Hindawi
Mobile Information Systems
Volume 2020, Article ID 1475164, 13 pages
https://doi.org/10.1155/2020/1475164

Research Article

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

Daegyu Choe,[1] **Eunjeong Choi**,[1] **and Dong Keun Kim**[2]

[1]*Department of Computer Science, Graduate School of Sangmyung University, Seoul, Republic of Korea*
[2]*Department 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

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 cameraswere 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 pretrained 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.

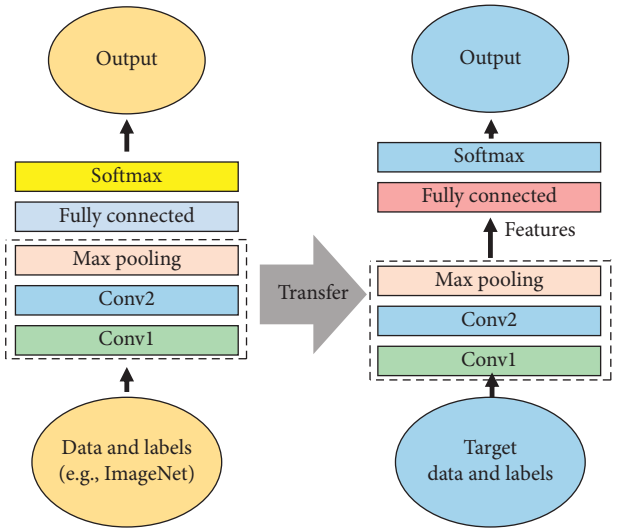

Figure 1: Diagram of the transfer learning.

Second, we create and train a classifier using the preprocessed 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 preprocessing 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 preprocessed 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]:

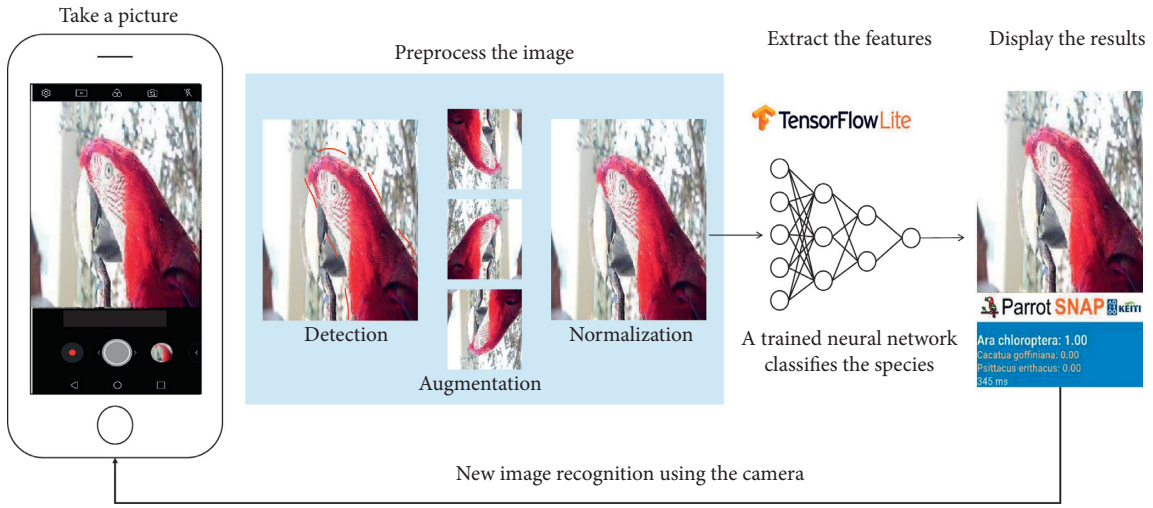

Figure 2: System configuration and scenario for classifying endangered parrot species using a mobile device.

