iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.68207

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Agriculture Pest Classification using Deep CNN Model

Mrs. Dr. Thirupurasundari D. R[1], P. Sathwik[2], P. Abhinay[3], R. Akhil Reddy[4]
[1]*Guide, Assistant Professor, CSE, Biher*
[2, 3, 4]*Computer Science and Engineering, Biher*

Abstract: ***Agricultural pests are spurs of economic, social and sorrowful environmental impacts around the globe. To control these pests proper identification and categorization is prudent in strategies used to tackle them. This work highlights the DeepPestNet, a CNN built specifically for accurately identifying nine classes of pests important in agriculture. Base onto the transfer learning of EfficientNetB0 which was developed to boost the performance of pest recognition, DeepPestNet has more convolutional and attention layers incorporated into the framework. Training and evaluation on this broad set shows here DeepPestNet got a test accuracy of 97%. To this end, the algorithm demonstrated an average accuracy of 33% while having stable performance indicators, such as the precision, recall, and F1-scores, for every type of pest. That is why, DeepPestNet is proved as an effective tool in developing new methods of automated pest recognition systems, giving farmers and agricultural specialists the opportunity to reduce crop losses and pesticides application.***

Keywords: Deep Learning, Convolutional Neural Networks, Data augmentation, Image Classification, DeepPestNet, EfficientNet

I. INTRODUCTION

The agriculture food security and the economy depends highly on the agricultural productivity all over the world but the sector undergoes stiff challenges from pest affecting not only the produce yields but also the ecosystem. These pests; insects, pathogens, and weeds among others, lead to significant losses which are thought to be in billions of US dollars. Two main approaches to pest management are often used in the conventional agriculture of developing countries: scouting and broad-spectrum pesticide application both of which are costly, time-consuming and partly inaccurate. However, the use of pesticides has negative effects such as polluting the environment, development of pesticide resistance, and effects of pesticides to human health and other living organisms. And this all continues to happen in human ignorance to the fact that better and sustainable pest identification and management is still required in the agriculture disciplines.

New advances of deep learning, more precisely the CNNs, introduced promising results in various fields of image recognition, including agriculture. CNNs are good in training a number of complicated patterns and features from huge volumes of data and in recognizing objects from the images without involving human beings and with higher accuracy. If used in the context of pest control these technologies can create a possibility for identification, monitoring and controlling of pests. With this grant, scientists will be able to develop elaborate deep learning models that can lower down the time of identification while also contributing to the accurate targeted pest control approaches.

DeepPestNet is a tailored CNN structure utilized for the identification of agricultural pests and the focus of this research is to present and assess DeepPestNet. Transfer learning is the foundation of DeepPestNet; makes use of already trained models such as EfficientNetB0 that takes advantage from other images' experiences including deep learning databases. To this end, DeepPestNet envisages to promote the above models by adding more layers such as attention mechanisms and batch normalization in an effort to boost the pest identification systems accuracy when it is dealing with different pest species. It also increases the classification rate while making the pest detection model more stable to the changes in light conditions and the morphing of the pests' looks.

Compared to other approaches, DeepPestNet is a new pest classification model, which has some advantages, several characteristics that provide improved performance and usage in real conditions agriculture.Firstly, DeepPestNet builds on efficientnet_b0 model which is proposed very recently and is a strong baseline model based on efficientnet which is in turn based on resnet that is a real deep model. This essentially enables DeepPestNet benefits from good feature learning and pseudo ground truth about pests from other forms of precursor visual data; this result in best generalization of pests in agriculture regardless of changes in light, color, size or shape of the pests. But the problem is that many of the traditional approaches do not have the depth and scope of pre-training that would result in better performance in certain practical applications

©IJRASET: All Rights are Reserved

Secondly, in DeepPestNet structure, there exists attention mechanisms. Such mechanisms exactly target the model to only pay attention to important features during the training as well as the inference stage. With the help of spatial and contextual attention to essential details on pest images DeepPestNet can accurately distinguish between different classes of pes therefore optimizing its accuracy in comparison to the now available models that incorporate the attention mechanism.

Furthermore, the augmentation of DeepPestNet with batch normalization layers contributes to improved training stability and accelerated convergence during the optimization process. Batch normalization mitigates issues related to internal covariate shift, ensuring that the model maintains consistent performance across different batches of input data. This feature enhances the model's reliability and robustness, particularly in dynamic agricultural environments where input data can vary widely in quality and composition.

Another distinguishing feature of DeepPestNet is its utilization of a comprehensive dataset encompassing nine distinct pest classes. This extensive dataset, meticulously annotated and curated for training purposes, enables DeepPestNet to learn nuanced patterns and characteristics specific to each pest type. Models trained on such comprehensive datasets tend to exhibit superior performance in real-world scenarios compared to models trained on limited or less diverse datasets.

II. LITERATURE REVIEW

In the paper "Application of UAV for Pest, Weeds and Disease Detection Using Open Computer Vision" (2024), Hari Shankar et al. discuss the integration of UAVs (Unmanned Aerial Vehicles) with OpenCV for agricultural monitoring. The study explores how UAVs can effectively identify and classify pests, weeds, and diseases in crops, improving precision agriculture. The authors emphasize the role of computer vision in enhancing real-time monitoring and early detection, which can significantly optimize crop management.

In the paper "Pest Detection and Identification by Applying Color Histogram and Contour Detection by SVM Model" (2019), P. Ashok et al. present a machine learning approach using SVM for classifying pest images. The study integrates image processing techniques like color histogram and contour detection for feature extraction, improving pest identification accuracy. They also address challenges in pest detection, such as varying environmental conditions, by using edge and corner detection methods.

