Large-Scale Insect Pest Image Classification

Thanh-Nghi Doan [1,2]
[1] Faculty of Information Technology, An Giang University, An Giang, Vietnam
[2] Vietnam National University Ho Chi Minh City, Vietnam
Email: dtnghi@agu.edu.vn

***Abstract*—One of the main issues with agricultural production is insect attack, which leads to poor crop quality. Farmers, however, have a complicated and time-consuming task in detecting and categorizing insects. Therefore, research on an effective system for image-based automated insect classification is crucial. The conventional "softmax" function is utilized to determine the category for new image occurrences and minimize "cross-entropy" loss in the bulk of current research, which focuses on employing deep convolutional neural networks to categorize insect images. This paper presents a novel method for large-scale insect pest image classification by combining fine-tuning EfficientNets and Power Mean Support Vector Machine (SVM). First, EfficientNet models are fine-tuned and re-trained on new insect pest image datasets. The retrieved features from EfficientNet models are then utilized to create a machine learning classifier. In the network's classification stage, the traditional "softmax" function is substituted with a non-linear classifier, Power Mean SVM. As a result, rather than "cross-entropy loss," the training process focuses on reducing "margin-based loss." Several benchmark insect image datasets were used to evaluate our proposed method. According to the numerical results, our method outperforms other cutting-edge methods for large-scale insect pest image categorization. With fine-tuning EfficientNets and Power Mean SVM, the classification accuracy of the proposed method for the Xie24, D0, and IP102 large insect pest datasets is 99%, 99%, and 72.31%, respectively. To our knowledge, these are the best performing image classification results for these datasets.**

***Keywords*—EfficientNets, power mean Support Vector Machine (SVM), large-scale insect image categorization**

I. INTRODUCTION

Insects are crucial to long-term agricultural growth and ecosystem. There are currently slightly over 5.5 million species of insects recognized and described out of a total of more than 30 million species [1]. However, only approximately 500 living insect species ruin crops; the remainder are helpful insects that eliminate harmful insects and preserve crops. Due to a lack of expert understanding, farmers are unable to effectively detect pests and diseases, therefore they employ ineffective pesticides. This results in the extinction of many insects and other useful creatures, as well as a significant negative impact on the human environment. As a consequence, it's vital to study a system that can automatically categorize enormous amounts of insect images. In recent years, several computer-based insect detection approaches have been investigated [2−9]. In these studies, image feature extraction methods are key factors in the effectiveness of a pest and disease image classification system. They can be divided into two groups: 1) Handcrafted features - image feature extraction methods that are manually designed, they are commonly used with conventional machine learning methods for object recognition and 2) Non-handcrafted features-image feature extraction methods that use Convolutional Neural Networks (CNNs). Handcrafted image feature generation approaches are useful for conveying low-level picture features including color, edge, and texture. In the study of Rani and Amsini [10], the Support Vector Machine (SVM) classifier was used to detect five different types of insects. These algorithms extract the primary visual features to build vector representations of insect images, which are subsequently tested on tiny insect datasets. However, the amount of living insect species on the planet has been approximated to be much larger, thus the design of handcrafted image feature extractors is inefficient.

The shortcomings of the above techniques have been solved by deep transfer learning features based on CNNs, a sort of non-handcrafted feature. In agriculture, CNN-based approaches have recently been used for weed identification, plant identification, insect and disease classification [11−19]. These CNNs have also outperformed other traditional approaches when it comes to insect pest image classification. Xia *et al.* [20] have developed a method for classifying insect pest and disease images based on VGG19 and the Region Proposal Network (RPN). RPN detects insect pest locations in images, whereas VGG19 extracts visual features from insect pest images. Thenmozhi and Reddy have proposed CNN models trained from scratch [21], and their method has been evaluated on three insect datasets: the National Bureau of Agricultural Insect Resources (NBAIR), Xie24 [22], and D0 [23]. Their method has been compared to various CNN models trained on ImageNet [24]. Furthermore, they investigated variables including batch size, iteration count, and learning rate. Recently, Wu *et al.* have created a sizable dataset of insects, IP102, by collecting insect images from numerous sources [25]. Then they have implemented a series of experiments to evaluate this dataset with both conventional machine learning methods and recent CNN approaches. Their

Manuscript received October 13, 2022; revised October 30, 2022; accepted December 19, 2022; published April 17, 2023.

experimental results have shown that ResNet-50 achieved the highest classification accuracy of 49.5%. Ren *et al.* have proposed an image feature synthesis method based on reusing the residual block structure to enhance image feature representation [26]. On the IP102 dataset, their approach was tested and found to have a classification performance of 55.24%. Nanni *et al.* have proposed an insect pest classification method that combined a variety of CNNs (AlexNet, GoogleNet, DenseNet, ShuffleNet, and MobileNetv2) that were trained on images preprocessed utilizing three saliency image methods, yielding nine new images for each original image in the datasets [27]. They claimed that their method achieved an accuracy of 92.43% and 61.93% on the Deng and IP102 datasets, respectively. Ayan *et al.* [28] have evaluated seven CNN models for refining and transfer learning on the D0 dataset [23]. The three models with the best classification accuracy were ensembled to increase overall classification performance using a sum of maximum probability technique. Weighted voting was used to combine these three models. The weights were calculated using a genetic algorithm that took into account the likelihood of success and predicted reliability of these models. The resulting ensemble model was called GAEnsemble. Their method has been compared to other methods and evaluated on the IP102 dataset, and it has achieved a classification accuracy of up to 67.13%. According to Setiawan *et al.*'s work [18], an effective training framework for small-sized model optimization on MobileNetV2 employing dynamic learning rate, utilizing CutMix augmentation, freezing layers, and sparse regularization, has been developed. The best accuracy of 71.32% was attained during training by combining those strategies.

As evidenced by the preceding research, there has been a surge in the usage of CNN approaches in the classification of insect images in the literature. However, there are still certain holes to fill in terms of developing more accurate CNN models for insect pest classification tasks. This research describes an innovative and effective method for enhancing the accuracy of insect pest image classification, especially for large-scale insect pest image datasets. Our approach employs fine-tuning EffcientNet [29] and transfer learning methods on new datasets. The image features are extracted from the datasets using the trained network model. In which batch data processing is utilized to avoid loading all data into the computer's main memory. Finally, in the insect image classification step, the Power Mean SVM classifier (PmSVM) [30] is utilized to replace the conventional softmax classifier in the output layer of CNN models. As a result, rather than focusing on "cross-entropy loss," our method focuses on minimizing "margin-based loss." Furthermore, we employ an SVM classifier with a non-linear kernel rather than a linear kernel as in Tang's wrok [31]. The following are the primary contributions to this reseach:

- An efficient approach is proposed for combining fine-tuning EfficientNet with PmSVM [30], in which a non-linear SVM classifier with the Power Mean Kernel replaces the conventional CNN classifier with the softmax function. This allows the training process to concentrate on decreasing margin-based loss rather than cross-entropy loss, which enhances CNN model classification performance.
- Several different CNN models have been evaluated on numerous large-scale insect pest datasets to find the best model for categorizing insect pest images. Specifically, EfficientNet models were chosen and fine-tuned with a new efficient transfer learning method for training these models. This training procedure is separated into two stages: (i) The first step involves training the full connected (FC) layer heads of the networks, and (ii) the second step involves unfreezing and re-training certain previous convolutional layers in the network at a lower learning rate. Therefore, the networks converge faster, resulting in improved classification performance.
- A technique for extracting image features from large-scale datasets that divides the datasets into several small batches for online and parallel processing. As a consequence, this method may be applied to a variety of large-scale datasets containing several classes.

