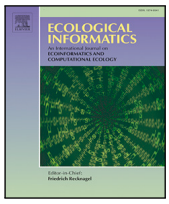
Ecological Informatics 92 (2025) 103437

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf



A comparison of deep neural network compression for citizen-driven tick and mosquito surveillance

Yichao Liu [a,*], Emmanuel Dufourq [b], Peter Fransson [a], Joacim Rocklöv [a,c,d]

[a] Interdisciplinary Center for Scientific Computing, Heidelberg University, 69120, Germany
[b] Department of Computer Science, University of Northern Iowa, 50614, United States of America
[c] Heidelberg Institute of Global Health, Heidelberg University Hospital, 69120, Germany
[d] Department of Epidemiology and Global Health, Umeå University, Sweden

ARTICLE INFO

Dataset link: https://github.com/Christian-lyc/model_compression-citizen-science-.git, https://colab.research.google.com/drive/1Lq1rPGPg3viidtC5pXP0AphMx-RxfTiA?usp=sharing

Keywords:
Deep learning
Pruning
Quantization
Object detection
Tick and mosquito citizen science

ABSTRACT

Citizen science has emerged as an effective approach for infectious disease surveillance. With advancements in machine learning, entomologists can now be relieved from the labor-intensive task of species identification. However, deploying machine learning models on mobile devices presents challenges due to constraints on battery life and memory capacity. In this study, we explore the potential of various model compression techniques for deploying machine learning models on resource-limited devices, enabling low-energy consumption and on-device processing for disease surveillance in remote or low-resource settings. We compared two main-stream model compression techniques, pruning and quantization on various mobile devices. Our findings indicate that quantization methods outperform pruning methods in terms of efficiency. Furthermore, we propose to integrate structured and unstructured pruning to enhance model performance while addressing key constraints of mobile deployment.

1. Introduction

1.1. Background of vectors

Vector-borne diseases, especially those transmitted by ticks and mosquitoes, pose a significant public health concern in the world (Palmer et al., 2017; Boulanger et al., 2019; Pustijanac et al., 2024). Ticks serve as vectors for various bacterial, viral, and parasitic diseases, including endemic Lyme borreliosis and tick-borne encephalitis, as well as the emerging Crimean-Congo hemorrhagic fever virus found within the WHO list of the 10 top-priority infectious diseases (Kuthyar et al., 2021). Similarly, mosquitoes serve as vectors for a variety of pathogens, including malaria parasites and arboviruses such as yellow fever, dengue, and Zika, which collectively impact hundreds of millions of people worldwide and result in over 700,000 deaths annually (Organization, 2024). Although tick- and mosquito-borne pathogens lack the capacity for human-to-human spread, arboviruses have increasingly been observed to affect urban populations and can cause re-occurring outbreaks and severe life-threatening disease. Climate change has, and is expected to further lead to expansion of vector habitats (Rocklöv and Dubrow, 2020). In addition to vector suitability, climate exhibits biological influences on vectors and pathogens through its effects on vector-traits such as longevity, reproduction and vector competence (Rocklöv and Dubrow, 2020).

Climate may additionally affect vector-borne disease risk through changes in reservoir and host abundance and mobility and amplification of pathogens between vectors and hosts (e.g., birds, rodents, livestock, or humans) (Rocklöv and Dubrow, 2020). For example, climate driven changes in habitat and host availability can have cascading effects on tick abundance and pathogen prevalence (Petit et al., 2024). For some range shifts in abundance of vectors, climate change appears to be one of the key drivers; while the impacts are particularly visible at the edges of the altitudinal and latitudinal range of the tick and pathogen in question, where climate is a limiting factor (Gilbert, 2021), impacts are likely much more widespread with climate often increasing the basic reproduction number, R0, of vector-borne diseases (Rocklöv and Dubrow, 2020). For example, *Ixodes ricinus* has displayed a range shift northwards in Europe, its northern limit now being 69° *N* in Arctic Norway, which is a 400-km northward shift since the 1940s (Jore et al., 2011). An increase in tick abundance or distribution does not necessarily lead to higher pathogen prevalence, disease risk, or incidence, as these also depend on the ecology of host–vector–pathogen interaction and human exposure. Human activities that alter land use patterns can create suitable habitats for ticks and their hosts, leading to increased

* Corresponding author.
E-mail address: yichao.liu@iwr.uni-heidelberg.de (Y. Liu).

https://doi.org/10.1016/j.ecoinf.2025.103437
Received 26 March 2025; Received in revised form 16 September 2025; Accepted 19 September 2025
Available online 8 October 2025
1574-9541/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

tick activity and a higher likelihood of human encounters with ticks and tick-borne pathogens (Bouchard et al., 2015; Allan et al., 2003; Gasmi et al., 2022). Similarly, mosquitoes — particularly *Aedes aegypti* — are known to be sensitive to temperature (Rocklöv and Dubrow, 2020) and their eggs require water to develop. Future projections have shown both changes in transmission intensity and geographical range shifts of dengue and malaria (Colón-González et al., 2021). The highly anthropophilic species predominantly feeds on human blood and breeds near human settlements, often in artificial habitats such as discarded tires and covered containers (Ferede et al., 2018). These phenomena highlight the necessity to investigate how climate change and human activities affect tick and mosquito activities, particularly *Culex* mosquitoes, their spatio-temporal distribution, and the incidence of disease among animals and humans, including West Nile Virus (WNV) and other zoonotic diseases (Liu-Helmersson et al., 2019; Ferraguti et al., 2021).

1.2. Related works on citizen science

Citizen science has become a widely used approach for monitoring animal populations and species diversity across various fields, including conservation, vector surveillance (mosquitoes and ticks), and pest control (Pataki et al., 2021; Palmer et al., 2017). Citizen science for vector-borne diseases addresses vector surveillance scalability and additionally early detection of emergence challenges by enabling the public to report photographic vector observations via web- or mobile-based apps (Jongejan et al., 2019; Palmer et al., 2017). Recent studies demonstrate the successful application of tick citizen science in the United States and Canada (Lewis et al., 2018; Chauhan et al., 2020; Nieto et al., 2018), and more recently, in the Netherlands (Jongejan et al., 2019). Similarly, citizen science has been effectively used to monitor mosquito populations. One example is Mosquito Alert (Palmer et al., 2017; Južnič-Zonta et al., 2022), which engages the public in reporting mosquito sightings and integrates data from tools such as ovitraps—devices specifically designed to attract and capture mosquito eggs. This combined approach helps track the distribution of species such as *Aedes aegypti* and *Aedes albopictus*, key vectors for diseases such as dengue, Zika, and chikungunya (Palmer et al., 2017; Južnič-Zonta et al., 2022; Pataki et al., 2021). However, in the case of vector surveillance, data collected through citizen science traditionally requires trained entomologists to manually review submitted images or mailed-in vector samples, which can be time-consuming when tens of thousands of observations are reported (Wiggins and Crowston, 2011). To date, there has been limited research, however, the few existing studies show potential to use deep learning for tick species classification from images.

The rapid advancement of deep learning (LeCun et al., 2015) has led to innovations in classification (He et al., 2016; Szegedy et al., 2015) and object detection (Girshick et al., 2014; Ren et al., 2016), facilitating the development of artificial intelligence (AI) in citizen science projects of ticks and mosquitoes. These innovations significantly reduce the manual classification workload for volunteers—a task traditionally performed only by highly trained specialists. Previously, processing large data volumes was both challenging and time-consuming, but deep learning now enables more efficient and scalable solutions. Additionally, identifying physical samples in laboratories and conducting expert reviews of submitted photographs often caused significant delays, which could be critical for administering timely and appropriate treatment (Justen et al., 2021). Recently, deep learning has been successfully applied to vector surveillance and control within citizen science initiatives (Pataki et al., 2021; Justen et al., 2021; Xu et al., 2021a; Luo et al., 2022; Omodior et al., 2021; Akbarian et al., 2021; Siddiqua et al., 2021; Huang et al., 2018; Mulchandani et al., 2019; Okayasu et al., 2019; Kittichai et al., 2021; Ong et al., 2022; Lee et al., 2023). However, most studies focus solely on model development and comparisons, with models typically occupying around 100 MB of hard drive space, and only one study explicitly considers real-world deployment by hosting the model on a server (Akbarian et al., 2021). While Huang et al. (2018) utilized edge devices, inference was performed on a remote server, making it unsuitable for citizen science applications. Additionally, Mulchandani et al. (2019) employed quantization to compress a sound recognition model, achieving 349 ms inference on an Android device with a Snapdragon 410 system on a chip, but did not address real-world image-based classification challenges. Similarly, Xu et al. (2021a) deployed a lightweight LeNet model (LeCun et al., 2002) on Samsung Galaxy S6 and Huawei P20 Pro devices, demonstrating mobile feasibility, but without addressing inference efficiency or model compression. However, from the studies reviewed, only a few explicitly considered the real-world challenges of citizen science scenarios, where ticks or mosquitoes appear relatively small in the image and are often embedded within a complex background. In practice, the images submitted by citizens pose even greater challenges for recognition, as variations in object size and inconsistent image quality further complicate the classification process (Touvron et al., 2019).

