Crop Pest Detection and Measures

Project Report

submitted by

HARRY A HAQUE

Reg. No. MAC20CD022

SHIBURAJ A.

Reg. No. LMAC20CD058

MUHAMMED BASHEER K.

Reg. No. LMAC20CD057

to

A P J Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering (Data Science)

Department of Computer Science & Engineering

Mar Athanasius College of Engineering

Kothamangalam, Kerala, India 686 666

JUNE 2023

Crop Pest Detection and Measures

Project Report

submitted by

HARRY A HAQUE

Reg. No. MAC20CD022

SHIBURAJ A.

Reg. No. LMAC20CD058

MUHAMMED BASHEER K.

Reg. No. LMAC20CD057

to

A P J Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering(Data Science)

Department of Computer Science & Engineering

Mar Athanasius College of Engineering

Kothamangalam, Kerala, India 686 666

JUNE 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAR ATHANASIUS COLLEGE OF ENGINEERING

KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **Crop Pest Detection and Measures** *submitted by* **Mr. Harry A Haque (MAC20CD022), Mr. Shiburaj A. (LMAC20CD058), Mr. Muhammed Basheer K. (LMAC20CD057)**, *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering (Data Science) from APJ Abdul Kalam Technological University for June 2023 is a bonafide record of the project carried out by them under our supervision and guidance.*

.....	.....	.....
Prof. Basil Joy	**Prof. Aby Abahai T**	**Prof. Joby George**
Project Guide	***Project Coordinator***	***Head of the Department***

Internals Examiner(s) External Examiner(s)

Date: Dept. Seal

ACKNOWLEDGEMENT

First and foremost, we sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the project.We express our sincere gratitude and thanks to Dr. Bos Mathew, Principal and Prof. Joby George, Head of the Department for providing the necessary facilities and their encouragement and support.

We owe special thanks to the project guide Prof. Basil Joy and project coordinator Prof. Aby Abahai T for their corrections, suggestions and sincere efforts to co-ordinate the project under a tight schedule.

We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in guiding and correcting us in conducting this project.

Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to us by our dear friends during the preparation of the project and also during the presentation without which this work would have been all the more difficult to accomplish.

ABSTRACT

Crop pests pose a significant threat to global agriculture, causing substantial economic, social, and environmental losses. Accurate pest identification is crucial for effective pest control strategies, yet existing approaches often face challenges in accuracy due to algorithmic complexity and limited data availability. Misclassification of pests can lead to detrimental consequences, including the improper use of pesticides, impacting agricultural yields and the environment. To address these challenges, we present DeepPestNet, an innovative end-to-end framework designed for precise pest recognition and classification. The proposed DeepPestNet model comprises 11 learnable layers, incorporating eight convolutional and three fully connected layers. Leveraging image rotation and augmentation techniques, this framework enhances dataset size and tests the generalizability of pest recognition. Evaluation on Deng's crops dataset demonstrates exceptional accuracy, achieving 100diverse crop pest classes. Comparative analysis against traditional pre-trained deep learning models further highlights the superior performance of DeepPestNet. Moreover, the framework's adaptability is showcased through its successful recognition, with 98.92accuracy, of nine distinct pest types in the widely used Kaggle Pest Dataset. The potential impact of DeepPestNet extends to providing immediate and precise aid to agricultural specialists and farmers, mitigating economic losses, and safeguarding crop yields by facilitating accurate pest identification. DeepPestNet stands as a promising solution, offering a robust and accurate automated system for pest recognition and classification in agriculture. Its potential application could significantly reduce the adverse effects of misclassified pests and improper pesticide use, thereby revolutionizing pest management strategies in the agricultural domain.

Table of Contents

List of Figures

List of Abbreviations

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
NB	Naive Bayesian
KNN	k-nearest neighbour

Chapter 1

Introduction

Crop pest detection is a critical application of technology in agriculture that leverages advanced techniques, particularly in the realm of computer vision and machine learning. With the global population steadily increasing, ensuring food security has become a paramount concern. However, the health and yield of crops face significant threats from various pests and diseases. Traditional methods of pest monitoring are often labor-intensive and time-consuming, making it challenging to detect and respond promptly to potential crop threats. In response to these challenges, modern technology offers innovative solutions through the development of crop pest detection systems.

The integration of computer vision allows for the automated analysis of visual data, such as images or videos of crops, to identify signs of pest infestations. Machine learning algorithms, particularly deep learning models, play a pivotal role in recognizing intricate patterns and anomalies associated with different pests. This technological synergy empowers farmers and agricultural practitioners to monitor vast expanses of crops efficiently, enabling early detection and timely intervention to mitigate potential damage.

Crop pest detection not only aids in pest management but also contributes to sustainable and precision agriculture. By accurately identifying and characterizing pest threats, farmers can optimize the use of pesticides, minimizing environmental impact and reducing costs. Additionally, these advanced systems facilitate a more targeted and proactive approach, ultimately enhancing crop yield, quality, and overall agricultural sustainability.

