People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Oum El Bouaghi
Faculty of : of Exact Sciences, Nature Sciences and Life

Thesis

Presented to obtain

3rd Cycle Doctorate

Branch: Computer Science

Specialty: Networks and Distributed Systems

Title :

Adaptability of computer vision and image processing algorithms to resource-constrained systems

Presented by :

Ramzi KHELIFI

Publicly defended on 20/02/2025 in front of the following committee members:

N°	first and last name	Grade	University	Quality
01	**BOUTEKKOUK Fateh**	Prof.	University of Oum El Bouaghi	President
02	**NINI Brahim**	Prof.	University of Oum El Bouaghi	Supervisor
03	**AMIRAT Abdelkrim**	Prof.	University of Souk Ahras	Examiner
04	**HEMAM Sofiane Mounine**	Prof.	University of Khenchela	Examiner
05	**ZERTAL Soumia**	MCA	University of Oum El Bouaghi	Examiner
06	**BELHOUCHETTE Kenza**	MCA	University of Oum El Bouaghi	Examiner

الحمد لله رب العالمين

الحمد لله الذي بنعمته تتم الصالحات، وبتوفيقه تسير الخطوات. إحمده سبحانه على ما إنعم به على مَن علم وتوفيق بإتمام هذه الأطروحة

الإهداء

أُهدي هذا العمل إلى والدتي الحبيبة، لدعمها اللامحدود، وتشجيعها الدائم، وحبها الغامر، ودعواتها الصادقة التي كانت و لاتزال مصدر قوتـي وإلهامي المسـتمر. لقد شكلـت تضحياتك وحكمتك وإيمانك بي ما أنا عليه اليوم، هذا الإنجاز هو إنجازك بقدر ما هو إنجازي.

Dedication

This thesis is lovingly dedicated to my mother, whose unwavering support, endless encouragement, boundless love, and heartfelt prayers have been my constant source of strength and inspiration. Your sacrifices, wisdom, and belief in me have shaped who I am today. This achievement is as much yours as it is mine.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Brahim Nini, for his invaluable guidance, unwavering support, and insightful feedback throughout the development of this thesis. His expertise and encouragement have been instrumental in shaping this work.

I am profoundly thankful to the members of my Research Laboratory on Computer Science's Complex Systems (ReLa(CS)²) and my teachers, especially Abdelhabib BOUROUIS and Toufik MARIR, for their collaboration, constructive discussions, and constant motivation. Your contributions have enriched my academic journey and fueled my determination to achieve excellence.

A special thanks goes to my family for their unconditional love, patience, and encouragement. To my wife, your endless support, sacrifices, and belief in me have been my anchor throughout this journey. To my children, your smiles, laughter, and understanding have been a source of inspiration and strength, reminding me of the importance of perseverance. To my mother, your prayers, sacrifices, and belief in me have been the driving force behind all my achievements. To my father, your wisdom, guidance, and steadfast support have been my foundation, shaping the values that have carried me through this journey.

Lastly, I extend my heartfelt appreciation to my colleagues, mentors, and peers who have provided inspiration, constructive feedback, or simply a kind word during challenging times. To the administrative staff, including the Dean Ousseif Taki Elddine, the Vice Deanship in Charge of Post-Graduation Ziar Aissaoui, and my dearest friend Baaloul Amir, thank you for your invaluable assistance and support.

To everyone who has contributed to this journey in any way, no matter how small, thank you.

This accomplishment would not have been possible without all of you.

Abstract

Adapting computer vision and image processing algorithms to resource-constrained systems is essential for enabling their deployment in diverse, real-world applications. Resource-constrained environments, such as embedded systems (ES), often face significant limitations in computational power, memory capacity, and energy availability. Extending hardware to address these limitations is often impractical due to cost and scalability concerns, making adaptability a more feasible and efficient solution. By adapting these algorithms, systems can overcome underlying limitations in physical resources, and power, enabling them to execute tasks effectively.

Various domains are increasingly adopting embedded computer vision systems, which play 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 dissertation 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), optimising their performance under the constraints of real-time limited-resource environments.

Keywords: Adaptability of algorithms, embedded computer vision, limited resource systems, machine learning, pattern classification, real-time image processing.

Résumé :

L'adaptation des algorithmes de vision par ordinateur et de traitement d'images aux systèmes à ressources limitées est essentielle pour permettre leur déploiement dans diverses applications du monde réel. Les environnements à ressources limitées, tels que les systèmes embarqués, sont souvent confrontés à des limitations importantes en termes de puissance de calcul, de capacité mémoire et de disponibilité énergétique. Étendre le matériel pour surmonter ces limitations est souvent impraticable en raison des coûts et des défis de scalabilité, rendant l'adaptabilité une solution plus pratique, viable et efficace. En adaptant ces algorithmes, les systèmes peuvent contourner les limitations sous-jacentes en ressources physiques et en énergie, leur permettant d'exécuter des tâches de manière efficace.

Les systèmes de vision embarqués sont de plus en plus adoptés dans divers domaines, jouant un rôle clé dans la mise en œuvre de technologies avancées telles que les véhicules autonomes et l'automatisation industrielle. Leur rapport coût-efficacité, leur taille compacte et leur portabilité les rendent particulièrement favorables à des implémentations et des opérations variées. Dans les scénarios en temps réel, ces systèmes doivent traiter les données visuelles avec une latence minimale, une exigence fondamentale pour une prise de décision immédiate. Cependant, ces solutions continuent de faire face à des défis importants liés à l'efficacité computationnelle, à l'utilisation de la mémoire et à la précision. Cette thèse aborde ces défis en améliorant les méthodologies de classification, notamment dans l'extraction de caractéristiques à l'aide de la matrice de cooccurrence des niveaux de gris (Gray Level Co-occurrence Matrix, GLCM) et les classificateurs à vecteurs de support (Support Vector Machine, SVM).

Pour maintenir un niveau élevé de précision tout en préservant les performances, un ensemble de caractéristiques réduit est sélectionné à l'issue d'une analyse approfondie de la complexité et ensuite est affiné par une méthode de sélection de caractéristiques basée sur la corrélation (Correlation-based Feature Selection, CFS). La méthode proposée atteint une précision globale de classification de 84,76% avec un ensemble de caractéristiques réduit de 79,2%, entraînant une diminution de 72,45% du temps de traitement, une réduction de 50% des besoins en stockage et une diminution allant jusqu'à 77,8% de la demande en mémoire pendant la phase de prédiction. Ces améliorations démontrent l'efficacité de l'approche proposée pour renforcer l'adaptabilité et les capacités des systèmes de vision embarqués (Embedded Vision

Systems, EVS), en optimisant leurs performances sous les contraintes des environnements à ressources limitées soumis aux exigences du temps réel.

Mots-clés : Adaptabilité des algorithmes, vision par ordinateur embarquée, systèmes à ressources limitées, apprentissage de la machine, classification des modèles, traitement d'images en temps réel.

الملخص:

يعد تكييف خوارزميات الرؤية الحاسوبية وخوارزميات معالجة الصور للتشغيل على الأنظمة محدودة الموارد أمرًا ضروريًا لتمكين نشرها في تطبيقات متنوعة في العالم الحقيقي. غالبًا ما تواجه البيئات المحدودة الموارد مثل الأنظمة المدمجة قيودًا كبيرة في القدرة الحاسوبية وسعة الذاكرة وتوافر الطاقة. وغالبًا ما يكون توسيع نطاق الأجهزة لمعالجة هذه القيود غير عملي بسبب مخاوف تتعلق بالتكلفة وقابلية التوسع، مما يجعل خيار التكييف حلاً أكثر جدوى وفعالية. من خلال تكييف هذه الخوارزميات، يمكن للأنظمة التغلب على القيود المتأصلة في الموارد المادية والطاقة، مما يمكنها من تنفيذ المهام بفعالية.

يتم اعتماد أنظمة الرؤية الحاسوبية المدمجة بشكل متزايد في مختلف المجالات، حيث تلعب دوراً محورياً في تمكين التقنيات المتقدمة مثل المركبات ذاتية القيادة والتشغيل الالي الصناعي. كما أن تكلفتها الاقتصادية وحجمها الصغير وقابليتها للنقل تجعلها مناسبة بشكل خاص للتطبيقات والعمليات المتنوعة. في سيناريوهات الوقت الحقيقي، يجب أن تعالج هذه الأنظمة البيانات المرئية بأقل قدر من التأخير في الوصول إلى البيانات، وهو أمر بالغ الأهمية لاتخاذ القرارات الفورية. ومع ذلك، لا تزال هذه الحلول تواجه تحديات كبيرة تتعلق بالكفاءة الحسابية واستخدام الذاكرة والدقة. تعالج الأطروحة هذه التحديات من خلال تعزيز منهجيات التصنيف، وتحديداً في استخراج ميزات مصفوفة التكرار المشترك للمستوى الرمادي (GLCM) ومصنفات آلة دعم المتجهات (SVM).

من أجل الابقاء على مستوى عالٍ من الدقة مع الحفاظ على الأداء، يتم اختيار مجموعة ميزات أصغر بعد إجراء تحليل شامل للتعقيد ويتم تنقيحها بشكل أكبر من خلال اختيار الميزات القائمة على الارتباط (CFS) وتحقق الطريقة المقترحة دقة تصنيف إجمالية تبلغ 84.76% مع مجموعة ميزات مخفضة بنسبة 79.2%، مما يؤدي إلى انخفاض بنسبة 72.45% في وقت المعالجة وانخفاض بنسبة 50% في متطلبات التخزين وانخفاض يصل إلى 77.8% في الطلب على الذاكرة أثناء التنبؤ. تُظهر هذه التحسينات فعالية المنهجية المقترحة في تحسين قدرة أنظمة الرؤية المدمجة (EVS) على التكيف وتحسين أدائها في ظل قيود بيئات الوقت الحقيقي محدودة الموارد.

الكلمات الدلالية: تكييف الخوارزميات، الرؤية الحاسوبية المدمجة، الأنظمة محدودة الموارد، التعلم الآلي، تصنيف الأنماط، معالجة الصور في الوقت الحقيقي.

Table Of Contents

List Of Figures

List Of Tables

Notations

Acronym	Term
ES	Embedded Systems
GLCM	Gray Level Co-Occurrence Matrix
SVM	Support Vector Machine
CFS	Correlation-Based Feature Selection
AI	Artificial Intelligence
CV	Computer Vision
SoC	System-On-Chip
SBCs	Single Board Computers
ROIs	Region Of Interest
IG	Information Gain
FCBF	Fast Correlation-Based Filter
mRMR	Minimum Redundancy Maximum Relevance
RFE	Recursive Feature Elimination
SFS	Sequential Forward Selection
SBE	Sequential Backward Elimination
Weka	Wrapper Subset Evaluation
SVM-RFE	Recursive Feature Elimination For SVM
FS-P	Feature Selection Perceptron
kNN	K-Nearest Neighbours
CNNs	Convolutional Neural Networks
GMM	Gaussian Mixture Models
NNs	Neural Networks
ReLU	Rectified Linear Unit
UAVs	Unmanned Aerial Vehicles
SV	Support Vectors
DF	Decision Functions
SMC	Sequential Monte Carlo
R-CNN	Region-Based Convolutional Neural Networks
MF R-CNN	Modified Faster R-CNN
DCNN	Deep Convolutional Neural Network

GSM	Global System for Mobile Communications
GPS	Global Positioning System
Real-Time EVS	Real-Time Embedded Vision System
FPGA	Field Programmable Gate Array
DT	Decision Tree
FPS	Frames Per Second
IoT	Internet Of Things
LFE-FRESH	Long-Fiber Embedded Fresh
ELAC	Electrochemically Aligned Collagen
EVS	Embedded Vision Systems
Ng	Number Of Grey
CM	Co-Occurrence Matrix
ASM	Angular Second Moment
IDM	Inverse Difference Moment
IMC I	Information Measures of Correlation I
IMC II	Information Measures of Correlation II
MCC	Maximal Correlation Coefficient
OvR	One-Versus-The-Rest
OvO	One-Vs-One
WTA	Winner-Takes-All
SVMs	Support Vector Machines
RBF	Radial Basis Function
CPU	Central Processing Unit
RAM	Random Access Memory
Ms	Millisecond
MiB	Mebibytes
kb	Kilobytes
TinyML	Tiny Machine Learning
PCA	Principal Component Analysis
RF	Random Forests
DL	Deep Learning Models

Chapter 1

General Introduction

1.1 Introduction

The rapid advancement of technology has led to the integration of embedded vision systems (EVS) into a wide range of applications. They are fundamentally transforming industries such as automotive, healthcare, security, and manufacturing. These systems combine imaging sensors with embedded processors to analyse visual data. It enables functionalities like autonomous navigation, intelligent surveillance, and automated quality control.

One of the most prominent examples is the development of autonomous vehicles, which rely heavily on embedded vision systems to interpret their surroundings and make split-second decisions without human intervention. Similarly, in industrial automation, EVS are employed by tasks like object recognition and defect detection, enhancing efficiency and precision while reducing reliance on manual labour.

Despite their ubiquitous presence in many fields, embedded vision systems face significant challenges that limit their performance and wider adoption. Among these challenges are constraints related to computational efficiency, and memory usage. These last have a negative impact on the accuracy and responsiveness in limited-resource environments. Common computer vision algorithms and machine learning models often require extensive processing power and memory, which conflicts with the compact size and limited power of embedded systems (ES).

Furthermore, the necessity for real-time processing adds another layer of complexity. In applications where immediate decision-making is critical, such as collision avoidance in autonomous vehicles or real-time monitoring in healthcare devices. Latency caused by computational delays can lead to serious repercussions.

In order to address these issues, some works have been directed toward optimising both hardware and software components of EVS. On the hardware side, advancements include the development of more powerful yet energy-efficient processors and specialised accelerators. However, these advancements often result in increased costs, which can limit the accessibility to such solutions. On the software side, efforts focus on algorithm optimisation, efficient coding practices, and the adoption of lightweight machine learning models, providing a low-cost alternative solution to expensive hardware developments.

Hence, in the context of software optimisation, enhancing classification methodologies plays a pivotal role. Techniques such as refining feature extraction methods and employing efficient classifiers can significantly reduce computational load. For instance, optimising the Gray-Level

Co-occurrence Matrix (GLCM) for feature extraction and utilising Support Vector Machines (SVM) with advanced feature selection methods can lead to models more efficient to EVS without substantially compromising accuracy.

This thesis contributes to this field by developing an enhanced classification system designed for real-time embedded vision applications. Through a comprehensive complexity analysis combined with correlation-based feature selection, this approach emphasises the adaptability. It ensures that the classification system is optimised to function efficiently within the constraints of resource-limited environments. This makes it well-suited for deployment in real-time scenarios were maintaining both performance and efficiency is critical.

1.2 Main Thesis

Embedded vision systems are witnessing growing adoption across critical domains, including autonomous vehicles, industrial automation, and healthcare. While these systems rely on compact embedded processors to analyse visual data, they target challenging applications that demand immediate decision-making, thus their deployment often faces significant obstacles.

Limitations in computational power, memory capacity, and energy efficiency delay real-world implementation, particularly in resource-constrained environments. These challenges become particularly significant when maintaining high accuracy is essential, while strict performance requirements simultaneously add further complexity. Established approaches to addressing these challenges often focus on developing hardware solutions, such as energy-efficient processors or specialised accelerators speeding up computational tasks.

While effective, such solutions can increase costs and make it difficult to scale systems across a wide range of applications, particularly those requiring low-cost and accessible solutions. The focus shifts attention to software-based approaches, where algorithm optimisation, efficient feature extraction, and the use of lightweight machine learning models emerge as promising alternatives and play a fundamental role. Within this context, adaptability becomes a key principle referring to the design of systems capable of functioning effectively under varying resource constraints while ensuring robustness and precision.

This thesis contributes to this field by presenting an enhanced classification system specifically refined for real-time embedded vision applications. The proposed system is designed to optimise resource utilisation without compromising accuracy, addressing the critical need for efficiency in the specific context of embedded environments.

Key advancements include conducting a comprehensive complexity analysis to minimise the feature set. Features derived from algorithms with low computational complexity are prioritised. Finally, the process is refined using correlation-based feature selection. These steps significantly reduce computational requirements, memory usage, and model size, enabling the system to adapt to the demands of resource-constrained and real-time scenarios.

Through these contributions, this dissertation establishes a practical approach for enhancing the performance and adaptability of computer vision and image processing algorithms for embedded systems. It advances current methodologies and also lays the groundwork for future exploration in lightweight and efficient algorithmic designs for embedded environments.

1.3 Methodology

The proposed methodology optimises embedded vision systems for resource-constrained environments, preserving high accuracy and maintaining effectiveness while minimising resource demands. It starts with extracting features using the Gray-Level Co-occurrence Matrix (GLCM) to identify an initial set of 24 features (12–14). It systematically refines these features through a complexity-based analysis (15) and then applies correlation-based feature selection, creating a streamlined feature set. This approach ensures computational efficiency and accuracy while enabling the system to adapt to application demands and real-time scenarios.

It employs a structured evaluation process, dividing the dataset (16) into training and testing subsets. The Support Vector Machine (SVM) classifier, selected for its balance between precision and efficiency, fine-tuned through an exhaustive grid search (17). This process optimises the model's hyperparameters, ensuring consistent and reliable performance. Through iterative refinement, it identifies the best settings to maximise model efficiency, enhancing adaptability and enabling the system to perform effectively across diverse embedded environments.

To validate the system's adaptability and practicality, we deployed it on a Raspberry Pi platform. We rigorously assessed performance metrics such as accuracy, processing time, and memory usage to confirm the system's efficiency under resource constraints. The results demonstrate that this methodology achieves successfully its aim. It enables embedded vision systems to deliver high performance while remaining flexible and efficient in real-time and resource-limited conditions.

1.4 Summary of related publications

This section outlines the published work associated to this thesis, highlighting its contributions to the field of embedded vision systems by advancing embedding methodologies.

Journal Paper

- R. Khelifi, B. Nini, and M. Berkane. **"Enhanced Classification System for Real-Time Embedded Vision Applications."** *IEEE Access, vol. 12, pp. 162311-162326, 2024.* doi: 10.1109/ACCESS.2024.3489476.

This paper introduces a novel classification system specifically tailored for real-time embedded vision applications. Key contributions include feature reduction through comprehensive complexity analysis, the application of correlation-based feature selection, and the development of a lightweight Support Vector Machine (SVM) classifier. These advancements significantly improve computational efficiency, reduce memory usage, and enhance accuracy, making the system suitable for deployment in resource-constrained environments. This thesis builds its comprehensive research on the methodologies and findings discussed in the paper.

1.5 Main Contributions

This thesis presents significant contributions to advancing embedded vision systems (EVS), with a focus on optimising performance for adaptability to resource-constrained environments. These contributions are organised into key areas:

1.5.1 Feature Reduction and Optimisation

A comprehensive complexity analysis was conducted to evaluate and prioritise low-complexity features, ensuring computational efficiency without compromising accuracy. We further refined this by applying correlation-based feature selection, which led to a 79.2% reduction in the feature set. This reduction directly improves system performance by minimising processing time and memory requirements.

1.5.2 Enhanced Classification Models

The development of a lightweight Support Vector Machine (SVM) classifier achieves a balance between performance and resource utilisation. This model reduces storage requirements by 50% and enables real-time operation in environments with limited computational and memory resources.

1.5.3 Adaptability in Resource-Constrained Environments

By focusing on low-complexity features and efficient classifiers, the proposed approach ensures adaptability, making the system suitable for diverse real-world applications. These advancements demonstrate the potential for deploying high-performance embedded vision systems in scenarios where standard methods are infeasible.

1.5.4 Real-Time Deployment Efficiency

The proposed system achieves an overall classification accuracy of 84.76%. It also reduces processing time by 72.45% and memory usage up to 77.8% during prediction. These improvements highlight the effectiveness of the approach in meeting the dual demands of real-time processing and resource constraints.

These contributions constitute a substantial progression in the state of the art, addressing serious challenges and advancing the field of embedded vision systems. They represent a cohesive effort to overcome the practical limitations of EVS. Therefore, demonstrate the feasibility of deploying resource-optimised, high-accuracy vision systems in real-world applications without the need for expensive hardware upgrades.

1.6 Organisation of the thesis

This thesis is organised into five chapters including the introductory one. Chapter [2], [3] , [4] and [5] each addressing a critical aspect of the research and its contribution to the field of embedded vision systems. It ends with a final conclusion and future works:

Chapter 2: Background on Vision Systems

This chapter provides a comprehensive background on embedded vision systems, discussing their fundamental principles, key components, and technological advancements. It establishes the theoretical foundation necessary for understanding the challenges and solutions explored in the subsequent chapters.

Chapter 3: Literature Review

This chapter reviews the existing research in embedded vision systems. It focuses on feature extraction techniques, classification methodologies, and optimisation strategies. The discussion identifies gaps in the current state-of-the-art and highlights the need for adaptable and efficient solutions in resource-constrained environments.

Chapter 4: Methodology

This chapter details the methodology adopted in this research. It includes feature extraction using Gray-Level Co-occurrence Matrix (GLCM), feature selection through complexity analysis and correlation-based techniques, and the development of an optimised Support Vector Machine (SVM) classifier. It also describes the experimental setup and evaluation framework used to validate the proposed system.

Chapter 5: Performance Assessment And Validation

This chapter presents the results of the experiments, including performance metrics such as accuracy, processing time, and memory usage. It provides a detailed discussion on the effectiveness of the proposed methodology, comparing it to existing approaches and emphasising its adaptability and efficiency in real-world applications.

Conclusion and Future Works

It summarises the key findings and contributions of the thesis. It also discusses the limitations of the current work. Additionally, it proposes directions for future research to further advance the field of embedded vision systems.

Chapter 2

Background on Vision Systems

2.1 Introduction

The integration of computer vision into embedded systems has transformed the way machines interpret and process visual data. In contrast to conventional vision systems that rely on high-performance computing infrastructure, embedded vision systems (EVS) are designed to operate within constrained resources. This evolution has enabled a wide range of applications, from real-time decision-making in autonomous systems to industrial automation and healthcare diagnostics.

Embedded vision relies on specialized hardware, optimized software, and advanced machine learning techniques to enhance processing efficiency. However, designing such systems presents significant challenges, including computational limitations, memory constraints, and energy efficiency requirements.

This chapter explores embedded vision systems (EVS) within the context of resource-limited environments. It outlines their transformative impact across various industries and the critical challenges they face. It also highlights the advancements driving their development and the importance of adaptability in ensuring their efficiency and relevance to modern applications.

The chapter also provides a foundation for the thesis, which focuses on developing an enhanced classification system optimised for resource-constrained environments. It introduces key definitions for ease of understanding the technical concepts and methodologies discussed throughout the thesis. Additionally, it discusses the opportunities and constraints associated with deploying EVS in real-world applications, providing the necessary context for the research. It emphasises the need for innovative approaches presented in subsequent chapters.

2.2 Computer vision

Computer vision is a computer science and engineering field focusing on enabling machines to interpret and understand visual data from the real world. It aims to emulate human vision capabilities. To achieve this, it constructs symbolic representations of the world based on images (1). The primary goal is to develop computational models that can visually perceive and analyse environments; effectively and digitally replicating certain aspects of the human visual system.

Computer vision originated in the 1960s as a branch of artificial intelligence (AI). Early research focused on simple tasks like edge detection and object recognition, which laid the

foundation of understanding images. Key contributions came from researchers like Larry Roberts, who applied geometric methods to image processing (2), and David Marr, who founded the theory of vision as an information-processing task. He developed a framework for understanding how the brain interprets images in stages (3), which is considered a fundamental contribution to cognitive science and artificial intelligence. Over time, computer vision developed through advancements in machine learning, leading to today's sophisticated image recognition and interpretation systems.

It is necessary to clarify the terms machine vision and Computer Vision (CV), as their scopes and applications have evolved significantly over time. Initially, CV focused on the study of visual perception and the design of software algorithms to interpret images and videos, with minimal emphasis on hardware. In contrast, machine vision was more engineering-oriented, covering both software and the hardware required for image acquisition, often for industrial applications (4).

However, with advancements in computer technology and a deeper understanding of visual systems, the distinction between the two fields has faded. Today, the terms are often used interchangeably. Both involve the integration of hardware and software to process visual information. As a result, CV has become a widely recognised and commonly used term in this context.

2.3 Vision Processing Algorithms

Vision algorithms are computationally intensive. This is because they rely on complex mathematical models and extensive data processing. Such demands are particularly evident in tasks like object detection, feature extraction, and image classification. These algorithms are characterised by their diversity, which arises from the absence of universally accepted standards (5).

The absence of standardised guidelines in vision algorithms results in diverse approaches to algorithm design and implementation. This introduces complexities in the development process (6,7). Designers face the challenge of selecting from a vast range of algorithms, customising them for specific tasks, and integrating these solutions into larger systems, especially within resource-constrained environments. This variability complicates optimisation for interoperability, performance, and efficiency compared to standardised fields.

According to a study published in the "Handbook of Quality System, Accreditation and Conformity Assessment" (8), standardisation plays an important role in driving innovation. It allows new technologies to spread more quickly and be adopted widely.

Moreover, vision algorithms are highly dynamic. They continuously evolve due to advancements in machine learning and computer vision techniques. This rapid progression leads to the algorithmic approaches with varying computational requirements and implementation complexities. The need for low-cost, energy-efficient solutions further constrains the deployment of these algorithms in industrial automation systems. This necessitates meticulous optimisation of both algorithmic structures and hardware designs to balance performance with the limitations of resource-constrained environments.

A review on resource-constrained embedded vision systems highlights the remarkable growth of low-cost ES and their use in robotics applications (9). It emphasises the critical task execution by implementing sophisticated control and computer vision algorithms.

Adaptability and efficiency are essential for further advancing computer vision and image processing algorithms in real-world applications. Enhancing these greedy algorithms to perform effectively under severe environmental conditions unlocks the full potential of embedded platforms, paving the way for their faster and broader adoption.

