MOX-Report No. 62/2021

Fast and accurate predictions of total energy for solid solution alloys with graph convolutional neural networks

Lupo Pasini, M.; Burcul, M.; Reeve, S.; Eisenbach, M.; Perotto, S.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it http://mox.polimi.it

Fast and accurate predictions of total energy for solid solution alloys with graph convolutional neural networks

Massimiliano Lupo Pasini[1], Marko Burčul[2], Samuel Temple Reeve[1],

Markus Eisenbach[3], Simona Perotto[4]

October 13, 2021

[1]Oak Ridge National Laboratory, Computational Sciences and Engineering Division
Oak Ridge, TN 37831, USA
[2] Department of Automation and Control Engineering, Politecnico di Milano
Piazza L. da Vinci 32, I-20133 Milano, Italy
[3] Oak Ridge National Laboratory, National Center for Computational Sciences
Oak Ridge, TN, 37831, USA
[4] MOX– Department of Mathematics, Politecnico di Milano
Piazza L. da Vinci 32, I-20133 Milano, Italy

Abstract

We use graph convolutional neural networks (GCNNs) to produce fast and accurate predictions of the total energy of solid solution binary alloys. GCNNs allow us to abstract the lattice structure of a solid material as a graph, whereby atoms are modeled as nodes and metallic bonds as edges. This representation naturally incorporates information about the structure of the material, thereby eliminating the need for computationally expensive data pre-processing which would be required with standard neural network (NN) approaches. We train GCNNs on ab-initio density functional theory (DFT) for copper-gold (CuAu) and iron-platinum (FePt) data that has been generated by running the LSMS-3 code, which implements a locally self-consistent multiple scattering method, on OLCF supercomputers Titan and Summit. GCNN outperforms the ab-initio DFT simulation by orders of magnitude in terms of computational time to produce the estimate of the total energy for a given atomic configuration of the lattice structure. We compare the predictive performance of GCNN models against a standard NN such as dense feedforward multi-layer perceptron (MLP) by using the root-mean-squared errors to quantify the predictive quality of the deep learning (DL) models. We find that the attainable accuracy of GCNNs is at least an order of magnitude better than that of the MLP.

This manuscript has been authored in part by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the

1 Introduction

Understanding and predicting the properties of materials with different atomic structures is critical for improved application performance and new technologies. There are of course many features that control material properties beyond the atomic level, from the mesoscale to macroscale. However, due to the combinatorial complexity of elements, crystal structures, and atomic disorder [1, 2, 3], there are still significant opportunities in materials discovery from information at the atomic scale. While many computational approaches have been developed to accurately model and predict the behavior of materials at the atomic scale from the electronic structure, including density functional theory (DFT) [4, 5], quantum Monte Carlo (QMC) [6, 7], and *ab-initio* molecular dynamics (MD) [8, 9], these techniques come with very high computational cost, even for relatively small numbers of atoms and/or small timescales [10]. To alleviate this cost, several techniques have been developed ranging from direct approximations of electronic structures methods to empirical models, which trade predictive accuracy for computational effort. For example, cluster expansion builds the total energy as a linear combination of contributions from interactions of different atom clusters, with input from smaller direct electronic structure calculations [11, 12, 13]. Classical MD starts from a larger scale approximation which ignores electrons entirely and fits an interatomic interaction model with DFT, or other quantum simulation algorithms [14, 15]. These models use empirical functional forms to represent bonding or approximate electronic effects. Quite importantly, MD is orders of magnitude faster than DFT for most atomic systems; however, MD models are difficult to develop even for a small number of elements and generally are not transferable from the dataset on which they were trained.

In spite of these issues, MD interatomic models built from DFT are widely used as surrogate models that do not require training from data. The one to one mapping between the high and low fidelity systems is a strength of the approach. Indeed, rather than a generic fitting procedure for the surrogate model parameters matching various output quantities of interest, forces and energies on each atom in each system can be used to train the MD model to match DFT. This force matching approach represents a significant advance for the field [16] and is now standard for training MD models, particularly as the increasing size of the systems modeled leads to more complex, multi-dimensional optimizations.

As the power of data-driven and machine learning (ML) approaches in sci-

ence continues to grow, ML surrogate models can also provide significant benefits. Neural networks (NN) are attractive as a general mathematical form which include non-linear interactions and, once trained, can be orders of magnitude faster than full physics simulations. There are many examples of NN surrogates for electronic structure calculations [17, 18, 19, 20, 21], as well as classical MD models [22, 23, 24, 25]. This has advanced classical MD significantly beyond the original empirical models. Critical features of these approaches include high accuracy (particularly when resolving atomic dynamics) and preservation of translational and rotational invariances.

However, complex NNs require significant effort to translate the atomic structure dataset into a form understood by the DL model. This data pre-processing inevitably leads to some loss of information originally contained in the raw DFT simulations and also requires additional effort to be performed.

The DL community has recently developed graph convolutional neural networks (GCNNs) [26, 27] which directly map the atomic input to graph structures, with atoms as graph nodes and chemical bonds as edges. This direct connection between the high-fidelity training data and the surrogate model is compelling as a DL equivalent of classical MD models. GCNNs not only reduce the cumbersome and expensive data pre-processing, but also, by abstracting the representation of the lattice structure using adjacency matrices, GCNNs can naturally be trained on lattices of different structures and sizes. Previous work with GCNN models in materials science includes crystal graph convolutional neural networks (CGCNN) [28] and material graph network (MEGNet) [29]. The focus of these efforts was on using GCNNs for prediction of material properties across broad classes of crystalline materials, sourced from the Materials Project and the Open Quantum Materials Database (OQMD) [2, 3]. This work showed significant flexibility of the DL model, simultaneously handling many materials across different properties, with good accuracy relative to the original DFT results. In this work, we focus on improving the predictive accuracy of DL models using GCNNs for chemically disordered binary solid solution alloys. Although the approach consists in training GCNN on DFT data, our goal is not to necessarily use GCNN as a replacement for DFT. Instead, we aim to use GCNN models to construct well educated initial guesses that could be used as starting point for DFT simulations, enabling the numerical study of large scale atomic structures that otherwise would not be computationally affordable.

In particular, we use open source DFT datasets, published on OLCF Data Constellation, to predict the total energy of copper-gold (CuAu) and iron-platinum (FePt) alloys with DL models. We train GCNNs on these datasets and compare their predictive performance with respect to multi-layer perceptron (MLP) architectures previously used. We show that GCNNs attain higher accuracy than MLPs and reduce the root mean squared error (RMSE) on validation data by an order of magnitude on both datasets. GCNNs also outperform the base-line DFT simulation by orders of magnitude in terms of computational time to produce the estimate of the total energy for a given atomic configuration

of the lattice structure.

