



# Programming Novel AI Accelerators for Scientific Computing – Cerebras Wafer Scale Cluster

Claire Zhang ([craig.zhang@cerebras.net](mailto:craig.zhang@cerebras.net))

Cerebras Systems

November 12, 2023





# Cerebras Wafer-Scale Engine (WSE-2)

Still the Largest Chip Ever Made

**850,000** cores optimized for sparse linear algebra

**46,225 mm<sup>2</sup>** silicon

**2.6 trillion** transistors

**40 gigabytes** of on-chip memory

**20 PByte/s** memory bandwidth

**220 Pbit/s** fabric bandwidth

**7nm** process technology

**Cluster-scale performance in a single chip**

# Cerebras CS-2 System

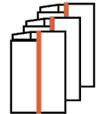
The world's fastest  
AI accelerator

- ✓ Deploy easily into existing racks
- ✓ Cluster-scale in a single system
- ✓ Datacenter-scale in a cluster
- ✓ Available on-prem or remote / in cloud

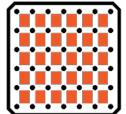


# Condor Galaxy 1 “CG-1” AI Supercomputer

64 Cerebras CS-2 systems, linked together with “*SwarmX*” for 4 ExaFLOPS of AI Compute



**64**  
CS-2 nodes



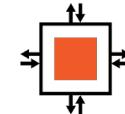
**54 million**  
AI cores



**4 exaFLOPS**  
AI compute  
at FP16



**82 TB**  
parameter  
memory



**388 Tbps**  
internal  
bandwidth



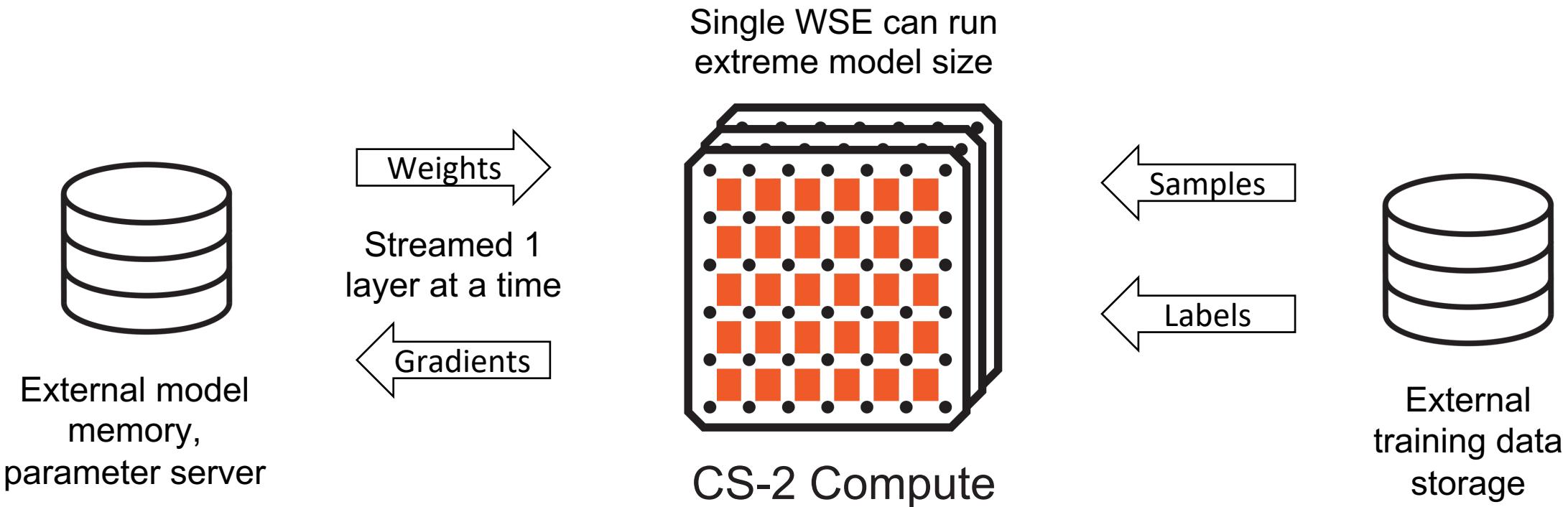
