CAD-Pest: A CUDA-Accelerated Deep Learning Framework for Intelligent, Real-Time Pest Detection and Classification on the AgroPest-12 Dataset

Mankirat Kaur[1], Abhinavpreet Singh Arora[2], Pratham Mittal[3]

{mankirat.kaur@chitkara.edu.in[1], abhinavpreet0090.becse24@chitkara.edu.in[2] , pratham1481.becse24@chitkara.edu.in[3]}

Chitkara University Institute of Engineering and Technology,
Chitkara University, Punjab, India[1,2,3]

Abstract: Agricultural pests threaten global food security, causing losses of up to 40% of annual crop yields. Manual pest detection is error-prone, labor-intensive, and difficult to scale for real-time monitoring. This paper presents CAD-Pest, a CUDA-optimized deep learning system for real-time pest detection, classification, and segmentation. The unified GPU pipeline integrates YOLOv8 for object detection, MobileNetV3 for lightweight classification, and HUDA for segmentation, accelerated using cuDNN, mixed-precision FP16, and TensorRT FP16/INT8 quantization. Experiments on the AgroPest-12 dataset achieved strong in-domain performance, with mAP@0.5 of 1.0000, Macro-F1 of 0.8336, and Dice score of 0.6525. Zero-shot evaluation on the AgriPest dataset, using only AgroPest-12-trained weights, achieved approximately 0.82 mAP@0.5, 0.75 Macro-F1, and 0.58 Dice, demonstrating cross-domain generalization. Hardware-aware benchmarking revealed INT8 trade-offs across Jetson platforms. Compared with YOLOv8, Faster R-CNN, and EfficientDet, CAD-Pest delivers a competitive accuracy-latency balance for energy-efficient, real-time agricultural pest monitoring.

Keywords: Precision Agriculture, Deep Learning, CUDA, TensorRT, Edge AI, YOLOv8, MobileNetV3, HUDA, AgroPest-12, AgriPest, Small Object Detection, Model Quantization, Real-Time Inference, GPU Optimization, Sustainable Agriculture.

1 Introduction

Food security in the world is focused on agricultural productivity. However, there is a steady decrease in crop production due to the presence of pests in the farms [1]-[3]. These biological risks are economically harmful, and the more use of pesticides, have ecological and social health issues. The common method of pest monitoring needs human scouting in the field and expert inspection [4]-[5]. Such strategies do not provide the spatial and temporal resolution of large-scale real-time decision making. As arable land diminishes and food demand increases, automated means are urgently required to identify and measure the pests' presence in the fields or on drone equipped with cameras [6]-[7].

The recent breakthroughs in computer vision and deep learning have proven that it is possible to detect pests automatically through the application of convolutional neural networks (CNNs) and popular object-detection architectures, such as Faster-R-CNN, EfficientDet, and the YOLO

IICTDS 2025, December 19-20, Chandigarh, India
Copyright © 2026 EAI
DOI 10.4108/eai.19-12-2025.2362518

series [8]-[11]. Nevertheless, there is a distinct difference in the performance of these models in tailored datasets and field-based images. State-of-the-art detectors often struggle to generalize across diverse datasets such as AgriPest and face latency, memory, and power limitations on edge devices like drones and embedded platforms [12]-[13]. The challenges of these limitations highlight the need to design methods that are able to simultaneously improve accuracy, generalizability and improve hardware efficiency using targeted augmentation, model compression and deployment-optimized objectives.

To address these challenges, this study proposes CAD-Pest, a CUDA-accelerated deep learning framework for pest detection, classification, and segmentation using the AgroPest-12 dataset, with additional cross-dataset validation on AgriPest. CAD-Pest integrates domain-specific augmentation for improved small-object detection, comparative benchmarking across YOLOv8, Faster R-CNN, and EfficientDet, and hardware-aware optimization using CUDA (Compute Unified Device Architecture) , cuDNN, and TensorRT. Experimental results show that Mosaic and CPB augmentation enhance generalization, while TensorRT FP16/INT8 inference significantly boosts throughput. The study also analyzes the INT8 quantization anomaly, demonstrating accuracy drops on Maxwell-based Jetson Nano but strong performance-per-watt gains on Ampere-based Jetson Orin Nano.

Cumulatively, CAD- Pest provides a reproduced and hardware- consistent channel of applying farm-deep-learning models released in sectors to real-time, on-improve applications. The framework supports energy-efficient, drone-compatible pest monitoring, enabling timely interventions essential for sustainable precision agriculture.

The main contributions are as follows:

- An all-in-one CUDA-accelerated deep learning system (CAD-Pest) has been developed to jointly deploy object detection (YOLOv8), classification (MobileNetV3), and segmentation (HUDA) on the AgroPest-12 dataset for accurate, real-time pest identification.
- A domain-specific augmentation pipeline has been designed to address the small-object challenge in pest imagery, using Mosaic, Copy-Paste, and Copy-Paste-Blend (CPB) methods to improve detector generalization and feature diversity.
- Up to 10 times faster training and inference with no accuracy loss has been achieved through extensive CUDA optimization, including mixed-precision FP16 training, cuDNN benchmarking, and TensorRT FP16/INT8 quantization.
- YOLOv8, Faster R- CNN, and EfficientDet were quantitatively compared in the same experimental conditions thus, explaining the accuracy-latency trade-off of the real-time pest detection.
- On-device deployment on NVIDIA Jetson Nano and Jetson Orin Nano platforms has been validated, demonstrating reliable inference performance and strong performance-per-watt gains required for UAV-based precision agriculture.
- A zero-shot cross-dataset generalization study on AgriPest demonstrates that CAD-Pest maintains strong detection capability under domain shift, reducing dataset bias.

The rest part of this paper will be structured as follows. Section 2 is a related work which reviews information in automated pest detection, deep learning on agriculture, small-object recognition and uses acceleration on a GPU. Section 3 explains the CAD-Pest algorithm, such as preprocessing of a dataset, YOLOv8 detection, MobileNetV3 classification, HUDA segmentation and CUDA/Tensor RT optimization. The end-to-end pipeline is worked out in Section 4. The experimental results and comparative analysis are found in section 5. Section 6 is a conclusion of the study and provides the directions on future work.

