


Article

GDnet-IP: Grouped Dropout-Based Convolutional Neural Network for Insect Pest Recognition

Dongcheng Li, Yongqi Xu, Zheming Yuan  and Zhijun Dai *

Hunan Engineering & Technology Research Center for Agricultural Big Data Analysis & Decision-Making, Hunan Agricultural University, Changsha 410128, China; dongchengli287@gmail.com (D.L.); xuyongqi@stu.hunau.edu.cn (Y.X.); zhmyuan@hunau.edu.cn (Z.Y.)

* Correspondence: daizhijun@hunau.edu.cn

Abstract: Lightweight convolutional neural network (CNN) models have proven effective in recognizing common pest species, yet challenges remain in enhancing their nonlinear learning capacity and reducing overfitting. This study introduces a grouped dropout strategy and modifies the CNN architecture to improve the accuracy of multi-class insect recognition. Specifically, we optimized the base model by selecting appropriate optimizers, fine-tuning the dropout probability, and adjusting the learning rate decay strategy. Additionally, we replaced ReLU with PReLU and added BatchNorm layers after each Inception layer, enhancing the model's nonlinear expression and training stability. Leveraging the Inception module's branching structure and the adaptive grouping properties of the WeDIV clustering algorithm, we developed two grouped dropout models, the iGDnet-IP and GDnet-IP. Experimental results on a dataset containing 20 insect species (15 pests and five beneficial insects) demonstrated an increase in cross-validation accuracy from 84.68% to 92.12%, with notable improvements in the recognition rates for difficult-to-classify species, such as *Parnara guttatus* Bremer and Grey (PGBG) and *Papilio xuthus* Linnaeus (PXLL), increasing from 38% and 47% to 62% and 93%, respectively. Furthermore, these models showed significant accuracy advantages over standard dropout methods on test sets, with faster training times compared to four conventional CNN models, highlighting their suitability for mobile applications. Theoretical analyses of model gradients and Fisher information provide further insight into the grouped dropout strategy's role in improving CNN interpretability for insect recognition tasks.



Citation: Li, D.; Xu, Y.; Yuan, Z.; Dai, Z. GDnet-IP: Grouped Dropout-Based Convolutional Neural Network for Insect Pest Recognition. *Agriculture* **2024**, *14*, 1915. <https://doi.org/10.3390/agriculture14111915>

Academic Editor: Roberto Alves Braga Júnior

Received: 4 September 2024
Revised: 19 October 2024
Accepted: 21 October 2024
Published: 29 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: common pests; lightweight CNN; insect image recognition; grouped dropout; activation function; batch normalization

1. Introduction

Pests have long imposed significant challenges on agriculture and human society, such as the Irish potato famine caused by *Phytophthora infestans* in the 1840s, the spread of coffee rust in Ceylon by *Hemileia vastatrix* in the 1860s, and the Bengal famine of 1943 caused by *Helminthosporium oryzae* [1]. Traditionally, the extensive use of pesticides has been the primary approach for pest management, but this has led to issues like pesticide resistance, water pollution, and other health hazards [2]. Consequently, there is a growing need for sustainable and efficient pest control measures. Insect monitoring, which enables early pest detection, is crucial to reducing pesticide dependency and enhancing crop quality and yield [3]. However, conventional monitoring methods involve manual insect recognition and counting, which are time-consuming and error-prone [4]. Therefore, while traditional pest management approaches have helped, they also show clear limitations, highlighting the need for modern, technology-driven solutions.

With the rapid development of artificial intelligence (AI), smart pest monitoring involving automatic data acquisition, remote sensing, big data analysis, and decision making has emerged and advanced enormously in recently years [5]. Machine learning (ML) involves

using algorithms and statistical models that allow systems to learn from data without explicit programming [6]. A simple image processing system that automatically estimates the density of whiteflies on sticky traps was developed using noise elimination and image binarization techniques [7]. However, this system was only suitable for detecting whiteflies on sticky traps. A system utilizing multiple-task sparse representation and multiple-kernel learning techniques was proposed for insect recognition, in which the techniques were used to combine multiple features of insect species to enhance the model's recognition performance [8]. This method's limitations include a high level of computational complexity due to sparse coding and multiple kernel learning, dependence on high-quality images for optimal performance, and limited scalability for datasets with large sample size and higher-dimensional features. To achieve automatic pest detection for strawberry greenhouse monitoring against pest attacks, a support vector machine with different kernel function was used for the classification of parasites and detection of thrips [9]. Despite demonstrating a strong performance in insect detection, ML-based algorithms encounter challenges such as time efficiency and accuracy constraints, which stem from the process of manually extracting image features and the substantial growth in image samples. Although ML-based algorithms demonstrate strong performance in certain specific tasks of insect detection, they often face challenges in terms of time efficiency and accuracy during feature extraction, especially with the rapid growth of image samples. Regularization can effectively improve model generalization and reduce overfitting, helping maintain robustness and accuracy across diverse datasets when dealing with complex data like image features.

In contrast, deep learning employs neural networks with multiple layers to automatically learn and extract features from data [4]. By utilizing convolutional operations, pooling, and fully connected layers, convolutional neural networks (CNNs) are an important branch of deep learning and are particularly effective for insect image recognition and classification [4]. To address the challenge of the multi-classification of crop insects, a CNN model based on an improved VGG19, a pretrained CNN 19 layers deep, was proposed for rapid and accurate insect detection in images [10]. The application of lightweight neural networks has also gradually become popular. A lightweight convolutional neural network named BerryNet-Lite was designed for accurate strawberry disease identification. By leveraging transfer learning and incorporating techniques like dilated convolution and efficient channel attention, BerryNet-Lite achieves a high accuracy with a low level of computational demand, making it suitable for real-time deployment in resource-limited environments [11]. Addressing the challenge of accurate potato pest identification, PotatoPestNet employs a customized CTInceptionV3-RS architecture [12]. With optimized hyper-parameters and regularization techniques, the model reduces the overfitting of various datasets. Its lightweight structure and computational efficiency make it highly practical for real-world agricultural applications, aiding farmers in timely pest management [12]. However, several issues remain to be addressed in this model, including augmenting the insect database, extracting relevant insect-like areas, and segmenting the periods of insect growth. By leveraging the advantages of Inception modules adopted by GoogleNet to specify different filter sizes within each block, a fine-tuned GoogleNet model was proposed to handle the complex backgrounds in farmland scenes. This model achieved a 6.22% improvement in pest classification accuracy compared to the state-of-the-art method [13]. A lightweight CNN model named CPAFNet (Common Pests in Agriculture and Forestry) was designed for pest identification using a specially constructed image dataset. This model, involving three specific Inception modules and optimized training parameters, presented a higher recognition accuracy and reduced training time compared to traditional CNN models [14]. Ensembles of CNNs, including EfficientNetB0, ResNet50, and GoogleNet, were developed using various data augmentation and Adam optimization techniques for pest identification. The top-performing ensemble matched human expert classifications on the Deng dataset and achieved state-of-the-art results on three insect datasets [15]. Although CNN models have achieved significant improvements in classification accuracy on image-based

pest datasets, the fine-tuning of network structure and training parameters often leads to overfitting problems.

Regularization is also an important method for controlling model complexity. In the training of CNNs, regularization methods are often employed to reduce the risk of overfitting, resulting in an improved generalization ability [16]. Dropout is a crucial regularization technique that mitigates overfitting by randomly deactivating neurons during the training process, effectively generating thinned networks. During testing, the effect of averaging the predictions of these thinned networks is easily approximated, significantly reducing overfitting and offering major improvements over other regularization methods [17]. However, for fully convolutional networks and natural images with a strong spatial correlation, standard dropout often fails [18]. Spatial dropout addresses this by dropping the entire channel (feature map) instead of individual neurons, encouraging the network to learn more robust features across channels [18]. Though effective, the randomly sampled sub-models caused by dropout during training are inconsistent with the full model used during model inference. To address this issue, a new training strategy called R-Drop was proposed. R-Drop regularizes dropout by forcing the output distributions of different sub-models generated by dropout to be consistent with each other [19]. Experiments on various deep learning tasks such as neural machine translation, language modeling, and image classification have shown that R-Drop is universally effective [19].

A common drawback of the mentioned dropout methods is that they fail to account for redundant information among feature maps generated within the same layer. This results in feature maps capturing similar information, which may not be uniformly retained and could be lost during the deactivation process. To address this issue, we propose a hypothesis that deactivating the group(s) of feature maps with similar expressions would enhance the effect of regularization and thus improve the generalization ability of a CNN model. Then, the key problem is how to design or categorize those relevant feature maps to form a representative group. The Inception module adopted by a CNN model has proven effective at classifying pests [13,14], as its multiple branches enrich feature information with different filter sizes. The feature maps within each branch can be intrinsically categorized into one group. However, the effectiveness of grouping by Inception is severely limited due to the manual design of the Inception structure and the variable dropout probabilities specified for different Inception modules. Aiming at addressing the disadvantages of Inception, we employ the clustering algorithm named WeDIV, which we developed recently [20], to group feature maps (or channels) at the tail of the network and then perform grouping dropout. For convenience, the proposed method is named the Grouped Dropout-based Network for Insect Pest recognition (GDnet-IP). The classification performance of the GDnet-IP with different dropout strategies was evaluated and compared with traditional CNN models using the Common Pests in Agriculture and Forestry (CPAF) dataset. Moreover, to enhance the interpretability of the proposed CNN model, the effectiveness of grouped dropout was also explained through analyses of model gradients [21] and Fisher information [22].

2. Dataset and Methods

2.1. Description of Insect Pest Dataset

The dataset used for evaluating the effectiveness of the proposed CNN model is the CPAF dataset [14], containing 4909 original images of 20 insect species (11 fruit tree pests, 4 cereal pests, and 5 beneficial insects). The name and sample size for each insect species are listed in Table S1. Among the original images, 973 were taken with a smartphone and 3936 were collected online. To enhance the network's generalization ability, data augmentation techniques were employed to expand the data volume. Each image was augmented through flipping (from 2 directions), rotating from 0 to 90 degrees in increments of 15 degrees, scaling by 0.8, 0.9, 1.1, and 1.2, respectively, adding Gaussian noise with the sigma set to 12, and transforming with fancy PCA. This resulted in 68,726 enhanced images, totaling 73,635 insect images.

For fair comparison, we followed the same data division as the dataset constructed in [12], in which the dataset was divided into three folds. To ensure that hyper-parameter optimization was performed only on the training set, we initially randomly extracted approximately 1/10th of the images (including the original and corresponding augmented images) from each of the three folds to form an independent Test Set 1, while the remaining images were used for hyper-parameter optimization (Table 1). During the hyper-parameter optimization stage, one fold (without any test samples) was used as the validation set, while the images in the other two folds served as the training set (more precisely, a sub-training set). Moreover, to validate the effectiveness of the models in real scenarios in which not all augmentations are available, only the original images from Test Set 1 were selected to form Test Set 2. For predictions on either test set, the model with optimal hyper-parameters was trained on all images except those included in Test Set 1. The training environment and the hyper-parameters to be optimized in the CNN model are detailed in Table 2.

