Pest Classification using CNN

Swetha Naga Sai Chirumamilla
Id: 811316952
Kent State University
Kent, Ohio
Mail: schirum4@kent.edu

Tarun Gopi Reddy Vajrala
Id: 811323522
Kent State University
Kent, Ohio
Mail: tvajrala@kent.edu

Anandagani reddy Nallamilli
Id: 811292429
Kent State University
Kent, Ohio
Mail: anallami@kent.edu

Sandhya Kamireddy
Id: 811292364
Kent State University
Kent, Ohio
Mail: skamired@kent.edu

***Abstract*— An innovative method for categorizing pests from picture data is presented in this report. Convolutional Neural Networks (CNN) with transfer learning implemented in the EfficientNetV2-L architecture are used. This research aims to correctly categorize photos into one of twelve different pest groups, which will support agriculture and pest management methods. Our model significantly improves classification efficiency and accuracy by utilizing the powerful feature extraction capabilities of EfficientNetV2-L and the benefits of transfer learning. We prove the efficacy of our approach in accurately identifying a range of pests using thorough evaluation and testing on diverse datasets, hence helping in the development of robust pest detection systems. Our research opens the way to improved ways to manage pests and sustainable farming methods by proving the potential of CNNs and transfer learning in solving real-world issues in agriculture.**

Keywords*— *CNN, Transfer learning

I. INTRODUCTION

The demand for innovative techniques in the agriculture sector has increased recently as a result of pests' adverse effects on crop productivity and quality. Considering their ability to cause large economic losses as well as environmental damage, pests create a danger to global food security. Conventional approaches to pest control frequently depend on chemical treatments and manual inspection, which can be expensive, time-consuming, and harmful to the environment. To help farmers with early identification and focused control efforts, there is an increasing demand for automated and effective pest detection systems.

This project intends to this challenge by constructing a deep learning technique and image-based categorization system for pests. especially, we use Convolutional Neural Networks (CNNs), an effective type of deep learning model known for their outstanding image classification capacity. Particularly, we use transfer learning, a method that allows us to modify pre-trained CNN models to our particular classification task, to improve the performance and efficacy of our model.

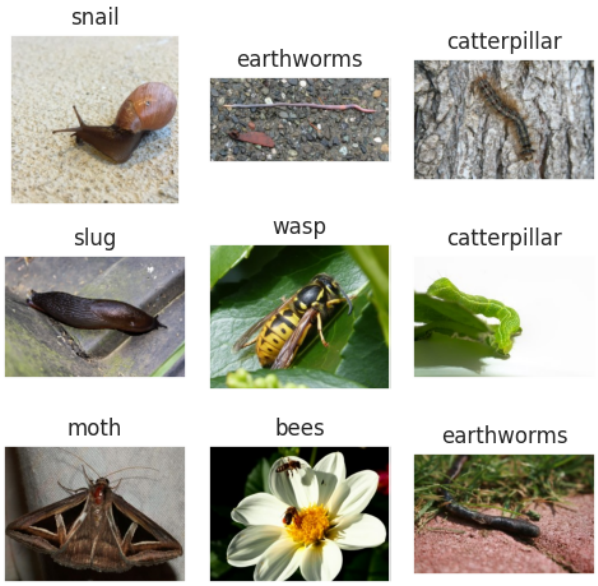


Fig1. Image visualizations - each class from the dataset

The EfficientNetV2-L approach a cutting-edge CNN architecture famous for its exceptional performance and computational efficiency, provides an essential component of our approach. With the use of EfficientNetV2-L's feature extraction powers, our goal is to accurately categorize pest photos into twelve different classes that cover a wide variety of species of pests that are frequently seen in agricultural environments.

