ASHESI

ASHESI UNIVERSITY

FALL ARMYWORM DETECTION IN GHANA USING MACHINE LEARNING

UNDERGRADUATE THESIS

B.Sc. Computer Science

Mustapha Tidoo Yussif

2020

ASHESI UNIVERSITY

FALL ARMYWORM DETECTION IN GHANA USING MACHINE LEARNING

UNDERGRADUATE THESIS

Thesis submitted to the Department of Computer Science, Ashesi University in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

Mustapha Tidoo Yussif

2020

DECLARATION

I hereby declare that this Undergraduate Thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

....

Candidate's Name:

....

Date:

....

I hereby declare that preparation and presentation of this Undergraduate Thesis were supervised in accordance with the guidelines on supervision of Undergraduate Thesis laid down by Ashesi University.

Supervisor's Signature:

....

Supervisor's Name:

....

Date:

....

Acknowledgements

Before recognizing the individuals for their specific contributions towards the success of this project, I want to give thanks to the Almighty God for my life and health.

I would like to express my sincere gratitude to my supervisor, Abdul Wasay, for his constructive criticism and motivations, which always put me on my toes throughout this endeavor. The regular meetings conducted by Abdul Wasay and his swift responses made it possible for me to work on this project joyfully by encountering a few bottlenecks which otherwise could have been a tough journey for me.

I would also like to sincerely thank the Computer Science Department of my school, Ashesi University, for buying the Floyd Hub virtual machine used to train the final models in this research.

Abstract

The economy of many African countries depends on the agriculture sector. However, the agriculture sector is at the mercy of FAW following the arrival of the pest in 2016. The pest is more dangerous to crops than other crop pests because they attack many crops, especially cereal crops, which are the staple food in Ghana. The most widely used technique to control the pest in the country is pheromone-baited traps and manual surveillance. Although these techniques are cheap to implement, they are prone to errors. This paper explores how to use machine learning techniques to detect Fall armyworm (FAW) in Ghana. The paper discusses various data preprocessing methods such as data augmentation, transfer learning, convolutional networks for classification as well as using object detection algorithms to detect the worm.

Keyword: *Fall armyworm, FAW, convolutional neural networks, CNN, Object detection.*

Table of Content

List of Tables

List of Figures

Chapter 1: Introduction and Background

The Fall armyworm [FAW; Spodoptera frugiperda(J. E. Smith)], a devastating pest endemic to tropical and subtropical regions of America, was first seen in Africa in January 2016 [8]. Since then, the pest continued to spread across the continent. The presence of the pest was recorded in about 30 African countries and reported to have caused extensive damage to essential economic crops, such as maize, millet, sorghum [8]. The FAW pest has three major characteristics that make it devastating and dangerous to Africa than other crops pest. (1) It can feed on over 80 different species of crops. (2) It reproduces and spreads quickly and attack places rapidly. (3) The environmental conditions of the sub-Saharan African countries, especially where there is binomial rainfall, are suitable for FAW to persist throughout the year. The last characteristic implies that the pest will affect not only the rain-fed crops but also the irrigated crops [4].

The FAW has a complete life cycle (egg, larva, pupa, and moth), just like many other insects. The female moth lays about 100-200 eggs on the host plant's leaves, usually near the base of the plant. In about 3-6 days, the eggs hatch young caterpillars that feed superficially on the leaves, creating windowpanes on the leaves. This stage is where the pest is very dangerous and difficult to control. The larvas (caterpillars) are the ones that cause mass defoliation. They feed on the vital parts of the plant, usually the leaves, stem, or the growing point rendering the crop to wither and die. Also, they feed on the plants in the night, making it difficult for farmers to find them. The young caterpillars in the whorls of young plants and on the leaves around the cobs silk of the older plants tend to damage the crop by creating ragged holes through the leaves. When it finishes destroying one crop, it moves to another one. In about 14 days, they would have fully grown and dropped on to the ground to begin the pupal stage. On the ground, it burrows into the soil or hides inside dead leaves (if the soil is hard) before pupating. After 8-9 days, the pupa forms the adult moths to restart the life cycle.

The economy of many African countries (e.g., Ghana) depends on agriculture. Sadly, the agricultural sectors of these countries are now at the mercy of the tenacious FAW. The Food

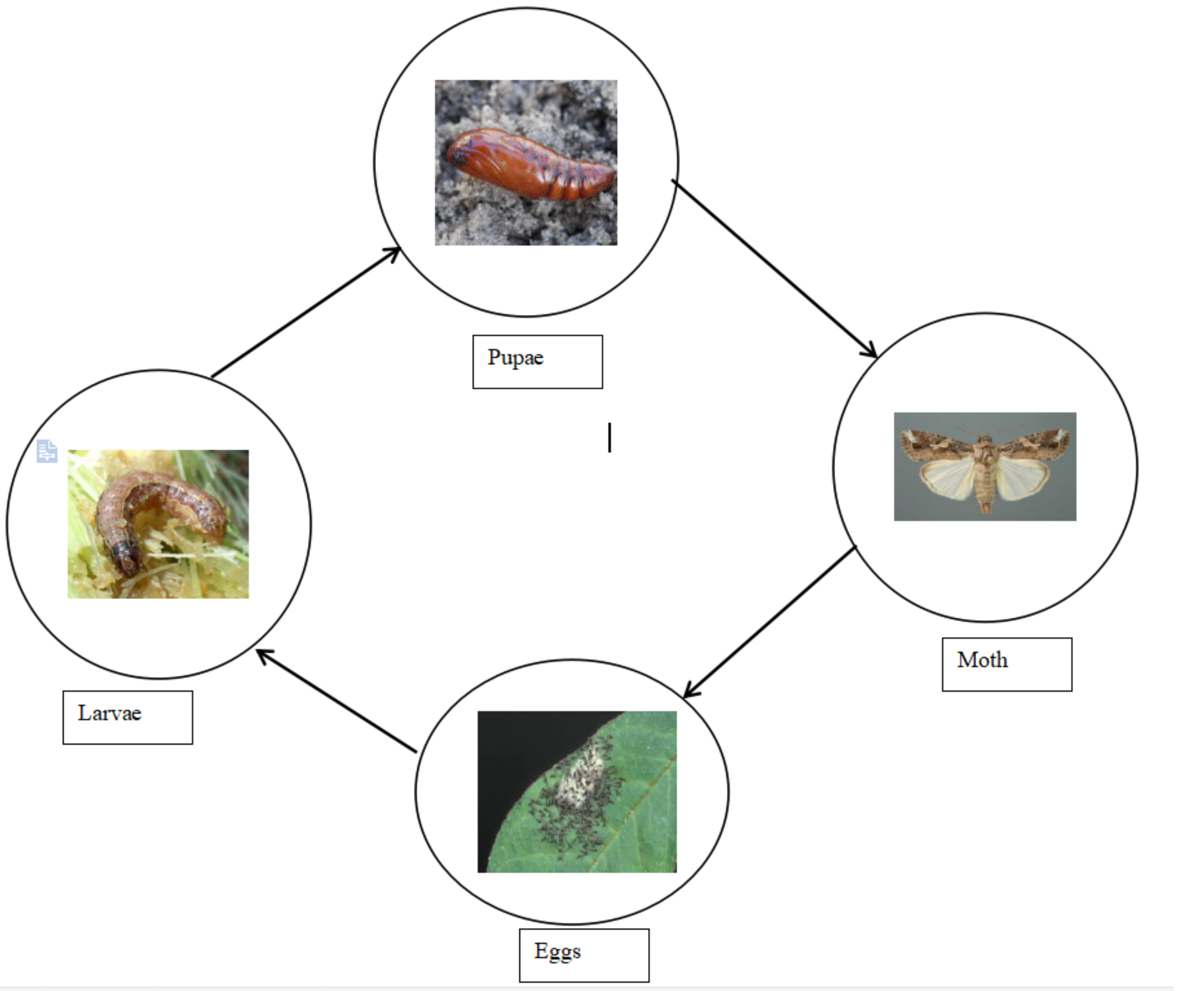

Figure 1.1: Life cycle of FAW

and Agriculture Organization (FAO) classified FAW as a threat to food security because of its devastating characteristics[7]. Ghana is endowed with a lot of natural resources such as gold, salt, diamond, manganese, iron (to mention only a few). Nonetheless, agriculture undoubtedly plays a vital role in providing employment to the citizenry as well as ensuring food security. Following the arrival of the pest, the lucrative looking agriculture is not secured anymore. In early 2017, the pest affected up to 100,000 ha of crops and rose up to 500, 000 ha by March 2017 [26]. The pest cost the country millions of money. According to [2], the impact of the FAW is between 22 and 67 percent of yield in Ghana, causing the country to lose humongous amounts of money. CABI report shows that the pest cost Ghana $162 million in 2017 by affecting up to 500, 000 tonnes of maize and sorghum [26]. Despite the imperative need to obtain effective methods to control FAW, traditional Pheromone-baited traps, manual surveillance, and chemical spraying are commonly used in the country. The pheromone-baited trap technique involves placing a trap in the field and then regularly observing it. The field expert counts the number of moths trapped. The data from the field is used to make informed decisions, such as deciding on a date to spray the area. Pheromone traps and pesticides spraying are time-consuming and tedious. Pheromone-baited trap is also prone to errors since the accuracy of the results depends on the field specialist who can underestimate or overestimate the count. Since there is a continuous observation of the trap, there is a need for automatic FAW detection.

