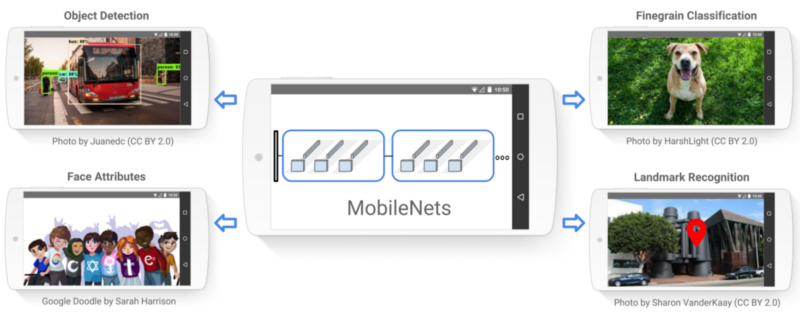
Surya Suravarapu Follow
Oct 11, 2017 · 3 min read

MobileNets: Efficient CNNs for Mobile Vision Applications


Object Detection
Photo by Juanedc (CC BY 2.0)
Face Attributes
Google Doodle by Sarah Harrison
MobileNets
Finegrain Classification
Photo by HarshLight (CC BY 2.0)
Landmark Recognition
Photo by Sharon VanderKaay (CC BY 2.0)

모든걸 놓는 얘기

Read full paper linked here: http://arxiv.org/abs/1704.04861

It's been six months or so since this paper came out from Google. Late to the party here, but better late than never, I guess! Straight to the point—the fundamental difference between regular Convolution Networks to MobileNets architecture is that it takes hugely less number of parameters to train and by doing so more efficient than the regular CNNs. Efficiency has a trade-off with accuracy: would that architecture fit to your use case? A question best answered based on the specifics of the problem one is trying to solve. For now, I would look at this architecture as a tool in my tool-set.

Overview

Fundamental approach of MobileNets is based on:

- A concept called ***depthwise separable convolutions***, and
- **Two global hyper-parameters** that you can use to arrive at the desired balance between ***latency and accuracy***.

원는 쓰면 안됨 !

Important reminder here is we are not talking about GPUs but the mobile as our target to perform Vision ML activities classification, facial recognition, object detection etc. and hence the discussion of trade-offs is an important one.

> *This paper proposes a class of network architectures that allows a model developer to specifically choose a small network that matches the resource*

> *restrictions (latency, size) for their application.*

Architecture

The architecture is based on depthwise separable filters. In the traditional models a convolution filter is applied for the entire depth. In the case of color pictures there are three channels—Red (R), Green (G), and Blue (B). The filters are applied across all the channels. In the MobileNets the filters are applied at a particular channel. Then the second step is a point wise convolution, a 1x1 convolution to combine the outputs of the depthwise convolutions. So, one layer for filtering (per channel) and one layer for combining.

> *Standard convolutions have the computational cost of:*

비용

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

> *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 .*

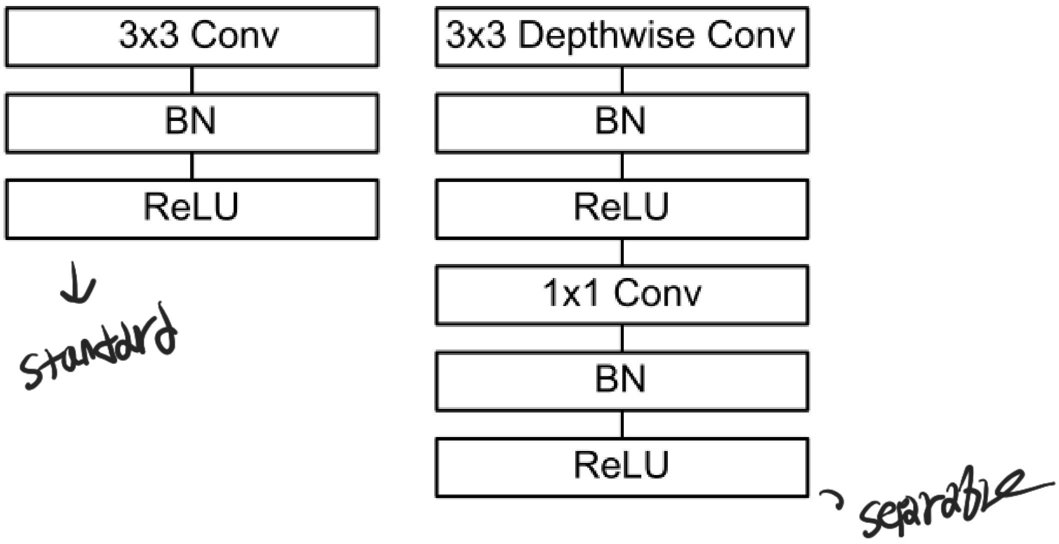
Computational cost of MobileNets depthwise convolutions is:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

The first portion before '+' is for the depthwise filtering, and the second one for the linear combination of the depthwise filters.

In other words —

> *MobileNet uses 3×3 depth-wise separable convolutions which uses between 8 to 9 times less computation than standard convolutions at only a small reduction in accuracy.*



Left: Standard Convolutional Layer with BatchNorm and ReLU activation. Right: Depthwise separable convolutions including depthwise and pointwise layers followed by Bathchnorm and ReLU activation

Global Hyper-parameters

비틀다.

Two hyper-parameters to tweak: alpha and resolution multiplier.

alpha: determines how much of the MobileNets architecture to use. From Keras documentation:

> ***alpha:*** *controls the width of the network.*

> *— If `alpha` < 1.0, proportionally decreases the number of filters in each layer.*

> *— If `alpha` > 1.0, proportionally increases the number of filters in each layer.*

> *— If `alpha` = 1, default number of filters from the paper are used at each layer.*

해상도!

> ***depth_multiplier:*** *The number of depthwise convolution output channels for each input channel. Also known as resolution multiplier*

Conclusion

I for one, would love to play with this architecture and see how my current CNN-based models perform. Great to see Keras, my current favorite framework, supports this architecture in their applications module. Why wait?

Before closing, linking a couple of nice write-ups on the topic using Tensorflow:

Creating insanely fast image classifiers with MobileNet in TensorFlow

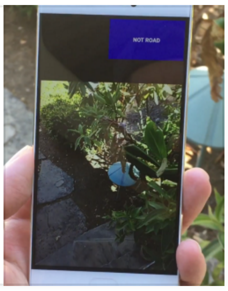
"It's like hot dog not hot dog, but for roads."

hackernoon.com

Building an insanely fast image classifier on Android with MobileNets in TensorFlow

Part two of a two-part series: It's like hot dog not hot dog, but for roads

hackernoon.com



Read complete paper here: http://arxiv.org/abs/1704.04861