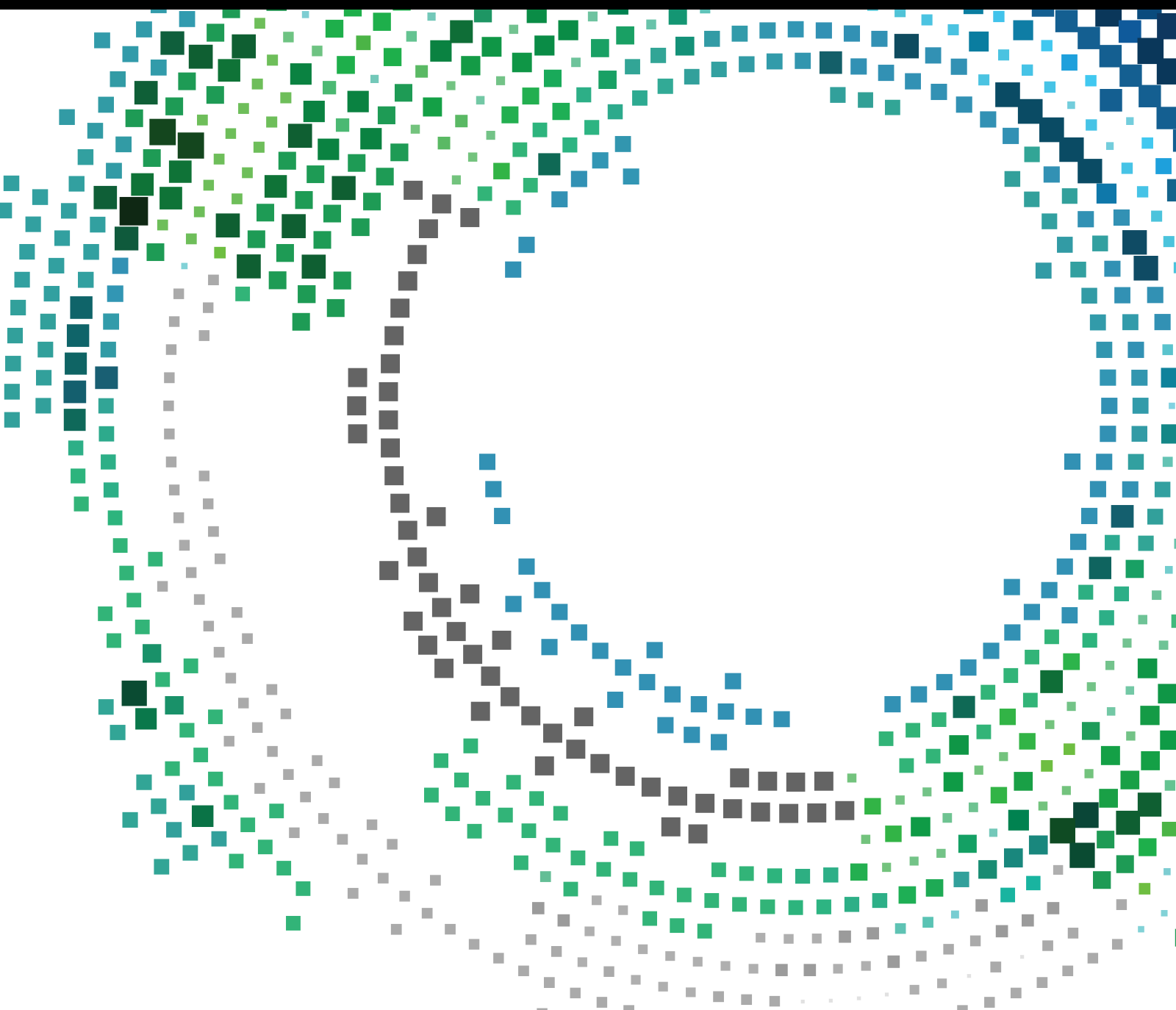


Deep Learning in Mobile Information Systems

Lead Guest Editor: Malik J. Khan

Guest Editors: Irfan Awan and Zeeshan A. Rana





Deep Learning in Mobile Information Systems

Mobile Information Systems

Deep Learning in Mobile Information Systems

Lead Guest Editor: Malik J. Khan

Guest Editors: Irfan Awan and Zeeshan A. Rana



Copyright © 2021 Hindawi Limited. All rights reserved.

This is a special issue published in "Mobile Information Systems." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Chief Editor

Alessandro Bazzi, Italy




Editorial Board

Ramon Aguero, Spain
Markos Anastassopoulos, United Kingdom
Marco Anisetti, Italy
Claudio Agostino Ardagna, Italy
Jose M. Barcelo-Ordinas, Spain
Luca Bedogni, Italy
Paolo Bellavista, Italy
Nicola Bicocchi, Italy
Peter Brida, Slovakia
Carlos Tavares Calafate, Spain
María Calderon, Spain
Juan-Carlos Cano, Spain
Salvatore Carta, Italy
Yuh-Shyan Chen, Taiwan
Wenchi Cheng, China
Massimo Condoluci, Sweden
Antonio de la Oliva, Spain
Almudena Díaz Zayas, Spain
Filippo Gandino, Italy
Jorge Garcia Duque, Spain
L. J. García Villalba, Spain
Michele Garetto, Italy
Romeo Giuliano, Italy
Prosanta Gope, United Kingdom
Francesco Gringoli, Italy
Carlos A. Gutierrez, Mexico
Wei Jia, China
Adrian Kliks, Poland
Quanzhong Li, China
Ding Li, USA
Jian-Xun Liu, China
Juraj Machaj, Slovakia
Mirco Marchetti, Italy
Sergio Mascetti, Italy
Elio Masciari, Italy
Eduardo Mena, Spain
Massimo Merro, Italy
Aniello Minutolo, Italy
Jose F. Monserrat, Spain
Raul Montoliu, Spain
Mario Muñoz-Organero, Spain
Francesco Palmieri, Italy
José J. Pazos-Arias, Spain
Marco Picone, Italy

Vicent Pla, Spain
Amon Rapp, Italy
Daniele Riboni, Italy
Michele Ruta, Italy
Neetesh Saxena, United Kingdom
Filippo Sciarrone, Italy
Florian Scioscia, Italy
Michael Vassilakopoulos, Greece
Ding Xu, China, China
Laurence T. Yang, Canada
Yugen Yi, China
Jianming Zhu, China


Contents

Deep Learning in Mobile Information Systems

Malik Jahan Khan , Irfan Awan , and Zeeshan Ali Rana 


Editorial (2 pages), Article ID 1296849, Volume 2021 (2021)

A Deep Learning Model for Quick and Accurate Rock Recognition with Smartphones

Guangpeng Fan, Feixiang Chen , Danyu Chen, Yan Li, and Yanqi Dong




Research Article (14 pages), Article ID 7462524, Volume 2020 (2020)

Research on Indoor Scene Classification Mechanism Based on Multiple Descriptors Fusion

Ping Ji, Danyang Qin , Pan Feng, Tingting Lan, and Guanyu Sun








Research Article (14 pages), Article ID 4835198, Volume 2020 (2020)

The Real-Time Mobile Application for Classifying of Endangered Parrot Species Using the CNN Models Based on Transfer Learning

Daegyul Choe , Eunjeong Choi , and Dong Keun Kim 







Research Article (13 pages), Article ID 1475164, Volume 2020 (2020)

Deep Learning on Computational-Resource-Limited Platforms: A Survey

Chunlei Chen , Peng Zhang , Huixiang Zhang , Jiangyan Dai , Yugen Yi , Huihui Zhang , and Yonghui Zhang 


Review Article (19 pages), Article ID 8454327, Volume 2020 (2020)

A Novel Image Classification Approach via Dense-MobileNet Models

Wei Wang , Yutao Li , Ting Zou , Xin Wang , Jieyu You , and Yanhong Luo 

Research Article (8 pages), Article ID 7602384, Volume 2020 (2020)

Hand Gesture Recognition Based on Single-Shot Multibox Detector Deep Learning

Peng Liu, Xiangxiang Li, Haiting Cui, Shanshan Li, and Yafei Yuan 

Research Article (7 pages), Article ID 3410348, Volume 2019 (2019)

Editorial

Deep Learning in Mobile Information Systems

Malik Jahan Khan ¹, Irfan Awan ², and Zeeshan Ali Rana ³

¹Department of Computer Science, Namal Institute, Mianwali, Pakistan

²Faculty of Engineering and Informatics, University of Bradford, Bradford, UK

³Department of Computer Science, NUCES-FAST, Lahore, Pakistan

Correspondence should be addressed to Malik Jahan Khan; malik.jahan@namal.edu.pk

Received 17 June 2020; Accepted 17 June 2020; Published 27 January 2021

Copyright © 2021 Malik Jahan Khan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The usage and dependence of mobile computing devices as well as mobile information systems have exponentially increased in the recent past. The adaptation of modern technologies in the developed world has completely redesigned the canvass of the daily life of a common person. Even in the developing world, it has penetrated and is still penetrating at a very fast pace. Such systems and applications now encompass a broad range of application domains in today's world including, but not limited to, healthcare, education, e-commerce, agriculture, forestry, weather, livestock, security, and social networking. Most of the applications address the problems of identification, recognition, diagnosis, predictions, reasoning, interpretation, and summarizations. Computer vision and machine learning have intersected decently to effectively cover these application as well as problem domains.

In the same era, machine learning has evolved as a panoramic science for almost all disciplines of computer science, with intelligence being embedded at the core of information systems. Theory and applications of supervised, unsupervised, and reinforcement learning have significantly been improved in the recent years. In most of the machine learning paradigms, theories of learning and cognitive sciences have contributed as base lines. Neural networks are one of such paradigms. The fundamental building block of a neural network is a neuron shown in Figure 1. One neuron takes inputs from its universe of discourse, aggregates the input while assigning a weight to each input, and passes the output through a transfer function and a final output is generated.

In conventional supervised learning, the generated output is compared with the target output provided by the trainer and an error is computed using the following equation:

$$E = o_j - t_j, \quad (1)$$

where o_j is the output of the neuron and t_j is the target output.

The error propagates back to compute the rate of change of error with respect to each weight using chain rule depicted in the following equation:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ij}}. \quad (2)$$

The step size of this differential is controlled through the learning rate and change in each weight is computed using the following equation:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \quad (3)$$

where Δw_{ij} is the change in the weight w_{ij} and η is the learning rate. Each weight is updated using the update rule given in equation (4) and the process is repeated with the expectation that the new weights would have gotten closer to the target weights.

$$w'_{ij} = w_{ij} + \Delta w_{ij}, \quad (4)$$

where w'_{ij} is the updated weight. It is applied across all edges of the neural network.

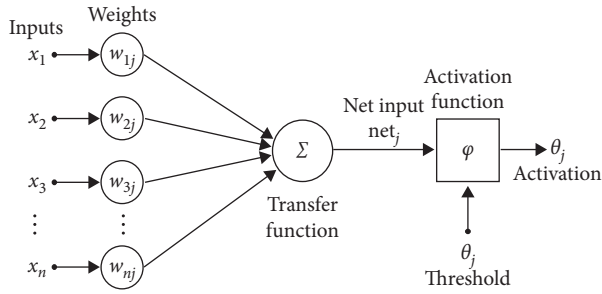


FIGURE 1: Building block of a neural network.

This back-propagation procedure is repeated across multiple iterations depending on the complexity of the target function. Different neurons are put together to generate a neural network to address the complex problems which may not be solvable by single neuron. Different neurons are arranged in different layers. Depth of the neural network is primarily determined by the number of hidden layers. The deeper neural networks are capable to learn the complex problem domains with the compromise on the computational complexity. Deep learning has successfully enhanced the effectiveness of mobile information systems in recent years to serve many different purposes, including object recognition, fault diagnosis, health monitoring, malware detection, and language translation. Over the last decade, mobile information systems have become more robust, autonomous, and self-organized, making tasks performed through these systems more reliable. Deep learning-based algorithms, models, and techniques, such as convolutional neural networks, probabilistic gradient algorithms, adaptive subgradient methods, and distributed deep learning over cloud methods, have been proposed, implemented, and deployed as the core decision-making engines in these systems.

The aim of this special issue of mobile information systems is to highlight the recent innovations where deep learning has been exploited to enhance the effectiveness of mobile applications and computing devices. This special issue includes papers that report innovative applications of deep learning for tasks such as visual analysis and classification in different fields including geology, indoor localization, animal species recognition, and human-computer interaction. This issue also includes a survey paper that summarizes typical applications of computational-resource-limited deep learning and presents a list of challenges to be addressed.

Fan et al. have presented a deep learning model for quick and accurate rock recognition with smart phones to help geological surveys. Recognition and classification of rock lithology is an important topic in geographical sciences. A lightweight convolutional neural network (CNN) has been trained to correctly recognize and classify rock images.

Ji et al. have proposed an indoor classification mechanism based on multiple descriptors fusion. Descriptor filter algorithms using a greedy approach have been proposed and implemented. Its performance has been analyzed and simulation results have been presented.

Choe et al. have presented a CNN to classify the endangered parrot species. The proposed approach has been deployed with a real-time mobile application. The application is quite innovative and significant to protect the endangered species of parrot. It has significant application to prevent smuggling and assist the relevant authorities identify breaches.

Liu et al. have presented a deep learning algorithm to recognize hand gestures. Hand gestures in complex scenarios have been chosen to be tested using single shot multibox detector deep learning algorithm. The proposed setup has been tested in real-time situations and high classification accuracy has been achieved. This adds significant value from the human-computer interaction perspective.

Wang et al. have presented a modification in MobileNet (a lightweight convolutional neural network) to improve its efficiency and capacity to integrate with mobile platforms. Image classification has been successfully tested in the implementation and analysis of the proposed framework.

Chen et al. have presented a survey on the deployment of deep learning in computational-resource-limited platforms. Deep learning is quite expensive in terms of resource requirements due to its inherent multilayer architecture and huge number of iterations required to execute the gradient-descent algorithm. The topic addressed in this survey paper is worth exploring to maintain a balance between the two extremes: computational expense and available resource-limited platforms.

With the intervention of sophisticated computational resources globally available through cloud computing, deep learning applications have been extensively developed and deployed. Conventional machine learning has been largely evolved into artificial neural networks and deep learning-based models primarily due to their potential to solve complex problems. With the advent of smarter computational resources for mobile devices, deep learning has immense potential to enable and embed intelligence in all kinds of applications and systems running over mobile devices.

Conflicts of Interest

The editors declare that there are no conflicts of interest regarding the publication of this special issue.

*Malik Jahan Khan
Irfan Awan
Zeeshan Ali Rana*

Research Article

A Deep Learning Model for Quick and Accurate Rock Recognition with Smartphones

Guangpeng Fan,^{1,2} Feixiang Chen ,^{1,2} Danyu Chen,^{1,2} Yan Li,^{1,2} and Yanqi Dong^{1,2}

¹School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

²Engineering Research Center for Forestry-oriented Intelligent Information Processing, National Forestry and Grassland Administration, Beijing 100083, China

Correspondence should be addressed to Feixiang Chen; bjfxchen@bjfu.edu.cn

Received 1 August 2019; Accepted 23 April 2020; Published 19 May 2020

Academic Editor: Zeeshan A. Rana

Copyright © 2020 Guangpeng Fan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the geological survey, the recognition and classification of rock lithology are an important content. The recognition method based on rock thin section leads to long recognition period and high recognition cost, and the recognition accuracy cannot be guaranteed. Moreover, the above method cannot provide an effective solution in the field. As a communication device with multiple sensors, smartphones are carried by most geological survey workers. In this paper, a smartphone application based on the convolutional neural network is developed. In this application, the phone's camera can be used to take photos of rocks. And the types and lithology of rocks can be quickly and accurately identified in a very short time. This paper proposed a method for quickly and accurately recognizing rock lithology in the field. Based on ShuffleNet, a lightweight convolutional neural network used in deep learning, combined with the transfer learning method, the recognition model of the rock image was established. The trained model was then deployed to the smartphone. A smartphone application for identifying rock lithology was designed and developed to verify its usability and accuracy. The research results showed that the accuracy of the recognition model in this paper was 97.65% on the verification data set of the PC. The accuracy of recognition on the test data set of the smartphone was 95.30%, among which the average recognition time of the single sheet was 786 milliseconds, the maximum value was 1,045 milliseconds, and the minimum value was 452 milliseconds. And the single-image accuracy above 96% accounted for 95% of the test data set. This paper presented a new solution for the rapid and accurate recognition of rock lithology in field geological surveys, which met the needs of geological survey personnel to quickly and accurately identify rock lithology in field operations.

1. Introduction

The recognition of rocks is not only an important part of geological survey but also the focus of geological research. The traditional recognition method consists of three steps [1, 2]: firstly, workers collect fresh rock samples in the process of exploration; secondly, after returning to the laboratory, the rock thin section with an area of about 2×2 cm is cut from the vertical stratification direction of the rock samples. When one side of the rock samples has been flattened on the grinding machine, it is glued to the carrier glass with glue such as adhesive. Then, the thickness of the other side is smoothed to 0.03 mm, and the cover glass piece is glued with the adhesive. Finally, an image of the rock sheet is viewed under a polarizing microscope by a knowledgeable

or experienced geologist. In this way, the rock type and structural parameters can be determined. This traditional identification method requires the observer to have very rich geological knowledge and experience. In addition, the method has many problems, such as strong subjectivity, long identification period, and poor field identification ability.

With the development of computer vision and image processing technology, great changes have been brought to rock recognition and mineral analysis [3, 4]. Many researchers analyzed the texture, fabric, granularity, and lithology of rocks based on image processing techniques such as image analysis and feature extraction. Patel used the probabilistic neural network (PNN) to develop a lab-scale vision-based model in which color histogram features are used as the input. The model has achieved good recognition

results, and the error of misclassification of limestone is less than 6%. The main limitation of this study is that the classification object is the entire rock sample of multiple rocks, and no further consideration is applied to identify rocks on site [5, 6]. Based on the basalt thin section image, Singh et al. extracted 27 characteristic parameters and identified 300 rock thin sections [7]. The recognition accuracy of the three texture categories is 92.22%, which is improved compared with previous studies, but the classification categories are fewer. Based on the research of image features, Cheng Guojian and Yin Juanjuan applied the support vector machine to realize the image classification of a total of 100 rock thin sections of 4 categories, with an accuracy of 80% [8]. The disadvantage is that the model performed poorly. With the continuous development of deep learning in the field of image intelligent recognition, many researchers used deep learning methods to automatically identify rock images [9–11]. Zhang Ye et al. used the transfer learning method for the first time to automatically identify and classify rock images. They have achieved the effective identification of three types of granite, Chiba, and breccia [12]. However, the experimental data are few and cannot meet the needs of on-site recognition. Li et al. used the transfer learning method to train the sandstone microscopic images to obtain a high-precision sandpaper slice microscopic image classification model [13]. The disadvantage is that the adaptability is poor, and it is only suitable for sandstone recognition. Cheng Guojian, Guo Wenhui and others realized automatic granularity recognition based on the rock thin section image [14]. The accuracy of rock identification is 98.5%. However, the identification objects of this study are rock thin section images, which need to be made in the laboratory and cannot be directly applied to the work site. Based on computer vision and machine learning, Marmo et al. used more than 1,000 carbonate flakes. Based on the gray scale digital image, they set up the multilayer sensory neural network model. Then, network training based on texture data was carried out, and the classification accuracy reached 93.3% [15]. Guo Chao et al. used the original color image of the rock to describe the feature space. Their method was to calculate the standard arithmetic values of different color channels by combining their morphologies [16]. The neural network is used to establish the mapping relationship between the feature space and the rock image category, and the algorithm is tested using 100 rock thin section images from the Ordos Basin. The results show that the automatic recognition rate of rock images in different color spaces is more than 95%.

The above research on rock image recognition uses the standard rock thin section image, rather than taking the more complex and direct rock image as the research object. It is based on various more complex feature parameter extraction algorithms, and the identified rock image data are less. The current research results reduce the problem of strong subjectivity and high recognition cost in the traditional method. But, it cannot meet the requirements of geological survey personnel to quickly identify rock lithology in real time in the field. “Smartphones” are now a ubiquitous handheld communication and computing device with

multiple sensors that all workers can use anytime, anywhere. In order to get a better solution, this paper proposes a method to identify the rock image. The method is suitable for smartphones, and the recognition is fast and accurate. An application running on an android smartphone is also designed and developed in this article. Because smartphone computing and storage resources are limited, this method is based on ShuffleNet, a light convolutional neural network. Combining with the transfer learning method, the learning results of ShuffleNet on ImageNet of a large data set are transferred. These are transferred to the experimental data set in this paper, namely, the rock image data set (a total of 30 categories). After retraining, the generated rock recognition model is exported. Finally, an app was designed and developed in this paper to help the staff quickly and accurately identify the rock lithology on site. The rock recognition model needs to be deployed on android-based smartphones. The model of this paper extracted features by searching image pixels without manual operation, which reduced the influence of subjective factors. Moreover, the training process has low requirements on the rock image size, imaging distance, and light intensity. Using smartphones, which are carried by workers, lithology can be quickly and accurately identified. Compared with the traditional method, it solves the problem of the traditional method. The solved problems include strong subjectivity, high identification cost, and long cycle. It also has advantages over the analysis and feature extraction techniques based on rock slice images. For example, the method can directly identify more complex images of rocks without making thin sections. This method has the advantage of quickly and accurately identifying rock lithology in the field, which can meet the requirements of workers to identify rocks quickly and accurately.

2. Materials and Methods

2.1. Rock Recognition Model Structure Design

2.1.1. ShuffleNet. ShuffleNet is an extremely efficient convolution structure designed for smartphones. It was proposed by Zhang Jian and others [17]. The depthwise separable convolution and group convolution introduced by Xception and ResNeXt can coordinate the ability and computation of the model, but their pointwise convolution occupies a large amount of computation [18–21]. ShuffleNet introduces pointwise group convolution to solve this problem. It has two characteristics: pointwise group convolution and channel shuffle. Compared with the existing advanced CNN models (such as MobileNets) [22, 23], the calculation amount can be greatly reduced under the similar precision, and the parameter amount can be greatly reduced. A large number of 1×1 convolutions consume a lot of computing resources, and pointwise group convolution helps reduce computational complexity. As shown in Figure 1, channel shuffle is to orderly disrupt the channels of each feature map to form a new feature map to solve the problem of “poor information flow” caused by group convolution.

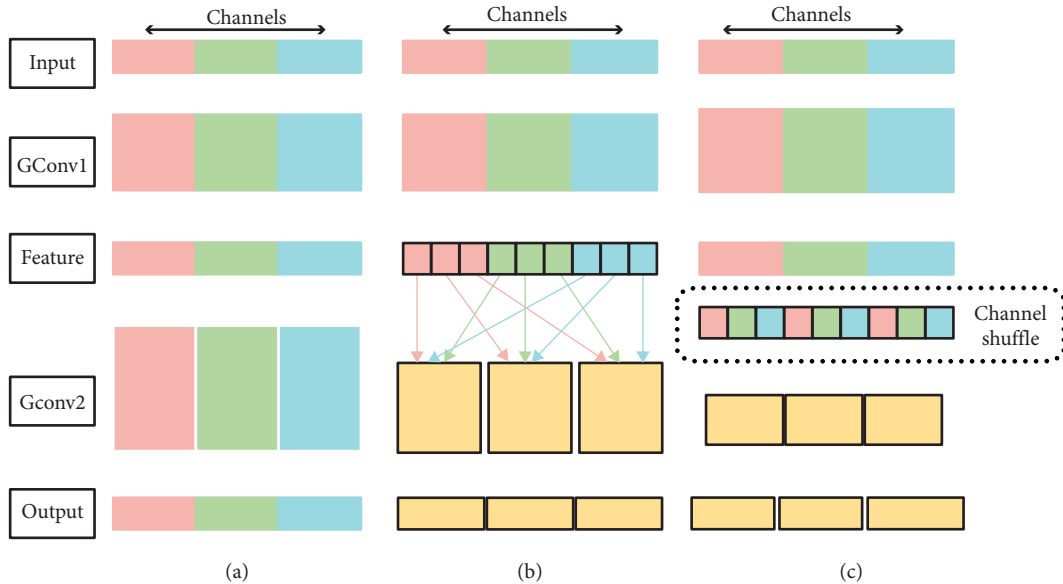


FIGURE 1: Channel shuffle with two stacked group convolutions. GConv stands for group convolution. (a) Two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; (b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; (c) an equivalent implementation to (b) using channel shuffle.

Group convolution can effectively reduce the computational cost, but the output only comes from some fixed input channels. It prevents feature exchange between channels and does not obtain the optimal representation. ShuffleNet uses channel shuffle to construct the association between the input channel and the output channel, including a convolutional layer with g groups and an output with $g \times n$ channels. The output dimension is reshaped into (g, n) , and it is transposed and flattened as the input to the next layer. Figure 2 shows that ShuffleNet is based on channel shuffle to construct the ShuffleNet unit.

The ShuffleNet architecture is primarily built from a set of ShuffleNet units. A ShuffleNet unit consists of a 1×1 pointwise group convolution layer and follows the channel shuffle operation layer. Under the same conditions, the calculation cost of this structure is low. The input is $c \times h \times w$ with bottleneck channels m . ShuffleNet only requires $hw(2cm/g + 9m)$ FLOPs, but ResNet requires $hw(2cm + 9m^2/g)$ floating-point operations per second (FLOPs). Compared to MobileNet, the ShuffleNet model achieves an absolute 7.8% performance in ImageNet Top-1 errors at a cost of approximately 40 million floating-point operations per second (MFLOPs). Channel split operation was proposed in the ShuffleNet V2. Firstly, the input of the feature channel is divided into two branch channels. One branch remains unchanged, and the other branch is computed a 1×1 convolution and 3×3 depthwise separable convolution. Then, the two branch features are connected, and the channel shuffle operation is implemented. After the channel is reorganized, the next unit is repeated. The report shows that ShuffleNet V2 is about 40% faster than ShuffleNet V1 and about 16% faster than MobileNet V2. With 500 MFLOPs, ShuffleNet V2 is 58% faster than MobileNet V2 and 63% faster than ShuffleNet V1 [24, 25].

2.1.2. Transfer Learning and Model Construction. The transfer learning method can apply the knowledge learned from other tasks (source tasks) to the target task. This method is conducive to the construction of the mathematical model of the target task and reduces the duplication of labor and the dependence on training data of the target task [26–28]. A comparison of transfer learning and traditional machine learning is shown in Figure 3. Traditional machine learning faces different learning tasks, and even if there is a similarity between tasks, different learning systems need to be established. However, in the face of different learning tasks, transfer learning can transfer the knowledge learned from the learning system. In other words, knowledge learned in solving the source task is transferred to the learning system that solves the target task.

From the perspective of the structure and function of the deep neural network, the convolutional layer of the neural network mainly extracts features and shares parameters and reduces the number of parameters through the use of the pooling layers [27–30]. The features extracted by the network are integrated through the final fully connected layer to obtain the high-level meaning of the image features. Then, it is classified by the classifier to get the final classification result [26, 31]. In some cases, the data set is small, and the distribution is not balanced, which makes the training results overfitting. The model performs well on the training data set but performs poorly on the verification data set and test data set. Using transfer learning method can improve this problem very well.

2.1.3. Model Structure Design. Through transfer learning, the model parameters trained by ShuffleNet on ImageNet of the large data set are migrated. In order to help train the rock image recognition model, it was migrated to 30 classes in the

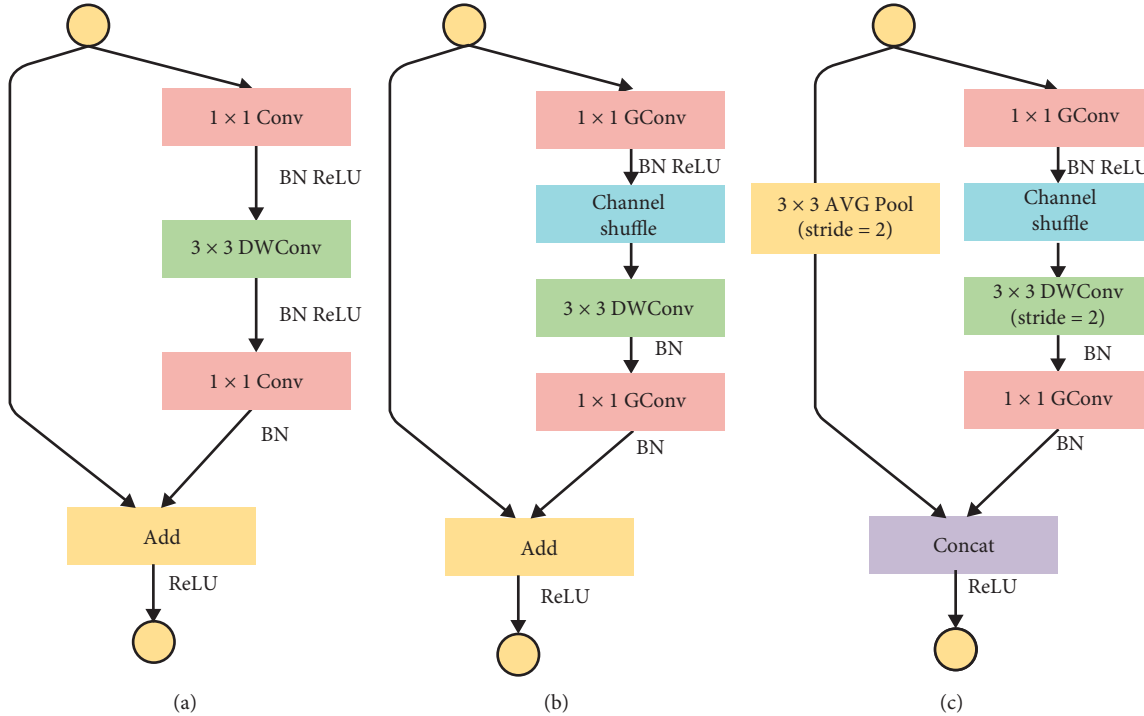


FIGURE 2: ShuffleNet units. (a) Bottleneck unit with depthwise convolution (DWConv); (b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; (c) ShuffleNet unit with stride = 2.

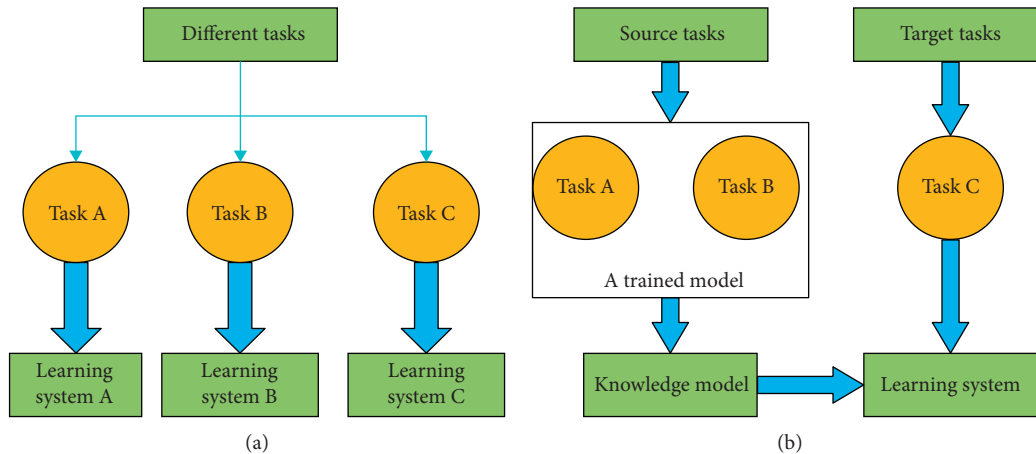


FIGURE 3: Comparison of traditional machine learning (a) and transfer learning (b).

experimental data set of the rock image. The ShuffleNet can extract valid information from images [32, 33]. The difference between rock image data sets and large data sets is relatively small. And in specialized fields, it is a small data set of fine-grained types. Therefore, the problem of rock image recognition belongs to the fine-grained classification of small data sets. This paper used the transfer learning method to perform rock image recognition. Most parameters of the network pretrained on large data sets were retained and adjusted to fit this data set. Input image resolution: 128, 160, 192, or 224px. Different sizes of input pictures will affect the classification results [34]. This article used 224 as the initial setting. The relative size of the model can be set to 1.0, 0.75,

0.50, or 0.25. This paper recommended 0.5 as the initial setting. Smaller models run significantly faster but at the expense of accuracy.

In order to deploy the trained model to the smartphone, the lightweight convolutional neural network structure ShuffleNet of 2.1.1 was used. ShuffleNet weights and parameters pretrained by ShuffleNet on the ImageNet data set were imported based on the characteristics of the rock image data set. Each convolutional layer used ReLU as the activation function. Batchnorm was used to normalize the distribution of the batch. The Softmax classifier of 2.1.4 was used for classification. A model was trained on the data set. The model structure is shown in Table 1.

TABLE 1: ShuffleNet architecture.

Layer	Output size	K size	Stride	Repeat	Output channels (g groups)				
					$g=1$	$g=2$	$g=3$	$g=4$	$g=5$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2	1					
Stage 2 ¹	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage 3			2	1	288	400	480	544	768
			1	7	288	400	480	544	768
Stage 4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity ²					143M	140M	137M	133M	137M

2.1.4. Softmax Regression Model. In the multiclassification problem, the Softmax regression algorithm was adopted in the rock recognition model in this paper to map the output values for multiple neural units into $(0, 1)$ with a total value of 1 [29, 35]. Therefore, the rock recognition model was classified as the probability of a sample being in a certain category to realize multiclassification. Let the training set consist of m labeled samples, i.e., $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$. The range of the category label y is $y^{(i)} \in \{1, 2, \dots, k\}$. Let probability $p(y = j|x)$ denote the probability that the sample is discriminated as being in category j in the case of input x . Therefore, the output of the k -class classifier is a k -dimensional vector, and the sum of its elements is 1. Analogical logistic regression using the hypothesis function $h_\theta(x^{(i)})$ can express the output of Softmax as

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}; \theta) \\ p(y^{(i)} = 2 | x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k | x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^k e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}. \quad (1)$$

Among the terms in the equation, the input x is a vector of dimension $m \times 1$, and the output model parameter is a matrix of order $m \times k$. The training process of the model is used to find the optimal value through continuous iteration so that the predicted value approaches the actual value. The cost function of the regression model can be expressed as

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{j=1}^k 1\{y^{(i)} = j\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right], \quad (2)$$

where $1\{\times\}$ is an indicative function, whose value rule is $1\{\text{expression whose value is true}\} = 1$; $1\{\text{expression whose value is false}\} = 0$.

As for solving the parameters by minimizing $J(\theta)$, there is currently no closed solution to minimize $J(\theta)$. In this paper, iterative algorithms such as gradient descent are used

to solve the problem. The gradient formula that was obtained after taking derivatives is as follows:

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^m [x^{(i)} (1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta))], \quad (3)$$

∇_{θ_j} is a vector whose l element $\partial J(\theta) / \partial \theta_{jt}$ is the partial derivative of $J(\theta)$ to the l component of θ_j .

In this paper, a weight attenuation function term is added to the cost function to make it strictly convex to ensure its convergence and unique solution. The cost function is modified by adding $\lambda/2 \sum_{i=1}^m \sum_{j=0}^k \theta_{ij}^2$, where n represents the number of input data, and this weight attenuation term will punish excessive parameter values. The cost function was converted to the following equation:

$$J(\theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{j=1}^k 1\{y^{(i)} = j\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^k e^{\theta_l^T x^{(i)}}} \right] + \frac{\lambda}{2} \sum_{i=1}^m \sum_{j=0}^k \theta_{ij}^2. \quad (4)$$

After adding the weight attenuation term ($\lambda > 0$), the cost function became a strict convex function so that a unique solution can be guaranteed, and since $J(\theta)$ is a convex function, the gradient descent method can guarantee convergence to the global optimal solution. Divide the data set into 3 parts: training set, validation set, and test set, take the number in λ from small to large, and then learn the model parameters on the training set, calculate the verification set error on the cross-validation set, and select the model with the smallest error, that is, choose λ . Finally, the evaluation is performed on the test set to obtain the best λ value. In order to use the optimization algorithm, the derivative of this new function $J(\theta)$ is required:

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^m [x^{(i)} (1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta))] + \lambda \theta. \quad (5)$$

An available Softmax regression model can be achieved by minimizing $J(\theta)$.

2.2. Model Training

2.2.1. Data Set and Data Preprocessing. Geological survey workers use fresh rocks by using a smartphone as an experiment in a data set of rock images. These rocks are generally rock foam rhyolite of different lithologies and dark gray stromatolite almond-like rough rocks. For example, there are 30 kinds of light gray rhyolite, purple red tuff, gray black obsidian, purple gray amphibolic rhyolite, and potassium gray white rock data. These images came from multiple locations in East China, with sizes between 3M and 6M. In this paper, the size of each image is compressed to 224 * 224 pixels on the condition of ensuring accuracy. Figure 4 is a sample map of the rock.

30 different kinds of rocks were collected, and a total of 3,795 images were taken. According to the ratio of 8:1:1, images were randomly selected from rock samples as the training data set, verification data set, and test data set. That is, there are 3,046 graphs of the training set, 381 graphs of the verification set, and 368 graphs of the test set. The detailed data distribution is shown in Figure 5.

It can be seen from the observation of the data set that the number of all types of data was unbalanced, and the number of pictures of some categories was very less. Methods such as rotation, flipping, cutting, and adjusting light and shade were used to randomly expand the training data set to improve the training performance.

2.2.2. Training Model. The transfer learning method was used to train the rock recognition model in the TensorFlow framework on the PC. The ShuffleNet network structure was built using the Python programming language, and the parameters pretrained by ShuffleNet on the ImageNet data set were imported. Experiments were evaluated on Core I9 series CPU, 32G RAM, NVIDIA GeForce GTX Titan Z 12G GPU, Linux OS PC. In the training process, the default iteration step number was 3600, and the learning rate was 0.008. The activation function was ReLU. During each iteration, 50 images were randomly selected from the data set for training, and 15 images were randomly selected for cross-validation. Softmax was used as a classifier to classify, and the optimization function used a method of stochastic gradient descent. Training accuracy refers to the percentage of accurate classification of currently trained images, while verification accuracy refers to the percentage of accurate classification of randomly selected images. Cross-entropy displays the learning effect of the model training process. The smaller the value, the better the learning effect.

As can be seen from Figure 6, the loss converges after the 160th iteration. After the 640th iteration, it remained stable, and the loss was close to zero. And at this point, the training accuracy is close to 100%. The accuracy of the verification set is slightly tight. After 3,600 iterations, the accuracy of the rock recognition model on the training set approaches 100%, and the loss is only 0.0004; the accuracy on the verification data set reached 97.65%, and the loss is only 0.1052. According to the training accuracy, verification accuracy, and cross-entropy changes, it can be seen that the training effect of the model is relatively ideal.

2.3. Accuracy and Run Time Comparison. This paper shows the superiority of the rock recognition model which is based on the combination of the ShuffleNet convolutional neural network and the transfer learning method. On the personal computer, the precision and running of the rock recognition model based on ShuffleNet were compared with the MobileNet model, the SqueezeNet model, and the standard convolutional network ResNet50 model. Originally designed for mobile and embedded visual applications, MobileNets are built primarily from the deep separable convolution operation, which decomposes standard convolution into deep convolution and point-by-point convolution. MobileNets apply a single filter to each input channel and then combine the output with linear combinations through point-by-point convolution. MobileNet V2 is proposed for further improvement. It is constructed by inversion residual and linear bottleneck technique, which can reduce the number of parameters and the loss of activation operation. Combined with the single-shot detector lite used for object detection, it was reported that MobileNet V2 was 35% faster than MobileNet V1, with 20 times less computation and 10 times fewer parameters than YOLO V2 [36]. SqueezeNet proposes to maintain precision with a small number of parameters, and its core structure is a new type of component Fire module. There are three main strategies for building Fire module [37, 38]. First, the 3 × 3 filter was replaced with a 1 × 1 filter. Second, the number of input channels of the filter was reduced. Finally, the network sampling was delayed. The evaluation results showed that the structural parameters of SqueezeNet were 50 times less than original AlexNet and maintained the horizontal accuracy of AlexNet on Imagenet.

2.4. Software Deployment. In this study, the trained rock recognition model was deployed on smartphones or embedded products. The significance of rock identification lies in the fact that geological investigators use smartphones for recognition on the work site instead of huge servers in the laboratory. Recognition in the laboratory is not a bad option. But, if workers take rock samples back to the laboratory to make rock thin sections, the recognition period and cost will increase. Many applications are often very sensitive to the response time of a program; even small latency in service response can have a significant impact on the user. Today, more and more applications provide core functionality through deep learning models. Whether people are deploying models to the cloud or to smartphones, low-latency reasoning is becoming increasingly important. One way to solve this problem is to perform model inference on a high-performance cloud server. Also, the input and output models are transferred between the client and the server. However, this solution brings many problems, such as high computing costs, massive data migration through mobile networks, user privacy, and increased latency. The top-level compressed ShuffleNet model takes an alternative approach to these scenarios and requires less resources to perform reasoning. This section describes the process of deploying the ShuffleNet model on a smartphone.

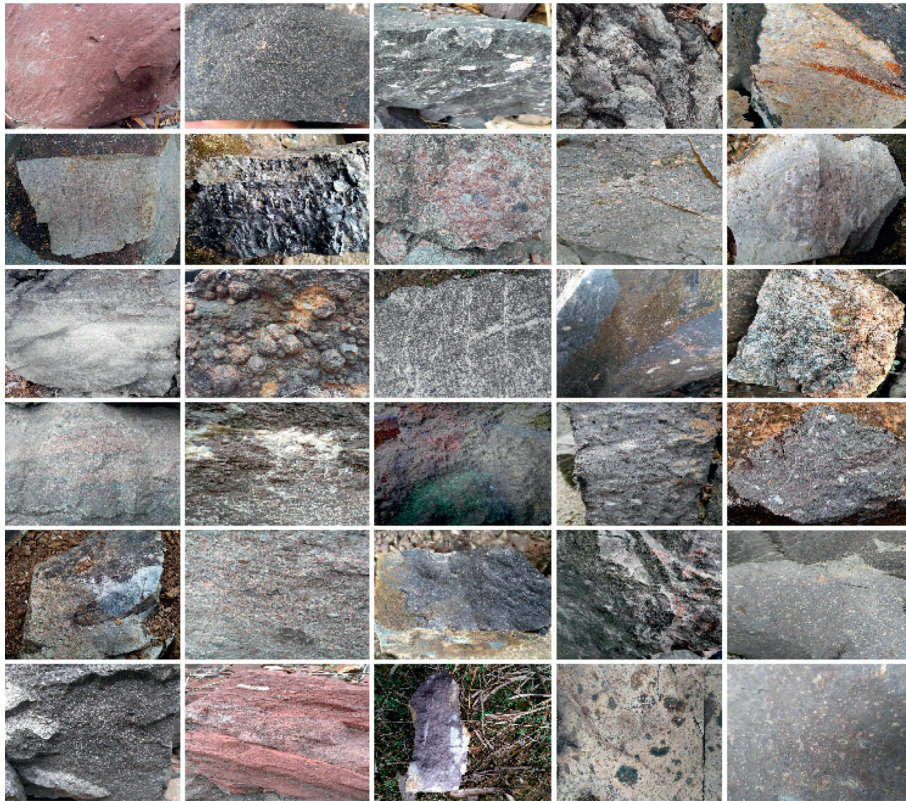


FIGURE 4: Rock sample data.

