MobileNets Paper Notes

Links:

Tags:

#ML #paper

Context:

Paper Notes

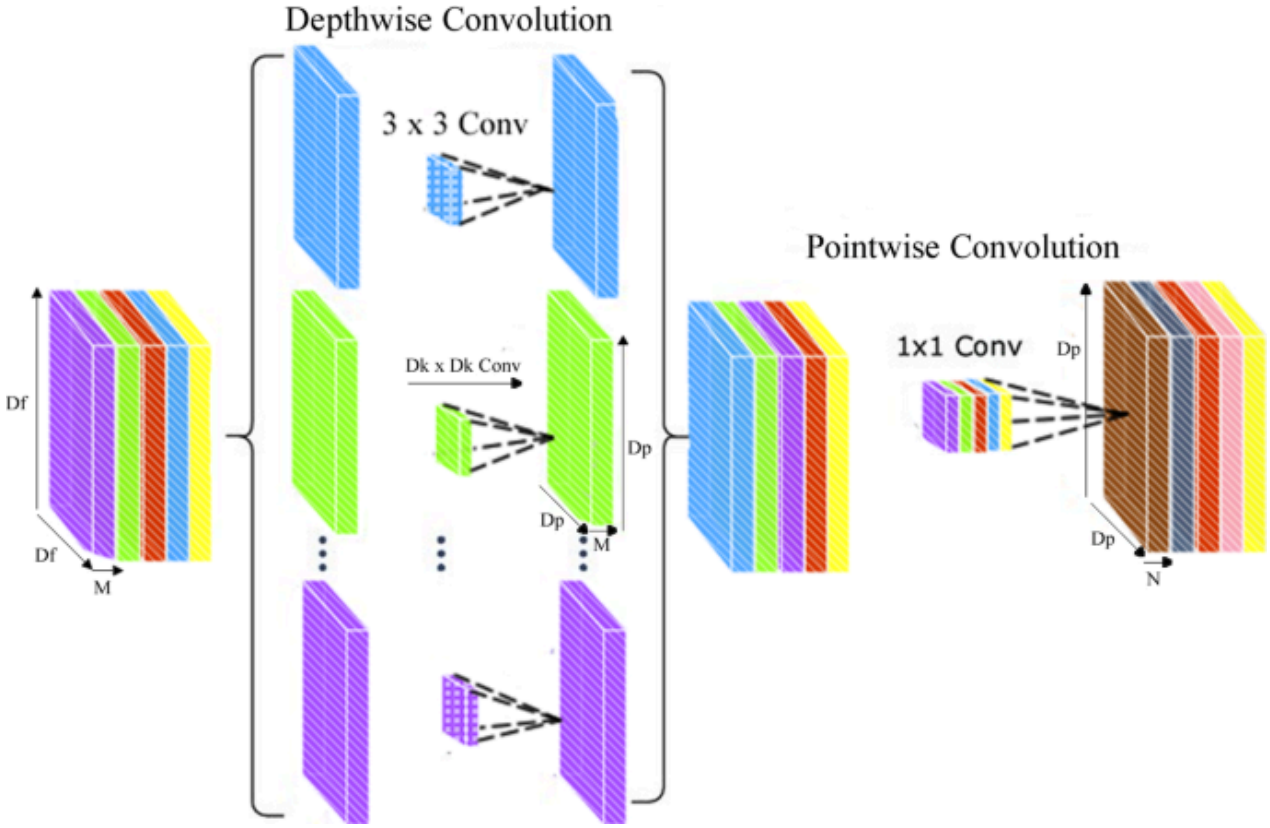
Findings:

Motivation

- The main goal of this paper is to build meaningful network architectures for high accuracy, latency and size
- This research is main done for the applications were small size and low latency models are req like mobiles, robotics, self-driving cars etc.
- From the prior works which refer to many different ways of obtaining a small network are shrinking, factorizing and compressing pre-trained networks. Compression based on product quantization, pruning, vector quantization and Huffman coding
- A method to train small networks if distillation that uses larger networks to teach smaller networks

