Efficient mobilenet architecture as image recognition on mobile and embedded devices

Barlian Khasoggi, Ermatita, Samsuryadi
Master of Informatics Engineering, Sriwijaya University, Indonesia

Article Info

Article history:
Received Jan 7, 2019
Revised Apr 10, 2019
Accepted May 15, 2019

Keywords:

CNNs
Deep learning
Image recognition
MobileNet
Tensorflow

ABSTRACT

The introduction of a modern image recognition that has millions of parameters and requires a lot of training data as well as high computing power that is hungry for energy consumption so it becomes inefficient in everyday use. Machine Learning has changed the computing paradigm, from complex calculations that require high computational power to environmentally friendly technologies that can efficiently meet daily needs. To get the best training model, many studies use large numbers of datasets. However, the complexity of large datasets requires large devices and requires high computing power. Therefore large computational resources do not have high flexibility towards the tendency of human interaction which prioritizes the efficiency and effectiveness of computer vision. This study uses the Convolutional Neural Networks (CNN) method with MobileNet architecture for image recognition on mobile devices and embedded devices with limited resources with ARM-based CPUs and works with a moderate amount of training data (thousands of labeled images). As a result, the MobileNet v1 architecture on the ms8pro device can classify the caltech101 dataset with an accuracy rate 92.4% and 2.1 Watt power draw. With the level of accuracy and efficiency of the resources used, it is expected that MobileNet's architecture can change the machine learning paradigm so that it has a high degree of flexibility towards the tendency of human interaction that prioritizes the efficiency and effectiveness of computer vision.

Corresponding Author:

Ermatita,
Master of Informatics Engineering,
Sriwijaya University,
30862 South Sumatera, Indonesia.
Email: e-mail: ermatitaz@yahoo.com

1. INTRODUCTION

Convolutional neural networks (CNNs) are current state-of-the-art architectures and widely used as image classification solutions [1]. CNNs make an important contribution to the great advances in computer vision, especially in applications and everyday utilities such as self-driving car, robotics, drones, medical diagnostics, and treatment for vision impairment. These successes spurred a new line of research that focused on finding higher performing convolutional neural networks. Starting in 2014, the quality of network architectures significantly improved by utilizing deeper and wider networks [2]. To improve network performance, many studies use larger datasets, better learning models, and better techniques to prevent overfitting [3]. In a study that has been done with relatively small image datasets with labels consisting of tens of thousands to millions of images such as DeCAF [3], NORB [4], Caltech-101/256 [5, 6], and CIFAR-10/100 [7] can accomplish simple tasks with good image recognition results, especially when coupled with labelpreserving transformations. One of the most excellent research results with the lowest current error rate is on the task of introduction of MNIST digits (<0.3%) closer to human performance [8].

In the model training of Convolutional neural networks (CNNs) [4, 6, 9, 10, 11, 12, 13], a model with a large learning capacity is required. However, the complexity of large datasets requires large devices and requires high computing power. Therefore, large computing resources do not have a high flexibility to the tendency of human interaction which prioritizes the efficiency and effectiveness of computer vision. Convolutional Neural Networks (CNNs) training with MobileNet architecture for image recognition on mobile devices and embedded devices on limited resources with ARM-based CPUs and work with moderate amount of training data (thousands of labeled images). This research get results with an accuracy of 86.3% by doing 30 minutes training with mobilenet architecture. With limited resources, these results are closely linked to numbers that are not too far away when compared to the inception v3 architecture (with same dataset).

2. CONVOLUTIONAL NEURAL NETWORKS ARCHITECTURES

Convolutional Neural Networks have been some of the most influential innovations in the field of computer vision. In 2012, year that CNNs grew to prominence as Alex Krizhevsky [1] used them to win that year's ImageNet competition, dropping the classification error record from 26% to 15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

CNNs are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function from the raw image pixels on one end to class scores at the other [14]. And they still have a loss function on the last (fully-connected) layer and all the guidelines developed for learning regular Neural Networks still apply. CNNs architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

CNNs take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a CNNs have neurons arranged in 3 dimensions: width, height, depth. That the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network, here is a visualization:

As described Figure 1, a simple CNNs is a sequence of layers, and every layer of a CNNs transforms one volume of activations to another through a differentiable function. This research use three main types of layers to build CNNs architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). Later, this research stack these layers to form a full CNNs architecture.

