Deep CNN-Based Planthopper Classification Using a High-Density Image Dataset

Mohd Firdaus Ibrahim 1,2, Siti Khairunniza-Bejo 1,3,4,*, Marsyita Hanafi 5, Mahirah Jahari 1,3, Fathinul Syahir Ahmad Saad 6 and Mohammad Aufa Mhd Bookeri 7

Abstract: Rice serves as the primary food source for nearly half of the global population, with Asia accounting for approximately 90% of rice production worldwide. However, rice farming faces significant losses due to pest attacks. To prevent pest infestations, it is crucial to apply appropriate pesticides specific to the type of pest in the field. Traditionally, pest identification and counting have been performed manually using sticky light traps, but this process is time-consuming. In this study, a machine vision system was developed using a dataset of 7328 high-density images (1229 pixels per centimetre) of planthoppers collected in the field using sticky light traps. The dataset included four planthopper classes: brown planthopper (BPH), green leafhopper (GLH), white-backed planthopper (WBPH), and zigzag leafhopper (ZIGZAG). Five deep CNN models—ResNet-50, ResNet-101, ResNet-152, VGG-16, and VGG-19—were applied and tuned to classify the planthopper species. The experimental results indicated that the ResNet-50 model performed the best overall, achieving average values of 97.28% for accuracy, 92.05% for precision, 94.47% for recall, and 93.07% for the F1-score. In conclusion, this study successfully classified planthopper classes with excellent performance by utilising deep CNN architectures on a high-density image dataset. This capability has the potential to serve as a tool for classifying and counting planthopper samples collected using light traps.

Keywords: planthoppers; convolutional neural network; machine vision; paddy cultivation

1. Introduction

Since 1910, rice has been a staple in the daily diet of Malaysians, either consumed directly as cooked rice or in the form of flour derived from the milling process [1]. Malaysians heavily rely on rice as their primary source of nutrition, consuming an average of 80 kg per person, which contributes to 26% of their daily caloric intake [2]. This indicates national consumption of approximately 2.7 million tons of rice per year. Despite local production, the quantity falls short of meeting demand, leading the Malaysian government to import approximately 30% of its rice from countries such as Thailand, India, Vietnam, and Pakistan [3]. Revision of policies and implementation of measures are necessary to ensure food security concerning rice.
Threats posed by planthopper insects have been a significant issue affecting rice production in Malaysia [4]. The Malaysian Agricultural Research and Development Institute (MARDI) has identified four major planthopper species in the country: the brown planthopper (*Nilaparvata lugens* (Stal)), the green leafhopper (*Nephotettix malayanus*), the white-backed planthopper (*Sogatella furcifera* (Harvath)), and the zigzag leafhopper (*Recilia dorsalis* (Motschulsky)). To prevent pest outbreaks, pesticides are commonly applied in the fields. However, excessive pesticide use can have negative impacts on both plants and the environment. These effects include the development of pesticide resistance by the pests and the presence of high pesticide residue concentrations in rivers [5]. Therefore, effective pesticide management strategies need to be implemented, with early detection of pest outbreaks playing a crucial role in the process.

In pest control, monitoring the occurrence pattern of pests plays a crucial role [6]. MARDI has developed a manual identification and counting process conducted by trained experts. To facilitate this process, they have designed a solar-powered light trap system specifically for capturing pests during nighttime [7]. The system consists of an LED light shielded by a transparent plastic sheet, approximately the size of A3 paper. The plastic sheet is coated with sticky glue to trap the pests. Since flying insects are attracted to light due to positive phototaxis, they are drawn toward the light source. The sticky sheet is enclosed within a transparent box that contains multiple small holes with a diameter of 5 mm. These holes prevent larger flying insects from becoming trapped in the sticky sheet. The collection of the sticky light trap is carried out the following day, and the trapped insects are manually counted by the experts. However, this counting process can be time-consuming, taking up to 6 h for a single light trap. Furthermore, the accuracy and efficiency of the manual counting process may be affected by factors such as fatigue and emotional aspects of the inspector. Consequently, applying this manual process on a large scale is challenging due to limitations in the counting procedure.