2 Physical system - solid solution binary alloys

The material systems on which we focus in this work are solid solution binary alloys, where two constituent elements are randomly placed on a fixed underlying crystal lattice. We consider two binary alloy systems, each with 32 atoms and with periodic boundary conditions considered within the DFT calculations. The first system is the CuAu [30] alloy arranged in a $2 \times 2 \times 2$ supercell and a face-centered cubic (FCC) structure, while the second system is the FePt alloy [31] arranged in a $2 \times 2 \times 4$ supercell with a body-centered cubic (BCC) structure.

For both CuAu and FePt datasets, each data point in the dataset provides the information about the atomic positions on a lattice. Denoting the total number of atoms in a configuration N_{atoms}, each input data point is represented as a $N_{\text{atoms}} \times 4$ matrix, where the first three columns provide the (x, y, z) coordinates of an atom and the fourth column provides the proton number that uniquely characterizes the atomic element located at a specific lattice point. The total energy is a single scalar for every configuration.

Figure 1 shows the local numbering of atoms inside a FCC or a BCC unit cell. For the atoms in CuAu arranged in $2 \times 2 \times 2$ FCC unit cells, the atoms are numbered starting from the $(x = 0, y = 0, z = 0)$ unit cell. The counting traverses through the unit cells in the x-direction first, then the y-direction, and the z-direction last. For example, atoms 1 to 4 are from unit cell (0, 0, 0), atoms 5 to 8 are from unit cell (1, 0, 0), atoms 9 to 12 are from unit cell (0, 1, 0), and so on. Within a unit cell, the order of counting follows the local numbering of atoms. The numbering of atoms for FePt follows the same manner, with a difference that the atoms are arranged in a supercell having $2 \times 2 \times 4$ BCC unit cells, each unit cell only has two atoms.

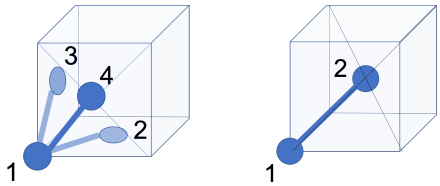

Figure 1: A face-centered cubic (FCC) unit cell for CuAu (left) and a body-centered cubic (BCC) unit cell for FePt (right). The local numbering of atoms in the unit cell for each structure is shown.

For each of these two alloys, 32,000 configurations with different chemical compositions between the two atom species (Cu and Au for the first system, and Fe and Pt for the second system) were chosen to construct the training dataset. More details about the dataset construction are presented in Section 4.

3 Graph convolutional neural networks (GCNNs)

A graph G is usually represented in mathematical terms as

$$G = (V, \mathcal{E}) \tag{1}$$

where V represents the set of nodes and $\mathcal{E}$ represents the set of edges between these nodes [32]. An edge is defined as a pair $(u, v) \in \mathcal{E}$ where $u, v \in V$, $\mathcal{E} \in V \times V$, and the edge starts at node u and ends in node v. The topology of a graph can be described through the adjacency matrix, a square matrix, A, with as many rows and columns as the number of nodes in the graph, whose entries are associated with edges of the graph according to the following rule:

$$\begin{cases} A[u, v] = 1 & \text{iff} \quad (u, v) \in \mathcal{E} \\ A[u, v] = 0 & \text{otherwise.} \end{cases} \tag{2}$$

The degree of a node $u \in V$ is defined as:

$$d_u = \sum_{v \in V} A[u, v] \tag{3}$$

and it represents the number of edges incident to a node. The node degree is used in the GCNN when aggregating the information from the neighborhood of a node.

In order to take advantage of the topology of the graph with N nodes, the DL model has to consider the following properties as input features:

- The number of neighbors ($L < N$) for each atom
- The distance between atoms (i.e., bond length).

If the input structure is defined with respect to N nodes in the graph, MLPs cannot take advantage of the information about neighboring nodes, as the fully connectivity of the MLP architectures forces all nodes to communicate with each other. This approach is difficult to scale for graphs of increasing size because the number of interactions increases combinatorially with N. One solution is to change the representation of the input, so that the input is defined in terms of edges instead of being defined in terms of nodes. However, this representation increases the dimensionality in the input from N to $L \times N$ and this further increases the computational cost to train the MLP model.

GCNNs embed the interaction between nodes without increasing the size of the input by representing the local interaction zone as a hyperparameter that cuts-off the interaction of a node with all the other nodes outside a prescribed local neighborhood. The fact that GCNN can naturally distinguish between short-range and long-range interactions without expanding the dimensionality of the input results into a computational saving with respect to an MLP.

GCNNs [26, 27] are DL models based on a message-passing framework, a procedure that combines the knowledge from neighboring nodes, which in our applications maps directly to the interactions of a central atom with its neighbors in the lattice structure. The typical GCNN architecture is characterized by three different types of hidden layers: graph convolutional layers, graph pooling layers, and fully connected layers. A schematic of a GCNN structure is provided in Figure 6. The convolutional graph layers represent the central part of the architecture and their functionality is to transfer feature information between adjacent nodes (in this case atoms). Every node $u_i \in \mathcal{V}$ is associated with a p-dimensional vector $h_i \in \mathbb{R}^p$ which contains the embedded nodal features for node u_i. Message passing is performed at each step of the training, and it requires performing in sequential the following operations:

1. Aggregate information from neighbors

2. Update hidden state information.

Through aggregation, the node u_i collects the hidden embedded features of its neighbors as shown in Figure 2 as well as the information on the edges (if available). After the aggregation is completed, the node u_i updates its hidden state

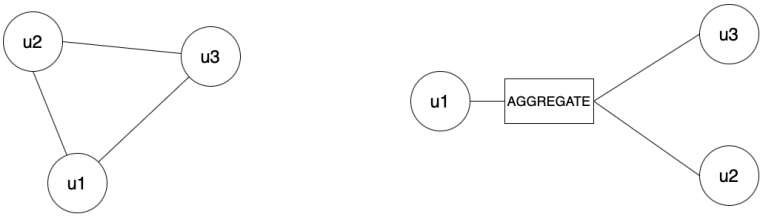

Figure 2: Message passing on a simple graph with three nodes.

h_i at iteration $(t+1)$ according to the following formula:

$$m_i^{t+1} = \sum_{j \in N(i)} m_j^t(h_j^t, e_{ij}^t) \tag{4}$$

where m_j^t is a message obtained from neighboring node u_j and the edge e_{ij} that connects them. Following, the nodal features h_i^{t+1} of the node u_j are updated at the $(t+1)$th step of the training as follows:

$$h_i^{t+1} = UPDATE(h_i^t, m_i^{t+1}) \tag{5}$$

where $UPDATE$ is a an arbitrarily differentiable function which combines aggregated messages m_i^{t+1} of node u_i neighbors with its nodal features h_i^t from the previous step t.