Other than the classification work, we also explore data visualization methods to learn more about the features and distribution of the pest image dataset. The image below shows where the different pest classes have been identified in the dataset:

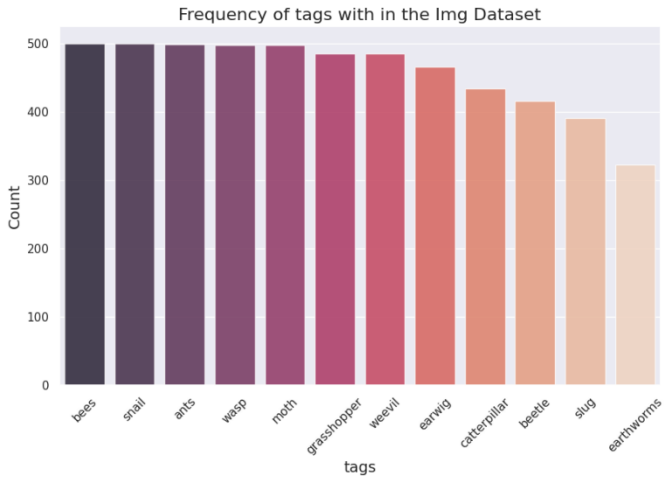


Fig2. Counts of images distribution for each class from the dataset

The above distribution plot shows a visual representation of the distribution of different pest classes across the dataset. This plot is providing opportunity to us to notice clustering or any dispersion patterns among the classes.

We try to identify patterns and trends in the data using visualization techniques like distribution plots, which will help to provide a more detailed understanding of the fundamental connections between various pest classes.

With the combination of state-of-the-art deep learning methods with extensive data visualization, our work aims to support the development of effective pest detection and management systems. We want to provide farmers with unique tools for sustainable pest control and crop protection by proving the effectiveness and practical utility of our method in real-world agricultural applications through empirical evaluations and analysis.

II. Survey

Using recent advancements in deep learning and computer vision, researchers have created a number of novel techniques for recognizing and categorizing agricultural pests. Yolov3 and the Adaptive Honey Badger Algorithm (AHBA) were utilized by Kathole et al. (2023) to enhance object detection parameters. AHBA was able to provide a classification accuracy of 34.6%.

In a similar vein, Liu (2020) presented DFF-ResNet, a pest recognition model that combines residual networks with feature fusion and achieved an overall accuracy of 53.43%.

Gradient-weighted Class Activation Mapping (Grad-CAM) was employed by Selvaraj (2021) to provide better visual explanations than c-MWP; the former outperformed the latter by a margin of 70.58% vs 60.30%.

Finally, Gill (2023) used EfficientNetV2 to do the classification job. The results showed that this deep convolutional neural network is effective with transfer learning, with an accuracy of 88%. These studies show areas for further improvement in agricultural pest identification and offer insightful information about new methods.

III. Background

The development of deep learning brings significant improvements in agricultural research and technology, mainly in the areas of pest detection and management. By using computer machines and artificial intelligence, scientists and professionals can be able to find creative ways to find the solutions to the problems that come because of the cause of pests in agricultural ecosystems cause.

A. Techniques:

Deep Learning: In deep learning, a branch of machine learning, multi-layered neural networks are trained to automatically extract characteristics from unprocessed input. A notable family of deep learning models known as convolutional neural networks, or CNNs, excels in tasks involving image identification and classification.

A machine learning technique called transfer learning makes it possible to apply previously learned knowledge to a new task or topic. Transfer learning enables researchers to take advantage of existing knowledge and accelerate the training process by optimizing pre-trained CNN models on datasets, especially in situations where data availability is limited.

B. Technologies:

EfficientNetV2-L: Its cutting-edge CNN architecture is effective at finding a balance between model performance and efficiency in a variety of tasks. EfficientNetV2-L, which was created as an improvement of the EfficientNet family, provides an ideal match for environments with limited resources because it combines cutting-edge architectural developments like compound scaling and stochastic depth to accomplish more accuracy with fewer variables.

Data Visualization: Model interpretation and exploratory data analysis both depend extensively on data visualization methods. Visualization tools, such scatter plots, distribution plots, and heatmaps, help researchers identify patterns, trends, and irregularities in complicated datasets. This allows researchers to get insights into the true nature of the data and informs decision-making.

Our research aims to create a reliable and effective pest categorization system for agricultural applications by using various methods and technologies. We want to improve the precision, effectiveness, and comprehensibility of pest detection systems using deep learning models, transfer learning methods, and data visualization techniques. This will ultimately help contribute to sustainable agriculture and food security.

