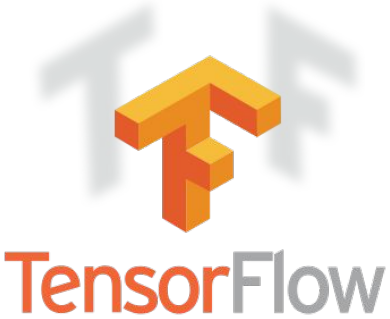
Tensorflow NN Complexity Scaling

Umberto Fazio

Advanced Computer Architectures - A.Y 2017-18
Prof. Cristina Silvano, Ahmet Erdem



Project description

For this project I implemented an algorithm in Tensorflow (python framework for Neural Network) in order to explore the ranges of parameters related to computation complexity.

The process of computation of the metrixes (inference time, FLOPS, memory and weights) is been automated as much as possible. In the configuration file, for each parameter, a list of different values can be chosen. The program will consider every possible configuration.

Parameters

- Number of layers: each layer has a conv layer and a max pool layer
- Number of filters for each conv layer
- Batch size
- Input size
- Number of classes
- Kernel size
- Fully Connected unites
- Kernel Stride

Computed Metrixes

- FLOPS: as sum of all FLOPS operations of every conv layer and fully connected layer

$$FLOPS_{fc} = NumInputs * NumNeurons$$

$$FLOPS_{conv} = 2(k*k)+NumChannels+NumFilters+W_{out}+H_{out}$$

- Memory (runtime memory)

$$Memory = W_{input} * H_{input} * NumFilters$$

- Weights (model memory)

$$Weights = k * k * NumChannels * NumFilters$$

Example

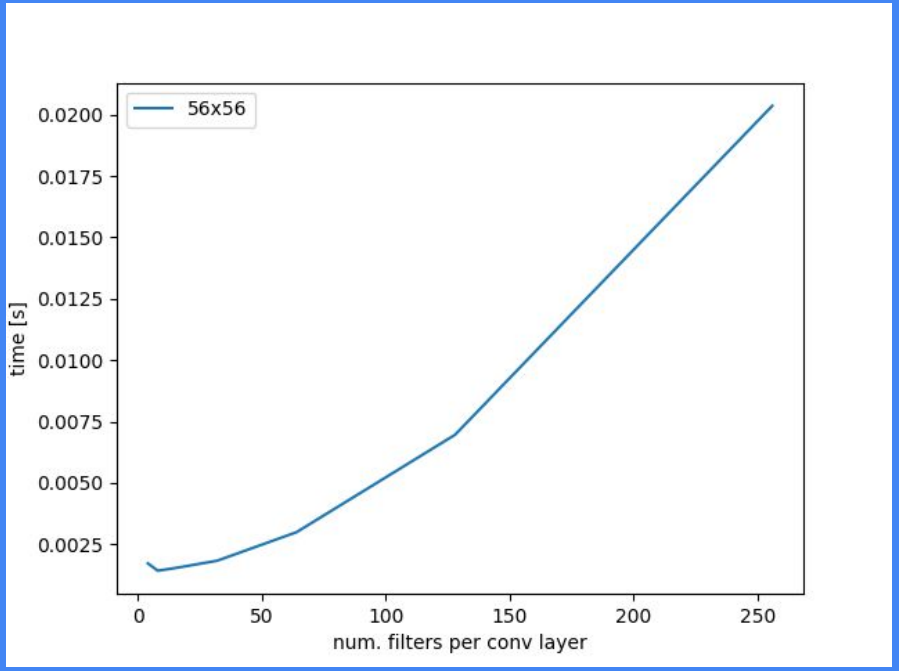
```
CONV0 [400x400x32] memory 5120000 weights 864 FLOPS 40053
POOL0 [100x100x32] memory 320000
CONV1 [100x100x32] memory 320000 weights 9216 FLOPS 2582
POOL1 [25x25x32] memory 20000
CONV2 [25x25x32] memory 20000 weights 9216 FLOPS 251
POOL2 [6x6x32] memory 1152
CONV3 [6x6x32] memory 1152 weights 9216 FLOPS 91
POOL3 [1x1x32] memory 32
FC [1x1x512] memory 512 weights 16384 FLOPS 16384
TOTAL PARAMETERS 44896
TOTAL MEMORY 23211392 bytes
TOTAL FLOPS 59361
inference took 0.014644440015157063 seconds
```