The limitations of the pheromone-baited traps and the pesticide spraying control measures have called for preventive and control measures to fight the pest. Automatic identification and classification of fall armyworm using machine learning algorithms have been identified as the state-of-the-art methods for detecting the pest. The machine learning-based approach is extensively discussed in chapter 2. The importance of identifying the fall armyworm cannot be underestimated since it helps farmers prepare adequately for the pest. An automatic detection system can predict fall armyworm invasion to admonish farmers in Ghana to prepare. The objective of this paper is to use convolutional neural networks to identify and classify fall armyworm.

Chapter 2: Related Work

This section outlines the direction of recent research on this topic. There have been a plethora of studies devoted to finding ways and means to detect and control the fall armyworm. The techniques proposed in the papers range from manual to automated systems. However, myriad of the findings favored automating operations to detect and identify the pest. The ultimate reasons for favoring automated over manual operations are explored in this section.

2.1: Pheromone-baited Trap-based Insects Identification

Most insects, including FAW moths, communicate with one another via pheromones. When a female moth is ready for mating, it releases sex pheromone to send a signal to the male moth. This observation gives rise to the development of pheromone-baited trap-based systems for sampling insects [20]. Scientists have taken advantage of this to create synthetic pheromones that are used as baits in traps for monitoring and controlling the pest.

In the crop field, farmers or extension workers set up the pheromone traps and visit them at least once in a week. They count the number of male insects captured to estimate the population size of the insect and to decide on the measures to take. According to [16, 18], pheromone-baited traps can detect the insect early before an infestation occurs. This can early-warn stakeholders to implement appropriate control measures to combat the insects.

Generally, pheromone-baited traps do not require a taxonomic expert to determine whether the captured pest is FAW or other insects: the sex pheromone used as the bait is targeted at an insect species and is expected to attract only the insects of that species [16]. The cost of involving a taxonomic expert in regular sticky traps (regular sticky traps captured every insect that falls on it) is not incurred in pheromone-baited traps systems. Pheromone-baited traps are also used to control the spread of the insects by keeping the male moth away from female moth to obviate mating. This is called *mating disruption*. In other words, pheromone can be used to confuse the insects, thus preventing them from mating and influencing their life cycle [16]. The male moth does not live long, and when it is withheld by the trap for long, it dies without mating. The population size of the insects would reduce because the female moths cannot lay eggs without mating. The efficacy of the pheromone-baited trap depends on factors such as the shape of the trap and the ability of the bait to be able to deliver the required sex pheromone for attraction [12]. [15] also emphasizes that reducing the trap opening width creates a condition that does not allow trapped male moths to escape hence increasing the number of moths captured.

Although pheromone-baited traps are cheap, it is prone to error, and it is labor-intensive as it involves inspectors visiting the field every week [15, 3]. Because the inspectors manually count the insects trapped, the number of moths captured can be underestimated or overestimated, which can significantly affect any decision taken based on this data. To this end,[15, 3] propose complementing pheromone traps with systems that can reduce human intervention and send data as and when they are captured.

2.2: Chemical Based Pest Management

Pheromone-baited traps are the most widely used method to detect the presence of FAW moths. Following the detection by the traps, farmers in Ghana often apply insecticides. However, controlling the Fall armyworm with insecticide is inconsistent because the fall armyworm larvae usually live in the host whorl, which is difficult for pesticides to penetrate [18]. The pest is difficult to control with pesticides, especially during the day. Usually, they feed on the plant in the morning and night. Farmers, therefore, cannot identify where the pest population is concentrated, and they end up missing their target. This contributes to the over 98 percent of the sprayed insecticides missing the target insects, but end up in the water, or affecting non-target species[20]. It is an undoubted fact that insects have a positive impact on humans, ranging from pollinating flowers to serving as natural control measures to plant pests. At the same time, some insect species, such as FAW, are pests to economically important crops and are threats to global food security [9]. Consequently, there is the need to implement measures that will control the harmful pest while causing no harm to the harmless insects.

2.3: Machine Leaning Based Insect Identification and Classification

Computer scientists have gained a keen interest in the field of biodiversity to investigate the feasibility of using machine learning to control crop pests without harming beneficial insects. For many years now, data on insects, including image data, have been gathered and labeled to use for insect identification and classification.

Machine Learning (ML) is a data-driven study that allows software or machine to learn from the gathered data and use the knowledge gained to detect or recognize objects. Different machine learning algorithms (Support Vector Machines, Artificial Neural Network, K-nearest neighbor, etc.) can be applied to ML tasks. For the most part, the algorithm that gives the best score on a particular data is preferred. Based on the nature of the input data, a particular algorithm can be selected. For instance, Convolutional Neural Network, CNN, or ConvNet (a variant of Artificial Neural Network) is mostly used for image classification [15]. CNN consists of several layers that determine the complexity of the network; a more significant number of layers means more complexity. The network takes an image as input, applies filters to it at each layer, and the output is served as an input to the next layer. These filters identify features like lines, sharp edges, a wing of an insect in the input image. The output of the CNN is a vector of features - another representation of the original image. In a classification task, the features vector from the last convolutional layer is flattened and fed to a classifier. CNN and different types of classifiers that can be used are explained in chapter 3. In an object detection task, however, feature maps from intermediate layers can also be used since they represent the original image at different convolutional layers. Detection algorithms have two convolutional layers.

1. A regression layer for localizing the position of the object.

2. A classification layer for identifying the class of the object.

The application of deep learning will effectively remove the human from the loop hence achieving an automated FAW monitoring system[8].

2.4: Summary of Related Work

So far, the Machine learning techniques, especially CNNs, perform better than the traditional pheromone-based traps, manual surveillance and application of pesticides in accurate recognition of FAW and other pests to facilitate timely preventive measures. CNN's learn features from the images and use the learned features to classify or detect the insects in unseen images. One challenge with CNNs is that they require large datasets for good performance. However, such datasets are usually not available in the agricultural field. The unavailability of the images in this domain often limits the application of the powerful CNNs. In tasks where the dataset is small, transfer learning, and data augmentation are commonly applied to improve performance. The literature shows that, in the past, CNNs have been used for classifying and detecting FAW in Etiopia, Zambia, etc. However, to the best of my knowledge, no CNN-based approach has been used to classify or detect the different growth life cycle stages of the worm in Ghana.

Chapter 3: Methodology

The current state-of-the-art approaches for classifying images and identifying objects in images use machine learning algorithms and convolutional neural networks to achieve high performance. This research explores how these state-of-the-art techniques can be used to detect FAW in Ghana. In this chapter, a summary of the dataset is covered before moving on to discuss the machine learning techniques.

3.1: Dataset

A subset of the IP102 dataset[27] (a Large-Scale Benchmark Dataset for Insect Pest Recognition dataset) was used for the binary classification, growth cycle classification, and detection tasks. The original dataset was collated by researchers from Nankai University and Cardiff University. The researchers used the internet as the primary source to gather most of the images. The researchers relied on the search engines to collect images using common names of the pest as search keys. In some cases, synonyms of the pest common names were used. Images were also captured from videos that contained the content of pests. A total of 75, 222 pest images belonging to 102 classes with an average size of 737 images per class were obtained [27]. The original dataset contained data on different pests, which was superfluous to this research since the main hypothesis for this project was to test how to classify and detect Fall armyworm. The Fall armyworm class was taken out to represent the positive class, and images from the other classes were randomly taken to form the negative class for the binary classification.

