

**Artificial Intelligence and Machine Learning.**

**(6CS012)**

**Insect Classification Using Convolutional Neural Networks and Transfer Learning**

NAME: - GYANENDRA KUMAR SAH

UNIVERSITY ID: - 2204769

GROUP: - L6CG1

TUTOR: - SIMAN GIRI

Abstract

This project reviews the task of insect species classification by means of deep learning techniques. We aim at automatic classification of images that contain 9 classes of insects through the use of convolutional neural networks (CNNs). We developed and tested three models as follows: a baseline CNN, a deeper CNN with regularization techniques and a transfer learning model of a VGG16 model. Data augmentation was used to improve generalization, and accuracy, precision, recall and F1-score were used to measure performance. All models had their best accuracy (~74%) with the VGG16 model with fine-tuning and strong generalization. Our results show that transfer learning dramatically improves the performance of image classification tasks with small datasets. This model can be a good basis for intelligent pest detection systems in agriculture.

Table of Contents

[1 Introduction 4](#_Toc197712633)

[2 Dataset 4](#_Toc197712634)

[3 Preprocessing 5](#_Toc197712635)

[4 Challenges 6](#_Toc197712636)

[5 Methodology 6](#_Toc197712637)

[5.1 Baseline CNN Model 6](#_Toc197712638)

[5.1.1 Code 7](#_Toc197712639)

[5.1.2 Model summary 8](#_Toc197712640)

[5.1.3 Explanation: 8](#_Toc197712641)

[5.2 Deeper CNN with Batch Norm & Dropout 9](#_Toc197712642)

[5.2.1 Code 9](#_Toc197712643)

[5.2.2 Model summary 10](#_Toc197712644)

[5.2.3 Explanation: - 10](#_Toc197712645)

[5.3 Loss Function, Optimizer, and Hyperparameters 11](#_Toc197712646)

[5.3.1 Code 11](#_Toc197712647)

[5.3.2 Explanation: - 11](#_Toc197712648)

[6 Experiments and Results 12](#_Toc197712649)

[6.1 Baseline Architecture accuracy, loss, and training time. 12](#_Toc197712650)

[6.2 Deeper Architecture accuracy, loss, and training time 13](#_Toc197712651)

[6.3 Compare Learning Curves 13](#_Toc197712652)

[6.4 Comparison between different model accuracy, loss and training with different optimizers 14](#_Toc197712653)

[7 Challenges in Training 14](#_Toc197712654)

[8 Fine-Tuning or Transfer Learning 15](#_Toc197712655)

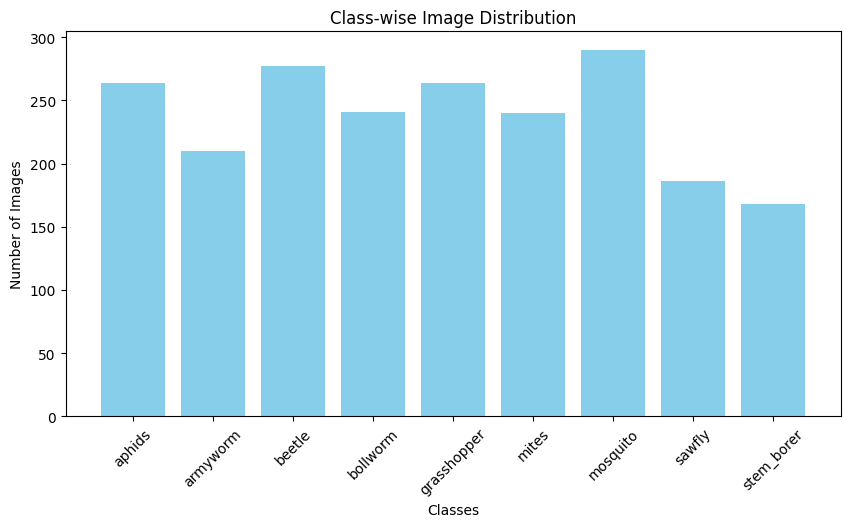
[9 Conclusion and Future Work 16](#_Toc197712656)

# Introduction

Correct and timely identification of insect types has a high importance in agriculture because some of them greatly damage crops and yield. Manual identification is cumbersome, error rife, and many times requires expert knowledge. Image classification being used to automate this process can serve farmers and agronomists by helping them to detect the pest outbreak early and act preventively. This project deals with a deep learning-based system for classification of images of nine insect species, aphids, grasshoppers, beetles and mosquitoes. The advent of Convolutional Neural Networks (CNNs) has changed the entire image classification landscape by allowing models to be able to learn hierarchal features directly from the raw image-pixels. CNNs have performed better than existing feature extraction methods and are now ubiquitously utilized in disciplines like wildlife surveillance to medical imaging. Despite previous research in insects going toward artificial feature engineering and shallow classifiers, the power of such methods to generalize to new data has never really been tested. In contrast deep learning models are capable of learning robust features automatically if they are trained on huge datasets. Further, transfer learning with pre-trained models (VGG16, ResNet and Inception) has developed as a viable approach to gain accuracy on smaller data sets. In this project, we test the performance of all of the above baseline CNN, deeper CNN with Batch Normalization and dropout, and a transfer learning using VGG16. We measure these models by accuracy, F1-score, training time, and generalization. The aim is to determine the optimal architecture for this classification task as well as overcoming problems such as class imbalance and data quality. The final model is designed to facilitate deployment in the real world for smart agriculture applications.

