Profiler-Gated Fusion of V:N:M Sparse Tensor-Core Kernels for By-Construction Non-Regressive CNN Speedups

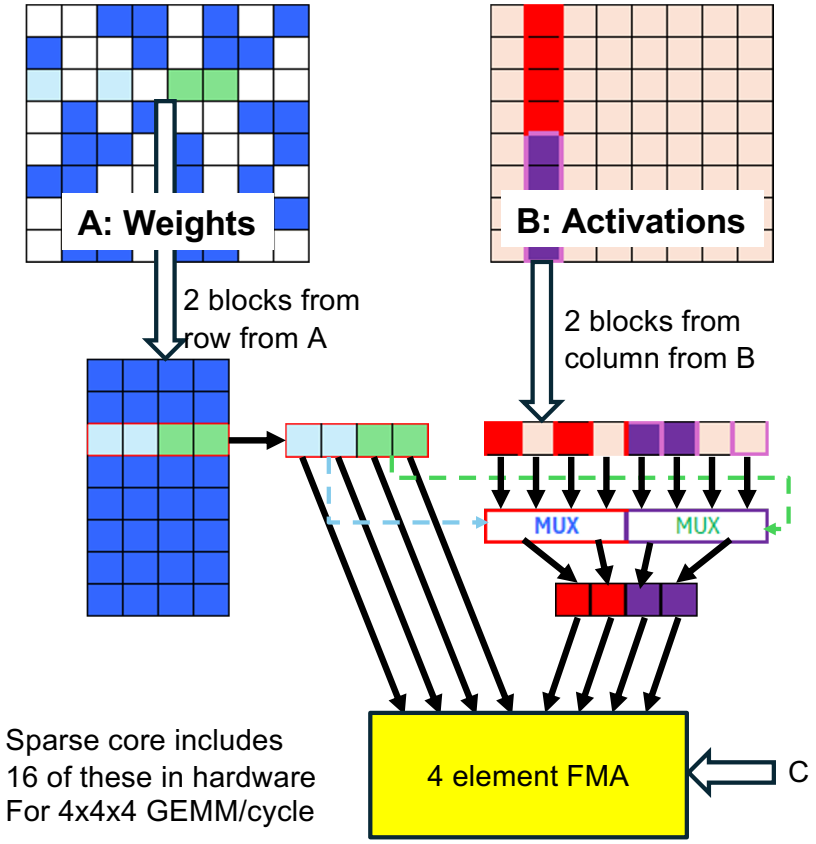
Jayden Vap, Reem Shehayib, Peter Kogge

Univ. of Notre Dame

Overview

- Modern techniques can “prune” 90+% of a net’s weights
- But on dense tensor cores ***no time savings***
- Emerging: *Sparse tensor cores* support “**2:4 sparsity**”
- Published V:N:M (“Venom”) algorithm expands this to arbitrary N:M
  - With uncertain gains for non linear/convolutional layers
- This paper:
  - Fused im2col-sparse-GEMM kernel for *convolutional layers up 2:16*
  - Accompanying profiler to decide when sparse version wins
  - Re-pruning procedure

2:4 Sparse Tensor Core



- Dense Tensor Core: A*B+C
  - A,B,C all 4x4
  - Core takes 1 block of 4x4 numbers each
- Assume A pruned so only 2 non-zeros for every 4 weights
- Sparse Tensor Core: Array of
  - One row block of 4 from compacted A
  - Two column blocks of 4 from dense B
    - A's metadata drives muxes
    - Two activations passed on from each block
- 4x4x4 core does eqvt of 4x8x4 tile on sparse data