The table 3.1 summaries the dataset for the binary classification task.

The Fall armyworm pest has four life cycle stages. I manually classified the images into

Task	no. Train images	no. Test images	no. Val images	class
Binary classification	940	147	441	Total
	581	92	272	Positive(FAW)
	359	55	169	Negative(No FAW)

Table 3.1: Dataset for the binary classification task

Task	no. Train images	no. Val images	no. Test images	class
Multi-class classification	27	15	16	Egg
	448	62	165	Larva
	20	11	12	Pupa
	89	27	43	Moth
	584	115	236	Total

Table 3.2: Dataset for the multiclass classification task

these four stages (eggs, larva, pupa, and moth) for multiclass classification. The training set contained 584 images. The validation set contained 115 images, and the test set has a total of 236 images.

For the detection task, however, the original dataset, IP102 dataset [27] did not contain bounding box annotations for the Fall armyworm class. I used the labelImg annotation tool to create bounding boxes in some of the images manually. Only 86 images in total were annotated; 71 for training and 15 for testing. The detection data was split into 71 training images and 15 testing images. Figure 3.1 shows a sample image with bounding boxes around the life cycle stages of the pest.

3.2: Data Preprocessing

The data preprocessing task marks the beginning of machine learning tasks. It involves cleaning the data and presenting it in a way that it is easy for the machine learning algorithm to process. For image processing tasks, this may include re-scaling the images. The data preprocessing task usually results in the improvement of model performance. The majority of the preprocessing was already done since the dataset was used to train baseline models. Primary data preprocessing tasks such as re-scaling the images to speed up the training process was carried out in this research. Also, normalization was used to standardize the input images.

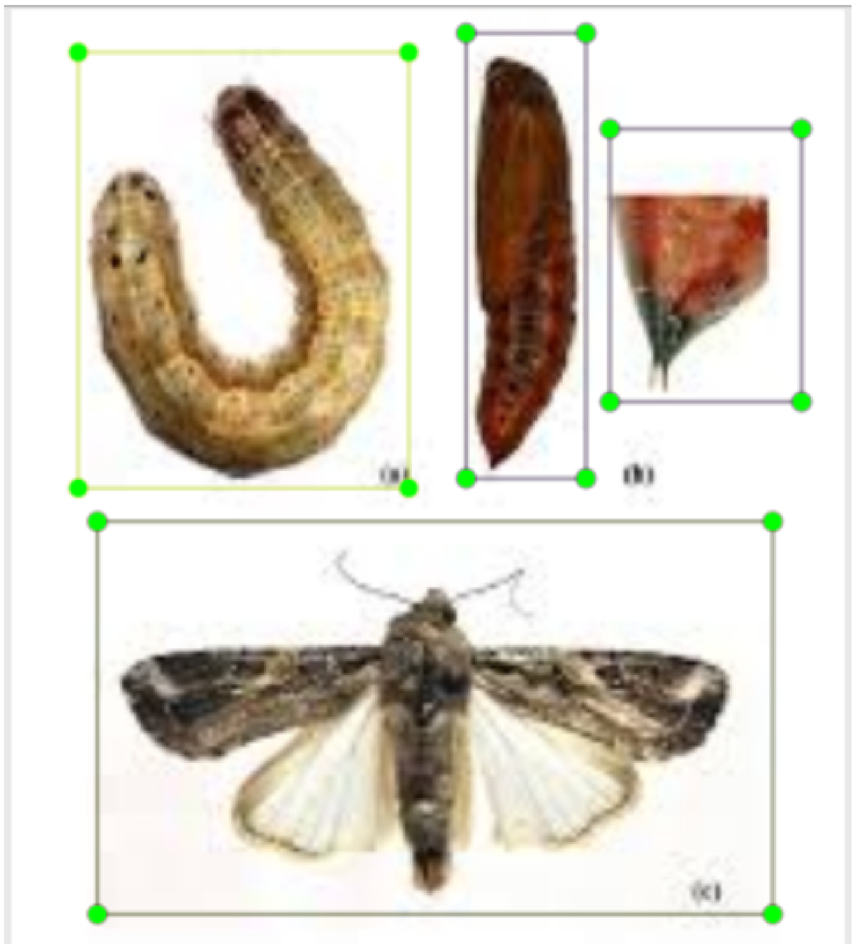

Figure 3.1: A sample image with bounding boxes. The first bounding box is around the larval stage of the pest. The second and third are for the pupal stage, and the fifth and sixth are for the moth. The egg stage is not in this image, but ar in other images.

3.3: Data Augmentation

CNNs are known to be data-hungry because they require large datasets to get good performance. In tasks where the dataset is small, data augmentation is used to increase the number of samples [17]. Data augmentation increases the number of samples in training, test, and validation datasets. Increasing the number of samples in the training dataset often improves performance. The following augmentation techniques were applied in the binary and growth cycle classification tasks:

- Random crop: focus on one part of the original image.

- Horizontal Flip: reflect the original image along its horizontal axis.
- Rotation 90: rotate the original image by 90 degrees clockwise.

However, only random crop and random flip were applied in the detection task.

The dataset for the growth cycle classification was highly imbalanced. The samples in the classes were more than each other by a very big margin. In the train split, for example, the number of samples in the three classes (eggs, pupa, moth) combined was less than the samples in only the larva class. Also, within the three classes, the data was imbalanced. Imbalanced data affect model performance because models tend to be biased towards the class with the most samples. In this research, data augmentation was used to handle the unbalanced data. Specifically, weighted random sampling without replacement was employed. With this technique, samples were randomly selected based on their weights to train, validate, or test the model.

3.4: FAW Classification: Convolutional Neural Networks

In the early years, image processing tasks comprise of: preprocessing of data, handcrafting features from the images using SIFT, HOG, SURF, GIST[27], training, and evaluating models. Manual engineering of features is labor-intensive and also requires experts, which is not always available [22]. Also, the handcrafted features may be over-engineered or too general for the given task. As a result of the limitation as mentioned above, a model may either over-fit or under-fit on the features. This calls for the need for a system to learn the essential features in the input data automatically. This is where CNNs come in to play.

Convolutional Neural Networks are feed-forward neural networks, comprising several layers, specialized in processing image data [13]. In its most basic form, a deep Convolutional Neural Network model consists of two layers: the convolutional blocks and a classifier.

3.4.1: The convolutional Blocks

The convolutional layers utilize convolving filters to extract patterns from an input image. The filters are also called kernels. They are n x n size. Usually, the preferred value for n is an odd number less than nine [13]. An input image is passed to the convolutional layer, a filter is moved across the image, and a value is calculated based on the filter. This operation is called convolving. During the convolution operation, some pixels called stride values are skipped. The most commonly used stride is 2 x 2. A filter used could be responsible for identifying edges (as shown in 3.2) or shapes in the images, which contributes massively to identifying semantic objects in the image. With filters, an important feature of the armyworm, such as the eyes, wings, or antenna, could be identified. As a result of this, convolutional layers are generally used as feature extractors.

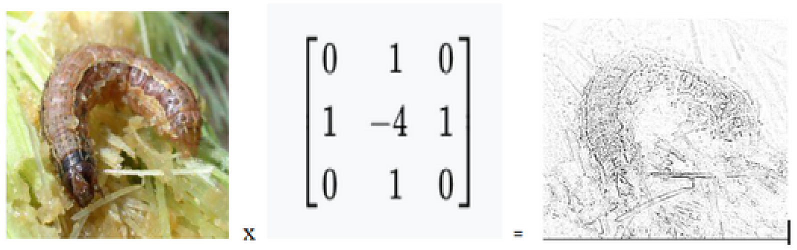
x
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
=

Figure 3.2: Detecting edges in images using edge detection filter.

A complete convolution of kernels over an image generates a vector called features map. The features map is passed to an activation function or a dropout layer or a normalization layer. Activation functions are discussed in the next sub-section.