While deep learning models have shown promising results, based on high-quality images, applying them in real-world citizen science projects involving ticks and mosquitoes presents unique challenges. Participants are not professional photographers, nor do they typically use high-resolution cameras, which often results in lower-quality images captured with mobile devices under uncontrolled conditions. The small size of the subjects contributes to the loss of critical visual information, such as the scutum shape and body texture in ticks, or distinctive markings on the thorax and abdomen in mosquitoes. Additionally, these small organisms are often overshadowed by a large, intricate background, which introduces significant noise that can confuse the model. Consequently, the classification model's performance is more likely to degrade under these conditions (Pawlowski et al., 2019).

As an initial step, a promising approach is to train an object detection model to localize the species of interest within an image and then crop those regions to isolate the organism from the background, thereby improving the accuracy of subsequent classification tasks. The hypothesis is that detection and cropping will help reduce background noise, thereby improving classification accuracy. Consequently, the object detection phase is crucial, serving as the foundation for the entire classification process. Given its importance, we have decided to focus on object detection as a preprocessing step prior to classification rather than relying on direct classification. There are two primary approaches to deploying the model: on a remote server, or directly on a mobile device. A server-based deployment allows users to receive real-time predictions of their submitted images; however, this approach may be limited by network bandwidth constraints when uploading images (Han et al., 2015). This issue is particularly critical in citizen science scenarios, where timely information is essential for individuals and public health agencies to receive timely information and provide relevant health advice. Alternatively, deploying the model directly on a mobile device can mitigate network-related delays, particularly in regions with limited or no mobile internet access. This approach may introduce computational limitations, depending on the hardware capabilities of the device. However, object detection models are significantly larger and more complex than the simple architectures used in the citizen science study of Xu et al. (2021a), requiring substantial storage and computational power. Moreover, mobile devices are battery-constrained, and running such large models can lead to excessive energy consumption. These challenges make deploying object detection models on mobile devices difficult, limiting both their accessibility and real-time effectiveness for the general public.

1.3. Model compression methods

Fortunately, advances in model compression have significantly reduced storage while preserving performance (Wu et al., 2016; Zhang

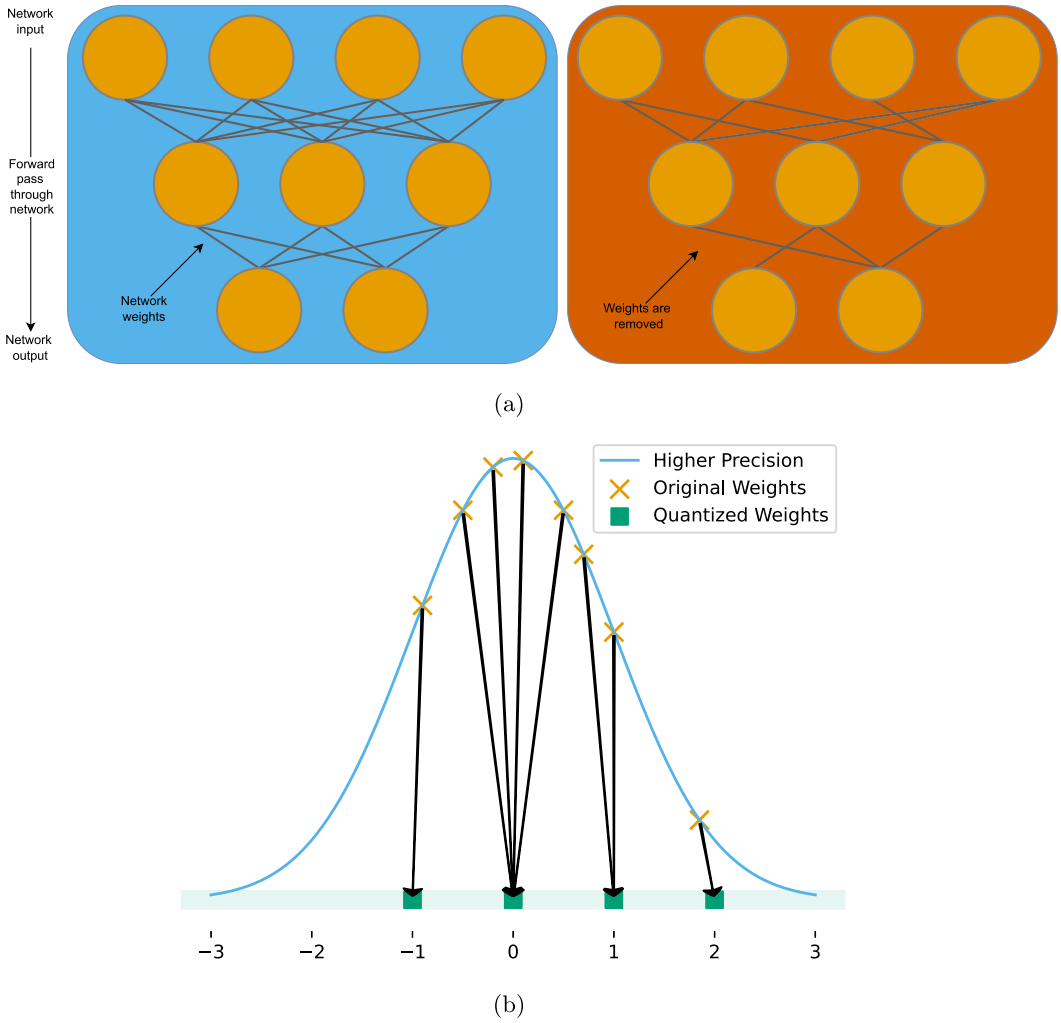


Fig. 1. Diagram of different model compression methods. (a) Left shows an model where each neuron in a layer is connected with every neuron in the layer after it. The right shows the same network after it has been pruned (some weights were removed). (b) Weights of model are quantized from higher precision values (along the curved line) to lower precision ones (one of the four discrete bins). This illustrates the idea of converting from 32 bits to 4 bits.

et al., 2025; Sun et al., 2023; Liu et al., 2017; Crowley et al., 2018; Tai et al., 2015). For example, Sun et al. (2023) demonstrated that compressing the large language model, LLaMA, achieves up to 2 times speedups on compatible hardware. Zhang et al. (2025) introduced Homogeneous-Device Aware Pruning for deployment across homogeneous edge devices, consistently achieving up to 2.86 times speedup at 1.0G Floating Point Operations Per Second (FLOPs) for ResNet50 (He et al., 2016) on homogeneous device clusters. Liu et al. (2017) showed that the popular classification model, VGG (Simonyan and Zisserman, 2014), can be reduced by up to 80% while preserving performance on CIFAR-10 (Krizhevsky et al., 2009), an image classification dataset. These developments make it more feasible to bring artificial intelligence-driven tools to the general public through mobile devices, allowing real-time vector detection and empowering citizen scientists with accessible, high-performance tools to contribute to public health surveillance efforts.

Deep learning models can be optimized for efficiency through various techniques, including pruning (Yang et al., 2017; Yu et al., 2018), quantization (Wu et al., 2016; Mellempudi et al., 2017), and knowledge distillation (Sau and Balasubramanian, 2016; Chen et al., 2017). A diagram of different model compression methods is shown in Fig. 1. Pruning removes redundant parameters to reduce model size and computational complexity. Quantization improves efficiency by lowering the numerical precision of model parameters and computations, converting 32-bit floating-point real values to lower-bit formats such as 8-bit, 4-bit, or even 2-bit. Knowledge distillation transfers knowledge from a larger, high-performing model to a more compact one, preserving accuracy while reducing complexity. In this study, we focus on quantization and pruning, as they are widely used in practice for deploying deep learning models on resource-constrained devices (Kuzmin et al., 2024; Bibi et al., 2024; Geng et al., 2024).

The two most common quantization methods supported by many machine learning frameworks for compressing pre-trained models are dynamic quantization and static quantization, as shown in Fig. 2. Notably, these methods do not require adjusting the pre-trained model's parameters. Dynamic quantization modifies the precision of the network parameters and data during model inference while processing new data. In contrast, static quantization can decrease the precision of the network parameters and does not require any adjustments to the data or network parameters during model inference (Nagel et al., 2021). This is achieved by using a model calibration on a small sample of data.