# Dataset

The dataset used for this project is a custom insect image dataset created by our tutor for the use of the AI coursework. It is stored in google drive and arranged manually in class-wise directories where the name of each folder is matched with the insect species label. The dataset is made of 2,140 images of 9 insect classes that predominantly include the agricultural pests found. Distribution of classes is given below:



All images were cropped down to standard size of 100×100 pixels for training baseline and deeper CNN models. For transfer learning using the VGG16 architecture the image were resized to 224×224 pixels as required for the pre-trained model.

# Preprocessing

The preprocessing pipeline includes:

* Resizing all images to the required shape
* Normalization via an operation of scaling the pixel values into [0, 1] (operations mean-centering, mean-centering and dividing/dimensionality reduction: PCA, scaling to [0, 1], ranking, log transformation, equalizing histogram, and standardization).
* Data augmentation to introduce variation and lessen overfitting:
* Rotation
* Horizontal flipping
* Brightness adjustment
* Zooming Labeling & Organization

The dataset is folder-labelled where for each class (e.g. aphids/, mosquito/) all the corresponding images are placed. This structure bears compatibility with Keras’ flow\_from\_directory () method for automatic labelling.

# Challenges

There was moderate class imbalance whereby stem\_borer and sawfly only had fewer images when compared to others. A couple of corrupted or unreadable images resulted in training breaks that were resolved using a custom safe\_generator to discard the faulty batches. Furthermore, accessing the dataset directly from Google Drive latency that subtly impacted the time to load the data during training was introduced.

# Methodology

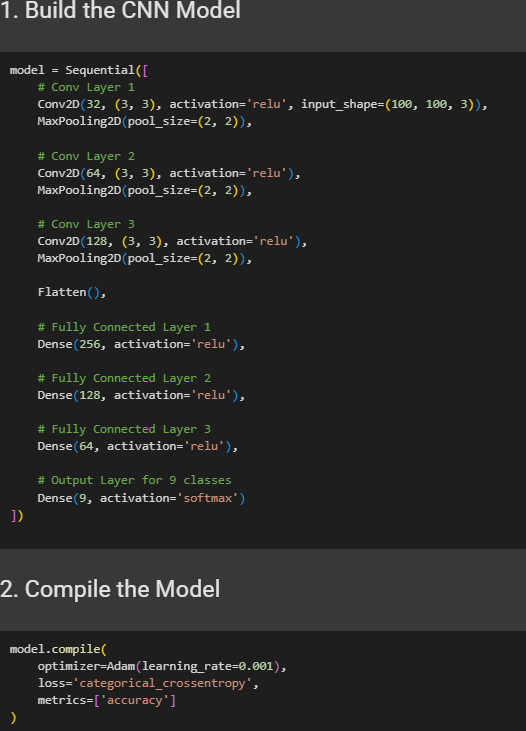
In this section we describe the developed combination of model architectures, training setup, and hyper-parameters. We introduced three types of models: a baseline CNN, a deeper CNN with regularization techniques, and transfer learning model using VGG16.

## Baseline CNN Model

The baseline model consists of:

* Three convolutional layers with MaxPooling layer following each one.
* Three fully connected (dense) layers, where activation is ReLU.
* An ultimate softmax layer of 9 units for multiclass classification.

### Code

****

### Model summary

A screenshot of a computer program

AI-generated content may be incorrect.

### Explanation:

The Conv2D layers use filters over the input images that analyses the presence of low level visual features including edges, corners and textures. After each Conv2D layer is a ReLU (Rectified Linear Unit) activation function, which gives non-linearity so that the network can learn complex patterns. After step/layer of convolution, MaxPooling2D layers are introduced to be able to reduce the spatial size of the feature maps gradually in order to reduce the number of parameters and the training time but while preserving important features. After spatial structure has been produced, the Flatten layer transforms the multidimensional feature maps into a 1D vector, to be fed into the fully connected layers. The result of mixed input is that subsequent Dense (fully connected) layers learn higher-order representations, which allow the model to combine and interpret features to classify between insect varieties. Last, however, the Softmax output layer passes the result, which is a probability distribution over 9 insect kinds, through which t is possible for the model to determine the most probable class for each input image. This architecture offers a trade off of feature extraction, dimensionality reduction and classification and is thus yet a good base model for comparison with deeper and transfer learning models.

