Grenze International Journal of Engineering and Technology, June Issue

Enhancing Crop Yield and Sustainability through an AI-Driven Pest Monitoring and Decision Support System

Anshuman Gaurav[1], Vikas Saini[2], Himanshu Kumar[3], Sonal Kumar[4], Rajneesh Singh[5] and Ravin Kumar[6]
[1-6]School of Computer Science and Engineering, Lovely Professional University Phagwara, India
Email: aryamangaurav@gmail.com, vikassn44@gmail.com, kumarhimanshu09032001@gmail.com, sonal.davbseb@gmail.com, rajneeshsingh5729@gmail.com, ravinpal2009@gmail.com

***Abstract*— Agricultural industry which is a large sector of the global economy continues to be struck by pest menaces which destroy crops and fill the economy with uncertainty. This particularly is crass to small farmers in such places as India. Old fashioned pest checks are a pain in the neck–they work too much, are too slow, and most of the time they miss the target until it is too late. It signifies that we are spraying chemicals all over and it is not economical as well as not environmentally friendly. We, therefore, addressed these issues through a project named as Smart-Crop-Monitor. It is an expensive automated system that detects and identifies pests in real-time. We trained the most recent YOLOv8 framework, which had been refined on 12 types of pests of a large and diverse dataset. It needs a full site, written in custom code. FastAPI is the in the background, managing data, on the front, a React interface makes it easy and enjoyable to see and interact with photos and videos. We designed the system to run on cheap edge hardware such as an old laptop or a single board computer in addition to a modular alerts system that can buzz you immediately through whichever channel you prefer. When we were on the validation, our YOLOv8n achieved 0.444 mAP50-95, which is something like, it is quite consistent and can be a practical tool. Overall, this paper provides a complete design of a hand-to-hand smart farming kit, demonstrating that the integration of the latest AI, web technologies, and IoT can be scaled up to make the pest control data-driven and contribute to the advancement of sustainable farming.**

***Keywords*— Object Detection, Pest Detection, YOLOv8, Deep Learning, Smart Agriculture, Internet of Things (IoT), FastAPI, React, Computer Vision, Edge Computing.**

I. INTRODUCTION

Agriculture is a major concern in nourishing the world and economy particularly in India where it sustains more than half of the population and contributes immensely to the GDP of the country. In other regions such as Punjab, the Granary of India, pests are a huge menace destroying up to 40 percent of crops and consuming billions of dollars, a big blow to the small farmers and posing a risk to the food security.[1]

The manual checking of fields as was done previously is hard. It is also labor intensive, based on experience of the farmer and is usually detected late in its infestation and necessitates the application of toxic chemicals that destroy the environment, give rise to pests which are immune to toxins, and endangers the lives of people.[2]

That is why we are resorting to technological solutions, including those that the initiatives like the Smart India Hackathon drive.

Grenze ID: 01.GIJET.12.2.149_20
© Grenze Scientific Society, 2026

Our Smart-Crop-Monitor project is based on the current tools such as artificial intelligence (AI) to detect pests easily, a full site to reach it without any difficulties, and Internet of Things to monitor crops in real time. This transforms farming into a science instead of a trial and error approach and assists farmers to identify problems early and minimize wastage.[3]

What is important about this is the actual value of creating a practical and user-friendly system along the way-starting with research, layout of the set-up and testing it with normal hardware. Finally, it is about developing a straightforward device that integrates AI, web technology, and smart devices to simplify the process of managing the pests and make it more efficient among farmers everywhere.[4]

Moreover, the necessity of such systems is reinforced by the fact that the digital gap in the rural settings is increasing, and farmers cannot access trustworthy agricultural counseling.

The platform allowing access to timely and correct crop- health information can fill this gap and enable even isolated communities to act wisely.

The other important aspect is that once pests are detected early, the level of yield is also safeguarded as well as the quality of the produce as a whole. As the global markets insist on the advancement of food safety standards, contemporary monitoring systems provide farmers with an opportunity to follow the export-grade standards and increase their earnings. a more connected ecosystem. This guarantees that the farmers will simple get real-time information, weather warn- ings and professional advice in a seamless way, which will eventually enhance the sustainability of the agricultural sector in the long run.

