Avoiding Communication in Convolutional Neural Networks

SURF Math Team
Anthony Chen Mason Haberle Jon Hillery Rahul Jain
Mentors: James Demmel Olga Holtz

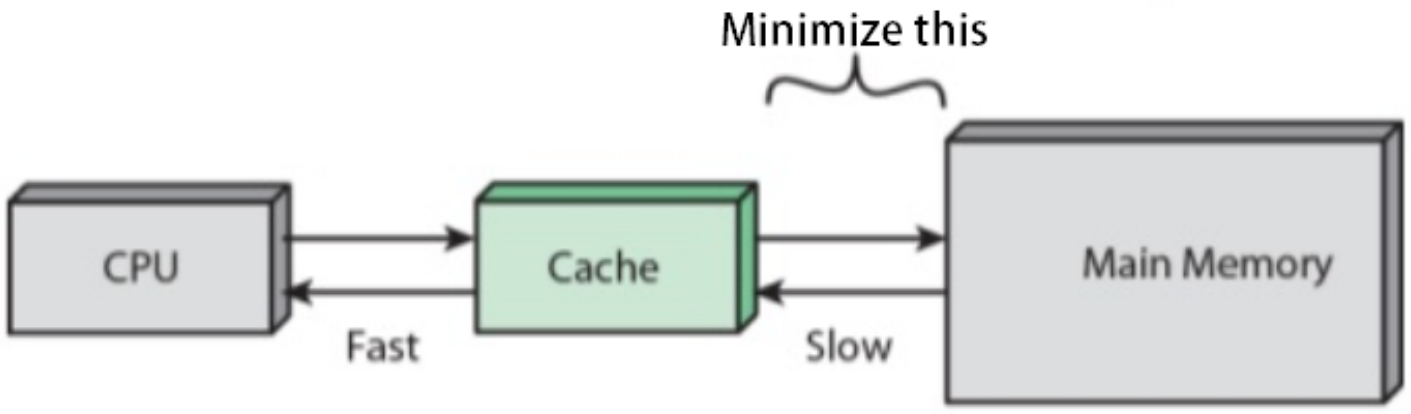
University of California, Berkeley

September 16, 2020

The Communication Model

- Arbitrarily large slow memory
- M spots in fast memory (cache)
- 1 communication = moving 1 word of data (1 integer) between slow and fast memory

Given a problem, minimize communication.



Communication-Avoiding Algorithm Speedups

- Up to **12x** faster for 2.5D matmul on 64K core IBM BG/P
- Up to **3x** faster for tensor contractions on 2K core Cray XE/6
- Up to **11.8x** faster for direct N-body on 32K core IBM BG/P
- Up to **6.2x** faster for All-Pairs-Shortest-Path on 24K core Cray CE6
- Up to **13x** faster for Tall Skinny QR on Tesla C2050 Fermi NVIDIA GPU

Massive speedups for a variety of important algorithms

Matrix Multiplication Bounds

Langou, Smith et al. show the following communication lower bound for 3-nested-loop matrix multiplication:

Theorem (Smith et al. 2019)

The number of words X communicated by an algorithm for 3-nested-loop matrix multiplication with matrices of size $m \times n$ and $n \times k$ on a machine with a fast memory size M is

$$X \geq \frac{2mnk}{\sqrt{M}} - 3M$$

Current optimal algorithms communicate about $\frac{2mnk}{\sqrt{M}} + mn$ words.

Proof Technique: Loomis-Whitney Inequality

- Smith et al. used the Loomis-Whitney inequality.
- Finite set $V \subset \mathbb{Z}^3$.
- Projections V_A, V_B, V_C onto each coordinate plane.
- $|V| \leq \sqrt{|V_A||V_B||V_C|}$

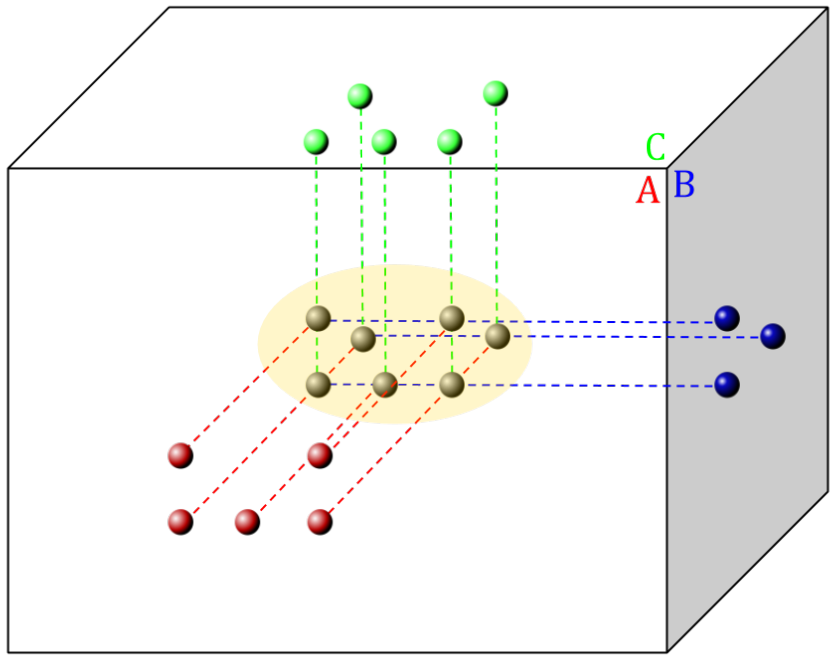


Figure: V in yellow represents all operations, projections V_A, V_B, V_C represent array accesses.