Several research studies have focused on utilising machine vision for pest identification and classification. One technique proposed for image analysis incorporates scene interpretation [8]. An automatic detection system for harmful insects inside the greenhouse has been developed using three feature extraction methods: the pyramidal histogram of gradient, Gabor filter, and colour data. The system utilised a support vector machine (SVM) for accurate prediction of whiteflies (1283 samples) with a 98.5% accuracy rate, as well as greenflies (49 samples) with a 91.8% accuracy rate. To capture the images of pests, a pan-tilt camera was employed as the acquisition device [9]. The study employed a centralised server to process and analyse the recorded field video. Images were extracted from the video on a frame-by-frame basis, and the SVM classification was performed individually for each frame.

Convolutional neural networks (CNNs) have recently been applied in pest classification. CNNs are a class of artificial neural networks (ANNs) that employ deep learning architecture and are commonly used for visual image classification [10–18] and detection [19–24]. The architecture of a CNN consists of an input layer, hidden layers, and an output layer as the final layer. The hidden layers include various types of layers, such as convolutional layers, pooling layers, and dense (also known as fully connected) layers. During the convolutional phase, the image is transformed into a feature map, also known as an activation map, with a specific shape. Convolutional layers combine the input with the output before passing it to the next layer. Global pooling layers may be incorporated to reduce the dimensionality of the data throughout the convolutional phase. The output of a cluster of neurons is aggregated at one layer and then transmitted to a single neuron in the subsequent layer. The dense layers establish connections between every neuron in one layer and every neuron in the next layer.

Although the application of CNN for planthopper classification has not been widely explored, its performance in classifying other pests has been proven. For instance, CNN was used to detect wheat sawfly, wheat mite, and wheat aphid in the paddy fields of Anhui province, China, achieving an accuracy of up to 90.88% for wheat sawfly detection and
a minimum accuracy of 70.2% for wheat mite detection [25]. In another study, VGG-19 was employed to train on 4800 images of 24 types of insects, achieving a mean average precision (mAP) of 0.8922 with a training time of 70 h [26]. Among other examples, moth [19,25], oilseed rape pest [26], bark beetles [20], forest pest [27], citrus pest [28], and rice pest [7,29–31] are a few other insects and pests that have been studied for classification using CNN. Ref. [32] used a CNN combined with a Euclidean distance map (EDM) to automatically recognise the brown planthopper captured on a sticky pad. Their method demonstrated successful results with 95% accuracy in identifying the BPH. However, the dataset used in the research was relatively small, comprising only 1374 samples. As imperfect planthoppers exhibit more variations, the addition of new samples may lead to misclassification. Moreover, the research only focused on differentiating between BPH and benign insects.

Therefore, to address the research gap, this study proposes a novel method for classifying four types of planthoppers using five different deep convolutional neural networks and a large dataset. Planthopper images were cropped from the full image captured by the light trap, resulting in a total of 7328 planthopper images. These images were then divided into training, validation, and test sets with a ratio of 80:10:10. Augmentation techniques were applied to the training dataset to create a larger dataset, resulting in a total of 187,456 samples for the training dataset. Subsequently, the dataset was trained using five different CNN architectures, namely ResNet-50, ResNet-101, ResNet-152, VGG-16, and VGG-19. The results of this experiment demonstrated the feasibility and validity of the model architecture for accurately classifying the four types of planthoppers. Quick and accurate classification can significantly reduce the time required to identify pests captured by the light trap. Implementing this approach can help reduce the reliance on manual labour while minimising the risk of human error during the classification process.

2. Materials and Methods

This study included three major stages: image acquisition, pre-processing, and classification. The flowchart for this study is shown in Figure 1.

