The Future of Machine Learning is Sparse

Future is Sparse @ SC'23

CASC
Center for Applied
Scientific Computing

11/17/23

Tal Ben-Nun
Computer Scientist

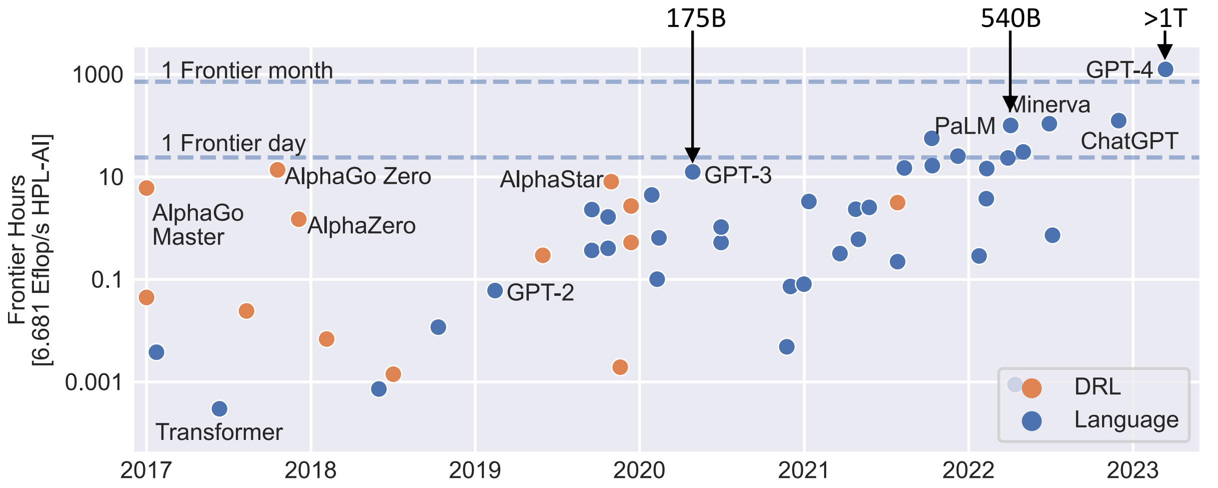
LLNL-PRES-857268

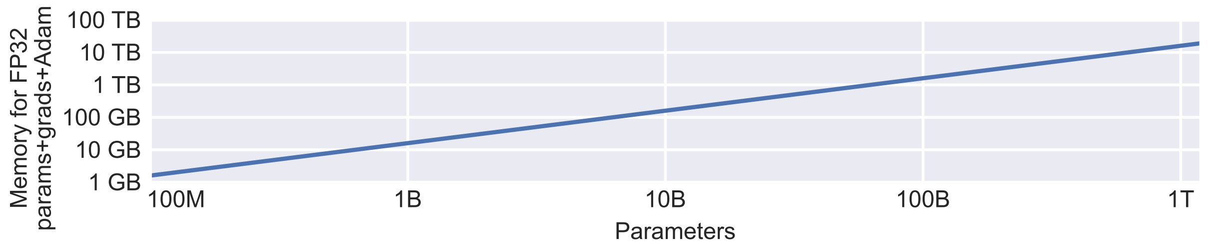
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore
National Laboratory

The large-scale era of Machine Learning

- Training time measured in Exaflop days
- Models (reportedly) exceed 1T parameters
  — and are actually better as they grow…
- Without reducing memory consumption, we will not have the capacity to expand
- Are all the parameters and features *necessary?*

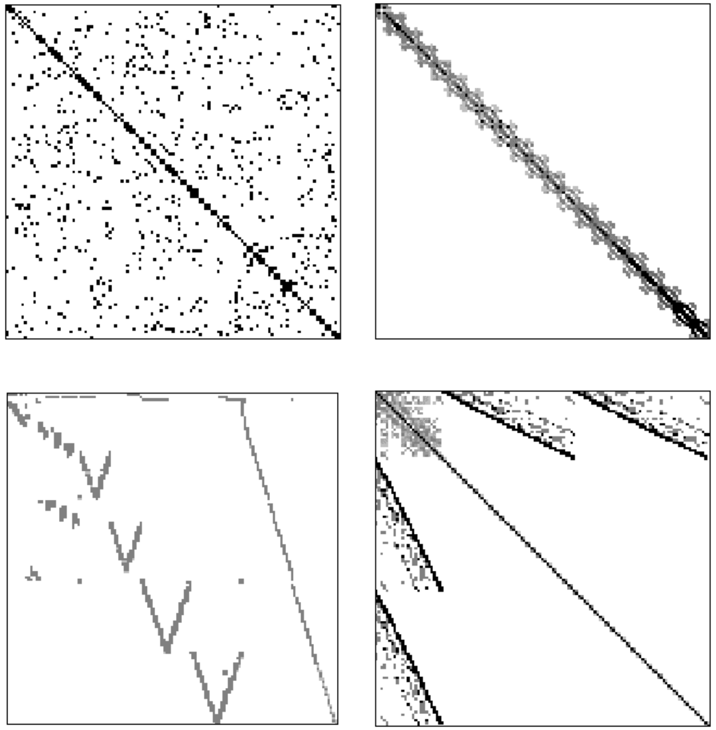




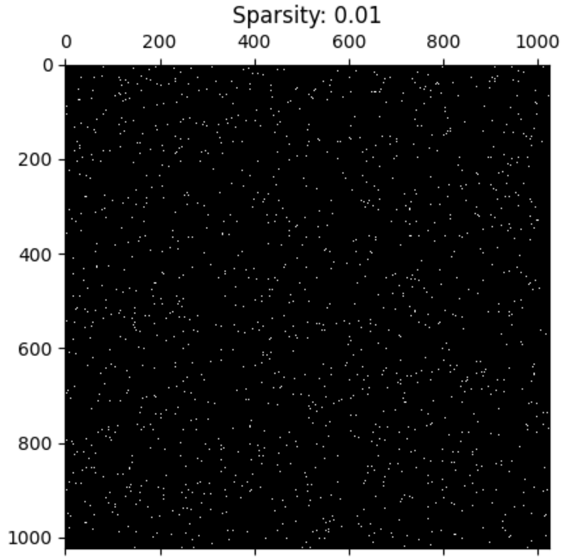
Massive GPU memory required **before** considering data + activations

