information

MDPI

Article

Edge-Optimized Deep Learning Architectures for Classification of Agricultural Insects with Mobile Deployment

Muhammad Hannan Akhtar, Ibrahim Eksheir and Tamer Shanableh *

Department of Computer Science and Engineering, American University of Sharjah, Sharjah P.O. Box 26666, United Arab Emirates; b00101092@aus.edu (M.H.A.); b00102157@aus.edu (I.E.)
* Correspondence: tshanableh@aus.edu

Abstract: The deployment of machine learning models on mobile platforms has ushered in a new era of innovation across diverse sectors, including agriculture, where such applications hold immense promise for empowering farmers with cutting-edge technologies. In this context, the threat posed by insects to crop yields during harvest has escalated, fueled by factors such as evolution and climate change-induced shifts in insect behavior. To address this challenge, smart insect monitoring systems and detection models have emerged as crucial tools for farmers and IoT-based systems, enabling interventions to safeguard crops. The primary contribution of this study lies in its systematic investigation of model optimization techniques for edge deployment, including Post-Training Quantization, Quantization-Aware Training, and Data Representative Quantization. As such, we address the crucial need for efficient, on-site pest detection tools in agricultural settings. We provide a detailed analysis of the trade-offs between model size, inference speed, and accuracy across different optimization approaches, ensuring practical applicability in resource-constrained farming environments. Our study explores various methodologies for model development, including the utilization of Mobile-ViT and EfficientNet architectures, coupled with transfer learning and fine-tuning techniques. Using the Dangerous Farm Insects Dataset, we achieve an accuracy of 82.6% and 77.8% on validation and test datasets, respectively, showcasing the efficacy of our approach. Furthermore, we investigate quantization techniques to optimize model performance for on-device inference, ensuring seamless deployment on mobile devices and other edge devices without compromising accuracy. The best quantized model, produced through Post-Training Quantization, was able to maintain a classification accuracy of 77.8% while significantly reducing the model size from 33 MB to 9.6 MB. To validate the generalizability of our solution, we extended our experiments to the larger IP102 dataset. The quantized model produced using Post-Training Quantization was able to maintain a classification accuracy of 59.6% while also reducing the model size from 33 MB to 9.6 MB, thus demonstrating that our solution maintains a competitive performance across a broader range of insect classes.

Keywords: on-edge classification; model quantization; TensorFlow Lite; insect classification; EfficientNet; deep learning

Academic Editor: Heming Jia

Received: 3 March 2025
Revised: 6 April 2025
Accepted: 22 April 2025
Published: 25 April 2025

Citation: Akhtar, M.H.; Eksheir, I.; Shanableh, T. Edge-Optimized Deep Learning Architectures for Classification of Agricultural Insects with Mobile Deployment. *Information* **2025**, *16*, 348. https://doi.org/10.3390/info16050348

1. Introduction

The deployment of machine learning models on mobile platforms has opened a multitude of opportunities across various sectors, including economics, environment, healthcare, education, and, notably, agriculture. In the realm of agriculture, these applications hold particular significance, offering transformative benefits to farmers, one of the backbones

of our society. By equipping farmers with state-of-the-art technologies, these applications provide them with unprecedented access to timely and actionable insights.

It has been proven over the years that insects pose the greatest dangers to crop yields during harvest, as seen by the rapid rise in the use of agricultural pesticides. However, due to evolution and climate change-related factors, a lot of dangerous insects become resistant and pose a greater threat [1]. Therefore, throughout the years, smart insect monitoring systems [2,3] and insect detection and classification models [4–6] have been used to tackle this problem by helping farmers and smart IoT-based systems identify and target these insects to protect crops.

Traditional methods of insect detection and pest management rely heavily on manual observation or cloud-based classification systems, which often introduce delays, inaccuracies, and privacy concerns. These methods are also impractical in remote or rural areas with limited or unreliable network connectivity, thus making rapid pest identification and timely intervention difficult.

Further worsening the problem, climate change and accelerated insect evolution have led to the emergence of resistant insect populations, complicating traditional pesticide-based management strategies. Existing digital insect monitoring systems often leverage heavy computational models which, while accurate, are unsuitable for direct deployment on resource-constrained mobile and edge devices commonly used by farmers. Such models either require continuous internet connectivity or are computationally prohibitive for real-time on-device analysis, thus failing to deliver the speed, efficiency, and convenience urgently needed in practical agricultural scenarios.

Therefore, there is a clear and pressing need for optimized, edge-deployable machine learning models that are capable of accurate and real-time insect classification directly on mobile devices. The existing literature largely focuses on cloud-based or resource-intensive deep learning architectures, neglecting the challenges associated with the lightweight deployment and quantization strategies that are critical for mobile scenarios. Moreover, datasets utilized by many existing studies often either lack specificity relevant to local pest populations or are excessively generalized, resulting in reduced model accuracy and applicability in localized agricultural contexts.

Now, there are many different insect image datasets with labels and annotations. Some popular ones include IP102 [7], which contains over 75,000 images with 102 classes of insects, an ultra-specific insect dataset [8], and BIOSCAN-1M [9], which contains over 1 million images. However, these large datasets may not always represent the species or environmental conditions relevant to a particular agricultural context. Consequently, more targeted datasets have begun to emerge—such as the Dangerous Farm Insects Dataset—focusing specifically on pests that pose significant risks to crop yields. These smaller, highly specific datasets present new opportunities and challenges for model development, especially when balanced against the computational constraints and on-device resource limitations inherent to mobile and edge platforms.

To date, relatively few studies have leveraged the Dangerous Farm Insects Dataset. One notable example is [10], where researchers employed transfer learning approaches using pre-trained deep learning architectures (e.g., ResNet, MobileNet, VGG) to identify the most effective model for classifying dangerous farm insects. Despite the dataset's limited size, ref. [10] demonstrated that models like Xception could achieve a validation accuracy of about 77.7%. However, the test performance remained around 70%, suggesting that there is room for methodological improvements and a need for techniques that better generalize from limited training samples while also maintaining efficient inference.

In this context, the present study aims to push state-of-the-art on-device insect classification by focusing on advanced classification models that are inherently well-suited

for mobile deployment. Specifically, we investigate architectures such as Mobile-ViT and EfficientNet, known for their balance of model accuracy and computational efficiency on constrained hardware platforms. While prior works like [10] provide a valuable benchmark for model performance on the Dangerous Farm Insects Dataset, our approach strives to go further by optimizing both accuracy and efficiency through a series of quantization strategies. These include Post-Training Quantization, Quantization-Aware Training, and representative dataset selection, all of which are evaluated for their impact on model performance and resource utilization. Beyond evaluating these techniques on our primary dataset, we also validate the approach on a larger dataset (IP102), demonstrating that our method is indeed generalizable, and it scales effectively.

To illustrate the practical implications of these advancements, we also develop a proof-of-concept mobile application using TensorFlow Lite. This application performs real-time insect classification directly on the device, enabling farmers and field workers to rapidly identify pests without relying on cloud-based services or continuous network connectivity. By improving both the effectiveness and the deployment feasibility of insect classification tools, this work contributes a generalizable methodology for building high-performance, resource-efficient ML models suited to the agricultural domain and beyond.

2. Literature Review

The ubiquity of mobile devices has motivated the search for rapidly prototyped and deployed machine learning models adept in running within the environments of these platforms. TensorFlow Lite (TFLite), a new trademark of Google's TensorFlow framework which is lighter and thus can be integrated into mobile applications easily, plays a role as the bridge connecting the development of machine learning models and mobile applications [11]. Developed for mobile and embedded platforms such as smartphones, tablets, and many portable devices, TensorFlow Lite enables developers to implement intelligent capabilities on their mobile applications.

TensorFlow Lite deals with the challenges in mobile environments, such as limited computing resources [12], power limitations, and the requirement for quicker real-time processing [13]. Through methods like model quantization [14], optimization [15], and hardware acceleration, TensorFlow Lite allows complex machine learning algorithms and neural networks to run on devices efficiently. This ensures the response time protects user privacy through computations performed locally on the device itself while delivering user experiences. Whether it is image recognition or natural language processing to gesture recognition or predictive analytics, TensorFlow Lite equips developers with tools to create applications that leverage machine learning capabilities at the device level.

There are some interesting applications of TFLite in different domains where the models have been used for a variety of tasks. For instance, ref. [16] introduces an IoT-based solution for real-time flash flood detection and alerting using TensorFlow Lite. It addresses the limitations of traditional flood warning methods by emphasizing the need for timely alerts to facilitate an effective public response. The proposed solution employs computer vision techniques with video cameras to monitor water levels. Implemented on low-powered Raspberry Pi devices, the system offers scalability and adaptability for deployment in flood-prone regions. Performance evaluation identifies TensorFlow Lite with an SSD-MobileNet-v2-Quantized model as the optimal configuration for achieving high detection accuracy and efficiency in IoT environments.

Some interesting applications have also been seen in image classification. The work in [17] introduces "AgroAId", a mobile app system utilizing deep learning and TensorFlow Lite for the visual classification of plant species and diseases, achieving 99% accuracy and providing spatiotemporal analytics on regional and seasonal disease trends. The work

in [18] presents a mobile application for on-edge medical diagnosis of lung diseases using TensorFlow Lite. The study experimented with a total of 18 models, which were created using various quantization techniques, including post-classification quantization, integer quantization, and Quantization-Aware Training. Quantization-Aware Training resulted in a 75.59% reduction in model size, with MobileNetV2 offering the best performance-to-size ratio, showing only a 4.1% accuracy loss.

Additionally, ref. [19] explores the use of vision transformers (ViTs) for on-edge medical diagnostics using the Kvasir-Capsule image classification dataset of gastrointestinal diseases. The study applied TensorFlow Lite quantization techniques, such as post-training float-16 (F16) quantization and Quantization-Aware Training (QAT), to reduce model sizes while maintaining performance. The study concluded that MobileViT_V2_175, with its F16 quantization and 27.47 MB size, offered the best balance between performance and efficiency.

Moreover, ref. [20] proposed "Leboh", an Android mobile application utilizing TFLite's EfficientNet-Lite model for waste classification, achieving an accuracy of 95.39% during model evaluation and 82.5% during user testing. Likewise, ref. [21] had an interesting idea and dataset, as they introduced a mobile application for oil palm fresh fruit ripeness classification, achieving a high test accuracy of 89.3%, with a 96 ms inference time per image using EfficientNetB0 optimized through transfer learning, 9-angle crop data augmentation, and float16 quantization.