1.1 Project Outline

Pest control is critical for increasing agricultural output and food quality while reducing costs and increasing profits, which has recently become significant. Insect pests are one of the most common causes of agricultural damage worldwide. The mitigation of these losses could save a significant amount of the yield and increase agricultural profitability. Pest infesta- tions on harvests cause various diseases and harm harvests, resulting in low yields [2]. Insect damage to harvest regions such as rice, wheat, and beans is one of the major reasons driving yield losses. Insecticides and other bio control tech- niques should be used to minimize insect crop losses and reduce insect populations and avoid them from spreading over broad areas. Pesticides and chemicals have a big impact on pest control. They will, however, have several detrimental effects on human health and the environment.

In addition, because pest management procedures change depending on the pest species, identifying the insect is critical. The capacity to recognize and classify insects, to distinguish between the healthy and dangerous ones, is the first step in preventing crop damage caused by insect pests. However, due to the intricate anatomy of insects and the resemblance between various insect species, insect classification is a difficult undertaking.

Traditional ML algorithms, on the other hand, have some drawbacks. Traditional ML techniques have been shown to work well when the quantity of crop pest species is limited, but they become ineffective when various fea- tures are manually retrieved. They necessitate a further level of data preprocessing known as feature engineering, which is critical. In addition, its capacity to generalize across datasets is limited. Furthermore, their effectiveness depends on the available data.

The motivation behind the research study is that despite the wide range of studies on pest classification and recognition, there is still interest in developing high-accuracy automated systems for pest classification. Although a few studies on pest classification and recognition have recently been provided, this research subject remains underexplored. Transfer learn- ing (TL) of pre-trained

DL frameworks and support vector machines (SVM) is the most commonly utilized pest classi- fication and recognition method in extant research. However, the SVM machine learning algorithm takes longer to train with larger datasets. Overfitting and negative transfer are the most concerning limits in TL. For this purpose, we developed the DeepPestNet model for pest identification to address these concerns. The method comprises of four main phases:

1. Oversample the minority pest class data to balance the dataset using DeepSmote.

2. Enhance the pest dataset through augmentation to expand the training data and improve the model's learning capabilities.

3. The crop pest image are detected using the DeepPestNet object detection method.

4. The pest is classified using the DeepPestNet method.

Overall, this method provides an effective solution for crop pest detection and classification. By employing advanced image processing techniques and machine learning algorithms, the proposed method can accurately detect and classify the crop pest, enhancing the performance and reliability of Crop Pest Detection and Classification systems.

Chapter 2

Background

2.1 Data Oversampling

Data oversampling plays a pivotal role in mitigating the challenges posed by class imbalance within machine learning datasets, where one class is significantly underrepresented compared to others. This issue can result in biased models that excel in predicting the majority class but struggle with the minority class. Oversampling methods, including Random Oversampling and more sophisticated techniques like Synthetic Minority Over-sampling Technique (SMOTE) and Adaptive Synthetic Sampling (ADASYN), aim to rectify this imbalance by either replicating instances from the minority class or generating synthetic samples to augment its representation. However, the application of oversampling demands careful consideration of potential pitfalls.

There's a risk of overfitting, where the model may memorize the minority class instances instead of learning general patterns. Evaluating the model's performance on an independent validation set becomes crucial to ensure its ability to generalize to unseen data. The computational cost also increases with a larger dataset, impacting training time and resource requirements. Often used in conjunction with undersampling or different sampling ratios, oversampling requires a thoughtful approach. Metrics like precision, recall, F1 score, and area under the Receiver Operating Characteristic (ROC) curve become more informative in assessing model performance on imbalanced datasets.

Cross-validation further enhances the robustness of the evaluation process. The decision to employ oversampling depends on the dataset's characteristics

and the specific goals of the machine learning task, particularly in applications where the minority class is of particular interest. In summary, while oversampling is a valuable tool for addressing class imbalance, it necessitates a thoughtful application and consideration of potential challenges to ensure its efficacy in enhancing model performance.

2.2 Image Processing

The study area of image processing is concerned with the editing and evaluation of digital images. It is a multidisciplinary field that incorporates elements of engineering, mathematics, and computer science. To make photos easier to study, extract information from, and visualise, image processing techniques are applied.

Techniques for image processing are used in a variety of industries, including entertainment, security, and health. For instance, image processing techniques are used in medical imaging to improve images for diagnostic purposes, such as enhancing contrast or reducing noise. Face recognition and surveillance are two security-related applications of image processing methods. Visual effects in movies and video games are created using image processing techniques.

Picture enhancement, image restoration, and image analysis are just a few of the different categories that image processing techniques fall under. Techniques for image enhancement are used to enhance the aesthetic appeal of photographs by boosting contrast or sharpness, for example. Image degradation caused by things like noise or blur can be fixed using image restoration techniques. In order to extract information from photos, image analysis techniques are used, such as object recognition and image segmentation.

Overall, image processing is a strong tool for evaluating and manipulating digital images and has many uses in a variety of industries.

2.3 OpenCV

A free, open-source computer vision and machine learning software library is called OpenCV (Open Source Computer Vision Library). It provides a com-

prehensive set of computer vision algorithms, including image processing, object detection, and machine learning. OpenCV is written in C++ and can be used in various programming languages, such as Python, Java, and C.

OpenCV is widely used in various fields, such as robotics, medical imaging, and security, because of its flexibility, efficiency, and wide range of algorithms. It also has a large community of developers and users, which contributes to its growth and improvement. With OpenCV, developers can build and deploy computer vision applications quickly and easily.
The Dark Channel Prior algorithm is one of the image dehazing techniques offered by OpenCV. Instead of developing the code from scratch, OpenCV allows you to perform the dehazing process by calling functions from the library, which can save you a tonne of time and effort. The Dark Channel Prior technique is implemented in OpenCV, which is performance-optimized and capable of handling huge images.