The dropout layer is used to handle over-fitting. The output of the dropout, normalization, or activation function could be passed to a max-pooling layer (there are other pooling layers; however, the max-pooling layer is mostly used). The max-pooling layer selects the largest values from the feature maps, which are then passed as an input to the next layer. As input is passed down the deep convolutional network, it gets down-sampled. Padding layers are included to maintain the shape of the input image throughout the network to prevent losing relevant features. The output of the last convolutional layer is flattened and passed to a classifier.

3.4.2: Activation Functions

In neural networks, activation functions introduce non-linear properties. They are responsible for converting an input signal of a node into an output signal. The activation functions used in this research were the ReLu and softmax function. ReLu was used in the convolutional blocks. There are other activation functions (e.g., tanh), but ReLu is faster. Because of its speed advantage, it is preferred over the other activation functions to reduce training time. Apart from its speed advantage, it is also simple and easy to implement. Mathematically, the ReLu function can be expressed as below:

$$f(x) = max(0, x)$$

It returns the input value if it is a positive number or returns zero if the input is a negative number. Because the output of the ReLu function can be zero or any positive value, it is only applied in the convolutional block. It is not used in classifiers because the output of the classifiers is a vector of probabilities, which is expected to be in the range [0, 1]. As a result of this, the softmax (for multiclass classification) and sigmoid (for binary classification) were used, in this research, to squash the output of the network to values within the range [0,1]. Mathematically, the Sigmoid function can be expressed as:

$$g(z) = \frac{1}{1 + \exp -z}, \quad where\ z = \theta^T x$$

and the softmax functions can be expressed as:

$$h(\theta^T x) = \frac{\exp \theta_i^T x}{\sum_{i=1}^{n} \exp \theta_i^T x}, \quad where\ i = 1, 2, 3, ...n(the\ number\ of\ classes).$$

3.5: The classifier Layer

A classifier takes the feature map from the last CNN layer and outputs discrete values representing the classes in the data. The output of the last layer of the CNN is flattened and passed to a classifier. Conventionally, a fully connected(FC) layer with a softmax function, is often used. For comparison, a FC layer with a softmax and a linear Support Vector Machine were employed as classifiers in this research. Specifically, the LIBLINEAR[10] implementation of SVM was used. The ultimate reason for using it is because of its speed advantage. The LIBLINEAR[10] is fast hence reduces training time. It is suitable for doing large-scale, regularized linear classification and regression. Figure 3.3 is a typical example of a model comprising a convolutional neural network and SVM classifier.

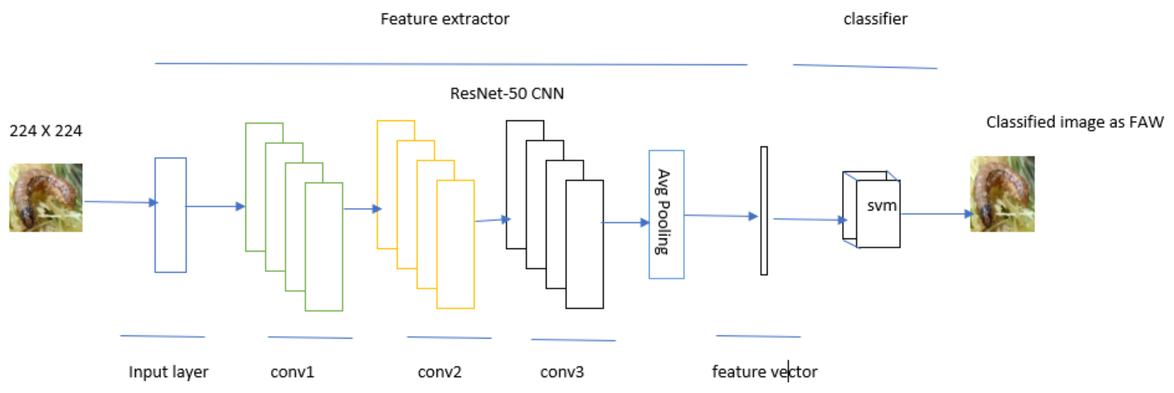

Figure 3.3: ResNet-50 convolutional network + SVM classifier.

3.6: FAW Detection: Using Single Shot Multibox Detector (SSD)

Object detection tasks deal with identifying instances of semantic objects in an input image or video [6]. Unlike a classification model, an object detection model finds the location of an object present in the image. Object detection models, such as Faster R-CNN[19] have two components. A region proposal component and detection component. The detection component comprises localization convolutions and classification convolutions. The object localization component (a regression network) predicts the bounding boxes of the objects, and the object classification component finds the object type present in the image. Non-maximum suppression is used to combine overlapping bounding boxes to form one bounding box. The SSD[14] detection algorithm experimented in this paper uses an entirely different approach than the approach of detection algorithms discussed above. The SSD[14] uses only one neural network to perform detection. In its basic form, it comprises three CNNs. Base convolutions (derived from pretrained models for image classification), extra convolutions (extra layers added on top of the base model), and prediction/multi-box convolutions (layers for localization of object and identification of object types). The only difference between the model in [14] and the implementation in this paper is the base convolutions network. [14] uses vgg16, but this paper uses RestNet50[11]. The following reasons inform the choice of the SSD[14] model:

- It is simple and relatively easy to implement because it uses a single neural network.
- It is fast and supports real-time object detection.
- ResNet50[11] can be used to derive the base convolutions.

3.7: Loss Functions and Optimization Method

A loss function is a function used to evaluate how well a model performs on a given data by measuring how predictions, y-hat, deviate from the ground-truth values, y. In other words, a loss function computes the average error between the true labels and the predicted labels of a specific model on a given data. A high loss indicates poor performance. During training, the model learns the underlying regularity (the ground-truth values), by calculating the average errors and updating its parameters based on the average error. Some of the most widely used loss functions are Cross-Entropy, Mean Square Error, Mean Absolute Error.

An optimization method is an algorithm model used to minimize the loss function.

3.7.1: For the Classification models

In this research, Cross-entropy loss was used for both the binary and growth-cycle classifications because it is the best loss function for classification tasks[5]. Adam optimization was utilized in the classification tasks.

3.7.2: For the Detection models

The SSD multibox loss was used in the detection task. A detailed technical explanation and equation of the multibox loss is explained in [14]. This section, however, gives a high level explanation of it. The multibox is a weighted loss. It comprises **confidence loss** and **location loss**. The confidence loss is a **crossentropy** loss. **The crossentropy** is used to calculate the average error by the model in assigning a label to each detected object. The location loss is a regression loss. Specifically, it is **smooth L1 loss**. The purpose of this loss is to make sure that the correct location of the object is determined. In a simple language level the multibox loss looks like the equation below:

$$weighted\_loss = confidence\_loss + alpha * location\_loss$$

The SGD optimization was used in the detection task.

3.8: Using Transfer Learning

Machine learning algorithms, especially deep neural networks, require large labeled datasets to train models. Getting such data to train models from scratch is often impractical because of the cost involved. Transfer learning is the technique employed to handle small datasets. Transfer learning means using a pre-trained model trained on a different dataset to solve a similar problem. Some of the pre-trained models are AlexNet[13], VGG-16[23], GoogleNet[25], ResNet[11]. In order to achieve higher performance, the source dataset should have similar characteristics as the target dataset [24]. However, if the source dataset and the target dataset are very different, transfer learning can still yield improvement in performance [1] . Besides, in a situation where there are no models that have been trained on a similar dataset, the last layers of a pre-trained model trained on a more generic dataset can be fined tuned. Fine-tuning models and training only some of the layers speeds up the training process. In this research, the RestNet50[11], was used as a feature extractor for both the classification and the detection tasks. This is because the ImageNet[21] (the dataset RestNet50[11] was trained on) is quite different from the pests dataset used in this paper. This model was used to extract generic features since there was no model trained on large pest datasets. Also, the total number of samples in the dataset used in this project was not enough to train a convolutional neural network from scratch to obtain a good performance. Ultimately, the RestNet50[11] model was chosen because IP102[27] dataset came with ResNet50[11] pretrained weights.

3.9: Evaluation Metrics

3.9.1: For the Classification Models

For the classification task, the models were evaluated on accuracy, precision, recall, and F1-score. Accuracy is the percentage of instances where the models predicted value is the same as the true label. Although accuracy might be a good metric for some problems, it might not be the best metric for other tasks. In tasks such as predicting FAW, the whole farm might be destroyed by the pests if the model predicted that the pests are not in the farm and farmers did not put measures in place. Also, accuracy is not an appropriate metric for imbalanced data. The ultimate reason is that the overwhelming majority class(es) will inundate the minority class(s). In this context, accuracy was a bad metric; hence was not the only metric used to evaluate the performance of a model. The reason for using this metric, however, was for comparing two techniques for training binary classifiers on the same data.

