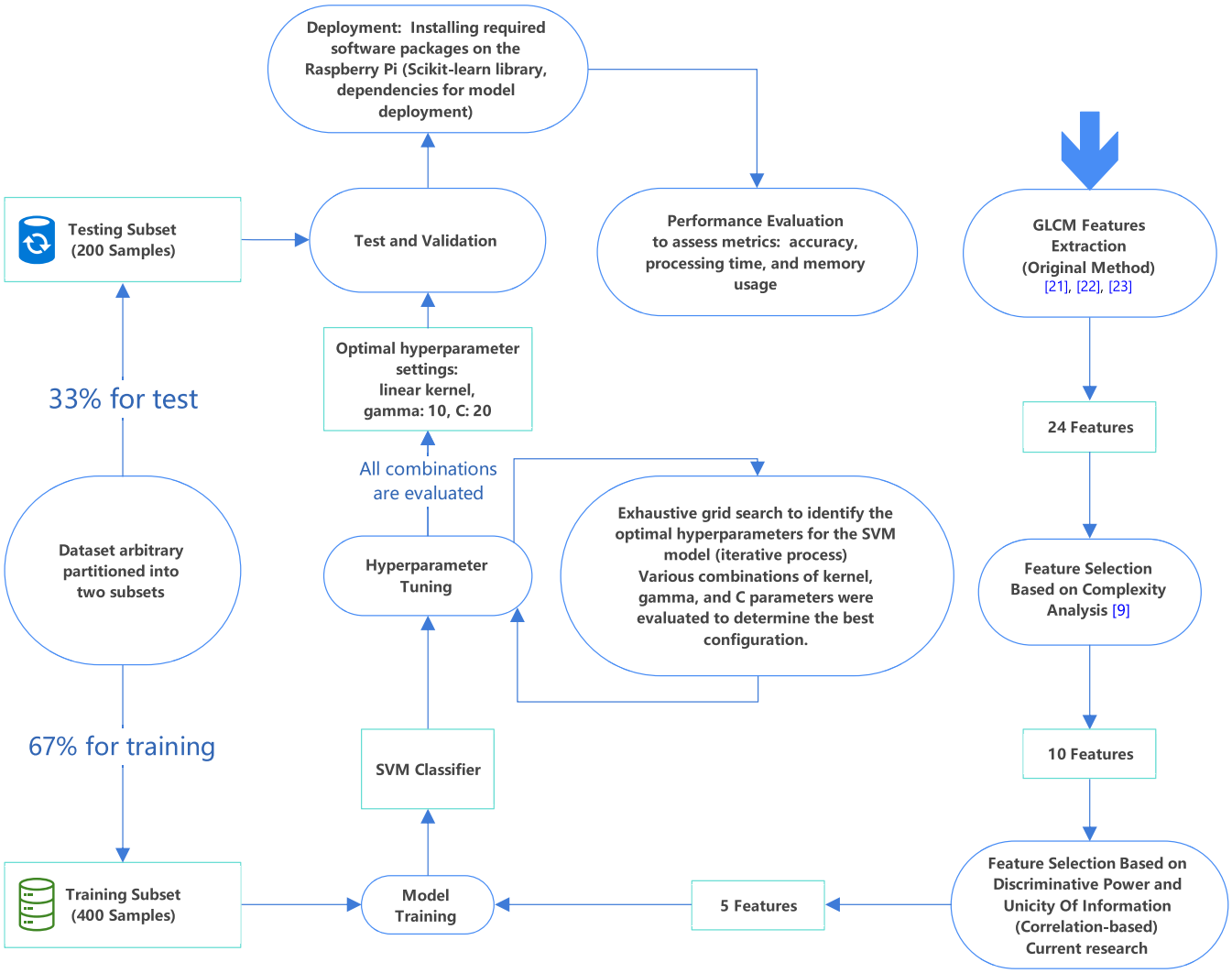

FIGURE 1. Flowchart visualizing the proposed methodology for enhancing the embedded classification system.

useful for understanding the worst-case scenario and ensuring that the algorithm's performance remains within acceptable limits [25].

However, in our context, where precise performance characterization is critical, especially for optimization in embedded vision platforms, $\Theta(n)$ is more appropriate. It offers a more accurate and comprehensive understanding of the algorithm's efficiency by reflecting its exact behavior across different scenarios.

While $\Theta(n^2+n)$ traditionally simplifies to $\Theta(n^2)$, similar to the other features, the "+n" term was retained to more accurately represent the nuanced differences in complexity particularly in the context of embedded systems where even small differences can significantly impact performance [9]. Additionally, retaining of constant factors like "+n" term allows for better distinction and grouping of features based on their computational demands, which helps reduce the overall number of features needed for effective optimization in performance-sensitive applications.

Our approach aims to compact the size of the feature's vector chosen in [9] to less than 10 features by combining complexity-based selection with Correlation-based Feature Selection (CFS). Selecting the most relevant features [26], with least complexity leads to reduce the computational cost for both, features extraction as well as classification, while preserving the highest performance of the classifier through optimal hyperparameter tuning.

Smaller classifiers require less computational power, leading to faster processing times. This is imperative for real-time applications, thereby unlocking the full potential of embedded systems in diverse and complex contexts [27].

The choice of Correlation-based Feature Selection (CFS) for this task is motivated by its ability to effectively select features that are most relevant to the target variable while minimizing redundancy among them [28]. Unlike

TABLE 2. List of GLCM's features grouped by time complexity [9].

	Running Time	Features
01	$\Theta(n^2)$	ASM, Contrast, IDM, Entropy, Homogeneity, Sum Mean, Maximum Probability, Dissimilarity, Difference Mean, Autocorrelation.
02	$\Theta(n^2 + n)$	Correlation, Inertia.
03	$\Theta(n^2 + 2n)$	Sum Average, Sum Entropy, Difference Variance, Difference Entropy.
04	$\Theta(n^2 + 4n - 2)$	Sum Variance.
05	$\Theta(2n^2)$	Sum of Squares, IMC I, IMC II, MCC, Cluster Tendency, Cluster Shade, Cluster Prominence.

other feature selection methods that may focus solely on relevance [29], [30], [31], [32], CFS evaluates both the correlation of each feature with the target variable and the inter-correlation among the features themselves. This ensures that each selected feature adds unique information, avoiding the selection of multiple features that provide overlapping information.

