Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Flower recognizer for mobile application

Authors:
Matteo Romanato - 816852 - m.romanato@campus.unimib.it
Lorenzo Pirola - 816418 - l.pirola13@campus.unimib.it
Youssef Karrati - 817435 - y.karrati@campus.unimib.it

January 21, 2021

Contents

List of Figures

Abstract

The aim of the project was to create a model suitable in a mobile context that can recognise a flower. To achieve this goal a flower dataset consisting of 102 classes was used. Three different fine-tuned architectures were compared and then they were compressed using iterative pruning and quantization. Analysing the results obtained, the final considerations were made, choosing the best model taking into account two fundamental aspects for the selected context: accuracy and size of the model.

1 Introduction

Flowers and the ability to identify them are very interesting topics that fascinating people of all ages and genders. Nevertheless, recognising and classifying a flower is not such an easy task in case of a low skills or knowledge.
Given the availability of digital and mobile technologies, it is easy to take a picture of a flower, but it is not so trivial to get information about it without knowing its name. So it can be seen that the missing piece is the link between obtaining an image of a flower and discover its name. This aspect was the starting point for the objective of this work: the aim of the project was to create a model for classifying flowers from images, which can be used for a mobile application to obtain information about them. So the problem addressed was the flowers classification, identifying them as members of different species. The classification was done by analysing the visual content of the images and assigning a label to each of them.
Given the objective of the project, which was to develop a model that handled the task of flowers recognition in a mobile context, there was two fundamental aspects to be based on: accuracy and size of the model. Once the dataset was retrieved and the necessary preprocessing operations was performed, three different architectures were chosen for comparison (MobileNetV2, DenseNet and EfficientNet) to identify which of the three was best suited to the above task. The first stage involved the fine-tuning of the three architectures just mentioned before and then comparison of the results obtained on them. During the second phase these models were pruned according to the magnitude-based weight pruning methodology, and then finally were applied a quantization technique. At the end, once the comparisons were made and the results obtained, the final choice and considerations were made.

2 Datasets

The dataset used for this project contained images of common flowers in the UK, belonging to 102 different categories. The images, totalling 8189, were collected mostly through web searches and to a small extent by taking photos manually. The dataset, called Oxford Flower Dataset, can be downloaded from the dedicated website [1].

[1] https://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html

Each class contained images ranging from a minimum of 40 to a maximum of 250. All the species of flowers in the dataset and the number of images related to each of them can be viewed on the dedicated site[2]. A property of this dataset that had made the classification task more challenging was the low inter-class variation, it means that there was several flowers belonging to different species that looked very similar[1].
The dataset was provided in two parts: images and labels. Therefore, after downloading both parts, the dataset was divided into a train set, using 80% of the available images (a total of 6551), and a test set using the remaining 20% of the images (1638). From the train set, for which the distribution of the number of images over all classes can be seen in Figure 1, the validation test was created using 20% of its data.

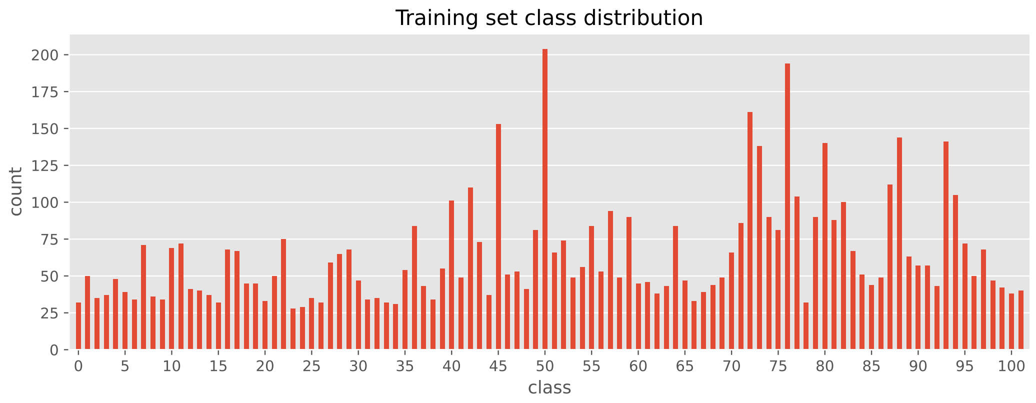

Figure 1: Training set class distribution.