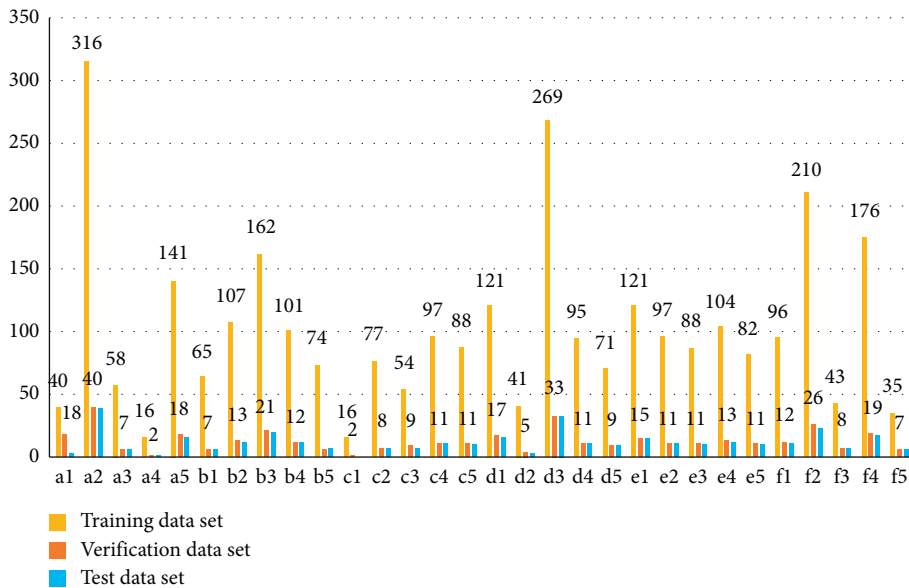


FIGURE 5: Data distribution.

The rock recognition model is trained on a PC server. This paper uses the TensorFlow framework to run properly on Linux. However, this cannot be done directly on the smartphone and requires some necessary conversions and deployments. The CNN model on Linux needs to be converted to the (.pb) format and deployed on Android smartphones. In order to implement the solution of

identifying rock lithology in the field, this paper developed an application running on Android smartphones. In addition, Huawei, Samsung, and Oppo, three common smartphones in the market, were selected as the experimental platform. The interfaces of the application are shown in Figure 7.

The application is written in the Java programming language and runs on an Android smartphone (Android

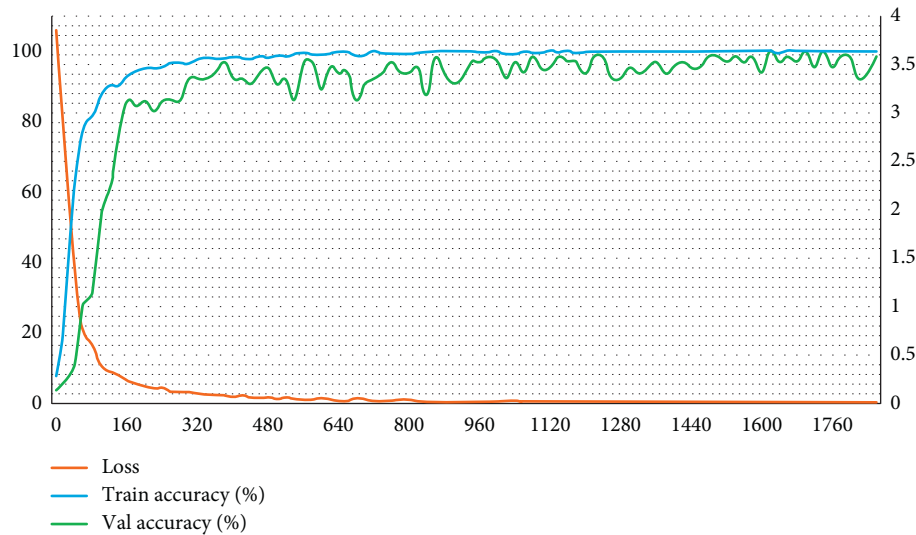


FIGURE 6: Training set accuracy, verification set accuracy, and loss during training.

4.4 operating system or higher). The operating memory of the phone should be greater than 4 GB, and the storage capacity should be 32 GB or more. The application can load and run the trained CNN model under the TensorFlow framework [39, 40]. It can identify not only the rock image captured by the smartphone in the field but also the rock photos stored in the gallery of the mobile phone. Software can output the recognition results (the type and nature of the rock, the accuracy of the recognition, and the time of execution) to the interface. The Huawei P20 mobile phone is used for the accuracy test of the model on the smartphone. The built-in main camera has a focal length of 3.95 mm and a resolution of 2244×1080 pixels, which is supported by most smartphones. Samsung Galaxy A8 s and Oppo R17 with a 16 megapixel rear camera (Android 7.0) are used to illustrate the performance of this app on other phones.

Operating the application is simple and convenient. Install the application on the smartphone carried by the investigator. Open the software at work. Then, click “Camera” to enter the camera photographing interface. Then, point the camera at the rock and click “photographing.” After this series of steps, the captured image is loaded into the interface to be recognized. Then, click “recognize,” and the recognition result (including the type and lithology of the rock, the recognition accuracy, and the recognition time) is displayed on the interface in less than 1 second.

2.4.1. Software Development Platform. On Windows 10, build the application development environment based on Android Studio 3.3, Android SDK (Java Development Kit), Java JDK 8 (Java Development Kit), TensorFlow Lite Development Kit (you can migrate the model trained under the TensorFlow framework to Android smartphone), and ADT (Android development tools). The application is suitable for the operating system of Android 6.0 and above.

2.4.2. Software Performance and Technical Overview. This software can be used by all types of Android smartphone users. It implements a friendly graphical user interface and features linear execution. In this interface, only the rock identification results (type, lithology, and accuracy) and the identification time are shown. It also allows the user to use the application without knowing its internal performance. The technical system of the application program is basically composed of two parts. One is the underlying part composed of the TensorFlow Lite development interface, and the other is the Android application layer composed of the Android native development interface (API) [41, 42]. TensorFlow Lite can deploy the CNN model trained under the TensorFlow framework to Android smartphones. According to the principle described in Section 2.1, TensorFlow Lite provides a Jar package written in Java and (.so) format dynamic library written in C++. The latter provides APIs for operational models such as functions for reading models, recognition functions, and output functions. The “Android native API” implements the main parts of the main application and the graphical user interface. It is responsible for coordinating tasks, invoking Android camera sensors to capture rock images and correctly store results. When rock images are captured in the “Android application layer” and needed to be identified and analyzed, a request was made for “TensorFlow Lite underlying part.” However, direct communication between the two parts is not feasible. So, this paper needs to use the Java Native Interface (JNI) to allow this interaction. The “Android application part” calls the required functionality through JNI, which is actually responsible for executing the C++ library and returning the results.

3. Results

3.1. The Accuracy and Time of the Rock Recognition Model Tested on the Smartphone. The purpose of this study is to facilitate geologists. The idea is for them to use smartphones

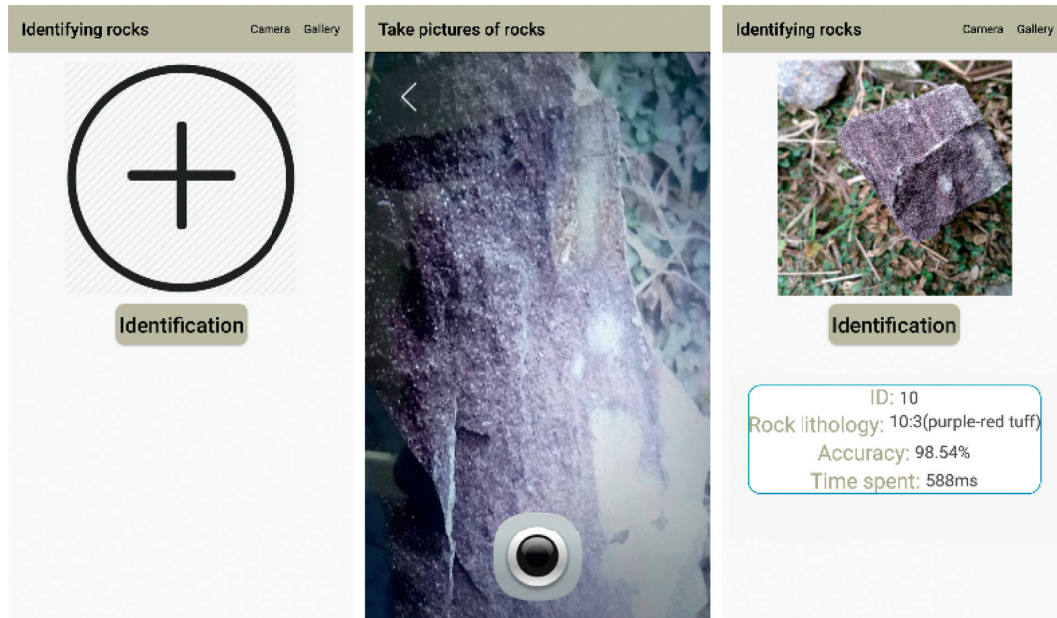


FIGURE 7: Application operator interface.

to quickly and accurately identify the types and lithology of rocks in the wild. Therefore, the accuracy and running time of identification are very important. The test data set and the (.pb) format model which has completed the training and the (.txt) format label file containing rock information are imported into the SD card of the Huawei smartphone, respectively. When the application in 2.2 is run, the rock recognition model is automatically loaded. Then, the test data set in the smartphone is read, and the recognition result of the test set is obtained. As shown in Figure 8, the accuracy of the model is represented by the confusion matrix.

Among them, the row value of the matrix is the true value. The column value is the predicted value, and the accuracy of the whole test set is 95.30%. And the accuracy of the single image is above 96%, accounting for 95% of the test data set.

The recognition time distribution of the single image of the test data set is shown in Figure 9. Among them, the average recognition time of the single image is 786 milliseconds. The maximum is 1,045 milliseconds. And the minimum is 452 milliseconds. The boxplot has no outliers, indicating stable model recognition.

3.2. Correlative Experiments and Analysis. A rock recognition model based on the ShuffleNet convolutional neural network combined with the transfer learning method was presented in this paper. In order to verify the superiority of this model, the same training data set and validation data set were used in this paper. And the accuracy and running time of this model were contrasted with other CNN models (MobileNet and SqueezeNet).

3.2.1. Compared with Other CNN Models. The training based on different batch sizes was evaluated at 8, 16, 32, and

48, respectively. Figure 10 shows the relationship between model accuracy and training period. This section takes advantage of the accuracy of the rock type and lithology. The data and other parameter settings were the same as the experimental data provided in Section 2.2.

All models converged after about 35 training epochs. The 32-batch training model achieved better performance, about 5% better than the other models. SqueezeNet was more stable and smooth during training, while MobileNet and ShuffleNet fluctuated more. It is reasonable to find that the calculation of gradient descent direction was more accurate and milder for larger batch sizes during model training. Smaller batch sizes resulted in more randomness and made it harder to achieve optimal performance.

3.2.2. Execution Time Evaluation. Different CNN models contain layers of various depths and widths, number of filters, and size and shape of filters, which lead to different structures, parameters, and complexity. In this paper, rock recognition models based on ShuffleNet training are compared with models based on other convolutional neural network training. The following results were obtained by evaluating the running time of different models. Running time of different CNN models was evaluated. Training and testing time of MobileNet, ShuffleNet, SqueezeNet, and ResNet50 models with various batch sizes (8, 16, 32, and 48) were evaluated. Table 2 gives the experiment results.

Using batch sizes 8, 16, 32, and 48 during each iteration, the MobileNet model had the longest training time, and these training times were 1.265 s, 2.364 s, 4.728 s, and 8.512 s, respectively. The ShuffleNet model had the shortest training time, about 75% of that of MobileNet. For test time, ShuffleNet took the shortest amount of time, with a time of 0.125 s. Comparing with the ResNet50 model, the running efficiency was greatly improved with compressed CNN

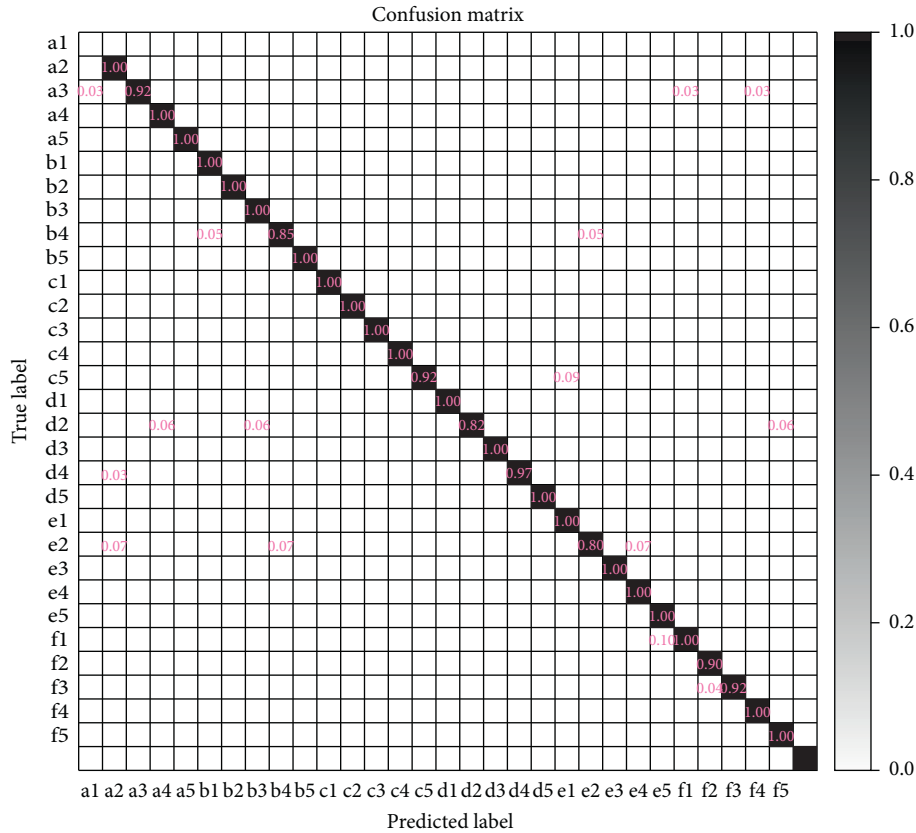


FIGURE 8: Confusion matrix.

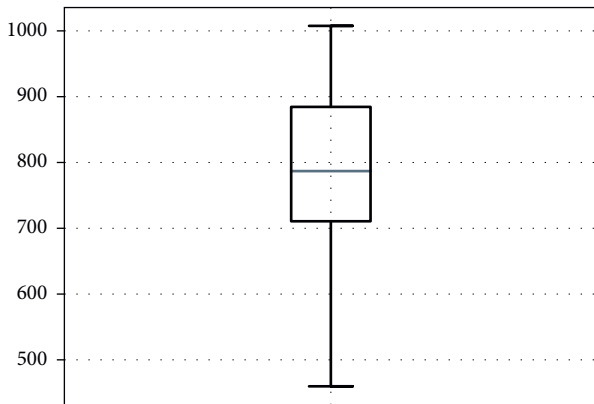


FIGURE 9: Recognition time of a rock image on the smartphone.

models. For the size of space occupied by the model, MobileNet, ShuffleNet, and SqueezeNet required 34.5M, 18.2M, and 25M, respectively, but 219.4M is needed for the ResNet50 model. The experimental results also show that the ShuffleNet model is efficient and occupies less space. It is 7 times faster than the standard convolutional network ResNet50 model and takes up 12 times less space.

3.3. Comparing the Recognition Times Using Different Android Smartphones. The compressed CNN model was deployed on the Android smartphone, and its performance was tested. After the model was converted, ShuffleNet, MobileNet, and

SqueezeNet files were, respectively, 15.2 MB, 8.4 MB, and 42.8 MB. Table 3 shows the model test results for the selected Android smartphone. Huawei, Samsung, and Oppo phones were used for testing, and the processors of the three phones are Kylin970, Qualcomm710, and Qualcomm670. The results showed that smartphones were very efficient and could perform operations in 0.5 seconds, enabling real-time applications. The Huawei phone achieved the best performance, taking 0.283 seconds to execute the model due to the neural network processing unit contained in it. Three models were deployed on the same smartphone. There were three different types of smartphones. As shown in Table 3, the model of ShuffleNet achieved the best performance.

3.4. The Advantages of the Presented Method. The rock identification method in this paper is compared with the traditional method and the method based on rock slice image processing. Table 4 shows the advantages of the presented method. The model of this paper can quickly identify the types and properties of rocks on the condition that the accuracy requirements are met. The presented method can quickly get the recognition results in less than 1 second after taking photos in the field. And there is no need to make rock flakes to reduce the cost of identification. The experimental results show that the convolutional network model has obvious performance in model compression and computation. It is suitable for rapid and accurate recognition of rock lithology under field offline conditions.



FIGURE 10: Model accuracy versus training epoch on various batch sizes: (a) result on the MobileNet model; (b, c) result on SqueezeNet and ShuffleNet models, respectively. Orange, yellow, green, and wine curves represent batch sizes equal 8, 16, 32, and 48, respectively.

TABLE 2: Execution time (millisecond).

Execution time	Batch size	MobileNet	ShuffleNet	SqueezeNet	ResNet50
Training time	8	657	158	553	1465
	16	1182	212	995	2935
	32	2364	424	1770	5048
	48	4256	954	2655	8993
Testing time	X	249	125	192	915
Space occupation	X	34.5M	18.2M	25M	219.4M

TABLE 3: Execution time on Android smartphones (millisecond).

CNN models	Oppo R17	Samsung Galaxy A8s	Huawei P20	Our PC server
MobileNet	1145	531	417	249
ShuffleNet	457	308	283	125
SqueezeNet	621	367	339	192

4. Discussion

In this paper, lightweight convolutional neural network ShuffleNet was used to solve the problem of recognizing types and lithology of rocks in the field. The pretraining model of ShuffleNet was fine-tuned in combination with the transfer learning method, and then the model was retrained on the rock image data set in this paper. Finally, this paper developed an intelligent program for quickly identifying rocks for geological survey. This program enabled effective

recognition of 30 types of rocks such as granite, rhyolite, tuff, and breccia before deploying trained rock models to Android smartphones. In this paper, the accuracy of the recognition model reached 97.65% in the verification data set of PC. And the accuracy of the recognition model on the test data set of the smartphone was 95.30%. The average recognition time of a single rock image was 786 milliseconds. The model size was only 18.2 MB. For the same rock model, there was no significant difference in the results of different smartphone recognition. The model extracted features by

TABLE 4: The presented method is compared with traditional recognition methods and feature extraction methods based on the rock thin section.

Recognition methods	Recognition period	Recognition process and cost estimate	Whether it can be applied in the field	Recognition accuracy
The presented method	Real-time recognition	Use a smartphone to take pictures of the rocks directly. Approximately \$300	Can be recognized in the field	Meet the accuracy of the survey
Traditional recognition methods	More than 2 days	Collect rock specimens, make rock thin sections, and observe the thin sections under the microscope by professionals. Costs over \$850	Unable to recognize in the field	Meet the accuracy of the survey
The method for analyzing and feature extraction based on the rock thin section	More than 2 days	Collect rock samples, make rock thin sections, and recognize in PC. Costs over \$850	Unable to recognize in the field	Meet the accuracy of the survey

searching image pixel points without manual operation, thus reducing the influence of subjective factors. Compared with the recognition of rocks using the technique of rock thin section image processing [3, 43], the presented method has lower requirements on the size, imaging distance, and light intensity of the rock image. In this paper, the rock recognition model trained by ShuffleNet was compared with MobileNet and SqueezeNet training models, respectively, in terms of accuracy and running time. It was found that the rock recognition model based on ShuffleNet training has many advantages. For example, it can effectively reduce model parameters, compress model size, improve model calculation speed, and shorten model running time. This method had short recognition time and high accuracy and was suitable for fast and accurate recognition of rock images under offline conditions in the field. Based on ShuffleNet's lightweight convolutional neural network, the characteristics of rocks were effectively identified in the image. Through the tests on PC and smartphones, there was no wrong situation, which fully proved the robustness and generalization ability of the model.

The greatest contribution of this paper was to provide a solution for geological survey to quickly and accurately recognize rock lithology. Traditional recognition methods need not only collection of fresh rock samples to make rock thin section but also knowledgeable or experienced professionals to recognize the rock type and structure parameters under the microscope. The traditional method has strong subjectivity, long period, and high difficulty in the field. Therefore, the traditional identification method requires the observer to have very rich geological knowledge and experience [42–44]. At present, it was found that most of the rock deep learning recognition techniques are used to identify rock slices. The same is true for image processing techniques. Workers need to collect rock samples and go back to the office or laboratory to make the rock thin section. The recognition accuracy can meet the requirements of professional standards. But, the biggest disadvantage is that the research results of rock recognition cannot be applied to the field. And workers cannot use the research results of rock recognition to quickly and accurately identify the rock in the field [6, 45, 46]. This paper used ShuffleNet combined with the transfer learning method to train the rock recognition model. There was no need to make the rock thin section.

Geological investigators use smartphones which they carried as tools to photograph rock images in the field.

The presented method also has some limitations. The size of sample data has a crucial influence on the recognition effect of the deep learning model. When the number of images of a certain type of the rock is less, its features will be submerged, leading to poor recognition effect. It is difficult to find similar rock images with low probability of classification and recognition. Therefore, the probability of identifying such rock images is low. In this paper, the original training data set is expanded after cutting the rock image. Then, a new classification recognition model is established by training. The second test was made by classifying and recognizing the rock images with low probability. For granite, which contains a lot of minerals and has a wide range of variation in its content, the recognition and classification effect in the model are poor. Because granite mainly consists of feldspar, quartz, and black and white mica, the mineral composition of different varieties is not the same. And there may be pyroxene and amphibole, so the image features are complex, and the difficulty of recognition is increased. In addition, there were 30 types of rocks identified in this paper, and more types and quantities of training samples were needed. The model in this paper was compared with the training models of MobileNet and SqueezeNet. Experimental results show that the ShuffleNet-based rock recognition model has advantages in precision and running time. The comparison involves fewer models and requires more lightweight compression models. Adding more models is to choose a better solution in terms of precision and time.

5. Conclusions

Recognition of rock types and lithology are an important part of geological survey. In this research, ShuffleNet, a lightweight convolutional neural network designed for smartphones, was used to recognize the types and lithology of rocks. The transfer learning method was used to train the rock recognition model on the PC, and the trained model was deployed on the smartphone. This paper designed and developed an application program that runs on a smartphone. This application is not only simple in operation but also highly accurate in rock recognition. This paper solved

the problems of long recognition period and high cost in traditional recognition methods. It also makes up for the defect that the methods based on the image processing and feature extraction of rock thin section cannot recognize the rock quickly and accurately in the field. Geological investigators can quickly and accurately identify rocks by using their smartphones in the field, which is of great help to geological surveys. In the future, this paper needs to compare the rock recognition model based on ShuffleNet with more models trained by the lightweight convolutional neural network. In order to improve the accuracy and efficiency of the method, more different kinds of rock training samples were added.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors thank other team members for their help with the experiment. This research was jointly supported by Geological Survey Projects of China Geological Survey (DD20190416).

References

- [1] H.-J. Massonne, H.-J. Bernhardt, D. Dettmar, E. Kessler, O. Medenbach, and T. Westphal, "Simple identification and quantification of microdiamonds in rock thin-sections," *European Journal of Mineralogy*, vol. 10, no. 3, pp. 497–504, 1998.
- [2] F. S. Reed and J. L. Mergner, "Preparation of rock thin sections," *American Mineralogist*, vol. 38, pp. 1184–1203, 1953.
- [3] C. A. Perez, J. A. Saravia, C. F. Navarro, D. A. Schulz, C. M. Aravena, and F. J. Galdames, "Rock lithological classification using multi-scale Gabor features from sub-images, and voting with rock contour information," *International Journal of Mineral Processing*, vol. 144, pp. 56–64, 2015.
- [4] L. Yu, A. Porwal, E.-J. Holden, and M. C. Dentith, "Towards automatic lithological classification from remote sensing data using support vector machines," *Computers & Geosciences*, vol. 45, pp. 229–239, 2012.
- [5] L. Lepistö, I. Kunttu, and A. J. E. Visa, "Rock image classification using color features in Gabor space," *Journal of Electronic Imaging*, vol. 14, no. 4, Article ID 040503, 2005.
- [6] A. K. Patel and S. Chatterjee, "Computer vision-based limestone rock-type classification using probabilistic neural network," *Geoscience Frontiers*, vol. 7, no. 1, pp. 53–60, 2016.
- [7] N. Singh, T. N. Singh, A. Tiwary, and K. M. Sarkar, "Textural identification of basaltic rock mass using image processing and neural network," *Computational Geosciences*, vol. 14, no. 2, pp. 301–310, 2010.
- [8] G. Cheng and W. Guo, "Rock images classification by using deep convolution neural network," *Journal of Physics: Conference Series*, vol. 887, Article ID 012089, 2017.
- [9] P. Y. Zhang, J. M. Sun, Y. J. Jiang, and J. S. Gao, "Deep learning method for lithology identification from borehole images," in *Proceedings of the 79th EAGE Conference and Exhibition 2017*, Paris, France, June 2017.
- [10] S. Rinnen, C. Stroth, A. Rißé, C. Ostertag-Henning, and H. F. Arlinghaus, "Characterization and identification of minerals in rocks by ToF-SIMS and principal component analysis," *Applied Surface Science*, vol. 349, pp. 622–628, 2015.
- [11] H. Izadi, J. Sadri, and M. Bayati, "An intelligent system for mineral identification in thin sections based on a cascade approach," *Computers & Geosciences*, vol. 99, pp. 37–49, 2017.
- [12] Y. Zhang, M. Li, and S. Han, "Automatic identification and classification in lithology based on deep learning in rock images," *Acta Petrologica Sinica*, vol. 34, no. 2, pp. 333–342, 2018.
- [13] N. Li, H. Hao, Q. Gu, D. Wang, and X. Hu, "A transfer learning method for automatic identification of sandstone microscopic images," *Computers & Geosciences*, vol. 103, pp. 111–121, 2017.
- [14] G. Cheng, Q. Yue, and X. Qiang, "Research on feasibility of convolution neural networks for rock thin sections image retrieval," in *Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 2539–2542, Xi'an, China, May 2018.
- [15] R. Marmo, S. Amodio, R. Tagliaferri, V. Ferreri, and G. Longo, "Textural identification of carbonate rocks by image processing and neural network: methodology proposal and examples," *Computers & Geosciences*, vol. 31, no. 5, pp. 649–659, 2005.
- [16] C. Guo and Y. Liu, "Recognition of rock images based on multiple color spaces," *Science Technology and Engineering*, vol. 14, pp. 247–251+255, 2014.
- [17] Zhang X., Zhou X., Lin M., Sun J. S. N., An extremely efficient convolutional neural network for mobile devices, arXiv:1707.01083 [cs] 2017.
- [18] F. Chollet, "Deep learning with depthwise separable convolutions," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1251–1258, Honolulu, HI, USA, July 2017.
- [19] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision—ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Springer International Publishing, Berlin, Germany, pp. 818–833, 2014.
- [20] F. Bordignon, L. P. D. Figueiredo, R. Exterkoetter, B. B. Rodrigues, and M. D. Correia, "Deep Learning for Grain Size and Porosity Distributions Estimation on micro-CT Images," in *Proceedings of the 16th International Congress of the Brazilian Geophysical Society & Expogef*, Rio de Janeiro, Brazil, August 2019.
- [21] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, February 2017.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.
- [23] Han S., Mao H., Dally W. J., Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding, arXiv:1510.00149 [cs] 2015.
- [24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: inverted residuals and linear

- bottlenecks,” in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT, USA, June 2018.
- [25] Niu W., Ma X., Wang Y., Ren B, 26 ms inference time for ResNet-50: towards real-time execution of all DNNs on smartphone, arXiv:1905.00571 [cs, stat] 2019.
- [26] Y. Ganin, E. Ustinova, H. Ajakan et al., “Domain-adversarial training of neural networks,” in *Domain Adaptation in Computer Vision Applications*, G. Csurka, Ed., Springer International Publishing, Cham, Switzerland, pp. 189–209, 2017.
- [27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 3320–3328, Curran Associates, Inc., New York, NY, USA, 2014.
- [28] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, Salt Lake City, UT, USA, June 2018.
- [29] Jang E., Gu S., Poole B., Categorical reparameterization with gumbel-softmax, arXiv:1611.01144 [cs, stat] 2016.
- [30] F. Zang and J. Zhang, “Softmax discriminant classifier,” in *Proceedings of the 2011 Third International Conference on Multimedia Information Networking and Security*, pp. 16–19, Shanghai, China, November 2011.
- [31] A. Caliskan, M. E. Yuksel, H. Badem, and A. Basturk, “Performance improvement of deep neural network classifiers by a simple training strategy,” *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 14–23, 2018.
- [32] Xie X., Zhou Y., Kung S.-Y., H. G. C.: hierarchical group convolution for highly efficient neural network, arXiv: 1906.03657 [cs] 2019.
- [33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105, Curran Associates, Inc., New York, NY, USA, 2012.
- [34] E. Zawadzka-Gosk, K. Wolk, and W. Czarnowski, “Deep learning in state-of-the-art image classification exceeding 99% accuracy,” in *Advances in Intelligent Systems and Computing*, pp. 946–957, Springer, Cham, Switzerland, 2019.
- [35] S. Tao, T. Zhang, J. Yang, X. Wang, and W. Lu, “Bearing fault diagnosis method based on stacked autoencoder and softmax regression,” in *Proceedings of the 2015 34th Chinese Control Conference (CCC)*, pp. 6331–6335, Hangzhou, China, July 2015.
- [36] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7263–7271, Honolulu, HI, USA, July 2017.
- [37] Shafiee M. J., Li F., Chwyl B. W. A., SquishedNets, Squishing SqueezeNet further for edge device scenarios via deep evolutionary synthesis, arXiv:1711.07459, 2017.
- [38] A. G. Santos, C. O. De Souza, C. Zanchettin, D. Macedo, A. L. I. Oliveira, and T. Ludermit, “Reducing SqueezeNet storage size with depthwise separable convolutions,” in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, Rio de Janeiro, Brazil, October 2018.
- [39] M. Abadi, P. Barham, J. Chen et al., “TensorFlow: a system for large-scale machine learning,” in *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming—ICFP 2016*, pp. 265–283, Nara Japan, September 2016.
- [40] M. A. S. Adhiwibawa, M. R. Ariyanto, A. Struck, K. R. Prilianti, and T. H. P. Brotosudarmo, “Convolutional neural network in image analysis for determination of mangrove species,” in *Proceedings of the Third International Seminar on Photonics, Optics, and Its Applications (ISPhOA 2018)*, Surabaya, Indonesia, August 2019.
- [41] H. Liu, X. Ma, M. Tao et al., “A plant leaf geometric parameter measurement system based on the android platform,” *Sensors*, vol. 19, no. 8, p. 1872, 2019.
- [42] J. Kang, J. Lee, and D.-S. Eom, “Smartphone-based traveled distance estimation using individual walking patterns for indoor localization,” *Sensors*, vol. 18, no. 9, p. 3149, 2018.
- [43] S. Karimpouli and P. Tahmasebi, “Segmentation of digital rock images using deep convolutional autoencoder networks,” *Computers & Geosciences*, vol. 126, pp. 142–150, 2019.
- [44] L. Shu, G. R. Osinski, K. McIsaac, and D. Wang, “An automatic methodology for analyzing sorting level of rock particles,” *Computers & Geosciences*, vol. 120, pp. 97–104, 2018.
- [45] S. Aligholi, R. Khajavi, and M. Razmara, “Automated mineral identification algorithm using optical properties of crystals,” *Computers & Geosciences*, vol. 85, pp. 175–183, 2015.
- [46] F. J. Galdames, C. A. Perez, P. A. Estévez, and M. Adams, “Classification of rock lithology by laser range 3D and color images,” *International Journal of Mineral Processing*, vol. 160, pp. 47–57, 2017.

Research Article

Research on Indoor Scene Classification Mechanism Based on Multiple Descriptors Fusion

Ping Ji, Danyang Qin , Pan Feng, Tingting Lan, and Guanyu Sun

Key Lab of Electronic and Communication Engineering, Heilongjiang University, Harbin, China

Correspondence should be addressed to Danyang Qin; qindanyang@hlju.edu.cn

Received 25 August 2019; Accepted 22 January 2020; Published 16 March 2020

Guest Editor: Malik Jahan Khan

Copyright © 2020 Ping Ji et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This study aims at the great limitations caused by the non-ROI (region of interest) information interference in traditional scene classification algorithms, including the changes of multiscale or various visual angles and the high similarity between classes and other factors. An effective indoor scene classification mechanism based on multiple descriptors fusion is proposed, which introduces the depth images to improve descriptor efficiency. The greedy descriptor filter algorithm (GDFA) is proposed to obtain valuable descriptors, and the multiple descriptor combination method is also given to further improve descriptor performance. Performance analysis and simulation results show that multiple descriptors fusion not only can achieve higher classification accuracy than principal components analysis (PCA) in the condition with medium and large size of descriptors but also can improve the classification accuracy than the other existing algorithms effectively.

1. Introduction

With the rapid development of the Internet and the increasing demand for applications based on location awareness, location-based services are getting extensive attention. Most people cannot live without the location service and the navigation system based on GPS (Global Position System) in their daily life. Obviously, outdoor localization technology has been relatively mature, and many mobile devices also refer to outdoor location technology [1, 2, 3, 4]. Due to the particularity of indoor environment, the GPS signal cannot directly meet the requirements of indoor localization service. At present, there are many indoor localization methods [4–6], mainly including WiFi, RFID, Bluetooth, Ultrawide band, and so on. Nowadays, the visual indoor localization system [7–9] is attracting more and more attentions of the researchers all over the world due to the advantages of low deployment cost, strong autonomy, and high localization accuracy.

A large visual database, namely, Visual Map, has occasionally been established at offline stage to achieve accurate indoor visual localization. Visual Map may contain a large number of images or image features of different scenes

and corresponding location information, which is the foundation of visual indoor localization. When the user performs a location query online, the image will be retrieved in the Visual Map. Traditional image retrieval algorithms rely on pixel point matching [10, 11], which can only give the results of image matching but does not contain the visual image location information. In addition, existing image retrieval algorithms often carry out global traversal search, which leads to excessive time overhead and is not conducive to real-time localization of mobile users. Therefore, an effective indoor scene classification mechanism is proposed in this paper based on multiple descriptors fusion. The images in Visual Map will be classified according to the scenes, so as to reduce the time overhead of visual images retrieval at online stage and improve the efficiency and accuracy of indoor scene classification. In this paper, both the visual information and the depth information of an image are fused. The visual image mainly contains color information, and each point on the depth image corresponds to the visual image and contains position information. Both types of images are captured by Microsoft Kinect 2.0.

In the indoor scene classification mechanism, the initial descriptor set containing two kinds of image descriptors will

be generated by the existing spatial pyramid model (SPM) [12, 13]. Then, the greedy descriptor filter algorithm (GDFA) will be proposed to find out the valuable descriptors. Multiple fusion descriptors will be generated by homologous and nonhomologous combination to further enhance the effectiveness of descriptors. Finally, support vector machine (SVM) will be adopted for classification. The overall framework of the indoor scene classification mechanism is shown in Figure 1.

The remaining of the paper is arranged as follows: Section 2 reviews the research progress of scene classification techniques and their applications in indoor scenes. Section 3 describes the generation of the initial descriptor set and the descriptor filtering in detail. Section 4 introduces the experimental database of this paper and shows descriptor evaluation results. In Section 5, two combinations of homologous and nonhomologous will be realized and the combination results will be evaluated. Section 6 concludes the article.

2. Motivation

At the Scene Understanding Symposium held at MIT in 2006, an important point was clearly stated for the first time, namely, scene classification is a new promising research direction for image understanding. Although existing classification methods claim to be able to solve any scene classification problems [14, 15], the experimental outcome shows that only the outdoor scene classification can be effectively solved by these methods, while the indoor scene classification problems may still be a challenging task. In addition, [16] shows that the classification accuracy of the indoor scene is far lower than that of the outdoor scene adopting the same feature extraction and classification recognition methods. Therefore, it is important to improve the classification accuracy of the indoor scene.

In early studies, low-level features of images were usually extracted to classify scenes, such as color, texture, and shape [17–19]. However, these methods based on low-level features have not been a hot topic in the field of scene classification due to its unsatisfactory classification effect. In order to overcome such problems, the methods based on middle-level features of image are proposed. The global feature Gist is adopted and improved in [20]. The good identification ability of scale invariant feature transform (SIFT) makes it always be adopted as the local features with the highest priority in many scene recognition algorithms [21]. Shi et al. [22] proposed an indoor scene classification algorithm based on the enhancement of visual sensitive area information. And local features and global features are integrated by the visual sensitive area information.

With the rise of Kinect, the scene classification algorithm based on depth information [24, 25] has received more and more attention. The histogram of oriented gradient (HOG) algorithm [26] is adopted to classify depth images and visual images, respectively [28]. SIFT is adopted to extract features of depth images and color images, and SPM coding is adopted to classify images after feature fusion [29]. SIFT of visual images and speeded up robust features (SURF) [27] of

depth images are fused to classify images [30]. Five deep core feature extraction algorithms are designed in [31] to extract the size, edge, and shape information of visual images, respectively, and the extracted information is fused for classification.

As research continues, the model based on the convolutional neural network (CNN) [16, 23] has attracted the researchers. However, massive training sets are required in CNN, which may result in relatively long training time. In addition, CNN usually has high computing requirement on the platform, so it is difficult to realize indoor scene classification on the platform with limited computing resource.

3. Multiple Image Descriptor Generation and Filtering

Inspired by [28–31], visual information and depth information will be fused in this paper. The higher accuracy indoor scene classification effect will be achieved by the spatial 3D information contained in the depth image, which is insensitive to light and reflects the position relationship between objects. Features of the original images will be extracted by D-SIFT (Dense SIFT) [32], and similar features will be clustered to form BoW (Bag-of-Words) [33–35] by K-means [36, 37]. Based on BoW, the initial descriptors set including visual image descriptors and depth image descriptors will be generated with the construction of SPM. It is true that the number of initial descriptors is large and the quality is uneven. In addition, combining directly with unfiltered initial descriptors will lead to an explosion of the combined results. Therefore, a simple and effective descriptor filtering algorithm ought to be proposed to obtain those valuable descriptors.

3.1. Initial Descriptors Generation. The descriptor generated expression could be derived from the following procedure. Let I be any input image and x be a descriptor generated by the image. \mathbb{L} is a set of predefined class tags, and l is one of them. The function of generating descriptor x from image I can be expressed as $g(I) = x$, and the probability of successfully matching descriptor x to class tag l is $P(l|x)$. Therefore, the expression of the most appropriate class tag \tilde{l} will be

$$\tilde{l} = \arg \max_{l \in \mathbb{L}} P(l|g(I)). \quad (1)$$

The key to the research will be turning the initial descriptors into valuable descriptors with high classification accuracy. In order to find such descriptors, equation (1) will be further optimized. On the premise of the best descriptor filtering and combination methods, a correct class label assigned to input image I will be \hat{l} ($\hat{l} \neq \tilde{l}$) and \mathcal{X} is adopted to express a set of multiple image descriptors. Then, the optimized descriptor generation expression will be

$$\tilde{g}(I) = \arg \max_{g(I) \in \mathcal{X}} P(\hat{l}|g(I)). \quad (2)$$

According to equation (2), the initial descriptors generated by the input image can only get the desired

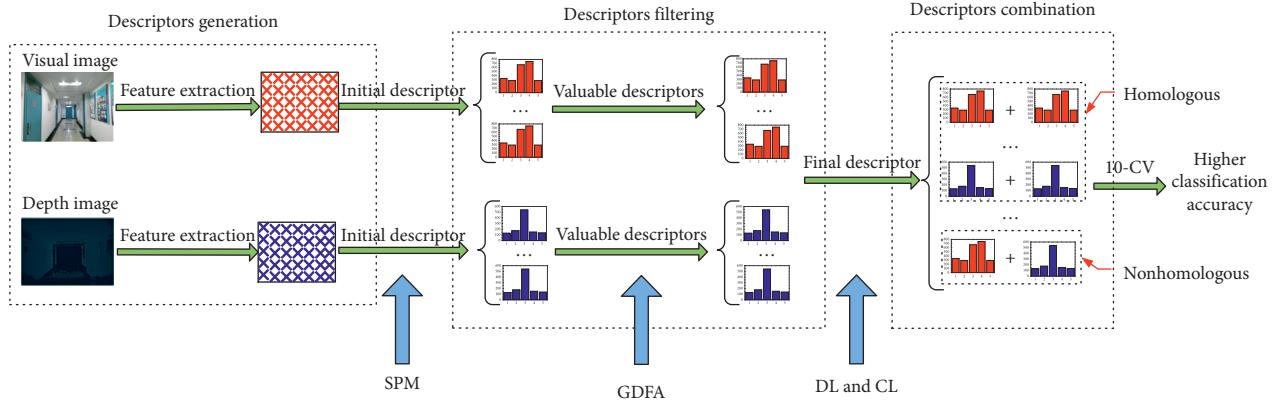


FIGURE 1: The indoor scene classification mechanism.

classification effect through filtering and combination. Initial descriptors are large in number and poor in quality, while descriptor filtering can discard worthless descriptors and descriptor combination can improve the effectiveness of descriptors. The descriptor generation process based on SPM will be described as follows.