VENOM V:N:M Format and SPATHA Package

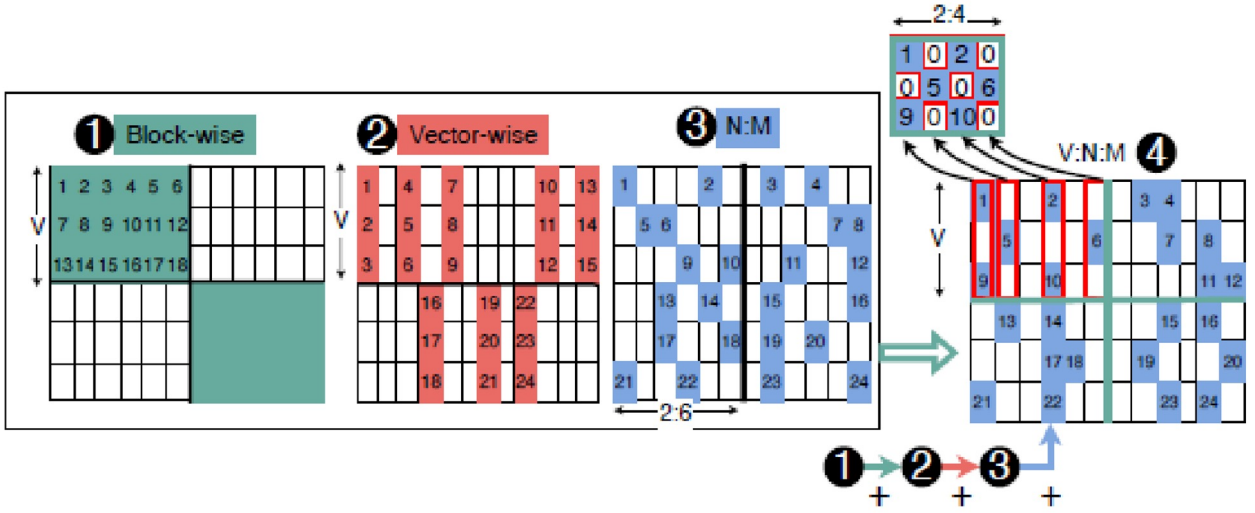


Fig. 2 from: R. L. Castro, A. Ivanov, D. Andrade, T. Ben-Nun, B. B. Fraguela, and T. Hoefler. 2023. VENOM: A Vectorized N:M Format for Unleashing the Power of Sparse Tensor Cores. In Proc. Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC).

Goal: reduce matrix to arbitrary N:M but suitable for 2:4 cores

1. Partition original matrix into VxM blocks
2. Select 4 "most significant" columns of each block
3. In each 4 consecutive elements, select 2 "most meaningful"

SPATHA package provides matching SpMM that run on 2:4 cores

VENOM Gains Due to Sparsity

Speedups over col. 2

GEMM ($M \times K \times N$)	dense (ms)	cuSPARSELt 2:4 ($\times$ dense)	Spatha 2:16 ($\times$ dense)	Spatha / cuSPARSELt
$64 \times 576 \times 131072$	0.279	1.00	8.65	8.6
$128 \times 576 \times 131072$	0.335	1.13	2.03	1.8
$128 \times 1152 \times 32768$	0.154	0.97	5.18	5.4
$256 \times 1152 \times 32768$	0.251	1.53	3.88	2.5
$256 \times 2304 \times 8192$	0.147	1.76	5.32	3.0
$512 \times 2304 \times 8192$	0.269	2.28	5.45	2.4
$512 \times 4608 \times 2048$	0.134	1.62	4.94	3.0
geomean	—	1.41	**4.70**	**3.3**