Pruning can be broadly categorized into structured and unstructured pruning, which has been extensively supported by popular machine learning frameworks. Structured pruning removes entire components of a model, such as neurons, channels, or layers, leading to a smaller and more efficient architecture. It has been extensively studied for its hardware-friendly nature (Cai et al., 2022; van Amersfoort et al., 2020;

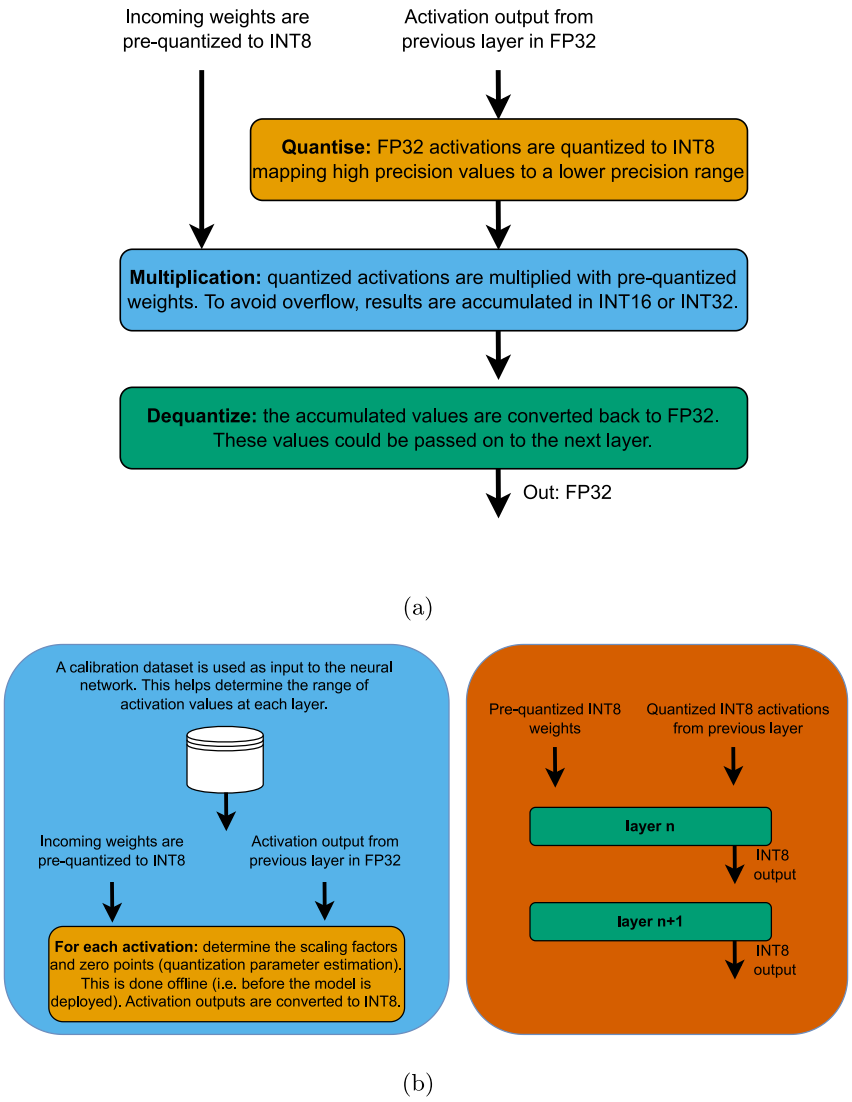


Fig. 2. Diagram of quantization methods. Normally, deep learning models use 32-bit floating-point (FP32) numbers to represent weights and activations (output of the network layers). (a) shows dynamic quantization, where the model is first fully trained in FP32. Quantization is only applied post-training, where the model's weights are pre-quantized from FP32 to 8-bit integers (INT8). These INT8 weights are stored in the model and remain fixed. The input to a layer is encoded as FP32. For each layer, the input to a layer is dynamically quantized to INT8 just before computation. The computation is conducted in INT8 (e.g., matrix multiplication). The result (activation output) is converted back to FP32 (dequantization) before being passed to the next layer. (b) shows static quantization. The training phase remains the same, i.e., a normal FP32 model is trained. A calibration dataset (samples from the dataset) is then used as input to the model. A quantization parameter estimation is performed (b, left) to determine, for each activation, the scaling factors and zero points needed to convert FP32 values into INT8. This is the calibration step. Once the range is estimated, activations can be pre-quantized (b; right). As in dynamic quantization, model weights are converted from FP32 to INT8 before inference. At this stage all matrix multiplications and computations are performed in INT8.

Wang and Fu, 2022; Nonnenmacher et al., 2021). In contrast, unstructured pruning eliminates individual weights with minimal impact on model performance, resulting in a sparse network. While unstructured pruning offers greater flexibility and can achieve higher compression rates, it requires specialized hardware and software optimizations to efficiently handle sparse computations (Cheng et al., 2024).

To the best of our knowledge, this is the first study to explore model compression in the context of image citizen science. Additionally, this work serves as a practical guide for researchers and practitioners on how to implement and evaluate different model compression techniques for mosquito and tick detection. We conduct experiments on both mosquito and tick datasets, assessing the impact of compression on model performance and comparing inference times across various mobile devices. Our contributions are as follows. Firstly, we integrate object detection into citizen science and systematically compare different quantization and pruning techniques on mosquito and tick datasets. We also deploy these compressed YOLOv8 object detection models on various mobile devices to assess the effectiveness of traditional model compression methods in real-world citizen science applications. Secondly, we construct a new tick object detection dataset (3258 images) using automated labeling tools. Finally, we integrate unstructured and structured pruning to mitigate their respective limitations — hardware inefficiency associated with unstructured pruning and performance degradation associated with structured pruning — and propose a hybrid approach that achieves a balance between model compression and accuracy.

2. Materials and methods

2.1. Datasets

In this study, we have used three datasets (Tick Dataset, 2024; Mosquito detection, 2024; Mosquito, 2024), which contain the two

Table 1
Datasets used in this study. The number of examples in each data split is provided, along with the number of classes and dataset source.

Data specification	GBIF tick dataset	GBIF mosquito dataset	Roboflow mosquito dataset
Training set	2282	1956	2283
Validation set	488	418	489
Test set	488	418	316
Classes	7	2	1
Source	GBIF	GBIF	Roboflow

major vectors, ticks and mosquitoes, that can transmit infectious diseases. Roboflow, a web-based platform for managing and preprocessing computer vision datasets, was used to assist in data preparation. We manually labeled and created two GBIF datasets using Roboflow's automatic annotation tool (Dwyer et al., 2024), while the third is an existing mosquito dataset available on Roboflow. All datasets were divided into train, validation, and test sets. Samples images are shown in Fig. 3.

The tick dataset includes 6 genera that commonly appear in European countries, *Carios*, *Dermacentor*, *Haemaphysalis*, *Ixodes*, *Rhipicephalus*, and *Hyalomma*. We collected the data from the GBIF data platform (GBIF, 2020). We filter the data based on the options, which include occurrence data such as images and human observation records. The dataset exhibits a significant genus imbalance, with over 8000 samples of *Ixodes* compared to only 76 samples of *Carios*. Additionally, it contains 4325 samples of *Dermacentor*, 240 samples of *Haemaphysalis*, 267 samples of *Hyalomma*, and 786 samples of *Rhipicephalus*. To address the overrepresentation of *Ixodes*, we downsampled the number of samples from the *Ixodes* class from 8000 to 850 to avoid class imbalance issues. The dataset is split into distinct sets: 2282 images for training, 488 images for validation, and 488 images for testing, with each split maintaining the class distribution.

We used an existing Roboflow mosquito dataset (Mosquito detection, 2024). This dataset contains images of a single species, each with a simple background and minimal or no visual clutter. The dataset was divided into 2283 images for training, 489 validation, and 316 for testing. Additionally, we created a separate mosquito dataset similar to the tick dataset using data from the GBIF platform (Mosquito, 2024). The GBIF mosquito dataset contains two classes, *Aedes aegypti* and *Aedes albopictus*. It was divided into 1956 images for training, 418 images for validation, and 418 images for testing. A summary of the datasets is shown in Table 1.

2.2. Neural network architecture

For our tasks, we applied YOLOv8n (You Only Look Once), a lightweight version of the YOLOv8 neural network (Jocher et al., 2023) that supports both object detection and image classification. This model was selected due to its exceptional speed and accuracy (Varghese and Sambath, 2024), combined with a streamlined design, enhanced adaptability across diverse applications (Jiang and Wu, 2024; Wang et al., 2024), and compatibility with various hardware platforms (Wang et al., 2023a; Yang et al., 2023; Wang et al., 2023b).

The YOLOv8n architecture consists of three main components: a backbone for feature extraction, a neck for feature aggregation, and a head for object detection. These components are built using four core types of submodules, as shown in Fig. 4. The backbone includes convolutional modules, CSP module and SPPF module that extract essential features from the input images. The neck is responsible for aggregating and refining these features across different spatial resolutions, enhancing the model's ability to detect objects at various scales. The head generates the final detection output, including object classes, confidence scores, and bounding box coordinates.

The four types of submodules are embedded in different parts of the network. C2f (Cross-Stage Partial with two fused branches) modules, an improved version of CSP (Cross Stage Partial) blocks, are used in both the backbone and neck to enhance feature reuse and reduce computational cost. SPPF (Spatial Pyramid Pooling—Faster) modules capture multiple scale spatial features by applying multiple pooling operations with different kernel sizes. ConvModule enables the model to learn meaningful patterns while maintaining training stability and non-linearity by applying SiLU activation function and batch normalization.