2.4 Typical vision system

Although applications of vision technologies vary, most computer vision systems follow a similar sequence of steps to process and analyse image data. This sequence, known as the vision pipeline, consists of a series of distinct stages that guide the flow of data.

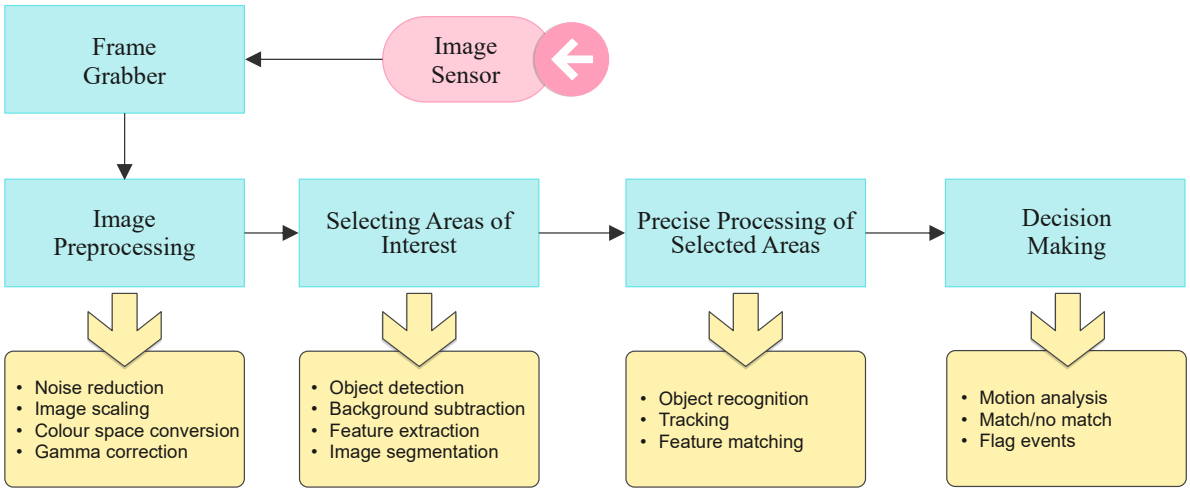


Fig. 2.1 Typical computer vision system pipeline

Fig. 2.1 depicts the structured stages of a typical computer vision system pipeline, illustrating the sequential processes required to analyse and interpret visual data (5). The pipeline begins with data acquisition, where an image sensor captures raw input, and prepares it for processing. This is followed by the image preprocessing stage. It applies operations such as noise reduction, colour space conversion, and gamma correction. This step enhances data quality and ensure compatibility for subsequent analysis.

The next stage involves selecting areas of interest and focusing computational resources on relevant regions. It employs techniques like object detection, background subtraction, and feature extraction. These selected regions undergo precise processing, including object recognition, tracking, and feature matching, to extract meaningful and actionable information.

Finally, the decision-making stage integrates the processed data to perform higher-level tasks, such as motion analysis, pattern matching, and event detection. This pipeline exemplifies the systematic approach employed in computer vision systems.

2.5 Embedded system

An embedded system is a specialised computing device designed to perform a specific and dedicated function within a larger mechanical or electronic system. It integrates tightly coupled hardware and software components, including a processor, memory, and input/output peripherals, to perform its task efficiently and reliably.

Unlike general-purpose systems, ES often operate autonomously with minimal human interaction (10). Operating in constrained environments with limited memory, computing power, and power supply, ES often need to react in real-time to inputs from processes or the environment.

Embedded systems are widely used in a variety of applications. Table 2.1 describes some real-world examples of ES, from digital watches and household appliances to industrial machinery and medical devices. In such as applications, cost and resource efficiency are the most critical considerations in design and manufacturing.

The evolution of ES dates back to the early days of digital computing. Initially, industrial and military applications utilised them. Advancements in microelectronics and software engineering have expanded their use to a wide array of consumer and commercial applications. The development of microcontrollers and system-on-chip (SoC) technologies has further revolutionised the field, enabling more complex and powerful ES (11).

Several emerging trends are shaping the future of ES:

- **Internet of Things (IoT):** The propagation of connected devices that communicate and interact with each other and the cloud.
- **Artificial Intelligence (AI):** Integration of AI and machine learning algorithms to enable smarter and more autonomous systems.
- **Edge Computing:** Processing data closer to the source (at the edge) to reduce latency and improve efficiency.
- **Cybersecurity:** Enhancing the security of ES to protect against cyber threats and vulnerabilities.

Table 2.1 Real-world examples of embedded systems

Example of Embedded System	Description
Electronic Speed Controllers (e.g., for RC vehicles)	ES regulate motor speed and direction in remote-controlled vehicles, enabling precise control and enhanced manoeuvrability.
Hearing Aids	ES in hearing aids amplify sound, reduce noise, and adjust settings dynamically based on user preferences and environmental conditions.
Elevator Control Systems	ES manage elevator operations, including motor control, floor selection, and implementing safety mechanisms.
Digital Audio Players	ES in digital audio players decode audio tracks, manage user interfaces, and control playback features.
Digital Thermometers	Digital thermometers use embedded systems to measure temperature precisely and display real-time readings.
Barcode Scanners	ES in barcode scanners interpret barcode data, decode information, and facilitate inventory management and retail operations.
GPS Navigation Devices	ES in GPS devices process satellite signals, calculate routes, and deliver step-by-step navigation instructions.
Drones	ES in drones control flight operations, stabilise the aircraft, analyse sensor data, and facilitate communication with ground stations.
Digital Cameras	ES in digital cameras handle image processing, adjust camera settings, and support features such as autofocus and image stabilisation.
Smartwatches	ES in smartwatches integrate sensors, processors, and wireless connectivity to monitor health metrics, deliver notifications, and communicate with smartphones.

2.5.1 Characteristics of Embedded Systems

To determine whether a system qualifies as an embedded system, it's essential to recognise its main defining characteristics:

- **Single Function:** An embedded system is engineered to perform a specific, dedicated function repeatedly.
- **Tightly Constrained:** The system operates under strict limitations regarding memory, processing power, and energy consumption.
- **User Interface:** May include interfaces ranging from simple buttons and LEDs to complex touchscreens, depending on the application.

These attributes help distinguish embedded systems from general-purpose systems, enabling their identification across various applications. Table 2.1 provides examples of real-world ES, illustrating their diverse implementations in different domains.

2.5.2 Types of Embedded Devices

Embedded systems can be classified into distinct categories based on their functionality and performance characteristics. This highlights their versatility and adaptability to various applications. Furthermore, illustrates the broad range of tasks embedded systems are designed to perform and the environments in which they operate. Below is an expanded overview of each type.

1. **Real-Time ES**

These systems are designed to process data and perform tasks in strict time constraints, ensuring reliability and safety. They are further classified into:

- **Soft Real-Time Systems:** Systems where timing constraints are important but not critical. Examples include audio systems and multimedia players.
- **Hard Real-Time Systems:** Missing a deadline can result in system failure or catastrophic consequences, For instance, aircraft control systems and medical monitoring devices.

2. **Stand-alone ES**

Stand-alone systems operate independently, performing specific, self-contained tasks without relying on external systems. Examples include microwave ovens, washing machines, and video game consoles.

3. Networked ES

Networked systems are connected to other devices through wired or wireless networks, enabling data sharing and task coordination within larger systems. Examples include Automated Teller Machines (ATMs), facilitating secure financial transactions by connecting to banking networks, home security systems that provide remote monitoring, and card swipe readers.

4. Mobile ES

Mobile systems are portable and compact, integrating sophisticated functionalities within constrained memory and processing capabilities. Examples include digital cameras, mobile phones, and smart watches.

2.6 Embedded computer vision

Computer vision involves using digital processing and intelligent algorithms to extract meaningful information from images or videos. Embedded vision combines computer vision with embedded systems for efficient visual data processing. Therefore, an embedded vision system is any microprocessor-based device with image sensing ability that is not a typical personal computer (5).

This category includes devices like tablets, smartphones, advanced medical diagnostic tools, and robots with image processing and recognition capabilities. In essence, embedded vision refers to machines with limited resources that interpret and understand their environment through visual inputs.

2.7 Advantages of Embedded Vision System Boards

In fact, embedded vision boards cannot entirely replace PC-based vision systems. However, their wide adoption is driven not only by cost-effectiveness but also by several other distinct advantages. These EVS boards are task-oriented, performing specific well-defined tasks rather than general-purpose computing. They are designed to meet specific application needs and provide primary benefits such as:

1. **Lightweight designs:** making them compact and portable.
2. **Lower manufacturing costs:** achieved through streamlined production processes.
3. **Lower energy consumption:** ensuring efficient operation in resource-constrained environments.
4. **Small footprint:** allowing for space-saving designs in compact implementations.

For a comprehensive comparison of widely used Single Board Computers (SBCs) and their technical specifications, Table 2.2 provides detailed insights into their configurations. Unlike PC-based systems, which typically aim for larger image processing tasks, these boards typically build to meet the precise needs of a given application.

The adoption of embedded vision is expected to grow rapidly in the coming years. This growth will revolutionise industries and open new avenues of innovation. It will be driven by ongoing advancements in software adaptability, which enable these systems to perform efficiently under diverse and constrained conditions without extra charges. This thesis witnesses these efforts, aligning with the wider push toward more adaptable and efficient embedded vision systems.

2.8 Conclusion

Embedded vision systems have emerged as a transformative technology, offering intelligent and efficient solutions across various fields. Their ability to integrate visual perception into compact, resource-constrained devices has enabled significant advancements. Despite challenges related to computational efficiency, memory usage, and real-time processing, continuous innovations in hardware design, algorithm optimization, and machine learning techniques are expanding their capabilities.

By leveraging adaptable and efficient computational models, embedded vision systems provide an opportunity to bridge the gap between high-performance vision processing and the constraints of embedded environments. The ongoing evolution of machine learning and feature optimization techniques further enhances their adaptability, allowing these systems to function effectively in real-world scenarios. These advancements contribute to the emergence of more accessible, cost-effective, and scalable vision solutions, making embedded vision a promising avenue for future research and technological innovation.

Table 2.2 Overview of Widely Used Single-Board Computers with Technical Details

SBC	Manufacturer (Country)	Dimensions (mm)	Weight (g)	Price (USD)	Programming Languages	Operating Systems	Description
Raspberry Pi 4	Raspberry Pi Foundation (UK) raspberrypi.com	85.6 × 56.5	46	$35–$75	Python, C/C++, Java	Raspberry Pi OS, Linux	A versatile, popular SBC with up to 8GB RAM, suitable for general computing, education, IoT, and media centres.
BeagleBone Black	BeagleBoard.org (USA) beagleboard.org	86.4 × 53.3	40	$55–$65	Python, C/C++, JavaScript	Debian, Ubuntu	Known for its flexibility, ideal for educational and industrial applications, with extensive I/O options and community support.
Odroid XU4	Hardkernel (South Korea) hardkernel.com	82 × 58	60	$60–$80	Python, C/C++	Ubuntu, Android	A high-performance SBC featuring an octa-core CPU, good for multimedia applications, gaming, and power-hungry tasks.
NVIDIA Jetson Nano	NVIDIA (USA) developer.nvidia.com	100 × 80	140	$99	Python, C/C++, CUDA	Ubuntu (JetPack)	Designed for AI and machine learning projects, supports deep learning and computer vision with a powerful GPU.
ASUS Tinker Board	ASUS (Taiwan) asus.com	85.6 × 56	55	$60–$70	Python, C/C++	TinkerOS, Debian, Android	A powerful alternative to the Raspberry Pi, with superior multimedia capabilities, including 4K support and high-quality audio.
Rock Pi 4	Radxa (China) radxa.com	85 × 54	50	$49–$75	Python, C/C++	Debian, Ubuntu, Android	Similar specs to the Raspberry Pi 4 but with faster eMMC storage options and an expandable M.2 slot.
Banana Pi M5	SinoVoip (China) banana-pi.org	92 × 60	48	$70–$80	Python, C/C++	Android, Linux	A Raspberry Pi-like SBC with quad-core processing, supporting Android and Linux operating systems.
Pine64 RockPro64	Pine64 (USA) pine64.org	90 × 60	54	$60–$80	Python, C/C++	Debian, Ubuntu, Android	Supports 4GB RAM and PCIe expansion, suitable for NAS setups and performance-intensive applications.
Arduino Portenta H7	Arduino (Italy) arduino.cc	66 × 25	12	$99–$120	Arduino (C/C++), MicroPython	Arm Mbed OS, FreeRTOS	A hybrid microcontroller/SBC with dual-core processing, designed for industrial IoT and edge computing applications.
Orange Pi 5	Orange Pi (China) orangepi.org	100 × 60	55	$60–$80	Python, C/C++	Debian, Ubuntu, Android	A cost-effective SBC with up to 8GB RAM, powered by a Rockchip RK3588S processor, suitable for multimedia, IoT projects, and light AI applications.

Chapter 3

Literature Review

3.1 Introduction

Optimising computer vision for resource-constrained environments requires in-depth understanding of the methods and their tunings involved during the vision pipeline. We concentrate on essential elements such as preprocessing, feature extraction, selection, and classification. The main focus is on strategies that enhance computational efficiency, minimise memory usage, and achieve high accuracy. Investigating techniques suitable for systems with limited resources identifies the challenges to the achievement of real-time performances.

We will discuss feature extraction methods, from classical to modern, considering their computational feasibility and practical effectiveness. We present various feature selection techniques that minimise dimensionality while preserving essential system capabilities. The discussion includes several classification models and decision-making procedures, highlighting those that optimise performance while ensuring resource efficiency.

The chapter then concludes by discussing some of the primary challenges that real-time vision systems face, including hardware-software optimisation, energy-efficient algorithms, and model compression. This analysis presents several significant gaps in current methods, such as the need for scalable solutions and effective balances between accuracy and efficiency. It lays the groundwork for the thesis's contributions.

3.2 Vision Classification Systems

Pattern recognition in vision systems emerged as a prominent engineering discipline during the American space program in the 1960s. Initially, it was strongly associated with the examination of digital images transmitted by spacecraft (18). Pattern recognition has later expanded into a wide array of applications in signal processing, and beyond.

Vision classification systems have become an integral component of modern computer vision applications. These systems apply principles from pattern recognition, machine learning and intelligent algorithms to categorise visual data into classes. Integrating a combination of feature extraction and classification techniques (19), vision classification systems effectively process and interpret visual inputs for tasks such as object detection, facial recognition, and medical diagnostics.

3.2.1 Components of a vision classification system

Vision classification systems consist of interconnected components that collaboratively analyse visual inputs and predict the target. Fig. 3.1 visualises the processing pipeline and the steps involved (20). These systems utilise various algorithms and feature extraction approaches to convert raw visual input into significant insights. This enables applications of detection, recognition, and medical imaging (21–24). Each component ensures efficient interpretation of visual information.

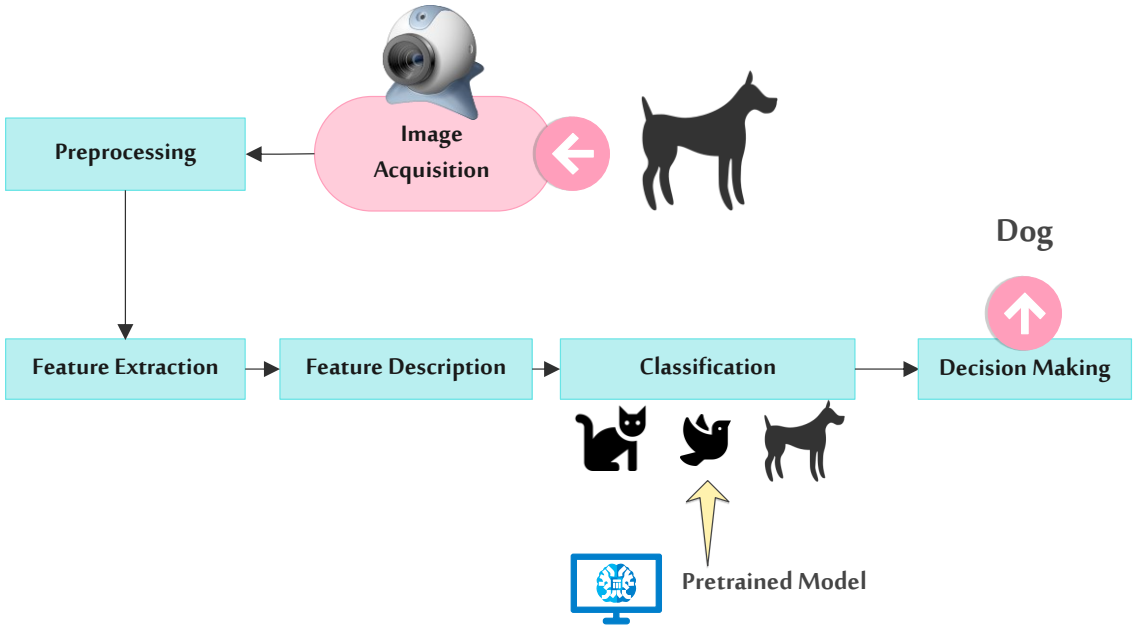


Fig. 3.1 Processing pipeline of a vision classification system.

3.2.1.1 Image Acquisition

Image acquisition is the initial step in the sequence of operations involved in the processing pipeline. It entails capturing visual information using a sensing device, such as a camera or scanner, and converting it into a digital format suitable for further analysis (25). This stage is critical, as the quality of the acquired data directly impacts the effectiveness of subsequent steps, including enhancement, segmentation, and classification.

3.2.1.2 Image Preprocessing

Image preprocessing is the process of converting raw image data into a format that is usable and meaningful, which can then be used for computer vision applications. By removing unwanted distortions and enhancing certain features, preprocessing ensures that the data is in the best shape for further analysis and modelling (26). As the first step in the vision pipeline that refines image quality and consistency, it forms the basis for robust performance in subsequent stages. It achieves this by employing techniques designed to overcome the challenges of raw data variability, preparing it for feature extraction and classification (25). The proper combination of these methods facilitates efficient and accurate problem-solving in a wide range of computer vision tasks. Table 3.1 summarises the key techniques in the field.

In limited-resource environments, preprocessing techniques must take into account both effectiveness and computational efficiency (5). Lightweight methods, such as simple filtering and resizing, are often preferred, while more computationally intensive techniques may require optimisation or hardware acceleration.

3.2.1.3 Feature Processing in Vision classification Pipelines

During this process, raw data is transformed into structured and meaningful representations for analysis. A feature refers to a measurable property or characteristic derived from data (27). Features serve as the foundation for predictive and analytical models, directly influencing their ability to learn patterns and make accurate decisions.

Features can take various forms, including numerical, ordinal, binary, and text. Effective feature processing involves identifying, normalising, and structuring these features to emphasise relevant attributes (20). This section explores the main aspects of feature processing, highlighting its critical role in vision pipelines and its impact on classifiers performance.

a) Feature Extraction

Feature extraction is the process of deriving meaningful attributes from raw data to represent its essential characteristics. In the context of imaging, this involves transforming pixel values into a set of descriptors that encode important patterns or structures within the data. Fig. 3.2 shows a concrete example. Unlike raw inputs, these extracted features provide a higher-level representation of the data, emphasising attributes that are most relevant for analysis or classification tasks (20).

The process of feature extraction often includes mathematical or computational transformations, such as spatial filtering to highlight edges or textures, or Fourier transforms to analyse frequency components. These operations capture specific properties, such as shape, texture, or frequency patterns, enabling systems to interpret and process the data more effectively (28). The focus is on simplifying the raw input into a more compact and informative form, without altering its underlying meaning.

In essence, feature extraction creates a structured representation of raw input data, serving as the foundation for further processing and analysis in pattern recognition or machine learning systems. These methods are categorised based on their fundamental principles and the types of features they aim to identify (29). Below is an overview of widely used feature detection techniques for various applications in computer vision and machine learning.

1. Feature Extraction Methods

- **Statistical-Based Algorithms**

Statistical-based methods rely on analysing the statistical properties of pixel intensity distributions. These algorithms compute metrics like mean, variance, and higher-order moments to capture the essential characteristics of an image. For example, the Gray-Level Co-occurrence Matrix (GLCM) examines the occurrence of specific pixel value pairs at defined spatial relationships. It provides insights into texture and structural patterns (12–14).

- **Frequency-Based Algorithms**

Frequency-based techniques transform image data from the spatial domain to the frequency domain. This makes it easier to analyse periodic patterns and repetitive structures (25). Methods like the Fourier Transform decompose an image into its frequency components, revealing information about texture, edges, and orientation. These techniques are especially effective in identifying recurring patterns within textures.

- **Statistical Moments and Simple Descriptors**

Statistical moments, such as mean, variance, skewness, and kurtosis, provide a concise description of the intensity distribution in an image (39). These simple descriptors are efficient to compute and offer a compact representation of an image's properties. Therefore, makes them ideal for tasks requiring fast and lightweight analysis.

Table 3.1 key techniques for image preprocessing

Technique	Description	Aim	Methods	Use Cases	References
Noise Reduction	Minimises random variations in pixel intensity to improve feature quality.	Enhance image clarity while preserving important details.	- Gaussian Filtering - Median Filtering - Bilateral Filtering	Medical imaging, reducing sensor noise.	(25)
Image Resizing	Standardises image dimensions for consistent processing.	Ensure uniformity in image size for algorithm compatibility.	- N Neighbor Interpolation - Bicubic Interpolation - Down sampling with Antialiasing	Standardising input sizes for neural networks.	(28)
Normalisation	Adjusts pixel intensity values to a consistent range for uniformity.	Facilitate consistent data representation and analysis.	- Min-Max Normalisation - Z-Score Normalisation	Normalising pixel intensity for machine learning models.	(30)
Colour Space Transformation	Converts images into different colour spaces for specific tasks.	Optimise the image format for computational tasks.	- RGB to Grayscale - HSV Conversion	Simplifying inputs for object detection and segmentation tasks.	(31)
Edge Detection	Highlights boundaries by detecting significant changes in intensity.	Extract prominent edges to assist in feature recognition.	- Canny Edge Detection - Sobel Operator - Prewitt Operator	Boundary detection for object recognition.	(32)
Image Thresholding	Converts grayscale images into binary images based on thresholding.	Simplify image data by isolating significant regions.	- Global Thresholding - Otsu's Method - Adaptive Thresholding	Isolating objects in image binarization.	(33)
Morphological Operations	Processes images based on their shapes to refine or emphasise regions.	Refine structural elements or remove noise artifacts.	- Erosion - Dilation - Opening - Closing	Improving image shapes for analysis in biomedical imaging.	(34)
Gamma Correction	Adjusts image brightness to correct non-linear intensity variations.	Improve image visibility and dynamic range.	- Power-Law Transformation - Log Transformation	Enhancing brightness in low-contrast images.	(35)

- **Texture-Based Algorithms**

Texture-based methods quantify the spatial arrangement of pixel intensities to describe the texture of an image (36). Local Binary Patterns (LBP), for instance, encode the relationship between a pixel and its neighbours into a binary representation, effectively capturing micro-patterns. This approach is commonly used in texture classification, defect detection, and face recognition.

- **Shape-Based Algorithms**

Shape-based techniques analyse geometric properties like edges, boundaries, and contours within an image. The Hough Transform (37) is a widely used method for detecting lines, circles, and other shapes by transforming points in image space into parameter space. These methods are crucial in applications such as road marking detection and industrial inspection.

- **Deep Learning-Based Algorithms**

Deep learning methods, particularly Convolutional Neural Networks (CNNs), have revolutionised feature detection by learning hierarchical representations directly from raw image data. These networks extract complex features at multiple levels, enabling accurate image classification and object detection (38). They have proven effective across domains, from medical imaging to autonomous driving.

- **Domain-Specific Algorithms**

Particular fields design these algorithms to address specific challenges. For instance, in medical imaging, algorithms may analyse textures to detect abnormalities in tissues. In remote sensing, vegetation indices like Normalised Difference Vegetation Index (NDVI) assess plant health from satellite images (40). By leveraging domain knowledge, these methods enhance the relevance and effectiveness of feature detection.

The diverse methodologies and applications of these feature detection techniques are summarised in Table 3.2 (29,41). It provides a comprehensive view of how different approaches contribute to effective feature extraction in various domains.

2. Feature Extraction Challenges

The feature extraction process encounters several challenges that can impact the performance and generalisation of models. Below, we delve into three primary challenges.

a) Robustness to Image Variations

Images often exhibit variations due to changes in lighting, scale, rotation, occlusion, and noise. These variations can lead to inconsistent feature representations, affecting the accuracy and reliability of computer vision systems (42,43).

- **Lighting Variations**: Changes in illumination can alter the appearance of objects, making it difficult for features to remain consistent across different lighting conditions.
- **Scale and Rotation Variations**: Objects may appear at different sizes or orientations, challenging the feature extraction process to recognise them as the same object.
- **Occlusion**: Partial obstruction of objects can result in incomplete feature representation, complicating accurate recognition.
- **Noise**: Random variations in pixel values can obscure important features, leading to potential misinterpretation.

b) Dimensionality Reduction

High-dimensional feature spaces can lead to increased computational costs, memory usage, and the risk of overfitting (44). Effectively reducing dimensionality while preserving essential information is a significant challenge (29,45).

- **Computational Complexity**: High-dimensional data requires more computational resources for processing and analysis, which can be impractical for large datasets.
- **Storage Requirements**: Storing high-dimensional data demands substantial memory, posing challenges for systems with limited resources.
- **Overfitting Risk**: Models trained on high-dimensional data may capture noise and irrelevant patterns, leading to poor generalisation to new data.

c) Overfitting and Underfitting

Achieving a balance between overfitting and underfitting is essential for model generalisation.

- **Overfitting**: Models that are too complex may fit the training data too closely, including its noise, resulting in poor performance on unseen data.
- **Underfitting**: Models that are too simple may not capture the underlying structure of the data, leading to poor performance even on training data.