**72,704**  
AMD EPYC™  
cores



**10 days**  
to first  
training run



# Weight Streaming technology disaggregates storage and compute to enable trillion parameter model training

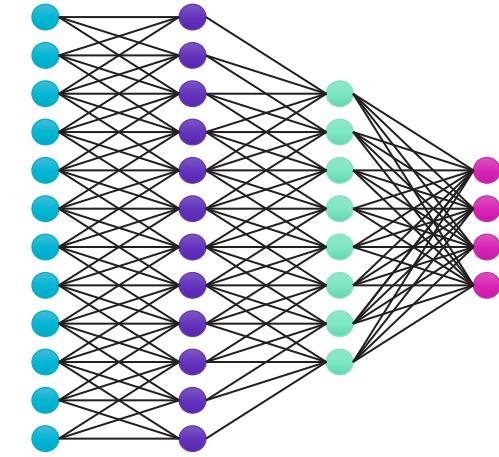


**Scale model size and training speed independently**

# Weight Streaming Execution Model

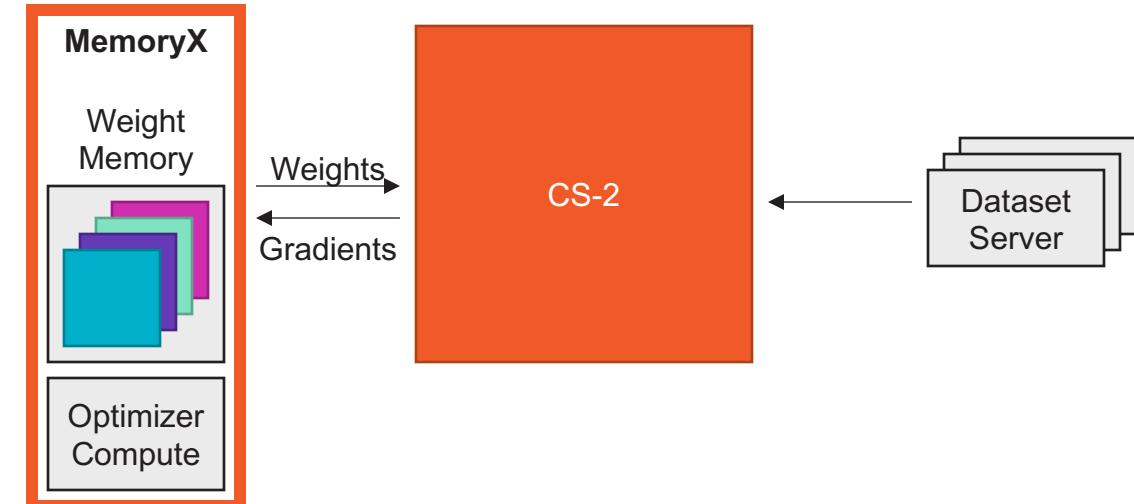
Built for extreme-scale neural networks:

- Weights stored externally off-wafer
- Weights streamed onto wafer to compute layer
- Activations only are resident on wafer
- Execute one layer at a time

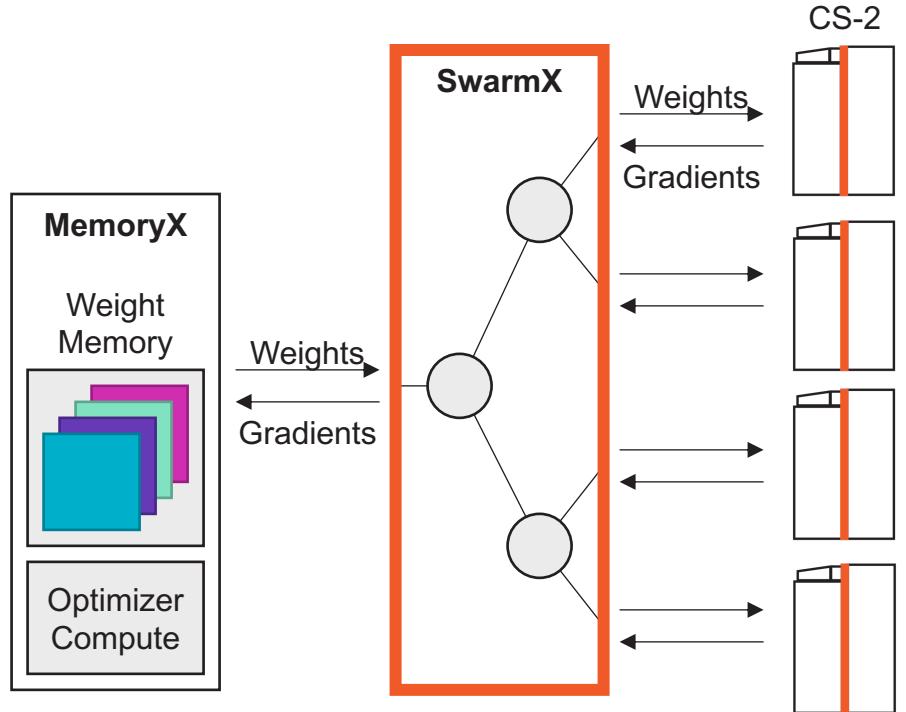


Decoupling weight optimizer compute

- Gradients streamed out of wafer
- Weight update occurs in MemoryX



# Simple Scaling to Multiple CS-2s



- Data parallel training across CS-2s
- Weights are **broadcast** to all CS-2s
- Gradients are **reduced** on way back
- **Multi-system scaling with the same execution model as single system**
  - Same system architecture
  - Same network execution flow
  - Same software user interface

**Scalable to extreme model sizes**  
**Compute scaling independent from capacity**

# Cerebras SDK

A general-purpose parallel-computing platform and API allowing software developers to write custom programs (“kernels”) for Cerebras systems.