3.2. Spatial Pyramid Model. In recent years, the BoW model has been widely adopted in computer vision. It takes the image features as visual words and classifies images by counting the number of visual words in each image. However, the traditional BoW lacks the spatial position information [29]. In this research, SPM will be established to cut the image into scale cells, then the number of visual words will be counted in each cell and the histograms can be drawn. Finally, histogram features at all scales will be linked together to form an eigenvector. We assume that a part of visual words has been selected as basic features. The steps of descriptor generation based on SPM are described in detail as follows:

- (i) Extracting the D-SIFT feature.
- (ii) Mapping each feature point to the corresponding visual word.
- (iii) Cutting the image and constructing spatial pyramid hierarchy (three cutting methods, such as vertical cutting method, horizontal cutting method, and grid cutting method, are adopted in this paper, as shown in Figures 2(a)–2(c), respectively).
- (iv) Counting the number of visual words in each cell and plotting histograms for each cell.
- (v) Connecting all histograms to form a feature vector as the image descriptor.

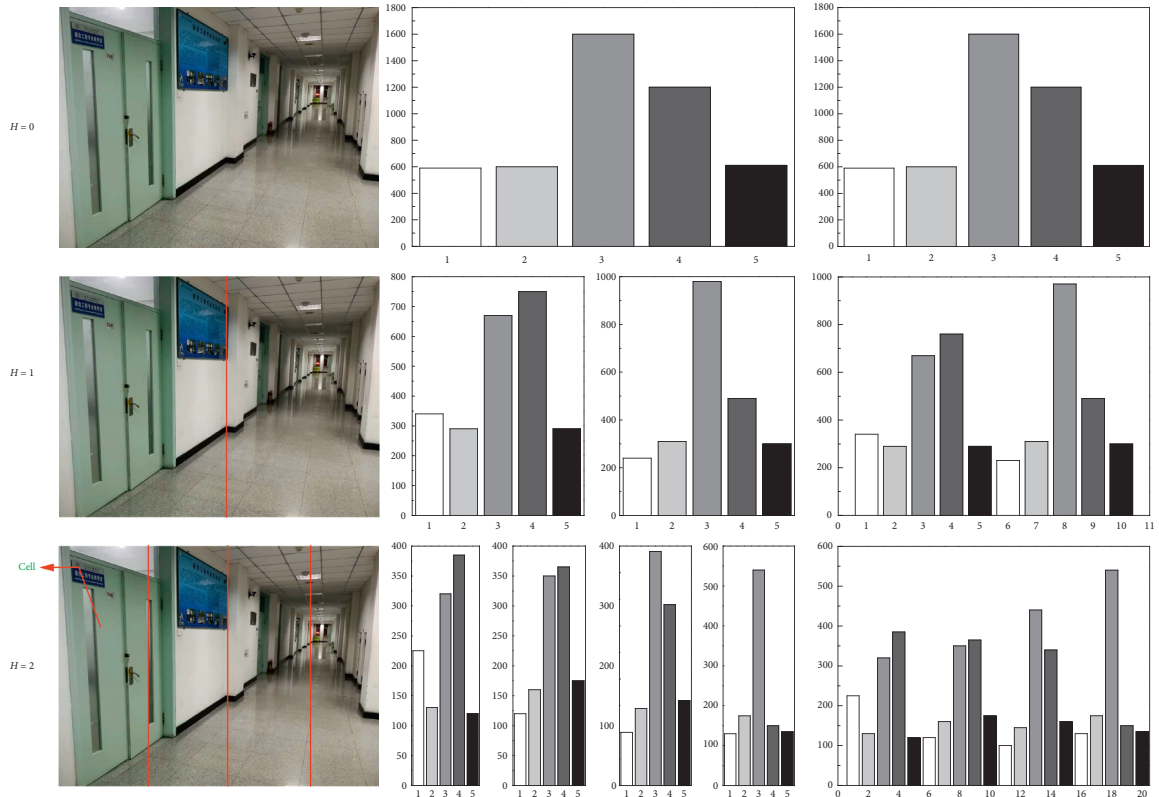
The SPM-based descriptor generation process is shown in Figures 2(a)–2(c), and each cutting type will be divided into three columns for clear explanation. As shown in Figure 2(a), the first column shows the cutting type of the initial image, the second column represents the statistical results of visual words for each cell, and the initial descriptors formed by connecting the second column histograms are shown in the third column. The

image contains 5 visual words; three pyramid hierarchies; and vertical, horizontal, and grid, the three cutting methods. The descriptors generation based on SPM mainly depends on three important parameters: BOW size (S), pyramid hierarchy (H), and cutting method (C). $H = 0$ represents the first hierarchy, and the image is cut 0 times. $H = 1$ represents the second hierarchy, and the image is cut 1 time; $H = 2$ represents the third hierarchy, and the image is cut 2 times. Therefore, the number of cutting depends on H . In other words, when $H = h$, the image will be cut h times, and the number of cells generated after cutting is 2^{hC} . Finally, seven different descriptors are obtained in Figure 2, whose size increases exponentially with the number of H and C and has a linear relationship with dictionary size S . The calculation formula of descriptor size η is as follows:

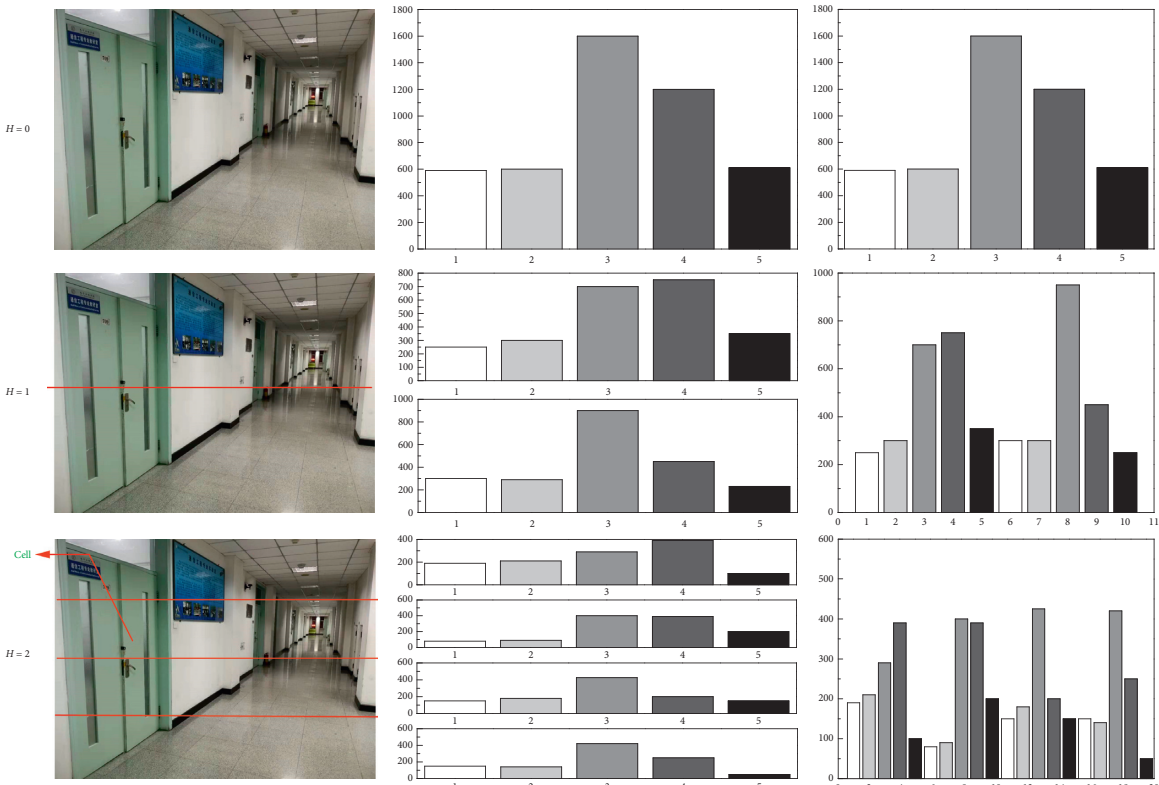
$$\eta = S \cdot 2^{hC}. \quad (3)$$

As we know, image descriptors contain semantic and spatial distribution information of the scene. S will determine the semantic meaning of descriptors, while H and C will focus on the spatial distribution of descriptors, ensuring that more detailed information can be provided. The larger S will provide more detailed semantic information, making features more obvious and more representative. However, if there are a lot of visual words, the histogram will become longer, which will affect the image retrieval and matching process, subsequently. Analogously, a higher pyramid hierarchy contains more detail, while a lower hierarchy is more general.

As can be seen from [12, 13, 38], the standard values of the three parameters are $S = 20, 50, \text{ and } 100$; $H = 0, 1, \text{ and } 2$; and $C = 1$ (horizontal and vertical segmentation) and 2 (grid segmentation), respectively. 21 different visual image descriptors and 21 depth descriptors can be obtained by combining these standard values. The reason why the number of descriptors is 21 instead of 27 (3^3) is that $H = 0$ in the pyramid model does not cut the image, with no demands for combination indeed. In other words, for any S , the first pyramid hierarchy will deal with only one descriptor, while the second and third pyramid hierarchies will deal with three descriptors.



(a)



(b)

FIGURE 2: Continued.

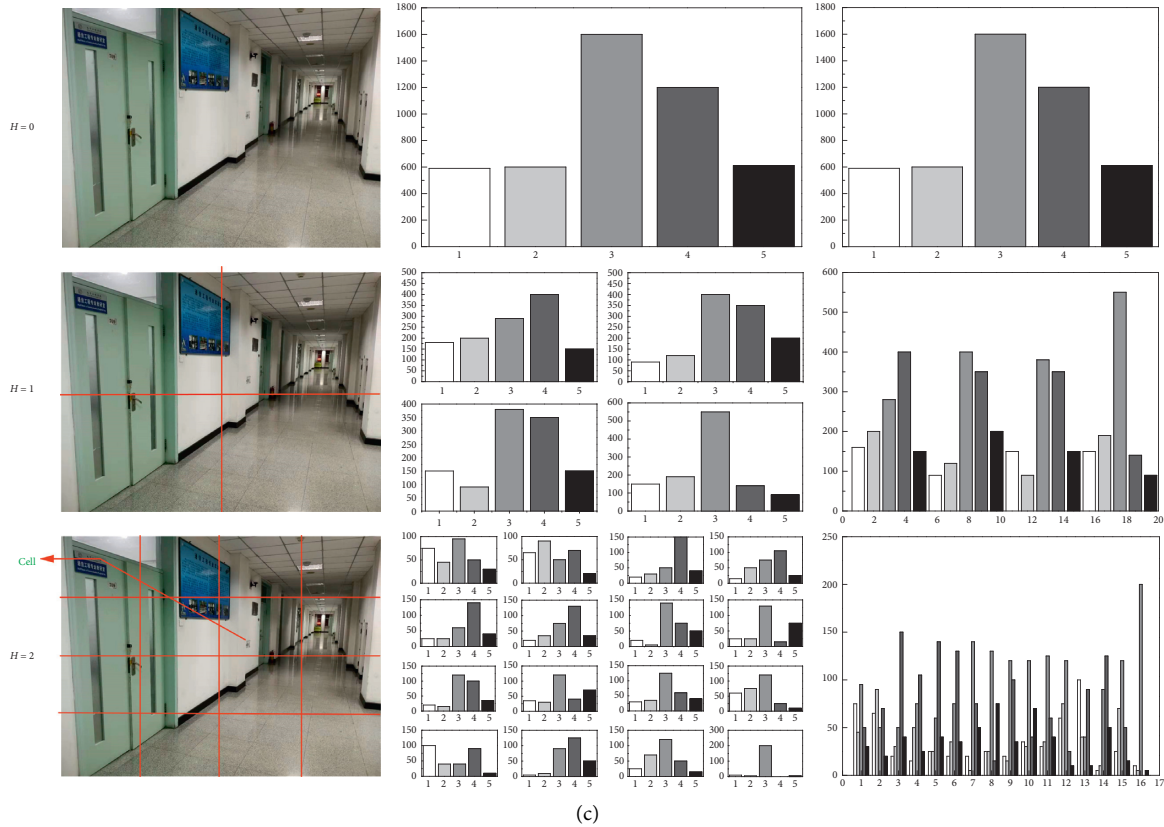


FIGURE 2: Construction of SPM and generation of the initial descriptor set. (a) Vertical cutting. (b) Horizontal cutting. (c) Grid cutting.

3.3. *Descriptors Filtering.* In this section, the greedy descriptor filter algorithm (GDFA) will be proposed to find the most valuable descriptors in the initial descriptor set. Since η of the initial descriptors mainly gathered in $(0, 400]$ (as shown in Figure 3), η is divided into three continuous intervals $(0, 150]$, $[150, 350]$, and $[350, \infty)$ for the convenience of descriptor filtering. We assume that large, medium, and small intervals are suitable for our data-gathering platform with small, medium, and high computing power configurations, respectively. The descriptor weight α is related to the descriptor classification accuracy ζ and descriptor size η . In order to obtain descriptors with smaller size and higher accuracy, the calculation formula of the weight α could be defined as follows:

$$\alpha = \frac{\zeta}{\log \eta}. \quad (4)$$

The greedy descriptor filtering algorithm (GDFA) flow is given in Algorithm 1.

At first, the weight of all descriptors is calculated according to equation (4). Next, the descriptor size is divided into $(0, 150]$, $[150, 350]$, and $[350, \infty)$ three continuous intervals, and then the descriptors are sorted in order of weight values from the largest to the smallest. The descriptor with the largest weight in \mathcal{N}_i is filtered and added to the first position in F . If the descriptor weight is greater than 95% of the weight of the previous selected descriptor, that is, $(\alpha_i > 0.95\alpha_{i-1})$, the descriptor is filtered out; otherwise, the next descriptor will be compared. GDFA not only could find

out the most valuable descriptors in each interval, but also could filter out descriptors with similar weights.

4. Descriptor Evaluation

4.1. *Experimental Database.* In order to study the indoor scene classification mechanism, as shown in Figure 4(a), the indoor image data gathering platform with Microsoft Kinect 2.0, independently developed by the laboratory, will be adopted to carry out image data gathering in the Heilongjiang University physical laboratory building. The database contains visual and depth images captured in 9 indoor scenes under different lighting conditions. To cite some examples, Figure 4(b) shows part of the database images.

The database images will be randomly divided into 5 sequences, namely, Training 1, 2, and 3 and Test 1 and 2. The image number for 9 scenes in 5 sequences is listed in Table 1.

4.2. *Evaluation Results and Analysis.* K-fold cross-validation could be a common accuracy test method, which can effectively avoid over-learning and under-learning. 10-CV (10-fold cross-validation) will be adopted to evaluate the classifier model in this section. To ensure that each cross-validation image is similar, a subset of 30 consecutive images will be randomly assigned to Fold1–Fold10 (represents 10 subsets of the 30 images), which effectively prevented any deviation caused by the time continuity in the data set. Figure 5 shows

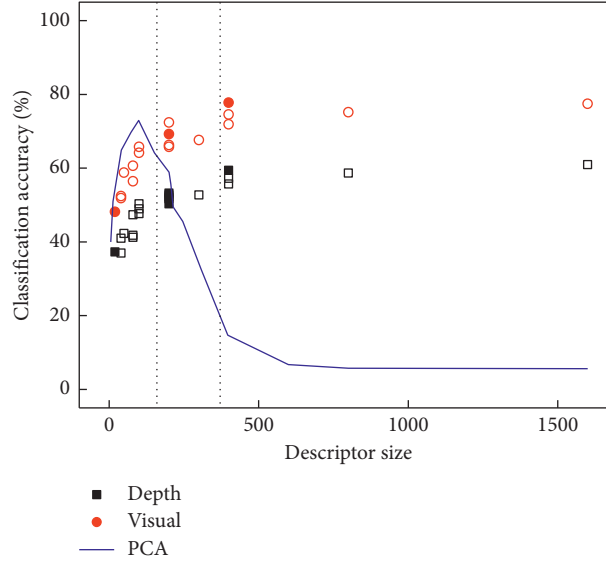


FIGURE 3: Greedy descriptor filtering algorithm versus PCA.

```

Input: descriptor list--- $\mathcal{L}$ 
         descriptor classification accuracy list--- $\zeta$ 
         descriptor size list--- $\eta$ 
(1) for  $j \in [0, \text{size}(\mathcal{L})]$  do
(2)    $\alpha[j] \leftarrow \zeta[j]/\log(\eta[j])$ 
(3) end
(4) Divide the descriptor size into  $(0, 150]$ ,  $[150, 350]$ , and  $[350, \infty)$  three continuous intervals
(5) for  $i \in [1, 2, 3]$  do
(6)   Divide  $\mathcal{L}$  into new lists  $\mathcal{N}_i$ 
(7)   Sort the descriptors in  $\mathcal{N}_i$  in order of weight values from largest to smallest
(8)   for  $j \in [0, \text{len}(\mathcal{N}_i)]$  do
(9)     Filter the descriptor  $\mathcal{N}_i[1]$  with the largest weight in  $\mathcal{N}_i$  and add  $\mathcal{N}_i[1]$  to  $\Phi_i$ 
(10)    if  $\mathcal{N}_i[j-1]$  is filtered and
(11)      $\alpha[j] > 0.95 * \alpha[j-1]$  then
(12)       Add  $\mathcal{N}_i[j]$  to  $\alpha_i$ 
(13)     else
(14)       end
(15)   end
Output: filtered descriptor list--- $\alpha$ 

```

ALGORITHM 1: Greedy descriptor filtering algorithm.

the distribution of each scene in the data set in each fold of 10-CV and global distribution. It is worth noting that scenes in the data set are not evenly distributed in Fold1–Fold10.

Table 2 shows the classification accuracy of initial descriptors of 42 visual image descriptors and depth image descriptors after 10 times of cross-validation. In SPM, when $H=0$, for any kind of segmentation type, there is no image cutting and the generated descriptors are identical, so the evaluation results are identical too. By comparing the results of visual images and depth images, we can find that the classification accuracy of depth images is significantly lower than that of visual images. The reason may be that the visual coding technology (visual coding is the mapping between

data and visual results) of the depth image is not accurate enough to obtain fine-grained data.

G DFA can find the valuable descriptors from the initial descriptor set, which will facilitate the descriptor combination work in Section 5. Table 3 shows the internal parameters and classification accuracy of the 4 visual image descriptors and 7 depth image descriptors filtered by G DFA, analogously, and the evaluation data are from 10-CV. In other words, the 42 initial descriptors given in Table 2 are reduced to 11 through the filtering of G DFA. These descriptors may have the highest weight in $(0, 150]$, $[150, 350]$, and $[350, \infty)$ intervals.

PCA is one of the classical and widely algorithms in current data preprocessing algorithms. Dimensionality

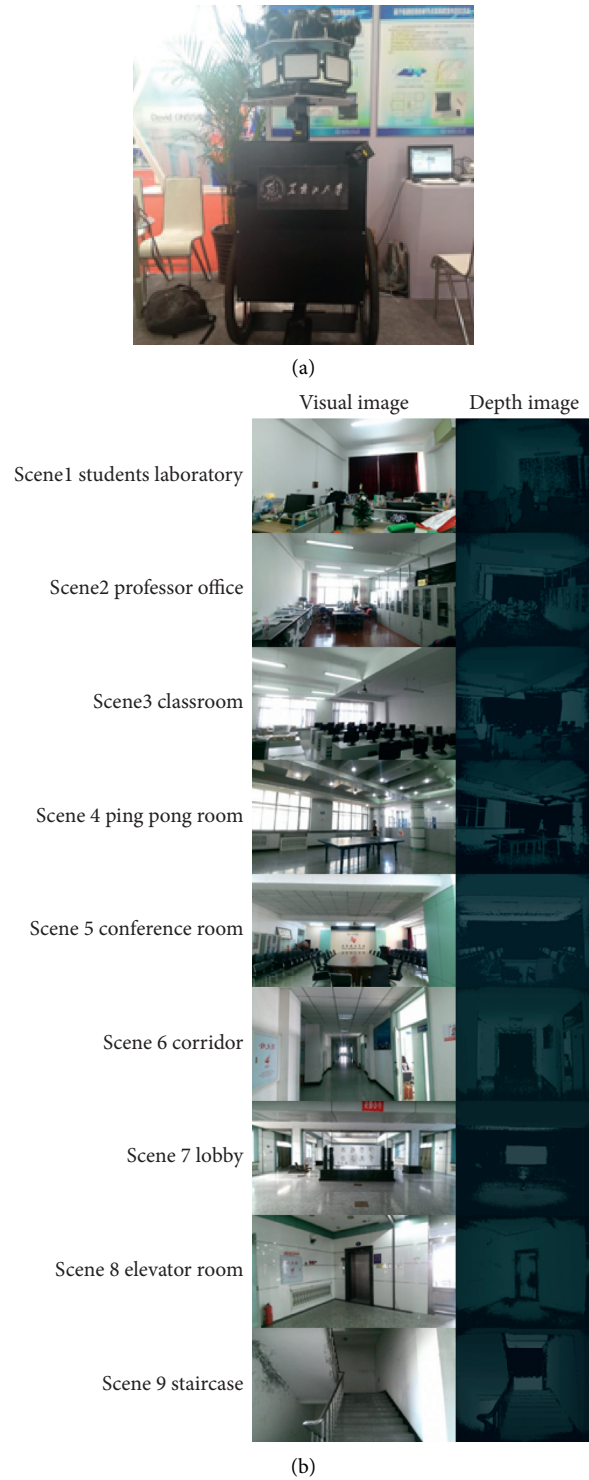


FIGURE 4: Indoor image data gathering platform (a) and part of database (b).

reduction with PCA can preserve the most important features in high-dimensional data and remove noise and worthless features, which could improve data quality and data processing speed. Figure 3 shows the comparison between the filtering result of G DFA and the dimensional reduction result of PCA (the solid point in Figure 3 is the descriptor obtained by the G DFA, and the dotted line

separates three intervals). As observed, when descriptor size is in $(0, 150]$, PCA outperforms both visual descriptors and depth descriptors. But when descriptor size is in $[150, 350]$ and $[350, \infty)$, the performance of PCA begins to decline, which may indicate that G DFA performs better than PCA, especially when the descriptor size is medium or large.

TABLE 1: The number of images of 9 scenes in 5 sequences.

Scene	Frame				
	Training 1	Training 2	Training 3	Test 1	Test 2
1	438	498	444	511	319
2	140	152	84	95	147
3	119	80	65	109	229
4	421	452	376	392	442
5	408	336	247	307	942
6	664	599	388	692	1287
7	126	79	60	95	223
8	153	96	118	140	193
9	198	240	131	104	241
All	2267	2532	1913	2445	4023

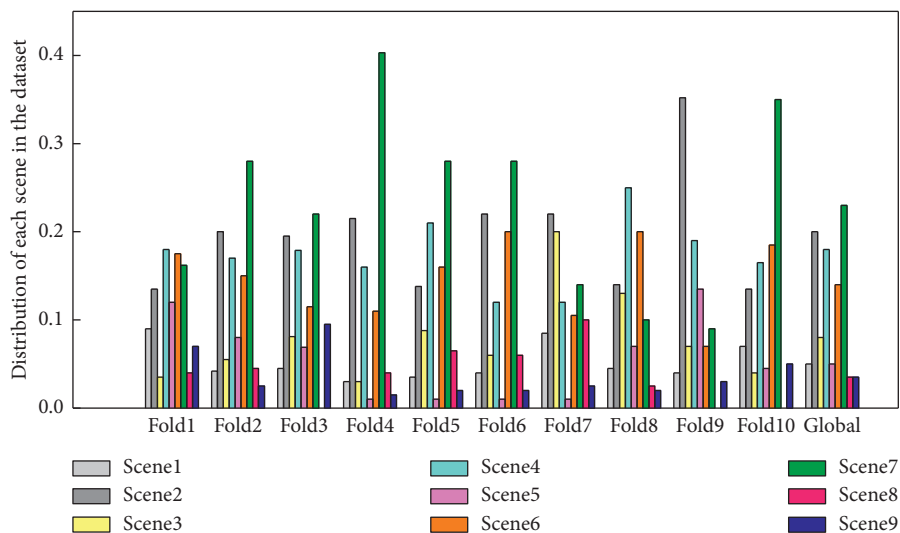


FIGURE 5: Distribution of each scene in Fold1-Fold10 and the global data set.

TABLE 2: Evaluation results of initial descriptors.

C	H	Visual			Depth		
		S = 20 (%)	S = 50 (%)	S = 100 (%)	S = 20 (%)	S = 50 (%)	S = 100 (%)
Vertical	0	48.13	58.75	66.20	37.07	42.49	50.06
	1	51.84	63.36	69.53	37.23	48.83	52.55
	2	56.38	65.88	72.09	41.03	50.51	55.44
Horizontal	0						
	1	52.51	64.68	72.01	40.93	47.53	51.78
	2	60.73	72.36	77.81	47.39	53.65	59.40
Grid	0						
	1	56.25	69.02	74.34	41.82	52.95	57.07
	2	67.53	75.26	77.24	52.76	58.37	60.86

5. Descriptor Combination

The most valuable descriptors have been selected by GDFA in Section 4. In order to further obtain the high-quality and highly efficient final descriptor, this section will propose a multiple descriptor combination algorithm (this section only combines two descriptors) although this step might increase the running time of scene classification. There will be two descriptor combination levels, as shown in Figure 6.

One is the descriptor level (DL), which can be input to SVM1 after the descriptors of Image1 and Image2 have been connected into one combination descriptor, as shown in Figure 6(a). The other one is the classifier level (CL), which weights the different response results after Image1 and Image2 have been input to SVM1 and SVM2 separately, as shown in Figure 6(b). Also, this section will discuss homologous combinations ($V+V$ or $D+D$) and nonhomologous combinations ($V+D$).

TABLE 3: Filtering results of GDFA.

Image type	Parameters			Filtering criteria	
	S	H	C	ζ (%)	η (interval)
V1	20	0	—	48.13	20 (1)
V2	50	2	Horizontal	72.36	200 (2)
V3	100	1	Horizontal	72.01	200 (2)
V4	100	2	Horizontal	77.81	400 (3)
D1	20	0	—	37.07	20 (1)
D2	50	2	Horizontal	53.65	200 (2)
D3	50	1	Grid	52.95	200 (2)
D4	100	1	Vertical	52.55	200 (2)
D5	100	1	Horizontal	51.78	200 (2)
D6	50	2	Vertical	50.51	200 (2)
D7	100	2	Horizontal	59.40	400 (3)

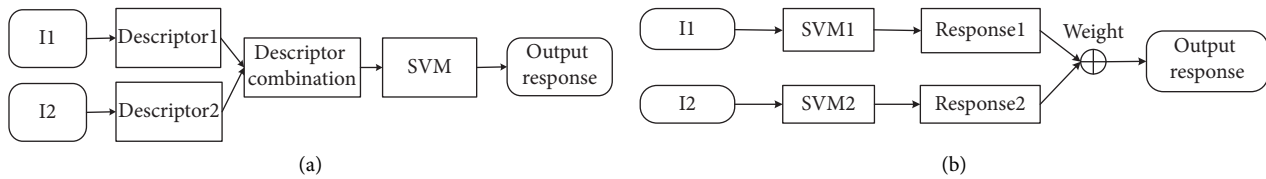


FIGURE 6: Descriptor combination level. (a) DL. (b) CL.

The combined sequences of training 1, 2, and 3 given in Table 1 will be used as the training set, while Test 1 and Test 2 will be used as the test set. These 5 sequences have the same scene. But it is noted that the light variation in Test 1 is stronger than that in Test 2.

5.1. Homologous Combinations. This section will combine two descriptors extracted from the same image type, namely, $V + V$ or $D + D$, which are called homologous combination. The combination will be carried out at DL and CL, respectively. The test set of SVM could have been composed of two groups of sequences with obvious light differences, Test 1 and Test 2, respectively.

5.1.1. $V + V$. There are 6 different combinations of the 4 depth image descriptors V1, V2, V3, and V4 given in Table 3, which will be applied to DL and CL, respectively. The classification accuracy obtained in Test 1 and Test 2 is shown in Figures 7(a) and 7(b), respectively.

5.1.2. $D + D$. There are 21 different combinations of the 7 depth image descriptors D1, D2, D3, . . . , D7 given in Table 3, which will be applied to DL and CL, respectively. The classification accuracy obtained in Test 1 and Test 2 is shown in Figures 8(a) and 8(b), respectively.

Comparing Figure 7 with Figure 8, we find that the classification accuracy of $D + D$ is generally lower than $V + V$. The highest classification accuracy in Test 1 and Test 2 achieved by the best depth image descriptor D7 is 48.79% and 65.45%, respectively (while the highest classification accuracy in Test 1 and Test 2 achieved by the best visual image descriptor V4 is 74.76% and 85.78%, respectively). When the best initial descriptor D7 acts as the parent

descriptor, the highest classification accuracy of DL is 56.07% in Test 1, while it is 71.86% in Test 2. Apparently, the classification accuracy in Test 2 is still higher than that in Test 1 in $D + D$.

Similar to $V + V$, DL always outperforms CL in $D + D$. The classification accuracy of combination descriptors in DL is always higher than the parents' descriptors (39 out of 42), while only a few combination descriptors have higher classification accuracy than parents' descriptors in the CL (16 out of 42). The internal parameters of D7 are $S = 100$, $H = 2$, and $C = \text{horizontal}$. $D5 + D7$ (56.07%) achieves a favorable effect, and the internal parameters of D5 are $S = 100$, $H = 1$, and $C = \text{horizontal}$. $D2 + D7$ (71.86%) also achieves a favorable effect, and the internal parameters of D2 are $S = 50$, $H = 2$, and $C = \text{horizontal}$. The similarity of the optimal combination is $C = \text{horizontal}$, which is verified in Section 4. In addition, the internal parameters of V4 and D7 are $S = 100$, $H = 2$, and $C = \text{horizontal}$. So, we can speculate that high classification accuracy could be obtained by descriptors with such a group of internal parameters, which will be verified in Section 6.

5.2. Nonhomologous Combinations. This section will combine two descriptors extracted from different image types, namely, $V + D$, which is called as nonhomologous combination. There are 28 different combinations of V1, V2, V3, and V4 and D1, D2, D3, . . . , D7 in Table 3, which will be applied to DL and CL, respectively. The specific evaluation process is the same as homologous combination, and the evaluation results are shown in Figure 9.

In Test 2, the highest classification accuracy of CL and DL reaches 80.36% and 92.64%, respectively, while in Test 1, it reaches 72.84% and 81.76%. This is consistent with what

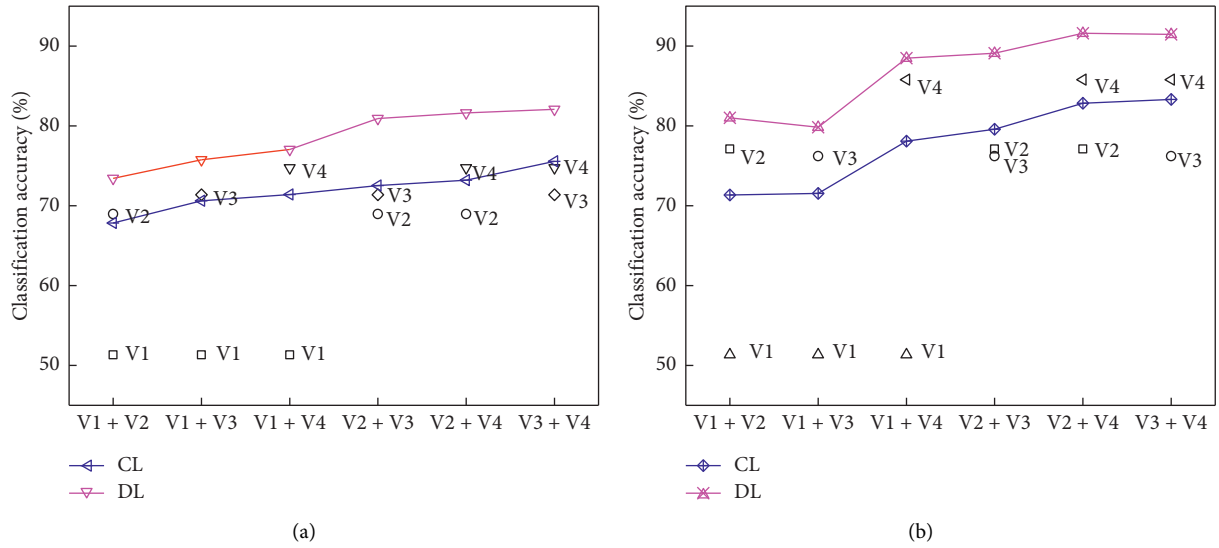


FIGURE 7: V + V combination evaluation results. (a) Test 1. (b) Test 2.

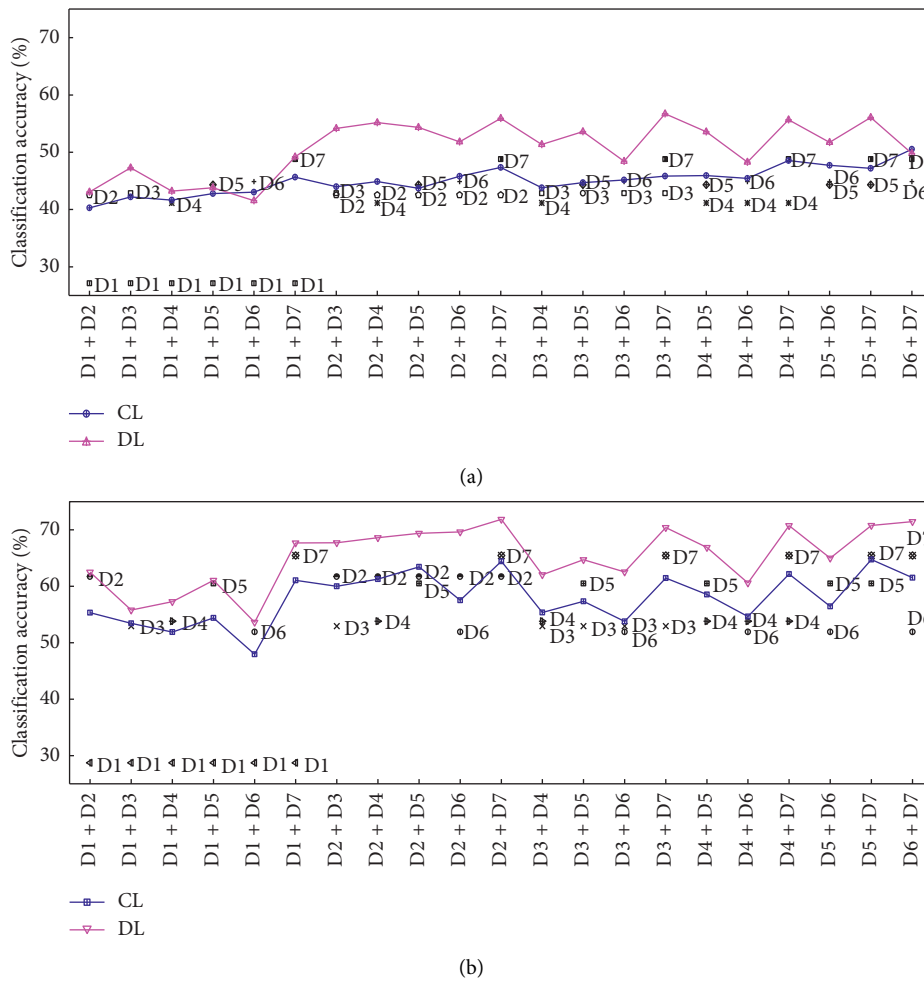


FIGURE 8: D + D combination evaluation results. (a) Test 1. (b) Test 2.

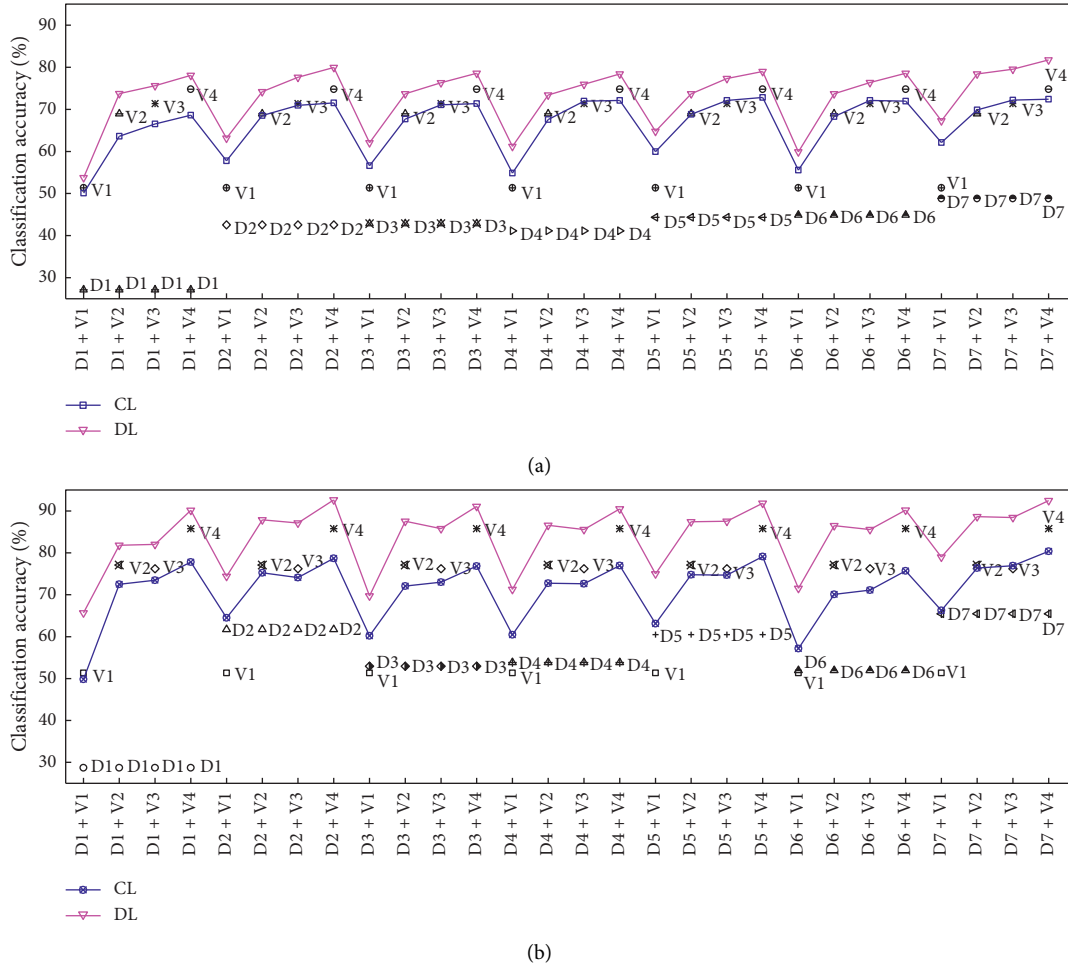


FIGURE 9: $D + V$ combination evaluation results. (a) Test 1. (b) Test 2.

we found before, the classification accuracy of Test 2 is always higher than Test 1, and DL always outperforms CL.

In CL, the combination with the highest classification accuracy is $D5 + V4$ (72.84%) in Test 1. In the meantime, the classification accuracy of $V4$, which acts a parent descriptor, is 74.76%. The combination with the highest classification accuracy is $D7 + V4$ (80.36%) in Test 2. The classification accuracy of $V4$, which acts as a parent descriptor, is 85.78%. As shown in Figures 9(a) and 9(b), only a few combination descriptors have higher classification accuracy than parent descriptors in the CL (18 out of 56), the same as in homologous combinations. It shows that the result of CL is not satisfactory.

In DL, the combination with the highest classification accuracy is $D7 + V4$ (81.76%) in Test 1. In the meantime, the classification accuracy of $V4$, which acts as a parent descriptor, is 74.76%. The combination with the highest classification accuracy is $D7 + V4$ (92.64%) in Test 2. The classification accuracy of $V4$, which acts as a parent descriptor, is 85.78%. As shown in Figures 9(a) and 9(b), the classification accuracy of combination descriptors in DL is always higher than that in parents' descriptors (56 out of 56).

We can conclude that DL outperforms CL in nonhomologous combination because most combination

descriptors in DL outperform their parent descriptor, while the combination descriptors in CL might be difficult to achieve. In addition, no matter in which level, the combinations of the descriptor with excellent performance and the descriptor with poor performance outperform other combinations. To cite some, $D1 + V4$ precedes $D1 + V1$, $D1 + V2$, and $D1 + V3$ in Figure 9(b).

Combining Figures 7–9, we can conclude that the overall effect of $V + V$ and $D + V$ outperforms $D + D$. Sometimes $V + V$ outperforms $D + V$ although nonhomologous combinations contain more comprehensive information. DL combines descriptors before entering a classifier, which may preserve characteristics of the descriptors completely. This may be the reason why DL is always better than CL. So, we only compare the evaluation results of $V + V$ and $V + D$ in DL.

Table 4 lists the best combinations of homologous and nonhomologous in DL, as well as the highest classification accuracy (bold data) obtained in Test 1 and Test 2. The best combination is $V3 + V4$ in Test 1, and the best combination is $D2 + V4$ in Test 2. We recall that the light variation in Test 1 is stronger than that in Test 2. So $V + V$ can be the best in Test 1, while $D + V$ can be the best in Test 2.

TABLE 4: The best combination.

	V + V	ζ (%)	D + V	ζ (%)
Test 1	V3 + V4	82.09	D7 + V4	81.76
Test 2	V2 + V4	91.60	D2 + V4	92.64

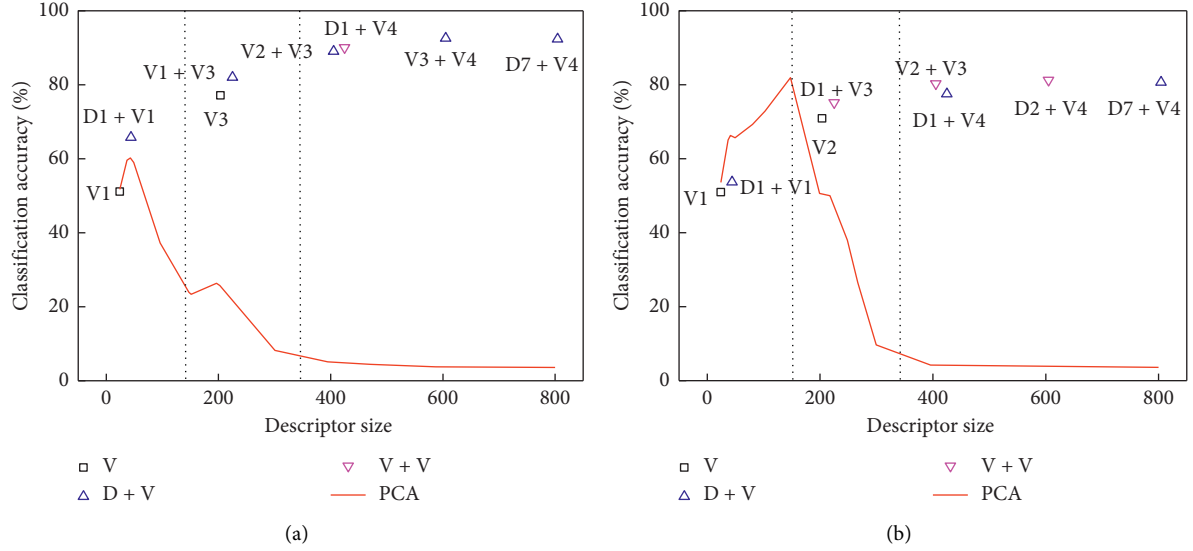


FIGURE 10: Descriptor size versus classification accuracy and PCA. (a) Test 1. (b) Test 2.

