# MobileNetV3 for Image Classification

Siying Qian[1st, *]
*College of Engineering and Computer Science*
*The Australian National University*
Canberra, Australia
*siying.qian@anu.edu.au

Yuepeng Hu[1st]
*School of Data Science*
*University of Science and Technology of China*
Hefei, China
huyp@mail.ustc.edu.cn

Chenran Ning[1st]
*School of Computer Science and Engineering*
*Fudan University*
Shanghai, China
17307130178@fudan.edu.cn

[1st] These authors contributed equally.

***Abstract*—Convolutional neural network (CNN) is a kind of deep neural networks, which extracts image features through multiple convolution layers and is widely used in image classifications. With the increasing number of image data processed by mobile devices, application of neural network for mobile terminals becomes popular. However, these networks need massive computation and advanced hardware support, making them difficult to adapt to mobile devices. This paper demonstrates that MobileNetV3 can get a superior balance between efficiency and accuracy for real-life image classification tasks on mobile terminals. In our experiments, classification performances are compared among MobileNetV3 and several other commonly used pre-trained CNN models on various image datasets. The chosen datasets are all good representatives of the application scenarios of mobile devices. The result shows that as a lightweight neural network, MobileNetV3 achieved good accuracy performance in an effective manner compared to other large networks. Furthermore, ROC confirmed the advantages of MobileNetV3 over other experimented models. Some conjectures are also brought out about the characteristics of image datasets that are suitable for MobileNetV3.**

***Keywords-Convolutional neural network; Image classification; Mobile devices; MobileNetV3***

## I. INTRODUCTION

Convolutional Neural Network (CNN) is recently given great attention because of its extended applications in image classification [1], segmentation, and other computer vision problems. CNN usually consists of 2 parts: features extracting part – made of convolutional layers and pooling layers, and classifying part – which contains lots of stacked fully connected layers. In the first part, kernels in convolutional layers scan the input image step by step, multiplying the weights in each kernel by the pixels' values and combining the sum to create a new image passed to the next layer. Pooling layers play a role in down-sampling to reduce the number of data and save computational resources. In the second part, the image first passes through a flatten layer to be converted to a one-dimensional array. The following fully connected layers use this array as input and produce the predicted label by applying the linear combination and the non-linear activation function.

Because of its advantage of extracting deep features layers by layers, nowadays, CNNs are widely used to solve real-world problems, such as auto-driving and medical image diagnosis. With a large number of tasks in daily scenarios, it comes very naturally to apply deep learning to daily tasks on mobile devices. Successfully applying CNN to mobile terminals is thus very important. Due to the limited computational power on mobile devices, model efficiency and small memory are highly required.

There are numerous CNN architectures designed for different tasks. Among them, some are often used in image classification problems, such as Inception [2], AlexNet [3] and VGG16 [4]. However, classification tasks in mobile devices require not only high accuracy but also, more practically, low memory costs and high computation efficiency, which give rise to models suitable for mobile terminals. We choose MobileNetV3 [5] as our main study object, which are lightweight CNN models with a good balance between accuracy and efficiency.

In this paper, we compare the performance of MobileNetV3, AlexNet, InceptionV3 [6] and ShuffleNetV2 [7] on different datasets, and validate the efficiency and adaptability of MobileNetV3. The main contributions of this paper can be summarized as follows:

1. Fruits 360 [8], 10 Monkey Species and Bird Species Classification, as the representations of common image datasets on mobile devices, are used to train and evaluate the chosen neural networks' performance.
2. MobileNetV3, AlexNet, InceptionV3 and ShuffleNetV2 are applied to multiple image classification tasks related to mobile scenarios.
3. Multiple evaluation metrics are adopted to analyse the performance of different models on various datasets.
4. Based on the experiment results, the characteristics of MobileNetV3 on image classification and its advantages on mobile devices are summarized.

The rest of this paper is organized as follows. In Section II, we describe the main structure of MobileNetV3 and the characteristics of other models in the experiment. In Section III, datasets are introduced, and the performance of MobileNetV3 and other models is evaluated. The experiment results and analysis are shown in Section III as well. There is a summarization of the characteristics of MobileNetV3 and a discussion of our future work in Section IV.

978-0-7381-3122-1/21/$31.00 ©2021 IEEE

## II. Model Formulation

CNN networks for mobile terminals are developing rapidly in recent years. The three versions of mobile nets have kept being improved in architecture from 2017 to 2019. MobileNetV1 [9] is developed referencing from the traditional VGG architecture while introducing depthwise separable convolutions. Based on that, MobileNetV2 [10, 11] is introduced one year later with linear bottleneck and inverted residual. With the help of NAS and NetAdapt network searching for architecture optimization, MobileNetV3 is further developed by dropping expensive layers and using h-swish non-linearity function instead of ReLU to increase its efficiency and relative accuracy at the same time in mid of 2019.

According to high or low resource use cases' targets, MobileNetV3 are defined as MobileNetV3-Small and MobileNetV3-Large two models with different architecture complexities.