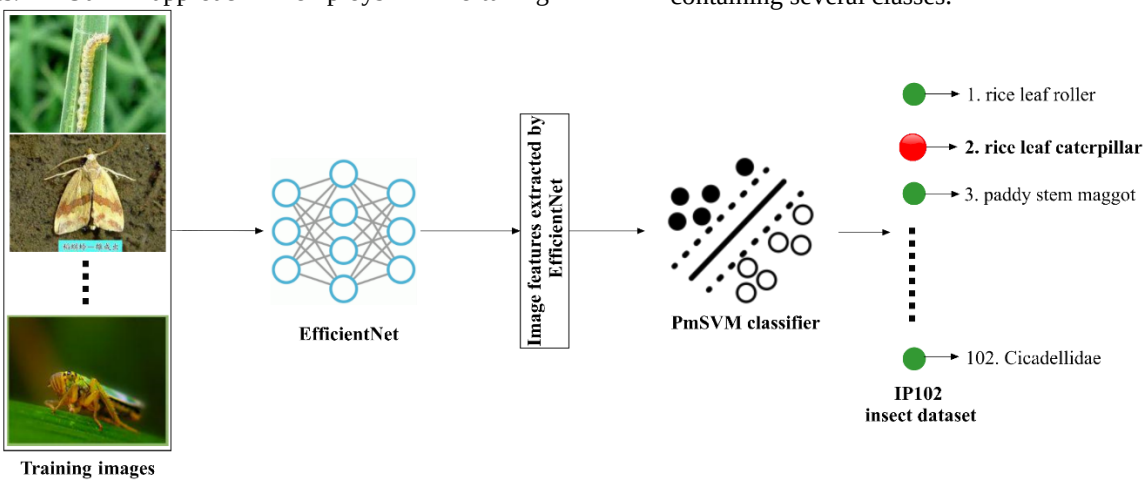

Figure 1. Overview of our insect image classification workflow.

Our approach has been tested on numerous benchmark insect image datasets, including Xie24 [22], D0 [23], and IP102 [25]. Experiments show that our approach beats several previous methods on these datasets. For instance, on Xie24, D0, and IP102, the approach achieved classification accuracy of 99%, 99%, and 72.31%, respectively. Furthermore, our method is easily applicable to enormous datasets that are larger than the computer's main memory capacity. The overall process of our proposed insect image classification approach is shown in Fig. 1. In terms of merits and limitations, Table I compares our approach to recent literature.

TABLE I. THE MERITS AND LIMITATIONS OF OUR METHOD IN COMPARISON TO RECENT LITERATURE

Methods	Merits	Limitations
FR-ResNets [26]	Feature reuse residual block which combines feature from the input signal of a residual block with the residual signal	- Require an effective feature reuse network structure - Non-incremental learning - Utilize the softmax classifier
Ayan *et al.* [28]	Combines different generalization capabilities of CNN models to detect insect species	- Non-incremental learning - Utilize the softmax classifier - High cost of training models - Complicated CNN models
Setiawan *et al.* [18]	Use small-sized models MobileNetV2 with dynamic learning rate, exploite CutMix augmentation, freezing layers, sparse regularization.	- Non-incremental learning - Utilize the softmax classifier
Our approach	Use state-of-the-art CNN EfficientNet, a fast and scalable framework, PmSVM classifiers instead of the Softmax classifier, and two rounds training	- Non-incremental learning - Not make use of feature confusion

The rest of the article is arranged as follows. The materials and methods related to insect pest image datasets, deep convolutional neural networks, visual explanations from deep networks, transfer learning methods, and the softmax classifier versus the Power Mean SVM are presented in Section II. Section III reports the results and discussion. Section IV contains the conclusions and recommendations for future research.

II. MATERIAL AND METHOD

This study presents a three-stage classification strategy for a large-scale insect pest image classification system. As illustrated in Fig. 1, the pre-trained CNN models from the ImageNet dataset [32] were fine-tuned and re-trained using a transfer learning method on the Xie24, D0, and IP102 insect datasets. Then, the CNN model with the greatest classification accuracy is chosen to retrieve image features from the insect image datasets. Finally, the PmSVM [30] is employed to train a classifier based on the extracted image feature datasets.

A. Insect Pest Image Datasets

As in recent studies, the three following common benchmark insect pest image datasets were utilized to evaluate our proposed approach.

Xie24: This dataset was suggested by Xie *et al.* [22]. It has 1600 RGB images of 24 different insect species. All images are preprocessed and then have a resolution of 227×227 pixels. To compare our method to that of Thenmozhi and Reddy's work [21], data augmentation techniques were utilized to improve the volume and variety of these image datasets before training the CNN models. The Xie24 dataset, after applying several image transformations, has a total of 6892 images. This dataset is then split into three subsets, with 70% samples of each class for training, 10% samples for validation, and 20% samples for testing. Therefore, there are 4653 training images, 516 validation images, and 1723 testing images as a consequence.

D0: This dataset was created by Xie *et al.* [23] and made public at the website https://www.dlearningapp.com/web/DLFautoinsects.htm. It includes 4508 RGB images with a resolution of 200×200 pixels from 40 different insect classes. We also split this dataset into three subsets with the same ratio as the Xie24 dataset. As a result, we have 2682 training images, 473 validation images, and 1353 testing images.

IP102: This large-scale insect pest collection recommended by Wu *et al.* [25] has 75,222 photos arranged in a hierarchical taxonomy. It contains 45,095 training photos, 7508 validation images, and the remaining 22,619 images for model evaluation. Fig. 2 presents some insect image samples from the IP102 dataset. This dataset comprises a number of characteristics that impact the success of image classification methods. First, due to their color and the similarity of the backdrop images, the insect in the photograph is challenging to identify. Second, because the morphology of an insect problem, such as a rice leaf roller, can vary significantly as it matures, the classes include photos depicting many stages of the insect's life cycle; this makes identifying each insect problem extremely difficult. Third, many insects and diseases are classified as different but have very similar appearances. Fourth, this dataset is highly imbalanced; the least represented class (Erythroneura apicalis) has only 71 images, while the most common class (Cicadellidae) has nearly 5740 images. Therefore, these are factors that generate numerous difficulties and obstacles for designing an efficient insect pest image classification algorithm on the IP102 dataset.

Data augmentation: By changing inputs while preserving output labels, "data augmentation" is a typical strategy for boosting the amount and variety of labeled training sets. In the field of computer vision, image augmentation techniques have become a prominent implicit regularization tool for reducing overfitting in deep convolutional neural networks, and they are commonly employed to improve performance [33, 34]. In this work, several image data augmentation techniques including rotation, zoom, shifting, shearing, flipping, and cropping operators are applied to three insect pest datasets. Fig. 3

shows how the Xie24, D0, and IP102 datasets' insect pest images of Cifuna locuples, Aulacophora indica, Red Spider, and Rice Leaf Roller are transformed into numerous images in the enhanced dataset using nine different operators.

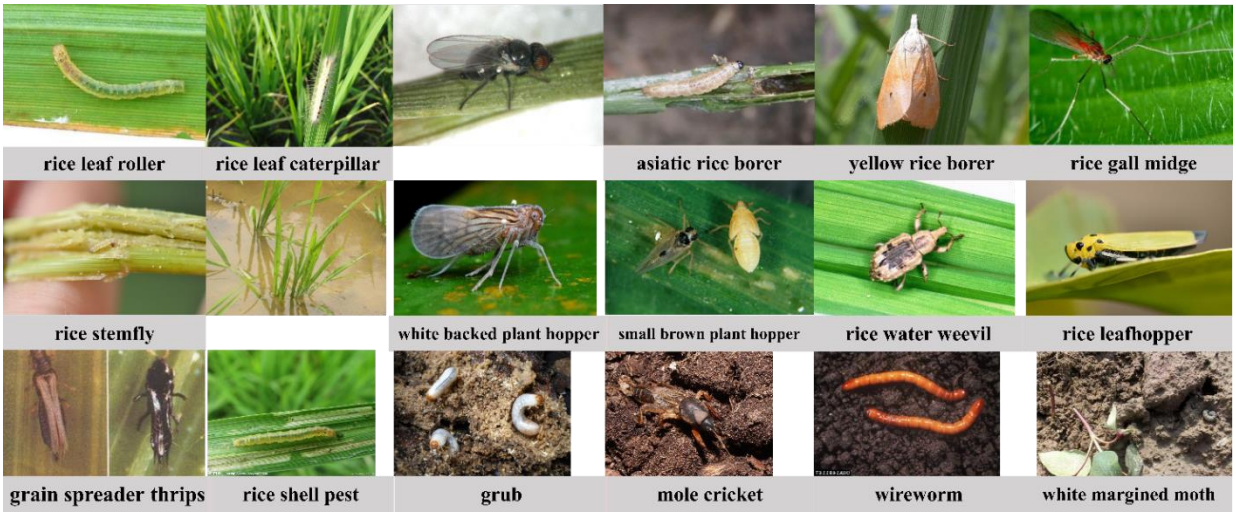

Figure 2. Image samples from a large-scale dataset for insect recognition, IP102.