## Language

CSL: Cerebras Software Language

Host APIs with Python

## Libraries

Optimized primitives

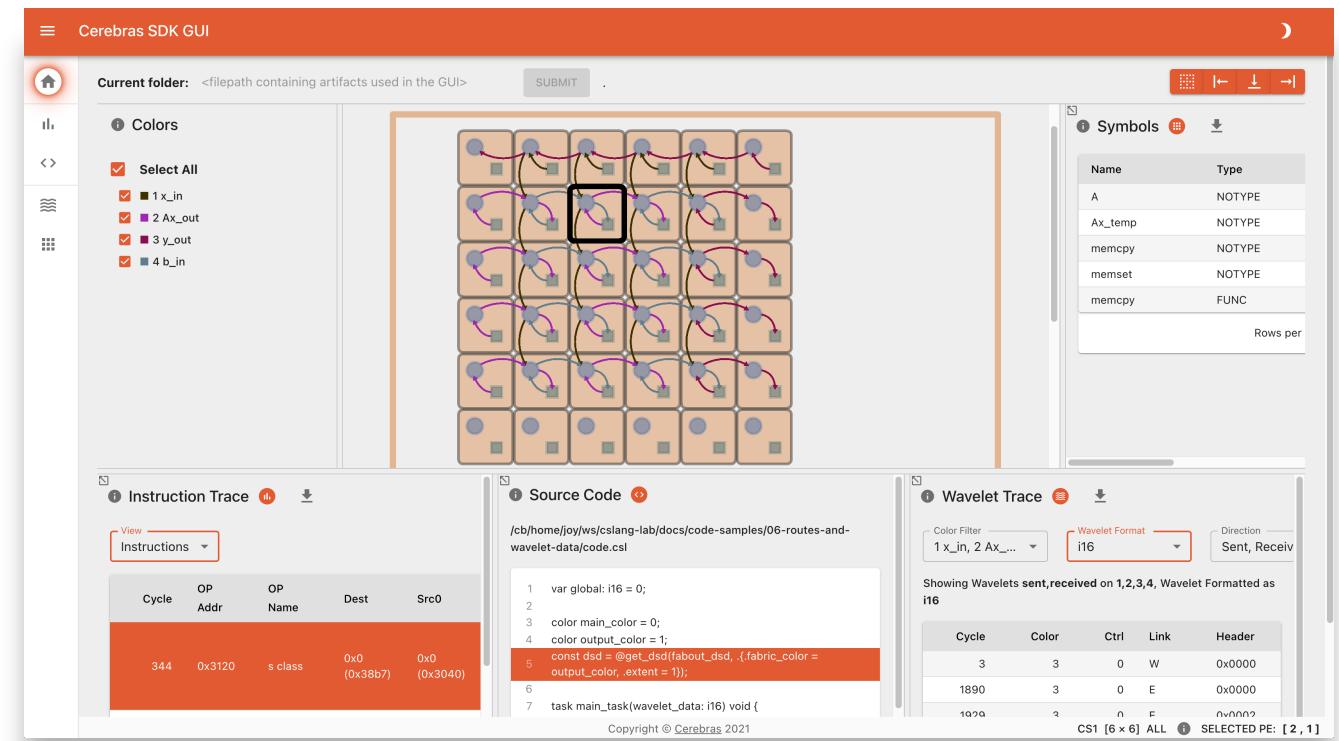
## Tools

Simulator

Debugger

Performance profiler

Visualization



# AI Case Study – Award-Winning COVID Research Accomplished on Cerebras 16-node Cluster



**Collaborator:** Argonne National Lab is a U.S. Department of Energy's national laboratory for science and engineering research



**The Ask:** Accelerate COVID-19 understanding by training 250M-25B GPT models on full-length SARS-CoV-2 genome data, using 10,240 token sequence length



**Challenge:** At the project's timeframe, team found pre-training LLMs with long sequence lengths intractable using conventional hardware, making experimentation difficult due to long training time.



**What we did:** We leveraged our CS-2 cluster and effortless programming model to pre-train the models of interest.



**Outcome:** The collaborative project won the ACM Gordon Bell Special Prize for High Performance Computing-Based COVID-19 Research

## GenSLMs: Genome-scale language models reveal SARS-CoV-2 evolutionary dynamics

Maxim Zvyagin<sup>1†</sup>, Alexander Brace<sup>1,2†</sup>, Kyle Hippe<sup>1\*</sup>, Yuntian Deng<sup>3,4#</sup>, Bin Zhang<sup>5</sup>, Cindy Orozco Bohorquez<sup>5</sup>, Austin Clyde<sup>1,2</sup>, Bharat Kale<sup>6</sup>, Danilo Perez-Rivera<sup>1</sup>, Heng Ma<sup>1</sup>, Carla M. Manni<sup>1</sup>, Michael Irvin<sup>1</sup>, J. Gregory Pauloski<sup>2</sup>, Logan Ward<sup>1</sup>, Valerie Hayot<sup>2</sup>, Murali Emani<sup>1</sup>, Sam Foreman<sup>1</sup>, Zhen Xie<sup>1</sup>, Diangen Lin<sup>2</sup>, Maulik Shukla<sup>1</sup>, Weili Nie<sup>2</sup>, Josh Romero<sup>3</sup>, Christian Dallago<sup>3,7</sup>, Arash Vahdat<sup>3</sup>, Chaowei Xiao<sup>3</sup>, Thomas Gibbs<sup>3</sup>, Ian Foster<sup>1,2</sup>, James J. Davis<sup>1,2</sup>, Michael E. Papka<sup>1,8</sup>, Thomas Brettin<sup>1</sup>, Rick Stevens<sup>1,2</sup>, Anima Anandkumar<sup>3,\*</sup>, Venkatram Vishwanath<sup>1\*</sup>, Arvind Ramanathan<sup>1\*</sup>  
<sup>1</sup>Argonne National Laboratory, <sup>2</sup>University of Chicago, <sup>3</sup>NVIDIA Inc., <sup>4</sup>Harvard University, <sup>5</sup>Cerebras Inc., <sup>6</sup>Northern Illinois University, <sup>7</sup>Technical University of Munich, <sup>8</sup>University of Illinois Chicago, <sup>9</sup>California Institute of Technology  
\*Joint first authors, <sup>#</sup>Contact authors: venkat@anl.gov, anima@caltech.edu, ramanathan@anl.gov

### ABSTRACT

Our work seeks to transform how new and emergent variants of pandemic causing viruses, specially SARS-CoV-2, are identified and classified. By adapting large language models (LLMs) for genomic data, we build genome-scale language models (GenSLMs) which can trace the evolutionary landscape of SARS-CoV-2 genomes. By pre-training on over 110 million prokaryotic gene sequences, and then finetuning a SARS-CoV-2 specific model on 1.5 million genomes, we show that GenSLM can accurately and rapidly identify variants of concern. Thus, to our knowledge, GenSLM represents one of the first whole genome scale foundation models which can generalize to other prediction tasks. We demonstrate the scaling of GenSLMs on both GPU-based supercomputers and AI-hardware accelerators, achieving over 1.54 teraflops in training runs. We present initial scientific insights gleaned from examining GenSLMs in tracking the evolutionary dynamics of SARS-CoV-2, noting that its full potential on large biological data is yet to be realized.