These challenges underscore the importance of developing sophisticated feature extraction methods that can handle variability in data (46). Additionally, it is essential to develop techniques

that reduce dimensionality effectively (47). Maintaining a balance between overfitting and underfitting is also critical for enhancing model performance across various applications (48).

b) Feature Description (Coding)

Feature description, usually referred to as feature coding, is the process of giving a meaningful representation to each extracted feature. Most of the time, this representation is vector-based since many classifiers in tasks like image recognition require fixed-length vector inputs (20).

In the context of image analysis, feature coding is the process of encoding local features derived from image patches. The local features are extracted from specific regions of an image, known as region of interest (ROIs) (27). These features are transformed into standardised, fixed-length vectors, which serve as inputs to the recognition system. Fig. 3.2 illustrates the sequence.

The main objective of feature coding is to ensure that these vectors really represent the fundamental information while being compatible with the input requirements of classification algorithms (49). This step mainly translates the raw feature data into an organised format that can be successfully worked out by the classification models.

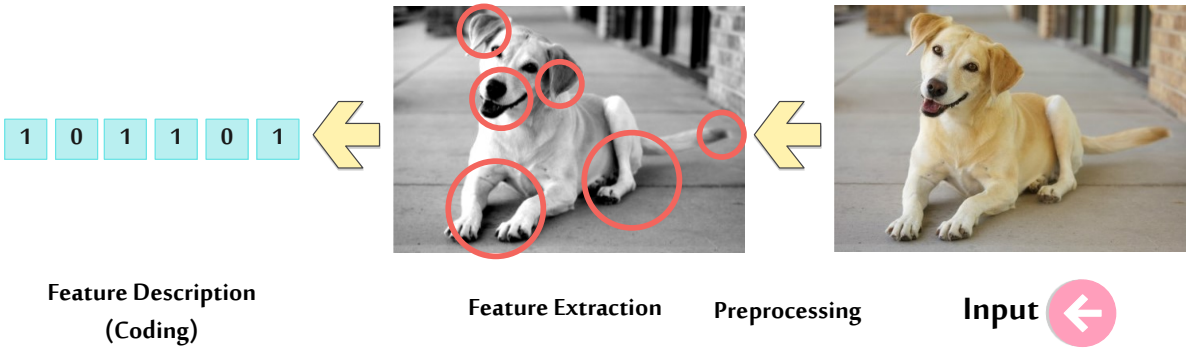


Fig. 3.2 Illustration of Feature extraction and Description

Table 3.2 Overview of Common Feature Detection Techniques and Their Applications

Technique	Examples	Applications	Output Format	Analysis Focus	Real-World Usage
Statistical-Based Algorithms	Mean, Variance, GLCM	Texture analysis, object recognition, image retrieval	Numerical vector	Analysing statistical properties of pixel intensity distributions	Image texture analysis in defect detection
Frequency-Based Algorithms	Fourier Transform, Wavelet Transform,	Texture classification, image compression, medical imaging	Frequency coefficients or numerical vector	analyse periodic patterns and structures	Texture classification in medical imaging
Texture-Based Algorithms	LBP, GLCM, Fractal Analysis	Material classification, defect detection, biometric identification	Histogram or texture descriptor vector	Capturing texture characteristics for classification	Facial recognition for biometrics
Shape-Based Algorithms	Hough Transform, Contour Analysis	Object recognition, scene analysis, shape-based matching	Shape descriptor vector	Analysing geometric shapes and contours in images	Road lane detection in autonomous vehicles
Deep Learning-Based Algorithms	CNNs	Face recognition, object detection, medical diagnosis	Hierarchical feature vector	Learning hierarchical features from data	Object detection in surveillance systems
Statistical Moments and Simple Descriptors	Mean, Skewness, Moment Invariants	Shape classification, feature simplification, image comparison	Statistical descriptor vector	Evaluating statistical moments for feature extraction	Shape analysis in medical imaging (e.g., tumour detection)
Domain-Specific Algorithms	Haralick Features, NDVI, Gabor Wavelets	Healthcare, agriculture, remote sensing, manufacturing	Application-specific descriptor (e.g., indices, wavelets)	Tailoring feature extraction techniques to specific applications	Satellite vegetation analysis using NDVI

c) Feature Selection

Selecting the optimal features is one of the most challenging aspects of pattern processing. This difficulty arises from a circular dependency. Determining the best features requires evaluating classifier accuracy. However, the classifier itself relies on the extracted features (50). This challenge complicates the objective assessment of feature extraction methods. In the absence of clear guidelines, researchers often rely on educated guesses or subjective choices based on personal judgment. Such approaches can introduce bias when designing feature sets (51). While these approaches can sometimes yield effective results, they become less reliable as the complexity of the patterns increases.

Feature selection can be defined as the process of identifying relevant characteristics while eliminating unnecessary and redundant ones. It aims to produce a subset that accurately represents the problem with minimal performance impact (52). This process offers numerous advantages (27), including:

- The enhancement of machine learning algorithm performance.
- General data reduction, minimising storage requirements and potentially aiding in cost reduction.
- Feature set reduction, conserving resources in subsequent data collecting or during application.
- Simplicity, the potential for employing more compact models, and enhanced speed

Despite these advancements, feature extraction methods continue to vary widely. This diversity highlights the complexity and subjectivity involved in selecting the most effective features for specific tasks (52).

1. Feature Selection Methods

Feature selection methods can be broadly classified into two categories: individual evaluation and subset evaluation (53). Individual evaluation, often referred to as feature ranking, assesses features individually by assigning weights based on their relevance to the target variable. While this approach is computationally efficient and easy to implement, it is limited in its ability to address redundancy among features. Redundant features are likely to have similar relevancies, making their ranking comparable and challenging to identify for removal.

In contrast, subset evaluation focuses on the generation of subsets of features using particular search strategies. Each generated subset is valued using a certain measure and compared with previously identified subsets to determine its effectiveness (54). This approach effectively handles feature redundancy by evaluating the relationships and interactions between features (55). It faces a significant challenge in computation since it is necessary to conduct a search through a number of feature subsets during the generation step, especially in high-dimensional datasets. Nevertheless, subset evaluation enables a more comprehensive way of feature selection through considerations of relevance and redundancy (53) .

Beyond these two approaches, feature selection methods can be further grouped by the level to which they are integrated into the learning algorithm that constructs the predictive model (52). The key of this classification underlines the interaction between feature selection and model building, especially methodological choices while designing efficient and accurate learning systems. Table 3.3 presents a comprehensive comparison of feature selection methods, categorised based on their Integration with Learning Algorithms.

- **Filter Methods**

These methods rely on the general characteristics of training data. They carry out the feature selection process as a preprocessing step, independent of the induction algorithm. This approach is advantageous for its low computational cost and good generalisation ability.

- **Wrapper Methods**

It involves using a learning algorithm as a black box. The algorithm's prediction performance is used to assess the relative usefulness of subsets of variables. In other words, the feature selection algorithm uses the learning method as a subroutine. This approach introduces a computational load, as the learning algorithm must evaluate each subset of features. However, this interaction with the classifier often results in better performance than filters.

- **Embedded Methods**

They perform feature selection in the process of training and are usually specific to given learning machines. Therefore, the search for an optimal subset of features is built into the classifier construction. This process can be viewed as a search within the combined space of feature subsets and hypotheses. This approach is able to capture dependencies at a lower computational cost than wrappers.

Table 3.3 Classification of Feature Selection Methods by Integration with Learning Algorithms

Category	Algorithms	Individual Evaluation	Subset Evaluation	Advantages	Disadvantages	Use Cases
Filters	Chi-Squared, Information Gain (IG), Correlation-Based Feature Selection (CFS), Consistency-Based Filter, Fast Correlation-Based Filter (FCBF), INTERACT, ReliefF, Minimum Redundancy Maximum Relevance (mRMR), Md Filter.	Chi-Square, IG, ReliefF, mRMR, Md	CFS, Consistency, FCBF, INTERACT	Independence of the classifier Lower computational cost than wrappers Fast Good generalisation ability	No interaction with the classifier	Preprocessing for high-dimensional data (e.g., text classification, genomics)
Wrappers	Recursive Feature Elimination (RFE), Sequential Forward Selection (SFS), Sequential Backward Elimination (SBE), Wrapper Subset Evaluation (Weka)	-	All	Interaction with the classifier Captures feature dependencies	Computationally expensive Risk of overfitting Classifier-dependent selection	Small to medium datasets with complex interactions (e.g., bioinformatics, image processing)
Embedded Methods	Recursive Feature Elimination for Support Vector Machines (SVM-RFE), Feature Selection Perceptron (FS-P)	FS-P	SVM-RFE	Integrated with model training Computationally efficient compared to wrappers Captures dependencies	Classifier-dependent selection Model-specific Results may not generalise to other algorithms	Regression and classification problems with regularisation (e.g., medical imaging, financial forecasting)

Combining feature selection methods present a promising opportunity to balance computational efficiency with performance. This approach leverages the strengths of both filters and wrappers (56). It consists of using filters for an initial reduction of irrelevant features, ensuring scalability and speed, followed by wrappers to refine feature subsets with model-specific evaluations.

This approach enables hybrid methods to address high-dimensional data challenges effectively and capture feature interactions and dependencies (57). Their flexibility allows for adaptation to specific tasks, making them a versatile solution in domains like bioinformatics (58), text mining, and image analysis (59). By bridging the gap between computational efficiency and optimisation, hybrid methods highlight the potential for innovative, domain-specific applications.

2. Feature Selection Challenges

Extensive datasets have presented new challenges for feature selection methodologies. Although conventional methods have demonstrated efficacy with smaller datasets, the emergence of "Big Data" and "High Dimensionality" necessitates flexible and scalable strategies (60).

Significant problems emerge from the excessive number of features, potentially resulting in inefficiencies, redundancy, and augmented computational expenses. To address these demands, scaling feature selection necessitates both creative algorithms and frameworks that can function efficiently in distributed and real-time settings (61,62).

This section examines the issues associated with large dimensionality, scalability, and novel techniques including distributed and real-time feature selection. Addressing these challenges enables the development of tools that meet the needs of data-intensive fields.

- **Ultra Large Number of Dimensions**

In the modern era of Big Data, machine learning techniques must be updated to effectively manage this enormous volume of data. The phrase “Big Dimensionality” is used to describe the extraordinary increase in the number of features. This increase has reached levels that render current state-of-the-art machine learning techniques insufficient (63).

- **Scalability**

Most existing learning algorithms were designed when datasets were relatively small. Today, handling small-scale and large-scale learning problems requires different trade-offs (64). For

small-scale problems, the trade-off typically revolves around balancing approximation and estimation.

In contrast, large-scale problems involve a more complex trade-off that considers not only accuracy but also the computational demands of the algorithm (65). Additionally, many algorithms assume that datasets can fit entirely in main memory. This assumption becomes a major limitation when datasets exceed memory capacity, rendering these algorithms ineffective.

- **Distributed Feature Selection**

Typically, feature selection is performed in a centralised way, where a single learning model is used to address the problem. However, with data now often distributed across different sources or locations, feature selection can benefit from processing multiple subsets either sequentially or concurrently. This shift introduces several approaches to distributing the feature selection task (66).

- **Real-Time Processing**

With the rise of Big Data, information is being collected at an exponential rate. This rapid growth creates a serious need for methods that can process data efficiently and quickly (52). In an era dominated by social media networks and portable devices, real-time solutions for tasks like content moderation or augmented reality filters in video analysis are becoming increasingly essential. These solutions ensure smooth user experiences and managing vast amounts of dynamic content (63).

The challenges discussed in this section underscore the importance of addressing the evolution of data dimensionality. A topic that remains insufficiently addressed in the global research literature (60). In recent years, datasets with millions of features have become common, and this trend is expected to continue as advancements in computing and information technologies progress (67).

3.2.1.4 Classification and Decision Making

a) Classification

Image classification seeks to establish a mapping function f that assigns a label y to a given image i, expressed mathematically as $y=f(i)$. The optimal function is one that minimises classification errors when evaluated on a labelled dataset (training data) (68). To make this process computationally efficient and achievable with a finite dataset, the problem is regularised by

constraining the set of potential functions. This restriction ensures that the optimal function can be identified within practical limits of computation and data availability.

During the inference phase, the function f is generalised to predict labels for previously unseen data images that were not included in the training process. This generalisation is a critical aspect of image classification, as it determines the model's ability to perform accurately on real-world datasets. By balancing computational constraints with the need for effective generalisation, classification models aim to deliver robust and scalable solutions to diverse image recognition tasks.

1. Strategies for classification

Generally, two different strategies for classification can be distinguished:

- **Matching an appearance model with image features**

This involves aligning a predefined model that represents the appearance of an object (e.g., a face, shape, or texture) with features extracted from an image. The goal is to identify and localise the object within the image by matching its features to the model. Techniques like template matching, feature matching (e.g., SIFT, ORB), or deep learning-based object detection algorithms (e.g., Faster R-CNN, YOLO) are often used for this purpose.

- **Mapping a feature vector to a class label**

This refers to the classification process, where a vector of features extracted from an image is mapped to a corresponding category or class. This step typically involves using a trained classifier, such as a Support Vector Machine (SVM), Decision Tree (DT), or a Neural Network (NN), to assign a label (e.g., "cat" or "dog") to the input based on the learned mapping between features and classes during training.

2. Types of Classification Models

- **Traditional Machine Learning Models**

Traditional models rely on explicit feature extraction followed by classification, Fig. 3.3 illustrates the process. These models include algorithms such as Support Vector Machines (SVMs), Decision Trees, and k-Nearest Neighbours (kNN) (4). These classifiers operate by mapping handcrafted feature vectors to class labels using pre-defined rules or statistical relationships.

While effective for smaller datasets with well-separated classes, traditional models often struggle with high-dimensional data and require domain expertise for feature engineering (69).

For instance, SVMs are highly effective in defining linear and non-linear decision boundaries when combined with kernel functions, but their scalability is limited in high-dimensional spaces (70).

- **Deep Learning Models**

Deep learning models integrate feature extraction and classification into a unified trainable framework. This eliminates the need for manual feature engineering. Fig. 3.3 demonstrates the concept. Convolutional Neural Networks (CNNs) are a main example, excelling in capturing spatial hierarchies of features from raw pixel data (71). These models utilise layers of convolutional and pooling operations to distil essential features from images. Fully connected layers then follow for classification (72).

While deep learning models are highly adaptable and effective in handling large-scale, complex datasets, they demand extensive computational resources and labelled data for training (73). Their end-to-end learning capability represents a significant advancement over traditional approaches.

- **Hybrid and Ensemble Models**

Hybrid and ensemble models combine multiple classifiers to enhance prediction accuracy and robustness. Techniques such as bagging (bootstrap aggregation) and boosting are commonly employed to aggregate decisions from diverse classifiers (74). For example, in ensemble models, weaker classifiers like decision stumps are sequentially trained in boosting schemes (e.g., AdaBoost), with each classifier focusing on errors made by the previous one. Ensemble strategies effectively address non-linearities and multimodal likelihood functions while mitigating overfitting (75). These methods are particularly useful for sparse and high-dimensional feature spaces.

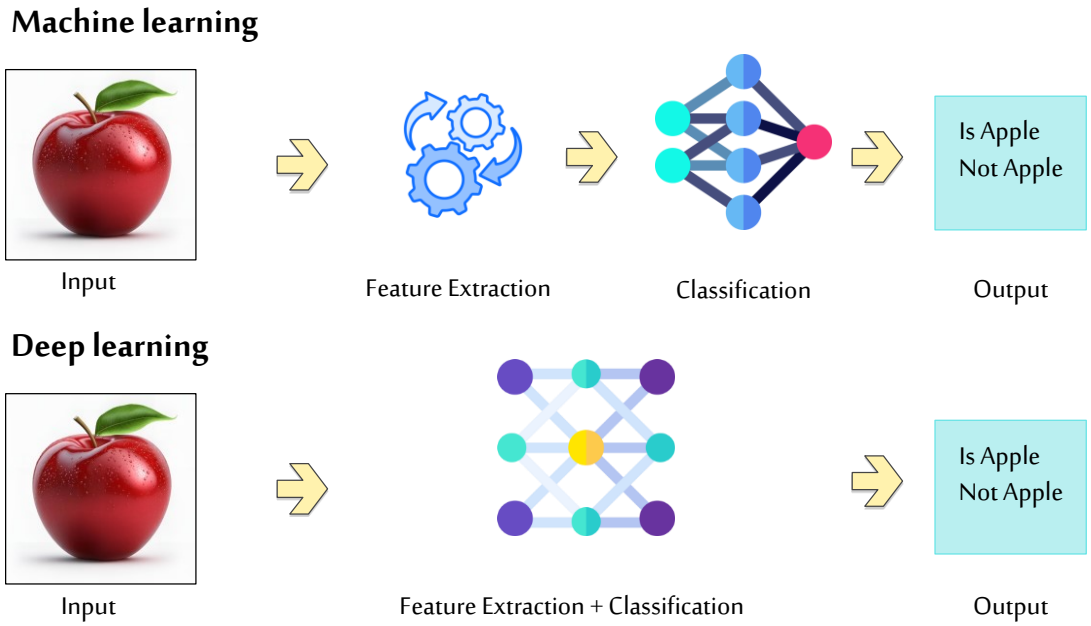


Fig. 3.3 Feature extraction in machine learning and deep learning

b) Decision Making

Decision-making is the final stage in image classification, where a system assigns a class label to an input based on extracted features. This critical step determines the performance and reliability of the classification system (68). The approaches to decision-making vary significantly depending on the nature of the model employed and the characteristics of the data. Broadly, decision-making strategies can be categorised into traditional feature-based methods (76) and neural network-based approaches (77). Each type has unique advantages and challenges, making the choice of method highly context-dependent.

1. Decision-Making Methods

- **Decision-Making in Traditional Feature-Based Methods**

Feature-based methods rely on explicit decision boundaries defined in a feature space constructed from manually designed descriptors. In these systems, algorithms such as Support Vector Machines (SVMs) (78) or decision trees use extracted features to partition the feature space into regions corresponding to different classes. For example, linear classifiers like SVMs with linear kernels create hyperplanes to separate classes. These kernels are optimal for linearly separable data. Fig. 3.4 shows the difference between linear and non-linear decision boundaries.

In contrast, kernel SVMs can be customised to model non-linear boundaries (79), enabling them to capture more complex relationships. Additionally, methods like template matching or active appearance models align image features with predefined templates. This provides a direct approach to decision-making. While these methods are computationally efficient and interpretable, they are often limited by their dependence on handcrafted features and their difficulty in handling high variability within classes (80).

- **Decision-Making in Neural Networks**

Neural networks take a fundamentally different approach by integrating feature extraction and decision-making into a unified, trainable framework (81). Through end-to-end learning, neural networks automatically learn the most relevant features and map them to class labels. Their ability to model both linear and non-linear relationships makes them highly adaptable (68). Neural networks often employ probabilistic outputs through activation functions like SoftMax, which provide class probabilities rather than hard labels.

This flexibility allows for threshold-based classification, accommodating uncertainty and improving robustness in complex datasets (82). However, neural networks require large amounts of labelled data and computational resources. In addition, their decision-making processes can sometimes lack interpretability compared to traditional methods.

Understanding the distinct characteristics of feature-based and neural network-based decision-making allows to select the most appropriate strategy for the specific classification task (77). Each approach has its place, with feature-based methods offering simplicity and clarity and neural networks excelling in scalability and adaptability to diverse data distributions.

2. Decision Boundaries and Strategies

- **Linear vs Non-linear Decision Boundaries**

Decision boundaries define the regions in the feature space where a classifier transitions from predicting one class to another. In Fig. 3.4 linear decision boundaries are represented by straight line in 2D, and could be a plane in 3D, or a hyperplane when the data exists in 4 or more dimensions (features), making them computationally efficient and interpretable (30). These are typically used when the data is linearly separable, as seen in classifiers like Support Vector Machines with linear kernels and Logistic Regression (83).

In contrast, non-linear decision boundaries are required for more complex datasets where classes cannot be separated by a straight line (84). These boundaries can take irregular shapes, capturing complex patterns within the data. In Fig. 3.4 non-linear decision boundaries are represented by a circle. Classifiers such as SVMs with non-linear kernels, neural networks, and decision trees are well-suited for modelling non-linear decision boundaries. They allow for greater flexibility in handling real-world problems with complex feature distributions (85). A comparative analysis of the fundamental attributes of linear and non-linear classifiers is summarised in Table 3.4.

Table 3.4 Comparative Analysis of Linear and Non-linear Classifiers Based on Decision Boundaries

Aspect	Linear Classifiers	Non-linear Classifiers
Decision Boundary	Straight line or hyperplane	Curved, irregular, or complex shapes
Examples	Logistic Regression, Linear SVM, Perceptron	Kernel SVM, Decision Trees, Random Forests, k-NN, Neural Networks
Working Mechanism	Separates data using linear equations	Uses transformations, kernels, or hierarchical rules to separate data
Data Assumptions	Assumes data is linearly separable or nearly separable	Handles non-linear patterns in data effectively
Interpretability	High (easy to understand weights and decision rules)	Moderate to low (e.g., decision trees are interpretable, but neural networks are not)
Training Complexity	Low (efficient on small datasets)	High (requires more computation, especially for large datasets or deep models)
Memory Requirements	Low (requires minimal storage for coefficients)	High (storing kernel matrices, decision paths, or network weights)
Strengths	- Fast training - Works well with structured/tabular data	- Flexible - Can model complex relationships - Handles unstructured data like images
Weaknesses	- Limited to linearly separable data - Struggles with complex patterns	- Computationally intensive - Risk of overfitting without proper regularisation
Applications	Text classification, spam filtering, small-scale predictive analytics	Image recognition, medical diagnostics, autonomous systems, speech analysis
Examples of Use Cases	Classifying emails as spam/non-spam	Predicting tumour malignancy in medical images

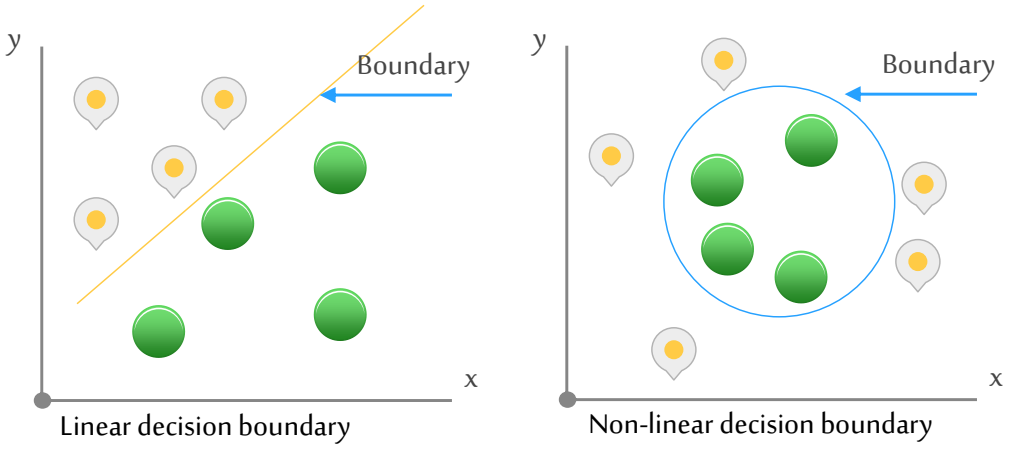


Fig. 3.4 Visual comparison of linear and non-linear decision boundaries

- **Probabilistic vs Deterministic Strategies**

Classification strategies are generally categorised as probabilistic or deterministic. Probabilistic strategies assign probabilities to each class, providing a measure of confidence in the prediction (86). These strategies, used in models such as Naïve Bayes and Logistic Regression, are particularly useful when decisions need to incorporate uncertainty or when threshold-based decision-making is required (87). For instance, in medical diagnostics, a probabilistic model can estimate the likelihood of a disease, aiding in informed decision-making (88).

On the other hand, deterministic strategies make definitive predictions without providing confidence scores (89). These strategies, such as those used in decision trees or rule-based systems, are valued for their simplicity and speed. This makes them appropriate for applications where interpretability and rapid decision-making are prioritised. The choice between probabilistic and deterministic approaches depends on the specific requirements of the task (87). These include the need for confidence measures, computational constraints, and application domain. Table 3.5 provides a comparative analysis of the two strategies.

Table 3.5 Comparison of Probabilistic and Deterministic Strategies in Classification Tasks

Feature	Probabilistic	Deterministic
Boundary Type	Soft: Decision boundaries are derived from probability thresholds, allowing flexibility and adaptability.	Hard: Decision boundaries are fixed and create strict separations.
Uncertainty Modelling	Yes: Provides probabilities that quantify confidence in classification decisions.	No: Assigns classes directly without modelling uncertainty.
Output	Probabilities or scores: Outputs real-valued probabilities or confidence scores.	Class labels only: Outputs binary or categorical labels.
Flexibility	Higher: Handles complex, overlapping data distributions effectively.	Lower: Works best when class distributions are clearly separable.
Computational Cost	Generally higher: Iterative optimisation is often required for probability calculations.	Generally lower: Simpler computations, with less reliance on iterative optimisation.
Examples	Logistic Regression, Naive Bayes, SoftMax Classifiers, Gaussian Mixture Models (GMM)	SVM, Decision Trees, K-Nearest Neighbours (kNN), Perceptron

- **Neural Networks Strategies**

Neural Networks (NNs) can operate using probabilistic strategies to provide more nuanced decision-making. These networks use activation functions like SoftMax for multi-class problems or sigmoid for binary classification, generating outputs that reflect the likelihood of belonging to each class (82). This probabilistic approach makes them especially effective in scenarios where understanding uncertainty is critical, such as in medical diagnosis or risk assessment (90). By quantifying confidence in predictions, probabilistic neural networks allow for more informed decision-making, particularly in cases where data is noisy or ambiguous.