TABLE I MobileNetV3-Large specification. #expand denotes the number of convolutional filters that are used to expand the feature space from the layer's input. SE indicates whether there is a Squeeze-and-Excite in that block or not. NL denotes the non-linearity type for that block. BN means batch normalization.

| Input | Operator type | Kernel size | #exp and | #out | str ide | S E | NL |
|---|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | - | 16 | 2 | - | *h-swish* |
| $112^2 \times 16$ | bottleneck | $3 \times 3$ | 16 | 16 | 1 | - | *ReLU* |
| $112^2 \times 16$ | bottleneck | $3 \times 3$ | 64 | 24 | 2 | - | *ReLU* |
| $56^2 \times 24$ | bottleneck | $3 \times 3$ | 72 | 24 | 1 | - | *ReLU* |
| $56^2 \times 24$ | bottleneck | $5 \times 5$ | 72 | 40 | 2 | ✓ | *ReLU* |
| $28^2 \times 40$ | bottleneck | $5 \times 5$ | 120 | 40 | 1 | ✓ | *ReLU* |
| $28^2 \times 40$ | bottleneck | $5 \times 5$ | 120 | 40 | 1 | ✓ | *ReLU* |
| $28^2 \times 40$ | bottleneck | $3 \times 3$ | 240 | 80 | 2 | - | *h-swish* |
| $14^2 \times 80$ | bottleneck | $3 \times 3$ | 200 | 80 | 1 | - | *h-swish* |
| $14^2 \times 80$ | bottleneck | $3 \times 3$ | 184 | 80 | 1 | - | *h-swish* |
| $14^2 \times 80$ | bottleneck | $3 \times 3$ | 184 | 80 | 1 | - | *h-swish* |
| $14^2 \times 80$ | bottleneck | $3 \times 3$ | 480 | 112 | 1 | ✓ | *h-swish* |
| $14^2 \times 112$ | bottleneck | $3 \times 3$ | 672 | 112 | 1 | ✓ | *h-swish* |
| $14^2 \times 112$ | bottleneck | $5 \times 5$ | 672 | 160 | 2 | ✓ | *h-swish* |
| $7^2 \times 160$ | bottleneck | $5 \times 5$ | 960 | 160 | 1 | ✓ | *h-swish* |
| $7^2 \times 160$ | bottleneck | $5 \times 5$ | 960 | 160 | 1 | ✓ | *h-swish* |
| $7^2 \times 160$ | conv2d | $1 \times 1$ | - | 960 | 1 | - | *h-swish* |
| $7^2 \times 960$ | pool | $7 \times 7$ | - | - | 1 | - | - |
| $1^2 \times 960$ | conv2d (no BN) | $1 \times 1$ | - | 1280 | 1 | - | *h-swish* |
| $1^2 \times 1280$ | conv2d (no BN) | $1 \times 1$ | - | $k$ | 1 | - | - |

The full specification of MobileNetV3-Large is shown in Table I.

According to the detailed layer specification, the model's overall architecture can be specified as Figure 1.

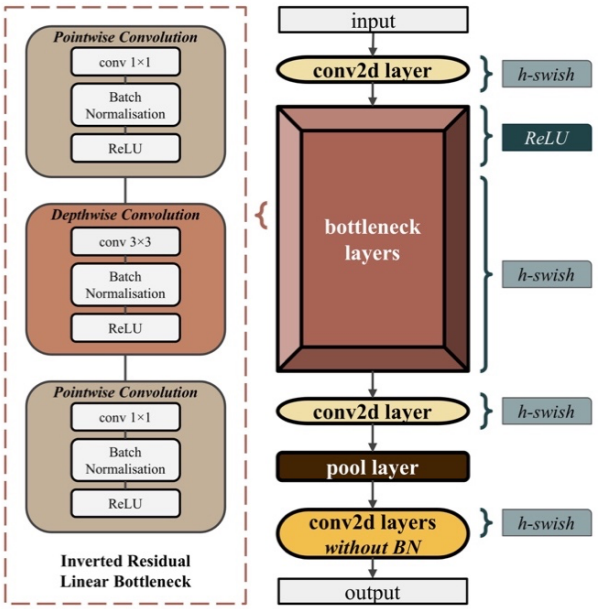



Figure 1 MobileNetV3 architecture. The general architectures are the same for both MobileNetV3-Large and MobileNetV3-Small.

The general structure of MobileNetV3-Small is almost the same as MobileNetV3-Large; besides that, MobileNetV3-Small is four layers shortened, as shown in Table II.

TABLE II MobileNetV3-Small specification. All notations are the same as Table 1.