Throughout this process of selection, the algorithm naturally considers various combinations and numbers of features, ultimately selecting the five features that offer the highest discriminative power with the least redundancy for the S3 model. While alternative combinations and feature sets were implicitly assessed, none provided a better balance of accuracy and efficiency than the final set used in S3. This dual approach of maximizing information and minimizing redundancy is fundamental for enhancing model performance and efficiency [33], as supported by promising results in previous research [34], [35], [36].

Among 10 features selected by complexity study approach [9], CFS in our research highlighted 5 features, Table 3 compares the original vector [9] and the maintained features by CFS. The accuracy of the classifier using the feature set obtained through a combination of complexity analysis and Correlation-based Feature Selection (CFS) is detailed in Table 4, column S3. The first column in Table 4 represents the target classes, denoted as C1, C2, C3, and C4. These classes correspond to the specific labels that the classification system aims to distinguish. Examples of samples belonging to these four classes are illustrated in Fig. 4.

A. RECOGNITION PIPELINE

The proposed classification system presented in Fig. 2 consists of several sequential stages, each playing an integral role in the overall process. The system is designed to classify images using the Gray-Level Co-occurrence Matrix (GLCM) [21] as an extractor of characteristics and Support Vector Machine (SVM) as a classifier [9]. Once a frame is grabbed in the first step (A), it undergoes to be preprocessed and enhanced to result a resized image in second stage (B), this image is ready for the next process (C) where the extraction of regions of interest (ROIs) is performed.

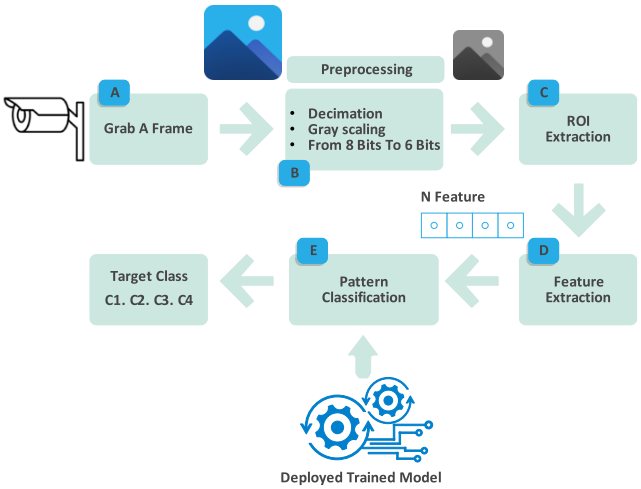

FIGURE 2. Classification pipeline [9].

Subsequently, the step (D) is features extraction, the GLCM is used as a descriptor. The image processing uses the GLCM as a statistical method for texture analysis. It examines the spatial relationship between pixels in an image, capturing the frequency with which pairs of pixel values (gray levels) occur at a specified distance and orientation. This matrix provides a detailed representation of the textural features of an image.

The classification phase (E) follows the feature extraction. It receives a vector of characteristics produced by the GLCM descriptor, initially comprising 24 original values. Based on a complexity analysis [9], these values are reduced to 10 selected features for class identification. In the current research, these ten low complexity features are further reduced to 5 based on their discriminative power and unicity of information, leading to a more efficient and informative feature set, tending toward real-time embedded deployment. This reduction in the number of features directly enhances the system's adaptability and its ability to perform efficiently in real-time limited-resource environments [37], which is the key aim of our paper. The SVM classifier was chosen for this task because it consistently outperforms other classifiers in various computer vision applications [12], [38], and [39]. SVMs are particularly effective in small-scale applications due to their computational efficiency, which becomes essential when working with smaller datasets. By reducing the number of features, the time and resources required for both training and prediction are significantly lowered, making SVMs a practical and efficient choice for real-time EVS [13]. This efficiency allows SVMs to provide robust and accurate classification results without the heavy computational burden associated with larger datasets [13].

The deployable model is an SVM classifier trained using the CSV dataset [40] with a reduced number of features. Before the training process, each 150 vectors' set of the four classes is arbitrary partitioned into two subsets according to the following rules: 100 lines equivalent to 67% for training, left 50 samples of each class for test.

The training and the testing of the classifier were performed with the optimal values of the hyperparameters: the kernel, C, and gamma calculated by an intensive operation called grid-search. Initially the training, testing and evaluating of the classification system was carried out on a machine equipped with Central Processing Unit (CPU) i5-11320H 3.20 GHz, 4 cores, 8 MB Intel ®Smart Cache, 16 GB of Random Access Memory (RAM), running windows 10 pro. We developed the model using Python and Scikit-learn library before deploying it into the embedded system's hardware environment, specifically a Raspberry Pi 4 Model B board, which is detailed further in Section III, subsection 'C. EVALUATING AND TESTING PLATFORM'.

B. DATABASE

The target database in this study is the Crosswalk Dataset, which is publicly available for download in [40]. It is divided into four classes: front crosswalk views (C1), left half-lane views (C2), right half-lane views (C3), and non-crosswalk images (C4) as shown in Fig. 4. Each class contains 150 samples, with a total of 600. The images were captured at 1280 × 720 resolution from videos recorded at 30 FPS in Fortaleza-CE, Brazil, during daylight. The images are named using the format class_p1_imageNumber.jpg (e.g., c3_p1_1.jpg).

Accompanying the dataset is the file 10_FEATURES_M17_CM6b_TH199.csv Fig. 3 integrated into the proposed classification system to serve training and prediction process. It details the data utilized on the embedded platform [9] generated using threshold images with T = 199, a decimation factor M = 17, and 10 GLCM features selected by the complexity analysis which are briefly described in Table 3, column description. They are coded on 64 bits as float64 data type, with the co-occurrence matrix considering only 6 bits of grayscale images.

C. EVALUATING AND TESTING PLATFORM

In our effort to target an embedded system, we deployed and evaluated the resulting classifier on a single-board computer to consider the limited resources of the hosting platform. Table 5 details the time consumed during the prediction of one sample by the three deployed SVM trained models according to the feature sets studied in this paper.

We used a Raspberry Pi 4 Model B as the hosting hardware. This device is equipped with a Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz, and 4GB LPDDR4-3200 SDRAM. We loaded The operating system and required software onto a Micro-SD card, with the Raspberry Pi OS, including the desktop environment and recommended software installed.