On the other hand, Neural Networks can also adopt deterministic strategies, where class assignments are made based on predefined thresholds without explicitly modelling probabilities (91). These networks are often used in real-time systems, where speed and decisiveness are prioritised over uncertainty quantification. For example, in binary classification, a hard threshold applied to a sigmoid output quickly assigns a class. Similarly, piecewise linear activations like Rectified Linear Unit (ReLU) create clear, fixed decision boundaries. Deterministic

approaches are well-suited for tasks requiring fast, direct responses, such as embedded systems or automated control applications (92).

In summary, neural Networks seamlessly integrate into both probabilistic and deterministic strategies, adapting to the specific needs of a given task. Probabilistic neural networks excel in scenarios requiring uncertainty quantification and nuanced decision-making, while deterministic networks are favoured in time-sensitive or resource-constrained environments. By leveraging the strengths of both approaches, neural networks demonstrate remarkable adaptability across diverse classification challenges.

3.3 Fine-Tune Classifiers

In many applications, the optimal setting of a set of parameters defining a solution for a given problem is decisive. Theoretically, an exhaustive search of the entire parameter space could identify the combination of values that best suits the system's desired operation (93). These values are denoted as hyperparameters. To evaluate how well a solution meets the desired criteria, we can also use a formalised measure, often referred to as cost, energy, or an objective function such as likelihoods, cost and loss functions. These functions quantify alignment with the desired outcome and guides optimisation processes to minimise errors or maximise performance (86). While exhaustive search would guarantee finding the optimal solution, this process very quickly becomes impractical as the number of hyperparameters increases (94).

This exponentially growing computational cost becomes excessive, even for systems with a moderately high number of parameters. Therefore, classification model fine-tuning typically relies on alternative optimisation methods that efficiently explore the parameter space for near-optimal solutions, avoiding the computational overhead of exhaustive search (17).

The choice of optimisation method depends heavily on the model architecture, dataset size, and available computational resources. For instance, fine-tuning the hyperparameters of a convolutional neural network for image classification often involves adjusting the learning rate, dropout rate, and the number of layers. Efficient optimisation techniques significantly reduce training time while ensuring high model performance.

The primary challenges and trade-offs in fine-tuning classification models are balancing time versus accuracy and managing the risk of overfitting:

1. **Time versus accuracy:** Faster optimisation methods, such as random search or basic gradient descent, may sacrifice accuracy for speed. In contrast, more thorough

methods like Bayesian Optimisation can yield higher accuracy but require significantly more time and computational resources.

2. **Overfitting risk:** Excessive fine-tuning of hyperparameters on validation data can lead to overfitting, where the model performs exceptionally well on the validation set but struggles with unseen data.

3.4 Validation and Performance Assessment

Intelligent systems depend critically on their performance being measurable against well-defined benchmarks and validated by research literature. Quantitative evaluation through standardised methodologies is essential for assessing the effectiveness of emerging technologies. However, the lack of universally accepted standards for performance measurement presents significant challenges, including limited reproducibility and comparability of results (95). This issue restricts collaboration and delays progress across diverse fields, including manufacturing, healthcare, and security. Establishing standardised tools, reference datasets, and open-source libraries could simplify performance assessments, enabling more frequent and effective cost-benefit analyses of intelligent systems.

Performance evaluation methods vary in complexity, accuracy, detail, and development time. The choice of assessment model depends on the specific evaluation purpose, ensuring the approach aligns with the context and objectives of the assessment (96). This section addresses key performance metrics essential for evaluating vision systems, emphasising the need for standardised and reproducible evaluation frameworks.

3.4.1 Cross-validation

Cross-validation is a widely used approach in machine learning for estimating the performance of a model regarding its generalisation ability when the data available is small (68). The main aim of cross-validation is to ensure that the model generalises well on unseen data and performs well on the training data to avoid overfitting.

The process operates by dividing the data into subsets or "folds", using one-fold for validation and the remaining folds for training. We repeat this numerous times, giving each fold an opportunity to serve as a validation set, thereby enabling an extensive performance assessment. There are various methods to perform cross-validation, including k-fold, stratified k-fold, leave-one-out, nested, and time-series cross-validation (97).

Cross-validation achieves this by utilising techniques that produce a more accurate performance estimate. These techniques also help identify overfitting or underfitting, and optimise the limited data by using every data point for both training and validation (98).

3.4.2 Classification Accuracy

The performance of a classifier is evaluated using its accuracy, defined as the ratio of correct predictions to the total number of predictions made (68). For a dataset containing samples $\mathbf{x_i}$ **(i=1..., M)** with corresponding ground-truth class labels $\mathbf{Y_i \in \{1,...,C\}}$, the classifier **cls** assigns a label $\mathbf{cls(x_i)}$ to each sample. The accuracy **acc(cls)** is computed as:

$$acc(cls) = \frac{|\{xi : cls(xi) = yi\}|}{M}$$

This metric reflects the proportion of correct classifications relative to the total number of samples. A classification model is said to demonstrate learning if its accuracy exceeds the expected accuracy from random guessing. For instance, in a scenario with C=4 classes, where each class has an equal prior probability, random guessing would result in an expected accuracy of 0.25 (25%). Any accuracy above this threshold signifies that the classifier has successfully captured relationships between features and class labels.

While overall accuracy provides a useful high-level measure, it may overlook class-specific differences in performance. To gain a more detailed understanding, a confusion matrix is commonly employed. This matrix is structured with C rows and C columns, corresponding to the number of classes. Each row **i** represents the true class, and each column **j** represents the predicted class. The matrix entry **(i , j)** indicates the number of samples from the true class **i** that were classified as class j. Correct classifications appear along the diagonal of the matrix, where **i = j**, while misclassifications are reflected in off-diagonal entries.

The confusion matrix Fig. 3.5 provides valuable insights into classifier performance (68). It enables quick identification of classes with high misclassification rates, facilitating targeted improvements. Additionally, it offers a reconfirmation of overall accuracy by comparing the proportion of diagonal entries to the total (99). Moreover, it highlights particularly challenging classes, where significant confusion with others occurs, guiding refinements to the model. By integrating overall accuracy with the detailed analysis offered by the confusion matrix, a more comprehensive and nuanced evaluation of classifier performance can be achieved.

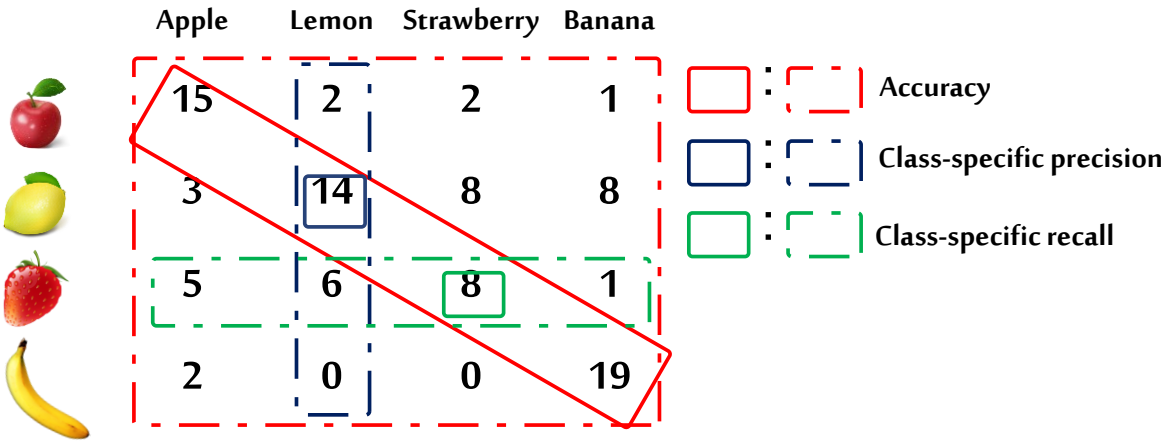


Fig. 3.5 Confusion matrix for a four-class problem

Precision score p: for a class with class label C measures the ratio of correctly labelled samples of a class C to all samples with this label. For instance, if 15 of 20 samples that were labelled as "Apple" truly are Apples and the remaining 5 samples were incorrectly labelled, then p=0.75 for this label.

Recall score r: for a class C measures how many members of a class were correctly labelled. If a database contains 25 samples with the label "Lemon" but just 15 of them were recognised as Lemons, the recall is r=0.60.

3.4.3 Resource-Specific Metrics

The evaluation of computational systems must extend beyond accuracy to address their operational effectiveness especially in resource-constrained environments. Systems deployed in real-world scenarios need to meet performance requirements while challenging with limitations in computational power, memory capacity, energy availability, and storage space (100). Adopting resource-specific metrics is essential to ensure these systems are viable for diverse applications.

The following metrics are critical to guide the design, assessment, and deployment of such systems:

1. **Computational Efficiency**

Systems must process data efficiently to ensure timely responses in scenarios where speed is critical (101). Metrics such as execution time and latency are indispensable for evaluating whether a system can meet performance demands while minimising computational overhead.

2. Memory Utilisation

For systems operating under constrained resources, particularly in embedded or edge environments (102), efficient memory usage is a fundamental requirement. Evaluating both peak and incremental memory needs during crucial phases ensures scalability and reliable performance under varying workloads.

3. Energy Consumption

Energy efficiency is a key metric for portable and battery-powered systems. It is vital to consider energy consumption when designing systems for long-term usability (103). Indicator measures, such as computational load and execution time, can provide practical insights when direct energy measurement is not feasible.

4. Model Compactness

Compact models are essential for systems deployed on devices with restricted storage capacities. Prioritising storage efficiency without significantly sacrificing accuracy ensures that systems can operate effectively across applications where memory and storage are limited (104).

5. Cost Efficiency

Adopting cost-efficient solutions helps balance performance with the financial and hardware resources required for deployment (105). This metric is critical for ensuring the scalability of systems while maintaining accessibility and affordability for practical use.

6. Scalability Across Applications

Scalability must be prioritised to enable systems to adapt to a variety of application domains and handle different dataset sizes (106). This metric ensures that solutions remain versatile and suitable for a range of environments, from resource-rich settings to low-power devices.

7. Real-Time Performance

Real-time processing capabilities are indispensable for systems that must respond rapidly to changing conditions (47,107,108). Metrics such as response time and frame rate are necessary to evaluate whether a system can meet the temporal requirements of dynamic applications like robotics and surveillance.

3.5 Applications of Adapted Vision Systems

The adoption of embedded systems across various industries marks a notable shift in advancing adaptability and operational efficiency and precision (109). These systems have transformed traditional processes by enabling real-time monitoring, data analysis, and automation (24,110). With their affordability, compact design, low power consumption, and robust processing capabilities, embedded systems have proven effective in tackling labour-intensive tasks (15,21,111).

Studies highlight the potential of these solutions to enhance accuracy, speed, and adaptability, leading to substantial improvements in productivity (22,107,112). This review examines the technological advancements achieved in the recent past years. It emphasises the distinct applications of embedded systems, the techniques utilised, the performance metrics evaluated, and their implications for future developments.

By analysing these innovations, this review aims to provide a thorough understanding of the current landscape and the promising prospects of these cost-effective technologies across diverse domains.

1. Autonomous Vehicles

Vision systems are optimised for embedded platforms to enhance safety in autonomous driving. (21) developed a vehicle-mounted real-time pedestrian trajectory prediction, the system was built utilising a compact computer known as Jetson Xavier. It combines pedestrian recognition, pose estimation, and optical flow data to estimate future pedestrian movements. The system demonstrated a reduction in displacement errors of 6.35% and 3.28% on the JAAD (Joint Attention in Autonomous Driving) and PIE datasets (Pedestrian Intention Estimation), respectively, as compared to the BiTraP model (Bidirectional Trajectory Prediction). These findings highlight the significance of multi-information fusion in trajectory predicting. Nonetheless, the study's dependence on certain datasets and the necessity for extensive real-world validation underscore opportunities for future inquiry.

Aeronautics is also an active domain for adapted computer vision applications. The research conducted by (113) introduces a vision-based in-flight collision avoidance system for Unmanned Aerial Vehicles (UAVs) exploiting an embedded computing platform. The system employs a set of methods, including dynamic background subtraction, denoising, grouping, and tracking, to identify and prevent collisions with moving objects.

The implementation entails background removal to emphasise moving items, employing morphological procedures and binarization for reducing noise. Moving objects are clustered, and noise blobs are excluded using Euclidean clustering. Finally, objects are tracked using the Kalman filter. Stereo cameras are utilised to calculate the three-dimensional trajectories of objects, and the system performs collision avoidance movements based on these calculated trajectories.

The system was evaluated on an affordable embedded platform and exhibited the ability to identify and monitor small moving entities, such as other drones, in real-time. The results demonstrate the system's efficacy in executing evasive moves, hence illustrating the possibility of implementing advanced vision-based collision avoidance methods on cheaper embedded systems. However, problems include depending on illumination conditions and residual noise in background subtraction indicate potential areas for enhancement to address these limits and guarantee reliable performance across varied operational contexts.

2. Environmental Monitoring

In the field of support for visually impaired individuals (15), techniques were assessed that decrease the memory and computational time necessary in a classifier system. These techniques adapted classification to low-power microcontroller systems. They preserve the precision achieved in workstations with significantly greater resources, conforming to the norms of a conventional implementation. The target application for experimental assessment is a crosswalk detector integrated into a wearable computer vision equipment designed to assist visually impaired individuals.

The implementation integrates the Gray Level Co-occurrence Matrix (GLCM) with the Support Vector Machine (SVM). This integration aids the development of a computer vision application based on a feature computation analysis. The analysis enabled the authors to reduce the feature length thereby achieving a lower complexity level compared to GLCM's twenty-four (24) statistical measures. In addition to employing resized images according to a proposed method for selecting the optimal resolution.

The authors indicated that the experimental results demonstrate that utilising the highest resolution is unnecessary. A reduced resolution can still achieve satisfactory accuracy with minimal memory usage. The suggested temporal complexity-based feature selection criteria demonstrated efficiency in both execution time and memory utilisation.

The main optimisation effort was on the feature extraction phase utilising GLCM. The results indicated that 94% of the classification timeframe is allocated to that step. The SVM classifier is complex, although the feature extraction is considerably more. The memory footprint of the SVM is reliant upon two factors: the number of Support Vectors (SV) and the number of Decision Functions (DF), both determined by the training procedure. Prior studies typically overlooked the latter due to SVM's classification as a binary classifier. However, applications with a greater number of classes typically require more data for decision functions.

On-board vision systems have also contributed to traffic surveillance. (24) presented an integrated computer-vision system intended for the detection of multiple objects in traffic scenarios. Their solution minimises the drawbacks of current systems, which frequently exhibit high costs, operational complexity, and difficulties with congestion, occlusion, and variable lighting conditions. The suggested system is engineered to operate precisely and rapidly in adverse conditions, including changes in lighting and varying daytime situations. The emphasis is on identifying and classifying traffic items in various contexts utilising a refined Sequential Monte Carlo (SMC) and Faster R-CNN architecture.

The improved Modified Faster R-CNN (MF R-CNN) framework presents an innovative architecture to boost the detection of small objects and a new probability function, that more effectively chooses target samples with accurate labels. This enhancement extends the likelihood function inside the SMC framework, utilising background-subtraction spatial-temporal cues to preferentially choose positive samples in particular scenarios. A tracklet-based method (short-term object trajectories) assigns weights to target samples depending on their significance. It emphasises positive samples and minimising the inclusion of inaccurately labelled instances in the training dataset. The findings demonstrate that the proposed likelihood function technique improves detector performance and speeds the training process.

The results reveal the efficacy of the novel likelihood function in precisely identifying positive samples and its superiority compared to the previously suggested SMC framework function. Subsequent research must concentrate on addressing the recognised limits. These include augmenting the system's robustness to varied traffic conditions. Additionally, optimising computational efficiency is a critical area of improvement. Another priority is guaranteeing high-quality and diversified training datasets.

Microcontroller-based vision systems are utilised in environmental monitoring and wildlife managing. (112) designed an IoT-based real-time image processing system combined with a Deep Convolutional Neural Network (DCNN) for the detection and classification of animals. The system aims to prevent human-wildlife conflicts via an early warning and monitoring module. It employs field cameras and radio frequency networks to collect and send images to a base station, where a pre-trained convolutional neural network model (AlexNet) processes them for feature extraction and labelling. Notifications are subsequently dispatched over the Global System for Mobile Communications (GSM) to the appropriate authorities. Global Positioning System (GPS) data, including latitude and longitude, species identification, gender classification for elephants, pose recognition, age estimation, strength assessment (group or individual), along with log date and time, is available via a secure web interface.

The study illustrates successful animal recognition and labelling, even under difficult circumstances. It highlights the practical method of computer vision adaptation for IoT with deep learning in wildlife surveillance. This method plays a crucial role in wildlife management, establishing a basis for subsequent technical progress in this domain. However, potential limits arise from reliance on data quality, environmental conditions, scalability, and technological infrastructure.

Agriculture, like prior fields, has seen advances powered by integrated vision systems. (107) designed a real-time embedded vision system (real-time EVS) for the online surveillance and classification of citrus fruits. It aims to enhance productivity and precision in agricultural practices. The system utilises a Field Programmable Gate Array (FPGA) for hardware execution, enabling fruit segmentation and categorisation according to colour and size. The decision tree (DT) classifier was trained on a balanced collection of reference images, guaranteeing effective pixel labelling.

The system showed superior performance metrics, reaching 97% accuracy in fruit segmentation, 94% in colour classification, and 90% in size prediction. It operates at a rate of 60 frames per second (fps), guaranteeing real-time processing capacity. The existing FPGA architecture is restricted to managing two video inputs, highlighting the necessity for higher-capacity FPGAs to enhance processing capabilities. Future efforts should concentrate on augmenting the system's capability to accommodate more types of fruits and increasing the volume of video feeds to improve processing power and output.

3. IoT and Edge Computing

Vision systems are also adapted to perform in extensive surveillance applications. (114) built a scalable architecture to enhance image processing and object detection in large-scale monitoring settings. The approach minimises data transfer and bandwidth usage while improving processing speed and scalability. This optimisation focusses on decreasing data transfer and bandwidth usage. The architecture has three tiers: the Perception Layer, the Network (Gateway) Layer, and the Application (Cloud) Layer.

The Perception Layer comprises cameras and sensors that acquire raw data. The Network (Gateway) Layer refines incoming data to lower its size and complexity before transmission, executing functions such as image enhancement and preliminary object detection. The Application (Cloud) Layer executes delicate processing functions, including object recognition and system coordination.

The system allocates image processing jobs between edge devices (gateways) and cloud servers. Simple preprocessing and object detection are performed at the edge to decrease bandwidth usage and reduce latency. Only the identified data, rather than the full image or video streams, is transmitted to the cloud for additional processing and identifying the target.

The implementation of microservices enables a modular system architecture. Every service, encompassing preprocessing, object detection, and recognition, is designed, deployed, and scaled autonomously. This modularity facilitates the allocation and scalability of discrete components as required, without affecting other system elements. The authors assert that the data demonstrate substantial advancements in large-scale surveillance operations through decreased bandwidth usage and improved processing speed, scalability, and adaptability.

Assessments demonstrated its suitability for use in smart buildings and industrial parks, providing a more resource-efficient and responsive alternative to conventional centralised systems. This research encounters multiple problems, including a significant reliance on internet connectivity and unclear distribution of workload between the edge and the cloud. These problems are crucial for optimising system performance. The experiment was performed in a controlled laboratory setting instead of a real-world Internet of Things (IoT) context, thus constraining the relevance of the findings to larger, more complex environments.

In the topic of trash management, optimised vision systems have demonstrated significant value. (111) developed a cost-effective computer vision system to enhance the sorting of recyclables. This system was set up on a Raspberry Pi computer and employed TensorFlow and OpenCV libraries to identify several materials, including glass, plastic, metal, paper, and cardboard. They utilised a big data set of waste photos known as TrashNet, supplemented by new pictures sourced from the internet, to train the system.

In trials, the solution successfully identified recyclables with 90% accuracy in a controlled, simulated setting. However, when they attempted to utilise it in real-time using a Raspberry Pi camera, the performance deteriorated. The accuracy decreased to 70%, and the system's processing speed declined to 1.4 images per second, in contrast to 10 images per second during simulations.

The issues primarily arose from the Raspberry Pi's constrained processing capabilities and the inferior quality of images produced by its camera. Although these obstacles, the study indicates that enhanced hardware and improved image clarity could render such a system highly beneficial. This would be particularly useful for automating the sorting of recyclables. Additional efforts are required to enhance the system's speed and precision for practical application in real-world recycling processes.

4. Healthcare Diagnostics

Embedded computer vision contributed to medical diagnoses. (23) introduced an embedded vision algorithm aimed at classifying retinal pictures for the identification of diabetic retinopathy. The researchers employed a convolutional neural network and k-fold cross-validation to train their model using 88,000 high-resolution photos sourced from the Kaggle/EyePacs database. The method attained a detection accuracy of 76%, suggesting applicability for non-specialist diagnostic purposes in underserved regions. The present accuracy is unsatisfactory for clinical implementation, requiring additional enhancement.

The research highlights the potential of integrated computer vision in improving early identification of diabetic retinopathy and lowering vision impairment in high-risk groups.

5. Industrial Automation

The research initiated by (110) in robotics presents an integrated computer vision system to improve the Long-Fiber Embedded FRESH (LFE-FRESH) hydrogel 3D printing process. The system

employs inexpensive components to monitor and regulate the Fiber embedding process in real-time, thereby reducing Fiber buckling and rectifying over-extrusion.

The technology revealed notable enhancements in the structural integrity and precision of 3D printed hydrogel components. These improvements were observed when tested using single-ply polyester and electrochemically aligned collagen (ELAC) fibres. Given its efficacy, the study underscores limitations, including dependence on real-time responses, indicating opportunities for further enhancement. This research demonstrates the capability of integrated computer vision systems to enhance hydrogel 3D printing applications, especially in tissue engineering and biohybrid robotics.

The literature investigation highlights the considerable progress of embedded computer vision systems in various fields, enhancing performance, precision, and overall productivity (24,111,113,114).

These systems have revolutionised conventional workflows in disciplines like healthcare (23), environmental monitoring (107), waste management (111), and autonomous navigation (21) through the integration of novel approaches, architectures, and techniques. Despite these advancements, a significant need remains for optimising computer vision algorithms and system architectures (109) to enhance their adaptability and performance in resource-limited settings. The subsequent section, examines these limits comprehensively, pinpointing areas for further investigation to advance the field.

3.6 Gaps in Existing Research

1. Balance Between Accuracy and Efficiency

Most of the embedded computer vision systems face a very serious trade-off between achieving a high degree of accuracy and retaining computational efficiency (110–112). Many techniques that represent the state of the art perform exceptionally well regarding accuracy, however they are often resource-consuming, hence unsuitable for deployment in constrained processing power and memory capacity contexts. This is especially critical for applications requiring real-time processing, where latency and optimisation of resources become very important (21). Current literature provides insufficient exploration of strategies that balance these competing requirements effectively.

2. Scalability of Solutions

The literature demonstrates that diverse embedded vision technologies are designed and refined for certain datasets (21,107,110) and restricted to experimental environments (111). However, their adaptability to practical applications continues to pose a considerable difficulty. Models designed for particular circumstances frequently exhibit a deficit of adaptation when confronted with varied, dynamic, or unpredictable contexts (24,113).

The scalability of these systems is not thoroughly investigated, which restricts their wider usability, especially in applications like extensive surveillance or challenging autonomous navigation tasks. This underscores the necessity for flexible frameworks that operate reliably across different scenarios and resource environments.

3. Adaptability for Extreme Constraints

Although progress in optimising computer vision systems for constrained environments, existing research frequently fails when addressing extreme limitations in power, memory, and computational capacity. For instance, applications in IoT (112) or wearable technologies (15) require models that operate efficiently with minimal energy consumption and storage. The development of algorithms that prioritise adaptability while maintaining acceptable accuracy levels remains an open research area. Addressing these constraints is vital for enabling the deployment of computer vision systems in under-resourced settings, such as remote healthcare diagnostics or wildlife monitoring.

4. Real-World Validation and Robustness

A considerable gap exists between laboratory-based research and real-world implementation. Many proposed solutions are validated on benchmark datasets or in controlled environments, which do not fully reflect the complexities of practical deployment (114). Factors such as varying lighting conditions, environmental noise, and hardware limitations are often overlooked (111). Bridging this gap requires rigorous real-world validation to ensure that proposed systems are robust and reliable under diverse operational conditions.

3.7 Conclusion

This chapter presented a comprehensive review of the literature on embedded computer vision systems. It highlighted their significant advancements and diverse applications across many domains. These vision systems have demonstrated their potential to enhance efficiency, precision, and adaptability while addressing critical challenges in resource-constrained environments. By leveraging novel techniques and architectures, they continue to develop and contribute to innovation across multiple fields.

Despite these achievements, the review also uncovered key research gaps, including the need for better scalability, enhanced adaptability in extreme conditions, and robust real-world validation. Addressing these challenges is essential for maximising the impact of embedded computer vision systems and extending their applicability to underexplored domains.

The next chapter, builds upon the insights gained from the literature review, focusing on addressing selected limitations from the identified gaps, relevant to the objectives of the thesis. It outlines approaches to optimise feature extraction and classifier design, aimed at improving computational efficiency, memory usage, and overall adaptability for resource-constrained environments. These methodologies pave the way for improving embedded vision systems (EVS) in real-world applications, while maintaining a balance between efficiency and accuracy.

Chapter 4

Methodology

4.1 Introduction

The analysis of existing literature revealed substantial gaps in adapting computer vision and image processing algorithms to environments with limited resources. The earlier chapter emphasised the need to overcome challenges related to computational constraints, memory efficiency, and flexibility. These aspects are crucial for enabling effective deployment in embedded systems, particularly for extreme constraints. This thesis bases its methodological approach on these challenges and the potential for enhancing real-time application performance.