Precision is calculated as the ratio of correctly positive predictions to the total number of positive predictions. In simple language, precision measures the number of correct positive predictions. Mathematically, precision for binary class classification is:

$$Precision = \frac{TruePositive(TP)}{TruePositive + FalsePositive(FP)}$$

And multi-class is:

$$Precision = \frac{SumofTP}{SumofTP + SumofFP}$$

The value of precision is in the range [0,1], where 0 represents no precision, and 1 shows perfect precision. Although precision is a good metric, it does not tell the whole story because it does not show how many real positive class examples are predicted as negative class examples. This is where recall comes in.

Recall measures the number of correct positive predictions out of all positive predictions. It is calculated as the ratio of true positive to the total number of true positive, and false negative. Mathematically, recall for binary classification is:

$$Recall = \frac{TruePositive(TP)}{(TruePositive + FalseNegative(FN))}$$

And multiclass is:

$$Recall = \frac{SumofTP}{(SumofTP + SumofFN}$$

The last metric used to evaluate the models was f1 score metric. F1 score combines both precision and recall metrics into a single metric and captures both properties the two metrics have. F1 score tells the score story neither precision and recall can not individually tell. Mathematically, f1 score is:

$$f1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

3.9.2: For the Detection Models

For the object detection task, the models were evaluated on mean average precision, mAP.

Chapter 4: Implementation

4.1: System Architecture

The flow chart diagram below shows a high-level overview of the research project. As shown in the flow chart, the application starts off by loading images from storage and preprocessing the images. The preprocessed images go through the training phase or testing phase. The details of the training and testing components are explained later.

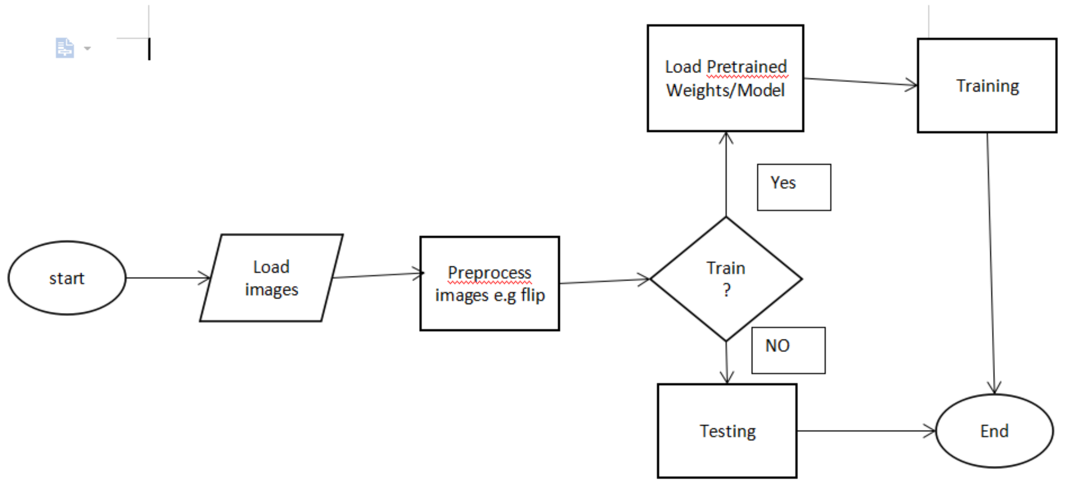

Figure 4.1: System Architecture

4.2: Resources

This section explains the resources used in this research. The models were built with the following technologies and packages: Pytorch, Google Collaboratory, LabelImg, and FloydHub.

4.2.1: PyTorch

Pytorch is a deep learning framework maintained by Facebook. Pytorch is known for its high-level tensor computations, as well as its ability to build a complex deep neural network with less code and effort. It is optimized for highly computationally expensive operations such as multiplication of a matrix with many dimensions, convolutional neural networks, recurrence neural networks, etc. Pytorch is pythonic, which makes it easy for people with python background to understand code written with the framework. The framework has many packages and pre-trained models that enhance the speedy training of models. It is a new framework compared to TensorFlow.

4.2.2: Google Collaboratory and Floyd Hub

Training deep learning models requires machines with very high computational power. The traditional personal computers lack this high power processing power requirement because most of them have only CPUs. Although deep learning models can be trained on computers with only CPUs, it could take longer time to train a slightly complex model. To be able to train models fast, traditional personal computers need Graphical Processing Units (GPUs). Since purchasing a GPU is costly, the cheapest option in training a deep learning model is to hire a cloud computer that comes to a GPU. In this project, google collaboratory platform and Floydhub were used to train the deep learning models. Google collaboratory was used when finding the hyperparameters, and the final training was done on Floydhub. Floyhub has preinstalled all the deep learning packages such as Tensorflow, Thano, Pytorch, Keras, and many more dependencies. Floyhub is also simple to use.

4.2.3: LabelImg

LabelImg is a graphical tool for annotating or creating bounding boxes in images. It saves the annotated images in XML format. The tool is simple and easy to use. It was used to create bounding boxes in the training images to experiment detection of FAW.

4.3: Training

All the models were trained on FloyHub cloud computers. Training CNNs involves a lot of matrix computation, which requires a high computational power machine. The smallest CNN model with only two convolutional blocks with 44,944 trainable parameters took 5 hours to train on a CPU machine. CPU executes instructions sequentially, making the training of CNN models from end to end longer. This neccessitates data and task parallelism where both loading the images and the high matrice computations are done in parallel. An interesting observation is that matrice multiplication is an element-wise operation and can be executed in parallel. Training on Graphical Processing Units (GPUs), designed for processing instructions in parallel, reduced the training time from 5 hours to 13 minutes.

Furthermore, the large number of trainable parameters in CNN is not the only limitation for training models on CPUs. The inputs of CNN models are images. For the most part, these images are in RGB formats with high resolutions. Such images form a matrix with very high dimensions. The large dimensionality, together with a large number of parameters, requires that Personal Computers (PCs) must have very large Random Access Memory. The standard PCs are designed in a way that the data required for executing the instructions must be loaded onto the RAM. Because of the design of PCs, it is impractical to train large models on them. Fortunately, the computational power and resources needed for deep learning tasks are accessible via cloud platforms such as Flodhub. All the models experimented in this research were trained on Floyhub's virtual machine, which was equipped with Tesla V100, 100GB storage, Compute Unified Architecture (CUDA) toolkits, and packages such as Pytorch, Tensorflow, etc.

The binary classification models were trained on 940 images and validated on 441 images. Each of the split contained two classes: positive (fall armyworm) and negative (not fall armyworm) classes. Because of the size of the data, the data were fed into the models in batches. The batch size used was 32. Each batch of data passed through the data augmentation step. The augmentation techniques were flip, rotation, and random crop. Performing this task generated more data samples out of the original ones and were used for training and evaluating the model.

4.3.1: Model 1 Architecture

This architecture used a ResNet50[11] pretrained model. However, the pretrained weights (imageNet weights) were not used for parameter initialization. It was initialized with a pre-trained weights from IP102[27] dataset. The architecture of the pretrained model was maintained, and only the last fully connected layer was changed from 1,000 to match the number of classes, 2, in the dataset for binary. The size of all the input images was resized to 224 x 224 x 3. When training the model, the learning rate was initialized as 0.0001 and dropped by a factor of 0.1 every seven epochs. The training and validation input data were supplied in min-batches of 32 images. Training the model on Tesla 100 took 1 hour, 23 minutes to complete the training.

4.3.2: Model 2 Architecture

This binary classification architecture used the ResNet50[11] model as a feature extractor. All the layers in this model, but the last one (classifier layer) was maintained. That is, the softmax layer was removed from the network leaving only the CNN portion. The CNN portion was used as a feature extractor. The feature extractor retrieved features from the input images. These features were flattened and passed to a linear Support Vector Machine (SVM).

Chapter 5: Experimental Results

In this research, there were a number of models built in an attempt to solve this problem. This chapter is devoted to the experiments that informed the final model selection and the findings of the research.