Furthermore, with the advancements in the fields of computer vision, object detection applications have made their way into the TinyML domain, and have been implemented using TFLite and MobileNetv2. For example, ref. [22] presents an object detection and classification system for urban actors, using TFLite with a re-trained Single Shot Detector (SSD) model. Another interesting use case was [23], which introduces a mobile-based application for traffic signs recognition, leveraging TFLite and transfer learning techniques to train a Single Shot MultiBox Detector (SSD) MobileNet V2 model, with the quantized model demonstrating four times faster detection compared to the original float model.

There are lots of other interesting applications in video classification [24], audio classification [25,26], and natural language processing that this paper does not delve into but are also worth looking into to gain a broader perspective on the potential and ability of TFLite in adding to the TinyML space.

In recent years, the integration of modern machine learning techniques has greatly advanced insect image classification and detection across a variety of contexts, from agricultural fields to forest ecosystems. Building on the versatility and real-time capabilities of TFLite demonstrated in previous applications, this literature review now shifts focus to insect classification. By synthesizing key findings from studies in this domain, we explore the datasets, algorithms, and methodologies employed, emphasizing the potential for mobile-based, real-time insect detection solutions to address existing challenges and enhance performance in the field.

Many of the existing workers performed experiments on their personal developed datasets. Ref. [27] developed a dataset of 225 images, representing nine common orders and sub-orders of insect species, with 25 specimen images in each. They utilized artificial neural networks (ANNs) and support vector machine (SVM) algorithms, reaching an accuracy of 93% with the SVM model.

Similarly, ref. [28] developed their own dataset, consisting of 60 samples of 24 common pest species found in-field, resulting in a 1440 image dataset. To enhance the classification accuracy in field crop insects, they developed recognition systems utilizing techniques such as multiple-task sparse representation and multiple-kernel learning (MKL). By lever-

aging the unique features of insect images, these techniques significantly improved the recognition performance, contributing to a classification result of 90.4%.

The work in [29] uses both of the datasets developed in [27,28], consisting of nine and twenty-four classes, respectively. Their methodology involved employing various machine learning techniques, such as artificial neural networks (ANN), support vector machine (SVM), k-nearest neighbors (KNN), naive Bayes (NB), and convolutional neural network (CNN) models. The study also proposed an insect pest detection algorithm involving foreground extraction and contour identification, contributing to the classification of insects across complex backgrounds. The evaluation of classification models was enhanced through nine-fold cross-validation, resulting in the highest classification rates of 91.5% and 90% for the nine and the twenty-four class insects, respectively, achieved using the CNN model.

Additionally, ref. [5] uses the dataset of 24 classes developed by [28], along with additional images sourced from the internet and incorporated into it for enhancements in generalization. The study implemented an improved network architecture based on VGG19, known for its progressive learning of image features. The model was compared to existing methods such as SSD and Fast R-CNN. Notably, the proposed methodology achieved a mean Average Precision (mAP) of 0.8922, surpassing both SSD and Fast R-CNN in performance.

Other studies that opted to develop their own dataset include [30], where a dataset of 29 k images was developed, covering 30 insect species. ResNet transfer learning techniques are applied to the dataset to classify the forest insects. The system achieved an average insect classification accuracy of 94%. To facilitate the classification process, the researchers developed an application that allows users to capture, edit, and transfer insect images.

Likewise, ref. [31] also utilized their own dataset, which comprised images of twenty classes of paddy field insect pests sourced from Google Images and photographs taken by the Faculty of Agriculture, University of Jaffna, Sri Lanka. A framework was developed to classify the images of paddy field insect pests using gradient-based features through the bag-of-words approach. The classification process involved several steps, including the identification of regions of interest and their representation as scale-invariant feature transform (SIFT) or speeded-up robust features (SURF) descriptors. Subsequently, codebooks were constructed to map these descriptors into fixed-length vectors in the histogram space. The feature histograms were then subjected to multi-class classification using support vector machines (SVMs). Notably, the combination of the Histogram of Oriented Gradients (HOG) descriptors with SURF features yielded approximately 90% accuracy in classification.

The work in [32] carries out a cascade architecture to identify Lepidoptera species from their images, combining deep convolutional neural networks (DCNNs) with Supported Vector Machines (SVMs). The dataset utilized in this research consisted of 1301 Lepidoptera images from 22 species. The architecture used part of the DCNN as a feature extractor, followed by SVMs serving as the insect classifiers. The proposed cascade architecture achieved a reported accuracy of 100% on the testing dataset.

Recent developments in automated insect identification further emphasize the feasibility and effectiveness of deploying machine learning-based methods for practical agricultural monitoring. The work in [33] demonstrated the use of convolutional neural networks (CNNs) to classify economically important insect species, such as the Mediterranean fruit fly (*Ceratitis capitata*) and the olive fruit fly (*Bactrocera oleae*), in real-time, even when insects are freely moving and changing postures. Their study highlighted the significant improvement in accuracy (93%) that is achievable using methods that go beyond traditional static-image-based approaches, emphasizing the potential for real-time, automated moni-

toring and precise pest management interventions. Similarly, the work in [34] leveraged optical sensors combined with machine learning to accurately identify flying insects in agricultural fields, achieving over 80% accuracy in the classification. This approach facilitated the timely and spatially optimized application of insecticides, significantly reducing unnecessary pesticide usage and supporting environmentally sustainable pest management. Collectively, these advancements illustrate the growing potential and practical utility of integrated sensor and machine learning technologies in precision agriculture. Lastly, the work in [35] proposed a workflow for holistic insect monitoring. The approach involves the use of large-scale DNA barcoding (megabarcoding) to classify species, validate them with morphology, and use specimen images to train AI for identification and trait analysis.

In summary, TFLite's adaptability for mobile and on-device machine learning has been demonstrated across various fields, and this flexibility is especially promising for insect classification. While previous research has utilized machine learning models for insect detection, reliance on cloud-based inference systems often introduces latency and privacy concerns. By applying TFLite for real-time, on-device insect classification, our study bridges this gap, offering a mobile-based solution that leverages the benefits of low-latency inference, efficient power usage, and privacy preservation. This approach not only enhances performance in the field, but also makes insect detection more accessible and scalable for practical applications such as wildlife monitoring and agricultural pest control.

In recent years, smart trap technology has emerged as a complementary approach to traditional deep learning-based insect classification. Ref. [36] developed an autonomous smart trap for the precision monitoring of hematophagous flies on cattle that integrates high-resolution imaging, environmental sensing, and on-device convolutional neural network processing. Their system achieved an overall classification accuracy of 96%, demonstrating a robust performance under field conditions, with low power consumption and real-time data transmission capabilities. This work not only validates the practical utility of edge-deployed AI in pest monitoring, but it also serves as a valuable benchmark for integrating sensor data with deep learning methodologies. By comparing our quantization-driven optimization approach to such real-world applications, we further underscore the relevance of our methods for achieving efficient and accurate insect detection in resource-constrained agricultural environments.

Furthermore, ref. [37] provide a comprehensive review of artificial intelligence applications in integrated pest management. They systematically examine how AI methodologies, when integrated with IoT-based systems, can enhance decision-making in pest control. This work, along with [38], highlights the potential for developing intelligent, real-time pest management systems that reduce the reliance on chemical treatments while promoting sustainable agriculture. By contextualizing our research within these broader AI-driven strategies, our approach to quantization-driven optimization for edge deployment is further validated as a critical step toward efficient, real-world pest monitoring.

3. Dataset

The first dataset used in this study is called the Dangerous Farm Insects Dataset. The dataset focuses on dangerous farm insects, consisting of approximately 1500 images across 15 distinct classes, with around 100 images per class. This localized dataset represents a more realistic approach compared to the larger, more generalized insect datasets, as it reflects the specific insect populations farmers encounter in certain regions. The classes include various pests such as locusts, aphids, and beetles, which pose significant threats to crops. By working with a smaller, focused dataset, we aim to demonstrate that targeted, region-specific data can be highly effective for building models that cater to local agricul-

tural needs, simulating real-world scenarios where the population control of specific insects is critical.

Table 1 provides a breakdown of the 15 insect classes, along with the number of images per class, offering insights into the dataset's structure. Additionally, Figure 1 showcases examples of dataset images, highlighting the visual diversity and complexity of the insects. This dataset encourages personalized data collection, empowering farmers and researchers to use tailored models for insect classification and control, ultimately promoting more efficient, localized pest management solutions.

Table 1. Classes and Image Distribution of Dangerous Farm Insects Dataset.

Class Name	Count
Africanized Honeybees (Killer Bees)	97
Aphids	88
Armyworms	96
Brown Marmorated Stink Bugs	114
Cabbage Loopers	104
Citrus Canker	104
Colorado Potato Beetle	112
Corn Borers	115
Corn Earworms	110
Fall Armyworms	113
Fruit Flies	101
Spider Mites	119
Thrips	109
Tomato Hornworms	109
Western Corn Rootworms	100
Total	1591

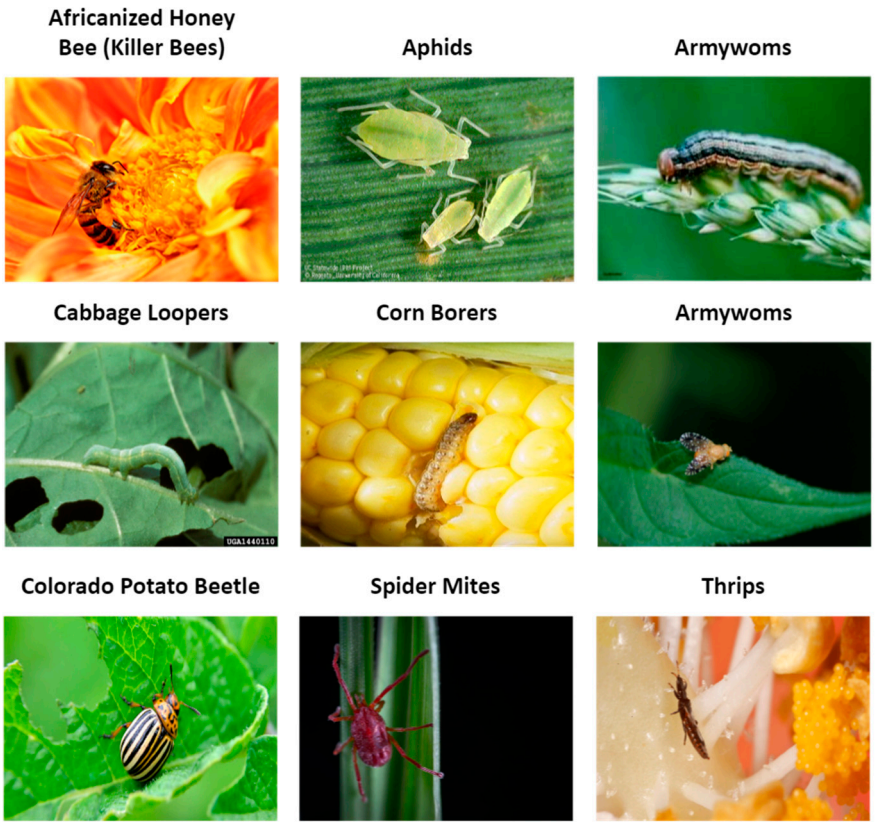

Figure 1. Example Images from the Dangerous Farm Insects Dataset.