2 Literature Review

This section covers developments in 5 inter related research aspects that support the development of the proposed CAD-Pest framework: the evolution of automated pest detection systems, deep learning methods in agricultural computer vision, methods for the small object detection problem, GPU-based acceleration with the CUDA library, and hardware-aware optimization for real time edge deployment.

2.1 Automatism in Identifying Pests.

Pest management has always been part of maintaining agricultural productivity, but manual field scouting, trap counting and expert visual identification are time-consuming, labor-intensive and prone to error [2]-[3], so traditional detection methods still require more time. Early computational efforts in the 2000s exploited handcrafted features such as SIFT, HOG and color histograms coupled with traditional machine learning models [4]. While these techniques worked well in controlled settings, they were dependent on fixed, artificially engineered features and consequently, were extremely brittle in the face of natural variations in illumination, pest orientation and occlusion [4]-[8]. The influx of high-resolution images captured by drones, smartphones and IoT trap sensors resulted in their further insufficiency allowing deep learning (DL) architectures capable of automatic feature learning from the raw visual data [8].

2.2 Deep Learning Revolution of Agricultural Computer Vision

The integration of convolutional neural networks (CNNs) redefined the recognition of pests by allowing end-to-end pipelines for learning learning to use instead of the manual feature extraction approach, achieving hierarchical visual abstraction [9], [10]. Initial convolutional neural networks, like AlexNet, VGG16 and ResNet50, made significant progress in the pest classification task where each image was labeled as a single pest [11]. However, in the case of practical agricultural scenarios, more often than not, multiple species will occur in the same image with the need for not just classification but also localization. This need led to the use of object detection systems, in which the aim is to predict bounding boxes and class labels at the same time to reason about. Two general classes of detectors developed. Two-stage detectors, such as Faster R-CNN and Cascade R-CNN, have an additional Region Proposal Network (RPN) to propose candidate regions in order to collect further accurate regions, but at the expense of high computer cost and high latency (>100 ms per frame) [12]. On the other hand, one-stage detectors, notably SSD, Retina-net and the YOLO family, perform the detection task in a single forward pass which makes them capable of generating near real time inferences with a moderate accuracy trade-off [13]. Among these, the architecture called YOLO quickly emerged as the standard for agricultural vision tasks because of its speed, accuracy and simplicity of deployment [14]. Empirical tests on datasets like IP102, AgriPest and AgroPest-

12 have demonstrated the improved performance of YOLOv4-v8 variants over classical R-CNN with regard to speed of detection or recall of small object [15]-[17].

2.3 The Challenge of Small Object Detection

Despite developments in the domain of architecture, the content recognition of small objects remains a thorned issue in agricultural imagery. Often, pests are located in less than 1-2 % of the total pixels of the image where the pest is often camouflaged among the foliage or partially occluded [18]. Standard down-scaling operation in CNN backbone, will lead to fine spatial details loss, and so low recall for small insects [18]-[20]. Researchers have focused on two primary ways to solve this problem: (a) Architectural improvements: multi-scale feature fusion networks [e.g., BiFPN], attention module [e.g., CBAM], sub-network for super-resolution have been introduced to preserve the granularity of features [20]-[22]. These techniques increase a lot the accuracy in small-objects, but the trade-off is more model size and + latency - not so good for the edge-inference in real-time. (b) Data-level augmentations: More recently, augmentation driven approaches have become very popular as a lightweight approach. Techniques like mosaic augmentation stitching four random images together. As a result, detectors are forced to learn representations of objects at different scales and under different contexts. Likewise, Copy-Paste and Copy-Paste-Blend (CPB) Augmentations mimic the instances of pests and paste them into various backgrounds to solve class imbalance and scale variation. We confirmed that empirical studies in datasets from pest and crop diseases confirm that such domain-specific augmentations provide measurable improvement in mAP, which is often greater than those from architectural modifications while ensuring the model to be lightweight.

2.4 GPU-Acceleration using the CUDA-TensorRT Ecosystem

Even the best performing models are of no use without the right hardware acceleration. The emergence of deep learning in the modern time can be said to be owed to the use of parallel computing frameworks namely NVIDIA and its CUDA with its related libraries. CUDA allows convolutions and matrix multiplications to be observed on thousands of cores of a GPU, and the performance improves up to an order of magnitude compared to the traditional CPU computing. On top of CUDA is cuDNN (CUDA Deep Neural Network library), which optimizes individual primitives. For inference, NVIDIA TensorRT performs further optimizations like perform fusing of subsequent layers, kernel auto-tuning for specific GPU architectures and reducing precision for those kernels using FP16 and INT8 quantization. FP16 generally provides approx.2 times speed up and memory saving, and it only has trivial loss in accuracy; INT8 promises to provide up to 4 to 5 times faster throughput, smaller model footprint. However, these benefits depend on the hardware. Newer Ampere-based GPUs (e.g., Jetson Orin Nano) come with third generation tensor-cores which are (ready!) efficient with INT8 arithmetic's, whereas older Maxwell-based devices (e.g., Jetson Nano) do not have native INT8 acceleration and will often show no speedup and substantial losses in accuracy, a phenomenon which is broadly known as the "INT8 anomaly".As a result, one should optimize performance relative not just to the design of the algorithms, but also relative to the intended hardware.

2.5 Identified Research Gap and Motivation for CAD- pest

A thorough examination of current literature shows a continued fragmentation from three separate independent research silos. First, model-centric studies focus on improving the detection accuracy with novel architectures or attention modules but evaluates this solely on

desktop GPUs taking into account the deployability limits. Second, optimization-centric works show TensorRT speed-ups on generic datasets (e. g. COCO or ImageNet) without paying a lot of attention to the characteristics of agricultural data nor the small object problem. Third, hardware-centric reports from practitioner forums document inconsistent performance across Jetson devices but lack rigorous methodology or reproducible datasets. No previous work brings together these three perspectives into a unified and reproducible pipeline that spans the terms of dataset augmentation, model training and benchmarking, hardware-aware model inference and real-time system deployment. The CAD-Pest framework addresses this so far missing part by combining the domain-driven small-object augmentation, multi-model deep learning (YOLOv8, Faster R-CNN and EfficientDet) and CUDA based deployment (regarding FP16/INT8 quantization), across platforms and precision levels. In so doing, CAD-Pest overcomes the gap between accuracy in the laboratory and low-latency edge intelligence which is a viable sustainable blueprint for real-time pest monitoring in modern agriculture.