For the preprocessing phase, it was decided to use the preprocessing modes already defined for each of different architectures. In particular:

- For the MobileNetV2 architecture, the preprocessing operations simply rescale the pixels in a range between -1 and 1;
- For the DenseNet architecture, a rescale of the pixels in a range between 0 and 1 and then each channel is normalised with respect to the ImageNet dataset;
- For the EfficientNet architecture, images are converted from RGB to BGR, and then each channel were normalised with respect to the ImageNet dataset.

Then all images were resized to 224 x 224 pixels.
The second operation performed on the data was data augmentation, applying various policies for which an example can be seen in Figure 2. The operations performed on images were:

- Images were randomly shifted in the x-direction and y-direction in a range from +10% to -10% of their total width or height, respectively. The outlines produced by shift operation were colored using black, avoiding changing the shape of flowers;

[2]`https://www.robots.ox.ac.uk/~vgg/data/flowers/102/categories.html`

– Images were randomly flopped;

– The brightness was randomly increased or decreased in range from +5% to -5%. It was decided to use such a small range because excessive variations increased the possibility that two different flowers would become too much similar also because of the low inter-class variation of the dataset, as can be seen in the Figure 3;

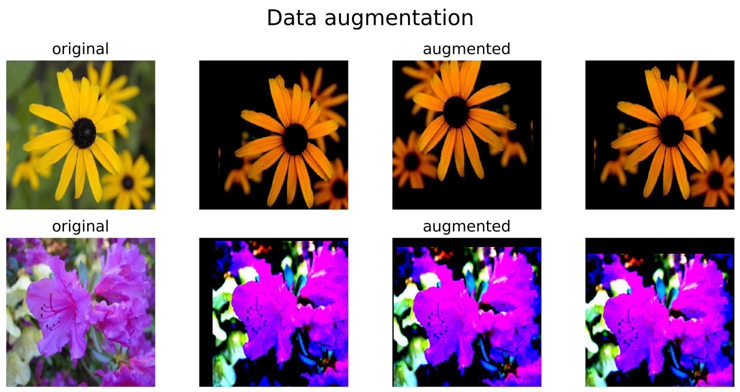

Figure 2: Data augmentation examples. The augmented images in the first row were obtained using the MobileNetV2 preprocessing, while the images in the second row were obtained using DenseNet preprocessing.

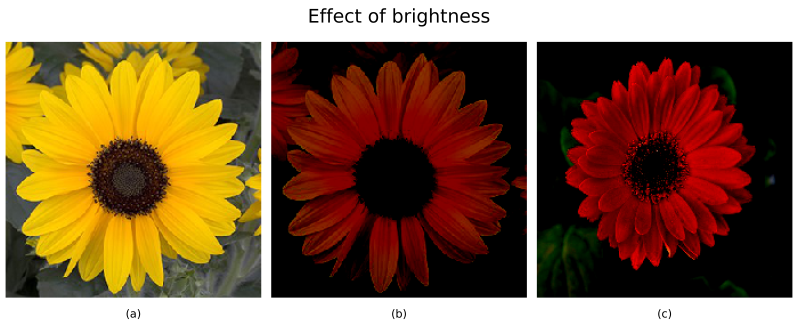

Figure 3: Effect of modifying brightness before MobileNetV2 preprocessing. The preprocessed image (b) of a Sunflower (a) and the preprocessed image (c) of a Barbeton Daisy result very similar after reducing brightness.