TABLE 5: Indoor scene classification execution time.

	Step	Parameters	Time (s)
Descriptor generation	Extracting D-SIFT feature	imageSize = 640 * 480	0.0840
		S = 20	0.0096
	Mapping feature point	S = 50	0.0140
		S = 100	0.0218
		H = 0	0.0006
	Counting histograms	H = 1, C = 1	0.0004
		H = 2, C = 1 or H = 1, C = 2	0.0003
H = 2, C = 2		0.0002	
Descriptor classification	Classifying the input descriptor	$\eta = 20$	0.0010
		$\eta = 50$	0.0016
		$\eta = 100$	0.0029
		$\eta = 200$	0.0062
		$\eta = 400$	0.0131
		$\eta = 800$	0.0291

As shown in Table 3, descriptor size has 8 possible values (including single descriptor or combination descriptor), respectively: 20, 40, 200, 220, 400, 420, 600, and 800. The maximum classification accuracy corresponding to each descriptor size value is compared with PCA results. Figure 10 shows the relationship between classification accuracy and descriptor size in Test 1 and Test 2. As we can see, the classification accuracy of the multiple descriptors fusion mechanism can be improved significantly with the descriptor size from small to middle. Also, the classification accuracy gradually tends to be stable with the descriptor size from middle to large. In Test 1, when descriptor size equals to 400 (large), V2 + V3 (80.94%)

gets the highest classification accuracy. In Test 2, when descriptor size equals to 600 (large), D2 + V4 (92.64%) gets the highest classification accuracy. PCA achieves high classification accuracy in the condition with small descriptor size. The superiority of the multiple descriptors fusion mechanism becomes obvious with the increasing descriptor size.

5.3. Execution Time. Indoor scene classification is divided into two stages: offline training and online testing. It is assumed that the construction of BoW and classifier training has been completed at the offline stage. Therefore, what affects the

TABLE 6: Comparison of classification accuracy.

Classification algorithm	ζ (%)
HOG + SVM [28]	77.2
SIFT + SPM [29]	84.2
SIFT + SURF [30]	85.7
Kernel Descriptors + Linear SVM [31]	89.6
Kernel Descriptors + Kernel SVM [31]	90.0
Kernel Descriptors + Random Forest [31]	90.1
Multiple descriptors fusion	92.6

running time of the online stage is the generation and classification of descriptors, including 4 steps, as shown in Table 5.

It is worth noting that step 1 adopts $\text{imageSize} = 640 * 480$. Step 2 is related to BoW size (S), so $S = 20, 50, \text{ and } 100$ are studied, respectively. Step 3 depends on the size and number of image cells, which is related to pyramid hierarchy (H) and cutting method (C). Step 4 is determined by η .

5.4. Algorithm Analysis and Comparison. Under the same database, the classification accuracy obtained by our mechanism will be compared with other fusion methods, as shown in Table 6. The classification accuracy obtained by the algorithms with single feature fusion [28–30] tends to be low for the indoor scene, largely because these algorithms do not filter descriptors. So it seems that the algorithm with single feature fusion is suitable for indoor scene classification. Higher classification accuracy is obtained by the algorithm with multiple features fusion [31], which extracted five different kernel descriptors from the images. After integration, they are trained and classified by Linear SVM, Kernel SVM, and Random Forest, respectively, and obtained 89.6%, 90.0%, and 90.1% accuracy in this experiment. 92.6% accuracy is achieved by our classification mechanism, which has a 2.5% higher value than in [31]. Above all, multiple descriptors fusion mechanism has good performance in indoor scene classification.

6. Conclusion

Aiming at the actual demands for indoor positioning applications, a multiple descriptors fusion model is established and an image classification strategy is proposed to improve the quality and efficiency of descriptors so as to achieve a better indoor scene classification effect. Firstly, the initial descriptor set is formed based on the established SPM. Then, the greedy descriptor filtering algorithm is adopted to select the descriptors with high weight in each descriptor size interval and a valuable descriptor set is obtained. Finally, the multiple descriptors combination algorithm is proposed to obtain high-quality and highly efficient multiple descriptors by combining homologous and nonhomologous images at DL and CL, respectively.

The generation, filtering, and combination of multiple descriptors proposed in this study improve the performance of the classifier. The evaluation results reflect that the multiple descriptors fusion mechanism proposed in this study outperforms the well-known PCA dimensionality

reduction technology, especially for the condition with medium or large descriptor size. This strategy not only achieves better results than other feature fusion algorithms, but also solved the limitations of existing scene classification algorithms applied to interior scenes.

Future research will focus on the improvement of the image feature extraction algorithm and the efficiency of constructing visual words by clustering features in the visual BoW model by other clustering algorithms. More attention will be paid to enhance the effectiveness of descriptors when describing image information. At the same time, the improvement of the quality of the depth image will be taken into account so as to make more efficient use of depth data in the process of descriptor filtering and descriptor combination. Alternatively, a more complete data set can be adopted.

Data Availability

The data results used to support the findings of this study are presented in this paper.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National High Technology Research and Development Program of China (2012AA120802), National Natural Science Foundation of China (61771186), Postdoctoral Research Project of Heilongjiang Province (LBH-Q15121), University Nursing Program for Young Scholars with Creative Talents in Heilongjiang Province (UNPYSCT-2017125), and Postgraduate Innovative Research Project of Heilongjiang University (YJSCX2019-166HLJU).

References

- [1] P. Nico, P. David, T. Jens et al., “TDoA-based outdoor positioning with tracking algorithm in a public LoRa network,” *Wireless Communications and Mobile Computing*, vol. 2018, 9 pages, 2018.
- [2] K. H. Lam, C. C. Cheung, W. C. Lee et al., “New RSSI-based LoRa localization algorithms for very noisy outdoor environment,” in *Proceedings of the IEEE 42nd Annual Computer Software and Applications Conference*, pp. 794–799, Tokyo, Japan, July 2018.
- [3] K. Zhang, S. Chong, Q. Zhou, H. Wang, Q. Gao, and Y. Chen, “A combined GPS UWB and MARG locationing algorithm for indoor and outdoor mixed scenario,” *Cluster Computing*, vol. 22, no. S3, pp. 5965–5974, 2019.
- [4] X. He, W. Manxing, L. Peng et al., “An RFID indoor positioning algorithm based on support vector regression,” *Sensors*, vol. 18, no. 5, pp. 1504–1519, 2018.
- [5] X. Yuan, Y. S. Shmaliy, Y. Li et al., “UWB-based indoor human localization with time-delayed data using EFIR filtering,” *IEEE Access*, vol. 5, pp. 16676–16683, 2017.
- [6] B. G. De, A. Quesada-Arencibia, C. R. Garcia, and J. C. Rodriguez, R. M. Diaz, A protocol-channelbased indoor

- positioning performance study for Bluetooth low energy," *IEEE Access*, vol. 6, pp. 33440–33450, 2018.
- [7] G. Xiang and Z. Tao, "Unsupervised learning to detect loops using deep neural networks for visual SLAM system," *Autonomous Robots*, vol. 41, no. 1, pp. 1–18, 2017.
 - [8] C. Yujin, C. Ruizhi, L. Mengyun, A. Xiao, D. Wu, and S. Zhao, "Indoor visual positioning aided by CNN-based image retrieval: training-free, 3D modeling-free," *Sensors*, vol. 18, no. 8, pp. 2692–2712, 2018.
 - [9] X. Aoran, C. Ruizhi, L. Deren, Y. Chen, and D. Wu, "An indoor positioning system based on static objects in large indoor scenes by using smartphone cameras," *Sensors*, vol. 18, no. 7, pp. 2229–2246, 2018.
 - [10] M. K. Alsmadi, "An efficient similarity measure for content based image retrieval using memetic algorithm," *Egyptian Journal of Basic and Applied Sciences*, vol. 4, no. 2, pp. 112–122, 2017.
 - [11] M. A. E. Aziz, A. A. Ewees, and A. E. Hassanien, "Multi-objective whale optimization algorithm for content-based image retrieval," *Multimedia Tools and Applications*, vol. 77, no. 19, pp. 26135–26172, 2018.
 - [12] L. Xie, F. Lee, L. Liu et al., "Improved spatial pyramid matching for scene recognition," *Pattern Recognition*, vol. 82, pp. 118–129, 2018.
 - [13] W. Zhao, H. Luo, J. Peng, and J. Fan, "Spatial pyramid deep hashing for large-scale image retrieval," *Neurocomputing*, vol. 243, pp. 166–173, 2017.
 - [14] L. Gupta, V. Pathangay, A. Patra et al., "Indoor versus outdoor scene classification using probabilistic neural network," *Eurasip Journal on Applied Signal Processing*, vol. 2007, Article ID 094298, no. 1, p. 123, 2007.
 - [15] L. T. L. Tao, Y. H. Kim, and Y. T. Kim, "An efficient neural network based indoor-outdoor scene classification algorithm," in *Proceedings of the International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, February 2010.
 - [16] L. Wang, S. Guo, W. Huang, Y. Xiong, and Y. Qiao, "Knowledge guided disambiguation for large-scale scene classification with multi-resolution CNNs," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 2055–2068, 2017.
 - [17] H. Kebapci, B. Yanikoglu, and G. Unal, "Plant image retrieval using color, shape and texture features," *The Computer Journal*, vol. 54, no. 9, pp. 1475–1490, 2011.
 - [18] J. K. Patil and R. Kumar, "Analysis of content based image retrieval for plant leaf diseases using color, shape and texture features," *Engineering in Agriculture, Environment and Food*, vol. 10, no. 2, pp. 69–78, 2017.
 - [19] A. Raza, T. Nawaz, H. Dawood, and H. Dawood, "Square texton histogram features for image retrieval," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 2719–2746, 2019.
 - [20] W. Tahir, A. Majeed, and T. Rehman, "Indoor/outdoor image classification using GIST image features and neural network classifiers," in *Proceedings of the International Conference on High-Capacity Optical Networks & Enabling/emerging Technologies (HONET)*, Islamabad, Pakistan, December 2015.
 - [21] L. Ju, K. Xie, H. Zheng, B. Zhang, and W. Yang, "GPCA-SIFT: a new local feature descriptor for scene image classification," *Communications in Computer and Information Science*, vol. 663, no. 4, pp. 286–295, 2016.
 - [22] J. Shi, H. Zhu, J. Wang et al., "Indoor scene classification algorithm based on information enhancement of vision sensitive area," *Moshi Shibie Yu Rengong Zhineng/Pattern Recognition and Artificial Intelligence*, vol. 30, no. 6, pp. 520–529, 2017.
 - [23] D. Lin, S. Fidler, and R. Urtasun, "Holistic scene understanding for 3D object detection with RGBD cameras," in *2013 Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1417–1424, Sydney, Australia, March 2014.
 - [24] Y. Zheng, J. Pu, H. Wang et al., "Indoor scene classification by incorporating predicted depth descriptor," *Pacific Rim Conference on Multimedia*, vol. 10736, pp. 13–23, May 2018.
 - [25] V. Bisot, S. Essid, and G. Richard, "HOG and subband power distribution image features for acoustic scene classification," in *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*, pp. 719–723, Nice, France, December 2015.
 - [26] A. Janoch, S. Karayev, Y. Jia et al., "A category-level 3D object dataset: putting the Kinect to work," in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1168–1174, Barcelona, Spain, November 2011.
 - [27] N. Silberman, D. Hoiem, P. Kohli et al., "Indoor segmentation and support inference from RGBD images," in *Proceedings of the 12th European conference on Computer Vision (ECCV)*, pp. 746–760, Springer, Berlin, Germany, October 2012.
 - [28] L. Jin, L. Quan, and A. Qingsong, "Research of image classification based on fusion of SURF and global feature," *Computer Engineering & Applications*, vol. 49, no. 17, pp. 174–177, 2012.
 - [29] R. Rani, S. Kumar Grewal, and K. Panwar, "Object recognition: performance evaluation using SIFT and SURF," *International Journal of Computer Applications*, vol. 75, no. 3, pp. 39–47, 2013.
 - [30] L. Bo, X. Ren, and D. Fox, "Depth kernel descriptors for object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems*, pp. 821–826, San Francisco, CA, USA, December 2011.
 - [31] K. Bregar, M. Mohorcic, and Y. Yang, "Improving indoor localization using convolutional neural networks on computationally restricted devices," *IEEE Access*, vol. 6, pp. 17429–17441, 2018.
 - [32] Y. Zhou, Y. Zhou, Q. Liu et al., "Research on a DSIFT algorithm applicable to image mosaicking," *Journal of Xian Jiaotong University*, vol. 49, no. 9, pp. 84–90, 2015.
 - [33] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikäinen, "From BoW to CNN: two decades of texture representation for texture classification," *International Journal of Computer Vision*, vol. 127, no. 1, pp. 74–109, 2019.
 - [34] L. Lifeng, M. Yan, Z. Xiangfen, Y. Zhang, and S. Li, "High discriminative SIFT feature and feature pair selection to improve the bag of visual words model," *Iet Image Processing*, vol. 11, no. 11, pp. 994–1001, 2017.
 - [35] Q. Zhu, Y. Zhong, B. Zhao, G.-S. Xia, and L. Zhang, "Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 6, pp. 747–751, 2016.
 - [36] S. Khanmohammadi, N. Adibeig, and S. Shanehbandy, "An improved overlapping k-means clustering method for medical applications," *Expert Systems with Applications*, vol. 67, pp. 12–18, 2017.
 - [37] E. Lee, M. Schmidt, and J. Wright, "Improved and simplified inapproximability for k-means," *Information Processing Letters*, vol. 120, pp. 40–43, 2017.
 - [38] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA, October 2006.

Research Article

The Real-Time Mobile Application for Classifying of Endangered Parrot Species Using the CNN Models Based on Transfer Learning

Daegyoo Choe ¹, Eunjeong Choi ¹, and Dong Keun Kim ²

¹Department of Computer Science, Graduate School of Sangmyung University, Seoul, Republic of Korea

²Department of Intelligent Engineering Information for Human, and Institute of Intelligent Informatics Technology, Sangmyung University, Seoul, Republic of Korea

Correspondence should be addressed to Dong Keun Kim; dkim@smu.ac.kr

Received 11 October 2019; Accepted 7 January 2020; Published 9 March 2020

Guest Editor: Malik Jahan Khan

Copyright © 2020 Daegyoo Choe et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Among the many deep learning methods, the convolutional neural network (CNN) model has an excellent performance in image recognition. Research on identifying and classifying image datasets using CNN is ongoing. Animal species recognition and classification with CNN is expected to be helpful for various applications. However, sophisticated feature recognition is essential to classify quasi-species with similar features, such as the quasi-species of parrots that have a high color similarity. The purpose of this study is to develop a vision-based mobile application to classify endangered parrot species using an advanced CNN model based on transfer learning (some parrots have quite similar colors and shapes). We acquired the images in two ways: collecting them directly from the Seoul Grand Park Zoo and crawling them using the Google search. Subsequently, we have built advanced CNN models with transfer learning and trained them using the data. Next, we converted one of the fully trained models into a file for execution on mobile devices and created the Android package files. The accuracy was measured for each of the eight CNN models. The overall accuracy for the camera of the mobile device was 94.125%. For certain species, the accuracy of recognition was 100%, with the required time of only 455 ms. Our approach helps to recognize the species in real time using the camera of the mobile device. Applications will be helpful for the prevention of smuggling of endangered species in the customs clearance area.

1. Introduction

With the development of information technology, deep learning-based image processing and classification is widely used in various applications [1]. In particular, the demand for image classification is increasing [2]. Deep learning-based classifiers, such as a convolutional neural network (CNN), increase the classification performance for various objects [2]. A common task in image processing is identifying similar types of objects with machine learning methods to classify and cluster animals [3]. Systems that automatically identify and classify animal species have become essential, particularly for the study of endangered species [4]. During the customs clearance of animals and plants, humans can directly examine the species to identify individual species, but this can be inefficient in terms of time and cost.

To improve the efficiency, automated classification of species can be conducted on mobile devices. However, this would require solving the problems of classifying species with similar shades of colors and shapes. Hence, custom machine learning models are needed to classify endangered species and address the complicated characteristics of animal images for specific applications.

Although various machine learning models can classify images of different animals, it remains a challenge to distinguish animal species. This is because there are some species with a high color similarity. It is a complicated process that requires expertise even for human beings. The CNN models are efficient modern recognition methods. Unlike the traditional image classification methods [5], a convolutional neural network uses multilayer convolution to automatically extract and combine features. These

algorithms are designed to be performed independently and are trained to solve specific tasks. Moreover, the neural network models have to be rebuilt once the feature-space distribution changes. To overcome these disadvantages, we adopted the transfer learning method to classify the endangered parrot quasi-species in this study. Transfer learning is a machine learning technique in which a model trained for one task is reused for another related task [6]. Among many ways to deploy deep learning models in production, one of the easiest ways is to deploy it on mobile devices. The advantages are that mobile devices are popular and easy to use. Users can get an answer in a few touches. Moreover, deep learning models can receive large amounts of data in real time thanks to the camera of the mobile device. When deploying a deep learning model on a mobile device, two aspects should be considered: model file size and process speed. If the size is too large, it is impossible to deploy the model on a mobile device. If the process is slow, it will cause inconvenience for the users.

In this study, a real-time mobile application was developed to classify endangered parrot quasi-species using the CNN models based on transfer learning. To clarify the purpose of this study, we suggested the following hypotheses:

- (i) The designed CNN-based transfer learning models can classify endangered parrot quasi-species with high color similarity
- (ii) The developed application can embed the designed CNN-based training

The rest of this paper is organized as follows. Section 2 presents related work on transfer learning with CNN models. Section 3 explains our real-time mobile application. Section 4 presents the experimental results of the classification of endangered parrot species for the designed mobile application. Section 5 discusses the contribution of the designed mobile application and the classification results. Finally, Section 5 concludes this study.

2. Related Work

2.1. CNN Models and Image Classification for Animals. Many well-known CNN model architectures exist for various applications. In 2016, Microsoft Research presented a solution for the problem of building deep models with shortcut connections [7]. Zoph and Le also presented a method to automatically find a new, optimized model architecture based on policy gradients called neural architecture search at ICLR 2017 [8]. Szegedy et al. have won the ILSVRC 2014 with a top-5 test error of 6.7% with a model built on the concept of “network in network.” The idea of this model is to reduce the computing cost using dimensionality reduction, constructing the network by stacking convolution operations, using filters of various sizes, and then combining them later [9]. Another model created by Szegedy et al. is Inception-ResNet, which combines the residual connections presented by Microsoft Research [10].

Many relevant studies exist to preserve the diversity of species. To acquire the data necessary for these studies,

unmanned cameras were installed to acquire images of the creatures. However, human resources are wasted on processing the obtained data. Because human’s judgment is subjective, the accuracy is inevitably deteriorated. Therefore, it is essential to create a system that automatically processes and classifies animal images. Norouzzadeh et al., in the “Snapshot Serengeti Project,” said that processing of information from animal image datasets by human beings is time-consuming; hence, much data remains unprocessed. They presented a system in which a machine can determine where the images belong to and check the number of entities and their behaviors in images [3]. Nguyen et al. also created a CNN model to classify three of the most commonly observed animal species in Victoria, Australia, and showed the real test results [11]. Zhuang et al. introduced a deep learning model that automatically annotates marine biological image data without relying on human experts. They experimented with their model with data from SeaCLEF2017 [12]. In this study, we also propose a system to classify image data acquired in real time using the camera of a mobile device.

2.2. Transfer Learning. Transfer learning is a state-of-the-art technique in deep learning research. Before the advent of this technique, people had to create and train a model from scratch. It was difficult to invent a model with remarkable performance on a specific task because of the lack of computing infrastructure. Moreover, it was impossible to collect enough meaningful data required to train a model, although many researchers attempted to gather them. However, various transfer learning methods have been proposed for transferring knowledge in the context of features, instant weights, parameters, or relationship information between data samples in a domain [13–16].

Figure 1 shows four steps of creating a complete model using transfer learning. First, we build an architecture of the model and train it on a large representative dataset. Second, we delete the final layer (known as “loss output”). Third, we replace it with another layer whose job is to finish the specific task. Fourth, we train a new model with a relatively small dataset suitable for the purpose. Transfer learning is literally to transfer the job of extracting features from data to the pre-trained model. For example, a model pretrained on the ImageNet dataset can detect low-level features on a bird image (such as curves, outlines, and lines) because these low-level features are almost the same in other animal images. The remaining task is to tune the high-level layers of the feature extractor and the final layer that classifies the bird (the process is called fine tuning). Some studies have already applied transfer learning [17, 18]. Transfer learning is expected to compensate for the lack of data, time, and computing.

3. Implementation of a Real-Time Mobile Application to Classify Endangered Parrot Quasi-Species

3.1. System Design and Image Classification in Mobile Devices. The system is divided into four parts, as shown in Figure 2. First, we preprocess the data to prepare it for deep learning.

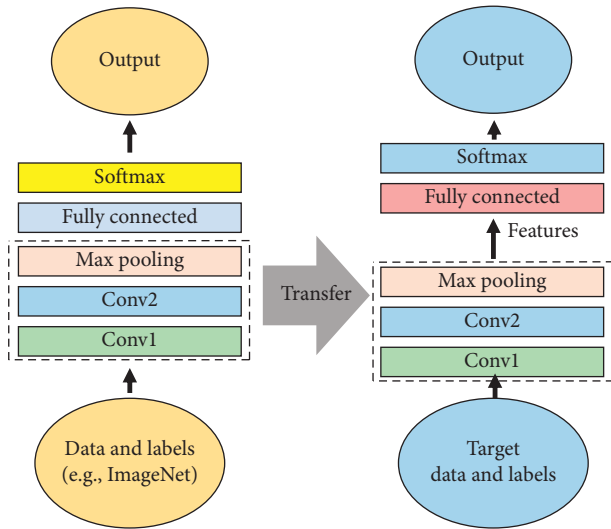


FIGURE 1: Diagram of the transfer learning.

Second, we create and train a classifier using the pre-processed data. Third, we convert the generated model into a file that can be deployed on a mobile device. Finally, we deploy the model. In this section, we describe data pre-processing and the process of creating and training the deep learning model.

In our study, we used Python (Anaconda) for the third step and Android Studio for the final step. Data were pre-processed using a Python library called “imgaug” [19] that provides image preprocessing methods (“ImageTransformation,” “AdditiveGaussianNoise,” “CoarseDropout,” “BilateralBlur,” etc.). We imported the “imgaug” library into our project in the Anaconda Jupyter notebook environment and performed data augmentation for the original images. The obtained images were saved in the folders together with the original images.

To develop an application, TensorFlow Lite provides a method that converts the generated model into a TensorFlow Lite FlatBuffer format file (.tflite), which can be deployed on a mobile device. According to the official TensorFlow Lite website, FlatBuffer is an open-source cross-platform serialization library that serializes data efficiently. TensorFlow Lite supports the conversion of files created by TensorFlow, concrete functions, and Keras [20]. We inserted this converted file into the demo project provided by TensorFlow Lite and then built the project. After this step, we created an Android package file (APK) and installed the application on a device. Figure 3 shows the overall process. Li et al. developed an optimized modeling technique for mobile devices using their reduction module, group convolution, and self-attention module. They claimed that this model was efficient for mobile applications compared with other models [21]. Subsequently, we explain how to deploy a CNN model created by TensorFlow Lite on a mobile device.

We use the Keras library to create and train deep learning models. Keras is a high-level open-source neural network API written in Python. It was developed as a part of the Open-Ended Neuro-Electronic Intelligent Robot Operating System (ONEIROS) project. A model produced by

Keras is built using a fast and intuitive interface based on TensorFlow, CNTK, and Theano [22]. In the field of computer vision, some model architectures that can effectively classify images have been previously introduced, and Keras provides them as open-source code [23]. In this study, we propose a way to customize these models, train them, and verify their performance.

3.2. Data Augmentation. One of the biggest limitations in deep learning model development is that it requires a large dataset. Thousands, millions, or even more data samples are required to create a reliable deep learning model. These limitations can be overcome by manipulating and transforming a small amount of data. This is called data augmentation. Data augmentation techniques have been used in many studies [24, 25]. The techniques include random cropping, horizontal flipping, brightness modification, and contrast modification. As illustrated in Figure 4, we extended the dataset by the horizontal and vertical flipping. Figure 4 shows the extended dataset as a result of four parrot species’ data augmentation. For this task, we imported “imgaug” Python library (as explained in Section 3.1). It contains the “Sequential” method, and manipulation techniques can be set as the parameters of this method [19]. In this study, because we only wanted to augment the images by the horizontal and vertical flipping, to check if the model can classify the quasi-species of parrots with a high color similarity, we inserted “Fliplr” and “Flipud” objects. Finally, 14,000 images including the original data were gathered (see the details in Section 3.5).

3.3. Feature Extraction and the CNN Model. Nguyen et al. set the two experimental scenarios on the model architectures of Lite AlexNet, VGG-16, and ResNet50 to classify wildlife images [11]. The first scenario was to train the model from scratch, and the second one was to use a technique called “feature extraction” that imports weights that had been pretrained on large images in ImageNet. To monitor and classify enormous animal image data, some pretraining techniques are needed to familiarize the model with extracting local features of a new image. Feature extraction solves the problem. It customizes the top layer of a model (fully connected layer) and lets the pretrained CNN extract the characteristics of the image. For our study, we used the feature extraction technique; we validated its performance by comparing it with the model with randomly initialized weights. The first model was generated with the pretrained weights in ImageNet. Our purpose was to verify if the model can capture the local differences of two species which are very similar such as “*Cacatua galerita*” and “*Cacatua goffiniana*.”

According to Lin et al., the fully connected layer commonly used in traditional CNN models is likely to overfit despite using the dropout. They proposed a global average pooling (GAP) technique that inserts the average value of each feature map into a vector and links it into the input of the SoftMax layer directly instead of a fully connected layer [26]:

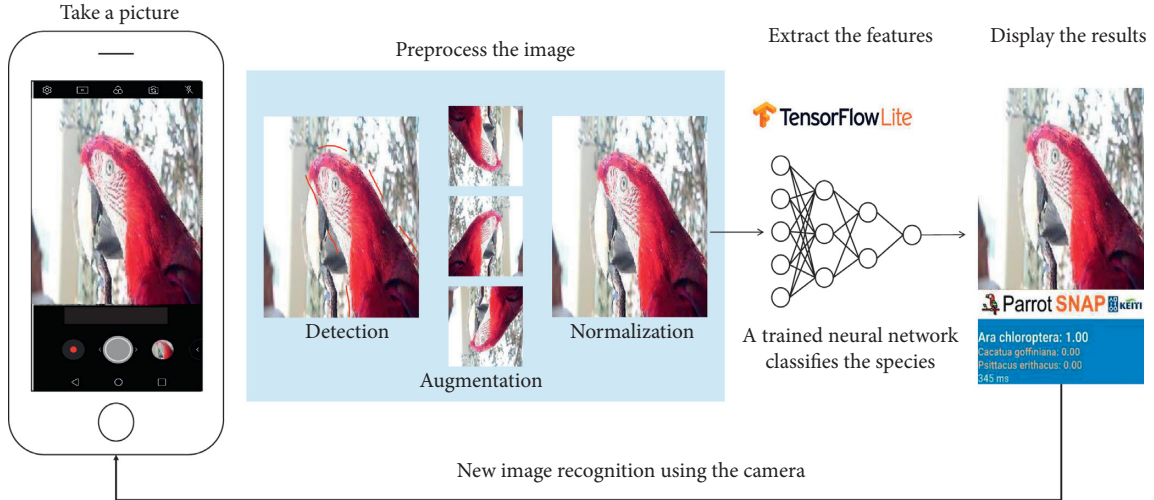


FIGURE 2: System configuration and scenario for classifying endangered parrot species using a mobile device.

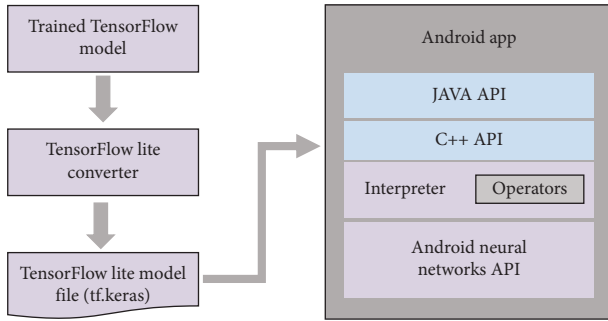


FIGURE 3: TensorFlow Lite conversion process graph.

$$GAP_i = \frac{1}{m \times n} \sum_a^m \sum_b^n x. \quad (1)$$

Formula (1) presents the approach suggested in their study. GAP is a vector of the average values of feature maps from the last convolutional layer. GAP_i indicates an element of the vector. Here, m is the number of rows in a feature map and n is the number of columns in a feature map. The meaning of the left term is summing all values in the feature map and then dividing them by m multiplied by n . The purpose is to obtain the average value of the feature map. GAP calculates averages of feature maps that are the outcomes of the convolutional process (Figure 5). Next, it creates a vector that consists of the average values.

According to their proposal, GAP has the following advantages over a fully connected layer. First, the computational cost can be reduced by decreasing the number of parameters to be handled by a human (hyperparameters). Second, some model parameters can be eliminated to reduce overfitting. Therefore, there is no need to rely on dropout. In this study, we will use GAP instead of a traditional fully connected layer to take advantage of this technique.

We imported the ResNet50, NASNetMobile, InceptionResNetV2, and InceptionV3 models from the Keras library for feature extraction. The imported models used

convolutional layers initialized with weights that had been pretrained on ImageNet. A global average pooling layer and a dense layer with SoftMax were added after the convolutional layers (Figure 6). The experiment compared two types of initialization: weights of ImageNet and random values. Moreover, we use a hyperparameter search library called “Hyperas” to optimize hyperparameters (such as optimizer and learning rate) without the researcher’s effort.

3.4. Transfer Learning. As explained in Section 2, we can apply the convolutional layers of a pretrained model to another classifier. Because an image consists of pixels, the local features of the image are almost the same as in other images. The convolutional layers can capture these patterns using the pretrained weights. At this point, the model’s ability to perform the abstraction of local parts affects the model’s performance. According to Krizhesky et al., the test results for the models with transfer learning showed that their top-5 accuracy was higher than in other cases [27]. Transfer learning does not train the convolutional layers but only lets them extract the features and then passes the extracted features to the classification layers. Moreover, there is an advanced technique to improve the model (called fine tuning) that trains the high-level layers of the convolutional layers and the classification layer together. In our study, we experimented with the models described in Section 3.3 (ResNet50, NASNetMobile, InceptionResNetV2, and InceptionV3) trained by transfer learning using the weights of ImageNet (Figure 7).

3.5. Experiments. Parrots are among the most common endangered species in South Korea because of social problems such as smuggling. Moreover, parrots are included in the list of the most endangered species by the Convention on International Trade in Endangered Species of Wild Flora and Fauna (CITES) (Table 1). We have previously studied parrots of distinct colors and shapes with conventional CNN models [28]. However, in this study, we hypothesize that the

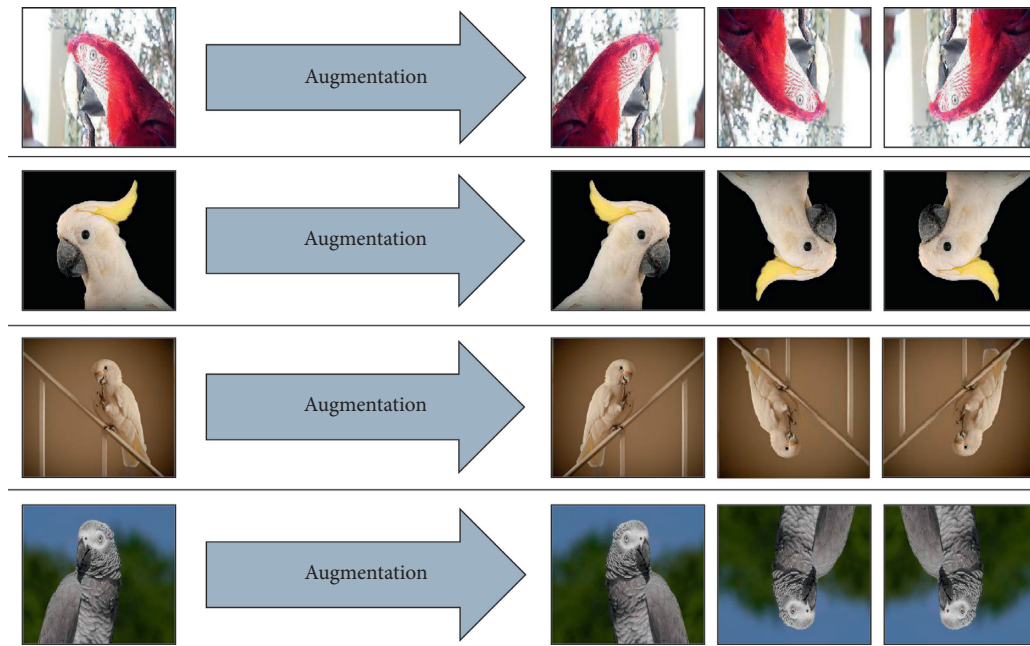


FIGURE 4: Data augmentation for images of endangered parrot species.

CNN models with transfer learning can classify the quasi-species well despite similar colors and patterns. This experiment used 14420 parrot images. The parrots were of four species, and we used 3605 images per species. As shown in Table 1, the four parrot species are *Cacatua goffiniana*, *Cacatua galerita*, *Ara chloroptera*, and *Psittacus erithacus*. Among these species, *Cacatua goffiniana* and *Cacatua galerita* have a high color similarity. Morphological information is very important to classify the parrot images using CNN. The morphological features of each species are shown in Table 1 [29]. Parrot images were divided into three subsets: training, validation, and test sets. They were crawled from Google and YouTube. There were 980 images per species originally, but we divided these into two groups and use only 875 for training because of the information leak. 3500 images were produced by data augmentation. 2800 images were for training and 700 images were for validation. The test set has 420 images, including 100 crawled images and 5 images provided by the Seoul Grand Park per each species. Because we focused on the color similarity of two species, we did not do any data augmentation affecting the color of images. Thus, 2800 images for the training set and 700 images for the validation set were provided to the models for each species. The test set did not undergo the process of data augmentation because it is not effective to use the augmented data not affecting the color for the actual test. The testing is divided into two steps. After the training, we carried out the test of each model’s performance by comparing the confusion matrix and F1-score values for 420 test samples. Next, we converted the file into a FlatBuffer format, deployed it on a mobile device, and then verified the results by using the video data obtained from the Seoul Grand Park.

Figure 8 depicts the entire experiment process. Original data were augmented using the “imgaug” library, as

described in Section 3.2. The image classifier was created using the Keras API in TensorFlow, a powerful tool to construct a deep learning model. We focused on a pretrained model for transfer learning; hence, we imported the models as shown in Figure 7. For example, “tensorflow.keras.applications.resnet.ResNet50” can set the weights initialization type [30]. We can obtain the desired results by setting the keyword parameter “weights” to “imagenet.” The models were completed with stacking a GAP layer and a dense layer. Once the models’ training was complete, we evaluated their performance with the test data using *t* “scikit-learn” Python library [31, 32]. Next, we converted it into a “FlatBuffer” file to be deployed on a mobile device [33]. Finally, we can see the result on a device, as illustrated in Figure 9.

4. Results

4.1. Experimental Results. Figure 10 shows the learning curves of training accuracy for eight models: ResNet50, NASNetMobile, InceptionResNetV2, and InceptionV3 with two types of initialization: pretrained ImageNet weights or random numbers (as described in the previous section). The horizontal axis shows the number of training iterations on the complete train dataset. The vertical axis shows the training accuracy (0.5 means the model correctly classified half of the data, and 1 means a perfect classification). As depicted in Figure 10, performance of the models was poor after the first epoch, but additional iterations improved the accuracy. After approximately twenty epochs, the accuracy of each model converged at 1, with no noticeable improvement afterward. Notably, the models that were initialized with the ImageNet weights and had nontrainable convolutional layers outperformed the others (we can check that the curves are located higher). Besides, their accuracy

[Featuremap calculation using Relu]

$$\text{Featuremap}_{a,b,c} = \max((\text{Weight}_c)^T * x_{a,b}, 0)$$

where (a, b) is a pixel index and c indicates a index of the channels

[Global Average Pooling]

$$\text{GAP}_i = \frac{1}{m \times n} (\sum_a^m \sum_b^n x_{a,b})$$

where m is the number of rows, n is the number of columns in a featuremap

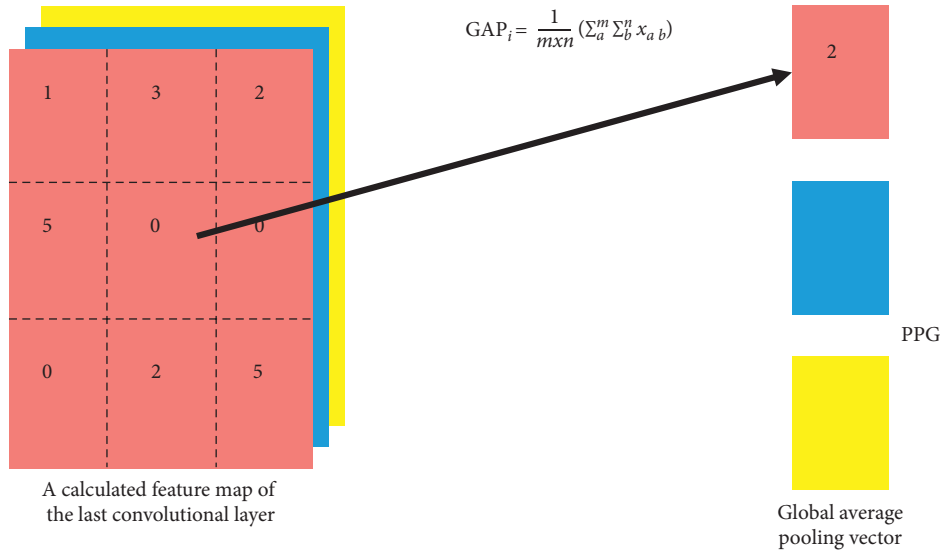


FIGURE 5: Concept diagram of global average pooling.

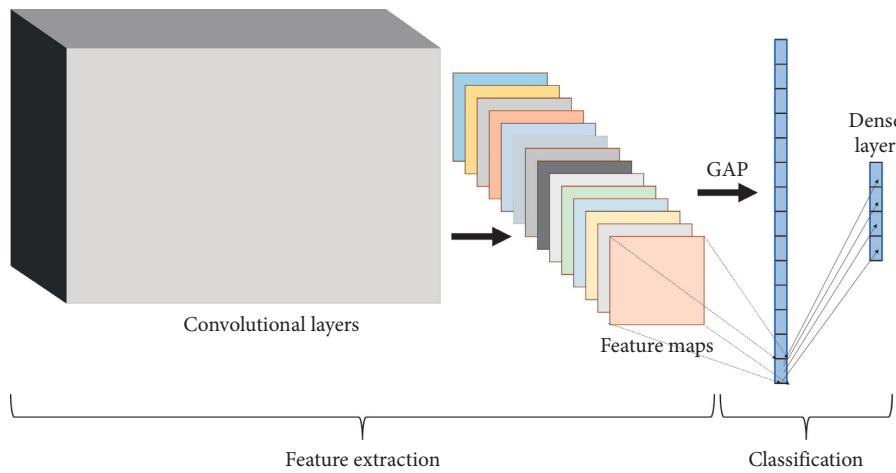


FIGURE 6: Convolutional layers and feature maps for feature extraction of endangered parrot species.

converged faster. Figure 11 illustrates the learning curves of validation accuracy for the models. The models were evaluated on the validation data after each epoch. Therefore, the accuracy measures the quality of predictions for the validation data. The curves look relatively uneven compared with the prior ones. This is because the models had never seen these data before. The models learned some features of parrots using the training images, and we tested what they learned using the validation data. The models experienced some

failures repeatedly. However, their accuracy converged to a point of minimal error. Likewise, the accuracy of ImageNet-initialized models is typically better than the others. Both graphs do not show any obvious drop as time passes (look at both graphs after twenty epochs). Thus, overfitting did not occur. Overfitting refers to the models that perform well on the training set but not on the validation set.