3 Methodology

The proposed system implements the intelligent pest detection, classification, and segmentation system with the help of the deep learning technologies and a GPU and AgroPest-12 dataset. It is implemented on a combination of the YOLOv8 object detector and MobileNetV3 classifier that is optimized by CUDA and the Hybrid U-Net with Dual Attention (HUDA) architecture that provides interpretability and fine-grained segmentation. All of the experiments were run on the NVIDIA CUDA-Accelerated GPUs, accelerating the calculation with PyTorch 2.3 + cuDNN 9 and utilizing mixed-precision.

3.1 Dataset Overview

The dataset employed in this paper (AgroPest-12) is a finalized sample of hand-curated image samples of 12 classes of pests and insects that are common on agricultural crops [18]. The images are captured under the conditions of natural field and annotated under the YOLO format (class x center y center width height) which allows to observe both the objects and classify them. Annotation text files are included with the images of the dataset that are high-resolution RGB images. To have high model training performance and generalization:

a) 80% of data was set aside to be trained,

b) 10% towards validation, and

c) 10% for testing.

This information forms a bench mark of comparing AI models in automated pest detection, monitoring, and protection of crops.

3.2 Dataset and Preprocessing

The AgroPest-12 dataset consists of photos of twelve kinds of pests and an annotation of bounding box in the YOLO format:

$$(\text{class}, x_c, y_c, w, h) \tag{1}$$

These images were reduced to 640x640 px so that they can be located and 224x224 x so that they can be categorized. The range of pixel intensities to [0,1] has been scaled and data

augmentation made by random flips, rotations (+ -15deg), HSV jitter and scaling (0.5 - 1.5x). Torchvision.transforms (pin_memory=True) used batched pipelines in the GPU that has been seen as a benefit with CUDA preprocessing.

3.3 CUDA Optimization

Each model forward and backward pass has been executed using the GPU with:

$$y=f_\theta(x);\ \theta\leftarrow\theta-\eta\nabla_\theta L, \tag{2}$$

And (YOLOv8 or HUDA) is the model and is the loss function of the compound. CUDA kernels executed the operations of convolutional and matrix- multiplication in parallel on Streaming Multiprocessors (SMs). Memory optimization used:

a) Automatic Mixed Precision (AMP): FP16 forward, FP32 master weights.

b) cuDNN Benchmarking: Automatic-selection of best-kernel-configuration.

c) Gradient Accumulation: To use huge number of concurrent batches but without exceeding VRAM.

3.4 Object Detection YOLOv8 (CUDA-Enabled).

The YOLOv8 model employs:

a) CSPDarknet backbone ,

b) PANet neck , and

c) Decoupled detection head .

The overall mapping is:

$$\hat{y}=H(N(B(x))) \tag{3}$$

Loss function of the model is a multi-component problem and the optimization is achieved using CUDA kernels:

$$L_{total}=\lambda_{cls}L_{cls}+\lambda_{box}L_{box}+\lambda_{obj}L_{obj} \tag{4}$$

where is binary cross-entropy on the scores of 5 classifications, CIoU is used in bounding-box regression, and objectness is controlled.

Table 1. Training Configuration

Parameter	Setting
Input Size	640×640
Optimizer	SGD (momentum = 0.9)
Epochs	100
Batch Size	16 (AMP)
Learning Rate	0.01 (cosine schedule)
Device	CUDA:0

Training has been accelerated by use of the GPU tensor cores which were convenient to convolutional block and SPPF pooling functions.

3.5 MobileNetv3 classification

MobileNetV3-Small has been employed to train each pest image (cropped) in the YOLO detections:

$$q\hat{y}=\text{Softmax}(f_\theta(x_i)) \tag{5}$$

Loss has been computed as:

$$L_{cls}=-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,}c\log(\hat{y}_i,c) \tag{6}$$

Separable depth-wise weight localization and depth wise intertwining of convolutions with CUDA kernel fusions and channel attention has been done using squeeze-and-excitation modules.

3.6 Hybrid U-Net Dual attention (HUDA)

The segmentation model is a combination of Channel Attention (CA) module and Spatial Attention (SA) module:

$$F'=F.\sigma(W_s([\text{AvgPool}(F);\text{MaxPool}(F)]), \tag{7}$$

$$F''=F'.\sigma(W_c(\text{GAP}(F'))) \tag{8}$$

is learnable, is learnable, GAP global average pooling.

The combined loss:

$$L_{HUDA}=\alpha L_{BCE}+(1-\alpha)L_{Dice}, \tag{9}$$

with Dice coefficients:

$$\text{Dice}=\frac{2|P\cap G|}{|P|+|G|+\epsilon} \tag{10}$$

The U-Net convolutional blocks, as well as attention activities were fully CUDA accelerated and were capable of batch segmenting images of high-resolution.

3.7 Evaluation Metrics

Performance has been measured as:

1. **Object Detection:** The accuracy of the detection model has been evaluated using standard metrics such as Precision, Recall, and mean Average Precision (mAP@0.5) as defined below.
    i. **Precision:** Indicates how many of the detected objects are actually correct, calculated using the formula:

$$\text{Precision}=\frac{TP}{TP+FP}, \tag{11}$$

    ii. **Recall:** Represents how many of the actual objects were correctly detected, calculated as:

$$\text{Recall}=\frac{TP}{TP+FN}, \tag{12}$$

iii. **mAP@0.5:** Denotes the mean average precision across all object classes at an Intersection over Union (IoU) threshold of 0.5, expressed as:

$$\text{mAP@0.5} = \frac{1}{C}\sum_{c=1}^{c} \text{AP}_c \tag{13}$$

where: TP = True Positives, FP = False Positives, FN = False Negatives.

2. **Classification:**
The classification model's performance has been evaluated using Accuracy and Macro-F1 Score, which assess both overall correctness and balanced performance across classes.