### KEYWORDS

SARS-CoV-2, COVID-19, HPC, AI Large language models, whole genome analyses

ACM Reference Format:

2020. GenSLMs: Genome-scale language models reveal SARS-CoV-2 evolutionary dynamics. In *Supercomputing '20: International Conference for High Performance Computing, Networking, Storage, and Analysis*. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3397274.3399700>

### 1 JUSTIFICATION

We demonstrate achieving >1.54 teraflops in training one of the largest foundation models on whole genome sequences and applying it to characterize SARS-CoV-2 variants of concern. Our models will inform timely public health intervention strategies and downstream vaccine development for emerging viral variants.

### 2 PERFORMANCE ATTRIBUTES

Performance	1.54 teraflops
Category	Whole genome analysis
Type	Large language models
Results reported	Training
Platform	GPU, AI-hardware accelerators
Measures	Accuracy, Speed

**ACM Gordon Bell Prize**  
Recognizing Outstanding Achievement  
in High Performance Computing

2022 Gordon Bell Special Prize Winner

# HPC Case Study – King Abdullah University of Science and Technology (KAUST) Breaks Record on Seismic Processing



**Collaborator:** KAUST is a world-renowned university that was ranked as one of the fastest-rising universities for high-quality research output.



**The Ask:** Deliver improved performance on seismic processing workload by leveraging Cerebras System's unique architecture and massive on-chip memory.



**Challenge:** Seismic processing algorithms are typically memory-bound problems, limited by the memory access speeds of other architectures. Researchers were challenged to re-design an algorithm to take advantage of Cerebras hardware to improve performance.

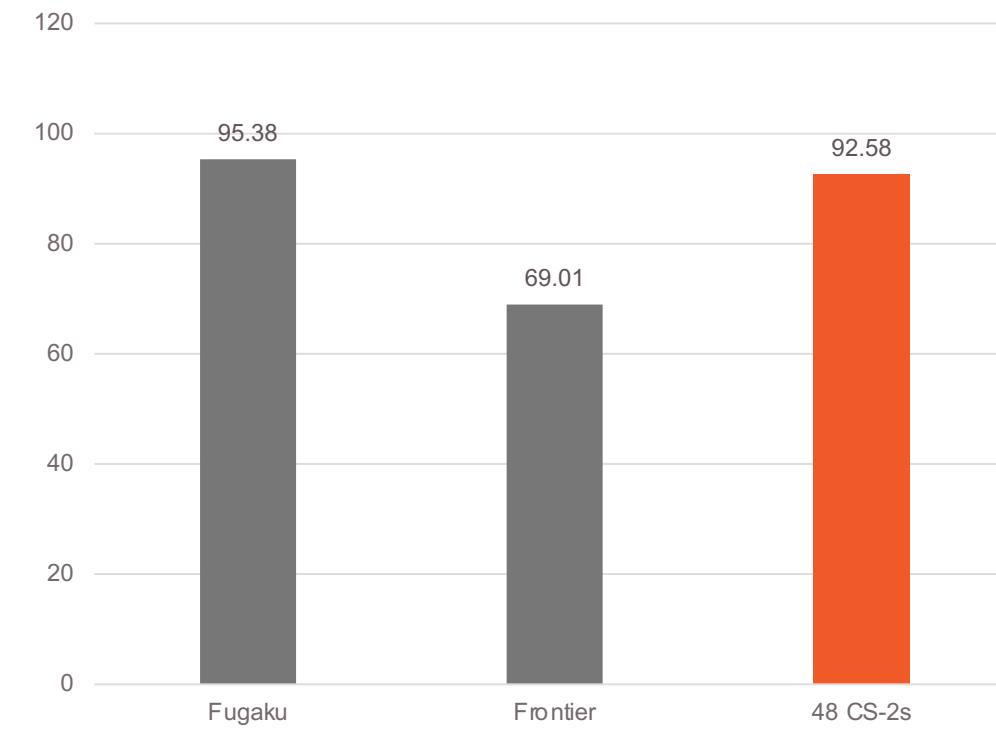


**What we did:** Provided researchers with Cerebras SDK to re-design a Tile Low-Rank Matrix-Vector Multiplication (TLR-MVM) algorithm to be optimized for Cerebras CS-2. We also provided researchers with CG-1, our AI Supercomputer, to run this simulation.



**Outcome:** Achieved a record sustained memory bandwidth of 92.58 Petabytes per second (PB/s) through the implementation of a TLR-MVM kernel that is uniquely tailored to exploit the advanced architecture of Cerebras CS-2 systems.