Sevilla et al., “Parameter, Compute, and Data Trends in Machine Learning”, 2021

Sparsity in HPC vs. Machine Learning

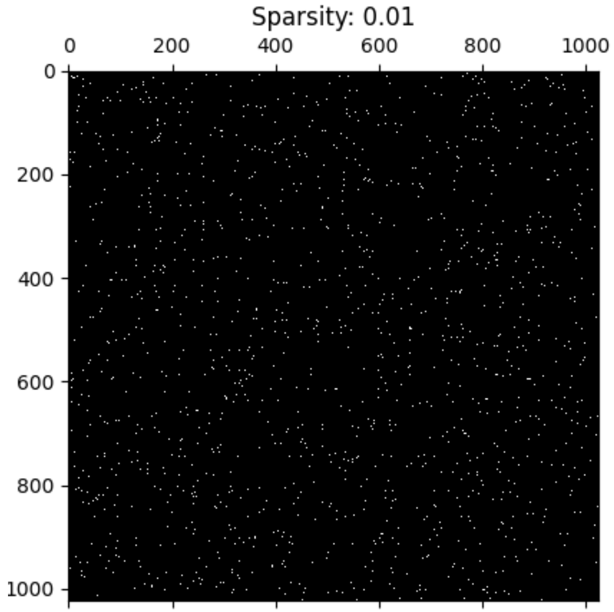


Source: SuiteSparse Matrix Collection



W_K

Source: Sparsified BERT

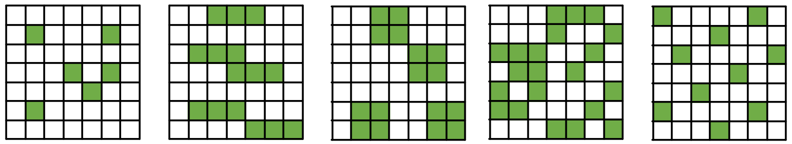


W_Q

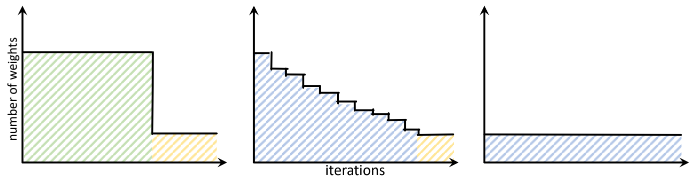
Overview

- Sparse elements in deep learning

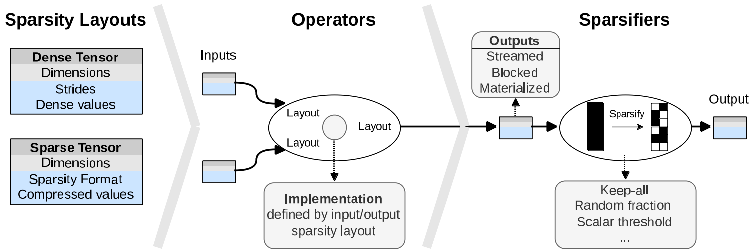
- Representations



- Scheduling strategies



- Hardware/software co-design and research tools



Primer on deep learning

$$f(\boldsymbol{w}; x)$$

Input distribution $\mathcal{X}$

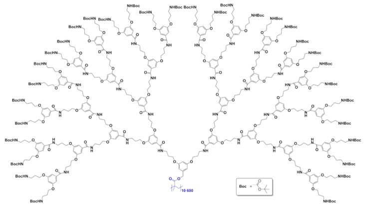
Output distribution $\mathcal{Y}$

Primer on deep learning

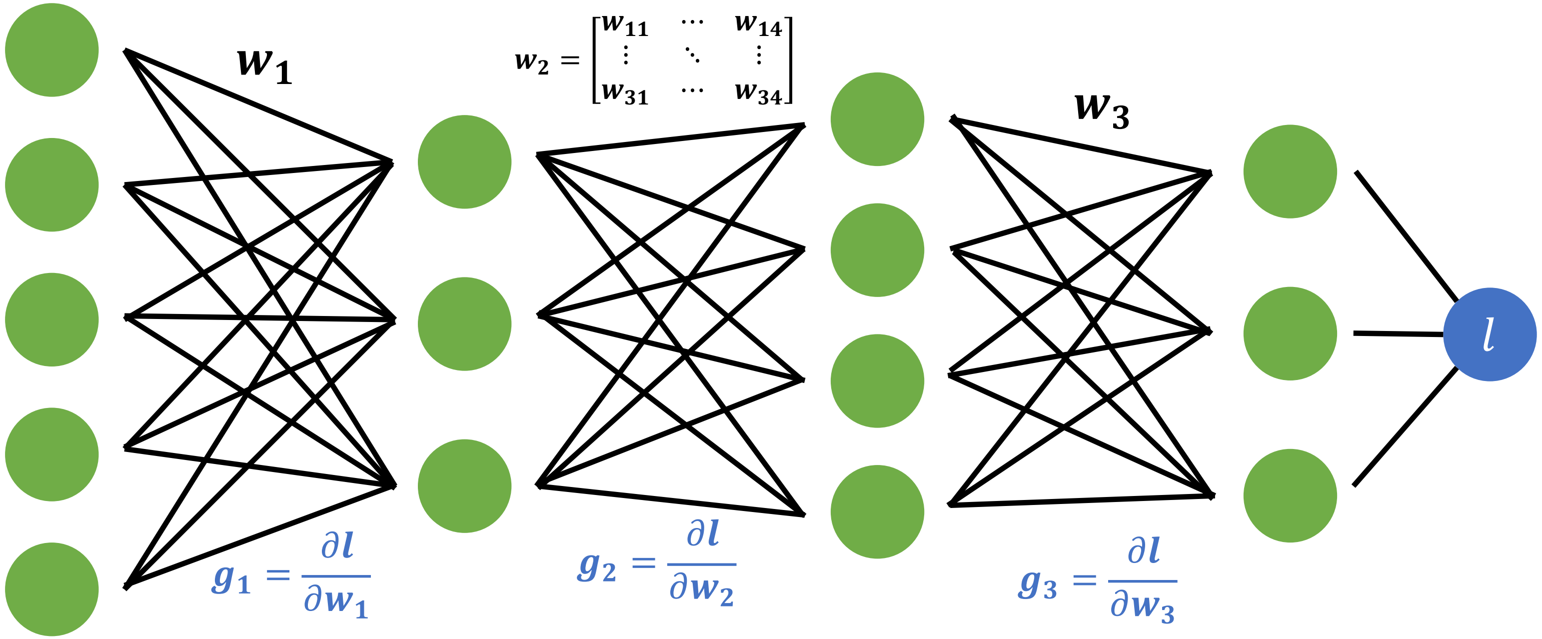
"This is a…"

Forward

"…presentation"



"Pruning a neural network"



Cat

Ω_M, σ_8, n_s

1.3 g/cc

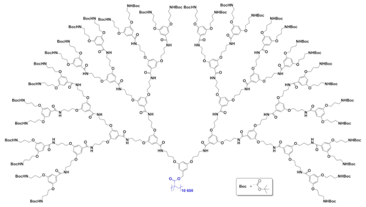
Backward

Input distribution $\mathcal{X}$

Output distribution $\mathcal{Y}$

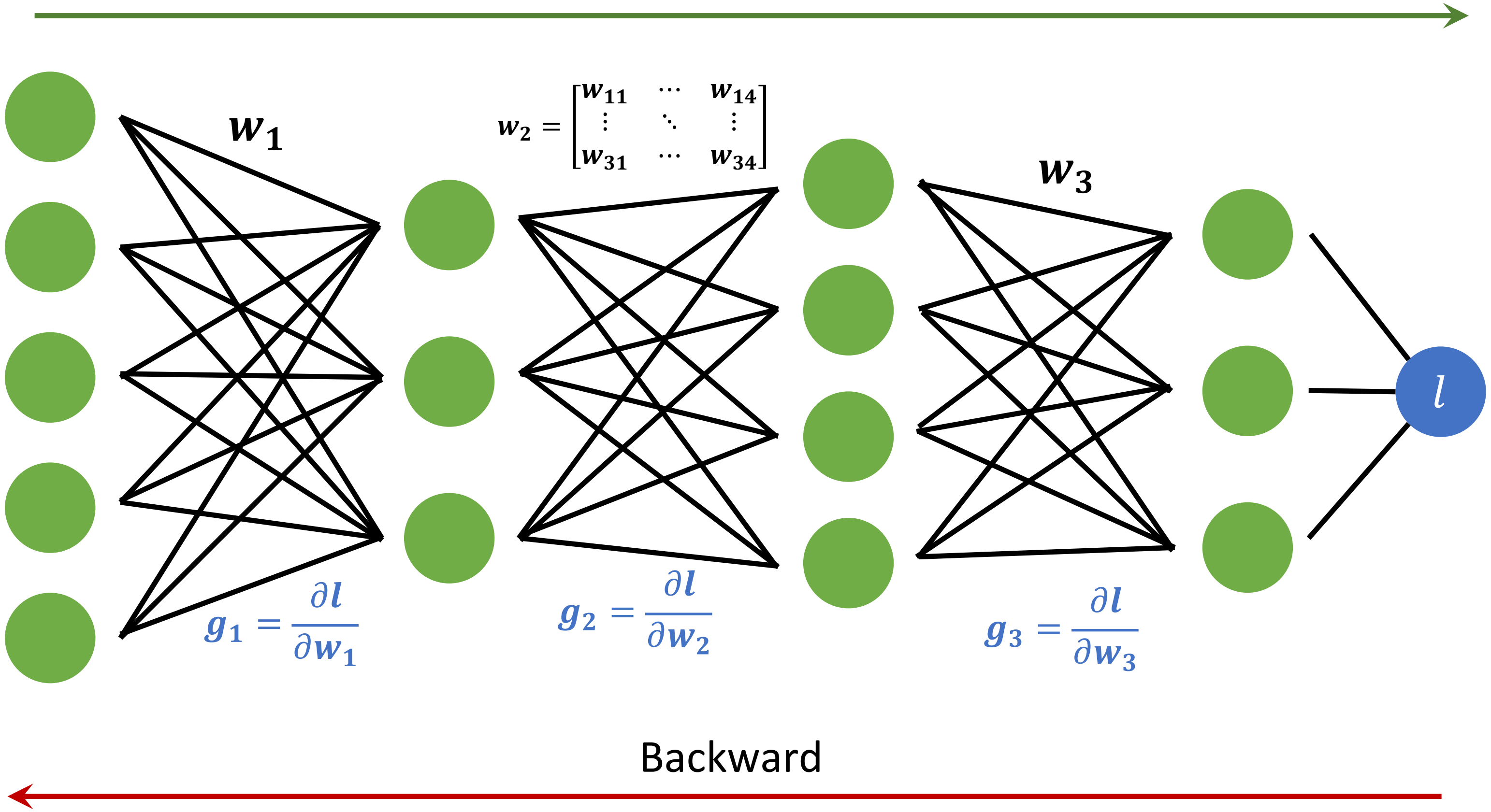
Where do we encounter sparsity?

"This is a…"



"Pruning a neural network"

Forward



Backward

"…presentation"

Cat

Ω_M, σ_8, n_s

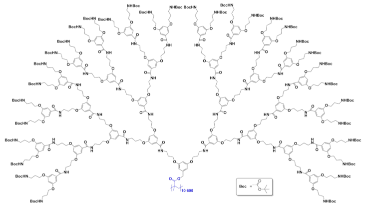
1.3 g/cc

Input distribution $\mathcal{X}$

Output distribution $\mathcal{Y}$

Where do we encounter sparsity?

"This is a…"



"Pruning a neural network"

"…presentation"

