Factorization in Deep Neural Networks - Part 2

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Course organisation

Sessions

1. Deep Learning and Transfer Learning,
2. Quantification,
3. Pruning,
4. Factorization,
5. Fact. pt.2 : Operators and Architectures,
6. Distillation,
7. Embedded Software and Hardware for DL.
8. Presentations for challenge.

Course organisation

Sessions

1. Deep Learning and Transfer Learning,
2. Quantification,
3. Pruning,
4. Factorization,
5. Fact. pt.2 : Operators and Architectures,
6. Distillation,
7. Embedded Software and Hardware for DL.
8. Presentations for challenge.

General principle

Complexity of 2D Convolutions

$N_{ops} = k.l.C_{in} C_{out}$
with kernel size (k, l), C_{in} input feature maps and C_{out} output feature maps.

To reduce the number of parameters, we can :

- Reduce the size of kernels
- Reduce the number of feature maps

Two strategies :

- Decompose kernels
- Depthwise convolutions

General principle

Complexity of 2D Convolutions

$N_{ops} = k.l.C_{in}\,C_{out}$
with kernel size (k,l), C_{in} input feature maps and C_{out} output feature maps.

To reduce the number of parameters, we can :

- Reduce the size of kernels
- Reduce the number of feature maps

Two strategies :

- Decompose kernels
- Depthwise convolutions

General principle

Complexity of 2D Convolutions

$N_{ops} = k\,.\,l\,.\,C_{in}\,C_{out}$
with kernel size $(k\,,\,l)$, C_{in} input feature maps and C_{out} output feature maps.

To reduce the number of parameters, we can :

- Reduce the size of kernels
- Reduce the number of feature maps

Two strategies :

- Decompose kernels
- Depthwise convolutions

General principle

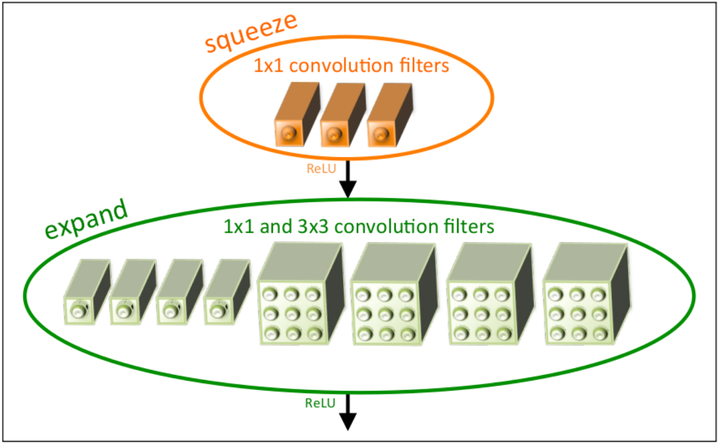
Complexity of 2D Convolutions

$N_{ops} = k.l.C_{in}C_{out}$
with kernel size (k, l), C_{in} input feature maps and C_{out} output feature maps.

Decomposing kernels

Assuming $C_{in} = C_{out}$, decompose (k, l) kernel by $(k, 1)$ and $(1, l)$:
$N_{ops} = k.1.C_{in}^2 + 1.l.C_{in}^2 = (l + k).C_{in}^2$
with kernel size (k, l), C_{in} input and out feature maps.

Introducing the Fire Module



Iandola et al. 2016, `https://arxiv.org/abs/1602.07360`

Depthwise convolutions

Instead of learning parameters that recombine all input feature maps to compute each feature maps,

use "groups" of D input feature maps to compute DC_{in} output feature maps.

Complexity of a Depthwise 2D Convolution

$N_{ops} = k.l.C_{in}.D.C_{in}$

with kernel size (k, l), C_{in} feature maps and D is Depth.