IV. METHODOLOGY

A. Data Collection and Preprocessing

This part shows methods to obtain and prepare the data set of images that can be used to train our pest classification model. This involves collecting photos of different pest species that are relevant to agricultural environments and putting them into a structured dataset. We sourced this data from Kaggle. We also report the planning measures performed to improve model generalization: we fix pixel intensities, standardize image dimensions, and augment the dataset.

We divided the image dataset into separate training and testing sets to guarantee the accuracy of our model evaluation. For this, the sci-kit-learn library's train_test_split function is used, with a test size of 20% of the entire dataset. For reproducibility, we additionally set up a random seed (42) and enable dataset shuffling. This section ensures sure the model is tested on data that hasn't been seen yet and trained on a subset of the data so we can evaluate how well it adapts.

We divided the data into three separate categories—training, validation, and testing sets—to make the training, validation, and testing of our pest classification model simpler. A part of the original dataset that was used to train the model and validation set also included in the training set (train_images). Batches of 32 images and labels are provided, and data augmentation techniques are used during training to improve model generalization:

Validation Set (val_images): A portion of the training data is reserved for validation purposes to monitor the model's performance during training and prevent overfitting. The validation set facilitates hyperparameter adjustment and evaluation of the model's capacity for generalization.

Testing Set (test_images): The remaining unseen data from the original dataset is used as the testing set. This set is used to evaluate the final performance of the trained model on unseen data, providing an unbiased assessment of its effectiveness in classifying pest images. The testing set is not shuffled to ensure consistent evaluation across multiple runs.

When training deep learning models on small datasets, data augmentation is essential for improving the models' resilience and capacity for generalization. Using TensorFlow's Sequential API, we define a sequential data augmentation pipeline in this phase.

The augmentation pipeline has the following modifications:

Resizing: Resizing the dataset input images to a standard size of 224x224 pixels.

Rescaling: Scaled the pixel values of the images to the range of [0,1] for normalization.

Random Flip: Applied horizontal flip with a probability of 50%, introducing variations in the orientation of the images.

Random Rotation: Randomly rotated the images by a maximum angle of 0.1 radians (approximately 5.7 degrees) to augment the dataset with rotate instances.

RandomZoom: Apply random zooming with a maximum zoom factor of 0.1, enlarging or shrinking the images within the specified range.

RandomContrast: Adjust the contrast of the images randomly by a maximum factor of 0.1, introducing variations in brightness levels.

```
# Data Augmentation Step
augment = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(224,224),
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.experimental.preprocessing.RandomFlip("horizontal"),
  layers.experimental.preprocessing.RandomRotation(0.1),
  layers.experimental.preprocessing.RandomZoom(0.1),
  layers.experimental.preprocessing.RandomContrast(0.1),
])
```

Fig3. Code for Data augmentation.

Our goal is to improve the model's capacity to generalize to new photos and broaden the training set, which will help the model perform better when it comes to identifying pest images. To do this, we plan to integrate these augmentation techniques into the training pipeline.

B. Model Architecture and Training

➔ ResNet50:

The ResNet50-based[5] model for image classification. We imported the ResNet50 architecture, freezes its layers to preserve pre-trained features, and establishes callbacks for checkpointing and early stopping during training. Custom layers are added for classification, and the model is compiled with specified optimization parameters. Finally, training is conducted using the dataset, with validation and early stopping mechanisms in place to ensure efficient training.

The pre-trained weights from the ImageNet dataset are incorporated via weights as 'ImageNet'. The model's predicted input form is defined as (224, 224, 3), which is an RGB picture with a pixel size of 224x224. Supplementary layers are appended to the model to construct a custom head. Two dense layers with ReLU activation and dropout are introduced. The final output layer comprises 12 units with softmax activation, tailored for multi-class classification.

The model is compiled using the Adam optimizer with a learning rate of 0.0001, categorical cross-entropy loss, and accuracy as the evaluation metric. Training is executed using the train_images dataset, with validation data sourced from val_images.

But coming to the performance of this model we achieved 63% Accuracy only over the test images.

```
35/35 [==============================] - 247s 7s/step - loss: 1.2878 - accuracy: 0.6379
Test Loss: 1.287814974784851
Test Accuracy: 0.6378526091575623
```

So, we have implemented our original model (EfficientNetV2-L) to see whether it gets any better or not.

