A Novel Inference Algorithm for Large Sparse Neural Network using Task Graph Parallelism

SNIG
Inference Engine

Dian-Lun Lin Tsung-Wei Huang

University of Utah

MIT/IEEE/Amazon HPEC Graph Challenge

https://github.com/dian-lun-lin/SNIG

Challenges: Performance and Memory Constraint

> 4 billion nonzero parameters

Neurons/Layers	120	480	1920	Bias	Size	Image Nonzeros
1024	3,932,160	15,728,640	62,914,560	-0.30	1.25 GB	6,374,505
4096	15,728,640	62,914,560	251,658,240	-0.35	5.40 GB	25,019,051
16384	62,914,560	251,658,240	1,006,632,960	-0.40	22.70 GB	98,858,913
65536	251,658,240	1,006,632,960	$\boxed{4{,}026{,}531{,}840}$	-0.45	94.70 GB	392,191,985

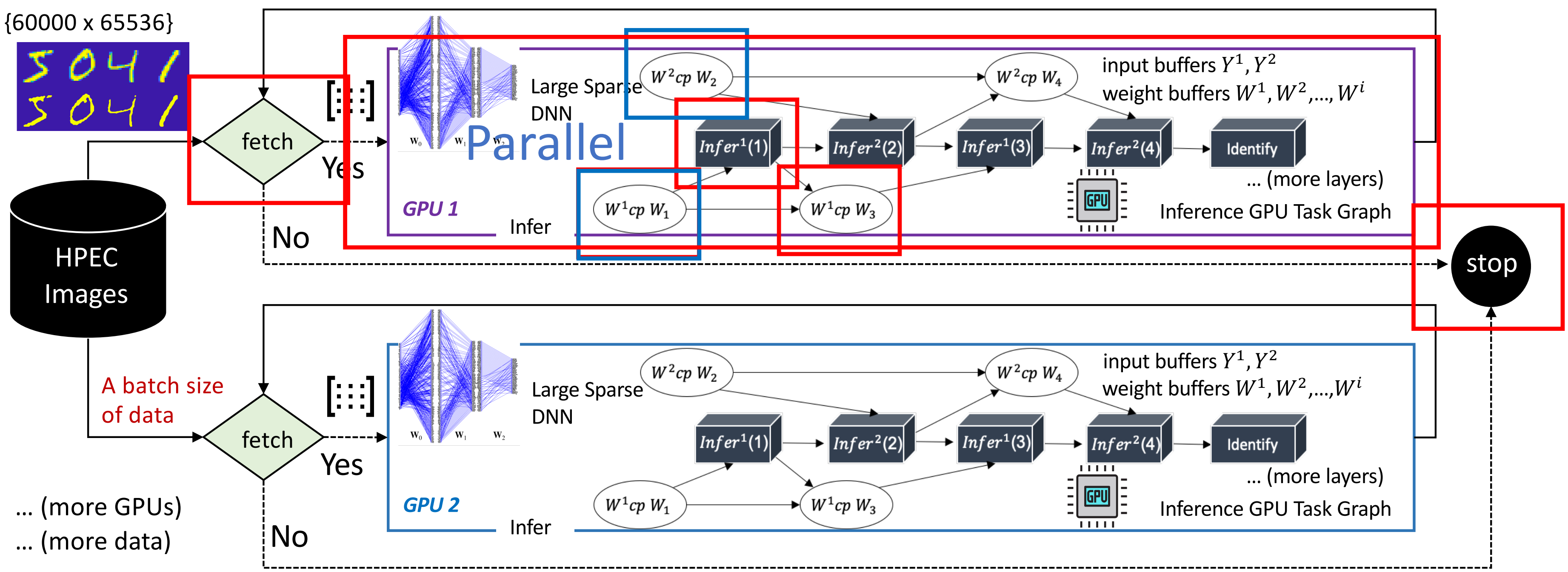
Previous year's champion (Bisson and Fatica, IEEE HPEC 2020, **BF** for short)

- Require the entire input data to sit in the GPUs under unified memory addressing
- Synchronize computation at each layer for load balancing

Pipeline parallelism (Yanping Huang et al., NIPS 2019, **GPipe** for short)

- Require the entire model to sit in GPUs
- Synchronize computation at each pipeline stage

Key Solution: Task Graph Parallelism (CUDA Graph)



End-to-end parallelism
Extensible to arbitrary LSNNs and input data under different numbers of GPUs