This chapter presents a detailed insight into the methodology employed to address the thesis objectives. It begins by outlining the research design, which includes the conceptual approach and theoretical principles that support this approach. This section is followed by a detailed description of the efforts made over all the stages of the classification pipeline for enhancements, focusing on the features extraction process.

Additionally, the chapter delves into the dataset and data acquisition techniques. It describes the software and hardware configurations, as well as the tools and platforms utilised during implementation. The systematic methodology presented here establishes a solid foundation for the subsequent analysis and discussion of the study's findings in the following chapter.

4.2 Research Design

The design is based on foundational research, specifically the methods outlined by (15). Our proposed approach, as illustrated by the flowchart in Fig. 4.1, introduces performance-oriented optimisations to tackle specific challenges in computational and memory efficiency. The most significant improvement is the systematic reduction of the feature set by 79.2%, which directly results in considerable gains in different metrics. These enhancements enable the system to perform in real-time while balancing the trade-offs between efficiency and accuracy. This balance is a critical factor influencing successful deployment in constrained environments.

The theoretical approach focuses on optimising feature selection and classification strategies to address the demands of resource-constrained systems. To achieve this, two complementary methods were utilised to compact the feature set. First, Complexity-based feature selection (15) was applied to prioritise features extracted through the least computationally demanding techniques. This was followed by correlation-based feature selection (CFS), which identified and maintained the most relevant and non-redundant features. The Support Vector Machine (SVM)

classifier was chosen for its demonstrated efficiency in such environments, offering high accuracy while minimising computational overhead (73,115,116).

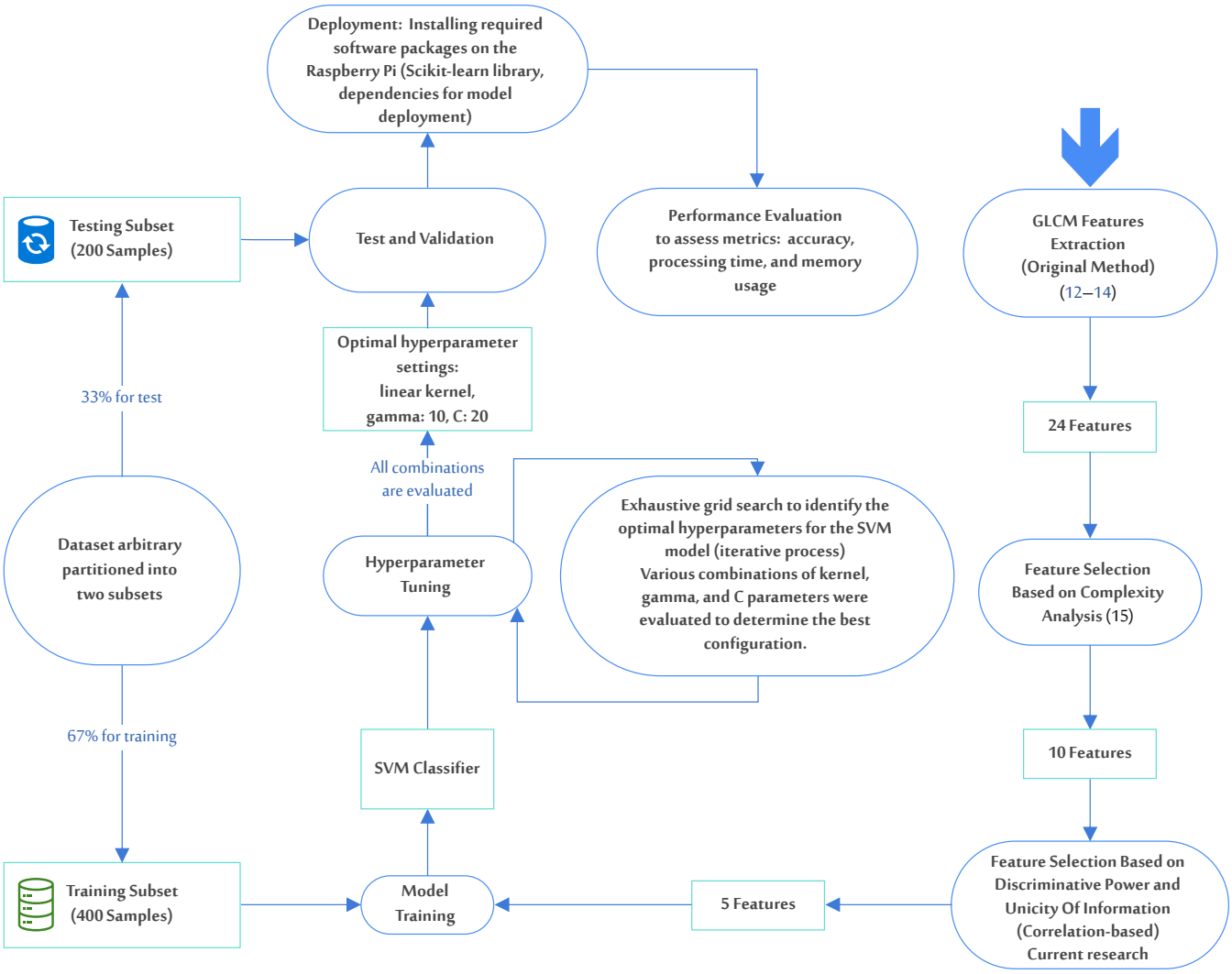


Fig. 4.1 Flowchart visualising the proposed methodology for enhancing the embedded classification system

4.3 Recognition Pipeline

Fig. 4.2 illustrates the sequential stages of the classification pipeline. It outlines visually the processes required to analyse and interpret visual data. The figure highlights the flow of information through each stage, emphasising the systematic approach employed in this work.

4.3.1 Image Preprocessing

In vision-based classification systems, preprocessing is essential for improving the quality of input data and ensuring consistent analysis. Through enhancing features and reducing irrelevant

information, preprocessing improves the system's ability to extract meaningful patterns. This step significantly impacts classification complexity, accuracy and efficiency. It facilitates models to achieve better performance and handle diverse input conditions effectively.

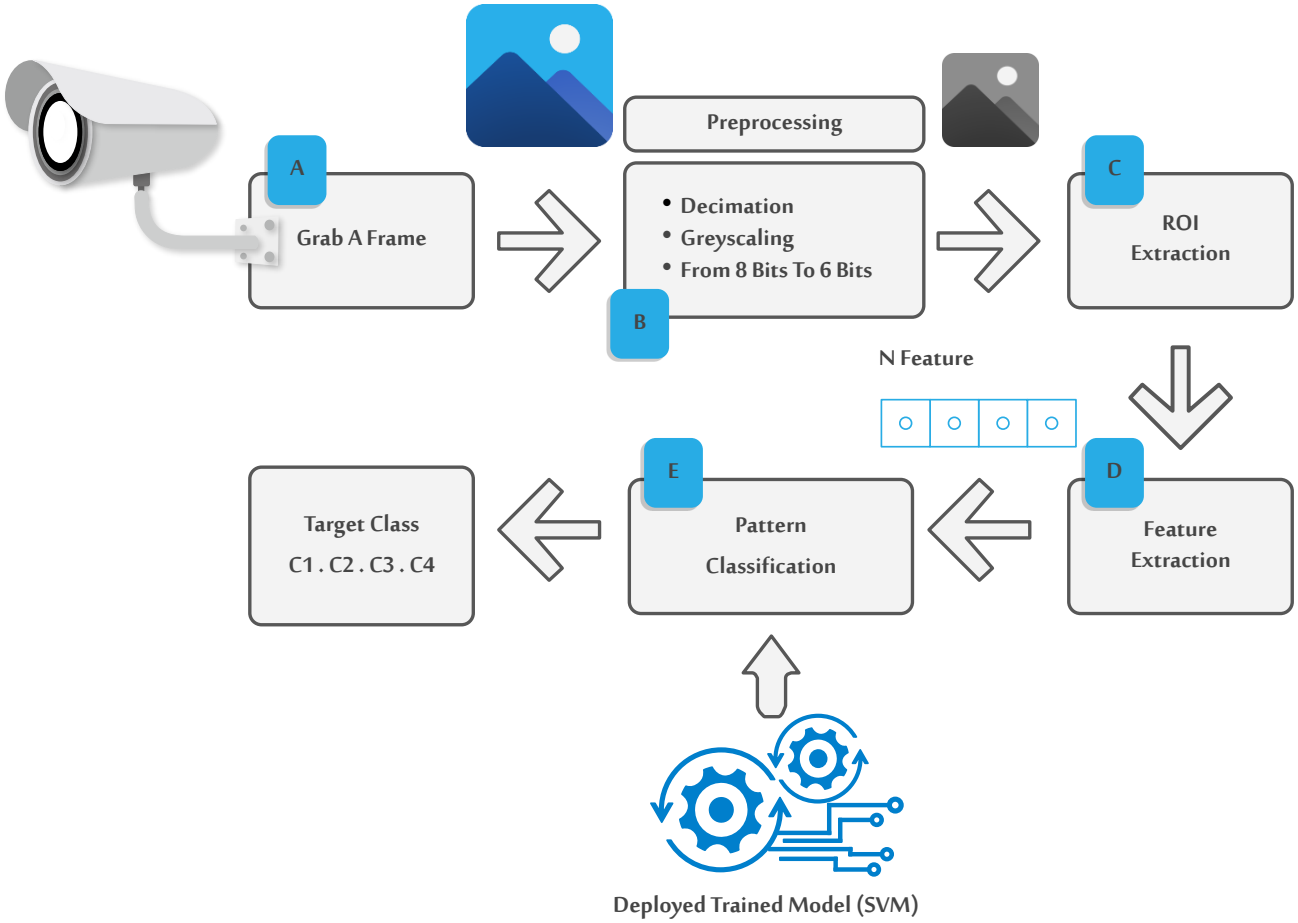


Fig. 4.2 Classification pipeline (15).

4.3.1.1 Greyscale Conversion

Greyscale conversion is a common preprocessing step in computer vision systems, particularly when colour information is not essential for the task, as in the current classification context. This process involves transforming a colour image, typically composed of RGB channels, into a single channel that represents intensity (117). The conversion is achieved by applying a weighted sum to the RGB values. The result is a greyscale image where each pixel intensity corresponds to a shade of grey (0 to 255).

The primary purpose of greyscale conversion is to simplify the image by reducing data complexity while preserving important structural features, like edges and textures (118). This

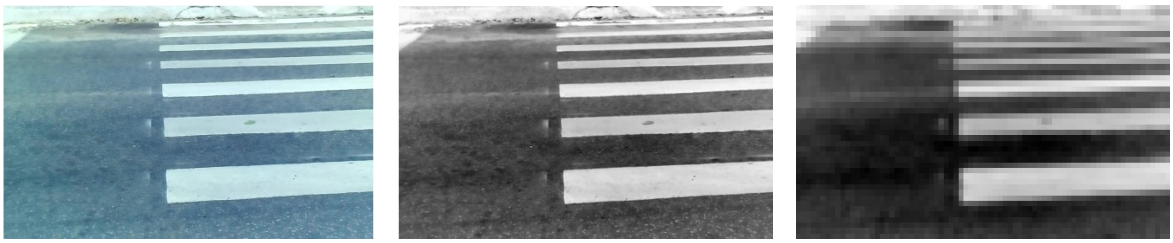
simplification improves computational efficiency, as the system no longer needs to process colour information.

4.3.1.2 Decimation

To further reduce computational complexity, the resolution of input images is decreased using decimation. While traditional resizing methods rely on interpolation to estimate new pixel values, decimation employs a low-complexity approach. it directly ignores certain pixels based on a predefined sampling pattern (119). For example, an image of size 1280×720 pixels can be decimated to 76×43 pixels by selecting every 17th pixel.

Fig. 4.3 illustrates the process. This approach significantly reduces the computational load without introducing artifacts from interpolation (120), making it well-suited for resource-constrained environments. Additionally, texture-based features, such as those used by GLCM, remain largely unaffected, ensuring the retention of critical information even in reduced-size images (15).

To preserve the same quality of the classifier attained by inputting images with larger dimensions, which consequently capture a higher level of detail, (15) chose the sample rate (M) = 17. This resulted in an image size of 76 × 43 pixels and average accuracy of 90.12%. This performance is close to 90.31%, achieved with the original input image size of 1280 × 720 pixels.



Original input (1280x720) pixels | **Grayscale input (1280x720) pixels** | **Decimated grayscale input by M=17 (76x43) pixels**

Fig. 4.3 Visual illustration of greyscale and decimation preprocessing results

4.3.1.3 Grey Levels Reduction from 8 to 6 Bits in Co-occurrence Matrix

Researchers In the study (15) investigated the effect of reducing the number of grey levels (Ng) in the Co-occurrence Matrix (CM) on classifier performance and computational efficiency. It compared different Ng values: 8, 7, 6, 5, and 4 bits, then evaluated classifier accuracy across various matrix sizes. The study found that reducing the grey levels does not significantly affect

classifier accuracy. Even when reduced to a 64x64 matrix with Ng = 6, the classifier maintained high accuracy, reaching 90%.

Reducing the grey levels led to substantial improvements in memory usage and computational efficiency, primarily by decreasing the time required for feature extraction. The results demonstrated an important reduction in computation time. The research (15) concluded that lowering grey levels provides a favourable balance between maintaining accuracy and significantly reducing both memory usage and processing time, as shown in Table 4.1.

Table 4.1 Execution Time as a Function of the CM Size (in milliseconds) (15)

Step	8 bits (256 x 256)	7 bits (128 x 128)	6 bits (64 x 64)	5 bits (32 x 32)	4 bits (16 x 16)
CM **Creation**	4.864	4.565	4.568	4.913	4.832
Feature Extraction	454.687	113.796	29.367	7.876	1.991
Classification (SVM)	0.019	0.019	0.023	0.023	0.021
Total (ms)	459.571	118.381	33.959	12.814	6.836

4.3.2 Roi Identification and Extraction

Regions of interest (ROIs) are parts of an image that are relevant to the classification task, allowing for targeted analysis instead of processing the entire image. By focusing on ROIs, computational resources are conserved, and the accuracy of the process is improved. They isolate information-conveying parts of the image, eliminate superfluous background noise, and concentrate on meaningful data. In the context of the vision system discussed in this thesis, the ROI is the crosswalk, which serves as the focal point for classification.

4.3.3 Feature Extraction

Feature extraction involves transforming raw image data into a set of measurable and meaningful characteristics that represent the essential patterns or properties of the image. These features are used to distinguish objects or patterns within the image, preparing them for classification or further analysis (25). The primary goal of feature extraction is to reduce the dimensionality of the data while retaining important information about the visual content of the image.

The adopted pipeline (15) utilised Grey Level Co-occurrence Matrix (GLCM) as an extractor. It is a statistical method used for textural analysis in images, resulting in a matrix of Nx N

values (4), where N corresponds to the number of distinct grey levels in the image. For example, an 8-bit image with 256 grey levels results in a 256 × 256 matrix.

The GLCM is constructed by analysing the spatial relationships between pixel intensities in an image. For each pixel, it records how often a neighbouring pixel, at a specified distance and orientation, has a particular grey level (121). This process generates a matrix where each entry corresponds to the frequency of a specific grey level combination, effectively capturing the textural structure of the image.

Once the matrix is generated, statistical features are extracted to describe the texture. These features have diverse real-world applications demonstrating the effectiveness of the extractor. For instance, they are used to detect abnormalities in medical imaging by analysing texture differences (122–124), classify land cover types in remote sensing (125,126), ensure quality control by identifying surface defects in industrial vision systems (127–129), support pattern recognition tasks such as face recognition (130) and object classification (131). The GLCM provides a compact, structured representation of textures, thereby delivering valuable insights that enhance analysis while reducing computational complexity.

GLCM generates 24 features (12–14) computable from a single image as input. These features are listed in Table 4.2. In the study conducted by (15), the researchers extracted all features from 600 images of the crosswalks (16) in their original captured resolution of 1280 x 720 pixel. They then calculated the accuracy of the full set, named FS1, after a prediction test run on 50 samples from the dataset.

The study revealed that using the complete features led to a good average precision of approximately 91%. However, to optimise the classifier for embedded devices, the researchers Implemented an approach based on a complexity analysis of feature computation. This approach will be further discussed and enhanced in the dedicated section titled "3.4 Feature Selection Process".

4.3.4 Pattern Classification

The classification stage (E) comes after the feature extraction. At this stage, it processes a vector of features derived from the GLCM, which originally consisting of 24 quantified values. These values serve as inputs for the SVM classifier. This process involves analysing incoming extracted relevant features to assign each sample to a predefined class. Fig. 4.4 represents samples from the

four labelled target classes. This is a fundamental step in pattern recognition, aimed at automating the labelling of data samples based on their features. Table 4.2 lists the GLCM's features as referenced.

The pretrained models can classify novel, unobserved data after learning from the tagged training dataset (132). The training subset consists of 400 samples selected arbitrarily among 600 instances of the target dataset (16). The samples are already matched with their respective class labels, allowing the model algorithm to learn the relationship between the input and the outputs. After completing the training by learning from given labelled examples (supervised learning), the model is capable of predicting the class labels for new input data (unseen data). In summary, the objective is to infer a function that maps inputs to outputs based on example input-output pairs.

Table 4.2 GLCM's features presented by reference

Reference	Haralick et al. (1973) **14 Features** (12)	Soh and Tsatsoulis (1999) **6 Features** (13)	Wang et al. (2010) **4 Features** (14)
Features	1. Angular Second Moment (ASM) 2. Contrast 3. Correlation 4. Sum of Squares 5. Sum Average 6. Inverse Difference Moment (IDM) 7. Sum Variance 8. Sum Entropy 9. Entropy 10. Difference Variance 11. Difference Entropy 12. Information Measures of Correlation I (IMC I) 13. Information Measures of Correlation II (IMC II) 14. Maximal Correlation Coefficient (MCC)	1. Homogeneity 2. Autocorrelation 3. Dissimilarity 4. Cluster Shade 5. Cluster Prominence 6. Maximum Probability	1. Sum Mean 2. Cluster Tendency 3. Difference Mean 4. Inertia
		Total	**24 Features**

(15) employed SVM for this task and it was retained in our work because it consistently surpasses other classifiers in various computer vision applications (73,115,116). It demonstrates a notable effectiveness and computational efficiency, particularly in small-scale applications. SVM excels at handling smaller datasets, making it an optimal choice for such scenarios. Reducing the number of features significantly decreases the time and resources needed for training and prediction. This, in turn, enhances, furthermore, the practicality and efficiency of SVMs, making them a highly suitable option for real-time EVS (133). This efficiency enables SVMs to consistently deliver precise classification outcomes while avoiding the high computational demands typically associated with larger datasets (134).

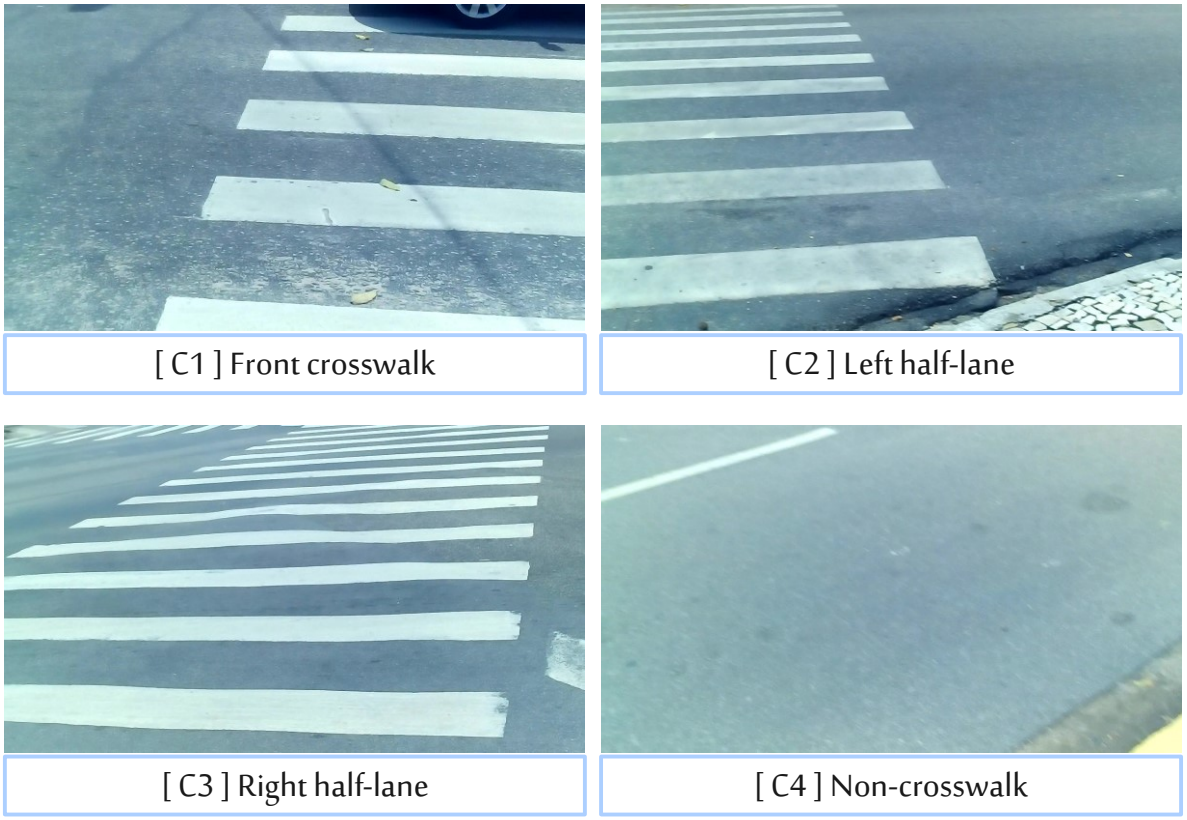


Fig. 4.4 Samples from the four classes (16).

In Chapter 02, Section 2.2.1.5, we discussed the concept of "linear discrimination". This refers to a straight line (or hyperplane in higher dimensions) that separates feature values into two distinct groups, denoted classes, one for each class (117). This concept aligns with the Support Vector Machine (SVM), which is fundamentally a two-class (binary) classifier (135). The SVM identifies the optimal decision boundary by maximising the margin between the two classes, ensuring efficient and accurate classification (136).

In the context of the current thesis, the classification problem involves handling multiple classes (K=4), which requires extending the Support Vector Machine (SVM) to operate in a multiclass setting. To address this challenge, the One-Versus-The-Rest (OvR) strategy (137), illustrated in Fig. 4.5 is adopted. In this approach, **K** separate SVM models are constructed, and each **k-th** model is trained by treating data from class $\mathbf{C_k}$ as positive examples. The remaining data from the K−**1** classes are treated as negative examples.

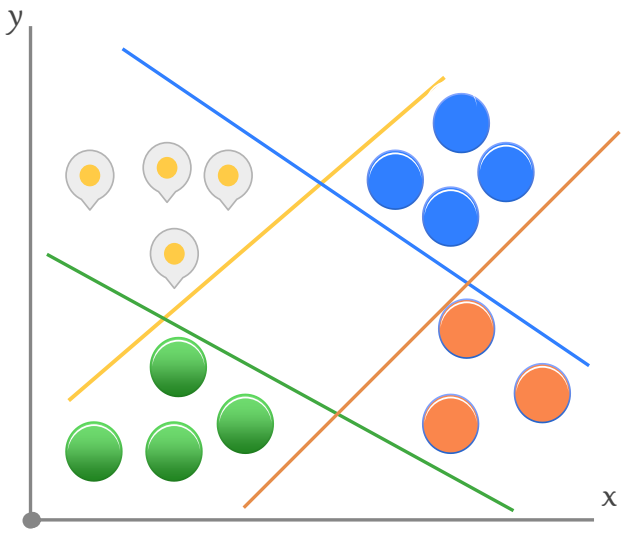


Fig. 4.5 The One-Versus-The-Rest (OvR) strategy of 4 classes

In the case of 4 classes, the number of SVMs required for a multiclass problem depends on the strategy used, for example:

1. **One-vs-Rest (OvR):**
    - For **K** classes, OvR constructs **K** binary classifiers.
    - For 4 classes, this results in 4 SVMs, where each SVM separates one class from the rest (e.g., Class 1 vs. {2, 3, 4}, Class 2 vs. {1, 3, 4}, and so on).
2. **One-vs-One (OvO):**
    - For **K** classes, OvO constructs $\frac{K\,(K-1)}{2}$ binary classifiers.
    - For 4 classes, this results in $\frac{4\,(4-1)}{2} = 6$ SVMs, where each SVM distinguishes between a pair of classes (e.g., Class 1 vs. Class 2, Class 1 vs. Class 3, etc.).

Thus:

- In OvR, **4** SVMs are required.
- In OvO, **6** SVMs are required.

Despite its widespread use and effectiveness, this method can occasionally produce inconsistent results, such as assigning an input to multiple classes simultaneously. This issue arises when the decision boundaries of individual classifiers overlap, leading to conflicts in classification outcomes (138). To address this, additional conflict resolution strategies are employed to ensure consistent predictions for new inputs. The following strategies are commonly used:

1. **Winner-Takes-All (WTA):**
    - The class with the highest decision score (output value) is selected as the final prediction.
    - This strategy resolves conflicts by prioritising the classifier with the strongest confidence level.
2. **Decision Thresholding:**
    - A threshold is applied to each classifier's decision score. If no class exceeds the threshold, the prediction can be flagged as uncertain, prompting additional steps such as rejection or manual review.
3. **Tie-Breaking Rules:**
    - In cases where multiple classifiers produce similar scores, predefined rules are used to break ties. For example, priority can be given to a specific class based on domain knowledge.
4. **Voting Schemes:**
    - In ensemble or aggregated approaches, predictions from multiple classifiers are combined using majority voting. The class receiving the most votes is chosen as the final output.
5. **Post-Processing Optimisation:**
    - Methods such as SoftMax normalisation are applied to the decision scores to ensure that the probabilities sum to 1. This helps to assign inputs consistently to a single class.
6. **Hierarchical Classification:**
    - Classes are organised into a hierarchical structure, and decisions are made at multiple levels. This strategy reduces the risk of conflicts by progressively narrowing the classification scope.