2.1. Data Collection

The study area was located in Felcra Seberang Perak, Malaysia (4.072710082450001, 100.86747760853657). The on-field data collection was conducted by an officer from MARDI Seberang Perai, Pulau Pinang, during the paddy planting season in 2020. To collect the data, a light trap device was utilised, which consisted of a light bulb, a stand pole, and a sticky trap. The sticky trap was created using a clear plastic sheet with the dimensions of a sheet of A3 paper, onto which sticky glue was sprayed on one side. The sticky light trap was then wrapped around the light bulb to capture any pests attracted to it. The light bulb was turned on from 7:30 p.m. to 8:30 p.m., when the insects were most active. Insects were drawn to the light source, flew toward it, and became trapped on the sticky trap in various positions. Some of them sustained damage in their attempt to escape from the light trap. The following day, the sticky trap was collected and taken to the lab for the image acquisition process.

2.2. Image Acquisition

Figure 2 illustrates a machine vision system used to capture light trap images. The system consisted of an industrial camera, a fixed focal length lens, a diffused LED white light (DLW2-60-070-1-W-24V, TMS Lite, Pulau Pinang, Malaysia), a flat platform for the light trap, and a 3-axis jig. To eliminate external light interference, all components were housed inside a black box. The LED light was powered by a 24 VDC light controller (SD-1000-D1, TMS Lite, Malaysia), with the controller output set to its maximum value of 2 A. A 6 Megapixel (MP) camera MV-CA060-10GC (HIK Vision, Hangzhou, China) with a sensor resolution of 2.4 µm per pixel was used to capture the images. The camera was paired with a 35 mm focal length lens (MVL-HF3528M-6MP, HIK Vision, Hangzhou, China) and
mounted on top of the platform. The distance between the platform and the lens was set to 127 mm, as depicted in Figure 3. The combined camera and lens configuration provided a field of view (FOV) of 24 mm in width and 15 mm in length. The camera was set to capture images in the red, green, and blue (RGB) colour format, with a size of $3072 \times 2048$ pixels. An example of a captured image is presented in Figure 4. The pixel density (ppcm) of the captured image was determined by dividing the number of pixels in the FOV by the actual measurement in centimetres. In this case, the pixel density of the captured image was calculated as 1229 pixels per centimetre (ppcm), indicating that each pixel on the image represented 0.0081 mm of the actual measurement.

Figure 1. Flow chart of the study.
The light trap was integrated into the machine vision system to facilitate the image acquisition process. Within the machine vision system, an xy-jig was employed to move the camera. The dimensions of the light trap were 420 mm in width and 294 mm in length. However, the field of view of the camera (FOV) could only cover an area of 24 mm × 15 mm. As depicted in Figure 5, the region occupied by the insects was measured to be 336 mm × 245 mm. Therefore, the camera only needed to be moved across 19 × 17 grids to capture the entire populated area. The xy-jig utilised 2 stepper motors to enable camera movement in the x and y directions. As a result, the stepper motor shifted the camera from one grid to another, covering a total of 323 grids. The operation of the enclosed black box, including stepper motor movement and image acquisition, was controlled using LabVIEW software (National Instruments, Austin, TX, USA) running on a Windows-based computer system equipped with a Ryzen 5-2600X CPU@3.6GHz.)
Figure 5. Sample of light trap inside the black box.

2.3. Dataset

Figure 6 illustrates the four types of planthoppers utilised in this study: BPH, GLH, WBPH, and ZIGZAG. The captured images from the sticky trap were manually cropped to extract individual planthopper images. In total, 7328 planthopper images were cropped from the light trap image and labelled according to their respective types. The labelling process was carried out manually with the assistance of experts from MARDI, who relied on the visual features and morphology of the planthoppers. The dataset was then divided into training, validation, and test sets in an 80:10:10 ratio. This resulted in 5858 samples for the training set, 730 samples for the validation set, and 736 samples for the test set. To enhance the variety of the training sample, augmentation techniques were applied. Firstly, the images were horizontally flipped. Then, they were rotated at three different angles: 10°, 20°, and 30°. Finally, a Gaussian blur was applied as the last step of the augmentation process. After augmentation, the training set comprised a total of 187,456 samples. Figure 7 provides examples of damaged and multi-orientated planthopper images from the dataset.

Figure 6. Four types of planthoppers gathered for the data collection. (a) BPH, (b) GLH, (c) ZIGZAG, and (d) WBPH.

