A Foray into CNN Efficiency

Yuning Wu
yanzu.wu@mail.utoronto.ca

Amir Azimi
amir.azimi@mail.utoronto.ca

Bojian Li
lafcadio.li@mail.utoronto.ca

Abstract

In this paper we reproduce the results of EfficientNet and MobileNet and compare them against the results stated in their respective papers. We train the two models on the Deep Weeds dataset to evaluate their transfer learning potential on a new classification task, and discuss the results and advantages to each model.

1 Introduction

EfficientNet is a model that is widely known for its efficiency, as it is able to obtain state of the art accuracy yet with a much lower number of model parameters. The goal of EfficientNet is to use a scaling method that provides an optimal scaling of network dimensions that will improve accuracy. Another model that presents state of the art mobile accuracy optimized to run resource-constrained environments is MobileNet. Our goal is to reproduce some of the results of these models on databases such as ImageNet and CIFAR-100, and also to extend these models to a new classification task that they had not been overexposed to during their training on ImageNet. We aim to compare the performance and efficiency of both of these models and to evaluate their respective advantages.

2 Related works

Scaling a ConvNet to improve accuracy is a widely used technique. For example, the famous ResNet(He et al., 2016) came out in 2016 along with some size-scaled variants of the model. It is clear that similar models with different sizes exhibit different performance and characteristics. EfficientNet(Tan et al., 2020) was built for seeking an optimal and uniform way to scale of ConvNet. Later on this year, the improved version EfficientNetV2(Tan et al., 2021) was introduced; the model is faster and also smaller.

In addtion to scaling, optimizing models to achieve better accuracy and more efficient training is a constantly evolving research research topic with the introduction of new architecture and tools. There have been multiple papers that have improved upon the MobileNet architecture. MobileNetV2 itself is an improvement of MobileNetV1 (G.Howard, et al., 2017), which first proposed depth-wise separable convolutions. The separable convolutions separates depth-wise convolutions from point-wise convolutions(Francois, et al., 2016) significantly reduced the number of parameters by obtaining the feature maps from point-wise convolutions and filter maps from depth-wise convolutions without loss of information thus the model can have more layers for the same cost. More improvements have been made in the form of MnasNet (Tan, et al., 2019), which aimed to improve model latency while simultaneously increasing accuracy by introducing an architecture search and squeeze and excitation layers. Further improvements have been made in the form of MobileNet V3 (Howard, et al., 2019) which introduced swish layers, obtaining better accuracy and lower latency than MobileNetV2. Even further improvement has been introduced with GhostNet (Han, et al., 2020), which uses new "ghost modules" to generate feature maps more efficiently.

3 Algorithm / methods / architecture

3.1 EfficientNet

We need to consider the additional amount of resources needed as a consequence of scaling up networks. In fact, we are trying to solve an optimization problem:

$$\max Accuracy$$
$$\text{subject to: } \begin{cases} Memory \leq \text{ target memory} \\ FLOPS \leq \text{ target flops} \end{cases}$$

It is determined empirically that there are relationships between depth(d), width(w) and resolution (r) of ConvNet. It is also empirically verified that scale up only one dimension results in sub-optimal results. Thus, it seems reasonable to assume that the three dimensions satisfy some constraints like below:

$$d = \alpha^\phi, w = \beta^\phi, r = \gamma^\phi$$
$$\text{subject to } \begin{cases} \alpha \times \beta^2 \times \gamma^2 \approx 2 \\ \alpha, \beta, \gamma \geq 1 \end{cases}$$

Above set of constraints is named compound scaling, and it describes the architecture of EfficientNet. After conducting a grid search, it is determined that the optimal values are $\alpha = 1.2$, $\beta = 1.1$ and $\gamma = 1.1$. By setting $\phi = 1$, a base network EfficientNetB0 is constructed using some combination of standard convolutional layers and inverted residual blocks. The scaling factor ϕ should be treated as a hyper-parameter; it should be manual adjusted according to the target resources or type of tasks.

3.2 MobileNetV2

In order to evaluate the efficiency of EfficientNet, we decided to test it against another model with a similar number of parameters that is also well-known for its "efficiency": MobileNet V2.

MobileNet is a neural network that is designed for mobile applications or scenarios where computational resources are limited. To achieve greater efficiency and state of the art performance, MobileNet makes use of an inverted residual with a linear bottleneck layer (Sandler, et al., 2019).