Through consecutive steps of message passing, the graph nodes gather information from nodes that are further and further away. As shown in Figure 3

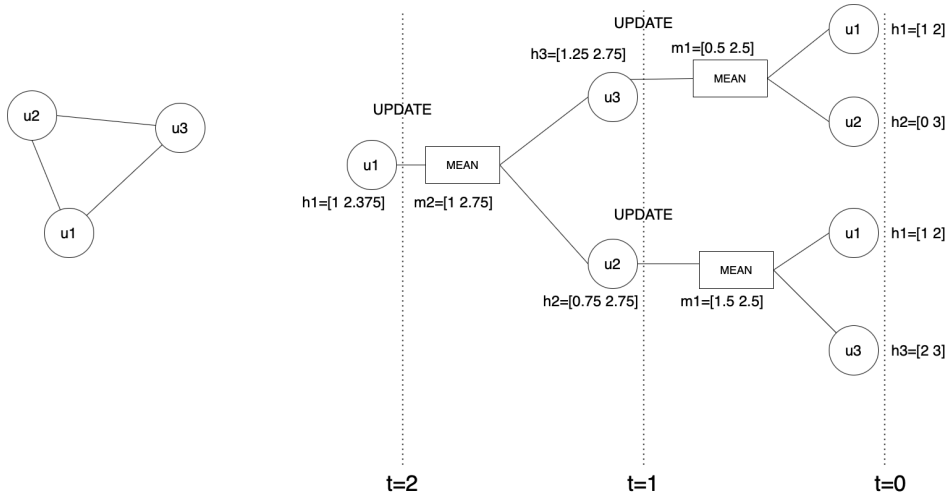

Figure 3: Example of 2-iteration message passing where the aggregation function used is mean of neighbors hidden states. The update function is also mean of the current node hidden state and aggregated message from node neighbors.

where k=2, node u_1 gets the information from neighbors of its neighbors. The type of information passed through a graph structure can be either related to the topology of the graph or features assigned to the nodes. An example of a topological information is the node degree, whereas an example of nodal feature in the context of this work is the proton number of the atom located at the node. The aggregation function aims at collecting information from adjacent nodes in a graph can be defined as:

$$AGGREGATE(u_i) = W^{(k)}_{neighborhood} \sum_{v \in N(u_i)} h^{k-1}_v + b^{k-1} \tag{6}$$

and the function that updates the nodal features is defined as:

$$UPDATE(u_i) = \sigma(W^{(k)}_{self} h^{k-1}_i + AGGREGATE(u_i))$$

where

$$W^{(k)}_{self}, W^{(k)}_{neighborhood} \in \mathbb{R}^{p \times p} \tag{7}$$

are the weights of one layer of GCNN and σ is an activation function (e.g, ReLU) that introduces nonlinearity to the model.

The most common types of graph convilutional layers are:

1. graph isomorphism network (GIN);
2. graph attention network (GAT); and
3. principal neighborhood aggregation (PNA).

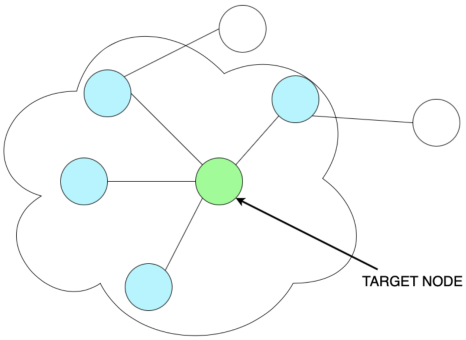

Figure 4: Convolution performed on target node of a graph where the number of neighbors varies in size and nodes are unordered.

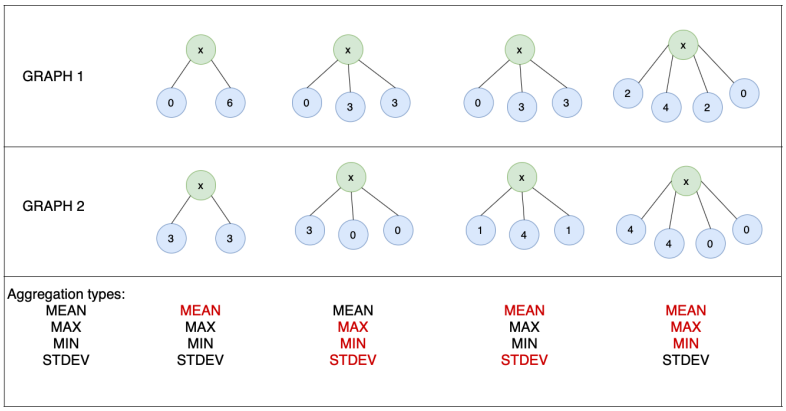

Figure 5: Examples of different aggregation schemes failing to distinguish between different graph pairs.

GIN [33] and GAT [34] use a single aggregating operation to perform message passing among adjacent nodes of a graph. In particular, GIN aggregates information using a sum, whereas GAT aggregates information using a weighted sum where each nodes is weighted according to the number of its neighbors. However, using a single aggregating operation to perform message passing may cause these aggregation schemes to confuse distinct graphs. Examples of aggregating operations that can fail in distinguishing different graphs are given in Figure 5. In contrast, PNA [35] combines multiple aggregating techniques and uses degree-based scalers that depend on a node degree and accordingly amplify or attenuate incoming messages; this results in an increase of the discriminating power of the model, as the model is less prone to classify two different graphs as identical. More details about the PNA aggregation scheme can be found in [36]. We compare the results of these different graph layers in Section 6.

