# Automated Insect Identification: A Comparative Study of Custom CNN and Cross-Fold CNN for Agricultural Insect Classification

Toluwalope Olateru-Olagbegi

May 2025

## 1 Introduction

Accurate identification of crop-damaging insects like Armyworms and Cabbage Loopers is critical for effective agricultural pest management. Manual identification is labor-intensive and prone to errors, necessitating automated solutions. Deep learning, particularly Convolutional Neural Networks (CNNs) and transfer learning, offers promising tools for image-based classification of insect species. The effectiveness of CNNs in image classification has been highlighted in various studies [Elngar et al., 2021, Krishna et al., 2018]. The provided scripts, implementing a Custom CNN and a Cross-Fold CNN, represent two distinct approaches to the binary classification of Armyworm and Cabbage Looper images. The Custom CNN is built from scratch. In contrast, the Cross-Fold CNN utilizes MobileNetV2, a pre-trained model, and employs K-Fold cross-validation for robust evaluation, a technique known to improve model generalization [Han et al., 2018]. This report compares these models based on their data processing, architecture, training, and performance, highlighting their strengths and limitations for agricultural applications.

## 2 Abstract

### 2.1 Background

This report evaluates two deep learning models to classify images of armyworms and cabbage loopers, two agricultural pests, based on the provided Python scripts (implementing a Custom CNN and a Cross-Fold CNN). The first model (Custom CNN) employs a custom Convolutional Neural Network (CNN) [Elngar et al., 2021], while the second model (Cross-Fold CNN) leverages transfer learning with MobileNetV2 and Stratified K-Fold cross-validation [Han et al., 2018]. The analysis compares their methodologies, performance, and suitability for real-world agricultural applications. Results indicate that the Cross-Fold CNN achieves superior generalization and accuracy, making it a more robust

solution for automated pest identification. The report concludes with practical implications and recommendations for future improvements.

## 2.2 Problem Statement: Challenges in CNN-Based Image Classification for Insect Species

Achieving high accuracy in CNN-based image classification of insect species remains a significant challenge, particularly to distinguish similar pests such as armyworms and cabbage loopers. These insects share similar green, elongated bodies with subtle marking differences, increasing the risk of misclassification, especially for custom CNN models that may struggle to learn distinguishing features from limited datasets [Krishna et al., 2018]. Furthermore, variability in image conditions—such as differences in lighting, angles, and backgrounds—can degrade model performance, leading to overfitting or poor generalization of unseen data. For example, the Custom CNN is based on a single train-test split, which may not adequately capture the variability of the dataset, resulting in a lower accuracy of 0.47 and limited robustness. In contrast, while the Cross-Fold CNN leverages transfer learning with MobileNetV2 and K-Fold cross-validation to improve accuracy to an average of 0.8746, it still faces challenges in fully adapting pre-trained features to insect-specific characteristics, as ImageNet weights may not be optimized for such specialized tasks. Therefore, this report aims to address the problem of achieving high classification accuracy in CNN-based models to distinguish Armyworms and Cabbage Loopers, by comparing the performance of a Custom CNN and a transfer learning approach (Cross-Fold CNN) and identifying strategies to improve the accuracy and generalization of the model for real-world agricultural applications.

## 2.3 Methodology

This study implements two deep learning models to classify images of Armyworms and Cabbage Loopers, using the Python scripts implementing a Custom CNN and a Cross-Fold CNN. The process begins with a dataset of insect images stored in Google Drive under folders named "Armyworms" and "Cabbage Loopers." The images are loaded using OpenCV, converted from BGR to RGB color space, resized to 224×224 pixels for uniformity, and normalized by dividing pixel values by 255.0 to scale them to the range [0,1]. The dataset is then shuffled with a random seed of 42 for reproducibility and split into 80% training and 20% testing sets using `train_test_split` from `scikit-learn`. To enhance model robustness, data augmentation is applied via TensorFlow's `ImageDataGenerator`, with the Custom CNN using moderate transformations such as 20-degree rotation and 10% width/height shifts, while the Cross-Fold CNN applies more aggressive augmentation, including 30-degree rotation and 20% shifts, to support its K-Fold cross-validation approach, a technique known to improve generalization [Han et al., 2018].

The script implementing the Custom CNN builds a custom Convolutional Neural Network (CNN) designed for binary classification of the insect images.