2.4. Model Architecture

In this study, two types of model architecture were used, namely Residual Network (ResNet) and Visual Geometry Group (VGG) network.

2.4.1. ResNet Model Architecture

Instead of learning unreferenced functions, Residual Networks (ResNets) train residual functions with reference to the layer inputs. They have an extremely deep architecture and are high-performance networks that enable the process of propagation of information to take place more directly through the network [33]. Residual nets allow each layer to match a residual mapping rather than requiring each few stacked layers to exactly match a desired
underlying mapping. This method, sometimes known as a “skip connection”, connects the activation of one layer to subsequent levels by bypassing some layers in between. It creates a network by stacking residual blocks on top of one another; e.g., ResNet-50 uses 50 layers of these blocks. This skip connection was designed to tackle the problem of CNN accuracy degradation. The inutile layer will be skipped during training. Table 1 shows the network structure of ResNet-50, ResNet-101, and ResNet-152; these three models were used in this study.

![Sample images](a) good, (b) misoriented, (c) damaged, and (d) overlapped.

**Figure 7.** Samples of planthopper images in different conditions (a) good, (b) misoriented, (c) damaged, and (d) overlapped.

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>Output Size</th>
<th>50-Layer</th>
<th>101-Layer</th>
<th>152-Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>112 × 112</td>
<td>7 × 7, 64, stride 2</td>
<td>3 × 3 max pool, stride 2</td>
<td></td>
</tr>
<tr>
<td>Conv2(x)</td>
<td>56 × 56</td>
<td>[1 × 1, 64] × 3</td>
<td>[1 × 1, 64] × 3</td>
<td>[1 × 1, 64] × 3</td>
</tr>
<tr>
<td>Conv4(x)</td>
<td>14 × 14</td>
<td>[1 × 1, 256] × 3</td>
<td>[1 × 1, 256] × 23</td>
<td>[1 × 1, 256] × 36</td>
</tr>
<tr>
<td>Conv5(x)</td>
<td>7 × 7</td>
<td>[3 × 3, 512] × 3</td>
<td>[3 × 3, 512] × 3</td>
<td>[3 × 3, 512] × 3</td>
</tr>
</tbody>
</table>

**Table 1.** Model architectures for ResNet. The number of stacked building blocks is indicated in brackets. Downsampling is performed by conv3(1), conv4(1), and conv5(1) with a stride of 2.

### 2.4.2. VGG Model Architecture

Visual geometry group (VGG) is a simple and effective CNN architecture proposed by [34]. A convolutional layer, an activation layer, a pooling layer, and a dense layer make up the VGG hierarchical structure of the CNN. Among these, the convolutional layer is one that is essential. By implementing “local perception” and “parameter sharing” in
two different methods, the objectives of feature extraction and dimensionality reduction processing are achieved. The convolution kernel is the main element of the convolution layer. The convolution kernel enables the retrieval of the shape of an object from several spots within an image, which minimises the dimensionality and the number of parameters that must be learned [35]. Smaller filters ($3 \times 3$) were used in this network, which minimised its computational complexity by lowering the number of parameters.

The VGG architecture starts by passing the image dataset through a stack of convolutional layers. VGG-16 has 13 convolutional layers and 3 fully connected layers, whereas VGG-19 has 16 convolutional layers and 3 fully connected layers. Both VGG models require an image size of $224 \times 224 \times 3$, which is an RGB image with pixel size of $224 \times 224$. A filter with size of $3 \times 3$ captures the concept of the left, right, top, bottom, and centre of the image. The convolution process was achieved with a 1 pixel stride. Spatial padding was utilised to maintain the spatial resolution of an image following convolution. Two 64 convolution kernels were processed in the first stage of the convolution process, and two 128 convolution kernels were processed in the second stage. During the third to fifth stage, VGG-16 and VGG-19 used different numbers of convolution. For VGG-16, 256, 512 and 512 convolution kernels were used from third to fifth stage, where in each stage the convolution was repeated 3 times. VGG-19 had the same convolution kernels with VGG-16, with the difference being on the number of convolution processes, which is 4. After each stage, max-pooling was performed on the output. Maximum pooling was carried out with a $2 \times 2$ pixel window and a stride size of 2. The next step was performing fully connected layers 3 times, followed by a softmax layer. Table 2 shows the two VGG architectures used in this study, i.e., VGG-16 and VGG-19.