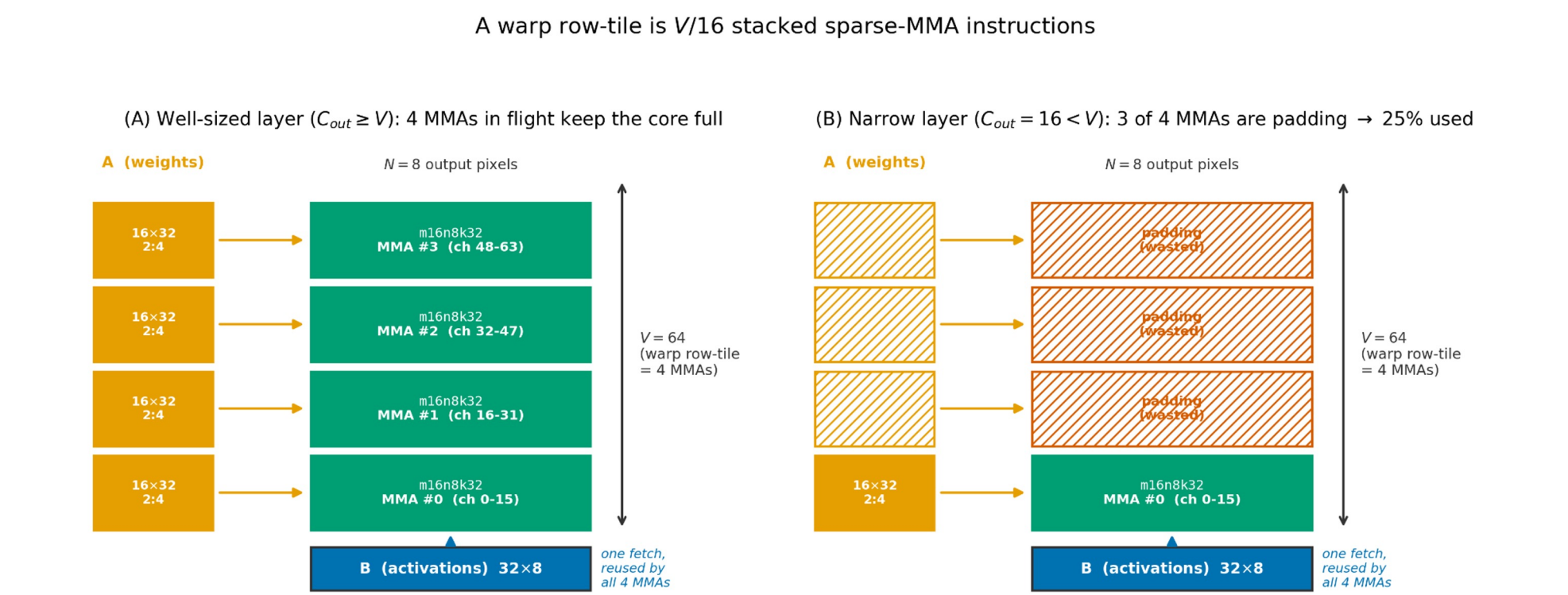


Driven by im2col (unfold),
Then 2:4 SPMM

Issues with use in Convolution Kernels

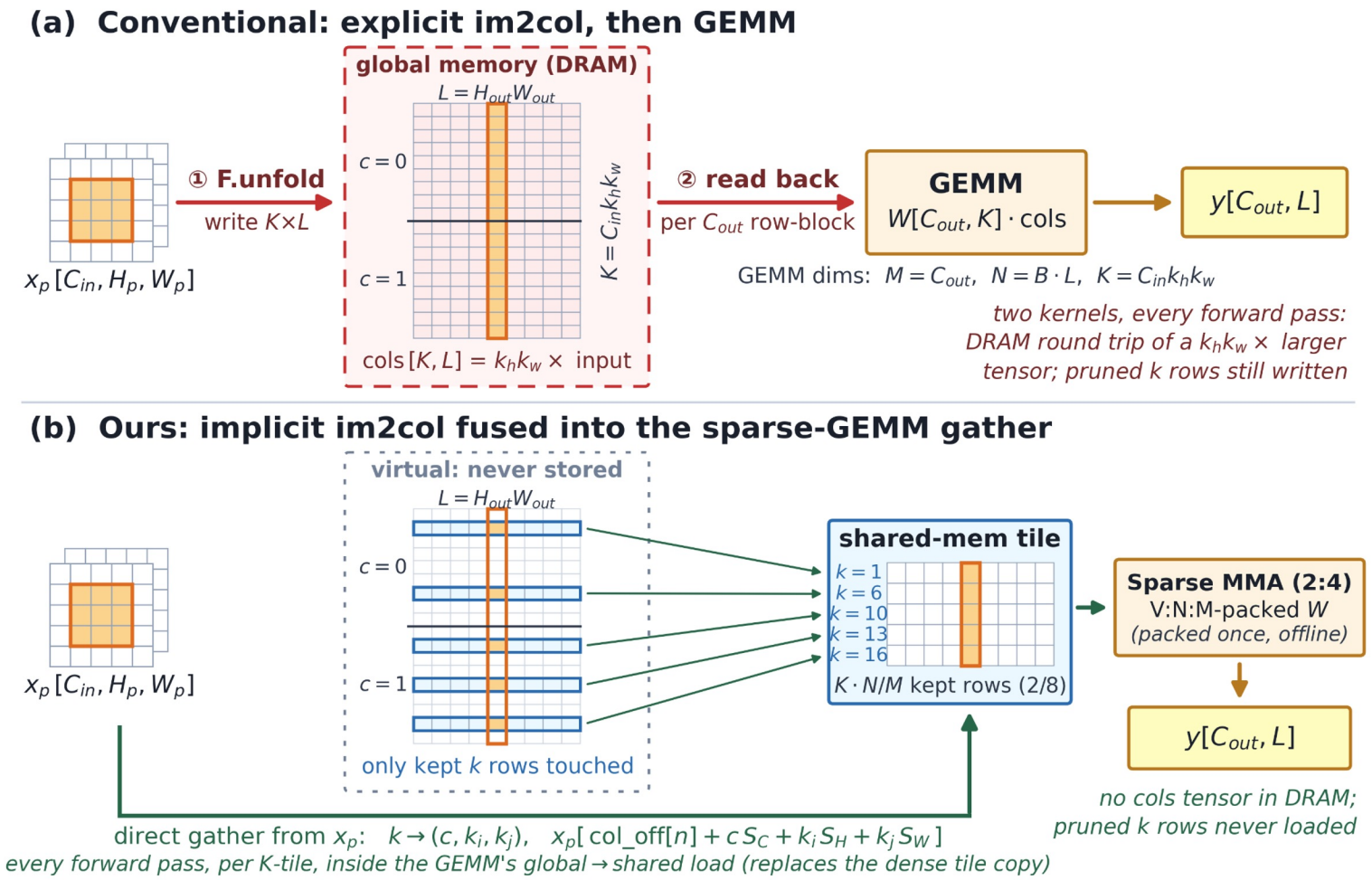
- Naïve im2col lowering converts convolutions into GEMM form
- ***But adds expensive memory-bound pass***
- Sparse MMA instruction (mma.sp.m16n8k32) on Ampere/Ada
  - Has output row granularity of 16, can stack block MMAs
  - But below size 32 row tiles are unrealizable or occupancy bound
- Output channel counts below tile height
  - Wastes rows to padding
  - Too small activation sizes don’t saturate sparse cores

