MagmaDNN and Ising Physics Simulations with Graph Convolutional Network

Chan, Kam Fai
kfchan1998@gmail.com

Nichols, Daniel
dnicho22@vols.utk.edu

Keh, Sedrick
sedrickkeh@gmail.com

July 2019

Abstract

Ising model is a model in materials science which simulates the properties of ferromagnetic materials by approximating the materials as groups of electric dipoles that are stacked in certain lattice structures. In our research we investigate the possibility of using techniques in machine learning, in particular the concept of a Graph Convolutional Network (GCN), a concept developed in 2013 that generalizes the concept of convolutions in general graph structures. Our result shows that despite GCN not performing as well as usual methods such as CNN with a slower convergence speed, it has shown its potential for processing problems in materials science with an intrinsic graph structure.

Contents

1 Introduction

In problems related to materials science, non-Euclidean graph structures are often encountered. These structures may come from the geometry of how the particles are stacked together, or the intrinsic difference in particles of various types that creates the irregularity in structure. It is hard to handle such graphs for the generality and the complexity of the graphs may vary greatly from sample to sample and a universal method would then be needed.

The Ising model, as a fundamental model in materials science, is closely related to these structures, for the geometry of the particles in an Ising model is not limited to regular Euclidean graphs such as square nets and cubic grids. Hence, it is crucial to start investigating how the information encoded in the structure can be extracted from investigating the problems related to Ising models.

1.1 Ising Model

Ising model [1] is one of the most fundamental models in materials science. As a model to simulate ferromagnetism of materials rich in magnetic dipoles, Ising model abstracts the properties of particles that contribute to the physical properties as discrete variables to simplify the computation with the definition as follows:

Let V be the set of dipoles in the material and $\sigma_v \in \{+1, -1\}$ be the spin of the dipole $v \in V$. Let $E \subseteq V \times V$ be the set of (undirected) edges in the model that denotes the neighbourship of the particles that determines the interactions and $J_e = J_{ij}$ is the interaction strength for the edge $e = (i, j) \in E$ between particles $i, j \in V$. Then $G = (V, E)$ forms the graph that represents the structure of the Ising model where E_{ij} is the (symmetric) weight matrix of the graph and σ is an Ising configuration for the Ising model defined by (G, J). In usual case where the numbers of neighbours the particles have are all small, G would be a sparse graph and E would be a sparse matrix.

In materials science, there is a quantity called Hamiltonian that has significant importance in mechanics. In the above Ising model and without the presence of an external magnetic field, the Hamiltonian is computed as

$$H = -\sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j$$

where E, J, σ are defined as above. According to a theory in materials science [1], a specific Ising configuration occurs in nature with probability related to its Hamiltonian. Several

physical properties of the material, including its heat capacity and magnetic susceptibility, are related to the expected value of the Hamiltonian with respect to all possible Ising configurations.

Despite the fact that Ising model originates from materials science, generalizations of such model are also used in different fields including neuroscience. [2]

1.2 Machine Learning

In our project, we focus on predicting the Hamiltonian with techniques from machine learning. As the computation of the Hamiltonian depends on the background graph structure, we consider this to be the perfect example and starting point for solving problems with intrinsic irregular structure in materials science with machine learning techniques: as far as we are aware of, there is no efficient algorithm to compute the Hamiltonian for a large number of configurations despite the importance of computing the Hamiltonian for a large number of configurations to do simulations. However, with machine learning, it is possible to do batch prediction and compute the Hamiltonian of thousands of configurations with the trade-off of losing some accuracy.

To simplify the situation we restrict ourselves to the settings of 8x8 2D planar grids with a periodic boundary condition and uniform interaction strength. In other words, $V = \{0, 1, ..., 7\}^2$ and $((i_1, j_1), (i_2, j_2)) \in E$ iff $|i_1 - i_2| + |j_1 - j_2| = 1$ or $i_1 = i_2 \in \{0, 7\}, \{j_1, j_2\} = \{0, 7\}$ or $j_1 = j_2 \in \{0, 7\}, \{i_1, i_2\} = \{0, 7\}$. Also, $J_e = 1$ for all $e \in E$. We choose such setting for the reasons that (1) the sample space is large enough for a neural network to do prediction, (2) extremal Hamiltonian configurations can be constructed and (3) it allows using regular CNN for benchmarking.

1.3 Prior Work

There are some previous results on developing algorithms on investigating Ising models on graphs, including using L1-regularization [3] and the graph convolution method proposed by Henaff et al.[4]. There are also results on investigating data with intrinsic graph structures, including documentation classification [5] and predicting graph-value outcome in biology [6].

At the best of our effort, we have yet to find research focusing on estimating Hamiltonian on graph Ising model, especially with machine learning. We believe the reason behind this is that in most cases where the number of configurations in interest is not large enough that the straightforward method is already sufficient to give a satisfying performance. However, as the size of data needed to be investigated increases along with the development of materials science, we believe further research in our topic is necessary.

