Ensino que te conecta

BACHARELADO EM
COMPUTER SCIENCE

DEVELOPMENT OF AN ARTIFICIAL INTELLIGENCE SYSTEM FOR DETECTION AND CLASSIFICATION OF AEDES AEGYPTI ON A LOW-COST EMBEDDED DEVICE

RAFAEL DE OLIVEIRA GUEDES NOGUEIRA

Brasília - DF, 2025

RAFAEL DE OLIVEIRA GUEDES NOGUEIRA

DEVELOPMENT OF AN ARTIFICIAL INTELLIGENCE SYSTEM FOR DETECTION AND CLASSIFICATION OF AEDES AEGYPTI ON A LOW-COST EMBEDDED DEVICE

Undergraduate Thesis presented as a partial requirement for the degree of Bachelor in Computer Science, by the Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa (IDP).

Advisor

M.Sc. Patrícia da Silva Oliveira

Brasília - DF, 2025

Código de catalogação na publicação – CIP

N778d Nogueira, Rafael de Oliveira Guedes

Development of an artificial intelligence system for detection and classification of aedes aegypti on a low-cost embedded device/ Rafael de Oliveira Guedes Nogueira. — Brasília: Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa, 2025.

86 f. : il.

Orientador: Profª. Ma. Patrícia Da Silva Oliveira

Monografia (Graduação em Ciência da Computação) - Instituto Brasileiro Ensino, Desenvolvimento e Pesquisa – IDP, 2025.

1. Aedes aegypti. 2. Edge artificial intelligence (Edge AI). 3. Shortcut learning. I. Título

CDD 006.3

Elaborada por Biblioteca Ministro Moreira Alves

RAFAEL DE OLIVEIRA GUEDES NOGUEIRA

DEVELOPMENT OF AN ARTIFICIAL INTELLIGENCE SYSTEM FOR DETECTION AND CLASSIFICATION OF AEDES AEGYPTI ON A LOW-COST EMBEDDED DEVICE

Undergraduate Thesis presented as a partial requirement for the degree of Bachelor in Computer Science, by the Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa (IDP).

Approved on

Examination Board

gov.br Documento assinado digitalmente
PATRICIA DA SILVA OLIVEIRA
Data: 07/01/2026 14:05:01-0300
Verifique em https://validar.iti.gov.br

M.Sc. Patrícia da Silva Oliveira - Advisor

gov.br Documento assinado digitalmente
KLAYTON RODRIGUES DE CASTRO
Data: 07/01/2026 10:33:45-0300
Verifique em https://validar.iti.gov.br

M.Sc. Klayton Rodrigues de Castro - Internal Examiner

Mathias Schneid Tessmann

Assinado de forma digital por Mathias Schneid Tessmann
Dados: 2026.01.07 03:06:22 -03'00'

Dr. Mathias Schneid Tessmann - Internal Examiner

DEDICATION

Dedico esta obra primeiramente a Deus, Fonte de toda sabedoria,
que não apenas sustenta a minha existência,
mas fomenta em minha alma a inquietude constante
de investigar e compreender a realidade em sua essência.
Aos meus pais, alicerces do meu caráter:
ao meu pai, Miguel, pela lição perene de buscar a verdade,
despida de aparências ou conveniências;
e à minha mãe, Marta, que traduziu o conceito de amor
através da linguagem silenciosa do sacrifício e da entrega.
Ao meu querido amigo de infância, João Luiz,
que a vida se encarregou de tornar um irmão.
Agradeço por me ajudar a enxergar o mundo sob diferentes perspectivas
e por ser um grande incentivador do meu desenvolvimento pessoal.
À minha cunhada, Glaucia Maria, a quem considero uma irmã.
Agradeço pela amizade sincera e pelo apoio constante.
Por fim, dedico este trabalho de forma especial ao meu irmão, Gabriel.
Você tem sido meu norte, meu conselheiro
e o espelho de humanidade onde busco refletir minhas ações.
Esta conquista também é sua.

ACKNOWLEDGMENTS

I would like to thank the Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa (IDP) for the excellence of the academic environment and for all the support and infrastructure provided throughout my undergraduate studies.

I acknowledge the Fundação de Apoio à Pesquisa do Distrito Federal (FAP-DF) for the financial support granted to the research project that originated this work, which was fundamental for providing the necessary resources.

I am immensely grateful to my advisor, Professor Patrícia de Oliveira, for her wise guidance, patience, and valuable academic contributions that made this work possible.

To Professor Jeremias Gomes, the most frequent instructor in my degree, I offer my deep gratitude for his example of dedication and his invaluable willingness to assist me whenever necessary.

To Professor Klayton Rodrigues, I thank you for accepting the invitation to serve on the examining board and for the important lessons taught throughout the course.

I would also like to extend my sincere appreciation to the entomology professionals at both the Instituto Aggeu Magalhães (Fiocruz-PE) and the ArboControl Laboratory (UnB). Their welcoming attitude, technical support, and willingness to share their expertise were fundamental to the success of the data collection phase.

To my godmother Miriam Rosa and godfather Antônio Silvestre, I extend my deepest gratitude for hosting me in their residence during my undergraduate studies, providing an invaluable foundation of support and stability.

Finally, I express deep gratitude to my friend and colleague Igor Caldeira, whose technical collaboration was indispensable. I am especially grateful for his effort in developing both versions of the smart trap used in this study.

ABSTRACT

Epidemiological surveillance of *Aedes aegypti* is a critical component in preventing arboviruses such as dengue, Zika, and chikungunya. However, traditional manual identification methods are labor-intensive, time-consuming, and difficult to scale. This work develops an embedded, real-time computer vision system for the automatic detection and classification of the species *Aedes aegypti*, *Aedes albopictus*, and *Culex quinquefasciatus*, optimized for low-cost edge devices, specifically a Raspberry Pi 4. A comprehensive dataset of 85,296 images, representing 753 unique mosquito specimens, was collected and curated in two distinct phases and locations to ensure biological diversity. To address the critical problem of Shortcut Learning, where models overfit to background features rather than mosquito morphology, a rigorous background normalization protocol was developed and validated. This strategy proved essential to ensure the model's generalization capability. The proposed system employs a sequence of Convolutional Neural Networks (CNNs). For object detection, the YOLOv8n model achieved a Mean Average Precision (mAP@0.5) of 99.44%. For species classification, the YOLOv8n-cls architecture achieved a test accuracy of 98.18%, successfully distinguishing the morphologically similar species of the *Aedes* genus. The complete pipeline was optimized for the embedded device using the NCNN framework, achieving an inference latency of 3.3 seconds for classification. The results demonstrate that high-precision automated vector surveillance is feasible on affordable hardware, offering a scalable solution for public health monitoring.

Keywords: Aedes aegypti, vector surveillance, convolutional neural networks (CNNs), shortcut learning, edge artificial intelligence (Edge AI), raspberry pi.

RESUMO

A vigilância epidemiológica do *Aedes aegypti* é um componente crítico na prevenção de arboviroses como dengue, Zika e chikungunya. No entanto, os métodos tradicionais de identificação manual são trabalhosos, demorados e difíceis de escalar. Este trabalho desenvolve um sistema de visão computacional embarcado e em tempo real para a detecção e classificação automática das espécies: *Aedes aegypti*, *Aedes albopictus* e *Culex quinquefasciatus*, otimizado para dispositivos de borda de baixo custo, especificamente uma Raspberry Pi 4. Um conjunto de dados abrangente de 85.296 imagens, representando 753 espécimes únicos de mosquitos, foi coletado e curado em duas fases e locais distintos para garantir a diversidade biológica. Para abordar o problema crítico do "Aprendizado de Atalho" (*Shortcut Learning*), onde os modelos se ajustam excessivamente às características do fundo em vez da morfologia do mosquito, um rigoroso protocolo de normalização de fundo foi desenvolvido e validado. Essa estratégia provou-se essencial para garantir a capacidade de generalização do modelo. O sistema proposto utiliza uma sequência de Redes Neurais Convolucionais (CNNs). Para a detecção de objetos, o modelo YOLOv8n alcançou uma Precisão Média (mAP@0.5) de 99,44%. Para a classificação de espécies, a arquitetura YOLOv8n-cls atingiu uma acurácia de teste de 98,18%, distinguindo com sucesso as espécies morfologicamente semelhantes do gênero *Aedes*. O pipeline completo foi otimizado para o dispositivo embarcado utilizando o framework NCNN, alcançando uma latência de inferência de 3,3 segundos para classificação. Os resultados demonstram que a vigilância vetorial automatizada de alta precisão é viável em hardware acessível, oferecendo uma solução escalável para o monitoramento de saúde pública.

Palavras-chave: Aedes aegypti, vigilância vetorial, redes neurais convolucionais (CNNs), inteligência artificial de borda (Edge AI), raspberry pi.

LIST OF FIGURES

LIST OF TABLES

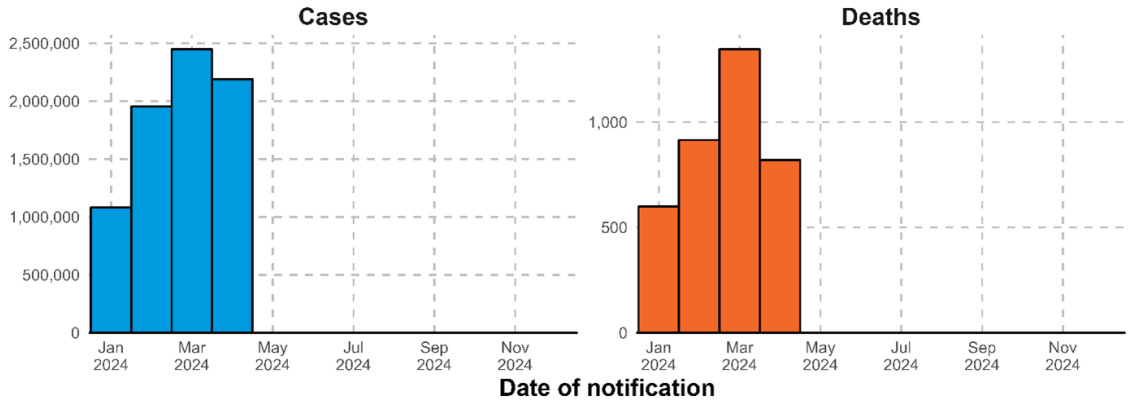
CONTENTS


1

1 INTRODUCTION

Mosquito-borne diseases, known as arboviruses, remain one of the leading threats to global public health. In this context, *Aedes aegypti* plays a central role as the primary vector of diseases such as dengue, Zika, and chikungunya. Figure 1 shows that between January and April 2024, approximately 7.6 million probable cases of dengue were reported worldwide, with over 3,000 deaths attributed to the disease, affecting more than 80 countries in all World Health Organization regions [1]. Dengue alone has shown an alarming growth trend: between 2000 and 2022, the number of reported cases increased more than tenfold, rising from approximately 500,000 to over 5.2 million cases annually. In 2024, large-scale outbreaks have already been recorded in several regions of Asia, Latin America, and the Caribbean, requiring urgent containment and entomological surveillance measures.

Figure 1: Epidemic curve of dengue cases and deaths as reported to WHO from January to April 2024



Source: [1].

In Brazil, the situation is equally alarming. The country bears one of the highest global burdens of arboviral diseases, with recurrent outbreaks that have significant social and economic impacts. In 2024, approximately 6.1 million probable cases of dengue were reported, with more than 6,000 resulting in death. [2]. The high incidence

has been accompanied by increased hospitalizations and deaths, primarily affecting populations in densely populated urban areas with inadequate sanitation infrastructure. The endemic presence of the *Aedes aegypti* mosquito, the primary vector of these diseases, further exacerbates the crisis, which demands continuous monitoring and control efforts.

In the Federal District, the capital of Brazil, the spread of dengue has also raised serious concerns. In 2024, the Federal District Health Department reported an increase of more than 900% in notified and probable dengue cases, increasing from 26,663 in 2023 to 274,802 in 2024. The fatality rate among severe cases reached 4.3%, according to the weekly bulletin from the Emergency Operations Center (COE) [3]. Indicators exceed national averages on several metrics, with some administrative regions reaching epidemic thresholds.

In general, the rapid proliferation of the vector is supported by local factors, including a hot and humid climate, areas with accumulated waste and standing water, and a lack of consistent vector surveillance measures.

In this context, vector surveillance emerges as an essential tool for preventive measures and rapid response to arboviral outbreaks. However, traditional methods of vector identification, despite being scientifically established, present significant limitations. These methods rely predominantly on the capture of adult mosquitoes using traps (such as CDC traps or ovitraps), followed by the transportation of specimens to specialized laboratories, where they undergo sorting and morphological identification by trained entomologists. This identification process requires careful observation of specific anatomical features under stereoscopic microscopes, including the pattern of scales on the thorax and abdomen, the shape of the antennae and proboscis, and the presence of white stripes on the legs—typical characteristics of *Aedes aegypti* [4].

Traditional methods are not only labor-intensive but also require specialized technical infrastructure and laboratory materials, making large-scale applications difficult, especially in regions with limited operational capacity. According to [5], even in urban centers with adequate infrastructure, the average time between mosquito capture and the issuance of entomological reports can exceed several days, compromising the speed required for effective public health decisions. Moreover, the shortage of professionals trained in medical entomology—particularly in remote or lower-income areas—worsens the issue and limits the coverage of surveillance efforts.

These factors reveal the operational bottlenecks that hinder the scalability of conventional vector surveillance methods. In a scenario marked by an overburdened public health system and a shortage of qualified field professionals, the development of innovative, scalable, and low-cost solutions for the early and accurate identification of vectors has become increasingly urgent. In this context, artificial intelligence-based technologies have gained prominence for offering automation, portability, and economic

feasibility. Recent studies have indicated that Convolutional Neural Networks (CNNs) are capable of achieving over 90% accuracy in mosquito classification from images, even when deployed on low-cost embedded devices such as the Raspberry Pi, as demonstrated by [6] and [7]. These technological advances enable a shift from the traditional centralized, lab-dependent surveillance model to a distributed, automated system that responds more effectively to local dynamics of vector proliferation.

However, the effective deployment of these technologies faces a significant dichotomy. Existing solutions typically fall into two categories: they either rely on deep, high-precision architectures that demand computational resources unavailable in low-cost devices (requiring cloud connectivity), or they utilize lightweight models that, when optimized for the edge, often sacrifices the taxonomic accuracy required to distinguish between morphologically similar species like *Aedes aegypti* and *Aedes albopictus*. Addressing this trade-off—achieving high accuracy within the constraints of embedded hardware—remains an open challenge necessary for scalable field surveillance.

Therefore, the general objective of this study is to develop an intelligent system capable of detecting and classifying *Aedes aegypti* mosquitoes from images, implemented on a low-cost embedded device, to support vector monitoring and control strategies. To accomplish this goal, the following specific objectives were defined:

- Collect and organize an image database containing specimens of *Aedes aegypti* and other morphologically similar species, captured directly inside the proprietary trap utilized in this study, ensuring the dataset represents the specific operational conditions of the proposed solution.
- Develop and validate a background normalization preprocessing strategy to mitigate shortcut learning, ensuring that the model's generalization capability is driven strictly by distinctive morphological features rather than spurious background correlations.
- Implement an object detection model, aiming for a mean Average Precision (mAP@0.5) greater than 95% in identifying mosquitoes within the trap.
- Develop a classification model capable of distinguishing *Aedes aegypti* from other species with an accuracy greater than 95%, leveraging transfer learning techniques.
- Optimize and deploy the computer vision pipeline on a Raspberry Pi 4, targeting an end-to-end accuracy greater than 90% and a viable inference latency for discrete monitoring events of less than 5 seconds per sample.

The methodological procedure will involve image collection and annotation, training of Convolutional Neural Network (CNN) models, application of compression and

quantization techniques for embedded execution, and experimentation on a physical embedded device. Tools such as Python, PyTorch, NCNN (Tencent), and OpenCV will be employed throughout the process.

Beyond this introduction, the structure of this work is organized as follows:

- *Literature Review*: This section begins by presenting the epidemiological landscape of arboviral diseases transmitted by *Aedes aegypti*, contextualizing their relevance to both global and national public health. It then introduces the foundations of computer vision and Convolutional Neural Networks (CNNs), covering topics from image acquisition and preprocessing to automatic feature extraction and species classification, as well as specific challenges in deep learning such as shortcut learning. It also details the performance metrics used to evaluate the models, such as mAP, Accuracy, and Latency.

- *Methodology*: This section details the experimental design, including the construction of an image dataset collected at the ArboControl insectary [8] of the University of Brasília (UnB) and the Aggeu Magalhães Institute (Fiocruz) in Pernambuco [9]. It specifically details the background normalization protocol adopted to prevent shortcut learning, the selection of YOLOv8 architectures, and the optimization pipeline for the NCNN framework on embedded hardware.

- *Results and Discussion*: This section presents the experimental findings, starting with an analysis of background removal techniques and the impact of the background normalization protocol. It then details the performance of the object detector (YOLOv8n) and the species classifier (YOLOv8n-cls). Finally, it benchmarks the complete embedded system on the Raspberry Pi 4, evaluating inference latency, throughput, and end-to-end accuracy.

- *Conclusion*: This section summarizes the main contributions, discussing the effectiveness of the proposed background normalization protocol and the feasibility of the embedded system. It also outlines limitations and suggests future research directions, such as sex classification and non-AI background removal methods.

2

2 LITERATURE REVIEW

The literature review aims to present the key concepts, theoretical foundations, and prior studies that support the development of this work. It is divided into two sections: the Theoretical Framework and Related Works.

2.1 THEORETICAL FRAMEWORK

This section presents the main concepts and technologies required for understanding and developing the proposed system.

2.1.1 *Aedes aegypti* Mosquito and its Epidemiological Importance

Aedes aegypti is a holometabolous insect undergoing the developmental stages of egg, larva (four instars), pupa, and adult [10]. Its eggs are exceptionally resilient, capable of withstanding desiccation, and remaining viable for approximately one year, which complicates control efforts [10]. This species exhibits behavior highly adapted to urban and domestic environments, with a strong preference for artificial water containers for oviposition and the ability to complete its life cycle in approximately 7–10 days under favorable conditions.

Adult *Aedes aegypti* mosquitoes are relatively small, black in color, with silvery-white scales arranged in a characteristic pattern (e.g., a lyre-shaped marking on the dorsal thorax) [11]. It differs from similar species based on these scale patterns. Behaviorally, *Aedes aegypti* is highly anthropophilic—it feeds preferentially on human blood even in the presence of alternative hosts—and frequently bites multiple times within a single gonotrophic cycle [11]. This multiple-bite behavior increases its vectorial potential by amplifying the chances of pathogen transmission. Furthermore, it is a diurnal/crepuscular and domestic mosquito, often found inside or around human dwellings, where it rests and seeks hosts; it is commonly found just a few meters from houses and easily enters homes [11]. Such behaviors (anthropophily, diurnal activity near humans, and repeated biting) contribute significantly to its efficiency as a disease vector.