We selected the Raspberry Pi Model B as the embedded system platform due to its cost-effectiveness and versatility. The Model B's compatibility with a wide range of software and peripherals, coupled with its robust community support [41], [42], [43], made it an optimal choice for our project. Additionally, its low power consumption aligns well with the requirements of an efficient embedded system. The choice of Raspberry Pi also facilitates portability and ease of integration with other components [5], which are necessary for the successful deployment of our solution.

The development environment consisted of the Thonny IDE and Python 3.9, both of which were pre-installed on the Raspberry Pi OS. For the machine learning component, we selected the scikit-learn (sklearn) library. Scikit-learn is a free and open-source machine learning library for Python that offers a wide range of classification algorithms, including support-vector machines. Its compatibility with Python's numerical and scientific libraries, along with its extensive documentation and large user community, made it an optimal choice for this project. Table 5 illustrates the efficiency improvements achieved by decreasing the number of features used in the classifier. The prediction time for the full feature set (24 features) is 23.70 ms. By reducing the feature set to 10 complexity-based features, the prediction time decreases significantly to 11.48 ms. Further reduction to 5 complexity and correlation-based features results in an even faster prediction time of 6.53 ms.

This performance evaluation demonstrates that even with limited computational resources, it is possible to achieve efficient real-time predictions by optimizing the feature set used in the classifier. This makes the Raspberry Pi a viable platform for deploying machine learning models in embedded systems, particularly in applications where resource constraints are a critical consideration.

D. HYPERPARAMETER TUNING

To achieve the highest performance for our Support Vector Machine (SVM) model, we conducted an exhaustive hyperparameter tuning process. This process was comprehensive, systematic, and careful, involving the consideration and evaluation of all possible combinations of model parameters in multiple iterations to find the optimal settings [44]. Grid search, recognized for its exhaustive nature, was employed to explore various hyperparameter configurations, ensuring that the best possible model performance was achieved [45]. The process was both resource-intensive and time-consuming, requiring substantial computational power and significant effort to achieve the best configuration for the model. The hyperparameters significantly influence the performance of the SVM, and optimal values are critical for achieving high accuracy and efficiency [46].

Our grid search process tested various combinations of these hyperparameters to identify the best configuration. After the extensive evaluation, we determined that the optimal settings for our SVM model were a linear kernel, a gamma value of 10, and a C value of 20. The linear kernel was selected for its simplicity and effectiveness in handling linearly separable data, which aligns well with our feature selection strategy. This decision is supported by findings in recent studies, where the linear kernel has been shown to be effective in high-dimensional spaces and

	ASM	Contrast	IDM	Entropy	Homogeneity	Sum Mean	Maximum Probability	Dissimilarity	Difference mean	Autocorrelation	Class
1	0.718484	167.824806	0.885132	0.488888	0.892471	6.818140	0.847442	3.271628	-0.077364	291.956279	1
2	0.503199	193.217674	0.815848	0.849199	0.828933	14.135969	0.708527	3.989147	-0.077054	682.605271	1
3	0.857481	101.301705	0.945159	0.232984	0.947614	2.914419	0.925891	1.983256	-0.041550	102.298605	1
4	0.840310	82.300465	0.940856	0.272157	0.944588	3.662481	0.916589	1.664186	-0.049767	156.233798	1
5	0.718484	167.824806	0.885132	0.488888	0.892471	6.818140	0.847442	3.271628	-0.077364	291.956279	1

FIGURE 3. Snapshot of the first five lines from the CSV Dataset [40].

TABLE 3. GLCM's selected low complexity features description [9], and feature maintained by CFS algorithm (current research).

No.	Feature	Description	CFS maintained features	Reference
1	ASM (Angular Second Moment)	Measures textural uniformity and energy. High values indicate uniform textures.	✓	Haralick et al. (1973) 14 Features [21]
2	Contrast	Measures the intensity contrast between a pixel and its neighbor. Higher values indicate more contrast.	✓	
3	IDM (Inverse Difference Moment)	Measures texture homogeneity. Higher values indicate smoother textures.	✓	
4	Entropy	Measures randomness. Higher values indicate more complexity and disorder in the texture.		
5	Homogeneity	Measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal.		Soh and Tsatsoulis (1999) 6 Features [22]
6	Autocorrelation	Measures the correlation between pixels. Higher values indicate repetitive patterns.		
7	Maximum Probability	The highest probability value in the GLCM, indicating the most dominant pair of pixels.		
8	Dissimilarity	Measures variation. Higher values indicate greater variation in the texture.		
9	Difference Mean	The mean of the differences in pixel pairs in the GLCM.	✓	Wang et al. (2010) 4 Features [23]
10	Sum Mean	The sum of the mean of the rows and columns of the GLCM.	✓	

TABLE 4. Classifiers accuracy for each feature set.

Target Classes Fig. 4	[S1] Full features (24 features)	[S2] Complexity based (10 features)	[S3] Complexity and Correlation based (05 Features)
C1	91.54%	91.65%	86.96%
C2	93.06%	92.77%	86.79%
C3	88.77%	89.15%	85.63%
C4	88.47%	88.73%	79.65%
Average	90.46%	90.58%	84.76%

TABLE 5. Prediction time consuming on raspberry Pi.

	[S1] Full features (24 features)	[S2] Complexity based (10 features)	[S3] Complexity and Correlation based (05 Features)
Prediction	23.70 ms	11.48 ms	6.53 ms

real-time applications, offering a good balance between computational efficiency and generalization performance. For instance, Martínez et al. [47] demonstrated the suitability of the linear kernel in predicting remaining useful life in real-time applications, highlighting its efficiency in terms of computational cost and generalization performance. Similarly, Alnuayri et al. [48] showed the effectiveness of a linear kernel in on-chip aging estimation, further supporting our choice for this application.

The gamma parameter controls the influence of individual training examples, and a value of 10 was found to provide optimal equilibrium between bias and variance. The regularization parameter C was set to 20, which helped to penalize misclassifications and normalize the trade-off between achieving a low error rate on the training data and preventing overfitting [46].

These hyperparameter settings were critical in ensuring that our lightweight classifier maintained high precision despite the reduction in feature set size [37]. By fine-tuning the hyperparameters, we were able to optimize the classifier's performance, ensuring that it delivered reliable and efficient results suitable for real-time EVA [44]. The combination of advanced feature selection and precise hyperparameter tuning highlights the robustness and practicality of our enhanced classification system [49].

FIGURE 4. Samples from the four classes [40].