(NN of 4 layers, 32 filters, batch size of 1, 400x400x3 input images, 2 classes, 3x3 kernel, 52 fc units and kernel stride of 2)

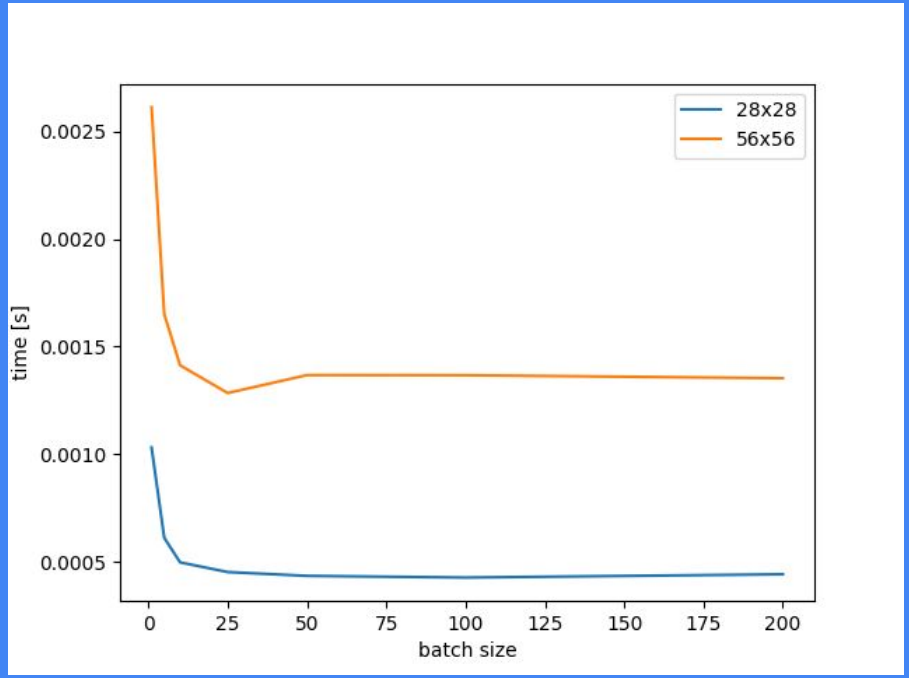
Some results

(in each graph, every the parameter of the NN is fixed, except for one which takes different values)