As Figure 1 shows, there was an obvious difference in the images belonging to the various classes, so it was an unbalanced dataset situation. Thus, to overcome the problem of getting high accuracy values due to an unbalanced dataset situation, rather than over or under sampling, we opted for a weight-balancing approach in which all classes contribute equally to the loss, assigning a greater weight to the smaller classes and a lower weight to the larger classes.

3 The Methodological Approach

The objective to reach was to train a model with optimal performance and small size that could be embedded on any mobile device. The project was focused on comparison of different architectures where at each steps the best choices were selected to achieve the designated goal.
The dataset was small and in part like the ImageNet dataset, so were exploited pre-trained architectures on this dataset, considered their compact dimensions and good capability to discriminate classes. All the assumptions drove to use fine-tuning techniques. The pre-trained networks were MobileNetV2, DenseNet and EfficientNet:

- MobileNetV2 is a neural network architecture that is specifically tailored for mobile and resource constrained environments[2]. Unlike the other architectures, MobileNet allowed to handle network width through a parameter α that manage the filters number in each layer. For a given layer and width multiplier α, the number of input channels M becomes αM and the number of output channels N becomes αN[3]. Settings used were 1, 0.75, 0.5, thus the first step was to select the best model based on these settings;

- DenseNet have connects each layer to every other layer in a feed-forward fashion. The effect of this dense connectivity pattern is that it requires fewer parameters than traditional convolutional networks, as there is no need to re-learn redundant feature-maps[4]. The model used is DenseNet-121, which is based on this but it's more compact, it compresses each 6 dense blocks in one;

- EfficientNets are a family of models developed applying scaling method that uniformly scales all dimensions of depth/width/resolution of a baseline neural network. It has been proved that they can achieve much better accuracy and efficiency than previous developed ConvNets[5]. The model used is Efficient-B0 which is the baseline architecture and it uses 5.4M parameters;

These architectures were cut after the last global avarage pooling layer. Cut on high layer provided better performance against a lower cut; this was due to the similarity between the used dataset and Imagenet dataset. Finally, these architectures were concatenated with a single fully connected layer of 102 units and softmax as activation function, because it was a classification problem. The initial optimizer was SDG, but it provided bad performance, for this reason it was preferred ADAM (Learning rate = 0.002) optimizer that exploited the momentum strategy to increase the loss descent. Categorical crossentropy was used as loss function while the models were evaluated using top-1, top-3, top-5 accuracy. The next steps were:

1. Comparison between models' performance.

2. Iterative pruning on models.

3. Quantization of models.

During the model training were handled two important issues: the unbalanced dataset and overfitting. The problem of unbalanced dataset was solved using a weight balancing approach, already discussed in Section 2. To avoid overfitting, which could make the model not able to discriminate images not coming from train dataset, different regularization techniques were used. One of them was data augmentation, that has already been cited in the Section 2. The other two were dropout (drop rate = 0.5)[6] applied on fully connected layer and early stopping. The latter was used to monitor the top-1 validation accuracy and to interrupt the training when there was a large offset between it and training accuracy.
Ended the comparison between the models, the compression phase started. Although the architectures on which the models were based were already small, the compression provided models with smaller size avoiding losing accuracy. Initially, an iterative pruning was applied to the models. Therefore, at each iteration the number of models' parameters were reduced, and the models were re-trained and re-evaluated to check how and how many the accuracy changed, finding the best trade-off between model size and accuracy. It was used the Keras pruning magnitude-based[3] technique: rather than remove the smallest weights it set them to zeros. The pruning was applied only on convolutional (normal and pointwise) and fully connected layers, since they contained the majority of parameters, as can be seen in Table 1.

Table 1: Amount of parameters per layer.