In this work, we also use a second, larger, and more diverse dataset to validate our solution's generalizability. For this purpose, we utilized the IP102 dataset [7], which consists of over 75,000 images, spanning 102 distinct classes of insects. This comprehensive dataset is one of the most widely recognized benchmarks in the field of insect pest classification, representing a diverse array of species and scenarios. Unlike the localized Dangerous Farm Insects Dataset, which focuses on specific pests in regional agricultural contexts, IP102 encompasses a broader spectrum of insect species, making it more reflective of global insect populations. This diversity presents a unique challenge, as inter-class similarities and greater variability in the images demand highly robust models capable of distinguishing fine-grained details across a wide range of classes. Figure 2 showcases examples of the dataset images.

Figure 2. Example Images from the IP102 Dataset.

The IP102 dataset exhibits a pronounced, long-tailed distribution across its 102 insect pest categories. Our analysis reveals a significant class imbalance that mirrors real-world ecological conditions, where certain pest species are far more prevalent than others. Figure 3 displays the number of images per class. Key statistics from our analysis indicate an average of 737 images per class, with a minimum of 71 images and a maximum of 5740 images per class, and a standard deviation of 971. Approximately 25% of the classes have fewer than 263 samples, while the top 10% of classes account for nearly 30% of the dataset, highlighting a strong imbalance. For instance, class 101 alone contains 5740 samples, whereas class 72 comprises only 71 samples. This disparity poses a challenge for deep learning models, which can become biased toward overrepresented classes. To mitigate these effects, we employed stratified sampling during training and evaluation to maintain proportional representation, applied class-weighted loss functions to penalize misclassification of minority classes, and used data augmentation techniques for underrepresented classes to effectively increase their sample size. This distributional skewness underscores the importance of developing robust models that generalize well under low-data regimes, especially for rare but agriculturally significant pests.

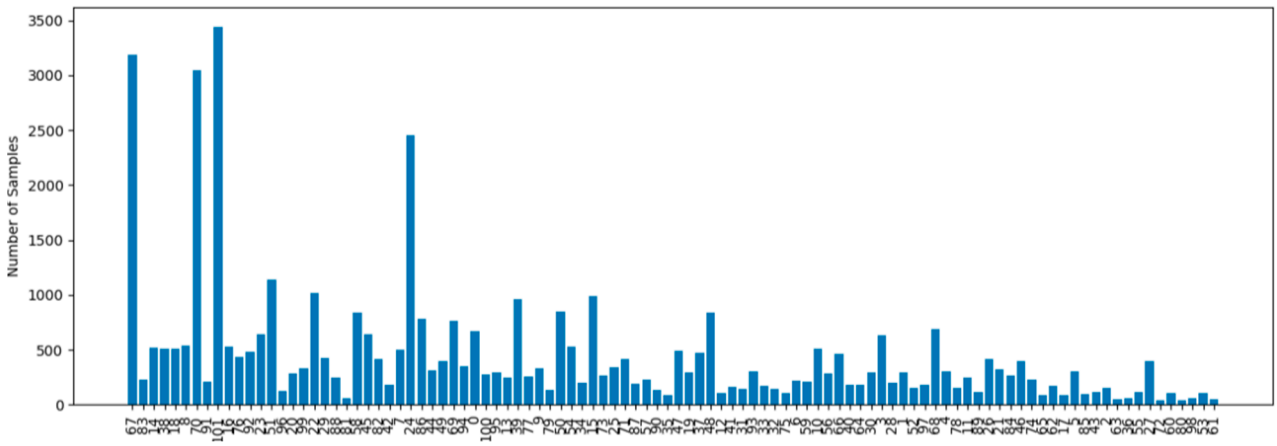

Figure 3. Class Distribution of IP102.

4. Methodology

This section reviews the existing Pre-Trained Classification Models used in this work, including MobileViT and EfficientNetV2B2. It then details how these models can be used and integrated into a deep learning architecture that is suitable for on-edge classification using various model quantization techniques and data augmentation solutions.

4.1. Pre-Trained Classification Models Used in This Work

MobileViT [39] is a novel vision transformer developed based on the ViT (Vision Transformer) architecture, and it addresses the challenge of deploying large-scale transformer models on resource-constrained platforms.

The key innovation of MobileViT lies in its compact design, achieved through a series of optimizations tailored for mobile deployment. MobileViT utilizes smaller batch sizes to reduce computational complexity. The number of attention heads and layers is carefully tuned to strike a balance between model size and performance. It further incorporates depth-wise separable convolutions and pointwise convolutions, which significantly reduce the number of parameters while preserving expressive power. Knowledge distillation techniques are also utilized during training to transfer knowledge from a larger pre-trained ViT model to the compact MobileViT architecture.

MobileViT represents a significant advancement in vision transformer research, offering a lightweight yet powerful solution for deploying transformer-based models on mobile devices.

EfficientNetV2 is a state-of-the-art neural network architecture designed to achieve a superior performance with highly efficient computational resources. One key feature of EfficientNetV2 is its novel compound scaling method, which optimizes model scaling across multiple dimensions, such as depth, width, and resolution. It also introduces a new Residual Group (ResG) module, which replaces the standard residual blocks used in previous architectures. The ResG module incorporates a combination of depth-wise separable convolutions and pointwise convolutions. A novel training regime called RandAugment is also utilized, which applies random data augmentation techniques during training.

Overall, EfficientNetV2 represents a significant advancement in neural network architecture design, offering smaller models with faster training speeds while maintaining the performance of larger, more computationally expensive models.

4.2. Proposed Solution

In this work, we adopted a transfer learning/fine-tuning strategy. Initially, we utilized a MobileViT model pertained on ImageNet, accessible via HuggingFace. The initial results mirrored those reported in the prior literature, with the validation accuracy peaking at approximately 76.9%. Subsequently, we employed an EfficientNetV2B2 model, also pre-trained on ImageNet and directly accessible through Keras, which achieved a peak validation accuracy of 72.3%.

Diverging from previous studies, we explore alternative data augmentation techniques with the aim of improving on existing results. Data augmentation plays a crucial role in improving the generalization and robustness of deep learning models [40], particularly when dealing with limited training data [41]. In this study, we employed a series of augmentation techniques to enhance the diversity and richness of the training dataset, no matter how small it is.

The dataset was split into 80–10–10, corresponding to the training, validation, and test datasets, respectively. Along with the standard resizing and rescaling operations, we incorporated a set of additional augmentations. Images were randomly rotated by 90-degree increments, with the rotation angle sampled uniformly from the set $\{0^\circ, 90^\circ, 180^\circ,$

270°}. Each image was also subjected to a random horizontal and/or vertical flip operation. Finally, to mimic changes in lighting conditions which would be realistic considering the domain at hand, a random brightness adjustment was applied to each image, with the magnitude of the adjustment controlled by a parameter.

Experimentation revealed that models trained exclusively on heavily augmented data performed worse than those trained on the original dataset. In the context of insect classification, where subtle visual details often distinguish one species from another, certain augmentations may have inadvertently distorted these critical features, making class boundaries less discernible [42]. As a result, the artificial variations introduced by augmentation alone could hinder the model's ability to extract meaningful patterns. To address this issue, we adopted a mixed approach, merging the augmented images with the original dataset. By doing so, the model retained exposure to authentic insect features while still benefiting from the broader variability offered by augmentation. This balanced strategy, effectively doubling the training size, ultimately improved performance and stability, especially given the dataset's limited scale.

As mentioned, we leveraged the EfficientNetV2B2 architecture initialized with the weights of the ImageNet dataset. This model was instantiated using the TensorFlow Keras API. We configured the model to include the top classification, while excluding any additional preprocessing steps. To adapt the pre-trained model for our specific classification task, we appended a custom classification layer on top of the pre-trained architecture. This layer consisted of a fully connected dense layer with 15 units, corresponding to the number of classes in our dataset.

In Figure 4, we display the overall proposed system solution. Firstly, full classification models with floating-point weights and biases are created using either MobileViT or EfficientNetV2B2. Secondly, the generated models are quantized and converted into models suitable for on-edge deployment using a mobile application. In the subsections to follow, we detail the quantization techniques used and show which ones are applicable to each of the used CNNs. We also elaborate on the effect of quantization on model size and accuracy.

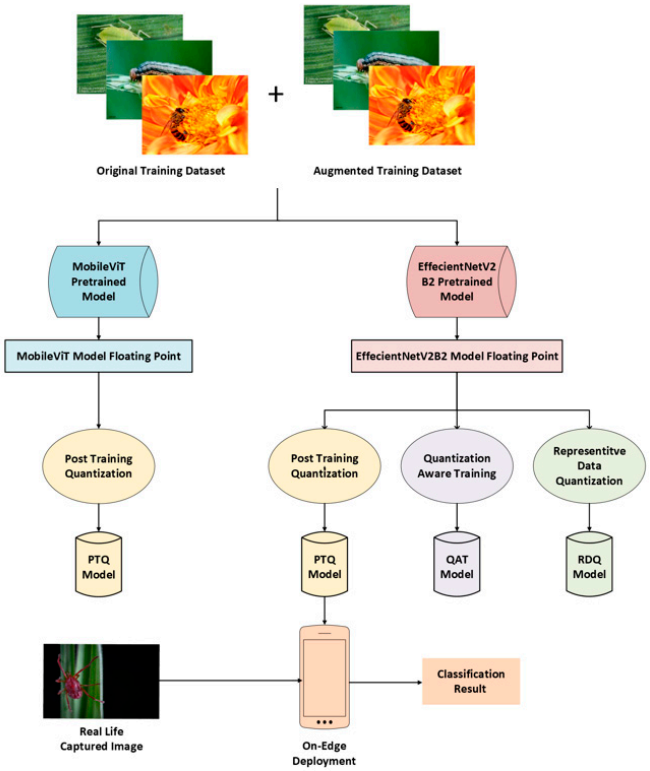

Figure 4. Overall Workflow for Model Training, Quantization, and Edge Deployment.

