MobileDets: Searching for Object Detection Architectures for Mobile Accelerators (Supplementary Material)

Yunyang Xiong *
University of Wisconsin-Madison
yxiong43@wisc.edu

Hanxiao Liu *
Google
hanxiaol@google.com

Suyog Gupta
Google
suyoggupta@google.com

Berkin Akin, Gabriel Bender, Yongzhe Wang, Pieter-Jan Kindermans, Mingxing Tan
Google
{bakin,gbender,yongzhe,pikinder,tanmingxing}@google.com

Vikas Singh
University of Wisconsin-Madison
vsingh@biostat.wisc.edu

Bo Chen
Google
bochen@google.com

This is the supplementary material for our submission. In this document, we describe further details of the relationship between the building blocks of MobileDet search space and the linear structure of Tucker/CP decomposition.

1. Connections with Tucker/CP decomposition

The proposed layer variants can be linked to Tucker/CP decomposition. Fig. 1 shows the graphical structure of an inverted bottleneck with input expansion ratio s, modulo nonlinearities. This structure is equivalent to the sequential structure of approximate evaluation of a regular convolution by using CP decomposition [2]. The Tucker convolution layer with input and output compression ratios s and e, denoted as Tucker layer shown in Fig. 3, has the same structure (modulo nonlinearities) as the Tucker decomposition approximation of a regular convolution [1]. Fused inverted bottleneck layer with an input expansion ratio s, shown in Fig. 2, can also be considered as a variant of the Tucker decomposition approximation.

Details of the approximation is as follows. CP-decomposition approximates convolution using a set of sequential linear mappings: a 1×1 pointwise convolution, two depthwise convolutions along the spatial dimensions, and finally another pointwise convolution. Since the kernel size of convolution is quite small, e.g. 3×3, or 5×5, the decomposition along the spatial dimensions does not save much computation. Without performing the decomposition along the spatial dimensions, the sequential graphical structure is equivalent to inverted bottleneck in Fig. 1. Similarly, a mode-2 Tucker decomposition approximation of a convolution along input and output channels involves a sequence of three operations, a 1×1 convolution, then a $K \times K$ regular convolution, then another 1×1 pointwise convolution. This sequential structure of Tucker layer is shown in Fig. 3. Combining the first 1×1 pointwise convolution and the second $K \times K$ regular convolution as one $K \times K$ regular convolution gives the fused inverted bottleneck layer in Fig. 2.

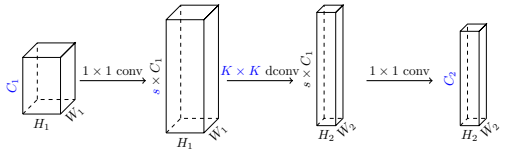

Figure 1: Inverted bottleneck layer: 1×1 pointwise convolution transforms the input channels from C_1 to $s \times C_1$ with input expansion ratio $s > 1$, then $K \times K$ *depthwise* convolution transforms the input channels from $s \times C_1$ to $s \times C_1$, and the last 1×1 pointwise convolution transforms the channels from $s \times C_1$ to C_2. The highlighted C_1, s, K, C_2 in IBN layer are searchable.

We therefore refer to the expansion operation as fused convolution layer, the compression operation as Tucker layer, and our proposed search space with a mix of both layers and IBNs as the MobileDets search space.

*Equal contribution.

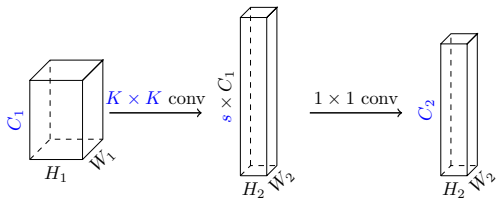

Figure 2: Fused inverted bottleneck layer: $K \times K$ *regular* convolution transforms the input channels from C_1 to $s \times C_1$ with input expansion ratio $s > 1$, and the last 1×1 pointwise convolution transforms the channels from $s \times C_1$ to C_2. The highlighted C_1, K, s, C_2 in the fused inverted bottleneck layer are searchable.

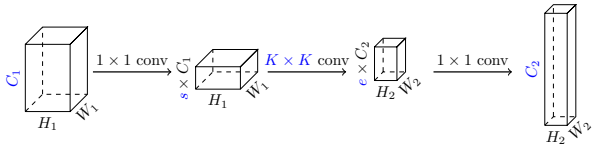

Figure 3: Tucker layer: 1×1 pointwise convolution transforms the input channels C_1 to $s \times C_1$ with input compression ratio $s < 1$, then $K \times K$ *regular* convolution transforms the input channels from $s \times C_1$ to $e \times C_2$ with output compression ratio $e < 1$, and the last 1×1 pointwise convolution transforms the channels from $e \times C_2$ to C_2. The highlighted C_1, s, K, e, C_2 in Tucker layer are searchable.

References

[1] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015. 1

[2] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 1