This model features three convolutional blocks with 128, 64, and 32 filters, respectively, each using a 5×5 kernel and ReLU activation, followed by 2×2 max-pooling layers to reduce spatial dimensions. The feature maps are flattened and passed through a dense layer with 256 units and ReLU activation, a dropout layer with a rate of 0.3 to prevent overfitting, and a final sigmoid output layer for binary classification. The model is compiled with the Adam optimizer at a learning rate of 0.004, using binary cross-entropy loss and accuracy as the evaluation metric, and is trained for 10 epochs with a batch size of 32, saving the trained model as `new_cnn_model.h5`. In contrast, the script implementing the Cross-Fold CNN utilizes transfer learning with MobileNetV2, a pre-trained model on ImageNet, where the base layers are frozen to retain learned features. A custom head is added, consisting of a GlobalAveragePooling2D layer, a dropout layer with a rate of 0.3, and a sigmoid output layer. This model is compiled with the Adam optimizer at a learning rate of 0.0005, binary cross-entropy loss, and accuracy metric, and is trained using Stratified K-Fold cross-validation with $k = 5$, running for up to 20 epochs per fold with a batch size of 32. Callbacks such as `EarlyStopping` with a patience of 5 and `ReduceLROnPlateau` with a patience of 3 are used to optimize training by halting early if validation loss stalls and reducing the learning rate if needed. The benefits of transfer learning in image classification have been well-documented [Han et al., 2018].

The evaluation process differs between the two scripts to reflect their training strategies. For the Custom CNN, the model's performance is assessed on the test set after training, reporting accuracy and loss to gauge its effectiveness. For the Cross-Fold CNN, accuracy and loss are computed for each fold's validation set during cross-validation, with the average accuracy across folds providing a robust performance estimate, and the final model from the last fold is evaluated on its validation set. This approach ensures that both models are thoroughly evaluated for their ability to generalize to unseen data, addressing the challenges of distinguishing visually similar insects under varying image conditions.

## 2.4 Data Processing

The data processing for this study begins with a dataset of Armyworm and Cabbage Looper images, stored in Google Drive under two folders named "Armyworms" and "Cabbage Loopers," with the assumption of a balanced number of images per class capturing the insects in diverse conditions such as varying lighting, angles, and backgrounds. In both scripts, implementing the Custom CNN and the Cross-Fold CNN, images are loaded using OpenCV, where they are read in BGR format and converted to RGB color space to align with deep learning input requirements, followed by resizing to a uniform 224×224 pixel resolution to ensure compatibility with the models' architectures. Pixel values are then normalized to the range of 0 to 1 by dividing by 255.0, a step that facilitates faster convergence during training by scaling the input features. The dataset is labeled with 0 for Armyworms and 1 for Cabbage Loopers to define the binary classification task, shuffled using a random seed of 42 for reproducibility,

and split into 80% training and 20% testing sets through `train_test_split` from `scikit-learn`. To enhance model robustness against overfitting, data augmentation is applied using TensorFlow's `ImageDataGenerator`, where the Custom CNN employs moderate transformations including 20-degree rotation, 10% width and height shifts, 10% shear, 10% zoom, and horizontal flipping, while the Cross-Fold CNN uses more aggressive augmentation with 30-degree rotation, 20% width and height shifts, 20% shear, 20% zoom, and horizontal flipping to better support its K-Fold cross-validation approach by simulating a wider range of variations in the smaller fold-specific training subsets, a strategy often employed to improve the generalization of deep learning models [Krishna et al., 2018].

### 2.4.1 Model Architectures

The two scripts implement distinct architectures for binary classification, tailored to balance computational efficiency and classification performance, as detailed below.

**Model Architecture 1** The script implementing the Custom CNN builds a custom Convolutional Neural Network (CNN) designed for binary classification of Armyworm and Cabbage Looper images, focusing on a lightweight architecture to capture spatial features efficiently. This model comprises three convolutional blocks, starting with 128 filters of size 5×5 and ReLU activation, followed by a 2×2 max-pooling layer to reduce spatial dimensions, then a second block with 64 filters of the same size and activation, again followed by 2×2 max-pooling, and a third block with 32 filters, also with ReLU activation and max-pooling. After the convolutional layers, the feature maps are flattened into a 1D vector, which is passed through a dense layer with 256 units and ReLU activation to learn higher-level features, followed by a dropout layer with a rate of 0.3 to mitigate overfitting, and finally a dense output layer with a single unit and sigmoid activation to produce the binary classification probability. The model is compiled using the Adam optimizer with a learning rate of 0.004, binary cross-entropy loss to match the binary classification task, and accuracy as the evaluation metric, preparing it for training on the preprocessed insect dataset. The design choices in custom CNNs significantly impact their performance in image classification tasks [Elngar et al., 2021].