Proof Sketch: Matmul Communication Bound

To derive a communication lower bound for directly multiplying two $n \times n$ matrices $C = AB$, use the following steps:

1. Split code into L segments, each one doing M communications.

Proof Sketch: Matmul Communication Bound

To derive a communication lower bound for directly multiplying two $n \times n$ matrices $C = AB$, use the following steps:

1. Split code into L segments, each one doing M communications.
2. Each segment has access to $2M$ elements of A, B, and C.

Proof Sketch: Matmul Communication Bound

To derive a communication lower bound for directly multiplying two $n \times n$ matrices $C = AB$, use the following steps:

1. Split code into L segments, each one doing M communications.
2. Each segment has access to $2M$ elements of A, B, and C.
3. L-W inequality: Number of operations in a segment $\leq \sqrt{8M^3}$.

Proof Sketch: Matmul Communication Bound

To derive a communication lower bound for directly multiplying two $n \times n$ matrices $C = AB$, use the following steps:

1. Split code into L segments, each one doing M communications.
2. Each segment has access to $2M$ elements of A, B, and C.
3. L-W inequality: Number of operations in a segment $\leq \sqrt{8M^3}$.
4. Need to do n^3 operations, so need $n^3/\sqrt{8M^3}$ segments.

Proof Sketch: Matmul Communication Bound

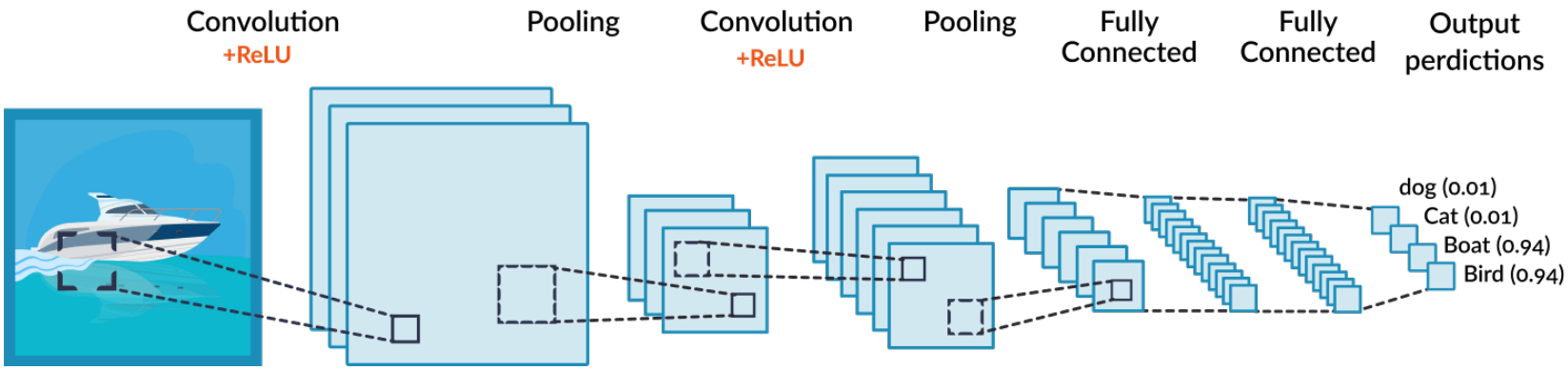
To derive a communication lower bound for directly multiplying two $n \times n$ matrices $C = AB$, use the following steps:

1. Split code into L segments, each one doing M communications.
2. Each segment has access to $2M$ elements of A, B, and C.
3. L-W inequality: Number of operations in a segment $\leq \sqrt{8M^3}$.
4. Need to do n^3 operations, so need $n^3/\sqrt{8M^3}$ segments.
5. M loads per segment, so need $n^3/\sqrt{8M}$ communications.

Smith et al. optimized the constant using Lagrange multipliers.