IV. RESULT AND DISCUSSION

This section presents the results obtained from the implementation of the proposed methodology, including the training, testing, and prediction phases. The results are analyzed in detail to evaluate the effectiveness and efficiency of the model, particularly in the context of its application to real-time EVS. We compare the performance of the proposed model with existing methods [9], highlighting key metrics such as accuracy, memory usage, and computational time. The discussion also explores the implications of these findings, considering the trade-offs between accuracy and efficiency, and how they align with the constraints typical of embedded systems.

A. COMPUTATIONAL TIME INVESTIGATION

This section focuses on pattern recognition aspects that affect computation time, primarily the feature extraction phase ensured by the use of GLCM [21].

Analysis of the results in Table 4 indicates that the classifier's mean accuracy was below 90%. Therefore, this combined approach did not yield optimal performance for this pattern recognition application. However, the reduction in feature vectors is expected to have a positive impact on computation time as demonstrated by Table 6.

While our approach of combined selection did not maintain the same accuracy as [9], it is important to consider the trade-offs between accuracy and computational efficiency. The reduced feature set significantly decreases the computational load, leading to faster processing times and lower resource consumption [37]. This is particularly beneficial in real-time applications or scenarios with limited computational resources [50].

Moreover, the use of fewer features would simplify the model, making it more compact, interpretable and easier to maintain. In practice, the balance between accuracy and efficiency should be tailored to the specific requirements and constraints of the end use [49].

The analysis of computational time of each feature set at different stages is demonstrated in Table 6. Moving from 24 features to 10 features (S1 to S2) [9] and then to 5 features (S3) achieved by our research, progressively reduces the training, testing, and one instance prediction times. This indicates that feature selection methods, both complexity-based and a combination of complexity and correlation-based, effectively improve computational efficiency.

TABLE 6. Computational time at different stages of each feature set.

Stages	[S1] Full features (24 features)	[S2] Complexity based (10 features)	[S3] Complexity and Correlation based (05 Features)
Training	2 min 21s	1 min 38 s	33.1 s
Testing	230.8 ms	125 ms	78.1 ms
Prediction	12.31 ms	7.03 ms	3.44 ms

The most substantial reductions in time are observed in the training phase, but significant improvements are also seen in testing and prediction phases. The provided chart Fig. 5 visually confirms the benefits of using the S3 feature set, which comprises only 5 features. The prediction times for S3 are the lowest across all phases, with a prediction time of 3.44 milliseconds compared to 7.03 milliseconds for S2 and 12.31 milliseconds for S1.

This sharp decline in prediction time illustrates the significant computational efficiency gained by combining complexity and correlation-based feature selection. Reducing the number of features not only decreases computational complexity and resource consumption, but also enhances model performance by preventing overfitting and improving data interpretability [9]. These advantages are critical for ensuring optimal performance in real-time applications and scenarios where computational resources are limited [50].

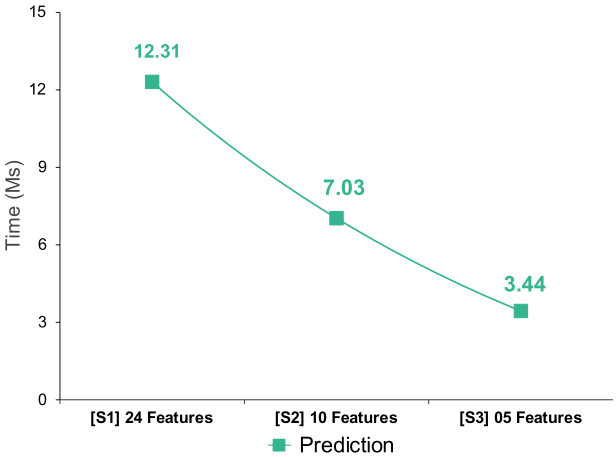

FIGURE 5. One instance prediction consuming time in milliseconds.

With larger datasets, the computational load increases exponentially due to the sheer volume of data and the complexity of processing a high number of features [49]. Reducing the number of features significantly moderates these effects, making the training, testing, and prediction processes more scalable and efficient [50].

As the number of features in a dataset decreases, the computational load also decreases because fewer data points need to be processed. Specifically, the total number of processed values is calculated as the product of the number of samples n and the number of features m. For instance, the used dataset [40] in current research contains 600 samples, as detailed in Section III: Methodology, Subsection: Database, using 24 features (S1) results in 14,400 processed values, reducing to 6,000 processed values with 10 features (S2), and further decreasing to 3,000 processed values with 5 features (S3).

Research by Guyon and Elisseeff [37] further validates the impact of reducing the number of features in a dataset. They highlight that feature selection plays a critical role in improving machine learning model performance by reducing computational complexity. Their study demonstrates that the small number of features not only speeds up the training and inference processes but also helps in addressing the challenges of high-dimensional spaces, leading to better generalization and more robust models, requiring less memory and computational power, which is especially beneficial when dealing with large datasets typically characterized by a high number of samples which makes them more computationally intensive, especially when deploying models on hardware with constrained resources.

B. MEMORY USAGE INVESTIGATION

The Table 7 lists both the peak memory usage and the increment in memory usage for each feature set and phase.

TABLE 7. Memory usage at different stages of each feature set.

Stages	[S1] Full features (24 features) Peak Memory Increment	[S2] Complexity based (10 features) Peak Memory Increment	[S3] Complexity and Correlation based (05 Features) Peak Memory Increment
Training	197.98 MiB 0.84 MiB	197.55 MiB 0.53 MiB	197.32 MiB 0.26 MiB
Testing	198.61 MiB 0.63 MiB	197.99 MiB 0.44 MiB	197.51 MiB 0.19 MiB
Prediction	198.98 MiB 0.36 MiB	198.23 MiB 0.24 MiB	197.45 MiB 0.08 MiB

These metrics were obtained by monitoring the memory consumption during the execution of the classifier system using the 'memory_profiler' library in Python which generates a detailed report of memory consumption. They provide valuable insights into the memory efficiency of different feature selection methods during the training, testing, and prediction phases. Peak memory is the highest amount of memory used at any point during the execution of a phase. It is measured in mebibytes (MiB), where 1 MiB equals 1,048,576 bytes. High peak memory usage indicates that the process requires a substantial amount of memory at its maximum point, which can affect the overall system performance and resource allocation. Increment, on the other hand, refers to the net increase in memory load from the start to the end of a phase, Fig. 6 demonstrates visually the increments in memory allocation for each feature set. Also measured in MiB, the increment value shows how much additional memory was needed to complete the phase. This metric is indispensable for understanding the dynamic memory demands of the process [19].