5.1: Binary Classification Results

There were two approaches adopted to train the models. The first approach was fine-tuning the last layers of a pre-trained RestNet50. The second approach was using the RestNet50 feature extractor and a linear SVM. The results for each model using the above-mentioned approaches are discussed below:

5.1.1: Model 1 Training Results

The model 1 architecture (discussed above) was trained for 25 epochs. After the 25th epochs, the gap between the validation and training accuracies curves was very wide as the training accuracy reached 99.7%, and validation accuracy is just 90%. Also, a similar thing was observed in the training and validation loss curves. (See figure 5.1). I increased the number of epochs to 75 and retrained the model. The gap decreased small as the training accuracy, and the validation accuracy reached 99.8% and 95.6%, respectively. In addition, the training loss significantly reduced to 0.002, but the validation loss did not reduce significantly. It was reduced to 0.90 (see figure 5.2. The number of training epochs was increased to 200. The validation accuracy and loss were still stacked at 95.6% and 0.90, respectively, regardless of the number of epochs. This resulted in the plot shown in figure 5.3. In principle, it turned out that the validation data did not provide enough information to evaluate the ability of the model to generalize. I added 294 images from the test split to the validation split and retrained the model. However, the addition of more data was futile. In spite of this unusual observation, this model, however, was still kept for testing.

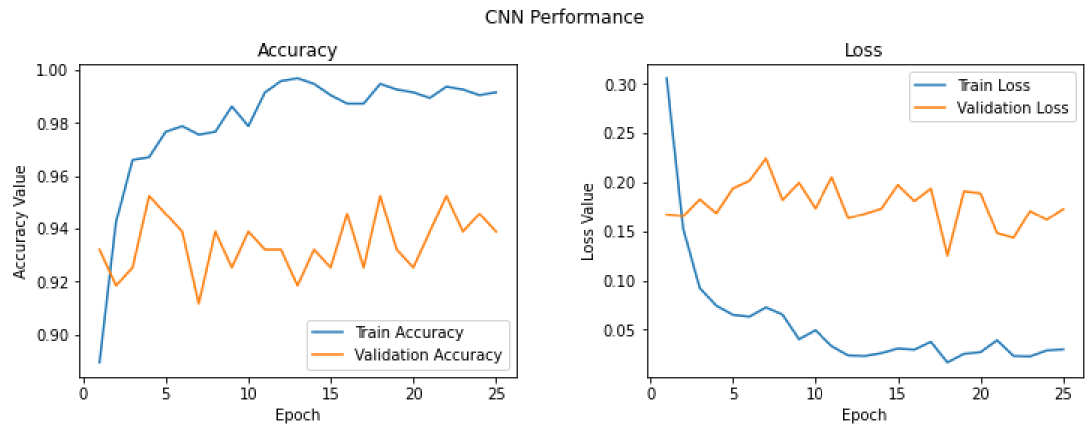

Figure 5.1: The first plot shows the training and validation accuracies of the model trained for 25 epochs. The second graph shows the training and validation losses

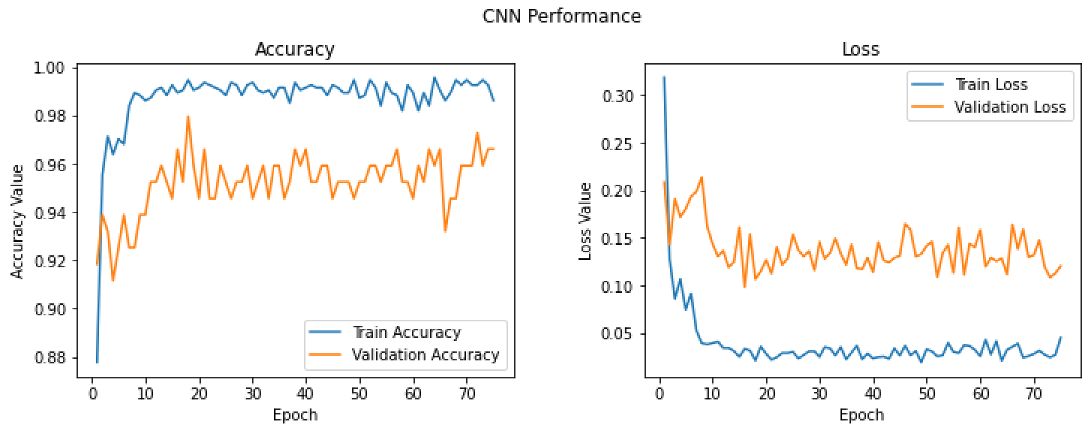

Figure 5.2: The first plot shows the training and validation accuracies of the model trained for 75 epochs. The second graph shows the training and validation losses trained for 75 epochs

5.1.2: Model 2 Training Results

Training this model was difficult as compared to the earlier one. The last layer of the convolutional network part of ResNet50[11], which is the average pooling layer, has an output size of 2048 for one image. For the most part, extracting features from the dataset (train/validation/test) means loading the data into memory, which is practically impossible given the size of the datasets. The training dataset alone contained 940 samples. At the end of the average pooling layer, there were 1,925,120 features that required at least 30GB RAM. Unfortunately, only 12GB RAM was available on my machine. The features were extracted in

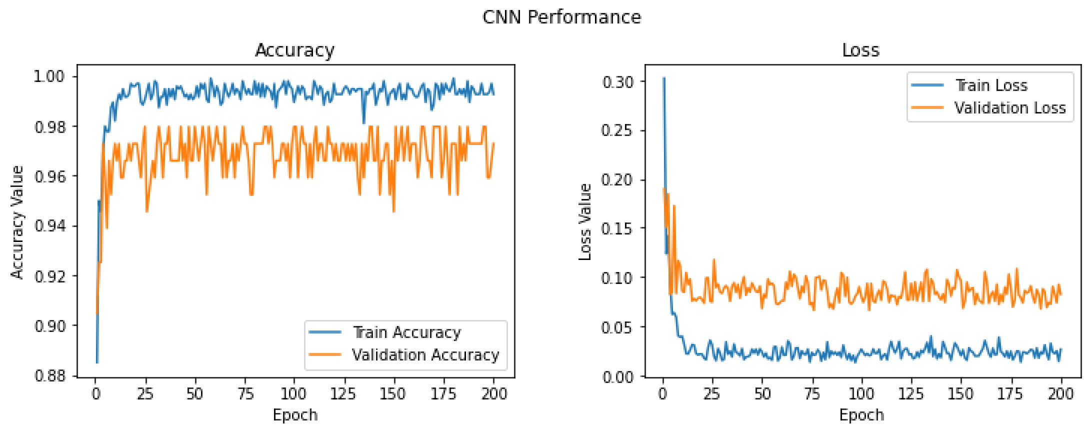

Figure 5.3: The first plot shows the training and validation accuracies of the model trained for 200 epochs. The second graph shows the training and validation losses trained for 200 epochs

mini-batches of 32 and saved in a CSV file. The classifier, LIBLINEAR[10], expects the data to be in a certain format. The CSV files were converted to the format the model expects. The time taken to train the LIBLINEAR[10] SVM classifier on the CPU machine with 16GB RAM was 7 minutes 45 seconds.

5.2: Multi-class Classification Training Results

The main reason for training the multiclass classification model was to ensure that the CNNs extract the detailed features of the pests. In the binary classification dataset, the positive class (Fall armyworm) contains images of the pest across all its life stages. There are images of FAW eggs, larva, pupa, and moth in one class. Intuitively, the FAW eggs, larva, pupa, and moth would all have different features hence splitting the dataset accordingly, and performing multiclass classification seems right. The multiclass architecture also uses the ResNet50 model as the base model. The training process of this model was similar to the training process of the CNN and softmax binary classification models. The only difference was the number of classes. Unlike the binary classifier, this model outputed four predictions representing the four life cycles of FAW. The multiclass classification model was trained on 584 images and validated on 115 images. The final training accuracy of the multiclass model was 99.2%, and the final validation accuracy was 96.6% after several iterations. The training and validation metrics (see figure 5.4) show that the model cannot evaluate unseen data when train on imbalanced data. The gap between the validation and training metrics is very wide. This is sign of overfitting the training data. While the training accuracy was at its optimal point, the validation curve stacked at 70 percent after the 20th epoch. The high accuracy achieved by the model on the training data explains that it is memorizing the training samples and can not generailize well on unseen data. This triggered the revision of the training process. A weighted random sampling was then applied to balance the dataset. The result is shown in figure 5.5. The training accuracy remained the same, but the validation accuracy increased. This means that if the model was trained on a large balanced dataset, it would yield better performance.