2.5. Experimental Setup

All of the training, validation, and testing processes were conducted using Jupyter Notebook (version 6.4.12) on a 64-bit Windows 11 operating system. The system was equipped with an AMD Ryzen 5 2600X processor running at 3.6 GHz and 16 GB RAM. The model training utilised the processing power of an NVIDIA GeForce RTX 3060 GPU with 12 GB VRAM, using CUDA API version 11.2. The algorithm was implemented using Keras, which was a deep learning API that operated on the TensorFlow library for machine learning platform. The experimental setups were carefully tuned to fully utilise the memory capacity of the GPU.

2.6. Pre-Processing

A total of 7328 original sample images were used in the experiment. These samples were randomly divided into the training set, validation set, and test set in the proportions of 80:10:10. The training samples were augmented, resulting in a total of 187,456 images after the augmentation process. The distribution of classes in the training dataset was as follows: BPH (35,264), GLH (40,992), ZIGZAG (29,568), and WBPH (81,632). Table 3 shows the detail of the dataset that been used for this experiment.

The training data exhibited class imbalance. Nevertheless, the extensive number of samples utilised in this study for training purposes could sufficiently mitigate concerns related to overfitting and bias. Additionally, to preserve the integrity of the images and prevent distortion caused by resizing, which could affect the geometry and shape of the samples, each image was scaled to 256 pixels in its smallest dimension while maintaining its original aspect ratio. Subsequently, the images were randomly cropped to a size of $224 \times 224$ pixels. These measures were implemented to expedite the model training process.

The model weights were randomly initialised. The group parameters of the last dense layer were modified for each model to accommodate four classes, which represented the total number of classes in this study. All models utilised a softmax activation function in the final layer. The SGD optimiser was employed, with a categorical cross-entropy loss function. The optimiser was configured with a learning rate of 0.0005, momentum of 0.9, and without Nesterov. A batch size of 32 was set for all models. The models were
trained for 20 epochs, with early stopping implemented when there was no improvement in validation loss after 3 epochs. An epoch refers to a complete iteration of the training data through the algorithm.

Table 2. Architecture for VGG-16 and VGG-19.

<table>
<thead>
<tr>
<th></th>
<th>VGG-16</th>
<th>VGG-19</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 weight layers</td>
<td>19 weight layers</td>
<td></td>
</tr>
<tr>
<td>Input (224 × 224 RGB image)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
<td></td>
</tr>
<tr>
<td>conv3-64</td>
<td></td>
<td>conv3-64</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
<td></td>
</tr>
<tr>
<td>conv3-128</td>
<td></td>
<td>conv3-128</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
<td></td>
</tr>
<tr>
<td>conv3-256</td>
<td></td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-256</td>
<td></td>
<td>conv3-256</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
<td>conv3-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-4096</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC-1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>soft-max</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Details of planthopper dataset.

<table>
<thead>
<tr>
<th>Planthopper Name</th>
<th>Number of Image Sample</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original Image</td>
<td>Training Image</td>
<td>Training Image (Augmented)</td>
<td>Validation Image</td>
<td>Testing Image</td>
</tr>
<tr>
<td>Brown Planthopper</td>
<td>1379</td>
<td>1102</td>
<td>35,264</td>
<td>139</td>
<td>137</td>
</tr>
<tr>
<td>Green Leafhopper</td>
<td>1603</td>
<td>1281</td>
<td>40,992</td>
<td>161</td>
<td>160</td>
</tr>
<tr>
<td>Zigzag Leafhopper</td>
<td>1156</td>
<td>924</td>
<td>29,568</td>
<td>116</td>
<td>115</td>
</tr>
<tr>
<td>White-Backed Planthopper</td>
<td>3190</td>
<td>2551</td>
<td>81,632</td>
<td>320</td>
<td>318</td>
</tr>
<tr>
<td>Image Total</td>
<td>7328</td>
<td>5858</td>
<td>187,456</td>
<td>736</td>
<td>730</td>
</tr>
</tbody>
</table>