➔ EfficientNetV2-Large:

We utilize TensorFlow's tf.keras.applications module to load a pre-trained EfficientNetV2-L model. This model instantiation includes the following specifications:

Input shape: Defines the dimensions of the input images fed into the model, with images resized to 224x224 pixels and 3 color channels (RGB).

Include top: Determines whether to incorporate fully connected classification layers above the base convolutional layers. By setting it to False, we exclude these top layers and plan to add our own classification head.

weights: Describes the model's weight initialization procedure. To take advantage of the learnt representations, we start the model using pre-trained weights from the ImageNet dataset.

pooling: Specifies the type of pooling operation applied after the convolutional layers. We opt for global max pooling ('max') to reduce spatial dimensions of the feature maps while retaining essential features.

Additionally, The trainable attribute of the pretrained_model is set to False, effectively freezing the weights of the pre-trained layers. This ensures that the weights remain unchanged during training, as our focus is solely on fine-tuning the top classification layers for our specific pest classification task.

Furthermore, The training will be monitored by three callbacks. They are Tensor board callback, early stopping, and model checkpoint. The following is a summary of the model hyperparameter:

- Batch size: 32
- Epochs: 100
- Input Shape: (224, 224, 3)
- Output layer: 12

Next we define the architecture of the pest classification model and compile it for training. Let's break down the key components and concepts involved:

1. Input and Augmentation:

The pretrained_model.input denotes the input layer of the pre-trained EfficientNetV2-L model. To introduce diversity into the training data and enhance the model's ability to generalize, augment(inputs) applies data augmentation techniques to the input images, including random flips, rotations, zooms, and contrast adjustments.

2. Classification Layers:

Following the pre-trained base model, we incorporate new classification layers tailored to our specific classification task. Flatten() is utilized to flatten the output tensor from the pre-trained model, preparing it for fully connected layers.

Dense layers with ReLU activation functions facilitate feature extraction and introduce non-linearity. A fraction of the units are randomly deactivated during training by dropout layers, which are integrated and have a dropout rate of 0.5 to prevent overfitting.

Furthermore, the activations of the previous layer are normalized by the application of Batch Normalization, which enhances stability and speeds up the training process.

3. Output Layer:

Twelve neurons make up the last layer (Dense), each of which represents one of the twelve pest classes that must be classified. The probability that a picture will belong to a certain class is generated for each class using the softmax activation function.

4. Model Compilation:

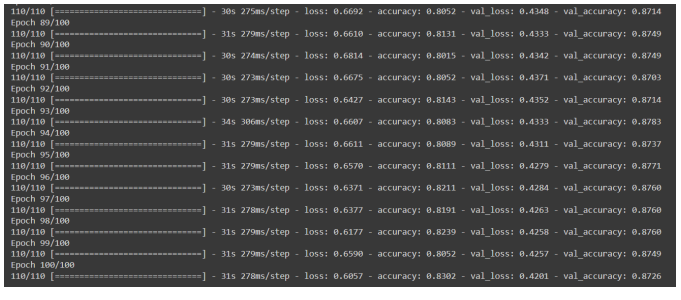
The Adam optimizer with a learning rate of 0.00001 is used to construct the model. During training, Adam, a popular optimization technique, dynamically modifies the learning rate to improve convergence. Categorical cross-entropy is selected as the loss function, given the multi-class classification nature of the task. The accuracy metric is designated to monitor the model's performance throughout training.

5. Training Process:

Training commences using the fit method on the training data (train_images), incorporating specified parameters such as steps_per_epoch, validation_data, validation_steps, and epochs. Steps_per_epoch and validation_steps dictate the number of batches yielded from the generators for each epoch, providing crucial control over the training process.

To track training progress, avoid overfitting, and save the optimal model weights based on validation results, callbacks for early stopping, tensorboard logging, and model checkpointing are added.

Fig4. Training epoch by epoch



By using transfer learning and data augmentation approaches to increase performance and generalization, our goal is to build a strong pest classification model that can correctly identify pest species from input photos.

C. Evaluation

During the evaluation phase, we assess the performance of the trained pest classification model by examining accuracy and loss plots generated throughout the training process.