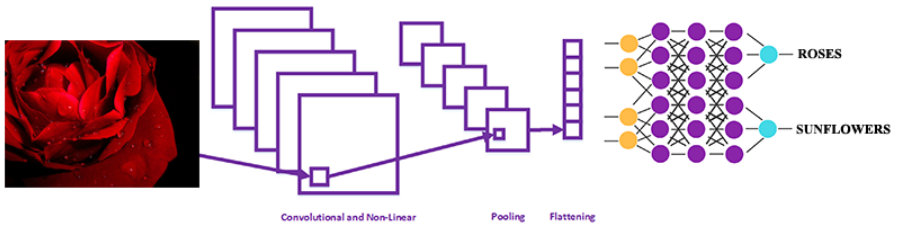


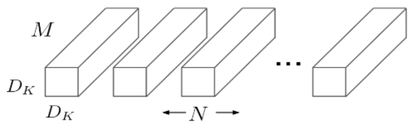
Figure 1. The regular convolutional neural networks

3. MOBILENET ARCHITECTURE

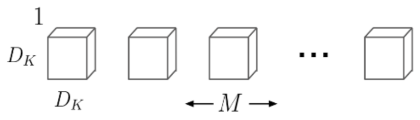
MobileNets is a model constructed mainly from depthwise separable convolutions originally announced in [15] and afterward used in Inception models [16] to decrease the volume of computation constraints in the first few layers. Flattened networks [17] construct a network out of fully factorized convolutions and presented the potential of exceptionally factorized networks. Factorized Networks [18] presents a similar factorized convolution in addition to use of topological networks. Successively, the Xception network [19] established how to scale up depthwise separable filters to perform better than Inception V3 networks. Another small network is Squeezenet [20] which uses a bottleneck method to design a very small network. Other decreased computation networks include arranged transform networks [21] and deep fried convnets [22]. This section will describe the principal layers that MobileNet is built on which are depthwise separable filters.

3.1. Depthwise separable convolution

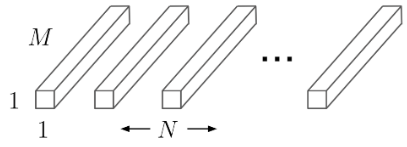
The MobileNet model is based on depthwise separable convolutions which is a procedure of factorized convolutions which factorize a regular convolution into a depthwise convolution and a 1 × 1 convolution named a pointwise convolution [26]. MobileNets depthwise convolution uses a single filter to every input channel. The pointwise convolution then applies a 1 × 1 convolution to merge the outputs the depthwise convolution. A regular convolution both filters and merge inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for merging. This factorization has the effect of extremely reducing computation and model size. Figure 2 shows how a regular convolution 2(a) is factorized into a depthwise convolution 2(b) and a 1 × 1 pointwise convolution 2(c).



(a) Regular Convolutional Filters



(b) Depthwise Filters



(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The regular convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

A regular convolutional layer takes as input a DF × DF × M featuremap F and produces a DF × DF × N feature map G where DF is the spatial width and height of a square input feature map1, M is the number of input channels (input depth), DG is the spatial width and height of a square output feature map and N is the number of output channel (output depth). The regular convolutional layer is parameterized by convolution kernel K of size DK ×DK ×M ×N where DK is the spatial dimension of the kernel assumed to be square and M is number of input channels and N is the number of output channels as defined previously.

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot \; F_{k+i-1,l+j-1,m} \tag{1}$$

Regular convolutions have the computational cost of:

$$D_K \,.\, D_K \,.\, M \,.\, N \,.\, D_F \,.\, D_F \tag{2}$$

where the computational cost depends multiplicatively on the number of input channels M, the number of output channels N the kernel size Dk × Dk and the feature map size DF × DF. MobileNet models report each of these terms and their relations. First it uses depthwise separable convolutions to break the relations between the number of output channels and the size of the kernel. The regular convolution operation has the result of filtering features constructed on the convolutional kernels and merging features in order to produce a new depiction. The filtering and combination steps can be split into two steps via the use of factorized convolutions called depthwise separable convolutions for substantial decrease in computational cost. Depthwise separable convolution are made up of two layers: depthwise convolutions and pointwise convolutions. Then depthwise convolutions to apply a single filter on each input channel (input depth). Pointwise convolution, a simple 1×1 convolution, is then used to create a linear combination of the output. This model use both batchnorm and ReLU nonlinearities for both layers.

