Summary of "*Xception*: Deep Learning with Depthwise Separable Convolutions"

Iván Vallés Pérez

Title : *Xception*: Deep Learning with Depthwise Separable Convolutions

Authors : François Chollet

Journal : Google report

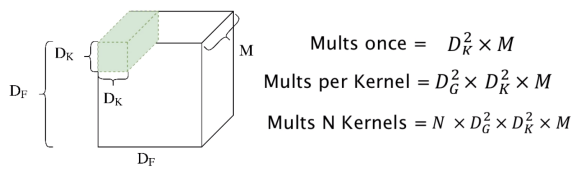
Date : April 2017

Convolution vs depthwise separable convolution[1]

Given an input tensor of shape (1, D_F, D_F, M), we are interested in applying a convolutional operation producing an output tensor of shape (1, D_G, D_G, N)

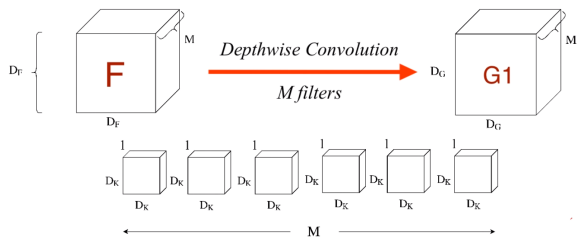
Convolution

A convolutional layer consists of N kernels with size D_K. In order to perform this operation, the number of needed operations are: $M \cdot D_K^2 \cdot D_G^2 \cdot N$

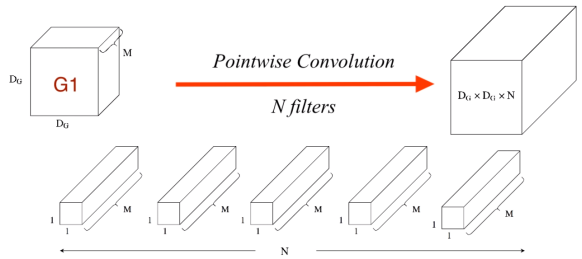


Depthwise separable convolution

1. Filtering stage: performed using the depthwise convolution operation. This operation consists of applying M convolutions of D_K size, separatedly (one for each channel); instead of one for all the channels. I.e. for each channel, only one different kernel is used. This operation produces an output tensor of shape (1, D_G, D_G, M). The number of needed operations at this stage is $M \cdot D_K^2 \cdot D_G^2$.



2. Combining stage: performed using the pointwise convolution operation. This operation consists of a set of N size 1 convolutions applied over the output of the previous step, leading to a tensor of shape (1, D_G, D_G, N). The number of needed operations at this stage is then $M \cdot D_G^2 \cdot N$

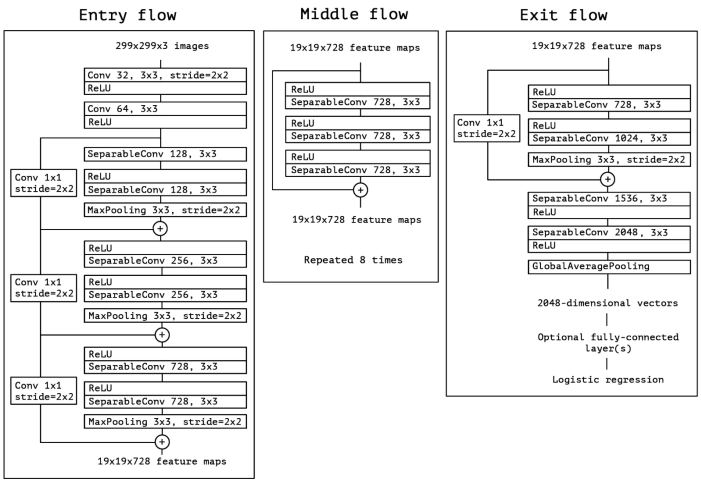


The total operations performed in the depthwise separable convolution layer sums to $M \cdot D_K^2 \cdot D_G^2 + M \cdot D_G^2 \cdot N$, which reduces to $M \cdot D_G^2 \cdot (D_K^2 + N)$. This is $\frac{1}{N} \cdot \frac{1}{D_K^2}$ times the operations needed by the standard convolution.

[1] Material borrowed from here `https://www.youtube.com/watch?v=T7o3xvJLuHk&t=337s`

Xception architecture

- The main goal of the paper is to compare *Inception* V3 with a *Xception* architecture with similar number of parameters to see which one performs the best.
- The *Xception* is inspired in the *Inception* architecture which achieved a much more efficient usage of the parameters of the network (less operations with better performance).
- The *Xception* algorithm exploits the deepthwise separable convolutions to reduce the number of operations needed in each step.
- One difference between *Xception* and *Inception* is the order of opperations: *Xception* applies a 1x1 convolution as a final operation (pointwise convolution) while *Inception* applies it at the begining.
- Another important difference between *Xception* and *Inception* is that in *Inception*, each of the convolutions are followed by a non-linearity while in *Xception* not including non-linearities between depthwise and pointwise convolutions showed better performance.
- The last but not less important difference is the addition of skip connections in the *Xception* architecture.
- The main hypothesis being tested in this paper is that the mapping of cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled. As it is a stronger hypothesis than the one tested in the *Inception* algorithm, the author decided to call it *Xception* (extreme *Inception*).
- In short, the *Xception* architecture is a linear stack of depthwise separable convolution layers with residual connections. The full architecture is described in the image below (borrowed from the original paper), in which all Convolution and SeparableConvolution layers are followed by batch normalization, even if it is not included in the diagram.



Results and conclusions

Both architectures were tested over two datasets. The *Xception* architecture achieved a significantly better performance without tunning the architecture to each specific problem.

- The residual connections (or skip connections) showed to be a key element for the high performance obtained.
- Intermediate activations in the depthwise separable layers have been tested leading to a worse result.
- The autor concludes with a recommendation: replace *Inception* modules by depthwise separable convolutions, in neural computer vision architectures.