Cerebras CS-2s achieves real memory bandwidth performance that rivals best-case performance on world's largest supercomputers



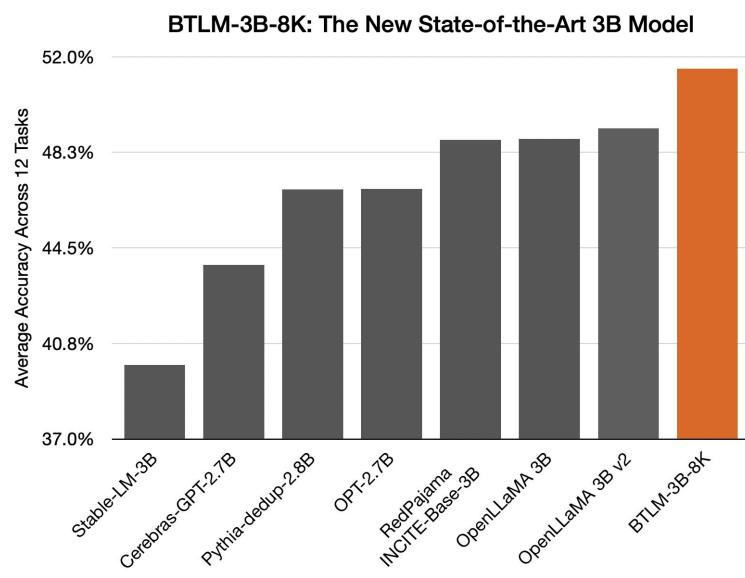
2023 Gordon Bell Prize Finalist

# Cerebras Open-Source Models

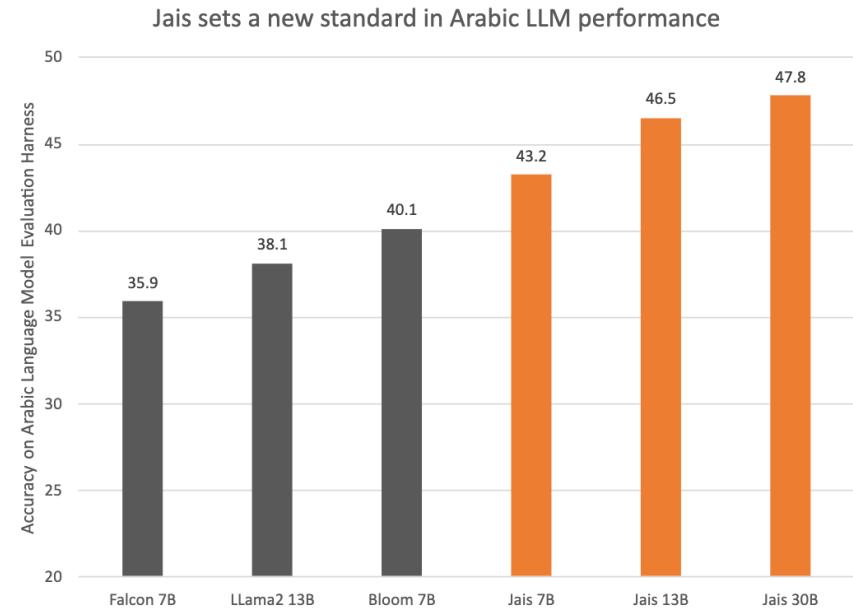
Models 7 Hugging Face

- cerebras/Cerebras-GPT-13B Updated Apr 7 · 20.1k · 601
- cerebras/Cerebras-GPT-6.7B Updated Apr 7 · 6.6k · 59
- cerebras/Cerebras-GPT-2.7B Updated Apr 7 · 10.4k · 33
- cerebras/Cerebras-GPT-1.3B Updated Apr 7 · 10.3k · 39
- cerebras/Cerebras-GPT-590M Updated Apr 7 · 3.44k · 16
- cerebras/Cerebras-GPT-256M Updated Apr 7 · 4k · 19
- cerebras/Cerebras-GPT-111M Updated Apr 7 · 22.5k · 52

**Cerebras-GPT**  
A family of open, compute  
efficient, large language models



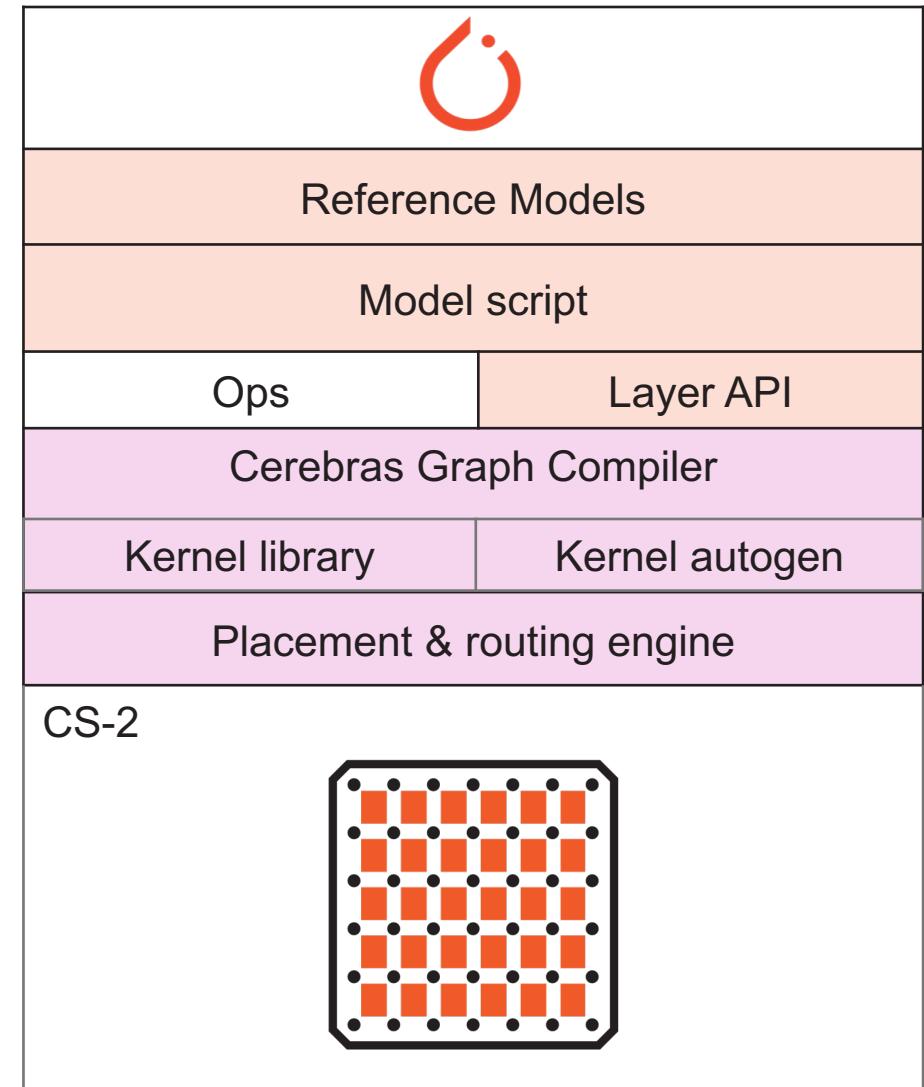
**BTLM-3B-8K**  
The state-of-the-art 3B open-  
source language model, in  
partnership with OpenTensor