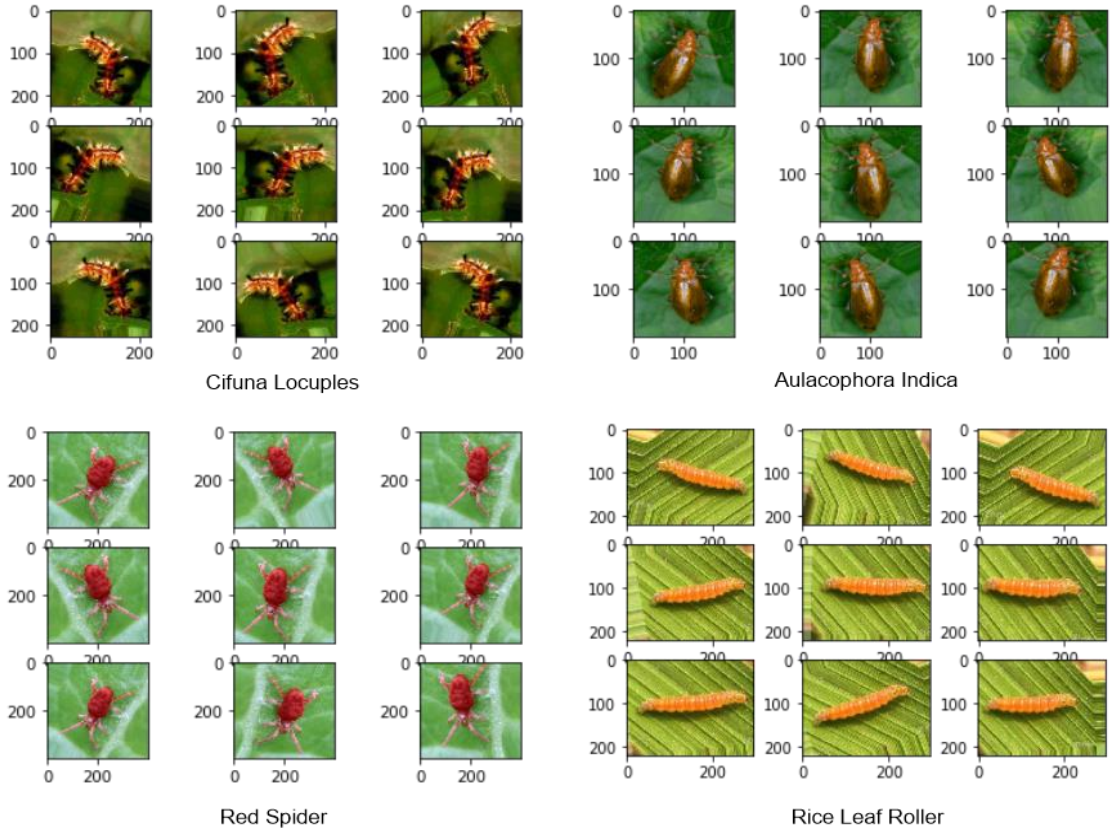

Figure 3. Data augmentation techniques with nine different operators are applied to some sample insect pest images of Cifuna Locuples from Xie24, Aulacophora Indica from D0; Red Spider, Rice Leaf Roller from IP102.

B. *Deep Convolutional Neural Network*

LeCun *et al.* [35] used a backpropagation approach to develop deep convolutional neural networks that can learn hierarchical features from data. The convolution layer, the pooling layer, and the fully connected layer are the three main components. To extract features from the image, the convolutional layer creates a set of filters of a predefined size (e.g., 3×3, 5×5, 7×7). An activation map is produced by applying each convolution operation to an image with each filter. Then this activation map is sent into the next layer. The pooling layer shrinks inputs to maintain distinguishing features while reducing model parameters. The output of CNNs is the FC layers, which use the extracted features from the convolutional layers to perform

classification tasks. In this work, many different CNN models were re-trained and evaluated on several benchmark insect pest datasets, consisting of Xie24, D0, and IP102.

1) VGG

VGG is a standard CNN architecture proposed by Simonyan and Zisserman [36]. The "deep" refers to the number of layers, with VGG-16 or VGG-19 having 16 or 19 convolutional layers, respectively. Cutting-edge object recognition models are built on top of the VGG architecture. On a range of tasks and datasets, the VGGNet, which was constructed as a deep neural network, beats baselines. It's also become one of the most widely used image recognition architectures. It has been shown that representation depth improves classification accuracy and that state-of-the-art results on the ImageNet challenge dataset may be reached using a traditional CNN architecture with much greater depth [32, 35]. This CNN architecture is frequently used in the identification of insects and pests in several research articles, such as [20, 23, 37].

2) ResNet

ResNet [38] is a proposed network architecture for addressing the issue that some non-linear layers do not learn the activation maps of the image. ResNet is designed with a network model based on many stacked residual units. ResNet uses a network model that consists of several stacked residual units. These leftover units are employed as network building components. Convolution and pooling layers are examples of these units. This architecture employs a 3×3 filter with a 224×224 pixel input image. The special architecture of ResNet helps the backpropagation process avoid gradient degradation. ResNet has parallel shortcuts to regular convolution layers that will aid the network in understanding global image features. After numerous weight levels, a shortcut is utilized to add the previous layer's input vector to the output of the following layer. These shortcuts allow the network to skip layers that are not useful, resulting in a more optimally tuned number of layers and making the training process faster. ResNet has been evaluated for insect pest classification in many studies [28, 39].

3) InceptionV3

InceptionV3 [40] is a GoogleNet variation based on factoring 7×7 convolutions into two or three sequential layers of 3×3 convolutions. InceptionV3 is a deep neural network design from the Inception series that contains Label Smoothing, 7×7 convolutions, and an extra classifier to transfer labeled data closer to the bottom of the network, an improvement over previous versions. InceptionV3 is primarily concerned with using fewer computing resources by altering prior Inception designs. This network has been demonstrated to be more computationally efficient than VGG networks, both in terms of the amount of parameters it creates and the cost it incurs (memory and other resources). If the Inception network is altered, special care must be taken to avoid losing the computational improvements. Because of the uncertainty surrounding the new network's efficiency, customizing an Inception network for a variety of use cases becomes a challenge. Several ways for enhancing the network in an InceptionV3 model have been developed in order to release the constraints and make the model more adaptable. Among the techniques employed are factorized convolutions, regularization, dimension reduction, and parallelized computations.

4) Xception

The convolutional neural network architecture Xception [41] only utilizes depth-wise separable convolution layers. An inception network is a deep convolution network with 71 layers and recurrent module designs called inception modules. It's possible that the network comes pre-loaded with a version that has been trained on over a million photographs from the ImageNet dataset [32]. Over a thousand different item categories, such as keyboards, mice, pens, and other animals, may be classified using the trained network. As a consequence, the network has learned to detect a variety of rich visual features in images. Images having a resolution of 299×299 pixels are accepted by the network. In general, each CNN layer is thought to extract some feature; hence, stacking these layers one on top of the other is not a good idea. Deep networks are prone to overfitting, and chaining many convolutional procedures together raises the training cost. Another problem is that each layer type draws a different type of information, making it difficult to determine which transformation (kernels) provides the most important information to the CNN.

5) DenseNet

DenseNet [42] is a high-performance, huge convolutional network with each layer connected to the ones before it. It recommends connecting any two layers with the same feature map dimension directly. DenseNet may expand to hundreds of layers without causing any problems during the optimization phase. In experiments, DenseNet usually has stable accuracy when the number of parameters is increasing and there is no sign of performance degradation or overfitting during training. Even though it has fewer parameters and a lower computational overhead than other CNN models, DenseNet consistently outperforms them on several benchmark datasets. This architecture allows gradients to be varied and features to be reused effectively. In several research studies, DenseNet has demonstrated high performance in insect pest and disease image classification [12, 39].