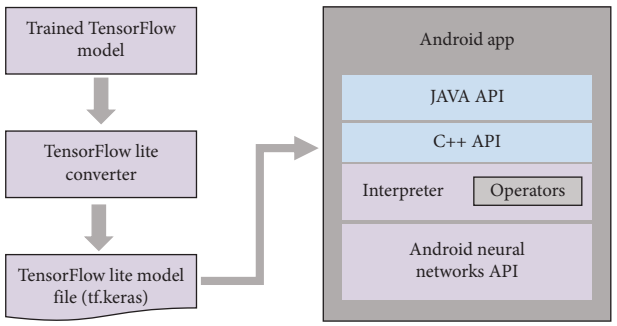

Figure 3: TensorFlow Lite conversion process graph.

$$\mathrm{GAP}_i = \frac{1}{mxn} \sum_{a}^{m} \sum_{b}^{n} x. \tag{1}$$

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

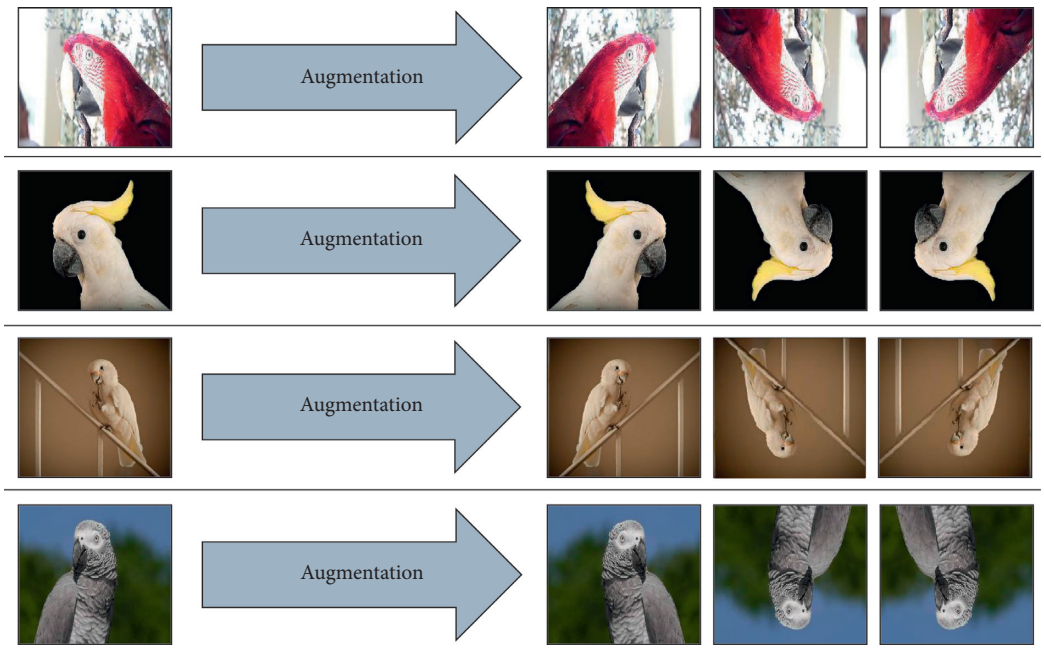

Figure 4: Data augmentation for images of endangered parrot species.

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]