Cat

Ω_M, σ_8, n_s

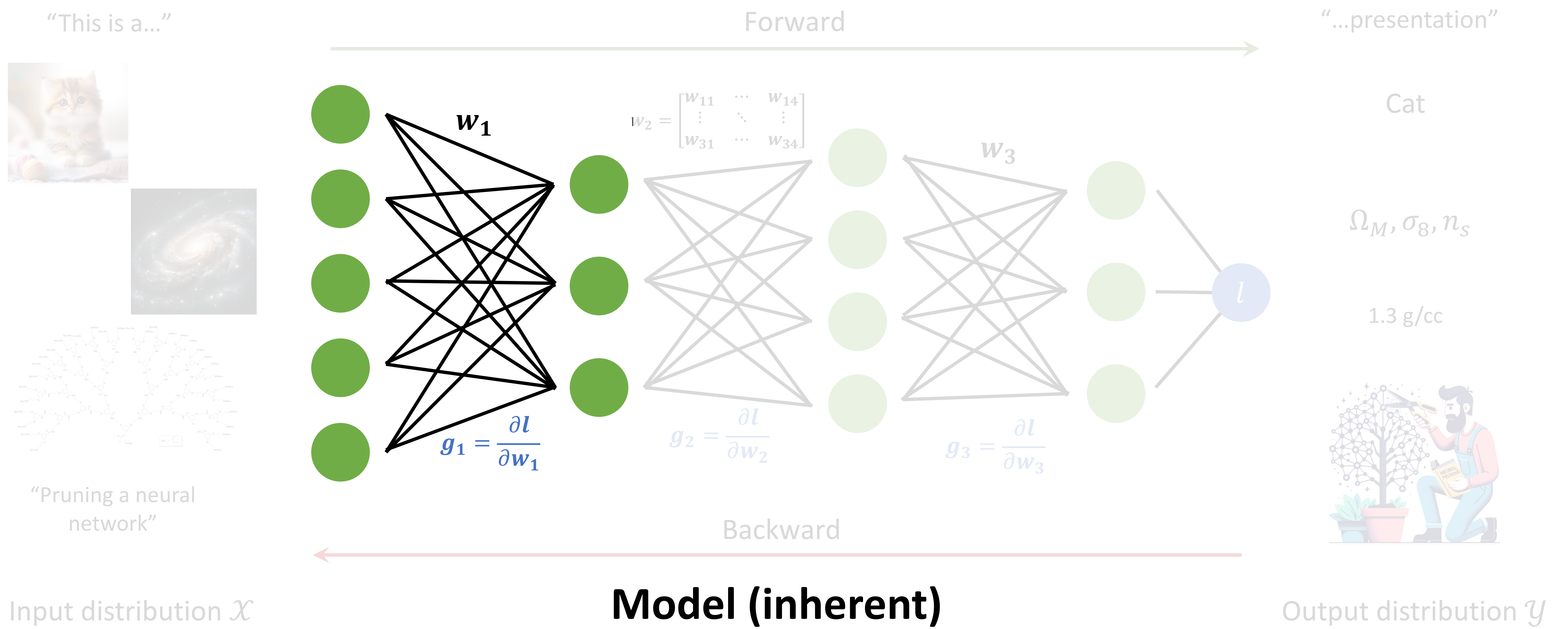
1.3 g/cc

Input distribution $\mathcal{X}$

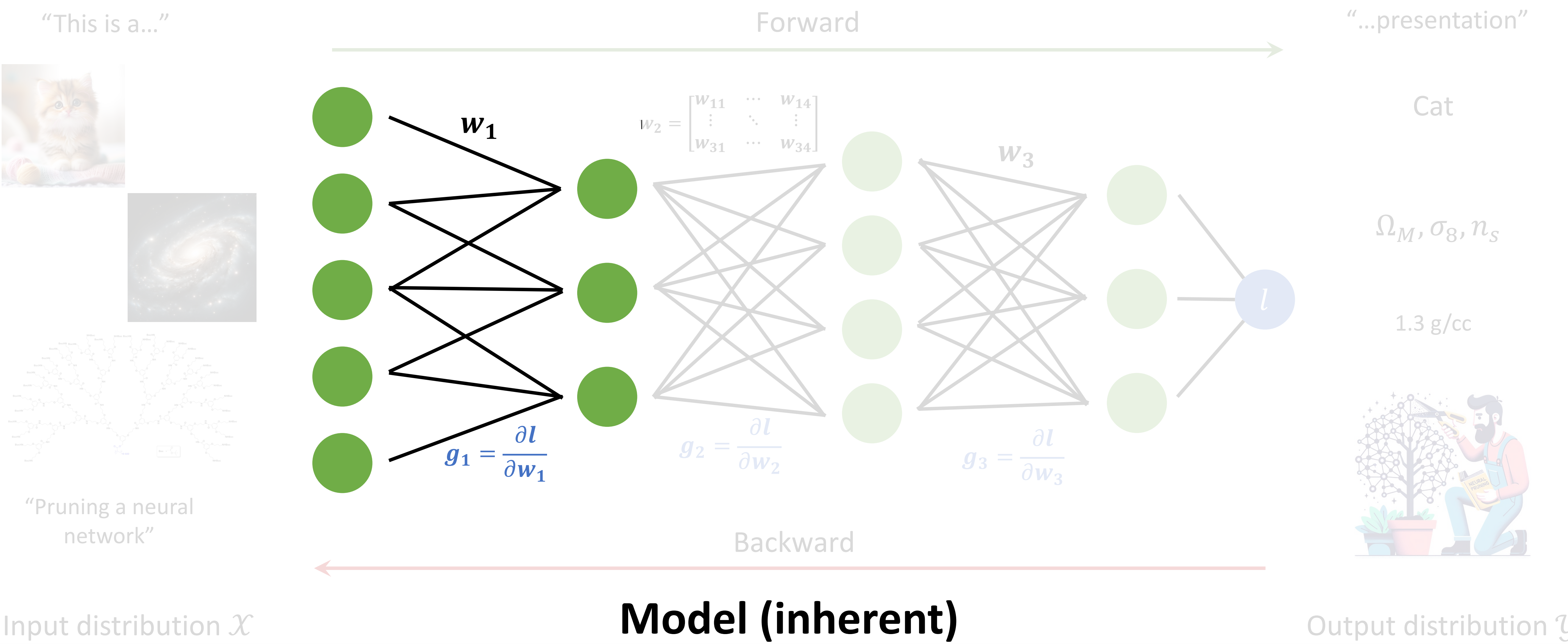
Input Representation

Output distribution $\mathcal{Y}$

Where do we encounter sparsity?

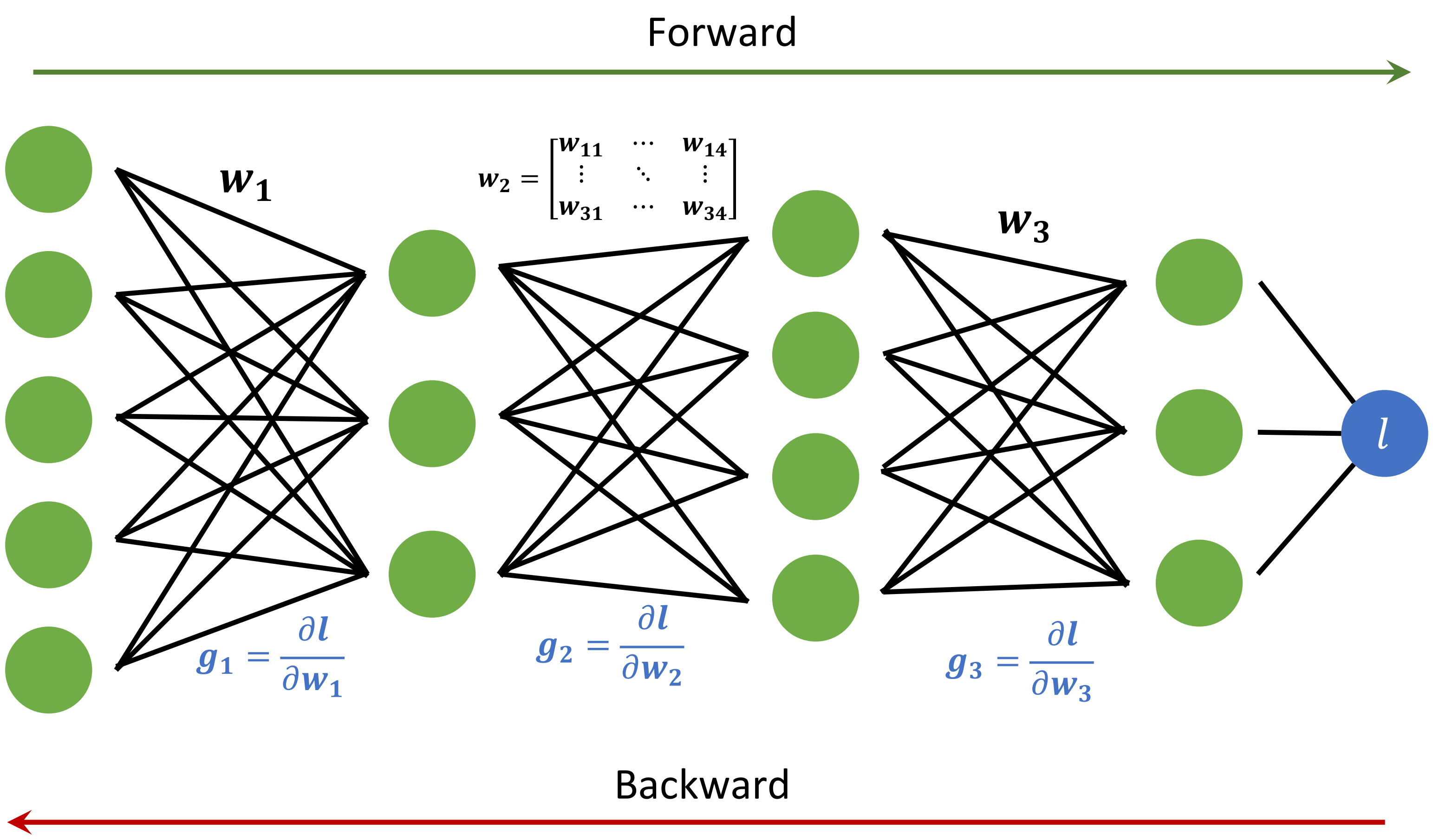


Where do we encounter sparsity?



Model (inherent)

Where do we encounter sparsity?



"This is a…"

"Pruning a neural network"

Input distribution $\mathcal{X}$

Model (sparsification)

"…presentation"

Cat

Ω_M, σ_8, n_s

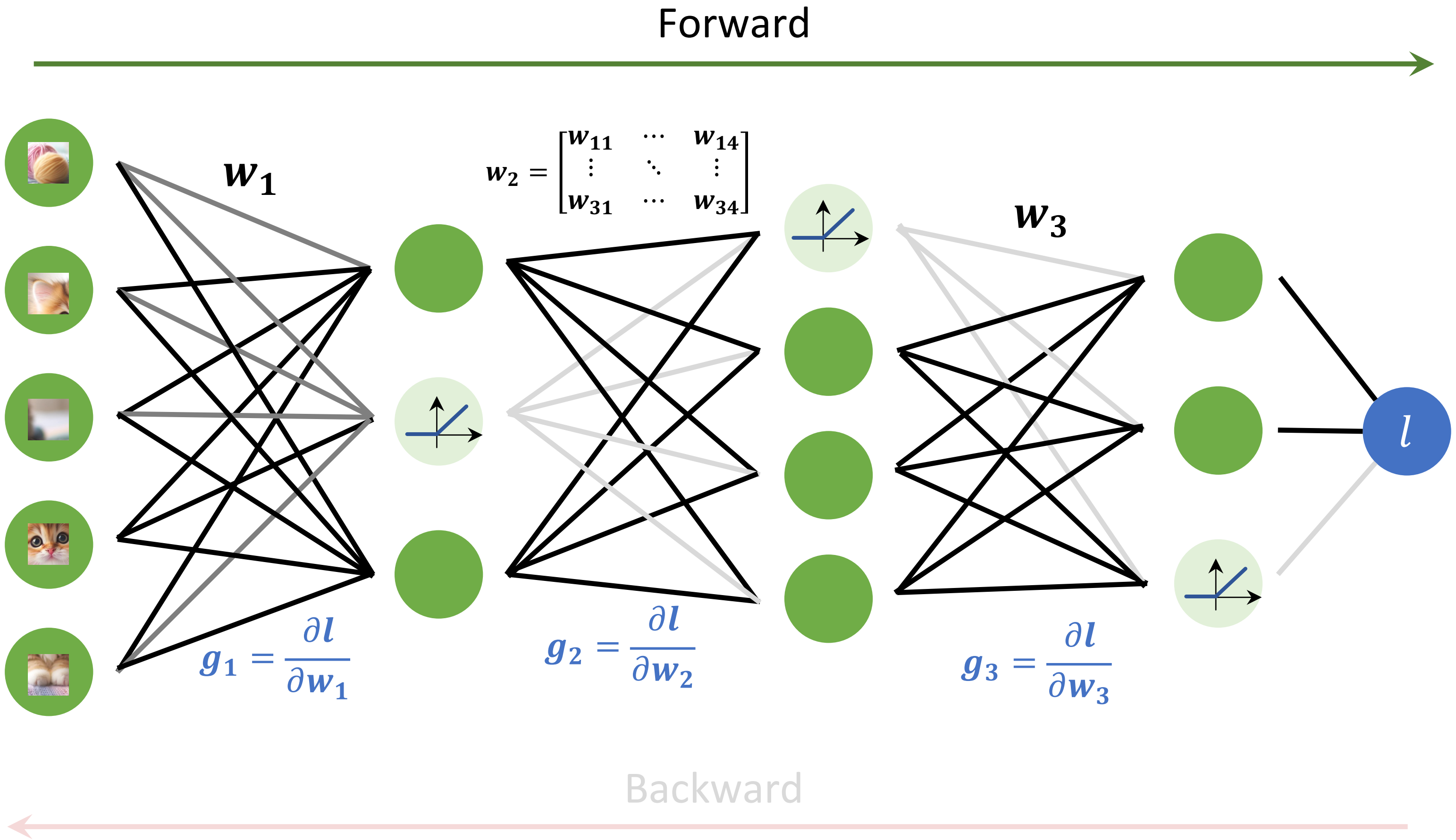
1.3 g/cc

Output distribution $\mathcal{Y}$

Where do we encounter sparsity?