Layer	Mobilenet	DenseNet-121	EfficientNet-B0
Convolution (normal and pointwise)	76.07%	96.19%	90.51%
Batch normalization	4.43%	2.34%	2.01%
Fully connected	15.61%	1.64%	3.13%
Depthwise convolution	3.89%	0%	4.43%
Other	0%	0%	0%

Then the models were quantized. This process constrains an input from a continuous or otherwise large set of values (such as the float32) to a smaller set (such as the float16). Quantization was done via TFlite[4] framework. Models were quantized using the dynamic range quantization[5] which converts weights to 8 bit precision (int) and converts activations dynamically at inference. Unfortunately, it was impossible to get a valid estimation of models' inference time without an appropriate CPU (ARM-based CPUs) that could take advantage of this kind of quantization. So, for this reason, also float16 quantization was tested, since this kind of quantization could take advantage of GPU acceleration.

[3]`https://www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras`

[4]`https://www.tensorflow.org/lite`

[5]`https://www.tensorflow.org/lite/performance/post_training_quant`

4 Results and Evaluation

4.1 Peformance of MobileNets

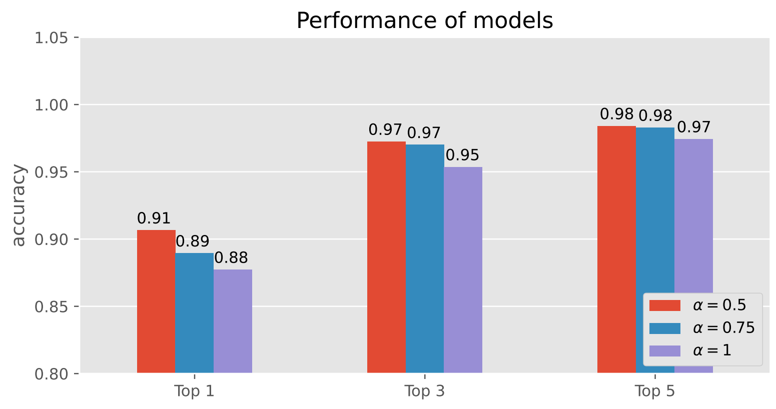

Figure 4: Top-1, top-3, top-5 accuracy obtained by MobileNetV2 using different values of width parameter α $(0.50, 0.75, 1)$.

4.2 Performance of models

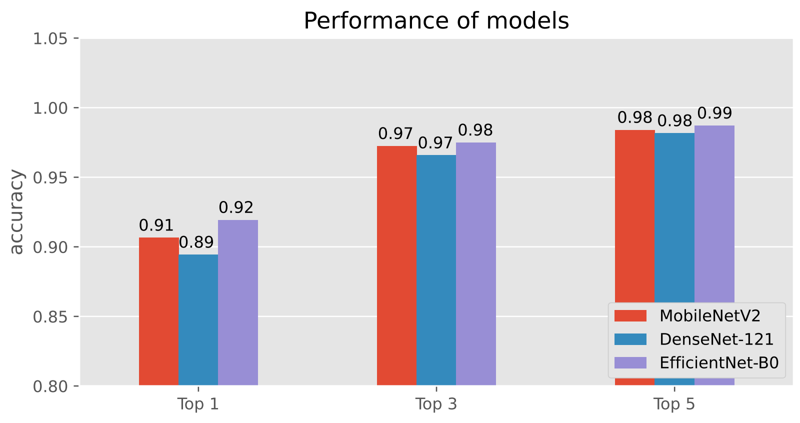

Figure 5: Top-1, top-3, top-5 accuracy obtained by MobileNetV2, DenseNet and EfficientNet.

4.3 MobileNetV2 iterative pruning

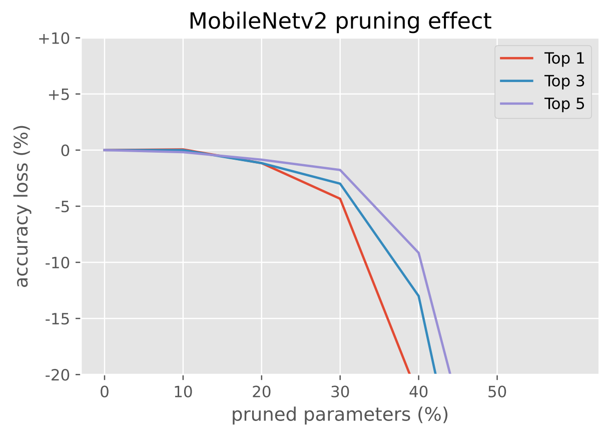

Figure 6: Accuracy-sparsity curve obtained by iteratively pruning MobileNetV2 (with $\alpha = 0.5$).