Network Dependent Issues

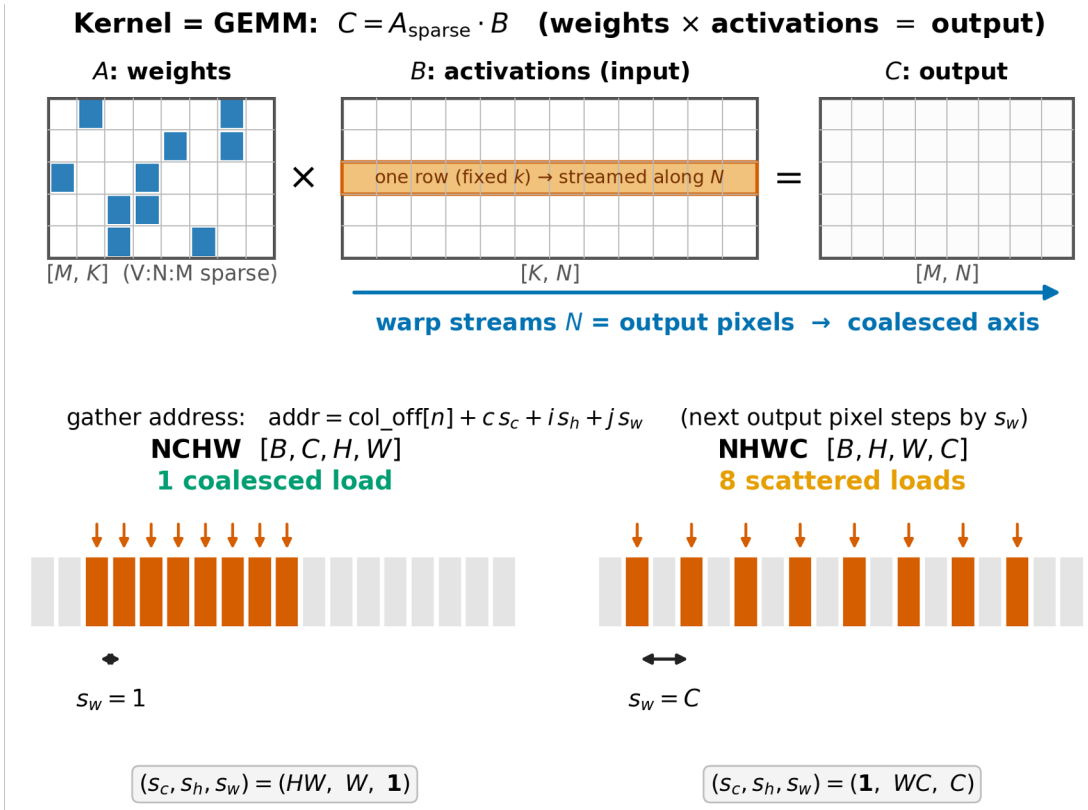


One `m16n8k32` MMA emits a 16×8 tile; stacking $V/16$ of them per warp (sharing one B fetch) hides load latency. $V = 16$ is a single idle MMA; $V = 32$ is occupancy-bound; $V = 64$ is the smallest tile that beats dense.