1. Accuracy Plot:

-The accuracy plot depicts the model's classification accuracy trends across consecutive epochs during training. It offers valuable insights into the model's learning trajectory and its capacity to accurately classify images across various classes.

-An upward trend in the accuracy curve signifies effective learning and continual improvement in performance over time. Conversely, if the accuracy plateaus or declines, it may indicate potential issues such as overfitting or convergence challenges.

- In Fig5., we can see the validation accuracy plot is higher than the Training Plot is because the validation data is smaller than the Training data by 220 images.

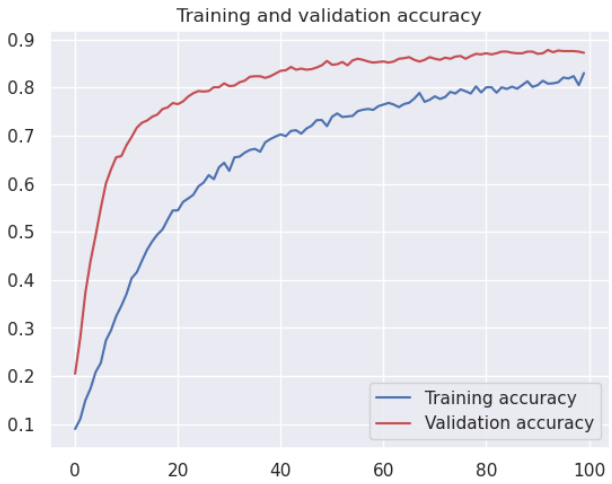


Fig5. Training & Validation Accuracy

2. Loss Plot:

- The loss plot visually represents the evolution of the model's loss function, specifically categorical cross-entropy, across epochs. It serves to quantify the disparity between predicted probabilities and actual labels for both training and validation datasets.

A diminishing loss curve indicates the model's success in minimizing prediction errors and moving closer to an optimal solution. Conversely, a sudden spike in loss might signal overfitting or other issues demanding attention.

By scrutinizing accuracy and loss plots together, we can gauge the effectiveness and generalization capability of the trained model. A model exhibiting consistent accuracy improvement and loss reduction throughout training signifies successful learning and reliable performance on unseen data. Conversely, anomalies or divergent trends in these plots may prompt adjustments to the model's architecture, hyperparameters, or training procedures to enhance performance and address potential concerns.

Fig6. Training & Validation Loss

Overall, accuracy and loss plots serve as invaluable diagnostic aids for monitoring and optimizing machine learning model training, empowering researchers and practitioners to make informed decisions and achieve desired outcomes in classification endeavors.

a) Evaluation using Classification Report:

A classification report offers a condensed presentation of essential performance measures for a classification model, encompassing precision, recall, and F1-score, alongside the model's overall accuracy. This report furnishes a succinct evaluation of the model's efficacy, typically segmented by class, facilitating rapid identification of its strengths and weaknesses. Typically presented in tabular format, each row corresponds to a class, with columns detailing diverse performance metrics. Additionally, the report may incorporate supplementary metrics like support (indicating the quantity of test samples attributed to a specific class) and the macro- and micro-averages of performance metrics across all classes.

	precision	recall	f1-score	support
ants	0.91	1.00	0.95	94
bees	0.87	0.91	0.89	92
beetle	0.91	0.71	0.80	94
catterpillar	0.64	0.81	0.72	85
earthworms	0.90	0.79	0.84	67
earwig	0.92	0.69	0.79	89
grasshopper	0.94	0.91	0.93	105
moth	0.87	0.91	0.89	99
slug	0.90	0.92	0.91	76
snail	1.00	1.00	1.00	102
wasp	0.89	0.90	0.90	104
weevil	0.91	1.00	0.95	92
accuracy			0.88	1099
macro avg	0.89	0.88	0.88	1099
weighted avg	0.89	0.88	0.88	1099

Fig7. Classification report

b) Evaluation using *Confusion Matrix*

To differentiate between accurate and wrong classifications, a confusion matrix provides a tabular overview of a classification model's prediction accuracy on a test dataset. Usually represented as a square matrix, with rows and columns denoting the true and anticipated class labels, respectively. Each cell in the matrix indicates how many test samples were allocated to each class and how many of them the model properly or erroneously recognized. A thorough study of the model's performance is given by the confusion matrix, which includes metrics for each class such as accuracy, precision, recall, and F1-score. It aids in locating hotspots for incorrect categorization and diagnosing issues with the model's capacity for prediction.