The graph pooling layers are interlaced between successive graph convolutional layers and reduce the dimensionality of the graph by aggregating the

information contained in adjacent nodes and edges of the graph (the atomic neighborhood for each atom). Fully connected (FC) layers are positioned at the end of the architecture to take the results of pooling and flatten them in order to match the dimensionality of the output. Batch normalizations are performed be-

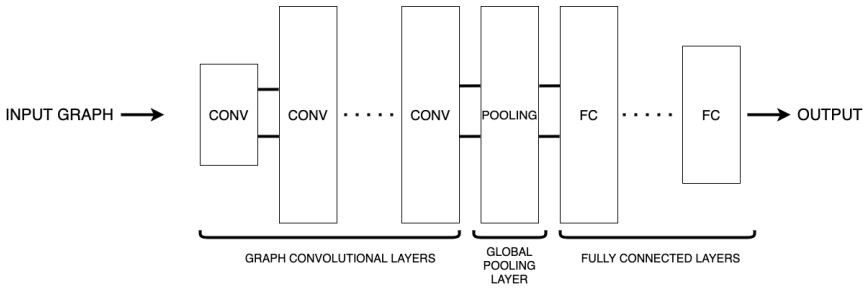

Figure 6: High level overview of the GCNN architecture used.

tween consecutive convolutional layers along with a rectified linear unit (ReLU) activation function to avoid vanishing gradients.

The set of hyperparameters that fully characterize the architecture of the DL model are the radius to define the local neighborhood of a graph node, the maximum number of nodes allowed in a neighborhood, the number of neurons in a hidden convolutional layer (also referred to as size of a layer), and the number of hidden convolutional layers. Hyperparameter optimization is performed to identify the architecture of the NN with best predictive performance.

Solid solution binary alloys consist of two types of atoms that randomly populate a crystal lattice. Studying the behavior of these materials is challenging due to the combinatoric complexity that quickly increases with the number of types and with the size of the lattice. Therefore, developing an accurate GCNN model that reliably estimates the material properties for each disordered configuration requires a large amount of data.

3.1 Software implementation

We use the `Pytorch` framework [37] to implement the GCNN in Python. `Pytorch` is not only a robust NN toolkit, but also serves as a performance portability layer for running on heterogeneous computing architectures, e.g., CPU and GPU. This method enables running our GCNN on any local machine, as well as the Summit and coming Frontier supercomputers. In addition, there are many packages which are extensions to `Pytorch`: `Pytorch Geometric` [38, 39] is particularly relevant to this work for graph structures. The code used in this work is openly available and developed on GitHub [40]. The hyperparameter configurations for each neural network architecture have been identified by performing hyperparameter optimization using the `Ray Tune` library [41, 42].

4 Dataset description

The dataset for each alloy comprises 32,000 configurations out of the 2^{32} available. The selection of the configurations is performed to ensure that every composition is adequately represented in the dataset. If the number of configurations for a specific composition is less than 1,000, then all those configurations are included in the dataset. For all other compositions, configurations are randomly selected up to the total of 32,000 . Moreover, the splitting between the training, and validation sets is performed at the level of each composition, to ensure that all the compositions are adequately represented in both the training and validation portions of the dataset. For each selected configuration, we computed the total energy of the system at each atomic position using the locally self-consistent multiple scattering (LSMS) DFT approach, which exhibits linear scaling with respect to the number of atoms [43, 44, 45]. We used the LSMS-3 code from Oak Ridge National Laboratory [46]. These data are openly available through the OLCF Constellation [30, 31], and describe material properties for atomic systems with chemical disorder. The chemical disorder makes the task of describing the material properties combinatorially complex, and the combinatorial complexity addressed by these datasets represents the main difference from other open source databases that only focus on ordered compounds [1, 2, 3].

Since different physical quantities have different units and different orders of magnitude, the inputs and outputs are respectively standardized (normalized) across all data points for each quantity, such that each quantity has a zero mean and unit standard deviation.

Raw input data is imported in the format of csv files and transformed into a `Python Data` object to handle the data through the `Pytorch Geometric` framework. The data object contains the following information:

1. **edge_index** $\in \mathbb{R}^{32\times 2}$ contains index pairs of neighboring nodes
2. **pos** $\in \mathbb{R}^{32\times 3}$ contains coordinates for each atom

Since `PyTorch Geometric` was built upon `PyTorch`, all of the primitive types are converted to `PyTorch` tensors. After loading the data, the total energy value for each configurations needs to be normalized using the min-max normalization defined as

$$e_i' = \frac{e_i - e_{\min}}{e_{\max} - e_{\min}} \tag{8}$$

where $e_{\min}$ and $e_{\max}$ minimum and maximum value of the total energy across all the nodes with the same index in the dataset, and use these values to perform the normalization. The range of values of the energy expressed in Rydberg is $[-1.22 \cdot 10^6, -1.41 \cdot 10^5]$ for CuAu and $[-1.12 \cdot 10^6, -1.16 \cdot 10^5]$ for FePt.

After standardizing the data, we extract the graph topology from the lattice structure. Since the connections between atoms in the lattice structure are not directly provided by the datasets, we created a procedure described in Algorithm

1 to calculate an approximation of the adjacency matrix that only considers local interactions between nodes of the graph with a maximum neighborhood radius and a maximum number of neighbors. The approximate adjacency matrix is then fed into the GCNN to characterize the structure of the graph convolutional layers that will perform successive aggregations of nodal features on adjacent nodes of the graph. The nodal feature for the first graph convolutional layer is the proton number, whereas the nodal features for successive convolutional layers are the result of convolutional operations that cannot be directly interpreted in terms of physical properties. Once the GCNN produces the prediction of the target quantity, a data post-processing is performed to remap the predicted quantities back to the physical space to restore their actual values and units.

Larger sizes of the local neighborhood lead to a higher computational cost to train the GCNN, as the number of regression coefficients to train at each hidden convolutional layer increases quadratically with the size of the local neighborhood. Moreover, a local interaction zone is also used in the LSMS-3 code to generate the DFT training data; therefore, setting the size of the local neighborhood to a large value causes the GCNN model to overfit as the model reconstructs interatomic interactions that are not captured in the DFT data.

5 Use of federated instruments, compute, and storage

The workflow pipeline described in this work benefits from the entire ORNL computing and data storage infrastructure, and in turn benefits it by developing additional robust and accurate DL capabilities. We illustrate how the research described in this work integrates in the entire OLCF infrastructure of federated instruments, compute, and storage in Figure 7. This research, part of the Artificial Intelligence for Scientific Discovery (AISD) Thrust of the Artificial Intelligence (AI) Initiative at ORNL, aims at developing and deplying fast and accurate surrogate models to accelerate the material design. As such, our GCNN take advantage of the existing ORNL resources:

- OLCF supercomputers
- OLCF data management
- OLCF-CADES edge computing.