"This is a…"

"Pruning a neural network"



"…presentation"

Cat

Ω_M, σ_8, n_s

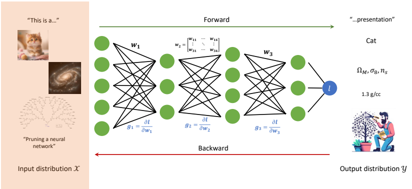
1.3 g/cc

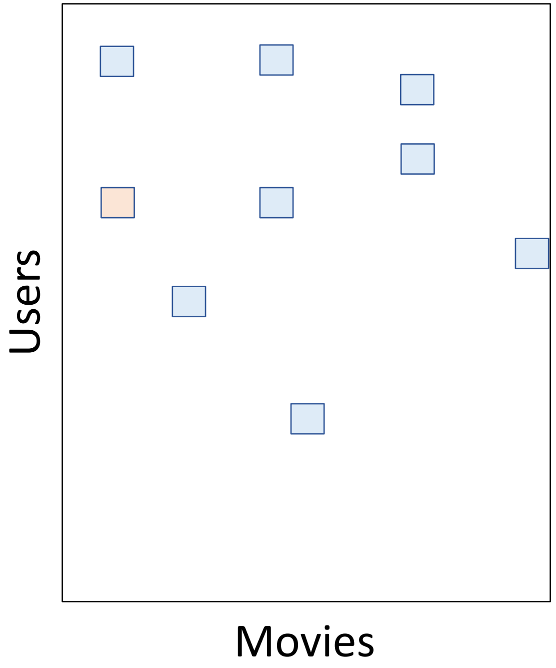
Input distribution $\mathcal{X}$

Ephemeral (input-induced)

Output distribution $\mathcal{Y}$

Input representations

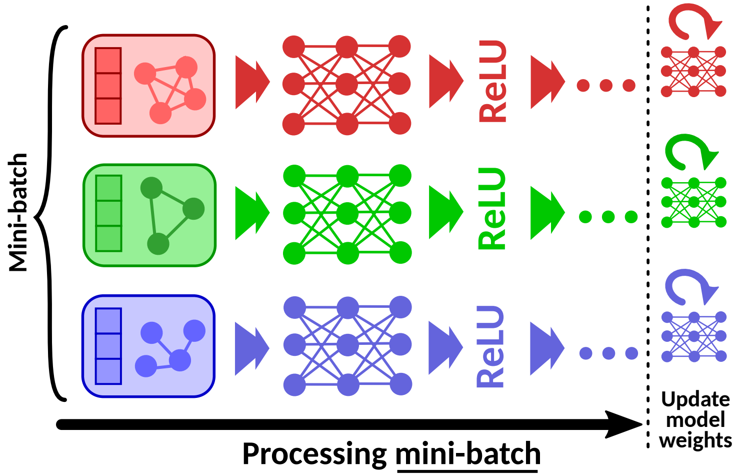




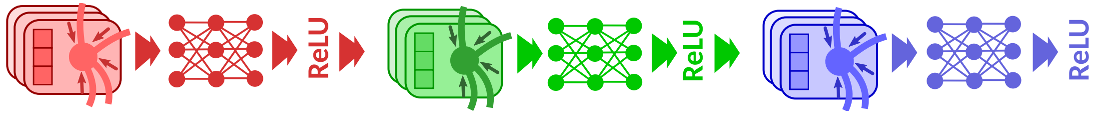
"X likes The Barbie Movie, what else might they like?"
Model output: "Oppenheimer"

Recommendation Systems

Data parallelism



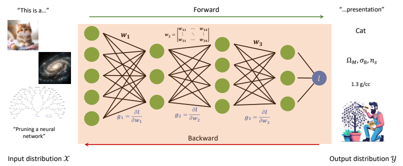
Pipeline parallelism

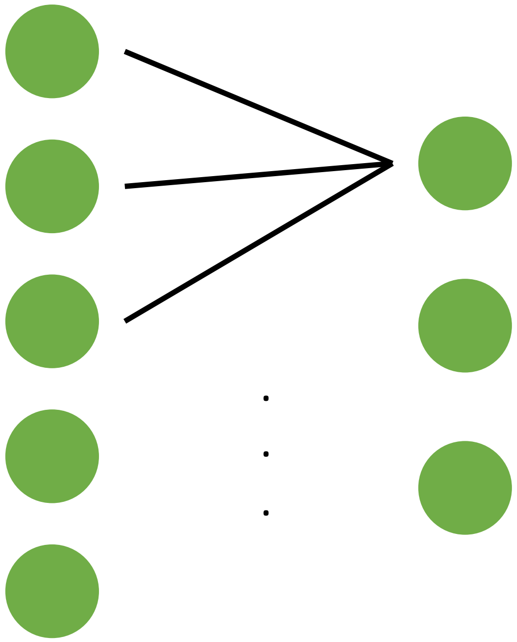


Graph Neural Networks

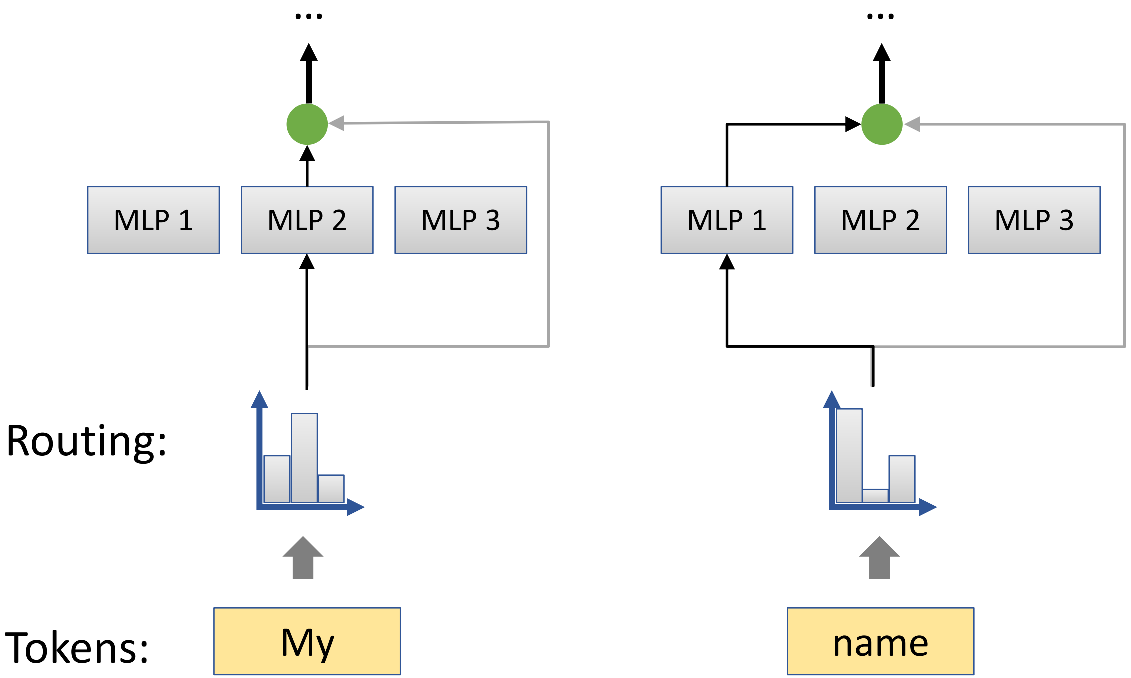
Besta & Hoefler, "Parallel and Distributed Graph Neural Networks: An In-Depth Concurrency Analysis", 2022

Inherent sparsity in models





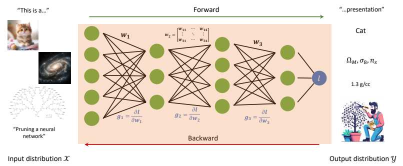
Convolution Operator



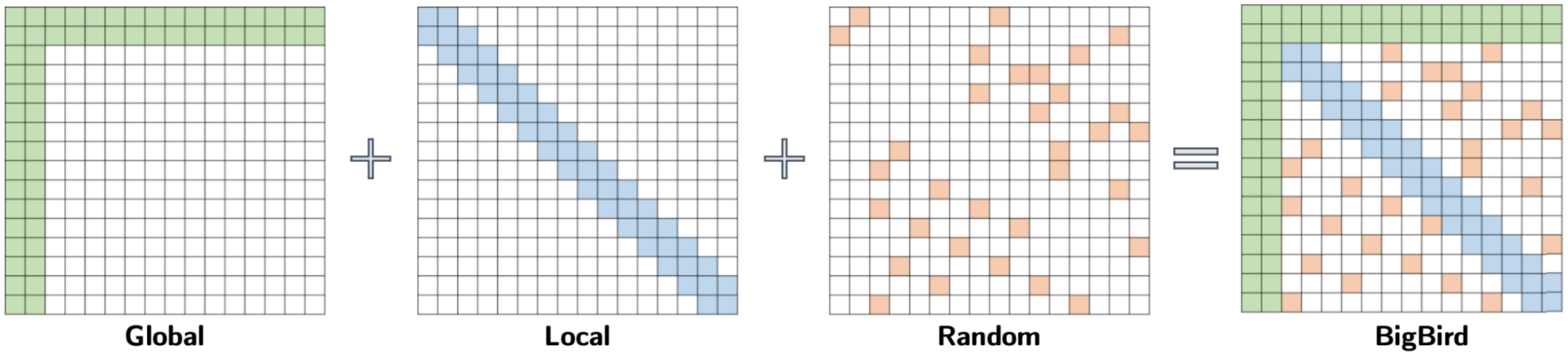
Mixture of Experts

Fedus, Zoph, Shazeer. "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", JMLR'22

Inherent sparsity in models – sparse attention



- Attention is $O(n^2)$. n is now 128,000



Source: https://blog.research.google/2021/03/constructing-transformers-for-longer.html

Model Architecture	Complexity per Layer	Sequential Operation
Recurrent	$O(n)$	$O(n)$
Transformer, (Vaswani et al., 2017)	$O(n^2)$	$O(1)$
Sparse Tansformer, (Child et al., 2019)	$O(n\sqrt{n})$	$O(1)$
Reformer, (Kitaev et al., 2020)	$O(n\log(n))$	$O(\log(n))$
Linformer	$O(n)$	$O(1)$

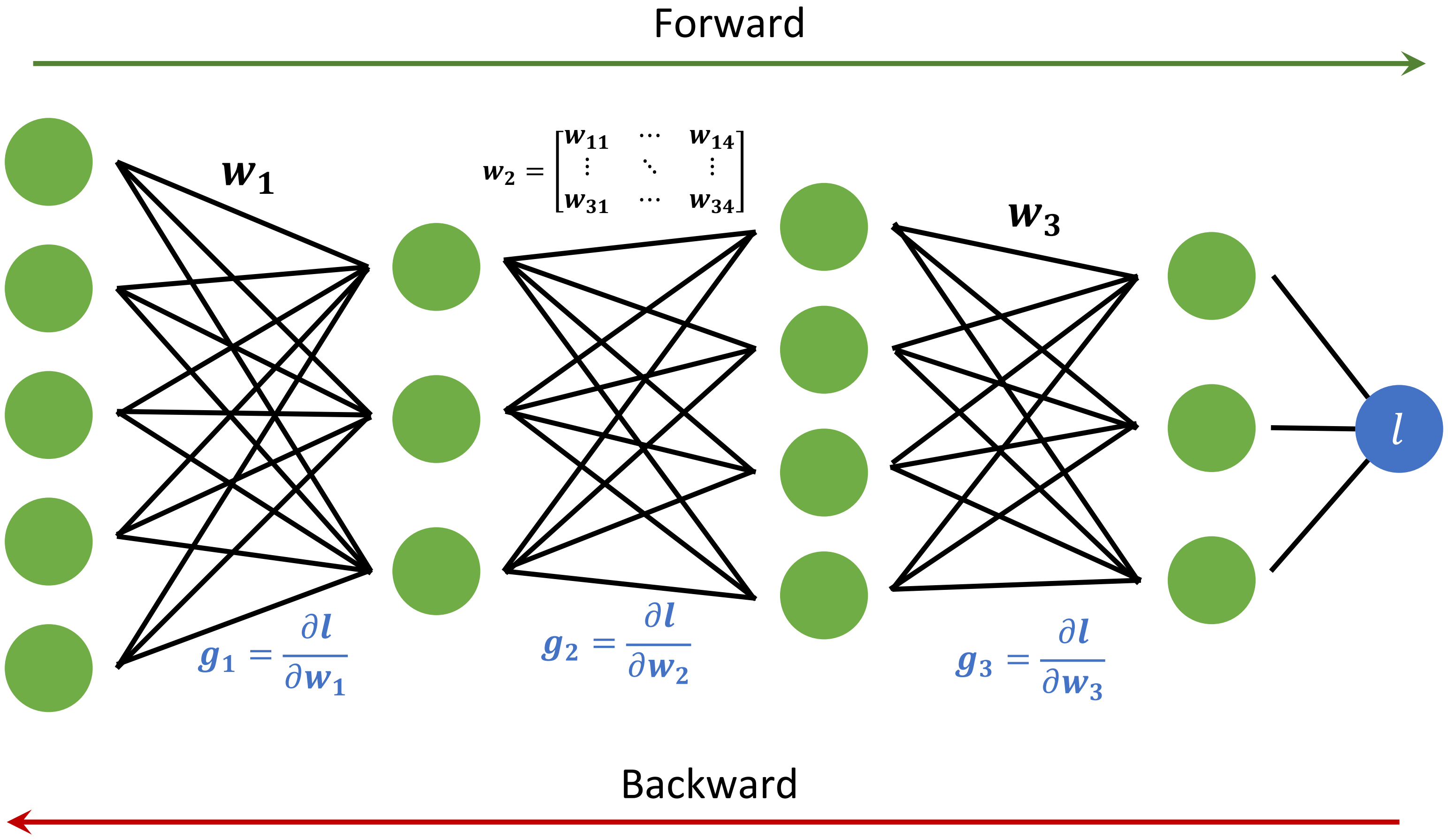
Zaheer et al. "Big Bird: Transformers for Longer Sequences". NeurIPS 2020
Wang et al. "Linformer: Self-Attention with Linear Complexity". arXiv:2006.04768