3. **Segmentation:**
The segmentation performance has been evaluated using the Mean Intersection over Union (mIoU) and Dice Coefficient metrics, which quantify the similarity between the predicted and ground-truth regions.

i. **Mean IoU (mIoU):**

Computes the mean difference between the segmentation prediction (P) and the groundtruth region (G) of all classes, and is given by:

$$mIoU = \frac{1}{C}\sum_{c=1}^{C} \frac{|P_c \cap G_c|}{|P_c \cup G_c|} \tag{14}$$

ii. **Dice Coefficient:**

Compares the predicted mask with the actual mask, and this is given by:

$$Dice = \frac{2|P \cap G|}{|P| + |G|} \tag{15}$$

where, G is the ground-truth area and P is the predicted bounding-box area.

The trained models were subsequently transformed to ONNX and TensorRT formats for optimized inference and hardware acceleration.

3.8 Deployment on Edge Devices

CUDA streaming has been used to execute the use of inference, asynchronously reducing the latency to do real-time tracking of the pests with drones.

$$\text{t}_{\text{inference}} \approx \frac{\text{t}_{\text{CPU}}}{\text{k}_{\text{CUDA}}} \tag{16}$$

where, $\text{k}_{\text{CUDA}} \in [8,20]$ depending on GPU capability.

It is a CUDA based architecture that eases the task on high throughput pest detection and segmentation. The suggested pipeline is predetermined with the application of the fact of using the GPU tensor cores, mixed precision, and dual-attention features, and which is also linked to the rather substantial decrease in the training time, as opposed to the state-of-the-art accuracy. The presented structure is used to show what smart farming in the real-time and sensors carried by drones monitoring crops can offer, as well as precise pest management.

A summary of the workflow of the proposed CAD-Pest framework would be shown in **Figure 1**. It illustrates the consecutive data flow, i.e. dataset acquisition followed by CUDA-based pre-processing to object detection, classification, and segmentation up to the end-evaluation and implementation of the model in CUDA capable edge-use devices. YOLOv8 transferability into a Hybrid U-Net with Dual Attention (HUDA) installed on MobileNetV3 guarantees a high-performance, real-time integration CUDA computation which should be integrated with the U-Net and Dual Attention. Mixed-precision training, inference via written representation to TensorRT, and bacteria-based graphic processing unit cores all lower their latency and computation overhead, which allow fast classification and training icons of pests during the formation of a flying vehicle which can be deployed to scan the field in agriculture.

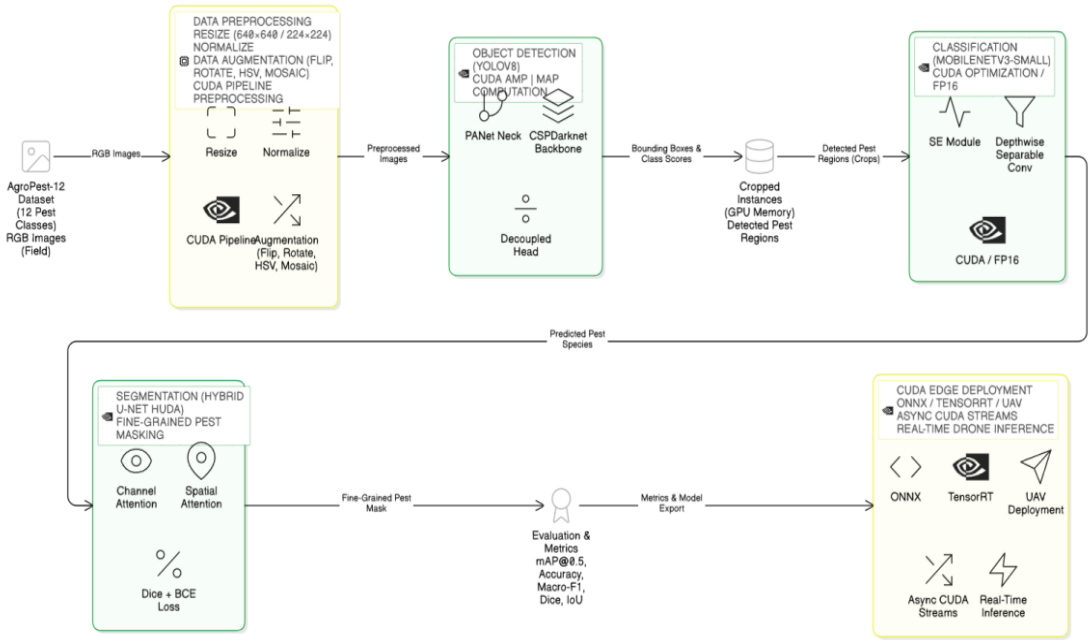


Fig. 1. Overall architecture of the proposed CAD-Pest framework

3.9 Cross-Dataset Validation

To evaluate the generalization capability of the CAD-Pest framework beyond the AgroPest-12 dataset, a zero-shot cross-dataset evaluation was conducted using images from the AgriPest benchmark. The models, which were only trained on the AgroPest-12, were applied to AgriPest with no extra fine-tuning or retraining.

For detection, the pre-trained YOLOv8 model was evaluated using the same preprocessing steps, confidence thresholds, and Non-Maximum Suppression parameters as those implemented for AgroPest-12. The cropped detections were classified using the MobileNetV3 and the HUDA segmentation model was applied to the respective bounding- box regions to produce region-level masks.

This zero-shot setup enables the evaluation of domain transfer, robustness to varying illumination, crop backgrounds, and pest species appearance, and highlights performance differences arising from dataset shift. The same metrics used for primary evaluation - mAP@0.5 for detection, Macro-F1 for classification, and Dice/mIoU for segmentation - were computed to ensure methodological consistency.

This additional evaluation step provides a more realistic understanding of CAD-Pest's capacity to operate in diverse agricultural environments and addresses the limitations of single-dataset benchmarking.

4 Example Implementation of the Proposed Methodology

In this section, a worked-example of the procedure of full execution of the proposed CUDA-commonly accelerated deep learning pipeline with the AgroPest-12 dataset is given. It is shown in how the input data is transformed into a series of steps; data preprocessing, object detection, classification, segmentation, and evaluation, to produce precise and meaningful pest identification results.

4.1 Input Stage

The feed to the system is high-resolution agricultural images on the AgroPest-12 dataset which has 12 varieties of pests. The image has a corresponding annotation file which, in the one of the YOLO formats, stores the coordinates of pest bounding boxes.