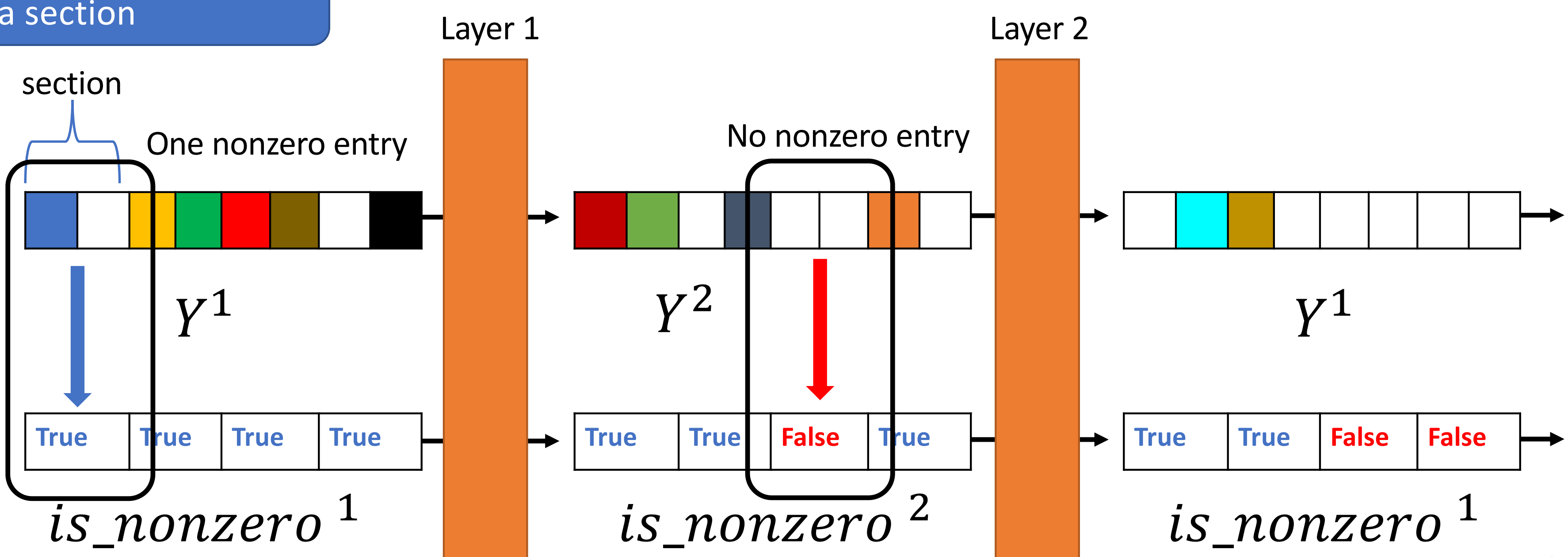
Reduce Memory Usage

During the inference, each GPU allocates

- two input buffers Y^1 , Y^2
- two Boolean arrays $is\_nonzero^1$, $is\_nonzero^2$

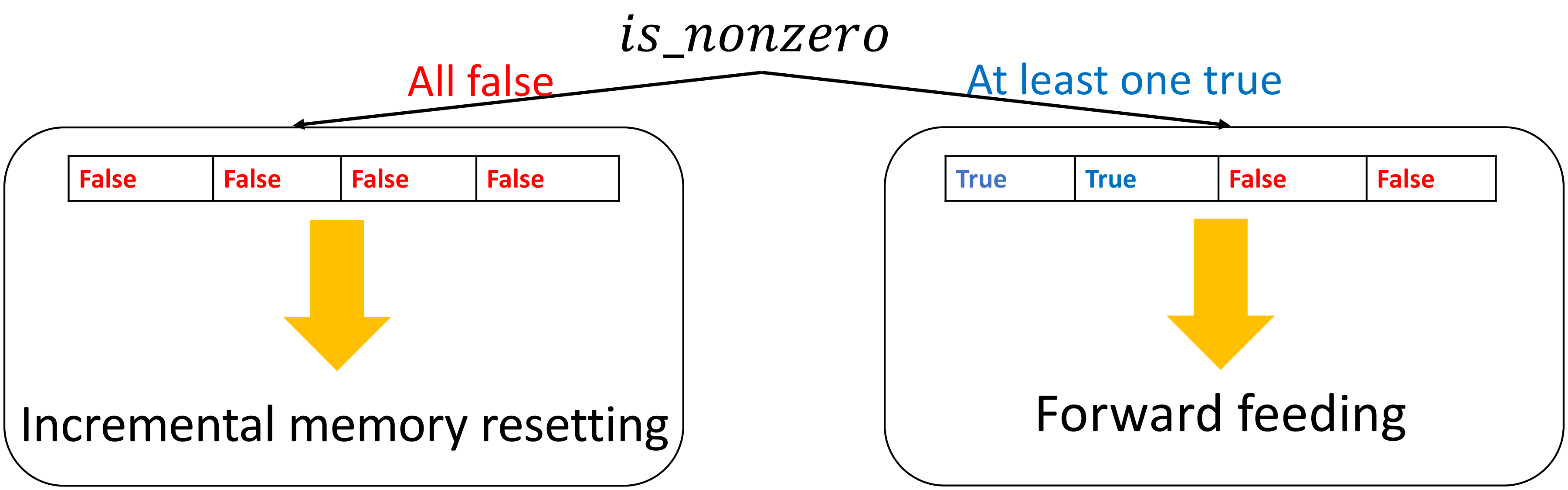
- Layer 1 takes Y^1 as input and outputs to the Y^2; Layer 2 takes Y^2 as input and outputs to the Y^1
- White entries represent zeros