MobileNet Architecture

- The architecture is based on depth wise separable convolutions
- The main focus in the architecture is Depth wise Convolution
- Depth wise Convolution very similar to conventional conv filters, but these are applied along the depths of channels instead of along the spatial dimensions



- This combination of Depth wise Convolution and point-wise Convolution is call Depth separable Convolution

By expressing convolution as a two step process of filtering and combining we get a reduction in computation of:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNet uses 3×3 depthwise separable convolutions which uses between 8 to 9 times less computation than standard convolutions at only a small reduction in accuracy as

- The above image shows by how much factor does the computation cost decreases by using this type of convolution
- **Network structure and training**:

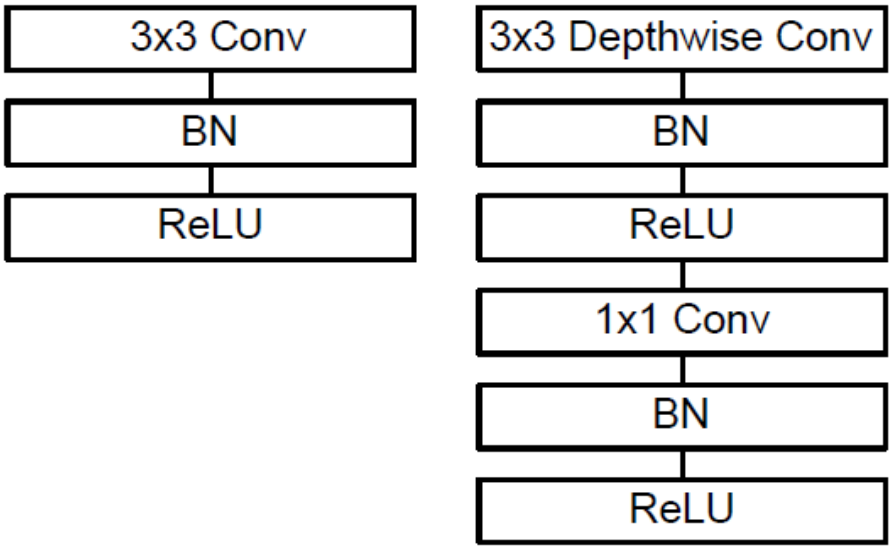


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

-
- This pic here shows how a normal Conv block is replaced by a depth wise conv block here
- Just mention in this paper an algorithm is used GEMM(general matrix multiply function) that is used to do very efficient matrix operations
- In these MobileNets 95% of the parameters are present in the point-wise convolution and the other 5% in the depth wise convolution
- While training these nets they had to implement near to nothing to reduce overfitting since smaller networks have a very hard time overfitting hence for pre-processing less regularization and a some data augmentation was done

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

-
- **Width Multipliers: Thinner Models**:
  - In this section to get the best architecture of the model for the specific use case, a hyperparameter was introduced $aplha$ which is used to basically thin the network uniformly at each layer
  - This loses the features from the image from which extract
  - This basically decreases the number of channels in the output feature maps

    - Due to this Width Multipliers the number of input channels and number of output channels decrease to '$aplha * number\ of\ channels$' where $alpha$ belongs between $(0, 1]$
    - The number of parameter is proportional to roughly $alpha^2$
    - This gives a decent trade-off between accuracy, latency and size
- **Resolution Multiplier: Reduced Representation**:
    - The second hyperparameter is rho
    - This basically reduces the resolution of the input image
    - This also has roughly a rho^2 proportionality to computational resources needed
    - This loses the detail in the input image

Table 3. Resource usage for modifications to standard convolution. Note that each row is a cumulative effect adding on top of the previous row. This example is for an internal MobileNet layer with $D_K = 3, M = 512, N = 512, D_F = 14$.