| Input | Operator type | Kernel size | #exp and | #out | str ide | S E | NL |
|---|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | $3 \times 3$ | - | 16 | 2 | - | *h-swish* |
| $112^2 \times 16$ | bottleneck | $3 \times 3$ | 16 | 16 | 2 | ✓ | *ReLU* |
| $56^2 \times 16$ | bottleneck | $3 \times 3$ | 72 | 24 | 2 | - | *ReLU* |
| $28^2 \times 24$ | bottleneck | $3 \times 3$ | 88 | 24 | 1 | - | *ReLU* |
| $28^2 \times 24$ | bottleneck | $5 \times 5$ | 96 | 40 | 2 | ✓ | *h-swish* |
| $14^2 \times 40$ | bottleneck | $5 \times 5$ | 240 | 40 | 1 | ✓ | *h-swish* |
| $14^2 \times 40$ | bottleneck | $5 \times 5$ | 240 | 40 | 1 | ✓ | *h-swish* |
| $14^2 \times 40$ | bottleneck | $5 \times 5$ | 120 | 48 | 1 | ✓ | *h-swish* |
| $14^2 \times 48$ | bottleneck | $5 \times 5$ | 144 | 48 | 1 | ✓ | *h-swish* |
| $14^2 \times 48$ | bottleneck | $5 \times 5$ | 288 | 96 | 2 | ✓ | *h-swish* |
| $7^2 \times 96$ | bottleneck | $5 \times 5$ | 576 | 96 | 1 | ✓ | *h-swish* |
| $7^2 \times 96$ | bottleneck | $5 \times 5$ | 576 | 96 | 1 | ✓ | *h-swish* |
| $7^2 \times 96$ | conv2d | $1 \times 1$ | - | 576 | 1 | ✓ | *h-swish* |
| $7^2 \times 576$ | pool | $7 \times 7$ | - | - | 1 | - | - |
| $1^2 \times 576$ | conv2d (no BN) | $1 \times 1$ | - | 1280 | 1 | - | *h-swish* |
| $1^2 \times 1280$ | conv2d (no BN) | $1 \times 1$ | - | $k$ | 1 | - | - |

### A. Depthwise Separable Convolution

To compute more efficiently, depthwise separable convolution is introduced. It is very similar to the traditional convolution. While unlike the traditional convolution with only one convolutional calculation for each layer, the convolutional calculation of the depthwise separable convolution is divided into two phases. In the first phase, depthwise convolution applies a single convolutional filter for each input channel. In the second phase, pointwise convolution (a $1 \times 1$ convolution) is applied to all the channels of outputs of the depthwise convolution. Altogether, the depthwise separable convolution improves the

calculating speed by reducing the computation amount, though it will sacrifice a little accuracy. Depthwise separable convolution is a core technique for many efficient models, such as MobileNetV1 - V3.

### B. Linear Bottleneck

To extract features from high-dimensional space without losing too much information, MobileNetV2 proposes linear bottlenecks to reduce the dimensionality of input. Linear bottleneck refers to a bottleneck layer, which is a convolutional layer with a $1 \times 1$ filter, combined with the linear activation function. Because the traditional ReLU function transformation provides non-linearity with a possibility of information loss, MobileNetV2 turns to insert linear bottleneck layers into convolutional blocks instead (assuming that the flow of features is low-dimensional and capturable).

### C. Inverted Residual

As a safer and more efficient way to extract all necessary information of the input data, bottleneck layers replace the ReLU layers. There is also an expansion layer at the beginning of the bottleneck block. Additionally, MobileNetV2 uses shortcuts directly between bottlenecks to better propagate gradients across multiple layers and prevent gradients loss and explosion. An inverted residual block is tested valid to act almost the same as a residual block, while it reduces memory costs considerably at the same time.

### D. Network Architecture Search

With reinforcement learning and recurrent neural network (RNN), network architecture search is applied to MobileNetV3 to determine the optimal architecture for a constrained hardware platform. It is a method that constructs a search space for the architecture of neural networks and searches efficiently in a hierarchical search space with reinforcement learning to approach the best structure of the model for specific tasks. For instance, the expansion layer of MobileNetV3 is redesigned as Figure 2, based on the original design for MobileNetV2.

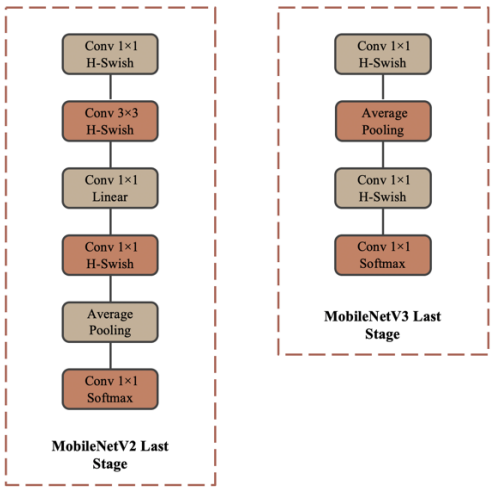



Figure 2 Comparison of original last stage and redesigned last stage.

Because of the similarity in the RNN controller and search space, MobileNetV3 uses MnasNet-A1 [13] as the initial model. Additionally, the reward design in reinforcement learning is modified to fit small mobile models better. After the types of layers are fixed, NetAdapt [14], a method for fine-tuning the hyper-parameters in each layer, is used to optimize the model.

### E. Swish Function

For higher accuracy, a new kind of activation function called swish is introduced to replace the ReLU function. This function is defined as:

$$swish(x) = x \bullet \sigma(x) \tag{1}$$

However, the sigmoid function in the swish formula may cost a large number of computational resources on mobile devices. To solve this problem, the authors of MobileNetV3 use ReLU6 function to approximate the sigmoid function in swish and produce an approximation of swish function called the hard version of swish (h-swish), defined as:

$$h-swish[x] = x\frac{\mathrm{Re}\,LU6(x+3)}{6} \tag{2}$$

### F. Additional Models in Experiments

AlexNet is a very initial deep CNN starting to use ReLU non-linearity instead of the tanh function and can be put on multiple GPUs. It is excellent to apply for high-resolution images. However, the major disadvantage of the model is a severe overfitting problem due to the massive amounts of parameters.