The training process involved feeding the combined training dataset (concatenation of the original train dataset and augmented dataset) to the model for a specified number of epochs. Additionally, we utilized the validation dataset to monitor the model's performance and prevent overfitting. To enhance the training efficiency and ensure the preservation of the best-performing model, we implemented two callbacks. The ModelCheckpoint callback was configured to monitor the validation accuracy and save only the best model based on this metric. Moreover, the EarlyStopping callback was employed to halt training if no improvement in the validation accuracy was observed after a certain number of epochs.

4.3. Quantization

To achieve our goal of deploying a high-performance insect detection model on mobile devices, we implemented various quantization techniques to convert our trained models into the TensorFlow Lite format. These techniques are essential for reducing the model size and computational load, ensuring that the models are suitable for running on resource-constrained devices without sacrificing significant accuracy. The quantization techniques applied include Quantization-Aware Training (QAT), Post-Training Quantization (PTQ), and Data Representative Training. Each of these approaches plays a crucial role in optimizing the model for mobile deployment.

4.3.1. Quantization-Aware Training

Quantization-Aware Training (QAT) is an advanced technique that simulates quantization during the model's training process. In this approach, quantization errors are modeled while the neural network is being trained, allowing the model to learn to mitigate potential inaccuracies caused by quantization [7]. By doing so, the model becomes more resilient to the numerical approximations made during deployment on mobile devices.

In our workflow, QAT was applied to fine-tune the model after initial training. During this stage, both the activations and weights are quantized to low-precision representations, typically 8-bit integers. Despite the lower precision, the model retains its predictive power, as it has been explicitly trained to adjust to these limitations. The primary advantage of QAT is that it helps preserve accuracy, making it highly suitable for scenarios where even minor drops in performance could compromise the model's effectiveness, such as in real-time insect detection on mobile devices.

4.3.2. Post-Training Quantization

Post-Training Quantization (PTQ) is a method applied after the model has completed its initial training. This technique is simpler than QAT because it does not involve any modification during the training process. Instead, PTQ transforms the trained model into a more compact form by reducing the precision of its weights and activations from 32-bit floating-points to lower-bit representations, such as 16-bit or 8-bit [39].

In our work, we applied dynamic range quantization, a form of PTQ that quantizes the model's weights from a 32-bit floating-point (float32) to 8-bit integers (int8), while leaving the activations in float32. This form of quantization provides a balance between reduced memory usage and inference performance, without requiring a calibration dataset. It is also compatible with a wide range of hardware backends.

4.3.3. Data Representative Training

Data Representative Training is a technique used to fine-tune quantized models by selecting a representative subset of the original dataset. This subset is used to guide the quantization process during PTQ, ensuring that the most critical features of the data are preserved in the lower-precision model [39].

In our case, a subset of insect images from each of the 15 classes was selected to represent the overall dataset. These representative data were used during the quantization process to ensure that the model's performance on mobile devices closely resembled its performance in the original 32-bit floating-point format.

5. Overall Workflow

As demonstrated by our experimental results, the most effective configuration of our proposed system involves training an EfficientNetV2B2 model in floating-point precision, and then converting it into a quantized model via Post-Training Quantization (PTQ). Figure 5 provides an illustration of the implementation pipeline of the proposed system, detailing how an input image progresses through each stage, and how the transformation from a floating-point model to an edge-optimized quantized model occurs.

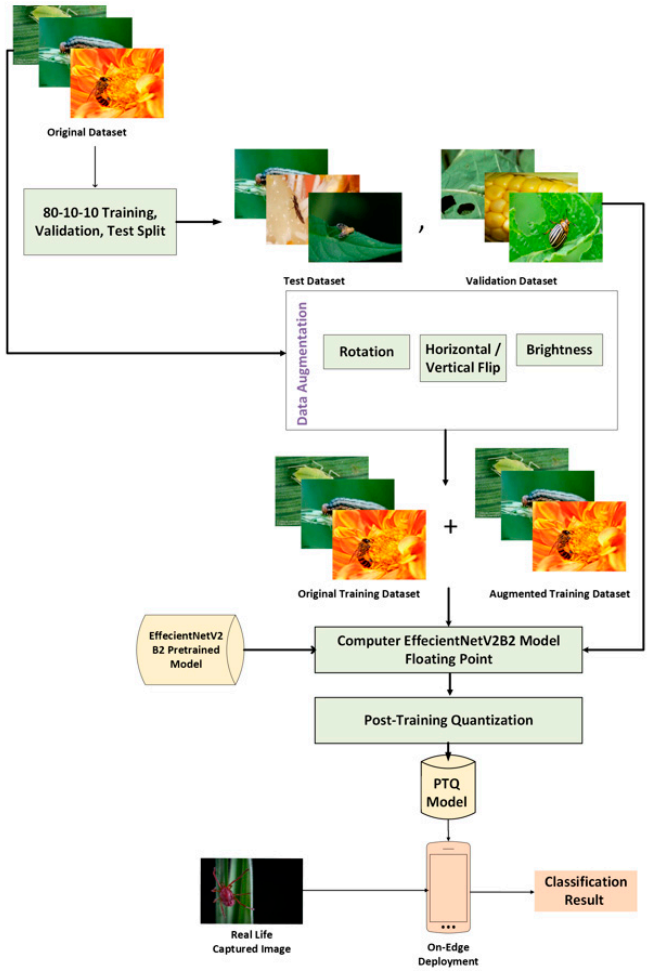

Figure 5. Detailed Implementation Pipeline of the Proposed System with the Best Model Generation Combination from Figure 2.

The process begins with comprehensive dataset preparation. Initially, a diverse collection of insect images is gathered, and each image is resized, normalized, and standardized to ensure consistency. The dataset is then split into training, validation, and test sets. Data augmentation techniques—such as rotations and flips—are applied to the training set to enhance diversity and improve the model's robustness.

Following preprocessing, the floating-point model is trained. The EfficientNetV2B2 model, pre-trained on a large-scale dataset, is fine-tuned using both the original and augmented training images. This fine-tuning occurs in floating-point (FP32) precision to maintain high fidelity during backpropagation. Throughout the training process, perfor-

mance is continuously monitored using the validation set, with early stopping and learning rate adjustments implemented as needed to optimize accuracy and prevent overfitting.

Once training is complete, Post-Training Quantization (PTQ) is employed to address the computational demands of edge devices. Although the floating-point model typically offers high accuracy, it is often computationally intensive. PTQ begins by passing a small calibration dataset—usually a subset of the training or validation data—through the trained model to collect essential activation statistics, such as the minimum and maximum values per layer. These statistics are then used to quantize the model's weights and activations, usually to 8-bit integers. This quantized model features significantly reduced memory requirements and faster inference speeds, while still preserving a level of accuracy close to that of the original floating-point model.

A direct comparison between the floating-point and quantized models highlights key differences. The floating-point model, operating in FP32, uses higher precision that ensures a robust performance, but at the cost of greater computational resources. In contrast, the PTQ model, which compresses weights and activations to int8, substantially reduces both the memory footprint and inference time due to hardware optimizations for integer operations. Although this quantization process can introduce a slight decrease in accuracy, our results demonstrate that a carefully calibrated PTQ model maintains performance levels that are nearly equivalent to the original.

Finally, the performance of both the floating-point and PTQ models is evaluated on the test set using metrics such as accuracy, model size (measured from the TensorFlow Lite quantized model representations), and inference time (time it takes the model to process and produce a result for a specific image). The empirical results confirm that the quantized model not only retains near-original accuracy, but also provides significant efficiency gains, aligning with our goal of achieving an edge-optimized performance. Table 2 highlights the main steps in the workflow as discussed above, and includes pseudo-code examples for each stage.

Table 2. Workflow Pseudo-Code.

Step	Description	Pseudo-Code Example
1	Load pre-trained FP32 model	model = tf.keras.models.load_model("EfficientNet_FP32.h5")
2	Prepare calibration dataset (representative subset)	calibration_data = select_subset(train_dataset)

3	Define representative data generator for calibration	def representative_data_gen():
		for input_value in calibration_data:
		yield [input_value]

4	Convert FP32 model to INT8 using PTQ	converter = tf.lite.TFLiteConverter.from_keras_model(model) converter.optimizations = [tf.lite.Optimize.DEFAULT] converter.representative_dataset = representative_data_gen tflite_quant_model = converter.convert()
5	Save quantized INT8 model	with open('EfficientNet_INT8.tflite', 'wb') as f: f.write(tflite_quant_model)
6	Deploy quantized INT8 model on edge device	deploy_model("EfficientNet_INT8.tflite")

The models were all trained on a Tesla P100 with a 16 GB GDDR6 VRAM. To demonstrate the practical application of our methodology, we developed a mobile application using the cross-platform Flutter framework, ensuring broad device compatibility. This application allows farmers to capture or upload images of insects, which are then processed by an on-device TensorFlow Lite model (the PTQ model) to classify insect species locally.

This design eliminates the need for an internet connection, ensuring a rapid and reliable performance in remote agricultural settings. Further details about the mobile app's design and operation are provided in Appendix A.

6. Results and Discussion

To our knowledge, the only prior study using the Dangerous Farm Insects Dataset is reported in [10]. Given the limited number of images available, the researchers of [10] implemented a transfer learning approach, utilizing pre-trained deep learning models to capture features of field pests more effectively. The method leveraged the use of ResNet, MobileNet, and VGG, with the goal of identifying the most suitable architecture for classifying dangerous farm insects.

Specifically, ref. [10] explored the use of models including ResNet-50V2, MobileNetV2, and Xception. While ResNet-50V2 showed potential in the initial testing, signs of overfitting suggested it might benefit from further refinement through hyperparameter tuning. ResNet-152V2 showed comparable behavior but did not offer notable performance gains over ResNet-50V2. MobileNetV2 maintained a stable performance during both training and validation, though the testing results revealed some limitations. In contrast, Xception provided significant performance improvements, especially in testing, positioning it as a leading candidate for the classification task. After hyperparameter optimization, Xception achieved a validation accuracy of 77.7%, outperforming baseline models. Although the study does not explicitly report test accuracy, a visual inspection of their results indicates a test performance with a classification accuracy of around 70%.