6) MobileNetV2

MobileNetV2 [43] is a mobile-friendly convolutional neural network architecture. It's based on an upturned residual architecture with residual bottleneck levels included. The middle expanding layer employs basic depth-wise convolutions to select features as a source of nonlinearity. MobileNetV2's architecture includes a fully convolutional layer with 32 filters, which is followed by 19 bottleneck layers. MobileNetV2 beats several cutting-edge real-time detectors on the COCO dataset in terms of reliability and computational cost. In comparison to YOLOv2 [44], the MobileNetV2 architecture requires 20 fewer calculations and 10 fewer parameters when used with the SSDLite detection module. This CNN has been

used in several studies of insect pest image classification [45, 46].

7) EfficientNet

EfficientNet [29] is a CNN design and scaling strategy that uniformly scales all depth, breadth, and resolution parameters using a compound coefficient. The researchers of EfficientNet argue that to achieve better accuracy, the network can be enlarged by making each layer wider, by ensuring that the input image has a higher resolution, or by a combination of all these factors. Unlike normal practice, which modifies these elements at random, the EfficientNet scaling technique uses a set of preset scaling coefficients to reliably enhance network width, depth, and resolution. Inspired by Tan *et al.* [47], they have developed the baseline EfficientNet-B0 network using a multi-objective neural architecture search that maximizes both precision and FLOPS, and then expanded it to create the EfficientNets family of models (from EfficientNet-B0 to EfficientNet-B7). The results of many experiments have shown that EfficientNets achieve higher efficiency and accuracy than other CNNs. Furthermore, EfficientNet has decreased the number of network training parameters dramatically. The experiments in [29] have also demonstrated the effectiveness of this method when extended to MobileNet and ResNet. On ImageNet, EfficientNet-B7 achieved the highest top-1 accuracy of 84.3%, but the network size is 8.4 times smaller and 6.1 times quicker than the other CNNs. EfficientNets also obtained 91.7% accuracy on the CIFAR-100 dataset and 98.8% accuracy on the Flowers dataset. In this research, we study an effective fine-tuning method to enhance the classification performance of EfficientNets on the three insect pest image datasets as presented in Section A, especially on the large-scale dataset IP102.

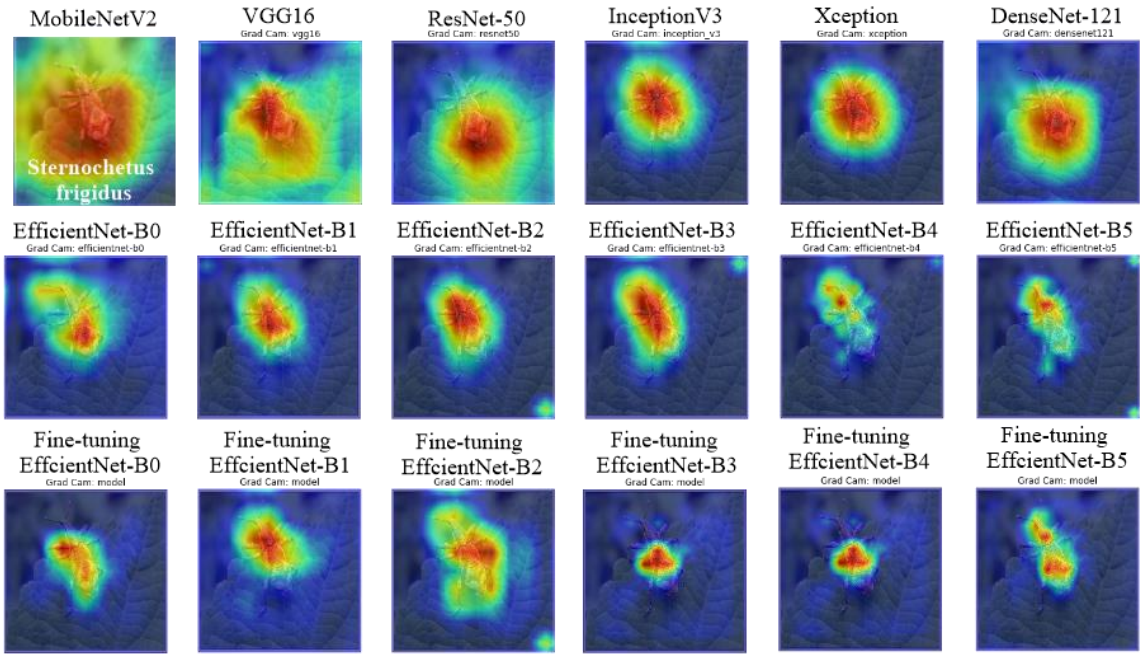

Figure 4. Class Activation Map of CNN models with different methods — The fine-tuning method allows the models (last rows) to focus on interesting areas of insect pest objects.

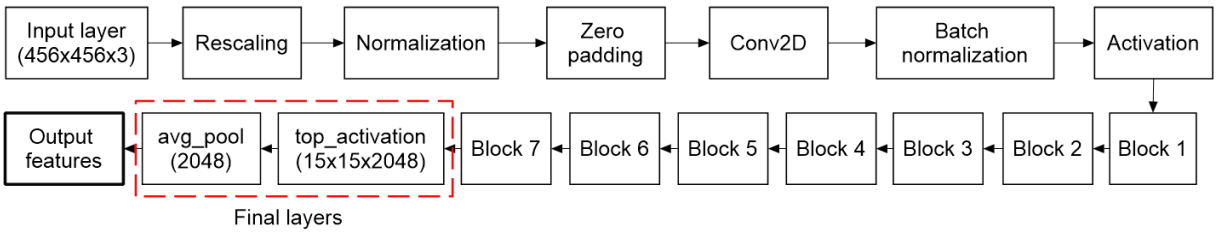

Figure 5. Fine-tunning EfficientNet-B5 model for extracting deep image features. Removing the FC layers from the network model and instead returning the final POOL layer (in the red dashed box). This output contains the extracted features.

C. *Visual Explanations from Deep Networks via Class Activation Map*

While deep learning has enabled outstanding accuracy in image classification, object recognition, and image segmentation, one of its most major issues is model interpretability, which is a critical component in model understanding and debugging. Deep learning approaches are treated as "black boxes", with no meaningful understanding of 1) where the network is "looking" in the input picture, 2) which series of neurons were engaged in the forward-pass during inference or prediction, and 3) how the network reached its final result. Therefore, we need a method to debug network models and visually confirm that they are "looking" and "activating" at the relevant spots in an image to guarantee that the model is

operating appropriately. Zhou *et al.* [48] have introduced a universal approach for CNNs with global average pooling, termed Class Activation Mapping (CAM), to assist deep learning researchers in debugging their network models. This method allows trained CNNs to locate objects without the use of bounding box annotations. In order to emphasize the distinguishable object portions that the CNNs were able to identify, CAM projects class scores onto each image. We can visually confirm where our network is looking by using CAM, ensuring that it is looking for and activating around the appropriate patterns in the image. The CNN models used to extract image characteristics must be carefully chosen since they have a considerable influence on the success of insect image categorization. Therefore, in this study, CAM visualizations of several CNNs and their fine-tuning models are evaluated for insect classification tasks. After that, the most effective CNN models are chosen for the insect image classification tasks. As illustrated in Fig. 4, the EfficientNet-B5 model provides the best-looking insect pest objects from the image of the IP102 validation dataset at random. In addition, versions of EfficientNet-B0 to EfficientNet-B5 with our fine-tuning strategy (last rows in Fig. 4) have concentrated on more interesting areas with more object information, whereas other CNN models either "look" at a lack of object information or are unable to "look" at relevant parts of objects in the image.