InceptionV3 first applied Batch Normalization layers to the propagation of data to accelerate the process of gradient descending. The inception modules in InceptionV3 are optimized and have more branches compared to InceptionV2 [16]. Furthermore, the key insight that factorization into small convolution, which means separating a 2-dimensional convolution kernel into two 1-dimensional ones, is implemented in InceptionV3.

ShuffleNet [15] uses group convolution and channel shuffle to reduce computational complexity. Features are extracted by depthwise separable convolutional layers, which contribute to lower computational requirement and higher speed of data propagation without significant decrease of accuracy.

## III. Experiments

### A. Datasets

In this paper, we used three different datasets downloaded from Kaggle. These datasets are to simulate daily scenarios where MobileNetV3 can be applied to mobile devices.

As shown in Figure 3, in row one, the first dataset is a fruit-360 dataset, including 131 fruit categories and 67692 training images. Each fruit is shot from different angles multiple times and in a clear background. Row two in Figure 3 demonstrates the second dataset, 1097 images of 10 monkeys. The third dataset contains 16 categories of birds and 150 images overall, shown in row three. This dataset is small and imbalanced, and birds are distributed randomly in various backgrounds.

Figure 3 Samples from three experimented datasets. The first row is fruit-360 dataset; the second row is monkey dataset; the third row is bird dataset.

As for data pre-processing, we performed the same operations for the three datasets: firstly, the training dataset was randomly split into a training subset and a validation subset, with the split ratio of 0.7; secondly, all images were resized to $224 \times 224$ and normalized as the models' input except for InceptionV3. Input images were resized to $299 \times 299$ for InceptionV3.

## B. Experiment Steps

In this paper, we compared the performance of 5 models (MobileNetV3-Large, MobileNetV3-Small, AlexNet, InceptionV3 and ShuffleNetV2) on 3 different datasets.

The comparisons of FLOPs and parameters' numbers among these 5 models are shown in Table III.

TABLE III COMPARISON OF FLOPS AND THE NUMBER OF PARAMETERS AMONG DIFFERENT MODELS.

| Model | *MobileNetV3-Large* | *MobileNetV3-Small* | *AlexNet* | *InceptionV3* | *ShuffleNetV2* |
|---|---|---|---|---|---|
| ***FLOPs (Millions)*** | 226.0 | 59.65 | 714.7 | 5731 | 148.8 |
| ***Parameters (Millions)*** | 5.48 | 2.54 | 61.1 | 23.8 | 2.28 |

The datasets include daily-life image classification problems related to fruit, bird and monkey.

We replaced the last classify layers for each model according to the number of unique classification classes for different datasets. Pre-trained weights [17, 18] on ImageNet [19] were used. The models were then fine-tuned by setting all the layers trainable. Using pre-trained weights made the training procedure easier and led to faster convergence. Fine-tuning the models was to adapt models to specific given tasks. The experiment procedure on the three datasets followed the same steps.

## C. Evaluation Metrics

### 1) Accuracy:

To evaluate the performance of models on a dataset, the number of samples predicted to be positive, which are actually positive, and the number of samples predicted to be negative, which are actually negative, are both calculated. It reflects the precision of models distinguishing true positive samples and true negative samples from the whole dataset.

Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where *TP*, *TN*, *FP*, *FN* are the number of true-positive, true-negative, false-positive and false-negative samples, respectively.

### 2) Cross-Entropy Loss:

In the stage of training a neural network, we usually define a target function and convert the training process to an optimization problem. To measure the turbulence of the neural network's predictions objectively and provide a reasonable direction in the searching space for us to optimize, Cross-Entropy is adopted as the loss function in the training stage of models in the rest of the paper.

Cross-Entropy Loss is defined as:

$$CrossEntropyLoss = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_{ij}) \log(q(x_{ij})) \tag{4}$$

where $N$ is the number of samples in the training set; $M$ is the number of classes in the training set; $p(x_{ij})$ is the true probability of the $i$th sample belonging to the $j$th class; $q(x_{ij})$ is the probability of the $i$th sample belonging to the $j$th class produced by the model.

### 3) ROC Curves:

To visually show the trade-off between the true positive rate (*TPR*) and the false positive rate (*FPR*), Receiver Operating Characteristics (ROC) curves are drawn, where $TPR = TP / P$ is the proportion of positive samples that are correctly labelled by the model and $FPR = FP / N$ is the proportion of negative samples that are mislabelled as positive.

The area under a ROC curve is often used to measure the performance of a model. The diagonal line on the graph represents a model that randomly labels the samples. A model better than random should appear above the diagonal. The further a model is to random (i.e., when the area under a ROC curve is much larger than 0.5), the model is better.