2.3. Pruning

We applied both unstructured and structured pruning techniques to the YOLOv8n model. The model was initially trained for 100 epochs on the mosquito and tick datasets, using the AdamW optimizer (Loshchilov and Hutter, 2017) with an early stop strategy. Early stopping was applied by monitoring the validation loss, and training was terminated if no improvement was observed over 10 consecutive epochs. The initial learning rate (0.01) and momentum (0.937) were the default settings in the Ultralytics package, based on prior empirical testing and best practices for training YOLO models. Images were preprocessed by resizing them to 640×640 pixels, to match the network's input size requirements. We normalized the pixel values to $[0, 1]$ by dividing the images by 255 to stabilize training. Then fine-tuning was applied to the pruned model.

In unstructured pruning, the individual weights within each layer of a pre-trained model are removed sequentially starting from the first layer, resulting in sparse weights (many of them are zero) without a regular pattern. This irregularity makes it challenging for hardware to optimize efficiently. A common approach is to deploy the model on a device that supports unstructured pruning, such as NVIDIA's Jetson series and certain Edge TPUs. We used PyTorch's L1 norm pruning to identify and zero out the lowest magnitude weights. In our experiments, a hyperparameter specifying the proportion of network channels to prune was set to either 20% or 40% for the entire network. These values were chosen based on preliminary experiments, where they provided a good trade-off between compression and accuracy degradation (Liu et al., 2017). After the pruning step, the pruned model was fine-tuned under the same settings used for pre-training. These settings were likewise applied during fine-tuning for structured pruning. To simplify hardware deployment by providing a standardized model format, we converted the model to the Open Neural Network Exchange (ONNX) format—a library designed to target framework interoperability and hardware optimization accessibility. To assess real-world performance, we integrated the ONNX model into an Android application and uploaded the final application package to AWS Device Farm for testing. We tested on multiple devices to measure RAM usage, inference time, and latency, simulating real-world citizen science scenarios where users require rapid tick and mosquito identification for timely decision-making.

For structured pruning, we constructed a dependency graph using DepGraph (Fang et al., 2023) to identify and remove redundant channels across layers. Channels were pruned based on their global importance ranking, determined by the L1 norm of their average magnitudes. We systematically remove a specified proportion of the least important channels in the network, based on their importance rankings. After pruning, the model was fine-tuned to recover the loss in accuracy from the removal of the redundant weights.

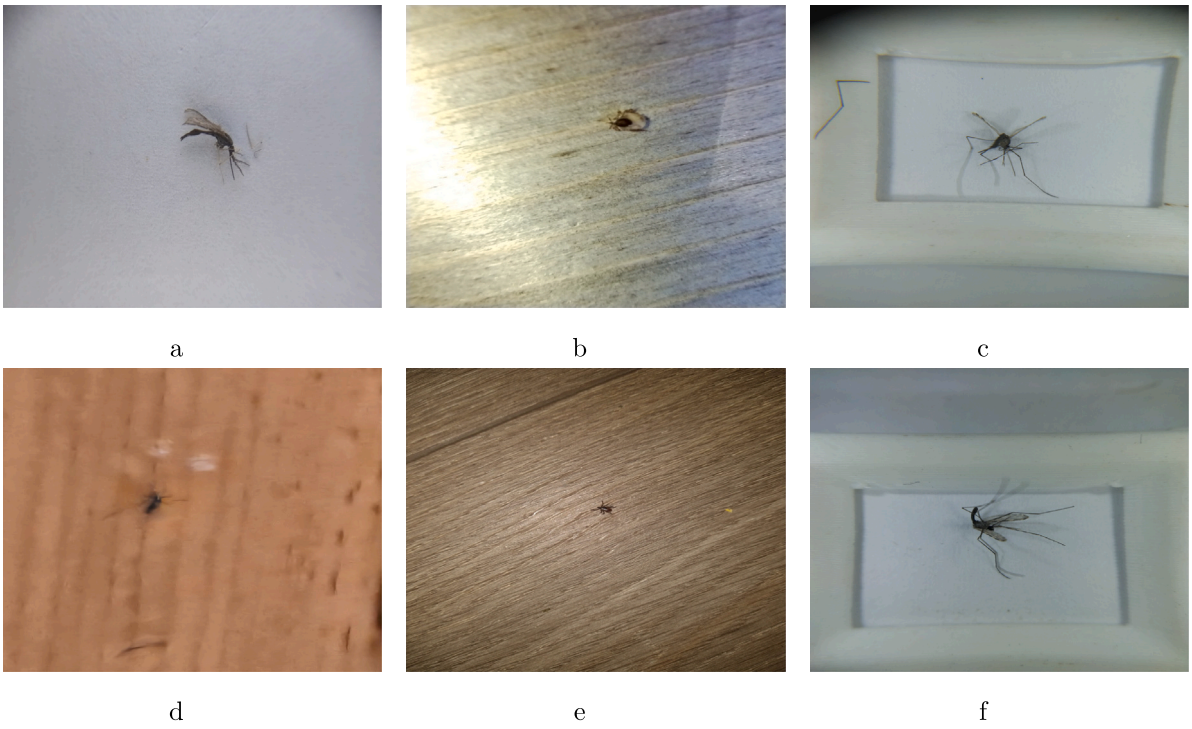


Fig. 3. Examples of ticks and mosquitoes from the different datasets. Images a and d are *Aedes albopictus* and are from the GBIF mosquito dataset, b and e are *Ixodes* and are from the GBIF tick dataset, and c and f are *Aedes albopictus* and are from the Roboflow mosquito dataset.

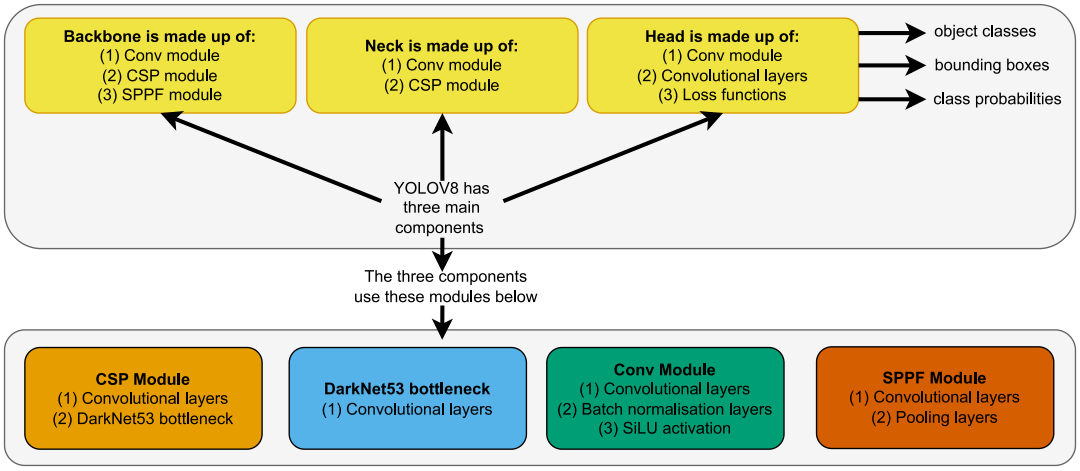


Fig. 4. Simplified YOLOv8 model architecture. There are three main components, each made up from different modules. YOLOv8 produces predictions for the object classes, bounding boxes, and class probabilities.

The L1 norm of the average magnitude of each channel does not sufficiently capture the number of less important weights within a channel, as it does not reflect the distribution of the weights. For example, a channel with high L1 norm does not necessarily mean most of the weights are far from zero. Similarly, a channel with low L1 norm does not mean most of the weights are close to zero. To address this, we combined unstructured and structured pruning, using unstructured pruning as prior information to estimate the count of low-magnitude weights per channel. Our approach assumes that the number of low-magnitude weights within each channel serves as a more accurate criterion for pruning than the average L1 norm.

Specifically, unstructured pruning identifies low-importance weights across the entire network, which are set to zero (algorithm 1, steps 6 to 9). The number of zero weights, in each channel, is determined (algorithm 1, steps 10 to 13), and then the channels are ranked based on these values (algorithm 1, step 14). The ranking is in descending order; the channel with the most number of zeros appear first. A user defined parameter ($p \in [0, 1]$) is then used to determine how many channels are removed (algorithm 1, step 15). If this is set to 20% ($p = 0.2$), this means that the first 20% are removed from the list (algorithm 1, step 16). Finally, the pruned network along with the modified weights are obtained. Our approach is presented in algorithm 1, and visually explained in Supplementary Material A.

