StrassenNets: Deep Learning with a Multiplication Budget

Michael Tschannen[1,*], Aran Khanna[2,*], Anima Anandkumar[3,4,*]

[1]ETH Zürich [2]Dolores Technologies [3]Amazon AI [4]Caltech [*]work done at Amazon

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

Abstract

We perform end-to-end learning of low-cost approximations of matrix multiplications in DNN layers by casting matrix multiplications as 2-layer sum-product networks (SPNs) (arithmetic circuits) and learning their (ternary) edge weights from data. The SPNs disentangle multiplication and addition operations and enable us to impose a budget on the number of multiplication operations. Combining our method with knowledge distillation and applying it to image classification DNNs (trained on ImageNet) and language modeling DNNs (using LSTMs), we obtain a first-of-a-kind reduction in number of multiplications (over 99.5%) while maintaining the predictive performance of the full-precision models. Finally, we demonstrate that the proposed framework is able to rediscover Strassen's matrix multiplication algorithm, learning to multiply 2×2 matrices using only 7 multiplications instead of 8.

Casting matrix multiplications as SPNs

Given matrices $A \in \mathbb{R}^{k\times m}$, $B \in \mathbb{R}^{m\times n}$ and $r \geq nmk$, the product $C = AB$ can be represented as a 2-layer SPN (arithmetic circuit)

$$\mathrm{vec}(C) = W_c[(W_b\mathrm{vec}(B)) \odot (W_a\mathrm{vec}(A))], \qquad (1)$$

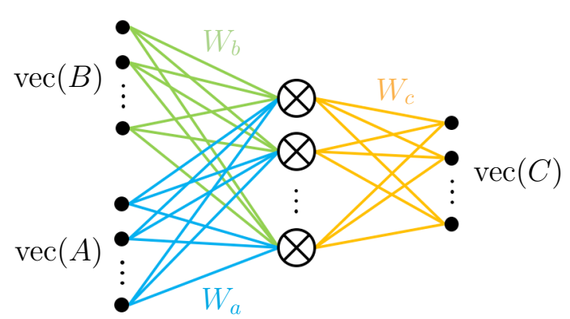
where $W_a \in \mathbb{K}^{r\times km}$, $W_b \in \mathbb{K}^{r\times mn}$ and $W_c \in \mathbb{K}^{kn\times r}$, $\mathbb{K} := \{-1, 0, 1\}$, and $\odot$ denotes the element-wise product (see Fig. 1 left).

Learning fast matrix multiplications via SPNs

- If A is fixed and B concentrates on low-dimensional subspace of $\mathbb{R}^{m\times n}$: We can find W_a, W_b s.t. (1) holds approximately even when $r \ll nmk$
- Associate A with (pretrained) weights/filters of a DNN and B with corresponding activations/feature maps, and **learn low-cost approximate matrix multiplications by learning W_a, W_b, W_c end-to-end**.

Benefits

- **Acceleration: 99.5%+ reduction in multiplications, potentially leading to massive inference speedups and energy savings on dedicated hardware such as FPGA and ASIC**
- **Generality: Our method can be applied to any general matrix multiplication (GEMM) including fully connected layers, N-dimensional convolutions, group convolutions, deformable convolutions, etc.**



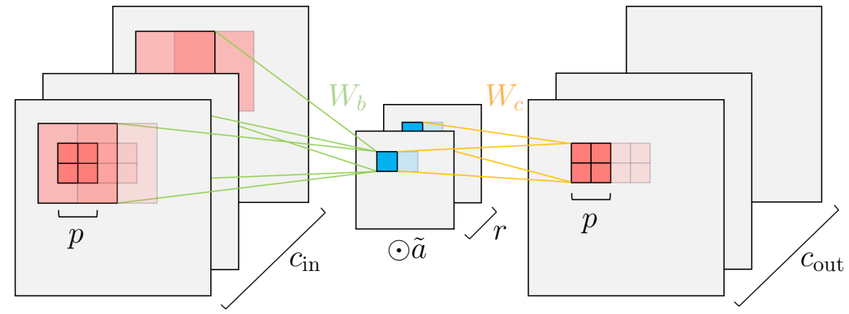


Figure 1: **Left:** 2-layer SPN (1) implementing an (approximate) matrix multiplication. The edges have weights in $\mathbb{K} = \{-1, 0, 1\}$. **Right:** Application to 2D convolution: p-strided ternary 2D convolution with W_b, channel-wise scaling by $\tilde{a} = W_a\mathrm{vec}(A)$, $1/p$-strided transposed ternary 2D convolution with W_c.