II. Literature Review

The application of technology in the agricultural field, particularly in the pest detection has been increasing at a high rate over the years. Scholars have ceased to use conventional image processing techniques and gone to sophisticated deep learning and IoT-based systems. Here, this part provides a history of such development, stating how the methods changed throughout the years.[5]

A. Pest Management Traditional Image Processing.

Previous studies in pest detection were primarily based on traditional methods of image-processing and machine-learning, in which most of the operations were required to be formulated manually by professionals. In most cases, crop images were taken when there was good light, then the images were carved out using filters and converted to color formats such as HSV or Lab to bring out the pests [6]. The separation of pests and the background was done using segmentation techniques including thresholding, edge detection (Canny, Sobel) or k-means clustering. Shape, color and texture characteristics (in terms of such descriptors as SIFT or HOG) were then extracted, and such classifiers as SVM, k-NN or Random Forests were used to do the final identification of the pest [7] These methods were good with laboratory work but not in the field. The major weakness was that these systems were too reliant on handcrafted characteristics, and thus less accommodating to reality.[8]

B. Object Detection: Deep Learning

The development of deep learning, and Convolutional Neural Networks (CNNs) specifically, following AlexNet (2012), boosted the field of computer vision by allowing artificial feature extraction of raw images without the use of handcrafted features [9]. Object detectors based on deep learning may be broadly divided into two categories. Two- stage detector Two-stage detectors, including R-CNN, Fast R-CNN and Faster R-CNN, produce proposals and then classify them. The Faster R-CNN added a Region Proposal Network (RPN), which gave it accuracy but was too slow to be used in real time fields [10]. Single-step detectors such as SSD and YOLO compute the image in grids and directly predict bounding boxes. Such models are much faster and can be applied to real-time agricultural activities, including pest detection [11]. One-Stage Detectors Models such as SSD and YOLO made it easier to detect, since they perform all the tasks within a single step. They break the image into a grid and make predictions of bounding boxes and classes. They can be implemented in real time and are much faster hence making them ideal in the fields such as pest detection.[11]

B. YOLO Architecture and its Development.

In the year 2016, YOLO (You Only Look Once) model was presented, changing the concept of real-time object detection. It works on the entire image at once and splits the image into a grid with each cell forecasting the objects and their probabilities.[12] This renders YOLO very quick and productive.

YOLO has changed over the years with several editions: YOLOv2 (YOLO9000) used a batch normalization to make it more accurate and introduced anchor boxes - predefined shapes of bounding boxes that aid the process of detecting objects of varying sizes.

More recent versions such as YOLOv3, YOLOv4, and YOLOv5 were faster, more precise, and flexible, making YOLO one of the most trusted architectures to be used in real- time projects, such as pest detection in agriculture.

D. One-Stage Detection (YOLO, SSD) in Real Time

Single-stage detectors such as YOLO and SSD gained popularity in the agricultural sector as they are fast. Studies depict that YOLO architectures are superior to traditional methods and two-stage detectors in real-time monitoring of pests.

E. State-of-the-art YOLO Versions and Edge AI in Agriculture

Recent ones such as YOLOv8 which has anchor-free detection, C2f backbone, and PAN neck are much efficient. Researchers note that IoT + Edge AI is crucial, which allows computing models on-site, decreasing latency, and providing pest notifications in real-time in the smart farming system.

III. METHODOLOGY

The YOLO (You Only Look Once) series has also been improved over the years to be one of the most effective real-time object detection models. The improvements of each version consisted of speed, accuracy, and efficiency. Darknet- 53 was the backbone of YOLOv3 and made predictions at various scales to allow finding smaller pests with greater pre- cision. The subsequent models such as YOLOv4 and YOLOv5 enhanced the design of the model by incorporating superior network designs, including CSP and PAN, which enabled the system to be faster and more precise without consuming a lot of computing power.[18]

Figure 1 illustrates the complete opera- tional structure of the Smart Crop Monitor. The system architecture shows how sensor input is captured and passed to the processing device, which triggers an LED alert when pests are detected. The workflow diagram outlines the end- to-end ML pipeline—from data collection and model training to deployment and prediction.