As for our proposed solutions, after fine-tuning our pre-trained models and utilizing augmentation, we achieved higher classification accuracy than the work reported in [10]. This is partially because we used newer and better models which are more suitable for image classifications such as MobileNet and EfficientNet. Additionally, the authors in [10] did not implement any specific augmentation techniques, which further distinguishes our approach. The results of our proposed solution are given below, in Table 3.

Table 3. Base Model Results in Comparison to Existing Work. The best results are highlighted in bold.

Model	Validation Accuracy (%)	Test Accuracy (%)
MobileNetV2 [10]	72.3	N/A
Xception [10]	77.7	~70
MobileViT (ours)	**82.6**	73.4
EfficientNetV2B2 (ours)	80.9	**77.8**

The results in Table 3 demonstrate that our proposed models, MobileViT and EfficientNetV2B2, achieved a superior performance compared to the baseline models used in previous works [10]. MobileNetV2, a baseline model, achieved a validation accuracy of 72.3%, but test accuracy was not reported in [10], suggesting limited applicability of this lightweight architecture to the complex features present in our insect dataset. Xception, another baseline model, improved upon MobileNetV2, with a validation accuracy of 77.7% and an approximate test accuracy of 70%, though its lower test accuracy may indicate susceptibility to overfitting, likely due to the absence of data augmentation in the original approach. In contrast, our MobileViT model, fine-tuned and supported by targeted augmentation techniques, achieved a validation accuracy of 82.6% and a test accuracy of 73.4%, highlighting the benefits of MobileViT's transformer-based design for capturing intricate insect image patterns. However, the highest test performance was achieved with our EfficientNetV2B2 model, which attained validation and test accuracies of 80.9% and 77.8%, respectively. The model's compound scaling method allowed for an optimal balance

between model size and depth, resulting in robust generalization across unseen data. These findings underscore the efficacy of combining advanced architectures with data augmentation to enhance model generalizability and address dataset limitations, positioning it as an optimal choice for on-device insect classification.

The ablation study summarized in Table 4 illustrates the incremental impact of individual augmentation techniques on our EfficientNetV2B2 model's accuracy. Without any augmentation, the model achieved a baseline accuracy of 74.5%, indicating a limited generalization capability. Horizontal and vertical flips individually improved the accuracy slightly, highlighting the benefit of these realistic transformations. Rotation augmentations showed the highest individual performance gain, reflecting the value of rotational invariance for insect images. Adjustments in brightness also contributed positively by simulating varying environmental lighting conditions. The combination of all augmentation techniques led to the highest model accuracy, underscoring the complementary benefits of applying multiple targeted data augmentation strategies to enhance model generalization and robustness.

Table 4. Ablation Study on Augmentation Techniques.

Augmentation Technique	Accuracy (%)
No Augmentation (Baseline)	74.5
Horizontal Flip only	75.8
Vertical Flip only	75.2
Rotation (90-degree increments)	76.3
Brightness Adjustment only	75.5
Combined (All Techniques, Final Model)	77.8

It is important to recognize that while augmentation can significantly enhance generalization, certain augmentation methods, when applied excessively or without careful consideration, may negatively impact performance. Excessive rotations or aggressive image transformations can inadvertently distort critical morphological features essential for insect classification, such as wing patterns, antennae positions, or body contours. These subtle but distinct visual characteristics, which the model relies upon to differentiate between similar insect classes, become difficult to discern when excessively manipulated. Therefore, augmentations must be thoughtfully balanced and applied strategically, preserving critical visual features to maintain high model accuracy and reliable generalization.

More relevant to this work, and as outlined in Section 4, we applied various quantization techniques to convert the full models into TensorFlow Lite formats optimized for edge deployment. Table 5 presents the impact of these quantization techniques on model size, classification accuracy, and inference times. The results illustrate the trade-offs associated with each quantization method and provide insights into achieving efficient, on-device inference while maintaining high classification accuracy.

Table 5. Quantized Models Results. The best results are highlighted in bold.

	Quantization Type	Model Size (MB)	Inference Time (Per Test Dataset) (s)	Inference Time (Per Image) (s)	Test Accuracy (%)
MobileViT	None	22	1.9	0.01	73.4
MobileViT	Post-Training Quantization	5.4	59.56	0.34	66.1
EfficientNetV2B2	None	33	41.09	0.24	**77.8**
EfficientNetV2B2	Post-Training Quantization	9.6	35.40	0.20	**77.8**
EfficientNetV2B2	Quantization-Aware Training	22.1	44.53	0.26	74.9
EfficientNetV2B2	Representative Data Quantization	10.4	39.67	0.23	77.2

In our experiments, the inference speed measurements reported in Table 5 were obtained using the TensorFlow Lite runtime integrated within our mobile application, developed with the Flutter framework. These results were recorded on a Nothing Phone (2), manufactured by Nothing Technology Limited in London, UK, sourced from a UAE retail shop, and running Android 15. This device is powered by the Qualcomm Snapdragon 8+ Gen 1 chipset built on a 4 nm process, featuring an octa-core CPU configuration with one Cortex-X2 core at 3.19 GHz, three Cortex-A710 cores at 2.75 GHz, and four Cortex-A510 efficiency cores at 2.0 GHz. The system is further supported by an Adreno 730 GPU, clocked at 900 MHz, and 12 GB of LPDDR5 RAM running at 3200 MHz, ensuring a flagship-level performance for on-device machine learning inference. This detailed hardware profile reflects a typical, high-performance mobile environment, and reinforces the practical relevance of our deployment results in real-world agricultural settings.

In addition to the performance metrics presented, our mobile application has been developed using the Flutter framework, ensuring cross-platform compatibility across both Android and iOS devices. In our experiments, the system was deployed on a Nothing Phone (2) running Android 15, which is equipped with a high-resolution camera that meets the standard pixel specifications required for accurate insect classification. The camera's resolution and sensor quality are sufficient to capture the fine details needed for a reliable model performance in real-world agricultural settings. Although our current evaluation focused on Android deployment, the Flutter-based development guarantees that our approach is equally compatible with iOS.

Table 5 presents the impact of various quantization techniques on model size, inference time, and test accuracy for MobileViT and EfficientNetV2B2, demonstrating the trade-offs in efficiency and accuracy that are critical for edge deployment. The base MobileViT model, known for its small and lightweight architecture, achieved the best inference time at 0.01 s per image, with a compact size of 22 MB and a test accuracy of 73.4%. However, after applying Post-Training Quantization, the inference time increased to 0.34 s per image, while the accuracy dropped to 66.1%. This performance degradation can be attributed to MobileViT's current lack of support in Keras as a pre-trained model, which limits our solution to Post-Training Quantization. This limitation also prevents the implementation of Quantization-Aware Training (QAT) and Representative Data Quantization for the MobileViT model.

Given these constraints, EfficientNetV2B2 emerged as the primary choice for deployment, primarily because its native Keras support allows for a comprehensive and flexible implementation of various quantization strategies. As shown in Table 3, the baseline EfficientNetV2B2 model achieved a 77.8% test accuracy at a size of 33 MB and with an inference time of 0.24 s per image. Applying Post-Training Quantization (PTQ)—a method that simply converts the trained weights and activations into low-precision formats after training—is particularly effective here. Because PTQ does not alter the model's learned parameters during training and leverages well-optimized integer arithmetic kernels, the model's representational quality remains virtually intact. This approach yields a 9.6 MB model that retains the full 77.8% accuracy and reduces the inference time to 0.20 s per image, demonstrating that PTQ can achieve sufficient compression with minimal trade-offs in the performance.

By contrast, Quantization-Aware Training (QAT) integrates quantization operations into the training loop itself. Although this technique theoretically helps the model adapt to the noise and reduced precision of quantization, the final quantization parameters may not perfectly match the actual data distribution at inference time. This mismatch can lead to suboptimal results, as seen in our QAT variant with a 74.9% accuracy, 22.1 MB size, and a 0.26 s inference time. Representative data quantization, which adjusts quantization

parameters based on a small dataset representative of the real-world input, helps mitigate this shortcoming of QAT. By tailoring the scaling factors to the actual operating conditions, Representative Data Quantization yields a better-aligned 10.4 MB model, with a 77.2% accuracy and a 0.23 s inference time, which is closer to PTQ's performance, but with the added benefit of data-driven fine-tuning. This improvement highlights how leveraging representative data can "correct" QAT's residual misalignments, providing a stronger balance between efficiency and accuracy. In essence, while PTQ offers a straightforward, near-optimal compression strategy, and QAT attempts to build quantization robustness during training, Representative Data Quantization steps in to refine the quantization parameters post hoc, ensuring that the final deployed model operates under conditions more closely matching those it will encounter in practice.

Notably, our test accuracy on TensorFlow Lite using the post-training quantized EfficientNetV2B2 model is effectively equivalent to the base model validation accuracy reported in the literature. This finding highlighted the effectiveness of our optimized approach, enabling the deployment of a lightweight yet accurate model for real-time insect classification in resource-constrained environments.

To assess the energy efficiency of our mobile insect classification application, we conducted experiments using the same Nothing Phone 2 (4700 mAh battery) running Android 15. The test was performed under controlled conditions—with the phone fully charged, screen brightness fixed, and minimal background activity—to ensure consistent measurements. During a 5-min session (300 s), the application processed 80 image classifications sequentially, consuming a total of 34.6 mAh. This corresponds to an average energy consumption of approximately 0.43 mAh per inference. By converting this energy usage using a nominal battery voltage of 3.85 V, we estimate an active power draw of 1.60 W. These results are summarized in Table 6 and demonstrate that our system maintains low power consumption, a critical factor for prolonged edge deployment in resource-constrained agricultural environments.

Table 6. Power Consumption Summary.

Metric	Value
Total Classifications	80
Duration	5 min (300 s)
Total Energy Used	34.6 mAh
Energy per Inference	0.43 mAh
Estimated Power Draw	1.60 W

To further evaluate the real-world performance of our insect classification application, we manually measured the end-to-end latency on a Nothing Phone 2. Using a stopwatch, we recorded the total time from user input to output, for each of 20 separate classification instances. The measurements revealed an average latency of 2.19 s, with a standard deviation of 1.19 s, a median latency of 1.58 s, and a range spanning from 1.00 to 3.81 s. These latency results, which are summarized in Table 7, offer additional insight into the system's responsiveness under typical operating conditions, and confirm that the application meets the requirements for practical on-device deployment.

The confusion matrix generated using the TensorFlow Lite Model of EfficientnetV2B2 with Post-Training Quantization is displayed in Figure 6.

Table 7. Latency Speed Summary.