Table 1. The number of images included in the training, validation, and test set.

	1st Fold	2nd Fold	3rd Fold	Total
Training/ Valid set	1483 (20,762)	1484 (20,776)	1484 (20,776)	4451 (62,314)
Test Set 1	152 (2128)	152 (2128)	154 (2156)	458 (6412)
Test Set 2	152	152	154	458
Total	1635 (22,890)	1636 (22,904)	1638 (22,932)	4909 (68,726)

Note: The number in parentheses represents the augmented images. During cross-validation for optimizing the hyper-parameters (first line), images in one fold were used as the validation set, while the remaining images were used as the training set. In the independent testing stage, the model with optimal hyper-parameters was trained on a total of 66,765 images. Test Set 1 comprised the original and corresponding augmented images randomly extracted from each of the 3 folds, while Test Set 2 included only the original images extracted.

Table 2. Details of computing environment and hyper-parameters to be optimized.

Environment	Item Name	Version/Values
Hardware	CPU	Intel(R) Platinum 8255C
	GPU	Nvidia GTX3080
Software	Pytorch	1.1.0
	Python	3.7
	CUDA	10.0
Hyper-parameters	Optimizer	Adam/AdamW/RMSProp
	Dropout probability	0.2/0.5/0.8 0.002/0.008/
	Learning rate	0.0002/0.0008/0.0008 (with decay)
	Activation function	ReLU/PReLU
	BatchNorm	With/Without

2.2. Development of Gdnet-IP Model

The Gdnet-IP is developed from the CPAFNet model [12]. It features four convolutional layers and three Inception modules, each followed by a max pooling layer. The last module is followed by a global pooling layer (see Figure 1A). This design utilizes layer-by-layer down-sampling to achieve a larger receptive field and extract higher-level features from input images. The main Inception module expands the network's width with four branches, in which the features computed with different filter sizes are combined to enhance their representation, ensuring richer information and reducing redundancy.

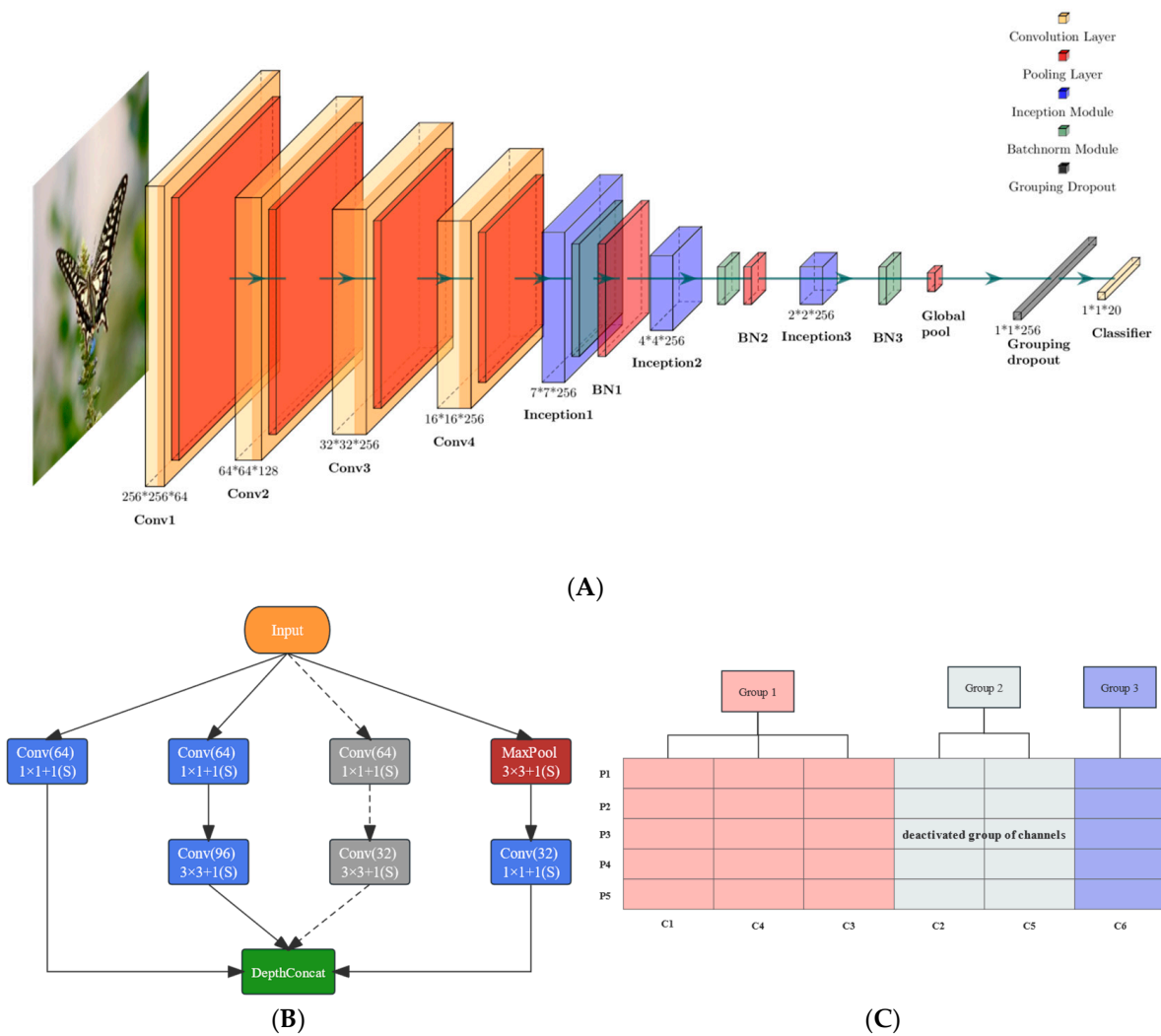


Figure 1. Architecture of GDnet-IP. (A) BatchNorm modules (BN) and clustering-based grouped dropout are introduced into the initial CPAFNet model; (B) Inception-based grouped dropout, in which one of the branches (grey branch) is randomly deactivated; (C) Clustering-based grouped dropout, in which the channels in ‘Group 2’ are randomly deactivated.

To enhance CPAFNet’s nonlinear learning capability, we replaced the default ReLU activation function with the Parametric Rectified Linear Unit (PreLU). Additionally, we added batch normalization layers after each Inception module to stabilize and accelerate network training. Based on the model with optimal hyper-parameters, two types of grouped dropout methods (i.e., Inception-based and clustering-based) were designed and introduced to form the improved Gdnet-IP model (Figure 1).

2.2.1. Reconstruction of CPAFNet

Before performing additional operations to enhance the accuracy of CPAFNet model, we tested its reliability based on the optimal hyper-parameters, data division, and performance assessment presented in [14]. However, the classification accuracy was found to be significantly lower than that in the reference. To address this, we first reconstructed the CPAFNet by setting different values for each hyper-parameter that needed optimization (Table 2). Specifically, the optimization algorithms considered include Adam [23], AdamW [24], and RMSProp [25]; and the basic dropout probabilities tested were 0.2, 0.5, and 0.8. For convenience, the optimal model from this step is named optiCPAFNet-1. Next, we experimented with learning rates set to 0.002, 0.008, 0.0002, 0.0008, and 0.0008 with

decay. The model optimized in this step is named optiCPAFNet-2 and is considered the recurrent and reliable CPAFNet.

2.2.2. Incorporating PreLU and BatchNorm

The ReLU activation function involving a simple threshold operation ($f(x) = \max(0, x)$) allows the network to overcome the vanishing gradient issue and produce sparse representations. This sparsity can lead to more efficient computations and help prevent overfitting. However, the biggest issue with ReLU during training is the occurrence of dead neurons [26]. PreLU addresses this issue by allowing a small, learnable negative slope, which helps keep neurons active. The flexibility of having a learnable parameter for negative slopes can lead to improved nonlinear learning in some cases, as the network can adapt better to the data [27].

In addition, batch normalization (often abbreviated as BatchNorm) is a powerful technique used in CNNs to improve training efficiency and model performance. BatchNorm normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation, followed by scaling and shifting operations using learnable parameters. By normalizing the outputs of layers, BatchNorm addresses issues related to internal covariate shifts, allows for higher learning rates, reduces dependency on initialization, and acts as a regularizer [28]. Therefore, we tested whether the combination of PreLU and BatchNorm operations could improve the classification accuracy based on the reliable CPAFNet model previously obtained. The model optimized in this step is named optiCPAFNet-3. The computing environment details and hyper-parameters to be optimized are listed in Table 1.

2.2.3. Grouped Dropout Based on Inception Modules or Clustering

Research into the interpretability of CNNs shows that feature maps produced by convolutional kernels do not operate independently but interact and collaboratively train a network [29]. This finding aligns with research suggesting that interactions and interpretability among different feature maps in hidden layers are not merely random linear combinations [30]. Inspired by this, we propose a hypothesis that feature maps exhibiting interactions and/or similar representation form groups to capture specific information. However, not all of the information captured by different groups is universally useful or improves the classification accuracy of a CNN model. Dropping group(s) out of feature maps with specific probabilities during training iterations could be an ideal alternative.

The Inception module with four branches adopted by the CPAFNet model captures enriched features using different filter sizes in each branch [14]. However, redundancy among these branches may still exist. Therefore, the grouped dropout strategy based on the Inception module, named the iGDnet-IP, was designed to test its contribution to the classification performance of neural networks (Figure 1B). There are three Inception modules sequentially deployed within the CPAFNet model. Performing grouped dropout on the first module would result in the loss of important features learned from shallow layers. Thus, grouped dropout was conducted only on the second and third modules. Low-level features detected in the shallow layers are generally considered to represent more basic and richer information than high-level features obtained from deep layers. Therefore, the drop probabilities to be optimized for the second module were set to 0.1, 0.2, and 0.3, while for the third module, they were set to 0.2, 0.3, and 0.4. This setting ensures that more information is preserved in the second module compared to the third module.

Considering the challenges of specifying different drop probabilities for the Inception-based Gdnet-IP model, we adopted a new strategy: automatically grouping feature maps using a clustering algorithm, followed by grouped dropout. However, clustering feature maps with 4 dimensions (length, width, depth, and the number of images) in hidden layers is both difficult and time-consuming. Thus, we performed clustering only on the tail (i.e., the output of the global pooling layer) of the network. The tail is a vector of size $1 \times 1 \times f$

(the number of filters) for a single image (Figure 1A) and a matrix of size $n \times f$ for n images in a batch, making it suitable for clustering.

We utilized the clustering algorithm WeDIV we recently proposed for grouping the feature maps [20]. The WeDIV is an improved k -means clustering algorithm incorporating weighted distance and a novel internal validation index. The former can effectively capture information with both global spatial correlation (reflected by Euclidean distance) and local variable trends (represented by Pearson distance) among features, while the latter is able to automatically estimate the optimal number of clusters. A hyper-parameter, w , in the weighted distance needs to be optimized within the range of $[0, 1]$ with a specified step (0.25 or 0.2 in this study) to measure the relative contributions of the two distances. The newly proposed model based on the clustering-based grouped dropout is named the Gdnet-IP (Figure 1A,C).