In addition to the Dark Channel Prior, OpenCV also offers Anisotropic Diffusion and Fast Marching methods for image dehazing. Additionally, the library offers a range of image processing tools, including as filtering, morphological operations, and colour space conversion, which can be helpful during the dehazing process' pre-processing phase.

2.4 TensorFlow

A well-known open-source software library called TensorFlow is utilised in applications for machine learning and artificial intelligence. The Google Brain Team created it, and it was made available in 2015.

TensorFlow provides a framework for building and training machine learning models. It uses a dataflow graph to represent the model, where nodes in the graph represent mathematical operations, and edges represent the data being processed. This allows for efficient computation on large datasets using multiple CPUs or GPUs.

One of the key features of TensorFlow is its flexibility and scalability. It supports various programming languages, including Python, C++, and Java.

Predictive analytics, natural language processing, image and audio recognition, and other processes can all be done with TensorFlow.

TensorFlow also provides a wide range of pre-trained models and tools, which makes it easy for developers to get started with machine learning without requiring significant expertise in the field. Additionally, TensorFlow has a large and active community of developers, who contribute to the development of new models, algorithms, and applications.

Overall, TensorFlow is a powerful and versatile tool for machine learning and AI, which is widely used in academia, industry, and research institutions.

2.5 NumPy

The Python programming language's NumPy package is used for numerical computation and data processing. It offers a high-performance multidimensional array object and a selection of mathematical operations, such as linear algebra, statistical analysis, and other operations, that may be carried out on arrays.

A crucial component of the toolkit, NumPy arrays offer a quick and easy way to store and work with massive volumes of numerical data. Multi-dimensional arrays with a specific data type, such integers or floating-point numbers, are also possible. NumPy is a crucial tool for data analysis and scientific computing because it also has functions for reshaping, sorting, and manipulating arrays.

NumPy is widely used in the scientific computing community and has many libraries built on top of it, including Pandas, Matplotlib, and SciPy, making it a fundamental part of the scientific Python ecosystem. Its fast and efficient performance, combined with its ease of use, makes it a popular choice for data analysis and scientific computing.

2.6 Python

The scripting language Python is robust, interactive, object-oriented, and interpreted. Python was designed to be extremely readable. It uses English terms more often than punctuation and has fewer syntactical elements than other lan-

guages. Python is interpreted Python is processed when it is active by the interpreter. You don't need to build your software before launching it. This is comparable to PHP and PERL. Interactive Python In fact, you can sit down at a Python prompt and work directly with the interpreter to create your programmes. Python is an Object-Oriented language. Python supports the Object-Oriented programming paradigm, which encapsulates code inside objects.Python is a Great Language for Novice Programmers Python is a terrific language for novice programmers and facilitates the creation of a variety of programmes, including simple text editors, web browsers, and games. Among Python's features are Python is straightforward to learn because it has a defined syntax, minimal keywords, and an easy-to-understand structure. The pupil can swiftly pick up the language thanks to this. Python code is simpler to read since it is better defined and easier to see. Python's source code is relatively simple to maintain. A large standard library that is particularly portable and compatible with UNIX, Windows, and Macintosh platforms makes up the majority of Python's library.Python features an interactive mode that enables testing and debugging of code snippets as they are being executed. Python is extremely portable and provides the same user interface across all hardware platforms. The Python interpreter is extensible; you can add low-level modules to it. These modules give programmers the ability to enhance or modify their tools to make them more useful. Databases Python offers access to all significant commercial databases.

2.7 Keras

Keras is an open-source deep learning framework that provides a high-level interface for building neural networks. It was developed by Francois Chollet and released in 2015. Keras is built on top of low-level libraries like TensorFlow and Theano, which provide the computational backends for the framework.

One of the key features of Keras is its user-friendliness. Keras allows users to build neural networks quickly and easily, without requiring a deep understanding of the underlying mathematical operations. It provides a simple and intuitive API for building models, allowing developers to focus on the high-level design of

their networks.

Keras supports a wide range of neural network architectures, including feed-forward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. It also includes a variety of pre-trained models and tools for common tasks, such as image classification, text analysis, and time-series forecasting.

Another important feature of Keras is its flexibility. It supports both CPU and GPU computation, and can be used with a variety of programming languages, including Python and R. Keras also provides support for distributed computing, making it easy to scale up and train large models across multiple machines.

Overall, Keras is a powerful and user-friendly deep learning framework that provides a flexible and intuitive interface for building neural networks. It has become a popular choice for both beginners and experts in the field of deep learning.

Chapter 3

Related works

Recently, some research studies have been concentrated on pest classification and recognition. The pest classification and recognition research can be divided into ML, DL, and hybrid- based approaches. Hybrid methods include techniques that employ both DL and ML techniques. Many pest classification research works use DL-based techniques, whereas ML-based techniques are rarely used. Below, we highlighted the most recent and relevant research work in automatic pest identifi- cation and classification.