The OLCF supercomputer Summit is used to quickly generate training data from large-scale ab-initio DFT simulations. The OLCF Constellation is used to permanently store the full results of the DFT simulations, enabling public access and citations through DOIs. The OLCF-CADES GPU-enabled edge computing clusters allow us to quickly train the GCNN model on the data downloaded from the OLCF Constellation, where eventually we can deploy the trained and validated model for scientific discovery. The outcome of this research, which

Algorithm 1 Calculating graph adjacency matrix in 3D

input : nodes - list containing nodes of the graph with coordinate information
r - radius within to search for the neighbors of a node
mnnn - maximum number of node neighbors
output: adj - adjacency matrix

```
adj = [len(nodes), len(nodes)] initialize adjacency matrix with zeros
 neighbors = {} dictionary of size n where key is index of a node and value is
 empty list
 distances = [len(nodes), len(nodes)] matrix of node distances
 i = 0 initalize index to zero
  for i < len(nodes) do
    j ← i + 1
     for j < len(nodes) do
       distances [i][j] = calculate_distance_in_3D (nodes [i],nodes [j])
        if distances [i][j] <= r then
          neighbors [i].append(j)
       end
    end
end

// Order neighbors for each node by their distance.
ordered_neighbors = order_candidate_neighbors_by_distance(neighbors, distances)
   // For each node find neighbors.

for node, neighbors in ordered_neighbors do
    neighbors = resolve_neighbor_conflicts (node, neighbors, adj, mnnn)
     adj [node, neighbors ] = 1
     adj [neighbors, node ] = 1
end
```

strongly relies on the Oak Ridge Leadership Computing Facility, will provide strategically relevant AI capabilities to the other ORNL user facilities such as the Manufacturing Demonstration Facility (MDF) and the Spallation Neutron Source (SNS).

6 Numerical results

We present numerical results to predict the total energy for the binary CuAu and FePt alloys using DL models by comparing the predictive performance of simple fully connected MLPs with GCNNs that use GIN, GAT, and PNA graph convolutional layers. The output of DFT calculations is considered as the exact reference that the DL model has to reconstruct. Therefore, the predictive performance of a DL model is tested by measuring the departure of quantities

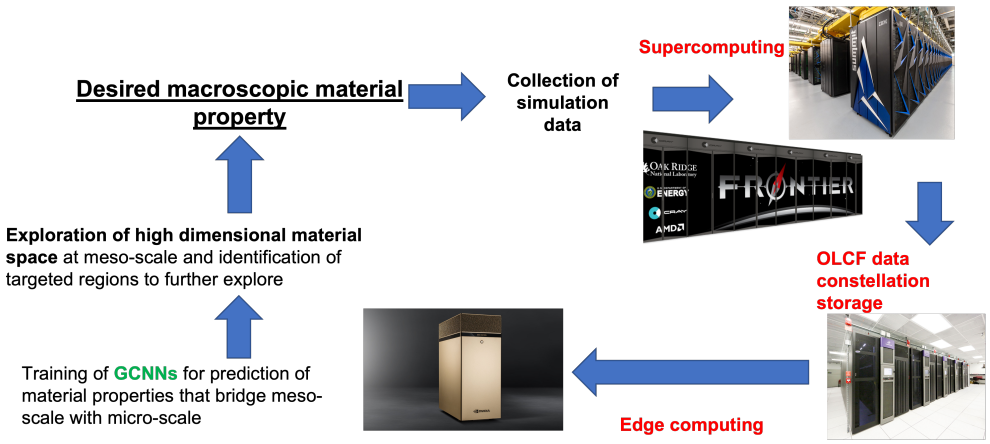

Figure 7: Illustration of the OLCF infrastructure.

predicted by the DL models from the results produced by DFT calculations.

We used the `HyperOpt` search algorithm based on the Tree-Structured Parzen Estimator approach (TPE) inside the `Ray Tune` library to identify the NN architecture that minimizes the RMSE. Details about GCNN architectures resulting from the hyperparameter optimization are presented in Table 1.

	NN model		
Hyperparameter	GCNN-GIN	GCNN-GAT	GCNN-PNA
Radius size	5	5	5
Maximum number of node neighbors	7	7	7
Hidden layer size	60	20	15
Number of convolutional layers	5	16	16

Table 1: Hyperparameter setting for GCNNs with GIN, GAT, and PNA graph convolutional layers.

The DL models are trained using the Adam method [47] with an initial learning rate equal to 0.0001, with a total number of 200 epochs performed. The batch size for each step of an epoch is 64 data points. Early stopping is performed to interrupt the training when the validation loss function does not decrease for several consecutive epochs, as this is a symptom that further epochs are very unlikely to reduce the value of the loss function. The training set for each of the NN represents 80% of the total dataset; the validation set represents the remaining 20%.

6.1 Comparison between computational times for first principles calculations and DL models

The DFT calculations to generate the dataset to train the NN models were performed with the LSMS-3 code on the Titan supercomputer at Oak Ridge National Laboratory (ORNL), each calculation used 8 Titan nodes (each Titan node had 1 NVIDIA K20X GPU) for a hybrid MPI-CUDA parallelization. For more details about the hardware specifics of Titan we refer to [48]. Because ORNL's Titan supercomputer has been decommissioned as of this writing, we present here the computational time of the same calculations on ORNL's current supercomputer Summit [49]. Each calculation was performed on one Summit node, utilizing all 6 NVIDIA V100 GPUs on the node.

Our DL approach demonstrates significant time reductions for both CuAu and FePt cases when NN models are used in place of DFT calculations for one configuration. The first-principles LSMS calculations take about 260 wall-clock seconds for CuAu and 300 wall-clock second for FePt on average on a Summit node, whereas the NN models predict the physical quantities in about one wall-clock second. The training of the NN models takes about 4,000 wall-clock seconds on an NVIDIA V100 GPU.

Computational approach	**Compute resources**	**Wall-clock time(s)**
1 DFT calculation	1 Summit node	263.7
MLP	1 NVIDIA-V100 GPU	0.9
GCNN	1 NVIDIA-V100 GPU	1.3
$\sim 10^6$ DFT calculations	1 Summit node	$2.63 \cdot 10^8$
32,000 DFT calc. + train NN + 10^6 NN pred.	1 Summit node	$9.71 \cdot 10^6$

Table 2: CuAu binary alloy - Average time in wall-clock seconds needed to estimate macroscopic physical properties on a random lattice configuration with first principles calculations, time for one single-tasking NN models evaluation, time for one multitasking NN evaluation, total wall-clock time to perform 10^6 DFT calculations and total wall-clock time to perform 32,000 DFT calculations, train the NN and evaluate the total energy for 10^6 configurations using the NN.