Learning the SPNs end-to-end

- Learn ternary W_a, W_b, W_c for each DNN layer using SGD with ternary quantization via a **straight-through gradient estimator**
- Variants and extensions:
  - Jointly learn W_a and A as $\tilde{a} = W_a\mathrm{vec}(A)$
  - Fine-tune A or $\tilde{a}$ after learning W_a, W_b, W_c
  - **Knowledge distillation (KD):** Cross-entropy between outputs of compressed and full-precision network as a regularizer
- Application to 2D convolution:
  - Compressing 2D convolution as matrix multiplication: Inefficient!
  - **Preserve weight-sharing:** Compress 2D convolution over $p \times p$ patches and across channels via p-strided ternary (transposed) convolutions and channel-wise scaling (see Fig. 1)
  - **Multiplication reduction:** $\mathbf{c_{in}c_{out}whp^2/r}$

Experiments

1. **Rediscovering Strassen's algorithm:** Learn exact multiplication of 2×2 matrices with 7 multiplications from synthetic data
2. **Image classification:** ResNet-18 on ImageNet
3. **Language modeling:** Character-level language model involving convolution, highway, and LSTM layers on Penn Tree Bank (PTB)

Results

Fast exact 2×2 matrix multiplication discovered by our algorithm:

$$W_a = \begin{pmatrix} -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 0 & 1 & 0 \\ -1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \; W_b = \begin{pmatrix} -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ -1 & -1 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}, W_c = \begin{pmatrix} 1 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & -1 \\ -1 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

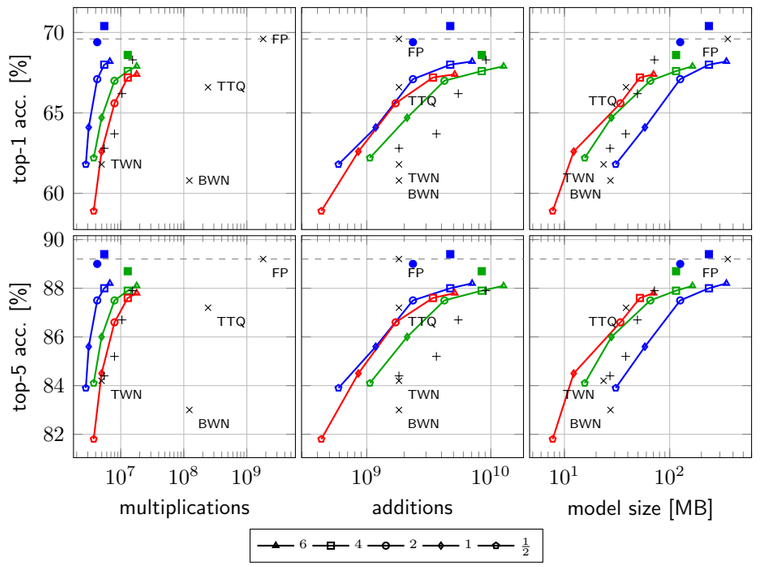


Figure 2: Validation accuracy of compressed ResNet-18 on ImageNet along with that of baselines. Marker types: r/c_{out} (solid markers with KD). Blue: $p = 2$, $g = 1$; green: $p = 1$, $g = 1$; red: $p = 1$, $g = 4$ (g: Groups in convolution realized by W_B).

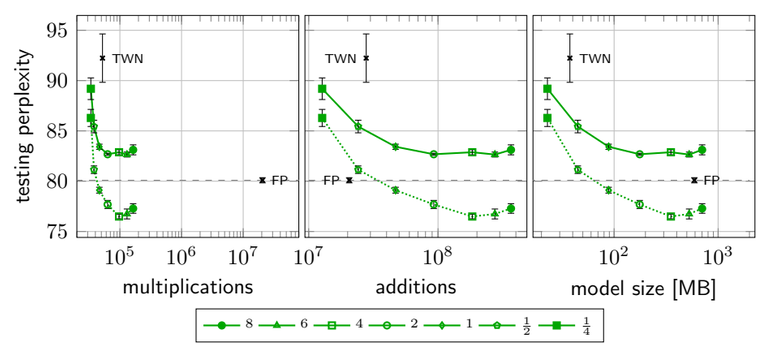


Figure 3: Testing perplexity for character-level language model on PTB. Solid line: Without KD; dotted line: With KD. Marker types: r/n_{out}.