What about tried and true models?

"This is a…"

"Pruning a neural network"

Input distribution $\mathcal{X}$



Model (sparsification)

"…presentation"

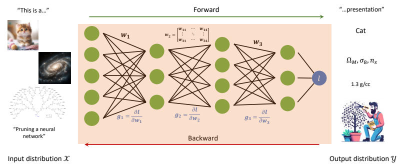
Cat

Ω_M, σ_8, n_s

1.3 g/cc

Output distribution $\mathcal{Y}$

Why should we sparsify?



- Reduces model parameters

- Improves generalization (Occam's Razor)

- Necessity: input representation or infeasibility

- State-of-the-art: 95% sparse ResNet-50, 50% sparse GPT models run at essentially same quality, up to 20x cheaper!

https://jmlr.org/papers/volume22/21-0366/21-0366.pdf

Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks

TORSTEN HOEFLER, ETH Zürich, Switzerland
DAN ALISTARH, IST Austria, Austria
TAL BEN-NUN, ETH Zürich, Switzerland
NIKOLI DRYDEN, ETH Zürich, Switzerland
ALEXANDRA PESTE, IST Austria, Austria

arXiv:2102.00554v1 [cs.LG] 31 Jan 2021

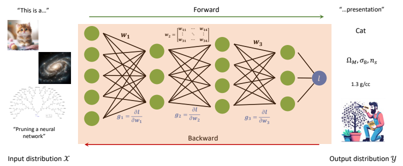
The growing energy and performance costs of deep learning have driven the community to reduce the size of neural networks by selectively pruning components. Similarly to their biological counterparts, sparse networks generalize just as well, if not better than, the original dense networks. Sparsity can reduce the memory footprint of regular networks to fit mobile devices, as well as shorten training time for ever growing networks. In this paper, we survey prior work on sparsity in deep learning and provide an extensive tutorial of sparsification for both inference and training. We describe approaches to remove and add elements of neural networks, different training strategies to achieve model sparsity, and mechanisms to exploit sparsity in practice. Our work distills ideas from more than 300 research papers and provides guidance to practitioners who wish to utilize sparsity today, as well as to researchers whose goal is to push the frontier forward. We include the necessary background on mathematical methods in sparsification, describe phenomena such as early structure adaptation, the intricate relations between sparsity and the training process, and show techniques for achieving acceleration on real hardware. We also define a metric of pruned parameter efficiency that could serve as a baseline for comparison of different sparse networks. We close by speculating on how sparsity can improve future workloads and outline major open problems in the field.

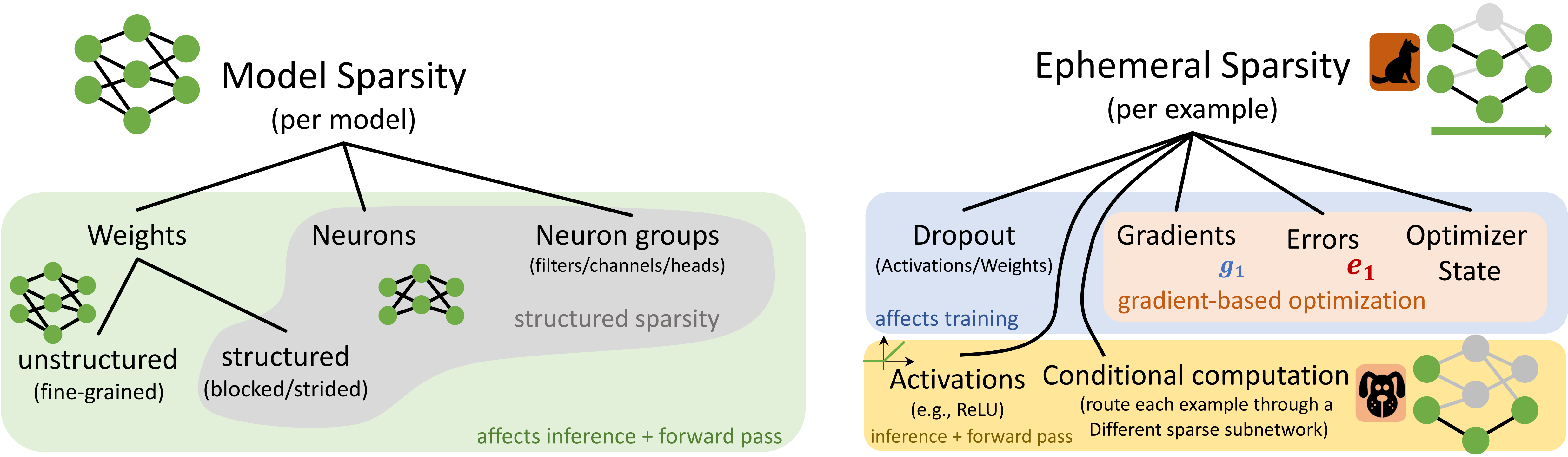
The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single

1 INTRODUCTION

Deep learning shows unparalleled promise for solving very complex real- such as computer vision, natural language processing, knowledge represer systems, drug discovery, and many more. With this development, the fi is moving from traditional feature engineering to neural architecture en

A taxonomy of model sparsification





g_1

Hoefler, Alistarh, Ben-Nun, Dryden, Peste. “Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks”, JMLR’21

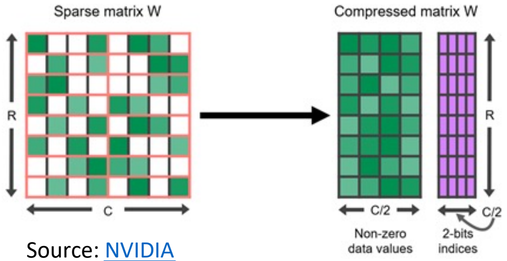
Sparse representations in ML

Dense	Bitmap	RLE / Delta	CSR/CSC/CSF	HYB	COO
[0,2,0,0,3,4,0,0,0,0,0,5]	[010011000001|2345]	[1|2,2|3,0|4,5|5]	[1] [1|2,2|3,0|4,5|5]		[1|2, 5|3, 6|4, 12|5]

0% dense — 30% low sparsity — 70% medium sparsity — 90% moderate sparsity — 99.9% high sparsity — 99.99999%

50%

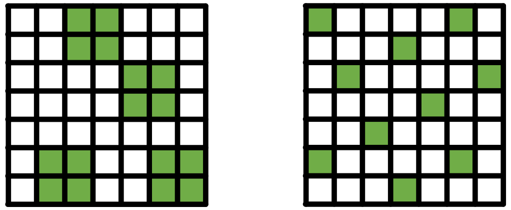
2:4 sparsity



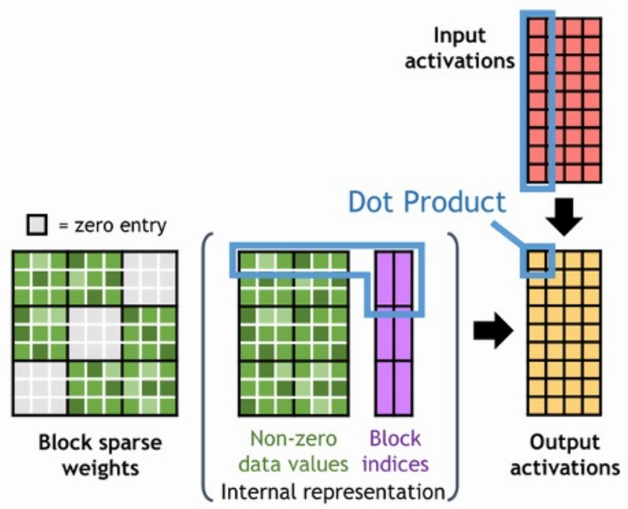
Source: NVIDIA

Considerations in picking a representation

- Sparsity can be conditioned
  - The more constraints applied, the worse the end result



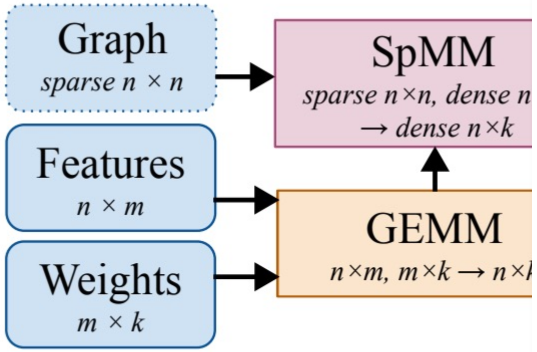
- Training and inference differ
  - Transposed representation necessary for backpropagation
  - Sparse representation may change during training!

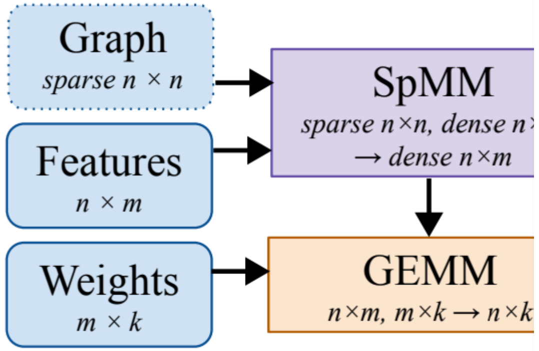


Source: NVIDIA

- Hardware support
  - NVIDIA Sparse Tensor Cores
  - CSR/CSC can be used effectively for inference

Operation order in GNNs matters!





