*Journal of Computer Science and Cybernetics, V.38, N.1 (2022), 15–29*
DOI no. 10.15625/1813-9663/38/1/16309

# AN EFFECTIVE DEEP LEARNING MODEL FOR RECOGNITION OF ANIMALS AND PLANTS

TRINH THI ANH LOAN[1], PHAM THE-ANH[1,*], LE VIET-NAM[1], HOANG VAN-DUNG[2]

[1] *Hong Duc University, 565 Quang Trung Street, Dong Ve Ward, Thanh Hoa City, Viet Nam*
[2] *Ho Chi Minh City University of Technology and Education,*
*01 Vo Van Ngan Street, Linh Chieu Ward, Thu Duc City, Ho Chi Minh City, Viet Nam*

**Abstract.** This paper presents a deep learning model to address the problem of recognition of animals and plants. The context of this work is to make an effort in protection of rare species that are seriously faced to the risk of extinction in Vietnam such as Panthera pardus, Dalbergia cochinchinensis, Macaca mulatta. The proposed approach exploits the advanced learning ability of convolutional neural networks and Inception residual structures to design a lightweight model for classification task. We also apply the transfer-learning technique to fine-tune the two state-of-the-art methods, MobileNetV2 and InceptionV3, specific to our own dataset. Experimental results demonstrate the superiority of our object predictor (e.g., 97.6% accuracy) in comparison with other methods. In addition, the proposed model works very efficiently with the inference speed of around 113 FPS on a CPU machine, enabling it for deployment on mobile environment.

**Keywords.** Deep learning models, classification losses, feature pyramid network.

## 1. INTRODUCTION

Nowadays, we are learning a substantial amount of information on daily media reports that a lot of species in the world are in danger of extinction such as Panthera pardus, Dalbergia cochinchinensis, Macaca mulatta. This situation is a source of many serious problems to our planet, for instance: floods, forest fires, heat waves, and climate changes. Therefore, it is urgent for humans to do necessary actions to protect natural environment, wildlife and plants, especially endangered species. One of the promising solutions is to apply modern technologies such as artificial intelligence or deep learning to build useful applications. This paper presents a deep neural network to recognize the animals or plants with a real-time response. The proposed model is scheduled to be embedded into a mobile application that are helpful to people who are working in the sector of environment protection. For instance, they can install the application on portable computing devices to quickly take a capture of an animal or plant and then to determine the identity as well as all related information of that individual.

*Corresponding author.
*E-mail addresses*: trinhthianhloan@hdu.edu.vn (T.T.A.Loan), phamtheanh@hdu.edu.vn (P.T.Anh), levietnam@hdu.edu.vn (L.V.Nam), dunghv@hcmute.edu.vn (H.V.Dung)

In recent years, deep learning techniques have been evolved and achieved outstanding performance for various problems in the areas of computer vision, pattern recognition, and object detection. In the specific field of image classification, the most representative methods have been derived from convolutional neural networks (CNN) such as VGG16 [1], AlexNet [2], ResNet50 [3], InceptionV3 [4]. Although these methods achieve high recognition rates, they suffer from intensively computational cost, making them difficult to be deployed on CPU machines. To handle the computational burden, a number of attempts have been presented, including ShuffleNet [5], SqueezeNet [6], MobileNetV1 [7], MobileNetV2 [8] and GoogLeNet [9]. Nonetheless, the obtained results are till not satisfactory for deployment on mobile platform in terms of either accuracy or efficiency.

In the present work, we bring an attempt to design a lightweight and accurate convolutional neural network (CNN) model to address the problem of image classification. In particular, the proposed method is specifically targeted to classify the species of animals and plants that are in the risk of extinction. To this aim, we have first collected a dataset of around sixty species of rare animals and plants (with approximately 23,000 images) from different places of Thanh Hoa province. On the other hand, we have designed an effective CNN model to perform image classification of these species. Finally, we have conducted various experiments to evaluate the proposed method in comparison with the state-of-the-art CNN models. The experimental results showed that our approach achieves a recognition rate of 97.6% while being very efficient with the inference speed of around 130 FPS on a moderate GPU (GTX 1070Ti) or 113 FPS on a CPU machine.

The rest of this paper has been structured as follows. Section 2 provides a comprehensive review of methods for image classification with special focus on deep learning approach. Section 3 presents the proposed method with detailed description of network architecture, computational analysis, and training settings. Section 4 is dedicated to the experiments and evaluation of the proposed method. Finally, Section 5 concludes the paper and discusses potential directions of further studies.

## 2. RELATED WORK

In this section, we revisit the main methods for image classification using deep learning approach. We shall restrict our discussions to the high performance CNN methods as well as real-time CNN predictors.

The authors in [2] introduced a CNN model, namely AlexNet, composed of five convolutional layers, several max-pooling layers, and three fully connected (FC) stages. Differing from traditional CNN models, AlexNet employs a novel activation function, rectified linear units (ReLUs) [10], to gain faster convergence as well as stable training. In addition, the convolutional layers of AlexNet employ large filter sizes (e.g., $11 \times 11$ and $5 \times 5$) and large stride (e.g., 4). The model takes as input the image size of $224 \times 224 \times 3$ and produces an output vector of 1000 channels corresponding to 1000 classes. At last, AlexNet applies the drop-out technique [11] to randomly discard 50% hidden nodes for reducing overfitting. Experiments conducted on the dataset of the ILSVRC-2012 competition [12] show that the proposed method performs best with a top-5 error rate of 15.3%. The model is heavy in terms of weights and memory with about 60 million parameters and 650,000 neurons.