D. Transfer Learning

Transfer learning [49] is a strategy that leverages previously learned information as a springboard for completing a separate but similar task. The fundamental goal is to create a learning curve that has a greater starting point, slope, and asymptote. Conventional machine learning methods that have been around for a long time were created to solve specific challenges. Transfer learning, however, is a deep learning technique that requires first training a model on a single dataset. The layers of the learned model are then reused in a new model that is trained on another dataset. As a result, this strategy can reduce the training time for a CNN. In this study, we apply robust, discriminative filters developed by cutting-edge ImageNet CNN models to detect insect images on which the models have never been trained. Data similarity and data volume are important elements to consider while setting the fine-tuning parameters for this approach. The transfer learning approach was employed through the fine-tuning of EfficientNet models in earlier CNN layers, and then these fine-tuned models were used to extract image features. Following the guidelines in Yosinski *et al.*'s work [49], and since the three insect image datasets described in Section A are tiny and distinct from the original ImageNet dataset. As a result, we keep some of the earlier layers fixed in the first step of training and fine-tune a few of the network's higher-level components. However, in the second pass of training, certain previous layers are unfrozen, and then backpropagation is utilized to fine-tune the weights of the pretrained network. This is based on the belief that the early layers of a CNN reflect more low-level characteristics that should be useful for a variety of tasks, however the successive levels of the CNN become more relevant to the specific information of the classes available in the original dataset. Therefore, the strong discriminative features learned by the pre-trained CNN models may still be used in this manner. Fig. 5 presents the architecture of the fine-tuned EfficientNet-B5 model for extracting deep image features from the datasets, which treats pre-trained networks as feature extractors. There are many layers in EfficientNet-B5 that are connected to each other. EfficientNet-B5 architecture has seven blocks, from block 1 to block 7, and each block has three modules. These modules are actually connected and repeated. Conv2D, Depthwise Conv2D, batch normalization, activation, global average pooling, rescaling, and zero padding are among the layers included. The top activation function in the EfficientNet architecture defines how the weighted sum of the input is transformed into an output from a node or nodes in the top layer of the network. When performing deep learning feature extraction, the pre-trained network is treated as an arbitrary feature extractor, allowing the input image to propagate forward, stopping at a pre-specified layer (the final POOL layer in the red dashed box), and taking the outputs of that layer as our features, with 2048 dimensions for every single feature.

E. Softmax Classifier Versus Power Mean SVM

1) Softmax classifier

Cross-entropy loss is used by the softmax classifier. Before applying the cross-entropy loss, the softmax classifier uses the softmax function to transform raw class scores into normalized positive values that add to one. When using deep learning algorithms, the softmax or 1-of-K encoding is used at the top of classification tasks. For example, if there are 10 classes, the softmax layer comprises ten nodes labeled by p_i, where $i = 1, \cdots, 10$. The notation p_i defines a gaussian distribution, therefore $\sum_i^{10} p_i = 1$. Assuming that W is the weight linking the last layer to the softmax layer and that h denotes the activation of the nodes in the last layer, the total input into the softmax layer is given by a (Eq. (1)).

$$a_i = \sum_k h_k W_{ki} \tag{1}$$

Then we have:

$$p_i = \frac{exp(a_i)}{\sum_j^{10} exp(a_j)} \tag{2}$$

The predicted class i is given by the formula:

$$\hat{\imath} = \arg\max_i p_i = \arg\max_i a_i \tag{3}$$

2) Power mean support vector machine

Consider a linear binary classification on $\mathbb{T} = \{(x_i, y_i)\}_{i=1}^n, x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$ where n is the quantity of training samples. The goal of SVM is to determine the optimal separating surface that is the farthest away from class $y = +1$ and class $y = -1$. It minimizes error while optimizing the distance between each class's

supporting planes. This is accomplished by the solution of the dual optimization problem (Eq. (4)).

$$\min_{\alpha\in\mathbb{R}^n} f(\alpha) = \frac{1}{2}\alpha^{\mathrm{T}}\mathrm{Q}\alpha - \mathrm{e}^{\mathrm{T}}\alpha \quad \text{s.t.} \begin{cases} y^T\alpha = 0 \\ 0 \le \alpha_i \le \mathrm{C}, \forall \mathrm{i} = 1,2,\cdots,n \end{cases} \tag{4}$$

where $e = [1,\cdots,1]_n^T$, C is a positive constant used to tune the margin and the error, $\alpha = (\alpha_1,\cdots,\alpha_n)$ are the Lagrange multipliers, Q is an $n \times n$ symmetric matrix, where $\mathrm{Q}_{ij} = y_i y_j K\langle x_i, x_j\rangle$, and $K\langle x_i, x_j\rangle$ is the kernel function.

The ideal solution of Eq. (4) determines the Support Vectors (SV) (for which $\alpha_i > 0$), and the separation surface and scalar b are subsequently defined by the support vectors. The following Eq. (5) is used to classify a new data point x.

$$sign\left(\sum_{i=1}^{\sharp SV} y_i\alpha_i K\langle x_i, x_j\rangle - b\right) \tag{5}$$

PmSVM [30] proposed that the Power Mean Kernel $M_p\langle x_i, x_j\rangle$ (x_i and $x_j \in \mathbb{R}_+^d$), as described in Eq. (6), substitutes the kernel function $K\langle x_i, x_j\rangle$ in Eqs. (4) and (5), which is well recognized as a generic form of various additive kernels.

$$M_p\langle x_i, x_j\rangle = \sum_{z=1}^{d}\left(x_{i,z}^p + x_{j,z}^p\right)^{\frac{1}{p}} \tag{6}$$

where $p \in \mathbb{R}$ is a constant. χ^2 kernel $(p = -1)$: $M_{-1}(x,y) = K_{\chi^2}(x,y) = \frac{2xy}{x+y}$, Histogram intersection kernel $(p = -\infty)$: $M_{-\infty} = K_{HI}(x,y) = min(x,y)$, Hellinger kernel $(p = 0)$: $M_0(x,y) = \sqrt{xy}$.

The research in [50] has examined the support vector machine as a classification alternative to the softmax function. According to the conducted research, using SVM in an artificial neural network design delivers better results than using the traditional softmax function. While the softmax function reduces cross-entropy or maximizes log-likelihood, SVM only finds the greatest hyperplane between data points of different classes. In this research, a hybrid model that incorporates CNNs and SVM is proposed to classify insect images. In which EfficientNet's softmax classifier has been replaced by the PmSVM classifier. We specifically assess the non-linear classifier PmSVM while classifying image features extracted from EfficientNet models.

III. Results and Discussion

A. Experimental Setup and Training

All numerical evaluation was carried out on an Ubuntu-based workstation with an Intel Core (TM) i7-8565U CPU running at 1.80 GHz and 1.99 GHz and 8 GB of RAM. The Keras deep learning framework and Python programming were used for all of the implementations. All images are converted from RGB (Red Green Blue) to BGR (Blue Green Red), then each color channel is normalized to zero-centered form with batch or layer normalization algorithms. After that, all of the input images were scaled to the standard size that each network model accepts. Accordingly, images were set to 224×224 pixels for MobileNetV2, VGG16, ResNet-50, DenseNet-121, EfficientNet-B0; 229×229 pixels for InceptionV3, Xception; 240×240 pixels for EfficientNet-B1, 260×260 pixels for EfficientNet-B2, 300×300 pixels for EfficientNet-B3, 380×380 pixels for EfficientNet-B4, 456×456 pixels for EfficientNet-B5, 528×528 pixels for EfficientNet-B6, and 600×600 pixels for EfficientNet-B7.

To build a reliable CNN capable of accurately categorizing images, several parameters must be adjusted. The most essential parameter is batch size, which relates to the number of samples utilized to learn a CNN model. The batch size is the quantity of data necessary for backpropagation's weight and bias updates. This value aids learning by optimizing network convergence speeds and allowing for precise prediction. The effect of batch size on CNN training is examined in further depth in Kandel and Castelli's work [51]. Their research found that a larger batch size does not always imply higher classification performance. In this investigation, the batch size was set to 32, which is the maximum amount allowed by computer resources across all models. The other parameters of the networks are set as follows: the gradient degradation factor is 0.9, the squared gradient degradation factor is 0.999, and the loss function is categorical cross-entropy. The optimizer is the Adam optimization algorithm for EfficientNets.