Aedes aegypti stands out as the principal vector of several arboviruses of significant public health concern, including the viruses responsible for dengue, Zika, chikungunya, and yellow fever [12]. Review studies on vector competence show that virtually no population of this mosquito is completely refractory to infection by these viruses—*Aedes*

aegypti populations worldwide are highly susceptible to dengue, Zika, and chikungunya, with no evidence of full natural resistance [12]. This high vector competence, combined with its strong adaptation to urban environments and close association with humans, results in a significant epidemiological impact. It is estimated that approximately 3.9 billion people are at risk of dengue infection across more than 120 countries where *Aedes aegypti* is present [13]. In recent years, the incidence of these arboviral diseases has risen sharply—for instance, in the Americas alone, 4.6 million dengue cases were reported in 2023 [13]. Modeling studies project the continued geographic expansion of *Aedes aegypti* due to climatic and anthropogenic factors, with potential colonization of up to 162 countries by 2080 [13]. In summary, the combination of a prolific life cycle, ecological plasticity (ability to survive in varied breeding sites and urban settings), and anthropophilic behavior makes *Aedes aegypti* an extremely relevant epidemiological vector associated with recurrent epidemics of dengue, Zika, chikungunya, and other arboviruses in tropical and subtropical regions.

2.1.2 Pipeline of a Computer Vision System for Insect Identification

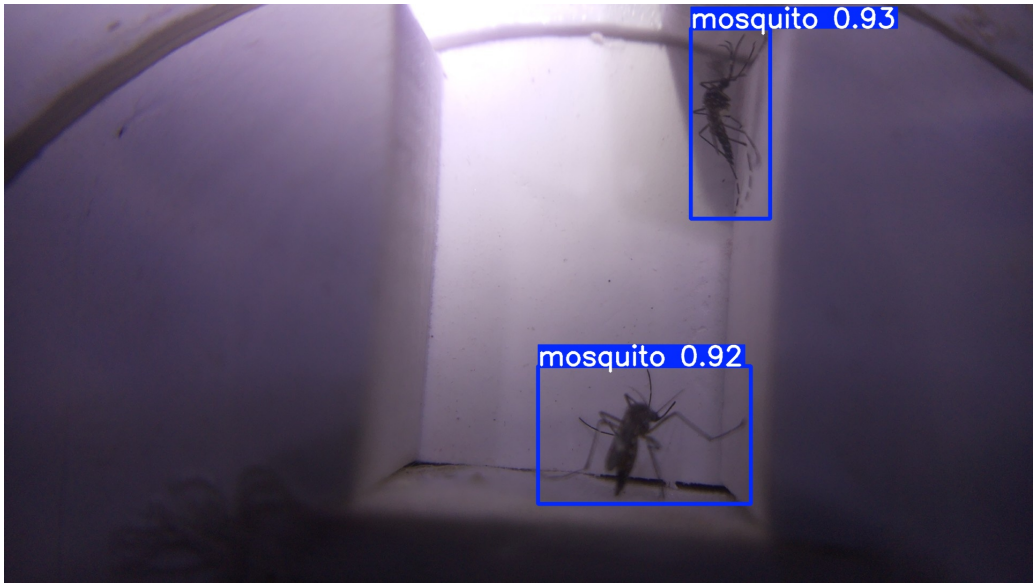
Accurate insect species identification is a fundamental pillar in various fields, including agriculture, ecology, and, most critically, public health, where disease vector surveillance is essential. Traditionally, such identification relies on morphological analysis performed by specialized entomologists—a process that, while accurate, can be time-consuming, costly, and limited by the availability of experts [14]. Given these challenges, computer vision has emerged as a promising tool for automating and scaling insect identification, offering the potential for more efficient and responsive monitoring systems. This section explores the core concepts of computer vision as applied to this task. A computer vision system for insect identification typically involves a sequence of stages, from image capture to final species classification. These stages aim to replicate and enhance the human visual analysis process using algorithms designed to process and interpret visual data [15].

- **Image Acquisition**: At the core of any computer vision system is the image itself. The quality of the acquired image is crucial for the downstream performance of the system. Factors such as resolution, lighting, focus, and camera perspective directly influence the amount of helpful information available for analysis [15]. In the context of insect identification, image acquisition may occur in controlled laboratory environments using cameras coupled with microscopes or stereoscopes or in the field through photographic cameras embedded in traps or mobile devices. Variability in field conditions (e.g., unstable natural lighting, complex backgrounds) presents additional challenges that must be considered in system design [14].

- **Object Detection**: Following image acquisition, the system must identify and locate the insects within the captured scene. In standard image classification tasks,

it is generally assumed that there is only one object of interest per image. However, as illustrated in Fig. 2, in field monitoring scenarios, it is common for several mosquitoes—or even other insects—to appear simultaneously in the same image. Therefore, object detection algorithms are employed to individually separate the specimens, generating bounding boxes that delimit each insect. This step is crucial to enable accurate counting and to isolate the specimens for the subsequent classification stages.

Figure 2: Conceptual illustration of mosquitoes processed by an AI-based detection system



Source: own elaboration.

Among the main object detection models, Faster R-CNN [16] stands out by adopting a two-stage approach—region proposal followed by classification, achieving high accuracy at the cost of higher computational demand. In contrast, single-stage models such as SSD MobileNetV2 (Single Shot MultiBox Detector) [17] and YOLO (You Only Look Once) [18] perform detection in a single pass, making them more suitable for real-time applications. These models offer good accuracy and are widely adopted in embedded scenarios due to their light weight and ease of implementation. More recently, EfficientDet [19] has gained attention for combining high accuracy with low computational consumption, using a scalable architecture based on EfficientNet and an efficient feature fusion module (BiFPN).

These models are evaluated using metrics such as IoU (Intersection over Union) and mAP (mean Average Precision), which respectively measure the overlap between predicted and actual bounding boxes and the average detection performance across categories. For applications such as this—fine-grained entomolog-

ical classification in the field—it is essential to choose architectures that balance strong generalization capabilities, low latency, and compatibility with low-cost devices, such as the Raspberry Pi.

- **Image Preprocessing**: Once the objects of interest have been detected and their bounding boxes defined, the isolated regions (ROIs) undergo a critical preprocessing phase. The goal is to standardize the data and ensure that the subsequent classification model learns from relevant information. The first step involves a Quality Filtering mechanism, which evaluates quantitative metrics—such as Edge Density, Michelson Contrast, and Contour Ratio—to automatically reject low-quality samples (e.g., blurred, low-contrast, or occluded images) that could degrade model performance. Following selection, Photometric Normalization techniques are applied. Contrast Limited Adaptive Histogram Equalization [20] (CLAHE) is employed to enhance local contrast and reveal morphological textures, mitigating lighting variations common in field traps. Finally, to prevent the model from learning spurious environmental patterns (shortcut learning), a Background Removal strategy is adopted. This involves segmenting the insect and replacing the original background with synthetic noise, ensuring the model focuses exclusively on the insect's morphological features.

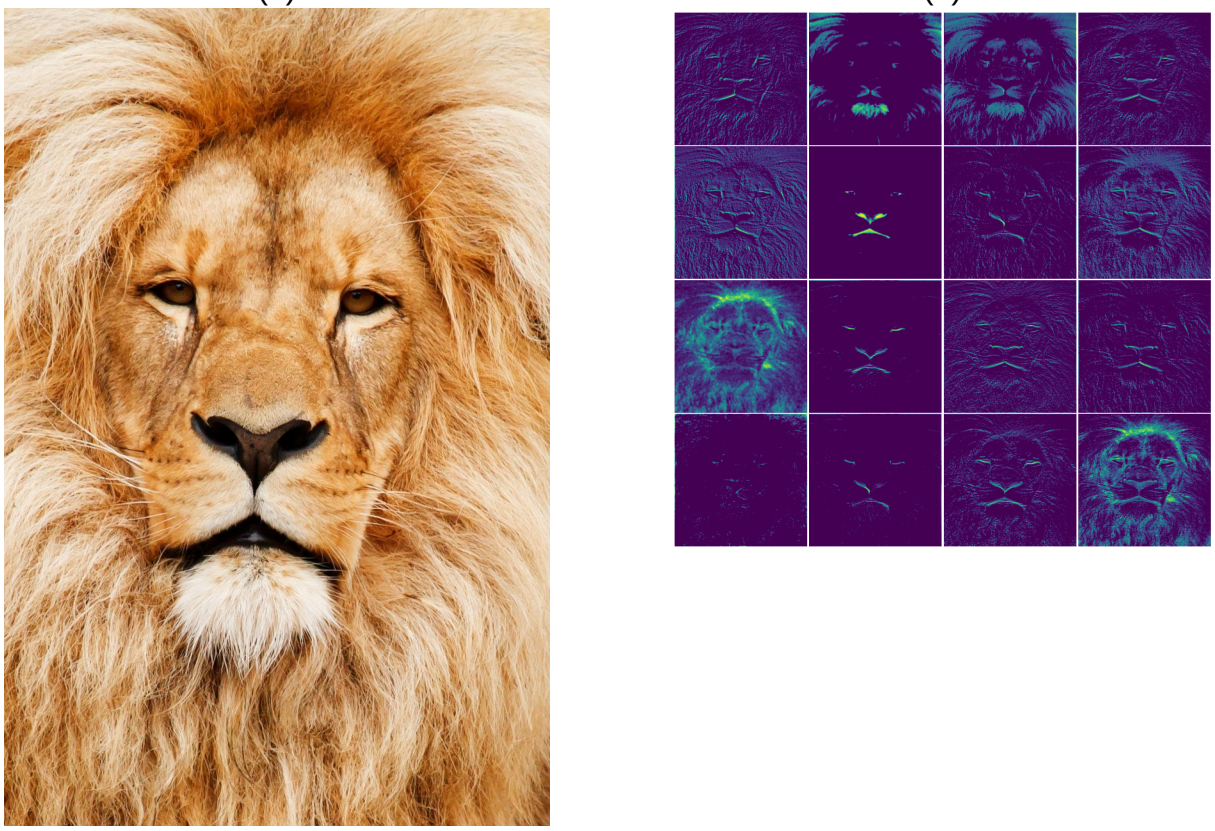
  Following these standardization steps, the data is prepared for the deep learning model. To improve the model's robustness and generalization capability, **Data Augmentation** is applied. This process artificially expands the training set by applying random geometric and photometric transformations such as rotations, zooming, and horizontal flips. These operations simulate the various ways the mosquito may appear to the camera, making the final model more invariant to positional and orientational variations [14]. Finally, each image crop is resized to match the input dimensions required by the CNN architecture (e.g., 224×224 pixels), and its pixel values are normalized to a standard numerical range, such as [0, 1] or [-1, 1]—a necessary condition for stable and efficient deep neural network optimization [14].

- **Feature Extraction**: Feature extraction is the process of identifying and quantifying distinctive attributes of an insect that enable its classification. Historically, this step relied on hand-crafted feature engineering, in which experts defined which morphological traits (e.g., wing shape, body color patterns, exoskeleton texture, proportions between body segments) would be measured. Classical computer vision algorithms, such as edge detectors (e.g., Canny), texture descriptors (e.g., Local Binary Patterns – LBP, Gabor filters), and shape descriptors (e.g., Hu moments, Fourier contours), were employed to extract quantitative information from insect images [15]. With the advent of deep learning, especially Convolutional

Neural Networks (CNNs), feature extraction has undergone a profound transformation. CNNs are capable of automatically learning the most discriminative features directly from raw input data (image pixels), eliminating the need for manual attribute selection and encoding [14]. During training, the network adjusts its internal filters (kernels) to detect hierarchical patterns—ranging from simple edges and textures in the initial layers to complex shapes and objects in the deeper layers — explicitly optimized for the target classification task, as illustrated in Fig. 3.

Figure 3: Feature extraction performed over the image of a lion using VGG19 CNN architecture. (a) Original picture of the lion (public domain, available at Pexels). (b) Feature maps generated by CNN

(a) (b)



Source: adapted from [21].

- **Species Classification**: Once the features have been extracted, a classification model is used to assign a species label to the insect. When using manually engineered features, traditional machine learning algorithms such as Support Vector Machines (SVMs), Random Forests, k-Nearest Neighbors (k-NN), or simpler Artificial Neural Networks (ANNs) can be trained to perform this task [15]. These classifiers learn to map the extracted feature set to the different species classes

present in the training dataset.

In the case of CNNs, feature extraction and classification are often integrated into a single end-to-end architecture. The convolutional and pooling layers act as feature extractors, while the final layers of the network (typically fully connected layers followed by an activation function such as softmax) are responsible for classification [14]. This integrated approach has demonstrated state-of-the-art performance in various visual recognition tasks, including species identification. The performance of any insect identification system based on machine learning—particularly those employing deep learning—is inherently dependent on the quality and quantity of the data used for training. A large and representative dataset is required, containing images of various species of interest that are adequately labeled by domain experts [14]. Building datasets tailored to insect species presents specific challenges, which are outlined below:

- **Intraspecies Variability**: Individuals of the same species may exhibit variations in size, color, and shape due to factors such as age, sex, diet, or geographic location
- **Interspecies Similarity**: Different but closely related species may be morphologically very similar, requiring the capture of subtle distinguishing details.
- **Field Image Quality**: Images acquired in field conditions may suffer from inconsistent lighting, partial occlusions, physical damage to specimens (e.g., missing legs or antennae), and complex backgrounds.
- **Labeling Scalability**: Properly annotating large volumes of images is a time-consuming and expertly demanding task.

Techniques such as data augmentation, which consists of generating new training samples by applying transformations to existing images (e.g., rotations, zooming, brightness adjustments), can help increase dataset diversity and model robustness [14].

2.1.3 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are computational models inspired by the biological neural networks that constitute animal brains [22]. They are designed to recognize patterns and solve complex problems by learning from data. The fundamental unit of an ANN is the artificial neuron, or perceptron, which mathematically models the function of a biological neuron.

Each neuron receives a set of inputs, multiplies them by specific weights, sums them up with a bias term, and passes the result through a non-linear activation function to produce an output. This process can be described by Eq. 1:

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1}$$

where x_i are the inputs, w_i are the weights, b is the bias, and ϕ is the activation function.

A single neuron has limited computational power. However, when organized into layers—an input layer, one or more hidden layers, and an output layer—they form a Multilayer Perceptron (MLP). MLPs are capable of approximating any continuous function, a property known as the Universal Approximation Theorem [23].

The learning process in ANNs involves adjusting the weights and biases to minimize the difference between the predicted output and the actual target. This is typically achieved using the backpropagation algorithm combined with an optimization method such as Stochastic Gradient Descent (SGD). During training, data flows forward through the network (forward propagation) to calculate the loss, and then error gradients are propagated backward (backpropagation) to update the parameters.

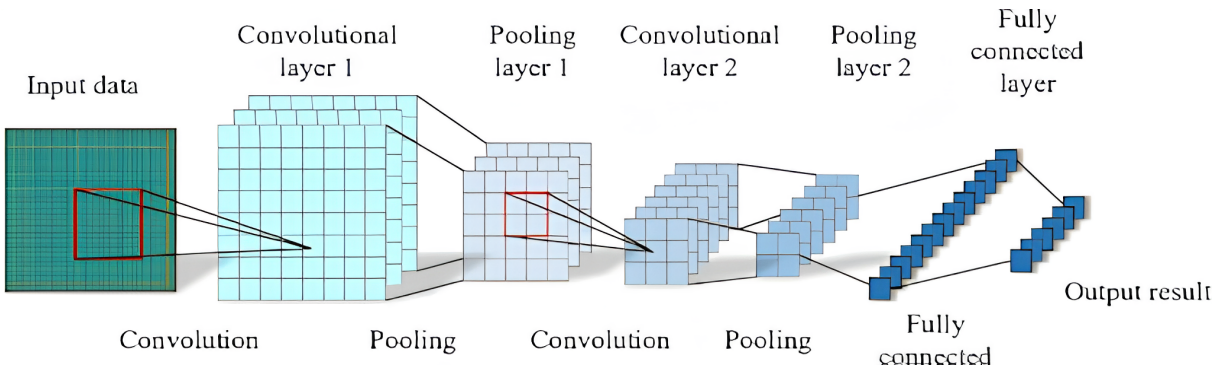
While ANNs are powerful for many tasks, they face significant challenges when applied directly to image data. To process an image with an MLP, the 2D grid of pixels must be flattened into a 1D vector. This operation destroys the spatial structure and local correlations present in the image (e.g., the relationship between adjacent pixels forming an edge). Furthermore, for high-resolution images, the number of trainable parameters in a fully connected network explodes, leading to high computational costs and a high risk of overfitting. These limitations paved the way for the development of Convolutional Neural Networks, which are specifically designed to preserve spatial hierarchies in visual data.

2.1.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) represent a specialized class of deep learning models designed for processing and analyzing grid-structured data, such as images and videos [14]. Their design is bioinspired, emulating aspects of the human visual cortex, and their effectiveness lies in the ability to learn hierarchical features directly from the data automatically.

A CNN is defined as a multilayer neural network distinguished by the use of three main types of layers: convolutional, pooling (subsampling), and fully connected layers. The structure of a CNN is illustrated in Fig. 4.

Figure 4: Schematic diagram of CNN structure



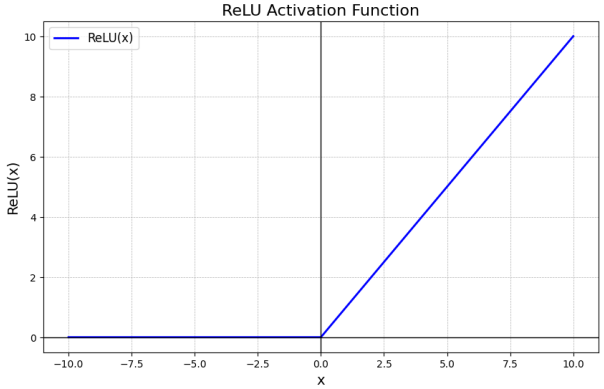
Source: [24].

Convolutional layers form the core of CNNs. Unlike traditional neural networks with full connectivity, neurons in a convolutional layer are connected only to a small region of the previous layer, known as the receptive field [14]. A key element is weight sharing: the same set of weights, forming a filter or kernel, slides over the entire input image, allowing the network to detect the same pattern (e.g., a vertical edge) regardless of its position in the image. This not only drastically reduces the number of parameters to be learned but also provides the network with a degree of translation invariance [14].

The 2D discrete convolution operation between an input **I** (image or feature map) and a kernel **K** para gerar um mapa de características de saída **O** pode ser definida pela Eq. 2.

$$\mathbf{O}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{I}(i-m, j-n)\,\mathbf{K}(m,n) \tag{2}$$

Figure 5: Rectified Linear Unit (ReLU) activation function graph



Source: own elaboration.

Parameters such as stride (the step size of the kernel) and padding (filling the input borders) are used to control the spatial dimensions of the resulting feature maps [14].