2.7. Performance Metrics

The primary objective of this research was to develop a classification model for distinguishing four different planthopper types. The accuracy of classification was deemed
the most crucial aspect of this multi-class task. Therefore, performance metrics based on
the confusion matrix, including accuracy, precision, recall, and F1-score, were utilised
to compare the performance of various models. Accuracy, which represents the proportion
of correctly predicted observations to all observations, is the simplest performance metric
to understand. However, when dealing with imbalanced datasets, accuracy may not provide
a clear picture of the performance of the model, as imbalanced datasets often exhibit a bias
toward the dominant class [36]. Hence, the F1-score, which combines precision and recall,
is a more suitable metric for imbalanced datasets. Precision measures the proportion of
correctly predicted positive observations out of all projected positive observations, while re-
call assesses the proportion of correctly predicted positive observations out of all instances
in the true positive class. Accuracy, precision, recall, and F1-score are computed using
indices such as true positive ($tp_i$), false positive ($fp_i$), true negative ($tn_i$), and false negative
($fn_i$). True positive and true negative refer to the model correctly predicting the positive
class or negative class, respectively. A false positive occurs when the model incorrectly
predicts the positive class, whereas a false negative occurs when the model inaccurately
predicts the negative class. In this study, the Scikit-learn library was utilised to plot the
performance metrics for the training and test results. The equations used to calculate
accuracy, precision, recall, and F1-score for each individual class (i) are provided in Table 4.
Table 5 displays the performance metrics used to calculate the average performance across
all classes (n), which include average accuracy, macro-average precision, macro-average
recall, and macro-average F1-score.

Table 4. Metrics for assessing the performance of individual classification classes.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy$_i$</td>
<td>$\frac{tp_i + fn_i}{tp_i + fp_i + fn_i + tn_i}$</td>
</tr>
<tr>
<td>Precision$_i$</td>
<td>$\frac{tp_i}{tp_i + fp_i}$</td>
</tr>
<tr>
<td>Recall$_i$</td>
<td>$\frac{tp_i}{tp_i + fn_i}$</td>
</tr>
<tr>
<td>F1-Score$_i$</td>
<td>$2 \times \frac{Precision$_i$ \times Recall$_i$}{Precision$_i$ + Recall$_i$}$</td>
</tr>
</tbody>
</table>

Table 5. Metrics for assessing the average performance of all classification classes.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Accuracy</td>
<td>$\frac{\sum_{i=1}^{n} Accuracy$_i$}{n}$</td>
</tr>
<tr>
<td>Macro Average Precision</td>
<td>$\frac{\sum_{i=1}^{n} Precision$_i$}{n}$</td>
</tr>
<tr>
<td>Macro Average Recall</td>
<td>$\frac{\sum_{i=1}^{n} Recall$_i$}{n}$</td>
</tr>
<tr>
<td>Macro Average F1-Score</td>
<td>$\frac{\sum_{i=1}^{n} F1-Score$_i$}{n}$</td>
</tr>
</tbody>
</table>

3. Results and Discussion
This section provides an in-depth analysis of the training, validation, and test outcomes
achieved using the sample. The results are presented by comparing the performance metrics
of all of the models.

3.1. Model Comparisons
The stopping criterion for each model was set to 20 epochs. However, we also intro-
duced an early stopping criterion where the model would stop if there was no improvement
in the validation loss after three epochs. Figure 8 displays the validation loss results, while
Figure 9 illustrates the validation accuracy results for each of the five CNN models. ResNet-
152 and ResNet-50 stopped at the fifth epoch, ResNet-101 stopped at the eighth epoch,
VGG-16 stopped at the ninth epoch, and VGG-19 stopped at the 15th epoch. Initially,
ResNet-152 and ResNet-50 exhibited high loss values, surpassing 1. On the other hand, the
loss values for VGG-16 and VGG-19 models showed gradual and minimal fluctuations. ResNet-101 exhibited fluctuations in the loss rate, with a final loss rate of 0.5647 before stopping, compared to its lowest value of 0.181. ResNet-152 had the lowest loss rate of 0.1348, followed by ResNet-101 with 0.1819. The highest loss value was observed in VGG-16, with a value of 0.2515.