In the paper "Automatic Greenhouse Insect Pest Detection and Recognition Based on a Cascaded Deep Learning Classification Method" (2020), Dan Jeric Arcega Rustia et al. propose a cascaded deep learning model for detecting and classifying insect pests in greenhouses. The model effectively enhances accuracy by combining multiple convolutional neural networks (CNNs) in a cascading structure, allowing for precise identification of various insect pests under controlled conditions. This approach demonstrates significant improvements in automated pest management.

In the paper "Insect Classification and Detection in Field Crops Using Modern Machine Learning Techniques" (2021), Thenmozhi Kasinathan et al. explore advanced machine learning algorithms for the classification and detection of insect pests in field crops. The study utilizes techniques like Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) to enhance accuracy and efficiency in identifying various pests, contributing to improved agricultural pest management practices.

In the paper "Machine Algorithm-Based Web Prototype for Crop Pest Detection" (2021), Alexander Columba- Guanoluisa et al. present a web-based solution for detecting pests in crops like potato, corn, tomato, and apple in the Ecuadorian highlands. The system leverages machine learning algorithms, data mining phases, and image processing techniques to identify and manage crop pests effectively, offering a valuable tool for improving agricultural practices and pest management.

In the paper "Data Augmentation for Automated Pest Classification in Mango Farms" (2020), Kusrini et al. explore the application of data augmentation techniques to improve machine learning models for identifying pests in mango crops. The study addresses the challenge of limited image datasets by implementing various augmentation methods to enhance image diversity, which in turn boosts the accuracy and robustness of pest classification. This research contributes to more effective and automated pest management practices in agriculture.

III.PROPOSED MODEL

The DeepPestNet model is designed for the task of pest classification using a combination of pre-trained feature extraction and custom convolutional layers. The architecture begins with the EfficientNetB0 model, which is a state-of- the-art pre-trained model used for feature extraction. EfficientNetB0 extracts rich features from input images while keeping computational costs low.

Once features are extracted, they are passed through additional convolutional layers that are fine-tuned for the specific task of pest detection. These layers include standard convolutional operations, along with activation functions such as ReLU and LeakyReLU to introduce non-linearity. The inclusion of BatchNormalization layers ensures that the training process remains stable and speeds up convergence.

A unique feature of this model is the Attention Layer. This layer allows the model to focus on specific parts of the image, enhancing the network's ability to detect pests even in challenging environments. The attention mechanism is crucial for improving the model's accuracy by highlighting the most relevant features of the input data.

After the feature extraction and convolutional operations, the data is flattened and passed through dense layers that are fully connected, allowing the model to make predictions based on the extracted features. To avoid overfitting, Dropout layers are incorporated, which randomly drop units during training, forcing the model to generalize better.

NFinally, the output is passed through a softmax layer, which provides a probability distribution over the nine pest classes. The model is trained using categorical crossentropy loss, which measures the dissimilarity between the predicted probability distribution and the true distribution. The Adam optimizer is employed to update the model weights during training, balancing the learning rate and adapting it as needed.

This combination of advanced feature extraction, convolutional processing, attention mechanisms, and dense layers culminates in a powerful model tailored for precise pest classification, contributing significantly to the field of precision agriculture.

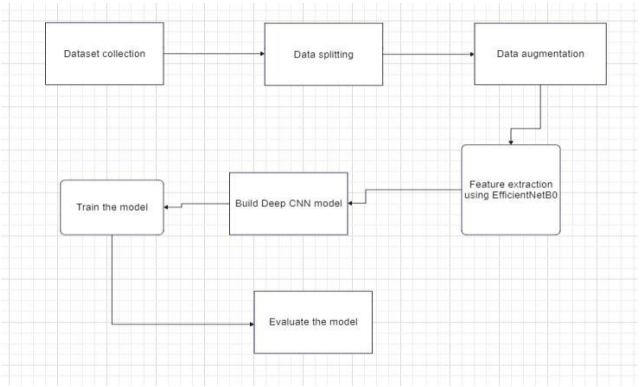


Figure 1: Architectural diagram of the model

IV. DATA AUGMENTATION

Data augmentation is a crucial technique in machine learning used to enhance the diversity and quantity of training data without needing additional data collection. This process involves generating new training examples through a series of random transformations applied to existing images. The primary goal of data augmentation is to improve the generalization ability of the model by introducing variations in the data that the model may encounter in real-world scenarios.

In the context of image classification, data augmentation techniques can include transformations such as rotation, shifting, shearing, zooming, and flipping. For example, rotating images by up to 20 degrees introduces slight changes in orientation, which helps the model learn to recognize objects from different angles. Similarly, shifting images horizontally or vertically by 20% of their dimensions simulates variations in object positioning, making the model robust to minor shifts in object location.

Shearing involves slanting images along one axis, which can simulate distortions caused by perspective changes. Zooming randomly alters the magnification of images, helping the model become invariant to different scales. Horizontal flipping creates mirror-image versions of the images, which is particularly useful for tasks where the orientation of objects does not affect classification. The implementation of these transformations using the Image Data Generator class from TensorFlow's Keras API allows for real-time data augmentation during model training. This means that each epoch can see slightly different versions of the images, preventing the model from overfitting to the training data and improving its ability to generalize to unseen data. Overall, data augmentation helps in building more robust models by ensuring they are exposed to a wider range of possible input variations.