Within the scope of this thesis, adopting Winner-Takes-All (WTA) effectively resolves conflicts. This method is particularly suitable for resource-constrained environments, as it maintains computational efficiency while ensuring consistent and reliable predictions (139).

Support Vector Machines (SVMs) are fundamentally deterministic classifiers. They determine a hard margin hyperplane in the feature space when the algorithm finds the optimal one that divides the data points into distinct classes with no misclassification. Alternatively, a soft margin is used when the data is not perfectly separable, allowing for errors and overlap (18). For instance, consider a dataset with two classes represented as blue and red dots:

- If all blue dots are on one side of the hyperplane and all red dots are on the other side, a **hard margin** can perfectly separate the data.
- If some blue and red dots overlap or are misclassified due to noise or measurement errors, a **soft margin** would allow the SVM to find the best trade-off between a wide margin and acceptable errors.

Once the hyperplane is calculated, predictions are made deterministically by evaluating which side of the hyperplane a given data point lies on. This process ensures that the SVM produces a clear and fixed decision boundary for classification tasks.

The outputs of an SVM are raw scores, such as the distance of a point from the hyperplane. These scores do not directly represent probabilities; instead, a data point is assigned to a class based on its relative position to the hyperplane (30). Additionally, for a given dataset and set of hyperparameters, the SVM will always produce the same output, underscoring its deterministic nature.

4.4 Feature Reduction Process

In modern machine learning, representing data in an effective and compact form is crucial for optimising model performance (140). Embedding techniques, such as transforming high-dimensional data into lower-dimensional spaces, are widely employed to achieve this goal (44). Dimensionality reduction techniques improve computational efficiency by simplifying the representation of complex datasets (141–143).

Among the various approaches to embedding, feature reduction emerges as a prominent and essential process (144–147). Unlike general embedding methods that may prioritise interpretability or visualisation (148), feature reduction focuses on optimising the extraction

process. This directly impacts subsequent stages in vision systems by retaining the most informative components representing the patterns and images (149). Selecting the most relevant features enhances the performance of machine learning classifiers, manages overfitting, and ensures scalability, specifically when dealing with high-dimensional datasets (52,150,151).

In this study, we propose an innovative feature reduction methodology to enhance the adaptability and performance of machine learning models for real-time embedded vision systems. Building on the principles of dimensionality reduction, our approach combines complexity-based selection (15) with Correlation-Based Feature Selection (CFS) to optimise the feature set for resource-constrained environments. By reducing the initial Grey Level Co-occurrence Matrix (GLCM) feature set from 24 to 5, the proposed method achieves a remarkable 79.2% decrease in features while maintaining an accuracy of 84.76%. This streamlined feature set minimises computational and memory demands ensuring adaptability and effectiveness, making it particularly well-suited for real-time applications.

The proposed approach demonstrates its impact across various metrics, which are discussed in detail within the next chapter. These metrics include a 72.45% decrease in processing time and a 50% reduction in model size, highlighting its effectiveness for deployment in embedded systems and real-time environments.

4.4.1 Complexity-Based Selection

The complexity study investigates the relationship between an algorithm's execution time and the input size n (152). The input value represents the dimensions of the Co-occurrence matrix. This analysis evaluates the algorithm's performance as the input size increases, enabling an understanding of its efficiency and scalability. Table 4.3 presents all characteristics delivered by GLCM, categorised by their time complexity (15), organised into five levels.

Considering the lowest complexity, (15) selected the 10 features from the first group to construct a novel classifier. The researchers conducted an accuracy study, labelled FS3, which demonstrated performance comparable to FS1, the feature set incorporating all 24 original GLCM features.

Table 4.3 (15) shows that theta notation $\Theta(n)$ is utilised, indicating that the algorithm's time complexity is symmetrically bounded above and below by the input n, thereby accurately

limiting the execution time. The algorithm's performance increases linearly with input size (153), making it a reliable measure of efficiency in scenarios where performance scales with input size.

It is essential to differentiate the theta notation $\Theta(n)$ from the Big O notation. $\Theta(n)$ provides an exact bound, indicating that the algorithm's growth rate is precisely constrained above and below by n. In contrast, Big O notation specifies an upper bound on time complexity, ensuring that the algorithm's growth rate does not exceed a specific threshold (153). Big O is crucial for understanding the worst-case scenario and ensuring that the algorithm's performance remains within acceptable parameters.

In the current situation, where accurate performance characterisation is essential, particularly for optimisation in embedded vision platforms, $\Theta(n)$ is more appropriate. It offers a precise and comprehensive understanding of the algorithm's effectiveness by capturing its specific behaviour in various scenarios.

Although $\Theta(n^2+n)$ conventionally simplifies to $\Theta(n^2)$, the '$+n$' term is retained to more precisely express the slight variation in complexity. This is especially relevant in embedded systems where minor differences can greatly influence performance (15). Moreover, preserving smaller terms like '$+n$' helps better distinguish and group features based on their computational demands, facilitating system optimisation for high-demand applications.

Table 4.3 List of GLCM's features grouped by time complexity (15)

	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	$\Theta\ (n^2 + 4\ n - 2)$	Sum Variance
05	$\Theta\ (2n^2)$	Sum of Squares, IMC I, IMC II, MCC, Cluster Tendency, Cluster Shade, Cluster Prominence

4.4.2 Correlation-based Feature Selection (CFS)

Our methodology aims to reduce the dimensionality of the feature vector identified in (15) to less than 10 features by merging complexity-based selection with Correlation-based Feature Selection (CFS). Selecting the most pertinent features (55) with minimal complexity decreases computing expenses for both feature extraction and classification, while maintaining the classifier's highest accuracy through effective hyperparameter tuning.

Compact classifiers require lower computer resources, resulting in shorter processing times. This is essential for real-time applications, thereby unlocking the complete potential of integrated systems in diverse and challenging environments (154).

The use of Correlation-based Feature Selection (CFS) for this task is motivated by its ability to efficiently identify features that are most pertinent to the target variable while reducing redundancy among them (54). In contrast to other feature selection techniques that primarily emphasise relevance (79,155–157), CFS assesses both the correlation of each feature with the target variable and the inter-correlation among the features themselves. This guarantees that each chosen feature contributes distinct information, avoiding the inclusion of features that provide redundant insights.

During the selection phase, the algorithm evaluates multiple combinations and sets of features, ultimately identified five features that provide the greatest discriminative strength with minimal duplication. Although multiple combinations and feature sets were deliberately evaluated, none demonstrated a higher level of precision and efficiency compared to the final set utilised in S3. The dual strategy of optimising information while reducing redundancy provides the key for improving model performance and efficiency (158), as evidenced by successful outputs in prior studies (159),(160),(58).

In the complexity investigation method used by (15), 10 features were initially selected. However, after applying Correlation-Based Feature Selection (CFS) in our research, this was refined to 5 features. Table 4.4 compares the original vector from (15) with the features retained by CFS. The classifier's accuracy, utilising the feature set derived from complexity analysis and Correlation-based Feature Selection (CFS) is presented in Table 5.1, under column S3. The first column in the table represents the target classes, labelled as C1, C2, C3, and C4. These classes correspond to the specific labels that the classification system seeks to distinguish. Fig. 4.4 provides samples from these four classes.

4.5 Training And Fine Tuning

The deployable model is an SVM classifier trained on the CSV database (16) with a smaller set of features. Prior to the training operation, each set of 150 vectors from the four classes is randomly divided into two folds to facilitate validation process (97). This is based on the following standards: One hundred lines correspond to 67%, are allocated for training, with fifty samples from each class reserved for testing (15).

The classifier's training utilised optimal hyperparameter values, namely the kernel, C, and gamma, which were carefully refined through a rigorous process known as grid-search (30,93) , a systematic method that evaluates diverse parameter combinations to determine the most effective settings for the classifier (17).

- The **kernel** specifies the type of function used to transform the input data into a higher-dimensional space, enabling the classifier to effectively handle non-linear relationships. Common kernel types include linear, polynomial, radial basis function (RBF), and sigmoid.
- The **C** parameter, also known as the regularisation parameter, controls the trade-off between minimising training error and maintaining a margin that generalises well to unseen data. A smaller value of **C** allows for a wider margin but may tolerate some misclassifications, while a larger value of **C** focuses on minimising training errors, which may lead to overfitting.
- The **gamma** parameter defines the influence of individual training example. A small value of gamma means that each training example has a far-reaching influence, creating a smoother decision boundary. In contrast, a large value makes the influence more localised, capturing finer details but risking overfitting.

Table 4.4 GLCM's selected low complexity features description (15), and features 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 (12)
2	Contrast	Measures the intensity contrast between a pixel and its neighbour. 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 (13)
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 (14)
10	Sum Mean	The sum of the mean of the rows and columns of the GLCM .	✓	

The fine-tuning process identified the linear kernel for its simplicity and efficacy in managing linearly separable data, which aligns with the nature of the used features. Recent studies support this selection, demonstrating the linear kernel's effectiveness even in high-dimensional, linearly separable data and real-time applications. If offers an optimal balance between computational efficacy and generalisation performance. (161) highlighted the success of the linear kernel in determining remaining useful life in real-time applications, emphasising its computational efficiency and generalisation capability. Similarly, (69) demonstrated the reliability of a linear kernel in on-chip ageing estimation, further validating its suitability for this application.

The hyperparameter configurations were essential for maintaining high precision in our lightweight classifier, despite the reduction in feature set size (50). By carefully adjusting these parameters, we optimised the classifier's performance, ensuring consistent and efficient outputs appropriate for real-time EVA (93). The integration of advanced feature selection and meticulous hyperparameter optimisation highlights the robustness and effectiveness of our enhanced classification system (162).

4.6 Conclusion

This chapter outlined the methodological framework established to address the research objectives. It began by presenting the research design, highlighting the conceptual foundations and theoretical principles supporting the study. The methodology employed a systematic approach to overcoming computational challenges. It emphasises preprocessing steps, feature reduction strategies, and classification techniques for resource-constrained environments.

Key contributions of this chapter include the use of Gray Level Co-occurrence Matrix (GLCM) for feature extraction, the integration of complexity-based and correlation-based feature selection methods, and the adoption of a Support Vector Machine (SVM) classifier optimised through exhaustive hyperparameter tuning. These techniques collectively aimed to ensure an effective balance between computational efficiency and classification accuracy. Furthermore, we thoroughly documented the experimental setup and deployment on an embedded system, which laid the groundwork for assessing the system's performance in real-time applications.

The next chapter will evaluate the effectiveness of the proposed methodology. Metrics such as accuracy, processing time, and memory usage will be assessed through empirical analysis. This evaluation will validate the system's adaptability and robustness, thereby demonstrating its ability to address the challenges identified in earlier chapters.

Chapter 5

Performance Assessment and Validation

5.1 Introduction

The primary focus of this chapter is to provide a comprehensive performance evaluation and validation of the proposed classification system. This classifier is systematically assessed across various indicators, including computational time, memory usage, storage requirements, and accuracy. These metrics are critical in determining the suitability of the system for real-time applications, as delays in processing, limitations in computational and memory resources can significantly affect its efficiency.

Our evaluation methodology involves rigorous testing on the Raspberry Pi platform, which represents severe resource-constrained settings and simulates real-world operating conditions. Additionally, we compare the performance of our system against established state-of-the-art methods to highlight its advantages and challenges. This comparative analysis provides insights into how the developed system performs relative to existing solutions, offering a clear perspective on its practical applicability.

Furthermore, we examine the implications of the novel approach by evaluating how its observed performance meets the strict requirements of embedded systems. We investigate the trade-off between classification accuracy, computational and memory demands, which is an essential consideration for the deployment of vision systems in resource-constrained settings. This analysis aims to provide a nuanced understanding of the operational efficacy of the system through inspecting these trade-offs.

In summary, this chapter conducts an in-depth analysis and a review of our improved classification system in terms of performance metrics and its suitability for real-time embedded vision applications. The insights gained from this assessment will be invaluable for future developments in this domain, contributing to the design of efficient and effective EVS solutions capable of operating within the constraints of embedded platforms.

5.2 Dataset and Data Acquisition

The dataset associated with this research is the Crosswalk Dataset (16), which is publicly accessible for exploration and use. The classification targets four classes: crosswalk viewed from the font (C1), from the left half-lane (C2), from the right half-lane (C3), and non-crosswalk images (C4), as illustrated in Fig. 4.4. Each class contains 150 instances, resulting in a total of 600 samples. The images were acquired at a resolution of 1280x720 from videos, recorded at 30 frames per

second in Fortaleza, Ceará, Brazil, during daytime hours. The images have been named using the format class_p1_imageNumber.jpg (e.g., c4_p1_1.jpg).

The dataset is supplemented by the file 10_FEATURES_M17_CM6b_TH199.csv. Fig. 5.1 illustrates the first ten lines. This dataset is integrated into the proposed classification system to support the training and prediction processes. The data employed on the embedded platform (15) originates from threshold images with T = 199, a decimation factor M = 17, and 10 GLCM features identified through complexity analysis, which are summarised in Table 4.4, under column description. The data is encoded as float64 using 64 bits, while the co-occurrence matrix reflects only 6 bits of greyscale values of images.

	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

Fig. 5.1 Snapshot of the first five lines from the CSV Dataset (16).

5.3 Experimental and Deployment Setup

5.3.1 Development environment

The training, testing, and evaluation of the classification system were initially conducted on a machine equipped with an i5-11320H 3.20 GHz Central Processing Unit (CPU) featuring 4 cores, 8 MB Intel® Smart Cache, 16 GB of Random Access Memory (RAM), running on Windows 10 Pro. This process was completed before deploying the model in the embedded system's physical environment.

The development environment included Anaconda, a platform for managing Python packages and environments, and the JUPYTER Notebook IDE, which provided an interactive web-based interface for coding and testing. We utilised Python 3.9 as the programming language. For machine learning, we selected the scikit-learn library, a free and open-source Python package.

In scikit-learn, the Support Vector Machine (SVM) implementation for multiclass problems adopts the one-vs-rest (OvR) strategy by default, though it can be configured to use other strategies like one-vs-one (OvO) (163). In the OvR approach, Scikit-learn resolves conflicts using the Winner-Takes-All (WTA) strategy. Each binary classifier outputs a decision score for its

respective class. The class with the highest decision score (confidence level) is selected as the final prediction. This method prioritises the classifier with the strongest confidence in its decision, ensuring consistent outcomes even in cases where decision boundaries overlap.

5.3.2 Deployment Setup

We deployed the classifier on a Raspberry Pi 4 Model B board as hosting hardware. Fig. 5.2 visualises the target single-board system. This device features a Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC operating at 1.8GHz, along with 4 GB LPDDR4-3200 SDRAM. We set up the operating system and necessary software onto a MicroSD card: the Raspberry Pi OS with the desktop environment and suggested applications.

We chose The Raspberry Pi Model B, because of its affordability and adaptability. The Model B's compatibility with various applications and peripherals, together with its strong community support (164–166), rendered it an excellent selection for our project. Moreover, its minimal power consumption aligns well with the demands of an efficient embedded system. The choice of Raspberry Pi enhances portability and simplifies integration with other components (111), which are important to the effective implementation of our solution in diverse scenarios.

The deployment environment included the Thonny IDE and Python 3.9, both pre-installed on the Raspberry Pi OS (167). We preferred the scikit-learn (sklearn) library for learning, while it is a free and open-source machine learning package for Python that provides an extensive array of classification techniques, including support vector machines (168). Its compatibility with Python's numerical and scientific libraries, as well as its comprehensive documentation and an active user community, rendered it an excellent selection for this project (169).

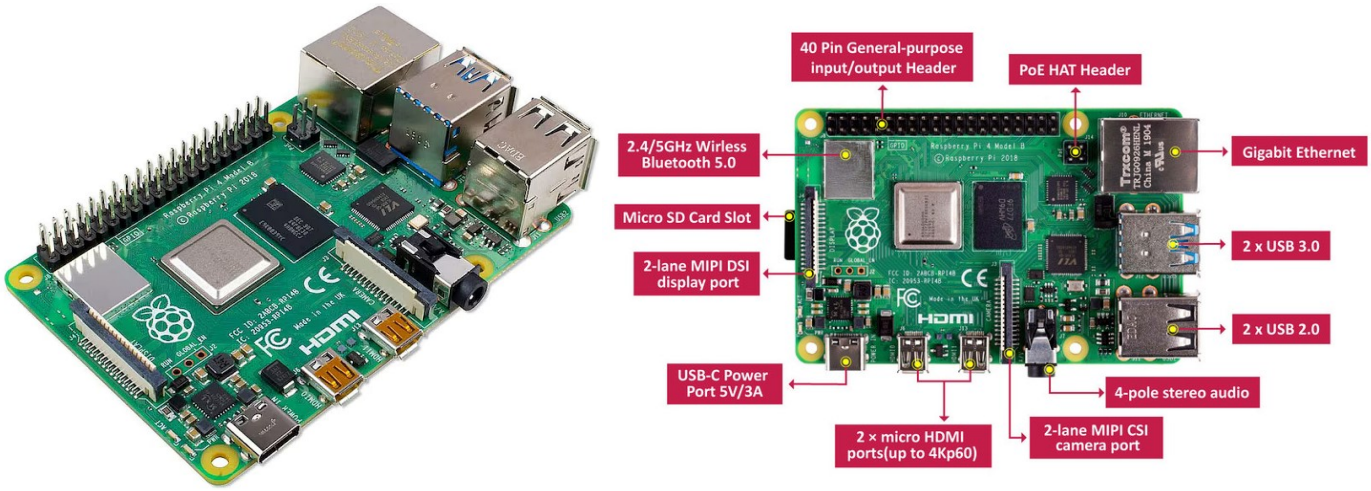


Fig. 5.2 Raspberry Pi 4 Model B board

5.4 Performance Metrics

The assessment of any classification system requires a clear understanding of the performance metrics that quantify its efficiency and effectiveness, alongside consideration of the specific context of deployment and purpose (96,100,170). In this section, we introduce the key metrics used to evaluate the proposed system, focusing on aspects critical to real-time embedded vision applications. These metrics include computational time, memory usage, storage requirements, and classification accuracy, each reflecting a distinct dimension of the system's operational performance.

Computational time metrics measure the speed at which the system processes data, ensuring timely decision-making in real-world scenarios. Memory usage examines the system's ability to operate within the strict memory constraints of embedded platforms, while storage requirements assess the compactness of the deployed model. Lastly, classification accuracy evaluates the system's ability to correctly identify and categorise input data, balancing precision with the resource limitations characteristic in embedded environments.

By systematically analysing these metrics, this section provides a detailed examination of the proposed system's performance under various conditions, paving the way for its validation and comparison against existing solutions.

5.4.1 Computational Time Investigation

This metric prioritises pattern recognition process, targeting its computation duration, precisely the feature extraction stage driven by GLCM (12–14).

The analysis of the results presented in Table 5.1 reveals that the classifier's average accuracy fell below 90%. Therefore, this proposed method did not achieve optimal performance for this pattern recognition task. However, the reduction in the feature vector positively influenced the computational time, as illustrated in Table 5.2.

Table 5.1 Classifiers accuracy for each feature set

Target Classes Fig. 4.4	[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%

Although our method of a combined selection did not maintain the same accuracy as (15), it is essential to evaluate the trade-offs between accuracy and computing efficiency. The reduced feature set significantly lowers the computational cost, resulting in accelerated processing times and less use of resources (50). This advantage is particularly valuable in real-time applications and scenarios with constrained processing resources (171).

Moreover, utilising a smaller feature set optimises the model, making it more concise, comprehensible, and manageable. The equilibrium between accuracy and efficiency must be customised to the particular demands and limitations of the intended application (162).

Table 5.2 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

Table 5.2 illustrates the computational time measurement for each feature set across the different phases. Starting by 24 features, then reducing to 10 features (S1 to S2) (15), and finally concluding with 5 features (S3), as established by our research, substantially decreases the durations for training, testing, and one sample prediction. This highlights that feature selection techniques, including both complexity-based and the combination of complexity and correlation-based approaches, significantly enhance computing efficiency.

The most substantial time savings occur during the training phase. Meaningful enhancements are also evident in the testing and prediction stages. Fig. 5.3 visually demonstrates the advantages of the S3 feature set, consisting of just five features. The prediction times for S3 are the shortest among all stages on both platforms, workstation and Raspberry Pi, registering at 3.44 milliseconds, and 6.53 milliseconds, respectively.

The observed reduction in prediction time demonstrates significant computational speed gains, achieved through the integration of complexity and correlation-based feature selection. Minimising the size of features not only reduces computational complexity and resource utilisation, but also enhances model performance by lowering the probability of overfitting and enhancing data interpretability (15). These improvements are essential to ensuring optimal efficiency in real-time applications and settings with constrained computational resources (171).

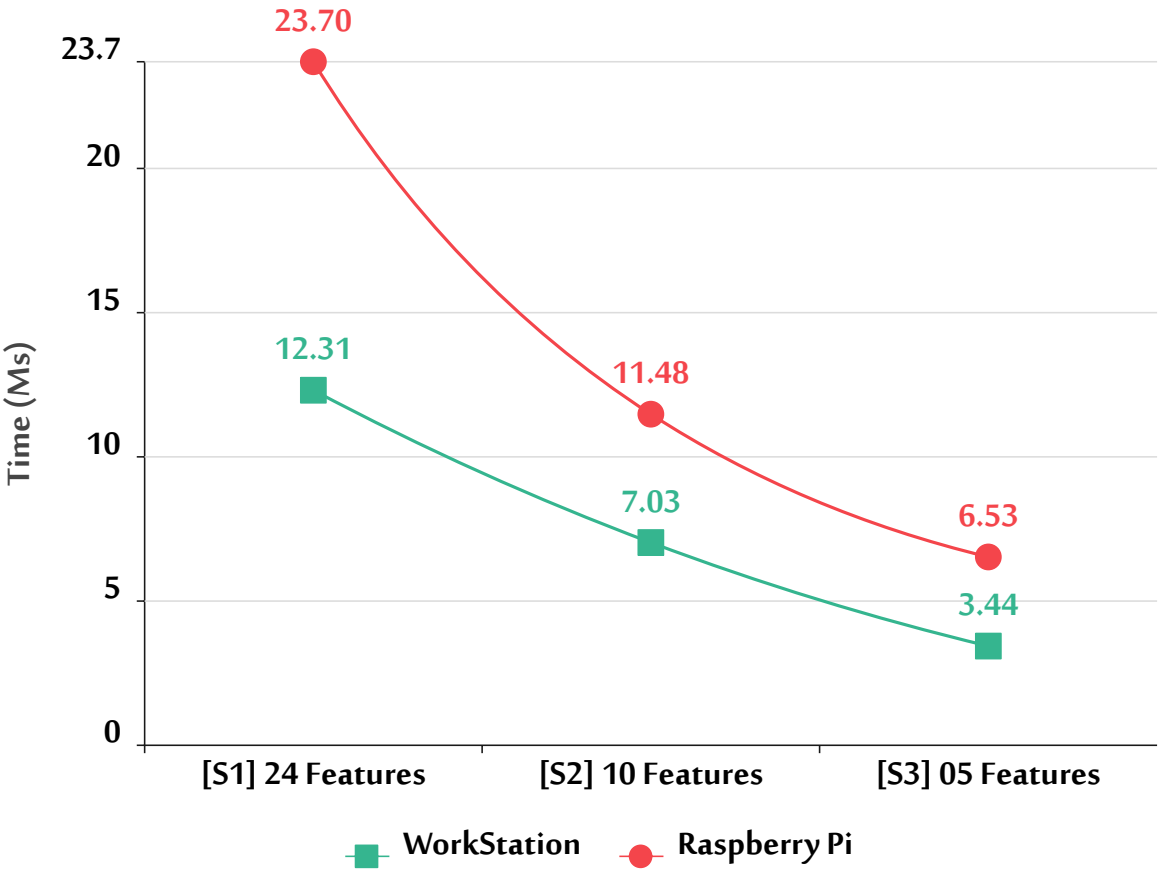


Fig. 5.3 One instance prediction consuming time in milliseconds

Table 5.3 examines the time required for the prediction of a single sample by the three implemented SVM-trained models for Raspberry Pi, based on the feature sets analysed in this study. It demonstrates the enhancements in efficiency, gained by reducing the number of

characteristics utilised in the classifier. The prediction time for the full feature set (24 features) is 23.70 milliseconds. Reducing the feature set to ten, complexity-based features shows a significant improvement, with a 51.52% reduction in prediction time compared to the baseline method. Further reduction to five features, using complexity-and-correlation-based selection, results in an even greater gain, achieving a 72.45% reduction in prediction time.

This performance evaluation illustrates that, despite constrained computer resources, efficient real-time predictions can be achieved by optimising the feature set utilised in the classifier. The Raspberry Pi exemplifies the feasibility of implementing machine learning models on embedded systems.

Table 5.3 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

The computational effort rises rapidly with larger datasets given the huge data volume and the complex nature of processing (162). These demands are significantly lowered by reducing the number of features, which increases the scalability and efficiency of testing, training procedures (171).

As the size of the feature set decreases, the computational cost correspondingly reduces due to the smaller amount of data points requiring processing. The total number of processed values is computed by multiplying the number of samples n by the number of features m. The dataset employed in the present study comprises 600 samples, as outlined in the previous chapter, Section 3.5. Employing 24 features (S1) results in 14,400 processed values, which is subsequently reduced to 6,000 processed values with 10 features (S2), and further lowered to 3,000 processed values with 5 features (S3).