Convolutional Neural Networks

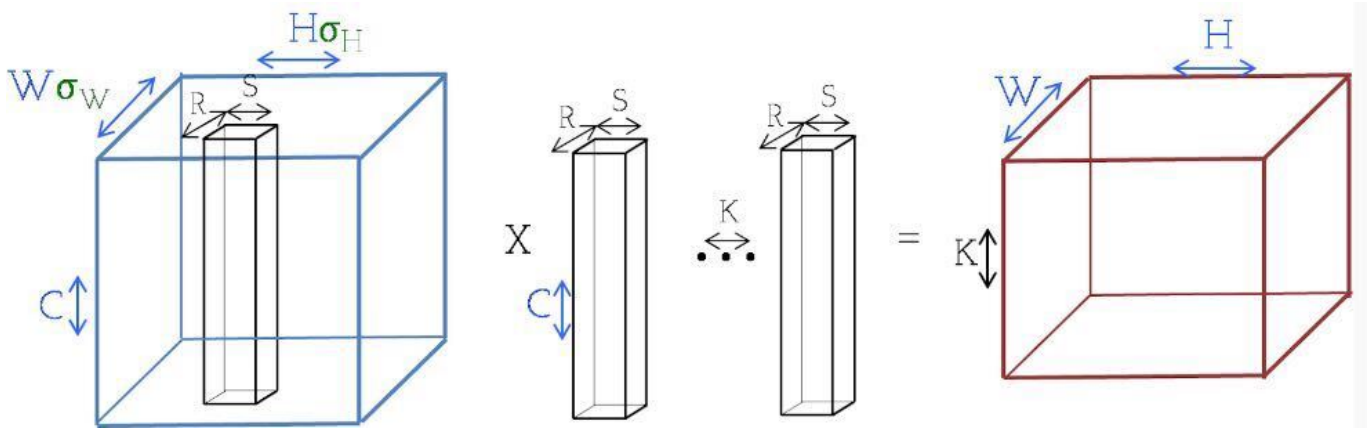
A **convolutional neural network** (CNN) is a simple but versatile machine learning tool based on the operation of convolving data.



A direct convolutional neural network is implemented by the following 7-nested-loop pseudocode:

```
for (b, c, k, w, h, r, s) = 0 to (B, C, K, W, H, R, S) - 1
    Out(k, h, w, b) += Image(r + σ_w w, s + σ_h h, c, b) × Filter(k, r, s, c)
```

Direct Convolution Model



The diagram above is a representation of the arrays involved.
The pseudocode operates on a collection of images with multiple channels and applies filters of size $R \times S$ to each channel.

```
for (b, c, k, w, h, r, s) = 0 to (B, C, K, W, H, R, S) - 1
```
$$Out(k, h, w, b) \mathrel{+}= Image(r + \sigma_w w, s + \sigma_h h, c, b) \times Filter(k, r, s, c)$$

Small Filters

For small filters, it will be helpful to divide out the filter loops by the strides. This changes the pattern of array accesses.
The mapping lifts our 7-dimensional lattice into 9 dimensions.

$$\begin{aligned}
&\texttt{for } (b,c,k,w,h)=0:(B,C,K,W,H) \texttt{ - } 1\\
&\quad\texttt{for } (r',r'',s',s'')=0:(R/\sigma_w,\sigma_w,S/\sigma_h,\sigma_h) \texttt{ - } 1\\
&\qquad Out(k,h,w,b) \texttt{ += } Image(r''+\sigma_w(r'+w),s''+\sigma_h(s'+h),c,b)\\
&\qquad\qquad\qquad\qquad \times Filter(k,r'',r',s'',s',c)
\end{aligned}$$

This lift is necessary to attain even asymptotically optimal lower bounds if the filters are small relative to the fast memory size.

Previous Asymptotic Bounds

Previous bounds are asymptotically optimal, but have no constants.

Theorem

The number of words X communicated by a CNN which does G operations, uses filters of size $R \times S$, image strides of σ_w, σ_h, and runs with a fast memory size M is

$$X \geq \Omega\left(\max\left\{\frac{G}{M}, \frac{G\sqrt{\sigma_w\sigma_h}}{\sqrt{RSM}}, |Out|, |Filter|, |Image|\right\}\right)$$

Precise Lower Bounds

We derive a communication lower bound for direct convolution (with constants!):

Theorem

The number of words X communicated by a CNN which does G operations, uses filters of size $R \times S$, image strides of σ_w, σ_h, and runs with a fast memory size M is

$$X \geq \max\left\{\frac{9G}{4M} - M, \frac{2G\sqrt{\sigma_w\sigma_h}}{\sqrt{RSM}} - 2M, |Out| + |Filter| + |Image|\right\}$$

Hölder-Brascamp-Lieb Inequalities

We use a generalization of the Loomis-Whitney Inequality:

Theorem (CDKSY'13)

Let $\phi_j : \mathbb{Z}^d \to \mathbb{Z}^{d_j}$ be group homomorphisms and let $s \in [0,1]^m$. Suppose for all subgroups $H \leq \mathbb{Z}^d$

$$rank(H) \leq \sum_{j=1}^{m} s_j rank(\phi_j(H))$$

Then for all nonempty finite $V \subseteq \mathbb{Z}^d$:

$$|V| \leq \prod_{j=1}^{m} |\phi_j(V)|^{s_j}$$

Optimizing Constants

To produce optimal constants, we combine several techniques.

