UNet Performance with Wafer Scale Engine (optimization case study)

Vyacheslav Romanov
NETL
U.S. Department of Energy
Pittsburgh, USA
romanov@netl.doe.gov

***Abstract*—Science-based simulations can significantly improve efficiency and effectiveness of the carbon storage operations and subsurface reservoir management and help to mitigate the impact of atmospheric carbon dioxide on climate change. However, they can be computationally very expensive, especially when inversion modeling is employed. Training robust, trustworthy, and reliable site-specific, field-scale, science-informed artificial intelligence models is also computationally and data intensive. In this work, compute acceleration of such model training was achieved by taking advantage of extreme workflow parallelism of single-wafer-embedding of the compute resources. Successful kernel mapping to system fabric and accelerated training were accomplished using a prototype UNet architecture and dynamic loss scaling.**

Keywords—AI accelerators, carbon management, UNet, image processing

I. Introduction

Advances in digital technologies accelerated the progress in science and engineering [1, 2]. However, conventional high-performance and distributed computing architectures cannot keep up with the growing demand for real-time data intensive applications [3-5]. Artificial intelligence (AI) based on machine learning (ML) can help to reduce the computational cost associated with the high-fidelity simulation techniques [5, 6]. Science-informed AI/ML models can vastly accelerate both forward and inversion modeling if the model training and fine-tuning are performed effectively and efficiently [7]. Yet, training robust and trustworthy science-informed models is also computationally and data intensive. That opens the door for AI accelerators [8] which exploit parallelism built in the hardware design. There are different ways to accomplish that (Fig. 1):

- NVIDIA's Graphics Processing unit (GPU) and spatial accelerators, e.g., Google's Tensor Processing Unit (TPU) are built to rely more on data parallelism [9, 10].
- Graphcore's Intelligence Processing Unit (IPU) and Cerebras' Wafer Scale Engine (WSE) are built for model parallelism, by exploiting efficiency in both General Matrix Multiply (GEMM) operations and inter-layer parallelism (pipeline) [11].
- Hybrid parallelism contingent on model and batch size, the overall compute architecture, and hyperparameter sets can be implemented on any hardware [12].

In this work, compute acceleration of AI/ML model training for subsurface applications was explored by taking advantage of extreme workflow parallelism of single-wafer-embedding of the compute resources [13, 14]. The use case was optimization of carbon storage operations, specifically in depleted oil fields.

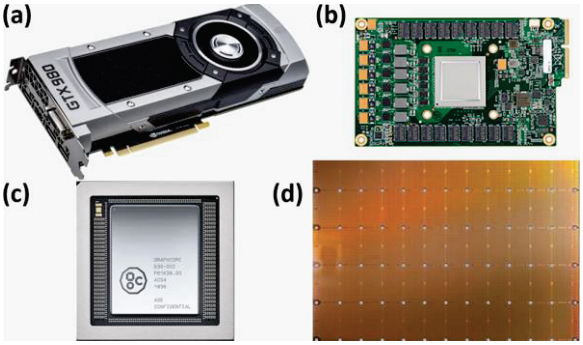

Fig. 1. AI accelerators: (a) NVIDIA's GPU, (b) Google's TPU, (c) Graphcore's IPU, and (d) Cerebras' WSE. Source: Adapted from [10, 11].

Carbon capture, utilization, and storage (CCUS) has been identified as bridge technology for the transition from fossil fuels to a sustainable energy future [15]. Large-scale deployment of this technology requires fast and accurate numerical simulations to support near-real-time operational scenario exploration as well as decision making [16]. Scurry Area Canyon Reef Operators Committee (SACROC) oilfield, located in the Permian Basin, TX, USA was selected here for the case study, because CO_2-enhanced oil recovery (CO_2-EOR) projects can provide the detailed geological and project performance data required for robust and accurate reservoir modeling [17, 18]. Selection of the geological setting was also based on availability of the simulation data for multiple geological realizations and injection scenarios, with reservoir potential to accept and retain 50 million tons of CO_2 over 50 years (during and post injection) [19].