The study conducted by (50), approves the significance of decreasing the number of features within a dataset. It demonstrates that feature selection is essential for enhancing machine

learning model effectiveness by decreasing computational complexity. The study illustrates that a limited number of features accelerates both training and inference processes while alleviating the challenges associated with high-dimensional spaces. This results in improved generalisation and more adaptable models that demand less memory and computational power. These advantages are particularly valuable for managing large datasets, which are characterised by a substantial number of samples, therefore demand additional computational effort, especially when setting up models on resource-constrained devices.

5.4.2 Memory Usage Investigation

Table 5.4 enumerates the peak memory use and the increments in memory consumption for every feature set through each stage. The metrics were obtained by observing memory usage during the classifier system's operation with the "memory_profiler" package in Python, which produces a comprehensive report on memory usage. This report offers valuable insights into the memory efficiency across different feature selection approaches during the training, testing, and prediction phases.

Peak memory refers to the maximum volume of memory utilised at any point during the execution, quantified in mebibytes (MiB). Higher peak memory consumption indicates that the process demands a considerable volume of memory at its peak, potentially impacting overall system performance and allocated resources. Increment, on the other hand, denotes the net increase in memory allocation from the beginning to the ending of a phase. Fig. 5.4 visually illustrates the increments in memory allocation for each feature set. The increment value, also measured in MiB, highlights the additional memory required to complete the phase. This statistic is essential for comprehending the dynamic memory requirements of the process (15).

During the training period, the feature set S1 (Full Features) exhibits the highest peak memory consumption and increase, reflecting the significant memory demand associated with managing a larger number of features. S2 (Complexity-Based Features) demonstrates a notable reduction in both peak memory and increment compared to S1, indicating improved efficiency. Finally, S3 (Complexity and Correlation-Based Features) demonstrates the lowest peak memory and increment, underscoring the substantial efficiency gains achieved through further feature set reduction.

During the testing phase, S1 consistently exhibits the highest peak memory and increment values, continuing its established behaviour of high memory demand. S2 significantly reduces

Table 5.4 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 values compared to S1, demonstrating smoother memory management. S3, on the other hand, achieves the lowest values, indicating optimal memory efficiency throughout the testing phase. This trend continues during the prediction phase, where S1 records the highest memory usage, both at its peak and increment. While S2 improves upon S1, it still consumes more memory than S3. The latter consistently displays the lowest peak memory and increment values, highlighting the benefits of integrating complexity and correlation-based feature selection for minimising memory usage.

Table 5.4 clearly illustrates the advantages of reducing the size of the feature set in terms of memory allocation. As the number of features decreases, both peak memory usage and memory increment decline across all phases. This decrease is particularly significant with the S3 feature set, which involves complexity and correlation-based selection to achieve the lowest memory allocation.

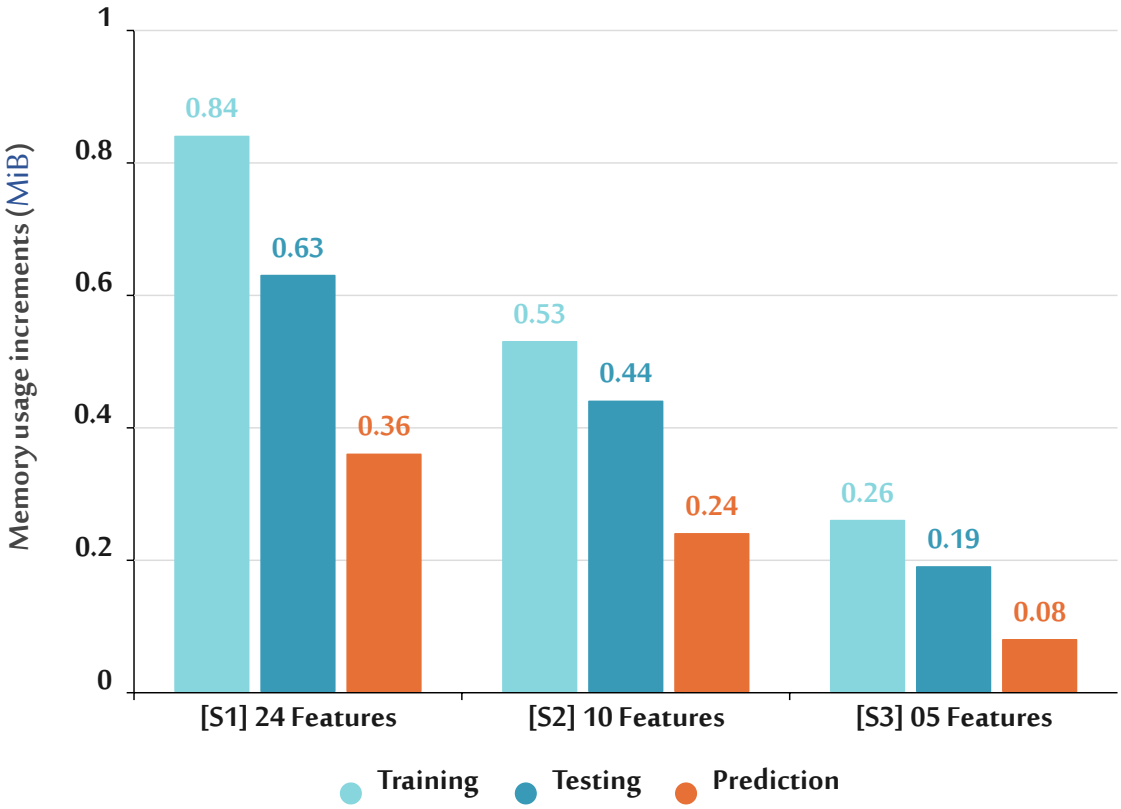


Fig. 5.4 Increments in memory usage at different stages of the feature sets, measured in MiB

In resource-limited contexts like embedded systems, even minor reductions in memory usage can significantly improve overall system performance (111). The percentage decreases in memory allocation achieved through feature reduction are significant. A 69.0% decrease in memory usage during training, and up to a 77.8% decrease during prediction, significantly impacts the practicality and efficiency of implementing machine learning models in these settings.

This demonstrates the applicability of our approach, where fewer features result in significant gains in memory efficiency, making the model more appropriate for implementation in environments with constrained memory space. The benefits of our technique are more evident with larger datasets, when the requirements for memory and processing resources are considerably higher.

Furthermore, although the peak memory usage remains relatively constant, the decrease in incremental memory consumption is highly advantageous for systems where memory optimisation is critical. This is particularly beneficial when tasks such as image acquisition and feature extraction are decoupled from prediction. Such a methodology aligns with scalable

architectures (114), where distributing of tasks between low-resource (edge devices) and high-resource (cloud servers) elements leads to considerable resource efficiency. Adopting a comparable method restricts the computational and memory requirements at various phases of the model's workflow, therefore enhancing the system's efficiency in resource-limited settings.

While the reduction of features resulted in a slight decline in accuracy, the trade-off is justified by important improvements in computational efficiency and resource management (50). The feasibility of implementing a more compact models is important to real-time applications and situations involving extensive datasets. These findings underscore the significance of our feature selection methodology in enhancing machine learning classification models for implementation on resource-limited devices.

Table 5.5 presents an analysis of the vector lengths for each feature set employed in our study. The memory size of the feature vector is a critical measure of the memory footprint and overall efficiency of machine learning models, including feature extraction, classification, and model deployment.

The S3 feature set, which integrates complexity and correlation-based selection and utilising only 5 features, lowers the feature vector length to 320 bits. This signifies an important reduction of nearly 79.2% compared to the S1 feature set. The compact feature vector minimises memory allocation and enhances model performance by allowing quicker training, testing, and prediction phases. The reduction in feature vector length directly results in reduced memory demands, which is particularly beneficial for implementation in resource-limited settings (171).

Table 5.5 Features' vector length for 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 compact vector offers additional benefits, it enhances data transport in distributed computing environments, such as cloud-based applications or edge computing. Smaller feature vectors necessitate less bandwidth for data transfer between storage and processing units. This

can result in faster data transmission and lower latency, thereby enhancing the model's real-time processing abilities upon possible use in such settings.

5.4.3 Storage Investigation

In addition to examining memory utilisation across the various stages, it is essential to evaluate the size of the deployable trained SVM model, particularly for implementation in storage-constrained contexts such as embedded systems and IoT devices (111). Table 5.6 displays the size of the deployable trained SVM model corresponding to each feature set.

Table 5.6 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

The full feature set S1, including 24 features, produces the largest model size of 56 KB. This is comprehensible due to the increased complexity and the greater volume of data necessary for conveying the full feature set. The S2 feature set, which employs complexity-based selection to limit the features to 10, results in a model size reduction to 36 KB. This reduction illustrates the advantage of eliminating less important components, reaching an approximate 35.70% decrease in the model's overall size, while maintaining performance integrity. The S3 feature set, which incorporates complexity and correlation-based selection to include only 5 features, leads to an optimal model size of 28 KB. This demonstrates a major decrease in model size of 50% relative to S1, making it exceptionally appropriate for deployment in limited storage settings.

The size of the deployable model strongly impacts the feasibility of using the SVM model in diverse applications, especially those necessitating compact and efficient solutions. The decrease in the number of features enhances computational efficiency and noticeably decreases the model's storage requirements.

5.4.4 Accuracy Investigation

The novel feature selection method introduced in this study aims to achieve an optimal balance between accuracy and computational demands, it maintains an average performance of 84.76%. The accuracy dropped by an absolute difference of 5.7%, representing a 6.31% relative decrease compared to the baseline method's accuracy of 90.46% (full features subset). This a reasonable trade-off which is appropriate within the field of embedded systems, where computational and power resources are key limitations. Prior studies have shown that in these contexts, a minor reduction in accuracy, generally between 2% and 10%, may be tolerable if it leads to substantial improvements in efficiency and decreased computational demands.

This is evident in applications such as energy-efficient signal processing and wearable health monitoring systems (171,172). Moreover, in the broader field of machine learning, it is a widely recognised practice to lower model complexity to address computational requirements. This approach is also prevalent in deep learning, where precision reductions of approximately 2% to 7% are often considered acceptable to ensure computational feasibility (81). These trade-offs are vital for ensuring that the models function efficiently within the constrained resources of real-time applications.

An accuracy of 84.76% is very applicable for a variety of real-world applications, especially those that prioritise computational efficiency rather than absolute precision. Research in embedded deep learning demonstrates that computational accuracy trade-offs, typically between 5% and 10%, are used for maintaining performance within the strict limitations of embedded systems (173). In real-time vision applications, it is fundamental to attain a balance between precision and processing speed, as minor accuracy decreases of up to 5% are generally negated. The results highlight the significance of our feature selection strategy due to the enhanced responsiveness and accuracy of the system (22).

In sophisticated deep learning systems, it is acceptable to balance the number of layers in the model (depth) against computational feasibility, allowing for tolerable accuracy reductions of 3% to 7% to enhance performance for real-world applications (72). These principles align closely with our methodology, where the accuracy trade-off resulting from feature reduction facilitates speed optimisation in resource-limited settings.

Our methodology is particularly relevant in embedded systems, where even small changes in computational loads can considerably affect overall performance. By reducing model complexity

while retaining valuable features, we ensure that the model remains both practical and efficient, capable of functioning within the rigorous limitations of embedded systems.

Research in platform-based design highlights the significance of these trade-offs, demonstrating that little reductions in accuracy are typically compensated by the advantages of lowered complexity and enhanced system efficiency (174), (175). The 84.76% accuracy attained by S3 is both adequate and strategically optimal for the specified real-world applications, where achieving a balance between performance and computational efficiency is imperative.

5.5 Overall Performance and Validation

The findings of our investigation demonstrate important facts regarding the trade-offs among accuracy, computational time, memory utilisation, and model size when implementing various feature selection techniques. Table 5.7 presents a comprehensive evaluation of three classification method: Baseline Method (Full Features), Complexity-Based Method, and Complexity followed by Correlation-Based Method, across several performance criteria.

The comparative analysis of feature selection techniques demonstrates a balancing between retaining high accuracy and optimising computational effectiveness (21,110,111). The Complexity Based Method (15) achieves a marginal accuracy improvement of 0.13%, while significantly reducing computational time by 42.89% on standard computers and 51.52% on embedded systems. It also leads to a small memory savings about 0.38%, and a substantial model size reduction of 35.71%. This approach is particularly advantageous for applications that require both performance and efficiency, such as real-time EVS.

Meanwhile, the Complexity and Correlation Based Method (our proposed method) exhibits superior computational efficiency, achieving a notable 72.06% decrease in computational time on work station, a 72.45% decreasing on Raspberry Pi, a 0.77% decline in memory usage, and a 50.00% reduction in model size.

The increment in memory usage during prediction decrease from 0.36 MiB for S1 to 0.08 MiB for S3, indicating a significant 77.78% enhancement. Similarly, the increment in memory usage during training decreases from 0.84 MiB for S1 to 0.26 MiB for S3, indicating a 69.05% reduction. These metrics show the considerable memory efficiency attained by minimising the number of features, thereby enhancing the system's adaptability for resource-limited contexts and real-time applications. Although peak memory appears consistent across models, the incremental memory

improvements during training and prediction are essential for embedded systems, where resource optimisation is necessary.

The impact of this method becomes particularly evident in scenarios involving high memory increments, even when peak memory usage remains relatively consistent across all subsets. This impact is achieved by distributing operations such as image acquisition, feature extraction and prediction. Task distribution enhances memory management and makes these incremental reductions very significant. This principle is illustrated by the technique applied in the study presented in (114), in which distribution of tasks notably enhanced resource efficiency.

Although there is a 5.7% drop in accuracy, this method is highly suitable for applications prioritising speed and resource efficiency over absolute precision (22,72,81,173). This makes it optimal for scenarios such as real-time monitoring and immediate data processing under constrained conditions.

5.6 Broader Implications

The findings of this study provide significant contributions to the field of embedded vision systems. The notable reduction in computational time enhances the feasibility of deploying machine learning models in real-time applications, meeting stringent processing time requirements. This advancement aligns with recent research emphasising the importance of efficient algorithms for embedded systems (9).

Moreover, the observed improvements in memory consumption and substantial decrease in model size are pivotal for extending the operational lifespan of battery-powered devices and reducing hardware demands. Such optimisations are central for the advancement of Tiny Machine Learning (TinyML) applications, enabling sophisticated machine learning tasks on low-power devices (176).

These enhancements not only contribute to the current capabilities of embedded vision systems but also set a precedent for future designs in resource-limited settings. By demonstrating that high-performance machine learning models can operate efficiently within stringent resource constraints, this study paves the way for more widespread adoption of intelligent systems in various applications, from mobile robotics to IoT devices (177).

In summary, the integration of complexity and correlation-based feature selection methods, as evidenced by the results of this study, offers a promising pathway for developing efficient,

Table 5.7 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%
Variation (%)		0.12 ↑	5.7 ↓		
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%

scalable, and sustainable embedded vision systems in resource-constrained environments.

5.7 Conclusion

This chapter has presented a comprehensive evaluation of the proposed classification system, rigorously analysing its performance across multiple fundamental dimensions. Through extensive testing on the Raspberry Pi platform, the study highlighted the system's adaptability to severe resource constraints, demonstrating its potential for real-world embedded real-time vision applications.

Key findings reveal a significant reduction in computational time and memory usage achieved by leveraging complexity and correlation-based feature selection. These optimisations notably enhanced the model's scalability and operational efficiency, paving the way for responsive and real-time resource-limited deployments without compromising essential functionality. The feature reduction strategy proved instrumental, achieving up to a 72.45% decrease in computational time on embedded systems, all while maintaining an acceptable accuracy level for real-time applications.

The comparative analysis emphasised the importance of trade-offs in balancing accuracy with computational efficiency. While the proposed method exhibited a minor 5.7% reduction in classification accuracy, this trade-off is justified by the significant gains in speed and resource management. These attributes make the method particularly suitable for time-sensitive applications, such as real-time monitoring and immediate data processing in constrained environments.

Moreover, the study has profound implications for the field of embedded vision systems. It sets a precedent for designing lightweight and efficient models that extend the operational lifespan of battery-powered devices, reduce hardware demands, and enable seamless integration into TinyML applications. By demonstrating the feasibility of deploying high-performance machine learning models on resource-limited platforms, this chapter translates theoretical advancements into practical real-world applications.

In conclusion, this chapter has established a strong foundation for the practical deployment of intelligent systems in embedded contexts, laying the groundwork for future innovations in real-time vision systems.

Conclusion and Future Works

This thesis has investigated the adaptability of computer vision and image processing algorithms within resource-constrained environments, with a focus on real-time embedded vision applications. By addressing the challenges associated with computational limitations, memory constraints, and energy efficiency, this research has bridged the gap between theoretical advancements and practical deployments.

The proposed classification system demonstrated significant improvements in efficiency, achieving reductions of up to 72.45% in computational time and substantial gains, including a reduction of the feature set by 79.16% (from 24 features to 5) and a 50% decrease in storage requirements, while maintaining high classification accuracy. These results, validated through rigorous testing on a Raspberry Pi platform, underscore the feasibility of deploying sophisticated machine learning models in embedded systems. Moreover, the integration of complexity and correlation-based feature selection has shown to be instrumental in balancing performance trade-offs, paving the way for scalable and efficient deployments in various fields, including robotics, healthcare, and IoT.

The insights gained from this research reinforce the potential of adaptive machine learning approaches while clearly highlighting their relevance in addressing global technological demands. By providing an efficient methodology for optimising resource usage without compromising accuracy, this thesis contributes to advancing the field of embedded vision systems, promoting innovation in both academic and industrial domains.

This research has achieved notable advancements in computational efficiency and scalability; however, a few challenges remain to inspire further exploration and refinement. A specific consideration is the observed trade-off of a 5.7% reduction in classification accuracy. This reduction is tolerable for many applications where small errors in classification or detection do not significantly impact overall effectiveness or outcomes, such as vehicle counting and pedestrian flow estimation for traffic optimization, gesture recognition in gaming systems, and automated identification of animal species. However, it may require careful attention in critical domains such as safety-critical systems, nuclear energy, and military and defence applications.

Additionally, the testing environment focused on a single-board computer platform (Raspberry Pi), which, while effective for validation, may narrow the scope of hardware

generalisation. Furthermore, external factors such as power stability, or environmental conditions were not extensively studied and may influence performance in real-world scenarios.

As the number of target classes increases, the computational demands of the system also grow. When employing the One-versus-the-Rest (OvR) strategy, the number of Support Vector Machines required increases linearly with the number of classes, despite it being an optimal strategy for multi-class classification. Exploring optimisation techniques or alternative classification strategies may help manage these demands effectively.

Lastly, this study focused on Support Vector Machines, offering promising results, but presenting an opportunity to extend these methods to other advanced classifiers and deep learning frameworks. Addressing these aspects will provide a more comprehensive understanding of the system's full potential and open pathways for broader applicability.

This thesis lays the groundwork for several promising avenues for future exploration and development. Future research should focus on enhancing feature selection techniques by incorporating hybrid approaches that combine statistical, neural, and heuristic-based methods, as well as dimensionality reduction techniques like Principal Component Analysis (PCA).

Expanding the application of the proposed approach to other classifiers, such as Decision Trees (DT), Random Forests (RF), and Deep Learning models (DL), would help evaluate its generalisability across various domains, including healthcare, finance, and safety-critical systems. Furthermore, real-world deployment and testing in industrial IoT devices and edge computing platforms, under varying environmental conditions like temperature fluctuations and network latency, would provide valuable insights into the system’s robustness.

Future research should also investigate adaptive resource management techniques that dynamically adjust algorithmic operations based on the available hardware capabilities and environmental constraints. This can involve real-time monitoring of power consumption and workload distribution to achieve optimal efficiency without sacrificing performance.

Exploring these directions in future research can refine and expand the methods proposed in this study, driving innovation and practical utility in resource-constrained environments. These advancements are expected to significantly enhance the performance, scalability, and societal impact of intelligent embedded vision systems, ensuring their broader acceptance, long-term sustainability, and relevance in an increasingly interconnected world.

References

1. Ikeuchi K. Computer Vision: A Reference Guide. 2nd ed. Springer International Publishing; 2021. https://link.springer.com/referencework/10.1007/978-3-030-63416-2
2. Roberts LG. Machine perception of three-dimensional solids. Massachusetts Institute of Technology; 1963.
3. Marr D. Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information. W.H. Freeman; 1982. https://books.google.dz/books?id=GcKywwEACAAJ
4. Davies ER. Computer Vision: Principles, Algorithms, Applications, Learning. 5th ed. Academic Press; 2017. https://www.lehmanns.de/shop/mathematik-informatik/42142411-9780128092842-computer-vision
5. Vijayalakshmi SR, Muruganand S. Embedded Vision: An Introduction. Dulles, VA: Mercury Learning & Information; 2019.
6. Zielke T. Is Artificial Intelligence Ready for Standardization? In 2020. p. 259–74.
7. Golenkov V, Guliakina N, Golovko V, Krasnoproshin V. Artificial Intelligence Standardization Is a Key Challenge for the Technologies of the Future. In 2020. p. 1–21.
8. Roy Chowdhury J. Standardization and Innovation: Standards Development Principles. In: Bhatnagar Anuj and Yadav S and AV and HLU and RS, editor. Handbook of Quality System, Accreditation and Conformity Assessment. Singapore: Springer Nature Singapore; 2024. p. 1–30. Available from: https://doi.org/10.1007/978-981-99-4637-2_5-1
9. Beltrán-Escobar M, Alarcón TE, Rumbo-Morales JY, López S, Ortiz-Torres G, Sorcia-Vázquez FDJ. A Review on Resource-Constrained Embedded Vision Systems-Based Tiny Machine Learning for Robotic Applications. Algorithms. 2024;17(11). https://www.mdpi.com/1999-4893/17/11/476
10. Jiménez M, Palomera R, Couvertier I. Introduction to Embedded Systems: Using Microcontrollers and the MSP430. New York: Springer; 2014.
11. Ibraheem SM, Adrees S. Embedded Systems: For Engineers and Students. 2nd ed. Sheikh Muhammad Ibraheem; 2024.
12. Haralick RM, Shanmugam K, Dinstein I. Textural Features for Image Classification. IEEE Trans Syst Man Cybern. 1973;SMC-3(6):610–21.
13. Soh LK, Tsatsoulis C. Texture Analysis of SAR Sea Ice Imagery Using Gray Level Co-Occurrence Matrices. Vol. 37, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. 1999.
14. Wang H, Guo XH, Jia ZW, Li HK, Liang ZG, Li KC, et al. Multilevel binomial logistic prediction model for malignant pulmonary nodules based on texture features of CT image. Eur J Radiol. 2010 Apr;74(1):124–9.
15. Silva ET, Sampaio F, da Silva LC, Medeiros DS, Correia GP. A method for embedding a computer vision application into a wearable device. Microprocess Microsyst. 2020 Jul 1;76:103086. https://linkinghub.elsevier.com/retrieve/pii/S0141933119304089
16. Medeiros DS. crosswalk-dataset. Kaggle; 2020. https://www.kaggle.com/dsv/1362295

17. Liu R, Liu E, Yang J, Li M, Wang F. Optimizing the Hyper-parameters for SVM by Combining Evolution Strategies with a Grid Search. In: Intelligent Control and Automation. Springer, Berlin, Heidelberg; 2006. p. 344–53.
18. Braga-Neto U. Fundamentals of pattern recognition and machine learning. Springer; 2020.
19. Devi MRSS, Kumar VRV, Sivakumar P. A review of image classification and object detection on machine learning and deep learning techniques. In: 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA). 2021. p. 1–8.
20. Huang Y, Tan T. Feature coding for image representation and recognition. Springer; 2014.
21. Liu Q, Sang H. Design of Real-time Pedestrian Trajectory Prediction System based on Jetson Xavier. Frontiers in Computing and Intelligent Systems. 2023;4(3):109–13.
22. Fanelli G, Dantone M, Van Gool L. Real time 3D face alignment with random forests-based active appearance models. In: IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG). IEEE; 2013. p. 1–8.
23. Vora P, Shrestha S. Detecting diabetic retinopathy using embedded computer vision. Applied Sciences (Switzerland). 2020 Oct 2;10(20):1–10.
24. Mhalla A, Chateau T, Gazzah S, Amara NE Ben. An Embedded Computer-Vision System for Multi-Object Detection in Traffic Surveillance. IEEE Transactions on Intelligent Transportation Systems. 2019 Nov 1;20(11):4006–18.
25. Gonzalez RC, Woods RE. Digital Image Processing. 4th ed. Hoboken, NJ: Pearson; 2018.
26. Umbaugh SE. Digital image processing and analysis: computer vision and image analysis. CRC Press; 2023.
27. Guyon I, Gunn S, Nikravesh M, Zadeh LA. Feature Extraction: Foundations and Applications. Berlin, Heidelberg: Springer; 2006. https://link.springer.com/book/10.1007/978-3-540-35488-8
28. Nixon M, Aguado A. Feature Extraction and Image Processing for Computer Vision. 3rd ed. London: Academic Press; 2012.
29. Salau AO, Jain S. Feature Extraction: A Survey of the Types, Techniques, Applications. In: 2019 International Conference on Signal Processing and Communication (ICSC). IEEE; 2019. p. 158–64.
30. Bishop CM. Pattern Recognition and Machine Learning. New York: Springer; 2006.
31. Forsyth DA, Ponce J. Computer Vision: A Modern Approach. Upper Saddle River, NJ: Prentice Hall; 2002.
32. Hartley R, Zisserman A. Multiple View Geometry in Computer Vision. 2nd ed. Cambridge, UK: Cambridge University Press; 2003.
33. Trucco E, Verri A. Introductory Techniques for 3D Computer Vision. Upper Saddle River, NJ: Prentice Hall; 1998.
34. Pratt WK. Digital Image Processing: PIKS Scientific Inside. 4th ed. Hoboken, NJ: Wiley-Interscience; 2007.
35. Chaki J, Dey N. A Beginner's Guide to Image Preprocessing Techniques. CRC Press; 2018.
36. Ojala T, Pietikäinen M, Harwood D. A comparative study of texture measures with classification based on featured distributions. Pattern Recognit. 1996 Jan;29(1):51–9.