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

- **Model Shrinking Hyperparameters**:

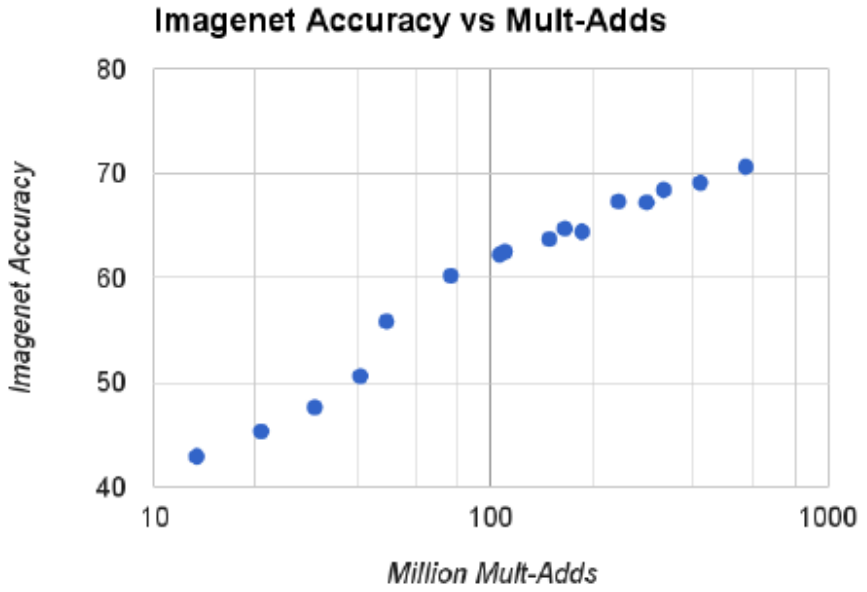


Figure 4. This figure shows the trade off between computation (Mult-Adds) and accuracy on the ImageNet benchmark. Note the log linear dependence between accuracy and computation.

- cross product of width multiplier $\alpha \in \{1, 0.75, 0.5, 0.25\}$ and resolutions $\{224, 192, 160, 128\}$. Results are log linear
- with a jump when models get very small at $\alpha = 0.25$.

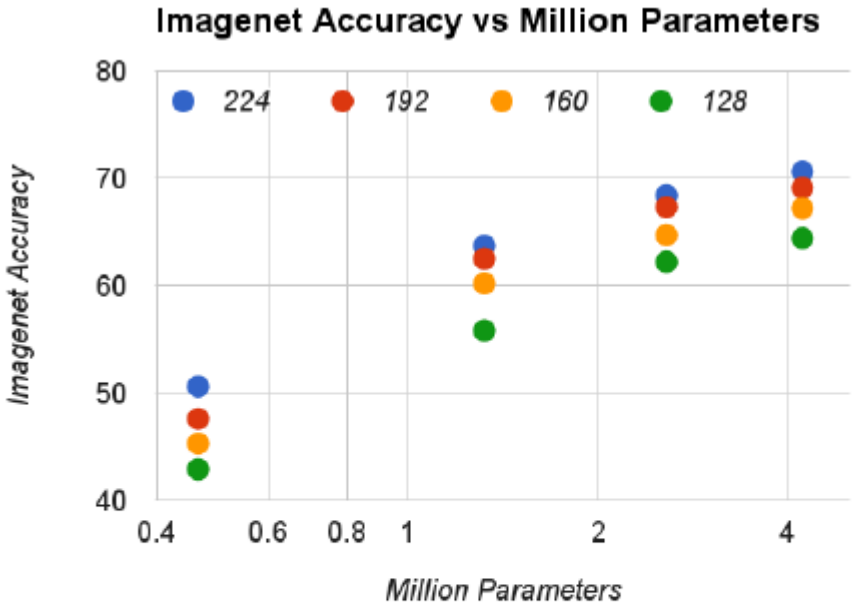


Figure 5. This figure shows the trade off between the number of parameters and accuracy on the ImageNet benchmark. The colors encode input resolutions. The number of parameters do not vary based on the input resolution.

-

made from the cross product of width multiplier $\alpha \in \{1, 0.75, 0.5, 0.25\}$ and resolutions $\{224, 192, 160, 128\}$.

- **Comparisons**:

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

  - This is one comparisonwhich talks about the model efficiency of MobileNet
- MobileNets are used and tested in many different fields like Fine Grain Recognition, Face Attributes, Object Detection, Face Embeddings.