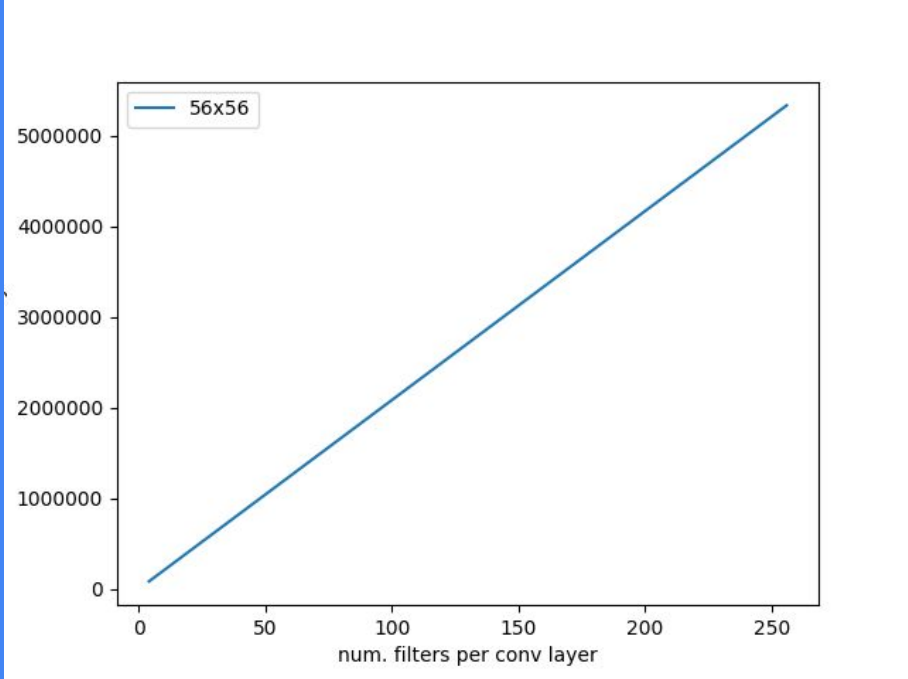
Inference time over number of filters



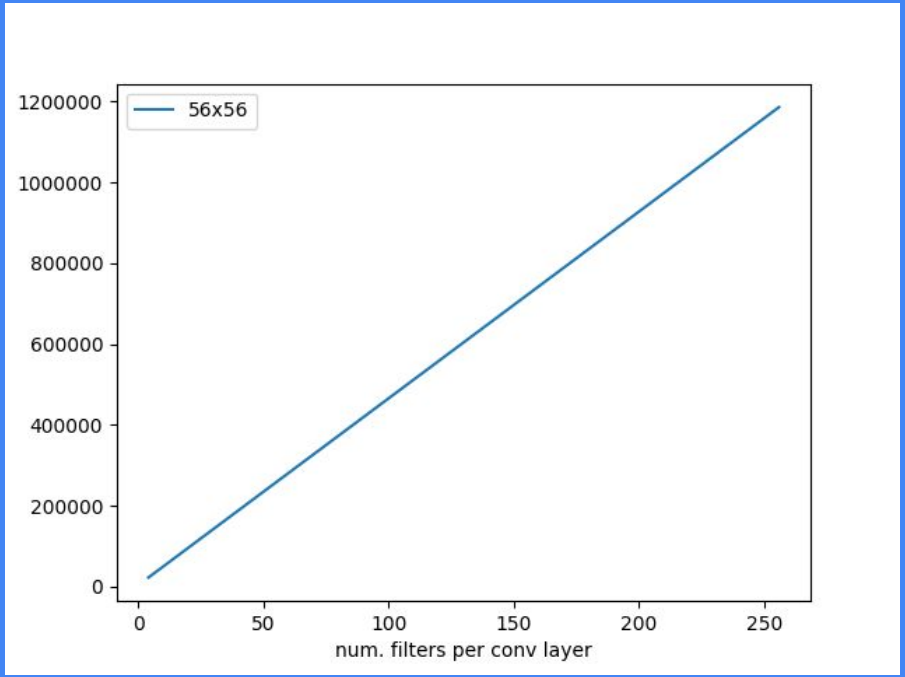
Inference Time over batch size



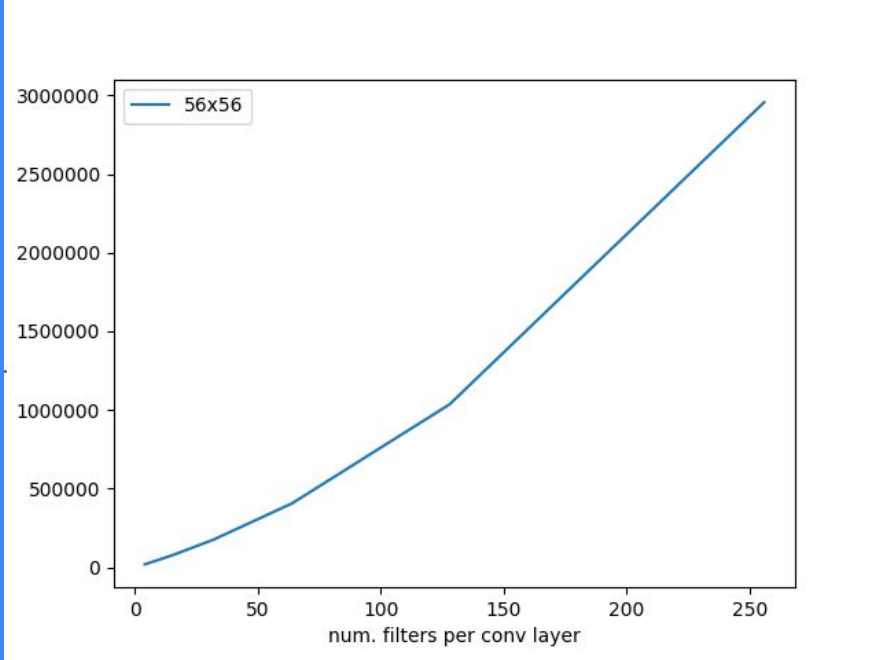
Total memory usage over number of filters