## D. Experimental Results

### 1) Experiments on Fruit Dataset:

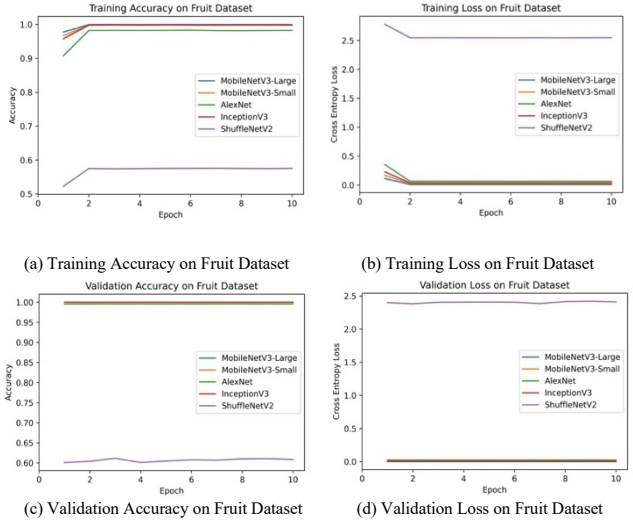
Fruit dataset contains 90483 images from 131 fruit categories. All training and testing photos of one category of fruit are captured from different angles of a single fruit representative from that category with a white background. It means there is no apparent difference between training and testing data of one specific category. The training size is large enough, and the fruit in the images are nearly without any background interference. There is a large number of testing data as well. The testing results can thus be considered convincing.

The fine-tuning training time is exceptionally long for fruit datasets, as shown in Table IV, compared to the experiments on the other two datasets. It may be due to the large dataset and the complex fruit varieties.

TABLE IV APPROXIMATE FINE-TUNING TRAINING TIME ON FRUIT DATASET FOR DIFFERENT EXPERIMENTED MODELS. THE ABSOLUTE TIME HIGHLY DEPENDS ON THE TRAINING ENVIRONMENT, WHILE THE TIME COMPARISON AMONG MODELS IS RELATIVELY MEANINGFUL.

| Model | *Approximate Training Time (in minutes)* |
|---|---|
| MobileNetV3-Large | 40 |
| MobileNetV3-Small | 33 |
| AlexNet | 417 |
| InceptionV3 | 225 |
| ShuffleNetV2 | 42 |

Though the training time is relatively longer, the actual necessary number of training epochs for fruit dataset is smaller to achieve good performance for fruit dataset. In Figure 4, all of MobileNetV3-Large, MobileNetV3-Small, AlexNet and InceptionV3 can reach nearly 1 as accuracy from the first epoch. Only the ShuffleNetV2 model shows much lower validation accuracy in the whole training process. By comparing training and validation evaluation, the validation accuracy is even generally higher than training accuracy for all models. No overfitting problem thus exists in these experiments.




(a) Training Accuracy on Fruit Dataset (b) Training Loss on Fruit Dataset

(c) Validation Accuracy on Fruit Dataset (d) Validation Loss on Fruit Dataset

Figure 4 Comparison of training and validation accuracy and loss throughout the fine-tuning training process on the fruit dataset experiment.

From both the test accuracies and ROC curves comparisons among all experimented models on the fruit dataset, the same result is concluded: On the one hand, MobileNetV3-Large achieves the best performance, and MobileNetV3-Small stays closely at the second place. InceptionV3 has a similar while slightly lower accuracy. Although AlexNet ranks fourth for this dataset, its accuracy is still high. On the other hand, the performance of ShuffleNetV2 is not satisfied, with the 0.79 test accuracy and an obvious classification ability gap compared to the other models. It is also clearly shown in Figure 5 that all models' performances are much better than a random guess.

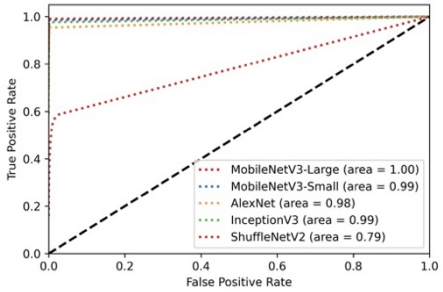



Figure 5 Comparison of ROC curves for different models on fruit dataset.

Considering fine-tuning training time, testing time, and the final performance for all models on the dataset, ShuffleNetV2 corresponds to high efficiency while low accuracy. AlexNet and InceptionV3 can achieve high accuracy with much higher cost on both time and resources. Only the two MobileNetV3 models achieved an outstanding balance between accuracy and efficiency as expected. MobileNetV3 not only cost the shortest time but also brought the best classification performance.

*2) Experiments on Monkey Dataset:*

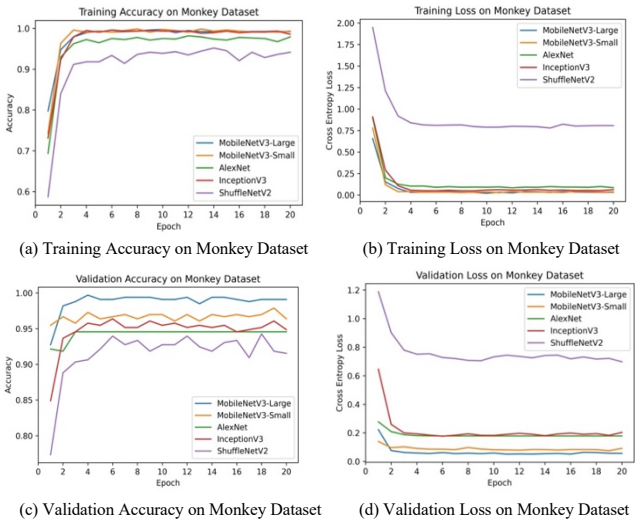
Monkey dataset consists of almost 1400 pictures of 10 species of monkeys. Since the photos produced by mobile devices are in relatively low resolution and animals are the objects that mobile devices may commonly capture, we picked up this dataset to validate the performance of MobileNetV3 in the classification of animals' graphics in low resolution. Each image in the dataset contains a monkey from one of those ten species. The background of the image is the place the monkey lives in. The monkeys are usually in the centre of the pictures, and the backgrounds are blurry.