Including:

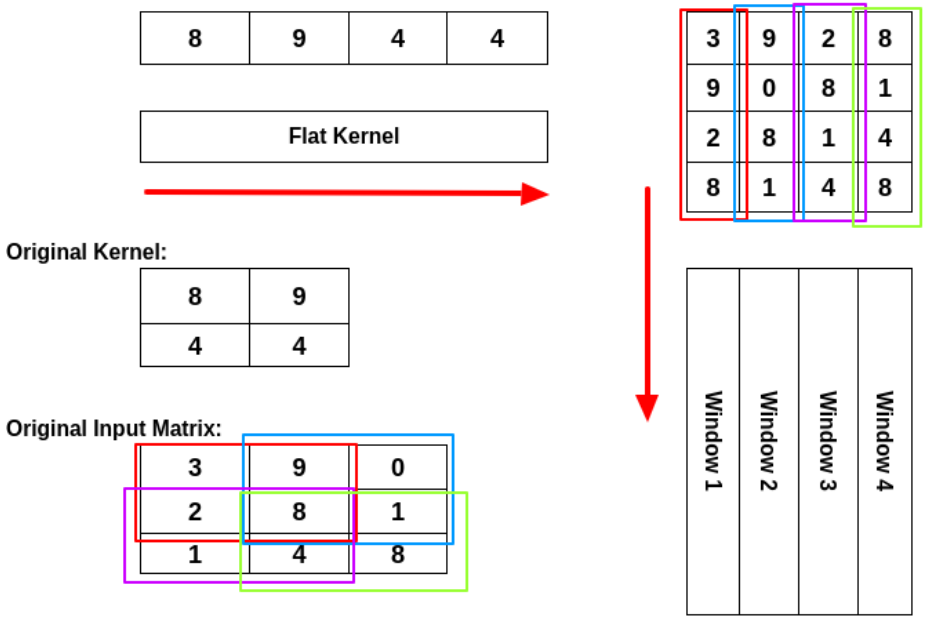
- Varying lengths of code segments and other parameters chosen in the proof.
- Optimizing over HBL exponents.
- Taking advantage of nonoverlapping arrays.

Like for matmul in Smith et al., we suspect the resulting precise lower bounds are nearly attainable.

Efficient Convolution Techniques: im2col

Image to Column:

- Goal: convert to a matrix multiplication.
- Duplicates entries of image
- Subject to communication-avoiding matmul algorithms with communication costs of about $\frac{2mnk}{\sqrt{M}}$



Efficient Convolution Techniques: FFT

If we duplicate the filter instead of the image, we end up with a block Toeplitz filter matrix F^* and a image vector $v(I)$.

We can turn F^* into a Toeplitz matrix T_F by adding appropriate rows. Toeplitz matrices can be diagonalized by the DFT matrix.

An optimal FFT has a communication volume of $2n \log_M n$, and with three FFTs/iFFTs, we get a total communication volume of

$$BCK(12Cw_I h_I \log_M(2w_I h_I) + 2w_I h_I)$$

Efficient Convolution Techniques: Winograd Convolutions

These are a Strassen-like family of techniques for computing convolutions with less multiplications and more additions.

If $F(m, r)$ is the number of multiplications needed for computing m outputs with a length r filter, then usually $F(m, r) = mr$.
However, it is a theorem that $\min_{\text{algs}} F(m, r) = m + r - 1$.
In particular we can write a convolution as:

$$Y = A^T((Gg) \odot (B^T d))$$

where g and d come from the image and filter. We can nest these one dimensional algorithms for a two dimensional minimal filter:

$$Y = A^T((GgG^T) \odot (B^T dB))A$$

Winograd Convolution Communication Analysis

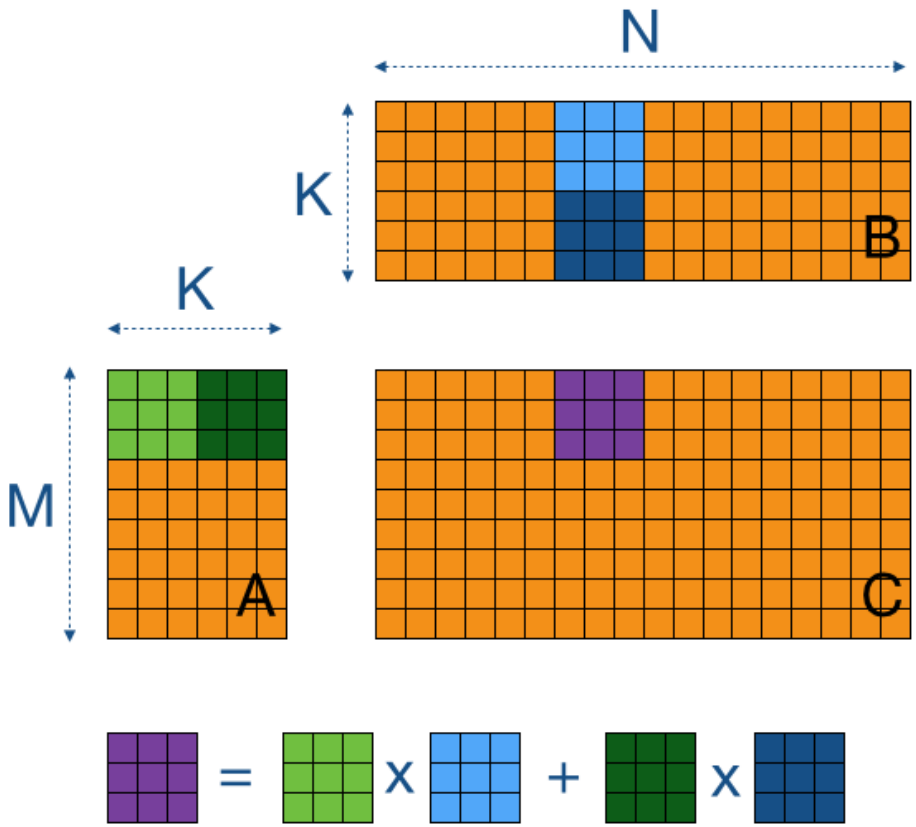
g is a $r \times r$ filter and d is a $(m + r - 1) \times (m + r - 1)$ image tile, with overlapping tiles. We have $P = \frac{h_I}{m} \frac{w_I}{m}$ tiles. $U = GgG^T$ only needs to be computed once for each g, and $V = B^T dB$ is computed once for each d so for the output

$$Y_{\tilde{x},\tilde{y}} = A^T \left(\sum_{c=1}^{C} U \odot V_{\tilde{x},\tilde{y}} \right) A$$

we have a substantial number of communication terms which we minimize in m.

Efficient Convolution Techniques: Blocking

- Access block of entries.
- Do all the operations you can with them.
- Move to a new block.
- Blocking comes from solving a linear program



Some Inequalities