In the work of [1], a CNN model (VGG16) has been presented that consists of sixteen convolutional layers, several max-pooling layers, and three FC layers. The input of the model is normalized to the size of $224 \times 224 \times 3$. The three FC layers create 4096, 4096, and 1000 outputs, respectively. In contrast to the AlexNet, VGG16 employs small kernel size $(3 \times 3)$ with stride of 1. The authors argued that using several small filters could be good alternatives for a large kernel. The model contains about 180 million parameters and yields a top-5 test error of 7% (single model) in the dataset of the ILSVRC-2012 competition.

The authors in [4] propose replacing convolutions having large filter sizes by the ones with smaller kernel sizes. To do so, they employed two sorts of operations. First, they applied factorization of a large convolution into smaller ones. For instance, a $5 \times 5$ convolution can be replaced by two two convolutional layers having the filter size of $3 \times 3$. Second, they performed spatial factorization of a symmetric convolution into asymmetric convolutions. For example, a $3 \times 3$ convolution can be replaced by a $3 \times 1$ convolution followed by another $1 \times 3$ convolution. Besides, they also incorporated the idea of using auxiliary classifiers [9] to improve the convergence and accuracy. At last, spatial down-sampling of the feature maps is done by applying an efficient grid size reduction method that combines max-pooling and convolution of stride $> 1$. Putting all together, a novel CNN model, so-called InceptionV3, is presented. InceptionV3 contains 42 layers but is more efficient than VGG16, and yields a top-5 error rate of just 5.6% on the ILSVRC-2012 dataset.

Another high performance model for image classification has been presented in [3] with the popular alias of residual learning. As the residual networks are easier to train and optimize, much deeper models can be created, resulting in an improvement of accuracy. In their paper, different versions of deeper models have been studied, including ResNet34, ResNet50, ResNet101, and ResNet152. All the models mostly discard the use of max-pooling layers; more precisely each model applies only one max-pooling operator after the first convolutional layer. Alternatively, they use directly convolutional layers with stride $> 1$ for spatial down-sampling. For example, ResNet34 contains 34 parameter layers (e.g., 33 convolutional layers, a global average pooling, and a FC layer to create a 1000-output vector). It is worth noting that the average pooling and max-pooling layers do not contain trainable parameters. In term of efficiency, the computational complexity of ResNet34 is 5 times smaller than that of VGG19 [1] (e.g., using FLOPs metric). As for performance, all the models achieve competitive accuracy compared with the state-of-the-art methods. Especially, the single-model ResNet152 obtains a top-5 error rate of 4.49% on the ILSVRC 2015 classification challenge[1].

Apart from these methods, there is a streamline of studies targeted to real-time inference speed such as GoogLeNet [9], ShuffleNet [5], SqueezeNet [6], MobileNetV1 [7], and MobileNetV2 [8]. The authors in [9] has created a lightweight model, codenamed GoogLeNet, that embeds a number of advanced features to create a deeper yet less parameter architecture. First, the model exploits the benefit of using Inception module to learn multi-scale features in the dimension of network depth. Here, the Inception structure applies different filter size convolutions (e.g., $1 \times 1$, $3 \times 3$, $5 \times 5$) to handle better the learning of multi-scale features. Furthermore, the use of many $1 \times 1$ convolutions helps reduce the trainable parameters while also creating a deeper network. Second, the author integrated a number of auxiliary classifier for training to improve the accuracy. Next, they also applied global average pooling before the FC layer to reduce a large number of parameters while till gaining the top-1 accuracy

[1] https://image-net.org/challenges/LSVRC/2015/

by 0.6%. Finally, the drop-out technique is applied to discard 30% hidden nodes. Overall, the model contains 22 deep layers (consisting of about 7 million parameters) and becomes the winner at ILSRVRC 2014 [13] with the top-5 error rate of 6.67% in classification task.

SqueezeNet [6] applies three strategies to create a small CNN architecture. First, it tries to replace as many as possible the 3 × 3 filters by the smaller ones having the size of 1 × 1. Second, it further forces feeding fewer input channels into the 3x3 filters, if any. Finally, it applies delayed down-sampling to enrich the features learned at the beginning layers. To handle the strategies 1 and 2, a Fire module is presented that consists of a squeeze convolution layer and an expand layer. The former is composed of 1 × 1 filters, while the latter employs convolutions having the sizes of 1 × 1 and 3 × 3 with the constraint that the number of kernels in the squeeze layer is less than that of the expand layer. To the end, SqueezeNet composes of two convolutional layers and eight Fire modules with the absence of FC layer. Experimental results showed that SqueezeNet is competitive with AlexNet in terms of accuracy but is much more efficient (e.g., 50 times fewer parameters).

In [7], the MobileNetV1 is presented that exploits the idea of depthwise separable convolution. A depthwise separable convolution is primarily composed of two components: depthwise convolution and pointwise convolution. The former applies a single filter per each input channel to create a set of separated feature maps. The latter creates linear combinations of the feature maps via the channel dimension. MobileNetV1 is finally built as a 28-layer stack of depthwise convolutions and pointwise convolutions in conjunction with the batch normalization [14] and ReLU activation function. MobileNetV1 performs on par with VGG16 but is 27 times less compute intensive.