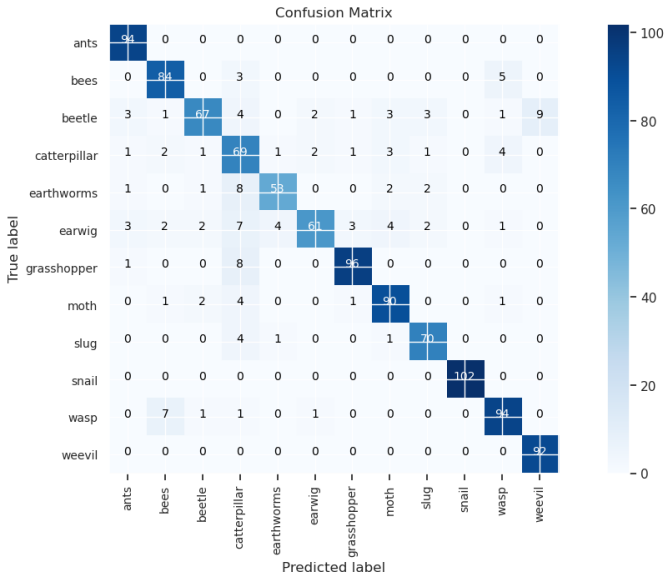


Fig8. Confusion Matrix

D. Prediction

Predictions are done for the random images in the test set.

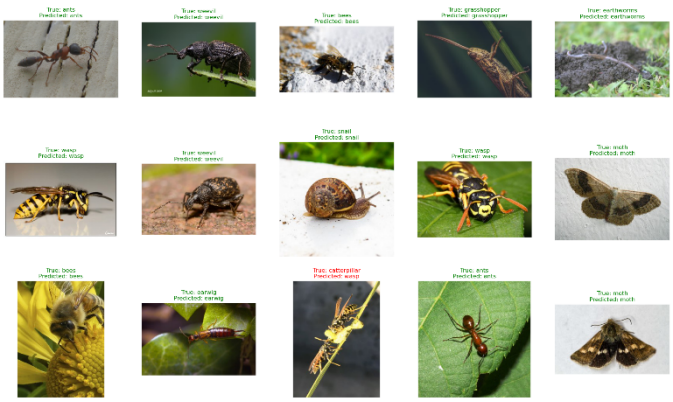


Fig9. Predictions and images

D. Model Interpretabilty using Grad-Cam Visualization

Grad-CAM (Gradient-weighted Class Activation Mapping) [3] is a method utilized for visualizing the significant regions within an input image that influenced a neural network's prediction the most. This technique provides insights into the specific areas of focus within the image that the model considered during its decision-making process. An extension of the CAM technique, Grad-CAM broadens its applicability to any model employing a convolutional neural network (CNN) as its fundamental architecture.

Setup and Visualization:

- Preprocesses input images using EfficientNet's preprocessing function.
- Removes the SoftMax activation from the last layer of the model.
- Displays Grad-CAM heatmaps overlaid on test images (3 rows, 5 columns).
- Titles show true labels and model predictions.

We have visualized the areas of an input image that were most important for a neural network's prediction by implementing Grad-CAM (Gradient-weighted Class Activation Mapping). It includes of preprocessing, Grad-CAM heatmap computation, and overlaying routines to produce visually appealing heatmaps over original photos.

The Grad-CAM visualizations are generated and shown with the true and predicted labels of an image collection by iterating over a series of photos after preprocessing and model architecture definition.

Finally, we have highlighted significant picture regions to demonstrate how Grad-CAM may be used to learn more about a model's decision-making process.