2.3. Reference Models

To evaluate the effectiveness of the proposed models in this study, we used the reliable CPAFNet (i.e., the optiCPAFNet-2) and four famous models (VGG11, VGG16, ResNet50, and Inception V3), which were compared with CPAFNet in [14] as reference models. In addition, two traditional dropout methods (Spatial-Dropout [18] and R-Drop [19]) were also compared to assess the performance of the two grouped dropout strategies.

The VGG11 (with 11 weight layers) and VGG16 (with 16 layers) models are part of the VGGNet family [31], which are widely used in image classification due to their simplicity and effectiveness. The ResNet50 (with 50 layers) is known for its use of residual blocks, which allows for training very deep networks by addressing the vanishing gradient problem [32]. This is achieved through shortcut connections within the residual blocks which enable gradients to propagate directly through multiple layers, improving training stability and accuracy [33]. The Inception V3 network is a powerful and efficient deep CNN architecture that builds upon the inception module, factorized convolutions, auxiliary classifiers, and batch normalization [34]. Inception V3 also incorporates factorized convolutions, auxiliary classifiers, and batch normalization to further optimize performance and computational efficiency, making it highly effective for complex image classification tasks [33].

2.4. Model Evaluation

Considering the imbalanced samples among different insect species, balanced accuracy (BA, Equation (1)) was computed as the evaluation metric. In Equation (1), N is the number of categories, n_i is the number of images in the i -th category, and m_i represents the number of images correctly predicted in this category. The batch size and the epoch for training were set to 256 and 40, respectively. The average BA of the training iterations in last epoch was computed as the identification accuracy. Additionally, we discussed the reasons why the grouped dropout-based network models perform well in classifying insect pest images by analyzing model gradients [21] and Fisher information [22]. The code implementing the two grouped dropout-based CNN models is written in Python and is available at <https://github.com/ZhijunBioinf/GDnet-IP> (accessed on 20 October 2024).

$$BA = \frac{1}{N} \sum_{i=1}^N \frac{m_i}{n_i} \quad (1)$$

3. Results and Discussion

3.1. Improving Model's Performance with Hyper-Parameter Optimization

We first performed hyper-parameter optimization to improve the performance of the baseline model using the cross-validation of the three-fold data utilized in [14]. The optimization paths of different hyper-parameters and their cross-validation accuracy are shown in Figure 2.

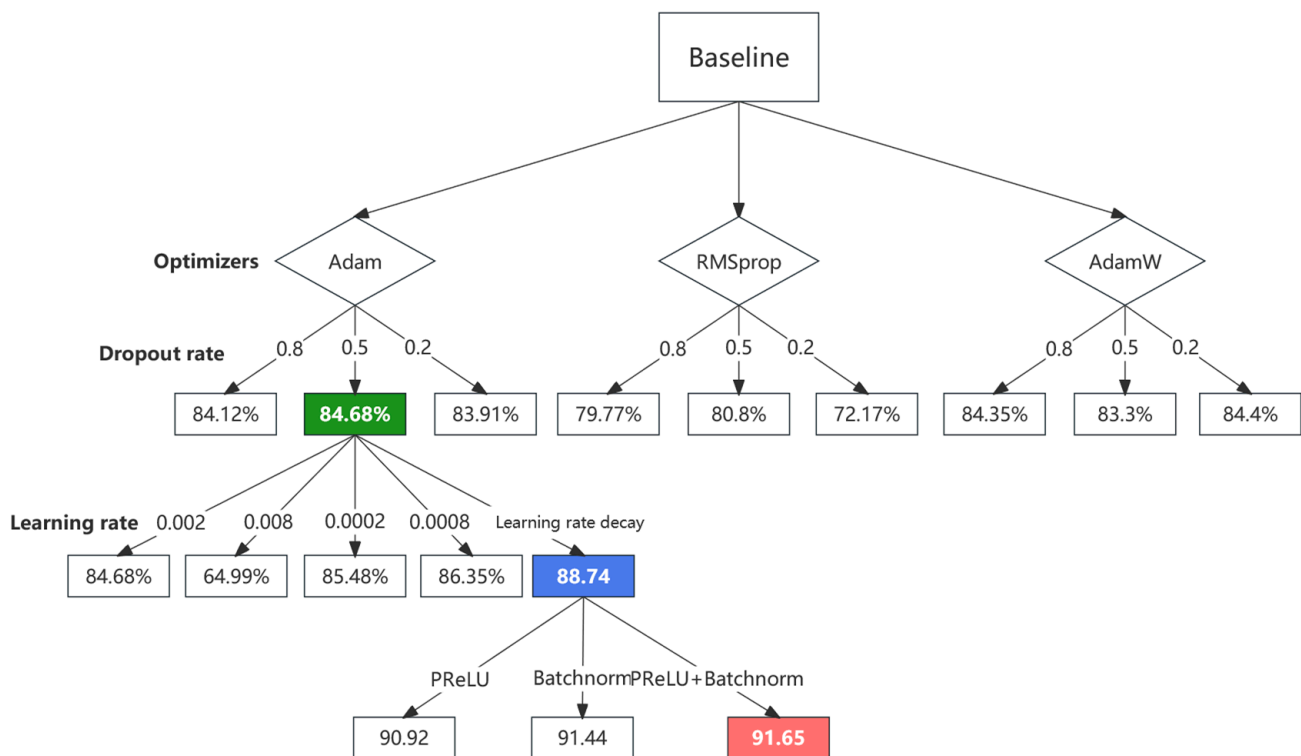


Figure 2. Optimization path of hyper-parameters for the CPAFnet baseline model. The classification accuracy (BA) colored in green represents optiCPAFnet-1 with an Adam optimizer and a classic dropout rate of 0.5. The BA in blue represents optiCPAFnet-2 (considered a reliable reproduction of the CPAFnet model) with an optimized learning rate of 0.0008 and a decay constraint. The BA in red represents optiCPAFnet-3 with the PreLU activation function and the inclusion of BatchNorm modules.

3.1.1. Impact of Optimizer and Learning Rate on Insect Classification Accuracy

First, the performance of three optimizers (Adam, AdamW, and RMSprop) was validated. We found that the model's classification performance was better with the Adam optimizer, with the model using AdamW performing comparably and that using RMSprop performing relatively worse. This may be due to the Adam optimizer's momentum mechanism, which adds an exponentially weighted average after completing the gradient calculation in each iteration. It causes certain neuron weights to update more significantly, helping prevent the model from getting stuck in local minima. Further tests with various dropout probabilities and optimizer combinations showed that the model (optiCPAFNet-1) performed best with the Adam optimizer and a dropout probability of 0.5 (with the accuracy shown in green in Figure 2). This suggests that a moderate dropout probability effectively prevents overfitting while retaining sufficient useful features.

Then, the impact of the learning rate on the model classification accuracy was validated. We found that the cross-validation accuracy of the optiCPAFNet-1 model increased steadily as the learning rate decreased (from 64.99% to 86.35%), reaching its maximum at a learning rate of 0.0008, while slightly decreasing at a learning rate of 0.0002 (85.48%). This suggests that a high learning rate (e.g., 0.008) causes the parameter updates to be too large during training, preventing the model from converging. On the other hand, a low learning rate (e.g., 0.0002) can yield a good classification accuracy but may slow down training, requiring more time to reach the optimal solution. Furthermore, after applying a learning rate decay strategy in the optiCPAFNet-2 model, the accuracy improved further to 88.74% (the accuracy is shown in blue in Figure 2).

By comparing the changes in the loss function of the model with and without decay constraints at a learning rate of 0.0008 (Figure 3), we found that around 700 training

iterations (shown with the vertical dashed line in Figure 3), the loss function gradually started to fit and did not decline significantly. This indicates that the model learned features with a certain amount of information from the data, but the loss function change without decay constraints is still very noticeable (as shown by the red line in the inset of Figure 3), making it difficult for the model to converge stably. However, with the introduction of the decay strategy, the variance in the model's loss function significantly decreased (as shown by the blue line in the inset of Figure 3), indicating that this strategy helps the model update parameters with smaller steps in the later stages of training, allowing for a more precise exploration of the parameter space and gradually approaching the global optimal solution.

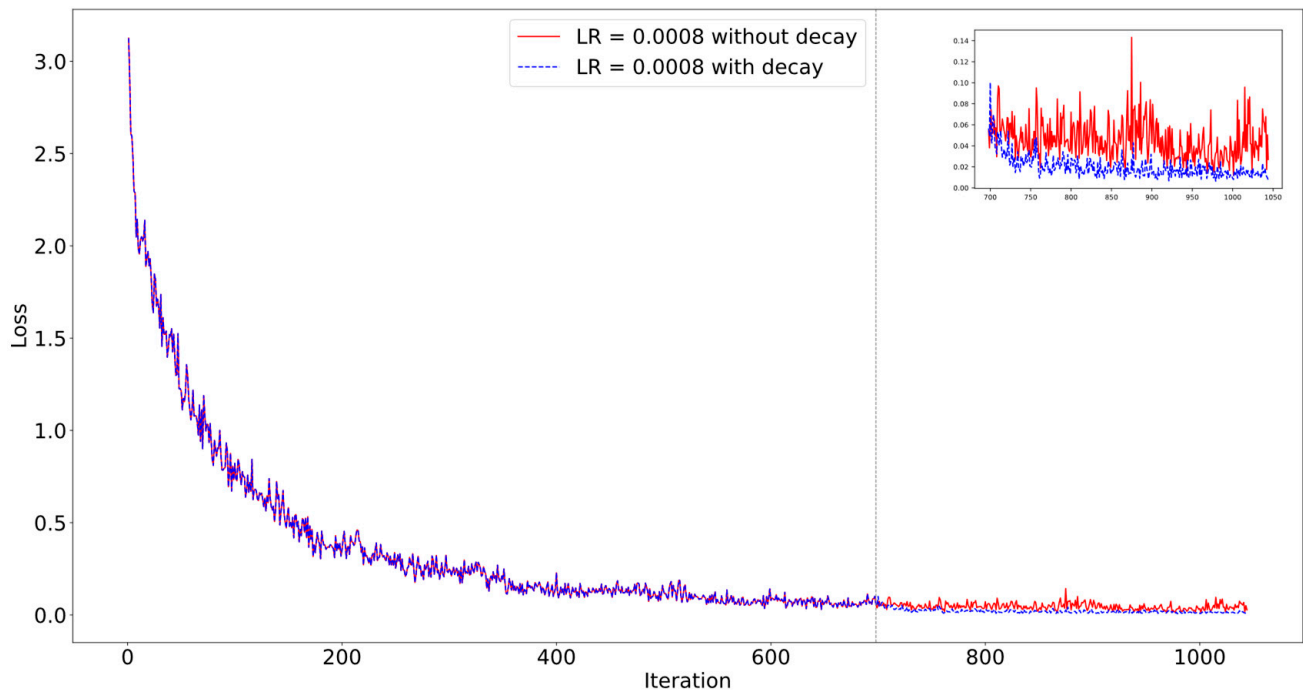


Figure 3. Comparison of the model's loss function changes before and after adding the learning rate decay. The subgraph in the upper-right-hand corner is a zoomed-in view after 700 training iterations; the red curve represents no learning rate decay, while the blue curve represents learning rate with decay.