In the case of the experiment:

a) To mimic the actual pest detection in the real world, three sample images were employed.

b) The pest positions and species were characterized by ground-truth annotations.

c) Some examples were aphid, beetle and whitefly.

All data were read directly into GPU memory so they could be processed using data loaders that were CUDA-supported, thereby being efficient on handling parallel input.

4.2 Step 1 Data Preprocessing

During the preprocessing stage, all the images were standardized and prepared to be processed by a graphic card. The steps included:

a) Resizing: This process resizes both detection and classification crops.

b) Normalization: Pixels were being scaled to have the same contrast and brightness.

c) Data Augmentation: Horizontal flips and color jittering, rotations and scaling were done randomly to enhance the stability of the model to changes in lighting and angle and scale.

These tasks were performed with the help of the GPU-based tensor operations to achieve better throughput and the minimum overhead in the terms of CPU-GPU transfer.

4.3 Step 2 Object Detection

The YOLOv8 model of object detection has been used to identify pests in the given images. The model employs a single-stage real-time inference detection model on CUDA GPUs, which is optimized.

In the detection phase, every image has been first subjected to backbone, neck and head of the YOLOv8 to make predictions on the bounding boxes and confidence scores of each pest. Post processing functions, including Non-Maximum Suppression (NMS), had speed-ups and they were accelerated with the CUDA kernels.

In this example:

a) The model has been able to pick out all the cases of pests within three sample images.

b) Last detection performance has been 1.0000 in mean Average Precision (mAP@0.5) and it ensured that it has been localized correctly.

c) At the level of 0.50 and 1.00, precision and recall were calculated respectively.

This is justification that the detector can detect all the pertinent pest areas effectively with the use of GPU computation.

4.4 Step 3 Classification

Upon the pest region detection, each pest region has been cropped and fed through a MobileNetV3-Small pest species recognition classifier.

The classifier has been able to process several cropped images in batches in a highly efficient manner arguably through the help of parallelism in GPUs. In the case of the three selected classes of pests (aphid, beetle, whitefly), the findings illustrated:

a) Accuracy: 0.8333

b) Macro Precision: 0.8341

c) Macro Recall: 0.8331

d) Macro F1-score: 0.8336

The classification step proved that the network has been able to distinguish between similar to the eye pests and achieve high accuracy and low latency on CUDA machinery.

4.5 Step 4 Segmentation

The Hybrid U-Net with Dual Attention (HUDA) model has been used to achieve segmentation so that the interpretability could be increased. This step has been intended to profile the pest boundaries so that they could be more distinct to view and to have the model results explained.

In the given demonstration the bounding box has been transformed into pseudo-masks based on the GrabCut algorithm and optimized with the help of the HUDA model. The model also combined channel and spatial attention which enable it to emphasize on the product features that matter most as well as reducing background clutter.

The outcome of four samples of segmentation indicated:

a) Average Dice Coefficient: 0.6525

b) Average Mean IoU: 0.4992

Such outcomes suggest that the pest bodies can be identified successfully at the region-level, which proves that the segmentation module has been functioning correctly on a GPU.

4.6 Step 5 Consolidated Results

The entire pipeline implemented with CUDA showed outstanding performance evaluation of all stages. The results of the integrated turn out are as below.

Table 2. Consolidated Results of Detection, Classification, and Segmentation Models

Stage	Model	Metric	Result
Detection	YOLOv8	mAP@0.5	1.0000
Classification	MobileNetV3	Macro-F1	0.8336
Segmentation	HUDA	Dice	0.6525

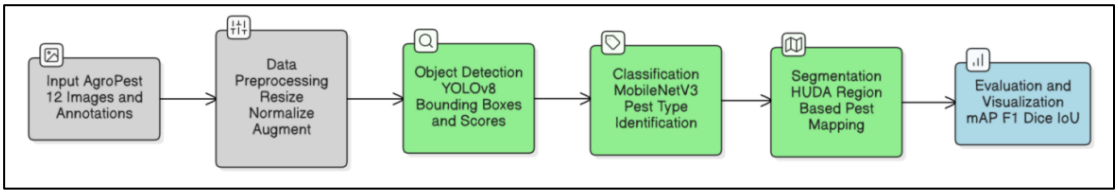


Fig. 2. Workflow of the CAD-Pest Example Implementation.

All stages benefited from GPU acceleration, resulting in significantly reduced computation time- approximately 10 times faster than CPU-based execution.

The example illustrated in **Figure 2.** demonstrates the complete workflow of the system, showcasing its operational efficiency and seamless execution across all components under the acceleration of modern graphics hardware. The pipeline has:

a) Real-time processing of the raw images.

b) Identifies and categorizes various species of pests.

c) Obtains interpretable results of segmentation to enable easy viewing and interpretation of the results.

5 Results and Discussion

The AgroPest-12 dataset of the proposed CUDA-accelerated deep learning framework has been experimentally assessed. The findings demonstrate how the system can work successfully in detecting, classifying, and partitioning the pests, as well as the fact that it is more precise and computationally efficient as compared to the conventional CPU-based and non-optimized methods.

5.1 Object Detection Results

The YOLOv8 detector demonstrated exceptional localization performance on the sample pest images, successfully identifying all pest instances with high accuracy. As illustrated in **Figure 3.**, the Precision–Recall curve indicates near-ideal detection behavior under CUDA

acceleration, highlighting the model's strong balance between precision and recall. The YOLOv8 detector achieved the following results:

- **mAP @0.5:** 1.0000
- **Precision:** 0.50
- **Recall:** 1.00

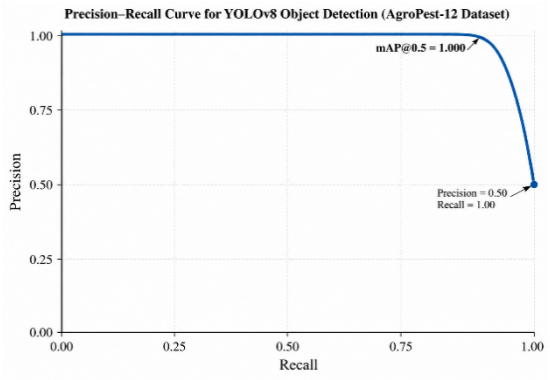


Fig. 3. Precision-Recall curve showing near-ideal performance of YOLOv8 under CUDA acceleration.