Depthwise convolution with one filter per input channel (input depth) can be written as:

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{H}_{i,j,m} \cdot \; F_{k+i-1,l+j-1,m} \tag{3}$$

where Ĥ is the depthwise convolutional kernel of size DK × DK × M wherethe m filterin Ĥ is applied to the m channel in F to produce the mth channel of the filtered output feature map Ĝ.
Depthwise convolution has a computational cost of:

$$D_K \,.\, D_K \,.\, M \,.\, D_F \,.\, D_F \tag{4}$$

Depthwise convolution is extremely efficient relative to regular convolution. However it only filters input channels, it does not combine them to create new features. So an additional layer that computes a linear combination of the output of depthwise convolution via 1 × 1 convolution is needed in order to generate these new features. The combination of depthwise convolution and 1 × 1 (pointwise) convolution is called depthwise separable convolution which was formerly introduced in [15].
Depthwise separable convolutions cost:

$$D_K \,.\, D_K \,.\, M \,.\, D_F \,.\, D_F \;+\; M \,.\, N \,.\, D_F \,.\, D_F \tag{5}$$

which is the sum of the depthwise and 1 × 1 pointwise convolutions. By using convolution as a two step process of filtering and combining to get a reduction in computation of:

$$\frac{DK\,.DK\,.M\,.DF\,.DF \;+\; M\,.N\,.DF\,.DF}{DK\,.DK.M\,.N\,.DF\,.DF} = \frac{1}{N} + \frac{1}{D_K^2} \tag{6}$$

MobileNet uses 3 × 3 depthwise separable convolutions which increase efficiency than regular convolutions at only a small reduction in accuracy. Additional factorization such as in [17, 2] does not show increase of computation efficiency as very little computation is spent in depthwise convolutions.

3.2. Network structure and training

The MobileNet structure is built on depthwise separable convolutions as mentioned in the previous section except for the first layer which is a full convolution. By defining the network in such simple terms are able to easily explore network structure to find a good network. All layers in MobileNet are followed by a batchnorm [16] and ReLU nonlinearity with the exception of the final fully connected layer which has no nonlinearity and feeds into a softmax layer for classification. Figure 3 contrasts a layer with regular convolutions, batchnorm and ReLU nonlinearity to the factorized layer with depthwise convolution, 1 × 1 pointwise convolution as well as batchnorm and ReLU after each convolutional layer. Down sampling is handled with strided convolution in the depthwise convolutions as well as in the first layer. A final average pooling reduces the spatial resolution to 1 before the fully connected layer. Counting depthwise and pointwise convolutions as separate layers, MobileNet has 28 layers.

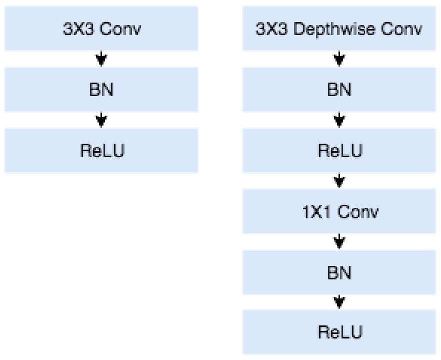


Figure 3. Left: Regular convolutional layer with batchnorm and ReLU. right: depthwise separable convolutions with depthwise and pointwise layers followed by batchnorm and ReLU

Instance formless light matrix operations are not typically faster than thick matrix operations until a very high level of sparsity. This model structure puts nearly all of the computation into dense 1 × 1 convolutions. This can be implemented with highly optimized general matrix multiply (GEMM) functions. Often convolutions are implemented by a GEMM but require an initial reordering in memory in order to map it to a GEMM. This method is used in the popular Caffe package [23]. 1 × 1 convolutions do not require this reordering in memory and can be implemented directly with GEMM which is one of the most improved numerical linear algebra algorithms.

MobileNet models were trained in TensorFlow [24] using RMSprop [25] with asynchronous gradient descent similar to Inception V3 [2]. However, opposing to training large models use less regularization and data augmentation techniques that small models rarely have trouble with overfitting. When training MobileNets, use side heads or label smoothing and additionally reduce the amount image of distortions by limiting the size of small crops that are used in large Inception training [2]. Additionally, found that it was important to put very little or no weight decay on the depthwise filters since their are so few parameters in them.