Depthwise convolutions

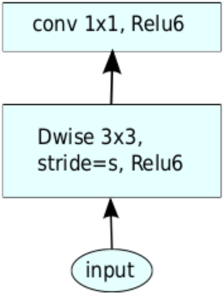
Instead of learning parameters that recombine all input feature maps to compute each feature maps,
use "groups" of D input feature maps to compute DC_{in} output feature maps.

Complexity of a Depthwise 2D Convolution

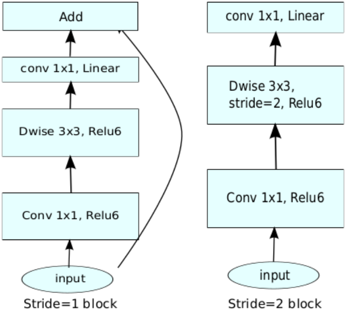
$N_{ops} = k.l.C_{in}.D.C_{in}$
with kernel size (k, l), C_{in} feature maps and D is Depth.

MobileNet

MobileNetV1



MobileNetV2



https://arxiv.org/abs/1704.04861 and https://arxiv.org/abs/1801.04381

Accuracy obtained on ImageNet

Network	Accuracy(%)	Params (M)
SqueezeNet	57.5	1.24
MobileNetV1	70.6	4.20
MobileNetV2	72.0	3.40

https://arxiv.org/abs/1704.04861 and https://arxiv.org/abs/1801.04381

Introducing Shift Attention Layer

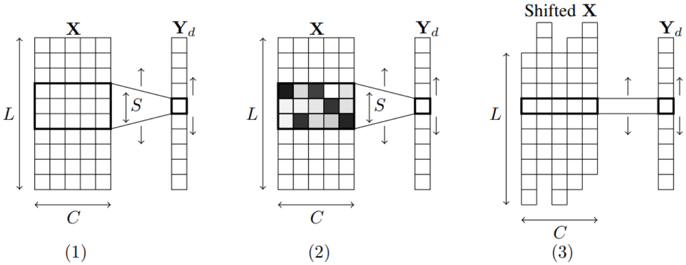


Figure 1: Overview of the proposed method: we depict here the computation for a single output feature map d, considering a 1d convolution and its associated shift version. Panel (1) represents a standard convolutional operation: the weight filter $\mathbf{W}_{d,\cdot,\cdot}$ containing SC weights is moved along the spatial dimension (L) of the input to produce each output in $\mathbf{Y}_d$. In panel (2), we depict the attention tensor $\mathbf{A}$ on top of the weight filter: the darker the cell, the most important the corresponding weight has been identified to be. At the end of the training process, $\mathbf{A}$ should contain only binary values with a single 1 per slice $\mathbf{A}_{d,c,\cdot}$. In panel (3), we depict the corresponding obtained shift layer: for each slice along the input feature maps (C), the cell with the highest attention is kept and the others are disregarded. As a consequence, the initial convolution with a kernel size S has been replaced by a convolution with a kernel size 1 on a shifted version of the input $\mathbf{X}$. As such, the resulting operation in panel (3) is exactly the same as the shift layer introduced in Wu et al. [2017], but here the shifts have been trained instead of being arbitrarily predetermined.

Hacene et al. 2019, `https://arxiv.org/abs/1905.12300`

Alternatives to Convolution

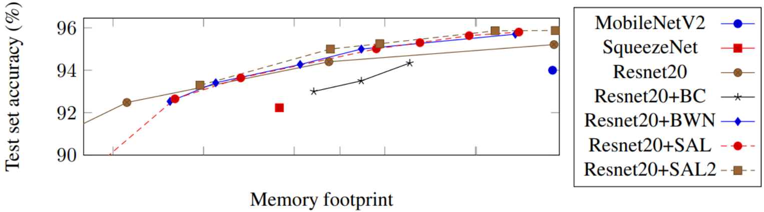


Figure 7: Evolution of accuracy when applying compression methods on different DNN architectures trained on CIFAR10.

Hacene et al. 2019, `https://arxiv.org/abs/1905.12300`