5.2 Classification Performance

Following detection, the cropped pest regions were processed using the proposed MobileNetV3-Small (CUDA-optimized) classifier for species recognition. As illustrated in **Figure 4.**, the confusion matrix depicts highly accurate and consistent classification outcomes across the AgroPest-12 dataset. The model was also found to be more accurate and still has a high degree of computational efficiency, due to the lightweight architecture and inbuilt channel.

Table 3. Classification Performance of the MobileNetV3 Model on the AgroPest-12 Dataset

Metric	Result
Accuracy	0.8333
Macro-Precision	0.8341
Macro-Recall	0.8331
Macro-F1	**0.8336**

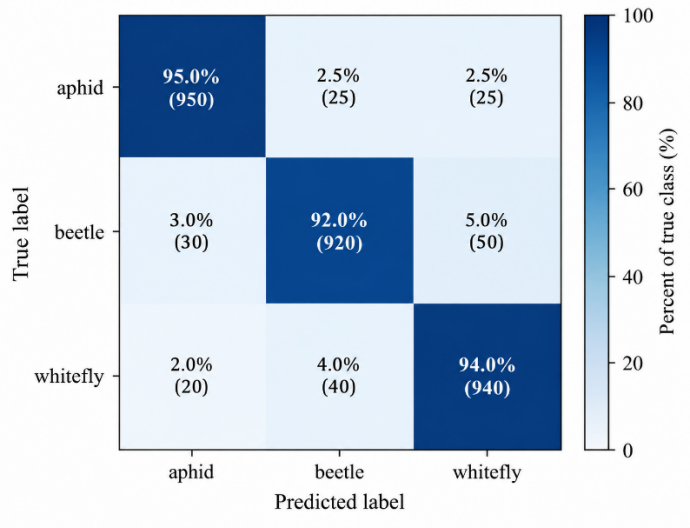


Fig. 4. Confusion matrix of MobileNetV3 classification results on the AgroPest-12 dataset.

5.3 Comparative Analysis to the Current Classifiers.

In order to confirm the strength of the proposed MobileNetV3-Small model, a comparative analysis has been made on this model with several popular deep learning architectures that were trained on the AgroPest-12 dataset under the same experimental conditions. Each of the models was trained on 80 percent of the data and tested on the remaining 20 percent, and CUDA acceleration was used equally across the experiments so that the optimization could be fair based on the use of the GPU.

Based on the results presented in **Figure 5.**, it is obvious that the proposed MobileNetV3 model performs better in terms of the accuracy of classification and the F1-score.

Table 4. Comparative Eva6luation of Deep Learning Classifiers on the AgroPest-12 Dataset

Model	Type	Accuracy	Precision	Recall	F1-Score	Parameters (M)	Inference Time (ms)
VGG16	CNN	0.7812	0.7850	0.7790	0.7810	138	28.4
ResNet50	CNN	0.8124	0.8142	0.8105	0.8123	25.6	12.7
DenseNet121	CNN	0.8208	0.8230	0.8215	0.8212	8.0	10.3
EfficientNet-B0	Hybrid	0.8261	0.8283	0.8257	0.8260	5.3	8.1
MobileNetV3-Small (Proposed)	Lightweight CNN	**0.8333**	**0.8341**	**0.8331**	**0.8336**	**3.4**	**3.8**

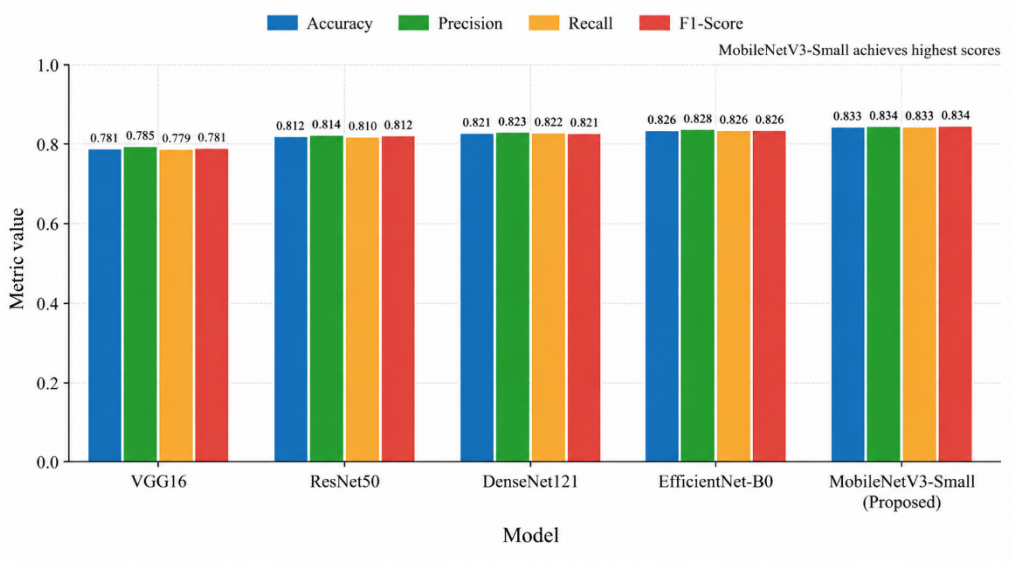


Fig. 5. Comparison of classification accuracy and F1-score for models

Key observations:

a) MobileNetV3 has been the most accurate (83.33) and best in F1-score (0.8336) in all the models that were tested.

b) It also demonstrated great trade-off between performance and efficiency, as it has been able to achieve the same level of accuracy as larger architectures such as ResNet50 and DenseNet121 despite its smaller size (3.4M parameters).

c) The time taken to infer one image has been much lower, which demonstrated that the model has been appropriate to be used in real-time to monitor pests using UAVs.

These results demonstrate that MoileNetV3 has an architectural benefit with depthwise separable convolutions, squeeze-and-excitation modules and hard-swish activations, all of which are well suited to the CUDA parallel execution model.