After the convolution operation, a non-linear activation function is applied to each element of the feature map. Non-linearity is essential for the network to learn complex representations, as stacking multiple linear layers would be equivalent to a single linear transformation [14]. An example is the ReLU (Rectified Linear Unit) function, shown in Fig. 5, which is widely used due to its simplicity and effectiveness in mitigating the vanishing gradient problem. It is mathematically defined in Eq. 3.

$$f(x) = \max(0, x) \tag{3}$$

The pooling layers aim to reduce the spatial dimensionality of the feature maps, decreasing the number of parameters and computational costs in the network while also providing a degree of invariance to small translations and distortions [14].

After several convolutional and pooling layers, which progressively extract increasingly complex and abstract features (from simple edges and textures in the initial layers to object parts or entire objects in the deeper layers), the architecture typically ends with one or more fully connected layers. In these layers, each neuron is connected to all activations from the previous layer, similar to a traditional multilayer neural network [14]. The final fully connected layer typically has several neurons equal to the number of classes. It utilizes an activation function, such as softmax, to produce a probability distribution over the classes.

When training end-to-end, the network itself learns the most relevant features, eliminating the need for manual feature engineering. This ability to automatically and robustly learn representations is what has made Convolutional Neural Networks (CNNs) highly successful in computer vision tasks, outperforming traditional vision methodologies across numerous challenges.

The resurgence of CNNs began with the remarkable victory of the AlexNet model in the 2012 ImageNet challenge, demonstrating far superior performance in image classification compared to previous methods [25]. Since then, a series of increasingly deeper and more efficient architectures have been proposed to improve accuracy in vision tasks. Notable examples include the VGG-16/19 networks [26], which standardized the use of multiple stacked convolutional layers with small filters (3x3); GoogLeNet/Inception [27], which introduced blocks with multiple parallel convolutions; and ResNets [28].

ResNets introduced the innovative concept of shortcut residual connections, allowing part of the information flow to “skip” intermediate layers [28]. This simple idea solved the vanishing gradient problem in very deep networks, enabling the stable training of models with more than 100 layers [28]. ResNet-152 (with 152 layers) won the ILSVRC 2015 competition, achieving record-breaking accuracy and proving that ultra-deep net-

works are feasible when shortcuts are added to facilitate gradient propagation [28]. Other notable architectures include Squeeze-and-Excitation Networks (SENets), which introduced channel-wise attention mechanisms, and DenseNets [29], which connect each layer to all previous ones to maximize feature reuse. As a result, the literature highlights a dramatic evolution in CNNs over the past decade – they have become deeper, with sophisticated modular blocks and training optimization strategies, leading to leaps in image recognition performance [25].

A significant challenge that emerged was using CNNs on resource-constrained devices (such as smartphones and embedded systems). To address this, optimized architectures were developed to reduce computational cost with minimal loss of accuracy. A prominent example is MobileNet, proposed by [30] as a family of lightweight convolutional neural networks (CNNs) for mobile and embedded applications. MobileNet introduced depth-wise separable convolutions, which decompose the traditional convolution operation into two steps – spatial filtering by channel and then pointwise channel combination – drastically reducing the number of parameters and operations required [30]. With its compact architecture and configurable width and resolution hyperparameters, it is possible to obtain small and fast models suitable for low-power hardware. Despite its simplicity, MobileNet models achieved outstanding performance in image classification (e.g., 70% Top-1 accuracy on ImageNet for MobileNetV1), becoming a benchmark for computer vision on edge devices. Other efficiency-focused architectures include ShuffleNet and EfficientNet [31], which apply compound scaling principles to optimize the accuracy-cost ratio by adjusting depth, width, and resolution.

In summary, modern CNNs range from profound and accurate models (VGG, ResNet, Inception, etc.) to compact and fast models (MobileNets and variants), all based on the same fundamental principles of convolution, pooling and hierarchical feature learning [30]. This theoretical framework provides the basis for selecting architectures that match the project's needs – for instance, choosing a lightweight model like MobileNet or a compressed variant, given the hardware constraints (e.g., Raspberry Pi) in the study at hand.

2.1.5 Performance Evaluation Metrics

To quantitatively assess the effectiveness of the developed models, standard metrics from the computer vision literature are employed. These metrics provide an objective basis for comparing different architectures and validating the system's reliability.

- **Classification Metrics**: For the classification stage, performance is evaluated based on the Confusion Matrix, which categorizes predictions into four types: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). From these, the following metrics are derived [32]:
  - **Accuracy**: The ratio of correctly predicted observations to the total obser-

vations. It is the most intuitive performance measure but can be misleading in imbalanced datasets, as defined in Eq. 4.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives. It indicates the model's ability to avoid false positives, as shown in Eq. 5.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

- **Recall (Sensitivity)**: The ratio of correctly predicted positive observations to the all observations in the actual class. It measures the model's ability to find all relevant cases, described by Eq. 6.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

- **F1-Score**: The weighted average of Precision and Recall. It is particularly useful when seeking a balance between precision and recall, or when the class distribution is uneven, as calculated in Eq. 7.

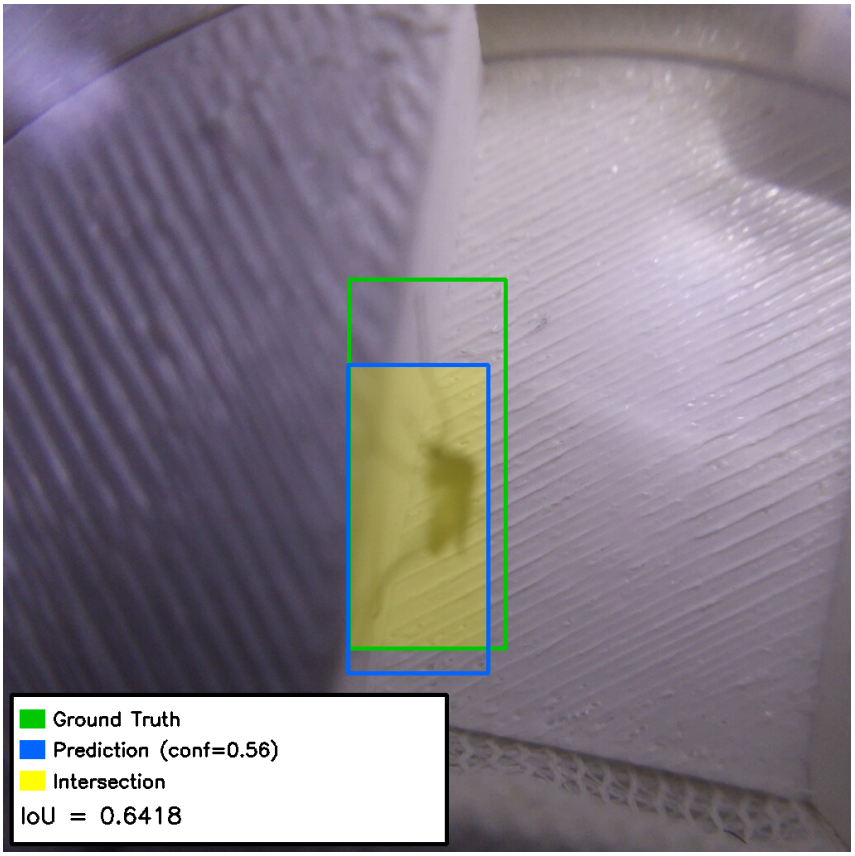
$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{7}$$

- **Detection Metrics**: For the object detection stage, evaluation is more complex as it involves both localizing the object and classifying it. The primary metrics used are [33]:

  - **Intersection over Union (IoU)**: A metric used to evaluate the accuracy of an object detector on a particular dataset. It computes the area of overlap between the predicted bounding box (B_{pred}) and the ground truth bounding box (B_{gt}), divided by the area of their union, as formulated in Eq. 8.

$$IoU = \frac{Area(B_{pred} \cap B_{gt})}{Area(B_{pred} \cup B_{gt})} \tag{8}$$

Figure 6: Visual representation of Intersection over Union (IoU) on a dataset sample



Source: own elaboration.

Figure 6 illustrates a raw dataset sample showing the ground-truth bounding box (B_{gt}), the predicted bounding box (B_{pred}), and their intersection area used to compute the Intersection over Union (IoU) metric (IoU) for detection evaluation.

- **Mean Average Precision (mAP)**: The standard metric for object detection. It calculates the average precision (AP) for each class and then averages these values. The AP is the area under the Precision-Recall curve, computed at specific IoU thresholds (e.g., mAP@0.5 considers a detection correct if IoU $\geq$ 0.5). This metric provides a single score that summarizes the model's performance across all classes and confidence thresholds.

2.1.6 Challenges in Deep Learning: Shortcut Learning

Despite the remarkable success of Convolutional Neural Networks, they are susceptible to a phenomenon known as "Shortcut Learning." This occurs when a model achieves high performance on a specific dataset by learning decision rules that are computationally simple but semantically irrelevant to the intended task [34]. Instead of learning the complex morphological features that define an object (e.g., the lyre-shaped

markings on the thorax of *Aedes aegypti*), the network may rely on spurious correlations, such as the background color, lighting conditions, or specific watermarks present in the training images.

Research indicates that shortcut learning is a significant cause of the lack of generalization in deep learning models [34]. A classic example is a classifier trained to recognize cows that fails when the cow is not on a green pasture, simply because it learned to associate the texture of grass with the class “cow” rather than the animal itself. In the context of entomological surveillance, this risk is critical. If a model is trained on images where *Aedes aegypti* always appears in a specific laboratory cage while other species are photographed in different environments, the network may learn to classify the “cage” rather than the mosquito. This leads to a system that performs perfectly during validation but fails when deployed in the field or a new environment. Addressing this challenge requires rigorous dataset curation and bias mitigation strategies, such as background removal or domain randomization, to ensure the model learns robust and generalizable features.

2.1.7 Artificial Intelligence on Embedded Systems

Embedded systems are computing systems designed to perform dedicated functions within larger mechanical or electrical systems, often with real-time computing constraints [35]. Unlike general-purpose computers (such as PCs or servers), which are designed to handle a wide range of tasks, embedded systems are optimized for specific applications, prioritizing efficiency, reliability, and low power consumption. Common examples range from simple microcontrollers in household appliances (e.g., washing machines, microwaves) to complex systems in automotive control units (ECUs), medical devices (pacemakers), and industrial automation robots.

Embedded devices, such as the Raspberry Pi, offer an affordable platform for the local deployment of artificial intelligence algorithms; however, they impose severe hardware limitations. [36] observed a growing trend toward decentralizing deep learning applications onto mobile/embedded devices to meet real-time and privacy requirements (avoiding cloud dependency) [36]. However, they emphasize that the limited resources of these devices—computational power, memory, and energy—make it challenging to run conventional deep neural network models, which demand high computational and storage capacity [36].

According to [36], the main technical difficulties in enabling deep learning at the edge include slow inference speed, overheating, high power consumption, and the inability to train complex models on-device. To mitigate these issues, several optimization techniques have been employed, including network pruning (removing irrelevant weights and connections) and weight quantization (reducing numerical precision) to shrink model size [36], as well as architecture design strategies focused on efficiency

(e.g., separable layers, inverted blocks). In parallel, dedicated hardware solutions have gained traction—such as mobile GPUs, FPGAs, and specialized ASICs (NPUs, TPUs)—alongside optimized libraries and frameworks (e.g., TFLite, OpenVINO, ARM NN) that leverage these resources to accelerate inference under constrained environments [36]. This body of research suggests that although challenging, embedded AI is feasible when combining lightweight models, network compression, and accelerator hardware support.

The Raspberry Pi, especially in its newer versions (e.g., Pi 4 or Pi 5), has been used as a testing platform for neural network inference in the field. Practical applications, such as facial recognition systems, object classification, or obstacle detection in robotics, have been implemented on the Raspberry Pi, typically using lightweight network versions (often trained in the cloud and deployed for local inference), which achieve satisfactory real-time performance at modest scales. However, it is essential to emphasize that on a typical Raspberry Pi (e.g., ARM quad-core CPU at 1.5 GHz, without a powerful GPU), deep neural networks usually do not achieve high FPS rates for complex vision tasks – model optimization and possible accuracy trade-offs may be necessary to reach the desired speed [36].

A complementary strategy to enable AI on embedded systems is to add AI acceleration coprocessors. In the Raspberry Pi ecosystem, for instance, one can connect modules like the Google Coral USB Accelerator, which contains an edge TPU (Tensor Processing Unit) dedicated to neural network inference. As reported by [37], adding a Coral accelerator via USB allows the Raspberry Pi to perform deep learning inference at high speeds and with much greater efficiency than the CPU, enabling previously impractical applications [37]. This Edge TPU can process vision models (e.g., image detection) in near real-time with low energy consumption, making it ideal for field projects that require quick response and continuous operation [37].

The combination of Raspberry Pi and accelerators has been utilized in industrial and IoT contexts, such as predictive maintenance image analysis or real-time agricultural pest recognition, where the embedded device must make decisions locally without relying on the cloud [37]. Even with accelerators, concerns remain regarding energy consumption and thermal dissipation in long-term AI deployments. In such cases, it is essential to strike a balance between system performance and autonomy, especially when powered by batteries or solar cells.

In addition to model optimization and the use of specialized hardware, enabling artificial intelligence on embedded devices heavily depends on software frameworks designed for resource-constrained environments. Notable examples include LiteRT (formerly TensorFlow Lite) and OpenCV, both of which are widely used on platforms such as the Raspberry Pi. LiteRT is Google's on-device AI runtime environment, which enables TensorFlow models to be converted and optimized through techniques such

as quantization and pruning, thereby reducing model size and accelerating inference. Its compact interpreter can delegate operations to accelerators like GPUs and TPUs – e.g., the Google Coral – maximizing performance and energy efficiency [14].

Complementarily, OpenCV (Open Source Computer Vision Library) offers a comprehensive collection of optimized algorithms for various computer vision tasks, including image preprocessing, feature extraction, object tracking, and geometric transformations [15]. It also features its deep neural network module (DNN), which is compatible with models trained in various frameworks. The integration of LiteRT and OpenCV enables the construction of efficient and flexible pipelines, combining lightweight inference with robust visual processing – a crucial combination for the development of intelligent embedded systems.

Thus, the literature and use cases suggest that embedded artificial intelligence is a rapidly evolving field. Model compression techniques and efficient architectures, combined with specialized hardware like edge TPUs and supported by optimized frameworks, are overcoming processing and energy barriers, allowing compact devices like the Raspberry Pi to run convolutional neural network algorithms for complex tasks (computer vision, pattern recognition, etc.) directly “at the edge,” in field environments. This paves the way for low-cost solutions in environmental monitoring, healthcare, agriculture, and other fields. However, it requires a solid theoretical foundation to guide system design decisions for the intelligent system being developed.

2.2 RELATED WORKS

2.2.1 Applications of AI in Public Health Systems

AI-based intelligent systems are gaining ground in public health, whether to assist in medical diagnoses, conduct epidemiological surveillance, or predict disease outbreaks. One of the most advanced areas is the application of machine learning to analyze large volumes of epidemiological and environmental data, enabling the detection of early signs of outbreaks. Predictive models for dengue, for example, have evolved from simple statistical techniques to more sophisticated machine learning approaches. A recent systematic review found that approximately 40% of dengue outbreak prediction models currently utilize machine learning techniques, particularly those developed in recent years [38].

These models use climate data (temperature, rainfall, humidity, etc.), demographic information, and case history to learn patterns that precede incidence surges. In general, they have demonstrated performance gains over traditional models while also providing insights into risk factors. For example, a study in Bangladesh combined interpretable decision tree algorithms (Random Forest, XGBoost, LightGBM) with data from 2000 to 2021 to build a dengue alert system. The model detected non-linear ef-

fects of climate variables and identified key factors (land use, population density, minimum temperature) associated with outbreaks [39]. Notably, the best-performing model (LightGBM) proved effective in forecasting dengue cases in advance, functioning as an alert system that can improve understanding of outbreak-triggering factors [39]. According to the authors, this system provides a sophisticated analytical framework for public health, enabling authorities to anticipate epidemics and implement control strategies proactively [39].

In Brazil, similar approaches have been explored. Recurrent Neural Networks (RNNs) of the Long Short-Term Memory (LSTM) type have been trained to predict weekly dengue cases in various cities, successfully detecting early signs of potential outbreaks [24]. These deep learning models can capture complex temporal dependencies in incidence and climate data, outperforming conventional methods in data-rich scenarios.

There are also efforts to apply explainable machine learning—for example, models that indicate the contribution of each variable (rainfall, temperature, mosquito indices) to the projected risk—which aids transparency and acceptance of forecasts [40]. Big Data and AI tools have also been used to indirectly monitor diseases. During Zika and chikungunya outbreaks, researchers analyzed internet search data and social media mentions, combined with predictive models, to estimate the real-time spread of these arboviruses, complementing traditional notification systems.

Another crucial AI application in public health is the automated epidemiological surveillance of vectors and reservoirs. Besides dengue, similar systems have been proposed for malaria, West Nile fever, and other zoonoses. Projects such as Microsoft Premonition exemplify a “digital epidemiology” trend by predicting risks using AI from automated vector monitoring, with cloud-based algorithms analyzing these signals to forecast biological threats, much like weather forecasts are made [41]. By combining robotics, IoT, and AI, Premonition aims to identify abnormal increases in vector populations or the presence of pathogens in those vectors before outbreaks occur [41]. The platform has already examined trillions of genetic sequences in captured mosquitoes, identifying viruses and other circulating agents in a given region [41]. In practice, such a system could trigger early alerts. For instance, if captured mosquitoes start carrying an emerging virus, authorities would be notified to intensify vector control or public health actions before human cases spike.

At the local level, public health agencies have adopted AI in pilot surveillance projects. In Texas (USA), the Harris County Health Department integrated automated mosquito traps that, with the help of AI, identify target species (vectors of Zika, dengue, West Nile, etc.) and record environmental data in real-time for each capture [42]. These data feed into geographic information systems that map the spatial and temporal distribution of vectors, allowing control teams to be directed precisely to high-risk areas and times [42]. The observed benefits include faster and more accurate information about

where to focus spraying or campaigns, saving resources, and increasing the impact of interventions [42]. Similarly, in Brazil and other tropical countries, it is envisioned that smart connected traps (such as [43]) could complement the fieldwork of health agents. Instead of manual periodic visits to ovitraps or visual larva readings, AI could provide continuous 24/7 monitoring, automatically alerting teams at the first sign of *Aedes aegypti* infestation increase in a given area. This would expedite transmission-blocking actions and the elimination of breeding sites.

Beyond mosquito surveillance, smartphones and AI have been employed in diagnosing and monitoring infectious diseases. For instance, computer vision algorithms analyze exam images (such as chest X-rays for tuberculosis detection, blood smears for malaria, and skin lesions for identifying leishmaniasis) with accuracy comparable to that of specialists, assisting in diagnosis in regions with limited medical personnel. In arboviruses, AI studies already support the differential diagnosis of dengue, Zika, and chikungunya based on symptom profiles and laboratory results, probabilistically indicating the most likely disease to inform clinical decision-making. Predictive models have also been used to forecast yellow fever outbreaks in specific regions by combining data on primate epizootics, vaccination coverage, and entomological indices, helping prioritize preventive campaigns in these areas [5].