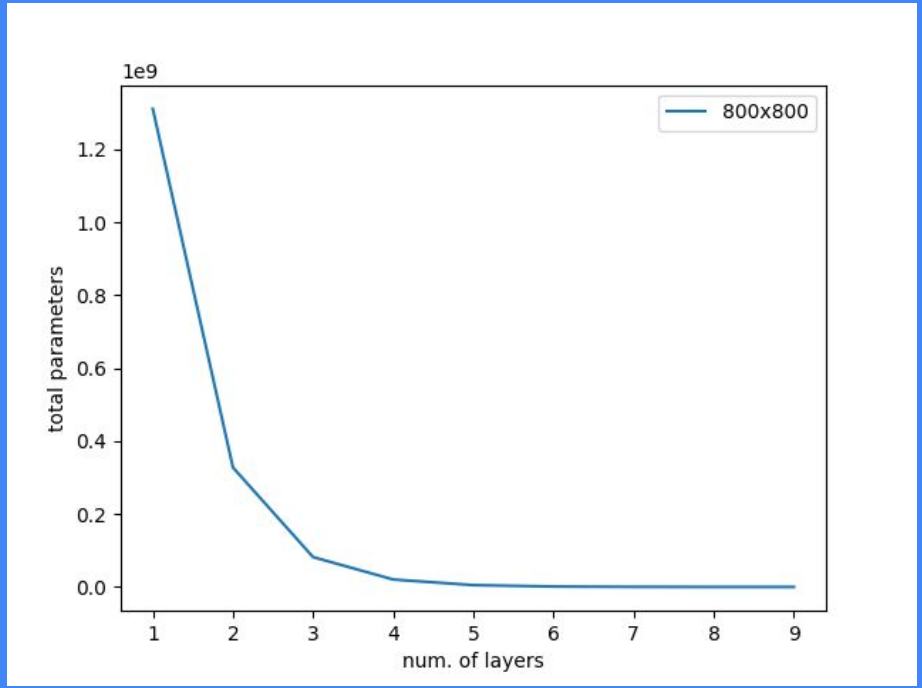
FLOPS over number of filters



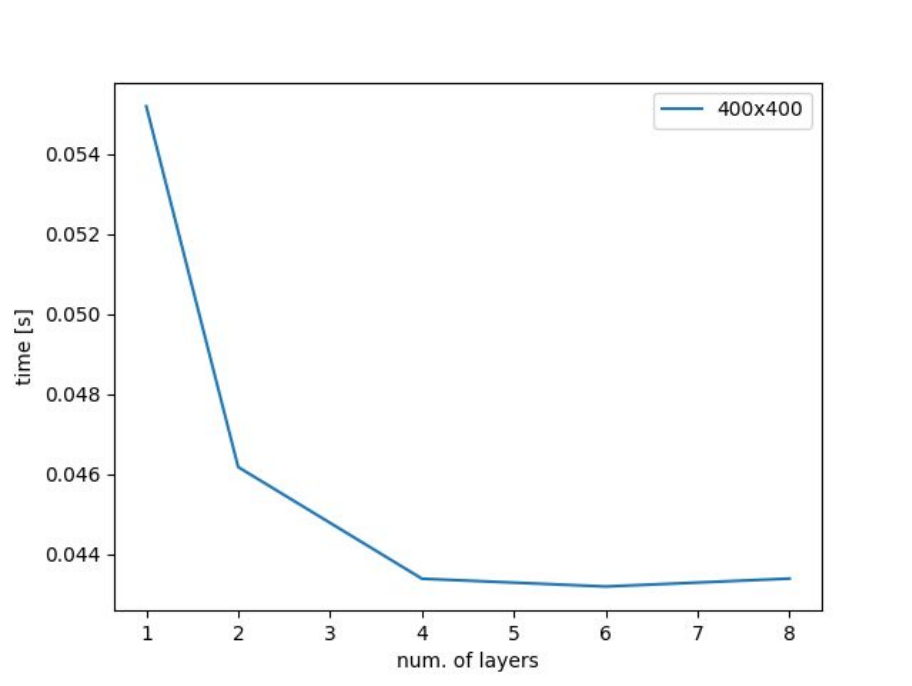
Total parameters over number of filters



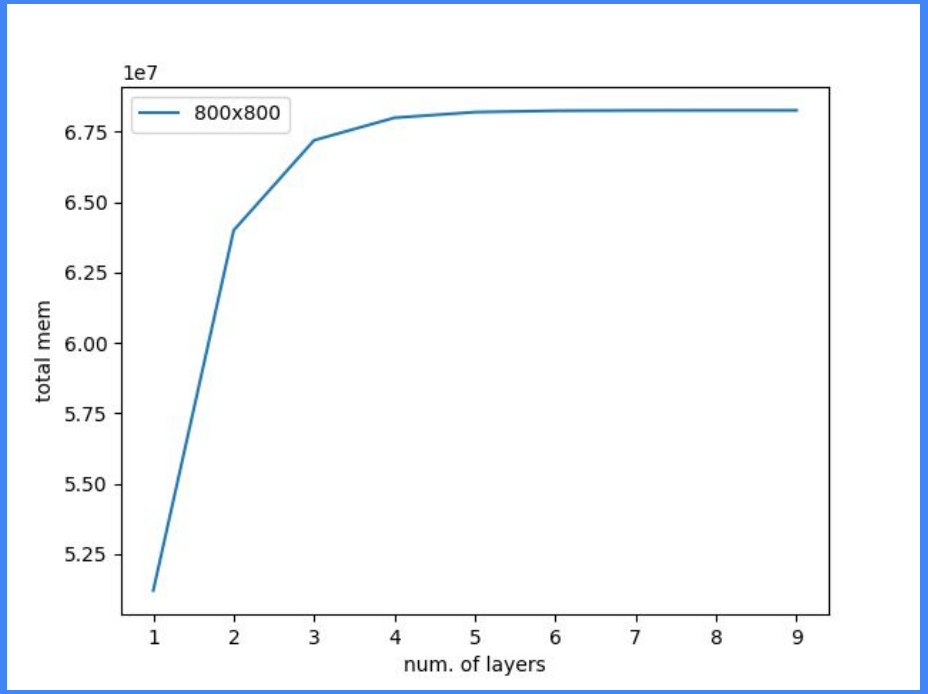
Total parameters over number of layers



Inference time over number of layers



Total memory over number of layers



Grazie!

You can find the code on GitHub at:

https://github.com/Umbi14/ACA-Project-TF-NN-Complexity-Scaling