In the training step, transfer learning methods were used to retrain all CNN models. To speed up learning, pre-trained network models on the ImageNet dataset were used to fine-tune CNN models to detect and categorize all categories in the datasets. There are over 1.2 million images and 1000 distinct classifications in the ImageNet. Therefore, the final FC layers of all models with 1000 outputs were altered to 24, 40, and 102 outputs in accordance with Xie24 (24 classes), D0 (40 classes), and IP102 (102 classes). The early-stop technique is used during the training phase if the validation accuracy does not increase after three epochs. The successful models' parameters were preserved for testing at the end of the training process. The number of FC layers in all models was maintained as uniformly as possible in order to better understand and compare their feature extraction performance. The total number of validation photos divided by the batch size was used to validate all models, and each model was trained in two rounds: 1) The first round is called the "warm-up" process; this starts training the networks but only trains the FC layer heads; As shown in the left of Fig. 6, all earlier layers in the network (blue color) are set as untrainable; the learning rate is set at 0.01, and the number of epochs is set at 10 for all network models; 2) In the second round, according to the network architecture, some of the earlier convolutional layers (green color) are unfrozen to perform the second training

phase (as shown in the right of Fig. 6) with a smaller learning rate of 0.0001 and the number of epochs set at 15 for D0, Xie24, and 30 for IP102. Online data augmentation methods are utilized to improve classification performance and minimize overfitting during training.

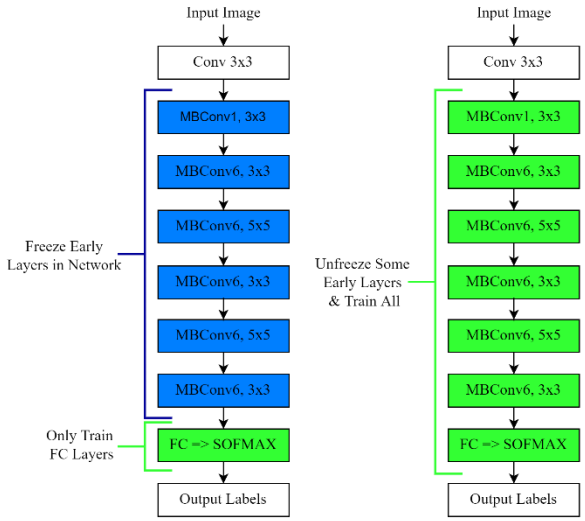

Figure 6. The process of training the CNN models in two rounds.

In the classification step, SVM models are trained with linear and non-linear kernels on the extracted image features datasets. All multi-class classification models were implemented by using a one-versus-all approach. Then the performance of the PmSVM classifier on the insect datasets is compared with that of LIBLINEAR [52] and LIBSVM [53] in terms of classification accuracy. LIBLINEAR is configured with default parameters and $C = 1$. LIBSVM is trained with an RBF Kernel. A grid parameter search method is applied to optimize the LIBSVM parameters (C, gamma, and degree) using cross validation. PmSVM is trained with the parameters as shown in [30], that is $p = -1$ (equivalent to χ^2 kernel) and $C = 0.01$.

B. Evaluation Metrics

The most common method to evaluate the performance of multi-class object classification is to calculate Average Precision (AP) at Eq. (7), Average Recall (AR) at Eq. (8), Average F1-score (AF1) at Eq. (9), and Accuracy (A) at Eq. (10). Precision measures how accurate our model is by calculating the fraction of correctly classified instances or samples among the ones classified as positives. Recall indicates how well the model recalls classes from images; it is a metric that quantifies the number of correct positive predictions made out of all possible positive predictions. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Classification accuracy is a metric that summarizes the performance of a classification model as the number of correct predictions divided by the total number of predictions. The metrics given between Eqs. (7) and (10) are generated utilizing indices such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) based on the values in the confusion matrix produced by such classifications. The number of properly categorized images in each class is represented by TP, whereas the total number of properly identified images in all other classes save the relevant class is represented by TN. FN stands for the number of photos in the relevant class that were incorrectly classified. The number of incorrectly classified images in all other classes save the relevant one is given by FP.

$$Precision(P) = \frac{TP}{TP + FP} \tag{7a}$$

$$Average\ Precision(AP) = \frac{1}{\#classes} \sum_{k=1}^{\#classes} P \tag{7b}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{8a}$$

$$Average\ Recall(AR) = \frac{1}{\#classes} \sum_{k=1}^{\#classes} R \tag{8b}$$

$$F1 - score(F1) = 2 \times \frac{P \times R}{P + R} \tag{9a}$$

$$Average\ F1 - score(AF1) = \frac{1}{\#classes} \sum_{k=1}^{\#classes} F1 \tag{9b}$$

$$Accuracy(A) = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

C. Insect Classification Results

MobileNetV2, VGG16, ResNet-50, DenseNet-121, InceptionV3, Xception, and eight different models of EfficientNets from B0 to B7 have been implemented to evaluate their classification performance on the three datasets as described in Section A. Figs. 7 and 8 present the accuracy and loss curves of the warm-up and unfrozen processes of the EfficientNet-B0 model for training and validation datasets of Xie24. Figs. 9 and 10 demonstrate the accuracy and loss curves of the warm-up and unfrozen processes of the EfficientNet-B0 model for training and validation datasets of D0. Figs. 11 and 12 show the accuracy and loss curves of the warm-up and unfrozen processes of the EfficientNet-B5 model for training and validation datasets of IP102. Tables II and III summarize the performance of network models on the Xie24 and D0 datasets. Table IV summarizes the performance for each network model on the IP102 dataset. The notation "-" means the authors did not report the results. Tables II–IV's bolded values show instances where the best value for the applicable performance criterion is achieved.

D. Discussion

As described in Section A, the insect pest image datasets are small and highly different from the ImageNet dataset, and the training process of network models may suffer from "early overfitting". Therefore, the warm-up processes are employed to reduce the primacy effects of the early training examples. This allows the networks to gradually adapt to the training data, allowing adaptive optimizers to calculate the correct statistics of the gradients. Figs. 7–12 presents the training loss and accuracy of EfficientNet for B0 and B5 on three insect datasets. As shown in Figs. 7, 9, and 11, the accuracy of EfficientNet

models training and validation gradually increases over the warm-up phase at a learning rate of 0.01. However, after 10 epochs, the classification accuracy of CNN models is not improving. Thus, the unfrozen process of training the networks has started with a much smaller learning rate of 0.0001. As could be seen from Figs. 8, 10, and 12, the curve of training and validation accuracy of the EfficientNet models became more stable and achieved high classification performance after a few epochs on several insect datasets. For instance, on the IP102 dataset, it took 30 epochs to obtain the best classification accuracy.

Figure 7. EfficientNet-B0 accuracy/loss curves for the train and test datasets of Xie24 during the warm-up phase.

Figure 8. EfficientNet-B0 accuracy/loss curves for the train and test datasets of Xie24 during the unfrozen process.

Figure 9. EfficientNet-B0 accuracy/loss curves for the train and test datasets of D0 during the warm-up phase.

Figure 10. EfficientNet-B0 accuracy/loss curves for the train and test datasets of D0 during the unfrozen process.

Figure 11. EfficientNet-B5 accuracy/loss curves for the train and test datasets of IP102 during the warm-up phase.

Figure 12. EfficientNet-B5 accuracy/loss curves for the train and test datasets of IP102 during the unfrozen process.

TABLE II. THE CLASSIFICATION PERFORMANCE (%) OF METHODS ON THE XIE24 DATASET

Methods	Image size	Accuracy	Avg. Precision	Avg. Recall	Avg. F1-score
K. Thenmozhi *et al.* [21]	227×227	97.47	-	-	-
DenseNet-121	224×224	90.00	90.00	90.00	90.00
ResNet-50	224×224	98.00	98.00	98.00	98.00
EfficientNet-B0	224×224	**99.00**	**99.00**	**99.00**	**99.00**

TABLE III. THE CLASSIFICATION PERFORMANCE (%) OF METHODS ON THE D0 DATASET

Methods	Image size	Accuracy	Avg. Precision	Avg. Recall	Avg. F1-score
E. Ayan *et al.* [28]	-	98.81	98.88	98.81	98.81
DenseNet-121	224×224	85.00	86.00	85.00	85.00
ResNet-50	224×224	93.00	94.00	93.00	93.00
EfficientNet-B0	224×224	**99.00**	**99.00**	**99.00**	**99.00**

TABLE IV. THE CLASSIFICATION PERFORMANCE (%) OF METHODS ON THE IP102 DATASET. THE ASTERISK (*) DENOTES THE EFFICIENTNET-B0 IS TRAINED FROM SCRATCH WITH 50 EPOCHS

