Scalable training of graph convolutional neural networks for fast and accurate predictions of HOMO-LUMO gap in molecules

Jong Youl Choi[1*], Pei Zhang[2], Kshitij Mehta[1], Andrew Blanchard[2] and Massimiliano Lupo Pasini[2]

Abstract

Graph Convolutional Neural Network (GCNN) is a popular class of deep learning (DL) models in material science to predict material properties from the graph representation of molecular structures. Training an accurate and comprehensive GCNN surrogate for molecular design requires large-scale graph datasets and is usually a time-consuming process. Recent advances in GPUs and distributed computing open a path to reduce the computational cost for GCNN training effectively. However, efficient utilization of high performance computing (HPC) resources for training requires simultaneously optimizing large-scale data management and scalable stochastic batched optimization techniques. In this work, we focus on building GCNN models on HPC systems to predict material properties of millions of molecules. We use HydraGNN, our in-house library for large-scale GCNN training, leveraging *distributed data parallelism* in PyTorch. We use ADIOS, a high-performance data management framework for efficient storage and reading of large molecular graph data. We perform parallel training on two open-source large-scale graph datasets to build a GCNN predictor for an important quantum property known as the HOMO-LUMO gap. We measure the scalability, accuracy, and convergence of our approach on two DOE supercomputers: the Summit supercomputer at the Oak Ridge Leadership Computing Facility (OLCF) and the Perlmutter system at the National Energy Research Scientific Computing Center (NERSC). We present our experimental results with HydraGNN showing (i) reduction of data loading time up to 4.2 times compared with a conventional method and (ii) linear scaling performance for training up to 1024 GPUs on both Summit and Perlmutter.

Keywords: Graph neural networks, Distributed data parallelism, Surrogate models, Atomic modeling, Molecular dynamics, HOMO-LUMO gap

Introduction

Drug discovery and molecular design rely heavily on predicting material properties directly from their atomic structure. A particular property of interest for molecular design is the energy gap between the highest occupied molecular orbital (HOMO) and lowest unoccupied molecula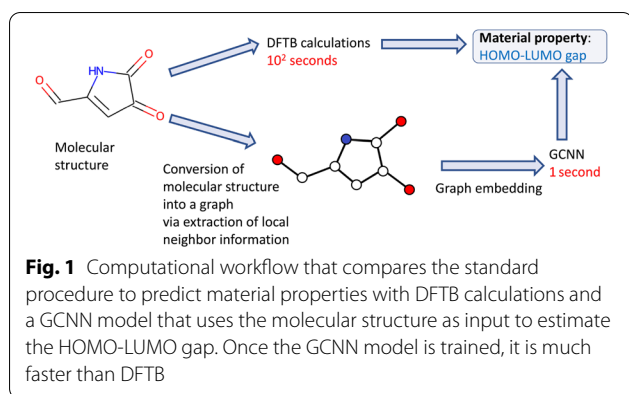r orbital (LUMO), known as the HOMO-LUMO gap. The HOMO-LUMO gap is a valid approximation for the lowest excitation energy of a molecule and is used to express its chemical reactivity. In particular, molecules that are more chemically reactive are characterized by a lower HOMO-LUMO gap. There are many physics-based computational approaches to compute the HOMO-LUMO gap of a molecule such as *ab initio* molecular dynamics (MD) [1, 2] and density-functional tight-binding (DFTB) [3]. While these methods have been instrumental in predictive materials science, they are extremely computationally expensive. The advent of deep learning (DL) models has provided alternative methodologies to

*Correspondence: choij@ornl.gov

[1] Computer Science and Mathematics Division, Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37831, USA
Full list of author information is available at the end of the article

Choi *et al. Journal of Cheminformatics*   (2022) 14:70

Page 2 of 10

Fig. 1 Computational workflow that compares the standard procedure to predict material properties with DFTB calculations and a GCNN model that uses the molecular structure as input to estimate the HOMO-LUMO gap. Once the GCNN model is trained, it is much faster than DFTB

produce fast and accurate predictions of material properties and hence enable rapid screening in the large search space to select material candidates with desirable properties [4–7].

In particular, graph convolutional neural network (GCNN) models are extensively used in material science to predict material properties from atomic information [8, 9]. When GCNN models are used as surrogates in screening of the vast chemical space, the models have to process large amounts of streaming data which is dynamically produced by molecular design applications [10] (Fig. 1).

To effectively process large volumes of data in training large complex GCNN models, both data loading and model training must scale on multi-node hybrid CPU-GPU high-performance computing (HPC) resources. HPC techniques to scale the training use distributed data parallelism (DDP) to distribute data in batches across different processes. Each process computes gradient updates for the coefficients of the DL model on the local batch, and combines local gradient updates of all processes by averaging them. Although DDP is a well established technique, the specific use of DDP to scale the training of GCNN models is still largely unexplored.

In this work we analyze the scalability of our library of GCNN models on two open-source graph datasets describing the HOMO-LUMO gap for a wide variety of molecules: the PCQM4Mv2 dataset from the Open Graph Benchmark (OGB) [11, 12] and the AISD HOMO-LUMO dataset [13] generated at Oak Ridge National Laboratory (ORNL). The scalability of the data loading and GCNN training on the two datasets has been tested on the Summit supercomputer at the Oak Ridge Leadership Computing Facility (OLCF) and the Perlmutter supercomputer at the National Energy Research Scientific Computing Center (NERSC). For our study, we use HydraGNN, a library we have developed for scalable data reading and GCNN training with portability on a broad

variety of computational resources [14]. HydraGNN is capable of multitask prediction of hybrid node-level and graph-level properties in large batches of graphs in different sizes, i.e., with varying number of nodes. We use the ADIOS [15] high-performance data management library for efficient storage and reading of graph data. Numerical results show that the training of HydraGNN scales linearly up to 1024 GPUs on both supercomputers.

