Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 10, No. 3, 2024

Implementing Insect Classification Based on Convolutional Neural Networks and Tensorflow

Zhiqiang Zhang *

Zhejiang Dongfang Polytechnic, Wenzhou Zhejiang, 325000, China
*** Corresponding author:** Zhiqiang Zhang

Abstract: Insects are one of the most diverse biological groups on Earth, playing a crucial role in human agricultural production. However, there is a relative shortage of professionals capable of classifying insects. With the rapid advancement of computer vision technology, image classification techniques have been employed to classify insects. Traditional insect image classification requires manual extraction of image features, a process that is both time-consuming and labor-intensive, and the accuracy of the identification results is relatively low. To address these issues, this study adopts deep learning technology and uses Google's TensorFlow framework to build a convolutional neural network (CNN) model for insect classification. The article further analyzes the impact of different optimizers and learning rates on the model's classification performance. Experimental results show that using the Adam optimizer with a learning rate of 0.009 yields the highest recognition accuracy for the CNN model, reaching up to 92%.

Keywords: Deep Learning Techniques; Insect Image Classification; Convolutional Neural Network Architecture; Tensorflow Framework.

1. Introduction

As a major agricultural country, China frequently faces the challenge of pests and diseases in traditional agricultural production activities. The key to solving this problem lies in the accurate identification of insects. Traditionally, entomologists rely on their unique expertise and observation of insects' external features, comparing them with insect atlases to classify insects, a process that is both time-consuming and laborious. With technological advancements, this traditional classification method is gradually being replaced by image-based insect classification techniques. Currently, commonly used insect classification techniques include image recognition, microwave radar detection, biophotonic detection, sample detection, near-infrared and hyperspectral techniques, as well as acoustic detection methods. In recent years, with the rapid development of the field of artificial intelligence, deep learning technology has made significant progress in various domains such as natural language processing and machine vision. Therefore, researchers have begun to explore the possibility of applying deep learning technology to insect image classification. This study aims to utilize an image classification method based on deep learning to address the problem of insect classification, with the goal of providing new support and assistance for pest and disease identification in the real world.

2. Deep Learning Approach

Deep learning (DL), especially Convolutional Neural Networks (CNN), plays a crucial role in the field of Machine Learning (ML), particularly for tasks such as image classification. Through a carefully designed multi-layer structure, CNNs possess the ability to automatically identify and extract data features. By utilizing local connections, weight sharing, and pooling techniques, CNNs can effectively capture key features in images, thereby significantly simplifying the complexity of feature extraction. Meanwhile, Recursive Neural Networks (RNN) are more suitable for processing data with sequential features, such as natural language, by using recurrent connections to remember previous information. These deep learning techniques have achieved remarkable results in various fields such as Computer Vision (CV) and Natural Language Processing (NLP), becoming important drivers for the development of artificial intelligence. Their ability to automatically learn and analyze complex data lays a solid foundation for higher-level AI applications.

3. TensorFlow and Convolutional Neural Networks

(1) TensorFlow

TensorFlow is a powerful open-source machine learning framework developed and maintained by Google, boasting a wide range of functionalities. Developers can construct complex numerical computation models through data flow graphs, particularly excelling in deep learning and various machine learning applications. This library is not only compatible with multiple hardware platforms but also provides rich Application Programming Interfaces (APIs) and numerous visualization development tools, greatly simplifying the process of building and optimizing algorithm models. Therefore, it has become an ideal choice for developing artificial intelligence applications.

(2) Convolutional Neural Networks

Convolution is a unique mathematical operation that produces a third function through the combination of two functions, f and g. Its core essence is a type of integral transform. The process of this operation involves flipping and shifting one function, multiplying it by another function, and then integrating the product over the shift, thereby obtaining the result of the convolution. Convolution operations are widely used in various fields such as signal processing, image processing, and machine learning.

Image convolution is a mathematical process involving two

matrices: the feature image matrix and the convolution kernel matrix. In this process, the convolution kernel (also known as a filter) slides over the input feature image with a specific stride. At each position, each element of the convolution kernel is multiplied by the corresponding element in the feature image, and then these products are summed to obtain a single value. This process is repeated across the entire feature image, typically with a fixed stride (as shown in Figure 1, where the stride is 1) for sliding the convolution kernel, until the entire feature image is traversed, resulting in the final output feature map.

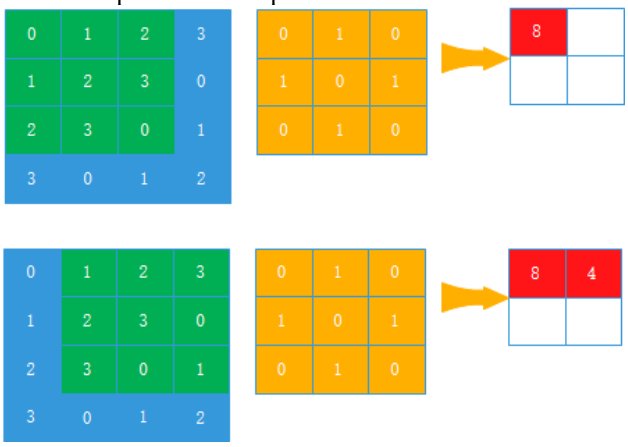


Figure 1. Convolution Operation Process

Pooling, also known as subsampling or downsampling, is based on the principle that neighboring regions in an image share highly similar feature. In the output of a convolutional layer, due to the high degree of overlap in features among adjacent image regions, there is a significant amount of redundant information. Pooling techniques aggregate global features of the image to reduce this redundancy, lower the computational complexity of convolutional neural networks, and prevent the neural network model from overfitting.

In image processing, two widely used pooling methods are average pooling and max pooling. Average pooling calculates the mean value of all pixel values within a specific region to represent the features of that region, while max pooling selects the maximum pixel value from the region as its representative. Both methods can effectively compress data while preserving key features in the image.

Taking Figure 2 as an example, it illustrates the computation process of image pooling, where the sliding stride is set to 2. This means that in the pooling operation, after processing one region, the system skips the adjacent region and proceeds to process the next one. This processing strategy not only improves computational efficiency but also reduces information redundancy to some extent, thereby helping convolutional neural networks more effectively extract and utilize image features. The pooling computation process can be explained through a concise diagram.

In Figure 2, there is a 4x4 feature map undergoing 2x2 Mean Pooling and Max Pooling. The pooling window slides over the feature map, selecting either the mean or the maximum value from within the window for the output. Ultimately, a 2x2 output feature map is obtained, achieving dimensionality reduction of the feature map. This process helps extract more prominent features while effectively reducing the number of model parameters and computational complexity.

Activation functions enable neural networks to possess powerful learning capabilities and effectively simulate complex nonlinear relationships by applying nonlinear transformations to each node in the network. Without activation functions, the functionality of neural networks would be severely diminished, limiting them to simple linear transformations. Undoubtedly, this would greatly restrict their expressive power and learning ability, rendering them unable to effectively capture the complex and diverse data relationships in the real world.

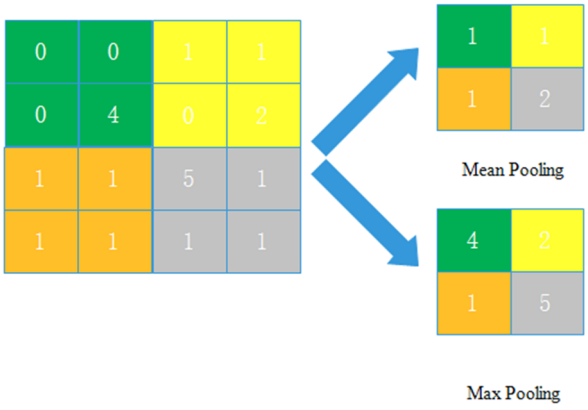


Figure 2. Pooling Operation Process

To meet the needs of different tasks, researchers have designed various activation functions, such as Sigmoid, Tanh, ReLU, Leaky ReLU, ELU, and Softmax. These activation functions each have unique characteristics, mathematical properties, and application scenarios, providing diverse options for neural network learning. For instance, the Sigmoid function compresses output values between 0 and 1, making it particularly suitable for the output layer of binary classification problems. The ReLU function is favored by many scholars due to its simple computation method and effectiveness in mitigating the vanishing gradient problem. However, it should be noted that it may suffer from the "dead zone" phenomenon when processing negative inputs. In contrast, the Softmax function is widely used in multi-class classification problems, effectively converting network outputs into probability distributions. In summary, activation functions play a crucial role in deep learning. They endow neural networks with the ability to learn and simulate complex nonlinear relationships, serving as a core technology driving the continuous evolution of deep learning techniques.

When constructing a neural network for multi-class classification problems, the softmax classifier is a common choice. It maps the input x to a k-dimensional probability distribution, where k is the total number of classes. For a given input x, the softmax function calculates the probability of each class j, using the following formula.

$$P(y = j|x) = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}}$$

Here, denotes the probability that the label y belongs to class j given the input x, typically computed as the dot product of the input x and the weights. In the formula above, the denominator represents normalization, ensuring that the sum of probabilities across all classes equals 1. To train this model, we define a loss function L(W) to measure the discrepancy between the predicted probability distribution and the true values. A commonly used loss function is the cross-entropy loss, which for the softmax classifier takes the following form.

$$L(W) = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} l\{y^{(i)} = j\}\log\frac{e^{W_i^T x^{(i)}}}{\sum_{l=1}^{k} e^{W_i^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=1}^{n} W_{ij}^2$$

Here, the indicator function takes the value of 1 when the variable equals j and 0 otherwise. The first component is the cross-entropy loss, which evaluates the deviation between the predicted probability distribution and the actual target values. Secondly, there is a regularization term aimed at mitigating overfitting of the model, which includes a regularization strength parameter indicating the degree of constraint on each element in the weight matrix W. During the training process of the model, an optimization algorithm (such as gradient descent) is employed to gradually adjust the weights W in order to minimize the value of the loss function. In this way, the model can more effectively fit the training data and make accurate classification predictions for new inputs.

4. Establishment of Insect Image Model

(1) Insect Image Data Collection

Prior to model training, we selected 12 different types of insects as our training dataset. Subsequently, utilizing advanced web scraping techniques, we collected 300 pieces of relevant data for each type of insect. After a rigorous manual review and screening process, inaccurate and unsuitable images were removed, leaving us with 215 high-quality training data entries. To construct a balanced and effective dataset, we split the data at a ratio of approximately 7:3, ensuring that each type of insect has 150 images in the training set and 65 images in the test set.

(2) Insect Image Data Preprocessing: One-Hot Encoding

One-hot encoding is a method of encoding where each state corresponds to an independent register bit, and at any given time, only one bit is active (i.e., has a value of 1) while the rest are 0. In the experiment, the image dataset is first converted into an array format and shuffled to enhance the model's generalization ability. Then, one-hot encoding is applied to the processed dataset to convert it into a form suitable for subsequent model training.

(3) Construction of Tensorflow Convolutional Neural Network

When constructing the convolutional neural network, the classic LeNet-5 model is chosen as the basic architecture, which includes two convolutional pooling layers and three fully connected layers. To mitigate the issue of model parameter overfitting, the dropout mechanism with a dropout rate of 0.5 is introduced into the network. The focus of this study is to explore how the application of different optimizers and the configuration of learning rates affect the final test performance of the model. During the comparative testing of optimizers, the experiment will refer to widely adopted parameter values and make adjustments and experiments within a small range of these reference values, with the aim of observing and analyzing changes in the model's prediction accuracy.

Adam, which stands for Adaptive Moment Estimation, is a variant of the gradient descent algorithm. Its characteristic lies in controlling the learning rate of the parameters within a certain range during each iteration, ensuring relative stability of the parameters even when faced with large gradients.

As shown in Figure 3, the Adam optimizer demonstrates excellent prediction accuracy. In the insect recognition task (as illustrated in Figure 4), despite the limited size of the training dataset, the Adam optimizer achieves a prediction accuracy of approximately 90% when the learning rate is set between 0.01 and 0.001, indicating good performance. However, when the learning rate exceeds 0.01, the prediction accuracy begins to gradually decline.

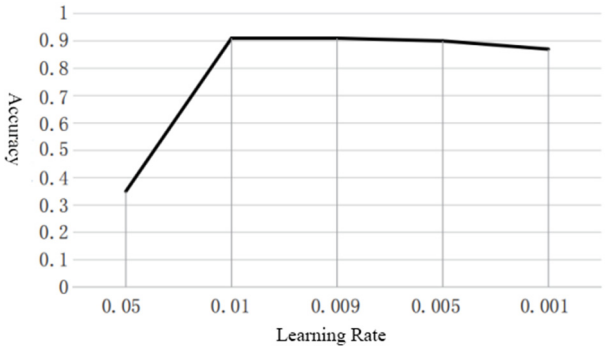
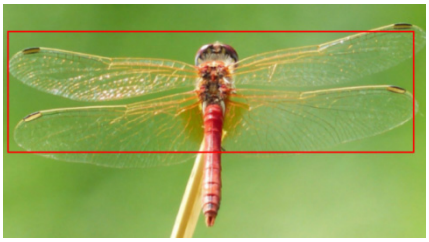
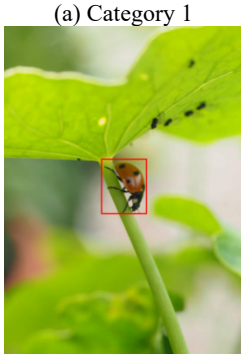


Figure 3. Prediction Accuracy of the Adam Optimizer



(a) Category 1



(b) Category 2

Figure 4. Illustration of Insect Recognition Results

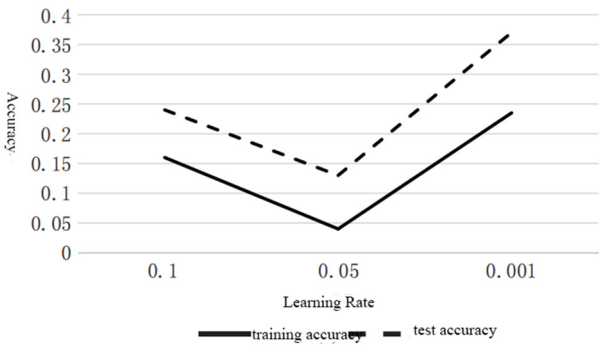
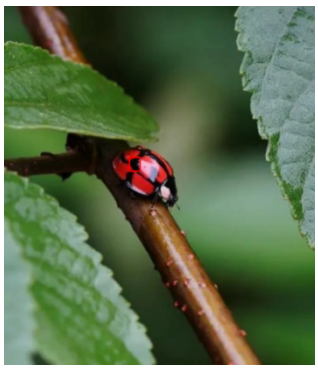


Figure 5. Training Results Using GD Optimizer

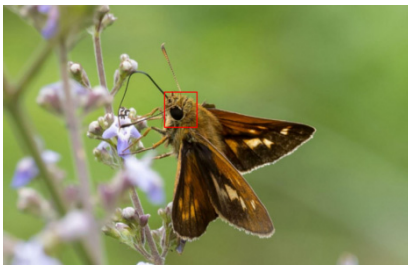
The standard Gradient Descent (GD) method is a commonly used approach in machine learning for solving model parameters. This method iteratively updates the parameters in the direction of gradient descent to gradually approach the minimum value of the loss function.



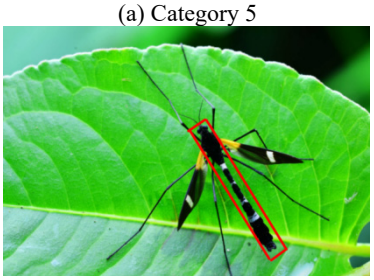
(a) Category 3

(b) Category 4

Figure 6. Schematic Diagram of Insect Recognition Effect



(a) Category 5



(b) Category 6

Figure 7. Illustration of Insect Recognition Results

Figure 5 displays the training results of the GD optimizer, while Figure 6 showcases its performance in the insect recognition task. However, in this experiment, the GD optimizer did not perform well. When the learning rate was set to 0.05, the prediction accuracy was very low, indicating that the model hardly learned effectively. Although attempts were made to significantly adjust the learning rate, the situation improved only slightly, and the prediction accuracy still failed to reach a satisfactory level. When the learning rate was set too low, there was no significant improvement in model performance. In summary, under the conditions of this experiment, the GD optimizer is not an ideal choice.

Adagrad is an optimization algorithm built upon Stochastic Gradient Descent (SGD), with its core idea being to adjust model parameters using different learning rates based on the frequency of data occurrence. Specifically, for frequently occurring datasets, a smaller learning rate is used for fine-tuning; whereas for rare datasets, a larger learning rate is employed to facilitate faster learning. This characteristic makes the Adagrad algorithm particularly excellent in handling sparse data scenarios.

In this experiment, the performance results of the Adagrad optimizer are listed in Table 1. Additionally, the effectiveness of insect classification is illustrated in Figure 7. When the initial learning rate of the Adagrad optimizer is set to 0.005, further reducing the learning rate leads to a significant decrease in prediction accuracy. Conversely, as the learning rate is gradually increased, the prediction accuracy shows an improving trend. Particularly when the learning rate is adjusted to 0.01, the test accuracy approaches 90%. Nevertheless, the overall accuracy remains stable at around 80%.

Table 1. presents the training results of the Adagrad optimizer.

Learning Rate	training accuracy	test accuracy	cost
0.05	1	0.8125	0
0.01	1	0.8950	0
0.009	1	0.8425	0
0.005	0.92308	0.7850	19063848
0.001	0.92308	0.6675	59120256

5. Conclusion

In this paper, a convolutional neural network model was constructed based on the TensorFlow framework for insect classification tasks. Additionally, the impact of three different optimizers and their learning rates on the model's final test performance was explored. Three mainstream optimizers—Adam optimizer, Gradient Descent (GD) optimizer, and Adagrad optimizer—were selected for comparative analysis, with different learning rates set for each optimizer.

Experimental validation revealed that the Adam optimizer exhibited the best performance when the learning rate was set to 0.009, achieving a prediction accuracy of up to 92%, which was significantly better than other configurations. However, due to the relatively small size of the dataset used in this experiment and the shallow architecture of the chosen convolutional neural network, there is still considerable room for improvement in prediction accuracy. This suggests that in future research, the accuracy of insect classification can be further enhanced by expanding the dataset size, deepening the network structure, or further optimizing model parameters.

Acknowledgments

Funded Project: 2024 Annual School-Level Project of Zhejiang Dongfang Polytechnic (Project Number: DF2024 SZZ01).

References

[1] Deng C, Han Y, Zhao B. High-Performance Visual Tracking With Extreme Learning Machine Framework[J]. IEEE Transactions on Cybernetics, 2019, 26(3): 1-12.

[2] Han H.G., Qiao J.F., Bo Y.C. Research on RBF Neural Network Structure Design Based on Information Strength[J]. Acta Automatica Sinica, 2012, 38(7): 1083-1090.

[3] Han H.G., Qiao J.F., Bo Y.C. On Structure Design for RBF Neural Network Based on Information Strength[J]. Acta Automatica Sinica, 2012, 38(7): 1083-1090.

[4] Zhao Z, Yang J, Che H, et al. Application of Artificial Bee Colony Algorithm to Select the Optimal Neural Network Architecture for Predicting Rolling Force in Hot Strip Rolling

Process[J]. Journal of Chemical and Pharmaceutical Research, 2013, 5(9): 563-570.

[5] Yan W, Tang D, Lin Y. A Data-Driven Soft Sensor Modeling Method Based on Deep Learning and Its Application[J]. IEEE Transactions on Industrial Electronics, 2017, 64(5): 4237-4245.

[6] Yao L, Ge Z. Deep Learning of Semi-Supervised Process Data with Hierarchical Extreme Learning Machine and Soft Sensor Application[J]. IEEE Transactions on Industrial Electronics, 2017, 65(2): 1490-1498.

[7] Vincent P, Larochelle H, Lajoie I, et al. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion[J]. Journal of Machine Learning Research, 2010, 11(Dec): 3371-3408.

[8] Zhang H, Zhang S, Yin Y. Online Sequential ELM Algorithm with Forgetting Factor for Real Applications[J]. Neurocomputing, 2017, 261: 144-152.

[9] Lekamalage C.K.L., Song K., Huang G., et al. Multi-Layer Multi-Objective Extreme Learning Machine[A]. In: Dong F. (ed.) 2017 IEEE International Conference on Image Processing [C]. Beijing: IEEE, 2017.

[10] Huang G.B., Zhu Q.Y., Siew C.K. Extreme Learning Machine: Theory and Applications[J]. Neurocomputing, 2006, 70(1-3): 489-501.

[11] He Q, Wang H, Jiang G.Q., et al. Research on Wind Turbine Main Bearing Condition Monitoring Based on Correlation PCA and ELM[J]. Acta Metrologica Sinica, 2018, 39(1): 89-93.

[12] He Q, Wang H, Jiang G.Q., et al. Wind Turbine Main Bearing Condition Monitoring Based on Correlation PCA and ELM[J]. Acta Metrologica Sinica, 2018, 39(1): 89-93.

[13] Toh K. Deterministic Neural Classification[J]. Neural Computation, 2008, 20(6): 1565-1595.

[14] Lu C.B., Mei Y. An Imputation Method for Missing Data Based on an Extreme Learning Machine Auto-Encoder[J]. IEEE Access, 2018, 6: 52930-52935.

[15] Gopakumar V, Tiwari S, Rahman I. A Deep Learning Based Data-Driven Soft Sensor for Bioprocesses[J]. Biochemical Engineering Journal, 2018, 136: 28-39.

[16] Su X, Zhang S, Yin Y, et al. Prediction of Hot Metal Silicon Content for Blast Furnace Based on Multi-Layer Online Sequential Extreme Learning Machine[A]. In: Chen X. (ed.) 37th Chinese Control Conference[C]. Wuhan: IEEE, 2018.