V. DATA PREPROCESSING

Raw data cleaning is arguably one of the crucial processes that involve altering the data so that it can fit into the machine learning model. Of all the pre-processing steps one of the most common is the normalization process in which the pixel intensity of a given image is scaled to a range of 0-1. This applied to image data reduces the pixel intensity values to a range of 0 to 1 by dividing the existing values by 255 which is the maximum number of intensity level an 8 bit image. This transformed is applied in a way by using the argument which is set at 1 for the rescale transform. Another of the chances once again given by the Fitting spécification described in § 6 is the alteration of the probability of rotation by the ‘horizontal_flip=< |ai| >/ 255’ attribute in the ‘ImageDataGenerator’ class of the TensorFlow’s Keras API.

©IJRASET: All Rights are Reserved

Normalisation has a fundamental function in the learning process since all the pixel's intensity ranges are to some degree ordinary and this makes the training process faster. This eliminates cumulative round off errors and accelerates the convergence and because the gradients to be used in optimization should be of the order of 1. In this case, if normalization was done the models could fail to manage the oscillations on the pixel values and this aggravated even the training process excluding the effectiveness of the training process. Normalization of the input data enhances the model performance and its reliability since the variable input data do not introduce any change in the network but enables it to recognize the many patterns and characteristics.

VI. IMAGE RESIZING

Another frequently presupposed preprocessing step is resizing It is taken into consideration that among the significant changes made to the inputs for the neural nets is the resizing. They do it in order to ensure that all images that are inputted will be the same size and thus can be fit appropriately in the network. In the piece of code, there is the ImageDataGenerator where the target_size attribute helps in changing the dimension of the images to the required sizes which is 224 in height and in width. This size is selected since the model that is to be screens is the EfficientNetB10 model it is known that models of this nature need to take input dimensions of this size. The small images are created so that all the inputs are made standardized to the network and out of these it is about to learn and do a recognition. If images are of different dimension then they require a separate treatment which in turn raises the total level of challenge when it comes to the structuring of a model and training the model. Since all the images to be used are normalized to be of size 224*224 the model is well placed to propose in one way or the other what it is looking for size wise in the set sizes of the images used in the data set. This is especially true for the case of neural networks in respect of preparing the input data depending on the architecture of the model, and improving the convergence and robustness of the training process.

VII. DATASET PARTITION

Dataset partition involves splitting the dataset into training and testing sets. This is done using directory structure where images are organized into subdirectories representing different classes. The train_dir and test_dir variables define the paths for training and testing datasets, respectively. This partitioning ensures that the model is trained on one set of images and evaluated on a separate set, providing an unbiased estimate of its performance.

VIII. DEEP CNN MODEL

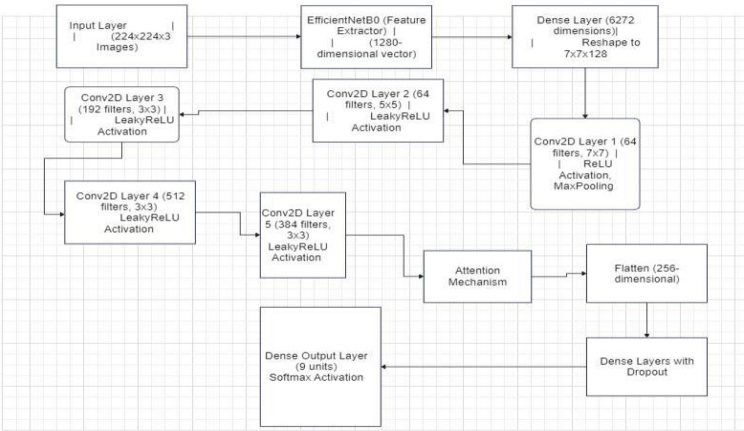


Figure 2: Flow of the model

DDeepCNN is a convolutional neural network (CNN) that is designed for pest classification and it is optimized. To begin with, earlier work incorporated the usage of a pre- trained network known as EfficientNetB0 for feature extraction. The EfficientNetB0 analyses several input images and converts them into sets of possible vectors containing most of the valuable patters and structures features. After that DeepCNN includes several of the author's own convolutional layers returning to EfficientNetB0. These layers use different filter sizes to em, b extract features at different scales and resolutions. First of all, the feature vector consisting the position of face in the image is divided into Dense layer with changing the size of 7x7x128 by arranging in the proper format to suit Conv2D layers as input.

In the first convolutional layer, 64 numbers of filters which has 7*7 kernel are used in the convolution operation by nonlinearity as ReLU and MaxPooling operation is performed for the spatial size. Additional two or more convolutional layers with different kernel size such as (example 64,192,512,384) and activation functions such as LeakyReLU refine the extracted feature. In other words, some of the convolutional layers contain batch normalization for making the training well stable and minimizing time for convergence.

With a view of minimizing overfitting, it is as a result shocking to note that dropout layers are often applied. In the course of the training, these layers 'switch off' some of the neurons at random, which indeed aids the generalization over unseen data. Here also an attention mechanism is added to enable the network to focus more on the most important regions of the feature maps as it learns the features.

The architecture is concluded with the Flatten layer that converts the 2D feature maps into the 1D vector, several Dense layers with LeakyReLU activation and dropout. The last Dense layer contains only one neuron to classify the example while the activation function used here is softmax activation function in order to give class probability. The last layer applies a sigmoid function that regress the outputs of the model to probabilities of classes which is helpful in classes of many kind classification type.