The inverted residual with linear bottleneck consists of a point-wise (kernel size of 1x1) convolution that expands the input by an certain expansion factor t and a ReLu activation (Gonzales L., 2019). The model then uses depth-wise separable convolutions (with a kernel size of 3x3) to achieve a higher dimensional representation of the input which comes before another ReLu activation. The model then uses more point-wise convolutions to project the previous activations back into a lower dimension (Gonzales L., 2019).

The significance of a "linear" layer is that the projection from high to low dimension results in a loss of data which acts as a sort of activation itself, thus another activation function is not required here (Gonzales L., 2019). Finally, a skip connection (i.e the residual) is added between the two bottleneck layers that will help with propagation.

The idea behind this change is to replace standard convolution operations with a more efficient factorized version that would require less computational resources (Sandler, et al., 2019). As described in the MobileNetV2 paper, applying a standard convolution on an input tensor L_i of dimension $h_i \times w_i \times d_i$ with kernel K of dimensions $k \times k \times d_i \times d_j$ results in a tensor with dimensions $h_i \times w_i \times dj$ has a computational cost of $hi \cdot w_i \cdot d_i \cdot d_j \cdot k^2$. The proposed inverted residual layer results in an output tensor with the same dimensions as before yet with a computation cost of $h_i \cdot w_i \cdot d_i(k^2 + d_j)$. Since MobileNet uses a kernel of size 3×3, we have that the computation cost for a standard convolution is $h_i \cdot w_i \cdot d_i \cdot d_j \cdot 9$, whereas the cost for this new layer is $h_i \cdot w_i \cdot d_i \cdot (9 + d_j)$, which as the paper states, is around 8-9 times less.

4 Experiments and Results

4.1 EfficientNet

It has been claimed that EfficientNet is able to achieve state-of-the-art performance on some well known dataset while the network has significant less number of parameters (Tan et al., 2020). In this section, we would like to conduct experiments on CIFAR100, CIFAR10 and deep_weeds dataset (Olsen, et al., 2019).

We used the same learning algorithm for all the datasets. We used stochastic gradient descent with weight decay set to 0.0005 and a scheduled learning rate decay which the learning rate will be multiply by 0.9 every five epochs.

For CIFAR100 and CIFAR10 dataset, we fine-tune the entire network which trained on imagenet. We stopped training at 500 epochs, and the best top-1 accuracy can be found in Table 1. Our experiement result on CIFAR100 dataset was closed to the claimed result which was 87.5%. However, our results on CIFAR10 were far from the claimed results in the original paper which achieved 98% with EfficientNetb0. This could be caused by the usage of different gradient descent algorithm since the original paper did not mention what algorithm was used on CIFAR10.

For deep_weeds dataset, we trained the models from scratch for 500 epochs and recorded the best top-1 validation accuracy. For the first 70 epochs, both train accuracy and validation accuracy increased rapidly; for EfficientNetb0, the top-1 validation accuracy is 90.5% at 70 epochs. The only hyper-parameter in EfficientNet is the compound scaling factor ϕ, the baseline model EfficientNetb0 has $\phi = 1$, the b1 and b2 model has correspondingly larger ϕ values. We observe that with larger compound scale factor, the model is able to achieve better accuracy.

For Rock-Paper-Scissors dataset we fine-tuned the entire model of both EfficientNetB0 and EfficientNetB2 for the input sizes of 150x150 on resized and 300x300 on raw data respectively. We obtained top-1 validation accuracy at 250 epochs of 86% and 92.4% with training time for each epoch on EfficientNetB2 being 50% slower. The results from this dataset also corresponds to the conclusion from the previous experiments

Table 1: EfficientNet results

Dataset	Model	Accuracy
Cifar10	EfficientNetb0	91.2%
Cifar10	EfficientNetb1	91.3%
Cifar10	EfficientNetb2	91.7%
Cifar100	EfficientNetb0	86.9%
deep_weeds	EfficientNetb0	93.5%
deep_weeds	EfficientNetb1	94.7%
deep_weeds	EfficientNetb2	95.0%

4.2 MobileNetV2

In order to benchmark the MobileNet model we attempted to reproduce the results used in the paper using a pretrained MobileNetV2 model imported from Tensorflow (Abadi, et al., 2015). We used MobileNetV2 to classify 50k validation images from the ILSVRC2012 dataset (Russakovsky, et al. 2015). We were able to achieve an average accuracy of around 71%, which is very close to the 72% reported in the MobileNetV2 paper.