For a blocking $[b_b, b_c, b_k, b_w, b_h, b'_r, b''_r, b'_s, b''_s]$ we know that

$$b_b b_k b_w b_h \leq \frac{M}{3}$$
$$b_c b_k b'_r b''_r b'_s b''_s \leq \frac{M}{3}$$
$$b_b b_c (b_w + b'_r)(b_h + b'_s) b''_r b''_s \leq \frac{M}{3}$$

so that the output, filter, and image blocks fit in memory. Taking logarithms, we can turn this into a linear program.

Linear Program for the Blocking

We take $l_i = \log_M b_i$ giving a linear program $\min c^T x$ subject to $Ax \leq l$ and bounds for each l_i. In particular, we have $c^T = \begin{bmatrix} -1 & \cdots & -1 \end{bmatrix}$

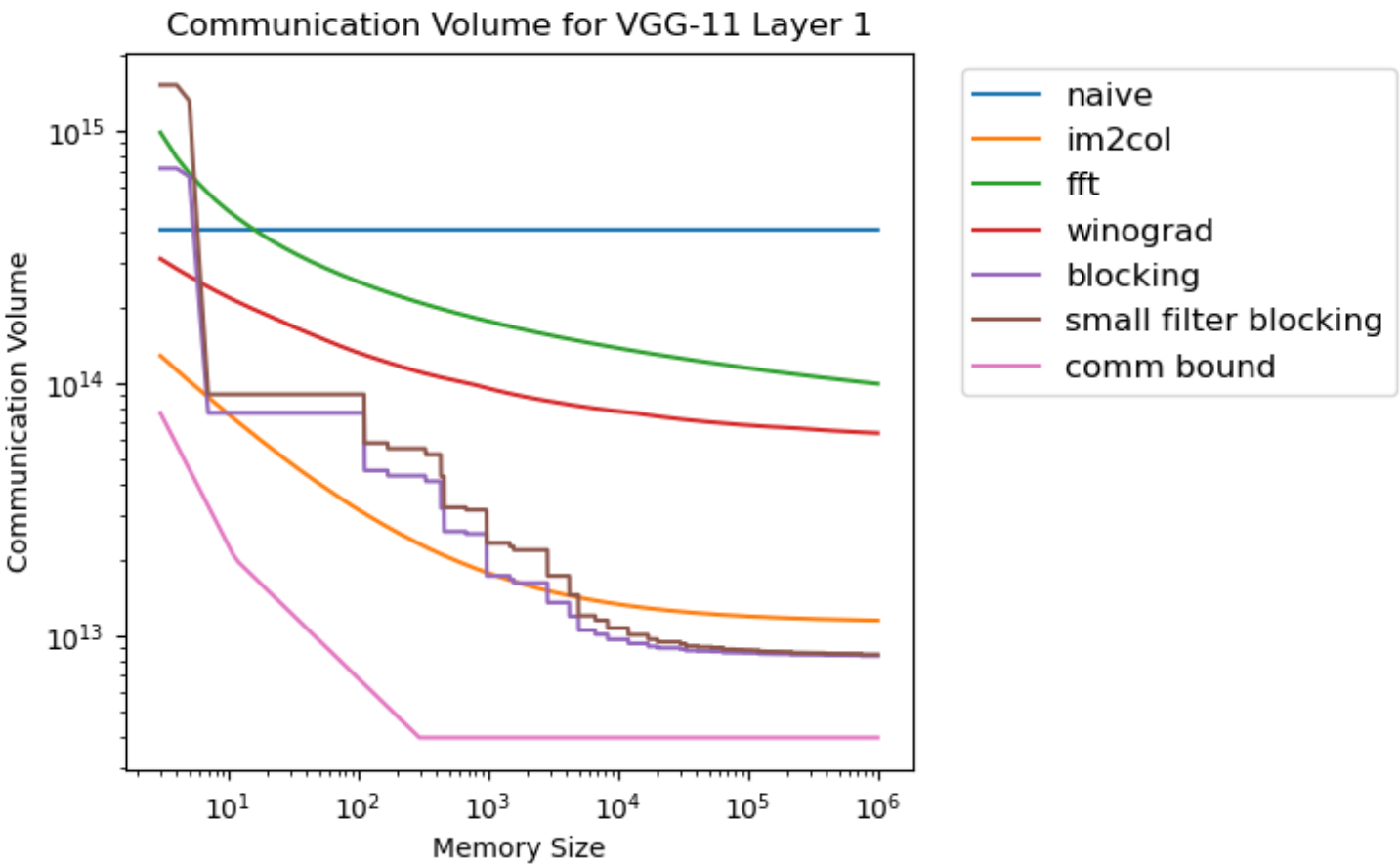
$$A = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$b^T = 1 - \begin{bmatrix} \log_M 3 & \log_M 3 & \log_M 12 & \log_M 12 & \log_M 12 & \log_M 12 \end{bmatrix}$$