Metric	Value
Number of Trials	20
Average Latency (s)	2.19
Standard Deviation (s)	1.19
Minimum (s)	1
Maximum (s)	3.81
Median (s)	1.58

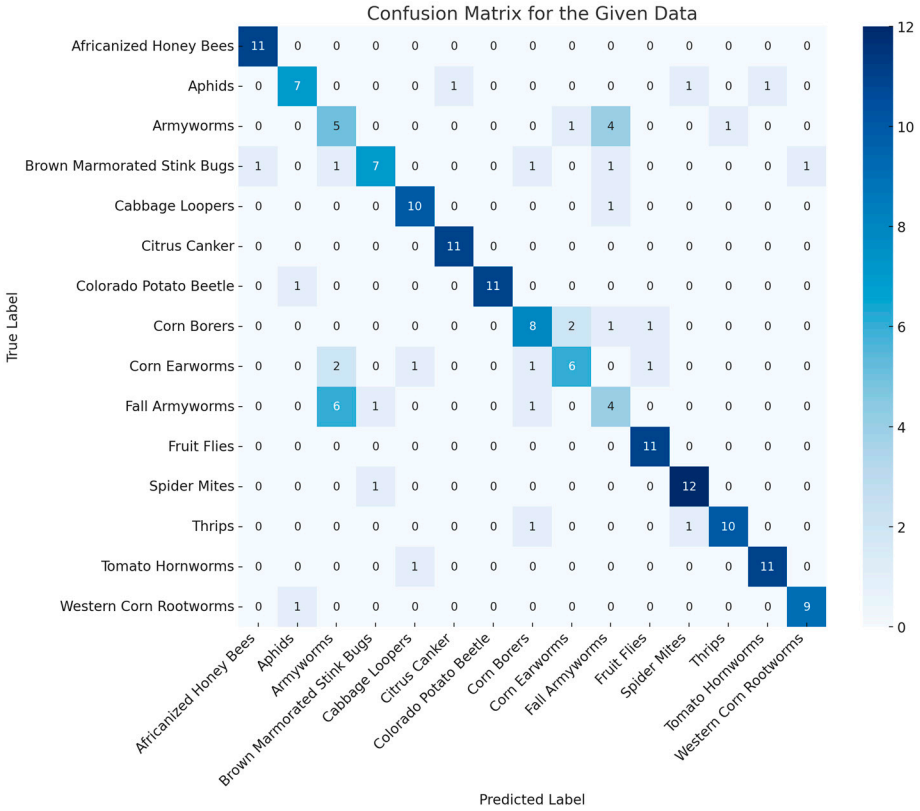

Figure 6. Confusion Matrix of EffecientNetV2B2 with Post-Training Quantization.

Having integrated the quantized EfficientNetV2B2 model using Post-Training Quantization into our proposed mobile application, we conducted a series of tests to evaluate the model's real-world performance in classifying insect species. For each of the fifteen classes, five random images were selected from the test set, resulting in a total of seventy-five tests. The model's predictions were recorded, and the results were compiled into a confusion matrix, which provides a detailed view of the model's accuracy and any misclassifications across the classes. This matrix highlights the model's accuracy, with the majority of predictions aligning with the ground truth, though some misclassifications exist between certain insect species with similar visual characteristics.

The model accurately classified 77.0% of the test images across 15 insect classes, showing good performance in categories like Africanized Honeybees, Fruit Flies, and Spider Mites. Such classes have high counts along the diagonal of Figure 6. An in-depth analysis of our confusion matrix reveals several specific misclassification patterns that warrant further investigation. For example, the model correctly classified only four out of eleven samples of Fall Armyworms (Row 9), resulting in an accuracy of approximately

36%. Notably, six samples of Fall Armyworms were misclassified as Armyworms and one as Western Corn Rootworms, indicating that the model frequently confuses these classes, likely due to their morphological similarities or overlapping features in the training data. Similarly, for Armyworms (Row 2), while five samples were correctly classified, there were additional errors, with one instance misclassified as Corn Earworms, and four as Fall Armyworms. Furthermore, Corn Borers were correctly identified in eight cases, but were confused with Corn Earworms (two samples) and Fall Armyworms (one sample).

These patterns suggest that certain insect species with similar visual characteristics are challenging for the model to distinguish. Given this, a potential strategy for future improvement could involve merging or re-evaluating visually similar classes—such as Armyworms and Fall Armyworms—if domain knowledge supports that their distinctions are ambiguous even to experts. Such refinements may help enhance model robustness and overall classification accuracy. The performance of the deployed mobile model was further evaluated using a confusion matrix based on user testing, as shown in Figure 7.

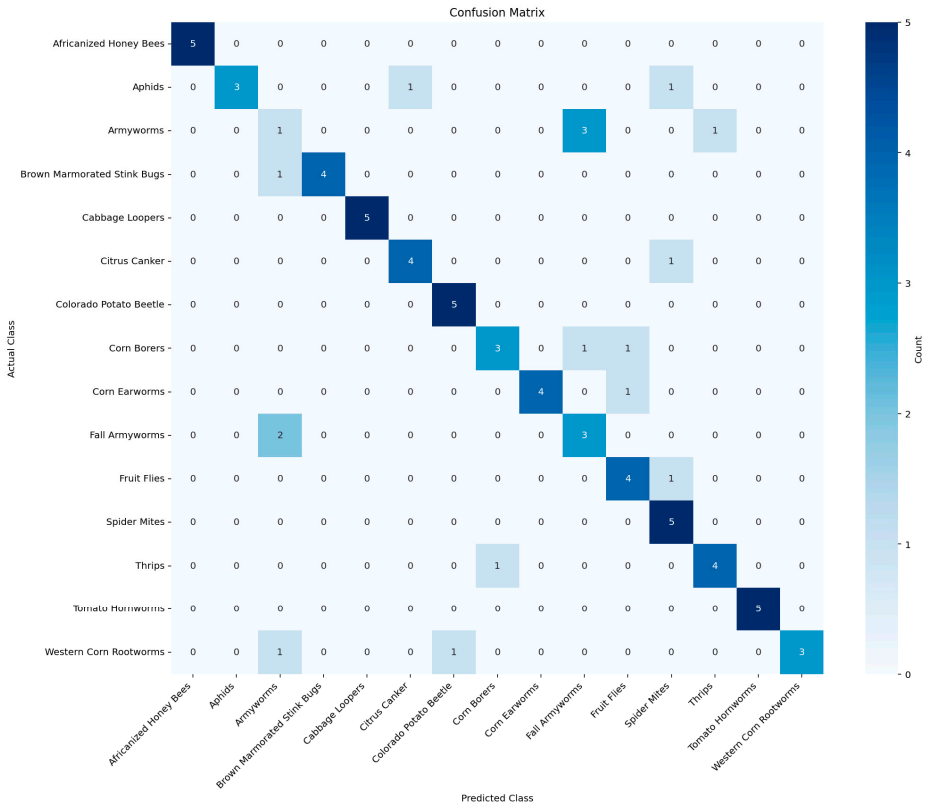

Figure 7. User-Tested Confusion Matrix of the Deployed Mobile Model.

To further validate the robustness and generalizability of our solution, we applied it to the IP102 dataset [7]. As mentioned in Section 3, this larger, more diverse benchmark is widely used for insect pest classification, making it ideal for this validation. With over 75,000 images across 102 insect pest classes, the dataset exhibited considerable variation in the number of samples per class. While some classes contained thousands of images, others were severely underrepresented, with as few as a few dozen samples. This imbalance posed a risk of biasing the predictive model towards the overrepresented classes, potentially leading to poor generalization on minority classes. Addressing this imbalance was critical to ensuring the fairness and robustness of the trained model.

To mitigate this issue, we adopted a systematic dataset balancing strategy, aimed at equalizing the number of samples per class while maintaining diversity. Initially, we determined a target of 300 samples per class, balancing the trade-off between dataset size

and class uniformity. For classes with more than 300 images, we employed under-sampling by randomly selecting 300 images, ensuring a representative subset was retained. For underrepresented classes, a combination of oversampling and augmentation techniques was used. Augmentation involved generating synthetic images through transformations such as rotation, zooming, flipping, brightness adjustments, and color shifts. These transformations helped create diverse variations of existing images, improving the model's ability to generalize to unseen data. Additionally, for classes with very few samples, existing images were duplicated to reach the target sample size, ensuring no class remained underrepresented.

We then followed the methodology outlined in Section 5, splitting the dataset and augmenting the training set before merging it with the original training dataset. Using this combined dataset, we trained an EfficientNetV2B2 model in floating-point precision, giving us an accuracy of 59.6%. Subsequently, we applied the three quantization strategies evaluated in this study, which are as follows: Post-Training Quantization (PTQ), Quantization-Aware Training (QAT), and Data Representative Quantization. The results obtained from these models are presented in Table 8, alongside a comparison with the results from related works.

Table 8. Quantized Models Results for IP102 Dataset.

	Quantization Type	Model Size (MB)	Test Accuracy (%)
InceptionNetV3 [43]	None	-	57.08
FR-ResNet-50 [44]	None	-	55.24
EfficientNetV2B2 (ours)	None	33	**59.6**
EfficientNetV2B2 (ours)	Post-Training Quantization	9.6	**59.6**
EfficientNetV2B2 (ours)	Quantization-Aware Training	22.1	58.3
EfficientNetV2B2 (ours)	Representative Data Quantization	10.4	59.6

To contextualize these results, we compared our quantized models to two baseline studies that utilized InceptionNetV3 [43] and ResNet [44]. We specifically selected these models because, like our own, they do not incorporate specialized hybrid strategies tailored to this dataset. As shown in Table 4, InceptionNetV3 and ResNet achieved accuracies of 57.08% and 55.24%, respectively. Our EfficientNetV2B2 model surpassed both baselines, even before quantization, reaching 59.6% accuracy. More importantly, after applying quantization, our solution maintained or closely approached this accuracy while reducing the model size, and thus the computational footprint.

Among the three quantization strategies evaluated—Post-Training Quantization (PTQ), Quantization-Aware Training (QAT), and Data Representative Quantization—PTQ stands out once again, as it did in our previous dataset. PTQ reduced the model size from 33 MB to 9.6 MB without any loss in accuracy, sustaining a 59.6% test accuracy. Representative Data Quantization also preserved the original accuracy at 59.6% with a slightly larger size (10.4 MB), while QAT achieved 58.3% accuracy at 22.1 MB. Although QAT proved useful, it did not offer the same degree of accuracy retention as PTQ or Representative Data Quantization.