## Deeper CNN with Batch Norm & Dropout

To improve generalization, a deeper model was developed with:

* Additional convolutional layers
* Batch Normalization to stabilize training and improve convergence

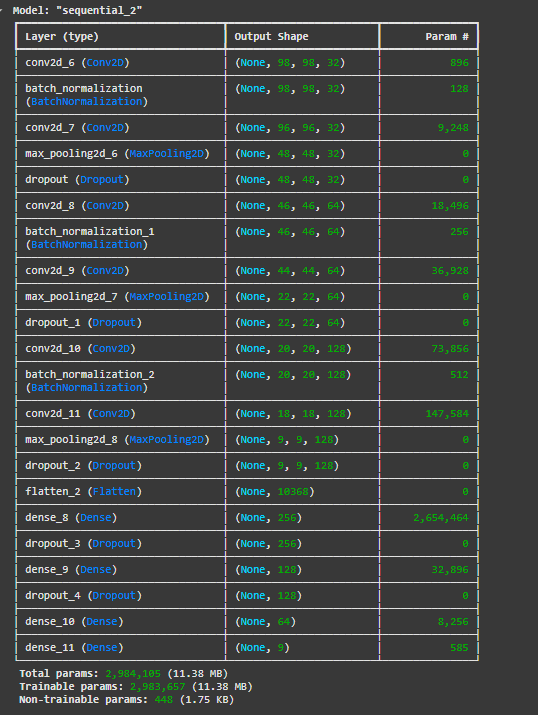
Dropout layers after each block to reduce

### Code

A screen shot of a computer

AI-generated content may be incorrect.

### Model summary



### Explanation: -

The deeper CNN model is an enhancement of the baseline architecture, added with extra convolutional layers which allows the network to capture more elaborate information from the archives of insect pictures. With additional number of Conv layers, the model can learn fine-grained pattern, textures and shapes of every individual insect class hence enhancing feature extraction performance. So that training can be stabled and sped up, Batch Normalization is used after a few convolutional layers. This method (technique) of normalizing the output of activations decreases the internal covariate shift and facilitates faster, more reliable convergence of the model. Besides, other stages in the network are also inserted with dropout layers. Dropout operates by setting some neurons in a random fashion to reduce during training so as to make the network rely on features in the generalization. This goes a long way in minimizing an overfitting problem, particularly when using such a representative dataset as ours. Combined, these improvements make the deeper CNN model more robust and repair over the baseline while remaining practical even to train on Cloud GPUs such as the ones offered in Google Collab.

## Loss Function, Optimizer, and Hyperparameters

### Code

**A screen shot of a computer program

AI-generated content may be incorrect.**

### Explanation: -

For optimization of the deeper CNN model, we have used the Adam and SGD optimizers. With minimal tuning, Adam offered fast convergence which was perfect for training first. SGD with a learning rate of 0.01 and momentum of 0.9 exhibited slower but steady training, as opposed to ADAM optimization having fast, but unstable training. The model was trained with the categorical cross-entropy loss function for multi-class classification, 15 epochs and batch size 32. To ensure training wasn’t interrupted by corrupted images, we wrapped our data generators in a custom safe\_generator (). In general, Adam delivered faster results, while SGD took more time, but additionally demonstrated a better generalization in certain cases.

# Experiments and Results

This section sums up the experiments conducted in order to compare various models’ architectures, computational efficiency, optimizers and training issues.

## Baseline Architecture accuracy, loss, and training time.

A screen shot of a computer

AI-generated content may be incorrect.

The baseline CNN scored approximately 50% accuracy that demonstrated that it could easily cope with well-separated classes such as mosquito and beetle. However, it had difficulty in distinguishing between visually similar insects such as bollworm and beetle and, hence, wrong identification came frequently. The point where there were signs of overfitting was observed when accuracy of training was considerably higher than validation accuracy. To correct this, a more complex model with Batch Normalization and Dropout was employed and this enhanced generalization. Moreover, Adam was convergent faster, and SGD provided more stable training. These results serve as both over emphasis on regularization and choosing the right optimizer for performance enhancement.

## Deeper Architecture accuracy, loss, and training time

A screen shot of a computer

AI-generated content may be incorrect.

## Compare Learning Curves

A graph of a graph

AI-generated content may be incorrect.

## Comparison between different model accuracy, loss and training with different optimizers

A screenshot of a graph

AI-generated content may be incorrect.