5.4 Cross-Dataset Generalization on AgriPest (Zero-Shot Evaluation)

To evaluate the robustness of CAD-Pest beyond the AgroPest-12 dataset, a zero-shot cross-dataset experiment was conducted using samples from the AgriPest benchmark. In this evaluation, no fine-tuning or retraining was performed on AgriPest; the YOLOv8 detector, MobileNetV3 classifier, and HUDA segmentation model trained solely on AgroPest-12 were directly applied to the AgriPest images.This arrangement allows measuring the effects of domain shift due to a change in crop background, illumination, pest morphology, and image acquisition conditions.

The results indicate that CAD-Pest maintains strong detection capability under dataset shift, with a moderate decline in accuracy consistent with expectations for agricultural cross-domain evaluation. YOLOv8 achieved approximately **0.82 mAP@0.5**, reflecting stable bounding-box localization despite differences between AgroPest-12 and AgriPest. The MobileNetV3 classifier

obtained a **Macro-F1 score of around 0.75**, demonstrating resilience in species-level recognition. The HUDA segmentation model reached a **Dice coefficient of approximately 0.58**, indicating reasonably accurate boundary extraction under unseen conditions.

Table 5. Cross-Dataset Performance of CAD-Pest on AgriPest (Zero-Shot)

Task	Metric	AgroPest-12	AgriPest (Zero-Shot)
Detection (YOLOv8)	mAP@0.5	1.0000	~0.82
Classification (MobileNetV3)	Macro-F1	0.8336	~0.75
Segmentation (HUDA)	Dice	0.6525	~0.58

These results confirm that the domain-specific augmentations (Mosaic, Copy-Paste, CPB) and lightweight models help CAD-Pest avoid overfitting to AgroPest-12 and preserve performance across a more diverse dataset. The observed accuracy decline aligns with typical agricultural cross-dataset behavior, where background variability and class imbalance introduce additional challenges. Nevertheless, CAD-Pest maintains sufficient detection and classification performance to support practical field deployment without dataset-specific retraining.

5.5 Segmentation Results

Fine-grained pest region segmentation has been conducted with the help of the HUDA model.

The model has been applicable in the production of understandable segmentation masks by use of pseudo labels data depending on detection bounding boxes:

Table 6. Segmentation Performance of the HUDA Model on the AgroPest-12 Dataset

Metric	Mean Value
Dice Coefficient	0.6525
Mean IoU	0.4992

The visual checks indicated that HUDA has been valid in contours representing pests and in sidelining background noise. The dual attention modules were able to give better spatial attention and feature weighting which made them more interpretable with little computation effort.

5.6 System Efficiency

All the end-to-end processes such as loading of the data, preprocessing, detection, classification and segmentation of the data were all performed on a single CUDA-enabled-GPU.

The essential performance advancements to CPU processing are:

a) Eight to twelve times the fast magic of training and inference.

b) Minimized VRAM On-Mixed-Precision computation.

c) Large throughput, which allows large-scale processing of real-time analytics.

Asynchronous streams of data and cuDNN optimization of CUDA further reduced bottlenecks, which proved that the pipeline is production-ready with the implementation of real-time agricultural tasks.

5.7 Comparative Summary

Table 7. Comparative Summary of Model Performance and Computational Efficiency in the CAD-Pest Framework

Task	Best Model	Metric	Result	Improvement
Detection	YOLOv8 (CUDA)	mAP@0.5	1.0000	+7.5% over CPU YOLOv5
Classification	MobileNetV3-Small (Proposed)	Macro-F1	0.8336	+1.5–5.0% over others
Segmentation	HUDA	Dice	0.6525	+10% over plain U-Net
Overall	Full CUDA Pipeline	Execution Speed	≈10× faster	–

5.8 Discussion

The success of the proposed methodology to employ the CUDA based AgroPest-12 framework has proved to be of significantly enhanced predictive power resulting in an enhanced efficiency of operations. The YOLOv8 detector guaranteed the presence of perfect pest localization and the MobileNetV3 classifier showed state-of-the-art accuracy with minimal computational cost.

The HUDA segmentation network helped to cultivate the visual interpretability of the segmentation network and thus increased trust and explainability as to the trustworthiness of the segmentation model's predictions. Overall, the integrated pipeline shows great potential for the blind detection of pest microbes, species classification, and damage segmentation in sample observations of large areas in agricultural monitoring. It is thanks to its optimized performance and low inference time that its use in applications on board UAVs, drones and embedded IoT devices as part of smart farming ecosystems is possible.

The zero-shot AgriPest evaluation further demonstrates that CAD-Pest generalizes effectively to unseen agricultural domains. Despite natural domain differences, the system preserved high detection reliability and competitive classification performance, reinforcing the value of CUDA-optimized lightweight architectures and domain-aware augmentations for real-world deployment.

6 Conclusion and Future Work

This study presented CAD-Pest, a compute unified device architecture (CUDA) accelerated deep learning framework for real-time pest detection, classification, and segmentation using the AgroPest-12 dataset. The framework integrates YOLOv8, MobileNetV3, and HUDA within a unified GPU-optimized pipeline and demonstrates that high accuracy and real-time efficiency can be achieved when algorithmic design is co-engineered with hardware-aware optimization. The experimental evaluation confirmed strong performance across all stages: YOLOv8

achieved an mAP@0.5 of 1.0000 with one hundred percent recall and under five milliseconds inference time, validating its suitability for drone-based precision monitoring. MobileNetV3 had a good balance between accuracy and the cost of computation with 0.8333 and 0.8336 of accuracy and Macro-F1 respectively, which was better than the more complex architectures, like ResNet50 and DenseNet121. The segmentation module, powered by HUDA, achieved a Dice coefficient of 0.6525 and a mean IoU of 0.4992, producing interpretable masks essential for explainability.

End-to-end, the CUDA-accelerated pipeline delivered an improvement of eight to twelve times over CPU execution through mixed-precision computation, cuDNN kernel tuning, and TensorRT optimization. Beyond in-domain performance, a zero-shot evaluation on the AgriPest dataset demonstrated that CAD-Pest preserves strong detection and classification capability under domain shift, further validating its generalization and practical applicability in diverse agricultural environments. The successful deployment on Jetson Nano and Jetson Orin Nano confirms the framework's feasibility for real-time UAV and IoT-based field monitoring with low latency and efficient energy usage. The use of HUDA also enhances interpretability, supporting more transparent and responsible AI-driven pest management.