We wanted to analyze the performance of the MobileNetV2 model on image classes that it had not been exposed to frequently during its training on ImageNet so that we may see how it would transfer its learned "knowledge" to an image classification task that is new to it.

We chose to analyse its performance on the Deep Weeds data-set (Olsen, et al., 2019) and compare its performance to EfficientNet. To train the model, we replaced the final layer of the MobileNetV2 model with a fully connected layer and trained it on the Deep Weeds dataset, freezing all weights except the ones of the newly connected layer. Like in the MobileNet paper we used the RMSProp

optimizer, but with a learning rate of 0.001. After 10 epochs, our model achieved an accuracy of 0.7995, as seen in Figure 1.

To fine tune the model, we unfroze the last 34 layers and retrained the model. This had no noticeable effect on the accuracy of the model on the validation data, although it seemed the model was slowly over-fitting to the training data as its training accuracy was increasing while its validation accuracy remained fixed. We repeated the above procedure using the SGD optimizer and a learning rate of 0.01 and received very similar results.

Since fine-tuning the model by unfreezing the weights did not improve performance, it would suggest that the MobileNetV2 model that was pre-trained on the ImageNet data-set is already able to capture many features of the weeds and does not require much fine-tuning.

To replicate the EfficientNet training, we attempted to train MobileNet in the same way. After adding the classification head, instead of training it separately we unfroze all weights and re-trained the model with a learning rate of 0.0001 using an RMSProp optimizer. After 20 epoch's our model was able to achieve an accuracy of around 0.89, which is considerably better than our previous attempt at training the model and much closer to EfficientNet in terms of accuracy. However, this model took considerably longer to train than the previous MobileNet model with frozen weights.

4.3 Comparison

Comparing the results of EfficientNet and MobileNet, it is clear that EfficientNet was able to achieve a much accuracy on the Deep Weeds data-set.Surprisingly, the MobileNet with unfrozen weights achieved a very high training accuracy early on during training, suggesting that the model would overfit to the training data. However, validation accuracy increased considerably as training went on, and the MobileNet model was able to achieve a much better validation accuracy than the previous MobileNet model with frozen weights. The MobileNet model was also able to achieve an accuracy very close to that achieved by the EfficientNetB0 model, which is closest to MobileNet in terms of the number of parameters it contains

However, it should be noted that training both MobileNet models took a much shorter amount of time than training the EfficientNet variants, clearly demonstrating their efficiency. Thus, while EfficientNet may be more efficient than other state-of-the-art-image classification models, and while the number of model parameters for both models are not drastically different compared to each other, EfficientNet is expectedly not nearly as optimized as MobileNet in terms of how much computational power it requires to train and deploy.

5 Summary

We were able to re-implement and train both EfficientNet and MobileNet and evaluate their performance against the benchmarks stated in their respective papers. We also conducted a number of experiments employing transfer learning to evaluate the performance of both pre-trained models and compare their advantages.

While EfficientNet took longer to train, all variants of the EfficientNet model were able to achieve a better accuracy than any MobileNetV2 model on the Deep Weeds dataset. It was noted that training EfficientNet took considerably more time to train. However, MobileNet was able to achieve an adequate validation accuracy while using considerably less time and resources. Thus it can be said that despite their relatively similar number of parameters, we can conclude that MobileNet is definitely a more feasible model to employ in mobile environments with limited resources.

6 Contributions

Amir Azimi contributed to sections 2, 3.2, 4.2, and 4.3.

Yuning Wu contributed to section 2, 3.1, 4.1 and 4.3

Bojian Li contributed to section 2, 4.1

References

[1] Abadi, M., Agarwal, A., Barham, P.,, Brevdo, E., Chen, Z., Citro, Z., S. Corrado, G., Davis,A., Dean,J., Devin, M., Ghemawat,S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, l., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, Pete., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015) TensorFlow: Large-scale machine learning on heterogeneous system. Tensorflow.org

[2] Culfaz, F. (2018) Transfer Learning using Mobilenet and Keras.

[3] Francois, C. (2016) Xception: Deep Learning with Depthwise Separable Convolutions. arXiv:1610.02357 [cs.CV]

[4] Gonzales, L. (2019) A Look at MobileNetV2: Inverted Residuals and Linear Bottlenecks.

[5] G. Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 [cs.CV]

[6] Han, K., Wang, Y., Tian, Q., Guo,J., Xu, C., & Xu, C. (2020) GhostNet: More Features from Cheap Operations. arXiv:1911.11907 [cs.CV].