The reason why epoch number is thirty is because we checked that it is useless to exceed thirty. We set some

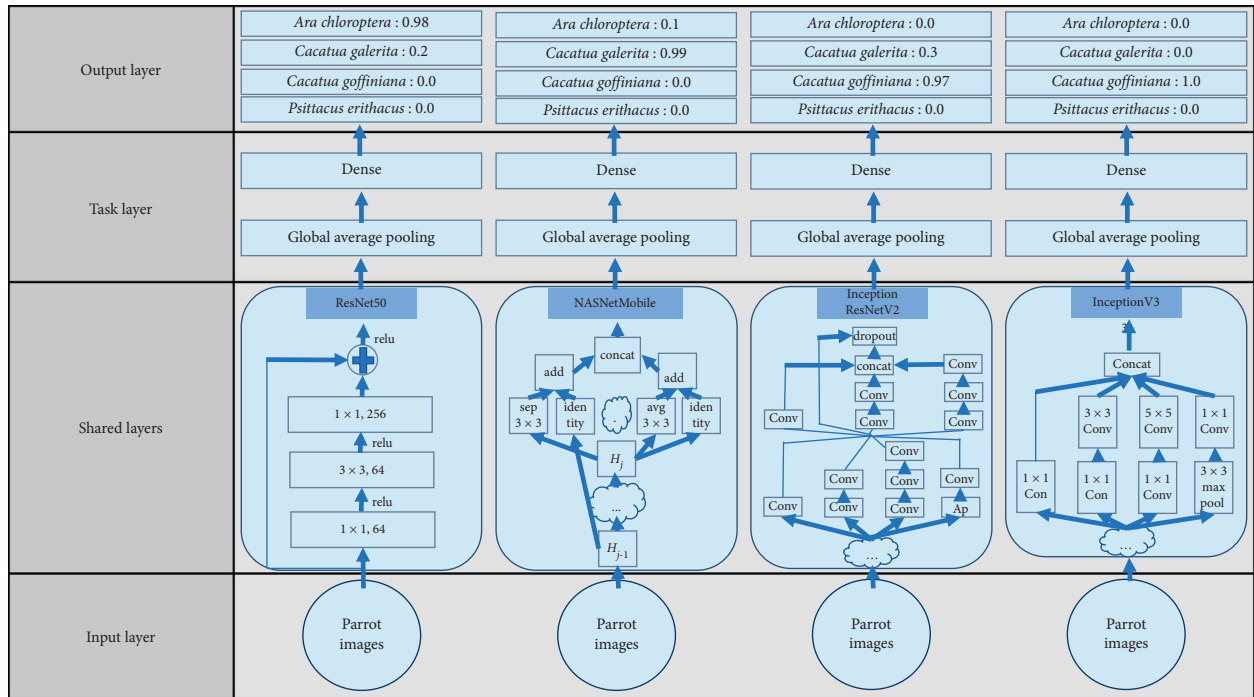






FIGURE 7: Convolutional layers and feature maps for feature extraction of endangered parrot species.

TABLE 1: Examples of four endangered parrot species.

Picture				
Name	Red and green macaw	Sulphur-crested cockatoo	Goffin's cockatoo	Gray parrot
Scientific name	<i>Ara chloroptera</i>	<i>Cacatua galerita</i>	<i>Cacatua goffiniana</i>	<i>Psittacus erithacus</i>
Appearance	Flight feathers, back, rump: darker red Tail-coverts: blue Median wing-coverts, scapulars, tertials: green Tail: dark red tipped blue Bare face with conspicuous lines of red feathers	Little yellow on ear-coverts or bases to feathers of head and underparts	Short, blunt bill Lores and bases to feathers of head salmon-pink: palest blue Almost white eye-ring	Gray parrot with short, squarish red tail
Cites appendices	Appendix II	Appendix II	Appendix I	Appendix I

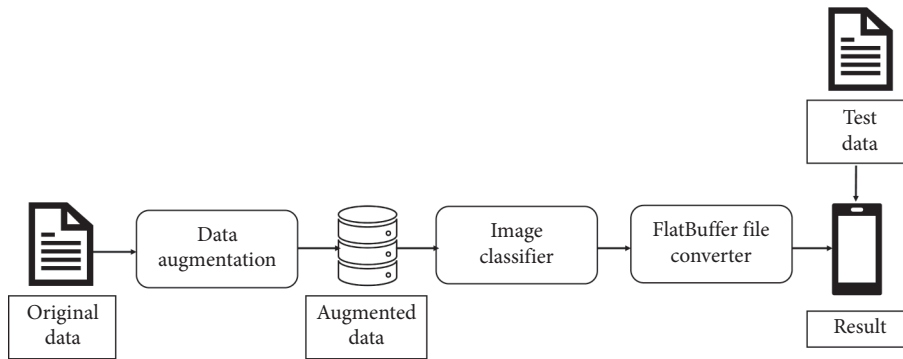


FIGURE 8: Overall experiment process.



FIGURE 9: Graphical user interface example of the designed system.

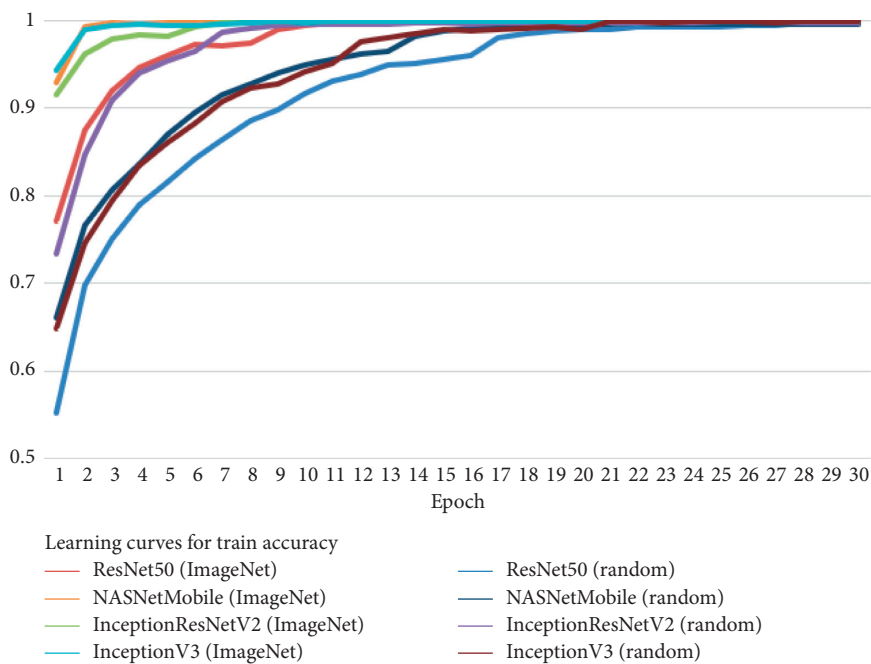


FIGURE 10: Learning curves of each model's train accuracy.

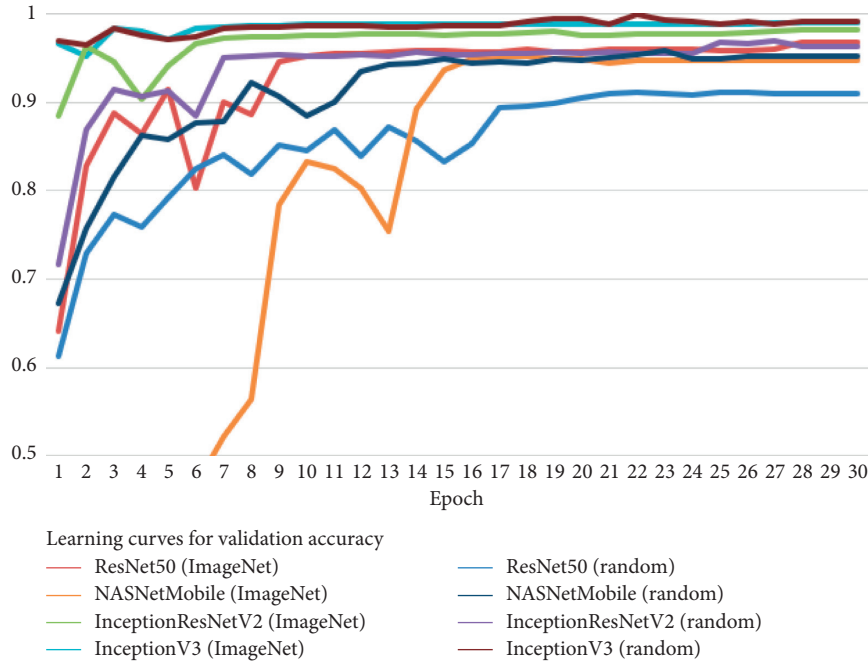


FIGURE 11: Learning curves of each model’s validation accuracy.

TABLE 2: Confusion matrix.

ResNet50 (ImageNet/random)		Prediction			
Actual	<i>Ara chloroptera</i>	<i>Cacatua galerita</i>	<i>Cacatua goffiniana</i>	<i>Psittacus erithacus</i>	
<i>Ara chloroptera</i>	100/92	0/1	0/3	5/10	
<i>Cacatua galerita</i>	0/1	98/66	6/38	1/0	
<i>Cacatua goffiniana</i>	0/2	15/40	88/55	2/8	
<i>Psittacus erithacus</i>	5/4	1/5	10/6	89/90	
NASNetMobile (ImageNet/random)		Prediction			
Actual	<i>Ara chloroptera</i>	<i>Cacatua galerita</i>	<i>Cacatua goffiniana</i>	<i>Psittacus erithacus</i>	
<i>Ara chloroptera</i>	99/95	0/1	5/1	1/8	
<i>Cacatua galerita</i>	0/0	100/76	2/23	3/6	
<i>Cacatua goffiniana</i>	0/3	12/32	89/54	4/16	
<i>Psittacus erithacus</i>	3/0	0/4	1/11	101/90	
InceptionResNetV2 (ImageNet/random)		Prediction			
Actual	<i>Ara chloroptera</i>	<i>Cacatua galerita</i>	<i>Cacatua goffiniana</i>	<i>Psittacus erithacus</i>	
<i>Ara chloroptera</i>	103/85	0/7	1/2	1/11	
<i>Cacatua galerita</i>	0/0	98/74	5/29	2/2	
<i>Cacatua goffiniana</i>	0/4	8/19	95/72	2/10	
<i>Psittacus erithacus</i>	8/5	0/4	1/4	96/92	
InceptionV3 (ImageNet/random)		Prediction			
Actual	<i>Ara chloroptera</i>	<i>Cacatua galerita</i>	<i>Cacatua goffiniana</i>	<i>Psittacus erithacus</i>	
<i>Ara chloroptera</i>	100/99	0/1	3/1	2/4	
<i>Cacatua galerita</i>	0/0	94/77	3/28	8/0	
<i>Cacatua goffiniana</i>	1/2	8/10	97/89	0/4	
<i>Psittacus erithacus</i>	8/6	0/1	0/2	97/96	

callback functions when we called the “model.fit()” in our experiment, “EarlyStopping()” and “ReduceLROnPlateau()”. It would have been stopped if the validation accuracy had not been improved during five epochs. We saw that the training epoch never exceeded twenty-five, so we set the number of epochs to thirty. Learning rate started from 0.001 and decreased gradually by 0.03 if the validation accuracy had not been improved during three epochs until the

termination of training. When we called “model.compile()”, we set loss equals to “categorical_crossentropy”, metrics equals to “acc”, and optimizer equals to “Adam.”

Table 2 shows the confusion matrix for all models. A confusion matrix is an evaluation approach that checks the performance of a classifier for all labels. Every model in this study is included, and each row shows the performance of the model depending on the labels. For instance,

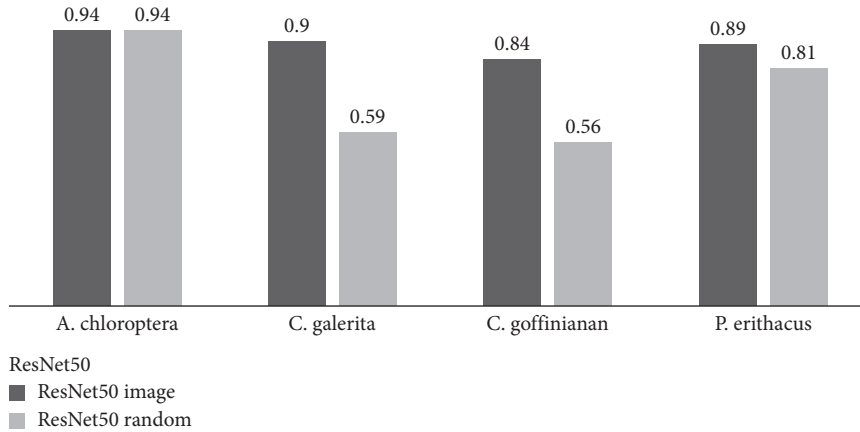


FIGURE 12: F1-score of RestNet50 for four different endangered parrot images.

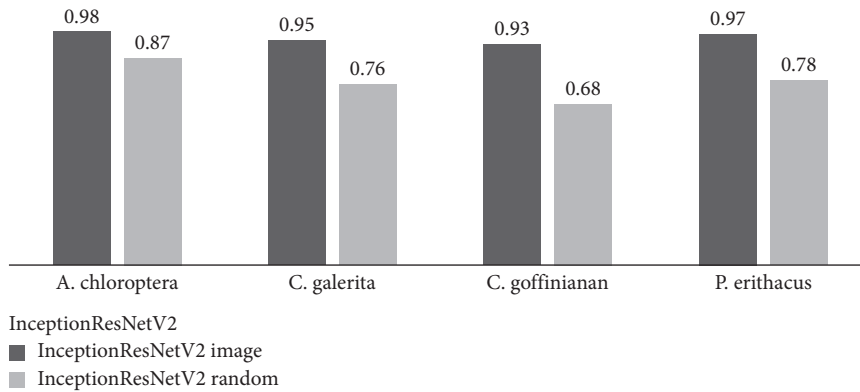


FIGURE 13: F1-score of InceptionResNetV2 for four different endangered parrot images.

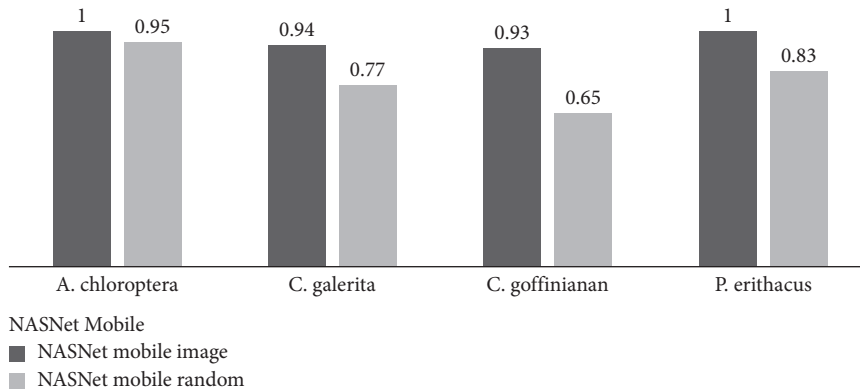


FIGURE 14: F1-score of NASNetMobile for four different endangered parrot images.

100/92 in the first row means that the number of correct predictions is 100 and 92 for the models initialized by the ImageNet weights and random weights, respectively. The number of test images for each species is 105, as mentioned earlier. Hence, ResNet50 with the ImageNet weights correctly classified 100 out of 105 samples. The confusion matrix is an important measure of the true performance of each model. Because the models were evaluated on

previously unseen data, we can verify whether they can recognize general features of the species. The results show that the models can classify the images in the training and validation sets with more than 90% of accuracy (learning curves of training and validation) but it does not seem to apply to the confusion matrix of random-number-initialized models (right-side values of the confusion matrix). Therefore, some pieces of information for validation were

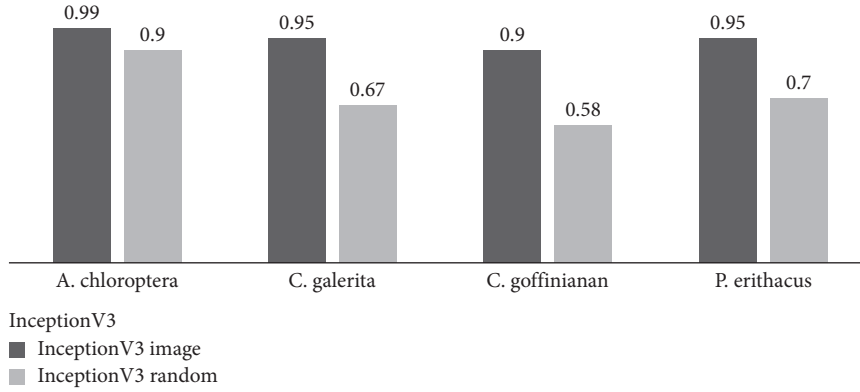


FIGURE 15: F1-score of InceptionV3 for four different endangered parrot images.

leaked out during the training; hence, the models memorized the features of validation instead of general features of species. According to our results, the models with ImageNet weights classify the images better than the other methods, even though the images are completely new. For example, the results are 98/66 and 88/55 for ResNet50 in Table 2. This finding stands not only for ResNet50 but also for the other models. The number of correct predictions for each model is 100 out of 105, 98 out of 105, and 94 out of 105 for *Cacatua galerita*; 88 out of 105, 89 out of 105, 95 out of 105, and 97 out of 105 for *Cacatua goffiniana*, respectively.

Figures 12–15 show F1-scores of the models. F1-score is a way to quantify the results of the confusion matrix. F1-score is calculated using precision and recall by

$$F1 = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2)$$

Precision reflects how many predicted items are correct. Recall reflects how many correct items are predicted. Precision can be calculated by dividing the number of true positives by the number of positive predictions. For instance, ResNet50 with ImageNet classified 105 images as *Ara chloroptera* in the test set. The number of true positives is 100. Therefore, the precision of ResNet50 is 100 out of 105. Recall can be calculated by dividing the number of true positives by the number of true cases. For ResNet50, the total number of true cases is 105; hence, the recall of the model is 100 out of 105. We can calculate the F1-score by substitution of the results:

$$2 * \frac{(100/105) * (100/105)}{(100/105) + (100/105)} \approx 0.95. \quad (3)$$

Figure 12 shows the F1-score of *Ara chloroptera*. The F1-score is more effective than simple accuracy when we measure the model’s performance because it considers the data distribution (unlike the accuracy). Let us suppose that we have 90 images with the first label and ten images with the second label. We can obtain 90% of accuracy if we classify all images as “the first label.. F1-score avoids this problem. Overall, we conclude that the ImageNet-based models are superior to the random-number-initialized models for quasi-species of parrots.

4.2. Mobile Application. The graphical user interface of the real-time mobile application developed in this study is shown in Figure 9. NASNetMobile model with ImageNet weights was converted into a FlatBuffer file (.tflite) and added to the application. Subsequently, we used Android Studio to edit the code and add visual elements. First, we checked that Android Studio, SDK version, and dependencies were compatible with TensorFlow Lite. After the model in a FlatBuffer file was located in a project, we built it, and then an APK was created. Finally, the application was installed on a device.

The parrot images were captured by the mobile device’s camera. Next, the trained model classified the image. Finally, the application showed the result of the model. We can check the result at the bottom of the screen, as seen in Figure 9. The first image of Figure 9 shows a preview of a parrot image: a text line presents that this parrot is “*Ara chloroptera*” as one hundred percent. “345 ms” is seen at the lowest part of the image: it means that it took 345 ms to classify this image. The average turnaround time was 460 ms, the minimum time was 229 ms, and the maximum time was 671 ms for 50 iterations. According to our findings, the application processed jobs under 1 second.

5. Discussion

In this paper, we proposed classifiers for endangered parrot species. The models extract the features of the parrot appearances at the convolutional layer, which has been pre-trained on a large amount of data, and then we classify the images at the last layer. Our proposed models require a relatively short time to conduct their job. They are more accurate than the models trained from scratch, especially for the species that have a similar color. This is because the pretrained models can already extract the low-level features of a new image. Another advantage of the models trained by transfer learning is that the model does not need to draw a bounding box to train the last layer. This approach will greatly reduce the inconvenience for humans by eliminating manual processes. We expect that the accuracy will be increased if fine tuning is applied. Finally, Tf.keras-based model can be easily deployed on an Android mobile device using the FlatBuffer file converter provided by TensorFlow

Lite. To clarify the key points of this study, we suggest the following highlights:

- (i) CNN models with transfer learning can be trained without any special difficulty
- (ii) The designed advanced CNN models do not require any manual preprocessing (such as labeling or drawing bounding boxes on the images)
- (iii) The CNN models can be easily converted into a file for deploying in a mobile application using TensorFlow Lite framework
- (iv) The mobile application can classify endangered quasi-species of parrots having a high color similarity in real time

6. Conclusions and Future Work

In our proposed system, the mobile application classifies the image acquired from the device camera in real time. To sum up, our system works as follows. We used two methods to create a high-quality model with a small amount of original data. First, we used data augmentation to increase the amount of data by manipulating the original data. Second, we used transfer learning to extract the characteristics of the image smoothly. Specifically, we used the convolutional layers pretrained on a large amount of data. Next, we used the FlatBuffer file converter provided by TensorFlow Lite to deploy this model on a mobile device. For quasi-species of parrots, the accuracy of the classification models with transfer learning is approximately 20% higher than that of the models trained from scratch.

Based on this study, we also expect that further studies on advanced topics could be explored as follows. First, the results can be improved when a fine-tuning process is added, as mentioned in Section 5. Second, in addition to the classification of the four species of parrots in this study, it is possible to carry out accurate classifications for parrots on more than ten species.

Data Availability

The image data used to support the findings of this study are available from the corresponding author upon request. But only some sample data are available because this study is under Ministry of Environment, Republic of Korea.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Ministry of Environment, Republic of Korea (2018000210004).

References

- [1] Y. Xue, S. Chen, J. Qin, Y. Liu, B. Huang, and H. Chen, "Application of deep learning in automated analysis of molecular images in cancer: a survey," *Contrast Media & Molecular Imaging*, vol. 2017, Article ID 9512370, 10 pages, 2017.
- [2] Z. Xie and C. Ji, "Single and multiwavelength detection of coronal dimming and coronal wave using faster R-CNN," *Advances in Astronomy*, vol. 2019, Article ID 7821025, 9 pages, 2019.
- [3] M. S. Norouzzadeh, A. Nguyen, M. Kosmala et al., "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 25, pp. E5716–E5725, 2018.
- [4] B. Mridula and P. Bonde, "Harnessing the power of deep learning to save animals," *International Journal of Computer Applications*, vol. 179, no. 2, 2017.
- [5] A. Sadaula, Y. Raj Pandeya, Y. Shah, D. K. Pant, and R. Kadariya, *Wildlife Population Monitoring Study Among Endangered Animals at Protected Areas in Nepal*, IntechOpen, London, UK, 2019.
- [6] Z. Huang, Z. Pan, and B. Lei, "Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data," *Remote Sensing*, vol. 9, no. 9, p. 907, 2017.
- [7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.
- [8] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the International Conference on Learning Representations*, Toulon, France, April 2017.
- [9] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015.
- [10] C. Szegedy, S. Ioffe, V. Vincent, and A. Alexander, *Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning*, AAAI Press, San Francisco, CA, USA, 2017.
- [11] H. Nguyen, S. J. Maclagan, T. D. Nguyen et al., "Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring," in *Proceedings of the 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Tokyo, Japan, October 2017.
- [12] P. Zhuang, L. Xing, Y. Liu, S. Guo, and Y. Qiao, "Marine animal detection and recognition with advanced deep learning models," in *Proceedings of the CLEF 2017*, Dublin, Ireland, September 2017.
- [13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [14] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, E. S. Olivares, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and A. J. S. López, Eds., pp. 242–264, IGI Global, Hershey, PA, USA, 2010.
- [15] R. Kumar Sanodiya and J. Mathew, "A novel unsupervised globality-locality preserving projections in transfer learning," *Image and Vision Computing*, vol. 90, 2019.
- [16] J. Ma, J. C. P. Cheng, C. Lin, Y. Tan, J. Zhang, and J. Zhang, "Improving air quality prediction accuracy at larger temporal resolutions using deep learning and transfer learning techniques," *Atmospheric Environment*, vol. 214, Article ID 116885, 2019.

- [17] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," in *Proceedings of the IEEE International Conference on Big Data*, pp. 1367–1376, Seattle, WA, USA, December 2018.
- [18] M. Sabatelli, M. Kestemont, D. Walter, and P. Geurts, "Deep transfer learning for art classification problems," in *Proceedings of the the European Conference on Computer Vision (ECCV) Workshops*, Munich, Germany, September 2018.
- [19] J. Alexander, "imgaug," 2019, <https://github.com/aleju/imgaug>.
- [20] TensorFlow, "TensorFlow lite converter," 2019, <https://www.tensorflow.org/lite/convert>.
- [21] X. Li, R. Long, J. Yan, K. Jin, and J. Lee, "TANet: a tiny plankton classification network for mobile devices," *Mobile Information Systems*, vol. 2019, Article ID 6536925, 8 pages, 2019.
- [22] Keras, "The Python deep learning LIBRARY," 2019, <https://keras.io>.
- [23] D. Rong, L. Xie, and Y. Ying, "Computer vision detection of foreign objects in walnuts using deep learning," *Computers and Electronics in Agriculture*, vol. 162, pp. 1001–1010, 2019.
- [24] A. Lin, J. Wu, and X. Yang, "A data augmentation approach to train fully convolutional networks for left ventricle segmentation," *Magnetic Resonance Imaging*, vol. 66, pp. 152–164, 2019.
- [25] D. Zhao, G. Yu, P. Xu, and M. Luo, "Equivalence between dropout and data augmentation: a mathematical check," *Neural Networks*, vol. 115, pp. 82–89, 2019.
- [26] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, <https://arxiv.org/abs/1312.4400>.
- [27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
- [28] D. G. Cheo, E. Choi, E. C. Lee, and K. Dong, "The mobile applications based on vision-object detections for classifying of endangered parrot species using the CNN deep model," in *Proceedings of the 2018 Americas Conference on Medical Imaging and Clinical Research (AMICR 2018)*, Panama, December 2018.
- [29] J. M. Forshaw, *Parrots of the World*, Princeton University Press, Princeton, NJ, USA, 2010.
- [30] Keras, "Applications," 2019, <https://keras.io/applications/>.
- [31] Scikit-Learn, "sklearn.metrics.Confusion_matrix," 2019, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.
- [32] Scikit-Learn, "sklearn.metrics.Classification_report," 2019, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html.
- [33] Tensorflow, "Tensorflow/tensorflow," 2019, https://github.com/tensorflow/tensorflow/blob/master/tensorflow/lite/python/tflite_convert.py.

Review Article

Deep Learning on Computational-Resource-Limited Platforms: A Survey

Chunlei Chen ¹, Peng Zhang ¹, Huixiang Zhang ², Jiangyan Dai ¹, Yugen Yi ³,
Huihui Zhang ¹ and Yonghui Zhang ¹

¹School of Computer Engineering, Weifang University, Weifang, China

²School of Cyberspace Security, Northwestern Polytechnical University, Xi'an, China

³School of Software, Jiangxi Normal University, Nanchang, China

Correspondence should be addressed to Chunlei Chen; chunlei.chen@wfu.edu.cn

Received 16 August 2019; Revised 28 December 2019; Accepted 1 February 2020; Published 29 February 2020

Guest Editor: Malik Jahan Khan

Copyright © 2020 Chunlei Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nowadays, Internet of Things (IoT) gives rise to a huge amount of data. IoT nodes equipped with smart sensors can immediately extract meaningful knowledge from the data through machine learning technologies. Deep learning (DL) is constantly contributing significant progress in smart sensing due to its dramatic superiorities over traditional machine learning. The promising prospect of wide-range applications puts forwards demands on the ubiquitous deployment of DL under various contexts. As a result, performing DL on mobile or embedded platforms is becoming a common requirement. Nevertheless, a typical DL application can easily exhaust an embedded or mobile device owing to a large amount of multiply and accumulate (MAC) operations and memory access operations. Consequently, it is a challenging task to bridge the gap between deep learning and resource-limited platforms. We summarize typical applications of resource-limited deep learning and point out that deep learning is an indispensable impetus of pervasive computing. Subsequently, we explore the underlying reasons for the high computational overhead of DL through reviewing the fundamental concepts including capacity, generalization, and backpropagation of a neural network. Guided by these concepts, we investigate on principles of representative research works, as well as three types of solutions: algorithmic design, computational optimization, and hardware revolution. In pursuant to these solutions, we identify challenges to be addressed.

1. Introduction

The last decade has witnessed exciting development of deep learning (DL) technologies, which contributes dramatic progress in signal and information processing applications including IoT and smart sensing. A deep neural network (DNN) comprises multiple neuron layers organized in a hierarchical structure. Parameters of every layer can be learned through iterative training. A well-trained DNN can distill useful features from raw data. All training samples are manually labeled. In one layer, input data can be mapped into a low-dimensional space through feature extraction. Subsequently, output features of the current layer are exported into the next layer. Outputs of the last layer imply the learned labels. A DNN can be fine-tuned through

minimizing the error between manual labels and learned labels [1].

Deep learning enjoys significant advantages over traditional machine learning [2, 3]. First, deep learning can achieve superior performance when data volume is massive. This means that deep learning can fully benefit from the huge amount of data collected by IoT. Traditional machine learning techniques are preferable when data volume is small. However, the performance prominently degrades when data volume is extremely large. In contrast, deep learning exhibits advantageous scalability with massive data. Second, deep learning relies less on feature engineering. IoT can gather diversified categories of data that are distinct in nature. Manually extracting features of heterogeneous data is a daunting task. Traditional machine learning requires a

domain expert to extract features. The manually identified features expose underlying patterns to algorithms. Nevertheless, deep learning autonomously extract features in a layer-wise manner to represent input samples with a nested hierarchy of features. Every layer defines higher-level features based on lower-level features extracted by the previous layer. Third, deep learning techniques can outperform traditional ones in terms of various smart-sensing-related tasks, such as computer vision, speech recognition, and human behavior understanding.

By contrast with traditional machine learning solutions, deep learning techniques are undergoing rapid development. Applications of deep learning involve information retrieval [4], natural language processing [5], human voice recognition [6], computer vision [7], anomaly detection [8], recommendation systems [9], bioinformatics [10], medicine [11, 12], crop science [13], earth science [14], robotics [15–18], transportation engineering [19], communication technologies [20–22], and system simulation [23, 24].

Deep learning is permeating into diversified aspects of human society, which puts forwards urgent demand on the ubiquitous deployment of DL-powered applications. In other words, deep learning is required to be fit into resource-limited platforms like smartphones or wearable devices. Nevertheless, matching DL and resource-limited platforms is a challenging task. Inferencing with DL is extremely resource-consuming (processor, memory, energy, etc) even though the more resource-consuming training phase can be offloaded onto high-performance-computing-powered mainframes. We investigate on typical resource-limited DL inferencing solutions by categorizing the solutions and discussing open questions. The rest of this paper is organized as follows. Section 2 clarifies impetus of developing resource-limited DL. Representative solutions are discussed in Section 3. Section 4 points out the challenges to be addressed. Section 5 concludes our work.

2. Computational-Resource-Limited Context of Deep Learning

2.1. Application Scenarios. Figure 1 shows typical applications of computational-resource-limited DL in the smart sensing context, including self-driving [25, 26], artificial intelligence APPs of smartphones [27], health/homecare robots [28–31], and intelligent wearable devices [32]. The DNN can be pretrained on remote cloud while the mobile DL platforms communicate with the cloud and perform inference based on local computational and energy resources [33]. All these applications rely on embedded computer with limited onboard resources such as processor, memory, and battery. Two fundamental technologies of such applications are sensor data processing and computer vision.

Recognizing and feeding back to user behavior and surrounding environment are the core functionalities of state-of-the-art Internet-of-Things (IoT) and mobile sensing applications. Nevertheless, raw sensor data are inevitably mixed with noise and uncertainty due to the complicated deployment environment. As a result, distilling precise and meaningful knowledge from raw sensor data is a challenging

task. DL is one of the most competitive methods to conquer this challenge [34].

The prevalence of wearable (head-mounted) augmented reality (AR) devices has open a way to a novel class of mobile computer vision applications, including the Microsoft HoloLens [35] and the Google Glass [36]. These applications vary from real-time traffic signal identification for navigation to human recognition for healthcare APPs. All these application scenarios propose the common demand to process continuous video streams in real time. The current leading-edge technology of video stream processing is DL, which handles video streams using a large-scale and pre-trained convolutional neural network (CNN) or recurrent neural network (RNN) [37].

2.2. A Perspective of Pervasive Computing. Deep learning can automatically extract features and achieve higher accuracy than traditional artificial intelligence techniques. As a result, deep learning is applicable to a broad range of scenarios. Additionally, open-source development tools like TensorFlow and Caffe are also speeding up progresses in deep learning. Research works on fitting deep learning into resource-limited mobile or embedded platforms will undoubtedly push a huge step forward towards the pervasive deep learning.

Deep learning is currently an indispensable impetus that advances the progress of pervasive computing. As shown in Figure 2, we summarize the development of pervasive computing into three stages. The hardware and software solutions of a former stage are incorporated into the latter stages. In the 1990s, researchers in this area try to facilitate the daily life of humans through Internet-interconnected desktops and mainframes. TCP/IP protocols account for the backbone of networks and the software layer of pervasive applications typically focuses on network organization and data delivery. In the following stage, the mobile Internet provides network access to users at any time and any place. IoT interconnects almost all digital sensors to collect raw data from diversified sources, which results in large data volume and puts forward high demand on the computing power of the data processing platform. Thus, distributed or parallel middleware like Hadoop aggregates the computing power of huge amounts of commodity servers. Additionally, cloud computing provides the aggregated supercomputing power to customers through Web Service. Data transmission between IoT and cloud platforms is further supported by WIFI and 3G/4G. However, applications of this stage mainly adopt traditional machine learning solutions, which cannot achieve constantly advancing performance with the continuous increase of input data volume. Nowadays, the learning and inference accuracy of DNN can efficiently scale with the input data amount. However, high time and memory overheads impede the deployment of DL on resource-limited platforms. Matching deep learning and hardware platforms is an active research area. Software layer solutions mainly focus on simplifying the trained DNN to approximate a full-status DNN. Hardware layer solutions involve embedded GPUs, artificial intelligence chips, or even

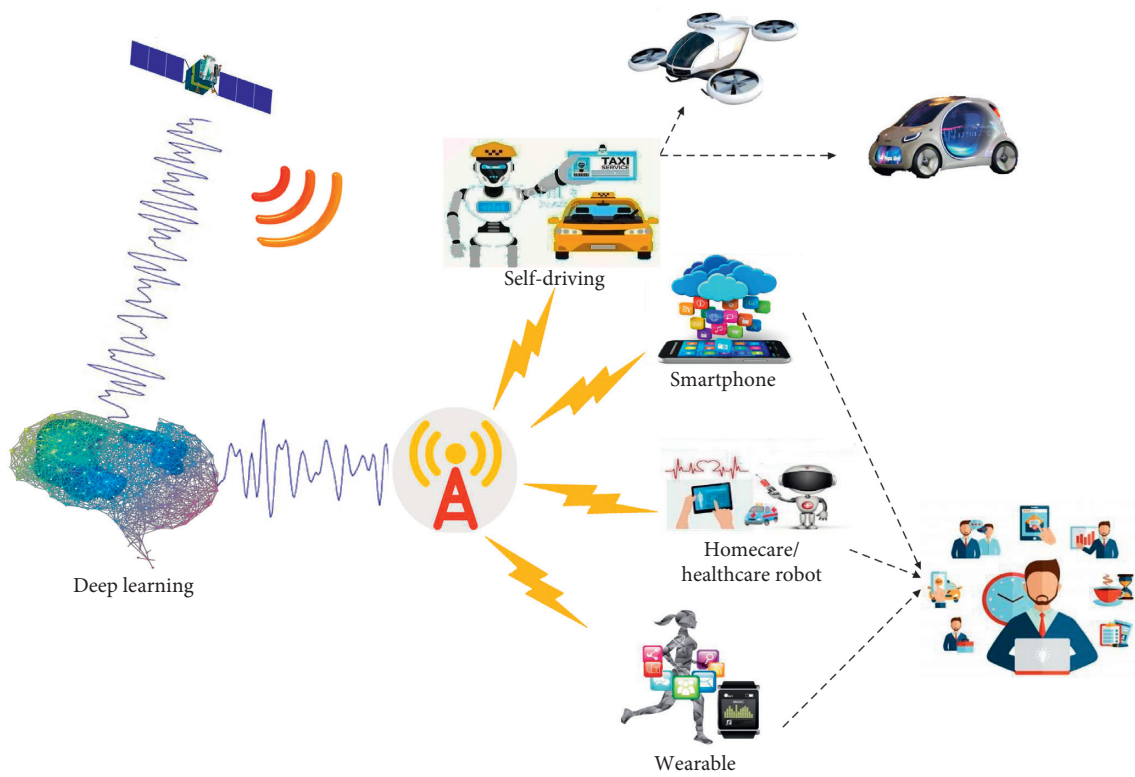


FIGURE 1: Typical smart-sensing-related application scenarios of resource-limited deep learning.

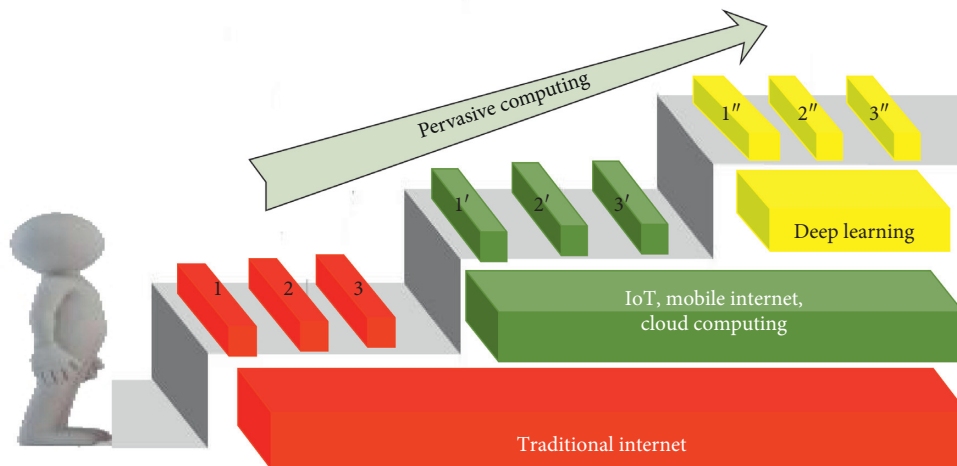
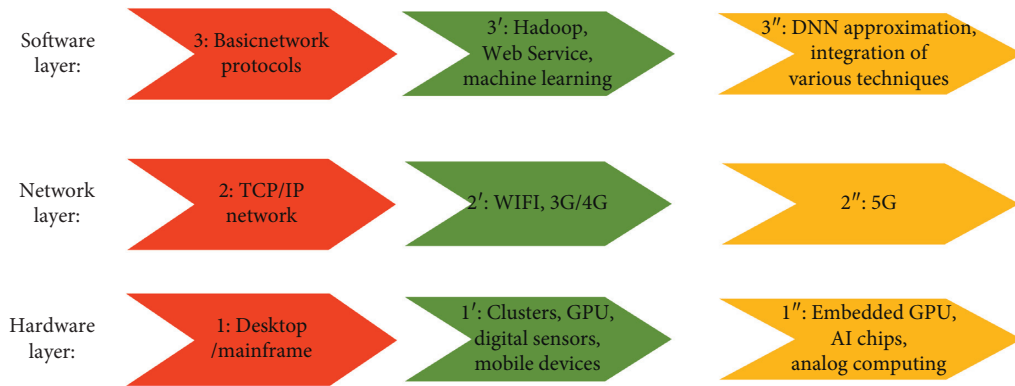


FIGURE 2: Retrospect of pervasive computing: deep learning is currently one of the dominant impetuses for pervasive computing.

analog computing based on new nonvolatile memory. Additionally, 5G will meet even higher bandwidth requirements.

3. Investigation on Existing Solutions

3.1. Computational Predicament of DNN: A Perspective of Underlying Principles. Classification is a typical application scenario of DNNs. Under this scenario, the target is to establish a mapping from input samples to corresponding labels. The following concepts are the cornerstones to exploit the learning and inference of DNNs: hypothesis space, capacity, stochastic gradient descent, and generalization [38].

Hypothesis space is the set of all functions generated by a neural network. One function is obtained by fitting part of parameters of the neural network and can map homogeneous samples to the same label. Training a neural network is to search the optimal functions in the hypothesis space, which can build mapping relationships specified by the training data (in other words, minimizing the training error). As a result, the size of hypothesis space determines the potential ability of a neural network to find optimal functions.

Capacity of a neural network reflects the size of hypothesis space, as well as the upper bound of ability to fit functions. The optimal functions may be beyond the hypothesis space, if the capacity is not sufficiently large. In this case, the neural network can only search in the limited hypothesis space and find functions that approximate the optimal functions with best efforts. Consequently, underfitting is inevitable.

A trained neural network is expected to correctly predict the label of previously unseen samples. Generalization reflects this kind of ability. Lower generalization error means higher generalization ability. Underfitting during the training phase can result in large generalization error in the inference phase.

Capacity sets the limit of the fitting ability, while generalization can measure the ability of scaling with unknown samples. Another vital issue with neural networks is the mechanism of searching the hypothesis space in the training phase. Conventionally, the searching is manipulated by stochastic gradient descent; searching is always along the direction in which training error drops fastest. The gradients are backpropagated from the deepest layer to the first layer to update weights in a layer-wise manner. Backpropagation converges when the difference of train errors between two successive iterations is smaller than a threshold. However, stochastic gradient descent commonly cannot reach the global optima. Despite that a near-optimal solution is generally sufficient to train a low-error neural network, this method typically requires a long time to converge. Moreover, parameters like step length should be carefully selected to avoid fluctuation of the gradient.

From the perspective of underlying principles, the computational predicament of DNNs is due to the following reasons.

The first is memory overhead. Oversized network is a conventional method to achieve low generalization error. A large capacity does not necessarily result in low generalization error. However, a large hypothesis space raises the upper bound of the generalization ability and thus increases the possibility of reaching a low error, especially when the target functions are not excessively complex.

The second is time and energy overhead. Back-propagation is inherently iterative and time-consuming. The gradient is calculated by minimizing the training error. The training error is a function of weights and other parameters. The huge number of weights leads to a slow convergence speed. Moreover, these weights need to be frequently transmitted between processing units and memory. Consequently, the long-time computation and intensive memory operations raise high demand on the processing ability and energy duration. In addition, values of hyperparameters are conventionally selected through fine-tuning, which multiplies the time overhead.

The third is the curse of dimensionality. High dimensionality of data aggravates the computational resource consumption. DNNs commonly need a large volume of training data to guarantee the generalization ability of the trained network. Higher dimensionality requires denser samples. If A_1 is the number of necessary training data points in the one-dimensional sample space, then the number of training data points is A_1^n in n -dimensional sample space [38]. More training data points of higher dimension inevitably exacerbate overheads of memory, time, and energy.

3.2. Challenges to Be Investigated. Deep learning is currently more art than a science. Neural networks are inherently approximate models and can often be simplified [39].