These examples illustrate that AI is becoming a valuable tool in public health, improving the ability to predict and respond to infectious diseases [39]. However, significant challenges remain: most models depend on high-quality and up-to-date data, and in many locations, underreporting or a lack of granular data reduces the effectiveness of forecasts. Moreover, integrating AI systems into public health workflows requires staff training and trust in the results. Ethical and privacy concerns (e.g., using cellphone or social media data to track diseases) must also be addressed. In summary, although AI does not yet replace traditional epidemiology, it is becoming a powerful ally, adding speed and predictive intelligence to infectious disease surveillance and control systems.

2.2.2 Applications of Computer Vision in Insect Identification

Recent research has demonstrated the potential of computer vision algorithms, particularly deep learning, in classifying mosquito species using images. [7] developed a convolutional neural network (CNN)-based model integrated into a multi-stage system for mosquito identification using images, addressing the open-set recognition problem where species not present during training must be handled [7]. Using a dataset of 2,696 specimens across 67 mosquito species, the model achieved an accuracy of approximately 97% in closed-set classification, where all task classes are previously represented in the training data—meaning the system only needs to distinguish among known species [7]. Published in Scientific Reports, the study highlights that computer

vision solutions can be both accurate and scalable for automating the identification of field-collected mosquitoes, thus supporting entomological surveillance and vector control programs [7].

In line with the advances in using CNNs for entomology, [5] proposed an innovative solution for vector surveillance of mosquitoes that transmit malaria. The system, named VectorBrain, features a two-stage architecture: first, the YOLOv5 (small version) model performs mosquito body detection and cropping, removing background noise and optimizing the region of interest; then, the cropped images are processed by a custom EfficientNet-B1 network, which is configured for multi-task learning. Experiments were conducted using images from the VectorCAM dataset [4], which comprises 4,195 specimens photographed under real-world conditions by community agents in three endemic African countries (Uganda, Ghana, and Zambia), resulting in a total of 17,612 multi-angle images. The system achieved 94.4% accuracy (macro F1-score of 94.1%) in species classification, 97% (macro F1-score of 95.1%) in sex identification, and 83.2% (macro F1-score of 81.6%) in abdominal status inference. In addition to high performance, the model was optimized for on-device execution in low-cost hardware, running offline and in real-time—reinforcing its applicability in field scenarios with limited infrastructure. Thus, VectorBrain represents a promising milestone for automating critical entomological tasks, enabling scalability, agility, and accuracy in vector control operations in Sub-Saharan Africa [5].

Furthermore, [44] demonstrated the effectiveness of CNNs in identifying parasitoid hymenopterans using only specific morphological traits. In their experiment, a CNN was trained (via transfer learning) to classify 57 genera of Braconidae wasps based on forewing images [44]. Despite a relatively small training set (488 images), the model achieved 96.7% accuracy in cross-validation classification, proving that taxonomic identification of insects is feasible using images of subtle morphological structures [44]. The authors highlighted that the automated approach performed comparably to human experts, underscoring the potential of computer vision in aiding the identification of hyperdiverse insect groups in entomology [44].

Beyond classification under controlled or lab conditions, computer vision techniques have progressed in detecting insects in natural environments. [45] developed a system combining cameras and deep learning models (YOLO algorithms) to monitor pollinators and pests in real-time in the field [45]. They compiled a large dataset of about 30,000 insect annotations (bees, hoverflies, butterflies, beetles, etc.) captured in over 2 million images from time-lapse cameras positioned above flowers [45]. The trained detection models achieved a mean precision of 92.7% and recall of 93.8% for the simultaneous detection and classification of nine functionally relevant insect groups [45]. Notably, the best-performing model (YOLOv5-based) maintained good performance even when facing previously unseen individuals—detecting around 80% of these “unknown” in-

sects, often classifying them as a close species [45]. These results demonstrate that modern CNNs can accurately identify insects in complex scenes with multiple species in natural backgrounds, enabling the development of new non-destructive methods for biodiversity and pest monitoring in the field [45].

Another study [46] focused on classifying morphologically similar mosquito species using a custom dataset of approximately 3,600 images across eight species. The photos exhibited realistic variations in pose and deformation, including missing body parts and simulated field-collected samples. To address data scarcity, the authors investigated the effectiveness of transfer learning, fine-tuning three pre-trained deep convolutional neural networks (DCNN) architectures (VGG-16, ResNet-50, and SqueezeNet) alongside data augmentation techniques. The approach proved highly effective, achieving a classification accuracy of up to 97.19% with the VGG-16 model. The combination of fine-tuning and augmentation was critical to the results. A distinguishing aspect of the study was its use of visualization techniques to analyze model behavior, demonstrating that the DCNN learned to utilize discriminative morphological features similar to those employed by human experts—such as patterns on the thorax and abdomen. Additionally, the study considered field applicability by reporting inference latency and energy consumption of the models on a resource-constrained embedded platform (Nvidia Jetson TX2).

In summary, the literature indicates that computer vision techniques (ranging from classical image processing algorithms to deep learning with CNNs) are revolutionizing entomology, enabling automated insect identification with high accuracy and speed—whether to assist in taxonomic recognition of disease vectors or to monitor pollinators and pests in ecology and agriculture projects.

2.2.3 Automated Classification of *Aedes aegypti* Using CNNs

Recent research has validated the use of Convolutional Neural Networks (CNNs) for the automated classification of mosquitoes, with several studies demonstrating high accuracy in identifying the *Aedes* genus. In a comparative analysis, [47] evaluated the LeNet, AlexNet, and GoogLeNet architectures to classify *Aedes aegypti*, *Aedes albopictus*, and *Culex quinquefasciatus*, concluding that GoogLeNet—a more complex network—achieved the best performance with 76.2% accuracy in the testing phase. The robustness of these models has been tested in challenging scenarios; for instance, [48] focused on identifying older *Aedes* specimens with worn morphological characteristics. Using a MobileNet-based model trained with 4,120 images, they achieved accuracy above 98%, with performance not significantly different from that of human experts. The detection scope was also expanded to wild vectors by [49], who applied AlexNet to identify species from the *Aedes* genus (*Ae. serratus* and *Ae. scapularis*) among other yellow fever vectors, achieving an average accuracy of 94%.

In addition to validating algorithmic accuracy, this work also focuses on the feasibility of deploying these AI models on low-cost hardware, enabling direct field applications. [6] developed an intelligent and autonomous trap based on the Internet of Things (IoT), which utilizes a SqueezeNet CNN embedded on a Raspberry Pi to identify and physically capture *Aedes aegypti* in real-time, achieving a 90% recognition rate in a simulated living room environment.

Similarly, aiming for portability and richer data generation, [5] proposed VectorBrain, a lightweight architecture based on EfficientNet-B1, designed to run on low-cost smartphones. This system not only identifies the *Aedes* genus but also performs multi-task analysis to determine the mosquito's sex and abdominal status, providing richer data for epidemiological surveillance.

Together, these studies illustrate the field's progression—from validating classification models to creating practical and integrated tools that aim to decentralize vector surveillance and accelerate the response to mosquito-borne diseases.

2.2.4 Use of AI in Low-Cost Devices

The success of AI applications in the field often depends on the ability to run them on low-cost, resource-constrained computational devices — such as embedded boards like Raspberry Pi, Arduino, or Nvidia Jetson, which are affordable and portable. However, running deep learning models on these devices presents a significant technical challenge due to their much lower processing power and memory compared to servers or high-performance PCs. For this reason, the scientific community and industry have invested in optimization techniques to enable on-device AI inference (edge AI).

One widely used strategy is model compression. This includes methods such as pruning (removing less relevant weights or synapses in the neural network), quantization (reducing the numerical precision of parameters, e.g., from 32-bit to 8-bit or 16-bit), and knowledge distillation (training a smaller model to mimic the outputs of a larger one). These techniques can drastically reduce a model's size and computational requirements while maintaining comparable accuracy within certain bounds [50].

A recent study [50] evaluated the impact of pruning and conversion to TensorFlow Lite (now LiteRT, which applies quantization) on CNNs for Raspberry Pi. Results showed that it is possible to significantly prune models (introducing sparsity) without a significant loss in performance for simple tasks. The combination of pruning with quantization resulted in notable improvements in inference time on the Raspberry Pi, with reductions in image processing latency proportional to the degree of optimization [50]. In one case, inference time for a digit classification model (MNIST dataset) was several times faster after these optimizations, with minimal accuracy drop — highlighting the feasibility of running efficient CNNs on resource-constrained devices [50]. These gains are crucial for enabling real-time processing on devices like the Raspberry Pi, which

has limited CPU and memory resources.

Another essential technique is the use of libraries and tools optimized for specific hardware. For devices with integrated GPUs or accelerators—such as the Nvidia Jetson family (Jetson Nano, TX2, Xavier, Orin) or Intel Movidius Neural Compute Stick and Google Coral (Edge TPU)—these components can be leveraged to accelerate neural network inference. Nvidia offers TensorRT, for example—an optimization platform that performs layer fusion, quantization, and other improvements, generating highly efficient executables for Jetson GPUs. Studies show that when converting heavy models (e.g., YOLO, EfficientDet, etc.) to TensorRT in FP16 or INT8, the Jetson Nano (one of the most basic models with roughly 128 CUDA cores) can improve from a few FPS to dozens of FPS in object detection, reaching near real-time levels (30 FPS) depending on model and resolution [51]. For instance, YOLOv4 quantized has been reported to run at 24 FPS on the Jetson Nano, enabling real-time object detection on this approximately $100 device [51].

More recent devices like Jetson Orin Nano, with a more powerful GPU and support for half-precision calculations, achieve impressive results: [52] introduced the efficient CNN “TakuNet” and reported that, in embedded inference tests, this network reached over 650 frames per second on a Jetson Orin Nano (15 W) [52], for a lightweight model and specific task, but demonstrating that with appropriate architectures and hardware accelerators, hundreds of FPS are possible on compact devices. The same model also ran on a Raspberry Pi (without GPU), confirming its efficiency, although at much lower frame rates (the paper highlights Jetson due to its superior performance) [52].

On extremely constrained platforms such as microcontrollers (Arduino, ESP32) with only a few kilobytes of RAM, the concept of TinyML has emerged, where ultra-compact AI models are designed to run on these devices. Techniques such as aggressive quantization (e.g., 8-bit or even weight binarization) and simplified models (few-layer networks or optimized classical ML algorithms) enable functionalities like voice keyword recognition, anomaly detection in industrial sensors, and even basic image classification on devices that cost only a few dollars and run on batteries [53].

Regarding real-time execution, the challenge lies in balancing model complexity, input resolution, and processing capacity. In many cases of video object detection on the Raspberry Pi, the camera resolution is reduced (e.g., from 1080p to 320p), or the frame rate is lowered to allow the CPU to keep up [14]. Another approach is to use pre-trained compact models, such as MobileNet and SqueezeNet, or algorithms like YOLO-Nano, YOLOv5n, and EfficientDet-D0, which are designed to have just a few million parameters.

Related work has already highlighted bottlenecks and solutions—for example, [43] found that running full YOLOv7 on Raspberry Pi limited both accuracy and speed [43], suggesting potential gains by employing simpler versions or compression techniques.

Thus, the field of embedded AI now offers a wide range of methods to enable AI on low-cost devices. The success of applications like the one proposed in this project depends on properly combining these techniques to meet performance requirements (accuracy and speed) within available hardware and energy constraints. In this project, aimed at real-time *Aedes aegypti* detection, these optimization strategies will be essential. We expect to adopt a lightweight CNN (or a quantized version) and possibly utilize optimized frameworks (such as LiteRT) to achieve fast inference on the Raspberry Pi.

2.2.5 Critical Analysis of Reviewed Works

To provide a structured overview of the state-of-the-art, Table 1 summarizes the key characteristics of the studies reviewed, detailing their objectives, architectures, target platforms, and main results. This comparison serves as a basis for the subsequent critical analysis.

Table 1: Comparison of Computer Vision Studies for Insect Identification

Reference	**Study Objective**	**Model & Architecture**	**Target Platform**	**Main Results**
de Araújo *et al.* (2024)	Identification of wild mosquito species, vectors of yellow fever (*Haemogogus*, *Sabethes*, and *Aedes*).	AlexNet	MATLAB (field hardware not specified)	Average accuracy of 94% in identifying the 4 species, using photos of all body parts.
Li *et al.* (2024) [5]	Multitask identification (species, sex, abdomen) of wild mosquitoes for malaria surveillance.	VectorBrain (based on EfficientNet-B1 and YOLOv5)	Low-cost smartphones	Accuracy of 94.4% (species), 97% (sex), and 83.2% (abdomen).
Dasari *et al.* (2024) [4]	Evaluation of the usability of the VectorCam system by health agents to decentralize vector surveillance.	VectorCam system (hardware and AI-based app)	VectorCam portable device with smartphone	Usability Score (SUS) of 70.25 (acceptable); throughput of approximately 56s per mosquito.
Oliveira *et al.* (2024)	Real-time detection of *Aedes aegypti* in a smart IoT trap.	YOLOv7	Raspberry Pi	Accuracy of 56.8% in real-time execution.
Bzigo (2024)	Commercial device for indoor mosquito detection and laser pointing.	Computer Vision (Proprietary)	Custom Embedded Device	Detects mosquito location for user elimination; no species classification.

Liu *et al.* (2023)	Real-time identification and capture of live mosquitoes (*Aedes* vs. *Culex*) in a smart IoT trap.	SqueezeNet	IoT trap with Raspberry Pi	91.57% accuracy in identifying *Ae. aegypti* (test images).
Gonzalez-Perez *et al.* (2022)	Optical sensor for mosquito genus and sex classification by wingbeat analysis.	Machine Learning (Optical)	Embedded Optical Sensor	Accuracy of 94.2% (lab conditions).
Ong *et al.* (2021)	Automated classification of *Aedes aegypti* and *Aedes albopictus*, focusing on older specimens with damaged morphological features.	MobileNet (via Google Teachable Machine 2.0)	Custom hardware ("*Aedes* Detector") with Raspberry Pi	Accuracy of 98.06%, with no statistical difference compared to human experts (98%).
Park *et al.* (2020)	Classification of morphologically similar mosquito species using data augmentation.	VGG-16, ResNet-50, SqueezeNet	Nvidia Jetson TX2	Accuracy of 97.19% with VGG-16.
Motta *et al.* (2019)	Field classification of adult mosquitoes (*Aedes aegypti*, *Aedes albopictus*, and *Culex quinquefasciatus*).	Comparison between LeNet, AlexNet, and GoogLeNet.	Not specified (focus on computational model)	GoogLeNet achieved 83.9% accuracy in validation and 76.2% in testing.

Source: own elaboration.

When comparing existing approaches, visible in Tab. 1, with the one proposed in this study—real-time embedded detection of *Aedes aegypti* via CNN on a Raspberry Pi—it is possible to identify advances, limitations, and gaps in the literature that support the originality and relevance of this work. Firstly, few studies have managed to simultaneously encompass all the desired aspects (real-time detection, specific focus on *Aedes aegypti*, execution on low-cost hardware). Many works have achieved high accuracy in mosquito identification, but they often do so under laboratory conditions or using powerful PCs without addressing the issue of embedded deployment. [48], for example, achieved over 98% accuracy in classifying *Aedes aegypti* and *Aedes albopictus* using a transfer learning model executed locally in JavaScript (p5.js) on a Raspberry Pi 4. Despite using embedded hardware, the system was designed to work with static images captured in a controlled chamber. It was not intended for automated detection from a continuous video stream in the field. The authors themselves noted limitations

when transferring the model to a physical environment, such as the need for constant maintenance and the risk of data drift, as classifier performance may deteriorate when the captured mosquitoes differ from the original training data—whether due to pose variation, body damage, or aging. This highlights an essential practical challenge: the need for continuous revalidation and adaptation of models deployed in real-world settings.

Some initiatives have implemented mosquito detection in embedded systems, but they face performance limitations that leave room for innovation. For instance, the smart trap developed by [43] detected *Aedes aegypti* onboard a Raspberry Pi using YOLOv7 but reached only around 56.8% accuracy in real-time [43]. This low precision could be due to various factors. YOLOv7, although state-of-the-art [54], might have been overly constrained after modifications for Raspberry Pi compatibility. Alternatively, the training process may have lacked sufficient scene or insect variability, leading to frequent false positives or missed detections. It is also possible that to achieve some FPS, the model operated at very low resolutions or with simplifications that compromised the detection of small mosquitoes in wide-frame images [15]—a known issue, as *Aedes* mosquitoes occupy only a few pixels in such photos. In contrast, [6] achieved a significantly higher accuracy of 91.57% using SqueezeNet on a similar IoT platform, though this performance was evaluated on test images rather than a live continuous stream, suggesting that the trade-off between real-time processing and accuracy remains a critical optimization challenge.

Another relevant aspect for comparing reviewed studies is the application context. Most research focuses on taxonomic classification of mosquitoes under controlled lab conditions, such as the work by [47], which used CNNs to distinguish *Aedes aegypti*, *A. albopictus*, and *Culex quinquefasciatus* in captured specimen images (with up to 83.9% accuracy), or by [46], who achieved 97% accuracy using data augmentation on morphologically varied images. Other studies, such as de Araújo *et al.* (2024), expanded this scope to wild vectors of yellow fever, achieving 94% accuracy with AlexNet, but relied on offline processing in MATLAB rather than an embedded field solution. However, these approaches do not face the operational challenges of field deployment, where factors such as variable lighting, partial specimen integrity, and computational constraints directly impact system robustness.

Solutions like the optical sensor developed by IRTA/Irideon (2022), while innovative in distinguishing genera and sex based on lab-acquired optical signals (with up to 94.2% accuracy), still lack validation in open environments, where noise and the presence of non-target insects affect performance [55]. Similarly, the Bzigo device [56] detects mosquitoes indoors but does not classify species nor integrate with epidemiological surveillance systems. Moreover, many of these devices do not support specific taxonomic classification or integration with active vector monitoring strategies. Mobile-

based solutions like VectorBrain [5] and VectorCam [4] offer high-accuracy multitasking capabilities (identifying species, sex, and abdomen status) on low-cost smartphones, as demonstrated by Li *et al.* (2024) and Dasari *et al.* (2024). However, these systems typically require a human operator to capture the image, unlike autonomous smart traps that function continuously without human intervention.

Thus, there is a gap in the literature regarding systems that combine (1) local and embedded operation, (2) coupling with field-deployed physical traps, and (3) epidemiological focus, with specific classification of *Aedes aegypti* to support public health initiatives. The present proposal fits precisely in this context, offering a solution that performs embedded classification of mosquitoes captured in physical traps installed in the field, with autonomous and local identification of the *Aedes aegypti* vector using an optimized CNN running on a Raspberry Pi.