In terms of validation accuracy, ResNet-152 achieved the highest accuracy with a value of 0.9583 at its lowest loss, followed by ResNet-50 with a value of 0.937. VGG-16 had the lowest accuracy value of 0.9 at its lowest loss.

Figure 10 displays the average time taken by each model to complete one epoch of training and validation. The plot reveals that ResNet-152 required the longest time, taking 50.17 min, while VGG-16 had the shortest time, at 12.53 min. This pattern indicated that models with more layers required more time to complete the training and validation process. Table 6 presents the average prediction time for a single sample. According to the table, VGG-16 exhibited the fastest prediction time, taking 0.022 s, followed by VGG-16
with 0.023 s. On the other hand, ResNet-152 had the longest prediction time, of 0.051 s. These results indicated that the training time directly influenced the prediction time for all models.

Figure 10. Model comparison based on time taken to complete training and validation.

Table 6. Average time for predicting an image sample.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prediction Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>0.026</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>0.037</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>0.051</td>
</tr>
<tr>
<td>VGG-16</td>
<td>0.022</td>
</tr>
<tr>
<td>VGG-19</td>
<td>0.023</td>
</tr>
</tbody>
</table>

Figure 11 presents a comparison of the results for each model based on accuracy, precision, recall, and F1-score for the test dataset. Each model achieved an average accuracy greater than 93.68%, which is impressive and meets the expected performance due to the utilisation of a large dataset. Among the models, ResNet-50 exhibited slightly superior performance with an accuracy value of 97.28%, while the model with the lowest performance was ResNet-152 with an accuracy of 93.68%. ResNet-101 demonstrated the second-highest performance with an accuracy of 97.15%, followed by VGG-16 (96.81%) and VGG-19 (95.92%). In terms of all performance metrics, ResNet-50 outperformed other models, with the lowest score in precision at 92.05%. Despite the imbalanced datasets, the performance of the models based on F1-scores exhibited a pattern similar to accuracy with slightly lower scores. ResNet-50 achieved the highest F1-score value at 93.07%, followed by ResNet-101 (91.91%), VGG-16 (91.34%), VGG-19 (89.36%), and ResNet-152 (86.59%). These results indicated that the utilisation of a large number of samples for training and testing in this study provided sufficient data to mitigate overfitting and bias concerns. Despite not having the fastest training time, ResNet-50 was considered the best choice for the classification model, as it only required 0.026 s to classify one sample during testing. Training was only conducted if there were new varieties of planthopper samples.
Agriculture 2023, 13, x FOR PEER REVIEW 14 of 18

Figure 11. Model comparison based on average accuracy, macro average precision, macro average recall, and macro average F1-score in percentage.

3.2. Error Analysis

This section discusses the analysis of the error on the prediction performed by the best performing model, which was ResNet-50. Figure 12 shows the confusion matrix for the ResNet-50 model. Recall was measured as the ratio of samples that were correctly assigned to a class to the total samples in that class. As for precision, it was determined by dividing the number of samples that were correctly classified by their classification by the number of anticipated classes.