**Jais**  
The World's Best Arabic-  
English Language Model, in  
collaboration with G42 and  
MBZUAI

# Develop in PyTorch, Execute on CS

- Dead-simple to run
- Program the same way as you would with a single GPU
- User does not worry about distributed compute or parallelism



# Introducing: Cerebras Model Zoo Repository

[github.com/Cerebras/modelzoo](https://github.com/Cerebras/modelzoo)

As a **starting** point

Reference model implementations (PyTorch):  
GPT-2, GPT-3, GPT-J, BERT and much more

Data preparation scripts

Configurations for multiple model sizes

For **benchmarking**

Tuned implementations for optimal performance

To **develop** new models

Cerebras PyTorch APIs

File organization as a template to start your own models

[docs.cerebras.net](https://docs.cerebras.net)

# Typical Anatomy of a model in Model Zoo

<b>run.py</b>	Main script to execute train, eval in CS-2
<b>configs/</b>	Folder with different parametrizations of the model in .yaml files
<b>model.py</b>	Creation of the NN model function
<b>utils.py</b>	Helper functions to set up run.py
<b>data.py</b>	Helper functions to prepare data

[github.com/Cerebras/modelzoo](https://github.com/Cerebras/modelzoo)

# Programming / training with the cluster is simple

## Define the model

- Write in PyTorch
- Parameterize based on yaml file
- Write *logical* model for *single* device

params\_gpt3xl.yaml

```
### GPT-3 XL 1.3B

hidden_size: 2048
num_hidden_layers: 24
num_heads: 16
```

## Train the model

- Point to the model parameters
- Specify the number of CS-2s
- Specify the number of steps
- Run!

training:

```
python run.py CSX \
--params params_gpt3xl.yaml \
--num_csx 1 \
--num_steps 100 \
--model_dir model_dir \
--mode train
```

# Demo

1.3B GPT-2 Model Training on 1 CS-2

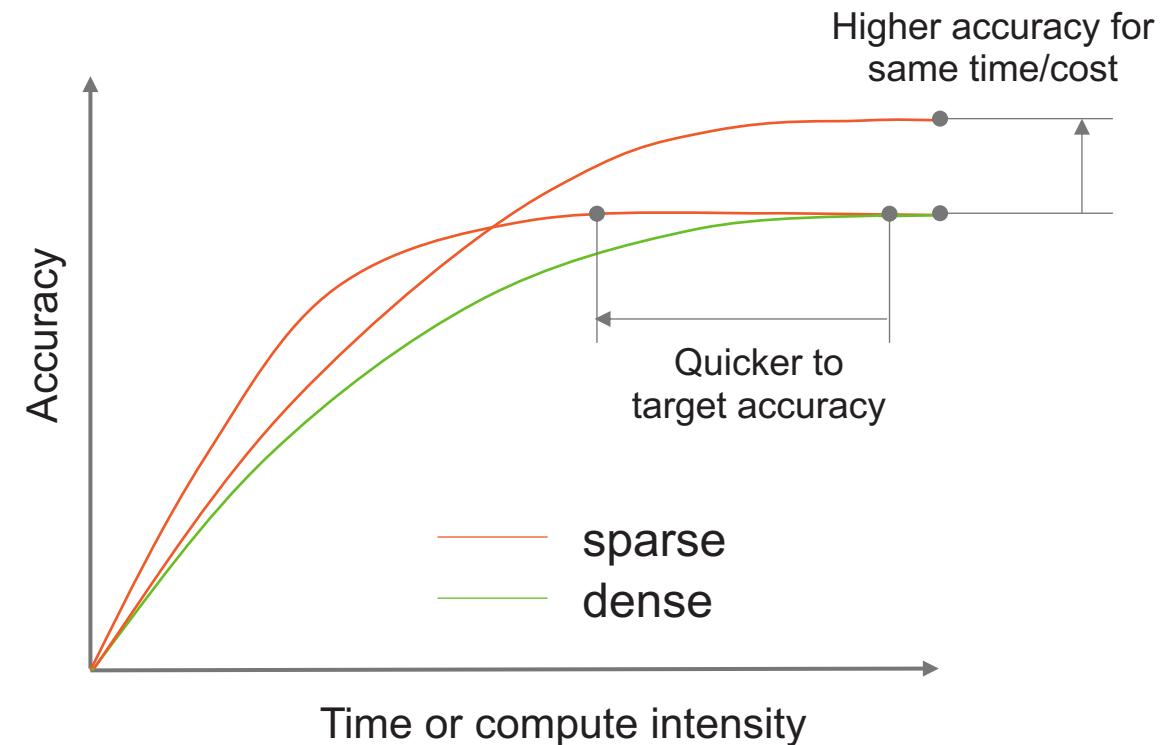
[

# Sparsity Accelerated Training

- Unstructured sparsity acceleration opens up another dimension of advancement beyond improving model architecture
- Sparsity improves training efficiency:
  1. Same accuracy, less training effort
  2. Higher accuracy, same training effort
- The ideal method would achieve both!