ShuffleNet [5] utilizes channel shuffle operation to create better feature representation of group convolutions. Traditionally, the outputs of a group convolution only concern with the specific input channels within the group. In contrast, channel shuffle operation allows the outputs of different group convolutions to be exchanged before feeding into the next layer. Hence the features are fully related and more robust. In addition, ShuffleNet improves the bottleneck unit in [3] by incorporating both channel shuffle operation and depthwise separable convolution resulting in a powerful and lightweight block, namely ShuffleNet unit. At the end, ShuffleNet consists of an initial convolutional layer, a number of ShuffleNet units structuring into three main stages, a global average pooling, and a FC layer. ShuffleNet is superior to MobileNetV1 in the means of both classification accuracy and inference speed.

The MobileNetV2 [8] improves MobileNetV1 in that it designs an advanced block, namely bottleneck depth-separable convolution with residual, to be the core component. Principally, a bottleneck convolution is composed of a thin input layer (i.e., low-dimensional volume) followed by one or several high-dimensional inner layers (i.e., expansion layers) then followed by a thin output layer. Here the input and output layers should contain only the linear features (e.g., by not using ReLU activation) thus not requiring to have a large number of channels. In contrast, the inner layers must capture the non-linearities (e.g., with ReLU included) and hence are required to work in high-dimensional spaces (e.g., using more filters or kernels) so that when ReLU destroys information in some channels, it is expected that the lost features might still be preserved in other channels. To reduce the computational overhead of using expand layers, MobileNetV2 applies depthwise separable convolutions that have been successfully exploited in MobileNetV1. In addition, a bottleneck is coupled with the residual block to form inverted residual bottleneck. To this end, the architecture of

MobileNetV2 consists of two convolutional layers (one at the beginning and the other at the end) and 19 residual bottlenecks. As for performance, MobileNetV2 yields better accuracy in the classification task while having 19% less parameters than MobileNetV1.

Differing from the afore-mentioned works, there are specific studies dedicated to solve the problem of forest species recognition as presented in [15, 16, 17]. The authors in [15] designed a light-weight model composed of two CNN layers, two max-pooling operations, followed by a local-connected layer and finally a full-connected layer. The proposed model is targeted to handle high resolution images of forest species with the consideration of efficient running times. To this aim, they proposed training the model by extracting random patches from images. For recognition, a number of patches are first extracted for each test image and then passed into the model. The patch results are combined with voting technique to determine the final result. In [17], the authors investigated a study on applying an off-the-shelf CNN model for four trap camera datasets of species. Specifically, the ResNet18 model [3] is employed as the basic model and was trained using the four datasets. For each dataset, two separate models have been trained and work in a cascade filtering fashion. One model is designed to predict empty or non-empty images. The other model is trained to recognize the species of non-empty images. In addition, the authors also study the effect of using transfer-learning technique where the basic models are pre-trained on the largest dataset. Experiments showed that the overall accuracies are ranged from 88.7% and 92.7% and the transfer-learning strategy helps improving the accuracies of both the empty model and species model. The work in [16] trains two CNN models, VGG16 and ResNet50, to classify 20 African wildlife species using a dataset of 111,467 images. The obtained accuracy is about 87.5% for both the models. Besides, the authors conducted a study on CNN-based feature interpretation that is helpful to explain the inner mechanism of feature learning of a CNN classifier.

## 3. THE PROPOSED APPROACH

### 3.1. Network architecture

In this section, we describe the proposed approach in detail. The design strategy of our model is two-fold. First, we further provide empirical demonstration that the Inception residual structure [18] is helpful for learning discriminated features in classification task. This structure has been successfully exploited in previous work [19, 20] and is a good alternative to the max-pooling layer that has been a common choice in the literature when designing CNN models [2, 1]. Specifically, an Inception residual (i.e., Inception-ResNet-A module) learns the features in a multi-scale manner and forces the model to explore the feature representation in the dimension of network depth. It is worth mentioning that the Inception residual module does not change the dimension of the input tensor (i.e., the spatial size and the number of channels are not changed). While the work in [19] is dedicated to handle the problem of face detection with a central focus on designing the semantic convolutional box, the present work emphasizes the design of an efficient model for the classification problem.

Second, we insist on creating a deeper model by adding more convolutions with stride of 1 while using less filters or channels at each layer. These two strategies help create a deeper and wider model while making it still efficient for real-time inference speed. Overall, the network architecture is presented in Table 1, composing of alternating the convolutional layers and (i.e., marked by Conv1 to Conv9) the Inception residual modules (i.e., IncRes1-IncRes6). As for the the standard selection of the stride parameter, a kernel with filter size of 3 × 3 is usually associated with a stride of 1 or 2 (i.e., see [8, 4] for examples). For each convolutional layer using the stride 2, the spatial size of the model is reduced four times and hence the entire network would be quickly pruned. To handle this point, we have added some convolutional layers with stride 1 to make the network deeper.

The model is ended by a global average pooling layer (i.e., AvgPool) which computes a mean value from the features in a 3 × 3 window. The network takes as input an image having the size of 192 × 192 × 3, has 15 parameter layers, and produces an output feature vector with the length of $K$ that is the number of classes or categories. As we will mention in the subsequent part, we have $K = 56$ classess of animals and plants in our dataset. To drive the learning process, cross entropy is selected as the loss function for the training.