4. MOBILENET ARCHITECTURE RESULTS AND ANALYSIS

In this section, first investigate the effects of depthwise convolutions as well as the choice of shrinking by reducing the width of the network rather than the number of layers. Then show the trade offs of reducing the network based on the two hyper parameters: width multiplier and resolution multiplier and compare results to a number of popular models. And then investigate MobileNets applied to a number of different devices. Result Performance of CNNs as shown in Table 1.

Table 1. Result performance of CNNs

Device	Processor	RAM(GB)	Architecture	Power(W)	Execution Time(s)	Final accuracy
Macbook(CPU)	Intel i5	8	Mobilenet V1 0.5 224	49.5	395.29	90.1%
Macbook(CPU)	Intel i5	8	Mobilenet V1 0.75 224	49.5	390.00	92.1%
Macbook(CPU)	Intel i5	8	Mobilenet V1 1.0 224	49.5	400.98	93.3%
PC(GPU)	GTX 1070	8	Mobilenet V1 0.5 224	192.7	396.44	91.8%
PC(GPU)	GTX 1070	8	Mobilenet V1 0.75 224	192.7	396.57	91.1%
PC(GPU)	GTX 1070	8	Mobilenet V1 1.0 224	192.7	386.83	92.3%
M8S PRO	Amlogic S905X	2	Mobilenet V1 0.5 224	2.1	3099.80	88.3%
M8S PRO	Amlogic S905X	2	Mobilenet V1 0.75 224	2.1	3103.25	91.0%
M8S PRO	Amlogic S905X	2	Mobilenet V1 1.0 224	2.1	3123.19	92.4%
RaspberryPI3	Cortex-A53	1	Mobilenet V1 0.5 224	3.5	4636.49	89.2%
RaspberryPI3	Cortex-A53	1	Mobilenet V1 0.75 224	3.5	5611.14	91.5%
RaspberryPI3	Cortex-A53	1	Mobilenet V1 1.0 224	3.5	5208.18	92.6%

5. CONCLUSION

This paper proposed model architecture that can run on eco-friendly devices with MobileNets based on depthwise separable convolutions. Result from investigated some of the important design decisions leading to an efficient model. Then demonstrated how to build smaller and faster MobileNets using width multiplier and resolution multiplier by trading off a reasonable amount of accuracy to reduce size and latency. This research compared different mobile devices and different model architecture to demonstrating how efficiency tradeoff with accuracy. Finally, this paper concluded that this can be solution for training data with moderate dataset in real world with eco-friendly mobile and embedded device.

REFERENCES

[1] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. *In NIPS*, 2012.

[2] C.Szegedy, V.Vanhoucke, S.Ioffe, J.Shlens, and Z.Wojna, Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567, 2015.

[3] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, Arxiv preprint arXiv: 1310.1531v1, 2013.

[4] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. *Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–97. IEEE, 2004.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[6] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, *California Institute of Technology*, 2007.

[7] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[8] D. Cires ̧an, U. Meier, and J. Schmidhuber. Multicolumn deep neural networks for image classification. *Arxiv preprint arXiv*:1202.2745, 2012.

[9] A. Krizhevsky. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 2010.

[10] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009

[11] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. Hand written digit recognition with a backpropagation network. In *Advances in neural information processing systems*, 1990.

[12] N. Pinto, D. Doukhan, J.J. DiCarlo, and D.D. Cox. A highthroughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS computational biology*, 5(11):e1000579, 2009.

[13] S.C.Turaga, J.F. Murray, V. Jain, F.Roth, M.Helmstaedter, K.Briggman, W.Denk, and H.S.Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

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

[15] L. Sifre. Rigid motion scattering for image classification. *PhD thesis*, Ph. D. thesis, 2014.

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

[17] J.Jin, A.Dundar, E.Culurciello. Flattened convolutional neural networks for feedforward acceleration. arXiv preprint arXiv:1412.5474, 2014.

[18] M. Wang, B. Liu, and H. Foroosh. Factorized convolutional neural networks. arXiv preprint arXiv:1608.04337, 2016.

[19] F. Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357v2, 2016.

[20] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet level accuracy with 50x fewer parameters and¡ 1mb model size. arXiv preprint arXiv:1602.07360, 2016.

[21] V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3088–3096, 2015.

[22] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large scale machine learning on heterogeneous systems, 2015. Software available from tensorflow. org, 1, 2015.

[25] T. Tieleman and G. Hinton. Lecture 6.5 rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[26] H. Andrew, Z. Menglong, C. Bo, K. Dmitry, W. Weijun, W. Tobias, A. Marco, A. Hartwig. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861v1, 2017.