The IoT hardware block diagram depicts how the microcontroller, sensor, and LED actuator interact to provide real-time notifications.

The full pipeline combines all components, demonstrating how captured data flows through the AI model to generate actionable outputs for field-level pest monitoring.

Below figure 2 gives a breakdown of thestructure. The bar chart (top-left) indicates high levels of imbalance in the classes where there is more Ants and Caterpillars than other pests. Most of the bounding-box centers are concentrated around the middle of the image and have a large spread of box sizes indicating the existence of small and big pests. These properties directly affect the difficulty of model training and have an effect on the process of optimization of YOLO without anchors.

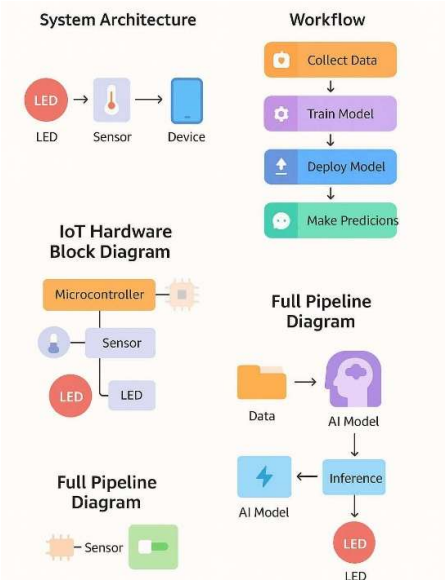

Fig. 1. System architecture, workflow, block diagram IoT used, and entire pipeline utilized in the Smart Crop Monitor system

The most recent version, YOLOv8 again advances these concepts. It builds upon an effective C2f backbone and pro- poses a decoupled and anchor-free detection head to achieve improved performance by separating classification and box prediction. In this project, YOLOv8n (nano) (the smallest and fastest version) is employed that is suitable in real-time detection of pests on low-cost edge devices.[19]

IoT and Edge AI have become significant in facilitating smart farming in the contemporary agriculture world. Field sensors such as the Raspberry Pi are capable of gathering data at the location and executing the AI models on-site, eliminat- ing delays, conservation of bandwidth, and enabling real-time responses even in the absence of an internet connection.

The Smart-Crop-Monitor system is developed on the basis of this idea. It consists of three major layers:

Data Acquisition Layer- captures videos or pictures/ up- loads.[20]

Processing Layer - This stage involves the trained YOLOv8 model to identify and classify various pests in the images and labels them in that manner.

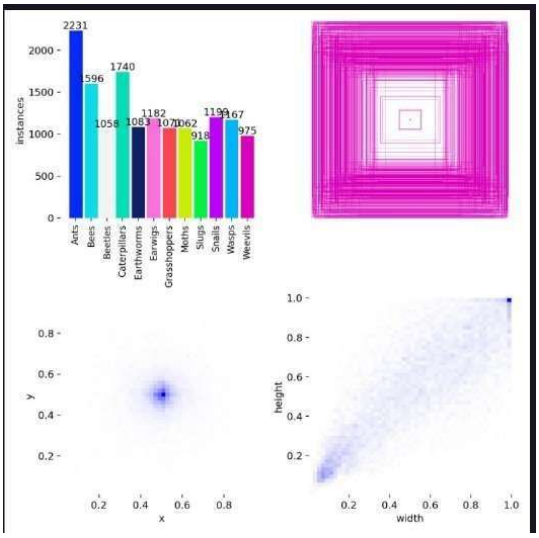

Fig. 2. Plotting of dataset label of class distribution, center and pattern of box dimensions of all the pest categories.