im2col Fusion Visualization

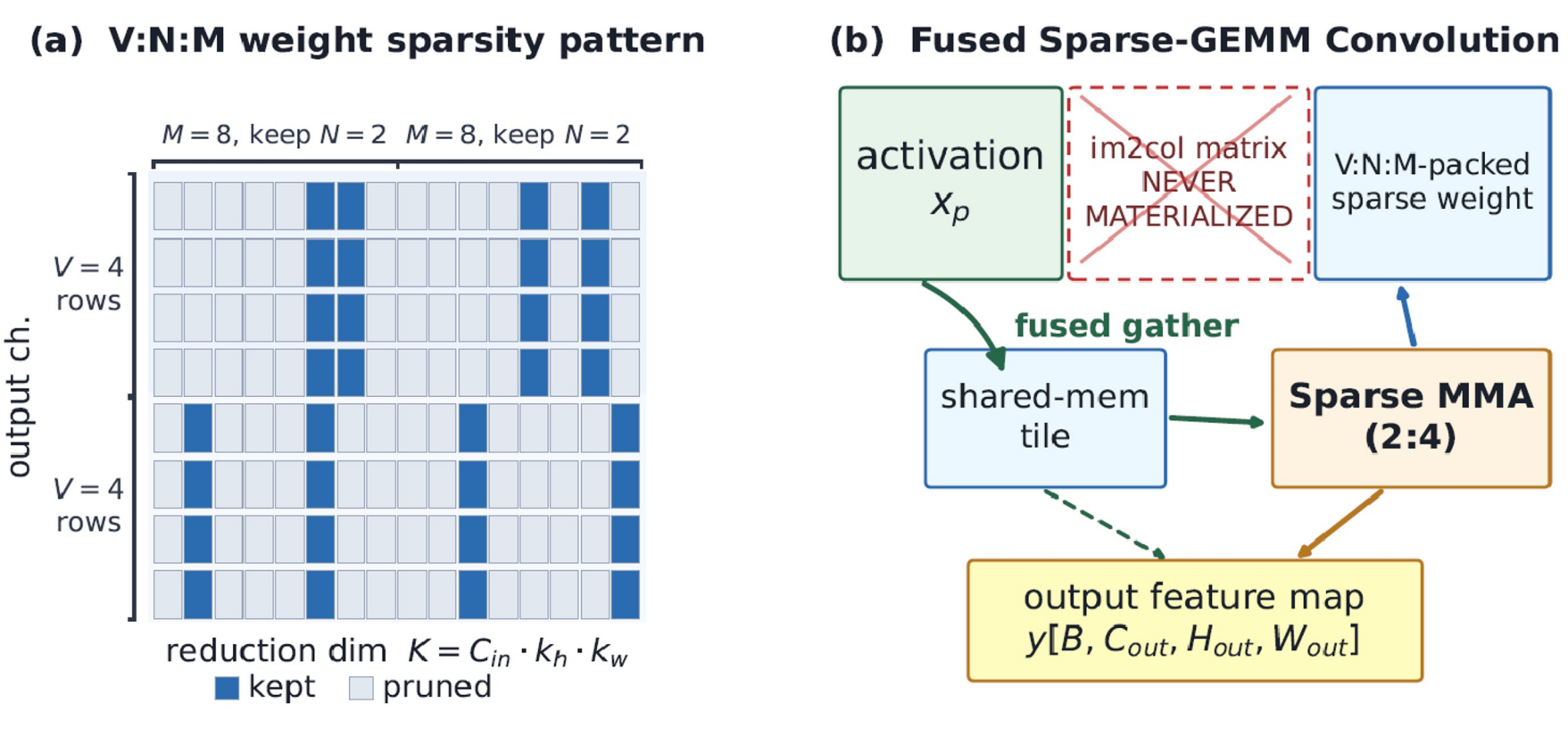


Channel First For Fused im2col



- Conventional GEMM orders computations (and thus memory) as Channel last (NHWC)
- This leads to scattered loads of data that follows same sparsity
- Switching to Channel first (NCHW) groups these

Fused im2col-Spares-GEMM Kernel



Profiler: determine when actual benefits

For each convolutional layer in the pruned network:

1. Measure wall-clock latency of fused im2col-sparse-GEMM kernel on representative input dimension sizes.
2. Measure wall-clock latency of the vendor cuDNN channel last convolution on the same input.
3. Assign layer to the sparse kernel iff difference is less than some fixed margin ε
    1. Otherwise assign layer to the dense channel last kernel.

Experimental Setup

- NVIDIA RTX 4080, FP16, batch size = 128
- Comparisons to cuDNN or cuSPARSElt 2:4 GEMM
- Network/datasets:
  - VGG-19 on CIFAR-10, CIFAR-100, and Tiny-ImageNet
  - ResNet-32 on CIFAR-10, CIFAR-100, and Tiny-ImageNet
  - ResNet-56 on CIFAR-10
  - DenseNet-40 on CIFAR-10
  - LeNet-5 on MNIST

Convolution Only vs Native 2:4 cuSPARSELt

Speedups over col. 2

Conv layer ($C_{in} \rightarrow C_{out}, r^2$)	dense (ms)	cuSPARSELt (×dense)	fused V:N:M (×dense)	fused V:N:M vs cuSPARSELt (×)
$64 \rightarrow 64, 32^2$	0.112	0.09	1.03	11.6
$64 \rightarrow 128, 32^2$	0.214	0.17	1.16	7.0
$128 \rightarrow 128, 16^2$	0.105	0.14	1.25	9.1
$128 \rightarrow 256, 16^2$	0.194	0.24	1.43	5.96
$256 \rightarrow 256, 8^2$	0.105	0.20	1.26	6.4
$256 \rightarrow 512, 8^2$	0.213	0.33	1.44	4.4
$512 \rightarrow 512, 4^2$	0.123	0.23	1.40	6.0
geomean	—	0.19	**1.27**	**6.9**



Driven by im2col (unfold),
Then 2:4 SPMM

- Batch 128, FP16, VGG-scale layers
- Nonfused sparse significantly *slower* than dense GEMM
- Fused provides real speedups over dense GEMM
- Fused significantly faster on same problems as cuSPARSELt alone

