StrassenNets: Deep Learning with a Multiplication Budget

Michael Tschannen[1,2,*], Aran Khanna[2,*], Anima Anandkumar[2,3,*]

[1]ETH Zürich [2]Amazon AI [3]Caltech [*]work done at Amazon

Notebook: http://bit.ly/2Akmerp

Abstract

Matrix multiplications can be cast as 2-layer sum-product (SP) networks (arithmetic circuits), disentangling multiplications and additions. We leverage this observation for end-to-end learning of low cost (in terms of multiplications) approximations of linear operations in neural network layers. Specifically, we propose to replace matrix multiplication operations by SP networks, with widths corresponding to the budget of multiplications we want to allocate to each layer, and to learn them end-to-end. Experiments on ImageNet show that this method applied to ResNet yields significantly higher accuracy than existing methods for a given multiplication budget, or leads to the same or higher accuracy compared to existing methods while using significantly fewer multiplications.

Learning fast matrix multiplications via sum-product networks

Given square matrices $A, B \in \mathbb{R}^{n\times n}$, the product $C = AB$ can be represented as a 2-layer sum-product (SP) network (arithmetic circuit)

$$\mathrm{vec}(C) = W_C[(W_B\mathrm{vec}(B)) \odot (W_A\mathrm{vec}(A))] \quad (1)$$

where $W_A, W_B \in \mathbb{K}^{r\times n^2}$ and $W_C \in \mathbb{K}^{n^2\times r}$, $\mathbb{K} := \{-1, 0, 1\}$, and $\odot$ denotes the element-wise product (see Fig. 1 left for an illustration).

1. General case $A \in \mathbb{R}^{k\times m}$, $B \in \mathbb{R}^{m\times n}$: $C = AB$ can be written in the form (1) if $r \geq nmk$.
2. If $A \in \mathbb{R}^{k\times m}$ is fixed and $B \in \mathbb{R}^{m\times n}$ concentrates on low-dimensional subspace of $\mathbb{R}^{k\times m}$: Can find W_A, W_B s.t. (1) holds approximately even when $r \ll nmk$

Idea

Leverage item 2 to learn low-cost (in terms of multiplications) approximate linear operations in neural network layers by associating A with (pretrained) weights/filters and B with corresponding activations/feature maps, and by learning W_A, W_B, W_C end-to-end.

Benefits

Acceleration: Potentially massive speedups on dedicated hardware such as FPGA at small (prediction) accuracy loss [2].

Generality: Can be applied to fully connected layers, N-dimensional convolutions, group convolutions, deformable convolutions, etc.

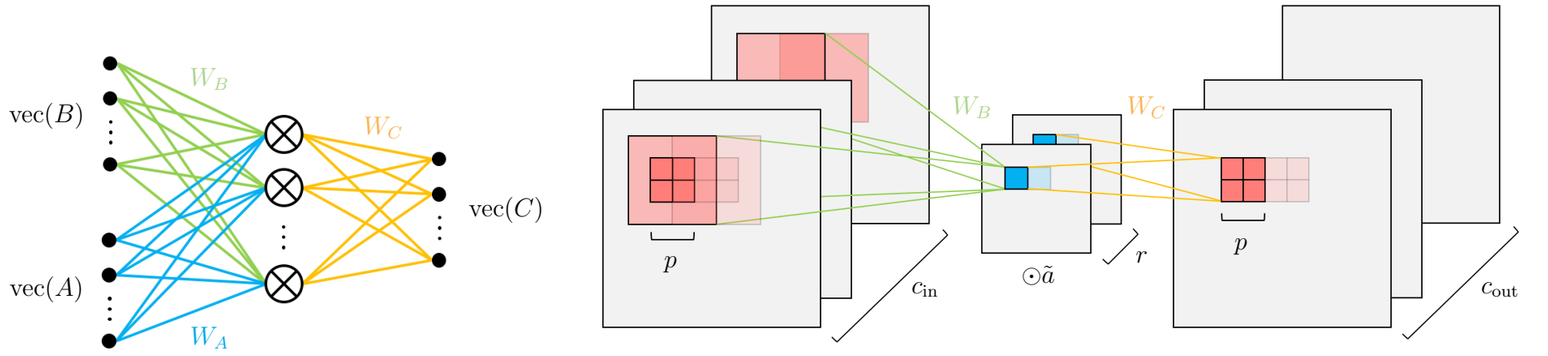

Figure 1: **Left:** Illustration of the SP network (1). **Right:** Application of the proposed framework to 2D convolution leads to p-strided 2D convolution with W_B, followed by channel-wise scaling with by $\tilde{a}$, followed by $1/p$-strided transposed 2D convolution with W_C.