2 Method

2.1 Graph Convolutional Network

One of the most significant developments made in the field of machine learning is the development of convolutional layers which are able to extract features from small local regions

and enable the advanced development of different branches such as computer vision. As the computation of Hamiltonian also requires extracting features of local regions, it is natural to do convolutions on general graphs. However, as the number of neighbours and their locations of a particle may vary greatly across the model, the filter used in convolutional layers can no longer be defined in the same way. It is then natural to use the Graph Convolution Network (GCN), a concept developed to capture the local properties of a graph.

2.1.1 Graph Fourier Transforms and Convolutions

The mathematical definition of convolution on graphs is developed by Shuman et al. [7]. The idea of this is to use a Fourier transformation to convert the signal on the graph to the spectral domain and do the convolution there. After transforming the signal back to the spatial domain, a convoluted signal would be obtained.

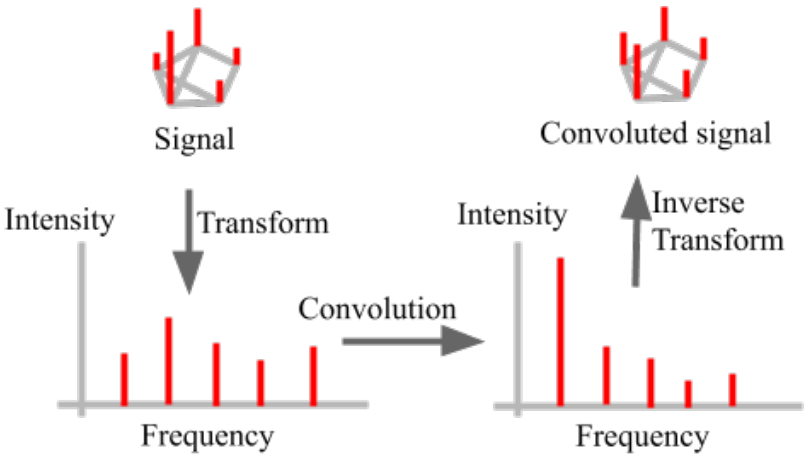

Figure 1: Schematic illustration of how graph convolution is computed

In their paper, Shuman et al. generalized the concept of convolution using the graph Laplacian, a matrix that represents the intrinsic relations between vertices [7]. For a simple undirected graph with negative edge weights $G = (V, E)$ with adjacency matrix A, the graph Laplacian is defined as $L = A - D$ where $D_{ii} = \sum_j A_{ij}$ is the (diagonal) degree matrix. A signal on the graph $f \in \mathbf{R}^V$ can be transformed as $\tilde{f} = U^T f$ and $f = U\tilde{f}$ where $L = U\Lambda U^T$ is a eigendecomposition of the graph Laplacian with U being orthogonal. Using the normalized graph Laplacian $\tilde{L} = D^{-1/2}LD^{-1/2}$ rather than the original graph Laplacian is also suggested.

2.1.2 Graph Convolutional Layers

In the original idea that Shuman et al. developed [7], in order to do graph convolution it is required to first do eigendecomposition on the graph Laplacian, which could potentially be a massive matrix and be time-consuming. Hence we adopted the graph convolution layer developed by Kipf and Welling[8] for its simplicity and efficiency.

The layer developed by Kipf and Welling is formulated as following [8]: with the input sample being X, the output of the layer is

$$Y = \bar{L}XW$$

where

$$\bar{L}_{ij} = (1 + \sum_k A_{ik})^{-1/2}(1 + \sum_k A_{jk})^{-1/2}(A + I)_{ij}$$

is the normalized graph Laplacian for the augmented graph where a self-loop of unit weight is added to each vertex and W is weight matrix representing the filter spectrum used for different channels.

One advantage of using this layer is that only 2 matrix multiplications, one of which is sparse-dense matrix multiplication, are needed for forward-propagation rather than doing eigendecomposition. Moreover, the new graph Laplacian can be precomputed efficiently with techniques such as parallelism. In later parts of this article, we would refer to this layer as the KW layer.

2.2 MagmaDNN

MagmaDNN [9] is an open source neural network framework in C++ which has excellent performance when compared to other mainstream frameworks including Tensorflow and Theano [10]. For its flexibility and extensibility, it is easy to have an efficient implementation of the graph convolution layer required. For these reasons, we choose to use MagmaDNN for implementing the models.

2.2.1 Computation and Sparse Linear Algebra

In most cases, the graph would be a sparse graph with high sparsity as particles usually only interact with particles within a short Euclidean distance. In this case the workload in the aforementioned layer can be reduced using routines about sparse-dense matrices multiplication especially when the order of the graph is too large for machine memory to hold the uncompressed graph Laplacian or when the sparsity is near 1. However when the library used lacks batch sparse-dense multiplication routine, using sparse matrix multiplication may damage the performance as multiplication may then be evaluated sample-by-sample. In our implementation, the graph Laplacian used in the layer is passed as a dense matrix rather than as a sparse matrix despite the sparsity is $1 - \frac{4 \cdot 64 + 64}{64 \cdot 64} = 92.19\%$ to utilize the gemmStridedBatched routine in cuBLAS due to the lack of appropriate strided and batched sparse-dense multiplication routine in cuBLAS[11] and Magma[12].