2.4. Quantization

We applied both dynamic and static Post-Training Quantization (PTQ) to our pre-trained YOLOv8 model using the ONNX Runtime quantization package, optimizing it for efficient deployment across diverse hardware platforms. PTQ was chosen over Quantization-Aware

Algorithm 1 Dual Pruning for YOLOv8n

Input: Pre-trained YOLOv8n network $F(\mathbf{X};\mathbf{W})$, input data $\mathbf{X}$, pre-trained YOLOv8n weights $\mathbf{W}$, channels $\mathbf{C}$ in F, pruning ratio $p \in [0,1]$
Output: Pruned YOLOv8n network F'
Load pre-trained model F
$N_W \leftarrow$ the number of weights in W
$N_C \leftarrow$ the number of channels in C
Stage 1: Unstructured Pruning (Weight Pruning)
Sort $\{|w_i| \mid w_i \in \mathbf{W}\}$ in ascending order ▷ Compute the absolute value of each weight in F, and sort the values in ascending order
$r_w \leftarrow \text{round}(N_W \times p)$ ▷ Determine the number of weights to prune
Set the first r_w weights (smallest magnitudes) from the sorted list (step 7) to zero
Stage 2: Structured Pruning (Channel Pruning)
for each channel c_i in C **do**
    Count number of zero weights in c_i
end for
Sort the channels in descending order based on steps 11 to 13
$r_c \leftarrow \text{round}(N_C \times p)$ ▷ Determine the number of channels to prune
Set the first r_c channels (least important) from the sorted list in step 14 to zero
Return pruned network F' with updated weights $\mathbf{W}$ that have been pruned to zero

Training due to its lower computational cost and sufficient accuracy retention in our application (Gholami et al., 2022). We used 8-bit quantization, as it balances compression and computational efficiency while maintaining compatibility with most hardware accelerators (Gholami et al., 2022). For all of the tick and mosquito datasets, dynamic quantization was applied to pre-trained models, converting weights and activations to 8-bit precision. In this method, activations are quantized at runtime, dynamically computing quantization parameters based on incoming data. This approach preserves accuracy, but increases inference time due to real-time parameter calculations (Gholami et al., 2022).

Conversely, static quantization first converts model parameters to 8-bit precision and determines fixed global quantization parameters (scale and zero point) by calibrating activations using representative sample data. To calibrate the model, a small range of sample data (validation set) is fed into the model to compute the range and distribution of activation values. Then, scaling factors and zero points are determined to map from floating-point values to integers. The quality of this calibration data is critical, as mismatches with the inference distribution can cause activation clipping or poor dynamic range utilization, leading to accuracy degradation. This reduces the inference cost, since quantization parameters remain constant (Gholami et al., 2022), though it may slightly reduce the accuracy due to potential mismatches between fixed parameters and real-time data distributions (Gholami et al., 2022). For our YOLOv8 application, we primarily used static quantization but left the detection layer unquantized. This decision was based on empirical evaluations showing that quantizing the detection layer significantly degraded the accuracy, likely due to its sensitivity for prediction.

2.5. Mobile device selection

Market share reports indicate that Android devices, particularly those from Samsung and Xiaomi, dominate the smartphone market compared to iOS (Mobile, 2024). Given this, we focused our testing efforts on Android platforms to ensure broader applicability for citizen science participants. Testing focused on measuring inference latency and RAM usage, we selected three Android devices representing a range of mid-tier smartphones and tablets, Samsung Galaxy Tab S7 (Snapdragon 865+, high-end CPU for parallelized tasks), Xiaomi Redmi Note 10 (Snapdragon 678, mid-range CPU for moderate workloads), and a Samsung Galaxy A53 (Exynos 1280, mid-range CPU). The device selection was based on several criteria, including market prevalence, hardware capabilities (e.g., chipset and memory), and affordability. These devices provide a balance between accessibility and computational power, ensuring that our model can run efficiently on commonly used consumer hardware.

3. Results

We evaluated six metrics to compare the different compression methods. Namely, mAP50 (Mean Average Precision at an IoU threshold of 0.50), a widely used metric in object detection (Chen et al., 2022; Wu et al., 2024; Xu et al., 2021b); number of network parameters (millions, M), measuring model size; bit operations (BOPS, in billions, G), which quantifies computational complexity by summing all operations weighted by the number of bits involved, offering a more hardware-aware alternative to FLOPs (Liu et al., 2023; Wang et al., 2018; van Baalen et al., 2022); inference time (milliseconds, ms) indicating real-time performance; RAM usage (megabytes, MB) evaluating memory efficiency; and finally energy consumption (kilowatt hours, kWh), measuring energy efficiency.

We conducted experiments using AWS Device Farm, an Amazon-hosted service that enables automated testing on real smartphones and tablets. Our results show that the model performs significantly better on the Roboflow mosquito dataset compared to both the tick dataset and the GBIF mosquito dataset (Tables 2, 3, 4). This discrepancy is likely due to two factors: first, the Roboflow dataset consists of single-background images, whereas the other datasets include more complex scenes, sometimes even without ticks; second, the tick dataset contains a larger number of classes, which may affect performance since classification loss contributes during training. Regarding model compression, unstructured pruning and quantization do not alter the total parameter count. This is because unstructured pruning only zeroes out low-magnitude weights without removing them, while quantization reduces precision but retains the original weight structure. From Tables 5, 6 and 7, static quantization consistently achieved the lowest inference time and RAM usage on the tablet. However, RAM savings were less pronounced on smartphones, likely due to differences in chipset architectures, memory management strategies, or operating system level optimizations. Our findings reveal that dual pruning (40% SP, 40% UP) achieves the best mAP50 and additionally resulted in the lowest number of parameters, on average 1.38 and 1.24 M (Tables 3 and 4). Notably, dual pruning (40% SP + 40% UP) achieved optimal mAP50 (97.50) while also demonstrating lower energy consumption compared to even quantization. If we instead consider operation times, quantization achieves the lowest BOPS/G, offering the fastest inference (328 BOPS/G compared to much higher values obtained by the other methods, Tables 2 and 3). From the hardware deployment perspective, performance varies across devices: high-end processors (Samsung galaxy tab s7) generally achieve shorter inference times but tend to consume more RAM, whereas mid-range processors (Xiaomi Redmi Note 10 and Samsung Galaxy A53) offer lower memory usage at the cost of slower inference. Static quantization consistently provided the shortest inference times and lower RAM usage, particularly on the Samsung Galaxy Tab S7. These results highlight the importance of selecting a compression method based on specific deployment needs. For scenarios prioritizing model accuracy, dual pruning with dynamic quantization is the best choice. When memory-constrained environments are crucial, structured pruning yields the most compact models and is also ideal for maximizing energy savings. We determined the minimum CPU RAM required to execute the different models. Our findings revealed that at least 300 MB of CPU RAM was needed to process two images using the original model, and that 200 MB of CPU RAM was needed for SP (40%) and DP (40% SP and 40% UP). Finally, if reducing computational cost and inference time is the priority, quantization proves to be the most effective approach.

Table 2
Model compression comparison on the GBIF tick dataset. 40% SP corresponds to structured pruning for 40% of the entire network channels, and 20% UP corresponds to unstructured pruning for 20% of the entire network channels, S- and D-Q correspond to static and dynamic quantization respectively, and DP corresponds to dual pruning.

Method	mAP50	Parameters/M	BOPS/G	Energy consumption/kwh
Original model	77.15	3.00	4298	0.24×10^{-1}
UP (20% prune)	76.30	3.00	4298	0.23×10^{-1}
UP (40% prune)	74.90	3.00	4298	0.23×10^{-1}
SP (20% prune)	72.80	2.12	3274	0.98×10^{-2}
SP (40% prune)	73.30	1.39	2260	0.92×10^{-2}
DQ	77.80	3.00	325.60	0.12×10^{-1}
SQ	74.20	3.00	325.60	0.12×10^{-1}
DP (20% SP, 40% UP)	**79.00**	2.12	3738	0.21×10^{-1}
DP (20% SP, 20% UP)	76.90	2.12	3738	0.22×10^{-1}
DP (40% SP, 40% UP)	78.00	**1.23**	3031	0.21×10^{-1}

Table 3
Model compression comparison on the Roboflow mosquito dataset. 40% SP corresponds to structured pruning for 40% of the entire network channels, and 20% UP corresponds to unstructured pruning for 20% of the entire network channels, S- and D-Q correspond to static and dynamic quantization respectively, and DP corresponds to dual pruning.

Method	mAP50	Parameters/M	BOPS/G	Energy consumption/kwh
Original model	97.50	3.00	4330	0.85×10^{-2}
UP (20% prune)	95.60	3.00	4330	0.72×10^{-2}
UP (40% prune)	96.20	3.00	4330	0.74×10^{-2}
SP (20% prune)	96.80	2.10	3274	0.64×10^{-2}
SP (40% prune)	95.30	**1.37**	2313	0.59×10^{-2}
DQ	97.20	3.00	328	0.76×10^{-2}
SQ	96.80	3.00	328	0.81×10^{-2}
DP (20% SP, 40% UP)	97.45	2.13	3918	0.67×10^{-2}
DP (20% SP, 20% UP)	96.80	2.12	3918	0.67×10^{-2}
DP (40% SP, 40% UP)	**97.50**	1.38	3548	0.63×10^{-2}

Table 4
Model compression comparison on the GBIF mosquito dataset. 40% SP corresponds to structured pruning for 40% of the entire network channels, and 20% UP corresponds to unstructured pruning for 20% of the entire network channels, S- and D-Q correspond to static and dynamic quantization respectively, and DP corresponds to dual pruning.