***Puts control of compute performance into hands of ML practitioner***

- This is an active research area in the ML community and at Cerebras



# Demo

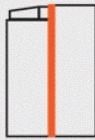
1.3B GPT with Sparsity Training on 1 CS-2

# Scaling compute to more CS-2s is simple

## Scaling compute

- Change the number of CS-2s
- Fully data-parallel training
- Run!

```
python run.py  
--params params.yaml           ← Where's your dataset?  
--num_csx = 1                  ← How many nodes?  
--model_dir = model_dir         ← Where to store weights?  
--num_steps = 1000              ← How many training steps?  
--mode=train                    ← Train, evaluate or infer?
```



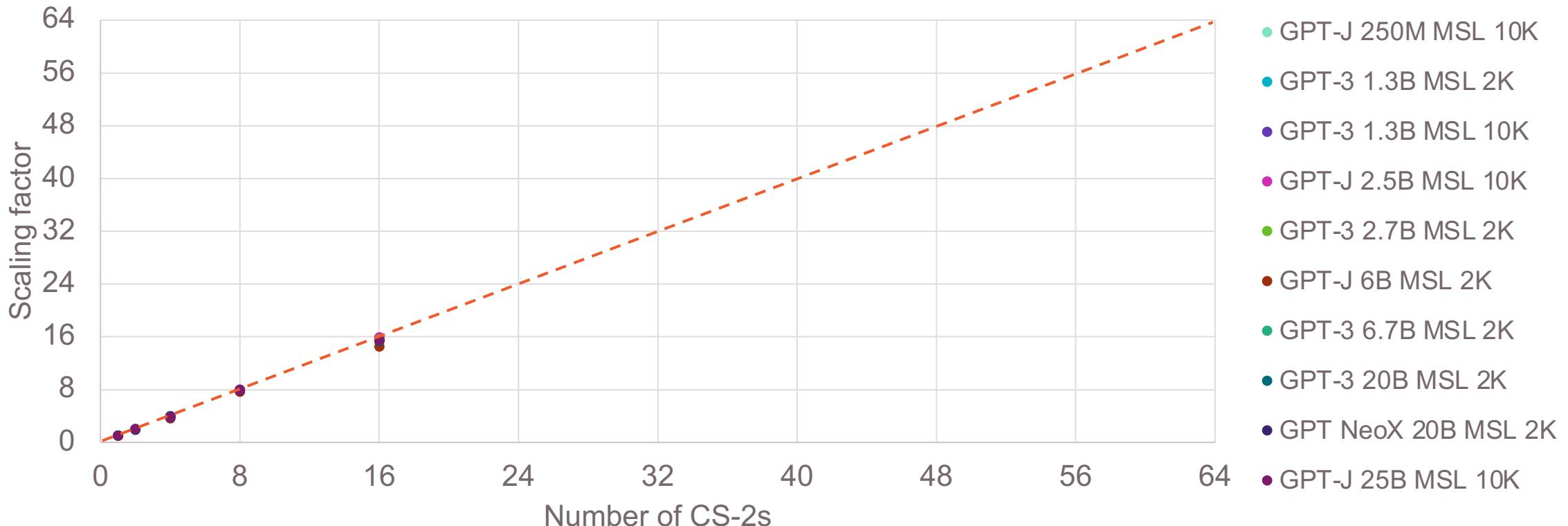
# Demo

1.3B GPT with Sparsity Training on 16 CS-2s

# Data Parallel Models Enables Near Linear Scaling

- Even the largest state-of-the-art models can train on a single CS-2
- Near-linear time to solution scaling across multiple CS-2s in a wafer-scale cluster

Cerebras cluster scaling – GPT training throughput



**Figure.** Measured training throughput scaling for 250M-20B GPT models over 1-16 CS-2 systems; projected scaling to 64 systems.

# Scaling to larger models is simple

## Scaling the model

- Change the model parameters in yaml
- Fully data-parallel training
- Run!

params\_gptneox.yaml

```
### GPT-NeoX 20B  
  
hidden_size: 6144  
num_hidden_layers: 44  
num_heads: 64
```

training:

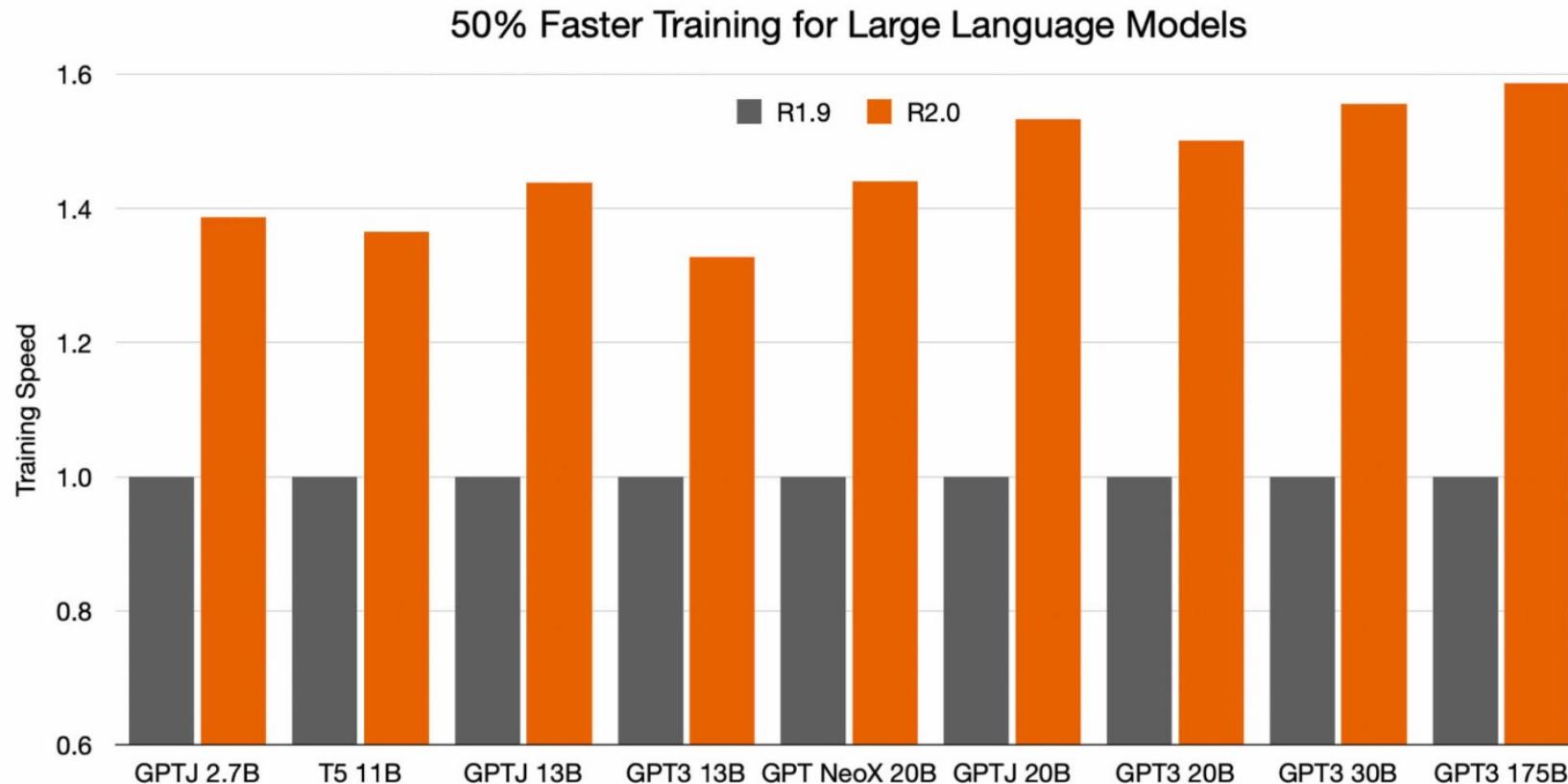
```
python run.py \  
--params params_gptneox.yaml \  
--num_csx 16 \  
--num_steps 100 \  
--model_dir model_dir \  
--mode train
```

# Demo

20B GPT NeoX with Sparsity Training  
on 16 CS-2s

# R2.0.2 Landed!

- All the demo videos were recorded in the timeframe of Release 1.9.1.
- We just released Release 2.0.2 this past week, bringing faster model training, PyTorch 2.0 support, dynamic sparsity, expanded model support, and more!



# Cerebras Documentation

[docs.cerebras.net/](https://docs.cerebras.net/)

Using Cerebras for your entire training workflow

docs.cerebras.net/

**CEREBRAS WAFER-SCALE CLUSTER (R2.0.2)**

**Get started**

Get started by setting up your Cerebras virtual environment. Try out our state-of-the-art deep learning models that reside in our Model Zoo and proceed to launch your first job in the Cerebras Wafer-Scale cluster:

- Set up a Cerebras virtual environment
- Clone the Cerebras Model Zoo
- Launch your first job on the Cerebras Wafer-Scale cluster

Familiarize yourself with the [Cerebras PyTorch package](#) and learn how to [write your own training loop](#) for your PyTorch model by using the [Cerebras PyTorch API](#). This package includes functions and data structures specifically tailored to facilitate the utilization of Cerebras Wafer-Scale cluster. By mastering this package, you'll be well-equipped to craft your own training loop for your PyTorch models.

**Beginner Tutorials**

Get accustomed with using Cerebras's Wafer-Scale to preprocess data, load data, train, and fine-tune deep neural networks using our step-by-step beginner-level instructional tutorials:

- Train and fine-tune a Large Language Model (LLM)
- Train a GPT model using Maximum Update Parametrization
- Train a UNet model for large image segmentation

**How-to Guides**

Take advantage of our advanced how-to guides that use the Cerebras PyTorch API and other Cerebras-enhanced features:

- Write a custom training loop using the Cerebras PyTorch API
- Train a GPT model using Maximum Update Parametrization
- Train a model with a large or small context window
- Fine-tune an LLM on a dataset using instructions
- Train a model with weight sparsity
- Restart a dataloader
- Port a trained and fine-tuned model to Hugging Face
- Control numerical precision level
- Deploy Cerebras models



Comprehensive tutorials

Quick “how-to” walkthroughs

# How to contact Cerebras?

- Email us at [developer@cerebras.net](mailto:developer@cerebras.net)
- Sign up for our monthly newsletter at [info.cerebras.net/subscribe](http://info.cerebras.net/subscribe)
- Join our Discord at [discord.gg/hZp5MUyw](https://discord.gg/hZp5MUyw)
- Join our Discourse at [discourse.cerebras.net/](https://discourse.cerebras.net/)
- LinkedIn - [linkedin.com/company/cerebras-systems/](https://linkedin.com/company/cerebras-systems/)
- Twitter - [twitter.com/CerebrasSystems](https://twitter.com/CerebrasSystems)



Talk to researchers and our  
ML/SDK Engineers here!



# Thank you