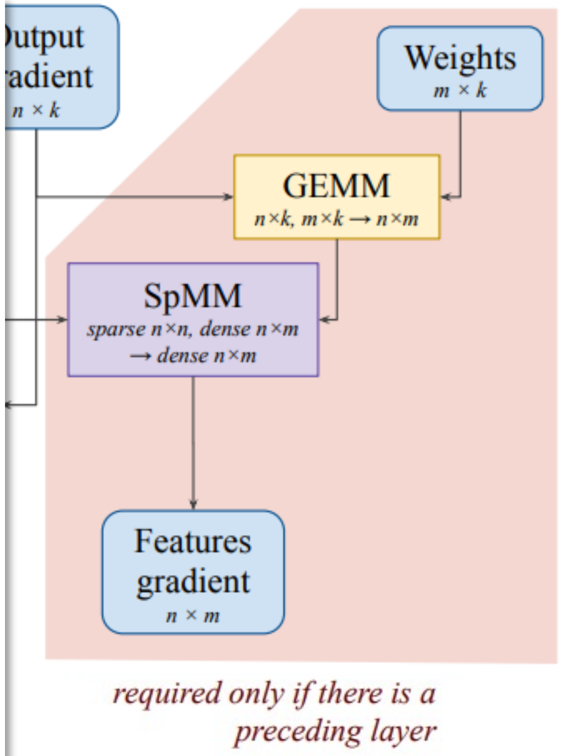
Forward

Table 1: Summary of differences between alternative GCN implementations. Entries in the "Usage condition" column were derived to minimize the total number of operations and memory transfers in the computation. In each pair of alternative schemes, the GEMMs are executed on the same shapes, while the SpMMs are executed on different shapes.

Operation	Scheme (Figure)	GEMM input sizes	SpMM input sizes	# Transient values	Usage condition
GCN forward	Transform-first (1a)	nm, mk	nn, nk	nk	$k < m$
	Propagate-first (1b)		nn, nm	nm	$k \geq m$
GCN backward with feature gradient	Fused-propagate (12a)	nm, mk nm, nk	nn, nk	nk	$k < 2m$
	Split-propagate (12b)		$2\times$: nn, nm	$2nm$	$k \geq 2m$
GCN backward no feature gradient	Fused-propagate (12a)	nm, mk nm, nk	nn, nk	nk	$k < m$
	Split-propagate (12b)		nn, nm	nm	$k \geq m$

Table 2: Comparison of GCN computation schemes with caching. The case presented in Figures 1b and 12b is omitted because it always executes more operations than the scheme using caching. In each pair of alternative schemes, the GEMMs are executed on the same shapes, while the SpMMs are executed on different shapes.

Operation	Forward scheme (Figure)	Backward scheme (Figure)	GEMM input sizes	SpMM input sizes	# Transients (fwd; bwd)	Usage condition
GCN, with feature gradient	Transform-first (1a)	Fused-propagate (12a)	nm, mk nk, mk nm, nk	$2\times$: nn, nk	$nk; nk$	$k < m$
	Propagate-first with caching (13a)	Split-propagate with caching (13b)		$2\times$: nn, nm	$nm; nm$	$k \geq m$
GCN, no feature gradient	Transform-first (1a)	Fused-propagate (12a)	nm, mk nm, nk	$2\times$: nn, nk	$nk; nk$	$k < \frac{1}{2}m$
	Propagate-first with caching (13a)	Split-propagate with caching (13b)		nn, nm	$nm; 0$	$k \geq \frac{1}{2}m$

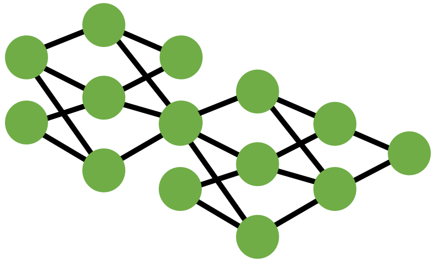


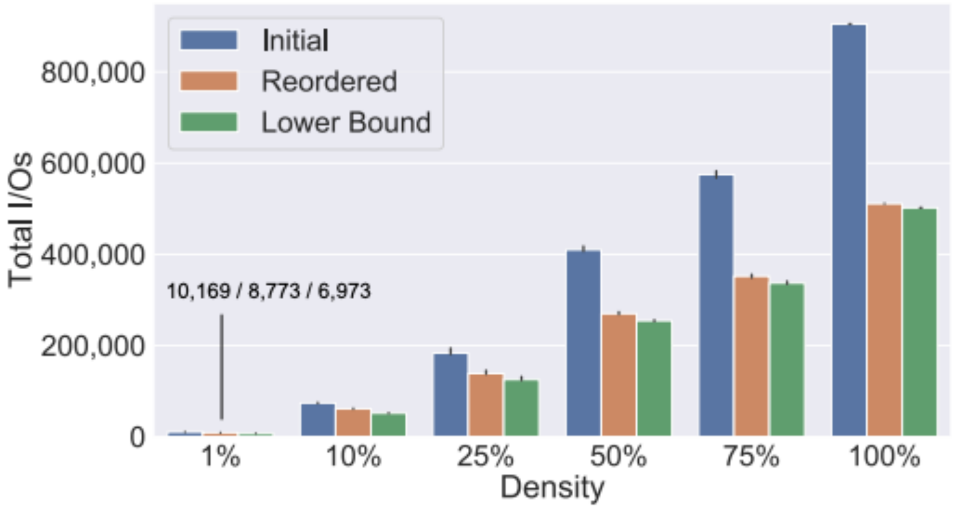
ckward

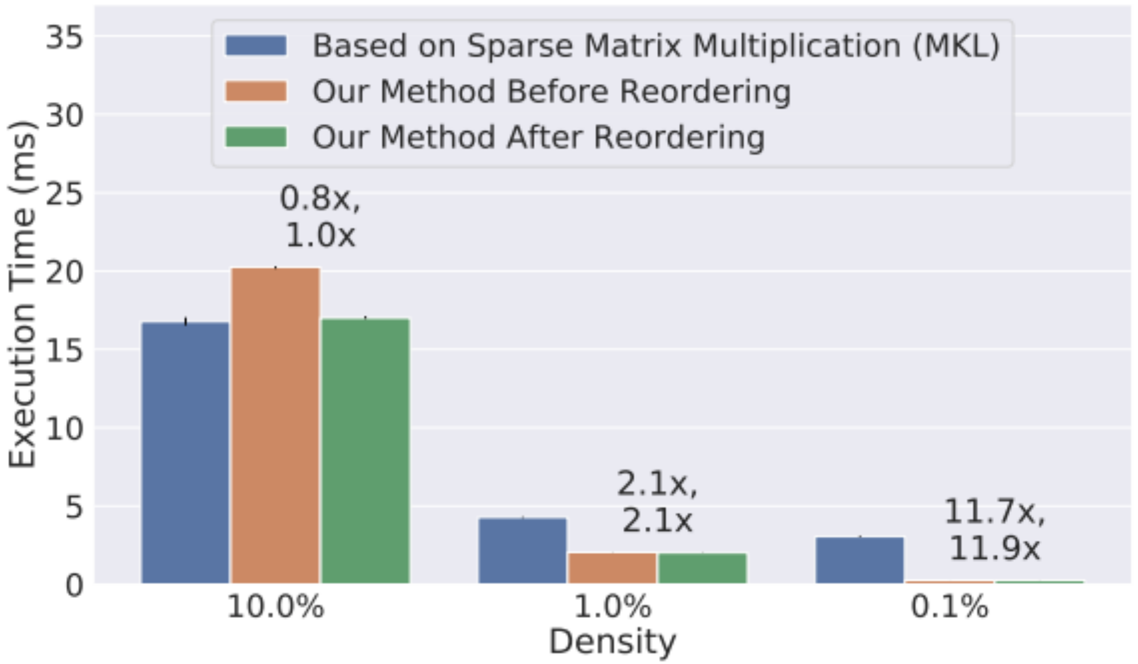
Up to 1.94x speedup over PyTorch Geometric!

Bazinska et al. "Cached Operator Reordering: A Unified View for Fast GNN Training". arXiv:2308.12093

NN evaluation can be performed as graph traversal



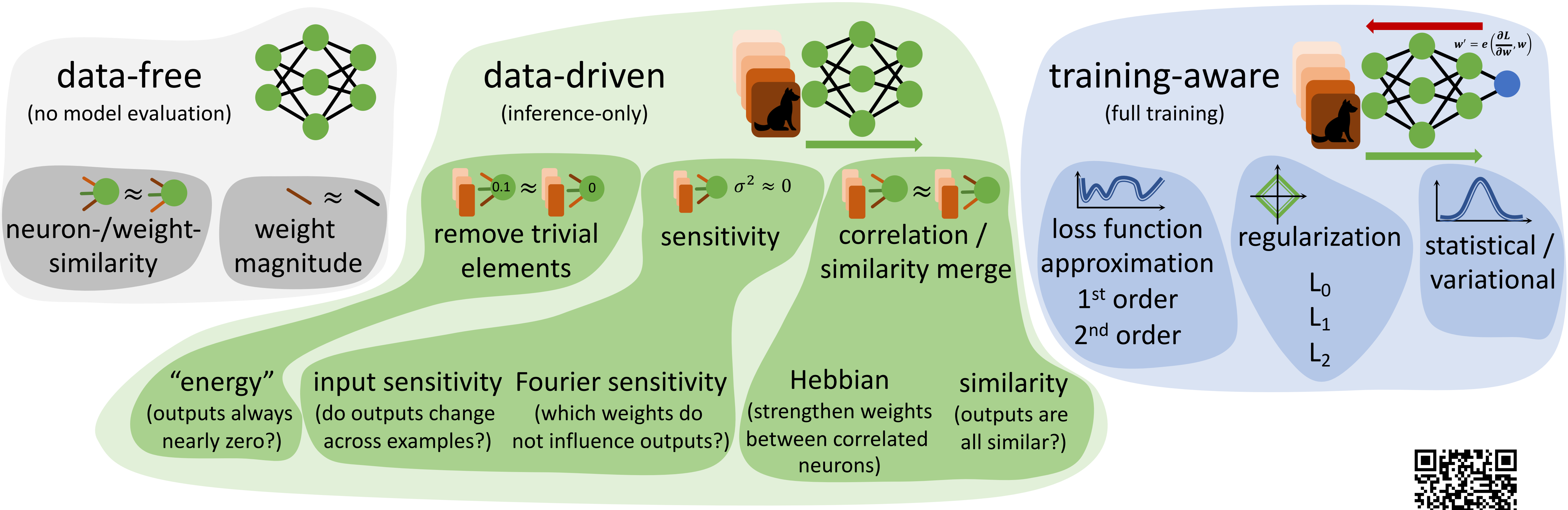




Performance on $\text{BERT}_{\text{LARGE}}$ vs. MKL

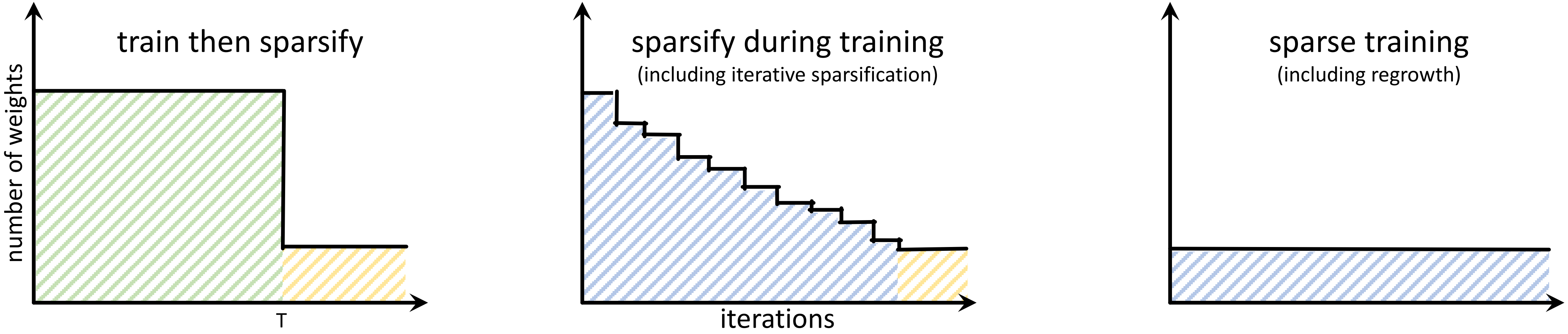
Gleinig, Ben-Nun, Hoefler, "A Theory of I/O-Efficient Sparse Neural Network Inference". arXiv:2301.01048