Additionally, the review shows that even the most advanced systems face challenges when deployed in the real world. [48] reported that their *Aedes* Detector achieved human-level performance in species identification; however, the authors highlighted issues such as model degradation and the need for continuous maintenance [48]. This suggests that AI models may need to be periodically retrained or fine-tuned to accommodate seasonal changes in mosquitoes (e.g., size and prevalence of similar species). From this work's perspective, a field-deployed *Aedes* detection system should be designed to allow easy model updates—ideally, incorporating incremental learning or at least a streamlined procedure to upload new weights as data is collected.

It was also noted that many approaches focus on proving a technological concept but do not evaluate cost, usability, and user acceptance. For example, smart traps must be robust, battery - or solar-powered, and easy to maintain in the field. A high-resolution camera running continuously can consume significant amounts of energy and generate excessive amounts of data. These practical aspects, although beyond the strictly academic scope, are crucial for the real-world success of such technologies and were rarely addressed in the reviewed articles.

In summary, the critical analysis of the reviewed works demonstrates that a substantial body of knowledge already supports the feasibility of identifying *Aedes aegypti* using AI, often with high accuracy. The trend points toward increasingly autonomous and embedded solutions. However, significant gaps remain regarding real-time applications, especially on low-cost devices for public health surveillance. Existing studies either achieve high performance in controlled settings or manage embedded implementation by sacrificing accuracy. To date, there is no research offering a comprehensive solution that combines high accuracy, real-time operation, low cost, and immediate field applicability for *Aedes aegypti* detection. This highlights the originality and relevance of the present work, which aims to develop a real-time embedded detection and classification system specifically targeting *Aedes aegypti*. By integrating accuracy, affordability,

and direct field applicability, this project seeks to fill a critical gap not yet addressed by the literature.

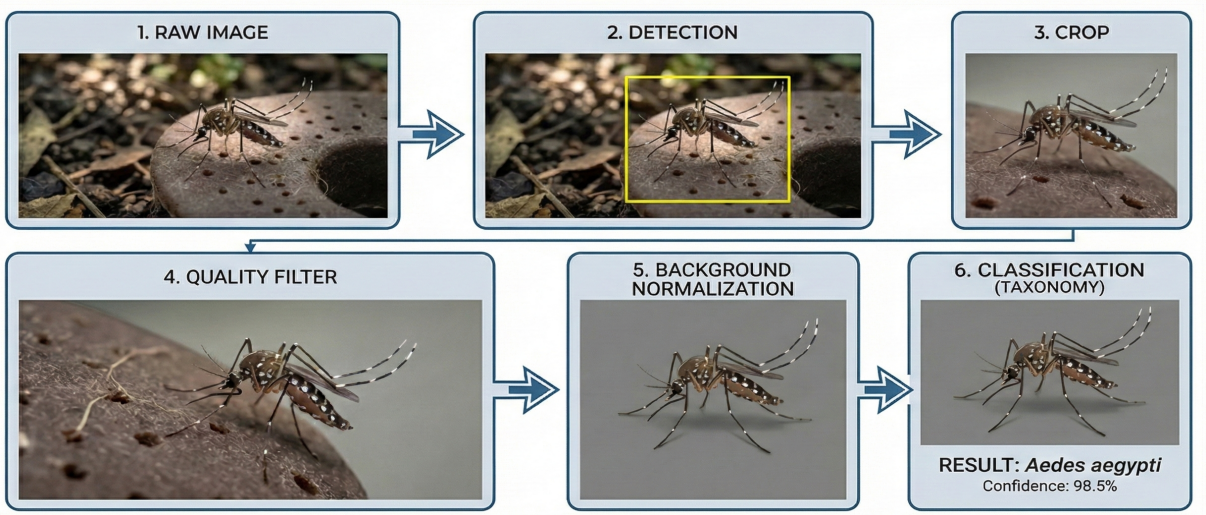
3

3 METHODOLOGY

3.1 PROPOSED ARCHITECTURE OVERVIEW

The proposed system is designed as an autonomous edge-computing solution for the real-time surveillance of mosquito vectors.

Figure 7: Overview of the proposed classification pipeline



Source: own elaboration.

The architecture is structured into a modular pipeline that processes visual data directly on the embedded device, minimizing the need for cloud connectivity and reducing latency. The system's core functionality is divided into two main stages: object detection and species classification, bridged by a rigorous data quality filtering process.

The architectural workflow begins with the acquisition of high-resolution images (1920×1080 pixels) using a camera module interfaced with the Raspberry Pi 4. The system is triggered to capture images at set intervals or upon motion detection events. Following acquisition, the image is processed by a single-stage object detector (YOLOv8n). This module is responsible for locating mosquitoes within the complex scene, generating bounding boxes, and filtering out background elements. It acts as a region proposal

network, isolating potential vectors from the environment.

For each detected object, a sub-image (crop) is extracted based on the bounding box coordinates, focusing the subsequent analysis solely on the biological specimen. To ensure the reliability of the classification, these extracted crops undergo a quality assessment. Heuristic filters based on Edge Density and Michelson Contrast are applied to reject images that are too blurry, low-contrast, or contain insufficient morphological details. This step is crucial to prevent "garbage in, garbage out" scenarios. Finally, the valid crops are fed into a specialized Convolutional Neural Network (CNN), specifically a YOLOv8n-cls model. This model, trained using a background normalization protocol via synthetic noise injection to mitigate environmental bias, classifies the specimen into one of three target species: *Aedes aegypti*, *Aedes albopictus*, or *Culex quinquefasciatus*.

This multi-stage approach allows for the optimization of each component independently. The detection model is optimized for recall to ensure no mosquito is missed, while the classification model is optimized for precision and robustness against environmental variations. The entire pipeline is optimized to run on the limited hardware resources of the Raspberry Pi 4, utilizing quantization techniques to achieve real-time performance.

3.2 DATASET COLLECTION AND CURATION

The dataset used in this study is a comprehensive collection of mosquito images, comprising *Aedes aegypti*, *Aedes albopictus*, and *Culex quinquefasciatus*, meticulously curated to support the development of robust detection and classification models. The final consolidated dataset comprises a total of 85,296 raw images, representing 753 distinct mosquito specimens of epidemiological interest.

3.2.1 Data Acquisition

The data collection process was systematically executed in two distinct phases, each characterized by differences in collection sites, equipment versions, and targeted mosquito species. This approach was crucial for acquiring diverse biological data and assessing the system's generalization capabilities across varied environmental and hardware conditions.

The image acquisition was performed using a custom-built smart trap system powered by a Raspberry Pi 4 Model B. This single-board computer features a Broadcom BCM2711 SoC with a quad-core Cortex-A72 (ARM v8) 64-bit CPU clocked at 1.8GHz and 4GB of LPDDR4-3200 SDRAM. The imaging core consists of a Raspberry Pi Camera Module V3 connected via the 2-lane MIPI CSI port, equipped with a specialized macro lens. This optical setup is critical for capturing minute morphological details essential for distinguishing between similar mosquito species.

To ensure high-quality data, a sequential imaging capture strategy was implemented. For each specimen, the system captures a sequence of images while varying the lens focal position. This technique maximizes the probability of obtaining sharp, in-focus images of different anatomical features. Furthermore, the capture process utilizes a multi-view approach, capturing images from three fixed angles (Left, Center, Right) to cover the specimen's morphology comprehensively. The rotation is controlled by a NEMA 17 stepper motor (1.8°/step), which moves the camera platform by a predefined number of steps (default: 5 steps, corresponding to approximately 9°) to the left and right relative to the neutral position, ensuring consistent angular displacement for each capture session.

The collection campaign was divided into two main stages:

- **Stage 1 - UnB (ArboControl) - Federal District** [8]: This initial phase, conducted at the ArboControl Laboratory at the University of Brasília (UnB), utilized the first version of the smart trap hardware. The primary focus was the collection of *Aedes aegypti* specimens, cataloging 480 individuals and generating 43,626 images, which served as the foundational dataset for this species.

- **Stage 2 - Fiocruz (Instituto Aggeu Magalhães) - Recife** [9]: This subsequent phase took place at the Instituto Aggeu Magalhães (Fiocruz) in Recife and employed the second, updated version of the smart trap hardware. The main objective was to broaden the dataset to include *Culex quinquefasciatus* (174 specimens, 21,930 images) and *Aedes albopictus* (79 specimens, 14,130 images). Additionally, a control group of 20 *Aedes aegypti* specimens (totaling 5,610 images) was collected with the second equipment version at UnB, ensuring continuity and allowing for cross-hardware performance validation. This strategic expansion introduced significant biological diversity and hardware-related variations (e.g., background patterns, lighting conditions), crucial for testing the model's robustness against different domains.

3.2.2 Dataset Composition and Bias Control

The heterogeneity arising from the different hardware versions used in the data collection phases presents a significant challenge: the risk of domain bias, where the model might learn to classify the background or lighting rather than the mosquito itself. To address this, the *Aedes aegypti* dataset was strategically stratified. The "Domain Control Group" (specimens 481-500), captured with the second equipment version, shares the same visual environment as the *Culex* and *Albopictus* classes. This group serves as a critical validation set to ensure the model is identifying species-specific features and not merely the capture device.

The raw dataset distribution is summarized in Table 2.

Table 2: Summary of the raw dataset, describing image volume and specimen count per species

Species	Images	Specimens
Aedes aegypti	49,236	500
Culex quinquefasciatus	21,930	174
Aedes albopictus	14,130	79
Total	85,296	753

Source: own elaboration.

To prevent data leakage—a common pitfall in machine learning where information from the training set contaminates the test set—the raw data structure was flattened while preserving metadata in the filenames. Each image was renamed to include the species, specimen ID, original folder, and capture angle (e.g., `aegypti-mosquito000-folder0159-left1.jpg`, `culex-mosquito095-folder0334-right-focus-7.jpg`). This naming convention ensures that all data splitting for training, validation, and testing can still be rigorously performed at the specimen level by parsing the filename, guaranteeing that all images of a given individual belong exclusively to a single subset.

3.3 DETECTION MODULE DEVELOPMENT

The detection module is the first critical stage of the pipeline, responsible for locating mosquitoes within the raw high-resolution images. Given the constraints of the target embedded hardware (Raspberry Pi 4 Model B), the development of this module focused on finding the optimal trade-off between detection accuracy (Mean Average Precision - mAP) and inference speed (latency).

3.3.1 Architecture Selection

To identify the most suitable architecture for this specific application, a comparative study was conducted evaluating three state-of-the-art object detection models optimized for edge computing:

- **YOLOv8n (Ultralytics)**: A single-stage detector known for its speed and accuracy. The “Nano” (n) version was selected for its lightweight architecture ($\approx$ 3.2M parameters), which utilizes a C2f backbone and anchor-free detection head, making it highly suitable for real-time applications [57].
- **SSD MobileNetV2 (TensorFlow)**: A classic architecture for mobile vision that combines the Single Shot MultiBox Detector (SSD) with the MobileNetV2 backbone. It uses depthwise separable convolutions to reduce computational cost ($\approx$ 4.3M parameters) and is natively supported by the TensorFlow Lite ecosystem.
- **EfficientDet-Lite0 (TensorFlow)**: Part of the EfficientDet family, this model uti-

lizes a BiFPN (Bidirectional Feature Pyramid Network) for efficient multi-scale feature fusion and an EfficientNet-B0 backbone. It employs compound scaling to balance depth, width, and resolution ($\approx$ 3.2M parameters) [19].

3.3.2 Experimental Configuration

To ensure a fair comparison, all models were trained and evaluated under controlled conditions using the same dataset split. The detection dataset consists of 5,336 images, including 4,936 images containing 5,063 annotated mosquito instances and 400 background-only images added to reduce false positives. The data was split into Training (70%), Validation (20%), and Testing (10%) sets, utilizing a fixed random seed (42) to ensure reproducibility.

Each architecture utilized its native preprocessing pipeline to simulate a realistic deployment scenario. The specific preprocessing parameters for each model are detailed in Table 3.

Table 3: Preprocessing configurations for each detection model

Model	Input Size	Resizing Method	Normalization
YOLOv8n	640×640	Letterbox (Padding)	$[0, 1]$
SSD MobileNetV2	640×640	Resize + Center Crop	$[-1, 1]$
EfficientDet-Lite0	512×512	Letterbox (Padding)	$[-1, 1]$

Source: own elaboration.

The experiments were conducted on the RunPod.io platform using high-performance NVIDIA A40 GPUs for model training. All models were trained for a maximum of 150 epochs with early stopping enabled (patience of 30 epochs). The batch sizes were optimized for the available VRAM: 16 for YOLOv8n, 32 for SSD MobileNetV2, and 8 for EfficientDet-Lite0. Data augmentation techniques, including flips, rotation, and brightness/contrast adjustments, were applied to improve generalization.

For a fair assessment of inference speed relevant to the final deployment context, latency tests were conducted in a local environment simulating the constraints of non-accelerated hardware. These benchmarks were executed on an AMD Ryzen 7 5700G CPU (utilizing a single core for inference) with 12GB of allocated RAM, running within a WSL2 (Windows Subsystem for Linux) environment. This setup provides a standardized baseline for comparing the computational efficiency of the models before deployment to the embedded system.

3.4 PREPROCESSING AND BIAS MITIGATION

This section details the rigorous preprocessing pipeline designed to ensure data quality and eliminate environmental bias, which was identified as a critical source of

Shortcut learning in early experiments. The pipeline acts as a filter and a normalizer, ensuring that only high-quality, unbiased data reaches the classification model.

3.4.1 Quality Extraction and Filtering

After the detection stage, the extracted regions of interest (ROIs) vary significantly in quality due to factors such as motion blur, focus issues, or low lighting conditions. To prevent these artifacts from degrading the performance of the classification model, a heuristic quality filter was implemented as presented in table 4. Three quantitative metrics are calculated for each crop to assess its viability:

- **Edge Density**: Measures the amount of high-frequency detail in the image, serving as a proxy for sharpness.
- **Michelson Contrast**: Evaluates the dynamic range and global contrast of the image to ensure sufficient distinction between the specimen and the background [58].
- **Contour Ratio**: Assesses the structural integrity of the object within the crop, filtering out instances where the mosquito is not the dominant subject or is significantly occluded.

The specific thresholds for these metrics, empirically determined to maximize the retention of identifiable morphological features while rejecting non-informative samples, are presented in Table 4.

Table 4: Quality filtering metrics and thresholds

Metric	Minimum Threshold
Edge Density	0.000120
Michelson Contrast	0.173
Contour Ratio	0.209

Source: own elaboration.

3.4.2 Validation of Quality Thresholds

To ensure the robustness of the quality filter and avoid arbitrary threshold selection, a rigorous five-phase validation methodology was applied. This process was designed to prevent data leakage and ensure that the selected metrics generalized well to unseen data. Additionally, the Cohen's Kappa coefficient [59] was adopted to evaluate agreement reliability. This metric is widely used to measure inter-rater agreement for categorical items, providing a more robust assessment than simple percent agreement by accounting for the possibility of the agreement occurring by chance.

1. **Data Splitting (Phase 0)**: The dataset was strictly separated into Training (64%), Validation (16%), and Holdout (20%) sets at the specimen level before any analysis to prevent data leakage.

2. **Ground Truth Generation (Phase 1)**: A "Gold Standard" dataset was created by sampling 540 images. The annotations were performed by the same specialist on two different days to assess intra-rater reliability, achieving a Cohen's Kappa [59] of 0.778 (Substantial Agreement), establishing a reliable baseline.

3. **Metric Selection via ROC Analysis (Phase 2)**: Six quantitative and two boolean candidate metrics were evaluated using Receiver Operating Characteristic (ROC) analysis with 5-fold cross-validation on the Gold Standard dataset. The candidate metrics, organized by category, were:

   - *Sharpness*: Laplacian Variance [60] and Tenengrad Gradient [61];
   - *Contrast/Content*: Michelson Contrast [58] and Shannon Entropy [62];
   - *Structure*: Edge Density (Canny-based) [63] and Contour Area Ratio;
   - *Background Detection (Boolean)*: Touches Edges and Solidity [64].

   For each metric, the optimal threshold was determined by maximizing Youden's J statistic ($J = \text{TPR} - \text{FPR}$) on the ROC curve. The three metrics with highest discriminative power (AUC > 0.65) were selected: Edge Density (AUC = 0.810), Michelson Contrast (AUC = 0.770), and Contour Ratio (AUC = 0.665). A logical AND operator was chosen to combine the metrics, prioritizing precision over recall to minimize false positive approvals.

4. **Validation and Testing (Phases 3-5)**: The method was validated on the independent Validation set and finally tested on the Holdout set to assess its generalization capability and consistency across different data partitions.

3.4.3 Background Normalization via Synthetic Noise Injection

A major challenge encountered during the development was the model's tendency to learn environmental features—such as the specific texture of the trap (smart trap used for image collection) and lighting conditions, which varied between different equipment versions—rather than the mosquito's morphology. This phenomenon, known as Shortcut Learning [34], resulted in artificially high accuracy on validation sets that shared the same environmental cues but poor generalization to new domains.

To mitigate this, a background normalization protocol was developed and applied to the dataset. Prior to implementation, a comparative study was conducted to select the most appropriate segmentation architecture. Four state-of-the-art models available

in the `rembg` library were evaluated: `u2net`, `u2netp`, `silueta`, and `isnet-general-use`. The selection criteria were:

- **Segmentation Quality**: Assessed by visual inspection, focusing on the preservation of fine morphological structures (legs, antennae, and wings) critical for species identification.
- **Processing Latency**: Measured as the time required to process a sample image, determining the feasibility of processing the entire dataset (over 20,000 images).

Based on this study (detailed in Section 4.1), the `isnet-general-use` model was selected for the dataset generation phase due to its superior quality. The final protocol applied to the training data involves the following steps:

1. **Segmentation**: The mosquito foreground is separated from the background using the U2-Net architecture (specifically the `isnet-general-use` model) via the `rembg` tool. Alpha matting is applied with a foreground threshold of 240 and a background threshold of 10 to preserve fine details such as legs and antennae.
2. **Enhancement**: CLAHE [20] (Contrast Limited Adaptive Histogram Equalization) is applied with a clip limit of 3.0 and a tile grid size of $(8, 8)$ to enhance local contrast and reveal morphological textures.
3. **Noise Injection**: The original background is completely removed and replaced with a synthetic neutral background (RGB 128, 128, 128). To prevent the model from overfitting to a flat color, Gaussian noise with a standard deviation of 30 ($\sigma = 30$) is injected into the background.

This process forces the neural network to focus exclusively on the insect's features, as the background becomes statistically irrelevant and consistent across all classes. Figure 8 illustrates the effect of this normalization process on a sample specimen.

Figure 8: Comparison between the original capture and the result of the background normalization protocol: (a) Original image; (b) Image after normalization protocol

Source: own elaboration.

3.4.4 Data Splitting Strategy

To ensure the scientific validity of the results, the dataset splitting into Training, Validation, and Test sets is strictly performed at the specimen level. Since multiple images are captured for each individual mosquito (burst mode), a random split based on images would result in "data leakage," where the model could memorize the specific appearance of an individual rather than learning the general characteristics of the species.