4.4 DenseNet iterative pruning

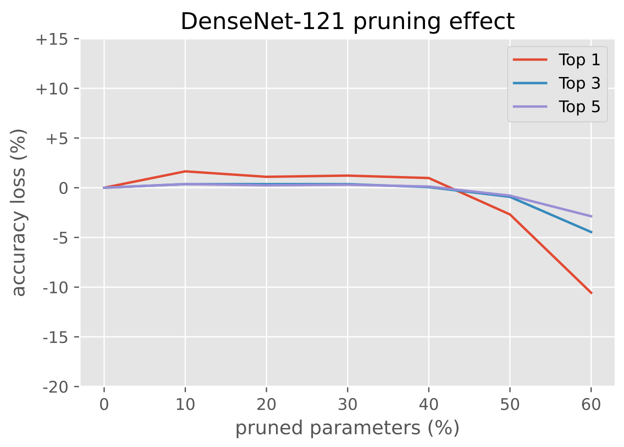

Figure 7: Accuracy-sparsity curve obtained by iteratively pruning DenseNet.

4.5 EfficientNet pruning

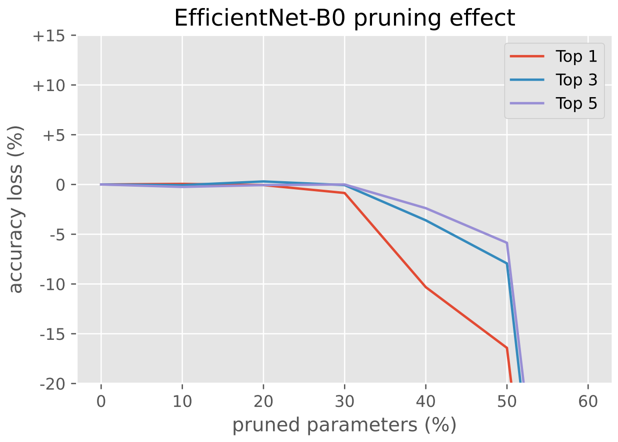

Figure 8: Accuracy-sparsity curve obtained by iteratively pruning EfficientNet.

4.6 Quantization

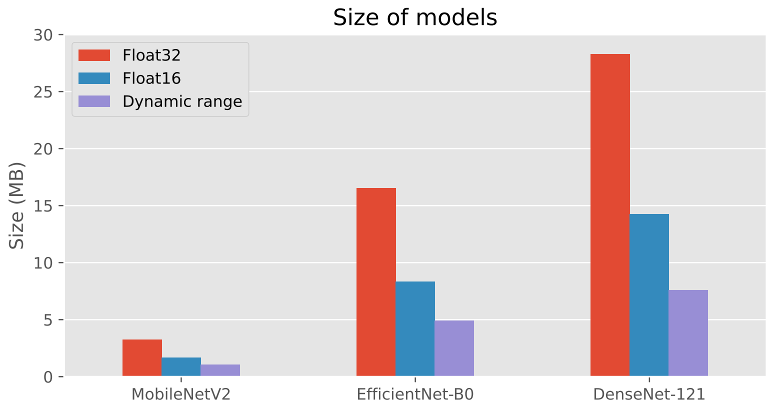

Figure 9: Size of models before quantization (float32), after half-precision (float16) quantization and dynamic range quantization.

Table 2: Effects of quantization on MobileNetV2.