We group several entries into a section

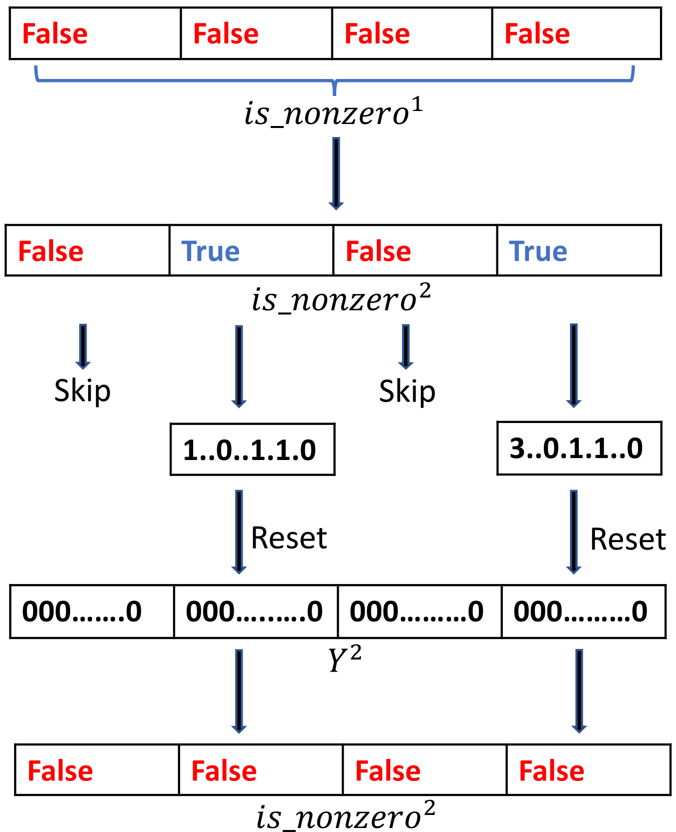


Two Key Components in Our Kernel

Our Goal: avoid unwanted computation on zero entries



Component #1: Incremental Memory Resetting

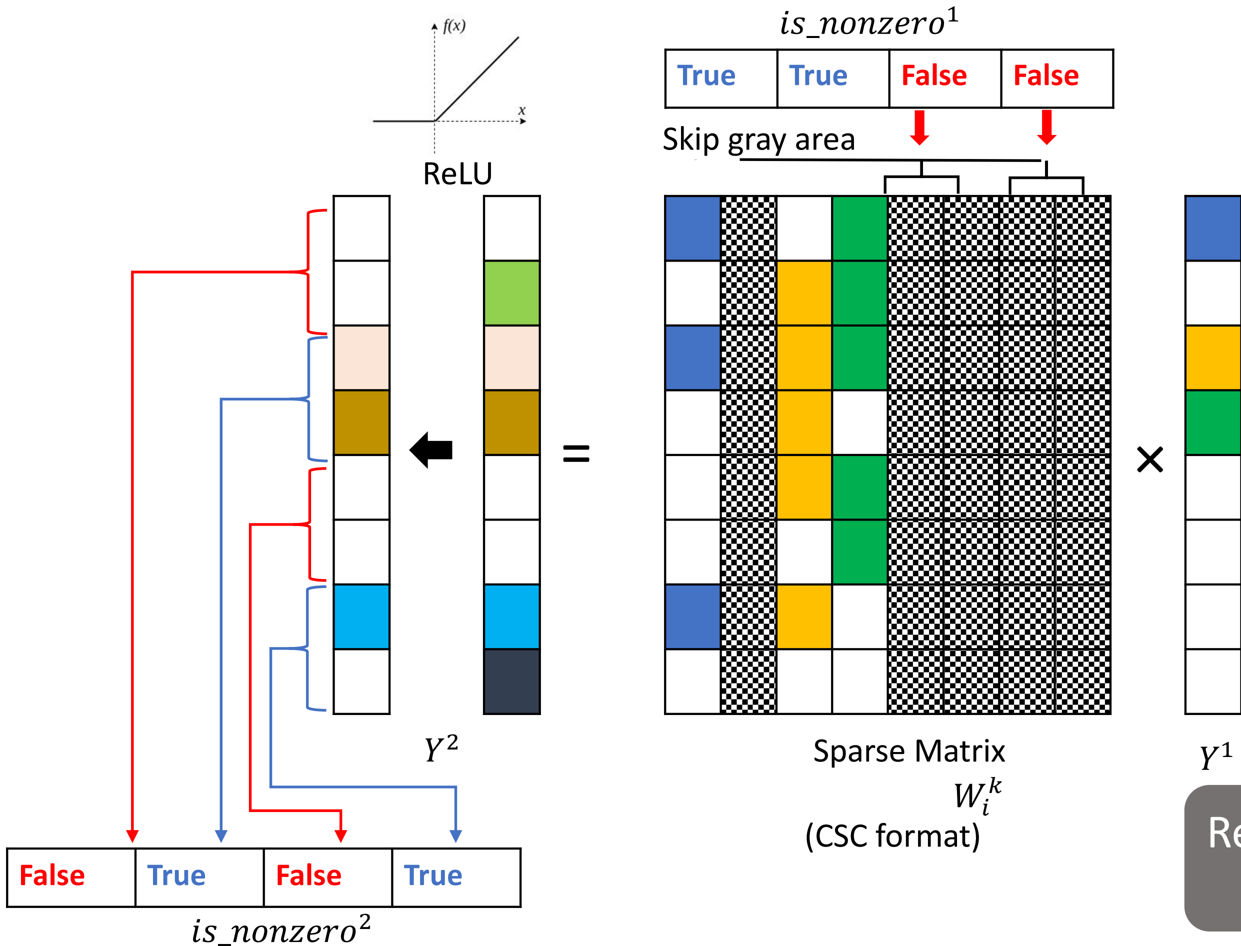


Observation:
If inputs are all zeros, outputs are still all zeros

Contains at least one nonzero entry

Results:
1. Skip lots of sections
2. Resetting rather than computing

Component #2: Forward Feeding



Our Goal: avoid unwanted computation on zero entries