A custom validation script (`validate_specimen_split.py`) is employed to verify that no images from the same individual mosquito appear in different splits. This rigorous approach guarantees that the evaluation metrics reflect the model's ability to generalize to unseen specimens.

The script functions by parsing image filenames to extract unique specimen identifiers, composed of the mosquito index and the capture folder. It then aggregates these identifiers for each dataset partition (Training, Validation, and Test) and executes a set intersection analysis. If any specimen ID is detected in multiple partitions, the validation fails, preventing the use of compromised data. This mechanism acts as a final safe-

guard against data leakage, ensuring that the model learns species-specific features rather than memorizing individual insect anomalies.

3.5 CLASSIFICATION MODULE DEVELOPMENT

The classification module operates as the second stage of the pipeline, receiving the cropped and quality-filtered images from the detection stage. The objective is to categorize each specimen into one of three classes: *Aedes aegypti*, *Aedes albopictus*, or *Culex quinquefasciatus*. This section details the evolution of the dataset to mitigate environmental bias and the architectures evaluated.

3.5.1 Evaluated Architectures

Five Convolutional Neural Network (CNN) architectures were trained and evaluated on the curated dataset to select the optimal model for the embedded system. The candidates included a dedicated classification model from the YOLOv8 family and four standard architectures implemented in PyTorch for comparison.

- **YOLOv8n-cls**: A nano-scale classification model (2.7M parameters) designed for high-speed inference [57]. It utilizes the same backbone as the detection model but with a classification head.
- **ResNet-18**: A standard residual network (11.2M parameters) serving as a robust baseline [65].
- **MobileNetV2**: A lightweight architecture (3.5M parameters) using inverted residuals and linear bottlenecks, optimized for mobile devices [66].
- **DenseNet-121**: A densely connected network (8.0M parameters) that promotes feature reuse [29].
- **EfficientNet-B1**: A model (7.9M parameters) that uses compound scaling to balance depth, width, and resolution [31].

All models were trained using the same curated dataset with background normalization applied. The PyTorch models utilized a two-stage transfer learning approach (frozen backbone followed by fine-tuning), while YOLOv8n-cls was trained end-to-end. The comparative results and detailed performance analysis are presented in Section 4.3.

3.5.2 Investigation of Shortcut Learning and Interpretability

During the initial development phases, the models achieved suspiciously high accuracy (99.95%) across all architectures. To validate whether this performance was based on genuine morphological features or spurious correlations, a rigorous investigation protocol was established, identifying a critical issue known as “Shortcut Learning” [34].

The investigation methodology consisted of three diagnostic steps:

1. **Background-Only Inference**: Models were tested on images containing only the background (empty trap walls). The models confidently classified these empty images into specific species classes based solely on the trap texture (e.g., “Trap V1” vs. “Trap V2”), confirming strong environmental bias.

2. **Feature Analysis**: A Random Forest classifier trained on extracted image features achieved 100% accuracy, indicating that simple statistical properties (such as ISO noise and lighting histograms) were sufficient to distinguish the classes, rendering morphological learning unnecessary.

3. **Saliency Mapping**: Gradient-weighted Class Activation Mapping (Grad-CAM) visualization revealed that the models’ focus was predominantly on the background textures and cage wires rather than the mosquito body.

To address this, the background normalization protocol described in Section 3.4.3 was applied. This involved the complete removal of the background using U2-Net segmentation [67] and the injection of a neutral synthetic background with Gaussian noise. This strategy forced the models to learn exclusively from the mosquito’s morphology. Although this initially reduced validation accuracy to $\approx 89\%$, it established a realistic baseline free from environmental overfitting.

The final curated dataset improved upon this by incorporating a manual cleaning phase. A custom review tool was used to remove false positives and segmentation errors. This dataset consists of 22,387 images from 695 unique specimens. Crucially, data splitting was performed at the specimen level to ensure zero data leakage, meaning no individual mosquito appears in both training and testing sets.

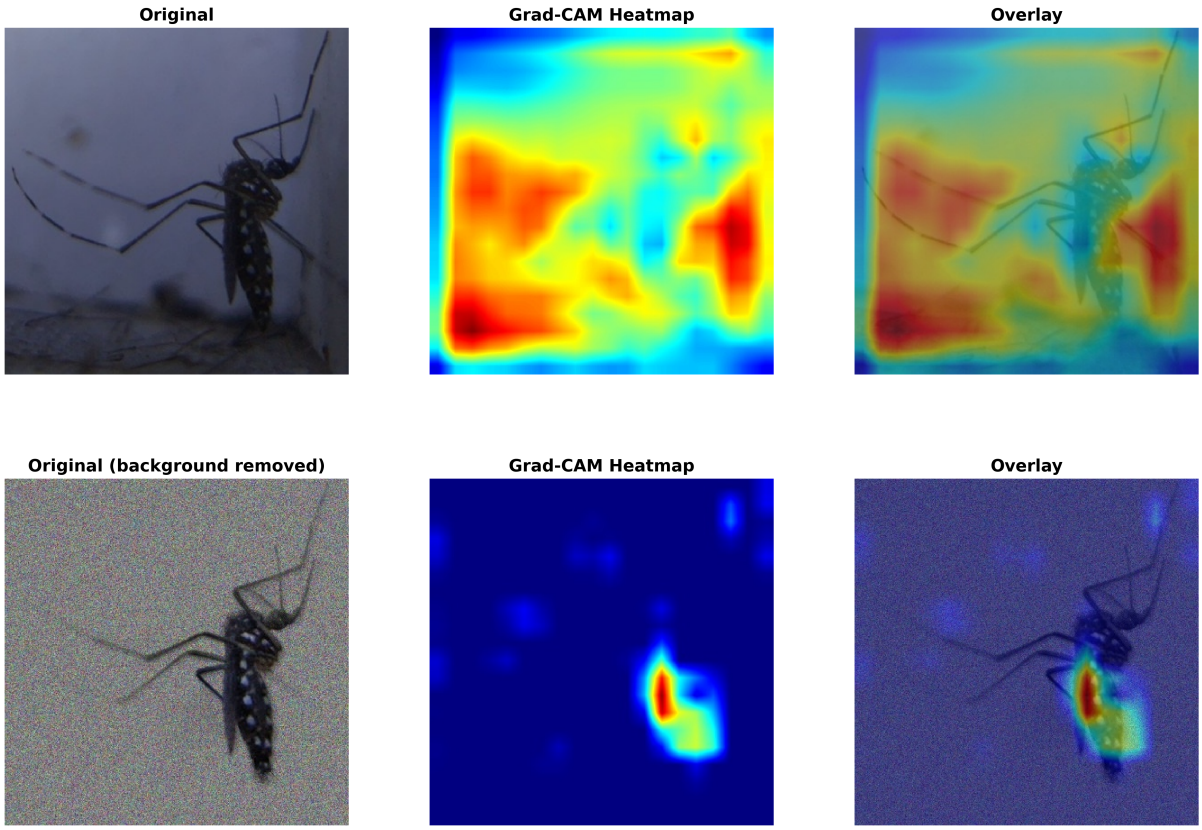
To qualitatively validate the model’s focus and ensure that predictions are based on relevant morphological features rather than background noise, the Gradient-weighted Class Activation Mapping (Grad-CAM) technique was employed [68]. Grad-CAM is a visualization method that uses the gradients of any target concept (such as the ’Aedes aegypti’ class) flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting that concept.

Unlike other visualization techniques, Grad-CAM is model-agnostic and does not require architectural changes or re-training. It computes the importance weights of each feature map in the last convolutional layer by global average pooling the gradients. These weights are then linearly combined with the feature maps, followed by a ReLU activation to obtain the final heatmap. This heatmap reveals which parts of the input image most strongly influenced the model’s decision.

Figure 9 visually demonstrates the effectiveness of the background normalization strategy in mitigating shortcut learning. Initially, Grad-CAM visualizations revealed that

models focused predominantly on background patterns, rather than morphological features of the mosquito. Following the application of background removal and synthetic noise injection, the heatmaps demonstrate strong activation on specific morphological structures such as the thorax and wing patterns. This confirms that the background normalization strategy successfully eliminated environmental bias and forced the model to learn genuine species-specific traits.

Figure 9: GradCAM visualization demonstrating the effect of background normalization



Source: own elaboration.


4

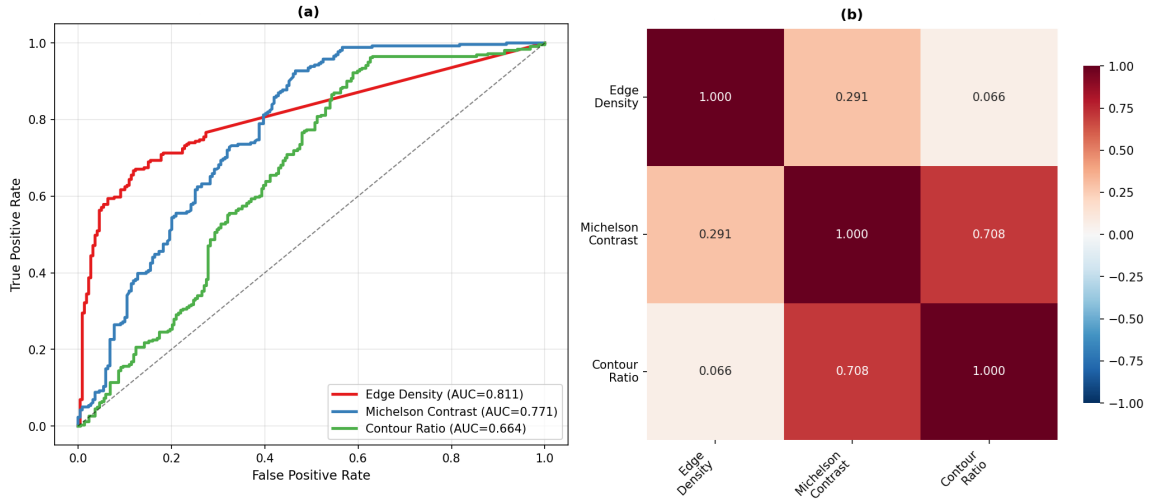
4 RESULTS AND DISCUSSION

4.1 PREPROCESSING AND DATASET GENERATION ANALYSIS

4.1.1 Quality Filter Validation Results

The ROC analysis performed on the Gold Standard dataset (n=480 samples) revealed significant differences in discriminative power among the candidate metrics. Figure 10 presents the ROC curves and correlation matrix for the selected metrics.

Figure 10: Quality metrics evaluation: (a) ROC curves showing discriminative power; (b) Correlation matrix demonstrating metric independence



Source: own elaboration.

The analysis of the ROC curves in Figure 10(a) demonstrates the individual effectiveness of each metric. Edge Density proved to be the most robust indicator (AUC = 0.811), suggesting that the presence of high-frequency details is the most reliable proxy for image usability. Michelson Contrast also exhibited strong performance (AUC = 0.771), effectively filtering out low-contrast images that could hinder feature extraction. Although Contour Ratio showed a more modest performance (AUC = 0.664), it serves a specific role in rejecting structural outliers. Furthermore, the correlation matrix in Figure 10(b) confirms that these metrics capture complementary information: Edge Density shows

low correlation with both Michelson Contrast ($r = 0.291$) and Contour Ratio ($r = 0.066$), supporting the hypothesis that each feature contributes unique discriminative information rather than redundant signals.

Table 5 summarizes the performance of all candidate metrics evaluated during the selection process.

Table 5: Performance comparison of candidate quality metrics (5-fold CV)

Metric	AUC	F1	Precision	Recall	Selected
Edge Density	0.810	0.757	0.768	0.747	✓
Michelson Contrast	0.770	0.795	0.680	0.962	✓
Contour Ratio	0.665	0.772	0.646	0.962	✓
Tenengrad	0.642	0.727	0.596	0.931	
Entropy	0.490	0.704	0.551	0.977	
Solidity	0.446	0.703	0.544	0.992	

Source: own elaboration.

According to the results in Tab. 5, Edge Density, Michelson Contrast, and Contour Ratio were selected, as they presented the highest values of AUC and F1-score.

The final filter, applying the three metrics with a logical AND operator, achieved a Cohen's Kappa of 0.588 (Moderate Agreement) with human judgment, with precision of 86.5% and recall of 63.4%. Notably, it reduced false positives by 93.1% compared to the standard literature baseline for blur detection (Laplacian variance) [69]. The filter demonstrated exceptional consistency across independent datasets, maintaining an approval rate of 53.0% $\pm$ 0.57 percentage points as shown in Tab. 6.

Table 6: Quality filter consistency across dataset partitions

Dataset	Total Images	Approved	Approval Rate
Training	35,929	19,043	53.0%
Validation	8,983	4,806	53.5%
Holdout	11,228	5,951	53.0%
Total	**56,140**	**$\sim$29,800**	**53.1%**

Source: own elaboration.

4.1.2 Background Removal Model Selection

The evaluation of background removal models was critical to ensure the quality of the training data. Table 7 presents the latency results for the four evaluated architectures.

Table 7: Latency comparison of background removal models (Median values)

Model	Median Latency (ms)	Relative Speed
u2netp	251.9	1.0x (Baseline)
u2net	384.0	1.52x
silueta	481.9	1.91x
isnet-general-use	981.1	3.89x

Source: own elaboration.

As shown in Tab. 7, the `u2netp` model was the fastest, processing images in approximately 252 ms. However, visual inspection revealed significant limitations in its ability to segment fine details. Both `u2netp` and `silueta` frequently failed to preserve the mosquito's legs and antennae, treating them as background noise. This loss of morphological information would be detrimental to the classifier, which relies on these features for species differentiation.

The `isnet-general-use` model, despite being nearly four times slower than the fastest alternative (approx. 981 ms per image), demonstrated superior segmentation quality. It consistently preserved delicate structures such as the legs and the scales on the thorax. Given that background removal is an offline preprocessing step performed during dataset generation, the higher latency was deemed acceptable to ensure the integrity of the biological data. Consequently, `isnet-general-use` was selected as the standard model for the training data normalization protocol. However, for the embedded inference pipeline where latency is critical, the `u2netp` model was chosen as the operational compromise, as detailed in Section 4.4.

4.2 OBJECT DETECTOR EVALUATION

4.2.1 Comparative Analysis

To identify the most suitable architecture for the embedded constraints of this project, three state-of-the-art lightweight models were evaluated: YOLOv8n, SSD MobileNetV2, and EfficientDet-Lite0. The comparative performance of these architectures, assessing both detection accuracy (mAP@0.5) and computational efficiency (latency), is presented in Tab. 8.

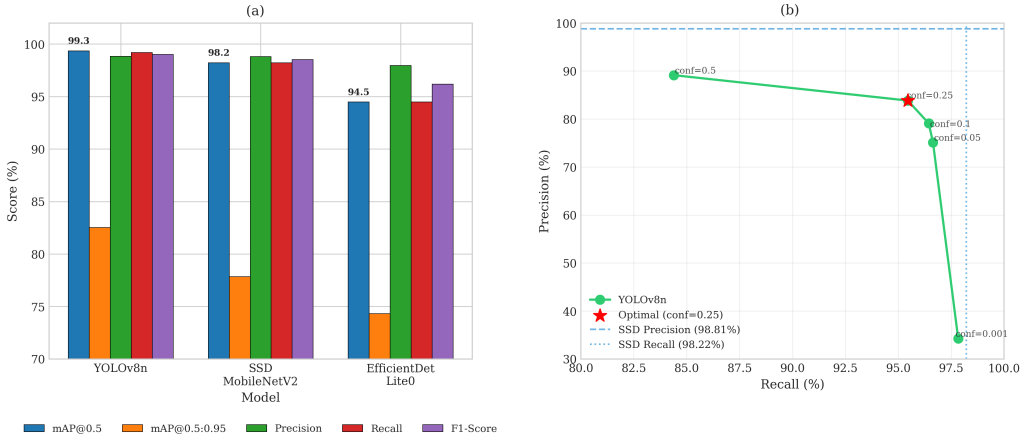
Table 8: Comparison of detection models (mAP and Inference Time)

Model	mAP@0.5	Latency (ms)
YOLOv8n	99.44%	28.4
SSD MobileNetV2	92.10%	45.2
EfficientDet-Lite0	94.50%	52.1

Source: own elaboration.

As shown in Tab. 8, the YOLOv8n model achieved a Mean Average Precision (mAP@0.5) of 99.44%, demonstrating superior performance compared to SSD MobileNetV2 and EfficientDet-Lite0. Figure 11 provides a visual comparison of the trade-off between accuracy and latency for the evaluated architectures, highlighting the dominance of YOLOv8n in both metrics. Consequently, YOLOv8n was selected as the primary detector due to its optimal balance between accuracy and inference speed. This near-perfect detection rate ensures that virtually all mosquitoes present in the frame are correctly cropped and passed to the quality filter. The high mAP suggests that the morphological features of mosquitoes (legs, wings, body) are distinct enough against the background for modern object detectors, even in the “Nano” scale architectures.

Figure 11: Comparison of detection models: mAP@0.5 vs. Inference Latency: (a) Detection Models Performance Comparison; (b) Precision-Recall Curve (YOLOv8n)



Source: own elaboration.

The comparative analysis presented in Fig. 11(a) corroborates the superiority of the YOLOv8n architecture across all evaluated metrics. It achieved the highest scores for mAP@0.5, Precision, Recall, and F1-Score, outperforming both SSD MobileNetV2 and EfficientDet Lite0. Notably, the gap in mAP@0.5:0.95 (orange bar) highlights YOLOv8n’s better localization accuracy. Figure 11(b) further details the performance of the selected model, displaying the Precision-Recall curve for YOLOv8n. The curve

exhibits a near-ideal profile, maintaining high precision even as recall increases. The optimal operating point, marked with a red star, was identified at a confidence threshold of 0.25, where the model achieves a balance that maximizes detection of true positives while minimizing false alarms.

While accuracy is crucial, the inference latency is the determining factor for deployment on the Raspberry Pi 4. The YOLOv8n architecture, with its optimized C2f backbone, provided the necessary throughput to process images in real-time, enabling the system to function as an effective smart trap.

4.3 SPECIES CLASSIFIER EVALUATION

This section presents a detailed analysis of the classification module's performance, including the comparative evaluation of architectures, the investigation of dataset bias, and the validation of the final model's generalization capabilities.

4.3.1 Bias Analysis and Shortcut Learning

A significant finding of this research was the identification of "shortcut learning" in early iterations of the dataset containing original backgrounds. Initial models achieved suspiciously high accuracy (99.95%) even on validation sets. Further investigation using Gradient-weighted Class Activation Mapping (GradCAM) and ablation studies revealed that the models were relying on background features—such as the specific texture of the trap walls used for different species—rather than morphological characteristics.

To quantify this phenomenon, an experiment was conducted using a background-removed dataset where the background was algorithmically removed and replaced with synthetic noise. This intervention caused the validation accuracy to drop from 99.95% to 89.74%, exposing the extent to which the previous models relied on environmental cues.