Methods	Params	Image size	Accuracy	Avg. Precision	Avg. Recall	Avg. F1-score
Setiawan et al [18]	3.5M	224×224	71.32	-	-	-
Ayan *et al.* [28]	-	-	67.13	67.17	67.13	65.76
FR-ResNets [26]	31M	224×224	55.24	-	-	54.18
AlexNet	57M	256×256	49.41	-	-	48.22
DenseNet-121	8M	224×224	54.59	-	-	52.97
ResNet-50	26M	224×224	54.19	54.19	54.19	48.22
ResNet-101	45M	224×224	53.07	-	-	52.00
GoogleNet	10M	224×224	52.17	-	-	51.24
VGG16	138M	224×224	51.84	-	-	51.20
MobileNetV2	3.5M	224×224	51.00			
EfficientNet-B0 (*)	5M	224×224	51.00	54.00	51.00	49.00
EfficientNet-B0	5M	224×224	67.00	67.00	67.00	67.00
EfficientNet-B1	8M	240×240	69.00	69.00	69.00	69.00
EfficientNet-B2	9M	260×260	69.00	69.00	69.00	69.00
EfficientNet-B3	12M	300×300	70.00	70.00	70.00	70.00
EfficientNet-B4 (Softmax)	19M	380×380	71.00	71.00	71.00	70.00
EfficientNet-B4+LIBSVM	19M	380×380	49.85	-	-	-
EfficientNet-B4+LIBLINEAR	19M	380×380	69.31	-	-	-
EfficientNet-B4+PmSVM	19M	380×380	71.84	-	-	-
EfficientNet-B5 (Softmax)	31M	456×456	72.00	71.00	72.00	71.00
EfficientNet-B5+LIBSVM	31M	456×456	46.35	-	-	-
EfficientNet-B5+LIBLINEAR	31M	456×456	70.28	-	-	-
EfficientNet-B5+PmSVM	31M	456×456	**72.31**	-	-	-
EfficientNet-B6	43M	528×528	67.00	68.00	67.00	67.00
EfficientNet-B7	66M	600×600	68.00	69.00	68.00	68.00

Our experimental results have shown that the combination of fine-tuning EfficientNet and PmSVM has provided superior classification accuracy compared to previous methods for all three insect image datasets. On the Xie24 dataset, all models produced average accuracy values that were quite similar to one another, as shown in Table II. The EfficientNet-B0 showed an improved accuracy of 99% compared to the method by Thenmozhi *et al.* [21] with 97.47% and other CNN models (DenseNet-121 with 90% and ResNet-50 with 98%). On the D0 dataset, EfficientNet-B0 also obtains the highest accuracy of 99% compared to the methods of Ayan *et al.* [28] with 98.81%, DenseNet-121 with 85%, and ResNet-50 with 93% (Table III).

On the IP102 dataset, Table IV shows that EfficientNet-B5 with our fine-tuning methods provided superior performance than other methods, with a classification accuracy of up to 72% with the softmax classifier. We also

trained the EfficientNet-B0 (*) from scratch; however, the performance is no better than transfer learning methods, despite having up to 50 epoch iterations. As demonstrated in Table IV, EfficientNet-B0 (*) achieved a classification accuracy of 51%, which is not higher than EfficientNet-B0's 67%. This result shows that training the network from the ground up is inefficient in terms of training time and classification performance. We have also evaluated EfficientNet-B6 and EfficientNet-B7, but the accuracy did not increase despite the fact that the network training parameters were quite large (43M and 66M parameters, as shown in Table IV). This phenomenon shows that larger EfficientNet variants may not always imply better performance, especially for applications with less data or classes. The bigger the EfficientNet variation employed in this scenario, the more difficult it is to change the netwoks' hyperparameters. On the other hand, the combination of fine-tuning EfficientNet-B5 with PmSVM (EfficientNet-B5+PmSVM) has outperformed other methods in terms of classification accuracy. As shown in Table IV, our proposed method achieved the highest accuracy with 72.31% for IP102, an improvement of more than 0.84% in the case of EfficientNet-B4 and more than 0.31% in the case of EfficientNet-B5 with the softmax classifier. And then EfficientNet-B5+PmSVM outperformed 0.99% over the method of Setiawan *et al.* [18] with 71.32%, 5.18% over the method of Ayan *et al.* [28] with 67.13%, FR-ResNets [26] with 55.24%, and other CNN models (DenseNet-121 with 54.59%, ResNet-101 with 53.07%). We have evaluated the combination of fine-tuning EfficientNet-B5 with LIBSVM and LIBLINEAR, but the accuracy was 46.35% and 70.28%, respectively (Table IV), not higher than fine-tuning EfficientNet-B5 with the softmax classifier and its combination with PmSVM.

It is worth noting that EfficientNet-B5 has fewer parameters (31M) than other CNN models, resulting in lower processing costs during the training and testing phases. This demonstrates that our proposed method, which combines the fine-tuning of EfficientNet-B5 and PmSVM, has a high capacity for scaling up to a variety of large-scale insect datasets.

IV. Conclusion

An automatic insect classification system plays a vital role in developing smart agriculture. This research has proposed an efficient method based on fine-tuning EfficientNets models and their combination with PmSVM to classify insect images with complex backgrounds and at different stages of their life cycle. Specifically, eight EfficientNets models have been evaluated on several different benchmark insect image datasets, and the best model was picked for extracting the image features. All models were trained over a small number of epochs, but they achieved significantly high accuracy on several datasets for all four criteria: precision, recall, accuracy, and F1-score. Experiments have shown that the combination of fine-tuning EfficientNet-B5 and PmSVM delivers the best performance at the lowest cost during the training and testing phases. It has established a new state-of-the-art classification performance of 72.31% on a large-scale insect dataset, IP102. Moreover, the small size of the EfficientNet-B5 model makes our proposed method easy to apply to several embedded control systems for autonomous machines, such as drones and robots utilized in smart agriculture. However, when the extracted feature dataset of network models is even larger and cannot be kept in the main memory of a computer, PmSVM encounters a problem during the classification stage. Therefore, in the near future, we may explore the method as proposed by Doan *et al.* [54]. They avoid loading the entire dataset into main memory by dividing it into manageable chunks of rows saved in separate files, instead loading one chunk of rows into main memory at a time for learning tasks. In addition, the current largest insect dataset, IP102, is still very small when compared to the number of millions of living insect species as described in [1]. Therefore, creating an insect image dataset with a larger number of classes and images for an automatic insect image classification system remains a major challenge that requires more contributions from computer vision researchers.

Conflict of Interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

[1] N. E. Stork, "How many species of insects and other terrestrial arthropods are there on earth?" *Annu. Rev. Entomol*., vol. 63, pp. 31–45, 2018.

[2] T. Kasinathan, D. Singaraju, and S. R. Uyyala, "Insect classification and detection in field crops using modern machine learning techniques," *Inf. Process. Agric.*, 2020. doi: 10.1016/j.inpa.2020.09.006

[3] C. Muppala and V. Guruviah, "Detection of leaf folder and yellow stemborer moths in the paddy field using deep neural network with search and rescue optimization," *Inf. Process. Agric.*, vol. 8, no. 2, pp. 350–358, 2021. doi: 10.1016/j.inpa.2020.09.002

[4] L. C. Ngugi, M. Abelwahab, and M. Abo-Zahhad, "Recent advances in image processing techniques for automated leaf pest and disease recognition — A review," *Inf. Process. Agric*., vol. 8, no. 1, pp. 27–51, 2021. doi: 10.1016/j.inpa.2020.04.004

[5] J. Liu and X. Wang, "Plant diseases and pests detection based on deep learning: a review," *Plant Methods*, vol. 17, no. 1, pp. 1–18, 2021. doi: 10.1186/s13007-021-00722-9

[6] L. Li, S. Zhang, and B. Wang, "Plant disease detection and classification by deep learning — A review," *IEEE Access*, vol. 9, pp. 56683–56698, 2021. doi: 10.1109/ACCESS.2021.3069646

[7] K. Zou, L. Ge, H. Zhou, C. Zhang, and W. Li, "Broccoli seedling pest damage degree evaluation based on machine learning combined with color and shape features," *Inf. Process. Agric*., 2021. doi: 10.1016/j.inpa.2020.12.003

[8] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, "Computer vision technology in agricultural automation — A review," *Inf. Process. Agric.*, vol. 7, no. 1, pp. 1–19, 2020. doi: 10.1016/j.inpa.2019.09.006

[9] J. G. M. Esgario, P. B. C. de Castro, L. M. Tassis, and R. A. Krohling, "An app to assist farmers in the identification of diseases and pests of coffee leaves using deep learning," *Inf. Process. Agric.*, pp. 1–10, 2021. doi: 10.1016/j.inpa.2021.01.004

[10] R. U. Rani and P. Amsini, "Pest identification in leaf images using SVM classifier," *Int. J. Comput. Intell. Informatics*, vol. 6, no. 1, 2016. doi: 10.13140/RG.2.2.11632.30721

[11] S. Lim, S. Kim, S. Park, and D. Kim, "Development of application for forest insect classification using CNN," in *Proc. 2018 15th Int. Conf. Control. Autom. Robot. Vis.*, 2018, pp. 1128–1131.