Method	mAP50	Parameters/M	BOPS/G	Energy consumption/kwh
Original model	83.67	3.00	4330	0.92×10^{-2}
UP (20% prune)	85.65	3.00	4330	0.96×10^{-2}
UP (40% prune)	85.59	3.00	4330	0.99×10^{-2}
SP (20% prune)	86.86	2.04	3337	0.85×10^{-2}
SP (40% prune)	85.79	1.35	2270	0.80×10^{-2}
DQ	84.27	3.00	328	0.65×10^{-2}
SQ	79.19	3.00	328	0.61×10^{-2}
DP (20% SP, 40% UP)	85.46	2.00	3696	0.86×10^{-2}
DP (20% SP, 20% UP)	84.84	2.00	3664	0.83×10^{-2}
DP (40% SP, 40% UP)	**86.95**	**1.24**	3052	0.70×10^{-2}

Table 5
Evaluation of different devices for model compression methods on a random image of the GBIF tick dataset. 40% SP corresponds to structured pruning for 40% of the entire network channels, and 20% UP corresponds to unstructured pruning for 20% of the entire network channels, S- and D-Q correspond to static and dynamic quantization respectively, and DP corresponds to dual pruning.

Method	Samsung galaxy tab *s7*		Xiaomi Redmi Note 10		Samsung Galaxy A53	
	Time/ms	RAM/MB	Time/ms	RAM/MB	Time/ms	RAM/MB
Original model	148	214	259	118	277	112
UP (20% prune)	133	193	246	118	283	112
UP (40% prune)	139	190	237	120	328	113
SP (20% prune)	106	152	214	110	221	100
SP (40% prune)	82	167	152	109	178	101
DQ	104	150	193	111	197	97
SQ	56	141	94	110	109	106
DP (20% SP, 40% UP)	120	171	225	114	256	102
DP (20% SP, 20% UP)	132	182	225	104	276	104
DP (40% SP, 40% UP)	109	168	188	108	223	101

4. Discussion

On our datasets, quantization is generally the most effective choice in most scenarios in terms of inference time and operation numbers. Specifically, our findings indicate that static quantization is suitable for single-background object detection, offering the shortest inference time with minimal accuracy loss. For example, on the Roboflow mosquito dataset, static quantization reduced inference time by 57% on the Samsung Tab and 61% on the Xiaomi Note 10, with only a 0.7% drop in mAP50. In contrast, dynamic quantization preserved mAP50

Table 6
Evaluation of different devices for model compression methods on a random image of the Roboflow mosquito dataset. 40% SP corresponds to structured pruning for 40% of the entire network channels, and 20% UP corresponds to unstructured pruning for 20% of the entire network channels, S- and D-Q correspond to static and dynamic quantization respectively, and DP corresponds to dual pruning.

Method	Samsung galaxy tab s7		Xiaomi Redmi Note 10		Samsung Galaxy A53	
	Time/ms	RAM/MB	Time/ms	RAM/MB	Time/ms	RAM/MB
Original model	129	206	240	115	264	111
UP (20% prune)	163	209	236	116	248	107
UP (40% prune)	134	205	239	122	284	109
SP (20% prune)	109	167	200	106	233	107
SP (40% prune)	97	146	153	102	158	105
DQ	118	155	175	97	237	106
SQ	55	150	92	109	112	107
DP (20% SP, 40% UP)	118	188	229	108	275	103
DP (20% SP, 20% UP)	147	184	229	109	279	100
DP (40% SP, 40% UP)	117	164	210	111	248	103

Table 7
Evaluation of different devices for model compression methods on a random image of the GBIF mosquito dataset. 40% SP corresponds to structured pruning for 40% of the entire network channels, and 20% UP corresponds to unstructured pruning for 20% of the entire network channels, S- and D-Q correspond to static and dynamic quantization respectively, and DP corresponds to dual pruning.

Method	Samsung galaxy tab s7		Xiaomi Redmi Note 10		Samsung Galaxy A53	
	Time/ms	RAM/MB	Time/ms	RAM/MB	Time/ms	RAM/MB
Original model	126	184	289	146	274	104
UP (20% prune)	140	211	293	152	288	111
UP (40% prune)	151	194	257	153	293	111
SP (20% prune)	117	184	244	131	237	103
SP (40% prune)	82	167	185	117	192	91
DQ	105	170	245	116	224	93
SQ	50	164	108	122	130	106
DP (20% SP, 40% UP)	142	180	230	139	262	111
DP (20% SP, 20% UP)	133	181	258	135	253	98
DP (40% SP, 40% UP)	114	153	203	124	206	96

but only reduced inference time by 27%. However, the reduction in energy consumption was less pronounced, likely due to the simplicity of the dataset. Nevertheless, for complex object detection tasks, dynamic quantization maintains comparable or superior performance to the original model while achieving reasonable inference time and RAM usage. Additionally, quantization significantly reduces model size, making it well-suited for mobile deployment. For instance, on the GBIF tick dataset, dynamic quantization reduced energy consumption by 50% and inference time by 29.7% on the Samsung Tab, all while maintaining similar detection performance. Quantization aware training methods that rely on additional training can be costly, the practical advantages of PTQ make it a more efficient and accessible solution for real-world deployment.

While unstructured pruning is theoretically appealing, it remains impractical for mobile devices due to limited support for sparse computations. Structured pruning, however, effectively reduces inference time and improves energy efficiency, making it beneficial for battery-constrained environments. For example, in both the tick dataset and Roboflow mosquito dataset, 40% structured pruning reduced energy consumption by approximately 22%–25% compared to dynamic quantization. However, on the tick dataset, while structured pruning achieved faster inference, it came at a cost—a 4% drop in mAP50 compared to dynamic quantization. This performance decline may stem from the removal of entire channels that contain important features, which structured pruning is prone to. We investigated the combination of structured and unstructured pruning and found that it reduced inference times while enhancing overall performance. This accurately identifies which channels contain more unimportant weights. However, in most cases, dual pruning led to longer inference times when compared to structured pruning alone, suggesting that pruned network architecture influences inference efficiency. This may be due to too many channels pruned in certain layers relative to others, leading to imbalanced computational loads across layers. Despite this, dual pruning remains valuable in applications where maintaining accuracy is a priority and the simple task scenarios where energy consumption is important. An interesting finding is that energy consumption is correlated with task complexity. For instance, in the most complex scenario — the tick dataset — structured pruning and quantization reduced energy consumption by 50% compared to the original model. However, this reduction was less significant for the simpler mosquito dataset, highlighting the influence of task complexity on compression benefits.

Across the evaluated datasets, quantization consistently outperforms pruning, aligning with the findings of Kuzmin et al. (2024), although their study did not examine hardware-specific comparisons. To the best of our knowledge, model compression has only been applied once in the current citizen science literature, since most of citizen science models are deployed on servers (Mulchandani et al., 2019). Our study provides comprehensive guidance for citizen science researchers on deploying models on mobile devices, with particular emphasis on applications in regions lacking systematic surveillance infrastructure, such as Low- and Middle-Income Countries, which often face additional computational challenges. The results demonstrate that the minimum inference time and energy consumption are reduced to half that of the original model, suggesting a reduction in inference-related battery consumption. Based on our experiments, we recommend integrating quantization and pruning to maximize model size reduction, particularly for the deployment of small and medium-sized YOLOv8 models on mobile devices. Combining dual pruning with quantization has the potential to enhance quantization performance while mitigating the inference-time drawbacks of dual pruning, especially in complex object detection tasks. However, the highly imbalanced nature of citizen science data, especially regarding rare species, poses challenges to using classification models following object detection.

Future work will extend in two complementary directions. First, from a methodological perspective, we will explore applying model compression to sequential object detection and classification models for citizen science, such as YOLO with ResNet (He et al., 2016) or MobileNet (Howard et al., 2017). Additionally, mixed-precision quantization offers a promising direction, as it combines high-bit and low-bit precision to enhance traditional quantization performance (Peng et al., 2023; Sun et al., 2022). Exploring adaptive quantization strategies (Gholami et al., 2022) may further optimize the trade-off between accuracy, latency, and energy efficiency in citizen science applications. Second, from a deployment perspective, while this paper primarily provides a theoretical framework and empirical results, future work will focus on developing and evaluating a user-friendly front-end interface. Such an interface will enable non-technical users to easily upload or capture images, receive model predictions in plain language, and provide simple feedback to improve system usability in practice.

In various ecological fields, model compression could be very useful. For example, low-powered edge devices are frequently used in wildlife monitoring to better understand migration and climate change, especially in the forest without reliable connectivity to remote servers. Additionally, in some citizen science projects, users might prefer to maintain privacy and would prefer to execute a model locally on a mobile device. A compressed model can help ecologists run predictions on field-deployed laptops, or other small devices especially in remote areas, instead of cloud computation. Our work takes an initial step toward integrating model compression into ecological studies, enabling more efficient and accessible AI applications. We envisage that future advancements in this area will further enhance real-time conservation efforts, empower citizen scientists, and support large-scale ecological monitoring.