Figure 12. Confusion matrix for ResNet-50 model.
The confusion matrix revealed that the model accurately categorised 161 GLH samples, demonstrating almost perfect classification. The majority of misclassifications occurred between BPH and WBPH, with 8 out of 116 WBPH samples misclassified as BPH, and 6 out of 139 BPH samples misclassified as WBPH. This was because WBPH and BPH were nearly identical in size and shape, with identical head and body proportions. When the backs of the planthoppers were clearly visible, these two groups could be easily distinguished. WBPH had a white line on their head and wings, while BPH had a distinct shape on their wings. However, if the backs of the planthoppers were not visible, it became extremely difficult to identify them. Alternative methods included reviewing the body colour of a BPH, which should be entirely brown, or searching for a white stripe on the face or side of a WBPH. WBPH generally had a darker body colour than BPH, but it could still be challenging to distinguish them if the BPH had a body colour similar to that of the WBPH. For GLH, it could be distinguished from other classes by its green body with a black stripe on its back. However, if viewed from the side, other samples could also be misclassified as GLH. As indicated by the confusion matrix, six WBPH samples were misclassified as GLH, while nine ZIGZAG samples were misclassified as GLH. This was because GLH also had a black body beneath the wings, sharing a characteristic with the other classes.

Figure 13 presents a comparison of accuracy, recall, precision, and F1-score for each planthopper class classified using ResNet-50. From the plot, it can be observed that GLH had the highest values for accuracy (95.52%), precision (100%), recall (96.41%), and F1-score (98.17%). On the other hand, WBPH had the lowest values for accuracy (95.52%), precision (79.31%), and F1-score (85.58%). Comparing these results with those obtained by [32], our model outperformed their proposed method with 2.28% higher accuracy for four types of planthoppers instead of only two types of planthoppers.

4. Conclusions

Detecting insect pests is crucial in agriculture, particularly in paddy fields, as it facilitates the assessment of their population dynamics and density. Accurate detection allows for precise and targeted application of pesticides. However, automatically detecting insects using image processing presents challenges due to the unpredictable nature of their environment. The presence of imperfect samples of trapped insects and inconsistent categorisation by humans further complicates the task. To overcome these challenges, this study proposed an automated detection method for planthoppers using deep CNN.
Five models, namely ResNet-50, ResNet-101, ResNet-152, VGG-16, and VGG-19, were employed to train on randomly initialised weights and identify the characteristics of four planthopper classes: BPH, GLH, WBPH, and ZIGZAG. The planthopper images were converted to RGB format and augmented to increase the sample size. A total of 7328 images were used, with 80% allocated for training, 10% for validation, and 10% for testing. The performance of these five approaches was evaluated using accuracy, precision, recall, and F1-score. The results demonstrated that ResNet-50 achieved the highest performance, with an average classification accuracy of 97.28% and individual class accuracies of 96.74% for BPH, 99.18% for GLH, 95.52% for WBPH, and 97.28% for ZIGZAG. It is important to note that the classification was performed on image samples that were previously manually cropped by an expert. Although the proposed method demonstrated promising results, it had a limitation in the case of borderline cases. In this study, we observed that these borderline cases were frequently misclassified. Furthermore, overlapping samples posed a substantial issue in the classification process. Overlapping insects were more prevalent when data collection was conducted over a longer duration. We limited the sample collection duration to one hour per sample. Consequently, the collected samples contained fewer overlapping insects. The overlapping issue can be addressed in the future to enhance the robustness of the classification process. Developing methods to handle overlapping samples could significantly improve the performance of the classification process. Furthermore, an additional effort can be undertaken to integrate the processes of object detection and classification to integrate them into a single step, aiming to fully automate the counting of planthoppers. Additionally, the capability of other deep learning architectures for planthopper classification can also be studied in the future.

Author Contributions: Conceptualisation, M.F.I. and S.K.-B.; methodology, M.F.I. and S.K.-B.; software, M.F.I.; formal analysis, M.F.I. and S.K.-B.; validation, M.F.I. and S.K.-B.; investigation, M.F.I. and M.A.M.B.; resources, M.F.I. and M.A.M.B.; data curation, M.F.I. and M.A.M.B.; writing—original draft preparation, M.F.I.; writing—review and editing, S.K.-B.; visualisation, M.F.I. and S.K.-B.; supervision, S.K.-B., M.H., M.J. and F.S.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to restrictions.

Acknowledgments: The authors would like to give thanks to MARDI for providing a machine vision system for data collection, assisting in field sampling and helping in identifying the pest dataset.

Conflicts of Interest: The authors declare no conflict of interest.

References
Agriculture 2023, 13, 1155


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.