[12] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agric.*, vol. 147, no. February, pp. 70–90, 2018. doi: 10.1016/j.compag.2018.02.016
[13] O. L. P. Hansen, *et al.*, "Species-level image classification with convolutional neural network enables insect identification from habitus images," *Ecol. Evol.*, vol. 10, pp. 737–747, 2020.
[14] B. Tugrul, E. Elfatimi, and R. Eryigit, "Convolutional neural networks in detection of plant leaf diseases: A review," *Agriculture*, vol. 12, no. 8, p. 1192, 2022. doi: 10.3390/agriculture12081192
[15] D. J. A. Rustia, *et al.*, "Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method," *J. Appl. Entomol.*, vol. 145, pp. 206–222, 2020.
[16] M. Rahat *et al.*, *Deep CNN-Based Mango Insect Classification*, 2021.
[17] F. Rajeena, *et al.*, "A novel method for the classification of butterfly species using pre-trained CNN models," *Electronics*, vol. 11, no. 13, 2022.
[18] A. Setiawan, N. Yudistira, and R. C. Wihandika, "Large scale pest classification using efficient convolutional neural network with augmentation and regularizers," *Comput. Electron. Agric.*, vol. 200, 107204, 2022. doi: https://doi.org/10.1016/j.compag.2022.107204
[19] J. Andrew, J. Eunice, D. E. Popescu, M. K. Chowdary, and J. Hemanth, "Deep learning-based leaf disease detection in crops using images for agricultural applications," *Agronomy*, vol. 12, no. 10, pp. 1–19, 2022. doi: 10.3390/agronomy12102395
[20] D. Xia, P. Chen, B. Wang, J. Zhang, and C. Xie, "Insect detection and classification based on an improved convolutional neural network," *Sensors (Switzerland)*, vol. 18, no. 12, pp. 1–12, 2018. doi: 10.3390/s18124169.
[21] K. Thenmozhi and U. S. Reddy, "Crop pest classification based on deep convolutional neural network and transfer learning," *Comput. Electron. Agric.*, vol. 164, 104906, 2019. doi: 10.1016/j.compag.2019.104906
[22] C. Xie, *et al.*, "Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning," *Comput. Electron. Agric.*, vol. 119, pp. 123–132, 2015.
[23] C. Xie, *et al.*, "Multi-level learning features for automatic classification of field crop pests," *Comput. Electron. Agric.*, vol. 152, pp. 233–241, 2018. doi: 10.1016/j.compag.2018.07.014
[24] O. Russakovsky, *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis. Vol.*, Sep. 2014.
[25] X. Wu, C. Zhan, Y. K. Lai, M. M. Cheng, and J. Yang, "IP102: A large-scale benchmark dataset for insect pest recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8779–8788. doi: 10.1109/CVPR.2019.00899
[26] F. Ren, W. Liu, and G. Wu, "Feature reuse residual networks for insect pest recognition," *IEEE Access*, vol. 7, pp. 122758–122768, 2019. doi: 10.1109/ACCESS.2019.2938194
[27] L. Nanni, G. Maguolo, and F. Pancino, "Insect pest image detection and recognition based on bio-inspired methods," *Ecol. Inform.*, vol. 57, 2020. doi: 10.1016/j.ecoinf.2020.101089
[28] E. Ayan, H. Erbay, and F. Varçın, "Crop pest classification with a genetic algorithm-based weighted ensemble of deep convolutional neural networks," *Comput. Electron. Agric.*, vol. 179, Dec. 2020. doi: 10.1016/j.compag.2020.105809
[29] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. ICML 2019*, 2019, pp. 10691–10700.
[30] J. Wu, "Power mean SVM for large scale visual classification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2344–2351. doi: 10.1109/CVPR.2012.6247946
[31] Y. Tang, "Deep learning using linear support vector machines," arXiv:1306.0239, 2013. doi: http://arxiv.org/abs/1306.0239
[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, 2012.
[33] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *Proc. 2018 International Interdisciplinary PhD Workshop, IIPhDW 2018*, 2018, pp. 117–122. doi: 10.1109/IIPHDW.2018.8388338
[34] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, 2019. doi: 10.1186/s40537-019-0197-0
[35] Y. LeCun, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989. doi: 10.1162/neco.1989.1.4.541
[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[37] I. Ahmad, M. Hamid, S. Yousaf, S. T. Shah, and M. O. Ahmad, "Optimizing pretrained convolutional neural networks for tomato leaf disease detection," *Complexity*, 2020. doi: 10.1155/2020/8812019
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Dec. 2016, pp. 770–778.
[39] Z. Shi, H. Dang, Z. Liu, and X. Zhou, "Detection and identification of stored-grain insects using deep learning: A more effective neural network," *IEEE Access*, vol. 8, pp. 163703–163714, 2020. doi: 10.1109/ACCESS.2020.3021830
[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308
[41] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. the 30th IEEE Conf. Comput. Vis. Pattern Recognition*, 2017, pp. 1800–1807. doi: 10.1109/CVPR.2017.195
[42] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. the 30th IEEE Conf. Comput. Vis. Pattern Recognition*, 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243
[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474
[44] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6517–6525. doi: 10.1109/CVPR.2017.690.
[45] B. Ramalingam, *et al.*, "Remote insects trap monitoring system using deep learning framework and iot," *Sensors (Switzerland)*, vol. 20, no. 18, pp. 1–17, 2020. doi: 10.3390/s20185280.
[46] L. Nanni, A. Manfè, G. Maguolo, A. Lumini, and S. Brahnam, "High performing ensemble of convolutional neural networks for insect pest image detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2021.
[47] M. Tan *et al.*, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2815–2823. doi: 10.1109/CVPR.2019.00293
[48] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2921–2929. doi: 10.1109/CVPR.2016.319
[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Adv. Neural Inf. Process. Syst.*, vol. 4, pp. 3320–3328, 2014.
[50] A. A. A. Ali and S. Mallaiah, "Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout," *J. King Saud Univ. - Comput. Inf. Sci.*, pp. 1–7, 2021. doi: 10.1016/j.jksuci.2021.01.012
[51] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312–315, 2020. doi: 10.1016/j.icte.2020.04.010
[52] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, no. 2008, pp. 1871–1874, 2008. doi: 10.1145/1390681.1442794
[53] C. C. Chang and C. J. Lin, "LIBSVM: A Library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–39, 2011. doi: 10.1145/1961189.1961199
[54] T.-N. Doan, T.-N. Do, and F. Poulet, "Large scale classifiers for visual classification tasks," *Multimed. Tools Appl.*, vol. 74, pp. 1199–1224, 2014.

Thanh-Nghi Doan received his doctorate degree in computer science from University of Rennes 1, France, in 2013. He has worked as a Ph.D. candidate in TEXMEX Research Team, IRISA, France. He is currently working at An Giang University, Vietnam National University Ho Chi Minh City, Vietnam. His research is mainly focused on machine learning, data mining and high-performance computing in computer vision, agricultural applications including insect pest image classification systems, prediction of the damage of rice diseases. Especially he made a major contribution to 2D and 3D image understanding systems in agriculture in Mekong Delta.