[7] Howard, A., Sandler, M., Chu, G., Chieh Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R Vasudevan, V., V. Le, Q.,& Adam, H. (2019) Searching for MobileNetV3. arXiv:1905.02244 [cs.CV]

[8] Keshav, V., (2019) ImageNet from tensorflow_datasets.

[9] Leifuer (2019) Flowers with Transfer Learning (MobileNet-Keras).

[10] Olsen, A., A. Konovalov, D., Philippa, B., Ridd, P., C. Wood, J.,Johns, J., Banks, W.,Girgenti, B., Kenny, O., Whinney, J., Calvert, B., Azghadi, M.R., & D. White, R. (2019) DeepWeeds: A Multiclass Weed Species Image Dataset for Deep Learning. doi: 10.1038/s41598-018-38343-3

[11] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh,S., Ma, S., Huang, Z.,and Karpathy,A., Khosla,A., Bernstein, M., C. Berg, A., & Fei-Fei, L. (2015) ImageNet Large Scale Visual Recognition Challenge. doi: 10.1007/s11263-015-0816-y.

[12] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2019) MobileNetV2: Inverted Residuals and Linear Bottlenecks.arXiv:1801.04381 [cs.CV]

[13] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & V. Le, Q (2019) MnasNet: Platform-Aware Neural Architecture Search for Mobile. arXiv:1807.11626 [cs.CV]

[14] Tan, M., V.Le, Q. (2020) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1812.00332 [cs.LG]

[15] Tan, M., V.Le, Q. (2020) EfficientNetV2: Smaller Models and Faster Training. arXiv:2104.00298 [cs.CV]

[16] Yixing, F. (2020) Image classification via fine-tuning with EfficientNet.

7 Appendix

7.1 MobileNet Code

Note that the provided MobileNet notebooks make use of code from Yixing, F. (2020), Leifuer (2019), and Culfaz, F. (2018), and Keshav, V. The code was mainly used to understand how to operate with tensorflow. As such, our code for the re-implementation of MobileNet makes use of a different model as that of Yixing, F. (2019), in addition to original code used to evaluate the MobileNetV2 model.

Our code for the extension of MobileNet to the Deep Weeds dataset makes use of some code from Leifuer (2019), and Culfaz, F. (2018), while using a different classification head and a different training procedure.

7.2 MobileNet Training Results

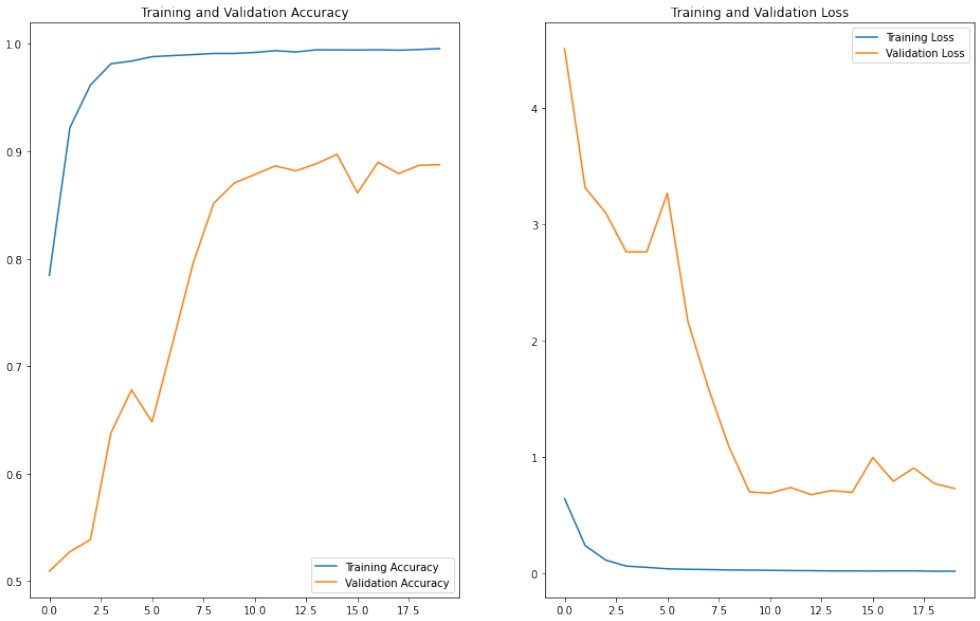

Figure 1: MobileNet accuracy and loss on Deep Weeds data-set with no frozen weights