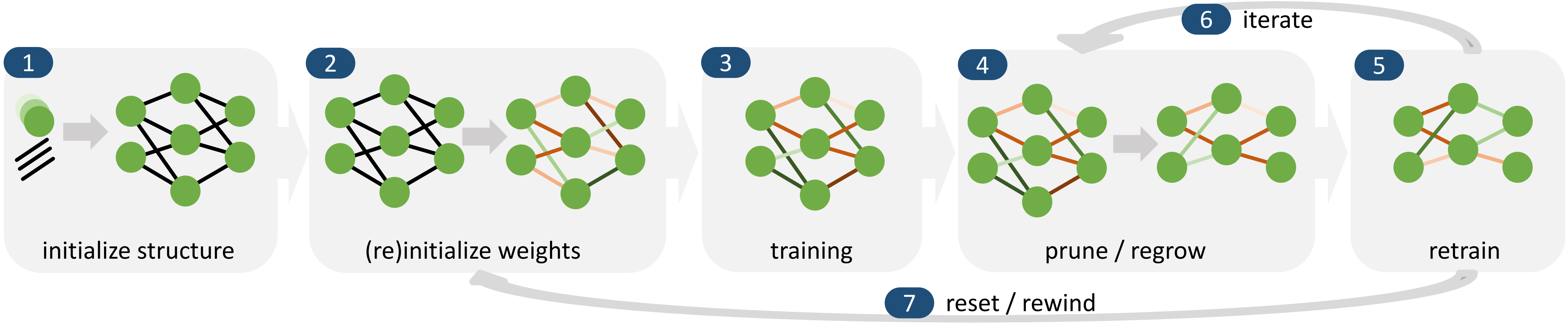
Model sparsification techniques



Hoefler, Alistarh, Ben-Nun, Dryden, Peste. "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks", JMLR'21

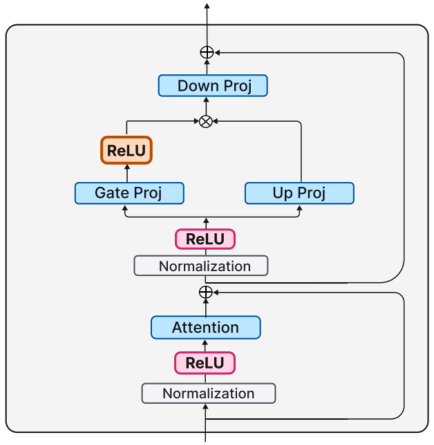
When to sparsify?





When to sparsify?

- Early structure adaptation takes place
- In large models: we may not have the budget to retrain
  — SparseGPT uses second-order information on a set of examples
- Model can be adapted: Inject ReLU to promote activation sparsity
- Repositories such as the Sparse Zoo contain recipes for many models

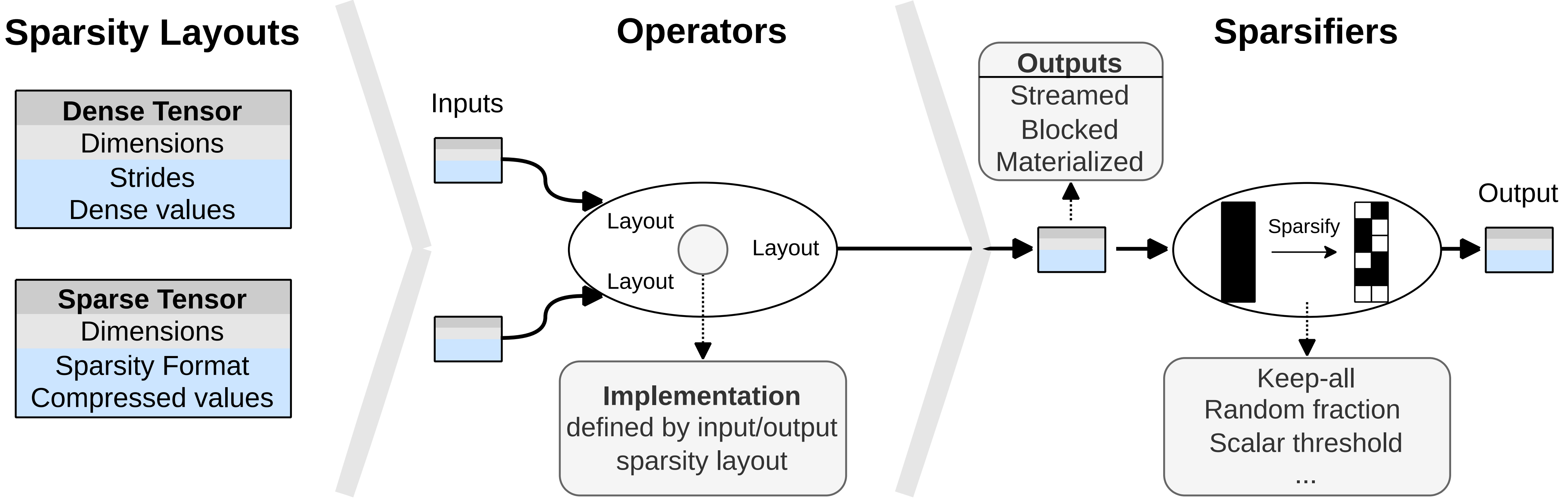


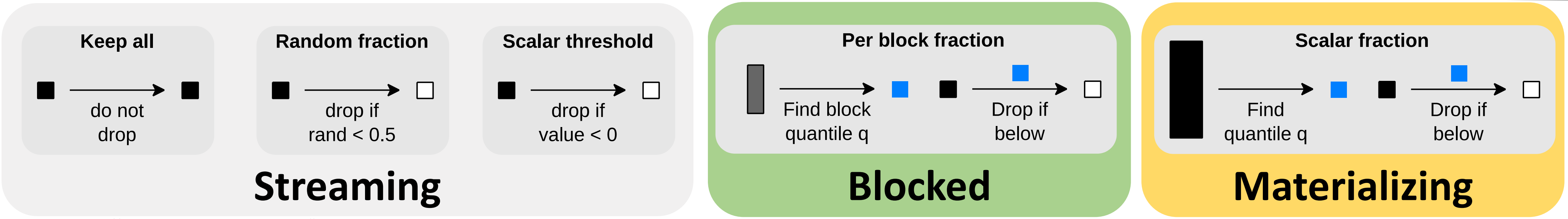
“Best” practices:

- Gradual magnitude pruning will get you most of the way to 90%
  — Higher sparsity or less drop will require more advanced techniques
- Be mindful of which layers you sparsify and their position in the model

Frantar & Alistarh, “SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot”. arXiv:2301.00774
Mirzadeh et al. “ReLU Strikes Back: Exploiting Activation Sparsity in Large Language Models”. arXiv:2310.04564
Sparse Zoo, https://sparsezoo.neuralmagic.com/

Programming Sparse Models – Meet PyTorch STen

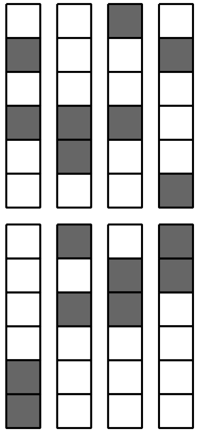




Ivanov et al. "STen: Productive and Efficient Sparsity in PyTorch". arXiv:2304.07613.

STen Performance

2:6 sparse format

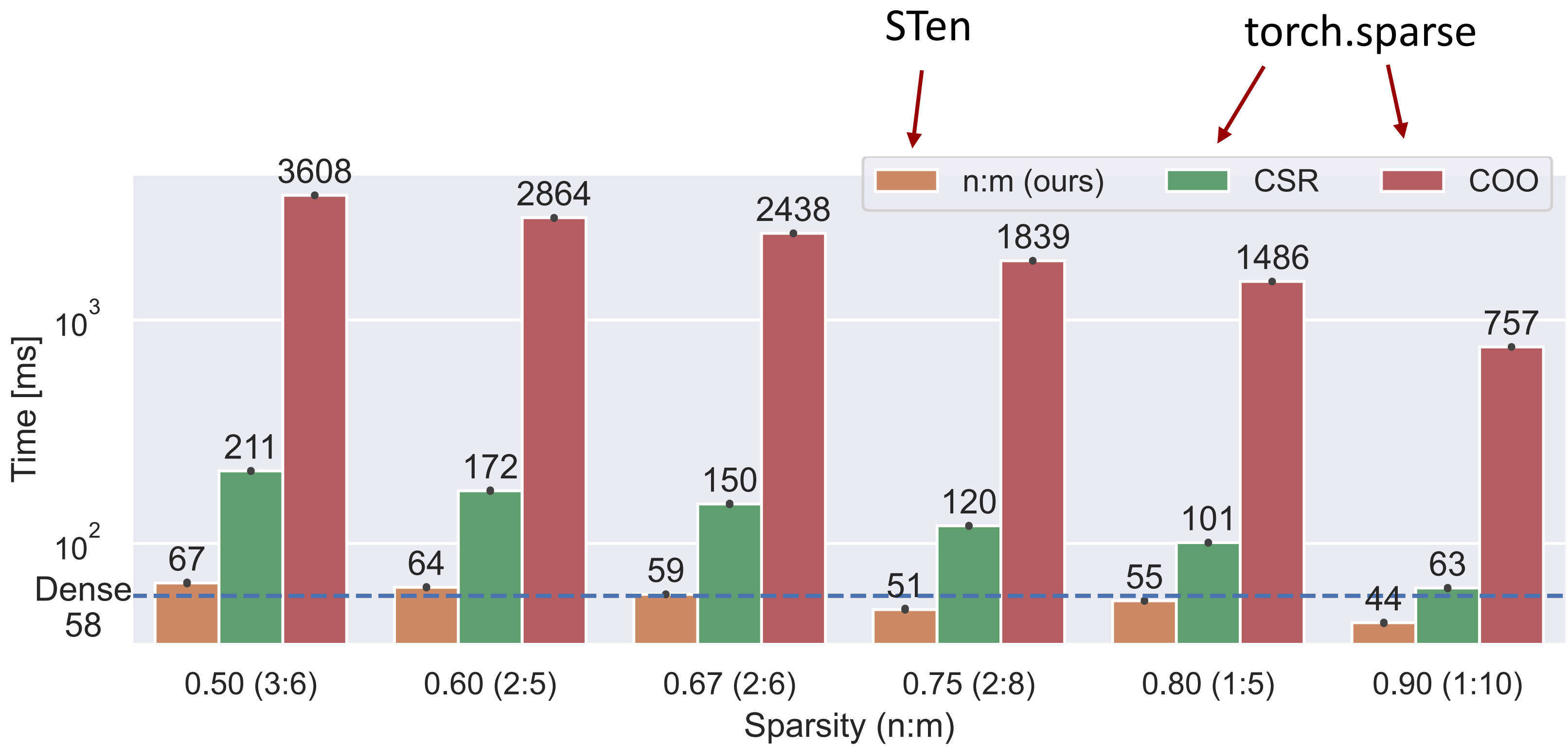


Custom implementation of matrix multiplication:

sparse @ dense -> dense

Linear layer: y = x **W** + b

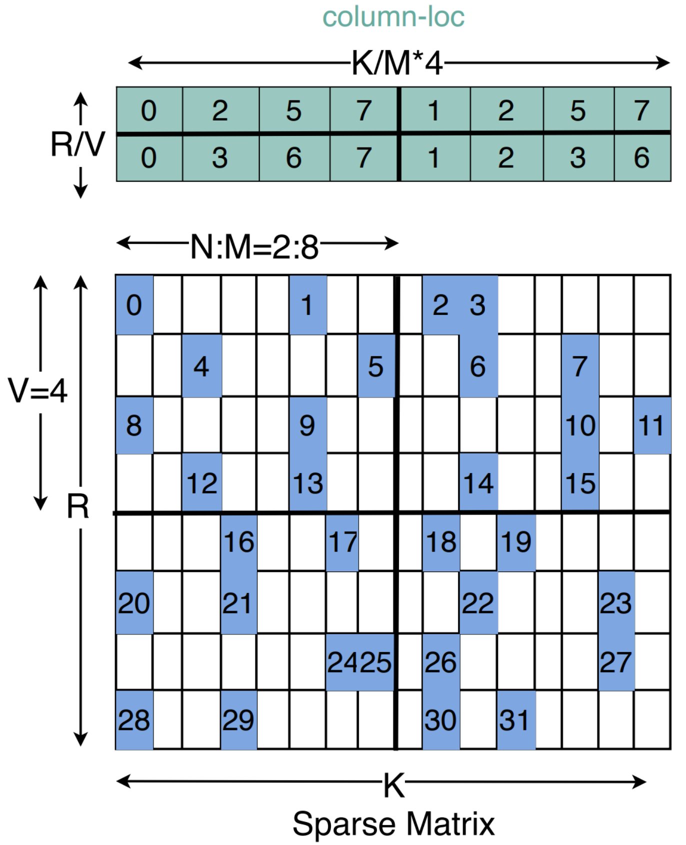
dense dense **sparse** dense

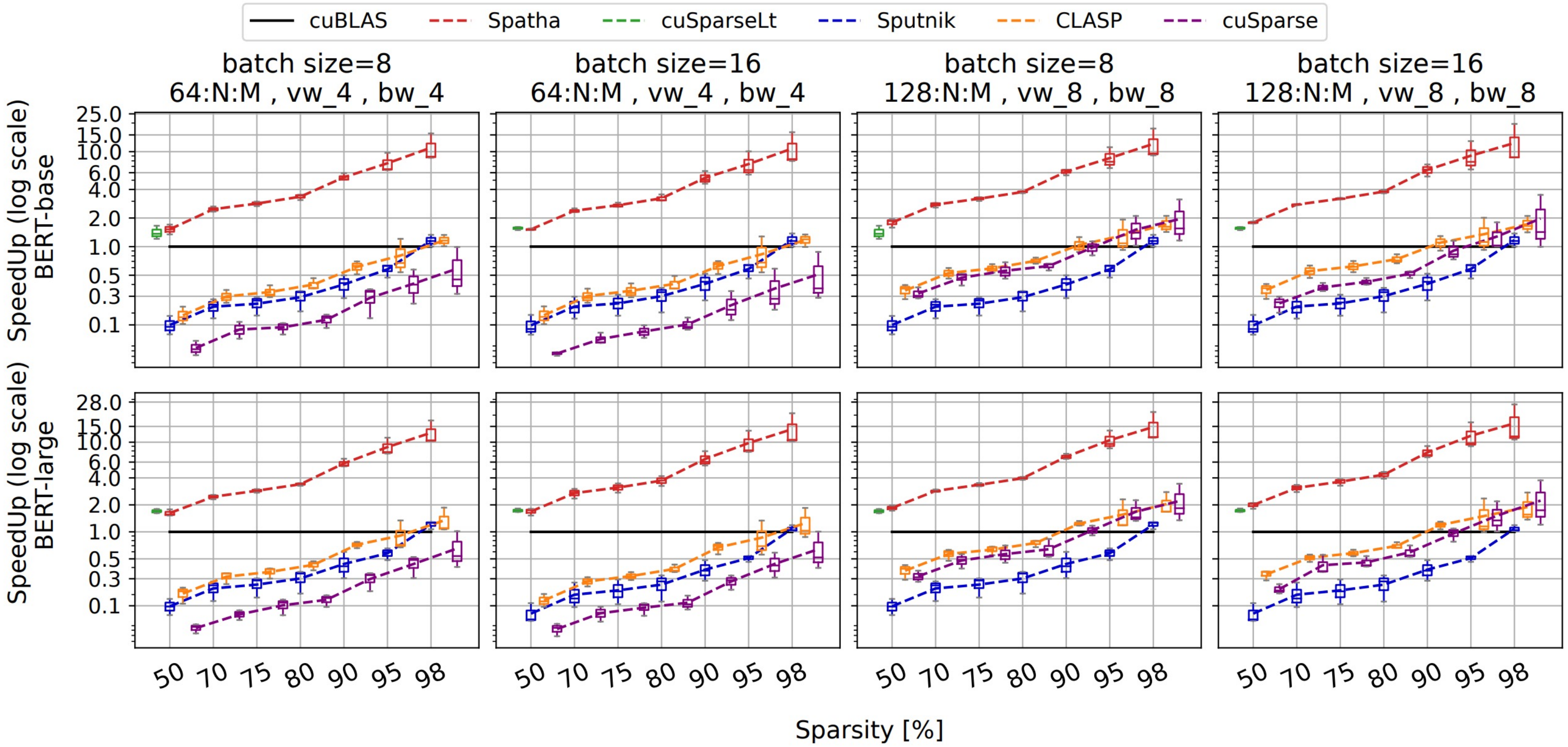


32% speedup

Ivanov et al. "STen: Productive and Efficient Sparsity in PyTorch". arXiv:2304.07613.

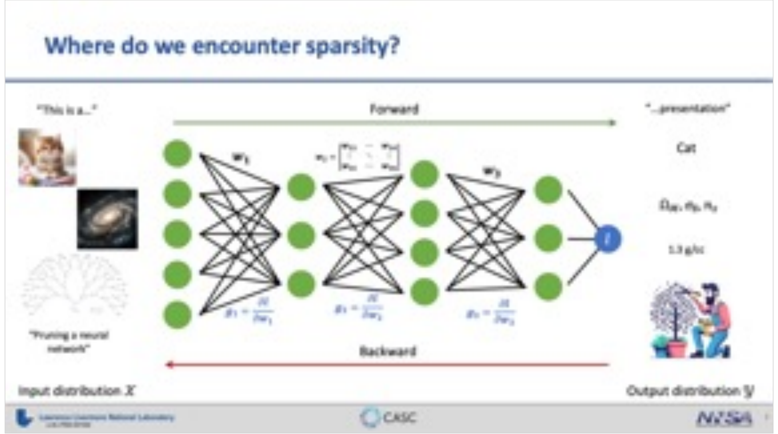
VENOM Performance





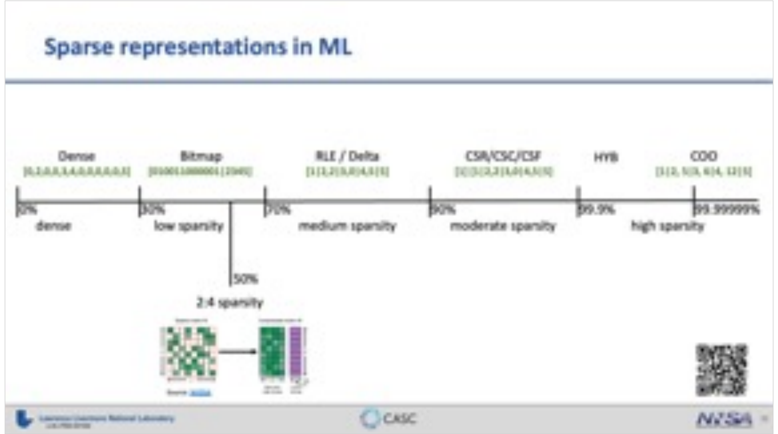
Castro et al.: “VENOM: A Vectorized N:M Format for Unleashing the Power of Sparse Tensor Cores“. SC23

Conclusion

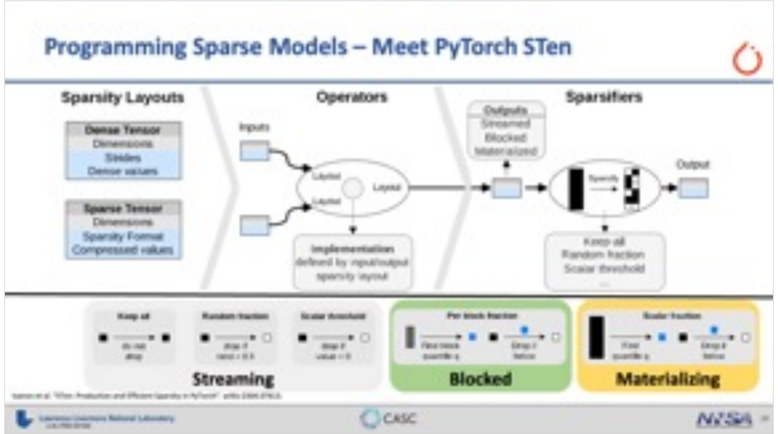


Where do we encounter sparsity?
Forward
Backward
Input distribution 𝒳
Output distribution 𝒴
CASC

Model sparsification techniques
data-free
data-driven
training-aware
CASC



Sparse representations in ML
Dense
Bitmap
RLE / Delta
CSR/CSC/CSF
HYB
COO
dense
low sparsity
medium sparsity
moderate sparsity
high sparsity
2:4 sparsity
CASC



Programming Sparse Models – Meet PyTorch STen
Sparsity Layouts
Operators
Sparsifiers
Streaming
Blocked
Materializing
CASC

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory