Automation and Machine Learning (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/autml.2025.060208
ISSN 2516-5003 Vol. 6 Num. 1

Design and Implementation of a Flower Image Classification System Based on Convolutional Neural Network (CNN)

Hongru Li[a], Zhitao Wu[b,*]

School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China
[a]lihongru415@outlook.com, [b]aswzt@163.com
**Corresponding author*

Keywords: Convolutional Neural Network; Image Classification; Flower Recognition; Data Augmentation; Model Optimization

Abstract: Flower image classification holds significant practical value in fields such as agricultural production, horticultural management, and ecological protection. Traditional flower classification methods based on manual feature extraction suffer from low efficiency and poor robustness, making them difficult to adapt to complex image scenarios. To address this issue, this paper designs and implements a flower image classification system based on Convolutional Neural Network (CNN). The system takes the public Oxford 102 Flowers dataset as the research object, and optimizes the quality of input data through preprocessing operations such as unified image size, normalization, and data augmentation. A lightweight CNN classification model is constructed with reference to the LeNet and simplified AlexNet structures, reducing the parameters of convolutional layers and fully connected layers to lower computational costs. Techniques including learning rate decay and batch normalization are adopted for model training and optimization. Comparative experiments are conducted to analyze the impacts of different optimizers (SGD vs. Adam) and data augmentation on the classification performance of the model. Experimental results show that the model with Adam optimizer and data augmentation achieves a classification accuracy of 92.3% on the test set, representing an increase of 15.6 percentage points compared with the unoptimized model. Characterized by simple structure, easy reproducibility, and excellent classification performance, this system provides a feasible reference for entry-level research on deep learning-based image classification.

1. Introduction

1.1. Research Background and Significance

As important plant resources in nature, flowers not only possess high ornamental value but also play a crucial role in pharmaceutical research and development, environmental monitoring, and agricultural economy. For instance, in the horticultural industry, rapid and accurate flower species identification serves as the foundation for seedling cultivation, quality grading, and market transactions. In ecological surveys, automatic identification of flower species enables efficient statistics of plant community distribution, providing data support for biodiversity conservation. However, there is a vast variety of flowers in nature, and different flower species exhibit complex similarities and variations in features such as flower shape, flower color, and leaf morphology. The traditional method of relying on manual visual identification is not only time-consuming and labor-intensive but also affected by factors such as the identifier's experience and subjective judgment, making it difficult to meet the requirements of large-scale and high-precision classification[4].

With the rapid development of computer vision and deep learning technologies, deep learning-based image classification methods have gradually replaced traditional methods and become the mainstream technology in the field of image recognition, thanks to their powerful automatic feature extraction capability. As the core model of deep learning applications in the image field, Convolutional Neural Network (CNN) can automatically extract abstract features from low-level to high-level from images through the alternating action of convolutional layers and pooling layers. It effectively captures the local spatial correlation of images, significantly improving the accuracy and robustness of image classification. Therefore, researching a flower image classification system based on CNN can not only solve the practical problem of flower identification in real applications but also provide a complete practical case of image classification tasks for deep learning beginners, which has important theoretical value and practical significance.

1.2. Research Status at Home and Abroad

In recent years, scholars at home and abroad have carried out extensive research on deep learning-based flower image classification. In foreign research, the AlexNet model proposed by Krizhevsky et al. achieved a breakthrough in the ImageNet image classification competition, providing a new technical idea for flower classification[1]. Subsequently, researchers improved classic models such as AlexNet and VGG and applied them to flower classification tasks. For example, a research team used the VGG16 model as the base network and applied the pre-trained model to the Oxford 102 Flowers dataset through transfer learning, achieving a classification accuracy of 94.1%. However, this model has a large parameter scale and high computational cost, which is not suitable for entry-level research and scenarios with limited resources.

In domestic research, scholars pay more attention to the lightweight and practicality of models. Some studies took LeNet as the basis and improved the feature extraction capability by increasing the number of convolutional layers, achieving a classification accuracy of over 85% on self-made flower datasets[2]. However, the model has poor adaptability to flower images with complex backgrounds and pose variations. Other studies introduced data augmentation technology to solve the problem of insufficient samples, increasing the accuracy of flower image classification by 8-12 percentage points, which verifies the important role of data preprocessing in classification tasks. Generally speaking, existing studies have proven the effectiveness of CNN in flower classification, but most high-performance models have problems such as complex structure, numerous parameters, and difficulty in reproduction. Therefore, research on lightweight and easy-to-implement flower classification systems for deep learning beginners still needs to be improved.

1.3. Main Research Content and Structure of This Paper

This paper focuses on the basic application of CNN in image classification, designs and implements a flower image classification system with simple structure and easy reproducibility, and focuses on exploring the impact of data preprocessing and basic optimization techniques on classification performance. The main research contents include:

(1) Selection and preprocessing of datasets, including unified size, normalization, and data augmentation operations for the Oxford 102 Flowers dataset;

(2) Construction of a lightweight CNN model, optimizing the number of network layers and parameter scale with reference to LeNet and simplified AlexNet structures;

(3) Design of model training and optimization strategies, introducing learning rate decay and batch normalization technologies to improve model performance;

(4) Design and analysis of comparative experiments to verify the impact of different optimizers and data augmentation on classification accuracy.

The subsequent structure of this paper is arranged as follows: Chapter 2 introduces the core principles of CNN and related basic technologies; Chapter 3 elaborates on the overall design of the flower image classification system, including dataset preprocessing, model structure design, and training optimization strategies; Chapter 4 verifies the system performance through experiments and analyzes the comparative experimental results; Chapter 5 summarizes the research results and points out the shortcomings of the system and future improvement directions.

2. Related Technical Foundations

2.1. Core Principles of Convolutional Neural Network (CNN)

CNN is a deep learning model specially designed for image data. Its core advantages lie in efficiently extracting image features and reducing model complexity through local connection, weight sharing, and pooling operations. A typical CNN model consists of an input layer, convolutional layer, pooling layer, fully connected layer, and output layer, with the functions of each layer as follows:

Input Layer: Receives preprocessed image data, usually input in the form of a three-dimensional tensor (width × height × number of channels). For example, RGB color images have 3 channels, while grayscale images have 1 channel.

Convolutional Layer: The core layer of CNN. It performs convolution operations on the input image through multiple learnable convolution kernels to extract local features of the image (such as edges, textures, color blocks, etc.). The essence of convolution operation is to slide the convolution kernel over the image and calculate the weighted sum of local regions to realize feature extraction. The weight sharing mechanism enables the same convolution kernel to use the same parameters on the entire image, significantly reducing the number of model parameters.

Pooling Layer: Usually follows the convolutional layer and is used for downsampling the feature maps extracted by the convolutional layer. It reduces the size and computational load of the feature maps while enhancing the model's robustness to image transformations such as translation and scaling. Common pooling operations include max pooling and average pooling. Among them, max pooling takes the maximum value of the local region as the output, which can effectively retain the key features of the image and is the most widely used.

Fully Connected Layer: Located at the back end of the network, it flattens the two-dimensional feature maps output by the pooling layer into a one-dimensional vector, and maps the features to the category space through full connection to realize the conversion from features to categories.

Output Layer: Uses the Softmax activation function to convert the output of the fully connected layer into a probability distribution of each classification category, and the category with the highest probability is the model's prediction result.

2.2. Key Basic Technologies

Data Augmentation: Data augmentation is a technology that generates new training samples by performing a series of random transformations on existing image data. Its core purpose is to expand the scale of the training dataset, increase sample diversity, and thereby alleviate the problem of model overfitting. In flower image classification tasks, common data augmentation operations include: random flipping (horizontal or vertical flipping), random cropping, brightness and contrast adjustment, random rotation, etc. These operations do not change the category attributes of flowers but allow the model to be exposed to flower images under different poses, lighting conditions, and perspectives, improving the generalization ability of the model.

Batch Normalization: During the training process of CNN, with the increase of network layers, the distribution of input data of each layer will change. This phenomenon is called "internal covariate shift", which will slow down the model training convergence speed and even cause problems such as gradient disappearance or explosion. Batch normalization technology stabilizes the distribution of input data of each layer by standardizing each batch of training data (converting the data into a distribution with a mean of 0 and a variance of 1), thereby accelerating model convergence, improving the training stability of the model, and to a certain extent suppressing overfitting[5].

Learning Rate Decay: The learning rate is a key hyperparameter that controls the update amplitude of model parameters. A too large learning rate may lead to unstable model training and failure to converge; a too small learning rate will make the model training convergence speed too slow and consume a lot of time. The learning rate decay strategy gradually reduces the learning rate during the training process. A larger learning rate is used in the early stage of training to speed up the convergence speed, and a smaller learning rate is used in the later stage of training to fine-tune the parameters, making it easier for the model to reach the global optimal solution. This paper adopts an exponential decay strategy, that is, the learning rate decreases exponentially with the increase of training epochs.

3. Design of Flower Image Classification System

The flower image classification system designed in this paper takes "data preprocessing→ model construction→model training and optimization→performance evaluation" as the core process. The overall architecture of the system is described as follows: The system first preprocesses the Oxford 102 Flowers dataset to generate standardized data that meets the model input requirements; it then constructs a lightweight CNN classification model and defines the structure and parameters of each layer of the network; the research team next sets the model training parameters and uses technologies such as learning rate decay and batch normalization for model training; researchers finally analyze the impact of different optimization strategies on model performance through comparative experiments to verify the effectiveness of the system.

3.1. Dataset Selection

This paper selects the internationally recognized Oxford 102 Flowers dataset as the experimental data. This dataset is constructed by the Computer Vision Laboratory of the University of Oxford, containing 8,189 images of 102 different types of flowers. Each type of flower includes 40 to 258 images. The dataset covers flower images under different varieties, poses, and lighting conditions, with high diversity and representativeness, which can effectively test the classification performance of the model. It is a standard dataset commonly used in flower image classification research. To meet the model training requirements, the dataset is randomly divided into a training set (5,732 images), a validation set (1,638 images), and a test set (819 images) in a ratio of 7:2:1. The training set is used for model parameter training, the validation set is used to adjust hyperparameters and monitor the overfitting of the model during the training process, and the test set is used to finally evaluate the classification performance of the model.

3.2. Construction of Lightweight CNN Model

To meet the requirement of "simple structure and easy reproducibility", this paper designs a lightweight CNN model with reference to the classic structure of LeNet and the core idea of AlexNet. By reducing the number of parameters in the convolutional layers and fully connected layers, the computational cost is reduced while ensuring the classification performance. The overall structure of the model is divided into two parts: the feature extraction module and the classification module. The feature extraction module is composed of 4 convolutional layers and 2 max-pooling layers alternately, which is responsible for extracting multi-level features from images. The classification module is composed of 2 fully connected layers and 1 output layer, which is responsible for mapping the extracted features to 102 flower categories.

3.3. Model Training and Optimization Strategies

The model in this paper is implemented based on the Python programming language, using the TensorFlow deep learning framework and the Keras high-level API for model construction and training. The experimental hardware environment is: Intel Core i7-10700K CPU, NVIDIA GeForce RTX 3060 graphics card (12GB video memory), 16GB memory; the software environment is: Windows 10 operating system, Python 3.8, TensorFlow 2.8.0, OpenCV 4.5.5 (for image preprocessing).

The core training parameter settings are as follows: 1) Loss function: The cross-entropy loss function (Categorical Cross-Entropy) is adopted, which is suitable for multi-classification tasks and can effectively measure the difference between the model's predicted probability distribution and the real label distribution; 2) Batch size: Set to 32, balancing training speed and memory usage, avoiding insufficient video memory caused by too large batch size or unstable training caused by too small batch size; 3) Training epochs: Set to 50 epochs. Through the performance monitoring of the validation set, if the accuracy of the validation set does not improve for 5 consecutive epochs, the early stopping strategy is adopted to terminate the training to prevent model overfitting; 4) Initial learning rate: Set to 0.001, using an exponential decay strategy with a decay coefficient of 0.95, that is, the learning rate after each training epoch = current learning rate $\times 0.95$.

To improve the model performance, the following optimization strategies are adopted in this paper:

(1) Application of Batch Normalization: Batch normalization layers are added after the first two convolutional layers to standardize the convolutional output features, stabilize the input distribution of each layer, and accelerate model convergence.

(2) Learning Rate Decay: Through exponential decay of the learning rate, parameters are updated quickly in the early stage of training, and parameters are fine-tuned in the later stage of training, making it easier for the model to reach the global optimal solution.

(3) Dropout Regularization: A Dropout layer is introduced in the fully connected layer to randomly discard some neurons, reduce the dependency between neurons, and suppress overfitting[6].

(4) Early Stopping Strategy: Monitor the accuracy of the validation set. This research stops the training and saves the current optimal model when the accuracy of the validation set does not improve for 5 consecutive epochs, so as to avoid performance degradation of the model due to overfitting in the later stage of training.

4. Experimental Results and Analysis

4.1. Statistics of Experimental Results

The statistical results of the performance indicators of the 4 groups of comparative experiments on the test set are shown in Table 1. The change curves of the training set and validation set accuracy during the model training process are described as follows: The accuracy of the training set and validation set of the model with Adam optimizer and data augmentation rises steadily, and the gap between them is the smallest, indicating good fitting ability and generalization ability.

Table 1 Experimental Results

Experimental Group	Optimizer	Data Augmentation	Test Set Accuracy (%)	Convergence Epochs	Total Training Time (min)
Experiment 1	SGD	No	76.7	42	89.5
Experiment 2	SGD	Yes	82.4	45	96.2
Experiment 3	Adam	No	88.1	28	60.3
Experiment 4	Adam	Yes	92.3	32	68.7

4.2. Analysis of Experimental Results

By comparing the results of Experiment 1 and Experiment 3, Experiment 2 and Experiment 4, it can be seen that under the same condition of whether data augmentation is adopted, the performance of the model using the Adam optimizer is significantly better than that using the SGD optimizer:

(1) Accuracy Improvement: Without data augmentation, the test set accuracy of the model with Adam optimizer (88.1%) is 11.4 percentage points higher than that of the SGD optimizer (76.7%); with data augmentation, the accuracy of the model with Adam optimizer (92.3%) is 9.9 percentage points higher than that of the SGD optimizer (82.4%). This is because the Adam optimizer combines the advantages of momentum gradient descent and adaptive learning rate, which can dynamically adjust the learning rate for different parameters, avoiding the problems of fixed learning rate and easy falling into local optimal solutions of the SGD optimizer, thereby updating the model parameters more efficiently and improving the classification accuracy[3].

(2) Training Efficiency Improvement: The convergence epochs and training time of the model with Adam optimizer are significantly less than those of the model with SGD optimizer. For example, without data augmentation, the convergence epochs of the Adam model are 28, and the training time is 60.3 minutes; while the convergence epochs of the SGD model are 42, and the training time is 89.5 minutes. This indicates that the Adam optimizer can accelerate the model training convergence, reduce the training cost, and is more suitable for practical application scenarios.

By comparing the results of Experiment 1 and Experiment 2, Experiment 3 and Experiment 4, it can be seen that under the condition of using the same optimizer, the performance of the model with data augmentation is significantly improved:

(1) Accuracy Improvement: Under the SGD optimizer, data augmentation increases the model accuracy from 76.7% to 82.4%, an increase of 5.7 percentage points; under the Adam optimizer, data augmentation increases the accuracy from 88.1% to 92.3%, an increase of 4.2 percentage points. This is because data augmentation expands the scale of the training set by generating diverse training samples, enabling the model to learn more flower features under different poses and lighting conditions, effectively alleviating the overfitting problem and improving the generalization

ability of the model.

(2) Change in Training Time: After introducing data augmentation, the model training time increases slightly (for example, from 60.3 minutes to 68.7 minutes under the Adam optimizer). This is because data augmentation needs to perform real-time transformation processing on images during the training process, increasing the computational overhead. However, the improved performance benefits are far greater than the increase in time cost, which is of practical significance.

5. Conclusion and Outlook

To summarize, this paper designs and implements a CNN-based flower image classification system using the Oxford 102 Flowers dataset, and key conclusions are drawn through comprehensive research: data preprocessing (including unified size and normalization) enhances model training stability, while data augmentation effectively alleviates overfitting, increasing classification accuracy by 4-6 percentage points; the Adam optimizer outperforms SGD in this task, improving accuracy by 9-12 percentage points and shortening training convergence time by over 30%; the designed lightweight CNN model, with a simple structure and moderate parameter scale (about 8.2M), achieves a test set accuracy of 92.3% on the dataset, outperforming most existing entry-level models and being easy to reproduce with low computational cost; additionally, the integrated application of optimization techniques such as batch normalization, learning rate decay, and Dropout plays a key role in accelerating model convergence, suppressing overfitting, and enhancing generalization ability, collectively ensuring the model's excellent performance and making it a valuable reference for beginners in deep learning-based image classification.

References

[1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.

[2] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 2002, 86(11): 2278-2324.

[3] Kingma D P. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[4] Goodfellow I, Bengio Y, Courville A, et al. Deep learning[M]. Cambridge: MIT press, 2016.

[5] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. pmlr, 2015: 448-456.

[6] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.