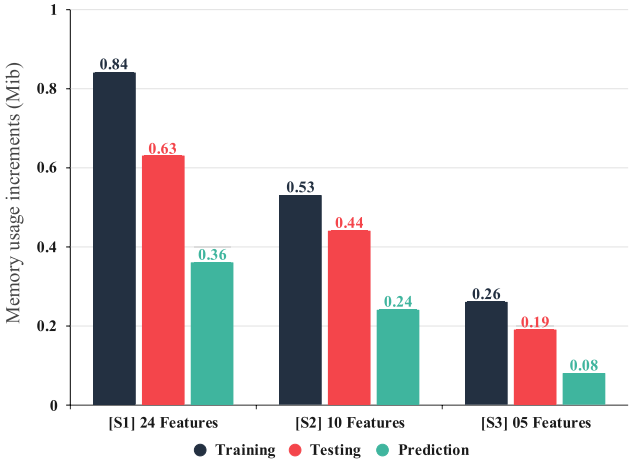

FIGURE 6. Increments in memory usage at different stages of the feature sets, measured in MiB.

In the training phase, the feature set S1 (Full Features) has the highest peak memory usage and increment, reflecting the substantial memory required for handling a larger number of features. S2 (Complexity-Based Features) shows a noticeable reduction in both peak memory and increment compared to S1, indicating improved efficiency. S3 (Complexity and Correlation-Based Features) demonstrates the lowest peak memory usage and increment, highlighting the significant efficiency gains achieved by further reducing the feature set.

During the testing phase, S1 again exhibits the highest peak memory and increment values, maintaining its trend of memory demand. S2 significantly reduces these values compared to S1, showcasing better memory management. S3 maintains the lowest values, indicating optimal memory efficiency during the testing phase. This pattern persists in the prediction phase, where S1 shows the highest memory consumption, both in peak and increment. S2 improves upon S1 but still uses more memory than S3. S3 has the lowest peak memory and increment, demonstrating the benefits of combining complexity and correlation-based feature selection in reducing memory consumption.

Table 7 clearly demonstrates the benefits of reducing the number of features in memory usage. As the number of features decreases, both peak memory and memory increment are reduced across all phases (training, testing, and prediction). This reduction is most pronounced with the S3 feature set, which combines complexity and correlation-based selection to achieve the lowest memory allocation.

In resource-constrained environments such as embedded systems, even modest memory savings can have a significant

impact on overall system performance [5]. The percentage reductions in memory load we achieved through feature reduction are substantial. A 69.0% reduction in memory during training, and up to a 77.8% reduction during prediction, can make a critical difference in the feasibility and efficiency of deploying machine learning models in such environments. This highlights the practical significance of our approach, where reducing the number of features leads to meaningful improvements in memory efficiency, directly contributing to the model's suitability for deployment in resource-limited settings. As highlighted earlier in the paper, the advantages of our approach become increasingly pronounced with larger datasets, where the demands on memory and computational resources grow substantially.

Moreover, while the peak memory remains almost the same, the reduction in incremental memory usage is highly advantageous in systems where memory optimization is imperative, particularly when we separate image acquisition and feature extraction from prediction. This scenario mirrors the approach demonstrated in scalable architectures [8], where task separation between low-resource (edge devices) and high-resource (cloud servers) components leads to significant resource efficiency. By applying a similar strategy, the computational and memory demands are reduced at different stages of the model's workflow, making the system more efficient in constrained environments.

Although reducing the number of features led to a minor decrease in accuracy, the trade-off is justified by significant gains in computational efficiency and resource management [37]. The slight decrease in accuracy is negligible compared to the substantial improvements in processing speed, memory usage and the potential of deploying a more compact model [5]. which are essential for real-time applications and scenarios involving large datasets. These results highlight the importance of our feature selection method in optimizing machine learning models for deployment on resource-constrained devices.

Table 8 provides a comparison of one features' vector length for each feature set used in our study, memory size of the feature vector is a critical factor in determining the memory footprint and overall efficiency of machine learning models, starting from feature extraction to classification, and ending by model deployment.

TABLE 8. Memory usage at different stages of each feature set.

	[S1] Full features (24 features)	[S2] Complexity based (10 features)	[S3] Complexity and Correlation based (05 Features)
Feature Vector Length (Bits)	1536 Bits	640 Bits	320 Bits

The S3 feature set, which combines complexity and correlation-based selection and uses only 5 features, reduces the features vector length to 320 bits. This represents a significant reduction of approximately 79.2% compared to the S1 feature set. The compact features vector not only reduces the allocated memory space but also improves the model's performance by enabling faster training, testing, and prediction phases. The reduction in features vector length directly implies lower memory requirements, which is particularly advantageous for deployment in resource-constrained environments [50].

Small sized features' vector provides another significant advantage: improved data transfer efficiency. In distributed computing environments, such as cloud-based applications or edge computing [8], smaller features' vectors require less bandwidth for data transfer between storage and processing units. This can lead to faster data transmission and reduced latency, significantly improving the model's real-time processing capabilities in the event of eventual implementation in such environments.

C. STORAGE INVESTIGATION

In addition to the analysis of memory usage during the different phases of the classifier operation, it is important to consider the size of the deployable trained SVM model, this is especially vital for deployment in storage-limited environments like embedded systems and IoT devices [5]. Table 9 illustrates the size of the deployable trained SVM model for each feature set.

TABLE 9. Deployable trained SVM model size in kilobytes (KB).

	[S1] Full features (24 features)	[S2] Complexity based (10 features)	[S3] Complexity and Correlation based (05 Features)
Model size	56 KB	36 KB	28 KB

For the S1 feature set, the full set of 24 features results in the largest model size of 56 KB. This is understandable given the higher complexity and the larger amount of data required to represent the full feature set. When moving to the S2 feature set, which uses complexity-based selection to reduce the number of features to 10, the model size decreases to 36 KB. This reduction highlights the benefit of removing less significant features, which reduces the overall size of the model by approximately 35.7%, without compromising too much on its performance. The S3 feature set, which combines complexity and correlation-based selection to use only 5 features, results in the smallest model size of 28 KB. This shows the most significant reduction in model size, approximately 50% compared to S1, making it highly suitable for deployment in resource-constrained environments.

TABLE 10. Comprehensive comparison of the three classification methods across several performance metrics.