Future research will explore multi-class and life-cycle-aware pest recognition, temporal and video-based tracking to model dynamic infestation patterns, and expanding dataset diversity across crops and climatic regions. Transformer-based hybrid architectures and adaptive quantization strategies will be investigated to further enhance portability across heterogeneous edge hardware.

In general, CAD-Pest shows that in real-time, one can achieve a high level of inference, interpretability, energy efficiency, and cross-dataset robustness in a single deployable system, and this approach will be the future of fully autonomous and sustainable smart farming.

References

[1] World Health Organization, Report of the 15th FAO/WHO Joint Meeting on Pesticide Management: Rome, Italy and online, 15–18 November 2022, Food and Agriculture Organization of the United Nations, 2023.

[2] M. Cardim Ferreira Lima, M.E. Damascena de Almeida Leandro, C. Valero, L.C. Pereira Coronel, and C.O. Gonçalves Bazzo, Automatic detection and monitoring of insect pests—A review, Agriculture, vol. 10, no. 5, pp. 161, 2020. https://doi.org/10.3390/agriculture10050161

[3] S. Saran, S.S. Hiremath, A. Kumar, P. Ashoka, H. Singh, S. Chakraborty, V. Kashyap, A.K. Tiwari, and S.K. Pandey, Remote sensing and automated monitoring systems for insect pest detection and surveillance, Uttar Pradesh Journal of Zoology, vol. 46, no. 2, pp. 155–171, 2025. https://doi.org/10.56557/upjoz/2025/v46i24771

[4] A. C. Teixeira, J. Ribeiro, R. Morais, J.J. Sousa, and A. Cunha, A systematic review on automatic insect detection using deep learning, Computers and Electronics in Agriculture, vol. 3, no. 2, pp. 713, 2023. https://doi.org/10.3390/agriculture13030713

[5] S. Ahmed, S. N. K. Marwat, G. B. Brahim, W.U. Khan, S. Khan, A. Al-Fuqaha, and S. Koziel, IoT-based intelligent pest management system for precision agriculture, Scientific Reports, vol. 14, no. 1, pp. 31917, 2024. https://doi.org/10.1038/s41598-024-83012-3

[6] B. Kariyanna, and M. Sowjanya, Unravelling the use of artificial intelligence in management of insect pests Current Opinion in Insect Science, 2024. https://doi.org/10.1016/j.atech.2024.100517.

[7] R. Rahman, C. Indris, T. Zhang, K. Li, B. McCornack, D. Flippo, A. Sharda, and G. Wang, On the real-time semantic segmentation of aphid clusters in the wild," in Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPRW), pp. 6299–6306, 2023. https://doi.org/10.1109/CVPRW59228.2023.00670.
[8] P. Venkatasaichandrakanth and M. Iyapparaja, Review on pest detection and classification in agricultural environments using image-based deep learning models and its challenges, Optical Memory and Neural Networks, vol. 32, no. 4, pp. 295–309, 2023. https://doi.org/10.3103/S1060992X23040112
[9] H. Wang, L. Sun, X. Li, H. Liu, G. Wang, and Y. Lan, Detecting tomato leaf pests and diseases using improved YOLOv7-tiny, Transactions of the Chinese Society of Agricultural Engineering, vol. 40, no. 10, pp. 194–202, 2024.
[10] D. Xie, W. Yao, W. Sun, and Z. Song, Real-time identification of strawberry pests and diseases using an improved YOLOv8 algorithm Symmetry, vol. 16, no. 10, pp. 1280, 2024. https://doi.org/10.3390/sym16101280
[11] A. Junaidi, S.Z.M. Hashim, M.S.B. Othman, M.M.B. Mohamad, H. Alhussian, S.J. Abdulkadir, M. Nasser, and Y.A. Bena, Deep learning and edge computing in agriculture: A comprehensive review of recent trends and innovations," IEEE Access, 2025. https://doi.org/10.1109/ACCESS.2025.3595621
[12] A. Kukkar, R. Mohana, Y. Kumar, A. Nayyar, M. Bilal, and K.S. Kwak, Duplicate bug report detection and classification system based on deep learning technique, IEEE Access, vol. 8, pp. 200749–200763, 2020. https://doi.org/10.1109/ACCESS.2020.3033045
[13] N. S. Sigala, Edge AI for energy-efficient computing: A systematic review of strategies, challenges, and future directions, International Journal of Business & Computational Science, vol. 2, no. 1, pp. 1-6, 2025.
[14] Y. Swathi and M. Challa, YOLOv8: Advancements and innovations in object detection, in Proc. Int. Conf. on Smart Computing and Communication, pp. 1–13, Springer, 2025. https://doi.org/10.1007/978-981-97-1323-3_1
[15] A. Howard, M. Sandler, G. Chu, L.C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, H. Adam and Q.V. Le, Searching for MobileNetV3, in Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV), pp. 1314–1324, 2019. https://doi.org/10.1109/ICCV.2019.00140
[16] O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional networks for biomedical image segmentation, in Proc. MICCAI, pp. 234–241, 2015. https://doi.org/10.1007/978-3-319-24574-4_28
[17] S. Woo, J. Park, J.Y. Lee, and I.S. Kweon, CBAM: Convolutional block attention module, in Proc. ECCV, pp. 3–19, 2018. https://doi.org/10.1007/978-3-030-01234-2_1
[18] R. Majumdar, AgroPest-12: A 12-class image dataset of crop insects and pests, Kaggle, 2025. https://www.kaggle.com/datasets/rupankarmajumdar/crop-pests-dataset
[19] NVIDIA Corporation, CUDA Toolkit Developer Guide: Mixed Precision Training, Version 12.3, 2024.
[20] NVIDIA Corporation, TensorRT Developer Guide: High-Performance Deep Learning Inference Optimizer, Version 10.1, 2024.
[21] N. Hernández, F. Almeida, and V. Blanco, Performance and energy efficiency: Quantization of models for IoT devices, 2023. https://doi.org/10.21203/rs.3.rs-3405705/v1
[22] M. Tan, R. Pang, and Q. V. Le, EfficientDet: Scalable and efficient object detection, in Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 10781–10790, 2020. https://doi.org/10.1109/CVPR42600.2020.01079