Recently, advanced ML-based techniques have effec- tively performed well in pest categorization and detec- tion [19]–[21]. Multiple classifiers are trained using extracted features from pests, and multiple types of pest images were categorized in these works. The UAV dataset was used to forecast armyworm contaminated and healthy corn regions, and the armyworm occurrence levels were then categorized. The best combination of image features for recognizing armyworm insects in corn-planted regions was discovered utilizing Gini-importance. The authors compared four types of ML methods: Random Forest (RF), Multilayer Perceptron (MLP), Naive Bayesian (NB), and SVM. The RF model performs the best compared to other ML approaches (MLP, NB, and SVM) for classifying the armyworm pest and normal corn. The authors [23] proposed an automatic pest detection (thrips) system based on SVM for greenhouse pest attack monitoring. Researchers used a novel image pro- cessing technique to detect parasites on strawberry plants. The SVM approach with a different kernel function was applied for parasite classification and thrips identification. The SVM structure was designed using the main diameter to minor diameter ratio as a region index,

Hue, Saturation, and Intensify as colour indices. The results demonstrate that utilizing the SVM method with area index and intensify as colour index produces the best classification, with a mean percent error of under 2.25 percent. In [24], the authors used two feature extraction techniques for the identification and categorization of tomato pests, namely, Histogram of Ori- ented Gradient (HOG) and Local Binary Pattern techniques (LBP). HOG outperforms its competitor, according to the comparison results. However, these ML-based pest recogni- tion systems rely on 'handcrafted' features extracted from the actual domain by comparing individual appearances. As a result, the empirical parameters must be manually changed to account for changes in image acquisition conditions. As a result, detecting pests in outside environments using hand- crafted techniques based on colour, structure, and texture remains a serious difficulty.

The importance of DL methods in computer vision has inspired researchers to utilize them for pest recognition and classification [25]–[27]. But unfortunately, DL algorithms in the domain of pest recognition have been constrained by a scarcity of pest image datasets and the inexplicability of DL frameworks. A novel and robust dataset for crop pest recog- nition was created in [15], and three different DL models were trained to employ TL and fine-tuning. The recognition rate of the three Architectures was greater than 80.00 per cent. Using the gradient-weighted class activation technique, the authors presented appropriate visual descriptions for the most crucial portions of the recognition layers. According to this study, the recognition process concentrates more on visual details than the entire image, and general differences are overlooked. An end-to-end pest detection system com- bining DL and hyperspectral imaging (HSI) techniques is presented in [16]. This technique can be used to quickly rec- ognize pests for successful pest control. To address noise and duplicate details in the HSI spectral space, one-dimensional convolution and attention techniques across spectral channels are employed to develop a spectral feature extraction unit to effectively use spectrum information. The HSI feature extractor secures rich spectral-spatial information using a three-dimensional convolution branch structure with various resolutions in parallel. The output feature map maintains its higher resolution throughout its use. Each branch contains an adjustable spectral-spatial feature

extraction unit that dynam- ically weights different inputs, limiting the HSI's dispropor- tionate effect and improving the network's feature extraction skills. Pest HSI was acquired utilizing hyperspectral imaging equipment, resulting in a dataset containing nine different pests.

Furthermore, it is commonly known that the hybrid models (DL and ML models) produce better classification results, which can be used to classify insects. In [28], DL models (TL technique) were used to classify eight different types of tomato pests. Using DL models, the extracted pest features were merged with three ML classifiers, i.e., discriminant analysis (DA), SVM, and the k-nearest neighbour approach (KNN). Bayesian optimization was used to effectively tune hyper-parameters. Following image augmentation, the VGG16 framework performed better than the other algo- rithms. The ResNet50 with discriminant analysis classifier had the best accuracy among the CNN and ML frameworks. The authors [29] developed a new DL model TPest-RCNN for pest detection. The faster regional-convolutional neural network is used as a based model in the proposed model. Moreover, VGG16 was used for feature extraction. Then, a region of tiny pests was generated by a region proposal framework. Finally, extracted features and regions of small pests are fed to RoIAlign for classification and detection.

Following a review of prior research, it is discovered that DL models, particularly those used to categorize crop pests, cannot achieve superior identification and classifica- tion performance. Instead of using real-world scenes, most models were trained and tested using images captured in highly controlled lab environments. However, in the field, the complicated surroundings, various viewpoints and postures, varying degrees of colour and texture alteration, changes in lighting conditions, and different locations of pests' wings and limbs constitute a considerable barrier to pest recog- nition. Although few studies addressed automatic detection in natural settings (testing is performed on natural images), most of them focused on a single species or used only one dataset for validation purposes. Detecting pests effectively and quickly and extracting traits independent of viewpoint, scale, and lighting conditions are critical for crop pest recog- nition. This paper developed an effective DL framework for identifying and classifying crop

pests into ten different classes. Also, the data set size is boosted using image rotation and data augmentation techniques to obtain generalized results. To test the generalizability of the proposed approach, we tested it on a different dataset with 9 different types of crop pests.