and each $l_i \in [0, \log_M B_i]$ where B_i is the relevant bound. Then, to recover a blocking, we take M^{l_i} and round.

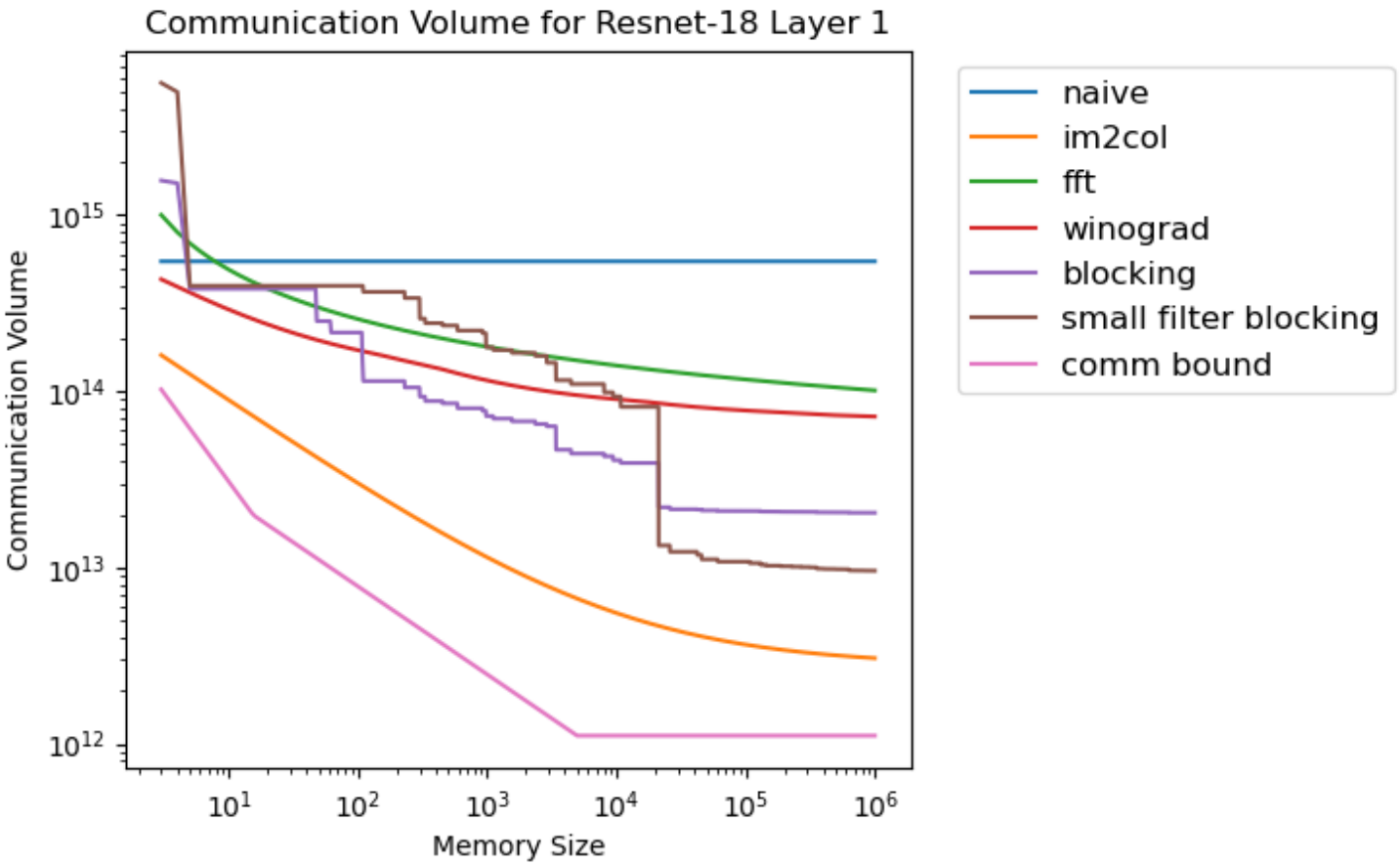
Attainability of the Bounds

With a batch size of 1.2 million, 3 input channels,
64 output channels, a 224 by 224 image, a 3 by 3 filter,
and strides of 1, we have communication volume as is shown:



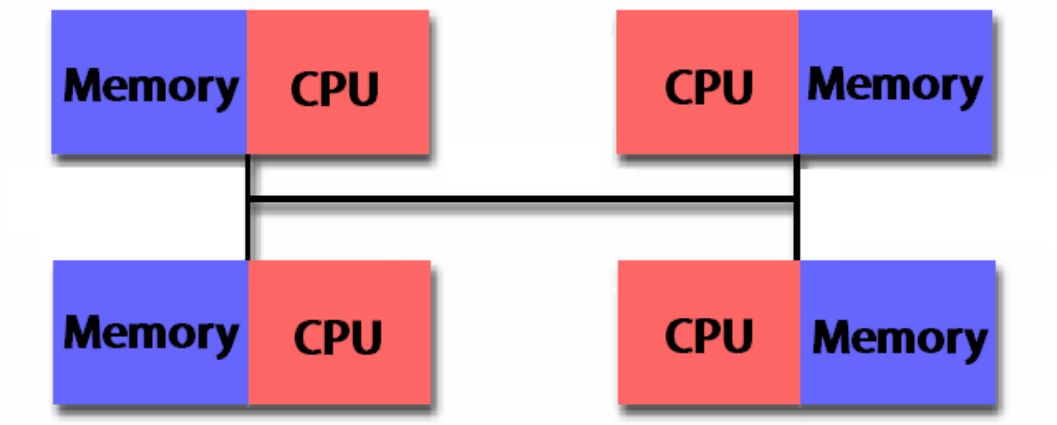
Attainability of the Bounds

With a batch size of 1.2 million, 3 input channels,
64 output channels, a 224 by 224 image, a 7 by 7 filter,
and strides of 2, we have communication volume as is shown:



Parallel Architectures

- Machine learning computations are often done in architectures with multiple parallel processors.
- We consider P parallel processors in a distributed architecture with M words of memory available to each processor.
- We'd like to minimize the number of words communicated between pairs of processors during a computation.



Memory-Dependent Parallel Lower Bounds

We derive a memory-dependent communication lower bound for direct convolution on parallel architectures:

Theorem

The number of words X communicated by a CNN which does G operations, uses $R \times S$ filters, strides of σ_w, σ_h, and runs on a distributed architecture with P processors with memory size M is

$$X \geq \max\left\{\frac{9G}{4PM} - M, \frac{2G\sqrt{\sigma_w\sigma_h}}{P\sqrt{RSM}} - 2M\right\}$$

Only nontrivial for $M \leq \max\left\{\frac{3\sqrt{G}}{2\sqrt{P}}, \frac{G^{2/3}(\sigma_w\sigma_h)^{1/3}}{P^{2/3}(RS)^{1/3}}\right\}$

Memory-Independent Parallel Lower Bounds

We derive a memory-independent communication lower bound for direct convolution on parallel architectures.

These bounds manage to dominate the memory-dependent bounds for certain large memory sizes.

Theorem

The number of words X communicated by a CNN which does G operations, uses $R \times S$ filters, strides of σ_w, σ_h, has largest array of size A, and is load-balanced on a distributed architecture with P processors with memory size M is