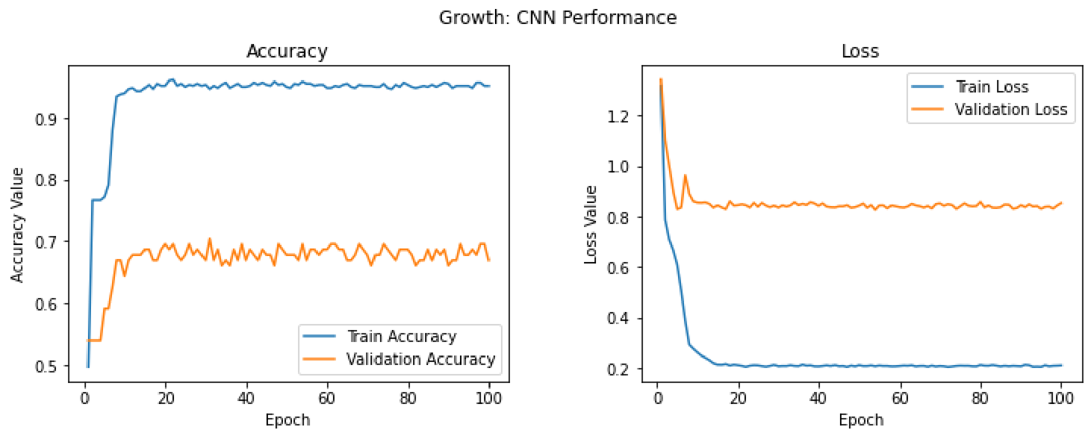

Figure 5.4: The first plot shows the training and validation accuracies of the model trained for 75 epochs on the data without the application of weighted random sampling. The second graph shows the training and validation losses trained for 100 epochs on the data without the application of weighted random sampling

5.3: Detection Model Training Results

The model was trained with 71 bounding box images and validated with 15 bounding box images. The images were loaded in min-batches of 2. They were randomly flipped and cropped. The training learning rate was set to 1e-3, and the model was trained for two epochs. As discussed in [14], the model was assigned the ground truth bounding boxes. Once they were assigned, the multi-box loss (discussed above) and hard negative mining (also discussed in [14])

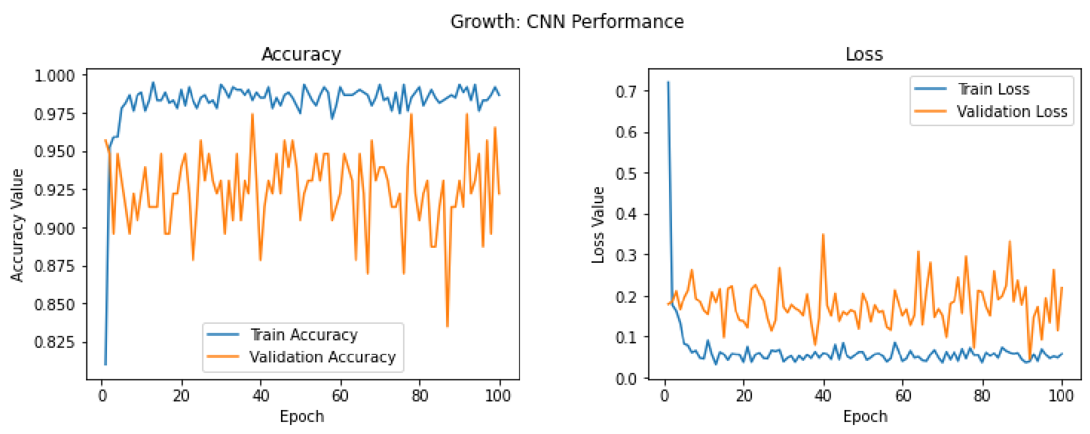

Figure 5.5: The first plot shows the training and validation accuracies of the model trained for 100 epochs on weighted random sampled data. The second graph shows the training and validation losses trained for 100 epochs on weighted random sampled data

were applied. Also, default bounding boxes, offset to find the location of the object of interest, were pre-computed [14]. Training the model for two epochs seemed to produce the best result since it was able to detect the location of the pest. This resulted in the figure below 5.6. It was observed that when the number of epochs was increased, the model performed poorly by predicting many overlapping bounding boxes. It turned out that many predicted boxes had Jaccard overlap more than the threshold (0.5) when the number of epochs is increased. The threshold was increased to 0.75, but the outcome was still the same. While the models were localizing the objects with an appreciable accuracy, it was observed that the models were all predicting one class (pupa) for all the objects. Despite this unusual behavior by the models, the slightly working model was saved for testing, because I ran out of time to revise it. I leave debugging it for future work. Training the final detection model took 1 hour 30 minutes, and validating it took 20 seconds.

5.4: Binary Classification Test Results

147 images were used to evaluate the binary classification models. The models were evaluated using f1 score, precision, and recall. Generally, mode 1 performed better in classifying unseen data than model 2. Out of the 147 images, model 1 classified 138 correctly and

Method	Acc	Prec	Rec	f1
CNN with Softmax (model 1)	0.943	0.956	0.952	0.954
CNN with SVM (Model 2)	0.879	0.907	0.897	0.902

Table 5.1: Binary classification models performance.

model 2 classified only 129 correctly. Table 5.1 summaries the classification performance of the two models under the specified evaluation metrics on the test dataset.

5.5: Multi-class Classification Test Results

The multiclass model was tested using 236 images. Like the training data, the test data were preprocessed. The test images were loaded in batches because of the size. They were re-scaled and converted to tensors. The data augmentation techniques discussed earlier were applied. The model showed relatively low performance on both the test data and the validation data. It achieved an accuracy of 0.87% on the test data. This means that out of the 236 of images, the model classified 205 images correctly. The model obtained a precision of 0.89. Though this value is high in the some task, it was rather considered low in this research. This is because of the nature of the FAW pest: how they quickly spread and destroy crops. Incorrect classification may result in a loss of thousands of cedis.

5.6: Detection Model Test Results

Images that were not seen by the model were randomly selected and passed to the final detection model. The model predicted the location(s) and class(es) of the object(s). The figures below show samples images predicted by the final model.

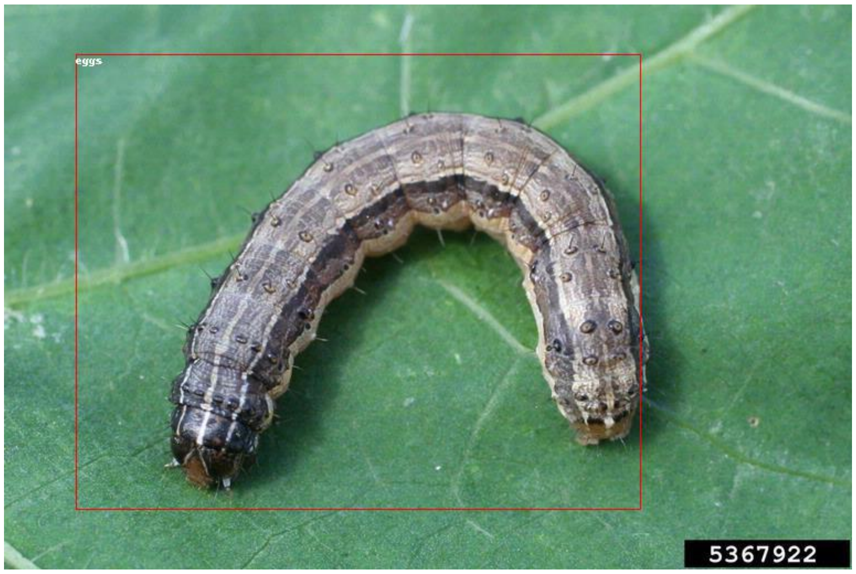

Figure 5.6: Detection model prediction 1. Although the model detected the location of the object, the class assigned to it was wrong.