**Model Architecture 2** In contrast, the script implementing the Cross-Fold CNN leverages transfer learning with MobileNetV2, a pre-trained model optimized for efficient image classification, using weights from ImageNet to enhance feature extraction for the insect classification task. The base MobileNetV2 model is loaded with its top classification layer removed and its layers frozen to retain the pre-trained features, minimizing training time and the need for a large dataset, while accepting input images of shape 224×224 pixels with three color channels. A custom head is added to adapt the model for binary classification, starting with a GlobalAveragePooling2D layer to reduce the spatial

dimensions of the feature maps into a 1D vector, followed by a dropout layer with a rate of 0.3 to prevent overfitting, and concluding with a dense output layer with a single unit and sigmoid activation to output the binary classification probability. This model is compiled with the Adam optimizer at a lower learning rate of 0.0005 to fine-tune the pre-trained weights effectively, using binary cross-entropy loss and accuracy as the evaluation metric, and is designed to be trained with Stratified K-Fold cross-validation to ensure robust performance across varied data splits. Transfer learning is highly effective in improving the performance of image classification models, especially when dealing with limited datasets [Han et al., 2018].

### 2.4.2 Training and Evaluation

The training and evaluation processes for the two models are tailored to their respective architectures and validation strategies to optimize performance and assess their ability to generalize to unseen data in the binary classification of Armyworm and Cabbage Looper images. For the Custom CNN, training is conducted over 10 epochs with a batch size of 32, utilizing the augmented training data to improve robustness, and the model's performance is evaluated on the test set after training by computing accuracy and loss, with the final trained model saved as `new_cnn_model.h5` for future use. The challenges of training CNNs from scratch, particularly with limited data, are well-documented [Krishna et al., 2018]. In contrast, the Cross-Fold CNN employs Stratified K-Fold cross-validation with $k = 5$ to ensure balanced class distribution across folds, training for up to 20 epochs per fold with a batch size of 32 and incorporating callbacks such as `EarlyStopping` with a patience of 5 to halt training if the validation loss does not improve and `ReduceLROnPlateau` with a patience of 3 to reduce the learning rate by a factor of 0.1 down to a minimum of 0.00001 if the validation loss plateaus, thereby optimizing the training process. The evaluation for the Cross-Fold CNN involves computing accuracy and loss for each fold's validation set, averaging the accuracy across all folds to provide a robust performance estimate, and assessing the final model from the last fold on its validation set to ensure comprehensive evaluation of its generalization capabilities under varying image conditions. The use of K-Fold cross-validation is a standard practice to obtain a more reliable estimate of a model's performance on unseen data [Han et al., 2018].

# 3 Results

Custom CNN: Test accuracy reached 0.47 after 10 epochs. Training accuracy varied between 0.5018 and 0.5588, while validation accuracy mostly stayed at 0.4750 (peaking at 0.5250 in Epoch 2). Validation loss ranged from 0.6933 to 0.6969, with a final test loss of 0.6971, indicating limited learning progress. This aligns with challenges in training CNNs from scratch, as noted in prior studies on CNN-based image classification [El-

ngar et al., 2021, Krishna et al., 2018]. Cross-Fold CNN: Across 5-fold cross-validation, accuracies were [0.8500, 0.8750, 0.9250, 0.8000, 0.9231], averaging 0.8746. Validation accuracy improved consistently (e.g., Fold 1: 0.5250 to 0.8500; Fold 5: 0.6154 to 0.9231), and validation loss decreased progressively (e.g., Fold 1: 0.7415 to 0.4240; Fold 5: 0.6363 to 0.1981), reflecting strong learning and generalization. These results highlight the effectiveness of transfer learning, as demonstrated in prior work on CNN transfer learning and deep learning for image classification [Han et al., 2018, Krishna et al., 2018].

# 4 Conclusion

The Cross-Fold CNN significantly surpasses the Custom CNN (0.8746 vs. 0.47 average accuracy), benefiting from transfer learning and K-Fold cross-validation for better generalization [Han et al., 2018, Krishna et al., 2018]. The Custom CNN exhibits poor convergence, with stagnant validation accuracy and high loss, likely due to its high learning rate (0.004), small dataset, and lack of cross-validation, a common issue in CNN training [Elngar et al., 2021, Krishna et al., 2018]. The Cross-Fold CNN is the better choice for insect classification, though further improvements could come from dataset expansion, hyperparameter tuning, and fine-tuning. The Custom CNN needs adjustments like a lower learning rate and cross-validation to enhance its performance.

# References


- Ahmed A Elngar, Mohamed Arafa, Amar Fathy, Basma Moustafa, Omar Mahmoud, Mohamed Shaban, and Nehal Fawzy. Image classification based on cnn: a survey. *Journal of Cybersecurity and Information Management*, 6(1): 18–50, 2021.

D. Han, Q. Liu, and W. Fan. A new image classification method using cnn transfer learning and web data augmentation. *Expert Systems With Applications*, 95:43–56, 2018. doi: 10.1016/j.eswa.2017.11.028.

M. M. Krishna, M. Neelima, M. Harshali, and M. V. G. Rao. Image classification using deep learning. *International Journal of Engineering & Technology*, 7 (2.7):614, 2018. doi: 10.14419/ijet.v7i2.7.10892.