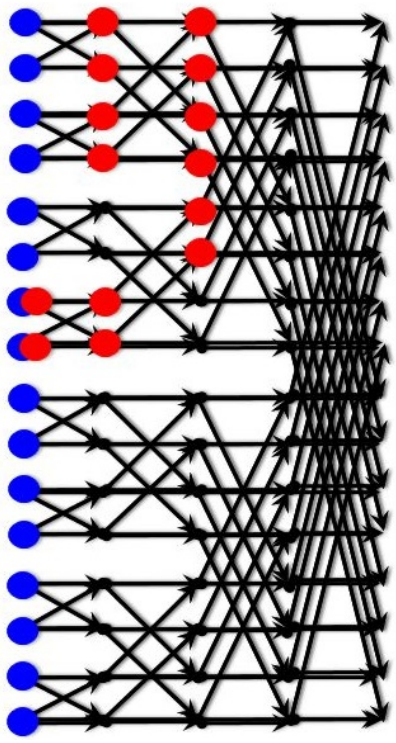
$$X \geq \max \left\{ \frac{\sqrt{G}}{\sqrt{P}} - \frac{A}{P}, \min \left\{ \frac{\sigma_w \sigma_h}{RS}, 1 \right\} \frac{G^{2/3}}{P^{2/3}} - \frac{A}{P} \right\}$$

Red-Blue Pebble Game

Another technique for proving communication lower bounds:

Pebbling game on a directed graph.

- Nodes represent operations.
- Arrows represent dependencies.
- Pebbles represent locations.
- Communication = how many red/blue switches



Partition Method

Let V be the vertices of the given graph, and assume no recomputation. We turn our problem into a partitioning problem:

1. Split code into L segments, each one doing M communication.
2. $V_i :=$ vertices computed (or inputed) during segment C_i
3. Inputs$(V_i) \leq 2M$ by I/O and memory bounds.
4. Output$(V_i) \leq 2M$ because each output vertex ends with a pebble.

Theorem (Hong and Kung, 1981)

For any calculation, the communication Q satisfies

$$Q \geq M(L_{min} - 1)$$

Tighter Partitions

This method was generalized to arbitrary partition size:

Theorem (Kwasniewski et al. 2019)

Define $R(M)$ to be maximum reuse size and $T(M)$ to be minimum I/O size. For any X, communication is bounded by

$$Q \geq (X - R(M) + T(M)) \cdot (L_{min} - 1)$$

This improved the matrix multiplication lower bound:

$$Q \geq \frac{2n^3}{\sqrt{M}} + n^2 - 2M \longrightarrow Q \geq \frac{2n^3}{\sqrt{M}} + n^2$$

Optimal Matrix Multiplication

Theorem (Kwasniewski et al. 2019)

Any matrix multiplication algorithm that multiplies matrices of sizes $m \times k$ and $k \times n$ by performing mnk operations requires at least $\frac{2mnk}{\sqrt{M}} + mn$ communications for the serial case or

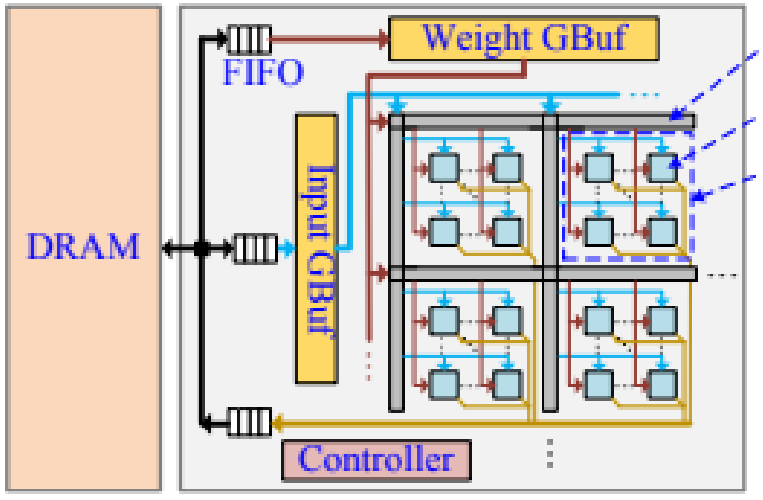
$$\min \left\{ \frac{2mnk}{P\sqrt{M}} + M, 3\left(\frac{mnk}{P}\right)^{\frac{2}{3}} \right\}$$

for the parallel case with P processors, assuming that each local memory size $M \geq \frac{mn+mk+nk}{P}$

They provide an algorithm, COSMA, that is very close to meeting these bounds.

Further Work

- Closing the gap between current CNN algorithms and lower bounds.
- Can we get a slightly tighter bound with RBPG?
- Attainability of lower bounds in the parallel case.
- Efficient hardware: Designing accelerators for CNNs (CHW'20).
- What about a multi-level memory model?



Selected References

- Online collection of papers: bebop.cs.berkeley.edu
- Survey: J. Demmel et al. 2014, *Communication lower bounds and optimal algorithms for numerical linear algebra*.
- Lower bounds using HBL: M. Christ, J. Demmel et al. 2013, *Communication lower bounds and optimal algorithms for programs that reference arrays*.
- CNN lower bounds: J. Demmel, G. Dinh 2018, *Communication-optimal convolutional neural nets*.
- Pebbling techniques: J. Hong, H. Kung 1981, *I/O complexity: The red-blue pebble game*.
- Communication-optimal matmul: T. Hoefler, G. Kwasniewski et al. 2019, *Red-blue pebbling revisited: Near optimal parallel matrix-matrix multiplication*.