3.1.2. Impact of PreLU and BatchNorm on Insect Classification Accuracy

Based on the optiCPAFNet-2 model, we sequentially performed PreLU activation function replacement, BatchNorm module addition, and an operation for their combination. We found that the model (optiCPAFNet-3) performed best when both operations were added simultaneously, achieving a classification accuracy of 91.65% (the accuracy is shown in red in Figure 2). Using the accuracies of the final nine epochs of the training iterations (Table S2) to conduct hypothesis testing, the performance of optiCPAFNet-3 is significantly better than that of the optiCPAFNet-2 model (one-tailed t -test, $p = 6.11 \times 10^{-13}$). Additionally, adding the BatchNorm module alone (91.44%) provided a more significant improvement than PreLU (90.92%).

The Introduction of the BatchNorm module, which normalizes the mean and scales the variance in the feature maps in each mini-batch, reduces the variation in input distribution across different layers of the model. This allows each layer to train on a more stable data distribution. Additionally, the scaling and shifting parameters in BatchNorm are used to restore the network's capacity to express inputs, reducing the information loss caused by normalization. Recent studies have found that layer normalization (LN) and its simplified version, RMSNorm, possess nonlinear expression capabilities. When the network structure is sufficiently deep, a simple neural network, LN-Net, containing only

linear layers and LN can classify given samples and sample categories arbitrarily [35]. This discovery challenges the conventional view of various normalization techniques as linear transformations, showing that nonlinear layers and normalization layers are no longer mutually exclusive neural network modules. PreLU introduces a learnable slope parameter, giving negative input values a non-zero gradient, thus addressing the “neuron death” problem caused by the default activation function ReLU. This design provides neural networks with richer nonlinear expression capabilities, helping capture more complex data patterns.

3.2. Further Improving Model’s Performance with Grouped Dropout

Based on the previously optimized optiCPAFNet-3 model, we introduced grouped dropout based on Inception modules (the iGDnet-IP) and a clustering analysis (the Gdnet-IP) separately and validated the impact of each of them on the classification accuracy.

3.2.1. Impact of Inception-Based Grouped Dropout on Insect Classification Accuracy

By simultaneously deactivating the second Inception module (with dropout probabilities of 0.1, 0.2, and 0.3) and the third module (with probabilities of 0.2, 0.3, and 0.4), we found that setting higher dropout probabilities in shallower Inception layers is detrimental to model fitting, whereas deeper Inception layers can accommodate higher dropout probabilities (Table 3). This may be because deeper layers tend to contain more abstract and complex feature representations, which can be learned in part of the branches in the Inception module; dropping a branch with varying probabilities has a smaller impact on the entire module. When the dropout probability for the second Inception module was 0.1, the model’s classification performance remained similar across different dropout probabilities for the third Inception module. Conversely, when fixing the dropout probability of the third Inception module, the model performance worsened as the dropout probability of the second Inception module increased (Table 3). This is because the features learned in shallower layers are more fundamental and relatively important, and complementary information can be learned by different branches. Deactivating any branch affects the learning performance of the entire module, thereby impacting the model’s final classification performance. These results also validate the rationale in setting different dropout probabilities for different Inception layers (modules). Compared to the optiCPAFnet-3 model (Table S2), the iGDnet-IP model achieved a significant improvement in cross-validation accuracy for insect classification ($p = 3.47 \times 10^{-6}$).

Table 3. Cross-validation performance of the model with different grouped dropout probabilities in the two Inception modules.

Prob B ^b	Prob A ^a		
	0.1	0.2	0.3
0.2	92.41%	92.34%	91.26%
0.3	92.39%	92.11%	91.31%
0.4	92.40%	92.31%	91.38%

^a: Grouped dropout probabilities set in the second Inception module; ^b: Probabilities set in the third Inception module.

3.2.2. Impact of Clustering-Based Grouped Dropout on Insect Classification Accuracy

For the output of the global pooling layer in the neural network, each image generates a vector with a length of 256, resulting in an $n \times 256$ matrix for each mini-batch, in which n is the number of samples and 256 is the number of feature channels. Using the WeDIV algorithm to cluster feature channels allows for the adaptive determination of the optimal number of clusters, followed by grouped dropout. However, estimating the optimal number of clusters for each iteration over thousands of training iterations increases the model’s computational complexity. Therefore, in this study, clustering was performed every 50 iterations, and the distribution of the number of clusters was analyzed

to obtain a general optimal number of clusters, thereby improving modeling efficiency. We found that the frequency of a cluster number of four was significantly higher than other cluster numbers (Figure 4), but fewer clusters did not necessarily mean a higher classification accuracy. Additionally, when optimizing the distance (the Euclidean distance and correlation distance) weights in the WeDIV algorithm, the step size needs to be defined, and different cluster numbers combined with different step sizes result in different model classification performances. When the step size was 0.25, the optimal number of clusters was five; when the step size was 0.2, the optimal number of clusters was four, achieving the best cross-validation accuracy of 92.12% (Figure 4). Compared to the optiCPAFnet-3 model (Table S2), the Gdnet-IP model also achieved a significant improvement in cross-validation accuracy for insect classification ($p = 3.37 \times 10^{-5}$), preliminarily indicating the effectiveness of the group deactivation strategy. Meanwhile, the model with WeDIV reached a higher cross-validation accuracy across different configurations, significantly outperforming the model involving the traditional K-means clustering (Table 4). The results suggest that, compared to K-means, the weighted distance in the WeDIV clustering algorithm may capture more complex data patterns and relationships between feature maps. Additionally, unlike K-means, which requires manually specifying the number of clusters, WeDIV can achieve adaptive clustering of feature maps, reducing reliance on network structure and data variability. These two factors make the group dropout based on WeDIV more efficient in insect image recognition.

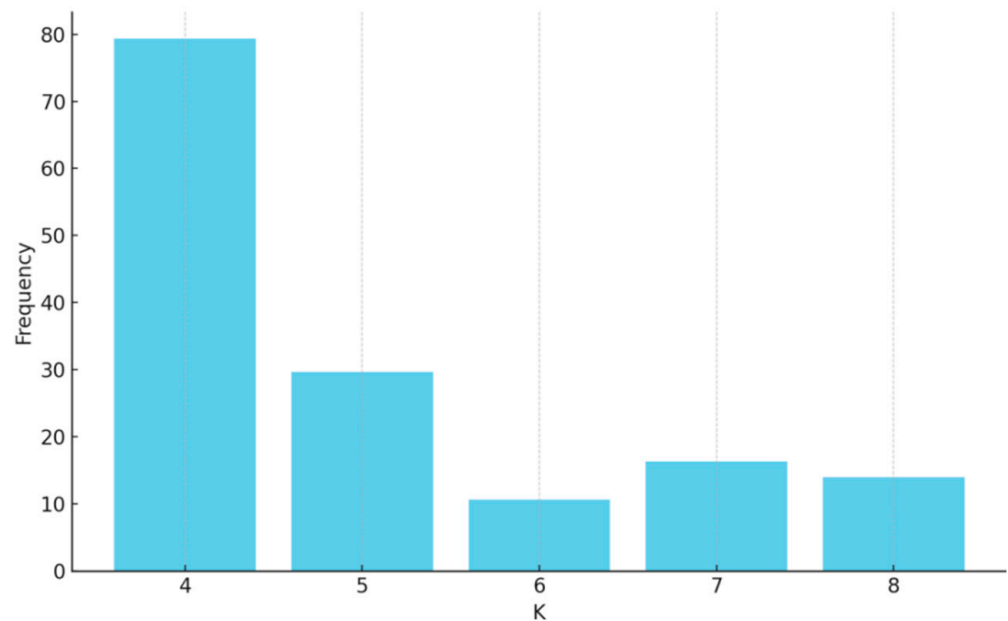


Figure 4. Frequency distribution of the clusters obtained from feature channels. Clustering was performed on the output of the global pooling layer using WeDIV every 50 training iterations.

Table 4. Cross-validation accuracies of the grouped dropout adopting the WeDIV (under the combinations of weighting step and clustering number) or the K-means clustering.

Clustering Number (K)	WeDIV		K-Means
	Step: 0.25	Step: 0.2	
4	91.58%	92.12%	91.49%
5	91.96%	91.84%	91.24%
6	91.94%	92.10%	91.71%
7	91.94%	92.11%	91.12%
8	91.89%	91.97%	91.06%

3.3. Improvements of Gdnet-IP for Specific Insect Species

The insect recognition model Gdnet-IP designed in this study includes two major innovations: the introduction of the grouped dropout strategy as well as the incorporation of the PreLU activation function and BatchNorm module. Although the latter is not a strict methodological innovation, this attempt to alter the network structure significantly improved the insect classification accuracy (as shown by the accuracy improvement of optiCPAFnet-3 over optiCPAFnet-2 in Figure 2). We can observe in detail which specific insect's recognition accuracy was impacted by these two innovations. The confusion matrices for the insect recognition of the optiCPAFnet-2, optiCPAFnet-3, and Gdnet-IP models on independent Test Set 1 are shown in Figure 5. For the optiCPAFnet-2 model, the classification accuracy for *Parnara guttatus* Bremer and Grey (PGBG) and *Papilio xuthus* Linnaeus larvae (PXLL) were both below 50% (Figure 5A). Upon examining the original images, we found that these two insect classes only contained 40 and 39 images, respectively, posing a challenge for insect recognition. In cases in which the sample size for certain categories in the dataset is very small, few-shot meta-learning methods can be applied in future research [36]. For example, an inter-class knowledge transfer approach with few-shot meta-learning can be utilized to enhance the model's generalization ability on small samples. By extracting shared features and transferring knowledge across classes, this method enables the model to generalize effectively even with a limited number of insect samples [37,38]. Additionally, optiCPAFnet-2 misclassified a significant number of PXLL images as *Gongylopus adiposus* Brunner (GA), while the classification of adult *Papilio xuthus* Linnaeus (PXL) remained unaffected (98%). This may be because PXLL larvae resemble GA more closely and both have green backgrounds in the images, leading the model to fail to learn rich enough morphological features to distinguish them sufficiently.