Metric	Baseline Method (Full features)	Complexity Based Method	Complexity and Correlation Based Method	Percentage Change (Complexity Based)	Percentage Change (Complexity and Correlation Based)
Accuracy (%)	90.46	90.58	84.76	0.13%	-6.31%
Computational Time on Station (Prediction) (ms)	12.31	7.03	3.44	42.89%	72.06%
Computational Time on Embedded System (Prediction) (ms)	23.70	11.48	6.53	51.52%	72.45%
Memory Usage (MiB) Peak Memory	198.98	198.23	197.45	0.38%	0.77%
Prediction Increment memory (MiB)	0.36	0.24	0.08	33.33%	77.78%
Training Increment memory (MiB)	0.84	0.53	0.26	36.90%	69.05%
Model Size (KB)	56.00	36.00	28.00	35.71%	50.00%

The deployable model size directly impacts the feasibility of using the SVM model in various applications, especially those requiring compact and efficient models. The reduction in the number of features not only improves computational efficiency but also reduces significantly the storage requirements for the model.

D. ACCURACY INVESTIGATION

In our research, the S3 subset was specifically designed to ensure an optimal balance between precision and computational complexity, achieving an average accuracy of 84.76%. While this represents a 5.7% decrease from the S1 and S2 subsets, it is a calculated trade-off that is both necessary and justifiable within the context of embedded systems, where resources such as computation and power are critical constraints. Previous research has demonstrated that in such environments, a slight decrease in accuracy typically in the range of 2% to 10% can be acceptable if it results in significant gains in efficiency and reduced computational load, as seen in energy-efficient signal processing and wearable health monitoring systems [18], [50]. Furthermore, in the broader field of machine learning, reducing model complexity to manage computational demands is a well-accepted practice, even in deep learning models, where accuracy losses of approximately 2% to 7% are often tolerated to achieve computational feasibility [51]. These trade-offs are essential for ensuring that models can operate effectively within the limited resources available in real-time applications.

Moreover, an accuracy of 84.76% remains highly practical for many real-world applications, particularly those that prioritize computational efficiency over maximum precision. For instance, studies in embedded deep learning have shown that computational accuracy trade-offs, often in the range of 5% to 10%, are leveraged to maintain performance within the stringent constraints of embedded devices [52]. Similarly, in real-time vision applications, achieving a balance between accuracy and processing speed is vital, as slight reductions in accuracy up to 5% are often compensated for These results underscore the importance of our feature selection method by the increased responsiveness and reliability of the system [14]. Even in advanced deep learning architectures, trade-offs between depth and computational feasibility are necessary, with accuracy reductions of 3% to 7% being acceptable to optimize performance for real-world deployment [15]. These principles align with our approach, where the accuracy trade-off associated to reduction in features led to optimizing performance in resource-constrained environments.

Our approach is especially relevant in embedded systems, where even small differences in computational demands can significantly impact overall performance. By reducing the model complexity while maintaining essential features, we ensure that the model is both efficient and effective, capable of operating within the stringent constraints typical of embedded applications. Research in platform-based design underscores the importance of these trade-offs, showing that modest decreases in accuracy are often outweighed by the benefits of reduced complexity and improved system efficiency [16], [53]. Therefore, the 84.76% accuracy achieved by S3 is not only sufficient but also strategically optimal for the intended real-world scenarios, where balancing performance with computational efficiency is essential.

E. OVERALL PERFORMANCE ASSESSMENT OF THE CLASSIFIER

The results from our study reveal significant insights into the trade-offs between accuracy, computational time (on both standard and embedded systems), memory usage, and

model size when employing different feature selection methods. Table 10 provides a detailed comparison of the three classification methods across several performance metrics, Baseline Method (Full features), Complexity Based Method, and Complexity and Correlation Based Method across several performance metrics.

The comparative analysis of feature selection methods highlights the delicate balance between maintaining high accuracy and maximizing computational efficiency [5], [6], [19]. The Complexity Based Method stands out with a minimal accuracy gain (0.13%) while delivering substantial reductions in computational time (42.89% on standard systems and 51.52% on embedded systems) and modest memory savings (0.38%), alongside with a significant reduction in model size (35.71%). This method is particularly advantageous for applications requiring both performance and efficiency, such as real-time EVS.

In contrast, the Complexity and Correlation Based Method (our proposed method) excels in computational efficiency, demonstrating a remarkable 72.06% reduction in computational time on standard systems, a 72.45% reduction on embedded systems, a 0.77% decrease in memory usage, and a 50.00% reduction in model size.

The prediction increment memory shows a reduction from 0.36 MiB for S1 to 0.08 MiB for S3, representing a substantial 77.78% improvement. Similarly, the Training Increment memory decreases from 0.84 MiB for S1 to 0.26 MiB for S3, reflecting a 69.05% reduction. These metrics highlight the significant memory efficiency achieved by reducing the number of features, further validating the system's suitability for resource-constrained environments. Even though peak memory usage may appear similar across models, the incremental memory savings during training and prediction are important for embedded systems, where optimizing resources is essential.

The effectiveness of this approach becomes more evident in scenarios where memory increments are substantial, even when peak memory remains relatively small. By distributing tasks like image acquisition, feature extraction and the prediction process, memory demands are managed more effectively, making these incremental savings particularly valuable. This reflects the strategy employed in the study [8], where task distribution significantly improved resource efficiency.

Despite a 6.31% accuracy drop, this method is highly suitable for applications prioritizing speed and resource efficiency over absolute precision [14], [15], [51], [52], making it optimal for scenarios like real-time monitoring and rapid data processing in constrained environments.

The outcomes of this study have significant implications for the development and deployment of machine learning models in various domains. The marked reduction in computational time highlights the potential for real-time applications, particularly in environments with stringent processing time requirements [5], [6], [10], [19]. The modest memory usage gains and significant reductions in model size further emphasize the methods' applicability in resource-constrained scenarios, extending the operational life of battery-powered devices and reducing hardware demands.

V. CONCLUSION

The combined application of complexity-based feature selection and Correlation-based Feature Selection (CFS) in our study, demonstrated significant benefits in optimizing Support Vector Machine (SVM) classification models, particularly for real-time embedded vision applications. By reducing the number of features from 24 to 5, we achieved promising results. Our approach resulted in a 50% reduction in model size compared to the full feature set, a notable decrease in memory increments reached 77.78% when predicting outputs, and significantly faster computational times, primarily a 72% reduction in prediction time, from 12.31 milliseconds to 3.44 milliseconds, making our model highly suitable for real-time processing. Although there was a minor decrease of 6.31% in accuracy, this trade-off proved tolerable given the achieved substantial performance improvements.