In each class of the dataset, the number of images is almost the same. It can help to prevent over-fitting problem caused by the imbalance of the training dataset. Additionally, there are nearly 30 pictures in each class to validate the trained models, showing each model's performance objectively. Labels and the number of images from each category are shown in Table V.

TABLE V NUMBER OF IMAGES IN EACH CLASS OF MONKEY DATASET.

| Label | *Training images* | *Testing images* |
|---|---|---|
| Mantled howler | 105 | 26 |
| Patas monkey | 111 | 28 |
| Bald uakari | 110 | 27 |
| Japanese macaque | 122 | 30 |
| Pygmy marmoset | 105 | 26 |
| White headed capuchin | 113 | 28 |
| Silvery marmoset | 106 | 26 |
| Common squirrel monkey | 114 | 28 |
| Black-headed night monkey | 106 | 27 |
| Nilgiri langur | 106 | 26 |

In the experiments, the training images were split into two parts: a training set consisting of 770 images of monkeys and a validation set containing 328 images. Before the training stage, all the pictures were rescaled to $224 \times 224$ pixels to fit the input size of most convolutional neural network (except $299 \times 299$ pixels for InceptionV3). During training, we loaded the weights from ImageNet transfer learning and used them as the initial weights.




(a) Training Accuracy on Monkey Dataset (b) Training Loss on Monkey Dataset

(c) Validation Accuracy on Monkey Dataset (d) Validation Loss on Monkey Dataset

Figure 6 Comparison of training and validation accuracy and loss throughout the fine-tuning training process on the monkey dataset experiment.

As shown in Figure 6, both MobileNetV3-Large and MobileNetV3-Small reached convergence within 4 epochs and achieved even the same high training accuracy as InceptionV3, which is a considerably larger neural network than MobileNetV3. Compared to ShuffleNetV2 that also designed for mobile devices, MobileNetV3 converged at a higher accuracy within fewer epochs, which shows that MobileNetV3 is easier to be trained in this monkey dataset and some other similar animals' datasets in low resolution.

At the testing stage, all the five trained models are applied to the test set. The results of each model's performance are shown in Figure 6.

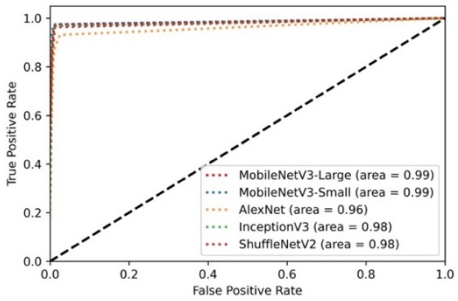



Figure 7 Comparison of ROC curves for different models on monkey dataset.

MobileNetV3-Large reached the highest testing accuracy among all the five models and is followed by MobileNetV3-Small. Figure 7 illustrates the same result. It reveals that MobileNetV3 can perform well in extracting features in low-resolution images and be applied to image classification. According to the huge number of the FLOPs and parameters (as shown in Table 3), InceptionV3 should have the best generalization ability. However, it didn't perform as expected. We suppose the reason lies in the training size of the monkey dataset is too small for such a large model to reach its best performance, which also shows that MobileNetV3 can be trained with small datasets and achieve high accuracy. According to the huge number of the FLOPs and parameters (as shown in Table 3), InceptionV3 should have the best generalization ability. However, it didn't perform as expected. We suppose the reason lies in the training size of the monkey dataset is too small for such a large model to reach its best performance, which also shows that MobileNetV3 can be trained with small datasets and achieve high accuracy.

To test each model's efficiency, we recorded the time of all five models to predict all labels in the test set. The results and analysis are shown together with the other two datasets in Section III *D.4* Figure 10.

*3) Experiments on Bird Dataset:*

The bird dataset is difficult to be trained and easily becomes overfitting because it only contains 150 images and has less than 10 images for each category. The various backgrounds and the small birds' sizes in the entire large images made experiments on this dataset even harder to reach a good result. Compared to the other two datasets, bird dataset required the least training time due to its smallest size.

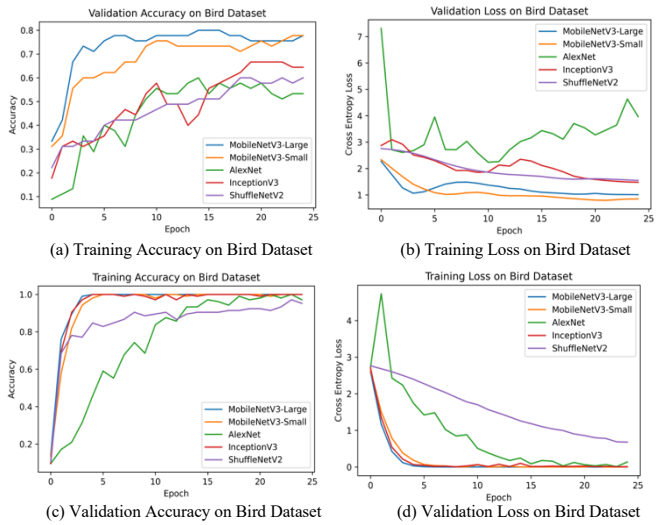
TABLE VI FINE-TUNING TRAINING TIME ON BIRD DATASET FOR DIFFERENT EXPERIMENTED MODELS. THE ABSOLUTE TIME IS ALMOST THE SAME. THE BEST VALIDATION ACCURACY IS RECORDED DURING TRAINING.