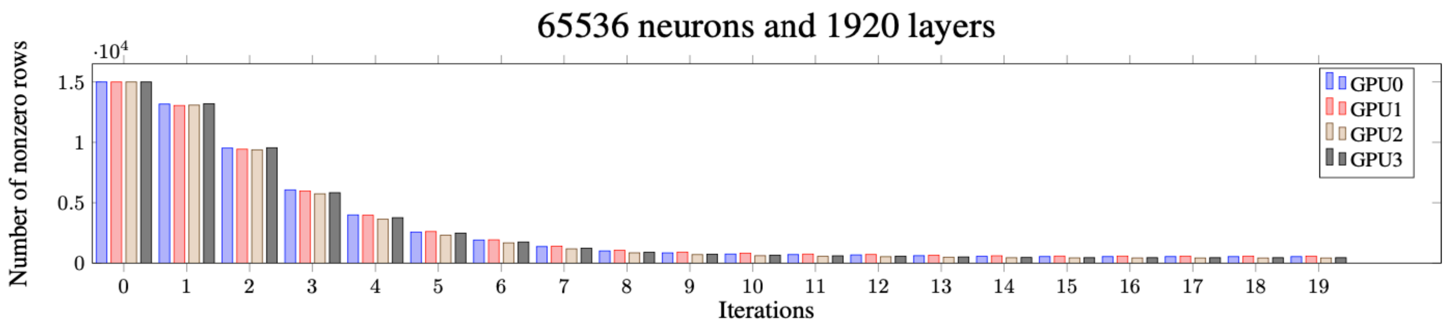
Inspect $is\_nonzero^1$ array first

Results:
Skip lots of sections

Experimental Results

- Baseline
  - BF (Bisson and Fatica, IEEE HPEC 2019) without NVLink (Doesn’t affect performance)

https://github.com/dian-lun-lin/SNIG



  - GPipe (Yanping Huang et al., NIPS 2019)
- Software
  - Ubuntu Linux 5.0.0-21-generic x86 64-bit machine
  - C++14 and CUDA NVCC 10.1 (CUDA Graph Library)
- Hardware
  - 40 Intel Xeon Gold 6138 CPU cores at 2.00 GHz
  - 4 GeForce RTX 2080 Ti GPUs with 11 GB memory
  - 256 GB RAM