37. Ballard DH. Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognit. 1981 Jan;13(2):111–22.
38. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM. 2017 May 24;60(6):84–90.
39. Ming-Kuei Hu. Visual pattern recognition by moment invariants. IEEE Trans Inf Theory. 1962 Feb;8(2):179–87.
40. Khodadadi N, Towfek SK, Zaki AM, Alharbi AH, Khodadadi E, Khafaga DS, et al. Predicting normalized difference vegetation index using a deep attention network with bidirectional GRU: a hybrid parametric optimization approach. Int J Data Sci Anal. 2024;1–28.
41. Işk Ş. A comparative evaluation of well-known feature detectors and descriptors. International Journal of Applied Mathematics Electronics and Computers. 2014;3(1):1–6.
42. Raj EFI, Balaji M. Shape Feature Extraction Techniques for Computer Vision Applications. In: Smart Computer Vision. Springer; 2023. p. 81–102.
43. Chu G, Peng Y, Luo X. ALGD-ORB: An improved image feature extraction algorithm with adaptive threshold and local gray difference. PLoS One. 2023;18(10):e0293111.
44. Khoei TT, Singh A. Data reduction in big data: a survey of methods, challenges and future directions. Int J Data Sci Anal. 2024;1–40.
45. Kumar D, Pandey RC, Mishra AK. A review of image features extraction techniques and their applications in image forensic. Multimed Tools Appl. 2024;1–102.
46. Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vis. 2004;60:91–110.
47. Khelifi R, Nini B, Berkane M. Enhanced Classification System for Real-Time Embedded Vision Applications. IEEE Access. 2024;12:162311–26.
48. Montesinos López OA, Montesinos López A, Crossa J. Overfitting, model tuning, and evaluation of prediction performance. In: Multivariate statistical machine learning methods for genomic prediction. Springer; 2022. p. 109–39.
49. Huang Y, Wu Z, Wang L, Tan T. Feature Coding in Image Classification: A Comprehensive Study. IEEE Trans Pattern Anal Mach Intell. 2014;36(3):493–506.
50. Guyon I, Elisseeff A. An introduction to variable and feature selection. Journal of Machine Learning Research. 2003;3:1157–82.
51. Sabato S, Kalai A. Feature Multi-Selection among Subjective Features. In: Dasgupta S, McAllester D, editors. Proceedings of the 30th International Conference on Machine Learning. Atlanta, Georgia, USA: PMLR; 2013. p. 810–8. (Proceedings of Machine Learning Research; vol. 28). https://proceedings.mlr.press/v28/sabato13.html
52. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A. Feature Selection for High-Dimensional Data. Cham: Springer International Publishing; 2015 [cited 2024 Nov 29]. https://link.springer.com/book/10.1007/978-3-319-21858-8
53. Yu L, Liu H. Efficient Feature Selection via Analysis of Relevance and Redundancy. Journal of Machine Learning Research. 2004;5:1205–24. https://jmlr.org/papers/v5/yu04a.html
54. Alhassan S, Abdul-Salaam G, Micheal A, Missah YM, Ganaa ED, Shirazu AS. CFS-AE: Correlation-based Feature Selection and Autoencoder for Improved Intrusion Detection System Performance. Journal

of Internet Services and Information Security. 2024 Feb 1;14(1):104–20.

55. Blessie EC, Karthikeyan E. Sigmis: A feature selection algorithm using correlation based method. J Algorithm Comput Technol. 2012;6(3):385–94.
56. Kang Y, Peng L, Guo J, Lu Y, Yang Y, Fan B, et al. A Fast Hybrid Feature Selection Method Based on Dynamic Clustering and Improved Particle Swarm Optimization for High-Dimensional Health Care Data. IEEE Transactions on Consumer Electronics. 2024;70(1):2447–59.
57. Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in bioinformatics. Bioinformatics. 2007 Nov;23(19):2507–17. https://doi.org/10.1093/bioinformatics/btm344
58. Balasubramanian K, N.P. A. Correlation-based feature selection using bio-inspired algorithms and optimized KELM classifier for glaucoma diagnosis. Appl Soft Comput. 2022;128:109432. https://www.sciencedirect.com/science/article/pii/S1568494622005579
59. Sidhom O, Ghazouani H, Barhoumi W. Three-phases hybrid feature selection for facial expression recognition. J Supercomput. 2024;80(6):8094–128. https://doi.org/10.1007/s11227-023-05758-3
60. Fan J, Han F, Liu H. Challenges of Big Data analysis. Natl Sci Rev. 2014 Nov;1(2):293–314. https://doi.org/10.1093/nsr/nwt032
61. Bobda C, Velipasalar S. Distributed embedded smart cameras: Architectures, design and applications. Vol. 9781461477, Distributed Embedded Smart Cameras: Architectures, Design and Applications. 2014. 1–273 p.
62. Mariatos V, Adaos KD, Alexiou GP. Design and implementation of a reconfigurable, embedded real-time face detection system. Proceedings of the International Workshop on Rapid System Prototyping. 2007;65–8.
63. Zhai Y, Ong YS, Tsang IW. The Emerging "Big Dimensionality." IEEE Comput Intell Mag. 2014;9(3):14–26.
64. Peteiro-Barral D, Bolon-Canedo V, Alonso-Betanzos A, Guijarro-Berdinas B, Sanchez-Maroono N. Scalability analysis of filter-based methods for feature selection. Advances in Smart Systems Research. 2012;2(1):21.
65. Fahad A, Tari Z, Khalil I, Habib I, Alnuweiri H. Toward an efficient and scalable feature selection approach for internet traffic classification. Computer Networks. 2013;57(9):2040–57. https://www.sciencedirect.com/science/article/pii/S1389128613001163
66. Bramer M. Principles of Data Mining. London: Springer; 2007. https://link.springer.com/book/10.1007/978-1-84628-765-7
67. Aghazadeh A, Spring R, LeJeune D, Dasarathy G, Shrivastava A, others. Mission: Ultra large-scale feature selection using count-sketches. In: International conference on machine learning. 2018. p. 80–8.
68. Toennies KD. An Introduction to Image Classification: From Designed Models to End-to-End Learning. Springer; 2024.
69. Alnuayri T. MALHKS, Rossi D. A Support Vector Regression based Machine Learning method for on-chip Aging Estimation. In: 2021 4th International Conference on Computing & Information Sciences (ICCIS). 2021. p. 1–6.
70. Roth V, Steinhage V. Nonlinear discriminant analysis using kernel

functions. Adv Neural Inf Process Syst. 1999;12.

71. Kim Hong-Sik AND Joe I. An XAI method for convolutional neural networks in self-driving cars. PLoS One. 2022 Sep;17(8):1–17. https://doi.org/10.1371/journal.pone.0267282
72. Szegedy C, others. Going Deeper with Convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015. p. 1–9.
73. Shao Y, Lunetta RS. Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points. ISPRS Journal of Photogrammetry and Remote Sensing. 2012 Jun;70:78–87.
74. Sarwar J, Khan SA, Azmat M, Khan F. An Application of Hybrid Bagging-Boosting Decision Trees Ensemble Model for Riverine Flood Susceptibility Mapping and Regional Risk Delineation. Water Resources Management. 2024;1–31.
75. Ardabili S, Mosavi A, Várkonyi-Kóczy AR. Advances in machine learning modeling reviewing hybrid and ensemble methods. In: International conference on global research and education. 2019. p. 215–27.
76. Sarker IH. Machine learning: Algorithms, real-world applications and research directions. SN Comput Sci. 2021;2(3):160.
77. Wu M, Zhou J, Peng Y, Wang S, Zhang Y. Deep Learning for Image Classification: A Review. In: International Conference on Medical Imaging and Computer-Aided Diagnosis. 2023. p. 352–62.
78. Cortes C, Vapnik V. Support-vector networks. Vol. 20, Machine learning. Springer; 1995. 273–297 p.
79. Quinlan JR. Induction of decision trees. Mach Learn. 1986;1(1):81–106.
80. Brunelli R. Template matching techniques in computer vision: theory and practice. John Wiley & Sons; 2009.
81. LeCun Y, Bengio Y, Hinton G. Deep Learning. Nature. 2015;521(7553):436–44.
82. Pourkamali-Anaraki F, Husseini JF, Stapleton SE. Probabilistic Neural Networks (PNNs) for Modeling Aleatoric Uncertainty in Scientific Machine Learning. arXiv preprint arXiv:240213945. 2024;
83. Castro HM, Ferreira JC. Linear and logistic regression models: when to use and how to interpret them? SciELO Brasil; 2022.
84. Bishop CM, Winn JM. Non-linear Bayesian image modelling. In: Computer Vision-ECCV 2000: 6th European Conference on Computer Vision Dublin, Ireland, June 26–July 1, 2000 Proceedings, Part I 6. 2000. p. 3–17.
85. Sannasi Chakravarthy SR, Rajaguru H. Detection and classification of microcalcification from digital mammograms with firefly algorithm, extreme learning machine and non-linear regression models: A comparison. Int J Imaging Syst Technol. 2020;30(1):126–46.
86. Murphy KP. Probabilistic Machine Learning: An Introduction. MIT Press; 2022. (Adaptive Computation and Machine Learning series). https://books.google.dz/books?id=wrZNEAAAQBAJ
87. Bazionis IK, Georgilakis PS. Review of deterministic and probabilistic wind power forecasting: Models, methods, and future research. Electricity. 2021;2(1):13–47.
88. Arco JE, Ortiz A, Ram\'\irez J, Mart\'\inez-Murcia FJ, Zhang YD, Broncano J, et al.

Probabilistic combination of eigenlungs-based classifiers for COVID-19 diagnosis in chest CT images. arXiv preprint arXiv:210302961. 2021;

89. Mustafa Khan M, UI Islam MS, Siddiqui AA, Qadri MT. Dual deterministic model based on deep neural network for the classification of pneumonia. Intelligent Decision Technologies. 2023;17(3):641–54.
90. He W, Jiang Z, Xiao T, Xu Z, Li Y. A Survey on Uncertainty Quantification Methods for Deep Learning. arXiv preprint arXiv:230213425. 2023;
91. Liu J, Lin Z, Padhy S, Tran D, Bedrax Weiss T, Lakshminarayanan B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. Adv Neural Inf Process Syst. 2020;33:7498–512.
92. Sun C, Wu W, Wang C. Time series classification of dynamical systems using deterministic learning. Nonlinear Dyn. 2023;111(23):21837–59.
93. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research. 2012;13:281–305.
94. Shetty AM, Aljunid MF, Manjaiah DH, Shaik Afzal AMS. Hyperparameter Optimization of Machine Learning Models Using Grid Search for Amazon Review Sentiment Analysis. In: International Conference on Data Science and Applications. 2023. p. 451–74.
95. Madhavan R, Tunstel E, Messina E. Performance evaluation and benchmarking of intelligent systems. Springer; 2009.
96. John LK, Eeckhout L. Performance evaluation and benchmarking. CRC Press; 2018.
97. Allgaier J, Pryss R. Cross-Validation Visualized: A Narrative Guide to Advanced Methods. Mach Learn Knowl Extr. 2024;6(2):1378–88. https://www.mdpi.com/2504-4990/6/2/65
98. Yates LA, Aandahl Z, Richards SA, Brook BW. Cross validation for model selection: a review with examples from ecology. Ecol Monogr. 2023;93(1):e1557.
99. Markoulidakis I, Markoulidakis G. Probabilistic Confusion Matrix: A Novel Method for Machine Learning Algorithm Generalized Performance Analysis. Technologies (Basel). 2024;12(7):113.
100. Maciel PRM. Performance, reliability, and availability evaluation of computational systems, volume I: performance and background. Chapman and Hall/CRC; 2023.
101. Xing Y, Lv C, Chen L, Wang H, Wang H, Cao D, et al. Advances in vision-based lane detection: Algorithms, integration, assessment, and perspectives on ACP-based parallel vision. IEEE/CAA Journal of Automatica Sinica. 2018;5(3):645–61.
102. Batista BG, Estrella JC, Ferreira CHG, Filho DML, Nakamura LHV, Reiff-Marganiec S, et al. Performance evaluation of resource management in cloud computing environments. PLoS One. 2015;10(11):e0141914.
103. Paulino N, Ferreira JC, Cardoso JMP. Improving performance and energy consumption in embedded systems via binary acceleration: A survey. ACM Computing Surveys (CSUR). 2020;53(1):1–36.
104. Chen S, Gao P, Wang X, Liao K, Zhang P. Highly compact adaptive network based on transformer for RGBT tracking. Infrared Phys Technol. 2024;139:105310. https://www.sciencedirect.com/science/article/pii/S1350449524001944
105. Sabattini JA, Sturniolo F, Bollazzi M, Bugnon LA. AntTracker: A low-cost and

efficient computer vision approach to research leaf-cutter ants behavior. Smart Agricultural Technology. 2023;5:100252. https://www.sciencedirect.com/science/article/pii/S2772375523000825

106. Li K, Wang D, Liu G, Zhu W, Zhong H, Wang Q. DiagSWin: A multi-scale vision transformer with diagonal-shaped windows for object detection and segmentation. Neural Networks. 2024;180:106653. https://www.sciencedirect.com/science/article/pii/S089360802400577X
107. Nuño-Maganda MA, Dávila-Rodríguez IA, Hernández-Mier Y, Hugo Barrón-Zambrano J, Carlos Elizondo-Leal J, Díaz-Manriquez A, et al. Real-Time Embedded Vision System for Online Monitoring and Sorting of Citrus Fruits. 2023; Available from: https://doi.org/10.3390/electronics
108. Shiao YS. Design and implementation of real-time tracking system based on vision servo control. Tamkang Journal of Science and Engineering. 2001;4(1):45–58.
109. Berger AH. Embedded Systems Design: An Introduction to Processes, Tools, and Techniques. Taylor & Francis; 2002. (CMP Books). https://books.google.dz/books?id=3vY35UkvXrAC
110. Sun W, Webster-Wood V. An integrated computer vision system for real-time monitoring and control of long-fiber embedded hydrogel 3D printing. Mater Today Proc. 2022 Jan 1;70:376–81.
111. Myers K, Secco EL. A Low-Cost Embedded Computer Vision System for the Classification of Recyclable Objects. Lecture Notes on Data Engineering and Communications Technologies. 2021;61:11–30. https://link.springer.com/chapter/10.1007/978-981-33-4582-9_2
112. Surya T, Chitra Selvi S, Selvaperumal S. The IoT-based real-time image processing for animal recognition and classification using deep convolutional neural network (DCNN). Microprocess Microsyst. 2022 Nov 1;95.
113. Park J, Choi AJ. Vision-Based In-Flight Collision Avoidance Control Based on Background Subtraction Using Embedded System. Sensors. 2023 Jul 1;23(14).
114. Luo X, Feng L, Xun H, Zhang Y, Li Y, Yin L. Rinegan: A Scalable Image Processing Architecture for Large Scale Surveillance Applications. Front Neurorobot. 2021 Aug 23;15.
115. Manzouri F, Heller S, Dümpelmann M, Woias P, Schulze-Bonhage A. A comparison of machine learning classifiers for energy-efficient implementation of seizure detection. Front Syst Neurosci. 2018 Sep 20;12.
116. Zhu Y, Tan Y, Hua Y, Wang M, Zhang G, Zhang J. Feature selection and performance evaluation of support vector machine (SVM)-based classifier for differentiating benign and malignant pulmonary nodules by computed tomography. J Digit Imaging. 2010 Feb;23(1):51–65.
117. Parker JR. Algorithms for image processing and computer vision. John Wiley & Sons; 2010.
118. Güneş A, Kalkan H, Durmuş E. Optimizing the color-to-grayscale conversion for image classification. Signal Image Video Process. 2016 Jul 29;10(5):853–60.
119. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Cham: Springer; 2022. (Texts in Computer Science).
120. Ebied M, Elmisery FA, Zekry A, Al-Zubi N, El Samie FEA. Utilization of decimation interpolation strategy for medical image communication and storage. In: 2018 8th

International Conference on Computer Science and Information Technology (CSIT). IEEE; 2018. p. 22–5.

121. Mirmehdi M. Handbook of texture analysis. Imperial College Press; 2008.
122. Raju P, Rao VM, Rao BP. Optimal GLCM combined FCM segmentation algorithm for detection of kidney cysts and tumor. Multimed Tools Appl. 2019 Jul 28;78(13):18419–41.
123. A JSP, Sankar D. GLCM based SVM Classifier for Covid-19 Detection from Chest X-rays. In: 2023 4th International Conference on Signal Processing and Communication (ICSPC). IEEE; 2023. p. 68–72.
124. Shahajad M, Gambhir D, Gandhi R. Features extraction for classification of brain tumor MRI images using support vector machine. In: 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE; 2021. p. 767–72.
125. Iqbal N, Mumtaz R, Shafi U, Zaidi SMH. Gray level co-occurrence matrix (GLCM) texture based crop classification using low altitude remote sensing platforms. PeerJ Comput Sci. 2021;7:e536.
126. Imani M, Ghassemian H. GLCM, Gabor, and morphology profiles fusion for hyperspectral image classification. In: 2016 24th Iranian Conference on Electrical Engineering (ICEE). IEEE; 2016. p. 460–5.
127. Asha V, Bhajantri NU, Nagabhushan P. GLCM based Chi-square Histogram Distance for Automatic Detection of Defects on Patterned Textures.
128. Raheja JL, Ajay B, Chaudhary A. Real time fabric defect detection system on an embedded DSP platform. Optik (Stuttg). 2013 Nov;124(21):5280–4.
129. Varsha A, Mundra K, Singh A, Bhosale H, Chattopadhyay C, Valadi J. From Pixels to Insight: Enhancing Metallic Component Defect Detection with GLCM Features and AI Explainability. In 2024. p. 289–301.
130. Salman AD, Talab MA, Al-Dahhan RR. Features Extraction for Robust Face Recognition Using GLCM and CS-LBP. In 2022. p. 175–91.
131. Tassi A, Vizzari M. Object-Oriented LULC Classification in Google Earth Engine Combining SNIC, GLCM, and Machine Learning Algorithms. Remote Sens (Basel). 2020 Nov 17;12(22):3776.
132. Silversides KL. Pattern Classification. In 2021. p. 1–3.
133. Abdiansah A, Wardoyo R. Time Complexity Analysis of Support Vector Machines (SVM) in LibSVM. Int J Comput Appl. 2015 Oct 15;128(3):28–34.
134. Akram-Ali-Hammouri Z, Fernández-Delgado M, Cernadas E, Barro S. Fast Support Vector Classification for Large-Scale Problems. IEEE Trans Pattern Anal Mach Intell. 2022;44(10):6184–95.
135. Chen CH. Handbook of pattern recognition and computer vision. World scientific; 2015.
136. Awad M, Khanna R. Support Vector Machines for Classification. In: Efficient Learning Machines. Berkeley, CA: Apress; 2015. p. 39–66.
137. Vapnik VN, Vapnik V, others. Statistical learning theory. 1998;
138. Psaltakis G, Rogdakis K, Loizos M, Kymakis E. One-vs-One, One-vs-Rest, and a novel Outcome-Driven One-vs-One binary classifiers enabled by optoelectronic memristors towards overcoming hardware limitations in multiclass classification. Discov Mater. 2024 Mar 3;4(1):7.
139. Zheng F, Shao L. A Winner-Take-All Strategy for Improved Object Tracking.

IEEE Transactions on Image Processing. 2018 Sep;27(9):4302–13.

140. Sampaio F, Silva LC Da, Filho PPR, Silva ET Da. Reducing Computational Costs of an Embedded Classifier to Determine Leather Quality. Brazilian Symposium on Computing System Engineering, SBESC. 2017;2017-Novem:211–6.
141. Ullah S, Nguyen DA, Wang H, Menzel S, Sendhoff B, Back T. Exploring Dimensionality Reduction Techniques for Efficient Surrogate-Assisted optimization. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE; 2020. p. 2965–74.
142. Ali I, Wassif K, Bayomi H. Dimensionality reduction for images of IoT using machine learning. Sci Rep. 2024 Mar 26;14(1):7205.
143. Wang L, Wang Y, Xiong S, Yang J. Some aspects of nonlinear dimensionality reduction. Comput Stat. 2024 Jun 16;
144. Zhang S, Qin F, Xu J, Zhang J. Small Sample Scene Cancer Classification Method Based on Combined Feature Dimensionality Reduction. In: 2024 16th International Conference on Computer and Automation Engineering (ICCAE). IEEE; 2024. p. 346–50.
145. Liu Y, Sui A. Research on Feature Dimensionality Reduction in Content Based Public Cultural Video Retrieval. In: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). IEEE; 2018. p. 718–22.
146. Ghosh S, Pramanik P. A Combined Framework for Dimensionality Reduction of Hyperspectral Images using Feature Selection and Feature Extraction. In: 2019 IEEE Recent Advances in Geoscience and Remote Sensing : Technologies, Standards and Applications (TENGARSS). IEEE; 2019. p. 39–44.
147. Alouache A. Feature Selection Using Teaching-Learning-Based Optimization Algorithm for Classification of Remote Sensing Images. In: 2024 21st International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). 2024. p. 1–6.
148. Vellido A. The importance of interpretability and visualization in machine learning for applications in medicine and health care. Neural Comput Appl. 2020 Dec 4;32(24):18069–83.
149. Jia W, Sun M, Lian J, Hou S. Feature dimensionality reduction: a review. Complex & Intelligent Systems. 2022 Jun 21;8(3):2663–93.
150. Miah ASM, Shin J, Hasan MdAM. Effective features extraction and selection for hand gesture recognition using sEMG signal. Multimed Tools Appl. 2024 Jun 3;83(37):85169–93.
151. Zhang Y, Han X, Yang J. Selection of optimal spectral features for leaf chlorophyll content estimation. Sci Rep. 2024 Oct 27;14(1):25598.
152. Edmonds J. Time Complexity. In: How to Think About Algorithms. Cambridge University Press; 2012. p. 366–73.
153. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed. Cambridge, MA: MIT Press; 2009. 44–46 p.
154. Zhang X, Li S, Yang J, Wang Y, Huang Z, Zhang J. Tactile Perception Object Recognition Based on an Improved Support Vector Machine. Micromachines (Basel). 2022 Sep 1;13(9).
155. Fisher RA. The use of multiple measurements in taxonomic problems. Ann Eugen. 1936;7(2):179–88.
156. Reshef DN, Reshef YA, Finucane HK, Grossman SR, McVean G, Turnbaugh PJ, et al. Detecting novel associations in large

data sets. Science (1979). 2011;334(6062):1518–24.

157. Battiti R. Using mutual information for selecting features in supervised neural net learning. IEEE Trans Neural Netw. 1994;5(4):537–50.
158. Yu L, Liu H. Efficient Feature Selection via Analysis of Relevance and Redundancy. Vol. 5, Journal of Machine Learning Research. 2004.
159. Palma-Mendoza RJ, de-Marcos L, Rodriguez D, Alonso-Betanzos A. Distributed correlation-based feature selection in spark. Inf Sci (N Y). 2019 Sep 1;496:287–99.
160. Jain I, Jain VK, Jain R. Correlation feature selection based improved-Binary Particle Swarm Optimization for gene selection and cancer classification. Appl Soft Comput. 2018 Jan 1;62:203–15.
161. Martínez A. L. H. KSAT, Rossi D. Online Remaining Useful Lifetime Prediction Using Support Vector Regression. IEEE Trans Emerg Top Comput. 2022;10(3):1546–57.
162. Kohavi R, John GH. Wrappers for feature subset selection. Artif Intell. 1997;97(1–2):273–324.
163. scikit-learn.org. https://scikit-learn.org/stable/modules/svm.html. 2024 [cited 2024 Dec 19]. Support Vector Machines. Available from: https://scikit-learn.org/stable/modules/svm.html
164. Molloy D. Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux. Wiley; 2016.
165. Upton E, Halfacree G. Raspberry Pi User Guide. John Wiley & Sons; 2016.
166. Richardson M, Wallace S. Getting Started with Raspberry Pi. O'Reilly Media; 2012.
167. raspberrypi.com. https://www.raspberrypi.com/software/. 2024. Raspberry Pi OS.
168. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. 2012 Jan 2;
169. toxigon.com. https://toxigon.com/integrating-numpy-with-scikit-learn. 2024. How to Integrate NumPy with Scikit-Learn: A Comprehensive Guide.
170. Lilja DJ. Measuring computer performance: a practitioner's guide. Cambridge university press; 2005.
171. Mahmoud M, others. Efficient Feature Selection for Wearable Health Monitoring Systems Using Genetic Algorithms. IEEE J Biomed Health Inform. 2019;23(6):2307–18.
172. Ghasemzadeh H, Jafari R. Energy-Efficient Signal Processing in Embedded Systems. IEEE Transactions on Signal Processing. 2009;57(10):4134–42.
173. Chowdhery A, Liu J, Ghandi MM. Exploiting Computation-Accuracy Trade-offs in Embedded Devices for Deep Learning. ACM SIGOPS Operating Systems Review. 2017;51(2):11–22.
174. Sangiovanni-Vincentelli A, Martin G. Platform-Based Design and Software Design Methodology for Embedded Systems. IEEE Design & Test of Computers. 2001;18(6):23–33.
175. Grover P, Sahai A. Shannon Meets Tesla: Wireless Information and Power Transfer. IEEE Trans Inf Theory. 2014;60(7):4553–69.
176. Elhanashi A, Dini P, Saponara S, Zheng Q. Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices. Electronics (Basel). 2024 Sep 8;13(17):3562.
177. Liu HI, Galindo M, Xie H, Wong LK, Shuai HH, Li YH, et al. Lightweight Deep Learning for Resource-Constrained Environments: A Survey. 2024 Apr 8;