Chapter 4

Implementation

4.1 Flowchart of Deepsmote

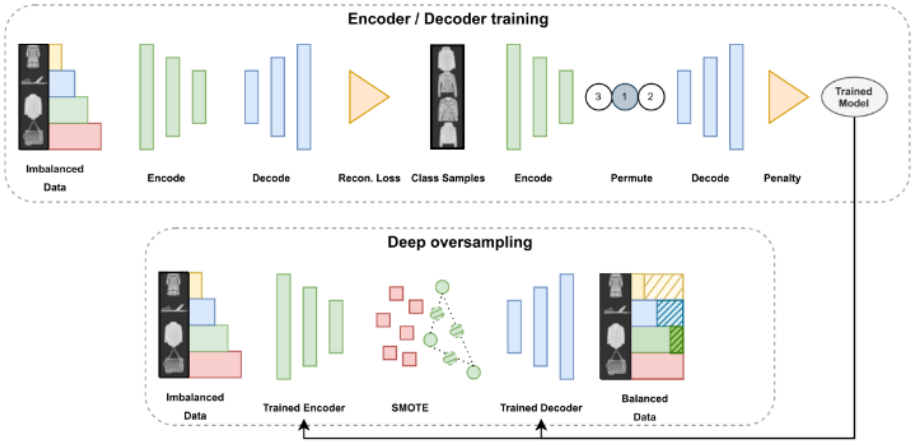

Figure 4.1: Flowchart representing the workflow of the Deepsmote

The main steps in the proposed solution is as follows:

1. Train Encoder/Decoder

   Feed imbalanced dataset in batches to encoder/decoder Calculate reconstruction loss between input and decoded output Sample images from a random class, encode and shuffle order Decode shuffled images and calculate penalty loss Jointly minimize reconstruction and penalty losses

2. Minority Class Oversampling

   For each minority class, select examples and encode them Apply SMOTE in the embedded space to generate new minority samples Decode the synthetic minority samples.

3. Dataset Augmentation

   Combine the oversampled minority class data with the original imbalanced dataset Use the balanced dataset to train the classifier (Resnet in the paper)

4.2 Dataset

To assess the performance of the PestDetNet framework, the evaluation utilized a dataset sourced from Deng et al. (2018), comprising images representing ten distinct pest categories commonly found in tea plants and other vegetation across Europe and Central Asia. The dataset encompasses pests such as Locusta migratoria, Euproctis pseudoconspersa Strand, Chrysochus Chinensis, Empoasca flavescens, Spodoptera exigua, larva of Laspeyresia pomonella, Parasa lepida, Acrida cinerea, larva of S. exigua, and L. pomonella. With a total of 562 pest images, each category consists of 40 to 70 images, providing a diverse set for evaluation. Notably, the images underwent rotations of 90 degrees twice, introducing variations in orientation. The dataset's compilation involved gathering images from Mendeley and various online sources, encompassing those captured with a Single Lens Reflex camera (SLR) and others obtained from platforms like Insert Images, IPM Images, and Dave's Garden. These RGB images exhibit disparities in resolution, size, posture, angle, lighting conditions, and backgrounds, collectively contributing to a comprehensive evaluation of the PestDetNet framework's robustness and adaptability to diverse environmental conditions.

4.3 DeepPestNet

DeepPestNet is a Deep Learning (DL) model proposed for the recognition and classification of crop pests in the field of image processing and computer science. The model employs eleven layers, including eight convolutional layers and three fully connected layers. It addresses the challenge of limited training data through dynamic data augmentations during each training phase. The architecture includes image resizing, dataset partitioning, and a 10-way softmax layer for classification. With a focus on efficiency, the model utilizes hyperparameter tuning and stochastic gradient descent during 80 training epochs, considering

overfitting prevention measures like dropout layers. The proposed DeepPestNet aims to provide an effective solution for pest recognition and classification, demonstrating its versatility on datasets such as Deng et al.'s and Kaggle's "Pest Dataset."

The DeepPestNet steps for detecting pests are as follows:

1. Data Augmentation: To overcome the challenge of limited data, image rotations, translations, and other augmentation techniques were applied. Data augmentation was done dynamically during each training phase using the imageDataAugmenter function. Augmented images were used exclusively for training, not testing.

2. . Image Resizing: Input images were resized to 224x224 pixels to ensure uniformity and expedite processing, meeting the model's input requirements.

3. Dataset Partitioning: The dataset was split into training (90

4. DeepPestNet Architecture Details: The model architecture includes convolutional layers, leaky relu (LR) layers, relu layer, maximum pooling layers, batch normalization (BN) layers, dropout layers, average pooling layers, softmax layer, and a classification layer. The first convolutional layer used 64 kernels of size 7x7. LR activation functions and max-pooling layers were employed after convolutional layers to enhance efficiency and reduce overfitting. The architecture ends with a 10-way softmax for the ten classes.

5. Hyperparameters: The study recognized the importance of hyperparameters in DL frameworks. Hyperparameters were determined through a trial-and-error method. Stochastic Gradient Descent (SGD) was used for training, with 80 epochs considered for pest recognition and classification, while mitigating overfitting.

6. Testing: Run the model on some test photos to see how it performs and where it needs work.

Chapter 5
Evaluation and Analysis

5.1 Average Class Specific Accuracy (ACSA)

ACSA is a performance metric designed specifically for evaluating classifiers on imbalanced datasets. It calculates the average of the true positive rates/recalls obtained for each class.This gives equal emphasis to minority and majority classes rather than focusing only on majority class accuracy. It avoids inflated performance measures when there is class imbalance. Values range from 0 to 1, with higher values indicating better ability of the classifier to handle imbalance and accurately classify minority cases. Being an average, all classes impact final score equally regardless of class distribution. In summary, ACSA gives a class-specific perspective rather than just overall accuracy, making it suitable for imbalanced evaluation where all classes need equal attention, especially the minority ones.