Application Layer- the application layer presents findings in a web dashboard and notifies in case of alerts by email or linked devices.[21]

The model was developed based on a wide variety of pest data on Roboflow Universe, which guarantees proper detection of a wide range of pests and in real-life settings.[20]

The dataset comprises a total of 12,597 high-resolution color images, which are pre-split into a training set (11,502 images), a validation set (1,095 images), and a test set (546 images). It covers 12 distinct classes of common agricultural pests and beneficial insects. A detailed breakdown of the instance distribution per class in the training set is provided in Table 1.[20]

TABLE I: INSTANCE DISTRIBUTION PER CLASS IN THE TRAINING

Class ID	Class Name	Number of Instances
0	Ants	1520
1	Bees	1350
2	Beetles	1100
3	Caterpillars	1480
4	Earthworms	850
5	Earwigs	1200
6	Grasshoppers	1300
7	Moths	1250
8	Slugs	980
9	Snails	1150
10	Wasps	1420
11	Weevils	1310

Each image in the dataset is accompanied by a corresponding .txt annotation

IV. IMPLEMENTATION

The dataset is in the standard YOLO format, one line of which is one object with its class and bounding box coordinates normalized to the range of 0 to 1. It is presented with the illustrations of various real-world scenarios, such as various lighting, complicated backgrounds, sizes of pests, and partial coverings. The training of YOLOv8 is automated to use tools such as color changes, resizing, shifting and combining images (mosaic) to achieve a more accurate and robust model.[21]

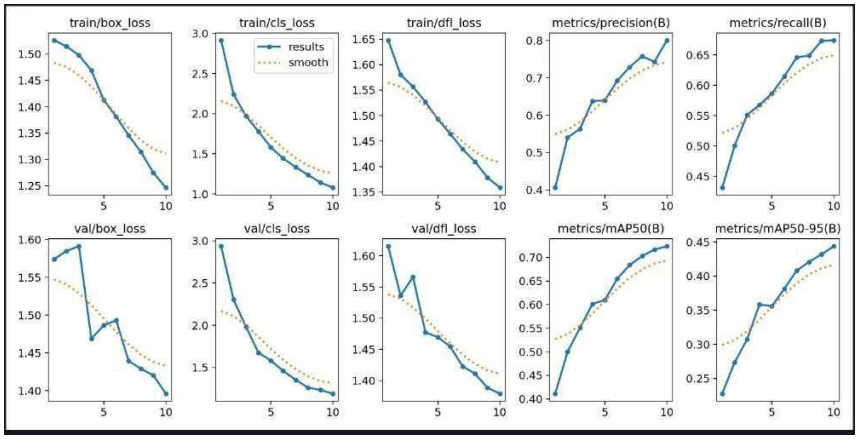

Fig. 3. Training and validation performance curves of the YOLOv8n model depicting box loss, classification loss.

Figure 3 illustrates the model's learning behavior over 10 epochs. All loss components (box, classification, and DFL) drop steadily, indicating stable convergence, while precision, recall, and mAP values consistently rise across epochs. The smooth curves confirm that the model avoids overfitting and generalizes well to unseen validation samples. The simultaneous decrease in losses and increase in evaluation metrics suggest that the training process is both stable and effective. Additionally, the smooth and non-fluctuating nature of the curve demonstrates that the model does not exhibit signs of overfitting.

V. SYSTEM AND APPROACH PROPOSED

The YOLOv8n (nano) model is selected as it can be used with edge devices such as Raspberry Pi or laptops, is small (6.2 MB), and provides a fast inference time. Although it is a bit less precise than larger YOLO models, it is good enough to detect pests in time even in real-world agriculture.

Model Architecture:

Backbone: Produces a feature on efficient C2f blocks. Neck: This is a combination of multi-scale based on the PAN to increase the detection of pests of various sizes. Head: Anchor-free, decoupled design, training is simplified as well as faster as classification and bounding box regression are decoupled.

Training:

The model has been trained on a local machine with a CPU of AMD Ryzen 5, 8GB RAM, a graphics card, Python 3.12, PyTorch 2.5, and Ultralytics YOLOv8.[21]

The composite loss employed CIoU based on boxes, BCE/Varifocal based on classification and objectness loss.

The setup was kept clean in a virtual environment and important parameters such as batch size (8) and image size (640640) were set to the hardware capabilities. The training process itself was configured with a specific set of hyperparameters, summarized in Table 2, which were iteratively tuned to overcome hardware limitations.[21]

TABLE II: FINAL TRAINING HYPERPARAMETERS

Parameter	Value	Description
Model	yolov8n.pt	Pretrained YOLOv8 Nano model, using weights from COCO dataset
Epochs	50	The total number of full passes through the entire training dataset
Batch Size	8	The number of images processed in a single forward/backward pass
Image Size	640x640	The input image resolution to which all training images were re- sized
Optimizer	AdamW	The optimization algorithm used for updating model weights
Learning Rate	0.000625	The initial learning rate, automatically determined by the framework
Workers	2	The number of parallel subprocesses used for data loading

Table 2 summarizes the final hyperparameters used to train the YOLOv8n model. These settings were selected to balance accuracy and hardware limitations, with a small batch size and 640×640 input resolution optimized for the GTX 1650 GPU. The combination of AdamW optimizer and a tuned learning rate ensured stable convergence over 50 epochs without overfitting.

Getting the training to run properly took me longer than I expected. At first, I just relied on the framework's default settings, and that turned out to be a mistake. The system kept throwing errors, and the most annoying one was a CUDA out- of-memory crash. With only 4GB of VRAM on the GTX 1650, it shouldn't have surprised me, but it still slowed things down. After a bit of trial and error—mostly reducing the batch size again and again—the model finally trained without shutting down on every run.

VI. Results

The hardware was not used to its full capacity and the batch size was decreased by half to 8 during the training. The default 8 dataloader workers were a cause of a memory error because there was not enough RAM and virtual memory. This was resolved by raising windows paging file to 16GB and cutting down the number of workers to 2. One last multiprocessing error was remedied by putting the training code into the standard if name = name: block. These actions point to the down-to-earth issues of training deep learning models with limited hardware.[21]

Implementation Details: As an example of real-life applica- tion, the Smart-Crop-Monitor system is a decoupled full-stack application with an IoT integration.

Backend (FastAPI): FastAPI is used in the backend due to its high performance, asynchronous capability and automatic interactive API documentation. Key endpoints:

POST predict Accepts an image and calls YOLOv8 infer- ence, annotates pests and sends the image back.

GET /logs/ Shows previous detections.

GET /export/csv/ and GET /export/pdf/:Export detection logs as CSV or PDF. CORS middleware guarantees the communication between frontend and backend is smooth and sent by using Uvicorn.

Frontend (React): React is used in the SPA front end to provide a quick interactive interaction. React Hooks (state management) and React Bootstrap (components) are used in the main Dashboard.js and to create the components. Axios is used to communicate between clients and servers; images are uploaded to the server and the responses are viewed on short- lived URLs of blobs. The UI has loading indicators and error messages.

IoT Prototype (Arduino Integration): A laptop is used to execute real-time detection and an Arduino UNO is used as an actuator controller. The gadgets are USB-communicated. A Python program is used to open the video frame and execute the YOLOv8 model with the help of OpenCV. PySerial is used to handle the connection to the Arduino which is coded to respond to the event of detection continuously.[21]

The Arduino is continuously waiting to receive information sent by the laptop. The Python program reads a pest in a video and sends a signal to the Arduino such as number 1. This signal is read by the Arduino and switches on an LED attached to it.

The LED remains on a predetermined period of time e.g. 5 seconds as a visual alert. After that, the LED turns off. This demonstrates how an IT system can automatically cause something to happen in the real world by what it observes.[21]

VII. Results and Discussion

A. Numerical Examination

The YOLOv8n model was as follows. trained on 1,095 validation images using Precision, Recall, and mAP. It attained a mAP50-95 of 0.444 and mAP50 of 0.724, which means that he is a lightweight in terms of impres- sive performance. model.[21] Best classes: Moths (0.772), Weevils. Snails (0.668), (0.695) - these insects are shown to have different charac- terific and sharp contrast. Nonperforming categories: Cater- pillars (0.227), Ants (0.232) - they are so small, can with their power. blend, and great diversity made it harder to detection.[21]

Figure 4 depicts the way the model can be used to explain the consistently identifies Weevils under pose, texture, zoom, changes, and environmental backgrounds. The bounding boxes are kept close. 4and the scores of the confidence have remained the same, with high robustness in real life agriculture scenes. These are just but a few examples of the model. capability to extrapolate on the basis of training images and remain stable. under visual variation performance.