In spite of the dramatic learning power of deep learning, computational cost has impeded their portability to resource-limited platforms [40]. DL algorithms are facing three kinds of barriers to optimize computational performance. *The first barrier* is the resource-consuming iterative nature of DL training. Moreover, the experiential nature aggravates this kind of iterative cost. Up to now, the success of deep learning mainly relies on empirical designs and experimental evaluations. Theoretical principles are still to be exploited. As a result, optimizing the performance of deep learning requires implementing and executing various possible models within the computational resource constraints to empirically recognize the optimal one [41]. Extracting meaningful knowledge from a single input sample can require enormous MAC operations. The number of MAC operations can reach the magnitude of billion [42]. Additionally, a single deep learning network can contain over a million parameters [43]. As a result, deep learning proposes high demands on processing ability, memory capacity, and energy efficiency. It is a vital issue to optimize deep learning networks by eliminating ineffectual MAC operations and parameters [42]. *The second barrier* is fitting DNNs into diversified modern hardware platforms. Different hardware platforms can be distinct in terms of clock

frequency, memory access latency, intercore communication latency, and parallelism mode. Designer of DL model can be categorized into two different types: data scientist and computer engineer. Data scientists mainly concentrate on optimizing training and inference accuracy through data and neural network techniques. However, they have little or even no concern with computational cost. Efforts to upgrade accuracy do not necessarily result in smaller network size and higher speed. Computer engineers focus on accelerating deep learning based on hardware platforms. They fine-tune or even reform DNNs to match the models to the design requirements for resource-constrained applications. *The third barrier* is lack of dedicated hardware. Traditional general-purpose digital computing hardware such as CPU, GPU, and FPGA neglect some unique characteristics of deep learning. For example, deep learning only involves limited kinds of computational operations. Additionally, deep learning is significantly tolerant to noise and uncertainty. Dedicated hardware may trade off universality for performance [44–48].

Cloud-powered DL has been an active research area. Such solutions can offload heavy computation onto the remote cloud hosts. Such methods assemble data from mobile or embedded devices, transfer the data to cloud, and perform deep learning algorithms (both training and inferencing) on cloud. Users are facing the risk of privacy leakage due to data transmission through computer networks, particularly if the data contain sensitive information. In addition, the reliability of cloud-based deep learning may be affected by network package loss or even network failure. *In this paper, we focus on three issues: first, trade-off between neural network capacity and generalization error using algorithmic design; second, fitting DNN into digital hardware through computational design; and third, next-generation hardware to cope with the computational predicament of DNN.* We categorize the existing solutions into three layers: the algorithmic, computational, and hardware layers.

Figure 3 summarizes typical solutions. A practical method may integrate more than one solutions.

3.3. Algorithmic Design. Algorithmic designs focus on reducing resource consumption through mathematically adjusting or reforming the DNN model and algorithm. Typical simplification techniques include depthwise separable convolution, matrix factorizing, weight matrix sparsification, weight matrix compression, data dimension reduction, and mathematical optimization.

Howard et al. designed a series of neural network models (MobileNets) to facilitate machine vision applications on mobile platforms [49]. MobileNets represent a kind of lightweight deep neural network based on depthwise separable convolutions. The main goal of MobileNets is to construct real-time and low-space-complexity models to satisfy the demands raised by mobile machine vision applications. The contributions of MobileNets are summarized as follows. First, core layers of MobileNets are derived from the depthwise separable convolution. The core concept of the depthwise separable convolution is to factorize a

conventional convolution into a depthwise separable convolution layer and a pointwise convolution layer [50]. MobileNets adopt this core concept to reduce the model size, as well as the total number of multiplication and addition operations. Second, pointwise convolutions account for 95% of the total computation while the im2col reordering optimization is unnecessary for pointwise convolutions [51]. Thus, MobileNets avoid massive computation of im2col reordering. Third, since MobileNets generate relatively small models and require comparatively few parameters, conventional anti-overfitting measures are adjusted. For instance, less regularization and data augmentation are used. Additionally, minimal weight decay (L2 regularization) is adopted on the depthwise filter. Fourth, two hyper-parameters called width multiplier and resolution multiplier are applied to further shrink the model size.

The core concept of [49] is factorizing a conventional convolution to lower the computation complexity. This factorization does not affect the inference accuracy and thus is a lossless simplification method. However, lossy simplification is necessary if superior simplification effect is demanded. Samrath et al. customize DL network to match FPGA platform [39]. This method simplifies the weight matrix through clustering and encoding. Additionally, matrix-vector multiplication operations are factorized to decrease computational complexity. First, elements of the weight matrix are clustered by k -means into K clusters. Thus, every element is affiliated to a cluster, and the center of every cluster is the mean of its affiliated elements. Consequently, every element in the weight matrix is replaced with the corresponding center. In other words, every weight is approximated with the center of its affiliated cluster. Second, the approximate weights are encoded with a bit width of $\log K$. And all cluster centers form a dictionary vector. As a result, encoding can significantly lower memory overhead. Third, the matrix-vector multiplication can be factorized due to the fact that the encoded matrix has abundant repetitive elements. Therefore, the number of floating-point multiplication operations is dramatically reduced, which means lower computational complexity. In addition to the aforementioned three basic steps, this method faces another problem: replacing weights with cluster centers inevitably induces numerical error to the DL network. This error can affect the inference accuracy. The method of [39] adopts two solutions to handle this error. One is increasing the length of the dictionary vector (in other words, designating a larger K to k -means). The other is to iteratively cluster and retrain the weights. The method of [39] focuses on compressing the already trained weight matrix. By contrast, methods like lasso regularization can sparsify the weight matrix during training [52].

Lane et al. propose a software framework named *DeepX* to reshape the DNN reference model under limited resource constraints [53]. By contrast to the clustering method of [39], *DeepX* uses SVD decomposition and reconstruction error minimization to compress the DNN model. On the first level, they adopt SVD decomposition to reconstruct and approximate the weight matrix of every DNN layer. Thus, *DeepX* dramatically reduces the amount of DNN

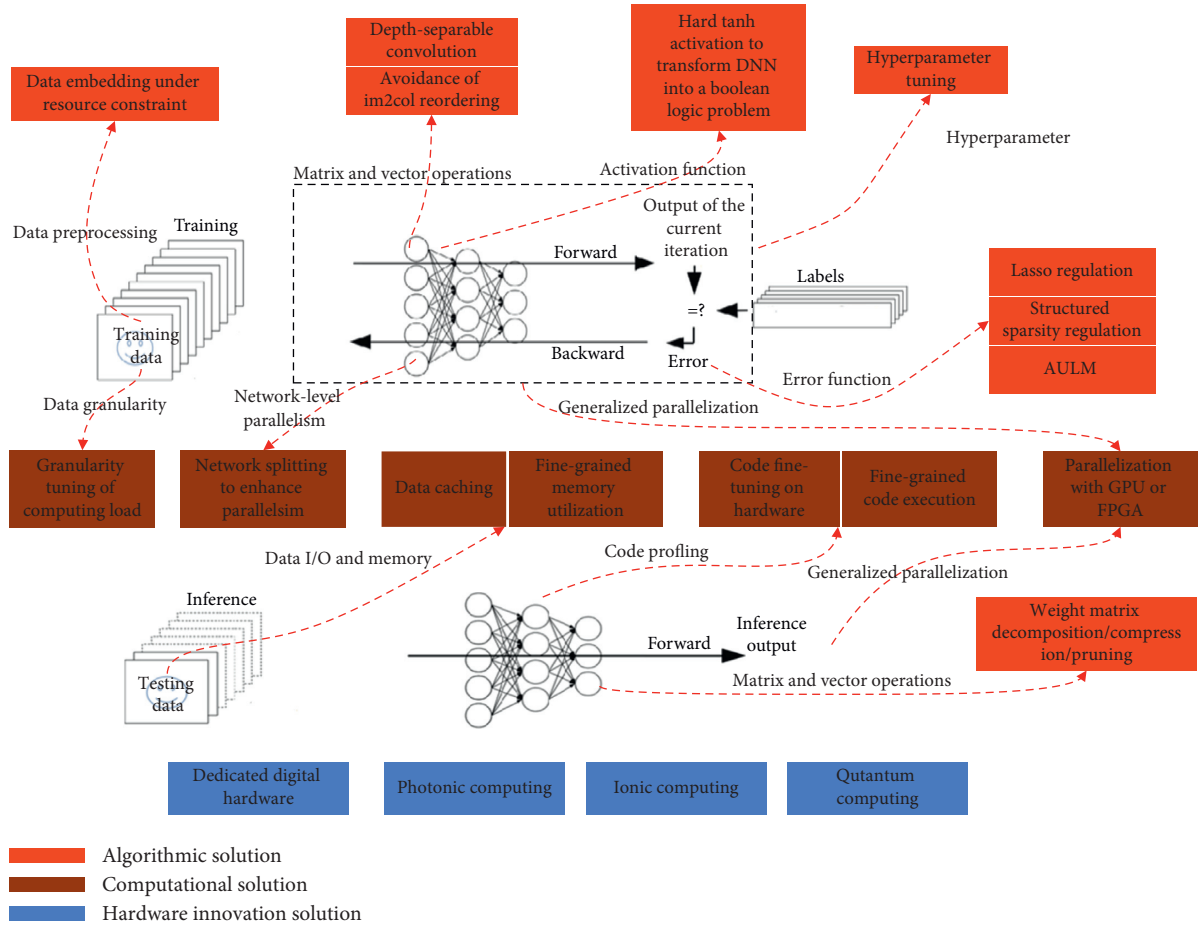


FIGURE 3: Categorization of existing resource-limited DL solutions: from the perspectives of training and inferring.

parameters in each layer. Additionally, the accuracy of this approximation is measured and tuned in pursuant to the reconstruction error. As a result, this reconstruction method avoids the predicament of retraining. On the second level, *DeepX* quantizes the computation loads of every neuron and formalizes workload scheduling as a constrained dynamic programming problem. In this manner, computation load can be automatically scheduled onto processors to meet energy and time constraints.

Pruning or compressing an already-trained DNN could result in large approximation error [54–57]. One alternative is to train a sparse DNN. Lin et al. propose a method named structured sparsity regularization (SSR) to achieve weight matrix sparsification during training [58]. They introduce two distinct structured-sparsity regularizers into the objective function of matrix weight sparsification. These two regularizers can constrain the intermediate status of DNN filter matrix to be sparse. Subsequently, they adopt an Alternative Updating with Lagrange Multipliers (AULM) scheme to alternatively optimize the sparsification objective function and minimize recognition loss. The SSR method enjoys significantly lower time and memory overhead than state-of-the-art weight matrix pruning methods. Nazemi et al. propose a DNN training method to remove redundant memory access operations. This method utilizes Boolean

logic minimization [59]. In the training process, the *sign* function is adopted as the activation. Consequently, activations are confined to binary values. Every layer of the DNN (except the first layer and the last layer) is modeled as a multi-input multioutput Boolean function. In the inference process, outputs of the DNN are obtained through synthesizing a Boolean expression other than computing the dot product of the input and weight. In other words, enormous memory accessing operations are avoided, which removes vast memory access latency and energy consumption.

The aforementioned algorithmic solutions focus on simplifying the DNN model so as to reduce MAC operations and memory consumption. Nevertheless, physical durability, especially energy efficiency, is still a daunting barrier to benefit various practical applications through deep learning. *DeLight* is a low-overhead framework that facilitates efficient training and execution of deep neural networks under low-energy constraints [60]. Authors of [60] restrain the DL network size through energy characterization in pursuant to pertinent physical resources. They design an automatic customization methodology to adaptively fit the DNN into the specific hardware while inducing minimum degradation of learning accuracy. The core concept of *DeLight* is to project data to low-dimensional embeddings (subspaces) in a context-and-resource-aware manner. Consequently,

insights into data samples can be achieved through dramatically less neurons. Moreover, trained models in every embedding are integrated to enhance learning accuracy.

The core concept of *DeLight* is fine-grained energy consumption control based on data dimension reduction. The framework *HyperPower* proposes to bound energy and memory consumption from the point of hyperparameter optimization [41]. This is a hyperparameter optimization framework based on Gaussian process (GP) and Bayesian optimization [61, 62]. This framework denotes test error as a function $f(x)$, where x is a data point in the design space of hyperparameters. Additionally, power and memory overhead is denoted as a function $g(x)$. Subsequently, hyperparameter tuning is formalized as an optimization problem: minimizing $f(x)$ under the constraint that $g(x)$ is lower than a threshold. Minimizing $f(x)$ is costly due to the fact that $f(x)$ has no close form. Consequently, *HyperPower* adopts GP to approximate distributions of $f(x)$. Moreover, the framework leverages Bayesian optimization to iteratively select optimal hyperparameters and update distribution of $f(x)$. $f(x)$ is assumed to obey Gaussian distribution. Let y denote the observations of $f(x)$. At the very beginning, an initial approximation of $f(x)$ can be resolved as $p_M(y|x)$ based on the assumption and a set of known (x, y) values (Gaussian process regression). Every iteration includes the following operations. The primary task is to select an optimal value of x from the design space to refine $p_M(y|x)$. And the selected x should push the $f(x)$ value along a direction of decrease. This value of x is identified through maximizing an expectation-improvement-based acquisition function. In addition, the acquisition function incorporates the constraint using an indicator function. The indicator function equals to one if the constraint is satisfied and zero if not. Second, the neural network is configured in accordance with the new design parameter (the newly identified x) and trained to obtain the test error (a new value of y). Third, the mean and covariance are updated using the new (x, y) , and thus, $p_M(y|x)$ is updated to $p_M(y)$.

3.4. Computational Optimization. Computational optimization relies on reengineering the algorithm implementation in accordance with a specific hardware architecture. Some conventional optimization techniques are code parallelizing, fine-tuning of parallel code, data caching, and fine-grained memory utilization.

Huynh et al. developed a tool *DeepMon* for continuous vision applications based on commodity mobile GPUs [37]. Large deep neural networks (DNNs) powered by commodity mobile GPU commonly cannot achieve strict real-time performance due to limited computational resources. However, the frame rate can be low (one to two frames per second) under some use cases, such as speaker recognition and elder nursing care. These application scenarios put forward comparatively low demands on real-time performance. *DeepMon* implements large DNNs for such applications based on commodity mobile GPUs and achieves near real-time performance. In the aforementioned applications, first-person-view images are not apt to exhibit significant

changes during a short time span. *DeepMon* divides each frame of image into equal-size blocks. *DeepMon* cached the intermediate results of each block when calculating the convolution of one frame. Subsequently, similar blocks are identified between this frame and the next frame. Consequently, the cached results can be directly utilized to calculate convolution of the next frame. Additionally, cached results expire after a certain time period. Similarity between two images is identified based on color distribution histogram and chi-square distance metric. In addition to this caching mechanism, *DeepMon* leverages Tucker-2 decomposition convolution layers [63] to factorize a traditional convolution layer into several small convolution layers. As a result, computation cost of convolution is reduced. Finally, *DeepMon* tunes GPU codes on various mainstream commodity mobile GPUs. Tuned and optimized GPU codes are encapsulated into separate kernels for each GPU model. As a result, *DeepMon* can adaptively adopt appropriate kernels at runtime so as to fit into a specific GPU with best efforts.

The main idea of *DeepMon* is caching the intermediate result to eliminate redundant computation. Another typical technique is GPGPU acceleration. Cao et al. proposed a GPGPU-powered RNN model that executes locally on mobile devices [64]. Recurrent neural network (RNN) can find wide applications such as speech recognition and robot chatting. Traditional mobile applications of RNN generally offload main computation onto the cloud. However, the cloud-based implementation induces security and efficiency issues. Cao et al. pointed out that existing GPGPU-accelerated methods for convolutional neural network (CNN) cannot directly be transplanted to mobile-device-based RNN. On the one hand, RNN inherently contains many sequential operations, which constrains the parallelism of RNN. On the other hand, existing GPGPU-powered RNN methods are specially designed for desktop GPGPUs. Such methods can not directly fit into mobile GPGPUs due to the fact that the mobile GPGPU possesses significantly less memory capacity and processing cores. In a RNN, the inevitable dependencies between adjacent cells dramatically increase the difficulty in exploiting parallelism among cells. Nevertheless, operations within a cell still exhibit considerable parallelism. In the work of [64], computation of the cell is factorized in fine granularity and elegantly fits into the mobile GPGPU.

The adaptive platform DL framework *Deep³* still adopts the idea of GPGPU-powered computing. However, *Deep³* exploit parallelism from three levels: data, network, and hardware. The ultimate goal of *Deep³* is to bridge the gap between data science perspective design of deep learning and computer engineering perspective optimization of deep learning. First is hardware parallelism. *Deep³* extracts basic operations (layers) of a deep learning network, including convolution, maximum pooling, mean pooling, matrix multiplication, and nonlinearities. Optimized implementation of a basic operation can be dramatically distinct with regard to the hardware platform. For example, by altering the dimensionality of matrices, we can observe that matrix multiplication is computation-intensive or data-intensive on

a specific platform. *Deep³* employs subroutines to perform hardware profiling. Each subroutine runs a specific operation with varying sizes on different platforms, separately. In this manner, *Deep³* recognizes the optimal size of a specific operation regarding a target platform. These optimal sizes are vital instructions to split an entire deep learning network into subnetworks, which adapt the computational, memory, and bandwidth resources of the target platform. Second is network parallelism. *Deep³* breaks down the entire deep learning network into overlapped subnetworks using a depth-first method. Each subnetwork has the same depth as the original network with significantly fewer edges. Every subnetwork can be independently updated, and such local updates are periodically collected by a parameter coordinator to optimize the entire network. Third is data parallelism. *Deep³* decomposes the high-dimensional input data into several low-dimensional subspaces through dictionary learning. Dictionary learning can be efficiently performed by machine learning algorithms like spectral clustering [65–67]. Subsequently, each subnetwork is dedicated to handling a specific subspace and different subspaces are processed in parallel.

Wu et al. exploit mobile deep learning in the joint perspective of software-and-hardware architecture. They propose a platform named *DeepShark* to capacitate commercial-off-the-shelf (COTS) mobile devices with the capability of adaptive resource scheduling [68]. Methods like *DeepX* try to compress the deep model. By contrast, *DeepShark* seeks trade-off between response speed and memory consumption. It splits a pretrained DNN into code blocks and incrementally runs the blocks on system-on-chip (SoC) to accomplish inference. Consequently, *DeepShark* only needs to load currently required data from external storage into memory rather than hold entire data in memory throughout the execution period. Thus, *DeepShark* remarkably lowers memory consumption. In addition, *DeepShark* induces no accuracy loss due to the absence of model compression or approximation. Moreover, privacy risks are avoided due to the fact that all user-relevant data are processed locally. Eventually, *DeepShark* is transparent to deep learning developers. It overloads default system functions of TensorFlow and Caffe. Developers can invoke *DeepShark* APIs in the same way as calling TensorFlow or Caffe APIs. By contrast, the work of [59] eliminates redundant memory operations in an algorithmic manner.

3.5. Hardware Revolution. Haensch et al. point out that the aspiration to apply DL to all fields of daily life is an inheritance of pervasive computing. However, academia and industry are facing challenging barriers to scale DL to fit DL into pervasive applications [69]. Overhead is a vital problem regarding pervasive application of DL, where overhead refers to time and computational resources required to construct, train, and run the model. Prior-art research works show that GPUs take a step further towards pervasive DL, whereas it is confirmed that customized hardware dedicated to DL can outperform general-purpose GPUs.

Han et al. design a dedicated processor for DNN-based real-time object tracking [70]. This processor achieves low power consumption through a DNN-specific processor architecture and a specialized algorithm. However, this dedicated processor still relies on digital computing.

A DL network only requires limited kinds of mathematical operations (for example, matrix multiplication). And such operations frequently reoccur in model training or inference. These two characteristics enable efficient execution of DL algorithms on not only GPUs but also analog computing circuits. Additionally, DL algorithms are highly tolerant to noise and uncertainty, which opens a way to trade numerical precision for algorithmic accuracy. Analog computing discussed by Haensch et al. [69] is an extension of in-memory computing. Prior-art nonvolatile memory materials cannot efficiently accommodate analog in-memory computing. Reengineering memory materials is a challenging task. A new generation of DL accelerating hardware has entered the vision of academia and industry. This kind of hardware trades versatility for low overhead. Nevertheless, complexity of constructing and training DL models is beyond the capacity of any single kind of hardware. As a result, researchers need to consider the solution in a systematic perspective and aggregate several kinds of accelerators into a perfect system. Vitality of new accelerators heavily depends on this issue. Moreover, Haensch et al. declare that analog accelerators will not completely replace the digital ones. Both digital and analog accelerators should be continuously developed to the maximum possible extent. The analog accelerators should be capable of seamless integration into digital ones.

Analog computing can be implemented based on electrochemical reactions. Such a mechanism has been investigated to establish hardware foundations for DL-related problems. For example, neuromorphic computing can circumvent immanent performance bottlenecks of traditional computing via parallel processing and crossbar-memory-enabled data accessing. Fuller et al. link a redox transistor to a conductive-bridge memory (CBM) and thus establish an ionic floating-gate memory (IFG) array [71]. The working life of redox transistors can reach up to over one billion “read-write” operations. Additionally, data access frequencies can achieve more than one megahertz. This IFG-based neuromorphic system shows that in-memory learning and inference can efficiently perform based on low-voltage electrochemical systems. The adaptive electrical features of IFG can hopefully pioneer neuromorphic computers that can significantly outperform conventional digital computers in power efficiency. Such neuromorphic analog computers could adjust deep learning to power-limited context, or even capacitate persistent lifelong learning of a product. Another electrochemistry-based hardware prototype is proposed in [72]. Tsushiya et al. design a solid-state ionic device to address decision-making issues like the multiarmed bandit problem (MBPs). This device opens a way to achieve decision-making through motion of ions, which could contribute to mobile artificial chips and find various applications including deep learning.

In addition to analog computing, photonic (or optical) computing is also a promising hardware solution. Currently, mainstream photonic computers replace components of electric digital computers with photonic equivalents, which can achieve higher speed and bandwidth. Some pioneering research works have adopted photonic computing to support DL-related computations. Rios et al. achieve all-photonic in-memory computations through combining integrated optics with collocated data storage and processing [73]. They fabricate nonvolatile memory using the phase-change material $\text{Ge}_2\text{Sb}_2\text{Te}_5$ and perform direct scalar and matrix-vector multiplications based on this nonvolatile photonic memory. The computation results are represented by the output pulses. This photonic computing system offers a promising shift towards high-speed and large bandwidth on-chip photonic computing, which circumvents electro-optical conversions. Such a system could be the cornerstone of the purely photonic computers. Feldmann et al. point out that conventional computing architectures differentiate real neural tissue by physically separating the functionalities of data memory and processing [74]. This separated design places a daunting barrier to achieving high-speed and power-efficient computing systems like human brains. A promising solution to conquer this barrier is to elaborate novel hardware to simulate neurons and synapses of human brains. Consequently, they investigate on wavelength division multiplexing techniques to implement a photonic neural network based on a scalable circuit, which can mimic the neurosynaptic system in an all-optical manner. This circuit maintains the intrinsic high-speed and large bandwidth characteristics of an optical system and capacitates efficient execution of machine learning algorithms.

Quantum computing is another prospective solution to support DL. Gao et al. adopt a quantum generative model to design quantum algorithm of machine learning. This model enjoys superior ability of representing probability distributions over conventional generative models. In addition, the model can achieve a speedup of exponential magnitude at least in some application scenarios that a quantum computer cannot be fully simulated through conventional digital computing paradigm. The work of [75] opens a way to quantum machine learning and demonstrates a dramatic instance where a quantum algorithm of both theoretical and practical values can reach exponentially higher performance over conventional algorithms.

Novel hardware paradigms like ionic memory, photonic computing, and quantum computing could set indispensable stages for resource-limited deep learning. Despite that these hardware evolutions may be initially motivated by facilitating deep learning applications, the next-generation hardware could find much broader applications in future.

3.6. Discussion. Table 1 summarizes representative works in the perspective of underlying principles that account for the computational predicament of DNNs. Existing research works commonly aim at dealing with one or more of the causes of the computational predicament.

The first is memory overhead induced by oversized network. Earlier algorithmic solutions tend to compress or prune the weight matrix of a pretrained DNN. Compressing or pruning is a trade-off between the capacity (or generalization ability) and memory efficiency. However, directly modifying a pretrained network inevitably results in unexceptable error. Despite that retraining is a choice, it will induce remarkable extra time overhead.

As a result, recent algorithmic solutions propose to achieve a sparse network through training. The core idea is to elaborately select a regularization item for the error function, which forces the network to form sparse weight matrices yet at little or even no loss in generalization ability. In addition to algorithmic solutions, digital computers can also capacitate large pretrained networks in the inference phase through fine-grained utilization of memory.

The second is time or energy overhead induced by backpropagation, memory operations, and hyperparameter tuning. From the point of algorithmic view, dramatic redundant computation can be eliminated, especially in matrix-matrix or matrix-vector multiplications. In this manner, time overhead as well as energy consumption is reduced. Time efficiency can also be promoted by reusing intermediate results of convolution, parallelization on digital processors, and code fine-tuning on digital processors. Unlike overhead caused by arithmetic processing, time consumption induced by memory operations is difficult to handle. The reason is that traditional digital computers adopt von Neumann architecture and thus have independent processing and memory units. Due to the statistical and approximate nature of DNNs, Boolean logic minimization can contribute to the reduction of memory operations, as well as energy consumption. This solution achieves efficient performance in handwritten digital recognition. However, it confines the activation functions to be *sign* functions, which limits the generalization ability. Regarding energy-related hyperparameter tuning, mathematical methods like Gaussian process can point out a more efficient searching path in the parameter space, other than merely rely on human experience or even random searching.

Energy consumption is mainly caused by arithmetic processing and memory operations. Consequently, the latter two are key problems. Regarding time overhead, most existing solutions focus on periphery issues like redundant computations. However, the problem roots in stochastic gradient descent. The training time will drop dramatically if we could fabricate an improved gradient that can lead to convergence more rapidly. With regard to memory operation overhead, it is an inherent problem of the von Neumann architecture. Resolving this problem requires new computing paradigms like in-memory computing.

The third is the curse of dimension. Conventional solutions like weight matrix decomposition and data embedding can reduce the feature dimension. As far as we know, there are limited research works of feature dimension reduction in the computational-resource-limited context. Relevant topics are to be investigated.

It should be noted that the above discussed aspects are not isolated to each other. A systematic view may imply a

TABLE 1: Representative research works in the perspective of underlying principles.

	Representative research works	Techniques
Memory overhead induced by oversized network	[39]	Weight matrix compression of a pretrained network through clustering: merging similar functions in the hypothesis space
	[56]	Weight pruning of a pretrained network: removing the weights that contribute little to fitting functions in the hypothesis space
	[39, 58]	Sparse training: lasso regularization, structured sparsity regularization
	[68]	Computational optimization on digital computers: fine-grained utilization of memory
Time or energy overhead induced by backpropagation, memory operations, and hyperparameter tuning	[37, 39, 49]	Algorithmic design to avoid computation redundancy: depth separable convolution, avoidance of im2col reordering, factorized matrix-vector multiplication based on SVD and Tucker-2
	[37]	Caching of digital computers: reuse intermediate results of convolution to avoid redundant computation
	[39, 40]	Parallelization on digital processors: FPGA, GPGPU
	[37, 40, 53]	Full utilization of digital processors: profiling and fine-tuning of CPU or GPGPU codes
	[59]	Avoidance of frequent memory operations through Boolean logic minimization
Curse of dimension	[41]	Hyperparameter tuning using Gaussian process
	[53]	SVD decomposition of the weight matrix
	[60]	Data embedding

more efficient solution. For example, a pretrained sparser network undoubtedly demands less inference time than a denser network. Another instance, reading/writing weights will induce less time and energy consumption if the weight matrix is sparser. Table 1 does not cover innovative computing paradigms like analog computing and quantum computing. We will discuss such computing paradigms in more detail later.

Table 2 provides more details on the representative research works. Three categories of solutions are all under rapid development. The overall motivation is to apply DL to mobile/embedded context efficiently. Algorithmic solutions are at the core position due to the fact that they directly cope with business logic of real applications and aim to reduce time and memory complexity on the mathematical logic layer. Existing solutions mainly focus on simplifying matrix-and-vector operations, data/network embedding, hyperparameter tuning, and sparsification through regularization. Further research is still needed to explore reducing computational overhead through activation function.

In addition to the mathematical logic layer, traditional general-purpose digital hardware bridges the gap between mathematical algorithms and real applications. To the best of our knowledge, most practical mobile/embedded DL-based applications are based on traditional hardware. In this case, classical computational optimization methods can be adopted to fully utilize computational resources, including data caching, parallelization, and code fine-tuning. However, many existing DNNs are designed by AI experts, who place little or even no concern on the adaptiveness of DNNs to hardware. As a result, the DNNs may need some reshaping to efficiently fit into a specific hardware device. In

view of this, we expect that researchers can design DNNs in a joint view of both AI experts and computer engineers.

Currently, representative computational performance metrics include memory overhead, memory access latency, parallelism (full utilization of processors), and power consumption. However, some topics still remain to be investigated. For instance, *DeepShark* uses external storage as the cache to support fine-grained memory utilization. Power consumption caused by data I/O is to be discussed. In addition, the balance between cache size and cache hit rate is also an interesting topic. Table 3 shows the datasets that were used to evaluate a DNN in pursuant to more than one performance metrics. These datasets and relevant algorithms are favourable choices to serve as benchmarks.

Nevertheless, traditional general-purpose digital hardware may be still inefficient under certain scenarios. Consequently, DL-dedicated digital hardware is becoming increasingly popular, whereas the computational performances of digital hardware are facing bottleneck due to physical constraints. Next-generation computing technologies such as quantum computing are promising solutions to conquer such constraints. Next-generation computing technologies will undoubtedly boost the progress of deep learning even if they are now in their infancies.

4. Challenges to Be Addressed

Despite the promising prospect of existing solutions, we are still facing some considerable challenges to be addressed.

4.1. Fundamental Support for Hardware Revolution. Analog computing is a promising technology to facilitate DL due to the fact that DL is tolerant to numerical errors.

TABLE 2: Details of representative research works.

Date	Name (ref. no.)	Resource	Representative method	Architecture of NN or topic of machine learning	Application scenario	Dataset
2016-4-11	<i>DeepX</i> [53]	Memory capacity, power	Inference phase: SVD decomposition-based weight matrix compression, fine-grained task scheduling to processors	AlexNet [76], 2-hidden layer DNN for SpeakerID, SVHN CNN, 2-hidden layer DNN for Audio Scene	Recognition of objects, human voice, audio environment	ImageNet [76], Speaker Verification Spoofing, and Countermeasures Challenge Dataset [77], SVHN dataset [78], Audio Scene dataset [79]
2016-8-8	<i>DeLight</i> [60]	Power	Training phase: data projection under energy constraint	4-Layer DNN	Imaging, smart sensing, speech recognition	Hyperspectral Remote Sensing Scenes [80], UCI Daily and Sports Activities [81], UCI ISOLET [82]
2017-4-17	<i>MobileNets</i> [49]	Memory capacity	Training phase: depthwise separable convolution, avoidance of im2col reordering, hyperparameter tuning	A 28-layer convolution neural net, PlatNet [87, 88], FaceNet [89, 90]	Large-scale geolocation, fine-grained image recognition, face recognition, object detection	ImageNet, Im2GPS [83], Stanford Dogs [84], YFCC100M [85], COCO [86]
2017-4-30	[39]	Memory capacity	Inference phase: weight encoding, weight sharing, factorization of vector-matrix multiplication	2-Hidden layer DNN	Speech recognition, indoor localization, human activity recognition, handwritten digital recognition	UCI ISOLET, UCI UJIIndoorLoc [87], UCI Daily and Sports Activities, MNIST [88]
2018-3-19	<i>HyperPower</i> [41]	Power	Training phase: hyperparameter tuning, GP-Bayesian optimization	Variants of AlexNet for MNIST and CIFAR-10	Handwritten digital recognition, image classification	MNIST, CIFAR-10 [89]
2019-1-21	[59]	Memory access latency, power	Training phase: transform the DNN realization problem into a Boolean logic optimization problem, Boolean logic minimization	Multiple layer perception [92], CNN	Handwritten digital recognition	MNIST
2019-2-28	[52]	Memory capacity	Training phase: group lasso regularization, intergroup lasso regularization	Fully convolutional network with 7 convolution layer initialized with pretrained VGG16	Face recognition	LFW face dataset [93]
2019-4-12	[58]	Memory capacity	Training phase: structured sparsity regularization, Alternative Updating with Lagrange Multipliers (AULM)	LeNet [94], AlexNet, VGG-16 [95], ResNet-50 [96], GoogLeNet [97]	Handwritten digital recognition, image classification	MNIST, ImageNet

TABLE 2: Continued.

Date	Name (ref. no.)	Resource	Representative method	Architecture of NN or topic of machine learning	Application scenario	Dataset	
Computational	2017-6-18	<i>Deep³</i> [40]	Processor	Training and inference phases: enhancing parallelism through computing load granularity altering, network splitting through depth-first traversal methodology, data dimension reduction using dictionary learning, parallelizing with GPU	Establish an universal framework for fitting DL network into specific hardware, AlexNet was used as an example	Imaging, smart sensing, speech recognition	Hyperspectral Remote Sensing Scenes, UCI Daily and Sports Activities, UCI ISOLET
	2017-6-19	<i>DeepMon</i> [37]	Processor, power	Inference phase: data caching, hardware-specific code fine-tuning, Tucker-2 matrix decomposition	VGG-Verydeep-16 [95], YOLO [98]	Continuous vision application	ILSVRC2012 train dataset [99], Pascal VOC 2007 train dataset [100], UCF101 dataset [101], LENA dataset [102]
	2017-6-23	<i>MobiRNN</i> [64]	Processor	Inference phase: fine granularity code execution, parallelizing with GPU	LSTM model [103]	Smart sensing	Mobile phone sensor dataset [104]
	2019-2-1	<i>deepshark</i> [68]	Memory capacity	Inference phase: fine-grained memory utilization	VGG, CaffeNet [105], GoogLeNet, AlexNet	Imaging	ILSVRC2012
	2018-10-4	[70]	Computing power, power	Training and inference phases: a unified core architecture, binary feedback alignment (BFA), dynamic fixed-point-based run-length compression (RLC), dropout controller	MDNet [106]	Real-time object tracking	Object tracking benchmark (OTB) dataset [107]
Hardware	2018-9-7	[72]	Computing power	Adopt voltage-charge relationship of electrochemical cells to achieve forgetting parameters, describe the decision-making problem using motion of ions	Multiarmed bandit problems (MBPs)	Reinforcement learning	—
	2018-12-7	[75]	Computing power	Quantum computing, model the correlation in data with underlying probability amplitudes of a many-body entangled state	Generative model	Generative model	—
	2019-2-15	[73]	Computing power	Electrochemical cells	Matrix-vector multiplications based on nonvolatile photonic memory	Basic arithmetic operations for machine learning or AI algorithms	—
	2019-5-9	[74]	Processing power, power	Separating the functionalities of data memory and processing, mimic the neurosynaptic system in an all-optical manner	Neural network consisting of four neurons and sixty synapses (and 140 optical elements in total)	Letter recognition	—

TABLE 3: Datasets relevant to more than one performance metrics.

	Performance metric	Relevant research work
UCI Daily and Sports Activities, UCI ISOLET	Processor utilization rate	[40]
	Memory overhead	[39]
	Power consumption	[60]
ILSVRC2012	Processor utilization rate	[37]
	Power consumption	[37]
	Memory overhead	[68]
MNIST	Memory overhead	[39, 59]
	Memory access latency	[59]
	Power consumption	[41]
Hyperspectral Remote Sensing Scenes	Processor utilization rate	[40]
	Power consumption	[60]

However, analog computing is a kind of in-memory computing and raises the demand for novel nonvolatile memory material. Analog-computing-powered DL calls for long-term joint efforts of computer scientists and material scientists.

Compared to the other innovative types of hardware, analog computing is temporarily taking the leading position. The analog array technology has been successfully applied to DNN to processing common datasets [108], while other innovative hardware technologies such as photonic computing and quantum computing are still to be applied to DNN [73, 75]. Superiority of the analog array lies in the fact that it adopts analog circuit to compute matrix-vector multiplication with constant time overhead irrelevant to size of the matrix. However, it is a predicament to straightly map convolutional neural network onto conventional analog arrays due to the fact that kernel matrices are commonly small and the constant-time multiplication operation has to be iterated for many times in a sequential manner. Rasch et al. parallelize the training through duplicating the kernel matrix of a convolution layer on different analog arrays and stochastically dispatching parts of the computation onto the arrays. As a result, the speedup ratio is proportional to the amount of kernel matrices per layer [106].

In addition to the high speed-up ratio, another advantage of analog computing is the splitting of processing and memory. Under a traditional von Neumann architecture, processing units and memory are separate. Data transmission between processing units and memory can consume orders of magnitude more energy than conventional arithmetic operations. In addition, a typical deep learning application routinely demands enormous data transmission operations, which raises dramatically higher energy consumption than that of computation. One promising solution is collocating processing units and memory using phase-change memory [109].

Despite that analog computing hardware has exhibited promising potential to outperform traditional von Neumann architecture hardware like GPUs, most existing research works focus on the functionality of such analog hardware. Efficiency and reliability issues like stability and durability are yet to be investigated before moving out of the lab to real applications [110].

4.2. More Efficient Algorithmic Solutions. Some algorithmic solutions like weight matrix compression and weight matrix decomposition are approximating the original pretrained neural network with a simplified one. Nevertheless, empirical nature of DL hinders solving an exact theoretical upper bound of approximation error. The absence of this upper bound makes it difficult to prove the robustness of such approximations. Additionally, due to the lack of theoretical principles, many algorithmic techniques require iterative tuning and running the model to select the optimal one. Nevertheless, the design space of model parameters is large. As a result, implementing such algorithmic techniques to large-scale real applications may be a daunting task, especially when we need to deal with hyperparameters within large ranges.

Posttraining simplification of the DNN may result in large error. Moreover, a large number of parameters hinder the stochastic gradient descent to achieve a near-optimal solution. Sparse training is a promising method to cope with these two problems.

Achieving high capacity of a deep neural network is a conventional solution to guarantee low generalization error. However, most deep neural networks obtain high capacity through harnessing a large number of weights, which means dense connections between consecutive layers. This explains the reason that many existing deep neural networks adopt fully connected layers. Nevertheless, real biological scale-free neural networks can significantly outperform state-of-the-art deep learning networks yet with sparse connections. Inspired by this observation, Mocanu et al. construct a sparse scale-free network topology with two consecutive layers [111]. This topology substitutes sparse layers for fully connected layers before training. Their sparse evolutionary training method quadratically decreases the amount of parameters, inducing no loss in accuracy. This sparse training method opens a way to lower the barrier to fitting deep learning into traditional hardware.

Based on the method of [111], Liu et al. train a sparse MLP (multiple layer perception) model with a million neurons to classify microarray genes [110]. This MLP model can be trained within the time of 101 seconds magnitude and achieve lower generalization error than traditional models

(dataset: Leukemia, dimension: 26, 1397 training data samples and 699 testing data samples).

The method of [111] mainly focuses on building a novel network topology, yet still adopts conventional stochastic gradient descent to train the model [111]. Dettmers et al. harness exponentially smoothed gradients to recognize layers and weights that efficiently decrease the training error of a sparse model. As a result, the model can converge significantly faster. In addition, the trained network is insensitive to hyperparameters.

In recent years, a rapidly increasing number of research works are investigating on sparse training of DNN [112–116]. These research works typically concentrate on sparse training of several types of DNN. In view of the diversity and complexity of DNNs, it is a highly valuable yet challenging job to exploit sparse training for various types of DNNs under specific application requirements.

4.3. Systematic Integration. As discussed in Section 2, the ultimate goal of resource-limited DL is ubiquitous deployment of DL. Diversified applications can put forward various requirements on ubiquitous DL. As a result, we need to systematically integrate various types of solutions.

Next-generation computing hardware should seamlessly collaborate with traditional digital hardware, with the ultimate target of accommodating the tachytely evolving DNNs.

Gil and Green argue that the future computing hardware is based on intersections of three aspects: mathematics and information, neuron-inspired biology and information, and physics and information. These intersections give rise to the concepts of digital computing, neural computing, and quantum computing, respectively. Gil and Green denote the three concepts as bit, neuron, and qubit, respectively. As shown in Figure 4, the next-generation AI-enabled computing system requires integration of the three [117]. In this figure, we adopt quantum computing (qubits) to represent future computing paradigms. Novel computing paradigms like analog computing should be also taken into consideration. We discuss this integration in detail as follows.

4.3.1. Digital Computing. The advantage of digital computing lies in its stable binary nature. With the same binary input, a digital computing system should always generate the same output. This nature is the cornerstone of building robust and stable systems for data storage and processing. Classical digital computing is still an efficient solution to not only mathematical and logical operations but also persistent data storage. In the future computing system, digital computing will still occupy an indispensable position due to its robust and reliable nature.

4.3.2. Neuron Computing. Despite the advantages of digital computing, current DNN-based AI methods require reshaping or even innovating this computing paradigm. AI has achieved dramatic progress in the last decade. AI is still in the phase of narrow AI, which demands large amount of

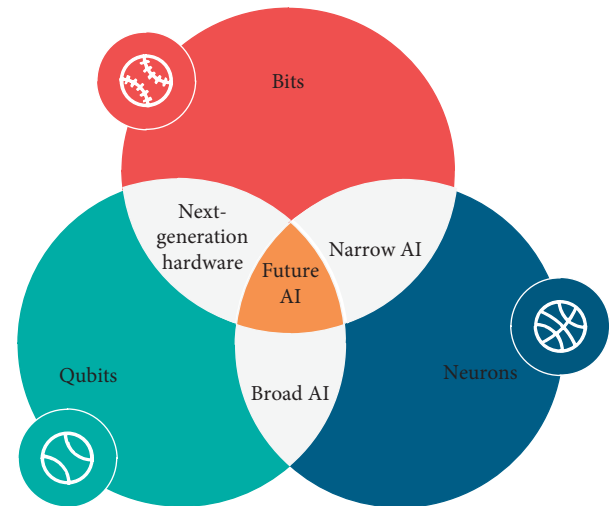


FIGURE 4: Road map to establish the next-generation AI-enabled computing systems.

manually labeled data to acquire knowledge of specialized tasks. In the next phase, we are expecting the broad AI that can adaptively and autonomously adapt to diversified tasks of various domains. Narrow AI is already computationally expensive in enormous scenarios. The vision of broad AI will even aggravate the computational predicament. Building efficient computing systems for such AI workload requires innovative reengineering of materials, architecture, and software.

The first category of solutions to AI-specific computing system stems from statistical and error-tolerant nature of deep learning. Such solutions sacrifice numerical precision for computational performance, yet generally achieve similar or even equivalent classification accuracy to the full-precision implementations [118–121]. We will witness a continuous decline in the precision demands of DNN training and inference in the coming decade. This trend is driven by the constant renovations of AI-specific digital hardware and matching algorithms, which will result in significant improvement in the performance of AI hardware.