In Figures 8 and 9 we show the trend of the training MSE and validation MSE with respect to different percentage of the entire CuAu and FePt datasets to train the GCNN model with PNA aggregation. The results show that using only 10% of the data for training increases the validation MSE of the GCNN model by two orders of magnitude with respect to the use of 90% of the dataset for training. Moreover, the fact that both training MSE and validation MSE reach plateau means that the neural network has been trained long enough to reach its expressive power on the dataset.

Tables 2 and 3 compare the computational time to compute the total energy for 10^6 configurations using the LSMS code, with the time needed for the scenario where the total energy is calculated just for $32,000$ configurations with LSMS to generate the dataset, train the NN model on this dataset, then use the NN model

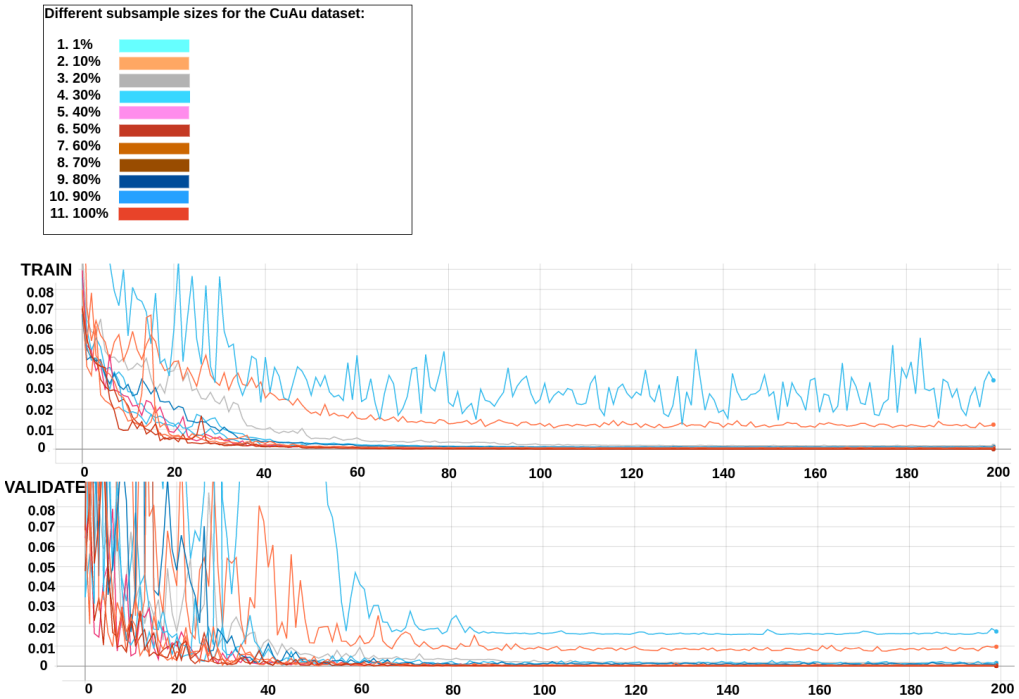

Figure 8: Training and validation MSE at different epochs for training a GCNN model with PNA aggregation policy on the CuAu dataset using different percentages of the dataset for training.

to predict the total energy for 10^6 alloy configurations. This comparison is of relevance for the possible use of surrogate DL models in Monte Carlo simulations to predict the thermodynamic properties of a solid solution alloy, in which the number of Monte Carlo samples required would at least be of the order of 10^6 for sufficient sampling. In this case, it is clear that using DL surrogate models for accurate predictions of total energy significantly reduces the computational time needed to predict material properties that would be otherwise unattainable given limited computational resources. Although the reduction of computational time has been achieved by sacrificing accuracy with respect to the DFT itself, we point out that our goal is not to use a GCNN surrogate model to compete against ab-initio calculations, but rather facilitating the use of DFT in Monte Carlo (MC) simulations to determine the thermodynamic properties of large scale lattices for solid solution alloys.

6.2 Comparison between statistical models for predictive performance

Numerical results obtained by training the NN models on CuAu and FePt datasets are presented in Tables 4 and 5 respectively. The metric used to mea-

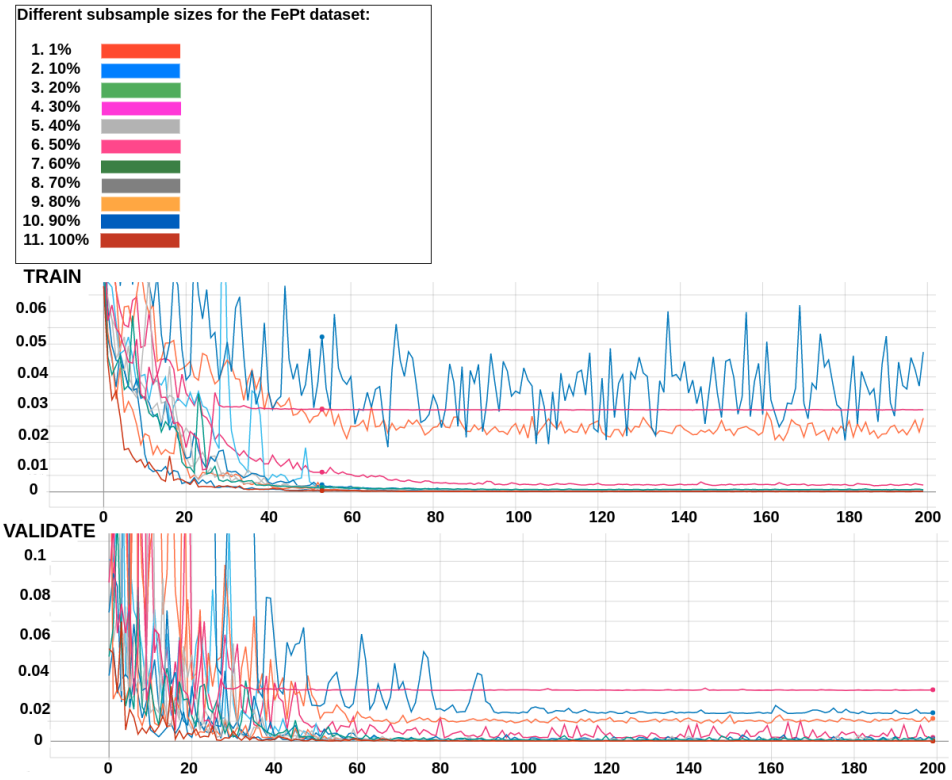

Figure 9: Training and validation MSE for training of a GCNN model with PNA aggregation policy on the FePt dataset using different percentages of the dataset for training.

sure the predictive performance of the models is the RMSE and it shows that GCNNs attain a higher accuracy than MLPs for the prediction of total energy for both CuAu and FePt. The GCNNs, and their natural mapping to the atomic input data, is indeed better to predict system properties. Moreover, these results show that PNA outperforms the other aggregation techniques by an order of magnitude. The results are in line with what expected by the DL theory, since the PNA layer has more expressive power than GIN and GAT. We also notice that MLP and GCNN-PNA attain lower accuracy on the FePt dataset than on the CuAu dataset (while the other two aggregations result in similar accuracy both datasets). This is partially due to the fact that the FePt binary alloy is a hard ferromagnetic metal, making it more difficult for the DFT to converge during self-consistent iterations. As the data for FePt is less accurate than the data for CuAu, so also is the attainable accuracy of the models trained on it.