Our implementation of BF method:
- Only partition at the beginning
- The number of non-empty rows per iteration remains balanced at each GPU

Overall runtime results of SNIG, BF, and GPipe

Reported in seconds

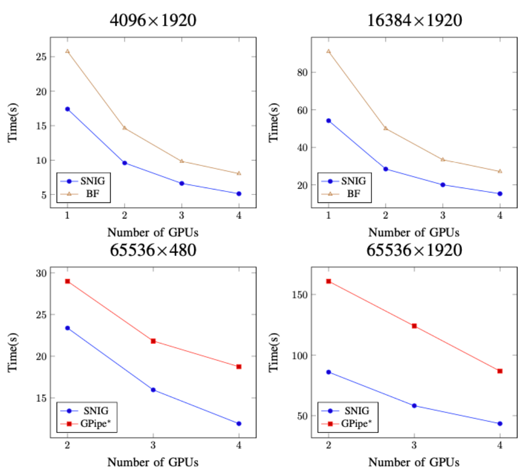
Bold text represents the best solution

		Number of GPUs										
		1		2			3			4		
Neurons	Layers	BF	SNIG	BF	GPipe*	SNIG	BF	GPipe*	SNIG	BF	GPipe*	SNIG
1024	120	**0.682**	0.799	0.409	**0.400**	0.518	**0.310**	0.339	0.342	0.272	0.307	**0.189**
	480	1.975	**1.609**	1.178	**0.928**	1.019	0.889	0.741	**0.700**	0.848	0.636	**0.476**
	1920	7.197	**5.252**	4.428	**3.178**	3.187	3.330	2.396	**2.291**	3.093	2.012	**1.748**
4096	120	2.305	**1.609**	1.265	1.010	**0.962**	0.853	0.896	**0.646**	0.681	0.810	**0.421**
	480	6.932	**4.696**	3.921	2.742	**2.695**	2.637	2.136	**1.830**	2.165	1.824	**1.367**
	1920	25.75	**17.41**	14.63	9.732	**9.584**	9.817	7.278	**6.610**	8.035	6.023	**5.121**
16384	120	8.165	**4.433**	4.283	2.925	**2.538**	2.896	2.481	**1.729**	2.328	2.241	**1.295**
	480	24.50	**14.02**	13.28	7.999	**7.683**	8.997	6.151	**5.346**	7.286	5.217	**4.041**
	1920	91.05	**54.22**	50.01	28.68	**28.39**	33.39	21.44	**19.98**	27.08	17.70	**15.24**
65536	120	524.3	**14.78**	262.5	11.41	**8.073**	11.33	10.16	**5.581**	9.136	9.643	**4.394**
	480	>1800	**43.00**	1027	28.99	**23.38**	33.23	21.82	**15.96**	26.94	18.74	**11.91**
	1920	>1800	**162.2**	>1800	160.9	**85.96**	123.2	124.0	**58.22**	98.59	86.77	**43.44**

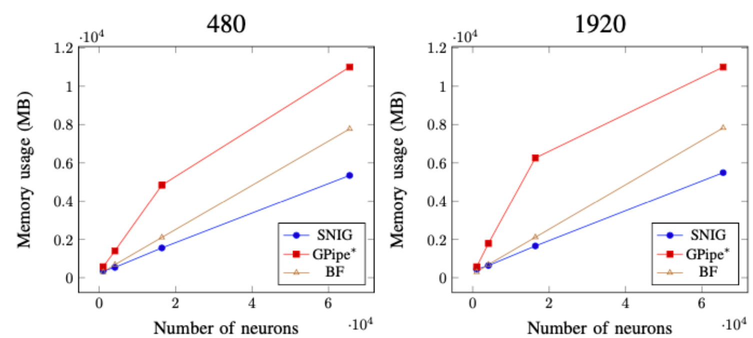
Execution Timeline & GPU/Memory Scalability

Execution Timeline (SNIG is 2x faster)

GPU Scalability



Memory Scalability



Thanks!

SNIG

Inference Engine

https://github.com/dian-lun-lin/SNIG
dian-lun.lin@utah.edu