Whole Network Time vs cuDNN Baseline

Model	Input	Fused / total conv layers	GPU × (measured)
VGG-19 (CIFAR-10)	32^2	8/16	1.13
VGG-19 (Tiny-ImageNet)	64^2	9/16	1.09
VGG-19 (CIFAR-100)	32^2	6/16	1.08
ResNet-32 (CIFAR-10)	32^2	0/33	$\sim$1.00
ResNet-32 (CIFAR-100)	32^2	0/33	$\sim$1.00
ResNet-32 (Tiny-ImageNet)	64^2	0/33	$\sim$1.00
ResNet-56 (CIFAR-10)	32^2	0/57	$\sim$1.00
DenseNet-40 (CIFAR-10)	32^2	0/39	$\sim$1.00
LeNet-5 (MNIST)	28^2	0/2	$\sim$1.00

Wide, purely feed-forward

Channel widths force smallest sparse-MMA tile

Conclusions

- Reported pruning sparsity does not provide measured speedup
  - Cost of materializing reordered matrices can be significant
- Closing gap requires specializing kernels to layer structure
  - Convolutional layers particularly sensitive
- Fusing im2col with sparse GMM can provide benefit
- On RTX 4080 1.13X real speedup for wide-channel CNNs
- Next steps:
  - Extend to strided/dialated convolutions
  - Replace off-line profiler with runtime adaptive cost model

Backup: SPATHA Performance

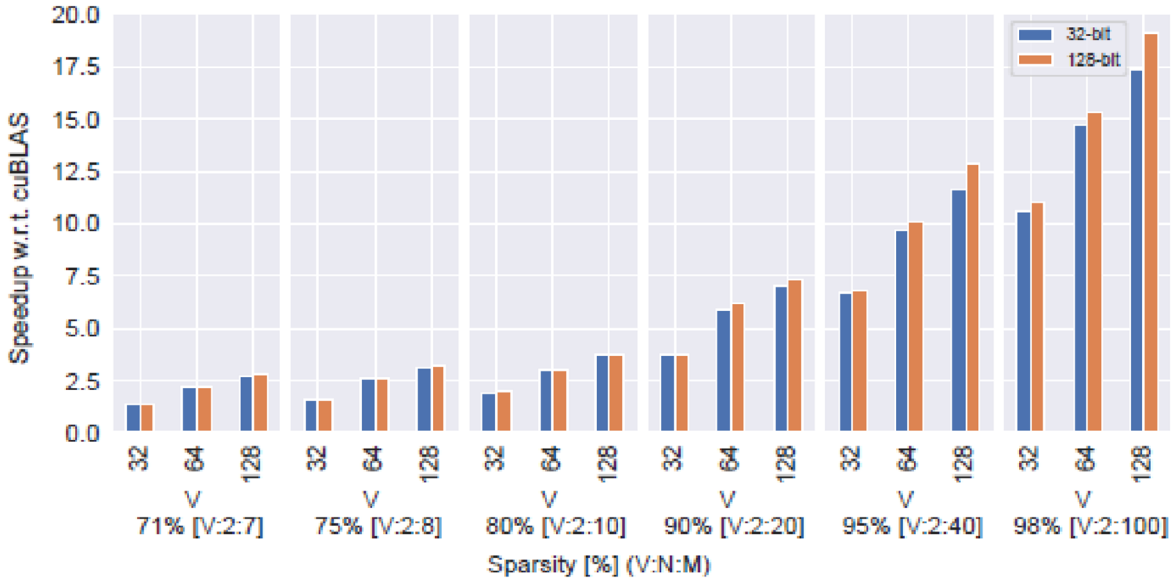


Fig. 10 from: R. L. Castro, A. Ivanov, D. Andrade, T. Ben-Nun, B. B. Fraguela, and T. Hoefler. 2023. VENOM: A Vectorized N:M Format for Unleashing the Power of Sparse Tensor Cores. In Proc. Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC).

- Experiment: 1024 x K x 4096, with V = 128
- K values: 32, 64, or 128
- Blue is 32 bit SMEM stores; orange is 128bit