The remainder of this work is structured as follows. "Related work" section describes related work. "Background" section presents the background of GCNN architectures, the HydraGNN library and the ADIOS data management framework. "Distributed data parallel training" section briefly describes our DDP approach along with efficient data loading of training data. "Numerical results" section presents the numerical results with comparisons between Summit and Perlmutter in terms of scalability, and "Conclusions and future work" section summarizes the analysis of this work and discusses future research directions.
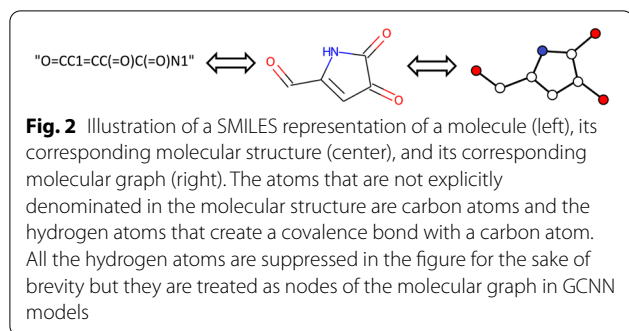
Related work

In this section, we review the relevant literature on GCNN models for predicting the HOMO-LUMO gap and DDP for GCNN.

GCNN modeling work has been reported on predictions of HOMO-LUMO gap [9, 16–19]. Most of the work focuses on the QM9 dataset [20] or the OE62 dataset [21]. QM9 has about 134 thousand molecules containing 5 element types (i.e., H,C,N,O and F), and OE62 covers 16 different elements types (i.e., H, Li, B, C, N, O, F, Si, P, S, Cl, As, Se, Br, Te, and I) and about 62 thousand molecules. The two datasets used in this work are significantly larger and more diverse and hence more challenging to process and predict—PCQM4Mv2 with 3.3 million molecules and 31 element types and AISD HOMO-LUMO with 10.5 million molecules and 6 element types. Some GCNN work [12, 22, 23] has been reported on PCQM4Mv2, but the implementations are not for large-scale distributed training.

With respect to distributed training of GCNN models, recent work surveyed different types of potential parallelization techniques, including DDP [24]. While most of the research has focused on distributing the GCNN training with graph partitioning techniques to process large-scale graphs, no specific contribution in the literature has analyzed the scalability of GCNN training using DDP. In this paper, we focus on exploring the challenges and demonstrating the capability to process millions of graphs with DDP at scale.

For efficient storage and loading of large datasets, we use the ADIOS [15] data management library, which is

Fig. 2 Illustration of a SMILES representation of a molecule (left), its corresponding molecular structure (center), and its corresponding molecular graph (right). The atoms that are not explicitly denominated in the molecular structure are carbon atoms and the hydrogen atoms that create a covalence bond with a carbon atom. All the hydrogen atoms are suppressed in the figure for the sake of brevity but they are treated as nodes of the molecular graph in GCNN models

Fig. 3 HydraGNN workflow to process molecular graph dataset. It performs iterative training phases, consisting of data loading, forward, backward, and optimization steps

commonly used for science applications. ADIOS provides a self-describing data format built upon a publish-subscribe framework. Other scientific data formats include the Hierarchical Data Format (HDF5) [25], but we selected ADIOS for its proven performance at extreme scale, in addition to the plethora of options for tuning I/O and streaming data, along with inbuilt support for data compression.

Background

In this section, we discuss our use of GCNN for the prediction of HOMO-LUMO gap in molecules. We describe the architecture of HydraGNN, our library of GCNNs. We discuss the design of the large-scale data loading in HydraGNN leveraging ADIOS, a high-performance scientific data management library for managing large-scale I/O and data transfers.

Graph convolutional neural networks

GCNNs [26, 27] are DL models for processing graph data. Representing molecules in the form of graphs is natural since the atoms can be viewed as nodes and chemical bonds as edges of the graph, as shown in Fig. 2. Nodes in the graph retain atomic features in molecules and edges retain the connectivity and bond properties (e.g., the distance between nodes). Each graph can have graph-level properties such as the HOMO-LUMO gap.

Graph convolutional (GC) layers are the core of GCNNs. They employ a message-passing framework, a procedure that combines the knowledge from neighboring nodes. The type of information passed through a graph structure can be either related to the topology of the graph or the nodal features. An example of topological information is the node degree, whereas an example of nodal feature in the context of this work is the atomic number. The message passing in a single GC layer in our applications maps directly to the pairwise interactions of an atom with its neighbors. Through consecutive steps of message passing (i.e., stacking multiple GC layers together), the graph nodes gather information from
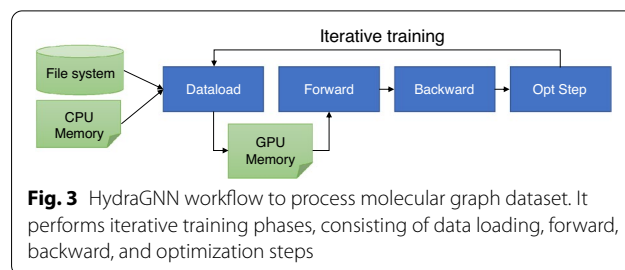
nodes that are further and further away, which implicitly represents many-body interactions.

A graph pooling layer is connected at the end of a stack of consecutive GC layers to gather feature information from the entire graph in prediction tasks of graph-level properties. It aims at aggregating the nodal feature associated with each atom across a graph into a single feature. In our work, we use global mean pooling layer, which averages node features across all the nodes in the graph. For atom(node)-level properties such as the atomic charge transfer and atomic magnetic moment, aggregating the information from all atoms into a global feature is not needed. Finally, fully connected (FC) layers take the results of pooling, i.e., extracted features, and provide the output prediction for global properties.

Differing in the policy adopted to aggregate, transfer, and update information through the message passing in GC layers, a variety of GCNNs have been developed, e.g., Principal Neighborhood Aggregation (PNA) [28], Crystal GCNN (CGCNN) [8] and GraphSAGE [29]. Many of them have been implemented in HydraGNN [30].

HydraGNN

HydraGNN is our in-house library for performant creation and testing of various GCNN models, and is designed to perform multi-task predictions of graph data. It is built on Pytorch [31, 32] and Pytorch Geometric [33, 34], and can run on small-scale workstations to large-scale HPC systems. It is openly available on Github [30].

HydraGNN loads molecular data encoded as graph structures consisting of a list of nodes (atoms) and edges (bonds) either from a file system or directly from memory. Training is performed over multiple iterations, where each iteration consists of a forward, backward, and optimization step as shown in Fig. 3. The forward phase computes the model output tensor from an input graph with a forward function consisting of GC layers, a global pooling layer, and FC layers.

It calculates the loss between the model output and the true tensor values corresponding to the input graph as the mean square error (MSE). The backward phase calculates the gradients of the loss with respect to each parameter in forward function. Finally, the optimization step updates parameters based on gradients calculated in the backward step and a user-defined optimization policy.

ADIOS data management framework

ADIOS is an open source, high-performance, I/O framework developed as part of the Exascale Computing Project (ECP).[1] It provides a custom, self-describing data format with optimized methods to read and write data for massively parallel applications. ADIOS's focus is to provide extreme-scale I/O capabilities on the world's largest supercomputers; it is used in science applications that generate data on the order of several petabytes. ADIOS is run in production in many HPC codes such as XGC [35], GENE [36], GEM [37], PIConGPU [38], WarpX [39], E3SM [40], LAMMPS [41], and others, providing over 1 Terabyte/second of I/O to the Summit GPFS file system [42] at ORNL.

At its core, ADIOS provides a concept of abstract "Engine". Engines execute the I/O heavy tasks and are conceived as workflows tackling specific applications' needs optimized for HPC filesystems, hierachical data management, or data access over the wide area network. We leverage ADIOS's optimized I/O writing and reading performance on HPC file systems in the presence of multiple concurrent processes, which is necessary for performing large-scale distributed data-parallel training for HydraGNN.

Besides the performance boost, ADIOS provides a scalable yet flexible data format that can manage millions of graphs with varying sizes. ADIOS stores data in a custom format termed BP (binary-packed). It is a highly efficient self-describing format for storing data along with metadata from various sources in a distributed application. An ADIOS file consists of variables and attributes, where variables can be scalars or arrays. Array variables can be local to a process or can be global arrays that are distributed amongst processes. Global arrays are used to construct large data structures that can be written and read efficiently in a parallel fashion. For example, edge attributes of all graphs in HydraGNN are stored as part of a single global array for fast storage and retrieval.

An ADIOS file is a container of one or more subfiles, with one subfile per writer. The number of writers for a group of ADIOS variables is configurable so that a user may tune the number of subfiles created in the ADIOS
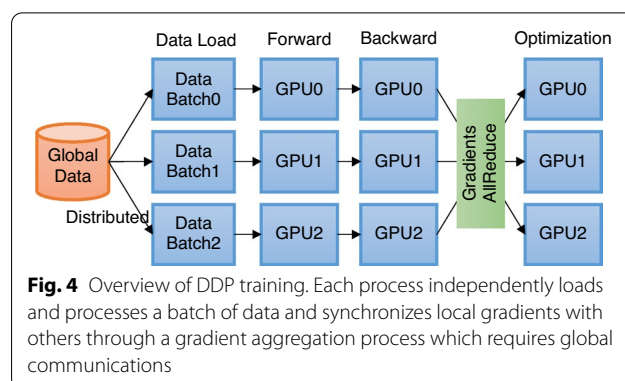

Fig. 4 Overview of DDP training. Each process independently loads and processes a batch of data and synchronizes local gradients with others through a gradient aggregation process which requires global communications

container. Internally, ADIOS transparently converts this N→M I/O pattern such that data from N processes is aggregated and output by M processes, where $1 \leq M \leq N$. At extreme scale where applications spawn hundreds of thousands of processes, this leads to significant performance benefits as users can avoid overwhelming the file system by creating a large number of subfiles. Several other options are available in the ADIOS library to further tune and optimize I/O performance.

Distributed data parallel training

Large supercomputers available at DOE national laboratories such as Summit and Perlmutter allow us to process millions of molecules (graph objects) concurrently to build and test various GCNN models. To explore the high degree of parallelism available on these nodes, we apply DDP to accelerate the training process in HydraGNN.

In deep learning, DDP refers to a method of performing training in parallel by distributing the same model across multiple nodes but assigning disjoint datasets to consume for each model. In other words, each process computes gradients of parameters in parallel for its assigned dataset. At the end of the computation, we aggregate gradients and re-distribute them to all processes to make each process maintain the same model parameters (Fig. 4). In general, gradients aggregation is an expensive operation in parallel computing. For Summit and Perlmutter, both equipped with NVIDIA GPUs, we use NCCL, an NVIDIA library to communicate directly between GPUs, without having to copy data to the host CPU first, which results in an efficient and inexpensive inter-process communication for gradient aggregation. The communication overhead will vary depending on the models' size and the frequency of aggregation.

Another challenge in DDP training in an HPC system is data management, including pre-processing and data loading during training. Each process regularly reads a group of data objects from the storage to form a batch of graph objects for training. The random I/O access

Choi *et al. Journal of Cheminformatics*   (2022) 14:70

Page 5 of 10

Table 1 ADIOS schema for graph dataset

Variable	Description	Array shape
x	Features associated to nodes	(#nodes, #node features)
edge index	Edge connectivity between nodes	(#2, #edges)
edge attr	Features associated to edges	(#edges, #edge features)
y	Target features in graph-level or node-level	(#target, #features)

pattern is common to maintain shuffled and disjointed subsets with other processes to avoid duplication during training and increase the fairness of training. However, without careful data management, loading training data can become a bottleneck for training. To mitigate this issue, we leverage ADIOS [15] for our distributed GCNN training.

We emphasize here the importance of pre-processing. Many open-source molecular databases use the simplified molecular-input line-entry system (SMILES) [43], the de facto standard format to represent 2D molecular structures in text strings (see Fig. 2 for an example). HydraGNN employs a pre-processing step to convert SMILES strings to graph representations in files. Depending on the scale of the input data (graph), an optimal pre-processing step for graph conversion is crucial to the performance. Without pre-processing, data loading has to perform conversion of SMILES data into a graph which can cause a bottleneck in training. To avoid this, our workflow includes a parallel pre-processing step in which graph data generated from SMILES strings is stored into the ADIOS high-performance data format, and is later read back for training. This avoids converting the SMILES data into graph objects for every iteration of the training.

To this end, we develop the ADIOS schema for graph datasets. The schema describes the mapping of graph objects to ADIOS components. We aggregate each graph attribute into a global multi-dimensional array and save it in ADIOS variable format. The main graph attributes are node features, edge attributes, and edge index, as summarized in Table 1.

We have developed an extensible data loader module in HydraGNN that allows reading data from different storage formats. In this work, we evaluate the following three methods for loading data from training datasets.

- Inline data loading: load SMILES strings written in CSV format into memory and then convert each SMILES into a graph object at every batch data loading. It has the smallest memory footprint.
- Object data loading: convert all SMILES strings into graph objects and export them in a serialized format (e.g., Pickle) during a preprocessing phase. A process loads each data batch directly from the file system and unpacks into a memory.
- ADIOS data loading: convert SMILES strings into graph objects mapped to ADIOS variables, and write them into an ADIOS file in a pre-processing step. Each process then loads its batch data during training in parallel along with other processes.

We will discuss performance comparisons in the next section.

Numerical results

In this section, we assess our development of DDP training in HydraGNN on two state-of-the-art DOE supercomputers, Summit and Perlmutter. We discuss the scalability of our approach, and compare the performance of different I/O backends for storing and reading graph data.

Setup

We perform our evaluation on two supercomputers of DOE. Both systems provide state-of-the-art GPU-based heterogeneous architectures.

Summit is a supercomputer at OLCF, one of DOE's Leadership Computing Facilities (LCFs). Summit consists of about 4600 compute nodes. Each node has a hybrid architecture containing two IBM POWER9 CPUs and six NVIDIA Volta GPUs connected with NVIDIA's high-speed NVLink. Each node contains 512 GB of DDR4 memory for CPUs and 96 GB of High Bandwidth Memory (HBM2) for GPUs. Summit nodes are connected in a non-blocking fat-tree topology using a dual-rail Mellanox EDR InfiniBand interconnection.

Perlmutter is a supercomputer at NERSC. Perlmutter consists of about 3000 CPU-only nodes and 1500 GPU-accelerated nodes. We use only the GPU-accelerated nodes in our work. Each GPU-accelerated node has an AMD EPYC 7763 (Milan) processor and four NVIDIA Ampere A100 GPUs connected to each other with NVLink-3. Each GPU node has 256 GB of DDR4 memory and 40 GB HBM2 per each GPU. All nodes in Perlmutter are connected with the HPE Cray Slingshot interconnect.

We demonstrate the performance of HydraGNN using two large-scale datasets, a previously

Table 2 Dataset description

Dataset	Graph size				Data file size (GB)		
	#Graphs	#Nodes	#Edges	#Avg[1]	CSV	Pickle	ADIOS
PCQM4Mv2	3.6 M	105.8 M	214.6 M	29.4	0.16	34	22
AISD HOMO-LUMO	10.5 M	550.6 M	1.1 B	52.4	0.88	94	60

[1] Average number of nodes (atoms) per graph

published benchmark dataset for graph-based learning (PCQM4Mv2) [11, 44] and a custom dataset generated for this work (AISD HOMO-LUMO) [13]. For both datasets, molecule information is provided as SMILES strings. The PCQM4Mv2 consists of HOMO-LUMO gap values for about 3.3 million molecules. In total, 31 different types of atoms (i.e., H, B, C, N, O, F, Si, P, S, Cl, Ca,Ge,As, Se, Br, I, Mg, Ti, Ga, Zn, Ar, Be, He, Al, Kr, V, Na, Li, Cu, Ne, Ni) are involved in the dataset. The custom AISD HOMO-LUMO dataset was generated using molecular structures from previous work [45]. It is a collection of approximately 10.5 million molecules and contains 6 element types (i.e., H, C, N, O, S, and F).

For scalability tests, we use HydraGNN with 6 PNA [28] convolutional layers and 55 neurons per PNA layer. The model is trained by using the AdamW method [46] with a learning rate of 0.001, local batch size of 128, and maximum epochs set to 3. The training set for each of the NN represents 94% of the total dataset; the validation and test sets each represent $1/3^{rd}$ and $2/3^{rd}$ parts respectively of the remaining data. For the error convergence tests, the HydraGNN model uses 200 neurons per layer.

Scalability of DDP

We perform DDP training with HydraGNN for PCQM4Mv2 and AISD data on Summit at ORNL and Perlmutter at NERSC using multiple CPUs and GPUs. The number of graphs and size in the datasets are summarized in Table 2.

We measure the total training time for PCQM4Mv2 and AISD HOMO-LUMO datasets over three epochs. As discussed previously, each training consists of a data loading phase, followed by forward calculation, backward calculation, and optimizer update. We test the scalability of DDP by varying the number of nodes on each system, ranging from a single node up to 256 nodes on Summit and 128 nodes on Perlmutter, corresponding to using 1536 Volta GPUs and 512 A100 GPUs respectively. Figure 5 shows the result. The scaling plot (top) shows the averaged training time for PCQM4Mv2 and AISD HOMO-LUMO on each system with a varying number of nodes, and the detailed timings of each sub-function during the training on Summit are shown at the bottom.

We obtain near-linear scaling up to 1024 GPUs for both PCQM4Mv2 and AISD HOMO-LUMO data. As we further scale the workflow on Summit, the number of batches *per GPU* decreases, leading to sub-optimal utilization of GPU resources. As a result, we see a drop in speedup as we scale beyond 1024 GPUs on Summit. We expect similar scaling behavior on Perlmutter, but we were limited to using 128 nodes (i.e., 512 GPUs) for this work.
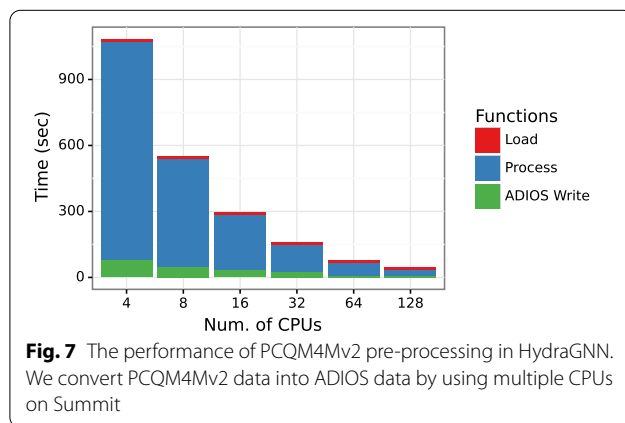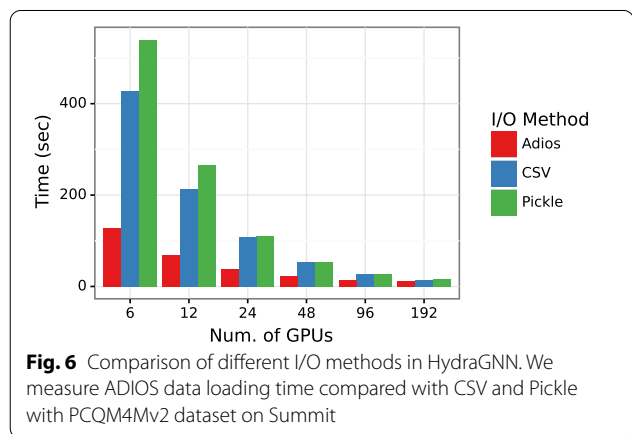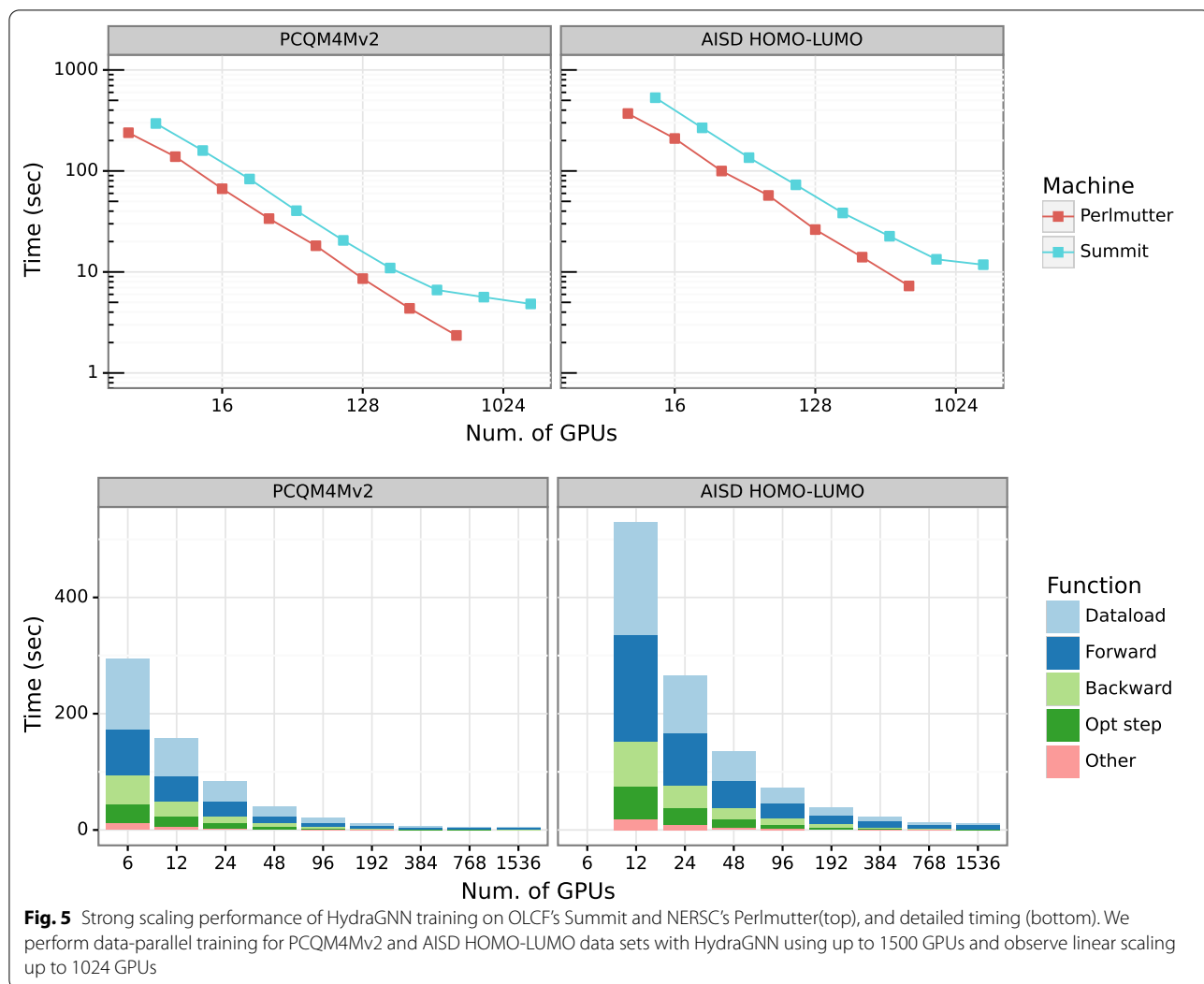
Comparing different I/O backends

Data loading takes a significant amount of time in training, as shown in Fig. 5, and hence is a crucial step in the overall workflow. We compare three different data loading methods—inline, object loading, and ADIOS data loading, as discussed in "Distributed data parallel training" section. Figure 6 presents time taken by the three methods for the PCQM4Mv2 data set on Summit. As expected, it outperforms the CSV data loading test case in which SMILES data is converted into a graph object for every molecule. ADIOS outperforms Pickle-based data loading by 4.2x on a single Summit node and 1.5x on 32 Summit nodes. To provide a complete picture of HydaGNN's data processing, Fig. 7 shows the pre-processing performance in converting the PCQM4Mv2 data into ADIOS2 on Summit. ADIOS supports parallel writing and shows scalable performance as we add CPUs in parallel.

Accuracy

Next, we perform long-running HydraGNN training for the HOMO-LUMO gap prediction with PCQM4Mv2 and AISD HOMO-LUMO datasets until training converges.

Figures 8 and 9 show the prediction results for PCQM4Mv2 and AISD HOMO-LUMO datasets respectively. With PCQM4Mv2, we achieve a prediction error of around 0.10 and 0.12 eV, measured in mean absolute error (MAE), for the training and validation set, respectively. We note that PCQM4Mv2 is a public dataset released without test data for the purpose of maintaining the OGB-LSC Leaderboards[2]. The reported validation MAE from multiple models varies from 0.0857 to 0.1760
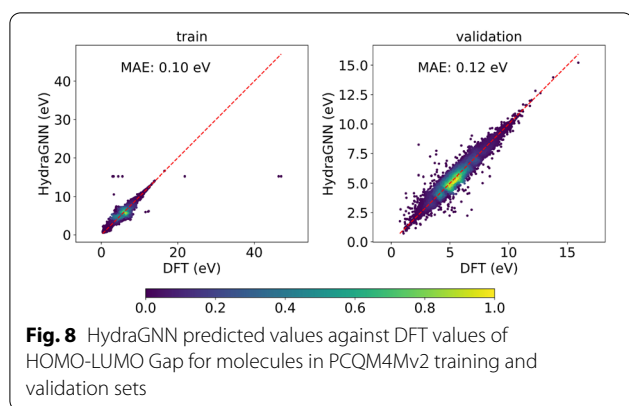
---

[2] https://ogb.stanford.edu/docs/lsc/leaderboards/

Fig. 5 Strong scaling performance of HydraGNN training on OLCF's Summit and NERSC's Perlmutter(top), and detailed timing (bottom). We perform data-parallel training for PCQM4Mv2 and AISD HOMO-LUMO data sets with HydraGNN using up to 1500 GPUs and observe linear scaling up to 1024 GPUs

Fig. 6 Comparison of different I/O methods in HydraGNN. We measure ADIOS data loading time compared with CSV and Pickle with PCQM4Mv2 dataset on Summit

Fig. 7 The performance of PCQM4Mv2 pre-processing in HydraGNN. We convert PCQM4Mv2 data into ADIOS data by using multiple CPUs on Summit

Choi *et al. Journal of Cheminformatics*   (2022) 14:70

Page 8 of 10

Fig. 8 HydraGNN predicted values against DFT values of HOMO-LUMO Gap for molecules in PCQM4Mv2 training and validation sets
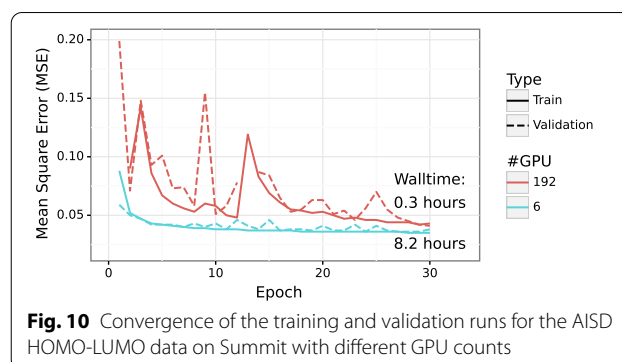
Fig. 10 Convergence of the training and validation runs for the AISD HOMO-LUMO data on Summit with different GPU counts

eV on the Leaderboard. The validation error of 0.12 eV in this work is within the accepted range. As for the AISD HOMO-LUMO dataset, it contains almost thrice as many molecules as the PCQM4Mv2 dataset. The MAE errors for training, validation, and test sets are 0.14 eV, which is similar to the PCQM4Mv2 dataset. Figure 10 shows the accuracy convergence on the AISD HOMO-LUMO dataset using different numbers of Summit GPUs. It shows that the HydraGNN training with 192 GPUs quickly converged in 0.3 h (wall time) to the similar accuracy level achieved by the 6 GPUs (a single Summit node) that took about 8.2 h.

We highlight that the convergence of the distributed GCNN training with 192 GPUs is deteriorated compared to the distributed training with only 6 GPUs. This is due to a well known numerical artifact that destabilizes the training of DL models at large scale and causes a performance drop because large scale DDP training is mathematically equivalent to large-batch training. In fact, processing data in large batches significantly reduces the stochastic oscillations of the stochastic optimizer used for DL training, thus making the DL training more likely to be trapped in steep local minima, which adversely affect generalization. Although the final accuracy of the

GCNN training with 192 GPUs is slightly worse than the one obtained using 6 GPUs for training, we emphasize the significant advantage that HPC resources provide in speeding-up the training. Better accuracy can be obtained when training DL models at large scale by adaptively tuning the learning rate [47, 48] or by applying quasi-Newton accelerations [49], but this goes beyond the focus of our current work.

Conclusions and future work

In this paper, we present a computational workflow that performs DDP training to predict the HOMO-LUMO gap of molecules. We have implemented DDP in HydraGNN, a GCNN library developed at ORNL, which can utilize heterogeneous computing resources including CPUs and GPUs. For efficient storage and loading of large molecular data, we use the ADIOS high-performance data management framework. ADIOS helps reduce the storage footprint of large-scale graph structures as compared with commonly used methods, and provides an easy way to efficiently load data and distribute them amongst processes. We have conducted studies using two molecular datasets on the OLCF's Summit and NERSC's Perlmutter
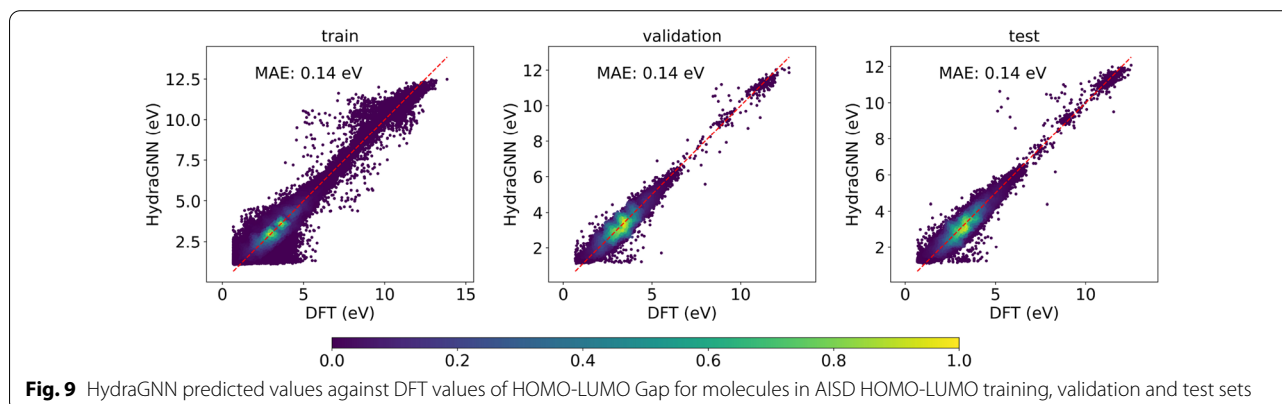

Fig. 9 HydraGNN predicted values against DFT values of HOMO-LUMO Gap for molecules in AISD HOMO-LUMO training, validation and test sets

Choi *et al. Journal of Cheminformatics*   (2022) 14:70

Page 9 of 10

supercomputers. Our results show the near-linear scaling of HydraGNN for the test datasets up to 1024 GPUs. Additionally, we present the accuracy and convergence behavior of the distributed training with increasing number of GPUs.

Through efficiently managing large-scale datasets and training in parallel, HydraGNN provides an effective surrogate model for accurate and rapid screening of large chemical spaces for molecular design. Future work will be dedicated to integrating the scalable DDP training of HydraGNN in a computational workflow to perform molecular design.

Author contributions
All authors contributed to the paper's conception and design. JYC and KM developed the ADIOS schema, and designed the experiments for evaluating different data loaders in HydraGNN. JYC implemented the ADIOS and Pickle backends, performed experiments on Summit and Perlmutter, and analyzed results. PZ contributed to the development of HydraGNN, results analysis and paper writing. AB performed material preparation, data collection and analysis. MLP contributed to the development of core capabilities in HydraGNN and wrote the main narrative of the manuscript. All authors contributed by commenting on the previous versions of the manuscript and revisions. All authors read and approved the final manuscript.

Availability of data and materials
The AISD HOMO-LUMO dataset has been generated and analysed for this work. It is open-source and accessible at the following URL https://doi.ccs.ornl.gov/ui/doi/394. Our main code HydraGNN is openly available on Github: https://github.com/ORNL/HydraGNN

Declarations

Competing interests
The authors have no competing interests to declare that are relevant to the content of this article.

Author details
[1] Computer Science and Mathematics Division, Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37831, USA. [2] Computational Sciences and Engineering Division, Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37831, USA.

References
1. Car R, Parrinello M (1985) Unified approach for molecular dynamics and density-functional theory. Phys Rev Lett 55:2471–2474. https://doi.org/10.1103/PhysRevLett.55.2471
2. Marx D, Hutter J (2012) Ab Initio molecular dynamics. Basic theory and advanced methods. Cambridge University Press, New York
3. Sokolov M, Bold BM, Kranz JJ, Hofener S, Niehaus TA, Elstner M (2021) Analytical time-dependent long-range corrected density functional tight binding (TD-LC-DFTB) gradients in DFTB+: implementation and benchmark for excited-state geometries and transition energies. J Chem Theory Comput. 17(4):2266–2282
4. Gaultois MW, Oliynyk AO, Mar A, Sparks TD, Mulholland GJ, Meredig B (2016) Perspective: web-based machine learning models for real-time screening of thermoelectric materials properties. APL Mater. 4:053213. https://doi.org/10.1063/1.4952607
5. Lu S, Zhou Q, Ouyang Y, Guo Y, Li Q, Wang J (2018) Accelerated discovery of stable lead-free hybrid organic-inorganic perovskites via machine learning. Nat Commun. 9:3405. https://doi.org/10.1038/s41467-018-05761-w
6. Gómez-Bombarelli R (2016) Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. Nat Mater 15:1120–1127. https://doi.org/10.1038/nmat4717
7. Xue D, Balachandran PV, Hogden J, Theiler J, Xue D, Lookman T (2016) Accelerated search for materials with targeted properties by adaptive design. Nat Commun 7:11241. https://doi.org/10.1038/nmat4717
8. Xie T, Grossman JC (2018) Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. Phys Rev Lett. 120(14):14530. https://doi.org/10.1103/PhysRevLett.120.145301
9. Chen C, Ye W, Zuo Y, Zheng C, Ong SP (2019) Graph networks as a universal machine learning framework for molecules and crystals. Chem Mater 31(9):3564–3572. https://doi.org/10.1021/acs.chemmater.9b01294
10. Reymond JL (2015) The chemical space project. Acc Chem Res 48(3):722–730. https://doi.org/10.1021/ar500432k
11. Hu W, Fey M, Zitnik M, Dong Y, Ren H, Liu B, Catasta M, Leskovec J (2020) Open graph benchmark: Datasets for machine learning on graphs. Advances in Neural Information Processing Systems 2020-Decem(NeurIPS), 1–34 arXiv:2005.00687
12. Hu W, Fey M, Ren H, Nakata M, Dong Y, Leskovec J (2021) OGB-LSC: A large-scale challenge for machine learning on graphs. arXiv preprint arXiv:2103.09430
13. Blanchard AE, Gounley J, Bhowmik D, Pilsun Y, Irle S AISD HOMO-LUMO. https://doi.org/10.13139/ORNLNCCS/1869409
14. Lupo Pasini M, Zhang P, Reeve ST, Choi JY (2022) Multi-task graph neural networks for simultaneous prediction of global and atomic properties in ferromagnetic systems. Mach Learn Sci Technol. 3(2):025007. https://doi.org/10.1088/2632-2153/ac6a51
15. Godoy WF, Podhorszki N, Wang R, Atkins C, Eisenhauer G, Gu J, Davis P, Choi J, Germaschewski K, Huck K et al (2020) ADIOS 2: the adaptable input output system. A framework for high-performance data management. SoftwareX 12:100561
16. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR
17. Choudhary K, DeCost B (2021) Atomistic line graph neural network for improved materials property predictions. NPJ Comput Mater 7(1):1–8
18. Nakamura T, Sakaue S, Fujii K, Harabuchi Y, Maeda S (2020) Iwata S Selecting molecules with diverse structures and properties by maximizing submodular functions of descriptors learned with graph neural networks. Sci Rep. 12:1124. https://doi.org/10.1021/acs.jcim.0c00687

19. Rahaman O, Gagliardi A (2020) Deep learning total energies and orbital energies of large organic molecules using hybridization of molecular fingerprints. J Chem Inf Model. 60(12):5971–5983. https://doi.org/10.1021/acs.jcim.0c00687

20. Ramakrishnan R, Dral PO, Rupp M, Von Lilienfeld OA (2014) Quantum chemistry structures and properties of 134 kilo molecules. Sci Data 1(1):1–7

21. Stuke A, Kunkel C, Golze D, Todorović M, Margraf JT, Reuter K, Rinke P, Oberhofer H (2020) Atomic structures and orbital energies of 61,489 crystal-forming organic molecules. Sci Data 7(1):1–11

22. Ying C, Cai T, Luo S, Zheng S, Ke G, He D, Shen Y, Liu T-Y (2021) Do transformers really perform badly for graph representation? In: Advances in Neural Information Processing Systems, vol. 34, pp. 28877–28888. https://proceedings.neurips.cc/paper/2021/file/f1c1592588411002af340cbaedd6fc33-Paper.pdf

23. Park W, Chang W-G, Lee D, Kim J, Hwang S-w (2022) GRPE: Relative positional encoding for graph transformer. In: ICLR2022 Machine Learning for Drug Discovery. https://openreview.net/forum?id=GNfAFN_p1d

24. Besta M, Hoefler T (2022) Parallel and distributed graph neural networks: an in-depth concurrency analysis. https://doi.org/10.48550/ARXIV.2205.09702

25. Folk M, Heber G, Koziol Q, Pourmal E, Robinson D (2011) An overview of the HDF5 technology suite and its applications. In: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, pp. 36–47

26. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2009) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80. https://doi.org/10.1109/TNN.2008.2005605

27. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc., Centre Convencions Internacional Barcelona, Barcelona Sain. https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

28. Corso G, Cavalleri L, Beaini D, Liò P, Veličković P (2020) Principal neighbourhood aggregation for graph nets. Adv Neural Inf Process Syst 33:13260–13271

29. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 1025–1035. Curran Associates, Inc., Long Beach Convention Center, Long Beach. https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

30. Lupo Pasini M, Reeve ST, Zhang P, Choi JY (2021) HydraGNN. Computer Software. https://doi.org/10.11578/dc.20211019.2. https://github.com/ORNL/HydraGNN

31. PyTorch. https://pytorch.org/docs/stable/index.html

32. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst. 32

33. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds

34. PyTorch Geometric. https://pytorch-geometric.readthedocs.io/en/latest/

35. Dominski J, Cheng J, Merlo G, Carey V, Hager R, Ricketson L, Choi J, Ethier S, Germaschewski K, Ku S et al (2021) Spatial coupling of gyrokinetic simulations, a generalized scheme based on first-principles. Phys Plasmas 28(2):022301

36. Merlo G, Janhunen S, Jenko F, Bhattacharjee A, Chang C, Cheng J, Davis P, Dominski J, Germaschewski K, Hager R et al (2021) First coupled GENE-XGC microturbulence simulations. Phys Plasmas 28(1):012303

37. Cheng J, Dominski J, Chen Y, Chen H, Merlo G, Ku S-H, Hager R, Chang C-S, Suchyta E, D'Azevedo E et al (2020) Spatial core-edge coupling of the particle-in-cell gyrokinetic codes GEM and XGC. Phys Plasmas 27(12):122510

38. Poeschel F, Godoy WF, Podhorszki N, Klasky S, Eisenhauer G, Davis PE, Wan L, Gainaru A, Gu J, Koller F et al (2021) Transitioning from file-based HPC workflows to streaming data pipelines with openPMD and ADIOS2. arXiv preprint arXiv:2107.06108

39. Wan L, Huebl A, Gu J, Poeschel F, Gainaru A, Wang R, Chen J, Liang X, Ganyushin D, Munson T et al (2021) Improving I/O performance for exascale applications through online data layout reorganization. IEEE Trans Parallel Distrib Syst 33(4):878–890

40. Wang D, Luo X, Yuan F, Podhorszki N (2017) A data analysis framework for earth system simulation within an in-situ infrastructure. J Comput Commun. 5(14)

41. Thompson AP, Aktulga HM, Berger R, Bolintineanu DS, Brown WM, Crozier PS, in 't Veld PJ, Kohlmeyer A, Moore SG, Nguyen TD, Shan R, Stevens MJ, Tranchida J, Trott C, Plimpton SJ, (2022) LAMMPS—a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. Comp Phys Comm. 271:108171. https://doi.org/10.1016/j.cpc.2021.108171

42. OLCF Supercomputer Summit. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/

43. Weininger D (1998) SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comput Sci. 28:31–36. https://doi.org/10.1021/ci00057a005

44. Nakata M, Shimazaki T (2017) PubChemQC project: a large-scale first-principles electronic structure database for data-driven chemistry. J Chem Inf Model 57(6):1300–1308. https://doi.org/10.1021/acs.jcim.7b00083

45. Blanchard AE, Gounley J, Bhowmik D, Shekar MC, Lyngaas I, Gao S, Yin J, Tsaris A, Wang F, Glaser J (2021) Language models for the prediction of SARS-CoV-2 inhibitors. Preprint at https://www.biorxiv.org/content/10.1101/2021.12.10.471928v1

46. Loshchilov I, Hutter F (2019) Decoupled weight decay regularization. In: 7th International Conference on Learning Representations, ICLR 2019. OpenReview.net, New Orleans, LA, USA. https://openreview.net/forum?id=Bkg6RiCqY7

47. You Y, Gitman I, Ginsburg B (2017) Large batch training of convolutional networks. arXiv:1708.03888 [cs.CV]. arXiv:1708.03888

48. You Y, Hseu J, Ying C, Demmel J, Keutzer K, Hsieh C-J (2019) Large-batch training for lstm and beyond. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '19. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3295500.3356137

49. Pasini ML, Yin J, Reshniak V, Stoyanov MK (2022) Anderson acceleration for distributed training of deep learning models. In: SoutheastCon 2022, pp. 289–295. https://doi.org/10.1109/SoutheastCon48659.2022.9763953

Publisher's Note