With the introduction of the PreLU activation function and BatchNorm module, the optiCPAFnet-3 model improved the recognition accuracy of PGBG from 38% to 53%, although many PGBG instances were still misclassified as *Pieris rapae* (PR). The recognition accuracy for PXLL increased from 47% to 67%, indicating an enhancement in the model's nonlinear fitting capabilities (Figure 5B). With the further introduction of the clustering-based grouped dropout strategy, the Gdnet-IP model elevated the recognition accuracy of PGBG to 62% and significantly boosted the accuracy of PXLL to 93%, demonstrating that grouped dropout indeed enhanced the model's ability to extract insect morphological features (Figure 5C). We employed Grad-CAM visualizations to analyze the reasons behind the enhanced recognition accuracy for PXLL. In the Grad-CAM visualizations, we observe that in the OptiCPAFnet-2 model (Figure 6A), the highlighted areas are relatively dispersed, covering general parts of the insect's body without focusing on specific details. In the OptiCPAFnet-3 model (Figure 6B), the attention is slightly more focused, particularly on the white textures of the insect, but it is still not very precise. In contrast, the Gdnet-IP model shows significantly improved attention, with concentrated highlights on the white textures of the insect's body (Figure 6C). This suggests that the Gdnet-IP model successfully focuses on crucial insect features, especially the white textures, enhancing its ability to recognize insects even against complex backgrounds. It is evident that the Gdnet-IP enables the model to focus more on the textures present on insect bodies (Figure 6). This improvement enhances the model's generalization capability, allowing it to accurately identify target insects within similar background environments. This substantial improvement in recognition performance, especially in the context of such small sample sizes, challenges the conventional understanding that deep neural networks require large training datasets. Although the recognition accuracy of some individual insect species, such as Tettigoniidae (Tet) and Paratenosera seu Hierodula (PH), showed slight declines, most other insects (eight species) achieved a higher recognition accuracy. The effective recognition of the aforementioned two insects, in particular, contributed to the overall optimal classification performance of the Gdnet-IP.

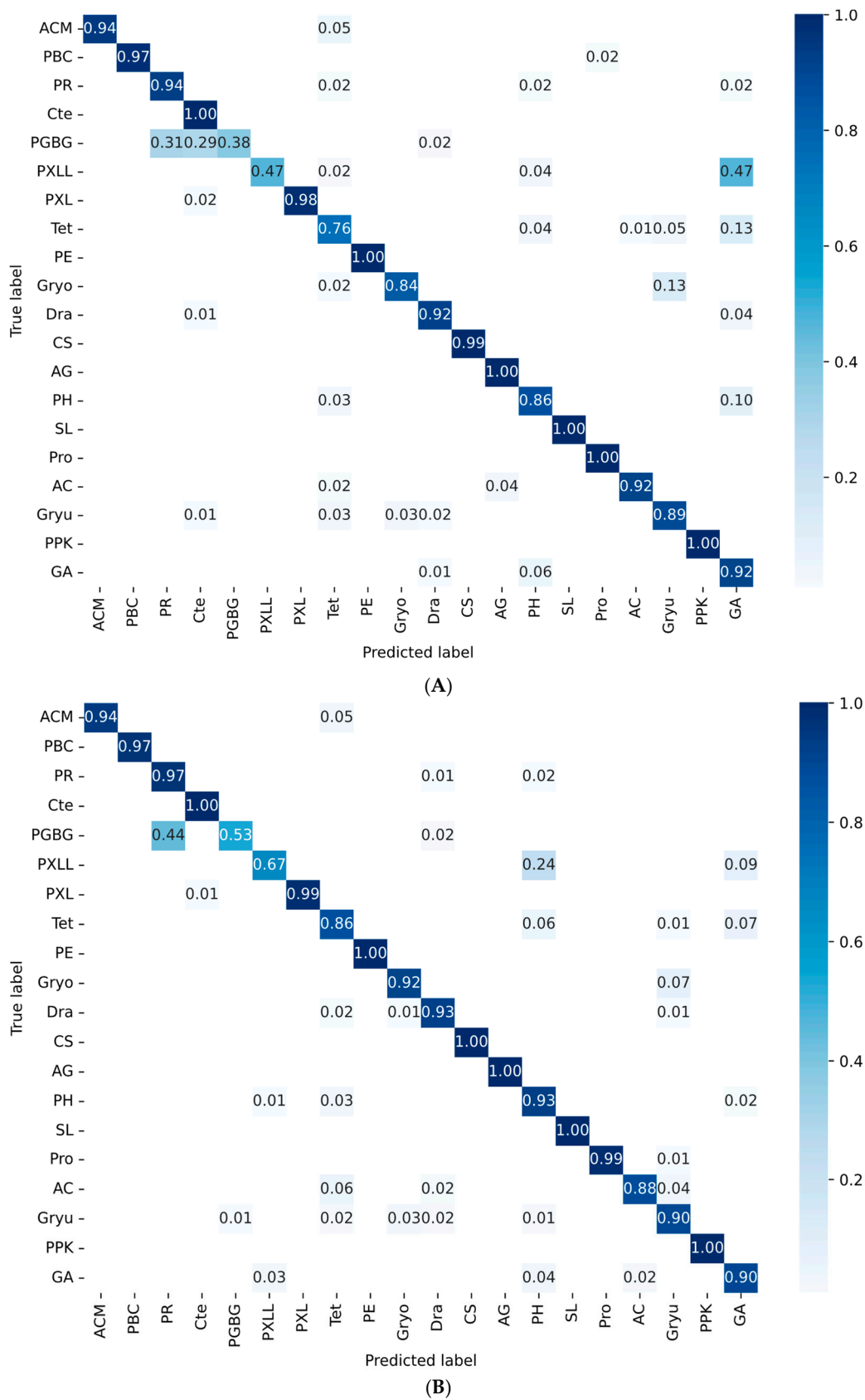


Figure 5. Cont.

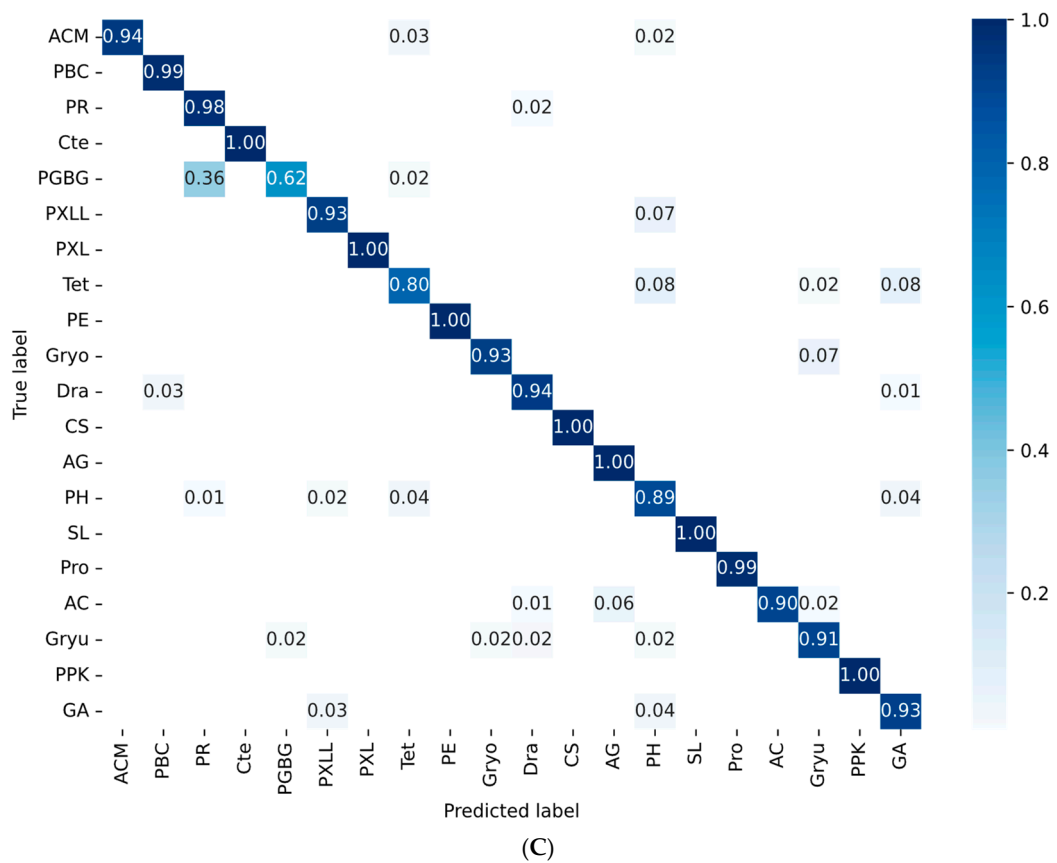


Figure 5. Confusion matrices for insect identification on independent Test Set 1 for different models. (A) Confusion matrix for the optiCPAFnet-2 model; (B) Confusion matrix for the optiCPAFnet-3 model; (C) Confusion matrix for the GDnet-IP model.

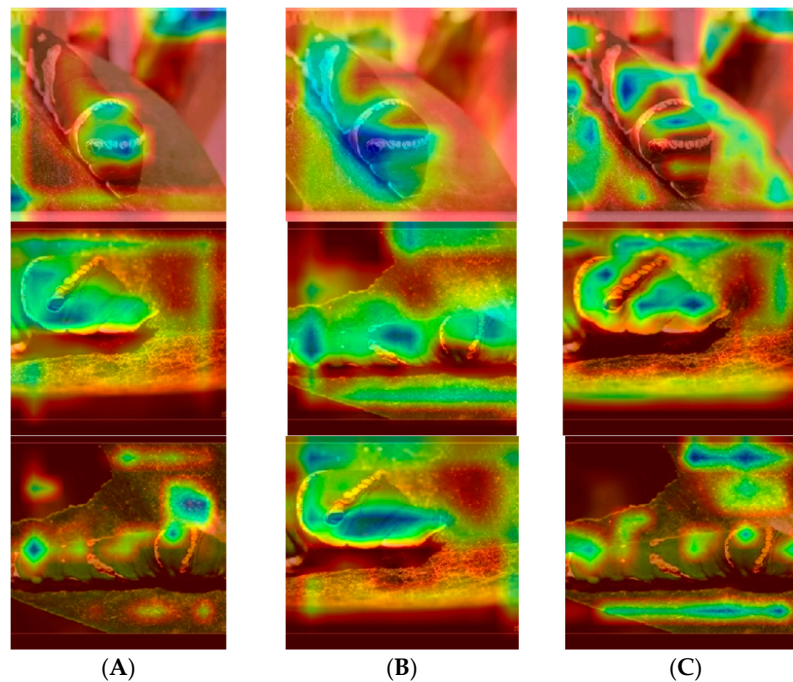


Figure 6. Grad-CAM heatmap of PXLL: (A) Grad-CAM heatmap of PXLL in the OptiCPAFnet-2 model, (B) Grad-CAM heatmap of PXLL in the OptiCPAFnet-3 model, (C) Grad-CAM heatmap of PXLL in the GDnet-IP model.

3.4. Comparison of Insect Recognition Models Adopting Different Dropout Methods

To validate the generalization capability of the new method, it is necessary to conduct independent testing. This study focuses on improving traditional dropout methods by comparing grouped dropout with three classic methods: dropout (as exemplified by optiCPAFnet-2), spatial dropout, and R-Drop. All reference dropout methods are based on the optiCPAFnet-3 model. First, we observed the classification performance of each method in cross-validation (Table S2, Figure 6). As previously noted, both grouped dropout strategies (i.e., the iGDnet-IP and Gdnet-IP) significantly outperformed the dropout method as well as the other two classic dropout methods, as shown in Figure 6. Among the models based on the three classic dropout methods, there were no significant differences in classification performance; however, they all significantly outperformed the optiCPAFnet-2 model. This indicates that regardless of the classic dropout method used, models incorporating PreLU and BatchNorm operations consistently exhibit superior insect recognition accuracies.