Table 1: The architecture of the proposed network

| No | Layer | Filter Shape/Stride (s) | Input Shape | FLOPs |
|---|---|---|---|---|
| 1 | Conv1 | $3 \times 3 \times 24$, s=2 | $192 \times 192 \times 3$ | 5,971,968 |
| 2 | IncRes1 | | $96 \times 96 \times 24$ | 297,271,296 |
| 3 | Conv2 | $3 \times 3 \times 24$, s=1 | $96 \times 96 \times 24$ | 47,775,744 |
| 4 | Conv3 | $3 \times 3 \times 32$, s=2 | $96 \times 96 \times 24$ | 15,925,248 |
| 5 | IncRes2 | | $48 \times 48 \times 32$ | 77,856,768 |
| 6 | Conv4 | $3 \times 3 \times 32$, s=1 | $48 \times 48 \times 32$ | 21,233,664 |
| 7 | Conv5 | $3 \times 3 \times 40$, s=2 | $48 \times 48 \times 32$ | 6,635,520 |
| 8 | IncRes3 | | $24 \times 24 \times 40$ | 20,348,928 |
| 9 | Conv6 | $3 \times 3 \times 48$, s=1 | $24 \times 24 \times 40$ | 9,953,280 |
| 10 | Conv7 | $3 \times 3 \times K$, s=2 | $24 \times 24 \times 48$ | 3,545,856 |
| 11 | IncRes4 | | $12 \times 12 \times K$ | 5,557,248 |
| 12 | Conv8 | $3 \times 3 \times K$, s=2 | $12 \times 12 \times K$ | 1,052,676 |
| 13 | IncRes5 | | $6 \times 6 \times K$ | 1,389,312 |
| 14 | Conv9 | $3 \times 3 \times K$, s=2 | $6 \times 6 \times K$ | 263,169 |
| 15 | IncRes6 | | $3 \times 3 \times K$ | 347,328 |
| 16 | AvgPool | $3 \times 3$, s=1 | $1 \times 1 \times K$ | 0 |
| 17 | Flatten | | $K$ | 0 |
| | | | Total FLOPs: | 515,128,005 |

To analyze the theoretical efficiency of our method, we provide in Table 1 the computational complexity in the means of FLoating-point OPerations (FLOPs). For a given convolutional layer, FLOPs is computed as follows [19]

$$\text{FLOPs} = C_{in} F^2 C_{out} W H, \tag{1}$$

where $F$ indicates the kernel width or height (typically we use square filters), $C_{in}, C_{out}$ denote the input and output channels, and $W, H$ be the width and height of the output feature map, respectively.

As for the Inception residual module, we adopted the computation of FLOPs in [19] for a given input feature map having the shape of $w \times h \times c$ (i.e., width, height, and depth, respectively), by $192wh(c + 144)$. On the other hand, FLOPs of other layers (e.g., AvgPool, Flatten) are often ignored because they are relatively small when compared to those of convolutional layers. As shown in Table 1, the computational complexity of our method is about 515.1 MFLOPs. To have a comparative evaluation, we provide in Table 2 the FLOPs of different methods for object classification [5]. Theoretically, the proposed model has lower computational cost than other models with an exception for the case of ShuffleNet 1× model. In the experiments, we further show that the practical running times of our model is significantly lower than those of the state-of-the-art methods.

Table 2: Comparison of FLOPs of the state-of-the-art methods

| Model | MFLOPs |
|---|---:|
| VGG-16 | 15300 |
| GoogleNet | 1500 |
| AlexNet | 720 |
| MobileNetV1-224 | 569 |
| ShuffleNet 2× | 524 |
| ShuffleNet 1× | 140 |
| Our model | 515 |

### 3.2. Dataset and training

As mentioned earlier, the present work is conducted in the context of making an effort to protect the endangered species. Hence we have collected the dataset of rare animals and plants on the Internet as well as from different places in Thanh Hoa province. Specifically, the dataset consists of 22,319 images distributing to 56 species (i.e., 36 animals and 20 plants). For the groundtruth, we have asked two experts in the field of wildlife conservation to verify the labels of all the images. Figure 1 demonstrates a few examples of animals and plants in the dataset. In addition, Figure 2 provides a summary of image distribution in the dataset for 56 species. As can be seen, one of the challenges of this dataset is concerned with the unbalance distribution over the classes. Furthermore, although each image contain one species, complicated and rich context information may be present, making the classification task more challenging.

The network is implemented in Python with Tensorflow 1.14 based on this source[2] and is trained on a moderate GPU configuration (8Gb RAM, GTX 1070 Ti). In addition, Table 3 describes other parameters used during the training phase. For model training, we divide the dataset into three subsets: training, validation, and testing with respect to the ratio of 80%, 5%, and 15%. The model is trained on the training set and validated on the validation.

[2] https://github.com/TropComplique/FaceBoxestensorflow

Figure 1: Examples of the collected animals and plants in the dataset

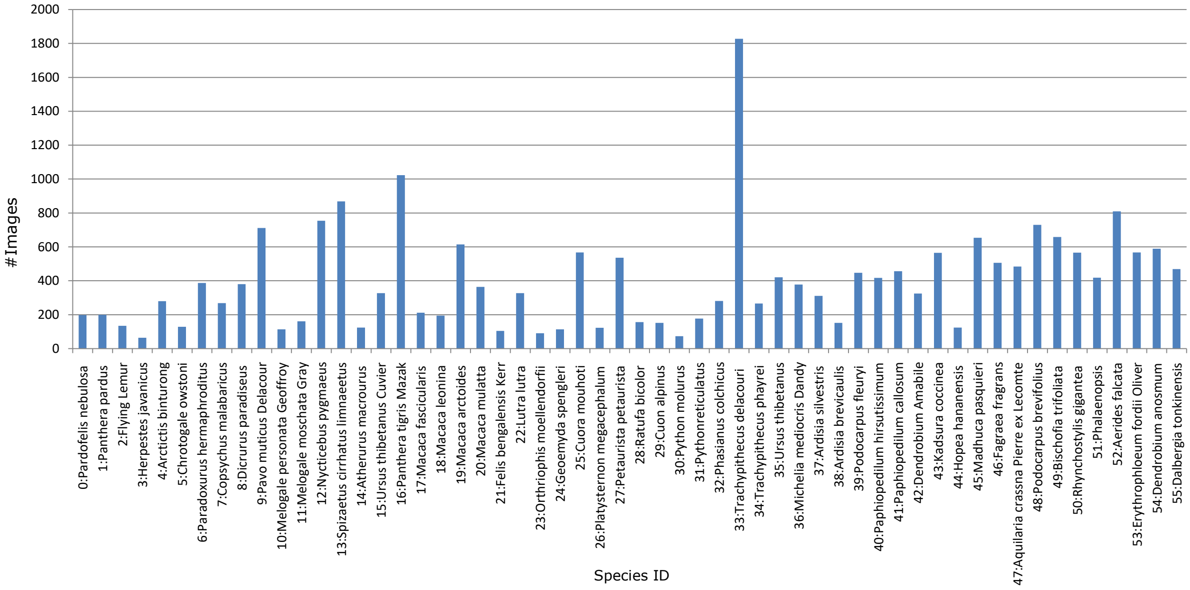


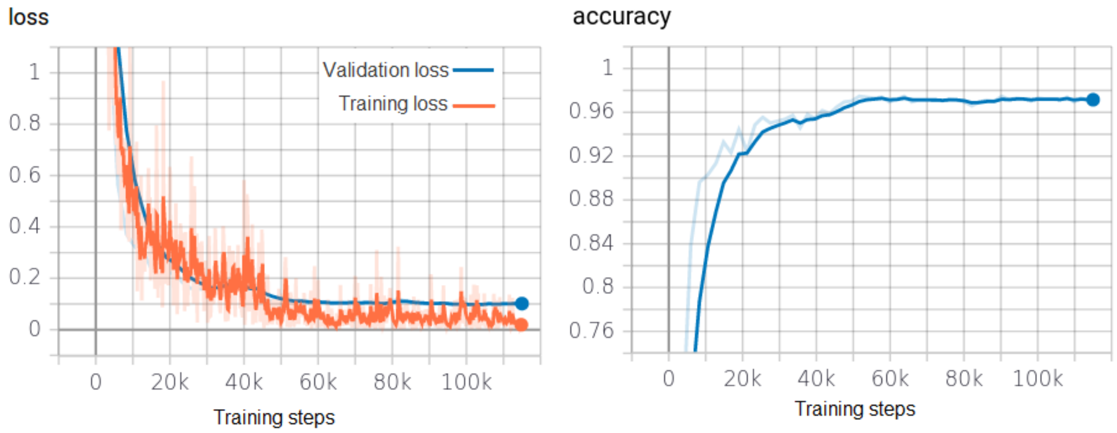
Figure 2: The statistics of image distribution of species in the dataset. Each species is represented by a pair of ID and scientific name.

In addition, we applied basic data augmentation techniques for training, including random color manipulation, pixel color scaling, horizontal and vertical flipping.

Table 3: Parameter settings for training phase

| Parameter | Value |
|---|---:|
| Number of training steps | 115,000 |
| Batch size | 32 |
| Learning rates | [0.01, 0.001, 0.0001] |
| Learning boundaries | [45000, 95000] |

When the training is done, we evaluate the model by using the test set. Figure 3(a) shows the behavior of loss functions for the training and validation sets. As can be observed, the two loss curves are very close to each other, indicating that the model has captured perfectly well the underlying structure of the data and is expected to perform equally in the test set. Figure 3(b) reveals the classification accuracy on the validation set during the training where the model yields a final accuracy of 97.1%.



(a) (b)

Figure 3: (a) Training and validation loss curves of our model, (b) Accuracy curve on the validation set.

## 4. EXPERIMENTS

### 4.1. Transfer learning of models

To have comparative results, we have included two state-of-the-art methods including MobileNetV2 [8] and InceptionV3 [4]. Specifically, we have adopted the transfer learning technique to fine-tuning these models. Both the MobileNetV2 an InceptionV3 are provided as off-the-shelf methods at the Tensorflow libarary (i.e., Keras packet)[3]. The two models have been pre-trained on the ImageNet dataset[4] with 1000 classes. The intuition of the fine-tuning technique is that we will unfreeze several top layers of a pre-trained model, add a new classification head, and retrain the model. It is, however, important to note that we are re-training only the unfreezed layers and the newly added classification head. The weights associated to the bottom layers are not updated during the fine-tuning process. In doing so, the model is adapted to learn new features particular to the new dataset.

Specifically, for each model we have added a classification head consisting of a FC layer having $K$ outputs followed by a Softmax activation to produce a prediction vector (i.e., $K = 56$). Using the FC layer for classification is a standard way used in many CNN models in the literature. Besides, we also study an additional head for classification that consists of a convolutional layer, an average pooling, a flatten layer, and finally a Softmax activation

[3] https://www.tensorflow.org/tutorials/images/transfer_learning
[4] https://image-net.org/index.php

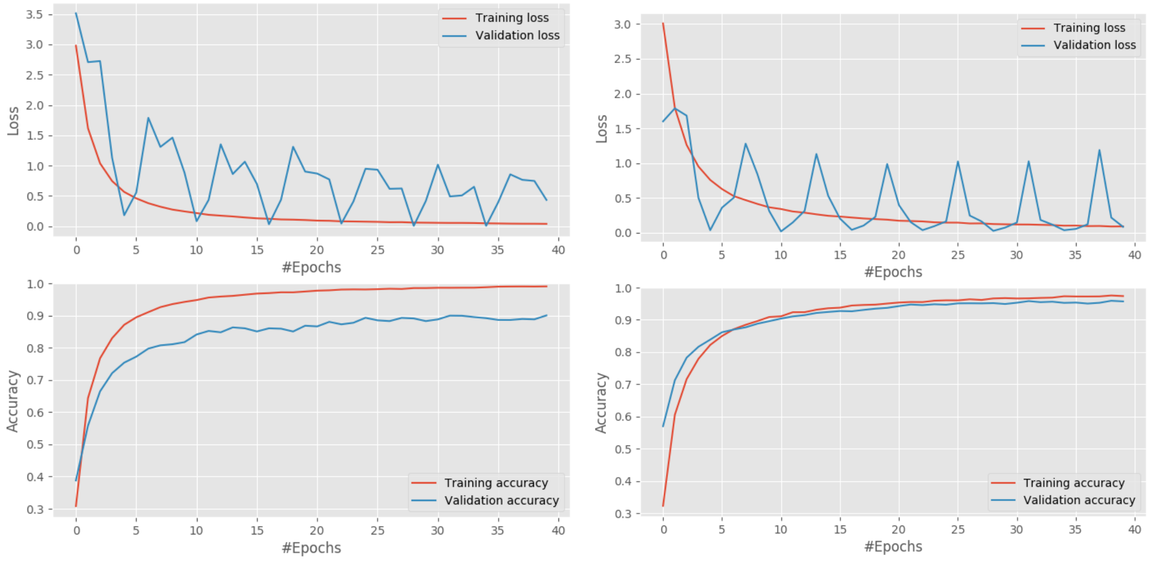
function. These new variants are denoted as MobileNetV2b and InceptionV3b in Table 4. All the models require an input size of $192 \times 192 \times 3$.

Table 4: Fine-tuning settings of two models

| Model | # Layers | Start tuning layer | Classification head |
|---|---|---|---|
| MobileNetV2 | 156 | 100 | FC layer $(K)$ + Softmax |
| InceptionV3 | 312 | 250 | FC layer $(K)$ + Softmax |
| MobileNetV2b | 156 | 100 | Conv + AvgPool + Flatten + Softmax |
| InceptionV3b | 312 | 250 | Conv + AvgPool + Flatten + Softmax |

To demonstrate how the new classification head works, taking the MobileNetV2b for instance, the last layer before appending the classification head has the shape of $6 \times 6 \times 1280$. Hence, we continue adding a convolutional layer having the filter size of $3 \times 3$, stride 2, the number of channels $K$. The output of this layer has the volume of $3 \times 3 \times K$. We then append an average pooling with stride 3 and filter size of $3 \times 3$, resulting in a new output of $1 \times 1 \times K$. Finally, we employ the flatten layer followed by the Softmax function to create a K-dimensional feature vector for prediction.

The number of layers and the layer onwards we start the fine-tuning are all detailed in Table 4. In addition, we have set the number of epochs (i.e., 40), batch size (i.e., 32), and the learning rate (i.e., 0.00001) for re-training the two models. The dataset used for fine-tuning is exactly applied as used for training our proposed system. The two models are fine-tuned using the training and validation sets. Figure 6 shows the loss functions and accuracy scores of the two models.



(a) (b)

Figure 4: Training loss and accuracy of the two baseline models: (a) MobileNetV2 and (b) InceptionV3.

### 4.2. Results and discussion

This part is dedicated to the presentation of the results of all the studied methods. We have computed the classification accuracy on the test set. Table 5 shows the accuracy results of our method, MobileNetV2, and InceptionV3. As can be observed, the proposed method significantly outperforms MobileNetV2 with a substantial gap of 6.5%. Our model is even superior to the heavy-weight InceptionV3 model with an improvement in accuracy of 2.6%. The obtained result consistently confirms the performance of our method as we have achieved the same scores on the validation set (i.e., 97.1%). For the impact of classification head, it was found that using dense layer gives a slight improvement of accuracy as indicated by the gap between the MobileNetV2 and MobileNetV2b, for instance.

Table 5: Classification results and inference time

| Model | Accuracy | FPS (GPU) | FPS (CPU) |
|---|---|---|---|
| MobileNetV2 | 91.1% | 53.5 | 17.1 |
| MobileNetV2b | 90.4% | 50.2 | 17.0 |
| InceptionV3 | 95.0% | 31.1 | 14.7 |
| InceptionV3b | 94.8% | 28.6 | 14.3 |
| Our method | 97.6% | 130.9 | 113.5 |

Figure 5 provides more insight of our method in the means of confusion matrix of recognition performance for 56 object classes. As can be seen, the proposed network is able to give perfect accuracies (100%) for many classes. There are only a few animals or plants that the proposed method works less effectively, for example, the following species IDs: 10 (81%), 14 (88%), 21 (85%), and 23 (87%). To further explore the source of this drop, we have manually checked the images in the test set of these classes and it was found out that the number of testing images is relatively small (e.g., around 10 images). Consequently, if one or two images are wrongly classified, the corresponding accuracy is just near 80%.

Table 5 also presents the empirical running times of three models on a GPU (GTX 1070 Ti) and a CPU (Core i7-7700). Here FPS is computed as the mean value from the running times of processing all images in the test set. For each image, the following tasks are taken into account when measuring the processing times: image loading from external disk, size normalization, and inference. To have stable timing measures, we have reported the results after three runs of model. Our method works very efficiently with the inference speed of around 130.9 FPS on a moderate GPU configuration or 113.5 FPS on a CPU machine. The proposed method is about two times faster than the efficient MobileNetV2. Especially, when working on a CPU enviroment, the proposed model even achieves more appealing speeds. The obtained results are very promising even thought the network design is conceptually simple and straightforward.

In addition, we also conducted an experiment to verify the efficiency of the proposed model when running on mobile platforms. To this aim, we have converted the trained model to the one that can work on mobile environment with TensorFlow.js library[5]. A simple application has been developed using React Native[6] and deployed on a mobile configuration

[5] https://www.tensorflow.org/js/guide/conversion

[6] https://reactnative.dev/

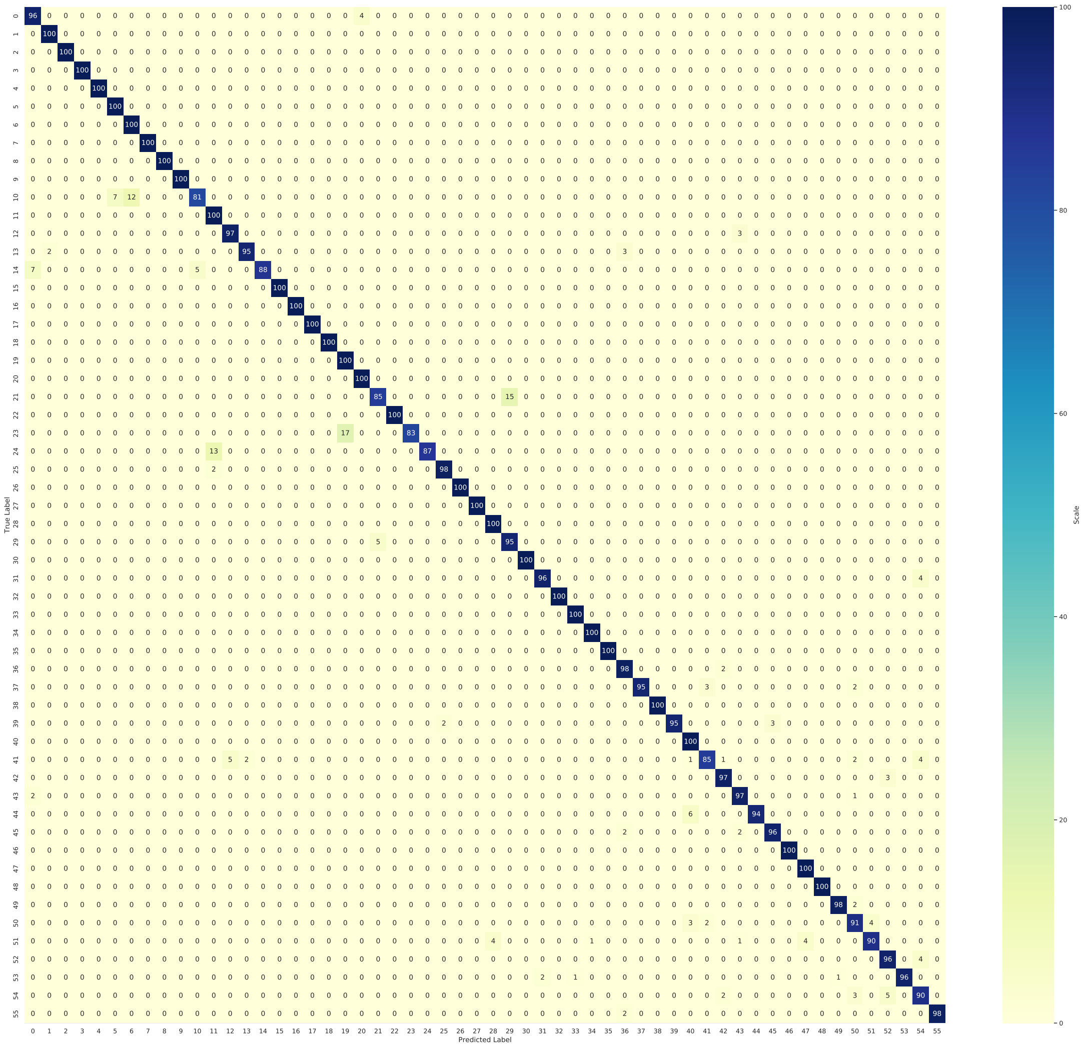
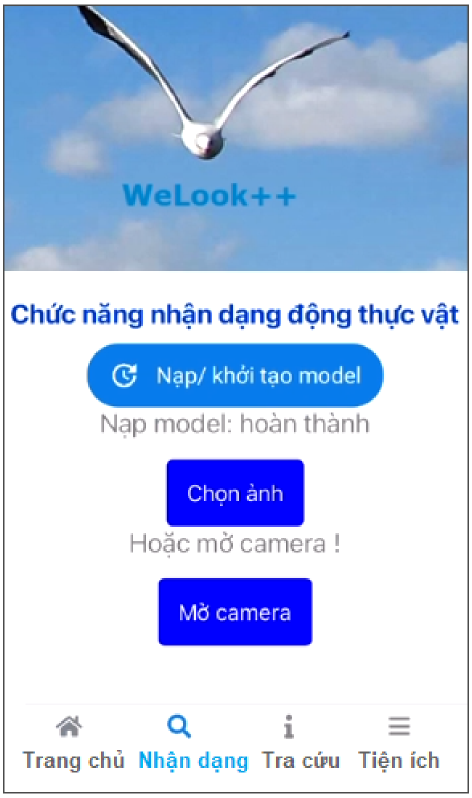


Figure 5: Confusion matrix of our method. The labels of animals and plants are derived from those in Figure 2.

(i.e., iPhone XR) by using the client tool of Expo Go. Figure 6 shows a report of running times given an input image having the size of 800 × 800. We provide here the timings for different steps, including: size normalization (i.e., resizing to model's input size of $192 \times 192$), image decoding (e.g., JPEG, PNG, or BMP), and finally inference for label prediction. As can be seen, the proposed model works very efficient with the total running times of 295 milliseconds that is appropriate for deployment in real life scenarios.

## 5. CONCLUSIONS

In this work, we have proposed an effective CNN architecture to handle the problem of image classification specific to the context of protecting the endangered species. The proposed method exploits the multi-scale feature representation in both the dimensions of



Chức năng nhận dạng động thực vật

Nạp model: hoàn thành

Hoặc mở camera !

Figure 6: The demonstrated application of animal and plant recognition

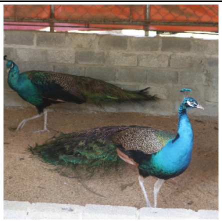
| Input image (800×800) | Running times (milliseconds) | | | | Predicted label |
|---|---|---|---|---|---|
| | Size normalization | Image decoding | Inference | Total | |
|  | 86 | 135 | 74 | 295 | Pavo muticus Delacour |

Figure 7: Demonstrated running times on mobile device of the proposed method

network width and depth. The former is accomplished by employing the Inception residual network that is composed of convolutions with different filter sizes (e.g., 1 × 1 and 3 × 3). The latter is handled by using more convolutions of stride 1 to create a deeper network. To make the model small and efficient, we carefully design and minimize the number of channels to every convolutional layer. Apart from the proposed model, we also perform transfer learning of the two state-of-the-art methods, MobileNetV2 and InceptionV3, on our specific dataset. Experimental results showed that our method outperforms the two others in terms of classification accuracy as well as processing time. Especially, the proposed network can work on CPU environment with real-time speed. The obtained results enable our CNN model to be deployment on portable computation devices. A next plan of fully optimizing the mobile application of this work has been scheduled and will be released soon.

## ACKNOWLEDGEMENTS

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2020.02

## REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556 [cs.CV]*, 09 2014.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *arXiv:1512.00567 [cs.CV]*, 2015.

[5] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *arXiv:1707.01083 [cs.CV]*, 2017.

[6] 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 ¡0.5mb model size," *arXiv:1602.07360 [cs.CV]*, 2016.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861 [cs.CV]*, 2017.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *arXiv:1801.04381 [cs.CV]*, 2019.

[9] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, 2010, p. 807–814.

[11] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv:1207.0580 [cs.NE]*, 2012.

[12] A. Berg, J. Deng, and L. Fei-Fei, "Large scale visual recognition challenge 2010," 2010. [Online]. Available: www.imagenet.org/challenges

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167 [cs.CV]*, 2015.

[15] L. G. Hafemann, L. S. Oliveira, and P. Cavalin, "Forest species recognition using deep convolutional neural networks," in *2014 22nd International Conference on Pattern Recognition*, 2014, pp. 1103–1107.

[16] Z. Miao, K. M. Gaynor, J. Wang, Z. Liu, O. Muellerklein, M. S. Norouzzadeh, A. McInturff, R. C. K. Bowie, R. Nathan, S. X. Yu, and W. M. Getz, "Insights and approaches using deep learning to classify wildlife," *Scientific Reports*, vol. 9, no. 1, pp. 1–9, 2019.

[17] M. Willi, R. T. Pitman, A. W. Cardoso, C. Locke, A. Swanson, A. Boyer, M. Veldthuis, and L. Fortson, "Identifying animal species in camera trap images using deep learning and citizen science," *Methods in Ecology and Evolution*, vol. 10, no. 1, pp. 80–91, 2019.

[18] 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*, ser. AAAI'17. AAAI Press, 2017, pp. 4278—-4284.

[19] T.-A. Pham, "Semantic convolutional features for face detection," *Machine Vision and Applications*, vol. 33, no. 3, pp. 1–18, 2021. [Online]. Available: https://doi.org/10.1007/s00138-021-01245-y

[20] S. Zhang, X. Wang, Z. Lei, and S. Z. Li, "Faceboxes: A cpu real-time and accurate unconstrained face detector," *Neurocomputing*, vol. 364, pp. 297–309, 2019.

*Received on July 20, 2021*
*Accepted on December 29, 2021*