| Model | Training Time | Best Validation Accuracy |
|---|---|---|
| MobileNetV3-Large | 14min 1s | **0.80** |
| MobileNetV3-Small | 14min 18s | 0.78 |
| AlexNet | 14min 13s | 0.60 |
| InceptionV3 | 14min 26s | 0.67 |
| ShuffleNetV2 | 14min 6s | 0.60 |

As shown in Table VI, the training time of all models is almost the same. It shows that time may not be a good comparison metric here because the time gaps among models for bird dataset are not as significant as those for fruit dataset. However, MobileNetV3-Large and MobileNetV3-Small still obtained much higher validation accuracy than other models within the same training time as the other two datasets. More accuracy and loss details in the experiment process are shown in Figure 8.

Figure 8 demonstrates that all models show different degrees of overfitting on bird dataset. This is mainly due to the imbalanced training data for each category and the small number of data in bird dataset. According to the training loss trends in Figure 8 (b), all models came to good convergence in the training process except ShuffleNetV2. Figure 8 (d) for validation loss shows that AlexNet failed at this classification task. Moreover, AlexNet, InceptionV3, and ShuffleNetV2 encountered heavy overfitting after 10 training epochs. MobileNetV3-Large and MobileNetV3-Small converged faster and are less overfitting.

The validation accuracy of these two models is generally higher than others.




(a) Training Accuracy on Bird Dataset (b) Training Loss on Bird Dataset

(c) Validation Accuracy on Bird Dataset (d) Validation Loss on Bird Dataset

Figure 8 Comparison of training and validation accuracy and loss throughout the fine-tuning training process on the bird dataset experiment.

Figure 9 also demonstrates that MobileNetV3-Large achieved the best performance among all models. MobileNetV3-Small had almost the same great performance, measured by the area under the ROC curve.

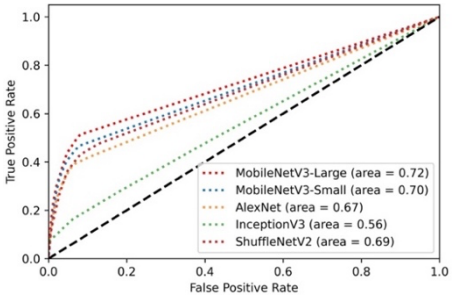



Figure 9 Comparison of ROC curves for different models on bird dataset.

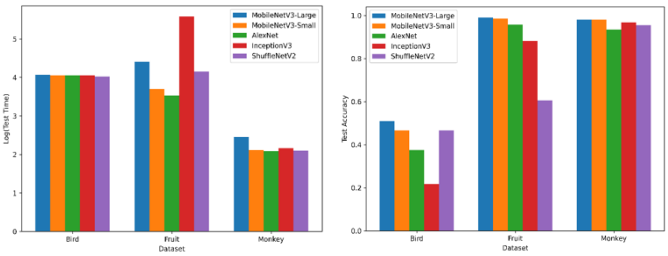
*4) Overall Testing Performance Comparison:*

From the above analysis for experiments of individual datasets, it can be found that MobileNetV3-Large can always achieve the best performance among all experimented models in the training process. MobileNetV3-Small usually comes as the second. Also, the training time of MobileNetV3 is significantly shorter than the other models.

We want to verify the above training-process results (in Section III *D.1* to III *D.3*) on testing performance again. To this end, experiments are conducted on testing data of the three datasets for all the five models. Figure 10 (a) and (b) demonstrate the test summary about efficiency and accuracy, respectively.

As shown in Figure 10 (b), both two models for MobileNetV3 kept the highest classification accuracy for the chosen datasets. MobileNetV3-Large performed slightly better than MobileNetV3-Small.

In addition to the classification accuracy, we also focus on model efficiency. Figure 10 (a) demonstrates that MobileNetV3 did not take significantly longer testing time than other experimented models.




(a) Log of Test Time on Datasets (b) Test Accuracy on Datasets

Figure 10 (a) Comparison of test time for all experimented models on the three different datasets. Due to the large difference in the original test size for the three datasets, we divide all testing time by the tested number in the corresponding dataset. (b) Comparison of test accuracy for all experimented models on the three different datasets.

From the accuracy and efficiency analysis in both training and testing process for all models, it can be seen that the two MobileNetV3 models take much shorter time (especially for fined-tuning training time) than large CNN models such as InceptionV3 and AlexNet, while achieving satisfying accuracy. MobileNetV3 is also slightly more efficient than other nets designed for mobile terminals such as ShuffleNetV2, but with much better performance.

Overall, our experiments' results validate that MobileNetV3 can reach high accuracy with a small amount of time resource, which means it is extremely suitable for mobile devices with constrained computational resources requiring high efficiency.

*5) Conjectures about Characteristics of Suitable Datasets:*

As a further step of our experiment results' analysis, we conclude some conjectures about what kinds of datasets are more suitable for MobileNetV3, by comparing performance for MobileNetV3 classifying the three different datasets.