Computational approach	Compute resources	Wall-clock time(s)
1 DFT calculation	1 Summit node	303.2
MLP	1 NVIDIA-V100 GPU	0.9
GCNN	1 NVIDIA-V100 GPU	1.3
$\sim 10^6$ DFT calculations	1 Summit node	$3.03 \cdot 10^8$
32,000 DFT calc. + train NN + 10^6 NN pred.	1 Summit node	$1.10 \cdot 10^7$

Table 3: FePt binary alloy - Average time in wall-clock seconds needed to estimate macroscopic physical properties on a random lattice configuration with first principles calculations, time for one single-tasking NN models evaluation, time for one multitasking NN evaluation, total wall-clock time to perform 10^6 DFT calculations and total wall-clock time to perform 32,000 DFT calculations, train the NN and evaluate the total energy for 10^6 configurations using the NN.

NN model	RMSE - Total Energy
MLP	6.65×10^{-3}
GCNN - GIN	5.10×10^{-3}
GCNN - GAT	7.14×10^{-3}
GCNN - PNA	2.52×10^{-4}

Table 4: Validation root mean squared error (RMSE) for neural networks that predict total energy for randomly sampled configurations of CuAu. The name of each training method refers to whether it is a multi-layer perceptron (MLP) or graph convolutional neural network (GCNN). Graph isomorphism network (GIN), graph attention network (GAT), and principal neighborhood aggregation (PNA) layers are used to perform the aggregation of neighboring nodes.

7 Conclusions

We trained GCNNs on ab-initio DFT data that produce fast and accurate predictions of the total energy for solid solution binary alloys CuAu and FePt. GCNN architectures use graph representations to directly map to the topology of the atomic structure, which enables GCNNs to attain higher accuracy than dense MLPs. Numerical results show that DL models outperform ab-initio DFT calculations in terms of computational time to provide estimates of the total energy for specific lattice configurations by compromising the accuracy within a tolerable margin of error. A comparison between MLPs and GCNNs in terms of predictive performance shows that GCNNs outperform MLPs on both datasets by reducing the RMSE by at least an order of magnitude. Our goal is not to necessarily replace DFT with a GCNN surrogate model, but rather integrate the two approaches to run large-scale Monte Carlo (MC) simulations for thermodynamic properties of solid solution alloys. Future work will combine two MC procedures in sequence: the first MC procedure will use a GCNN model to generate an educated initial guess, which can then be used as starting point for

NN model	RMSE - Total Energy
MLP	1.22×10^{-2}
GCNN - GIN	5.97×10^{-3}
GCNN - GAT	6.73×10^{-3}
GCNN - PNA	8.01×10^{-4}

Table 5: Validation root mean squared error (RMSE) for neural networks that predict total energy for randomly sampled configurations of FePt alloy. The name of each training method refers to whether it is a multi-layer perceptron (MLP) or graph convolutional neural network (GCNN). Graph isomorphism network (GIN), graph attention network (GAT), and principal neighborhood aggregation (PNA) layers are used to perform the aggregation of neighboring nodes.

a new MC run where DFT is called directly. Combining the two MC runs is expected to drastically reduce the total number of calls to the DFT code substantially reducing the computational time with respect to running DFT with uninformed initial guesses.

Acknowledgements

Massimiliano Lupo Pasini thanks Dr. Vladimir Protopopescu for his valuable feedback in the preparation of this manuscript.

This work was supported in part by the Office of Science of the Department of Energy and by the Laboratory Directed Research and Development (LDRD) Program of Oak Ridge National Laboratory. This work used resources of the Oak Ridge Leadership Computing Facility, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The last author acknowledges the research project GNCS-INdAM 2020 "Tecniche Numeriche Avanzate per Applicazioni Industriali".

References

[1] S. Curtarolo, W. Setyawan, G. L. W. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. J. Mehl, H. T. Stokes, D. O. Demchenko, and D. Morgan. AFLOW: An automatic framework for high-throughput materials discovery. *Comput. Mater. Sci.*, 58:218–226, June 2012.

[2] A. Jain, S. Ping Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson. Commen-

tary: The materials project: A materials genome approach to accelerating materials innovation. *APL Mater.*, 1(1):0–11, 2013.

[3] J. E Saal, S. Kirklin, M. Aykol, B. Meredig, and C. Wolverton. Materials design and discovery with high-throughput density functional theory: the Open Quantum Materials Database (OQMD). *JOM*, 65(11):1501–1509, November 2013.

[4] P. Hoenber and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864B871, 1964.

[5] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138, 1965.

[6] M. P. Nightingale and J. C. Umrigar. *Self-Consistent Equations Including Exchange and Correlation Effects*. Springer, 1999.

[7] B. L. Hammond, W. A. Lester, and P. J. Reynolds. *Monte Carlo Methods in Ab Initio Quantum Chemistry*. Singapore: World Scientific, 1994.

[8] R. Car and M. Parrinello. Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett.*, 55:2471–2474, 1985.

[9] D. Marx and J. Hutter. *Ab Initio Molecular Dynamics, Basic Theory and Advanced Methods*. Cambridge University Press New York, New York, USA, 2012.

[10] J. Aarons, M. Sarwar, D. Thompsett, and C. K. Skylaris. Perspective: Methods for large-scale density functional calculations on metallic systems. *J. Chem. Phys*, 145(22):220901, 2016.

[11] J. M. Sanchez, F. Ducastelle, and D. Gratias. Generalized cluster description of multicomponent systems. *Phys. A Stat. Mech. Appl.*, 128:334–350, 1984.

[12] D. De Fontaine. Cluster approach to order-disorder transformations in alloys. *Phys. A Stat. Mech. Appl.*, 47:33–176, 1994.

[13] O. Levy, G. L. W. Hart, and S. Curtarolo. Uncovering compounds by synergy of cluster expansion and high-throughput methods. *J. Am. Chem. Soc.*, 132(13):4830–4833, 2010.