	Float32	Float16	Dynamic range
Top-1 Accuracy	89.50%	89.50%	89.26%
Top-3 Accuracy	96.09%	96.15%	95.91%
Top-5 Accuracy	97.56%	97.56%	97.44%
Model size (MB)	3.27	1.67	1.07
Avg inference time (s)	0.01	0.01	0.26

Table 3: Effects of quantization on EfficientNet.

	Float32	Float16	Dynamic range
Top-1 Accuracy	91.09%	91.15%	91.33%
Top-3 Accuracy	97.44%	97.44%	97.25%
Top-5 Accuracy	98.72%	98.72%	98.53%
Model size (MB)	16.56	8.36	4.92
Avg inference time (s)	0.08	0.08	1.16

Table 4: Effects of quantization on DenseNet.

	Float32	Float16	Dynamic range
Top-1 Accuracy	90.42%	90.35%	- -
Top-3 Accuracy	96.64%	96.64%	- -
Top-5 Accuracy	98.29%	98.29%	- -
Model size (MB)	28.31	14.25	7.60
Avg inference time (s)	0.11	0.11	~ 7.35

5 Discussion

The first comparison was the one between the three MobileNetV2 models defined using different values for the α parameter. Figure 4 shows how well they performed on test set. Since the model with $\alpha = 0.5$ achieved the best performance and was the smallest in terms of size, it was chosen as candidate for the subsequent comparisons. Before proceeding with the pruning process, the DenseNet and EfficientNet models were also evaluated on the same test and the overall results are shown in Figure 5. To obtain a fair comparison, a k-fold cross validation should have been performed, but its execution would have exceeded the limits imposed by Google on Colab.
Figures 6, 7, 8 show the accuracy-sparsity curves obtained after the iterative pruning of MobileNetV2, DenseNet and EfficientNet, respectively. In MobileNetV2 were pruned the 20% of parameters, keeping an accuracy of 89.82%. DenseNet proved to be more resilient during the pruning process, thus it was chosen to reach a sparsity of 40%. The pruned model also achieved a better accuracy (90.42%) compared to the "full" model (89.44%). The accuracy of EfficientNet remained stable until it reached a sparsity of 30%, hence the model was only pruned up to this point, mantaining an accuracy of

91.09%.
Finally, all three models were quantized. Figure 9 shows the models' size without any kind of quantization (in red), using a float16 quantization (in blue) and using a dynamic range quantization (in purple). Tables 2, 3, 4 report more details about the quantization effect on size and performance of MobileNetV2, EfficientNet and DenseNet, respectively. The use of the dynamic range quantization did not affected the performance of models but it allowed a significant reduction of their dimension. As expected, the smallest model was MobileNetV2 (1.07 MB), but also EfficientNet (4.92 MB) and DenseNet (7.60 MB) reached a suitable size for a mobile device. As said before in Section 2, without an appropriate CPU that could take advantage of the dynamic range quantization, the models' inference time increased. So, it could not be possible to evaluate the accuracy of quantized DenseNet without exceeding the restrictions provided by Google Colab on computation (it took about 7.35 seconds to make prediction on a single image). Anyway, the float16 models showed a maximum inference time of one tenth of second, exploting the GPU acceleration.
In conclusion, the model chosen for the application was the EfficientNet, since its performance is higher compared to the one of MobileNetV2 (91.33% > 89.26%) and its size remains acceptable for a mobile device (4.92 MB).

6 Conclusions

The experiments ended up with a model based on EfficientNet which is suitable for the purpose of the project, given its performance (91.33%) and its size (4.92 MB). In case of future problems related to the size of the application, the final choice will fall on MobileNetV2. Future studies could investigate how different policies of data augmentation impact model perfomance, paying special attention to the problem of low inter-class variations between flower species.

References

[1] M.-E. Nilsback, “An automatic visual Flora – segmentation and classification of flowers images,” Ph.D. dissertation, University of Oxford, 2009. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/publications/2009/Nilsback09/

[2] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[3] 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,” *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[4] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[5] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *CoRR*, vol. abs/1905.11946, 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1, no. 2, ch. 7.12, p. 253.