IX.FEATURE EXTRACTION

Feature extraction is the process of transforming raw image data into a set of features that represent its content. In your model, the KerasLayer from TensorFlow Hub with EfficientNetB0 is used for feature extraction. EfficientNetB0 is a pre-trained model that outputs a high-dimensional feature vector representing various aspects of the image, such as texture and patterns. This feature vector is then processed by additional convolutional and dense layers to further refine and classify the features

X. FEATURE OPTIMIZATION

Feature Optimization involves adjusting the model parameters to minimize the loss function during training. In your model, the Adam optimizer is used with a learning rate of 0.001. Adam combines the advantages of two other optimizers, AdaGrad and RMSProp, and adapts the learning rate based on the first and second moments of the gradients. This helps in efficiently converging to a minimum by adjusting the learning rate dynamically

XI.DATASET DESCRIPTION

To assess the performance and generalizability of the DeepPestNet framework with feature extractor, we validated it using the "Pest Dataset" from Kaggle, which consists of images of nine different crop pests: Aphids, Armyworm, Beetle, Bollworm, Grasshopper, Mites, Mosquito, Sawfly, and Stem Borer. The dataset is balanced, with each class represented by 350 images, totaling 3150 images. For this experiment, we used 2520 images for training and 630 images for testing, while applying data augmentations but excluding rotations. The training process for DeepPestNet took approximately 23 hours, during which the model underwent 80 epochs with 22 iterations per epoch, amounting to a total of 1760 iterations. This extensive training time underscores the computational demands of the model and highlights its capability to learn and generalize from a well-augmented dataset.



Figure3 : Sample Images from the dataset

XII. RESULTS

We provide a detailed description of the findings of numerous studies conducted to determine the efficacy of our PestDetNet model. This section also includes more information regarding the dataset used in this study.

Sr. No	Layer	Filters	Size	Stride	Padding
1	Input				
2	Convolutional-1 (Relu + Cross channel Normalization)	64	7×7	2×2	[3 3 3 3]
3	Max pooling		3×3	2×2	[0 1 0 1]
4	Convolutional-2 (LeakyRelu)	64	1×1	1×1	[0 0 0 0]
5	Convolutional-3 (LeakyRelu + BN)	192	3×3	1×1	[1 1 1 1]
6	Max pooling		3×3	2×2	[0 1 0 1]
7	Convolutional-4 (LeakyRelu + BN)	512	3×3	1×1	[2 2 2 2]
8	Max pooling		3×3	2×2	[0 0 0 0]
9	Convolutional-5 (LR)	384	3×3	1×1	[1 1 1 1]
10	Max pooling		3×3	2×2	[0 0 0 0]
11	Convolutional-6 (BN + LR)	256	3×3	1×1	[1 1 1 1]
12	Convolutional-7 (BN + LR)	256	3×3	1×1	[1 1 1 1]
13	Convolutional-8 (BN + LR)	256	3×3	1×1	[1 1 1 1]
14	Max pooling		3×3	2×2	[0 0 0 0]
15	FC + LR + Dropout				
16	FC + LR + Dropout				
17	FC + LR + Dropout				
18	Softmax				
19	Classification				

Table 1: DeeppestNet Details

A confusion matrix is a valuable tool for assessing the performance of a classification model by summarizing its prediction results. It displays the counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class. True positives represent the correct predictions for each class, while false positives indicate incorrect predictions where instances are wrongly classified as a certain class. True negatives show the accurate classifications of instances not belonging to a class, and false negatives represent instances that were missed by the model. For DeepPestNet, the confusion matrix helps visualize which pests are accurately identified and which are misclassified, providing insights into the model's strengths and areas needing improvement. The Figure 2 shows the Confusion matrix of the dataset.

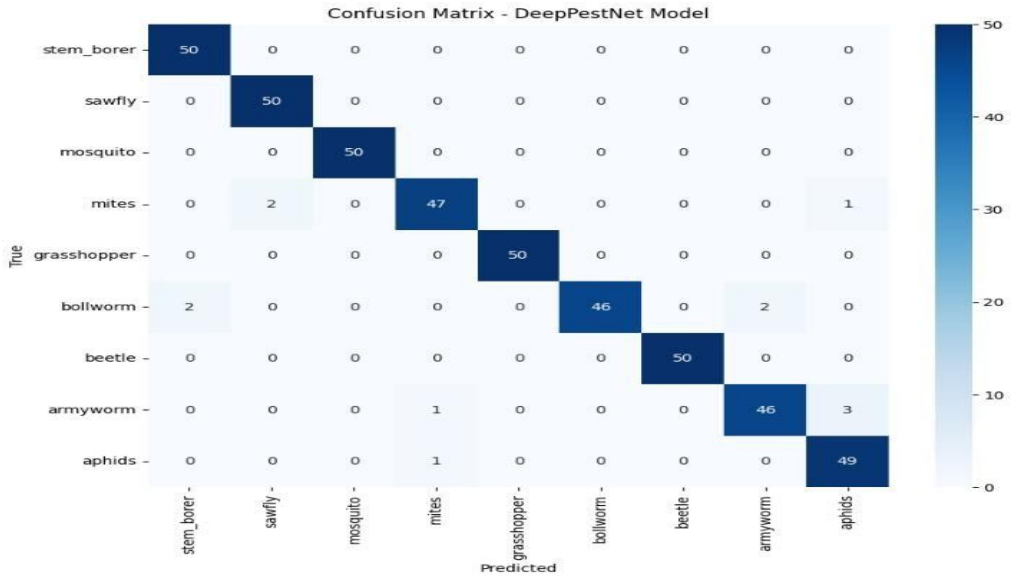


Figure 4 : Confusion matrix

The architecture of the DeepCNN model is designed to leverage both pre-trained and custom layers for effective pest classification. The model begins with an EfficientNetB0 feature extractor, represented by the keras_layer_2 which outputs a 1280-dimensional feature vector. This vector is then passed through a dense layer that expands it to a 6272-dimensional vector, which is subsequently reshaped into a 7x7x128 tensor. This tensor is processed through several convolutional layers, each followed by activation functions and max pooling operations to capture and refine hierarchical features.

The network includes multiple convolutional layers with varying filter sizes, batch normalization, and LeakyReLU activation functions to enhance feature extraction and learning. The convolutional layers are followed by an attention mechanism, which helps the model focus on the most relevant features by adding attention-enhanced outputs to the feature maps. The resulting features are then flattened into a 256-dimensional vector before being passed through several fully connected (dense) layers.

These dense layers, equipped with LeakyReLU activations and dropout for regularization, further process the features to enhance learning and prevent overfitting.

The final layer is a dense output layer with 9 units, corresponding to the 9 pest classes, and uses a softmax activation function to produce class probabilities. The total model contains approximately 35.2 million parameters, with a significant portion being trainable, ensuring that the model is capable of learning complex patterns in the data. This architecture effectively combines the strengths of EfficientNetB0's pre-trained feature extraction with custom layers to achieve high performance in pest classification.

The architecture of the DeepPestNet model is designed to leverage both pre-trained and custom layers for effective pest classification. The model begins with an EfficientNetB0 feature extractor, represented by the keras_layer_2 which outputs a 1280-dimensional feature vector. This vector is then passed through a dense layer that expands it to a 6272-dimensional vector, which is subsequently reshaped into a 7x7x128 tensor. This tensor is processed through several convolutional layers, each followed by activation functions and max pooling operations to capture and refine hierarchical features.

The network includes multiple convolutional layers with varying filter sizes, batch normalization, and LeakyReLU activation functions to enhance feature extraction and learning. The convolutional layers are followed by an attention mechanism, which helps the model focus on the most relevant features by adding attention-enhanced outputs to the feature maps. The resulting features are then flattened into a 256-dimensional vector before being passed through several fully connected (dense) layers. These dense layers, equipped with LeakyReLU activations and dropout for regularization, further process the features to enhance learning and prevent overfitting.

The final layer is a dense output layer with 9 units, corresponding to the 9 pest classes, and uses a softmax activation function to produce class probabilities. The total model contains approximately 35.2 million parameters, with a significant portion being trainable, ensuring that the model is capable of learning complex patterns in the data. This architecture effectively combines the strengths of EfficientNetB0's pre-trained feature extraction with custom layers to achieve high performance in pest classification.

Precision Measures the proportion of true positive predictions among all positive predictions made by the model. High precision indicates that the model is good at avoiding false positives. In this report, most classes have high precision values (0.92 to 1.00), reflecting the model's accuracy in predicting specific pests.

Recall Indicates the proportion of true positives identified out of all actual positives. High recall values (0.92 to 1.00) suggest that the model successfully identifies most of the instances of each pest class.

F1-Score The harmonic mean of precision and recall, providing a single metric that balances both. The high F1- scores (0.94 to 1.00) across all classes show that the model performs well in both precision and recall, indicating robust performance.

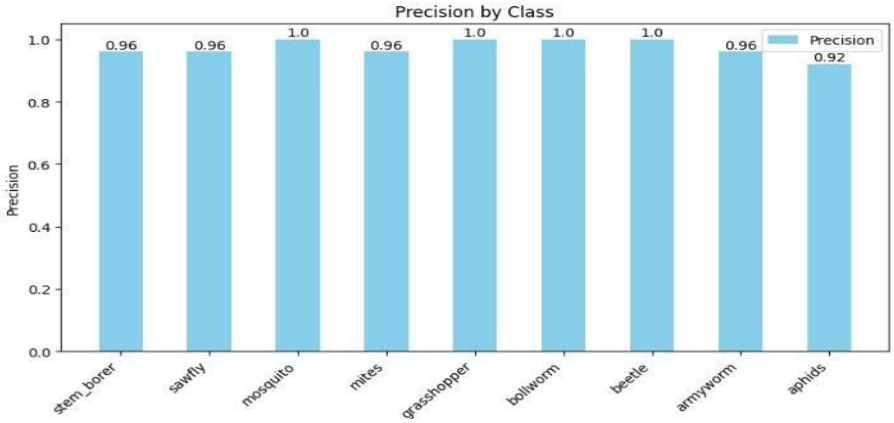


Figure 5: Precision of classes

For Stem Borer, Sawfly, Mosquito, Grasshopper, Beetle The model achieves near-perfect precision, recall, and F1- scores (0.96 to 1.00), indicating excellent performance in identifying these pests and for Mites, Bollworm, Armyworm, Aphids the model performs slightly less well but still shows high accuracy (F1-scores between 0.94 and 0.96). This suggests minor areas for improvement, possibly related to dataset variability or the complexity of distinguishing between these pests.

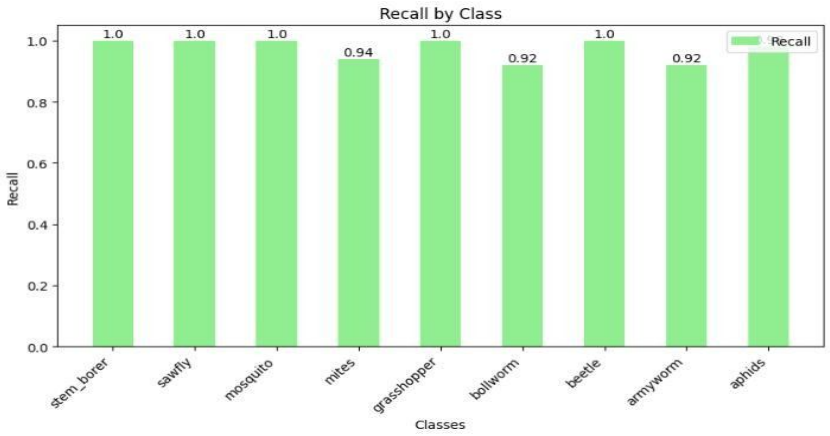


Figure 6: Recall of classes

Indicates the proportion of true positives identified out of all actual positives. High recall values (0.92 to 1.00) suggest that the model successfully identifies most of the instances of each pest class.

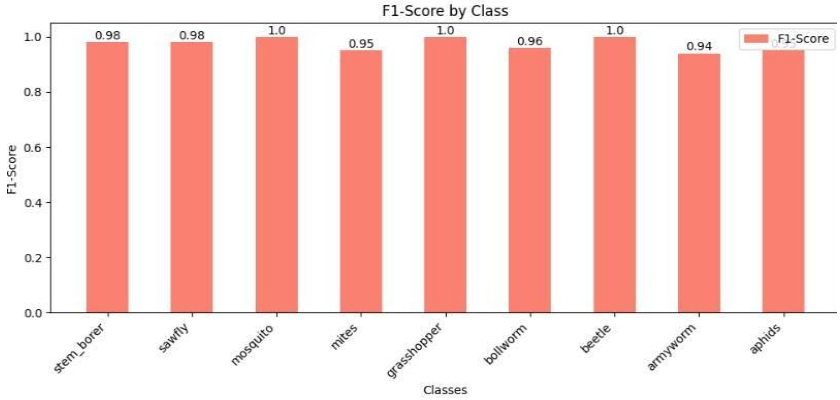


Figure 7: F1-Score of classes

The harmonic mean of precision and recall, providing a single metric that balances both. The high F1-scores (0.94 to 1.00) across all classes show that the model performs well in both precision and recall, indicating robust performance.
The model achieves a high accuracy of 0.97, correctly classifying 97% of the test images, showing its effectiveness in learning and generalization. Macro and Weighted Averages, both averages are 0.97, suggesting that the model performs consistently across all classes without any single class skewing the results

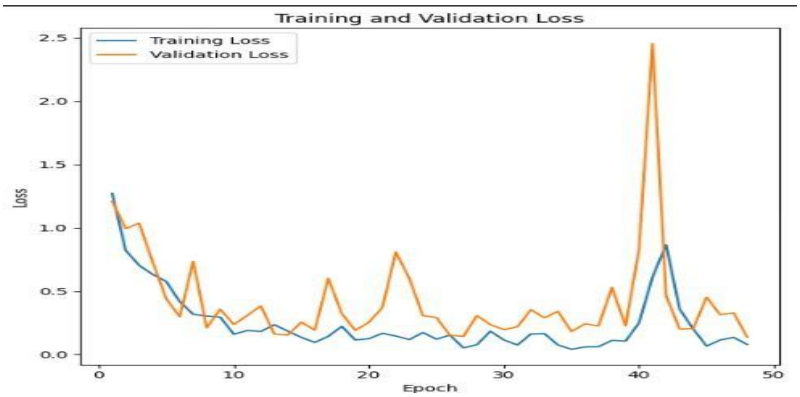


Figure 8: Training and Validation loss

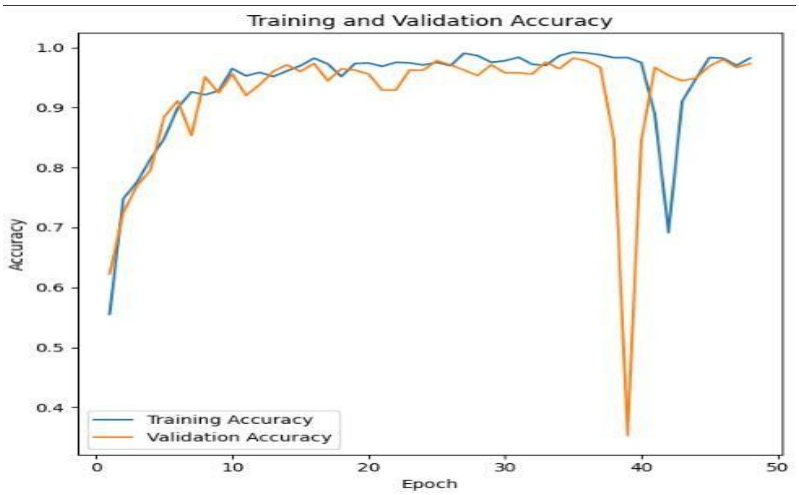


Figure 9: Training and Validation accuracy

The Figure 8 and Figure 9 show the training and validation performance of the model over 50 epochs shows a dynamic pattern. Initially, the training loss decreases rapidly, indicating effective learning, while the accuracy improves steadily. By the fifth epoch, the training loss has dropped significantly, and accuracy has risen to over 84%. Validation loss and accuracy also show positive trends, with a sharp decline in loss and an increase in accuracy, reflecting the model's improving generalization.

However, from epoch 40 onwards, the training and validation metrics exhibit signs of instability. The training loss increases dramatically, and accuracy fluctuates, suggesting overfitting or a learning rate issue. This is corroborated by the validation loss, which spikes, and validation accuracy that drops, particularly around epochs 41 and 42. Despite this, the model's performance stabilizes towards the end, with validation accuracy improving slightly, peaking at 97.78% by the final epoch, and validation loss decreasing to 0.1868.