5.2 Macro-averaged Geometric Mean (GM)

GM calculates the geometric mean of the true positive rates/recalls obtained for each class.Takes a harsher penalty for low scoring classes compared to arithmetic mean.Ranges from 0 to 1, with higher values indicating better performance across all classes.Gives equal weight to each class regardless of class distribution.Avoids inflated performance estimates for imbalanced datasets.

5.3 Macro-averaged F1 Measure (FM)

Calculates F1 score (harmonic mean of precision and recall) for each class first.Then takes the average of the F1 scores to get the macro-averaged F1.Ranges from 0 to 1, with higher values indicating better balance of precision and recall.Minority class performance affects the score as much as majority class.Avoid overestimates on imbalanced data by using macro-averaging.

Chapter 6

Results

We provide a detailed description of the findings of numerous studies conducted to determine the efficacy of our PestDetNet model. This section also includes more information regarding the dataset used in this study. Deng 1018 is used to evaluate the performance of our technique. We also tested our model's performance using the publicly available "Pest Dataset" to ensure that it is generalizable.

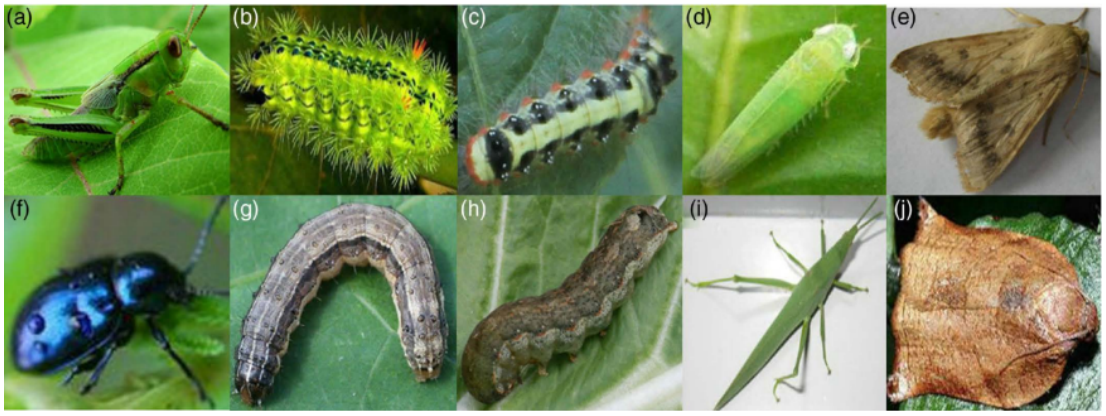

Figure 6.1: Sample images from Deng et al. dataset (a) Locusta migratoria, (b) Parasa lepida, (c) Gypsy moth larva, (d) Empoasca flavescens, (e) Spodoptera exigua, (f) Chrysochus chinensis, (g) Laspeyresia pomonella larva, (h) Spodoptera exigua larva, (i) Atractomorpha sinensis, (j) Laspeyresia pomonella.

To assess the performance of the proposed PestDetNet frame- work, we used the dataset presented in Deng et al. (2018). It comprises ten distinct pest categories primarily seen in tea plants and other plants throughout Europe and Central Asia.The total number of pests images in the Deng data is 562. Each

class of pests consists of 40 to 70 images. The number of images against each pest's category is shown in Table 3. Fig. 3 shows sample images after rotations. We rotated the images of the dataset twice by 90 degrees. The number of images against each pest's category after rotation is also mentioned in Table 3. The pest images in the dataset were gathered from Mendeley and other online sources and include pest images acquired with a Single Lens Reflex camera (SLR). The rest of the images were from Insert Images, IPM Images, Dave's Garden, and others. The dataset contains RGB images of different resolutions. The size, posture, angle, lighting conditions, and backgrounds of the sample images vary greatly.

We propose DeepSMOTE, a novel and breakthrough over- sampling algorithm dedicated to enhancing deep learning models and countering the learning bias caused by imbalanced classes. As discussed above, oversampling is a proven tech- nique for combating class imbalance; however, it has tradition- ally been used with classical machine learning models. Several We propose DeepSMOTE, a novel and breakthrough over- sampling algorithm dedicated to enhancing deep learning models and countering the learning bias caused by imbalanced classes. As discussed above, oversampling is a proven tech- nique for combating class imbalance; however, it has tradition- ally been used with classical machine learning models. Several attempts have been made to extend oversampling methods, such as SMOTE, to deep learning models, although the results have been mixed [84]–[86]. In order for an oversampling method to be successfully applied to deep learning models, we believe that it should meet three essential criteria. 1) It should operate in an end-to-end manner by accepting raw input, such as images (i.e., similar to VAEs, WAEs, and GANs). 2) It should learn a representation of the raw data and embed the data into a lower dimensional feature space, which can be used for oversampling. 3) It should readily generate output (e.g., images) that can be visually inspected, without extensive manipulation.

We show through our design steps and experimental evalu- ation that DeepSMOTE meets these criteria. In addition, it is capable of generating high-quality, sharp, and information-rich images without the need for a discriminator network.