These findings underscore the effectiveness of our approach in enhancing computational efficiency, scalability and adaptability, enabling the deployment of SVM models in limited-resources environments such as single-board computers like Raspberry Pi and IoT systems.

Looking forward, the integration of advanced feature selection methods in SVM classifiers opens new avenues for further research and practical application. Future studies could explore the combination of feature selection with other machine learning techniques, such as ensemble methods or deep learning, to further enhance model performance and efficiency. Additionally, investigating the impact of feature selection on other types of classifiers and across various domains, such as healthcare, finance, and safety, could provide broader insights into the versatility, adaptability and applicability of these methods.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] H.-S. Kim and I. Joe, "An XAI method for convolutional neural networks in self-driving cars," *PLoS ONE*, vol. 17, no. 8, Aug. 2022, Art. no. e0267282, doi: 10.1371/journal.pone.0267282.

[2] L. R. Polamreddy and Y. Zhang, "LaksNet: An end-to-end deep learning model for self-driving cars in udacity simulator," 2023, *arXiv:2310.16103*.

[3] K. Othman, "Public acceptance and perception of autonomous vehicles: A comprehensive review," *AI Ethics*, vol. 1, no. 3, pp. 355–387, Aug. 2021, doi: 10.1007/s43681-021-00041-8.

[4] T. Surya and S. Selvaperumal, "The IoT-based real-time image processing for animal recognition and classification using deep convolutional neural network (DCNN)," *Microprocessors Microsyst.*, vol. 95, Nov. 2022, Art. no. 104693, doi: 10.1016/j.micpro.2022.104693.

[5] K. Myers and E. L. Secco, "A low-cost embedded computer vision system for the classification of recyclable objects," in *Intelligent Learning for Computer Vision* (Lecture Notes on Data Engineering and Communications Technologies), vol. 61. Singapore: Springer, 2021, pp. 11–30, doi: 10.1007/978-981-33-4582-9_2.

[6] Q. Liu and H. Sang, "Design of real-time pedestrian trajectory prediction system based on Jetson xavier," *Frontiers Comput. Intell. Syst.*, vol. 4, no. 3, pp. 109–113, Jul. 2023.

[7] M. A. Nuño-Maganda, I. A. Dávila-Rodríguez, Y. Hernández-Mier, J. H. Barrón-Zambrano, J. C. Elizondo-Leal, A. Díaz-Manriquez, and S. Polanco-Martagón, "Real-time embedded vision system for online monitoring and sorting of citrus fruits," *Electronics*, vol. 12, no. 18, p. 3891, Sep. 2023, doi: 10.3390/electronics12183891.

[8] X. Luo, L. Feng, H. Xun, Y. Zhang, Y. Li, and L. Yin, "*Rinegan*: A scalable image processing architecture for large scale surveillance applications," *Frontiers Neurorobotics*, vol. 15, Aug. 2021, doi: 10.3389/fnbot.2021.648101.

[9] E. T. Silva, F. Sampaio, L. C. da Silva, D. S. Medeiros, and G. P. Correia, "A method for embedding a computer vision application into a wearable device," *Microprocessors Microsyst.*, vol. 76, Jul. 2020, Art. no. 103086, doi: 10.1016/j.micpro.2020.103086.

[10] A. Mhalla, T. Chateau, S. Gazzah, and N. E. B. Amara, "An embedded computer-vision system for multi-object detection in traffic surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4006–4018, Nov. 2019, doi: 10.1109/TITS.2018.2876614.

[11] P. Vora and S. Shrestha, "Detecting diabetic retinopathy using embedded computer vision," *Appl. Sci.*, vol. 10, no. 20, p. 7274, Oct. 2020, doi: 10.3390/app10207274.

[12] Y. Zhu, Y. Tan, Y. Hua, M. Wang, G. Zhang, and J. Zhang, "Feature selection and performance evaluation of support vector machine (SVM)-based classifier for differentiating benign and malignant pulmonary nodules by computed tomography," *J. Digit. Imag.*, vol. 23, no. 1, pp. 51–65, Feb. 2010, doi: 10.1007/s10278-009-9185-9.

[13] A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (SVM) in LibSVM," *Int. J. Comput. Appl.*, vol. 128, no. 3, pp. 28–34, Oct. 2015, doi: 10.5120/ijca2015906480.

[14] S. Tulyakov, "Real-time face alignment with a 3D constrained local model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3061–3068.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[16] A. Sangiovanni-Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *IEEE Design Test Comput.*, vol. 18, no. 6, pp. 23–33, 2001.

[17] A. H. Berger, *Embedded Systems Design: An Introduction to Processes, Tools, and Techniques*. New York, NY, USA: Taylor & Francis, 2002. [Online]. Available: https://books.google.dz/books?id=3vY35UkvXrAC

[18] H. Ghasemzadeh and R. Jafari, "Energy-efficient signal processing in embedded systems," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 4134–4142, Feb. 2009.

[19] W. Sun and V. Webster-Wood, "An integrated computer vision system for real-time monitoring and control of long-fiber embedded hydrogel 3D printing," *Mater. Today, Proc.*, vol. 70, pp. 376–381, Jan. 2022, doi: 10.1016/j.matpr.2022.09.272.

[20] J. Park and A. J. Choi, "Vision-based in-flight collision avoidance control based on background subtraction using embedded system," *Sensors*, vol. 23, no. 14, p. 6297, Jul. 2023, doi: 10.3390/s23146297.

[21] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973, doi: 10.1109/TSMC.1973.4309314.

[22] L.-K. Soh and C. Tsatsoulis, "Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 2, pp. 780–795, Mar. 1999.

[23] H. Wang, X.-H. Guo, Z.-W. Jia, H.-K. Li, Z.-G. Liang, K.-C. Li, and Q. He, "Multilevel binomial logistic prediction model for malignant pulmonary nodules based on texture features of CT image," *Eur. J. Radiol.*, vol. 74, no. 1, pp. 124–129, Apr. 2010, doi: 10.1016/j.ejrad.2009.01.024.

[24] J. Edmonds, "Time complexity," in *How to Think About Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2012, pp. 366–373, doi: 10.1017/cbo9780511808241.025.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed., Cambridge, MA, USA: MIT Press, 2009.

[26] E. C. Blessie and E. Karthikeyan, "Sigmis: A feature selection algorithm using correlation based method," *J. Algorithms Comput. Technol.*, vol. 6, no. 3, pp. 385–394, Sep. 2012.