[14] B. J. Alder and T. E. Wainwright. Phase transition for a hard sphere system. *The Journal of Chemical Physics*, 27(5):1208–1209, November 1957. Number: 5.

[15] A. Rahman. Correlations in the motion of atoms in liquid argon. *Phys. Rev.*, 136(2A):A405–A411, October 1964. Number: 2A.

[16] F. Ercolessi and J. B Adams. Interatomic potentials from first-principles calculations: The force-matching method. *Europhys. Lett.*, 26(8):583–588, June 1994. Number: 8.

[17] F. Brockherde, L. Vogt, M. E. Tuckerman, K. Burke, and K .R. Müller. Bypassing the Kohn-Sham equations with machine learning. *Nat. Commun.*, 8(872), 2017.

[18] C. Wang, A. Tharval, and J. R. Kitchin. A density functional theory parameterised neural network model of zirconia. *Mol. Simul.*, 44(8):623–630, 2018.

[19] A. V. Sinitskiy and V. S. Pande. Deep neural network computes electron densities and energies of a large set of organic molecules faster than density functional theory (DFT). `https://arxiv.org/abs/1809.02723`.

[20] C. A. Custódio, É. R. Filletti, and V. V. França. Artificial neural networks for density-functional optimizations in fermionic systems. *Sci. Rep.*, 9(1886), 2019.

[21] K. Ryczko, D. Strubbe, and I. Tamblyn. Deep learning and density functional theory. *Phys. Rev. A*, 100(022512), 2019.

[22] J. Behler and M. Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98(14):146401, April 2007.

[23] K. Schütt, P.-J. Kindermans, Sauceda F., H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Mller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 991–1001. Curran Associates, Inc., 2017.

[24] Justin S. Smith, Olexandr Isayev, and Adrian E. Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.

[25] L. Zhang, J. Han, H. Wang, R. Car, and W. E. Deep potential molecular Dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120(14):143001, April 2018.

[26] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. Lee,

M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[28] T. Xie and J. C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 120(14):145301, April 2018.

[29] C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chem. Mater.*, 31(9):3564–3572, May 2019.

[30] M. Lupo Pasini and M. Eisenbach. CuAu binary alloy with 32 atoms - LSMS-3 data - DOI:10.13139/OLCF/1765349.

[31] M. Lupo Pasini and M. Eisenbach. FePt binary alloy with 32 atoms - LSMS-3 data - DOI:10.13139/OLCF/1762742.

[32] U.S.R. Murty J.A. Bondy. Graphs and subgraphs. In *Graph theory with applications*. North-Holland.

[33] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv:1810.00826 [cs, stat]*, February 2019. arXiv: 1810.00826.

[34] T. N. Kipf and M. Welling. Graph attention networks. *arXiv:1609.02907 [cs, stat]*, February 2017. arXiv: 1710.10903.

[35] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Li, and Petar Velikovi. Principal neighbourhood aggregation for graph nets. *arXiv:2004.05718 [cs, stat]*, December 2020. arXiv: 2004.05718.

[36] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, September 2020.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[38] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[39] PyTorch Geometric. `https://pytorch-geometric.readthedocs.io/en/latest/`.

[40] M. Burčul and M. Lupo Pasini. GCNN. https://github.com/allaffa/GCNN.

[41] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[42] Ray Tune: Hyperparameter Optimization Framework. https://docs.ray.io/en/latest/tune/index.html.

[43] M. Eisenbach, J. Larkin, J. Lutjens, S. Rennich, and J. H. Rogers. GPU acceleration of the locally self-consistent multiple scattering code for first principles calculation of the ground state and statistical physics of materials. *Comput. Phys. Commun.*, 211:2–7, 2017.

[44] Y. Wang, G. M. Stocks, W. A. Shelton, D. M. C. Nicholson, Z. Szotek, and W. M. Temmerman. Order-N multiple scattering approach to electronic structure calculations. *Phys. Rev. Lett.*, 75:2867–2870, 1995.

[45] Y. Yang, C. C. Chen, M. C. Scott, C. Ophus, R. Xu, A. Pryor, L. Wu, G. Sun, J. Zhou, M. Eisenbach, P. R. C. Kent, R. F. Sabiranov, H. Zeng, and J. Miao. Quantitative evaluation of an epitaxial silicon-germanium layer on silicon. *Nature*, 542(7639):75–79, 2017.

[46] M. Eisenbach, Y. W. Li, O. K. Odbadrakh, Z. Pei, G. M. Stocks, and J. Yin. LSMS. https://github.com/mstsuite/lsms.

[47] Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv:1412.6980 [cs]*, January 2017. arXiv: 1412.6980.

[48] OLCF Supercomputer Titan. https://www.olcf.ornl.gov/for-users/system-user-guides/titan/.

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

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

59/2021 Stella, S.; Regazzoni, F.; Vergara, C.; Dede', L.; Quarteroni, A.
A fast cardiac electromechanics model coupling the Eikonal and the nonlinear mechanics equations

60/2021 Rosafalco, L.; Manzoni, A.; Mariani, S.; Corigliano, A.
Fully convolutional networks for structural health monitoring through multivariate time series classification

61/2021 Buchwald, S.; Ciaramella, G.; Salomon, J.; Sugny, D.
A greedy reconstruction algorithm for the identification of spin distribution

58/2021 Tassi, T., Zingaro, A., Dede', L.
Enhancing numerical stabilization methods for advection dominated differential problems by Machine Learning algorithms

56/2021 Zingaro, A.; Fumagalli, I.; Dede' L.; Fedele M.; Africa P.C.; Corno A.F.; Quarteroni A.
A multiscale CFD model of blood flow in the human left heart coupled with a lumped parameter model of the cardiovascular system

57/2021 Roberto Piersanti, Francesco Regazzoni, Matteo Salvador, Antonio F. Corno, Luca Dede', Chri
3D-0D closed-loop model for the simulation of cardiac biventricular electromechanics

55/2021 Buchwald, S.; Ciaramella, G.; Salomon, J.
ANALYSIS OF A GREEDY RECONSTRUCTION ALGORITHM

54/2021 Ciaramella, G.; Gander, M.J.; Mamooler, P.
HOW TO BEST CHOOSE THE OUTER COARSE MESH IN THE DOMAIN DECOMPOSITION METHOD OF BANK AND JIMACK

53/2021 Ciaramella, G.; Mechelli, L.
On the effect of boundary conditions on the scalability of Schwarz methods

52/2021 Ciaramella, G.; Mechelli, L.
An overlapping waveform-relaxation preconditioner for economic optimal control problems with state constraints