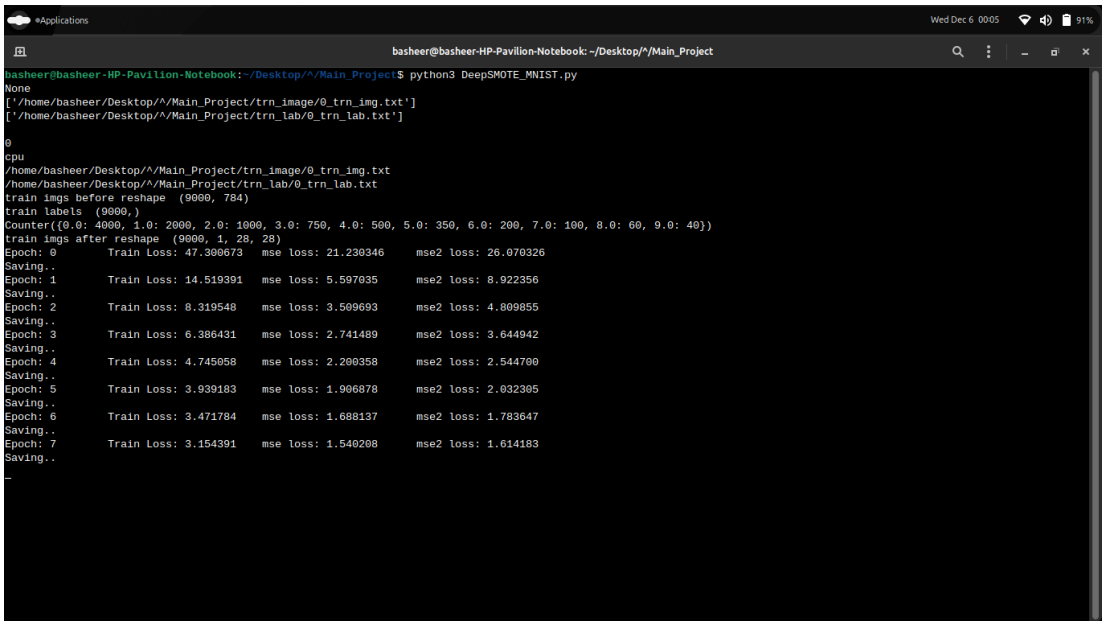

Figure 6.2: Deepsmote encoder-decoder model training

Chapter 7

Conclusion

In conclusion, this project has presented the DeepPestNet framework as a groundbreaking solution for the automatic recognition and classification of pests, addressing the global challenge of agricultural and financial losses caused by contamination. The remarkable achievement of 100

The study's versatility was demonstrated through experimental validations on various datasets, including the Pest Dataset from Kaggle, as well as medical datasets such as Breast Grading Carcinoma and Laryngeal datasets. This not only affirms the effectiveness of DeepPestNet in the agricultural domain but also highlights its adaptability to diverse applications.

An additional strength of this research lies in the incorporation of DeepSMOTE for data oversampling, a crucial element for addressing the challenges associated with limited training data. The integration of DeepSMOTE enhances the robustness of DeepPestNet, allowing for improved generalization and performance, especially in scenarios where data imbalances pose a significant concern.