We then observed the independent prediction performance of each model (Tables S3 and S4, Figure 6). The CNN models based on the two grouped dropout strategies still significantly outperformed the other reference models. Compared to the optiCPAFnet-3 model, the iGDnet-IP model improved the insect classification accuracy from 91.60% (Test Set 1) and 91.91% (Test Set 2) to 92.28% ($p = 1.01 \times 10^{-5}$) and 92.47% ($p = 9.96 \times 10^{-4}$), respectively. The Gdnet-IP model further increased the accuracy to 92.72% ($p = 5.08 \times 10^{-9}$) and 92.88% ($p = 2.75 \times 10^{-7}$). The grouped dropout strategy not only enhanced model performance during hyper-parameter optimization but also demonstrated superior generalization capabilities in independent prediction stages. Additionally, although the Gdnet-IP model's classification performance was slightly lower than that of the iGDnet-IP model in internal cross-validation, it outperformed the latter in independent predictions, especially in Test Set 1, including data augmentation (Figure 7). This indicates that the grouped dropout based on the cluster analysis has a superior level of resistance to overfitting.

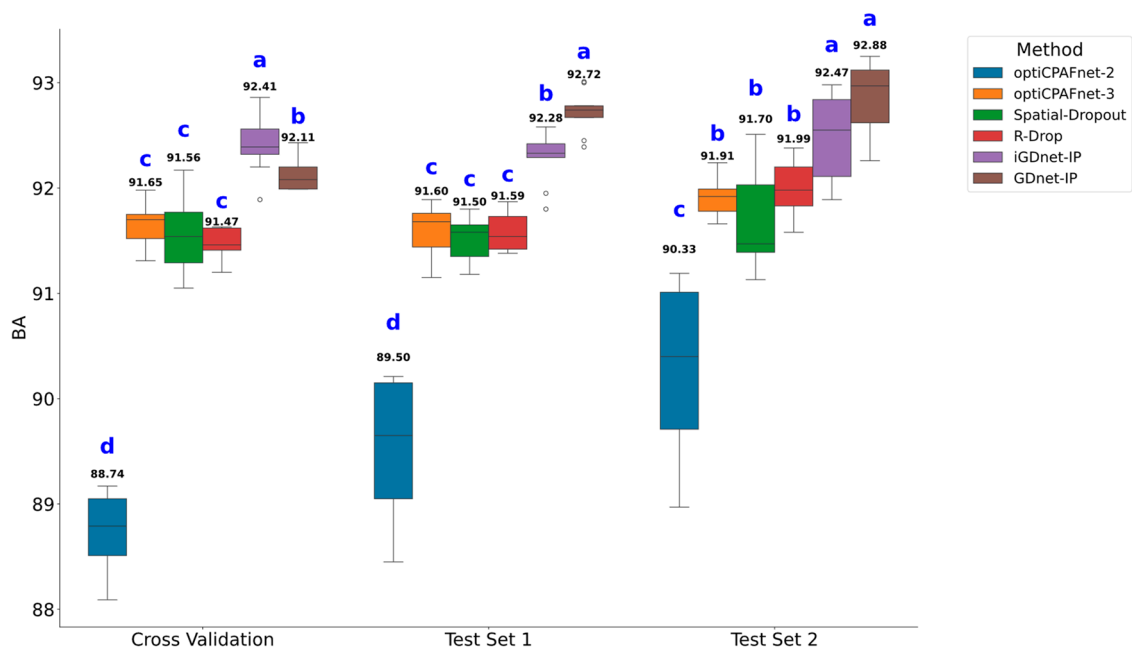


Figure 7. Comparison of insect classification models based on different dropout methods. Letters on the boxplot were derived from one-way ANOVA. Cross-validation accuracy was obtained from samples excluding test set images. Test Set 1 includes original and augmented images, while Test Set 2 consists of original images only. The two reference dropout methods were based on the optiCPAFnet-3 model. iGDnet-IP is a CNN model involving Inception module-based grouped dropout, and Gdnet-IP is a model adopting clustering-based grouped dropout.

3.5. Comparison of Insect Classification Accuracy Between Gdnet-IP and Classic CNN Models

All the comparisons of different dropout methods mentioned above are based on the CPAFnet model’s network structure. The hyper-parameters used in the reference models are shown in Table 5. To further validate the advantages of the new method, it is necessary to compare it with various classical CNN models. The cross-validation, independent test accuracy, and training time for each model on the CPAF dataset are shown in Tables S5–S7 and Figure 7. First, we compared the three-fold cross-validation accuracy of different models and found that the Gdnet-IP outperformed ResNet50 and significantly outperformed the other reference models. ResNet50 significantly outperformed the VGG11 model, while there were no significant differences among the other models (Figure 8A).

For independent Test Set 1, the Gdnet-IP also exhibited the best classification performance, significantly outperforming optiCPAFnet-2 ($p = 1.98 \times 10^{-3}$) and InceptionV3 ($p = 1.18 \times 10^{-5}$), and surpassing VGG11 (by 1.48%), VGG16 (by 1.87%), and ResNet50 (by 1.55%). InceptionV3 performed the worst, with no significant differences among the other four reference models (the leftmost bar group in Figure 7). For Test Set 2, the Gdnet-IP significantly outperformed optiCPAFnet-2 ($p = 2.43 \times 10^{-3}$) and InceptionV3 ($p = 2.45 \times 10^{-4}$), with improvements over the other three models as well. The two VGG models and ResNet50 showed no significant differences among themselves, but they all significantly outperformed optiCPAFnet-2 and InceptionV3 (right bar group in Figure 8B). In terms of training time, optiCPAFnet-2 and the Gdnet-IP, being lightweight models, trained significantly faster than the other classical models (Figure 8C).

Although the Gdnet-IP’s classification accuracy on independent test sets was not significantly superior to all of the reference models, its shorter training time facilitates deployment on mobile devices, enabling a broader range of applications. In addition, the four reference models feature deeper network structures (with their depths ranked as ResNet50 > InceptionV3 > VGG16 > VGG11). Except for InceptionV3, the other three models did not show significant differences in classification accuracy on the test set. This indicates that for datasets like CPAF, which have relatively fewer samples and categories than the ImageNet dataset containing ~22,000 categories, shallow network models can adequately handle the classification task. In contrast, deeper models are more prone to overfitting.

Table 5. Optimized hyper-parameters utilized in the reference models.

	VGG11	VGG16	ResNet50	InceptionV3
Batch size	256	256	256	64
Optimizer	SGD	SGD	SGD	RMSprop
Learning rate	0.0002	0.0002	0.0008	0.0002
Momentum	0.9	0.9	0.9	0.9

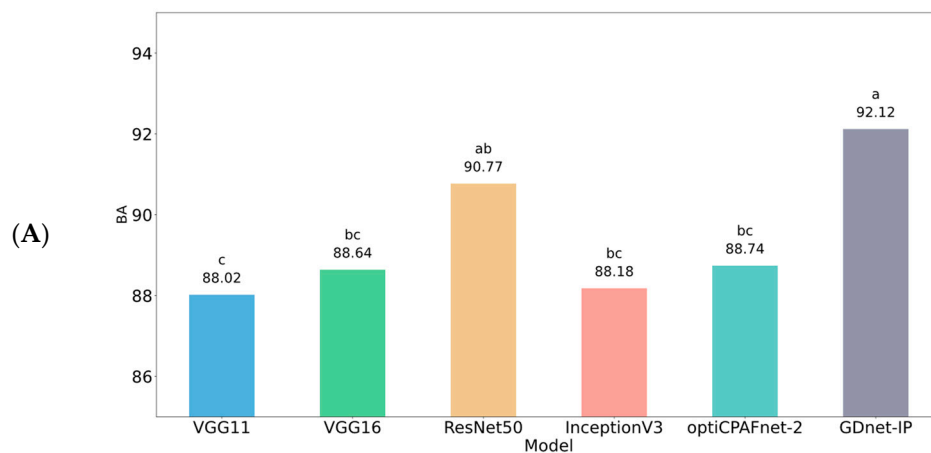


Figure 8. Cont.

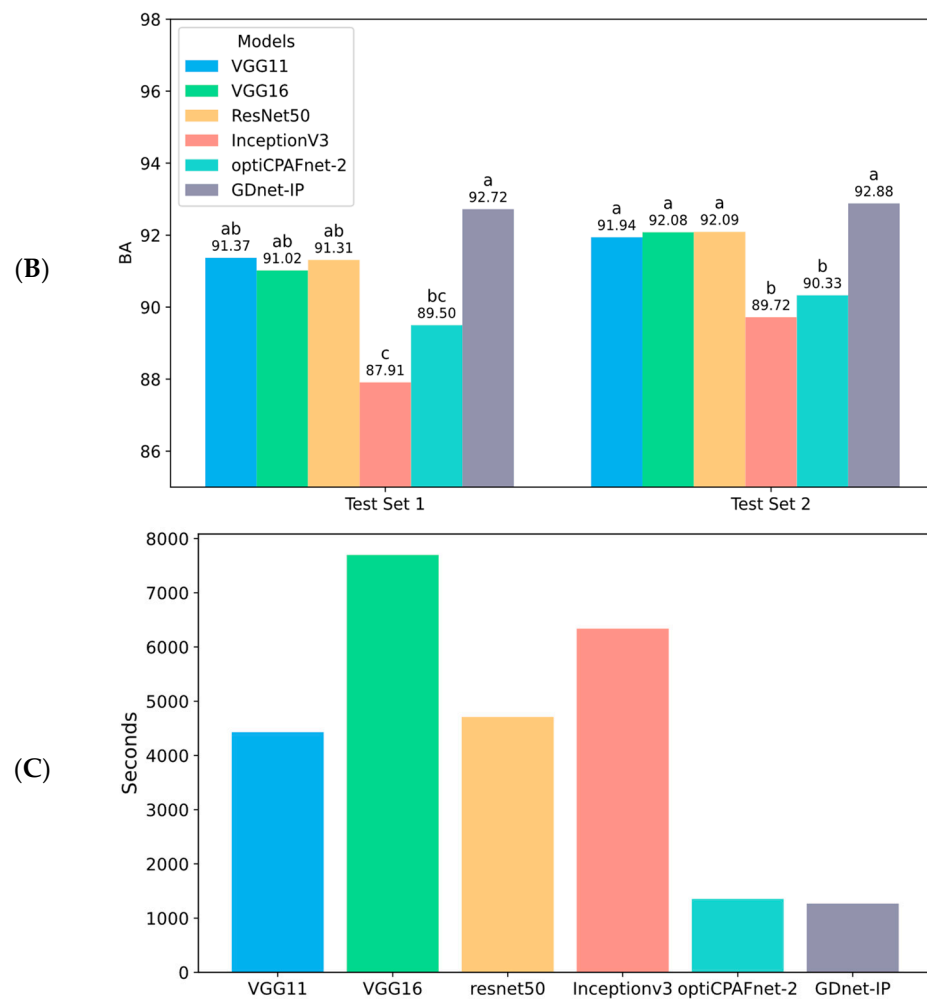


Figure 8. Comparison of insect classification accuracy between GDnet-IP and classic CNN models. The lowercase letters shown above the bars indicate the marks of statistical significance differences. (A) Cross-validation accuracy of each model; (B) Prediction performance of each model on Test Set 1 (with image augmentation) and Test Set 2 (without image augmentation); (C) Training time of each model (in seconds).