$\text{GAP}_i = \frac{1}{mxn}(\Sigma_a^m \Sigma_b^n x_{a\,b})$

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

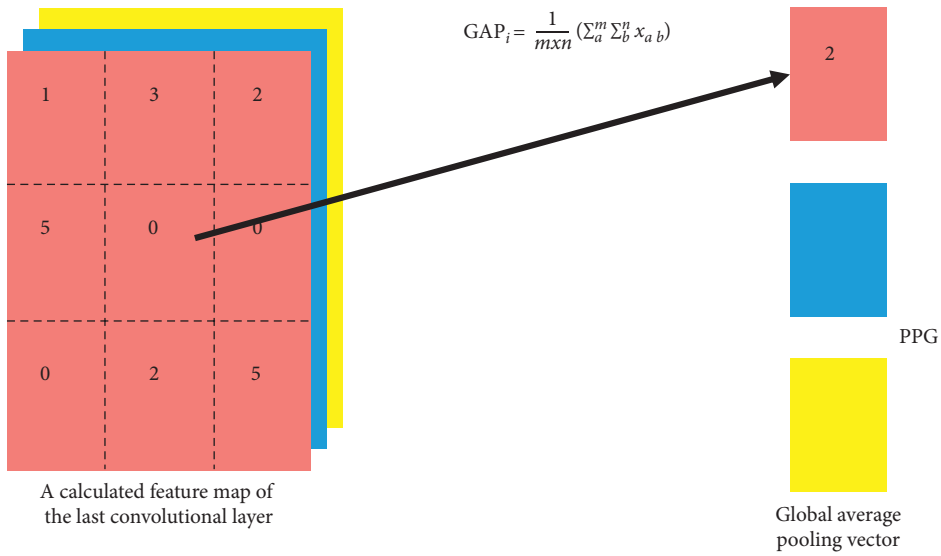

Figure 5: Concept diagram of global average pooling.

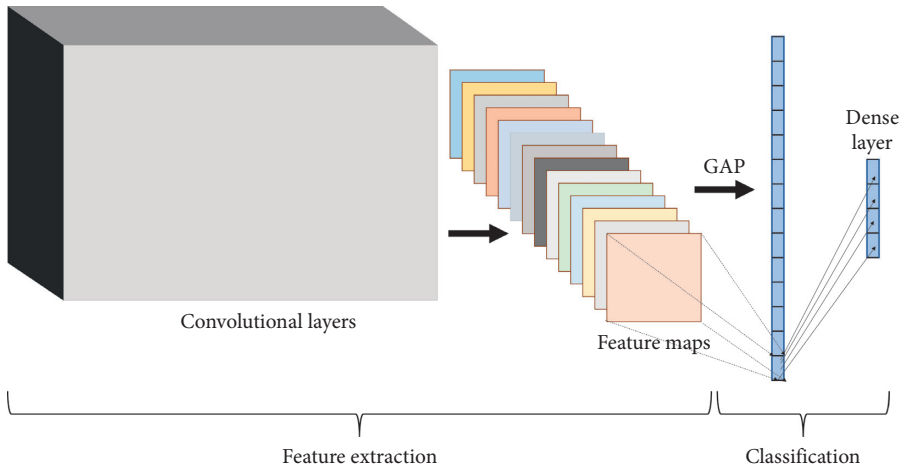

Figure 6: Convolutional layers and feature maps for feature extraction of endangered parrot species.

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

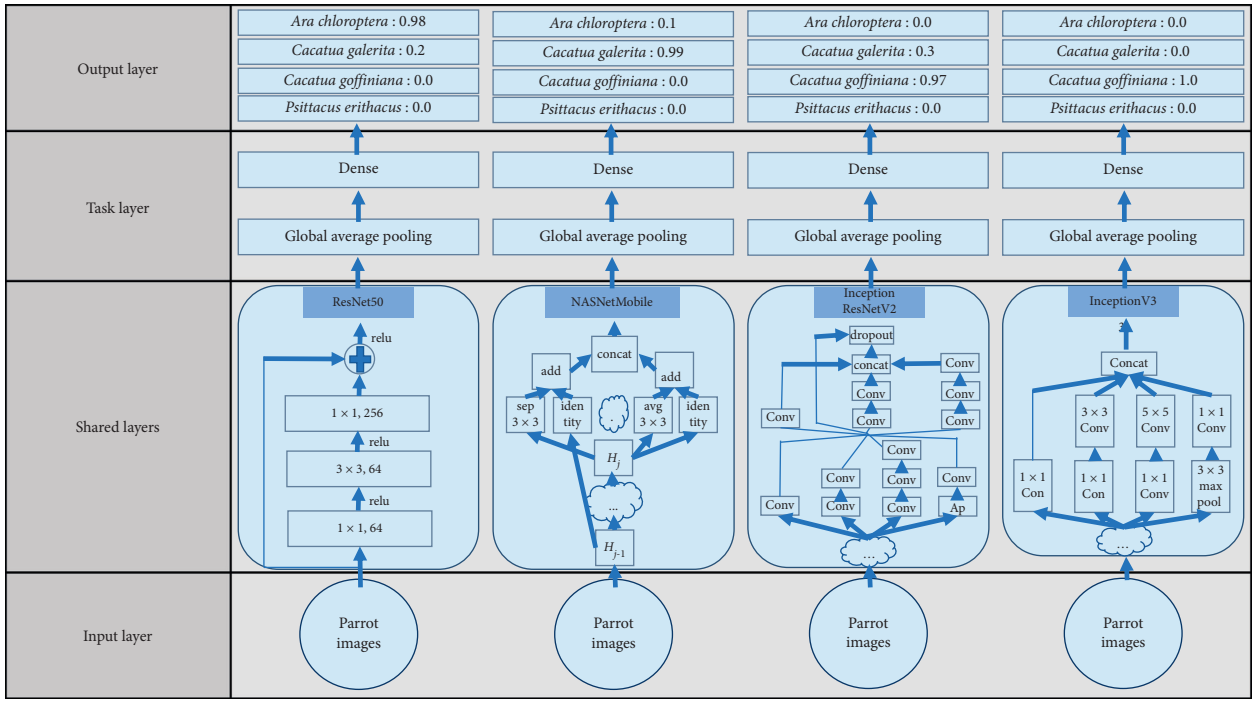

Figure 7: Convolutional layers and feature maps for feature extraction of endangered parrot species.

Table 1: Examples of four endangered parrot species.

Picture				
Name	Red and green macaw	Sulphur-crested cockatoo	Goffin's cockatoo	Gray parrot
Scientific name	*Ara chloroptera*	*Cacatua galerita*	*Cacatua goffiniana*	*Psittacus erithacus*
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

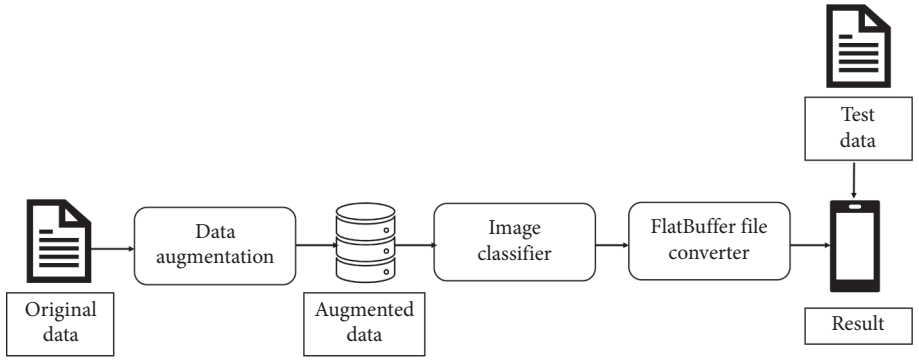

Figure 8: Overall experiment process.

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

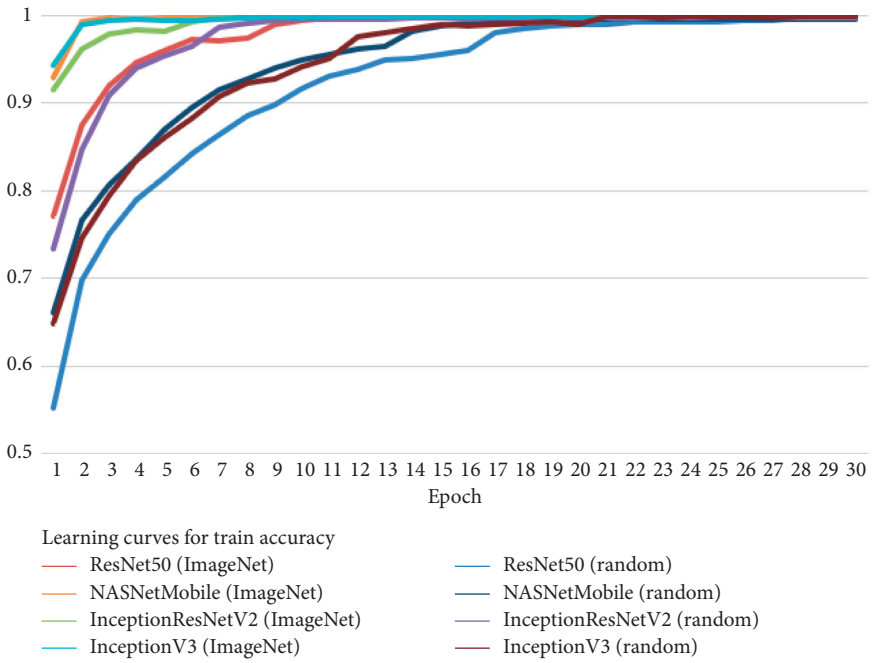

Figure 10: Learning curves of each model's train accuracy.

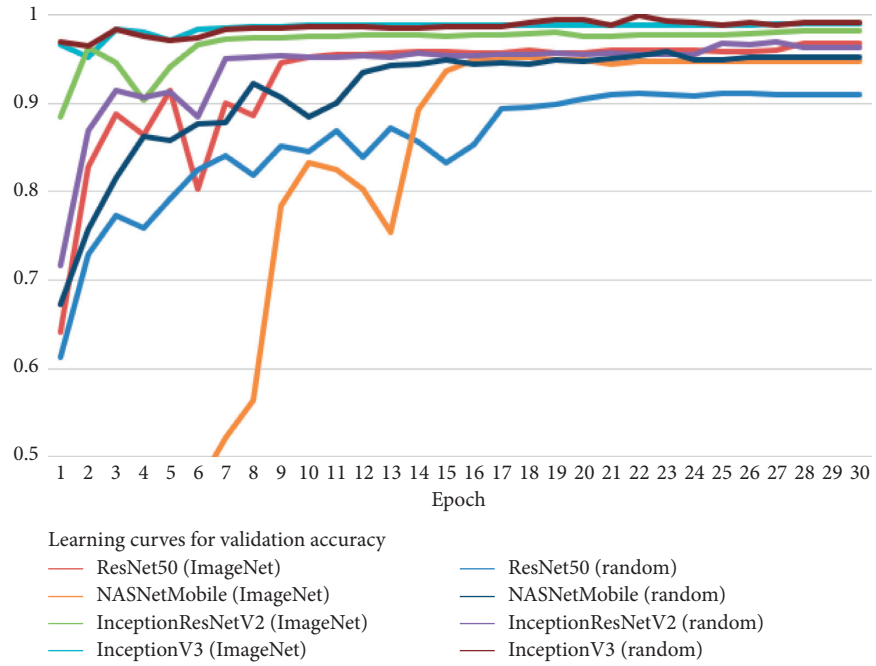

Figure 11: Learning curves of each model's validation accuracy.

Table 2: Confusion matrix.

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

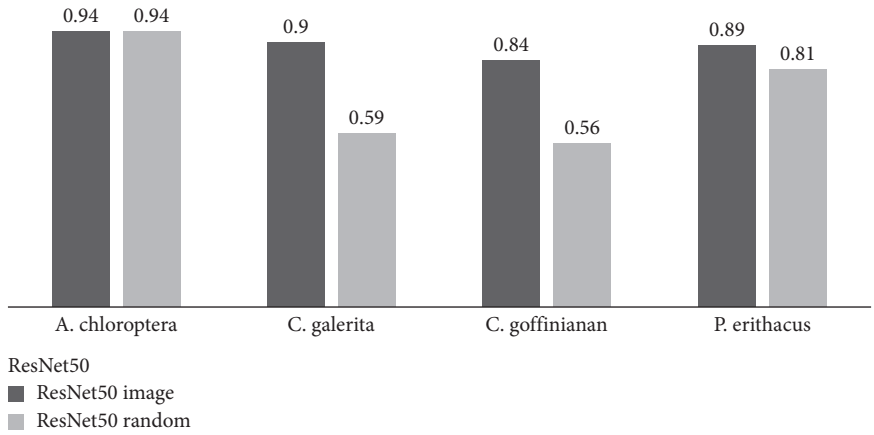

Figure 12: F1-score of RestNet50 for four different endangered parrot images.

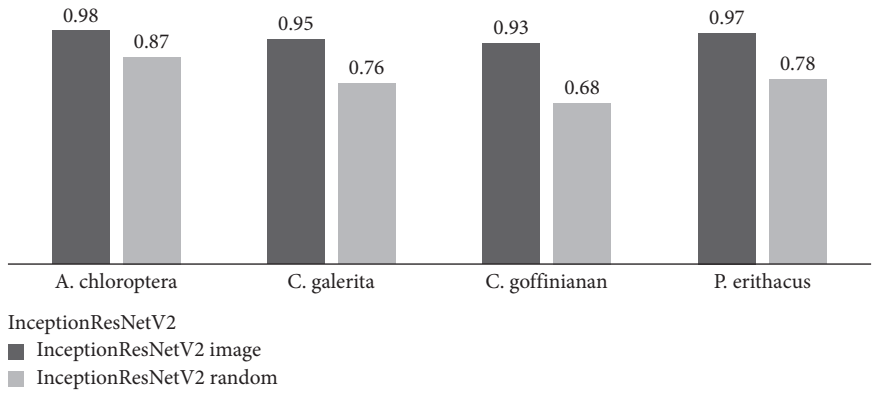

Figure 13: F1-score of InceptionResNetV2 for four different endangered parrot images.

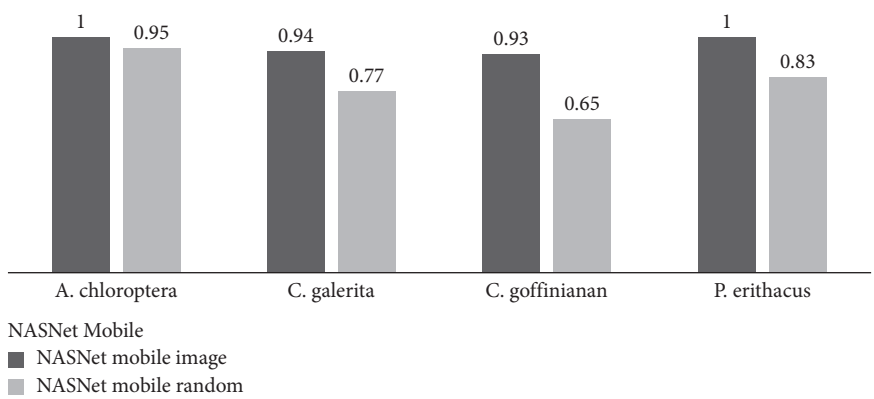

Figure 14: F1-score of NASNetMobile for four different endangered parrot images.

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

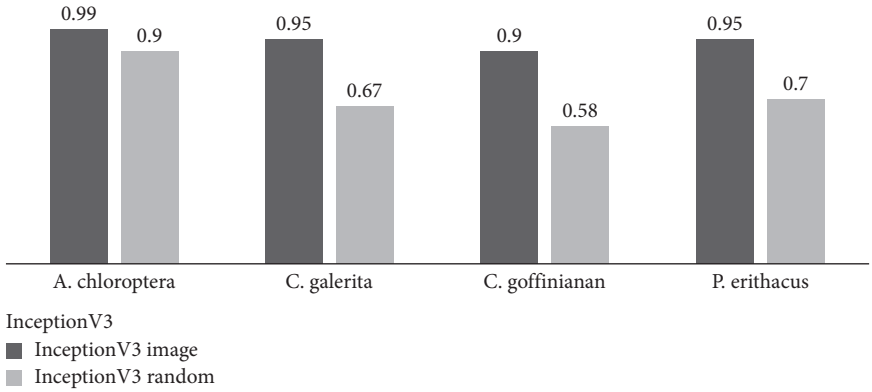

Figure 15: F1-score of InceptionV3 for four different endangered parrot images.

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 dependences 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 pretrained 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. Olivas, 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.