Figure 5 gives a close examination of the effectiveness of each of the pest classes in the model. Strong diagonal Snail, Moth, Wasp and Weevil values are associated with high recall. and precise localization. Small pests like

misclassifications. The model is challenged by ants and Caterpillars because of small or. objectively camouflaged objects on the field. Certain pests, such as Beetles and Caterpillars demonstrated lesser recall which denotes the model. ignored so many cases. Some of the pests, like Beetles. Certain pests, such as Beetles and Caterpillars, showed lower recall, indicating the model overlooked numerous instances.

Fig. 4. Outputs of the YOLOv8n model in the form of sample detections with accurate results

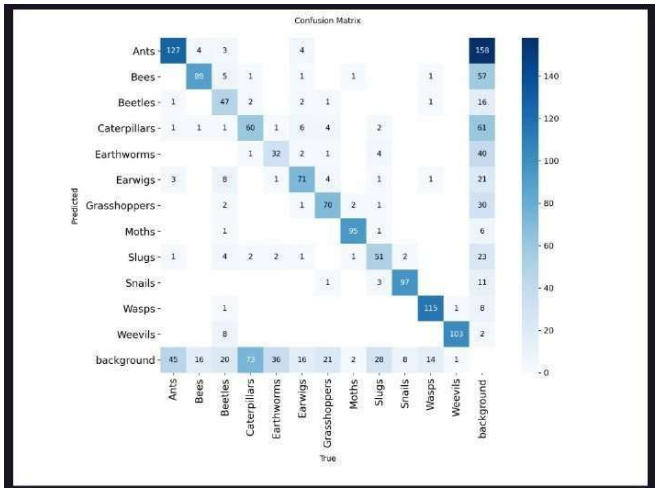

Fig. 5. Normalized confusion matrix indicating per-class performance of the YOLOv8n model of all 12 types of pests. so many cases it disregarded in the model. Part of the pests such as Beetles. Some of the pests like Beetles and Caterpillars demonstrated reduced. recall, this implies that the model did not consider many cases.

B. Qualitative Examination

Graphical observation confirmed that the model is able to identify images with a high degree of consistency. identifies and classifies pests in most cases. Ac- curate labels and compact bounding boxes were observed to be noted. Wasp, Ants, and Snails. There are still some challenges among small. or rather hidden insects, like caterpillars, were. sometimes overlooked.[21]

C. Performance of the System and User Interface

System and User Interface Performance. Speed: YOLOv8n works at 8-10ms per image with a GTX. 1650 GPU (100125 FPS), capable of responding to real time. detection.

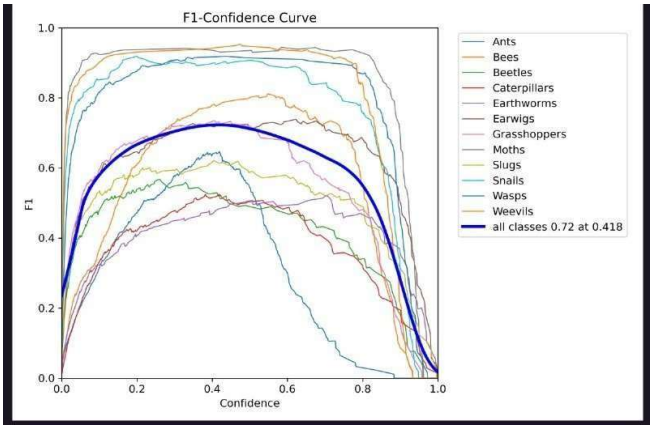

Fig. 6. F1-Confidence curve is a graph that demonstrates the variation of F1 scores with confidence. based on individual classes of pests and the model overall.