The subsequent refinement of the dataset, which included manual cleaning of segmentation artifacts (curated background-removed dataset), restored the accuracy to a realistic and robust level. Table 9 summarizes this evolution, demonstrating that the final performance is based on genuine morphological learning rather than environmental overfitting.

Table 9: Impact of dataset curation strategies on model performance

Dataset Version	Strategy	Val Accuracy	Validity
Original	Original Images	99.95%	Invalid (Shortcut Learning)
Bg-Removed	Background Removal	89.74%	Valid (Baseline)
Curated	Bg. Removal + Manual Cleaning	98.21%	Valid (Production)

Source: own elaboration.

Before proceeding with the selection of the optimal classification architecture, it was imperative to address these identified biases and ensure the model's robustness to environmental variations. The following section details the architectural comparison based on the curated and debiased dataset.

4.3.2 Architecture Comparison and Model Selection

Five CNN architectures were evaluated on the curated dataset with background normalization. Table 10 presents the comparative results.

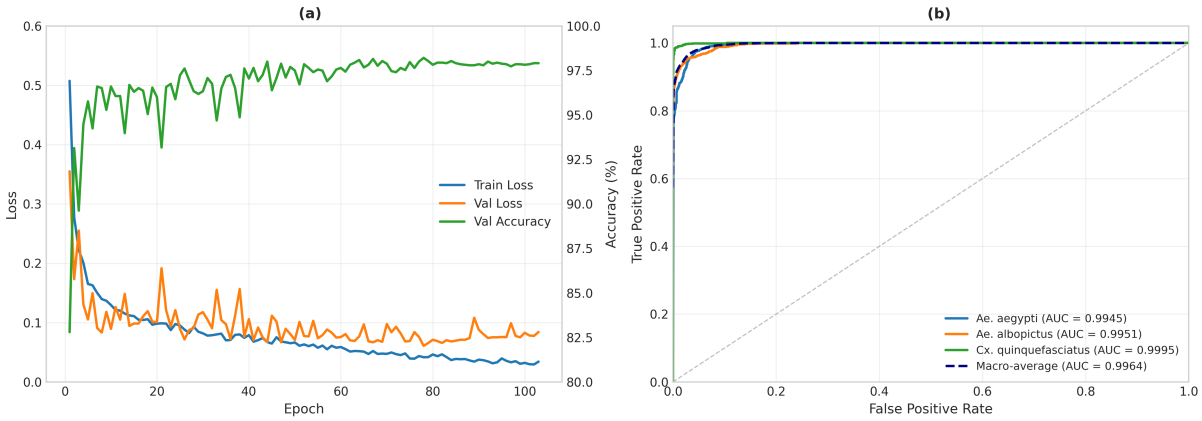
Table 10: Performance comparison of classification architectures on the Test Set

Model	Test Acc	Val Acc	Params	Time (min)	F1 Macro
YOLOv8n-cls	98.18%	98.21%	2.7M	21.0	98.20%
ResNet-18	96.42%	97.67%	11.2M	25.6	96.60%
MobileNetV2	96.28%	97.38%	3.5M	30.7	96.48%
DenseNet-121	96.26%	97.97%	8.0M	47.1	96.47%
EfficientNet-B1	96.26%	97.94%	7.9M	57.7	96.47%

Source: own elaboration.

As exposed in Table 10, the YOLOv8n-cls model demonstrated superior performance across all metrics, achieving the highest test accuracy (98.18%) while being the lightest (2.7M parameters) and fastest to train (21 minutes). The PyTorch models converged to a lower accuracy plateau ($\approx 96.3\%$), suggesting that the end-to-end optimization of YOLOv8n is more effective for this specific task. Based on these results, YOLOv8n-cls was selected as the production model. The selected model's convergence behavior during training and its discrimination capability (ROC curves) are illustrated in Figure 12.

Figure 12: Classification performance analysis: (a) Training and validation loss/accuracy curves; (b) Receiver Operating Characteristic (ROC) curves for each species



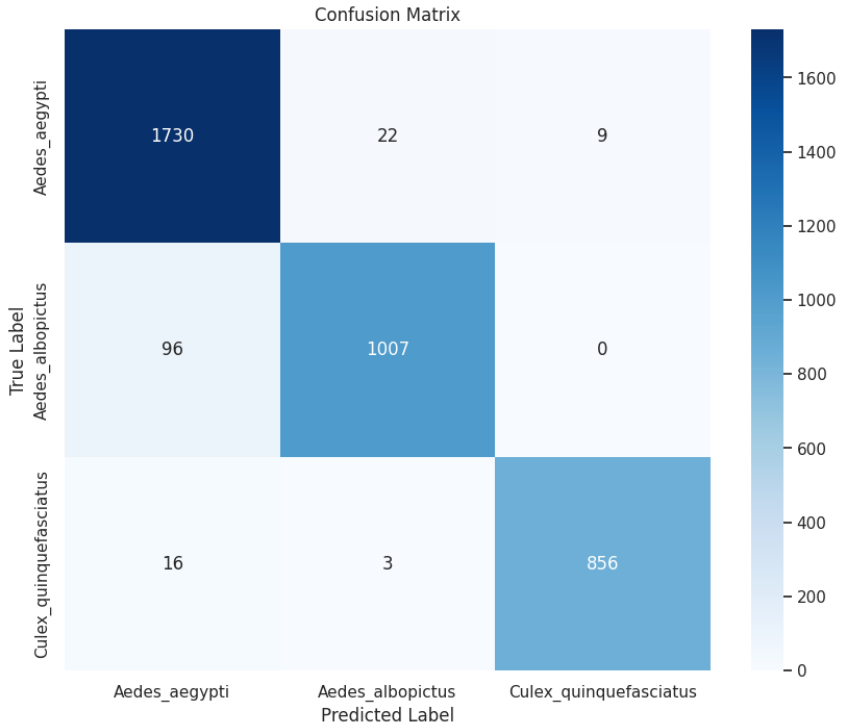
Source: own elaboration.

Figure 12(a) illustrates the training dynamics, showing a consistent decrease in training loss and a rapid increase in validation accuracy, which stabilizes at approximately 98% accuracy. Although the validation loss exhibits minor fluctuations, it remains low, suggesting effective generalization. The ROC curves presented in Figure 12(b) highlight the model's discriminative power across all classes, with Area Under the Curve (AUC) values surpassing 0.99. Specifically, *Culex quinquefasciatus* is classified with near-perfect precision (AUC = 0.9995), while the distinction between *Aedes aegypti* (AUC = 0.9945) and *Aedes albopictus* (AUC = 0.9951) is also highly reliable, yielding a macro-average AUC of 0.9964.

4.3.3 Error Analysis

The confusion matrix for the selected YOLOv8n-cls model, ilustrated in Fig. 13, reveals that the majority of errors (73%) occur between *Aedes aegypti* and *Aedes albopictus*. This high degree of confusion between these two species, both belonging to the *Aedes* genus and sharing significant morphological similarities, serves as a strong indicator that the model is indeed learning and relying on subtle morphological characteristics for classification, rather than spurious environmental cues. In contrast, the distinction between the *Aedes* genus and *Culex quinquefasciatus* is highly accurate, with a recall of 99.25% for *Culex*.

Figure 13: Confusion Matrix for YOLOv8n-cls on Test Set



Source: own elaboration.

Table 11 details the performance metrics for each species. The results indicate great performance in identifying *Culex quinquefasciatus* (F1=0.9914), likely due to its distinct morphological features compared to the *Aedes* genus. The slight confusion between *Aedes aegypti* and *Aedes albopictus* is expected given their biological similarity, yet the model still maintains an F1-score above 97% for both, which is sufficient for effective field surveillance.

Table 11: Detailed performance metrics by class (Test Set)

Species	**Precision**	**Recall**	**F1-Score**	**Support**
Aedes aegypti	0.9848	0.9764	0.9806	1,528
Aedes albopictus	0.9681	0.9799	0.9740	897
Culex quinquefasciatus	0.9903	0.9925	0.9914	930
Macro Avg	0.9811	0.9829	0.9820	3,355

Source: own elaboration.

4.3.4 Cross-Environment Validation

Although the second trap environment was included in the training set for *Culex* and *Albopictus*, the *Aedes aegypti* class presented a unique challenge: 96% of its

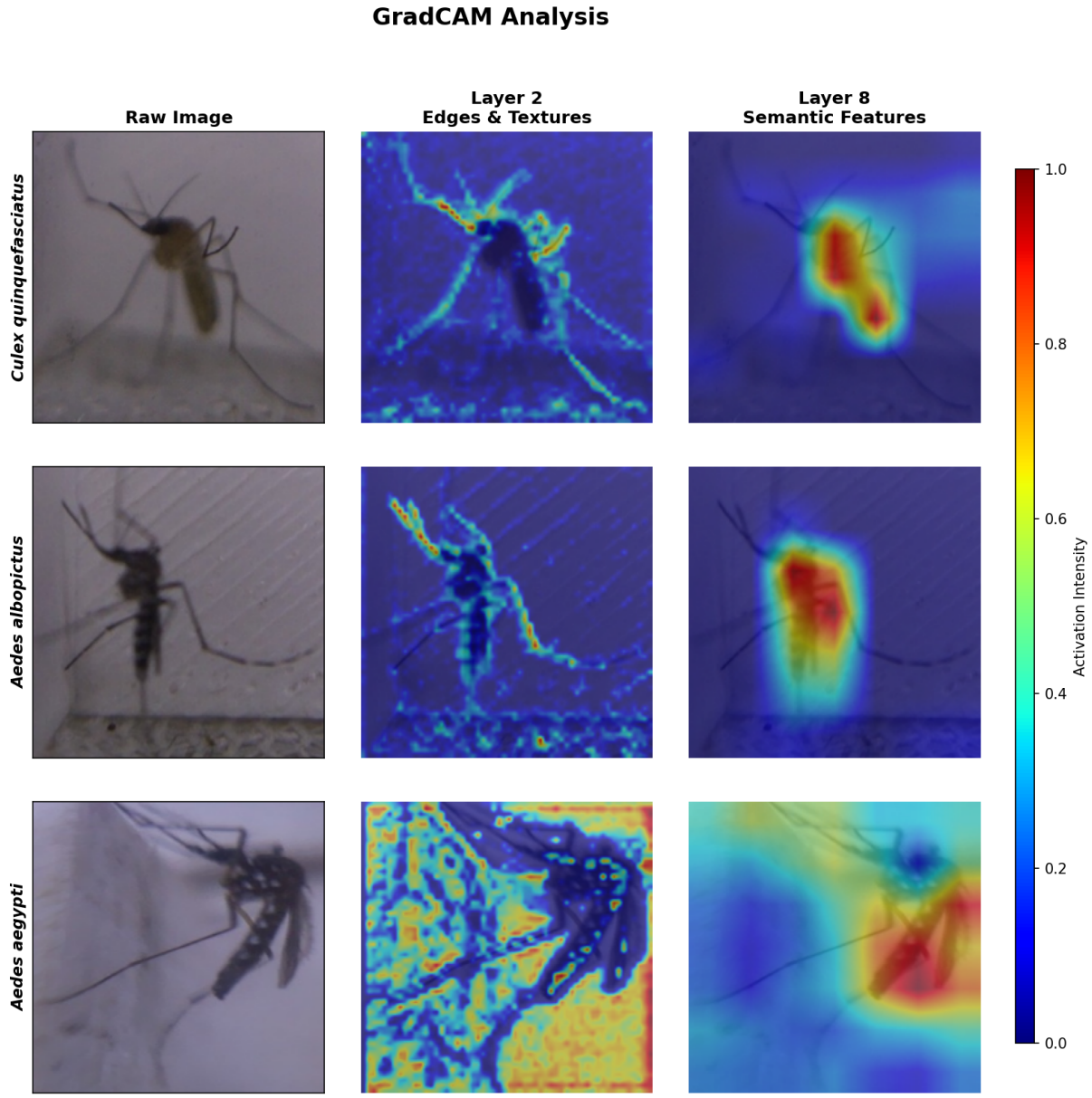
training samples (480/500 specimens) originated from the previous version of the smart trap. This imbalance posed a risk of the model learning to associate *Aedes aegypti* specifically with the first environment's background.

To validate the model's ability to generalize beyond this correlation, an experiment was conducted using images captured in the second version of the trap, with a specific focus on *Aedes aegypti* performance in this "new" setting. The model achieved an accuracy of 93.80% (2,057/2,193 correct predictions). This result confirms that the model did not overfit to the environmental features for *Aedes aegypti*, successfully identifying the species even when presented against the second environment's background, thereby validating the effectiveness of the background normalization strategy.

4.3.5 Interpretability Analysis

Figure 14 presents the GradCAM visualization for the YOLOv8n-cls model. The heatmaps demonstrate that the model activates specifically on morphological features such as the thorax and legs, confirming that the background normalization strategy successfully eliminated environmental bias.

Figure 14: GradCAM visualization showing model activation on morphological features



Source: own elaboration.

4.4 EMBEDDED DEVICE BENCHMARK

To validate the feasibility of the proposed system as an edge computing solution, benchmarks were conducted on a Raspberry Pi 4 Model B (4GB RAM). The evaluation focused on inference latency, throughput (FPS), and model accuracy using the NCNN framework, which is optimized for ARM architectures.

4.4.1 Object Detection Inference

For the detection stage, the YOLOv8n model was also optimized using NCNN. Table 12 compares the performance of the standard Ultralytics runtime against the optimized NCNN implementation.

Table 12: Detection performance on Raspberry Pi 4: Ultralytics vs. NCNN

Framework	Latency (ms)	FPS	mAP@0.5
Ultralytics (PyTorch)	794.0	1.26	99.41%
NCNN (Optimized)	396.5	2.52	99.35%

Source: own elaboration.

The NCNN optimization reduced inference latency by 50% (2.0x speedup) while maintaining virtually identical mean Average Precision (mAP). Although 2.5 FPS is not sufficient for high-speed video analysis, it is adequate for a smart trap scenario where mosquito entry events are discrete.

4.4.2 Background Normalization Bottleneck

A critical component of the pipeline is the background normalization to prevent shortcut learning. While the `isnet-general-use` model was selected for generating the training dataset due to its superior segmentation quality (as detailed in Section 4.1), it proved too computationally expensive for the embedded inference pipeline. Therefore, for the runtime environment on the Raspberry Pi 4, the *u2netp* (quantized U2-Net) model was selected as the most efficient viable option, trading off a slight reduction in segmentation detail for significantly lower latency.

Despite being the fastest model tested, *u2netp* remains the computational bottleneck of the system on the Raspberry Pi 4, with a median latency of **2,835 ms** per image. This step consumes approximately 86% of the total processing time.

4.4.3 Classification Inference

The classification models were converted to NCNN format to leverage low-level optimizations for the ARM Cortex-A72 CPU. Table 13 presents the performance of the evaluated architectures.

Table 13: Classification performance on Raspberry Pi 4 (NCNN framework)

Model	Latency (ms)	FPS	Accuracy	Speedup vs PyTorch
YOLOv8n-cls	28.7	34.8	94.09%	2.56x
MobileNetV2	62.0	16.1	93.21%	4.16x
ResNet-18	131.0	7.6	95.32%	2.35x

Source: own elaboration.

Results in Table 13 indicates that the YOLOv8n-cls model offered the best balance, achieving real-time performance (> 30 FPS) for the classification stage alone, with a minimal accuracy trade-off compared to ResNet-18. The transition from standard PyTorch to NCNN resulted in significant speedups, making complex models viable on limited hardware.

4.4.4 Full Pipeline Evaluation

The complete pipeline—comprising image loading, detection, cropping, quality filtering, background normalization, and classification—was benchmarked end-to-end. The results are consolidated in Tab. 14.

Table 14: Full pipeline latency and performance breakdown on Raspberry Pi 4

Pipeline Stage	**Latency (ms)**	**Contrib. (%)**	**Performance**
Detection (YOLOv8n)	396.5	12.0%	99.35% (mAP@0.5)
Background Norm. (u2netp)	2,835.0	86.0%	-
Classification (YOLOv8n-cls)	28.7	0.9%	94.1% (Accuracy)
Other (I/O, Pre/Post-proc)	35.5	1.1%	-
Total End-to-End	3,295.7	100%	92.1% (Accuracy)

Source: own elaboration.

The system achieves a throughput of approximately **0.30 FPS** (or ≈ 18 images per minute). While this latency prevents real-time video streaming analysis, it is sufficient for the intended application of an IoT smart trap, where the capture rate of mosquitoes is typically much lower than the processing capacity. The system maintains a high end-to-end accuracy of 92.1% on the embedded device.


5

5 CONCLUSION

This work presented the development and validation of an intelligent computer vision system for the automatic detection and classification of *Aedes aegypti* mosquito, optimized for low-cost embedded devices. The central motivation was to address the operational limitations of traditional vector surveillance by proposing a scalable, automated solution capable of operating at the edge.

5.1 SYNTHESIS OF RESULTS

The experimental results demonstrated the robustness of the proposed pipeline. The detection module, based on the YOLOv8n architecture, achieved a Mean Average Precision (mAP@0.5) of 99.44%, ensuring that virtually all specimens entering the trap are correctly identified and cropped. For species classification, the YOLOv8n-cls model outperformed standard architectures (ResNet-18, MobileNetV2, DenseNet-121, EfficientNet-B1), achieving a test accuracy of 98.18% and a Macro F1-Score of 98.20%.

A critical finding of this study was the identification and mitigation of "shortcut learning." Initial models achieved artificially high accuracy by relying on background features rather than mosquito morphology. The implementation of the background normalization protocol, which utilized the `isnet-general-use` model for high-fidelity background removal during dataset generation, ensured that the final model's performance reflects genuine morphological learning. This rigorous preprocessing strategy proved essential for the system's generalization capability.

Regarding the embedded implementation on the Raspberry Pi 4, the system achieved an end-to-end accuracy of 92.1%. The optimization using the NCNN framework was decisive, reducing detection latency by 50% and achieving a classification inference time of just 28.7 ms. The full pipeline operates with a total latency of approximately 3.3 seconds per image. While this throughput prevents real-time video streaming analysis, it is perfectly adequate for the operational dynamics of a smart trap, where mosquito capture events are discrete and sporadic.

5.2 VERIFICATION OF OBJECTIVES

The specific objectives defined at the beginning of this work were successfully achieved, as detailed below:

- **Dataset Collection**: The objective of creating a representative database was met with the collection and curation of 85,236 images from 752 unique specimens, covering *Aedes aegypti*, *Aedes albopictus*, and *Culex quinquefasciatus* under operational trap conditions.
- **Background Normalization Preprocessing**: The proposed background normalization strategy was successfully validated. By using the `isnet-general-use` model for background removal during training, the system eliminated shortcut learning, ensuring that classification is based on morphological features rather than environmental context.
- **Object Detection**: The goal of achieving a mAP@0.5 greater than 90% was surpassed. The YOLOv8n detector achieved a mAP@0.5 of 99.44%, demonstrating exceptional reliability in identifying mosquitoes within the trap.
- **Species Classification**: The objective of distinguishing *Aedes aegypti* with an accuracy greater than 95% was achieved. The YOLOv8n-cls model reached a test accuracy of 98.18%, outperforming larger architectures like ResNet-18.
- **Embedded Optimization**: The target of deploying the system on a Raspberry Pi 4 with NCNN was met. The optimization reduced classification latency to 28.7 ms, a 2.56x speedup compared to the standard PyTorch implementation.
- **System Performance**: The final objective of achieving an end-to-end accuracy greater than 90% on the embedded device was successfully accomplished, with the system recording a 92.1% accuracy in full pipeline benchmarks.