As is previously discussed, another category of solutions lies in the idea of eliminating the overhead of data transmission between processing units and memory.

We can envision the high demands raise by DNN-based AI in the near future. Quantum computing enjoys the greatest computing power among almost all existing computing paradigms and thus has the potential to boost high-time-complexity deep learning applications that are knotty to the other computing paradigms.

4.3.3. Quantum Computing. Quantum computing generates an exponential state space of qubits (quantum bit) through exploring quantum superposition and entanglement. Computing power exponentially scales with the number of qubits: one additional qubit means doubled computing power. Prototypes of quantum computers have come out in the lab of hardware vendors like IBM [122, 123]. The next topic is to bridge the gap between the technical prototype

and real applications. For instance, quantum error correction (QEC) codes are indispensable for fault-tolerant quantum computing. Quantum computers will be a core accelerator of future AI-enabled computing systems. Nevertheless, currently, the cost of building a fault-tolerant quantum computing is beyond the reasonable range [124]. Further in-depth investigation is urgent.

4.3.4. Integration of Bits, Neurons, and Qubits. As aforementioned, a deep-learning-enabled computing system relies on three cornerstones: digital computing (bits), neural computing (neurons), and quantum computing (qubits). Systematic solutions to computational-resource-limited deep learning will require the integration of bits, neurons, and qubits. Bits can provide fundamental data storage and guarantee the robustness of underlying hardware. However, bits alone can only support programmed tasks for specific narrow purposes. Integrating neurons with bits generates narrow AI or even broad AI, which can not only distill insightful knowledge from unimaginably huge amount of data but also assist humans in a collaborative and more human-like manner. Various science and engineering problems are hopefully to be resolved with the assistance of AI. The core principle of a neural network is to search a function in the hypothesis space of the network and thus map a category of samples to a corresponding output label. Due to the large scale and complexity of science and engineering problems, a typical neural network necessarily requires a high capacity to generate a large hypothesis space. A large hypothesis space can possibly contribute to reducing the generalization error. Nevertheless, a large hypothesis space means more degree of freedom and demands a long time to let the stochastic-gradient-descent-impelled backpropagation find an approximation to the optimal solution. The exponentially scaling computing power just matches the time overhead of the similar order of magnitude.

Digital hardware like GPGPU and FPGA currently account for the mainstream accelerator of DNNs. Time-consuming manual fine-tuning of parallel code is an unavoidable operation to achieve optimal performance, with regard to every “DNN model-GPGPU type” pair. As a result, digital-hardware-accelerated DL is facing a barrier to efficient and agile programming. Moreover, the developing toolkit of analog-computing-enabled or quantum-computing-based deep learning is undoubtedly an essence when we someday handover analog computers or quantum computers to investigators, programmers, and computing resource providers.

5. Conclusion

In this paper, we investigate typical solutions of resource-limited deep learning and point out the open problems.

Existing solutions have achieved successes under specific scenarios. However, we expect future breakthroughs in the following two aspects. The first aspect is dedicated hardware.

Most existing solutions depend on general-purpose digital hardware. Dedicated hardware, which takes into consideration unique characteristics of deep learning, is a promising direction to achieve further performance enhancements. The second aspect is the theoretical principles of deep learning. Simplifying the DNN is almost an inevitable method to reduce resource consumption. Nonetheless, such methods currently rely on empirical and iterative tuning. Additionally, the robustness of simplification is not theoretically guaranteed. Clarifying the theoretical principles of deep learning will enable more efficient simplification and guarantee robustness.

Disclosure

The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Chunlei Chen conceptualized the study. Jiangyan Dai and Huihui Zhang were responsible for resources. Chunlei Chen and Peng Zhang prepared the original draft. Huixiang Zhang, Yugen Yi, and Yonghui Zhang reviewed and edited the manuscript.

Acknowledgments

This work was supported by the following research funds: the National Natural Science Foundation of China (31872847, 61471269, and 71661015), Industry-University Collaborative Education Program granted by Ministry of Education of China (201802217002), Planning Project of the 13th Five-year Plan of China Information Industry Association Education Branch (ZXXJ2019019), the Nature Science Foundation of Shandong Province (ZR2019PF023), the Higher Educational Science and Technology Program of Shandong Province (J18KA130 and J16LN56), the Science and Technology Development Program of Weifang (2019GX009, 2018GX004, and 2017GX002), the Key Technology Research and Development Program of Sichuan Province and Chengdu Municipality (szjj2015-054), the Doctoral Program of Weifang University (2016BS03 and 2015BS11), and the Science and Technology Benefiting People Plan Project of Weifang High Tech Zone (2019KJHM13).

References

- [1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.
- [2] H. W. Lin, M. Tegmark, and D. Rolnick, “Why does deep and cheap learning work so well?,” *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1223–1247, 2017.

- [3] P. P. Brahma, D. Wu, and Y. She, "Why deep learning works: a manifold disentanglement perspective," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 10, pp. 1997–2008, 2016.
- [4] Y. Bayle, M. Robine, and P. Hanna, "SATIN: a persistent musical database for music information retrieval and a supporting deep learning experiment on song instrumental classification," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 2703–2718, 2019.
- [5] B. Xu, R. Cai, Z. Zhang et al., "NADAQ: natural Language database querying based on deep learning," *IEEE Access*, vol. 7, pp. 35012–35017, 2019.
- [6] R. V. Swaminathan and A. Lerch, "Improving singing voice separation using attribute-aware deep network," in *Proceedings of the 2019 International Workshop On Multilayer Music Representation And Processing(MMRP)*, pp. 60–65, IEEE, Milano, Italy, 2019.
- [7] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1027–1034, 2019.
- [8] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: a social multimedia perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566–578, 2019.
- [9] Z. Huang, J. Tang, G. Shan, J. Ni, Y. Chen, and C. Wang, "An efficient passenger-hunting recommendation framework with multitask deep learning," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7713–7721, 2019.
- [10] M. Zeng, M. Li, Z. Fei et al., "A deep learning framework for identifying essential proteins by integrating multiple types of biological information," *IEEE/ACM transactions on computational biology and bioinformatics*, p. 1, 2019.
- [11] F. Pasa, V. Golkov, F. Pfeiffer, D. Cremers, and D. Pfeiffer, "Efficient deep network architectures for fast chest X-ray tuberculosis screening and visualization," *Scientific Reports*, vol. 9, no. 1, p. 6268, 2019.
- [12] K. Li, J. Daniels, and C. Liu, "Convolutional recurrent neural networks for glucose prediction," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 603–613, 2019.
- [13] A. Ramcharan, P. McCloskey, and K. Baranowski, "A mobile-based deep learning model for cassava disease diagnosis," *Frontiers in Plant Science*, vol. 10, p. 272, 2019.
- [14] M. Reichstein, G. Camps-Valls, B. Stevens et al., "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.
- [15] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 6, Article ID eaav1488, 2019.
- [16] W. Zheng, H. B. Wang, and Z. M. Zhang, "Multi-layer feed-forward neural network deep learning control with hybrid position and virtual-force algorithm for mobile robot obstacle avoidance," *International Journal of Control, Automation and Systems*, vol. 17, no. 4, pp. 1007–1018, 2019.
- [17] Y. J. Heo, D. Kim, and W. Lee, "Collision detection for industrial collaborative robots: a deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.
- [18] F. Niroui, K. Zhang, and Z. Kashino, "Deep reinforcement learning robot for search and rescue applications: exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.
- [19] F. Ding, Z. Zhang, Y. Zhou, X. Chen, and B. Ran, "Large-scale full-coverage traffic speed estimation under extreme traffic conditions using a big data and deep learning approach: case study in China," *Journal of Transportation Engineering, Part A: Systems*, vol. 145, no. 5, Article ID 05019001, 2019.
- [20] D. Mochizuki, Y. Abiko, T. Saito, D. Ikeda, and H. Mineno, "Delay-tolerance-based mobile data offloading using deep reinforcement learning," *Sensors*, vol. 19, no. 7, p. 1674, 2019.
- [21] H. Ye, G. Y. Li, and B. H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.
- [22] H. Ye and G. Y. Li, "Deep reinforcement learning based distributed resource allocation for V2V broadcasting," in *Proceedings of the 2018 14th International Wireless Communications And Mobile Computing Conference (IWCMC)*, pp. 440–445, Kansas City, MO, USA, 2018.
- [23] W. Li, C. W. Pan, R. Zhang et al., "AADS.: Augmented autonomous driving simulation using data-driven algorithms," *Science Robotics*, vol. 4, no. 28, Article ID eaaw0863, 2019.
- [24] S. M. Aldossari and K.-C. Chen, "Machine learning for wireless communication channel modeling: an overview," *Wireless Personal Communications*, vol. 106, no. 1, pp. 46–70, 2019.
- [25] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. Barth, "Deep reinforcement learning enabled self-learning control for energy efficient driving," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 67–81, 2019.
- [26] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, 2019.
- [27] K. Z. Haider, K. R. Malik, S. Khalid, T. Nawaz, and S. Jabbar, "Deepgender: real-time gender classification using deep learning for smartphones," *Journal of Real-Time Image Processing*, vol. 16, no. 1, pp. 15–29, 2019.
- [28] A. Esteva, A. Robicquet, B. Ramsundar et al., "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, no. 1, pp. 24–29, 2019.
- [29] E. Kanjo, M. G. Y. Eman, and S. A. Chee, "Deep learning analysis of mobile physiological, environmental and location sensor data for emotion detection," *Information Fusion*, vol. 49, pp. 46–56, 2019.
- [30] S. Chung, J. Lim, K. J. Noh, G. Kim, and H. Jeong, "Sensor data acquisition and multimodal sensor fusion for human activity recognition using deep learning," *Sensors*, vol. 19, no. 7, p. 1716, 2019.
- [31] F. Mehmood, I. Ullah, S. Ahmad, and D. Kim, "Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, 2019.
- [32] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, "DeepWear: adaptive local offloading for on-wearable deep learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 314–330, 2020.
- [33] A. Alelaiwi, "An efficient method of computation offloading in an edge cloud platform," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 58–64, 2019.

- [34] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 International Workshop on Internet of Things towards Applications—IoT-App '15*, pp. 7–12, ACM, Seoul, Korea, 2015.
- [35] R. Affolter, S. Eggert, T. Sieberth, M. Thali, and L. C. Ebert, "Applying augmented reality during a forensic autopsy—Microsoft HoloLens as a DICOM viewer," *Journal of Forensic Radiology and Imaging*, vol. 16, pp. 5–8, 2019.
- [36] C.-H. Wang, N.-H. Tsai, J.-M. Lu, and M.-J. J. Wang, "Usability evaluation of an instructional application based on Google Glass for mobile phone disassembly tasks," *Applied Ergonomics*, vol. 77, pp. 58–69, 2019.
- [37] L. N. Huynh, Y. Lee, and R. K. Balan, "Deepmon: mobile GPU-based deep learning framework for continuous vision applications," in *Proceedings Of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 82–95, ACM, Niagara Falls, NY, USA, 2017.
- [38] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization? Convergence properties of backpropagation," Technical Report, NEC Research Institute, Princeton, NJ, USA, 1998.
- [39] M. Dong, S. Wen, F. Zeng, Z. Yan, and T. Huang, "Sparse fully convolutional network for face labeling," *Neuro-computing*, vol. 331, no. 28, pp. 465–472, 2019.
- [40] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Deep³: leveraging three levels of parallelism for efficient deep learning," in *Proceedings of the 54th Annual Design Automation Conference 2017 on—DAC '17*, p.61, ACM, Austin, TX, USA, 2017.
- [41] D. Stamoulis, E. Cai, D.-C. Juan, and D. Marculescu, "HyperPower: power-and memory-constrained hyper-parameter optimization for neural networks," in *Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 19–24, IEEE, Dresden, Germany, 2018.
- [42] M. A. Hanif, M. U. Javed, R. Hafiz, S. Rehman, and M. Shafique, "Hardware-software approximations for deep neural networks," in *Approximate Circuits*, pp. 269–288, Springer, Berlin, Germany, 2019.
- [43] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: a review," *IEEE Access*, vol. 7, pp. 7823–7859, 2018.
- [44] D. Shin and H.-J. Yoo, "The heterogeneous deep neural network processor with a non-von Neumann architecture," *Proceedings of the IEEE*, pp. 1–16, 2019.
- [45] F. Schuiki, M. Schaffner, F. K. Gurkaynak, and L. Benini, "A scalable near-memory architecture for training deep neural networks on large in-memory datasets," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 484–497, 2019.
- [46] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "Neurostream: scalable and energy efficient deep learning with smart memory cubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 420–434, 2018.
- [47] H. Fuketa, H. Fuketa, T. Ikegami et al., "Image-classifier deep convolutional neural network training by 9-bit dedicated Hardware to realize validation accuracy and energy efficiency superior to the half precision floating point format," in *Proceedings Of the 2018 IEEE International Symposium On Circuits And Systems (ISCAS)*, pp. 1–5, IEEE, Florence, Italy, 2018.
- [48] H. Tann, S. Hashemi, and S. Reda, "Lightweight deep neural network accelerators using approximate SW/HW techniques," in *Approximate Circuits*, pp. 289–305, Springer, Cham, Switzerland, 2019.
- [49] A. G. Howard, M. Zhu, B. Chen et al., "Mobilenets: efficient convolutional neural networks for mobile vision applications," 2017, <https://arxiv.org/abs/1704.04861>.
- [50] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings Of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, IEEE, Honolulu, HI, USA, 2017.
- [51] A. Vasudevan, A. Anderson, and D. Gregg, "Parallel multi channel convolution using general matrix multiplication," in *Proceedings of the 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 19–24, IEEE, Seattle, WA, USA, July 2017.
- [52] M. Dong, S. Wen, Z. Zeng, Z. Yan, and T. Huang, "Sparse fully convolutional network for face labeling," *Neuro-computing*, vol. 331, no. 28, pp. 465–472, 2019.
- [53] N. D. Lane, S. Bhattacharya, P. Georgiev et al., "Deepx: a software accelerator for low-power deep learning inference on mobile devices," in *Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, p. 23, IEEE, Vienna, Austria, 2016.
- [54] S. Ge, Z. Luo, Q. Ye, and X.-Y. Zhang, "MicroBrain: compressing deep neural networks for energy-efficient visual inference service," in *Proceedings of the 2017 IEEE Conference On Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1000–1001, IEEE, Atlanta, GA, USA, 2017.
- [55] C. Deng, S. Liao, Y. Xie, K. K. Parhi, X. Qian, and B. Yuan, "PermDNN: efficient compressed DNN architecture with permuted diagonal matrices," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Micro-architecture (MICRO)*, pp. 189–202, IEEE, Fukuoka, Japan, 2018.
- [56] W. Yang, L. Jin, S. Wang, Z. Cu, X. Chen, and L. Chen, "Thinning of convolutional neural network with mixed pruning," *IET Image Processing*, vol. 13, no. 5, pp. 779–784, 2019.
- [57] R. Yazdani, M. Riera, J.-M. Arnau, and A. Gonzalez, "The dark side of DNN pruning," in *Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium On Computer Architecture (ISCA)*, pp. 790–801, IEEE, Los Angeles, CA, USA, 2018.
- [58] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Towards compact ConvNets via structure-sparsity regularized filter pruning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 574–588, 2019.
- [59] M. Nazemi, G. Pasandi, and M. Pedram, "Energy-efficient, low-latency realization of neural networks through boolean logic minimization," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference on - ASPDAC '19*, pp. 274–279, ACM, Tokyo, Japan, 2019.
- [60] B. D. Rouhan, A. Mirhoseini, and F. Koushanfar, "DeLight," in *Proceeding of the 2016 ACM/IEEE International Symposium On Low Power Electronics And Design*, pp. 112–117, ACM, San Francisco, CA, USA, 2016.
- [61] J. Wang, A. Hertzmann, and D. J. Fleet, "Gaussian process dynamical models," *Advances in Neural Information Processing Systems*, vol. 19, pp. 1441–1448, 2006.
- [62] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: a review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

- [63] P. P. Markopoulos, D. G. Chachlakis, and E. E. Papalexakis, "The exact solution to rank-1 L1-norm TUCKER2 decomposition," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 511–515, 2018.
- [64] Q. Cao, N. Balasubramanian, and A. Balasubramanian, "MobiRNN: efficient recurrent neural network execution on mobile GPU," in *Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications—EMDL '17*, pp. 1–6, ACM, Niagara Falls, New York, USA, 2017.
- [65] L. Jing, M. K. Ng, and T. Zeng, "Dictionary learning-based subspace structure identification in spectral clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 8, pp. 1188–1199, 2013.
- [66] L. He and H. Zhang, "Iterative ensemble normalized cuts," *Pattern Recognition*, vol. 52, pp. 274–286, 2016.
- [67] L. He, N. Ray, Y. Guan, and H. Zhang, "Fast large-scale spectral clustering via explicit feature mapping," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 1058–1071, 2019.
- [68] C. Wu, L. Zhang, Q. Li, Z. Fu, W. Zhu, and Y. Zhang, "Enabling flexible resource allocation in mobile deep learning systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 2, pp. 346–360, 2019.
- [69] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: analog computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, 2019.
- [70] D. Han, J. Lee, J. Lee, and H.-J. Yoo, "A low-power deep neural network online learning processor for real-time object tracking application," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1794–1804, 2019.
- [71] E. J. Fuller, S. T. Keene, A. Melianas et al., "Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing," *Science*, vol. 364, no. 6440, pp. 570–574, 2019.
- [72] T. Tsuchiya, T. Tsuruoka, S. J. Kim et al., "Ionic decision-maker created as novel, solid-state devices," *Science Advances*, vol. 4, no. 9, Article ID eaau2057, 2018.
- [73] C. Rios, N. Youngblood, Z. Cheng et al., "In-memory computing on a photonic platform," *Science Advances*, vol. 5, no. 2, Article ID eaau5759, 2019.
- [74] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, "All-optical spiking neurosynaptic networks with self-learning capabilities," *Nature*, vol. 569, no. 7755, pp. 208–214, 2019.
- [75] X. Gao, Z. Y. Zhang, and L. M. Duan, "A quantum machine learning algorithm based on generative models," *Science Advances*, vol. 4, no. 12, Article ID eaat9004, 2018.
- [76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.
- [77] W. Zhizheng, *Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof 2015) Database*, University of Edinburgh, The Centre for Speech Technology Research (CSTR), Edinburgh, UK, 2015.
- [78] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Granada, Spain, December 2011.
- [79] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene detection," 2014, <https://arxiv.org/abs/1508.04909>.
- [80] Remote Sensing, 2015, <http://www.ehu.es/ccwintco/index.php/HyperspectralRemoteSensingScenes>.
- [81] UCI machine learning repository, 2015, <https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities>.
- [82] UCI Machine Learning Repository, 2015, <https://archive.ics.uci.edu/ml/datasets/isolet>.
- [83] J. Hays and A. Efros, "IM2GPS: estimating geographic information from a single image," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, June 2008.
- [84] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *Proceedings of the First Workshop On Fine-Grained Visual Categorization, IEEE Conference On Computer Vision And Pattern Recognition*, Colorado Springs, CO, USA, June 2011.
- [85] B. Thomee, B. Elizalde, D. A. Shamma et al., "Yfcc100M," *Communications of the ACM*, vol. 59, no. 2, pp. 64–73, 2016.
- [86] J. Huang, V. Rathod, C. Sun et al., "Speed/accuracy trade-offs for modern convolutional object detectors," 2016, <https://arxiv.org/abs/1611.10012>.
- [87] UCI Machine Learning Repository, <https://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc>.
- [88] T. Weyand, I. Kostrikov, and J. Philbin, "PlaNet-photo geolocation with convolutional neural networks," *Computer Vision—ECCV 2016*, vol. 9912, pp. 37–55, Springer, Berlin, Germany, 2016.
- [89] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998.
- [90] F. Schroff, D. Kalenichenko, J. Philbin et al., "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, IEEE, Boston, MA, USA, June 2015.
- [91] The CIFAR-10 Dataset, <http://www.cs.toronto.edu/kriz/cifar.html>.
- [92] H. Bourlard and C. J. Wellekens, "Links between Markov models and multilayer perceptrons," *Advances in Neural Information Processing Systems*, vol. 1, pp. 502–510, 1988.
- [93] Labeled Faces in the Wild, <http://vis-www.cs.umass.edu/lfw/>.
- [94] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [95] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, <https://arxiv.org/abs/1409.1556>.
- [96] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, IEEE, Las Vegas, NV, USA, 2016.
- [97] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, IEEE, Boston, MA, USA, 2015.
- [98] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," 2015, <https://arxiv.org/abs/1506.02640>.
- [99] O. Russakovsky, J. Deng, H. Su et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [100] PASCAL VOC2007, https://dbcollection.readthedocs.io/en/latest/datasets/pascal_voc2007.html.

- [101] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: a dataset of 101 human actions classes from videos in the wild," 2012, <https://arxiv.org/abs/1212.0402>.
- [102] S. Song, V. Chandrasekhar, N.-M. Cheung, S. Narayan, L. Li, and J.-H. Lim, "Activity recognition in egocentric life-logging videos," in *Proceedings of the Asian Conference On Computer Vision*, pp. 445–458, Springer, Singapore, 2014.
- [103] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, <https://arxiv.org/abs/1409.2329>.
- [104] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, 2013.
- [105] BVLC CaffeNet Model, https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.
- [106] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," 2015, <https://arxiv.org/abs/1510.07945>.
- [107] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.
- [108] M. J. Rasch, T. Gokmen, M. Rigotti et al., "RAPA-ConvNets: modified convolutional networks for accelerated training on architectures with analog arrays," *Frontiers in Neuroscience*, vol. 13, p. 753, 2019.
- [109] A. Sebastian, M. Le Gallo, and E. Eleftheriou, "Computational phase-change memory: beyond von Neumann computing," *Journal of Physics D: Applied Physics*, vol. 52, no. 44, Article ID 443002, 2019.
- [110] E. A. Cartier, W. Kim, N. Gong et al., "Reliability challenges with materials for analog computing?," in *Proceedings of the 2019 IEEE International Reliability Physics Symposium (IRPS)*, pp. 1–10, IEEE, Monterey, CA, USA, 2019.
- [111] D. C. Mocanu, E. Mocanu, P. Stone et al., "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Communications*, vol. 9, no. 1, p. 2383, 2018.
- [112] S. Liu, D. C. Mocanu, A. R. R. Matavalam et al., "Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware," 2019, <https://arxiv.org/abs/1901.09181>.
- [113] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: faster training without losing performance," 2019, <https://arxiv.org/abs/1907.04840>.
- [114] R. Moradi, R. Berangi, and B. Minaei, "SparseMaps: convolutional networks with sparse feature maps for tiny image classification," *Expert Systems with Applications*, vol. 119, pp. 142–154, 2019.
- [115] R. Ma, J. Miao, L. Niu, and P. Zhang, "Transformed ℓ_1 regularization for learning sparse deep neural networks," *Neural Networks*, vol. 119, pp. 286–298, 2019.
- [116] J. Yang and J. Ma, "Feed-forward neural network training using sparse representation," *Expert Systems with Applications*, vol. 116, pp. 255–264, 2019.
- [117] D. Gil and W. M. J. Green, "The future of computing: bits + neurons + qubits," 2019, <https://arxiv.org/abs/1911.08446>.
- [118] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalkrishnan, "Training deepneural networks with 8-bit floating point numbers," in *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montreal, Canada, 2018.
- [119] C. Sakr and N. Wang, "Accumulation bit-width scaling for ultra-low precision training of deep networks," in *Proceedings of the International Conference On Learning Representations*, New Orleans, LA, USA, 2019.
- [120] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalkrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," in *Proceedings of the 2nd SysMLConference*, Stanford, CA, USA, 2019.
- [121] K. Gopalkrishnan, "DNN training and inference with hyper-scaled precision," in *Proceedings of the of Joint Workshop On-Device Machine Learning and Compact Deep Neural Network Representations*, Long Beach, CA, USA, 2019.
- [122] IBM, *IBM Q Experience*, IBM, Armonk, NY, USA, <https://www.ibm.com/quantumcomputing/technology/experience>.
- [123] IBM, *Quantum Computation Center Opens*, IBM, Armonk, NY, USA, 2019, <https://www.ibm.com/quantum-computing/technology/experience>.
- [124] C. Vuillot, H. Asasi, Y. Wang et al., "Quantum error correction with the toric Gottesman-Kitaev-Preskill code," *Physical Review A*, vol. 99, no. 3, Article ID 032344, 2019.

Research Article

A Novel Image Classification Approach via Dense-MobileNet Models

Wei Wang ¹, Yutao Li ¹, Ting Zou ², Xin Wang ¹, Jieyu You ³, and Yanhong Luo ³

¹School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China

²Yiyang Branch, China Telecom Co., Ltd., Yiyang 413000, China

³Hunan Children's Hospital, Changsha 410000, China

Correspondence should be addressed to Jieyu You; yjy660111@sina.com and Yanhong Luo; mfxgz123@163.com

Received 1 September 2019; Accepted 12 December 2019; Published 6 January 2020

Guest Editor: Malik Jahan Khan

Copyright © 2020 Wei Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As a lightweight deep neural network, MobileNet has fewer parameters and higher classification accuracy. In order to further reduce the number of network parameters and improve the classification accuracy, dense blocks that are proposed in DenseNets are introduced into MobileNet. In Dense-MobileNet models, convolution layers with the same size of input feature maps in MobileNet models are taken as dense blocks, and dense connections are carried out within the dense blocks. The new network structure can make full use of the output feature maps generated by the previous convolution layers in dense blocks, so as to generate a large number of feature maps with fewer convolution cores and repeatedly use the features. By setting a small growth rate, the network further reduces the parameters and the computation cost. Two Dense-MobileNet models, Dense1-MobileNet and Dense2-MobileNet, are designed. Experiments show that Dense2-MobileNet can achieve higher recognition accuracy than MobileNet, while only with fewer parameters and computation cost.

1. Introduction

Computer image classification is to analyze and classify images into certain categories to replace human visual interpretation. It is one of the hotspots in the field of computer vision. Because the features are very important to classification, most of the researches on image classification focus on image feature extraction and classification algorithms. Traditional image features such as SIFT and HOG are designed manually. Convolutional neural networks have the ability of self-learning, self-adapting, and self-organizing; so, it can automatically extract features by using the prior knowledge of the known categories, and avoid the complicated process of feature extraction in traditional image classification methods. At the same time, the extracted features are highly expressive and efficient.

Deep convolutional neural network (CNN) has achieved significant success in the field of computer vision, such as image classification [1], target tracking [2], target detection [3], and semantic image segmentation [4, 5]. For example, in

the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012), Krizhevsky et al. won the championship with an AlexNet [1] model of about 60 million parameters and eight layers. In addition, VGG [6] with 16-layer, GoogleNet [7] with Inception as the basic structure, and ResNet [8] with residual blocks that can alleviate the problem of gradient disappearance have also achieved great success. However, the deep convolutional neural network itself is a dense computational model. The huge number of parameters, heavy computing load, and large number of memory access lead to huge power consumption, which makes it difficult to apply the model to portable mobile devices with limited hardware resources.

In order to apply the deep convolutional neural network model to real-time applications and low-memory portable devices, a feasible solution is to compress and accelerate the deep convolutional neural networks to reduce parameters, computation cost, and the power consumption. Denil et al. [9] proved that the parameters of deep convolutional neural network have a lot of redundancy, and these redundant

parameters have little influence on the classification accuracy. Denton et al. [10] found an appropriate low-rank matrix to estimate the information parameters of deep CNNs by singular value decompositions. The method requires high computational cost and more retraining to achieve convergence. Han et al. [11] deleted the unimportant connections in the pretrained network by parameter pruning, retrained and quantized the remaining parameters, and then encoded the quantized parameters by Hoffman coding to further reduce the compression rate. However, the method requires manual adjustment of superparameters. Chen et al. [12] used a low-cost Hash function to group the weights between the two adjacent layers into a Hash bucket for weight sharing, which reduces the storage of additional positions and realizes parameter sharing. Hinton et al. [13] compressed the network model by knowledge distillation, and extracted useful information. The useful information is migrated to a smaller and simpler network, which made the simple network and the complex network have similar performance.

In addition, many related researches have improved network models to compress networks. For example, SqueezeNet [14] is a network model based on fire module, MobileNets [15] is a network model based on depthwise separable filters, and ShuffleNet [16] is improved on the basis of residual structure by introducing group pointwise convolution and channel shuffle operation.

Compared with VGG-16 network, MobileNet is a lightweight network, which uses depthwise separable convolution to deepen the network, and reduce parameters and computation. At the same time, the classification accuracy of MobileNet on ImageNet data set only reduces by 1%. However, in order to be better applied to mobile devices with limited memory, the parameters and computational complexity of the MobileNet model need to be further reduced. Therefore, we use dense blocks as the basic unit in the network layer of MobileNet. By setting a small growth rate, the model has fewer parameters and lower computational cost. The new models, namely Dense-MobileNets, can also achieve high classification accuracy.

2. Fundamental Theory

2.1. MobileNet. MobileNet is a streamlined architecture that uses depthwise separable convolutions to construct lightweight deep convolutional neural networks and provides an efficient model for mobile and embedded vision applications [15]. The structure of MobileNet is based on depthwise separable filters, as shown in Figure 1.

Depthwise separable convolution filters are composed of depthwise convolution filters and point convolution filters. The depthwise convolution filter performs a single convolution on each input channel, and the point convolution filter combines the output of depthwise convolution linearly with $1 * 1$ convolutions, as shown in Figure 2.

2.2. Dense Connection. DenseNet [17] proposed a new connection mode, connecting each current layer of the

network with the previous network layers, so that the current layer can take the output feature maps of all the previous layers as input features. To some extent, this kind of connection can alleviate the problem of gradient disappearance. Since each layer is connected with all the previous layers, the previous features can be repeatedly used to generate more feature maps with less convolution kernel.

DenseNet takes dense blocks as basic unit modules, as shown in Figure 3. In Figure 3, a dense block structure consists of 4 densely connected layers with a growth rate of 4. Each layer in this structure takes the output feature maps of the previous layers as the input feature maps. Different from the residual unit in ResNet [8], which combines the sum of the feature maps of the previous layers in one layer, the dense block transfers the feature maps to all the subsequent layers, adding the dimension of the feature maps rather than adding the pixel values in the feature maps.

In Figure 4, the dense block only superimposes the feature maps of the previous convolution layers and increases the number of feature maps. Therefore, only the magnitude of x_l and x_{l+1} is required to be equal, and the number of feature maps does not need to be the same. DenseNet uses hyperparameter growth rate to control the number of feature map channels in the network. The growth rate k indicates that the output feature maps of each network layer is k . That is, for each convolution layer, the input feature maps of the next layer will increase k channels.

3. Dense-MobileNet

Dense-MobileNet introduces dense block idea into MobileNet. The convolution layers with the same size of input feature maps in MobileNet model are replaced as dense blocks, and the dense connections are carried out within the dense blocks. Dense block can make full use of the output feature maps of the previous convolution layers, generate more feature maps with fewer convolution kernels, and realize repeated use of features. By setting a small growth rate, the parameters and computations in MobileNet models are further reduced, so that the model can be better applied to mobile devices with low memory.

In this paper, we design two different Dense-MobileNet structures: Dense1-MobileNet and Dense2-MobileNet.

3.1. Dense1-MobileNet. MobileNet model is a network model using depthwise separable convolution as its basic unit. Its depthwise separable convolution has two layers: depthwise convolution and point convolution. Dense1-MobileNet model considers the depthwise convolution layer and the point convolution layer as two separate convolution layers, i.e., the input feature maps of each depthwise convolution layer in the dense block are the superposition of the output feature maps in the previous convolution layer, and so is the input feature maps of each deep convolution layer, as shown in Figure 5. Because depthwise convolution is a single channel convolution, the number of output feature maps of the middle depthwise convolution layer is the same

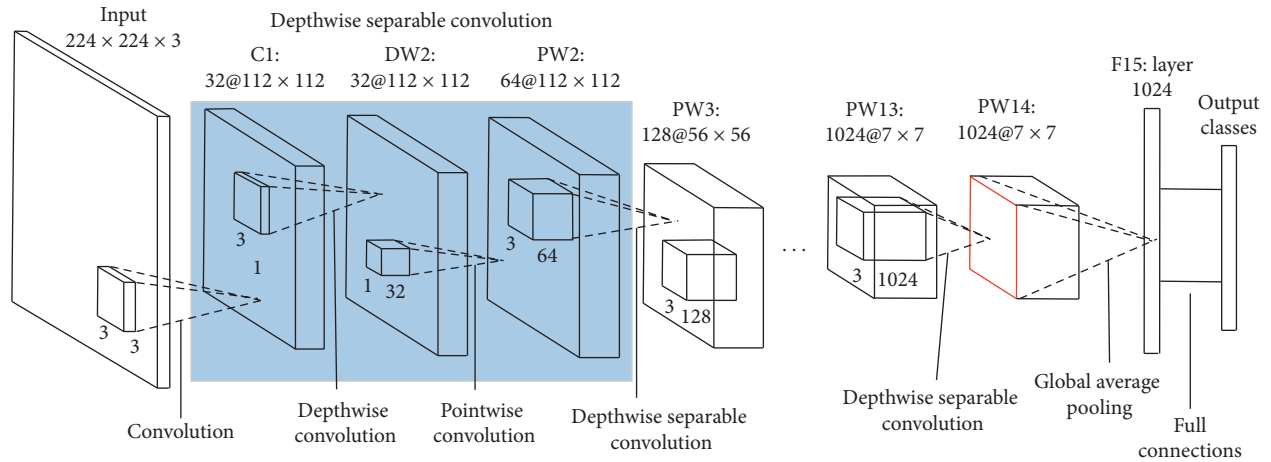


FIGURE 1: Architecture of MobileNet.

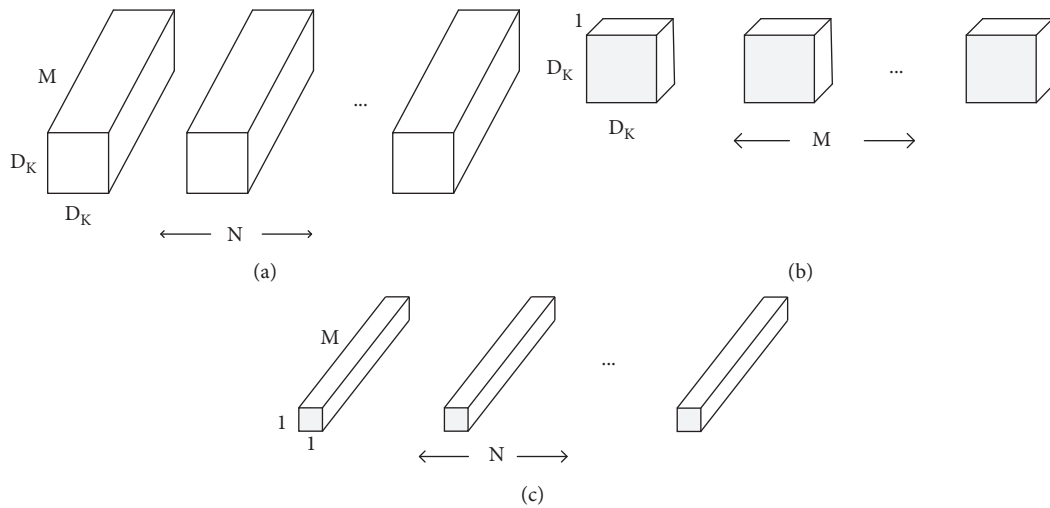


FIGURE 2: Standard convolutional filters and depthwise separable filters. (a) Standard convolutional filters, (b) depthwise convolutional filters, and (c) point convolutional filters.

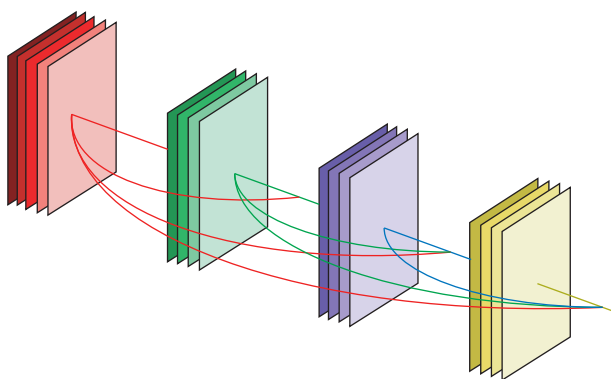


FIGURE 3: DenseNet structure.

as that of the input feature maps, which is the sum of the output feature maps of all the previous layers.

DenseNet contains a transition layer between two consecutive dense blocks. The transition layer reduces the number of input feature maps by using 1×1 convolution

kernel and halves the number of input feature maps by using 2×2 average pooling layer. The above two operations can ease the computational load of the network. Different from DenseNet, there is no transition layer between two consecutive dense blocks in Dense1-MobileNet model, the reason are as follows: (1) in MobileNet, batch normalization is carried out behind each convolution layer, and the last layer of the dense blocks is 1×1 point convolution layer, which can reduce the number of feature maps; (2) in addition, MobileNet reduces the size of feature map by using convolution layer instead of pooling layer, that is, it directly convolutes the output feature map of the previous point convolution layer with stride 2 to reduce the size of feature map.

3.2. Dense2-MobileNet. Dense2-MobileNet takes depthwise separable convolution as a whole, called a dense (depthwise separable convolution) block, which contains two point convolutional layers and a depthwise convolutional layer.

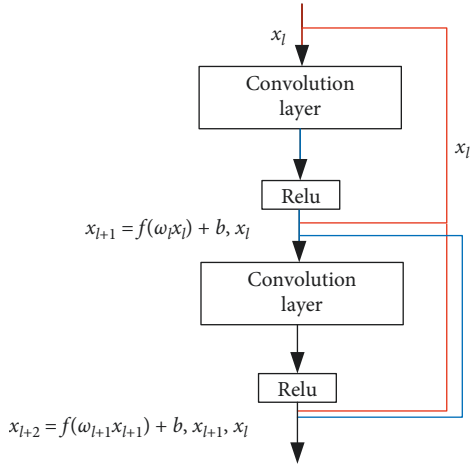


FIGURE 4: Two layers of dense connected modules.

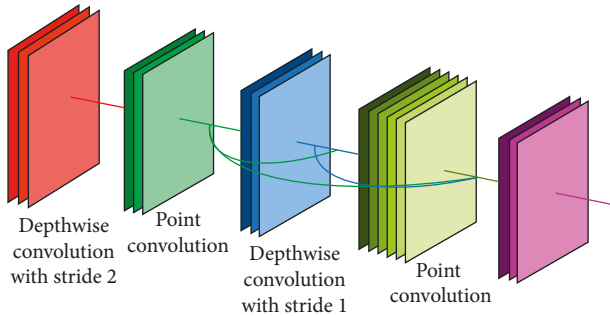


FIGURE 5: Schematic diagram of the Dense1-MobileNet model.

The input feature maps of depthwise separable convolution layer is the accumulation of output feature maps generated by point convolutions in all previous depthwise separable convolution layers, while the input feature map in point convolution layer is only the output feature map generated by the depthwise convolution in the dense block, not the superposition of the output feature maps of all the previous layers. So, the dense block structure in this model only has one dense connection, as shown in Figure 6.

In Dense2-MobileNet model, only one input feature map needs to overlay the output feature map of point convolution in the upper depthwise separable convolution layer. Because of the fewer cumulative times of structural feature maps, the number of output feature maps of all layers in a dense block is also less cumulative; so, it is not necessary to reduce the channel of feature maps by a $1 * 1$ convolution. After superimposing the output feature maps generated by the previous separable convolutions, the size of the feature map can be reduced by the depthwise convolution with stride 2; so, the Dense2-MobileNet model does not add other transition layers too. The MobileNet model is finally pooled globally and connected directly to the output layer. Experiments show that the classification accuracy of the global average prepooling depthwise separable convolution with dense connection before the global average pooling is higher than that of two-layer depthwise separable convolution without dense connection. Therefore, the depthwise separable

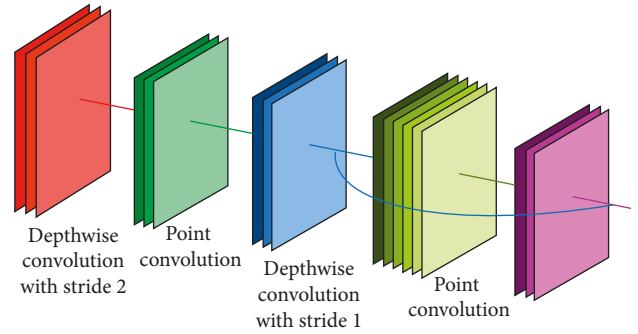


FIGURE 6: Diagram of the Dense2-MobileNet model.

convolution layer before global average pooling is also densely connected.

3.3. Dense-MobileNet Performance Analysis.

Dense-MobileNet model is constructed by adding dense connections in MobileNet. By setting a small hyper-parameter growth rate, it achieves less parameters and computational complexity than that in the MobileNet model. In the MobileNet model, every 2 depthwise separable convolution layers need to reduce the dimension of the feature map by depth convolution with stride of 2. Since the sizes of the input feature maps in same dense blocks need to be the same, there are only 2 depthwise separable convolution layers included in a dense block. The growth rate in Dense-MobileNet is set by using the least difference between the number of input feature maps of each layer in MobileNets and that in Dense-MobileNet. In fact, other optimal growth rates can be selected based on the balance between the compression rate and the accuracy rate of the model.

In this paper, the Dense1-MobileNet model decomposes depthwise separable convolution into 2 separate layers, and uses 4 convolutions as a dense block. The growth rate of dense blocks in Dense1-MobileNet is $\{32, 64, 64, 128, 128, 128, 256\}$. When the parameters of the Dense1-MobileNet model decrease to 1/2 of MobileNet, its calculation decreases to 5/11 of MobileNet.

The Dense2-MobileNet model takes depthwise separable convolution as a whole and 4 convolution layers as a dense block, but only one dense connection is used. The Dense2-MobileNet model has a growth rate of $\{32, 64, 128, 256, 256, 512\}$ for dense blocks. When its model parameters drop to 1/3 of MobileNet, its calculation decreases to 5/13 of MobileNet. The parameters and calculation of each model are shown in Table 1.

The DenseNet121 model in Table 1 contains 121 convolutional layers. With 16 as growth rate, the compression ratio of transition layer is set to 0.5. That is, all output feature maps in the previous dense block are used as input feature maps in transition layer, and the number of output feature maps in this layer is half of the number of input feature maps. As can be seen from Table 1, DenseNet121 model is affected by dense connections, which has fewer parameters but a large amount of computation. At the same time, the parameters and computations of the two improved

TABLE 1: The parameters and calculation of each model.