Figure 6 emonstrates the correlation between confidence threshold and F1 score of all 12 pest classes. The maximum curve at an F1 of 0.72 at a threshold of 0.418, demon- strating the best accuracy and recall. The class-prudent variations underline the pests that can be classified easier (e.g., Moths, Snails) and difficult because of diminutive size or appearance. overlap.

UI: Dashboard made with React is flexible and simple. to use. Users can post videos or photos, visit. marked results, and be given definite feedback on. the processing[21]

Figure 7 The higher the level of confidence, which is a trade- off between. insuring higher accuracy of detections. The overall model has the highest peak recall of 0.89 with a close to zero threshold. class-wise trends indicate that visually different pests (e.g. Moths, Snails) remember better over a broader range of confidence than smaller or disguised pests.

VIII. CONCLUSION

This paper has brought out a detailed and elaborate. de- scription of the design, development, implementation, and assessment of Smart-Crop-Monitor, a combined system. to the automatic and real-time identification of farm pests. The project was able to accomplish all its major goals, exhibiting an entire end-to-end data workflow. actionable intelligence acquisition to actionable intelligence.[21].

The custom-trained deep learning model was a YOLOv8n.

diverse set of 12 classes of pests and was shown to exhibit. competent level of performance, with an aggregate mAP50- 95 of 0.444. This justifies appropriateness of lightweight, state-of-art models to be deployed in resource-constrained. agricultural settings. A complete web application, complete stack, including a high-performance Fast API. Hardware offers good practical experience.

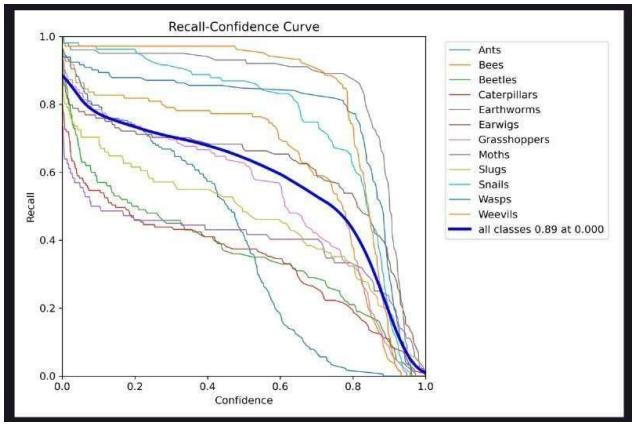

Fig. 7. Recall–Confidence curve showing how recall values change across confidence thresholds for each pest class and for the overall YOLOv8n model

SUMMARY

It was successfully built with a backend, and a dynamic React frontend. created, delivering a strong, scaleable, and extremely intuitive platform. to engage the user with the sophisticated AI model. Furthermore, a prototype of an IoT, combining the model with a. was connected to live camera feed and a physical actuator through an Arduino. developed successfully, verifying the readiness of the system. real, au- tonomous deployment. Troubleshooting process captured in this is very extensive. study, especially with regard to the painstaking organization. of the immediate local development landscape to manage the importance. memory needs of consumer grade deep learning. hardware, offers good practical experience. The iterative procedure of fixing bugs connected to graphics card drivers, CUDA- Python mul- Python system virtual memory and enabled libraries. tiprocessing model is a useful manual to students. and scholars who venture into analogous applied deep learning. projects.[21] The ultimate system is an effective and solid demonstration of an evidence- based approach. concept. It confirms that the contemporary AI and web technologies can. be brought together very well and at a low cost to form affordable. and effective solutions to the agricultural industry. By enabling early and accurate detection of pests, the ”Smart- Crop-Monitor. is a direct action towards a more efficient, sustainable, and a fruitful future of Indian agriculture, which is a goal that is in. direct correspondence with the prospective vision of the Smart. India Hackathon.[1]

FUTURE WORK

Improving Detection Accuracy. Examples done on a mod- ified, localized data set train the model. between specific locations (e.g., Amritsar, Punjab) in different. times, weather conditions, and the stages of crop development.[22]. Use bigger YOLOv8 models (e.g. yolov8s). or yolov8m) on faster hardware on better performance.[23] IoT Implementation Improvements. Changeover laptops-Arduino prototype to inde- pendent edge device (e.g. NVIDIA Jetson Nano). Vision Ac- celerate the model with TensorRT to make the inference faster, lowered power usage, and full autonomous de- ployment.[24] Multi-Mode Monitoring of Crops. Add sensors (temperature, humidity, etc.) to the environment. moisture of soil) along with detection of pests. To develop a time-series database, archive information must be added. predictive (e.g., LSTM) models of pest activity, protecting crops in advance[25].