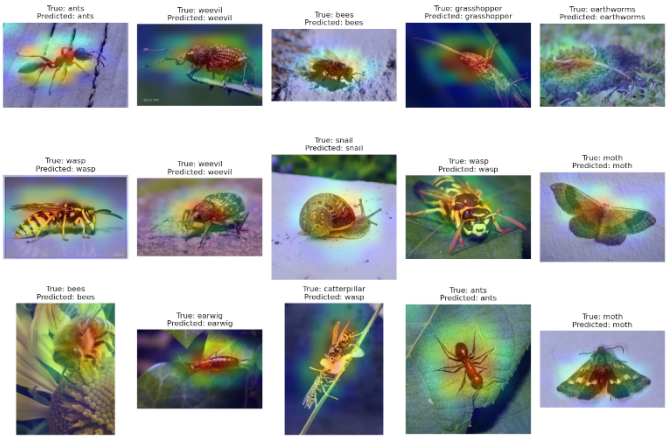


Fig 10. Grad-CAM Visualization

V. CHALLENGES:

Training Time:

- Due to the complex model architecture and data processing, training times were prolonged, and a significant amount of computing power was needed.
- When possible, take use of GPU acceleration. To avoid needless training, think about model checkpointing and early ending.

Data Quality:

- Due to imbalances in the data and problems with quality, it was unable to adequate coverage of all pest types.
- Developed a plan for cleansing the data and create synthetic data to supplement underrepresented classes using data Augmentation technique.

Potential Future Challenges:

- The model might struggle to generalize to real-world environments due to varying lighting conditions, backgrounds, and unseen pest species.
- Deploying the model for use in agricultural settings may pose latency and connectivity issues.

VI. CONCLUSION:

In Conclusion to help farmers identify and manage pests early on, we created a novel pest classification model in this work by utilizing the EfficientNetV2-L architecture and extensive data augmentation approaches. Across twelve distinct insect categories, our model demonstrated an astounding 88% classification accuracy, proving the efficacy of transfer learning and data augmentation in feature extraction from pest photos.

We used Grad-CAM (Gradient-weighted Class Activation Mapping) visualization to improve interpretability. This provided us with information about the specific regions of the pictures that affected the model's predictions the most. The interpretability of the model helps users to understand the logic underlying the model's conclusions, which increases their confidence in the predictions made by the model. Farmers may utilize a reliable tool to improve sustainable farming practices and lessen crop damage due to the model's high accuracy and interpretability. It also opens up new possibilities for breakthroughs in agricultural pest identification and encourages continuous innovation in sustainable agriculture by improving model parameters and expanding datasets to improve robustness and generalization in real-world scenarios.

REFERENCES

[1] https://en.wikipedia.org/wiki/Convolutional_neural_network

[2] https://paperswithcode.com/method/efficientnet#:~:text=**EfficientNet**%20is%20a,using%20a%20*compound%20coefficient*.

[3] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. ArXiv. https://doi.org/10.1007/s11263-019-01228-7

[4] Atul B. Kathole, Jayashree Katti, Savita Lonare. Identify and classify pests in the agricultural sector using metaheuristics deep learning approach(2023).https://www.sciencedirect.com/science/article/pii/S27731863 2300018X

[5] Wenjie Liu., Guoqing Wu., Fuji Ren., Xin Kang. DFF-ResNet: An insect pest recognition model based on residual networks(2020). https://ieeexplore.ieee.org/abstract/document/9259195

[6] Ramprasaath R. Selvaraj., Michael Cogswell., Abhishek Das., Ramakrishna Vedantam. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization(2021). https://ieeexplore.ieee.org/document/8237336

[7] Kanwarpartap Singh Gill; Vatsala Anand; Rahul Chauhan; Garima Verma; Rupesh Gupta. Agricultural Pests Classification Using Deep Convolutional Neural Networks and Transfer Learning(2023). https://ieeexplore.ieee.org/abstract/document/10425080?casa_token=f5j0o6R d0n4AAAAA:3M0lWgqiFYKAo8z2e0oAgwHLroJx0aPd7_toQXmSz2WRz Uon87JJ5sM4AyUitV-T9Rh0a2hu8MA

[8] Datasetlink: https://www.kaggle.com/datasets/gauravduttakiit/agricultural-pests-dataset/data

[9] Kathole, A. B., Katti, J., Lonare, S., & Dharmale, G. (2023). Identify and classify pests in the agricultural sector using metaheuristics deep learning approach. Franklin Open, 3, 100024. https://doi.org/10.1016/j.fraope.2023.100024.