5.3 CONTRIBUTIONS

1. **Feasible High-Precision Edge Solution**: The research demonstrates that high-accuracy taxonomic classification is possible on low-cost, accessible hardware (Raspberry Pi 4) without relying on cloud connectivity. It bridges the gap between high-precision/high-cost lab systems and low-precision/low-cost field systems.
    - *Performance*: The system distinguishes between morphologically similar species (specifically *Aedes aegypti* vs. *Aedes albopictus*) with 98.18% accuracy using a YOLOv8n-cls model.
    - *Optimization*: By leveraging the NCNN framework, the complete pipeline (detection + classification) runs with a latency of roughly 3.3 seconds, which is viable for the discrete nature of mosquito trapping.

2. **Mitigation of Shortcut Learning**: A significant methodological contribution is the identification and resolution of Shortcut Learning (environmental overfitting).
    - *Problem*: Initial models were found to be identifying mosquitoes based on background features rather than morphology.
    - *Solution*: The study developed and validated a rigorous background normalization protocol (using `isnet-general-use` for segmentation) to ensure the model learns genuine morphological features, significantly improving generalization.
3. **Comprehensive Image Dataset**: The work contributes a large-scale, specialized dataset designed for real-world operational conditions.
    - *Scale and Diversity*: The collection contains 85,236 images representing 753 unique mosquito specimens.
    - *Operational Context*: Data was collected in two distinct phases and locations (UnB and Fiocruz Pernambuco) using the specific trap hardware, ensuring the images represent the actual conditions the system will face in the field.
4. **Automated Detection and Classification Pipeline**: The work delivers a fully validated computer vision pipeline ready for deployment:
    - *Object Detection*: A YOLOv8n model achieved a Mean Average Precision (mAP@0.5) of 99.44%, ensuring reliable capture of mosquitoes within the trap.
    - *End-to-End Reliability*: The integrated system achieved an overall accuracy of 92.1% when running directly on the embedded device, proving its potential for scalable, decentralized public health monitoring.

5.4 LIMITATIONS

Despite the successful outcomes, this work encountered several limitations that should be acknowledged:

1. **Shortcut Learning (Environmental Overfitting)**: This was the most critical challenge identified. Initial models achieved artificially high accuracy (99.95%) by learning background features (such as specific trap wall textures) rather than the mosquito's morphology.
    - *Impact*: This rendered the initial models invalid for real-world use as they failed to generalize (accuracy dropped to 89.74% when tested on neutral backgrounds).

- *Mitigation*: It required the development of a complex background normalization strategy, involving algorithmic background removal and manual dataset curation, to force the model to focus on the insect.

2. **Computational Bottleneck of Background Normalization**: While the background normalization strategy solved the shortcut learning problem, it introduced a significant performance bottleneck on the embedded hardware.

    - *Limitation*: The high-quality segmentation model (`isnet-general-use`) used for training was too computationally expensive for the Raspberry Pi.
    - *Trade-off*: The system had to switch to a lighter, quantized model (*u2netp*) for runtime. Even with this lighter model, background removal consumes 86% of the total processing time (approximately 2.8 seconds per image), limiting the system's throughput.

3. **Hardware Constraints vs. Model Complexity**: Deploying modern Deep Learning models on a Raspberry Pi 4 required significant optimization effort.

    - *Limitation*: Standard PyTorch models were too slow for practical use (e.g., detection took nearly 0.8 seconds).
    - *Mitigation*: The entire pipeline had to be migrated to the NCNN framework (optimized for ARM CPUs) with quantization. While successful, this added engineering complexity to achieve the final 3.3-second latency.

4. **Morphological Similarity Between Species**: Distinguishing between *Aedes aegypti* and *Aedes albopictus* presented a persistent challenge due to their biological similarity.

    - *Impact*: The majority of classification errors (73%) occurred between these two species. While the final model achieved high accuracy, this inter-class confusion remains the hardest biological hurdle compared to distinguishing the morphologically distinct *Culex*.

5. **Latency for Real-Time Video**: The final system achieves a throughput of approximately 0.3 FPS (one image every 3.3 seconds).

    - *Limitation*: This latency prevents the system from being used for real-time video stream analysis. It is restricted to processing discrete capture events (images taken when a sensor triggers), which is sufficient for a trap but limits other potential applications.

5.5 FUTURE WORK

To advance the development of this technology towards a mass-deployable product and expand its scientific scope, the following future works are suggested:

- **Sex Classification**: Incorporate the classification of mosquito sex (male vs. female). This feature is crucial for sterile insect technique (SIT) programs and for risk assessment, as only female mosquitoes transmit arboviruses.
- **Expansion of Species Scope**: Extend the dataset and model capabilities to include other epidemiologically relevant vectors, such as Anopheles (malaria vector) and other local *Aedes* species, increasing the system's versatility for different geographic regions.
- **Field Validation with Wild Specimens**: Conduct long-term pilot studies with smart traps deployed in real-world urban environments. Testing with wild mosquitoes is essential to validate the model's robustness against natural morphological variations (e.g., size, wing wear) and uncontrolled lighting conditions.
- **Optimization of Background Normalization**: The current background removal step represents the main computational bottleneck. Future research should investigate:
  - **Non-AI Techniques**: Exploring traditional computer vision methods (e.g., color thresholding, edge detection, or static background subtraction) that are computationally inexpensive compared to neural networks.
  - **Model Optimization**: Applying pruning, quantization, or knowledge distillation specifically to the segmentation models to reduce latency without compromising the preservation of fine details (legs and antennae).
  - **Specialized Training**: Training a lightweight segmentation model (e.g., U-Net MobileNet) specifically on mosquito images, rather than using general-purpose salient object detection models, to achieve a better trade-off between speed and quality.
- **IoT Integration**: Integrate a LoRaWAN or cellular communication module to transmit classification results in real-time to a central dashboard, enabling the creation of dynamic infestation heatmaps.
- **Dataset Expansion**: Incorporate other relevant vectors (such as *Anopheles* or *Haemagogus*) and non-vector insects to further improve the model's robustness and practical utility.

REFERENCES

References

[1] World Health Organization. (2024) Disease outbreak news: Dengue — global situation. Atualizado com dados até 30 de abril de 2024, publicado em 30 de maio de 2024. [Online]. Available: https://www.who.int/emergencies/disease-outbreak-news/item/2024-DON518

[2] Ministério da Saúde. (2024) Monitoramento das arboviroses - aedes aegypti. Accessed on June 22, 2025. [Online]. Available: https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/a/aedes-aegypti/monitoramento-das-arboviroses

[3] ——. (2024) Informe semanal nº 05 — coe arboviroses 2024. Accessed on June 22, 2025. [Online]. Available: https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/a/arboviroses/informe-semanal/2024/informe-semanal-no-05-coe

[4] S. Dasari, B. Gopinath, C. J. Gaulke, S. M. Patel, K. K. Merali, A. Sunil Kumar, and S. Acharya, "A handheld tool for the rapid morphological identification of mosquito species (vectorcam) for community-based malaria vector surveillance: Summative usability study," *JMIR Hum Factors*, vol. 11, p. e56605, Aug 2024. [Online]. Available: https://humanfactors.jmir.org/2024/1/e56605

[5] D. Li, S. Hegde, S. A. Kumar, A. Zacharias, P. Mehta, V. Mukthineni, S. Srimath, S. Patel, M. Suin, R. Chellappa, and S. Acharya, "Towards transforming malaria vector surveillance using vectorbrain: a novel convolutional neural network for mosquito species, sex, and abdomen status identifications," *Scientific Reports*, vol. 14, no. 1, p. 23647, Oct 2024.

[6] W.-L. Liu, Y. Wang, Y.-X. Chen, B.-Y. Chen, A. Y.-C. Lin, S.-T. Dai, C.-H. Chen, and L.-D. Liao, "An iot-based smart mosquito trap system embedded with real-time mosquito image processing by neural networks for mosquito surveillance," *Frontiers in Bioengineering and Biotechnology*, vol. Volume 11 - 2023, 2023. [Online]. Available: https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2023.1100968

[7] A. Goodwin, S. Padmanabhan, S. Hira, and et al., "Mosquito species identification using convolutional neural networks with a multitiered ensemble model for novel species detection," *Scientific Reports*, vol. 11, p. 13656, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-92891-9

[8] ArboControl Project – University of Brasília. (2024) Arbocontrol: Integrated platform for vector surveillance and research. Accessed on June 22, 2025. [Online]. Available: https://arbocontrol.unb.br/

[9] Fiocruz Pernambuco — Centro de Pesquisas Aggeu Magalhães. (2025) Centro de pesquisas aggeu magalhães. Accessed on November 25, 2025. [Online]. Available: https://www.cpqam.fiocruz.br/

[10] A. P. M. Mundim-Pombo, H. J. C. Carvalho, R. Rodrigues Ribeiro, and et al., "Aedes aegypti: egg morphology and embryonic development," *Parasites & Vectors*, vol. 14, p. 531, 2021. [Online]. Available: https://doi.org/10.1186/s13071-021-05024-6

[11] European Centre for Disease Prevention and Control. (2024) Aedes aegypti – fact-sheet for experts. Accessed on June 22, 2025. [Online]. Available: https://www.ecdc.europa.eu/en/disease-vectors/facts/mosquito-factsheets/aedes-aegypti

[12] J. A. Souza-Neto, J. R. Powell, and M. Bonizzoni, "Aedes aegypti vector competence studies: A review," *Infection, Genetics and Evolution*, vol. 67, pp. 191–209, Jan 2019, epub 2018 Nov 19.

[13] C. Vinauger and K. Chandrasegaran, "Context-specific variation in life history traits and behavior of aedes aegypti mosquitoes," *Frontiers in Insect Science*, vol. Volume 4 - 2024, 2024. [Online]. Available: https://www.frontiersin.org/journals/insect-science/articles/10.3389/finsc.2024.1426715

[14] M. Elgendy, *Deep Learning for Vision Systems*. Manning Publications, 2020. [Online]. Available: https://www.manning.com/books/deep-learning-for-vision-systems

[15] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010. [Online]. Available: https://szeliski.org/Book/

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016. [Online]. Available: https://arxiv.org/pdf/1506.01497

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *SSD: Single Shot MultiBox Detector*. Springer International Publishing, 2016, p. 21–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. [Online]. Available: https://arxiv.org/pdf/1506.02640

[19] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” *arXiv preprint arXiv:1911.09070*, 2020. [Online]. Available: https://arxiv.org/abs/1911.09070

[20] K. Zuiderveld, *Contrast limited adaptive histogram equalization*. USA: Academic Press Professional, Inc., 1994, p. 474–485.

[21] R. Manthiram. (2020) Exploring feature extraction with cnns. Accessed: 2025-07-14. [Online]. Available: https://towardsdatascience.com/exploring-feature-extraction-with-cnns-345125cefc9a/

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[23] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, 1989.

[24] Q. Chen, F. Zhang, Y. Wang *et al.*, “Bearing fault diagnosis based on efficient cross space multiscale cnn transformer parallelism,” *Scientific Reports*, vol. 15, p. 12344, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-95895-x

[25] X. Zhao, L. Wang, Y. Zhang *et al.*, “A review of convolutional neural networks in computer vision,” *Artificial Intelligence Review*, vol. 57, no. 99, 2024, accepted 04 February 2024, Published 23 March 2024. [Online]. Available: https://doi.org/10.1007/s10462-024-10721-6

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

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” 2015. [Online]. Available: https://arxiv.org/abs/1512.00567

[28] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, “Review of deep learning: concepts, cnn architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, no. 1, p. 53, 2021, epub 2021 Mar 31, PMID: 33816053, PMCID: PMC8010506. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8

[29] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” 2018. [Online]. Available: https://arxiv.org/abs/1608.06993

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

[31] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” 2020. [Online]. Available: https://arxiv.org/abs/1905.11946

[32] D. M. W. Powers, “Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation,” *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[33] R. Padilla, S. L. Netto, and E. A. B. da Silva, “A survey on performance metrics for object-detection algorithms,” in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[34] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. S. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, “Shortcut learning in deep neural networks,” *Nature Machine Intelligence*, vol. 2, pp. 665–673, 2020.

[35] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2017.

[36] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, “Deep learning on mobile and embedded devices: State-of-the-art, challenges and future directions,” *ACM Computing Surveys*, vol. 53, 05 2020.

[37] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, “A comprehensive review on applications of raspberry pi,” *Computer Science Review*, vol. 52, p. 100636, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013724000200

[38] X. Y. Leung, R. M. Islam, M. Adhami, D. Ilic, L. McDonald, S. Palawaththa, B. Diug, S. U. Munshi, and M. N. Karim, “A systematic review of dengue outbreak prediction models: Current scenario and future directions,” *PLoS Neglected Tropical Diseases*, vol. 17, no. 2, p. e0010631, 2023.

[39] M. S. Rahman, M. Amrin, and M. A. Bokkor Shiddik, “Dengue early warning system and outbreak prediction tool in bangladesh using interpretable tree-based machine learning model,” *Health Science Reports*, vol. 8, no. 5, p. e70726, 2025.

[40] R. Aleixo, F. Kon, R. Rocha, M. S. Camargo, and R. Y. De Camargo, “Predicting dengue outbreaks with explainable machine learning,” in *2022 22nd IEEE Interna-*

tional Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2022, pp. 940–947.

[41] TechCrunch. (2020) Microsoft launches premonition, its hardware and software platform for detecting biological threats. Accessed on 22 June 2025. [Online]. Available: https://techcrunch.com/2020/09/22/microsoft-launches-premonition-its-hardware-and-software-platform-for-detecting-biological

[42] Microsoft News. (2020) Building a better mosquito trap: How a microsoft research project could help track zika's spread. Accessed on 22 June 2025. [Online]. Available: https://news.microsoft.com/features/building-a-better-mosquito-trap-how-a-microsoft-research-project-could-help-track-zikas-sp

[43] D. Oliveira and S. Mafra, "Implementation of an intelligent trap for effective monitoring and control of the aedes aegypti mosquito," *Sensors*, vol. 24, no. 21, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/21/6932

[44] D. Ward and B. Martin, "Trialling a convolution neural network for the identification of braconidae in new zealand," *Journal of Hymenoptera Research*, vol. 95, pp. 95–101, 2023. [Online]. Available: https://doi.org/10.3897/jhr.95.95964

[45] K. Bjerge, H. Karstoft, H. M. Mann, and T. T. Høye, "A deep learning pipeline for time-lapse camera monitoring of insects and their floral environments," *Ecological Informatics*, vol. 84, p. 102861, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954124004035

[46] J. Park, D.-I. Kim, B. Choi, W. Kang, and H. Kwon, "Classification and morphological analysis of vector mosquitoes using deep convolutional neural networks," *Scientific Reports*, vol. 10, p. 1012, 01 2020.

[47] D. Motta, A. □. B. Santos, I. Winkler, B. A. S. Machado, D. A. D. I. Pereira, A. M. Cavalcanti, E. O. L. Fonseca, F. Kirchner, and R. Badaró, "Application of convolutional neural networks for classification of adult mosquitoes in the field," *PLOS ONE*, vol. 14, no. 1, pp. 1–18, 01 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0210829

[48] S. Q. Ong, H. A. Ahmad, G. Nair, P. L. Lim, S. Sulaiman, K. F. San, N. Mustapha, W. A. Nazni, M. S. Jeffree, Y. W. Woon, N. A. Mohamed, A. H. Mohamed, and N. Hussin, "Implementation of a deep learning model for automated classification of *aedes aegypti* (linnaeus) and *aedes albopictus* (skuse) in real time," *Scientific Reports*, vol. 11, p. 9908, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-89365-3

[49] T. O. de Araújo, V. L. de Miranda, and R. Gurgel-Gonçalves, “Ai-driven convolutional neural networks for accurate identification of yellow fever vectors,” *Parasites & Vectors*, vol. 17, p. 329, 2024. [Online]. Available: https://doi.org/10.1186/s13071-024-06406-2

[50] A. Salem, T. Theodoridis, and K. Siriwardana, “Efficient convolutional neural networks on raspberry pi: Enhancing performance with pruning and quantization,” 04 2024.

[51] X. Yue, H. Li, M. Shimizu, S. Kawamura, and L. Meng, “Deep learning-based real-time object detection for empty-dish recycling robot,” pp. 2177–2182, 2022.

[52] D. Rossi, G. Borghi, and R. Vezzani, “Takunet: An energy-efficient cnn for real-time inference on embedded uav systems in emergency response scenarios,” pp. 339–348, 02 2025.

[53] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, “Tensorflow lite micro: Embedded machine learning on tinyml systems,” 2021. [Online]. Available: https://arxiv.org/abs/2010.08678

[54] D. L, P. Megharaj, P. P, N. Kiran, A. R. K P, and G. S Nath, “State-of-the-art object detection: An overview of yolo variants and their performance,” 09 2023, pp. 1018–1024.

[55] M. I. González-Pérez, B. Faulhaber, M. Williams *et al.*, “A novel optical sensor system for the automatic classification of mosquitoes by genus and sex with high levels of accuracy,” *Parasites & Vectors*, vol. 15, p. 190, 2022. [Online]. Available: https://doi.org/10.1186/s13071-022-05324-5

[56] Bzigo, “Bzigo – the first device that detects mosquitoes and points at them,” https://bzigo.com/, 2025, accessed: 2025-06-22.

[57] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics yolov8,” 2023, accessed: 2025-11-25. [Online]. Available: https://github.com/ultralytics/ultralytics

[58] A. A. Michelson, *Studies in Optics*. Chicago, IL: University of Chicago Press, 1927.

[59] J. Cohen, “A coefficient of agreement for nominal scales,” *Miscellaneous*, vol. 20, no. 1, pp. 37–46, 1960.

[60] S. Pertuz, D. Puig, and M. A. Garcia, “Analysis of focus measure operators for shape-from-focus,” *Pattern Recognition*, vol. 46, no. 5, pp. 1415–1432, 2013.

[61] E. Krotkov, “Focusing,” *International Journal of Computer Vision*, vol. 1, no. 3, pp. 223–237, 1988.

[62] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[63] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[64] D. Zhang and G. Lu, “Review of shape representation and description techniques,” *Pattern Recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[65] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[66] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[67] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand, “U2-net: Going deeper with nested u-structure for salient object detection,” *Pattern Recognition*, vol. 106, p. 107404, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320320302077

[68] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[69] J. L. Pech Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia, “Diatom autofocusing in brightfield microscopy: A comparative study,” vol. 3, 02 2000, pp. 314–317 vol.3.

idp

Ensino que
te **conecta**