Chapter 8

ANNEXURE

DeepSMOTEMNIST.py

```
# -*- coding: utf-8 -*-
#
# import collections
# import torch
# import torch.nn as nn
# from torch.utils.data import TensorDataset
# import numpy as np
# from sklearn.neighbors import NearestNeighbors
# import time
# import os
#
# print(torch.version.cuda) #10.1
# t3 = time.time()
# ##############################################################################
# """args for AE"""
#
# args = {}
# args['dim_h'] = 64         # factor controlling size of hidden layers
# args['n_channel'] = 1#3    # number of channels in the input data
#
```

```
# args['n_z'] = 300 #600     # number of dimensions in latent space.
#
# args['sigma'] = 1.0        # variance in n_z
# args['lambda'] = 0.01      # hyper param for weight of discriminator loss
# args['lr'] = 0.0002        # learning rate for Adam optimizer .000
# args['epochs'] = 200       # how many epochs to run for
# args['batch_size'] = 100   # batch size for SGD
# args['save'] = True        # save weights at each epoch of training if True
# args['train'] = True       # train networks if True, else load networks from
#
# args['dataset'] = 'mnist'  #'fmnist' # specify which dataset to use
#
#
# ##############################################################################
#
#
#
# ## create encoder model and decoder model
# class Encoder(nn.Module):
#     def __init__(self, args):
#         super(Encoder, self).__init__()
#
#         self.n_channel = args['n_channel']
#         self.dim_h = args['dim_h']
#         self.n_z = args['n_z']
#
#         # convolutional filters, work excellent with image data
#         self.conv = nn.Sequential(
#             nn.Conv2d(self.n_channel, self.dim_h, 4, 2, 1, bias=False),
#             #nn.ReLU(True),
#             nn.LeakyReLU(0.2, inplace=True),
#             nn.Conv2d(self.dim_h, self.dim_h * 2, 4, 2, 1, bias=False),
```

```
#             nn.BatchNorm2d(self.dim_h * 2),
#             #nn.ReLU(True),
#             nn.LeakyReLU(0.2, inplace=True),
#             nn.Conv2d(self.dim_h * 2, self.dim_h * 4, 4, 2, 1, bias=False),
#             nn.BatchNorm2d(self.dim_h * 4),
#             #nn.ReLU(True),
#             nn.LeakyReLU(0.2, inplace=True),
#
#
#             nn.Conv2d(self.dim_h * 4, self.dim_h * 8, 4, 2, 1, bias=False),
#
#             #3d and 32 by 32
#             #nn.Conv2d(self.dim_h * 4, self.dim_h * 8, 4, 1, 0, bias=False),
#
#             nn.BatchNorm2d(self.dim_h * 8), # 40 X 8 = 320
#             #nn.ReLU(True),
#             nn.LeakyReLU(0.2, inplace=True) )#,
#             #nn.Conv2d(self.dim_h * 8, 1, 2, 1, 0, bias=False))
#             #nn.Conv2d(self.dim_h * 8, 1, 4, 1, 0, bias=False))
#         # final layer is fully connected
#         self.fc = nn.Linear(self.dim_h * (2 ** 3), self.n_z)
#
#
#     def forward(self, x):
#         #print('enc')
#         #print('input ',x.size()) #torch.Size([100, 3,32,32])
#         x = self.conv(x)
#
#         x = x.squeeze()
#         #print('aft squeeze ',x.size()) #torch.Size([128, 320])
#         #aft squeeze  torch.Size([100, 320])
#         x = self.fc(x)
```

```
#         #print('out ',x.size()) #torch.Size([128, 20])
#         #out  torch.Size([100, 300])
#         return x
#
#
# class Decoder(nn.Module):
#     def __init__(self, args):
#         super(Decoder, self).__init__()
#
#         self.n_channel = args['n_channel']
#         self.dim_h = args['dim_h']
#         self.n_z = args['n_z']
#
#         # first layer is fully connected
#         self.fc = nn.Sequential(
#             nn.Linear(self.n_z, self.dim_h * 8 * 7 * 7),
#             nn.ReLU())
#
#         # deconvolutional filters, essentially inverse of convolutional filters
#         self.deconv = nn.Sequential(
#             nn.ConvTranspose2d(self.dim_h * 8, self.dim_h * 4, 4),
#             nn.BatchNorm2d(self.dim_h * 4),
#             nn.ReLU(True),
#             nn.ConvTranspose2d(self.dim_h * 4, self.dim_h * 2, 4),
#             nn.BatchNorm2d(self.dim_h * 2),
#             nn.ReLU(True),
#             nn.ConvTranspose2d(self.dim_h * 2, 1, 4, stride=2),
#             #nn.Sigmoid())
#             nn.Tanh())
#
#     def forward(self, x):
#         #print('dec')
```

```
#         #print('input ',x.size())
#         x = self.fc(x)
#         x = x.view(-1, self.dim_h * 8, 7, 7)
#         x = self.deconv(x)
#         return x
#
# ##############################################################################
# """set models, loss functions"""
# # control which parameters are frozen / free for optimization
# def free_params(module: nn.Module):
#     for p in module.parameters():
#         p.requires_grad = True
#
# def frozen_params(module: nn.Module):
#     for p in module.parameters():
#         p.requires_grad = False
#
#
# ##############################################################################
# """functions to create SMOTE images"""
#
# def biased_get_class(c):
#
#     xbeg = dec_x[dec_y == c]
#     ybeg = dec_y[dec_y == c]
#
#     return xbeg, ybeg
#     #return xclass, yclass
#
#
# def G_SM(X, y,n_to_sample,cl):
#
```

```
#     # determining the number of samples to generate
#     #n_to_sample = 10
#
#     # fitting the model
#     n_neigh = 5 + 1
#     nn = NearestNeighbors(n_neighbors=n_neigh, n_jobs=1)
#     nn.fit(X)
#     dist, ind = nn.kneighbors(X)
#
#     # generating samples
#     base_indices = np.random.choice(list(range(len(X))),n_to_sample)
#     neighbor_indices = np.random.choice(list(range(1, n_neigh)),n_to_sample)
#
#     X_base = X[base_indices]
#     X_neighbor = X[ind[base_indices, neighbor_indices]]
#
#     samples = X_base + np.multiply(np.random.rand(n_to_sample,1),
#             X_neighbor - X_base)
#
#     #use 10 as label because 0 to 9 real classes and 1 fake/smoted = 10
#     return samples, [cl]*n_to_sample
```

REFERENCES

[1] Naeem Ullah, Javed Ali Khan, Lubna Abdulaziz Alharbi, Asaf Raza, Wahab Khan, Ijaz Ahmad, "An Efficient Approach for Crops Pests Recognition and Classification Based on Novel DeepPestNet Deep Learning Model" , Volume 10, may 2022

[2] Adwan A. Alanazi a, Alkhansa A. Shakeabubakor b, Sayed Abdel-Khalek c, Salem Alkhalaf d, "IoT enhanced metaheuristics with deep transfer learning based robust crop pest recognition and classification" , Volume 84,December 2023.

[3] Thenmozhi Kasinathan, "Insect classification and detection in field crops using modern machine learning techniques" ,September 2021

[4] "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data" Damien Dablain , Bartosz Krawczyk , Member, IEEE, and Nitesh V. Chawla , Fellow, IEEE