3.6. Explanation of Grouped Dropout Based on Model Gradients and Fisher Information

It is known from the previous results that CNN models based on grouped dropout can significantly improve insect recognition accuracy, but the reasons behind this remain unclear. The interpretability of deep learning models has always been a research hotspot in this field. We attempt to analyze the reasons for the effectiveness of the grouped dropout strategy from the perspectives of model gradients and Fisher information.

3.6.1. Explaining the Effectiveness of Grouped Dropout with Model Gradients

From the perspective of model gradients, the effectiveness of model's performance can be explained by the parameter update magnitude (the amount of change in the parameters at each iteration), gradient norm (the size of the gradient vector), and gradient variance (the variability in gradients across different batches) [21].

We first analyzed the difference in parameter update magnitude between the Gdnet-IP and optiCPAFnet-3 model. The L2 norm of the difference between the parameter values at iteration t (W_t) and at iteration $t + 1$ (W_{t+1}) was calculated as $\|W_{t+1} - W_t\|_2$, and this is used as the metric. We observed that as the training iterations increase, the L2 norm of the parameters for the Gdnet-IP rises faster than that for optiCPAFnet-3, with the magnitude of the difference becoming progressively larger (Figure 9A). In models trained with stochastic

gradient descent, a smaller parameter update magnitude can lead to the model converging to a local optimum or oscillating during convergence [19]. Therefore, the Gdnet-IP model may be more likely to achieve a global optimum within the same number of iterations. This phenomenon (i.e., the larger parameter update magnitude) may be attributed to two causes: first, the model with grouped dropout has larger gradient norms, resulting in larger “steps” (parameter updates); and second, grouped dropout may cause the model optimization to proceed in a more consistent direction. By observing the gradient norm and gradient variance, we found that compared to optiCPAFnet-3, the Gdnet-IP model initially (before 1000 iterations) has relatively smaller gradient norms and does not show an advantage. However, as the number of iterations increases, the difference in gradient norms between the two models becomes more pronounced (Figure 9B). Meanwhile, the Gdnet-IP shows greater gradient variance from the early stages of training and this continues into the later stages (Figure 9C). These results indicate that after dropping a particular class (or group) of associated neurons, the model requires larger gradient adjustments to adapt to this change, resulting in more diverse parameter updates.

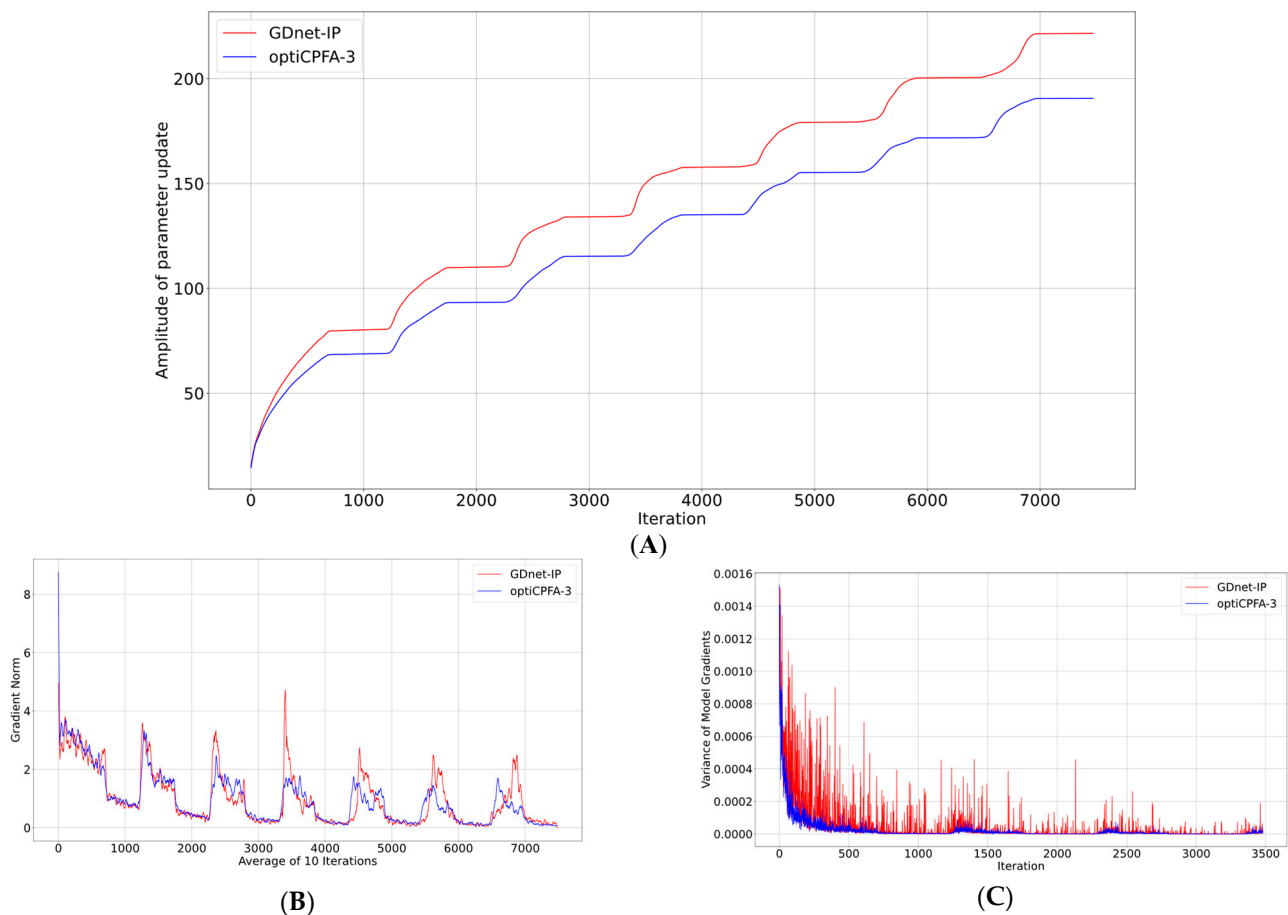


Figure 9. Parameter and gradient changes during the training iterations for two models. (A) Changes in the amplitude of model parameter update; (B) Changes in the model gradient norm calculated every 10 iterations; (C) Changes in the model gradient variance.

3.6.2. Explaining the Effectiveness of Grouped Dropout with Fisher Information

The process of neuron establishment in the early stages of training has a significant impact on the model’s final generalization ability [22]. In deep learning, Fisher information and the Fisher information matrix (FIM) are commonly used to analyze the effectiveness of neuron connections. Given parameters w and the prior distribution $Q(x)$ of the dataset, one can compute the posterior distribution $p_w(y|x)$ of the target variable and the gradient of its likelihood function. This allows for the calculation of the FIM value [22]. Fisher

information reflects the extent to which changes in parameters affect the data distribution (the smaller the FIM value, the smaller the impact), helping in understanding the model's sensitivity to parameter adjustments and the importance of these parameters.

We found that the FIM values for both models (the GDnet-IP and optiCPAFnet-3) exhibit two distinct phases of rising and falling in the early stages of training (the first 200 iterations) (Figure 10). During the rising phase, a large number of neuron connections are established in the training process, and the classification accuracy increases significantly. At this time, the FIM values of the GDnet-IP are significantly higher than those of optiCPAFnet-3, indicating that grouped dropout introduces more uncertainty in neuron weight updates, making it less likely for the model to get trapped in local minima. During the falling phase, the FIM value drops sharply from ~ 900 (the GDnet-IP model) to a higher scale (~ 400), but the model prediction accuracy does not decrease during this period; rather, it continues to rise slowly. This indicates that as the model performance stabilizes, the information content of the neuron weights gradually decreases. This is not a “forgetting” phenomenon, but rather an information “consolidation” process. The gradual decrease in information content suggests that the model is pruning unimportant connections. The larger decline in FIM for the GDnet-IP model indicates this process more strongly. This “forgetting”–“consolidation” process is highly similar to human brain development: in human infants, the number of synaptic connections per neuron peaks at around 12 months, and then gradually decreases with age, while cognitive abilities continue to increase [39]. This suggests that synaptic pruning or neuron “pruning” is essential for pattern learning and rule summarization.

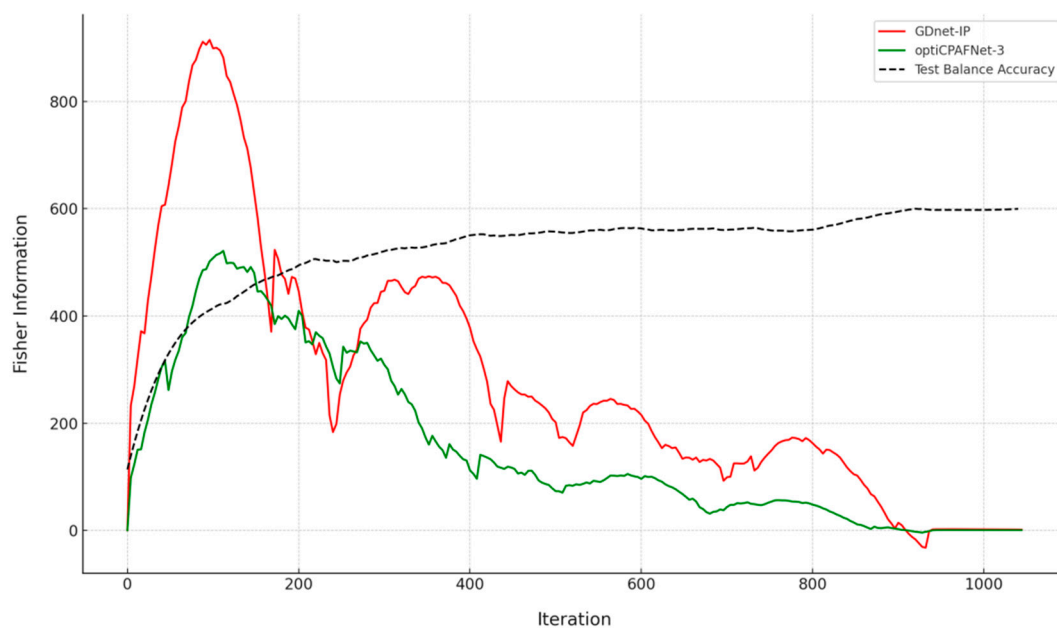


Figure 10. FIM scores and classification accuracy on Test Set 1 during the training iterations for two models.

By observing the trend in FIM changes after 200 iterations, we found that the FIM of the GDnet-IP exhibits a more pronounced and fluctuating decline, indicating that the model gradients still show an increasing trend during the rapid reduction process. This suggests that the corresponding neuron weight connections are undergoing multiple phases of pruning (declining periods) and re-establishment (rising periods). Additionally, there is often a critical learning period in the early stages of training, marked by the gradual decline in and stabilization of gradient variance [40]. The gradient variance of the GDnet-IP stabilizes around 3000 iterations (Figure 8C), while its FIM remains relatively high around 800 iterations (Figure 10), indicating a longer critical learning period for the GDnet-IP.