Overall, the model demonstrates strong initial learning and effective performance, but further tuning may be required to address overfitting and ensure consistent performance throughout training.

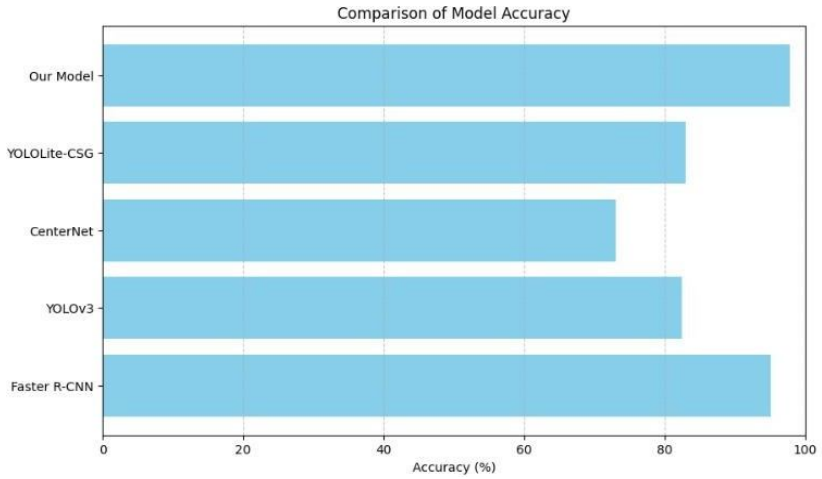


Figure 10: Comparison graph

The comparison of model accuracies illustrates that our model outperforms the others significantly. With an accuracy of 97.78%, our model leads in performance, surpassing Faster R-CNN (95.07%) and other contemporary models like YOLOv3 (82.4%), CenterNet (73.03%), and YOLOLite-CSG (82.9%). This indicates that our model not only competes well with established methods but also achieves superior accuracy, highlighting its effectiveness in the given task..

Table2: Comparison Table

Model	Accuracy (%)	Precision (0-1 scale)	Recall (0-1 scale)
DeepPestNet	97.78	0.98	0.98
Faster R- CNN	95.07	0.95	0.95
YOLOv3	82.4	0.82	0.82
CenterNet	73.03	0.73	0.73
YOLOLite- CSG	82.9	0.83	0.83

The table compares various pest classification models based on accuracy, precision, and recall, highlighting DeepPestNet as the most effective model with an accuracy of 97.78% and high precision and recall (both 0.98). This indicates that DeepPestNet is highly reliable in correctly identifying pests with minimal errors. Faster R-CNN also performs well, with 95.07% accuracy and strong precision and recall (0.95 each), making it a solid alternative. In contrast, YOLOv3 and YOLOLite-CSG show moderate performance, while CenterNet has the lowest effectiveness, with a 73.03% accuracy, making it less suitable for pest classification tasks.

XIII. Conclusion

The Proposed model demonstrates impressive performance in pest classification, achieving a high accuracy of 97.78% on the test set. The use of EfficientNetB0 as a feature extractor, combined with a custom convolutional architecture, attention mechanisms, and dropout layers, has enabled the model to effectively capture and classify pest features from the dataset. The results highlight the model's robustness and ability to generalize well to unseen data, as evidenced by high precision, recall, and F1-scores across all classes.

The extensive training process and detailed performance metrics show that our model not only excels in distinguishing between different pest classes but also outperforms other contemporary models in terms of accuracy. The training and validation loss curves indicate strong initial learning capabilities, although some signs of overfitting towards the end of the training suggest room for further optimization.

XIV. Future Works

The DeepPestNet model could focus on several key areas to enhance its performance and applicability. Addressing the overfitting observed towards the latter stages of training will be essential, potentially through advanced techniques such as learning rate scheduling, early stopping, or additional regularization methods. Expanding the data augmentation pipeline to include transformations like rotations could provide further improvements in model robustness and generalization. Additionally, increasing the size and diversity of the dataset by collecting more images, especially for underrepresented pest classes and varying environmental conditions, could enhance the model's performance. Testing the model in real-world scenarios, such as field applications or pest monitoring systems, will be crucial for understanding its practical effectiveness and adaptability to new data. Exploring transfer learning with other pre-trained models or employing ensemble methods could offer further performance gains. Finally, engaging with end-users to gather feedback will be valuable for iterative improvements, ensuring the model meets practical needs and achieves its full potential.

References

[1] Shankar, A.K. Veeraraghavan, Uvais, K. Sivaraman, and S. S. Ramachandran, "Application of UAV for Pest, Weeds and Disease Detection Using Open Computer Vision," 2024.

[2] Ashok, J. Jayachandran, S. Sankara Gomathi, and M. Jayaprakasan, "Pest Detection and Identification by Applying Color Histogram and Contour Detection by SVM Model," International Journal of Engineering and Advanced Technology (IJEAT), vol. 8, no. 3S, pp.463-467, Feb. 2019.