[27] X. Zhang, S. Li, J. Yang, Y. Wang, Z. Huang, and J. Zhang, "Tactile perception object recognition based on an improved support vector machine," *Micromachines*, vol. 13, no. 9, p. 1538, Sep. 2022, doi: 10.3390/mi13091538.

[28] S. Alhassan, D. G. Abdul-Salaam, A. Micheal, Y. M. Missah, D. E. D. Ganaa, and A. S. Shirazu, "CFS-AE: Correlation-based feature selection and autoencoder for improved intrusion detection system performance," *J. Internet Services Inf. Secur.*, vol. 14, no. 1, pp. 104–120, Mar. 2024, doi: 10.58346/jisis.2024.i1.007.

[29] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[30] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[31] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *Science*, vol. 334, no. 6062, pp. 1518–1524, Dec. 2011.

[32] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. Neural Netw.*, vol. 5, no. 4, pp. 537–550, Jul. 1994.

[33] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, Dec. 2004.

[34] R.-J. Palma-Mendoza, L. de-Marcos, D. Rodriguez, and A. Alonso-Betanzos, "Distributed correlation-based feature selection in spark," *Inf. Sci.*, vol. 496, pp. 287–299, Sep. 2019, doi: 10.1016/j.ins.2018.10.052.

[35] K. Balasubramanian and A. N. P., "Correlation-based feature selection using bio-inspired algorithms and optimized KELM classifier for glaucoma diagnosis," *Appl. Soft Comput.*, vol. 128, Oct. 2022, Art. no. 109432, doi: 10.1016/j.asoc.2022.109432.

[36] I. Jain, V. K. Jain, and R. Jain, "Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification," *Appl. Soft Comput.*, vol. 62, pp. 203–215, Jan. 2018, doi: 10.1016/j.asoc.2017.09.038.

[37] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[38] F. Manzouri, S. Heller, M. Dümpelmann, P. Woias, and A. Schulze-Bonhage, "A comparison of machine learning classifiers for energy-efficient implementation of seizure detection," *Frontiers Syst. Neurosci.*, vol. 12, Sep. 2018, doi: 10.3389/fnsys.2018.00043.

[39] Y. Shao and R. S. Lunetta, "Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points," *ISPRS J. Photogramm. Remote Sens.*, vol. 70, pp. 78–87, Jun. 2012, doi: 10.1016/j.isprsjprs.2012.04.001.

[40] D. S. Medeiros, 2020, "Crosswalk-dataset," *Kaggle*, doi: 10.34740/KAGGLE/DSV/1362295.

[41] D. Molloy, *Exploring Raspberry Pi: Interfacing to the Real World With Embedded Linux*. Hoboken, NJ, USA: Wiley, 2016.

[42] E. Upton and G. Halfacree, *Raspberry Pi User Guide*. Hoboken, NJ, USA: Wiley, 2016.

[43] M. Richardson and S. Wallace, *Getting Started With Raspberry Pi*. Sebastopol, CA, USA: O'Reilly Media, 2012.

[44] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[45] R. Liu, E. Liu, J. Yang, M. Li, and F. Wang, "Optimizing the hyper-parameters for SVM by combining evolution strategies with a grid search," in *Intelligent Control and Automation*. Berlin, Germany: Springer, 2006, pp. 344–353, doi: 10.1007/978-3-540-37256-1_87.

[46] C. Cortes and V. Vapnik, *Support-Vector Networks*, vol. 20. Cham, Switzerland: Springer, 1995.

[47] A. L. H. Martínez, S. Khursheed, T. Alnuayri, and D. Rossi, "Online remaining useful lifetime prediction using support vector regression," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 3, pp. 1546–1557, Jul. 2022, doi: 10.1109/TETC.2021.3106252.

[48] T. Alnuayri, A. L. H. Martínez, S. Khursheed, and D. Rossi, "A support vector regression based machine learning method for on-chip aging estimation," in *Proc. 4th Int. Conf. Comput. Inf. Sci. (ICCIS)*, Nov. 2021, pp. 1–6, doi: 10.1109/ICCIS54243.2021.9676376.

[49] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 273–324, Dec. 1997.

[50] M. Mahmoud, "Efficient feature selection for wearable health monitoring systems using genetic algorithms," *IEEE J. Biomed. Health Inf.*, vol. 23, no. 6, pp. 2307–2318, Feb. 2019.

[51] E. Akleman, "Deep learning," *Nature*, vol. 53, no. 9, p. 17, Sep. 2020.
[52] A. Chowdhery, J. Liu, and M. M. Ghandi, "Exploiting computation-accuracy trade-offs in embedded devices for deep learning," *ACM SIGOPS Operating Syst. Rev.*, vol. 51, no. 2, pp. 11–22, 2017.
[53] P. Grover and A. Sahai, "Shannon meets Tesla: Wireless information and power transfer," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 4553–4569, Jul. 2014.

RAMZI KHELIFI received the Applied University Studies degree in computer science and the License degree in information technologies from Larbi Ben M'Hidi University, Oum El Bouaghi, Algeria, in 2004 and 2015, respectively, and the master's degree in computer vision, in 2017. He is currently a member of the Research Laboratory on Computer Sciences Complex Systems (ReLa(CS)2). His research interests include artificial vision, image processing, and the development of algorithms for systems with limited resources.

BRAHIM NINI is a Professor with the Department of Mathematics and Computer Science, Larbi Ben M'hidi University, Oum El-Bouaghi, Algeria. He holds a professor title in computer science (image processing), in 2018. From 2013 to 2016, he was the Head of ReLa(CS)2 Laboratory. He is currently the Head of IMALEX Team within the ReLa(CS)2 Laboratory. His main areas of interests include image and video processing, augmented reality, and artificial vision. He has published more than 30 works touching at all of these domains. During this period, he was part of five national research projects.

MOHAMED BERKANE received the Engineer degree in computer science from Constantine University, Algeria, in 1992, the Magister degree in artificial intelligence from Batna University, Algeria, in 1998, and the Ph.D. degree in artificial vision from INSA Lyon, France, in 2010. He is currently an Assistant Professor with University of Oum El Bouaghi. He studies the motion estimation and optical flow in image sequences using neuronal approaches. His research focuses on the field of image processing. As a second line of research, he works on emotion recognition, and interested in the development of rapid and efficient approaches dedicated to embedded systems.