The objective is to use deep-learning techniques for building data-driven models based on field observations and/or synthetic data, that can provide fast and accurate predictions of 3D spatio-temporal evolution of the subsurface responses to operational stimuli, for uncertainty quantification and optimization tasks.

U.S. Dept. of Energy, Office of Fossil Energy and Carbon Management

II. Data Management

A. Use Case

SACROC unit, one of the largest (about 50,000 acres) CO_2-EOR projects in the U.S., with over 55 million metric tons of CO_2 sequestered to date, is geolocated in the southeastern segment of Horseshoe Atoll within Midland Basin in western Texas [17], on the eastern edge of the larger Permian Basin. The domain of geo-cellular numerical model that generated the source data for the case study is the northern platform of SACROC, with lateral dimensions of approximately 8 × 4 km (Fig. 2). For computational expediency, the original high-resolution model with 9,450,623 (287 × 149 × 221) cells was upscaled to the coarse model with 13,600 cells (34 × 16 × 25) [19]. That area was truncated along the eastern edge and split into two square subsets for training AI/ML model. The combined area of interest (two squares in Fig. 2) contained 3 injection and 2 production wells and was stimulated with the CO_2–water alternating gas (WAG) operations for 30 years combined, discretized as 361 injection and 631 post-injection steps each year [19].

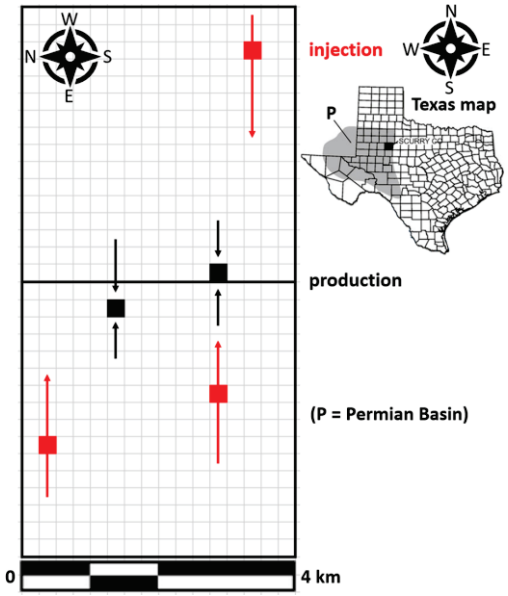

Fig. 2. Use case configuration: partition of the area of interest (length=34, width=16, depth=25) with three injection and two production wells, into two subsets. Inset: Texas map with the area's geolocation (black square).

SACROC unit is composed of shelfal bioclastic limestone and thin shale beds of late Pennsylvanian age delineated into Strawn, Canyon, and Cisco strata [20, 21]. Cisco and Canyon reef formations are responsible for oil production. Wolfcamp shale formation of lower Permian age serves as low permeability caprock above Cisco and Canyon groups. The top of Cisco and Canyon formations is at 1200 m, and the bottom is at 1400 m below sea level [22]. The overlying Permian Wolfcamp shale serves as primary seal for CO_2 storage. The underlying low-permeability Strawn formation assists in confining injected CO_2 within Cisco and Canyon formations [22]. Triassic Dockum formation above the seals houses shallow groundwater aquifers.

The simulations utilized here for AI/ML training explored variability in depositional environments and heterogeneity in terms of porosity realizations to capture geomodel uncertainty [19]. (Permeability data were empirically derived from the porosity distribution based on 3D seismic surveys and well logs but were not used in this case study.) Operational uncertainty was captured via variable injection rates. The maximum injection bottom hole pressure was set at the estimated reservoir fracture pressure of 21.24 MPa, which was used as a reference for re-scaling the AI/ML input data. Production well pressure of the source model varied based on the depth of perforation, with minimum bottom hole pressure (BHP) set at 300 kPa (~45 psi) over the initial BHP to effectively maintain near-initial reservoir pressure during CO_2 injection [19].

The fluid flow simulation data were generated with the SACROC model (the source) using CMG GEM [19] for three realizations of carbonate reef depositional environment (saline reservoir assumption) across a range of simulation parameters (injected CO_2 volume and porosity/permeability realizations) which resulted in 90 one-year data subsets total, split into 81 subsets for AI/ML model training and 9 for testing and evaluation. The simulation responses of interest were the pressure and saturation over time (4D arrays by space-time) and the water production rate at one or more monitoring wells (2D arrays by well/time). Only pressure responses were selected for the case study.

B. Static and Time-Sequence Data Preprocessing

- Due to the current hardware limitations, the maximum dimensions of input and output data arrays for AI/ML model training were 256 × 256 × 1. The framework for processing larger, multi-channel feeds are in the works, contingent on the hardware upgrades in the near future.

Accordingly, the tensors for each one-year evolution subset of 3D pressure maps were reshaped from (962, 2, 16, 16, 25) to (1024, 2, 256, 25) by padding the time dimension to a multiple of 256 and by flattening the horizonal space dimensions. The tensors were scaled to 0–255, rounded to 'uint8' data type, sliced into 256 × 256 arrays, and saved as PNG image files, i.e., 5400 (27 × 4 × 2 × 25) training images and 600 testing images for each of three geomodel variants. Additionally, similar images but without pressure evolution were generated for no-injection scenario. These images were designated as 'mask' images in AI/ML model function.

The input data (initial 3D pressure map of the area of interest, the matching horizontal layer information, the well placement and injection schedule, and the matching time quadrant) were mapped onto (256 × 256) pixel images for each time quadrant, each half-field, each depth level, and each geomodel variant, separately to match the corresponding mask images. One axis represented the flattened horizontal space dimensions. The other axis was used as proxy for compressed time (the first 128 pixels) and as depth index (the initial pressure map for all 25 depth levels and up to 25 lines reserved for the matching depth-level marker, 12 repeated initial-pressure lines for the matching depth level, and 32 lines of encoded geology values). The values for the depth marker were constant at –0.5 prior to re-scaling. The injection schedules were placed around the corresponding wells in sequential subsets of 8 (as shown in Figure 2) to fit the

compressed time allotment. The values for production wells were constant at –0.5 prior to re-scaling. The time quadrant markers (re-used as half-field markers) were placed as (32 × 32) pixel squares at the start of each matching compressed-time interval, along the left margin for the top half and the right margin for the bottom half of the area of interest (Fig.2). Subsequently, all encoded (256 × 256) array values were scaled to 0–255 and rounded to 'uint8' data type. The re-scaled input arrays were also saved as PNG image files.

III. Methods and Techniques

Data pre-processing was performed at WATT, the compute resource running on modern, bare metal, GPU accelerated servers as part of the Center for Artificial Intelligence and Machine Learning (CAML) at the U.S. Department of Energy, National Energy Technology Laboratory (NETL). Subsequent AI/ML model development and training was completed with Cerebras Software Platform (CSoft) which has two main parts:

- ML software integrated with TensorFlow and PyTorch to bring whitelisted models to CS-2 system.
- Software Development Kit (SDK) allowing users to develop custom kernels.

A. Computational Environment

Access to Cerebras CS-2 servers (cs2-1 and cs2-2, each powered by Wafer Scale Engine as the main compute engine) was available through HPE Superdome Flex version 1.6.0 (SDF) server of Neocortex, AI supercomputer at the Pittsburgh Supercomputing Center, with 32 Intel Xeon Platinum 8280L (28 cores, 56 threads each, 2.7-4.0 GHz, 38.5 MB cache) and 12 × 100 GbE (aggregate 1.2 Tbps) bandwidth interconnects to each Cerebras CS system. Neocortex is federated with Bridges-2 platform, a component in the National Science Foundation's (NSF) XSEDE cyberinfrastructure "ecosystem," which includes Ocean, the high-speed spinning-disk Lustre filesystem optimized to store large datasets.

Cerebras Graph Compiler (CGC) can input user-specified algorithms written in either Cerebras Software language (CSL) or other general-purpose languages, e.g., Python 3.7.4 (Linux) used in this study, to program for CS-2. Stack (first-in-first-out call) compilation using Cerebras intermediate representation CIR-H supports the following major tasks.

- Feature support: Building the graph representation of AI model (TensorFlow 2 framework) and matching layers and mathematical operations to kernel libraries.
- Hardware placement: Mapping kernels onto regions of CS-2 system fabric, routing kernel communication, resource allocation, configuring data routes, and creating executable.

There are two main execution modes on CS-2, pipelined and weight streaming. In the pipelined mode, the entire model is mapped onto the wafer at once, as opposed to streaming the model weights one layer at a time to accommodate for extremely large models. Alternatively, in the weight streaming mode, only the activations are resident on wafer; the weights are stored externally and streamed onto wafer to compute layer, while the gradients are streamed out of the wafer to decoupled weight-optimizer compute node.

Pre-trained model evaluation was performed using CPU within Singularity container at one of the SDF nodes (sdf-1 or sdf-2) using 14 tasks, for one chief (coordinator) and 13 workers. The task definition in Neocortex job orchestrator, Slurm was set to 28 CPU cores per task. Access to Bridges-AI GPU nodes was not available in this project.

B. Model Architecture

The computational research tracks available to the NSF grant recipients included:

- ML workflows leveraging the model(s) as-is from the Cerebras Model Zoo repository [23] or similar. Support by the Neocortex team was primarily offered for Transformer style models. Experimental 2D UNet models were supported with limited functionality.
- Domain specific programming on structured grids leveraging the Wafer scale engine Field equation Application programming interface (WFA) API [24].
- Projects leveraging Cerebras SDK.

The experimental 2D UNet model from the Model Zoo was selected a template for the case study, as UNet models appear to outperform the state-of-the-art Transformer-based networks in image processing applications [25]. The architecture consists of encoder blocks with progressively increasing number of channels and decreasing 2D feature-map dimensions, as well as decoder blocks with symmetrically decreasing number of channels and increasing 2D dimensions [26] as illustrated in Figure 3. Optionally, skip-connections can be used between the corresponding levels of encoder and decoder. Down-sampling was executed by max-pooling layer, after repeated application of two padded (using tf.image.ResizeMethod.BILINEAR) 3x3 convolutions, each followed by rectilinear (ReLU) activation; while every up-sampling step (by default via 2x2 transposed convolution layer or optionally via interpolation) that halved the number of feature channels in the expansive path, was followed by optional concatenation with the corresponding feature map from the contracting path, and then by two sets of the padded 3x3 convolutions each followed by ReLU activation.

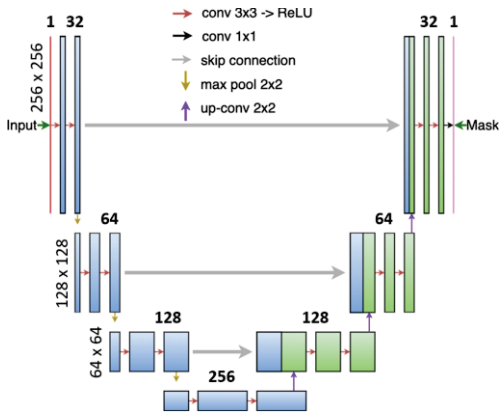

Fig. 3. Default encoder-decoder configuration (UNet).

- The default UNet architecture comprised three encoder-decoder levels, with {[32, 64, 128, 256], [128, 64, 32]} filter configuration (Fig. 3). A single-level architecture with {[32, 64], [32]} filter configuration was utilized to expedite debugging and hyperparameter tuning. In all tested instances, skip connections appeared to improve both train and test convergence; so, by default this option was set to True. The single-level model activated only 23-33% of the CS-2 fabric cores depending on the number of skip-connections and other parameters, while larger models actively needed 58-61% of the processing elements (PEs).

C. Data Flow

The images were uploaded using scp command through Ocean file system to local copy of the Model Zoo, which was initially created using rsync command. The input images were placed directly into corresponding Train and Test directories, while the mask images and text files with the matching image file names (for input and mask images) and categories (such as image class number and batching options) were placed into the corresponding Label sub-directories. Dataset class definition scripts were created for working with raw images and with tfrecords enabling high-performance input pipelines. The latter option was not used at the prototype development stage.

- Data normalization was set to "zero_centered" for input images and to "zero_one" for mask images.
- For loss computations, input and mask image data were flattened, and data types were unconditionally cast to tf.float32. For logging the image summaries, the mask image data type was cast to tf.float16 instead of tf.int32 in the original UNet template. The use of cbfloat16 was set to False.
- In input parameter specifications, the mixed-precision option was set to True to reduce memory usage. Loss scaling parameter value was changed from "dynamic" to null (NaN) in the yaml file and then converted to None. With dynamic scaling, some operations were not matched to handwritten kernels; the experimental automatic kernel compiler (DTG) was invoked in that instance but was unable to compile the multiplication broadcast scaling operations.
- The train batch size was limited to 256, while the test or evaluation batch was fixed at 32. The shuffle option was set to True, but data augmentation was opted out of because of the nature of spatio-temporal data encoding in images.

Data input pipeline was staged by the Python script that imported the dataset classes from Input directory and defined the dataset input functions for training and evaluation. Buffer size was set to tf.data.experimental.AUTOTUNE for prefetching, which resulted in different block sizes sent during the program run from sdf-1 node (1524) vs sdf-2 node (preset batch size) for very small batch sizes.

D. Optimization and Evaluation Metrics

The compute workflows available for the whitelisted UNet models (R_1.6.0) from the Cerebras Model Zoo repository only supported binary classification tasks. The only supported loss function was binary cross entropy (BCE). There were four experimental metrics available for post-training model evaluation: Mean-Intersection over Union (mIOU), Dice Similarity Coefficient (DIC), Mean Per-Class Accuracy (MPCA), and Pixel Accuracy (Acc). Only the Pixel Accuracy metric marginally provided some benefit to evaluation of AI/ML models sought in this case study, but it needed debugging to make it work. Additional experimenting was conducted to add the loss functions and evaluation metrics for regression tasks.

- The default loss function for the image regression tasks was Mean Squared Error (MSE). Direct replacement of BCE with MSE was not compatible with the batching and backpropagation workflows matched to handwritten kernels (whitelisted). So, the loss function was pushed to the output model layer added to the default UNet model.
- One of auto-compiling problems was associated with handling conditional (True/False) operations. Preset layer policies for compute data types helped to resolve the conditional precision scaling, specifically for the backpropagation task. Uniformly applied layer policies also resolved the tensor shape mismatch caused by batch processing. The reduction over the batch axis was applied using Reduction.SUM_OVER_BATCH_SIZE. Otherwise, the unreduced vector loss would be passed to the optimizer.
- Another loss function added for regression optimization was tf.compat.v1.losses.absolute_difference().
- For trained model evaluation purposes (using CPU for inferencing) tf.image.ssim(img1, img2,) was added to default metrics, where img1 and img2 tensors of the input and mask image evaluation batches were reshaped to [32, 256, 256, 1] and cast to tf.float16.

IV. Results and Discussion

UNet model is quite adaptable to re-purposing it from image classification to regression. However, due to the limited options offered in form of the whitelisted operations in the experimental model, complex base model policies, and tight precision limits, this task proved to be challenging. These challenges can be adequately addressed as demonstrated in this section.

A. Data Mapping Efficiency

The data preprocessing approach described in section II.B was a result of trial-and-error learning. Mapping tabular data onto the image frames limited to 65,536 (256 × 256) pixels was not a trivial task. Several attempts failed to generate images with the machine-readable feature encoding that would facilitate efficient training of UNet model. The key factors were feature prominence (such as dimensions vs contributor ranking) and space utilization. The ultimately successful approach (described in section II.B) was guided by domain science in assessing the feature importance, by maximizing the image space utilization, and by ensuring that important features are not diluted after three rounds of max-pool upscaling to the bottleneck's coarse feature map (32 × 32). That was the rationale behind using 8 pixels as the low-limit feature size in the image encoding (section II.B).

B. Model Training (CS-2) and Evaluation (CPU inference)

The initial training using all available data and MSE as loss function resulted in divergence (Fig. 4). The test error was on par with train error; while the structural similarity index (SSIM) initially increased as long as the root-mean-square error values decreased and then collapsed to zero at the onset of divergence.

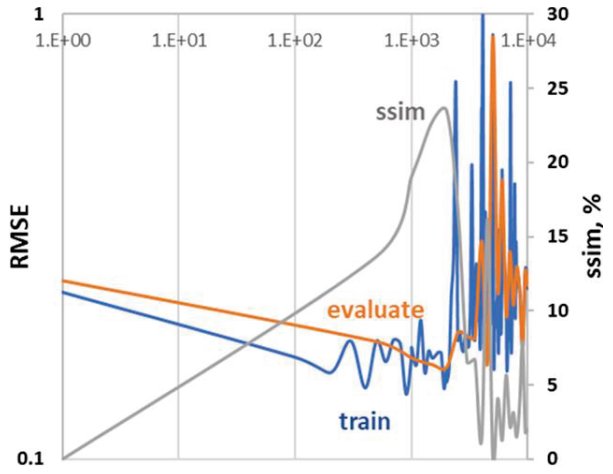

Fig. 4. Fixed-loss-scale optimization progress. Primary axis: RMSE (training and avaluation). Secondary axis: ssim (evaluation).

The root-cause analysis showed that the primary reason for divergence was heterogeneity of the source data along the depth axis. Specifically, according to the estimates, UNet model could not pick up the fractal structure of the seal formation's layers unless loss gradients for the permeable layers decreased by two orders of magnitude. At the same time, training on permeable layers alone achieved SSIM levels above 70% and RMSE of up to 0.95. Due to systemic precision limitations of CS-2 training policy (see sections III.C and IV.C) optimization with fixed-scale MSE loss function inevitably led to vanishing gradients.

C. Performance Optimization

a) Precision limitations: The code compilation for CS-2 system was only successful with mixed-precision option set to True, without activation of the dynamic-scaling policy. All that in combination with systemic half-precision limitations, results in exploding ratio of zero gradients during backpropagation and eventually training divergence, unless it is remedied through manual loss scaling. Half precision compute can be an order of magnitude faster than double precision, but it severely limits the up-/down-scaling range ($<2^{29}$).

b) Loss scaling: Attempts to alleviate the gradient scale discrepancy among the depth layers by using the pixel-based weight-averaging ran into unstable convergence. Instead, the loss scaling was manually adapted to the training progress; and MSE was replaced with absolute_difference() to alleviate the gradient scale problem further.

c) Incremental training: The supervised loss scaling with incrementally increasing number of images used in training (Table 1) ultimately succeded, without training divergence.

TABLE I. INCREMENTAL MODEL TRAINING: OPTIMIZATION STAGES

Iterations	Train configuration and performance		
	Depth channels	***Loss scale***	***Compute time*** [a]
0 – 1,000	0 and 4 with and without injection [b]	x 2^9	11.615 s
1,000 – 1,250	0 – 7 with injection	dynamic	7.984 s
1,250 – 4,000	0 – 7 with injection	x 2^9	39.371 s
4,000 – 10,000	0 – 15 with injection	x 2^9	86.255 s

[a.] Including 34 checkpoints in total (5, 11, 6, and 12 per iteration block)

[b.] Post-training evaluation performed on layers 0–4 with injection only

Other relevant CS-2 training and CPU evaluation metrics for three-level UNet (averaged over all stages listed in Table I):

- CS-2 Fabric Programming time = 149.5 s
- CS-2 (60% active PEs) Training time = 14.4 ms/step (optimization, including backpropagation)
- CS-2 Compute utilization = 30%
- CS-2 Data streaming: 8446.6 samples/s
- CPU (14 parallel tasks) Inference time = 217.8 s/step (forward model without shuffle and backpropagation)

Given the total number of cores in CS-2 (850,000) and the SDF node allocation (392) the CS-2 cores appear to run an order of magnitude more efficiently. However, the login node runs evaluations almost an order of magnitude faster than SDF nodes, with comparable number of CPUs (32). In that case, CS-2 and Bridges-2 login node compute efficiency is comparable on per-core basis, at least for the small number of samples used for training and evaluation of the prototype.

The incremental-training errors, both the error normalized per training summary (after the reduction by sum over batch size, as defined in section III.C) and evaluation MSE (using CPU and standard test batch size with train data) continued to trend down, while SSIM kept increasing (Fig. 5).

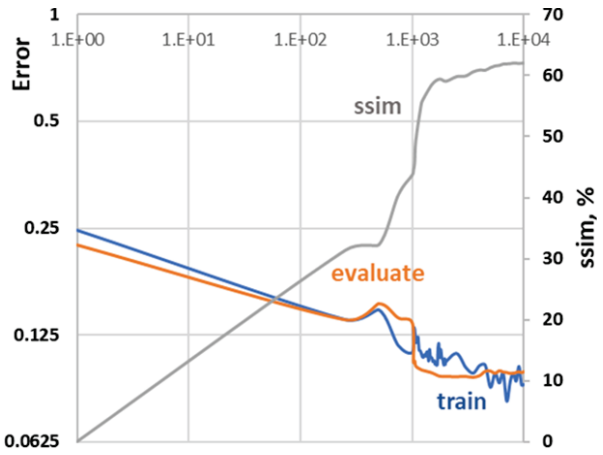

Fig. 5. Dynamic-loss-scale optimization progress. Primary axis: losses (training and avaluation). Secondary axis: ssim (evaluation).

- The model training for 10,000 steps on 60.52% of fabric cores with batch size of 1 on the same 16-image dataset, optimized scaling, loss function and hyperparameters converged within 61 seconds (net) to the same training loss and metrics values. Total compute time depended on the number of checkpoints (3 seconds/checkpoint). CS-2 fabric programming took 140 seconds.

ACKNOWLEDGMENT

Cerebras Systems and Neocortex teams at the Pittsburgh Supercomputing Center provided vital support during the early stages of setting up the computational environment under the NSF cooperative agreement. NETL research contractor support is recognized for assistance with the energy data exchange.

DISCLAIMER

This project was funded by the Department of Energy, National Energy Technology Laboratory, an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

[1] OECD, "The Digitalisation of Science, Technology and Innovation: Key Developments and Policies," OECD Publishing, Paris, February 2020. DOI: 10.1787/b9e4a2c0-en.

[2] A. Agrawal and A. Choudhary, "Perspective: Materials informatics and big data: Realization of the "fourth paradigm" of science in materials science," APL Materials, vol. 4, no. 1, 053208.1-10, January 2016.

[3] D. Zhao, et al., "FusionFS: Toward supporting data-intensive scientific applications on extreme-scale high-performance computing systems," in Proc. 2014 IEEE International Conference on Big Data, Washington, DC, pp. 61-70, January 2015. DOI: 10.1109/BigData.2014.7004214.

[4] AMD, "What Is Accelerated Computing, and Why Is It Important?" June 2023. [Online] https://www.xilinx.com/applications/adaptive-computing/

[5] Hyperion Research, "Hyperion Research HPC Market Update," November 2022. [Online]. Available: https://hyperionresearch.com/wp-content/uploads/2022/11/Hyperion-Research-SC22-HPC-Market_Combined-1.pdf. [Accessed June 2023].

[6] S. Ezell, "A New Frontier: Sustaining U.S. High-Performance Computing Leadership in an Exascale Era," Information Technology and Innovation Foundation (ITIF), September 2022. [Online]. Available: https://itif.org/. [Accessed June 2023].

[7] V. Romanov, "Convoluted Filtering for Process Cycle Modeling" Engineering Reports, e12657 (Early View), March 2023. [Online]. DOI: 10.1002/eng2.12657.

[8] Synopsys, "What is an AI Accelerator?" [Online]. Available: https://www.synopsys.com/ai/. [Accessed 20 May 2023].

[9] Saturn Cloud, "How to Run Identical Model on Multiple GPUs and Send Different User Data to Each GPU," June 2023. [Online]. Available: https://saturncloud.io/blog/how-to-run-identical-model-on-multiple-gpus-and-send-different-user-data-to-each-gpu/.

[10] K. Sato and C. Young, "An in-depth look at Google's first Tensor Processing Unit (TPU)," Google, May 2017. [Online]. Available: https://cloud.google.com/.

[11] M. Khairy, "TPU vs GPU vs Cerebras vs Graphcore: A Fair Comparison between ML Hardware," Medium, July 2020. [Online]. Available: https://khairy2011.medium.com/. [Accessed June 2023].

[12] S. Kumar, et al., "Exploring the limits of Concurrency in ML Training on Google TPUs," arXiv: 2011.03641v3 [cs.LG], May 2021.

[13] R. Sai, M. Jacquelin, F.P. Hamon, M. Araya-Polo, and R.R. Settgast, "Massively Distributed Finite-Volume Flux Computation," arXiv, abs/2304.11274, April 2023.

[14] P.A. Buitrago and N.A. Nystrom, "Neocortex and Bridges-2: A High Performance AI+HPC Ecosystem for Science, Discovery, and Societal Good," in Nesmachnow, S., Castro, H., Tchernykh, A., editors, High Performance Computing. CARLA 2020. Communications in Computer and Information Science, vol. 1327. Springer, Cham, February 2021. DOI: 10.1007/978-3-030-68035-0_15.

[15] T. Garrish and F. Birol, "Accelerating CCUS: New opportunities for deployment (Chair's Summary)," in Proc. IEA Ministerial Meeting 2019, Paris, December 2019.

[16] V. Nuñez-Lópeza and G.-E. Ramón, "A Sustainable Approach to Decision-Making in CCUS Systems," in Proc. 14th International Conference on Greenhouse Gas Control Technologies (GHGT-14), Melbourne, Australia, October 2018.

[17] V. Gholami, S.D. Mohaghegh, and M. Maysami, "Smart Proxy Modeling of SACROC CO_2-EOR," Fluids, vol. 4, no. 2, 85, May 2019.

[18] K.D. Romanak, R.C. Smyth, C. Yang, S.D. Hovorka, M. Rearick, and J. Lu, "Sensitivity of groundwater systems to CO_2: Application of a site-specific analysis of carbonate monitoring parameters at the SACROC CO_2-enhanced oil field," Int. J. Greenh. Gas Con., vol. 6, pp. 268–279, January 2012.

[19] V. Romanov, et al., Science-informed Machine Learning to Accelerate Real Time (SMART) Decisions Initiative—Storage FWP 1022462 – Phase 1 Final Summary Report. Pittsburgh, PA: U.S. Department of Energy, National Energy Technology Laboratory, February 2023.

[20] E.L. Vest, Jr., "Oil Fields of Pennsylvanian—Permian Horseshoe Atoll, West Texas," in Halbouty, M.T., editor, Geology of Giant Petroleum Fields: AAPG Memoir, vol. 14, pp. 185–203, January 1970.

[21] M.A. Ranies, J.K. Dobitz, and S.C. Wehner, "A review of the Pennsylvanian SACROC Unit," in Viveros, J.J., and Ingram, S.M., editors, The Permian Basin: Microns to Satellites, Looking for Oil and Gas at All Scales: West Texas Geological Society, no. 2001-110, pp. 67–74, October 2001.

[22] W.S. Han, B.J. McPherson, P.C. Lichtner, and F.P. Wang, "Evaluation of trapping mechanisms in geologic CO_2 sequestration: Case study of SACROC northern platform, a 35-year CO_2 injection site," American J. Sci., vol. 310, no. 4, pp. 282–324, April 2010.

[23] bhargav-cerebras. Cerebras "Model Zoo" (original repository). Available: https://github.com/Cerebras/modelzoo. [Accessed May 2023].

[24] M. Woo, T. Jordan, R. Schreiber, I. Sharapov, S. Muhammad, A. Koneru, M. James, and D. Van Essendelft, "Disruptive Changes in Field Equation Modeling: A Simple Interface for Wafer Scale Engines," arXiv:2209.13768 [cs.DC], September 2022.

[25] X. Jia, J. Bartlett, T. Zhang, W. Lu, Z. Qiu, and J. Duan, "U-Net vs Transformer: Is U-Net Outdated in Medical Image Registration?" arXiv:2208.04939 [eess.IV], August 2022.

[26] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, vol. 9351, pp. 234–241, November 2015.