From Figure 10 (b) and ROC curves for individual datasets (i.e., Figure 5, 7, and 9), it can be found that fruit-360 dataset leads to 1.0 accuracy in an extremely short time, and monkey dataset corresponds to a pretty good accuracy (higher than 0.95) after a few fined-tuning training epochs. However, bird dataset has far smaller accuracy (lower than 0.50) than the other two datasets.

Considering the characteristics of these datasets, fruit dataset is a representative of images of daily-life objects with no significant differences among a huge number of different categories. Training set for each category is large enough, and there is no background interference. Monkey dataset is made up of low-resolution animal images from only ten categories of monkeys. Monkeys that need to be classified are put in the centre of the images. Image backgrounds are environments that monkeys commonly live in, while they are blurry. Bird dataset has only a small number of images for ten categories of birds. Though those images are in high resolution, image backgrounds are messy, and the birds that need to be categorized are in tiny size of the entire images.

Thus, we suppose the characteristics of suitable image datasets for MobileNetV3 classification tasks are as follows:

- Classified objects are highlighted in the images with no strong background interference.
- Images have no high-resolution requirement (i.e., are friendly to low-resolution images).
- The training dataset is better in relatively large size for the classified category, which means the classified objects are better commonly seen in the training library.
- The classified object is in typical shape or state of its category, and the backgrounds are better to be common as well.
- There is no need to have apparent differences among various categories in the dataset, as CNN can capture features more precisely than eyes of human beings.

The experimental summary of characteristics of MobileNetV3 suitable datasets convinces that MobileNetV3 is perfectly designed for mobile devices and brilliant to handle classification tasks for daily-life photos in an extremely short time.

## IV. Conclusion

This paper compared the performance of image classification tasks among MobileNetV3 and certain standard CNN models based on the image datasets, which are usually captured and handled by mobile devices. By comparing and analysing the experimental results, we found that MobileNetV3 models can complete image classification tasks with much higher efficiency. At the same time, their final accuracy is significantly higher than the accuracy of other models. Besides, we note that MobileNetV3 is suitable for images that contain daily-life objects with less background interference, even if the images are in low resolution. Therefore, MobileNetV3 is very convenient to deal with image classification tasks for mobile terminals.

As further work to our research, we would like to summarize the advantages of other large CNNs that can be used to adjust the MobileNetV3 models further better, to improve their performance while avoiding the architecture to expand. We also plan to experiment on more image datasets with MobileNetV3 models, which aims to summarize and confirm more universal characteristics of images that perform excellently on MobileNetV3.

## References


[1] Al-Saffar A.A.M., Tao H. and Talab M.A., "Review of deep convolution neural network in image classification," International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET). IEEE, pp. 26-31, 2017.

[2] Szegedy C., Liu W., Jia Y., et al., "Going deeper with convolutions," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9, 2015.

[3] Krizhevsky A., Sutskever I. and Hinton G.E., "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[4] Simonyan K. and Zisserman A., "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[5] Howard A., Sandler M., Chu G., Chen L.C., Chen B., Tan M., Wang W., Zhu Y., Pang R., Vasudevan V. and Le Q.V., "Searching for mobilenetv3," Proceedings of the IEEE International Conference on Computer Vision, pp. 1314-1324, 2019.

[6] Szegedy C., Vanhoucke V., Ioffe S., Shlens J. and Wojna Z., "Rethinking the inception architecture for computer vision," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818-2826, 2016.

[7] Ma N., Zhang X., Zheng, H.T. and Sun, J., "Shufflenet v2: Practical guidelines for efficient cnn architecture design," Proceedings of the European Conference on Computer Vision (ECCV), pp. 116-131, 2018.

[8] Mureşan H. and Oltean M., "Fruit recognition from images using deep learning," Acta Universitatis Sapientiae, Informatica, vol. 10, no. 1, pp. 26-42, 2018.

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

[10] Sandler M., Howard A., Zhu M., Zhmoginov A. and Chen L.C., "Mobilenetv2: Inverted residuals and linear bottlenecks," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520, 2018.

[11] Xiang Q., Wang X., Li R., Zhang G., Lai J. and Hu Q., "Fruit image classification based on mobilenetv2 with transfer learning technique," Proceedings of the 3rd International Conference on Computer Science and Application Engineering, pp. 1-7, 2019.

[12] Hu J., Shen L. and Sun G., "Squeeze-and-excitation networks," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132-7141, 2018.

[13] Tan M., Chen B., Pang R., Vasudevan V., Sandler M., Howard A. and Le Q.V., "Mnasnet: Platform-aware neural architecture search for mobile," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2820-2828, 2019.

[14] Yang T.J., Howard A., Chen B., Zhang X., Go A., Sandler M., Sze V. and Adam H., "Netadapt: Platform-aware neural network adaptation for mobile applications," Proceedings of the European Conference on Computer Vision (ECCV), pp. 285-300, 2018.

[15] Zhang X., Zhou X., Lin M. and Sun J., "Shufflenet: An extremely efficient convolutional neural network for mobile device," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848-6856, 2018.

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

[17] Hussain M., Bird J.J. and Faria D.R., "A study on cnn transfer learning for image classification," UK Workshop on Computational Intelligence," vol. 840, pp. 191-202, 2018.

[18] Pan S.J. and Yang Q., "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, 2009.

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