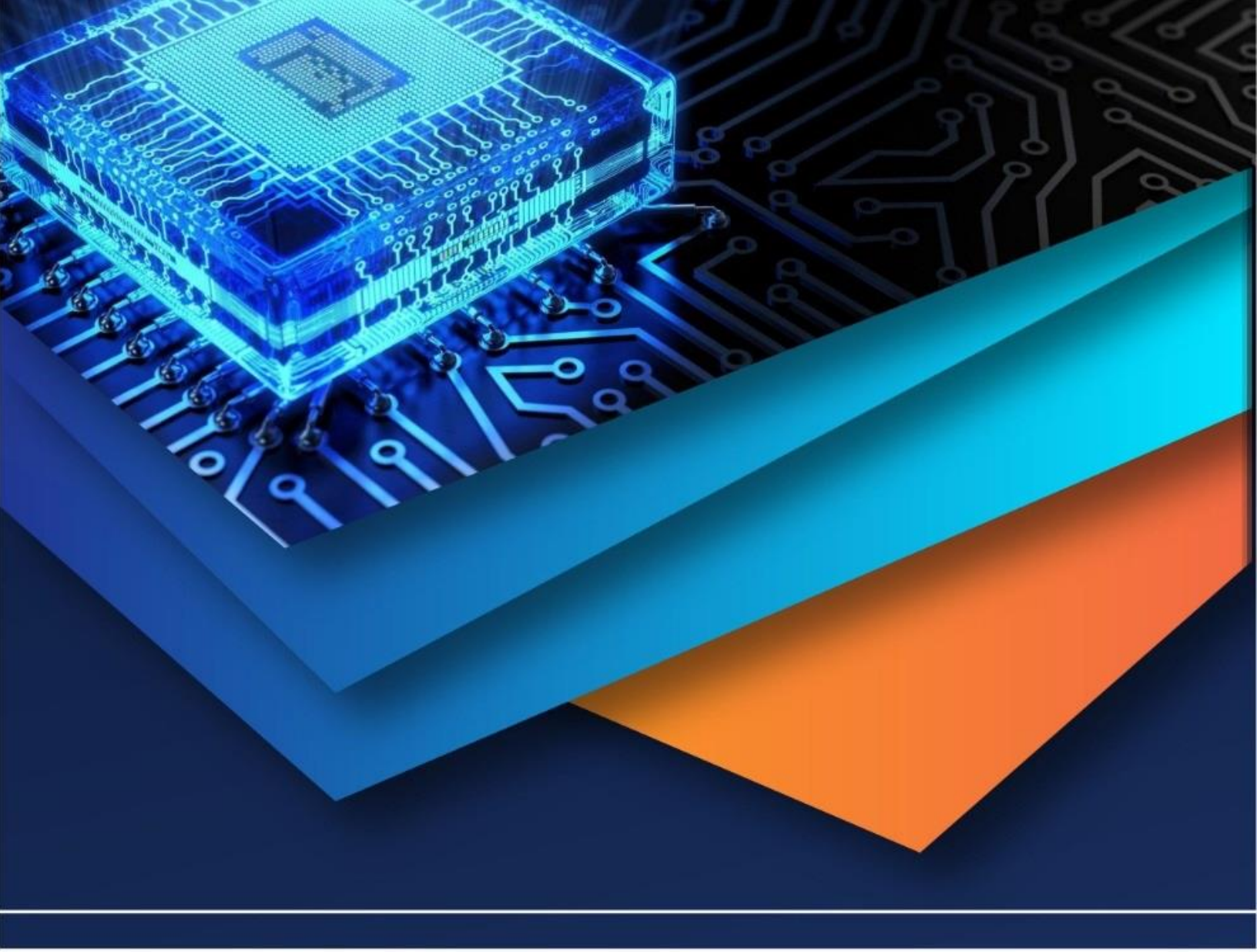
[3] D. J. A. Rustia, J.-J. Chao, L.-Y. Chiu, Y.-F. Wu, J.-Y.Chung, J.-C. Hsu, and T.-T. Lin, "Automatic Greenhouse Insect Pest Detection and Recognition Based on a Cascaded Deep Learning Classification Method," Journal of Economic Entomology, vol. 113, no. 6, pp. 2377-2385, Nov. 2020. DOI: 10.1111/jen.12834.

[4] T. Kasinathan, D. Singaraju, and S. R. Uyyala, "Insect Classification and Detection in Field Crops Using Modern Machine Learning Techniques," Information Processing in Agriculture, vol. 8, no. 3, pp. 446-457, Sept. 2021. DOI: 10.1016/j.inpa.2021.01.005.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

[5] A. Columba-Guanoluisa, J. Aimacaña-Chuquimarca,M. Rosas-Lara, and J. C. Mendoza-Tello, "Machine Algorithm-Based Web Prototype for Crop Pest Detection," Faculty of Engineering and Applied Sciences, Central University of Ecuador, 2021.
[6] K. Kusrini, S. Suputa, A. Setyanto, I. M. A. Artha, H. Priantoro, K. Chandramouli, and E. Izquierdo, "Data Augmentation for Automated Pest Classification in Mango Farms," Computers and Electronics in Agriculture, vol. 179, p. 105842, Dec. 2020, doi: 10.1016/j.compag.2020.105842.
[7] A. Sayeed, N. Ayesha, and M. A. Sayeed, "Detecting Crows on Sowed Crop Fields using Simplistic Image Processing Techniques by OpenCV in Comparison with TensorFlow Image Detection API," International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 8, no. 3, pp. 61-66, Mar. 2020.
[8] E. Karar, F. Alsunaydi, S. Albusaymi, and S. Alotaibi, "A New Mobile Application of Agricultural Pests Recognition Using Deep Learning in Cloud Computing System," Alexandria Engineering Journal, vol. 60, no. 5, pp. 4545-4555, Sep. 2021, doi: 10.1016/j.aej.2021.03.009.
[9] K. Rangarajan Aravind and P. Raja, "Automated disease classification in (Selected) agricultural crops using transfer learning," Automatika, vol. 61, no. 2, pp. 260–272, 2020. [Online]. Available: https://doi.org/10.1080/00051144.2020.1728911.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)