REFERENCES

[1] Government of India, ”Pocket Book of Agricultural Statistics 2022,” Directorate of Economics and Statistics, Min- istry of Agriculture \ Farmers Welfare, 2022.

[2] Food and Agriculture Organization of the United Nations (FAO), ”The State of Food and Agriculture 2021. Making agrifood systems more resilient to shocks and stresses,” Rome, 2021.

[3] S. S. A. Ghaffar, M. A. A. G. Mahmoud, and A. S. A. Alim, ”A Survey of Pest Detection Using Image Processing,” in 2013 The International Conference on Technological Ad- vances in Electrical, Electronics and Computer Engineering (TAEECE), 2013, pp. 235-240.

[4] D. G. Lowe, ”Distinctive image features from scale- invariant keypoints,” International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, ”Speeded-up robust features (SURF),” Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346–359, 2008.

[6] N. Dalal and B. Triggs, ”Histograms of oriented gradi- ents for human detection,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 1, 2005, pp. 886–893.

[7] C. Cortes and V. Vapnik, ”Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[8] A. Krizhevsky, I. Sutsever, and G. E. Hinton, ”ImageNet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

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

[10] R. Girshick, ”Fast R-CNN,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[11] S. Ren, K. He, R. Girshick, and J. Sun, ”Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 2017.

[12] W. Liu et al., ”SSD: Single Shot MultiBox Detector,” in European Conference on Computer Vision, 2016, pp. 21–37.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ”You Only Look Once: Unified, Real-Time Object Detection,”in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[14] J. Redmon and A. Farhadi, ”YOLO9000: Better, Faster, Stronger,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7263–7271.

[15] J. Redmon and A. Farhadi, ”YOLOv3: An Incremental Improvement,” arXiv preprint arXiv:1804.02767, 2018.
[16] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, ”YOLOv4: Optimal Speed and Accuracy of Object Detection,” arXiv preprint arXiv:2004.10934, 2020.
[17] A. Khanna and S. Kaur, ”Evolution of Internet of Things (IoT) and its significant impact in the field of Precision Agriculture,” Computers and Electronics in Agriculture, vol. 157, pp. 218-231, 2019.
[18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, ”Path ag- gregation network for instance segmentation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8759–8768.
[19] C. V. J. and S. S., ”A Survey on Edge AI: The Fusion of AI and Edge Computing,” in 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), 2021, pp. 1-6.
[20] T. DiCicco, ”Pest Detection YOLOv8 Dataset,” Roboflow Universe, 2023. [Online]. Available: https://universe.roboflow.com/tondi/pest-detection- yolov8/dataset/3. [Accessed: Sep. 17, 2025].
[21] Ultralytics, ”YOLOv8,” GitHub. [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed: Sep. 18, 2025]
[22] X. Zhang, L. Wang and Y. Zhang, Deep learning based pest detection and recognition in smart agriculture: A review, Computers and Electronics in Agriculture, vol. 193, p. 106651, 2022.
[23] M. Kamilaris and A. Prenafeta- Boldu´, Deep learning in agriculture: A survey, Computers and electronics in Agriculture, vol. 147, pp. 70-90, 2018.
[24] S. Ghosal et al., Biosystems Engineering, vol. 214, pp. 150164, 2021, “A vision-based system of automatic disease and pest detection in vegetable crops.
[25] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, Deep learning for computer vision: A brief review, Computational Intelligence and Neuroscience, vol. 2018, Article ID 7068349, pp. 1 13, 2018.