These results on IP102 affirm the scalability and versatility of our approach. By replicating the improved efficiency and robust accuracy previously demonstrated on a smaller dataset, we confirm that our methodology is not limited to a single context. Instead, it remains effective on a more complex and extensive dataset, reinforcing its potential for widespread application in resource-constrained, real-world agricultural environments.

Our analysis revealed a complex relationship between augmentation strategies and the characteristics of the Dangerous Farm Insects Dataset. Models trained exclusively on heavily augmented data performed worse than those trained on the original dataset. This

counterproductive effect was likely due to the dataset's small size and the subtle visual distinctions between insect classes, where excessive transformations, such as large rotations or aggressive brightness adjustments, inadvertently distorted critical features that define class boundaries.

These findings highlight the importance of domain-specific augmentation design. The application of more restrained transformations, such as brightness adjustments and flips, aligns with the dataset's real-world variability while preserving critical visual cues. This result emphasizes that augmentation strategies must be adapted to the dataset's specific properties to maximize their effectiveness, particularly for small, specialized datasets like ours.

7. Conclusions

In this study, we focused on optimizing machine learning models for real-time insect classification on edge devices, specifically targeting dangerous farm insects. This is highly relevant in the current climate crisis, as changing environmental conditions continue to influence both biodiversity and insect behavior. By employing transfer learning and fine-tuning techniques on MobileViT and EfficientNetV2B2 architectures, we achieved state-of-the-art accuracies of 82.6% and 77.8% on validation and test datasets, respectively. We further demonstrated that advanced quantization techniques—including Post-Training Quantization, Quantization-Aware Training, and Data Representative Quantization—can significantly reduce model size and computational load, facilitating seamless deployment on mobile devices without sacrificing accuracy.

To validate and strengthen the generalizability of our approach, we extended our experimentation to a larger, more diverse dataset (IP102), confirming that our method scales effectively across different domains and insect classes. We also developed a mobile application integrating the quantized EfficientNetV2B2 model, showing that comparable accuracy to the original unquantized version can be maintained while achieving an efficient, on-device performance.

Our study highlights the potential of leveraging deep neural networks and vision transformers for pest detection in scenarios where domain-specific datasets are small and challenging to collect. By utilizing pre-trained models, we achieve high accuracy, even in resource-constrained environments, addressing critical challenges in agriculture. As climate change alters ecosystems, it exacerbates pest management difficulties by enabling invasive species, extending breeding cycles, and increasing crop damage. Our approach, supported by EfficientNet's optimized architecture and advanced quantization techniques, ensures scalable, real-time pest detection on edge devices, eliminating the need for continuous connectivity. This enables targeted interventions, reduces reliance on broad-spectrum pesticides, and promotes sustainable agricultural practices, demonstrating the broader implications of scalable, localized pest management solutions in combating global food security threats.

Future works could explore further optimization strategies, such as domain-specific data augmentation or ensemble methods, to enhance classification accuracy. An interesting direction for future work is exploring the impact of applying data augmentation selectively based on class-specific characteristics. Given that insect classes differ significantly in morphology and behavior (e.g., flying versus crawling insects), tailoring augmentation strategies to these distinct features could further enhance model accuracy and robustness. A targeted investigation into class-specific augmentation could better preserve critical visual features, improve model interpretability, and potentially increase accuracy in nuanced classifications.

Author Contributions: Conceptualization, M.H.A., I.E. and T.S.; methodology, M.H.A., I.E. and T.S.; software, M.H.A. and I.E.; validation, M.H.A., I.E. and T.S.; formal analysis, M.H.A. and I.E.; investigation, M.H.A. and I.E.; resources, M.H.A. and I.E.; data curation, M.H.A. and I.E.; writing—original draft preparation, M.H.A., I.E. and T.S.; writing—review and editing, M.H.A., I.E. and T.S.; supervision, T.S.; project administration, T.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this research are publicly available and can be found on https://www.kaggle.com/datasets/tarundalal/dangerous-insects-dataset (accessed on 15 March 2024). Other supporting material is available on request. The code used for generating results is available on GitHub through https://github.com/ibeksheir/Classification-of-Agricultural-Insects-with-Mobile-Development (accessed on 15 December 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

To maximize the impact of this study on farmers and facilitate effective mitigation of crop insect infestations, a mobile app was developed with a focus on accessibility and usability for end-users. The application was built using the cross-platform Flutter Framework, ensuring it is both versatile and compatible with a wide range of mobile devices. It empowers users to either upload images from their phone's gallery or capture images directly using the camera, providing flexibility in how they interact with the app. This workflow of using the app is demonstrated in Figure A1.

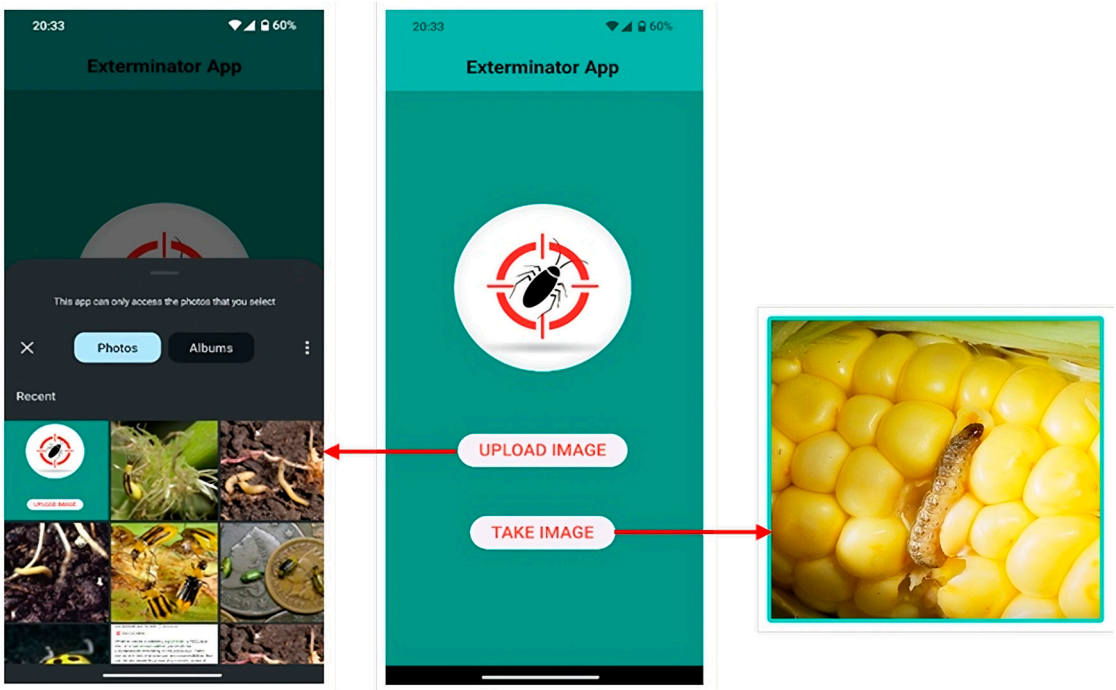

Figure A1. App's main page which allows a user to upload an Image or capture one using the camera.

The captured or selected image is then passed to the integrated TensorFlow Lite classification model. The model analyzes the image, identifies the insect species, and displays the classification result based on the class with the highest confidence. Additionally,

users are given the option to view the confidence level for each of the fifteen insect classes via the 'View Analysis' page, offering greater insight into the model's predictions.

To further enhance the user experience and provide actionable information beyond the classification result, the app interfaces with a Firebase Firestore database. This database contains comprehensive details on each insect species, including mitigation strategies that farmers can implement. This information can be updated by the system's admin when needed. Users can access this information by clicking the 'Mitigate' button on the result page or selecting any class from the 'Result Analysis' page. The results, along with related analysis pages, are illustrated in Figure A2.

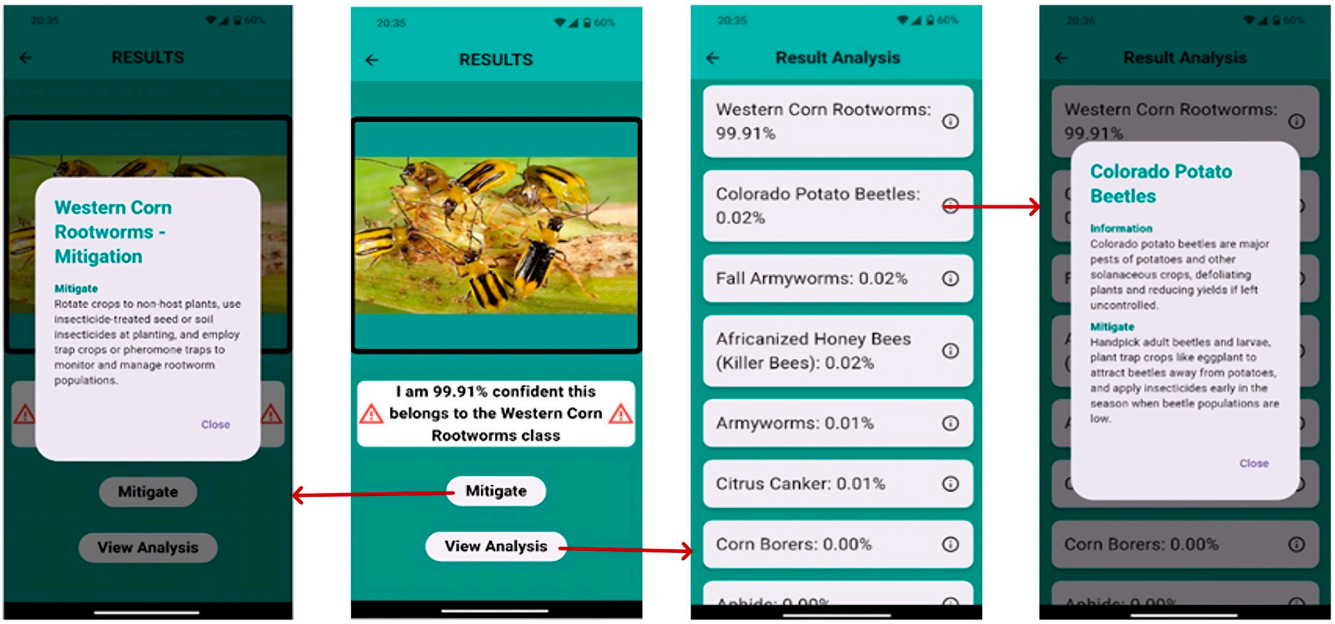

Figure A2. Results and Analysis via a Firebase Database.