The continuous fluctuation of FIM values reflects the ongoing dropout of different neuron combinations, suggesting that the GDnet-IP can learn more new combinational features.

4. Conclusions

This study developed the GDnet-IP convolutional neural network model based on the novel concept of grouped dropout regularization for the image recognition of insects and common pests. Firstly, by optimizing the original CPAFnet model from multiple angles, including optimizer selection, dropout probability, learning rate, and decay strategy, we found that a smaller learning rate (0.0008) with decay constraints helps improve the model's classification performance (its internal cross-validation accuracy increased from 84.68% to 88.74%). Secondly, by replacing the default ReLU activation function with PReLU and adding BatchNorm layers after each Inception layer in the network, the model's nonlinear expression capability was enhanced (its classification accuracy improved to 91.65%), simultaneously increasing training efficiency. Thirdly, leveraging the inherent branching structure of the Inception module and the self-developed clustering algorithm WeDIV for adaptive grouping, two grouped dropout insect classification models (the iGDnet-IP and GDnet-IP) were proposed, which further significantly improved the insect recognition accuracy (with p -values of 3.47×10^{-6} and 3.37×10^{-5} , respectively).

Importantly, the GDnet-IP significantly improved the recognition accuracy of three insect species, *Parnara guttatus* Bremer and Grey (PGBG), *Papilio xuthus* Linnaeus larvae (PXL), and *Gryllotalpa* (Gryo), from 38%, 47%, and 84% to 62%, 93%, and 93%, respectively. Additionally, the classification performance of the two grouped dropout models was significantly better than that of the reference dropout methods. Compared to four classic CNN models, although the grouped dropout method did not achieve statistically significant superiority, its training time was significantly shorter than that of the reference models, facilitating model deployment on mobile devices. This paper also analyzed and discussed the effectiveness of grouped dropout from the theoretical perspectives of model gradient and Fisher information, providing a theoretical reference for establishing the interpretability framework of CNN models.

To enhance the recognition accuracy of each insect species, it would be beneficial to increase the number of images for insects that are difficult to classify. To improve modeling efficiency, transfer learning using pre-trained models based on large datasets such as ImageNet can be considered. The optimization of hyper-parameters in deep learning models still relies on experience and poses challenges; this can be approached as a multi-factor, multi-level experimental design problem. Advanced experimental design methods (e.g., uniform design) can be combined with classical machine learning models (e.g., support vector machines) to find the optimal combination of hyper-parameters. Finally, integrating object detection with the improved insect classification model from this study can help enhance the model's generalizability and expand its range of applications.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/agriculture14111915/s1>, Table S1. The names and sample sizes of 20 insect species in the CPAF dataset. Table S2. Cross-validation accuracies of models using different dropout methods. Table S3. Classification accuracy on Test Set 1 for models using different dropout methods. Table S4. Classification accuracy on Test Set 2 for models using different dropout methods. Table S5. Cross-validation accuracies of different CNN models. Table S6. Classification accuracy on Test Set 1 for different CNN models. Table S7. Classification accuracy on Test Set 2 for different CNN models.

Author Contributions: D.L.: Visualization, methodology, software, validation, writing—original draft. Y.X.: Data curation, investigation. Z.Y.: Conceptualization. Z.D.: Supervision, conceptualization, methodology, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported by the National Natural Science Foundation of China (31701164), the Natural Science Foundation of Hunan Province, China (2018JJ3238), the Excellent Youth Scientific

Research Program of the Educational Department of Hunan Province, China (22B0208), and the Special Funds for Construction of Innovative Provinces in Hunan Province, China (2021NK1011).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data is contained within the article. The data presented in this study are available in [Common pests image recognition based on deep convolutional neural network].

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. IPPC Secretariat. *Scientific Review of the Impact of Climate Change on Plant Pests*; FAO on Behalf of the IPPC Secretariat: Rome, Italy, 2021.
2. Rani, L.; Thapa, K.; Kanojia, N.; Sharma, N.; Singh, S.; Grewal, A.S.; Srivastav, A.L.; Kaushal, J. An extensive review on the consequences of chemical pesticides on human health and environment. *J. Clean. Prod.* **2021**, *283*, 124657. [[CrossRef](#)]
3. Preti, M.; Verheggen, F.; Angeli, S. Insect pest monitoring with camera-equipped traps: Strengths and limitations. *J. Pest Sci.* **2021**, *94*, 203–217. [[CrossRef](#)]
4. Li, W.; Zheng, T.; Yang, Z.; Li, M.; Sun, C.; Yang, X. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. *Ecol. Inform.* **2021**, *66*, 101460. [[CrossRef](#)]
5. Lima, M.C.; de Almeida Leandro, M.E.; Valero, C.; Coronel, L.C.P.; Bazzo, C.O.G. Automatic detection and monitoring of insect pests—A review. *Agriculture* **2020**, *10*, 161. [[CrossRef](#)]
6. Kasinathan, T.; Singaraju, D.; Uyyala, S.R. Insect classification and detection in field crops using modern machine learning techniques. *Inf. Process. Agric.* **2021**, *8*, 446–457. [[CrossRef](#)]
7. Qiao, M.; Lim, J.; Ji, C.W.; Chung, B.-K.; Kim, H.-Y.; Uhm, K.-B.; Myung, C.S.; Cho, J.; Chon, T.-S. Density estimation of *Bemisia tabaci* (Hemiptera: Aleyrodidae) in a greenhouse using sticky traps in conjunction with an image processing system. *J. Asia-Pac. Entomol.* **2008**, *11*, 25–29. [[CrossRef](#)]
8. Xie, C.; Zhang, J.; Li, R.; Li, J.; Hong, P.; Xia, J.; Chen, P. Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning. *Comput. Electron. Agric.* **2015**, *119*, 123–132. [[CrossRef](#)]
9. Ebrahimi, M.A.; Khoshtaghaza, M.H.; Minaei, S.; Jamshidi, B. Vision-based pest detection based on SVM classification method. *Comput. Electron. Agric.* **2017**, *137*, 52–58. [[CrossRef](#)]
10. Xia, D.; Chen, P.; Wang, B.; Zhang, J.; Xie, C. Insect detection and classification based on an improved convolutional neural network. *Sensors* **2018**, *18*, 4169. [[CrossRef](#)]
11. Wang, J.; Li, Z.; Gao, G.; Wang, Y.; Zhao, C.; Bai, H.; Lv, Y.; Zhang, X.; Li, Q. BerryNet-Lite: A Lightweight Convolutional Neural Network for Strawberry Disease Identification. *Agriculture* **2024**, *14*, 665. [[CrossRef](#)]
12. Talukder MS, H.; Sulaiman, R.B.; Chowdhury, M.R.; Nipun, M.S.; Islam, T. PotatoPestNet: A CTInceptionV3-RS-based neural network for accurate identification of potato pests. *Smart Agric. Technol.* **2023**, *5*, 100297. [[CrossRef](#)]
13. Li, Y.; Wang, H.; Dang, L.M.; Sadeghi-Niaraki, A.; Moon, H. Crop pest recognition in natural scenes using convolutional neural networks. *Comput. Electron. Agric.* **2020**, *169*, 105174. [[CrossRef](#)]
14. Wang, J.; Li, Y.; Feng, H.; Ren, L.; Du, X.; Wu, J. Common pests image recognition based on deep convolutional neural network. *Comput. Electron. Agric.* **2020**, *179*, 105834. [[CrossRef](#)]
15. Nanni, L.; Manfè, A.; Maguolo, G.; Lumini, A.; Brahnam, S. High performing ensemble of convolutional neural networks for insect pest image detection. *Ecol. Inform.* **2022**, *67*, 101515. [[CrossRef](#)]
16. Santos, C.F.; Papa, J.P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Comput. Surv.* **2022**, *54*, 1–25. [[CrossRef](#)]
17. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
18. Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; Bregler, C. Efficient object localization using convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 648–656.
19. Liang, X.; Wu, L.; Li, J.; Wang, Y.; Meng, Q.; Qin, T.; Chen, W.; Zhang, M. R-drop: Regularized dropout for neural networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 10890–10905.
20. Ning, Z.; Chen, J.; Huang, J.; Sabo, U.J.; Yuan, Z.; Dai, Z. WeDIV—An improved k-means clustering algorithm with a weighted distance and a novel internal validation index. *Egypt. Inform. J.* **2022**, *23*, 133–144. [[CrossRef](#)]
21. Liu, Z.; Xu, Z.; Jin, J.; Shen, Z.; Darrell, T. Dropout reduces underfitting. In Proceedings of the International Conference on Machine Learning, Zhuhai, China, 17–20 February 2023; pp. 22233–22248.
22. Achille, A.; Rovere, M.; Soatto, S. Critical learning periods in deep networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
23. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
24. Shazeer, N.; Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 4596–4604.

25. Tieleman, T.; Hinton, G. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *17*.
26. Eckle, K.; Schmidt-Hieber, J. A comparison of deep networks with ReLU activation function and linear spline-type methods. *Neural Netw.* **2019**, *110*, 232–242. [[CrossRef](#)] [[PubMed](#)]
27. Zhang, Y.D.; Pan, C.; Sun, J.; Tang, C. Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU. *J. Comp. Sci.* **2018**, *28*, 1–10. [[CrossRef](#)]
28. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
29. Du, Y.; Yuan, C.; Li, B.; Zhao, L.; Li, Y.; Hu, W. Interaction-aware spatio-temporal pyramid attention networks for action classification. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 373–389.
30. Bau, D.; Zhou, B.; Khosla, A.; Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–27 July 2017; pp. 6541–6549.
31. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
33. Ashtiani SH, M.; Javanmardi, S.; Jahanbanifard, M.; Martynenko, A.; Verbeek, F.J. Detection of mulberry ripeness stages using deep learning models. *IEEE Access* **2021**, *9*, 100380–100394. [[CrossRef](#)]
34. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NE, USA, 27–30 June 2016; pp. 2818–2826.
35. Hospedales, T.; Antoniou, A.; Micaelli, P.; Storkey, A. Meta-learning in neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 5149–5169. [[CrossRef](#)] [[PubMed](#)]
36. Xu, Y.; Bao, Y.; Zhang, Y.; Li, H. Attribute-based structural damage identification by few-shot meta learning with inter-class knowledge transfer. *Struct. Health Monit.* **2021**, *20*, 1494–1517. [[CrossRef](#)]
37. Yang, Z.; Wang, J.; Zhu, Y. Few-shot classification with contrastive learning. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer Nature: Cham: Switzerland, 2022; pp. 293–309.
38. Ni, Y.; Guo, Y.; Jia, J.; Huang, L. On the Nonlinearity of Layer Normalization. *arXiv* **2024**, arXiv:2406.01255.
39. Kandel, E.R.; Schwartz, J.H.; Jessell, T.M.; Siegelbaum, S.; Hudspeth, A.J.; Mack, S. *Principles of Neural Science*; McGraw-Hill: New York, NY, USA, 2000.
40. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [[CrossRef](#)]

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.