CRediT authorship contribution statement

Yichao Liu: Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Emmanuel Dufourq:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization. **Peter Fransson:** Writing – review & editing, Supervision. **Joacim Rocklöv:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Given Emmanuel Dufourq's role as Associate Editor, he had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

YL and JR is supported by the Alexander von Humboldt Stiftnung. YL was supported by the BWS plus project "heiAIMS" which is carried out by the Baden-Württemberg Stiftung as part of the Baden-Württemberg-STIPENDIUM programme for university cooperations and the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences (HGS MathComp). The statements made and views expressed are solely the responsibility of the authors. We acknowledge the suggestions from Prof. Holger Fröning from the Institute of Computer Engineering (ZITI), Heidelberg University

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ecoinf.2025.103437.

Data availability

All code for training and testing the neural networks is available at https://github.com/Christian-lyc/model_compression-citizen-science-.git. An example code for dual pruning in algorithm 1 is available at https://colab.research.google.com/drive/1Lq1rPGPg3viidtC5pXP0AphMx-RxfTiA?usp=sharing.

References

Akbarian, S., Nelder, M.P., Russell, C.B., Cawston, T., Moreno, L., Patel, S.N., Allen, V.G., Dolatabadi, E., 2021. A computer vision approach to identifying ticks related to Lyme disease. IEEE J. Transl. Eng. Heal. Med. 10, 1–8.

Allan, B.F., Keesing, F., Ostfeld, R.S., 2003. Effect of forest fragmentation on Lyme disease risk. Conserv. Biol. 17 (1), 267–272.

Bibi, U., Mazhar, M., Sabir, D., Butt, M.F.U., Hassan, A., Ghazanfar, M.A., Khan, A.A., Abdul, W., 2024. Advances in pruning and quantization for natural language processing. IEEE Access.

Bouchard, C., Leonard, E., Koffi, J.K., Pelcat, Y., Peregrine, A., Chilton, N., Rochon, K., Lysyk, T., Lindsay, L.R., Ogden, N.H., 2015. The increasing risk of Lyme disease in Canada. Can. Vet. J. 56 (7), 693.

Boulanger, N., Boyer, P., Talagrand-Reboul, E., Hansmann, Y., 2019. Ticks and tick-borne diseases. Med. Mal. Infect. 49 (2), 87–97.

Cai, Y., Hua, W., Chen, H., Suh, G.E., De Sa, C., Zhang, Z., 2022. Structured pruning is all you need for pruning CNNs at initialization. arXiv preprint arXiv:2203.02549.

Chauhan, G., McClure, J., Hekman, J., Marsh, P.W., Bailey, J.A., Daniels, R.F., Genereux, D.P., Karlsson, E.K., 2020. Combining citizen science and genomics to investigate tick, pathogen, and commensal microbiome at single-tick resolution. Front. Genet. 10, 1322.

Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M., 2017. Learning efficient object detection models with knowledge distillation. Adv. Neural Inf. Process. Syst. 30.

Chen, Z., Duan, Y., Wang, W., He, J., Lu, T., Dai, J., Qiao, Y., 2022. Vision transformer adapter for dense predictions. arXiv preprint arXiv:2205.08534.

Cheng, H., Zhang, M., Shi, J.Q., 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Trans. Pattern Anal. Mach. Intell..

Colón-González, F.J., Sewe, M.O., Tompkins, A.M., Sjödin, H., Casallas, A., Rocklöv, J., Caminade, C., Lowe, R., 2021. Projecting the risk of mosquito-borne diseases in a warmer and more populated world: a multi-model, multi-scenario intercomparison modelling study. Lancet Planet. Heal. 5 (7), e404–e414.

Crowley, E.J., Gray, G., Storkey, A.J., 2018. Moonshine: Distilling with cheap convolutions. Adv. Neural Inf. Process. Syst. 31.

Dwyer, B., Nelson, J., Hansen, T., et al., 2024. Roboflow. Retrieved from https://roboflow.com. (Computer vision software).

Fang, G., Ma, X., Song, M., Mi, M.B., Wang, X., 2023. Depgraph: Towards any structural pruning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16091–16101.

Ferede, G., Tiruneh, M., Abate, E., Kassa, W.J., Wondimeneh, Y., Damtie, D., Tessema, B., 2018. Distribution and larval breeding habitats of Aedes mosquito species in residential areas of northwest Ethiopia. Epidemiol. Heal. 40.

Ferraguti, M., Heesterbeek, H., Martínez-de la Puente, J., Jiménez-Clavero, M.Á., Vázquez, A., Ruiz, S., Llorente, F., Roiz, D., Vernooij, H., Soriguer, R., et al., 2021. The role of different Culex mosquito species in the transmission of West Nile virus and avian malaria parasites in Mediterranean areas. Transbound. Emerg. Dis. 68 (2), 920–930.

Gasmi, S., Koffi, J.K., Nelder, M.P., Russell, C., Graham-Derham, S., Lachance, L., Adhikari, B., Badcock, J., Baidoobonso, S., Billard, B.A., et al., 2022. Vector-borne infections-part 1: Ticks & mosquitoes: Surveillance for Lyme disease in Canada, 2009–2019. Can. Commun. Dis. Rep. 48 (5), 219.

GBIF, G., 2020. The global biodiversity information facility (year) what is GBIF. Available from [13 January 2020].

Geng, X., Wang, Z., Chen, C., Xu, Q., Xu, K., Jin, C., Gupta, M., Yang, X., Chen, Z., Aly, M.M.S., et al., 2024. From algorithm to hardware: A survey on efficient and safe deployment of deep neural networks. IEEE Trans. Neural Netw. Learn. Syst..

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K., 2022. A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision. Chapman and Hall/CRC, pp. 291–326.

Gilbert, L., 2021. The impacts of climate change on ticks and tick-borne disease risk. Annu. Rev. Entomol. 66 (1), 373–388.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 580–587.

Han, S., Mao, H., Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Huang, L.-P., Hong, M.-H., Luo, C.-H., Mahajan, S., Chen, L.-J., 2018. A vector mosquitoes classification system based on edge computing and deep learning. In: 2018 Conference on Technologies and Applications of Artificial Intelligence. TAAI, IEEE, pp. 24–27.

Jiang, L., Wu, L., 2024. Enhanced Yolov8 network with Extended Kalman Filter for wildlife detection and tracking in complex environments. Ecol. Inform. 84, 102856.

Jocher, G., Chaurasia, A., Qiu, J., 2023. Ultralytics YOLOv8. Retrieved from https://github.com/ultralytics/ultralytics.

Jongejan, F., de Jong, S., Voskuilen, T., van den Heuvel, L., Bouman, R., Heesen, H., Ijzermans, C., Berger, L., 2019. "Tekenscanner": a novel smartphone application for companion animal owners and veterinarians to engage in tick and tick-borne pathogen surveillance in the Netherlands. Parasites Vectors 12, 1–9.

Jore, S., Viljugrein, H., Hofshagen, M., Brun-Hansen, H., Kristoffersen, A.B., Nygård, K., Brun, E., Ottesen, P., Sævik, B.K., Ytrehus, B., 2011. Multi-source analysis reveals latitudinal and altitudinal shifts in range of Ixodes ricinus at its northern distribution limit. Parasites Vectors 4, 1–11.

Justen, L., Carlsmith, D., Paskewitz, S.M., Bartholomay, L.C., Bron, G.M., 2021. Identification of public submitted tick images: A neural network approach. Plos One 16 (12), e0260622.

Južnič-Zonta, Ž., Sanpera-Calbet, I., Eritja, R., Palmer, J.R., Escobar, A., Garriga, J., Oltra, A., Richter-Boix, A., Schaffner, F., Della Torre, A., et al., 2022. Mosquito alert: leveraging citizen science to create a GBIF mosquito occurrence dataset. Gigabyte 2022.

Kittichai, V., Pengsakul, T., Chumchuen, K., Samung, Y., Sriwichai, P., Phatthamolrat, N., et al., 2021. Deep learning approaches for challenging species and gender identification of mosquito vectors. Sci. Rep. 11, 4838.

Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images.

Kuthyar, S., Anthony, C.L., Fashina, T., Yeh, S., Shantha, J.G., 2021. World health organization high priority pathogens: Ophthalmic disease findings and vision health perspectives. Pathogens 10 (4), 442.

Kuzmin, A., Nagel, M., Van Baalen, M., Behboodi, A., Blankevoort, T., 2024. Pruning vs quantization: which is better? Adv. Neural Inf. Process. Syst. 36.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 2002. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.

Lee, S., Kim, H., Cho, B., 2023. Deep learning-based image classification for major mosquito species inhabiting Korea. Insects 14 (6), 526.

Lewis, J., Boudreau, C., Patterson, J., Bradet-Legris, J., Lloyd, V., 2018. Citizen science and community engagement in tick surveillance—A Canadian case study. Healthcare 6 (1).

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C., 2017. Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2736–2744.

Liu, J., Zhuang, B., Chen, P., Shen, C., Cai, J., Tan, M., 2023. Single-path bit sharing for automatic loss-aware model compression. IEEE Trans. Pattern Anal. Mach. Intell. 45 (10), 12459–12473.