References

1. Pereira, V.J.; Cunha, J.P.A.R.D.; Morais, T.P.D.; Oliveira, J.P.R.D.; Morais, J.B.D. Physical-chemical properties of pesticides: Concepts, applications, and interactions with the environment. *Agric. Sci.* **2016**, *32*, 627–641. [CrossRef]
2. Li, W.; Zheng, T.; Yang, Z.; Li, M.; Sun, C.; Yang, X. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. *Ecol. Inform.* **2021**, *66*, 101460. [CrossRef]
3. Quach, L.-D.; Nguyen, Q.K.; Nguyen, Q.A.; Lan, L.T.T. Rice pest dataset supports the construction of smart farming systems. *Data Brief* **2024**, *52*, 110046. [CrossRef]
4. Khalid, S.; Oqaibi, H.M.; Aqib, M.; Hafeez, Y. Small Pests Detection in Field Crops Using Deep Learning Object Detection. *Sustainability* **2023**, *15*, 6815. [CrossRef]
5. Xia, D.; Chen, P.; Wang, B.; Zhang, J.; Xie, C. Insect Detection and Classification Based on an Improved Convolutional Neural Network. *Sensors* **2018**, *18*, 4169. [CrossRef] [PubMed]
6. Visalli, F.; Bonacci, T.; Borghese, N.A. Insects Image Classification Through Deep Convolutional Neural Networks. In *Progresses in Artificial Intelligence and Neural Systems*; Springer: Singapore, 2021; pp. 217–228. [CrossRef]
7. Wu, X.; Zhan, C.; Lai, Y.-K.; Cheng, M.-M.; Yang, J. IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 8779–8788. [CrossRef]
8. Abeywardhana, D.L.; Dangalle, C.D.; Nugaliyadde, A.; Mallawarachchi, Y. An Ultra-Specific Image Dataset for Automated Insect Identification. *arXiv* **2021**. [CrossRef]
9. Gharaee, Z. A Step Towards Worldwide Biodiversity Assessment: The BIOSCAN-1M Insect Dataset. Available online: https://github.com/zahrag/BIOSCAN-1M (accessed on 29 April 2024).
10. Satpute, B.S.; Yadav, R.; Yadav, P.K. Dangerous Farm Insects Detection Using Transfer Learning and CNN. In Proceedings of the 2023 2nd International Conference on Futuristic Technologies (INCOFT), Belagavi, Karnataka, India, 24–26 November 2023; pp. 1–8. [CrossRef]

11. Tensorflow. Tensorflow Lite. Available online: https://www.tensorflow.org/lite/ (accessed on 29 April 2024).
12. Li, D.; Wang, X.; Kong, D. DeepRebirth: Accelerating Deep Neural Network Execution on Mobile Devices. *arXiv* **2017**. [CrossRef]
13. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *arXiv* **2017**. [CrossRef]
14. Orăşan, I.L.; Seiculescu, C.; Caleanu, C.D. Benchmarking TensorFlow Lite Quantization Algorithms for Deep Neural Networks. In Proceedings of the 2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 25–28 May 2022; pp. 221–226. [CrossRef]
15. Zhao, T.; Xie, Y.; Wang, Y.; Cheng, J.; Guo, X.; Hu, B. A Survey of Deep Learning on Mobile Devices: Applications, Optimizations, Challenges, and Research Opportunities. *Proc. IEEE* **2022**, *110*, 334–354. [CrossRef]
16. Rashid, A.A.; Ariffin, M.A.M.; Kasiran, Z. IoT-Based Flash Flood Detection and Alert Using TensorFlow. In Proceedings of the 2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 27–28 August 2021; pp. 80–85. [CrossRef]
17. Reda, M.; Suwwan, R.; Alkafri, S.; Rashed, Y.; Shanableh, T. AgroAId: A Mobile App System for Visual Classification of Plant Species and Diseases Using Deep Learning and TensorFlow Lite. *Informatics* **2022**, *9*, 55. [CrossRef]
18. Aldamani, R.; Abuhani, D.A.; Shanableh, T. LungVision: X-ray Imagery Classification for On-Edge Diagnosis Applications. *Algorithms* **2024**, *17*, 280. [CrossRef]
19. Varam, D.; Khalil, L.; Shanableh, T. On-Edge Deployment of Vision Transformers for Medical Diagnostics Using the Kvasir-Capsule Dataset. *Appl. Sci.* **2024**, *14*, 8115. [CrossRef]
20. Handhayani, T.; Hendryli, J. Leboh: An Android Mobile Application for Waste Classification Using TensorFlow Lite. In *Intelligent Systems and Applications*; Arai, K., Ed.; Springer: Cham, Switzerland, 2023; pp. 53–67.
21. Suharjito; Elwirehardja, G.N.; Prayoga, J.S. Oil palm fresh fruit bunch ripeness classification on mobile devices using deep learning approaches. *Comput. Electron. Agric.* **2021**, *188*, 106359. [CrossRef]
22. Campoverde, A.; Barros, G. Detection and Classification of Urban Actors Through TensorFlow with an Android Device. In *Information and Communication Technologies of Ecuador (TIC.EC)*; Efrain Fosenca, C., Morales, G.R., Cordero, M.O., Botto-Tobar, M., Martínez, E.C., León, A.P., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 167–181.
23. Benhamida, A.; Varkonyi-Koczy, A.R.; Kozlovszky, M. Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics. In Proceedings of the 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary, 2–4 June 2020; pp. 537–542. [CrossRef]
24. Deshan, P.; Yapa, N.; Perera, D.; Lunugalage, D.; Wijekoon, J.L. Smart Snake Identification System using Video Processing. In Proceedings of the TENCON 2021–2021 IEEE Region 10 Conference (TENCON), Auckland, New Zealand, 7–10 December 2021; pp. 539–544. [CrossRef]
25. Elliott, D.; Otero, C.E.; Wyatt, S.; Martino, E. Tiny Transformers for Environmental Sound Classification at the Edge. *arXiv* **2021**. [CrossRef]
26. Zualkernan, I.; Judas, J.; Mahbub, T.; Bhagwagar, A.; Chand, P. An AIoT System for Bat Species Classification. In Proceedings of the 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), Bali, Indonesia, 27–28 January 2021; pp. 155–160. [CrossRef]
27. Wang, J.; Lin, C.; Ji, L.; Liang, A. A new automatic identification system of insect images at the order level. *Knowl.-Based Syst.* **2012**, *33*, 102–110. [CrossRef]
28. Xie, C.; Zhang, J.; Li, R.; Li, J.; Hong, P.; Xia, J.; Chen, P. Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning. *Comput. Electron. Agric.* **2015**, *119*, 123–132. [CrossRef]
29. Kasinathan, T.; Singaraju, D.; Uyyala, S.R. Insect classification and detection in field crops using modern machine learning techniques. *Inf. Process. Agric.* **2021**, *8*, 446–457. [CrossRef]
30. Lim, S.; Kim, S.; Park, S.; Kim, D. Development of Application for Forest Insect Classification using CNN. In Proceedings of the 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 18–21 November 2018; pp. 1128–1131. [CrossRef]
31. Venugoban, K.; Ramanan, A. Image Classification of Paddy Field Insect Pests Using Gradient-Based Features. *Int. J. Mach. Learn.* **2014**, *4*, 1–5. [CrossRef]
32. Zhu, L.-Q.; Ma, M.-Y.; Zhang, Z.; Zhang, P.-Y.; Wu, W.; Wang, D.-D.; Zhang, D.-X.; Wang, X.; Wang, H.-Y. Hybrid deep learning for automated lepidopteran insect image classification. *Orient Insects* **2017**, *51*, 79–91. [CrossRef]
33. Tannous, M.; Stefanini, C.; Romano, D. A Deep-Learning-Based Detection Approach for the Identification of Insect Species of Economic Importance. *Insects* **2023**, *14*, 148. [CrossRef]
34. Kirkeby, C.; Rydhmer, K.; Cook, S.M.; Strand, A.; Torrance, M.T.; Swain, J.L.; Prangsma, J.; Johnen, A.; Jensen, M.; Brydegaard, M.; et al. Advances in automatic identification of flying insects using optical sensors and machine learning. *Sci. Rep.* **2021**, *11*, 1555. [CrossRef] [PubMed]

35. Meier, R.; Hartop, E.; Pylatiuk, C.; Srivathsan, A. *Towards Holistic Insect Monitoring: Species Discovery, Description, Identification and Traits for All Insects*; Royal Society Publishing: London, UK, 2024. [CrossRef]
36. Santaera, G.; Zeni, V.; Manduca, G.; Canale, A.; Mele, M.; Benelli, G.; Stefanini, C.; Romano, D. Development of an autonomous smart trap for precision monitoring of hematophagous flies on cattle. *Smart Agric. Technol.* **2025**, *10*, 100842. [CrossRef]
37. Tomar, P.; Kaur, G. *Artificial Intelligence and Iot-Based Technologies for Sustainable Farming and Smart Agriculture*; IGI Global: Hershey, PA, USA, 2021.
38. Muhammed, D.; Ahvar, E.; Ahvar, S.; Trocan, M.; Montpetit, M.-J.; Ehsani, R. Artificial Intelligence of Things (AIoT) for smart agriculture: A review of architectures, technologies and solutions. *J. Netw. Comput. Appl.* **2024**, *228*, 103905. [CrossRef]
39. Mehta, S.; Rastegari, M. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. *arXiv* **2021**. [CrossRef]
40. Khosla, C.; Saini, B.S. Enhancing Performance of Deep Learning Models with different Data Augmentation Techniques: A Survey. In Proceedings of the 2020 International Conference on Intelligent Engineering and Management (ICIEM), London, UK, 17–19 June 2020; pp. 79–85. [CrossRef]
41. Ottoni, A.L.C.; de Amorim, R.M.; Novo, M.S.; Costa, D.B. Tuning of data augmentation hyperparameters in deep learning to building construction image classification with small datasets. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 171–186. [CrossRef]
42. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
43. Reza, T.; Mehedi, N.; Tasneem, N.A.; Alam, A. Identification of Crop Consuming Insect Pest from Visual Imagery Using Transfer Learning and Data Augmentation on Deep Neural Network. In Proceedings of the 2019 22nd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 18–20 December 2019; pp. 1–6. [CrossRef]
44. Ren, F.; Liu, W.; Wu, G. Feature Reuse Residual Networks for Insect Pest Recognition. *IEEE Access* **2019**, *7*, 122758–122768. [CrossRef]