Network model	Calculations (millions)	Parameters number (millions)
DenseNet121	1364.7	1.78
MobileNet	568	3.21
Dense1-MobileNet	258	1.51
Dense2-MobileNet	217	1.12

Dense-MobileNets models are less than those of the MobileNet model.

4. Experiments and Result Analysis

In order to prove the validity of D-MobileNet models, we carry out classification experiments on Caltech-101 [18] and Uebingen Animals with Attributes, and compare the experimental results with those of the MobileNet model and the DenseNet121 model.

The Caltech-101 data set contains 9145 images in 102 classes, including 101 object classes and one background class. The number of images in each class ranges from 40 to 800. Figure 7 shows some samples in the Caltech-101 data set. In the experiments, the images in the data set are firstly labeled, and then fully scrambled. 1500 pictures are randomly selected as testing images, and the remaining pictures are used as training images.

The Uebingen Animals with Attributes database has 30475 pictures in 50 animal classes. Because the picture number is not the same in different classes, 21 largest animal classes with little difference in sample numbers are selected as our data set. There are 22742 pictures in the data set. The picture numbers in each class range from 850 to 1600. Figure 8 shows the samples in Uebingen Animals data set. Before training network, pictures in the data set are labeled and 2,000 of them are randomly selected as the test set. The rest of the pictures are used as the training data set.

The experiment uses Python language under TensorFlow framework. The model is implemented on a server equipped with NVIDIA TITAN GPU. RMSprop optimization algorithm with an initial learning rate of 0.1 is used to optimize the experiment. Depending on the number of training samples, we set different epoch numbers to reduce the learning rate. The weight initialization adopts the Xavier initialization method, which can determine the random initialization distribution range of parameters according to the number of inputs and outputs at each level. It is a uniform distribution with zero initial deviation. A total of 50,000 batches are trained, with 64 samples in each batch. ReLU is used as the activation function.

Table 2 shows the classification accuracy of four classification methods on the Caltech-101 data set. From Table 2, we can see that after 30,000 iterations, the accuracy of the 4 classification models has reached a balance, and the accuracy of our 2 improved structures is higher than that of DenseNet121. Compared with the accuracy of the standard MobileNet model, the accuracy of the Dense1-MobileNets

model is lower than that of the standard MobileNet model, while the accuracy of the Dense2-MobileNets model is higher than that of the standard MobileNet model. When the number of iterations is 50000, the accuracy of the Dense1-MobileNet model decreases by 0.13%, and the structure reduces less parameters and computation. When the number of iterations is 50000, the accuracy of the Dense2-MobileNet model increases by 1.2%, and its parameters and computation are reduced relatively.

Table 3 shows the classification accuracy of 4 classification methods on the Uebingen Animals data set. From Table 3, we can see that after 30,000 iterations, the accuracy of the 4 classification models also has reached a balance, and the accuracy of our 2 improved structures is higher than that of DenseNet121. Compared with the accuracy of the standard MobileNet model, the accuracy of the Dense1-MobileNets model is lower than that of the standard MobileNet model, while the accuracy of the Dense2-MobileNets model is higher than that of the standard MobileNet model. When the number of iterations is 5000, the accuracy of the Dense1-MobileNet model decreases by 0.1%, while the accuracy of the Dense2-MobileNet model increases by 1.2%.

The above two experiments were conducted under the same hyperparameter conditions. When the number of iterations is 5000, the classification accuracy of dense network on the Uebingen Animals data set is 0.4% higher than that of the MobileNet model, but it is 4.7% lower than that of the MobileNet model on the Caltech-101 data set. From the above two experiments, it can be seen that the classification accuracies of dense connection in the Dense1-MobileNet model are lost about 1% in both data sets, while they are improved in the Dense2-MobileNet mode. The main reason is that depthwise convolution and point convolution in depthwise separable convolution realize spatial correlation and channel correlation in standard convolution, respectively. However, Dense1-MobileNet using depthwise convolution and point convolution as the separate convolution layers will destroy channel correlation and reduce classification accuracy. The input feature map of the average pooling layer in Dense2-MobileNet is the superposition of the output feature maps of the previous 2 deep separable convolutions. It makes full use of the previous feature maps, reduces the parameters and computation, and improves the classification accuracy.

In order to further illustrate the performance of our method, we tested different methods in real data and other experimental environment. In the experimental comparison, we added the comparison with DenseNet161 and MobileNetV2 [19], and the experimental settings are shown in Table 4. The data set is our own children's colonoscopy polyp data set. There are two types of samples. One includes the samples with polyps, and the other includes the samples without polyp. As shown in Figure 9, the upper row is the samples with polyps, and the lower row is the samples without polyp.

The expanded training set contains 31450 samples, including 4005 polyp samples. The test set contains 4005 samples, including 1005 polyp samples. The size of each sample is $260 * 260$. The batch size of test set is set to 10, and



FIGURE 7: Samples in the Caltech-101 data set.

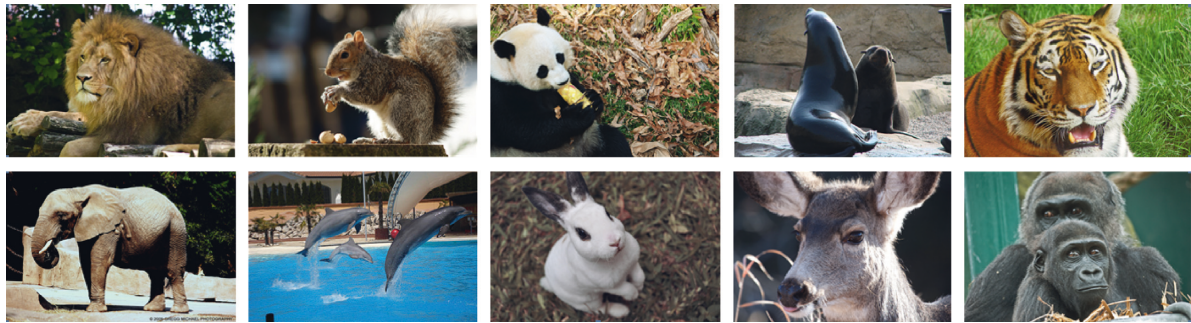


FIGURE 8: Samples in the Uebingen Animals (21) data set.

TABLE 2: Classification accuracy (%) on the Caltech-101 data set.

Number of iterations	30000	35000	40000	45000	50000
DenseNet	72.07	72.27	72.07	72	71.9
MobileNets	76.73	76.6	76.6	76.8	76.6
Dense1_MobileNet	76.6	76.53	76.47	76.4	76.47
Dense2_MobileNet	77.6	77.67	77.87	77.8	77.8

TABLE 3: Classification accuracy (%) on the Uebingen Animals data set (21classes).

Number of iterations	30000	35000	40000	45000	50000
DenseNet	91.85	92.15	91.95	92	92
MobileNets	91.6	91.6	91.6	91.55	91.6
Dense1_MobileNet	90.65	90.6	90.6	90.6	90.65
Dense2_MobileNet	92.1	92.05	92.1	92.05	92.05

TABLE 4: Experimental settings on children’s colonoscopy polyp data set.

Attribute	Configuration information
OS	Ubuntu 14.04.5 LTS
CPU	Intel® Xeon® CPU E5-2670 v3 @ 2.30 GHz
GPU	Nvidia GeForce GTX TITAN X
CuDNN	CuDNN 6.0.21
CUDA	CUDA 18.0.61
Framework	PyTorch

the initial learning rate is 0.1. Every network trains 200 epochs in total, and the learning rate decreases to half of the previous in the 50th epoch and then decays by half every 20 epoch. The average recognition accuracy of the last 100 epochs is taken as the final recognition result, as shown in Table 5.

Because there are only two types of test data sets, the classification accuracy of all methods is relatively high, all of which are over 96%. As can be seen from Table 5, the accuracy of Dense2_MobileNet (using full connection layer) is a little better than those of DenseNet121, MobileNet, and MobileNetV2, and slightly lower than that of DenseNet161. However, DenseNet161 is a deeper network with a large amount of parameters and calculation. In our experiments, the parameters and calculation of DenseNet161 are about 26.48 M and 10360.23 M, respectively, and the parameters of MobileNetV2 are about 2.23 M and 479.28 M, respectively. Although MobileNetV2 makes the network more lightweight, its parameter amount and calculation amount are still more than twice of our Dense_MobileNets. Therefore, the Dense_MobileNets still has certain advantages in the comprehensive evaluation of the

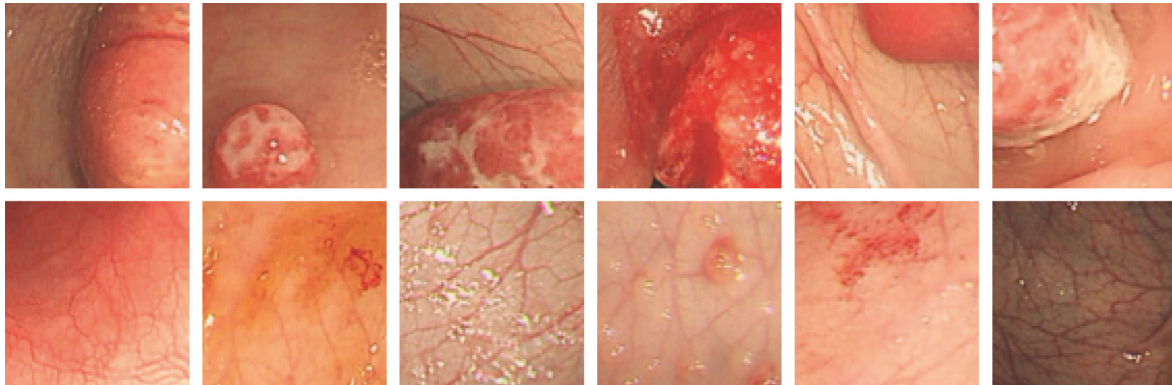


FIGURE 9: Samples in children's colonoscopy polyp data set.

TABLE 5: Classification accuracy (%) on children's colonoscopy polyp data set.

Network	Accuracy
DenseNet121	96.35
DenseNet161	96.57
MobileNet	96.45
MobileNetV2	96.46
Dense1_MobileNet	96.42
Dense2_MobileNet	96.48

accuracy of classification, the number of parameters, and the amount of calculation.

5. Conclusions

The memory intensive and highly computational intensive features of in deep learning restrict its application in portable devices. Compression and acceleration of network models will reduce the classification accuracy.

This paper introduces the Dense-MobileNet model with dense blocks for image classification. The dense blocks are used as the basic structure to improve the structure of MobileNet, and two improved models are proposed. These two models can reduce the parameters and calculation by setting the hyperparameter growth rate. At the same time, experiments show that Dense2-MobileNet can also increase the accuracy of classification. Compared with the MobileNet model, although the classification accuracy of Dense1-MobileNet is reduced, it reduces the number of parameters by at least half and the amount of calculation by nearly half. Generally speaking, the models proposed in this paper can be better applied to mobile devices.

Data Availability

All data sets are public data sets that can be downloaded online.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Defense Pre-Research Foundation of China (7301506), National Natural Science Foundation of China (61070040), Education Department of Hunan Province (17C0043), and Hunan Provincial Natural Science Fund (2019JJ80105).

References

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, MIT Press, Cambridge, MA, USA, 2012.
- [2] N. Wang and D. Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in Neural Information Processing Systems*, pp. 809–817, MIT Press, Cambridge, MA, USA, 2013.
- [3] W. Wang, C. Tang, X. Wang, Y. Luo, Y. Hu, and J. Li, "Image object recognition via deep feature-based adaptive joint sparse representation," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 8258275, 9 pages, 2019.
- [4] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "The development of convolution neural network and its application in image classification: a survey," *Optical Engineering*, vol. 58, no. 4, Article ID 040901, 2019.
- [5] F. Li, C. Wang, X. Liu, Y. Peng, and S. Jin, "A composite model of wound segmentation based on traditional methods and deep neural networks," *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 4967290, 1 pages, 2018.
- [6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015.
- [7] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Boston, MA, USA, June 2015.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.
- [9] M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 2148–2156, Lake Tahoe, NV, USA, December 2013.

- [10] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, pp. 1269–1277, MIT Press, Cambridge, MA, USA, 2014.
- [11] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding," 2015, <https://arxiv.org/abs/1510.00149>.
- [12] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proceedings of the International Conference on Machine Learning*, pp. 2285–2294, Lille, France, July 2015.
- [13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, <https://arxiv.org/abs/1503.02531>.
- [14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, <https://arxiv.org/abs/1602.07360>.
- [15] A. G. Howard, M. Zhu, B. Chen et al., "Mobilenets: efficient convolutional neural networks for mobile vision applications," 2017, <https://arxiv.org/abs/1704.04861>.
- [16] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: an extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, Salt Lake City, UT, USA, June 2018.
- [17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, Honolulu, HI, USA, July 2017.
- [18] F. Li, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," in *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop*, p. 178, Washington, DC, USA, June 2004.
- [19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT, USA, June 2018.

Research Article

Hand Gesture Recognition Based on Single-Shot Multibox Detector Deep Learning

Peng Liu,¹ Xiangxiang Li,² Haiting Cui,¹ Shanshan Li,¹ and Yafei Yuan ^{2,3}

¹Department of Sports Media and Information Technology, Shandong Sport University, Jinan 250102, Shandong, China

²Department of Optical Science and Engineering, Fudan University, 200433 Shanghai, China

³Department of Electronic Engineering, Fudan University, 200433 Shanghai, China

Correspondence should be addressed to Yafei Yuan; yyf@fudan.edu.cn

Received 20 July 2019; Revised 3 November 2019; Accepted 9 December 2019; Published 30 December 2019

Guest Editor: Malik Jahan Khan

Copyright © 2019 Peng Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Hand gesture recognition is an intuitive and effective way for humans to interact with a computer due to its high processing speed and recognition accuracy. This paper proposes a novel approach to identify hand gestures in complex scenes by the Single-Shot Multibox Detector (SSD) deep learning algorithm with 19 layers of a neural network. A benchmark database with gestures is used, and general hand gestures in the complex scene are chosen as the processing objects. A real-time hand gesture recognition system based on the SSD algorithm is constructed and tested. The experimental results show that the algorithm quickly identifies humans' hands and accurately distinguishes different types of gestures. Furthermore, the maximum accuracy is 99.2%, which is significantly important for human-computer interaction application.

1. Introduction

With the rapid development of computer technology and artificial intelligence, noncontact gesture recognition plays important roles in human-computer interaction (HCI) applications [1–4]. Due to its natural human-computer interaction characteristics, the hand gesture recognition system allows users to interact intuitively and effectively through a computer interface [5, 6]. Additionally, gesture recognition based on vision is widely applied in artificial intelligence, virtual reality, multimedia, and natural language communication [7–10].

However, traditional hand gesture recognition based on image processing algorithms was not widely applied in HCI because of its poor real-time capacity, low recognition accuracy, and complex algorithm. Recently, gesture recognition based on machine learning has been developed rapidly in HCI due to the introduction of the graphics processor unit (GPU) and the artificial intelligence (AI) image processing [11, 12]. The machine learning algorithms such as local orientation histogram, support vector machine (SVM) [13], neural network, and elastic graph matching are widely used in gesture

recognition systems [14–16]. Owing to its learning ability, the neural network does not need manual feature setting during the simulating human learning process and can carry out training the gesture samples to form a network classification recognition map [17, 18]. Deep learning models are inspired by information processing and communication patterns developed from biological nervous systems, which involve neural networks with more than one hidden layer. They can acquire the characteristics of the learning object easily and accurately under the complex object and exhibit superior performance in computer vision (CV) and natural language processing (NLP) [19–21]. Current state-of-the-art object detection systems are variants of Faster R-CNN [22]. The Single-Shot Multibox Detector (SSD) further optimizes object detection [23, 24]. As compared to Faster R-CNN, SSD is more simple and efficient as it completely eliminates proposal generation subsequent pixel and feature resampling stages, and it also encapsulates all computation in a single network which makes SSD easily trainable and straightforward to integrate into systems [5, 25–28].

This paper discusses hand gesture recognition in complex environments based on the Single-Shot Multibox

Detector. The approach is different from the work [28]. The image pyramid method is adapted to gesture recognition. More accurately, the system crops the image into blocks to detect far and small hand gestures. The experiment results show the SSD overcomes the interference signals in complex backgrounds and improves the accuracy and processing speed of gesture recognition.

2. Related Work

Generally, the process of vision-based hand gesture recognition system includes three steps which are hand segmentation, gesture model building, and hand gesture classification. To increase the efficiency, we simplify the process into two steps by using the SSD network. More precisely, we just need a convolutional neural network such as VGG16 [29] as a model system to identify the gesture features and then proceed with hand segmentation and gesture classification simultaneously by the SSD network. This makes our architecture much simpler and much faster than other methods based on the Faster R-CNN model.

The main purpose of gesture model building is to obtain useful semantic features, separate them from the complex backgrounds, and provide effective input information source for the following stage. In the stage of hand segmentation and hand gesture classification, hand postures with different sizes will be located with different bounding boxes. For these bounding boxes, simultaneously, we acquire the confidence for all gesture categories. Training is used for this unified framework to acquire an effective recognition model; recognition output is based on the model that has been trained to identify the gesture categories of input data. In other words, given an input image, we can acquire the location and classification score of hand gesture in this image end-to-end.

The standard hand gesture database is important for the hand gesture recognition system. Figure 1(a) shows the 36 hand gestures from the Massey University's 2D Static hand gesture image dataset which is about standard numbers and letters [30]. Note that some gestures are rather difficult to distinguish from each other. For example, "a" and "e," "d" and "l," "m" and "n," or "i" and "j." In this paper, we have chosen the characters of "w," "o," "r," and "k" as the study objects which are shown in Figure 1(b). The Canon EOS 6D camera was employed to capture the gesture with an EF 24–105mm/4L IS USM lens and a shutter time of 1/100 S. And the maximum distance is about five meters. Each hand gesture sample was obtained under three different complex backgrounds, aiming to prove the applicability and reliability of the hand gesture recognition system.

The hand gesture model building plays a vital role in a gesture recognition system that is regarded as the first step for processing the original input gestures. The inputs of this stage are images. When seeing an image, from the perspective of human beings, we can catch the sight of the scene described in the picture. However, the computer cannot capture these scenes from an original picture. The computer thinks an image is just a matrix with a variety of values in different spatial locations and channels. In other words, the computer can only obtain pixel-level information of an

image. Obviously, it is difficult to distinguish different objects using low-level information such as pixel values. Therefore, if we want to recognize hand gestures, one of the most efficient methods is extracting and summarizing high-level information such as their features and structures from the original image. This is exactly what gesture modeling does in our framework. We use the VGG16 convolutional neural network, which uses 13 convolutional layers and is deep enough to obtain high-level information of hand gestures. Given the original image as the input, the VGG16-Net will output feature maps of different resolutions which contain high-level information of the image. The reason for choosing 19 layers is that it is enough to extract high-level semantic information for classification and regression. And limited by the size of our dataset, using high-level layers can easily lead to overfitting.

The VGG-Nets are a series of convolutional neural networks with different depths which all use very smaller (3×3) convolution filters. The VGG16-Net (16 weight layers) is one of them which has 13 convolutional layers and 3 fully connected layers. The structure of VGG16 is shown in Figure 2. In this figure, the convolutional layer parameters are denoted as "conv < receptive field size > - < number of channels >." The ReLU activation function is not shown for brevity. The original image is passed through a stack of convolutional layers, which use filters with a small receptive field: 3×3 (which is the smallest size for capturing the notion of the left, right, up, down, and center). The convolution stride is fixed to 1 pixel; the spatial padding of a convolutional layer is such that the spatial resolution is preserved after convolution, i.e., the padding is 1 for 3×3 convolution filters. Spatial pooling is carried out by five maxpooling layers, which follow some of the convolutional layers (not all the convolutional layers are followed by maxpooling layers). Maxpooling is performed over a 2×2 pixel window, with stride 2.

All convolutional layers are equipped with the rectification nonlinearity (ReLU) [31]. After a stack of convolutional, maxpooling, and ReLU layers, we get feature maps with lower resolution and stronger semantic information. There are also fully connected layers and a soft max layer which are used for image classification in the original VGG16-Net. We replace these layers with SSD layers to implement hand segmentation and hand gesture classification.

The second stage, i.e., using the SSD network to perform hand segmentation and hand gesture classification, is the most important part in our framework. We have chosen the SSD model because it is both accurate and fast. The core of SSD is predicting category scores and bounding box offsets for a fixed set of default bounding boxes using very small (3×3) convolutional filters applied to feature maps. Beyond that SSD produces predictions of different scales from feature maps of different scales and separates predictions by aspect ratio. This architecture leads to simple end-to-end training and high accuracy, further improving the speed versus accuracy trade-off [5].

SSD is based on a feed-forward convolutional neural network (VGG16) that produces a fixed-size collection of bounding boxes and scores for the presence of object class

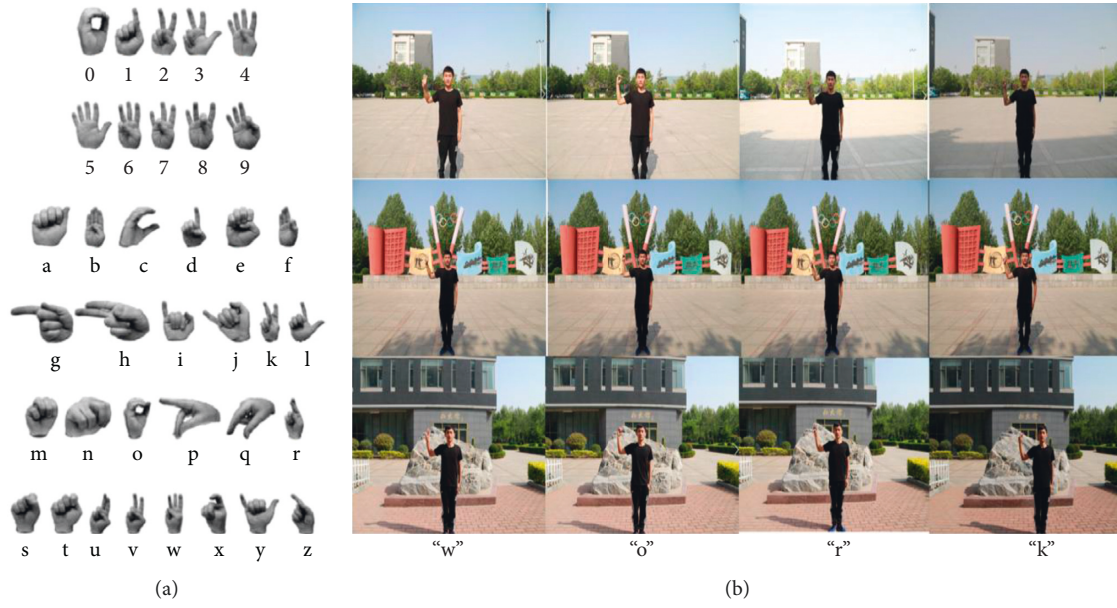


FIGURE 1: The graphs of hand gesture information: (a) standard library of hand gestures and (b) four hand gestures graphs in different backgrounds.

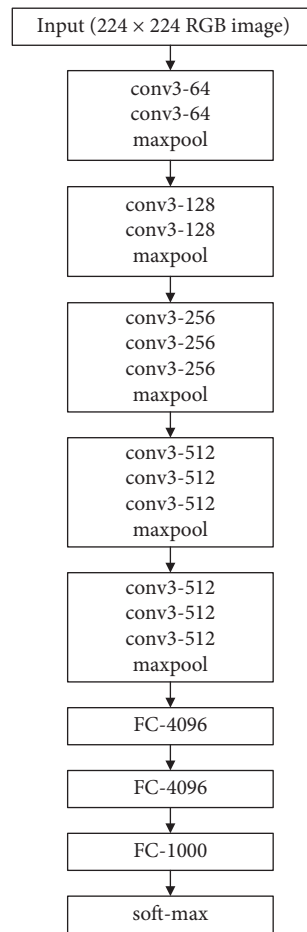


FIGURE 2: VGG16 Conv-Net structure.

instance in those boxes. This approach will produce a large number of bounding boxes, and most of them are covered by each other. Therefore, a nonmaximum suppression step

is executed to discard repetitive bounding boxes and produce the final detections. The structure of SSD is shown in Figure 3. The input image is an image with 300×300 pixels

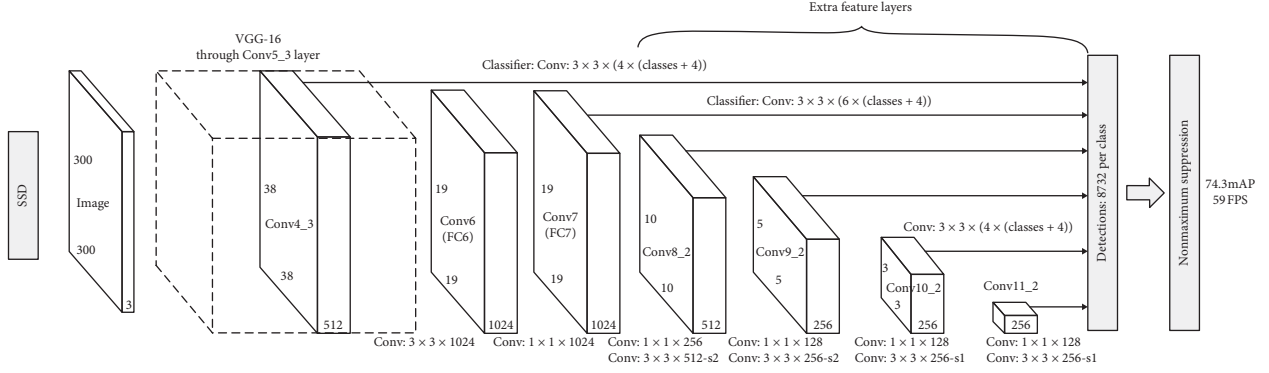


FIGURE 3: Structure of the SSD network.

and RGB channels. The part in the dotted box is the truncated VGG16 network. The SSD model adds several feature layers of different scales to the truncated VGG16 network. These layers decrease in size as depth increases and allow predictions of detections at multiple scales. Then, small convolutional filters apply to every position in selected feature maps. More precisely, these filters apply to a set of default boxes of different aspect ratios at each location in several selected feature maps to predict the shape offsets and the confident scores for all object categories. In our work, object categories include four hand gestures and the background.

Noting that we have the SSD framework, the next thing we need is an objective function to train the model end-to-end. The overall objective function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N} (L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)), \quad (1)$$

where N is the number of default boxes that match to ground truth boxes. The localization loss is a smooth $L1$ loss between the ground truth box (g) and the predicted box (l) parameters. These parameters are offsets for the center coordinate (cx , cy) of the default bounding box (d) and for its width (w) and height (h), which is similar to Faster R-CNN [22]:

$$L_{\text{loc}}(x, l, g) = \sum_{i \in \text{Pos}} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1} \left(l_i^m - \tilde{g}_j^m \right),$$

$$\tilde{g}_j^{cx} = \frac{(g_j^{cx} - d_i^{cx})}{d_i^w},$$

$$\tilde{g}_j^{cy} = \frac{(g_j^{cy} - d_i^{cy})}{d_i^h},$$

$$\tilde{g}_j^w = \log \left(\frac{g_j^w}{d_i^w} \right),$$

$$\tilde{g}_j^h = \log \left(\frac{g_j^h}{d_i^h} \right).$$
(2)

The confidence loss is the soft max loss over multiple class confidences (c), as is usually used in multiple classification tasks:

$$L_{\text{conf}}(x, c) = - \sum_{i \in \text{Pos}} x_{ij}^p \log(\tilde{c}_i^p) - \sum_{i \in \text{Neg}} \log(\tilde{c}_i^0), \quad \text{where } \tilde{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}. \quad (3)$$

During training, we match the default boxes to the ground truth boxes to calculate and reduce the loss of objective function. We do this recursively to optimize the parameters of the SSD model and finally get an ideal model. By using k-means clustering to guide the aspect ratio of anchor boxes, we get three different ratios. After that the ratios are 1.9, 1.6, and 1.1 with slight adjustment, respectively. Furthermore, the used optimizer is Adam with an initial learning rate of 0.0001.

3. Results and Discussion

The hand gesture recognition system was built by the SSD algorithm and training each character gesture with 1070 images with three different complex backgrounds. Then, we used 268 images which were not in the training set to test the building recognition model. The testing results of the recognition model on characters “w,” “o,” “r,” and “k” show good performance. In all 268 images, 261 of them are recognized correctly, with an accuracy of more than 93.8% and the highest recognition accuracy of 99.2%. The average prediction confidence for the 261 images recognized successfully is up to 0.96, which is very close to 1. Examples of visualization results are shown in Figures 4–7 with the character “w,” “o,” “r,” and “k,” respectively.

To evaluate the comprehensive performance of the gesture recognition system, the recognition accuracy for each hand gesture and response time was tested. The average accuracy of the gesture recognition system and response time are shown in Table 1. All the accuracies are more than 93.8%, and the character “o” owns higher accuracy. All

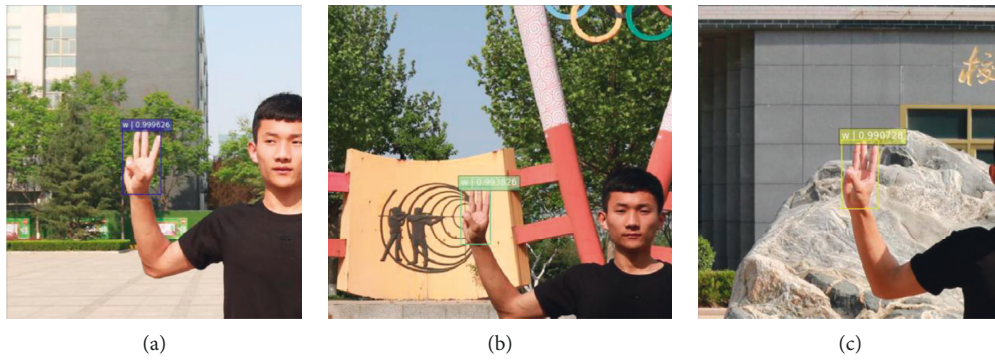


FIGURE 4: The automatic recognition result of character “w.”

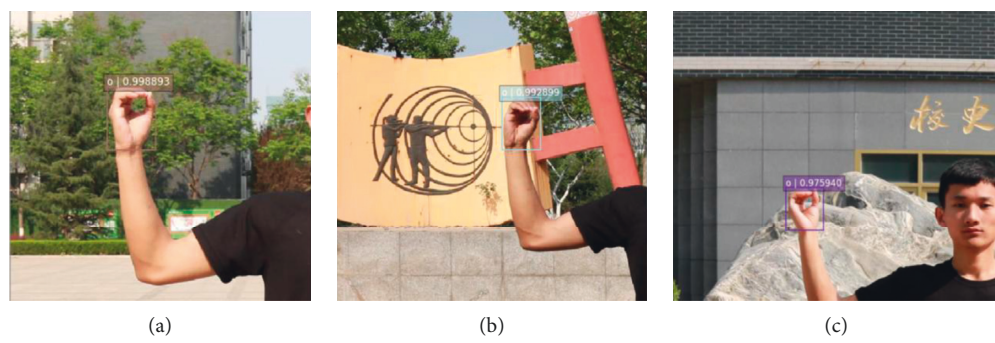


FIGURE 5: The automatic recognition result of character “o.”

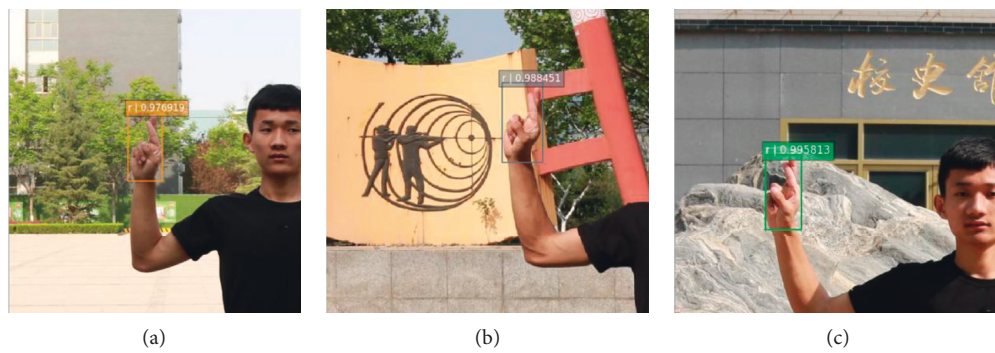


FIGURE 6: The automatic recognition result of character “r.”

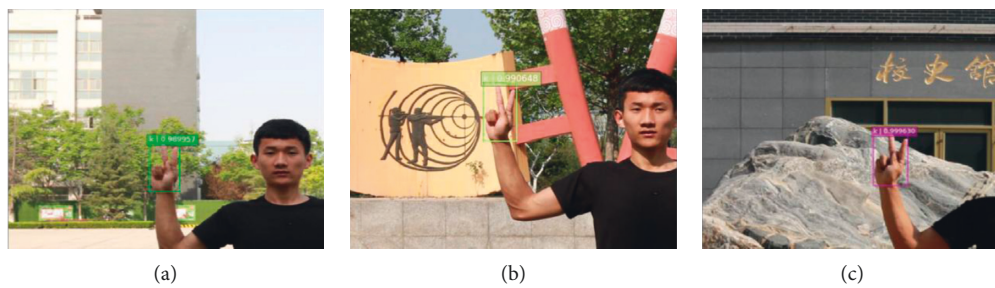


FIGURE 7: The automatic recognition result of character “k.”

TABLE 1: The accuracy and response time of the gesture recognition system.

Character	Accuracy	Response time (ms)
“w”	>96.9	17.1
“o”	>99.1	18.3
“r”	>93.8	18.2
“k”	>98.6	19.7

TABLE 2: Recognition accuracy comparison for the English alphabets.

Works	Method	Training number	Test number	Accuracy (%)
Liu et al. [32]	Baum–Welch and Viterbi Path Counting	520	260	85.77
Sahoo et al. [33]	Firefly-based back propagation	442	442	73.3
Kundu et al. [34]	26-point feature extraction and ANN	520	260	86.5
Zeng et al. [35]	Leap motion via deterministic	2340	2600	92.9

response times are less than 20 ms which shows that the system exhibits high real-time performance.

The proposed work contributes to promote the accuracy of the hand gestures recognition as alphabets (“w,” “o,” “r,” and “k”) with the employment of SSD and image cropping. The results show that the adopted classification approach exhibits superior performance, which clearly indicates that the proposed system is an effective method for the hand gestures recognition. It is found, by comparing with other works, that the accuracy of the proposed method adopted in our work is higher than that of others which are listed in Table 2.

4. Conclusion

The Single-Shot Multibox Detector (SSD) deep algorithm is proposed to apply to the hand gesture recognition. We chose four character’s hand gestures under three different complex backgrounds as the investigated objects. The 19-layer convolutional neural network is used as a recognition model with learning and training the selected characters end-to-end. The system test results show that the hand gesture recognition system based on the SSD model performs efficiently, reliably, quickly, and accurately. The response time of the system is less than 20 ms revealing high real-time performance. The minimum accuracy is more than 93.8%, and the maximum is 99.2%. The research results show that the SSD algorithm can be used in the hand gesture recognition system for the human-computer interaction application.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge the financial supports by the Shandong Key Research and Development Plan Project

(grant no. 2019GGX105018), National Key R&D Program of China (grant no. 2017YFE0112000), and Shanghai Municipal Science and Technology Major Project (grant no. 2017SHZDZX01).

References

- [1] S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: a survey,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2012.
- [2] S. Kim, G. Park, S. Yim, S. Choi, and S. Choi, “Gesture-recognizing hand-held interface with vibrotactile feedback for 3D interaction,” *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1169–1177, 2009.
- [3] O. K. Oyedotun and A. Khashman, “Deep learning in vision-based static hand gesture recognition,” *Neural Computing and Applications*, vol. 28, no. 12, pp. 3941–3951, 2016.
- [4] J. Jeong and Y. Jang, “Max–min hand cropping method for robust hand region extraction in the image-based hand gesture recognition,” *Soft Computing*, vol. 19, no. 4, pp. 815–818, 2014.
- [5] W. Liu, D. Anguelov, D. Erhan, S. Szegedy, C.-Y. Fu, and A. C. Berg, “SSD: single shot multibox detector,” *Computer Vision—ECCV 2016*, vol. 21, pp. 21–37, 2016.
- [6] Y. Wu, D. Jiang, X. Liu, R. Bayford, and A. Demosthenous, “A human-machine interface using electrical impedance tomography for hand prosthesis control,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 6, pp. 1322–1333, 2018.
- [7] X. Ma and J. Peng, “Kinect sensor-based long-distance hand gesture recognition and fingertip detection with depth information,” *Journal of Sensors*, vol. 2018, Article ID 5809769, 9 pages, 2018.
- [8] S. F. Chevtchenko, R. F. Vale, and V. Macario, “Multi-objective optimization for hand posture recognition,” *Expert Systems with Applications*, vol. 92, pp. 170–181, 2018.
- [9] C. Mummadi, F. Leo, K. Verma et al., “Real-time and embedded detection of hand gestures with an IMU-based glove,” *Informatcs*, vol. 5, no. 2, p. 28, 2018.
- [10] D. Xu, X. Wu, Y.-L. Chen, and Y. Xu, “Online dynamic gesture recognition for human robot interaction,” *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3-4, pp. 583–596, 2014.
- [11] J. Dongarra, M. Gates, J. Kurzak, P. Luszczek, and Y. M. Tsai, “Autotuning numerical dense linear algebra for batched

- computation with GPU hardware accelerators,” *Proceedings of the IEEE*, vol. 106, no. 11, pp. 2040–2055, 2018.
- [12] M. Morchid, “Parsimonious memory unit for recurrent neural networks with application to natural language processing,” *Neurocomputing*, vol. 314, pp. 48–64, 2018.
- [13] C.-C. Hsieh and D.-H. Liou, “Novel Haar features for real-time hand gesture recognition using SVM,” *Journal of Real-Time Image Processing*, vol. 10, no. 2, pp. 357–370, 2012.
- [14] A. Sultana and T. Rajapuspha, “Vision based gesture recognition for alphabetical hand gestures using the SVM classifier,” *International Journal of Computer Science & Engineering Technology*, vol. 3, no. 7, pp. 218–223, 2012.
- [15] J. Triesch and C. von der Malsburg, “A system for person-independent hand posture recognition against complex backgrounds,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, pp. 1449–1453, 2001.
- [16] J. Wang and G. Wang, “Hand-dorsa vein recognition with structure growing guided CNN,” *Optik*, vol. 149, pp. 469–477, 2017.
- [17] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, “Hand gesture recognition with 3D convolutional neural networks,” *IEEE on Computer Vision*, vol. 1, no. 3, pp. 1–7, 2015.
- [18] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, “Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm,” *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69–80, 2006.
- [19] J. Chen, Q. Ou, Z. Chi, and H. Fu, “Smile detection in the wild with deep convolutional neural networks,” *Machine Vision and Applications*, vol. 28, no. 1-2, pp. 173–183, 2016.
- [20] C. Zhang, Y. Tian, X. Guo, and J. Liu, “DAAL: deep activation-based attribute learning for action recognition in depth videos,” *Computer Vision and Image Understanding*, vol. 167, pp. 37–49, 2018.
- [21] P. Barros, G. I. Parisi, C. Weber, and S. Wermter, “Emotion-modulated attention improves expression recognition: a deep learning model,” *Neurocomputing*, vol. 253, pp. 104–114, 2017.
- [22] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [23] J. Yu, J. Li, B. Sun, J. Chen, and C. Li, “Multiclass radio frequency interference detection and suppression for SAR based on the single shot multibox detector,” *Sensors*, vol. 18, no. 11, p. 4034, 2018.
- [24] A. Fuentes, S. Yoon, S. Kim, and D. Park, “A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition,” *Sensors*, vol. 17, no. 9, p. 2022, 2017.
- [25] Y. Li, H. Huang, Q. Xie, L. Yao, and Q. Chen, “Research on a surface defect detection algorithm based on MobileNet-SSD,” *Applied Sciences*, vol. 8, no. 9, p. 1678, 2018.
- [26] Y. Wang, C. Wang, and H. Zhang, “Combining a single shot multibox detector with transfer learning for ship detection using sentinel-1 SAR images,” *Remote Sensing Letters*, vol. 9, no. 8, pp. 780–788, 2018.
- [27] X. Zhao, W. Li, Y. Zhang, and Z. Feng, “Residual super-resolution single shot network for low-resolution object detection,” *IEEE Access*, vol. 6, pp. 47780–47793, 2018.
- [28] C. Yi, L. Zhou, Z. Wang, Z. Sun, and C. Tan, “Long-range hand gesture recognition with joint SSD network,” in *Proceedings of the IEEE Internal Conference on Robotics and Biomimetics*, Kuala Lumpur, Malaysia, December 2018.
- [29] D. Zhao, D. Zhu, J. Lu, Y. Luo, and G. Zhang, “Synthetic medical images using F&BGAN for improved lung nodules classification by multi-scale VGG16,” *Symmetry*, vol. 10, no. 10, p. 519, 2018.
- [30] A. L. C. Barczak, N. H. Reyes, M. Abastillas, A. Piccio, and T. Susnjak, “A new 2d static hand gesture colour image dataset for ASL gestures,” *Research Letters in the Information and Mathematical Sciences*, vol. 15, pp. 12–20, 2011.
- [31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, pp. 1106–1114, 2012.
- [32] N. Liu, B. C. Lovell, P. J. Kootsookos, and R. I. A. Davis, “Model structure selection & training algorithms for an HMM gesture recognition system,” in *Proceedings of the IEEE 9th International Workshop on Frontiers in Handwriting Recognition*, pp. 100–105, Tokyo, Japan, October 2004.
- [33] M. K. Sahoo, J. Nayak, S. Mohapatra, B. K. Nayak, and H. S. Behera, “Character recognition using fire-fly based back propagation neural network,” *Computational Intelligence*, vol. 2, pp. 37–49, 2016.
- [34] S. Kundu, H. S. Chhabra, S. S. Ara, and R. P. Mishra, “Optical character recognition using 26-point feature extraction and ANN,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7, no. 5, pp. 156–162, 2017.
- [35] W. Zeng, C. Wang, and Q. Wang, “Hand gesture recognition using leap motion via deterministic learning,” *Multimedia Tools and Applications*, vol. 77, no. 21, pp. 28185–28206, 2018.