Liu-Helmersson, J., Brännström, Å., Sewe, M.O., Semenza, J.C., Rocklöv, J., 2019. Estimating past, present, and future trends in the global distribution and abundance of the arbovirus vector Aedes aegypti under climate change scenarios. Front. Public Heal. 7, 148.

Loshchilov, I., Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Luo, C.-Y., Pearson, P., Xu, G., Rich, S.M., 2022. A computer vision-based approach for tick identification using deep learning models. Insects 13 (2), 116.

Mellempudi, N., Kundu, A., Mudigere, D., Das, D., Kaul, B., Dubey, P., 2017. Ternary neural networks with fine-grained quantization. 2017. arXiv preprint arxiv:1705.01462.

2024. Mobile Operating System Market Share Worldwide. StatCounter Global Stats, Retrieved from https://gs.statcounter.com/os-market-share/mobile/worldwide/2024.

2024. Mosquito dataset. Retrieved from https://universe.roboflow.com/tickcitizenscience/mosquito-qi01f. (Computer vision dataset on Roboflow Universe).

2024. Mosquito detection dataset. Retrieved from https://universe.roboflow.com/venkat-mukthineni/mosquito-detection-fqch7. (Computer vision dataset on Roboflow Universe).

Mulchandani, P., Siddiqui, M.U., Kanani, P., et al., 2019. Real-time mosquito species identification using deep learning techniques. Int. J. Eng. Adv. Technol. 9 (2), 2249–8958.

Nagel, M., Fournarakis, M., Amjad, R.A., Bondarenko, Y., Van Baalen, M., Blankevoort, T., 2021. A white paper on neural network quantization. arXiv preprint arXiv:2106.08295.

Nieto, N.C., Porter, W.T., Wachara, J.C., Lowrey, T.J., Martin, L., Motyka, P.J., Salkeld, D.J., 2018. Using citizen science to describe the prevalence and distribution of tick bite and exposure to tick-borne diseases in the United States. PLoS One 13 (7), e0199644.

Nonnenmacher, M., Pfeil, T., Steinwart, I., Reeb, D., 2021. SOSP: Efficiently capturing global correlations by second-order structured pruning. arXiv preprint arXiv:2110.11395.

Okayasu, K., Yoshida, K., Fuchida, M., Nakamura, A., 2019. Vision-based classification of mosquito species: Comparison of conventional and deep learning methods. Appl. Sci. 9 (18), 3935.

Omodior, O., Saeedpour-Parizi, M.R., Rahman, M.K., Azad, A., Clay, K., 2021. Using convolutional neural networks for tick image recognition–a preliminary exploration. Exp. Appl. Acarol. 84, 607–622.

Ong, S.-Q., Nair, G., Yusof, U.K., Ahmad, H., 2022. Community-based mosquito surveillance: an automatic mosquito-on-human-skin recognition system with a deep learning algorithm. Pest. Manag. Sci. 78 (10), 4092–4104.

Organization, W.H., 2024. Vector-Borne Diseases. Tech. Rep., World Health Organization, Retrieved from https://www.who.int/news-room/fact-sheets/detail/vector-borne-diseases.

Palmer, J.R., Oltra, A., Collantes, F., Delgado, J.A., Lucientes, J., Delacour, S., Bengoa, M., Eritja, R., Bartumeus, F., 2017. Citizen science provides a reliable and scalable tool to track disease-carrying mosquitoes. Nat. Commun. 8 (1), 1–13.

Pataki, B.A., Garriga, J., Eritja, R., Palmer, J.R., Bartumeus, F., Csabai, I., 2021. Deep learning identification for citizen science surveillance of tiger mosquitoes. Sci. Rep. 11 (1), 4718.

Pawlowski, N., Bhooshan, S., Ballas, N., Ciompi, F., Glocker, B., Drozdzal, M., 2019. Needles in haystacks: On classifying tiny objects in large images. arXiv preprint arXiv:1908.06037.

Peng, P., You, M., Jiang, K., Lian, Y., Xu, W., 2023. MBFQuant: A multiplier-bitwidth-fixed, mixed-precision quantization method for mobile CNN-based applications. IEEE Trans. Image Process. 32, 2438–2453.

Petit, M.J., Johnson, N., Mansfield, K.L., 2024. Vectorial dynamics underpinning current and future tick-borne virus emergence in Europe. J. Gen. Virol. 105 (11), 002041.

Pustijanac, E., Buršić, M., Millotti, G., Paliaga, P., Iveša, N., Cvek, M., 2024. Tick-borne bacterial diseases in Europe: Threats to public health. Eur. J. Clin. Microbiol. Infect. Dis. 1–35.

Ren, S., He, K., Girshick, R., Sun, J., 2016. Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. 39 (6), 1137–1149.

Rocklöv, J., Dubrow, R., 2020. Climate change: an enduring challenge for vector-borne disease prevention and control. Nature Immunol. 21 (5), 479–483.

Sau, B.B., Balasubramanian, V.N., 2016. Deep model compression: Distilling knowledge from noisy teachers. arXiv preprint arXiv:1610.09650.

Siddiqua, R., Rahman, S., Uddin, J., 2021. A deep learning-based dengue mosquito detection method using faster R-CNN and image processing techniques. Ann. Emerg. Technol. Comput. (AETiC) 5 (3), 11–23.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sun, Z., Ge, C., Wang, J., Lin, M., Chen, H., Li, H., Sun, X., 2022. Entropy-driven mixed-precision quantization for deep network design. Adv. Neural Inf. Process. Syst. 35, 21508–21520.

Sun, M., Liu, Z., Bair, A., Kolter, J.Z., 2023. A simple and effective pruning approach for large language models. arXiv preprint arXiv:2306.11695.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9.

Tai, C., Xiao, T., Zhang, Y., Wang, X., et al., 2015. Convolutional neural networks with low-rank regularization. arXiv preprint arXiv:1511.06067.

2024. Tick dataset. Retrieved from https://universe.roboflow.com/tickcitizenscience/tick-qk4y9. (Computer vision dataset on Roboflow Universe).

Touvron, H., Vedaldi, A., Douze, M., Jégou, H., 2019. Fixing the train-test resolution discrepancy. Adv. Neural Inf. Process. Syst. 32.

van Amersfoort, J., Alizadeh, M., Farquhar, S., Lane, N., Gal, Y., 2020. Single shot structured pruning before training. arXiv preprint arXiv:2007.00389.

van Baalen, M., Kahne, B., Mahurin, E., Kuzmin, A., Skliar, A., Nagel, M., Blankevoort, T., 2022. Simulated quantization, real power savings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2757–2761.

Varghese, R., Sambath, M., 2024. Yolov8: A novel object detection algorithm with enhanced performance and robustness. In: 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems. ADICS, IEEE, pp. 1–6.

Wang, G., Chen, Y., An, P., Hong, H., Hu, J., Huang, T., 2023a. UAV-YOLOv8: A small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios. Sensors 23 (16), 7190.

Wang, H., Fu, Y., 2022. Trainability preserving neural pruning. arXiv preprint arXiv:2207.12534.

Wang, X., Gao, H., Jia, Z., Li, Z., 2023b. BL-YOLOv8: An improved road defect detection model based on YOLOv8. Sensors 23 (20), 8361.

Wang, T., Ren, S., Zhang, H., 2024. Nighttime wildlife object detection based on YOLOv8-night. Electron. Lett. 60 (15), e13305.

Wang, L., Zhan, J., Gao, W., Yang, K., Jiang, Z., Ren, R., He, X., Luo, C., 2018. BOPS, not FLOPS! A new metric and roofline performance model for datacenter computing. arXiv preprint arXiv:1801.09212.

Wiggins, A., Crowston, K., 2011. From conservation to crowdsourcing: A typology of citizen science. In: 2011 44th Hawaii International Conference on System Sciences. IEEE, pp. 1–10.

Wu, J., Jiang, Y., Liu, Q., Yuan, Z., Bai, X., Bai, S., 2024. General object foundation model for images and videos at scale. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3783–3795.

Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J., 2016. Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4820–4828.

Xu, Z., Ding, X., Yin, K., Li, Z., Smyth, J.A., Sims, M.B., McGinnis, H.A., Liu, C., 2021a. Tickphone App: A smartphone application for rapid tick identification using deep learning. Appl. Sci. 11 (16), 7355.

Xu, M., Zhang, Z., Hu, H., Wang, J., Wang, L., Wei, F., Bai, X., Liu, Z., 2021b. End-to-end semi-supervised object detection with soft teacher. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3060–3069.

Yang, T.-J., Chen, Y.-H., Sze, V., 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5687–5695.

Yang, G., Wang, J., Nie, Z., Yang, H., Yu, S., 2023. A lightweight YOLOv8 tomato detection algorithm combining feature enhancement and attention. Agronomy 13 (7), 1824.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V.I., Han, X., Gao, M., Lin, C.-Y., Davis, L.S., 2018. Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9194–9203.

Zhang, K., Li, G., Lu, N., Yang, P., Tang, K., 2025. Hardware-aware DNN compression for homogeneous edge devices. arXiv preprint arXiv:2501.15240.