2D convolution as example application

- Standard 2D convolution can be cast as matrix multiplication using the `im2col` format and compressed by (1). However, this results in impractically large W_A, W_B, W_C.
- Instead, we compress the computation of $c_{\text{out}} \times p \times p$ output elements of the convolution from $c_{\text{in}} \times (p-1+k) \times (p-1+k)$ input elements by applying (see Fig. 1 right)
  1. a p-strided 2D convolution with a *ternary* filter of size $r \times c_{\text{in}} \times (p-1+k) \times (p-1+k)$ (corresponding to multiplication with W_B; optionally implemented as a g-grouped convolution),
  2. a channel-wise multiplication with an $r \times 1 \times 1$ *full-precision* filter $\tilde{a} = W_A\mathrm{vec}(A)$ (corresponding to the filters of the original convolution),
  3. a $1/p$-strided transposed 2D convolution with a *ternary* filter of size $c_{\text{out}} \times r \times p \times p$ (corresponding to multiplication with W_C).

Experiment

Method applied to ResNet-18 and evaluated on ImageNet.

Implementation: Apache mxnet & gluon

Training protocol: (SGD with momentum 0.9, mini-batch size 128)

1. Train full-precision $\tilde{a}, W_A, W_B$ for 70 epochs with learning rate (LR) 0.05 (multiply the LR by 0.1 after 40 and 60 epochs)
2. Activate quantization (quantization method from [3]), train for 10 epochs with LR 0.0005
3. Fix W_A, W_B and train $\tilde{a}$ for 5 epochs with LR 0.0005

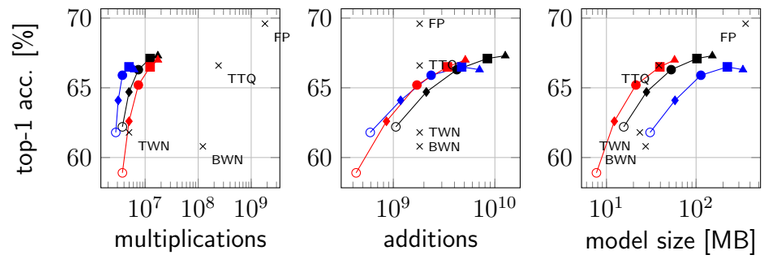

Figure 2: Top-1 validation accuracy. Blue: $p = 2$, $g = 1$; black: $p = 1$, $g = 1$; red: $p = 1$, $g = 4$.

model size

$(p,g) \setminus r$	$6c_{\text{out}}$	$4c_{\text{out}}$	$2c_{\text{out}}$	c_{out}	$\frac{1}{2}c_{\text{out}}$
$(1,1)$	57.45	71.33	85.22	92.16	95.63
$(2,1)$	6.64	37.46	68.28	83.69	91.4
$(1,4)$	83.88	88.95	94.03	96.56	97.83

multiplications

$(p,g) \setminus r$	$6c_{\text{out}}$	$4c_{\text{out}}$	$2c_{\text{out}}$	c_{out}	$\frac{1}{2}c_{\text{out}}$
$(1,1)$	99.04	99.32	99.59	99.73	99.79
$(2,1)$	99.66	99.73	99.79	99.83	99.85
$(1,4)$	99.04	99.32	99.59	99.73	99.79

additions

$(p,g) \setminus r$	$6c_{\text{out}}$	$4c_{\text{out}}$	$2c_{\text{out}}$	c_{out}	$\frac{1}{2}c_{\text{out}}$
$(1,1)$	-596.81	-364.61	-132.42	-16.32	41.73
$(2,1)$	-288.62	-159.15	-29.69	35.05	67.41
$(1,4)$	-181.56	-87.78	6.00	52.89	76.33

Table 1: Reduction [%] in the number of multiplications, additions, and model size.

Bibliography

[1] V. Elser. A network that learns Strassen multiplication. Journal of Machine Learning Research, 17(116):113, 2016.

[2] E. Nurvitadhi et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In FPGA, pp 514, 2017.

[3] F. Li, B. Zhang, and B. Liu. Ternary weight networks. arXiv:1605.04711, 2016.