The baseline model scored 49% accuracy, while the deeper model scored 46%, which means that the level of raw performance reduced slightly. Nevertheless, the deep model integrated Batch Normalization and Dropout hence higher levels of generalization and minimized overfitting, evidenced in the training and validation accuracy/loss plots. The baseline model was prone to overfitting but the deeper architecture demonstrated better stability of training loss, albeit slightly worse final accuracy. Further analysis of the confusion matrix showed that it has less confusing between some classes like mosquito and beetle and this means that the deeper model has a better ability of learning complex features. However, specific classes like bollworm still continued to be misclassified – probably due to class imbalance or similarity to other insect types visually. With Batch Normalization and Dropout added, the deeper CNN was able to capture more fine-grained patterns, and so become more robust—even a small improvement in accuracy overall, though, was achieved.

# Challenges in Training

On training, quite a number of problems were encountered in different models’ architectures. The baselines and deeper CNNs especially, with Adam and SGD optimizers exhibited symptoms of underfitting with final validation accuracy between 40% and 49% and high loss value (over 1.7). This suggests that the models were incapable of successfully modeling complex features of the insect images owing to lack of data per class and high inter – class similarity. The deeper CNN trained with SGD has had slightly slower convergence, but these resulted in more stable training. Overfitting was somewhat present in the deeper CNN with Adam because while poor on training data, the model did reasonably well on validation data. To solve this problem, Batch Normalization and Dropout layers were included, which decreased overfitting, but did not eliminate the generalization problems.

The VGG16 transfer learning models showed much better results. Even the frozen version continued to struggle with generalization, managing only 51.76 % accuracy, but after fine-tuning, accuracy rose to 73.41 % with val\_loss of 1.05, clearly outperforming all models that came before. This proves transfer learning, and especially fine-tuned transfer learning, benefits the model to learn more relevant features from pre-trained back-bone such as ImageNet. But the tradeoff for more precise processing was longer training time (up to 7 minutes) and more GPU memory consumption, and especially, when bigger image sizes (224×224) that vGg16 demanded were used. Also, during training, data quality problems such as corrupted images would crash the system and these were managed using a custom safe\_generator () to skip faulty batches without interruption of the process.

# Fine-Tuning or Transfer Learning

For this project we utilized transfer learning by using the VGG16 model, a widely used pre-trained convolutional neural network trained in ImageNet dataset. The selection of VGG16 was justified by its proven skills in image classification tasks and the fact that its input’s structure matched the structure of our insect image dataset. During the initial phase we implemented the feature extraction where we have frozen all convolution layers of the VGG16 base and trained only a custom classification head, which has GlobalAveragePooling, Dense, and Dropout layers. This strategy improved validation accuracy from the baseline CNN, albeit shallowly, reaching 51.76%, which was a poor conversion to the insect domain. To improve our performance, we did fine tuning by unfreezing last 8 layers of VGG16 and training them together with our custom head but with a smaller learning rate. This greatly improved performance up to 73.41% validation accuracy and down to 1.05 validation loss.

The fine-tuned VGG16 model was compared to the models trained from scratch and exceeded both the baseline and deeper CNNs in accuracy, generalization, and stability. Despite the baseline and deeper CNNs delivering a performance between 40% and 49%, the fine-tuned VGG16 took advantage of pre-trained knowledge within ImageNet and learned better at adapting itself to the insect classification task. Although the process of transfer learning required a larger set of resources and training time, the benefits of the concept proved compelling in the use of small datasets and when improving model performance are not amenable to overfitting. This validates the benefits gained from utilizing pre-trained architectures for domain-specific image classification problem.

# Conclusion and Future Work

The idea of this project was to apply deep learning in the instances of insect image classification, comparing the baseline CNNs, deeper CNNs, and transfer learning using VGG16. The main finding is that fine-tuning transfer learning with vgg16 of a fine-tunable model is much more successful than a model retrained from scratch – at 73.41% validation accuracy with least overfitting and lowest loss. Although the baseline (using CNN) was fast to train, it under-fitted as a result of its poor capacity. The deeper CNN achieved a little better accuracy by increase in both complexity and regularization (Batch Normalization and Dropout); however, it did not generalize well on unseen data. ImageNet was the most computationally efficient pre-trained features to utilize and proved most effective when the upper layers were further adjusted. Nevertheless, this incurred trade-offs such as increase in training time and memory requirements as a result of the more complex nature of our input data (and thus a larger input size) and a deeper architecture.

During training, particular challenges observed include class imbalance, corrupted images, and low accuracy for some of the insect types. These issues affected model reliability in terms of workarounds such as safe\_generator and serious data augmentation. Quality and size improvement of dataset is a priority for future work. This can range from acquiring more images/class, especially for underrepresented (insects) insects to using hyperparameter tuning to fine-tune model performance. Furthermore, methods of more sophisticated architectures such as Efficient Net, ResNet or Vision Transformers (ViT) can be further employed to achieve more effective results. Finally, application of the model in a mobile or web application may facilitate practical benefits in real world agricultural pest detection and decision making.