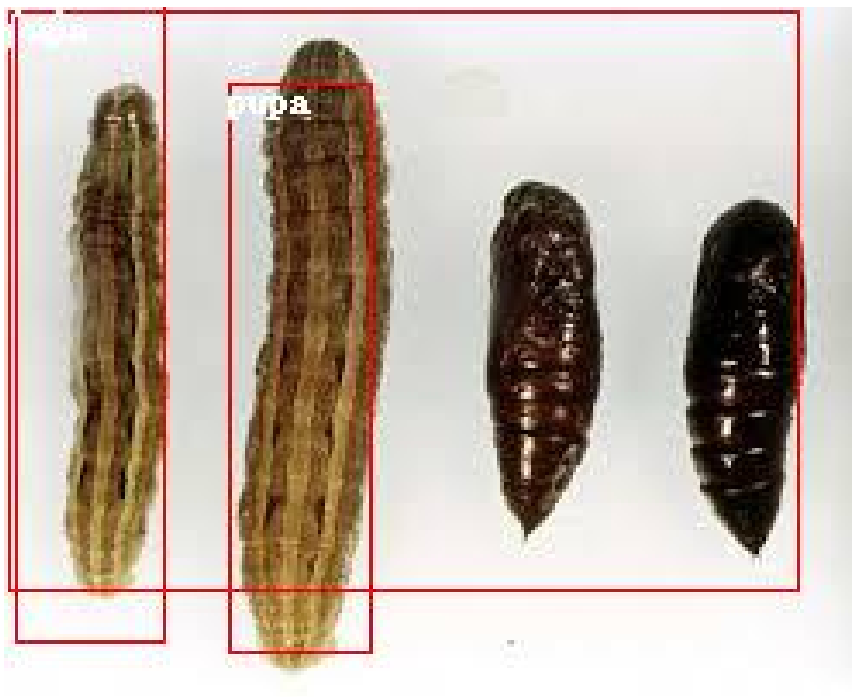

Figure 5.7: Detection model prediction 2.

Chapter 6: Conclusion and Future Work

6.1: Summary

The aim of this research was to experiment on how to use machine learning algorithms to detect Fall armyworms in Ghana. Binary classification, multiclass classification and detection models were built. In particular, the differences between CNN with softmax classifier and CNN with SVM in terms of, not just performance, but also implementation difficulty level were examined. Models were trained from scratch, but they performed woefully. By using Transfer learning and data augmentations, I could improve the performance.

6.2: Limitations

This section discusses the challenges encountered in the research that has a high potential of impacting the outcome of the results discussed in the paper.

1. The datasets that were used to experiment throughout the research were not sufficient. Given that machine learning techniques such as CNNs require large datasets to produce better performance, it is possible that the outcome of this research may have been hampered by the inadequacy of the data.

2. Apart from insufficient data, the limited data was highly imbalanced. Although the weighted random sampling technique was applied to alleviate this challenge, the technique introduced a "sufficient vs. balance" trade-off. In order to achieve balance, the technique down-sampled the class with a large number of samples.

3. The final limitation is that the pretrained model, ResNet50, was trained on unrelated datasets than the dataset used in this research. There was no publicly available FAW dataset. Also, I could not find any model pretrained on insect or pest dataset, which is closely related to my dataset and has a high potential of improving the performance obtained in this research.

6.3: Future work

This section suggests possible extensions of this work to researchers who are interested in using machine learning to classify and detect FAW. Below are the suggestions:

1. Gathering a large dataset to improve the results of the research. Data is very import in machine learning task. Without it, there is nothing a machine learning researcher can do. In this research, I was restricted by the dataset used. In the first place, the dataset was not collated for FAW tasks. Because one of the classes of the dataset was FAW, it was the only dataset suitable for this tasks since no public dataset was available at the time of this research. Future extension of this project should consider sourcing more data to detect the worm accurately.

2. Besides, researchers should also do extensive annotation and preprocessing of the data to extend this result.

References

[1] Afridi, M. J., Ross, A., and Shapiro, E. M. On automated source selection for transfer learning in convolutional neural networks. *Pattern recognition 73* (2018), 65–75.

[2] Chambers, J. Overview on stored-product insect pheromones and food attractants. *Journal of the Kansas Entomological Society* (1990), 490–499.

[3] Chiwamba, S. H., Phiri, J., Nkunika, P., Nyirenda, M., and Kabemba, M. M. An application of machine learning algorithms in automated identification and capturing of fall armyworm (faw) moths in the field. *ICICT2018, Lusaka, Zambai* (2018).

[4] Codling, E. Environmental impact and remediation of residual lead and arsenic pesticides in soil. In *Pesticides in the Modern World-Risks and Benefits*. IntechOpen, 2011.

[5] CS231n, S. Convolutional neural networks for visual recognition, 2017.

[6] Dasiopoulou, S., Mezaris, V., Kompatsiaris, I., Papastathis, V.-K., and Strintzis, M. G. Knowledge-assisted semantic video object detection. *IEEE Transactions on Circuits and Systems for Video Technology 15*, 10 (2005), 1210–1224.

[7] Day, R., Abrahams, P., Bateman, M., Beale, T., Clottey, V., Cock, M., Colmenarez, Y., Corniani, N., Early, R., Godwin, J., et al. Fall armyworm: impacts and implications for africa. *Outlooks on Pest Management 28*, 5 (2017), 196–201.

[8] Ding, W., and Taylor, G. Automatic moth detection from trap images for pest management. *Computers and Electronics in Agriculture 123* (2016), 17–28.

[9] Early, R., Gonzalez-Moreno, P., Murphy, S. T., and Day, R. Forecasting the global extent of invasion of the cereal pest spodoptera frugiperda, the fall armyworm.

[10] Fan, R., Chang, K., Hsieh, C., Wang, X., and Lin, C. Liblinear: A library for large linear classification journal of machine learning research 9.

[11] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[12] Kehat, M., Anshelevich, L., Dunkelblum, E., Fraishtat, P., and Greenberg, S. Sex pheromone traps for monitoring the codling moth: effect of dispenser type, field aging of dispenser, pheromone dose and type of trap on male captures. *Entomologia experimentalis et applicata 70*, 1 (1994), 55–62.

[13] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[14] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision* (2016), Springer, pp. 21–37.

[15] Madsen, H. F., and Vakenti, J. M. The influence of trap design on the response of codling moth (lepidoptera: Olethreutidae) and fruittree leafroller (lepidoptera: Tortricidae) to synthetic sex attractants. *Journal of the Entomological Society of British Columbia 70* (2019), 5–8.

[16] Millar, J. G., Daane, K. M., Steven Mcelfresh, J., Moreira, J. A., Malakar-Kuenen, R., Guillén, M., and Bentley, W. J. Development and optimization of methods for using sex pheromone for monitoring the mealybug planococcus ficus (homoptera: Pseudococcidae) in california vineyards. *Journal of Economic Entomology 95*, 4 (2002), 706–714.

[17] Peñas, K. E. D., Rivera, P. T., and Naval, P. C. Malaria parasite detection and species identification on thin blood smears using a convolutional neural network. In *2017*

IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE) (2017), IEEE, pp. 1–6.

[18] PRASANNA, B., HUESING, J., EDDY, R., AND PESCHKE, V. Fall armyworm in africa: a guide for integrated pest management.

[19] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99.

[20] ROELOFS, W., COMEAU, A., HILL, A., AND MILICEVIC, G. Sex attractant of the codling moth: characterization with electroantennogram technique. *Science 174*, 4006 (1971), 297–299.

[21] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., ET AL. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[22] SEVERYN, A., AND MOSCHITTI, A. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (2015), ACM, pp. 373–382.

[23] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[24] SIVARAMAKRISHNAN, R., ANTANI, S., AND JAEGER, S. Visualizing deep learning activations for improved malaria cell classification. In *Medical informatics and healthcare* (2017), pp. 40–47.

[25] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. Going deeper with convolutions. In *Pro-*

ceedings of the IEEE conference on computer vision and pattern recognition (2015), pp. 1–9.

[26] WALTER, H., JULIEN, G., AND DAY, R. Fall armyworm response in ghana: stakeholder workshop.

[27] WU, X., ZHAN, C., LAI, Y., CHENG, M.-M., AND YANG, J. Ip102: A large-scale benchmark dataset for insect pest recognition. In *IEEE CVPR* (2019), pp. 8787–8796.