2.3 Data augmentation

To train a model, it is necessary to first obtain a sufficient amount of training data. As configurations with Hamiltonian around 0 are the most common ones in number, special methods are required to generate the data for otherwise the model would naturally bias towards predicting near-zero values. Hence we used the Wang-Landau algorithm[13] to generate the data. However, despite the algorithm can generate configurations with less common

Hamiltonian values, it is still unable to generate configurations with extremal Hamiltonian as there are fewer than 100 configurations for each of such values out of around $2^{64} \approx 2 \times 10^{19}$ possible configurations.

To ensure there are enough samples for each possible Hamiltonian value, techniques in data augmentation are applied. Due to the intrinsic symmetry in our setting, configurations can be flipped, rotated and negated (i.e. swapping +1 and -1) while having the Hamiltonian unchanged. Moreover, it is easy to compute the new Hamiltonian from the original one if the configuration is slightly perturbed. With these techniques, we have generated 2079000 samples with similar proportion for each of the possible Hamiltonian value among which 176400 samples are used as the training dataset.

3 Results

We have trained a GCN model along with a CNN model for benchmarking on XSEDE Bridges on an NVIDIA Tesla P100 GPU. The structures of the models are shown in the graph.

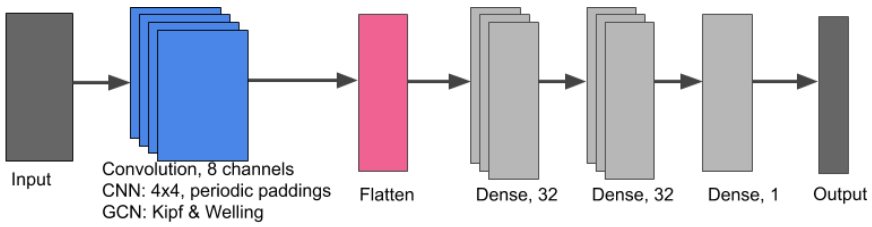

Figure 2: Structures of the models

To help the CNN capture the periodic boundary condition, periodic boundary padding is added to the data before feeding into the CNN model.

We have trained the models with 30 epochs, batch size 6300 and learning rate 0.2 and obtained the result as shown in the table:

	Train		Test	
	MAE	RMSE	MAE	RMSE
CNN	2.50	3.37	2.51	3.38
GCN	5.17	7.08	5.18	7.09

Table 1: Outcomes of the models. MAE stands for mean absolute error, RMSE stands for Root-mean-squared error

4 Analysis and Conclusion

One can notice that the loss for the GCN model drops rapidly only at the beginning but for the CNN model the initial rapid drop continues until the fourth epoch. We believe this

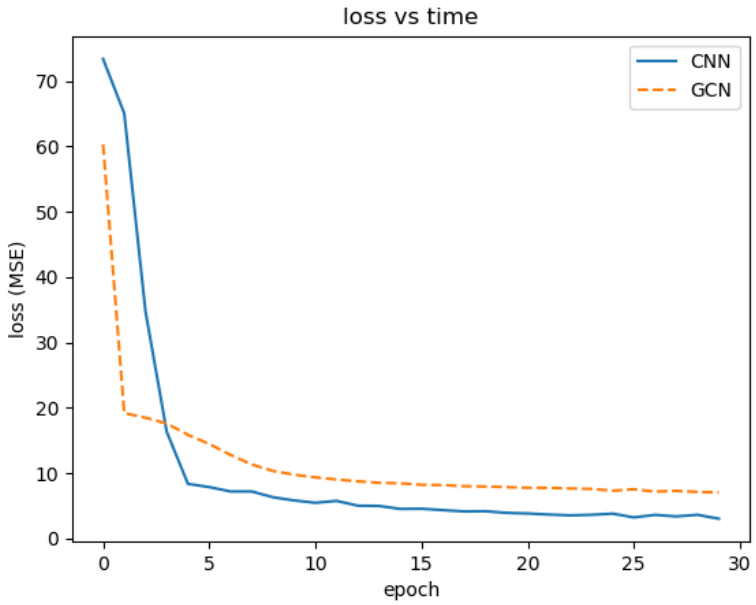

Figure 3: Evolution of the models

difference comes from the structural difference between the usual convolution layer and the KW layer. Our choice of learning rate (0.2) may also affect the situation.

It is also worth noticing that the GCN model converges slower than the CNN in later epochs. We believe that this is due to the fact that in usual convolutional layers a large part of the information of the filter, including the effective size of the filter, is defined as hyperparameters and only the weight is learned, while in graph convolution the whole filter is learned as a signal spectrum, hence increasing the amount of information that must to be learned by the model and slowing down the convergence speed.

However, with the ability to process general graphs with only some loss in convergence speed, we are convinced that GCN is a good substitute for CNN in problems related to materials science with non-Euclidean structure.

5 Future Work

As a starting point of using graph convolution techniques on materials science related problems, we have only used the KW layer [8] despite the existence of other schemes of GCN layer [14]. It would be interesting to compare the KW layer with different GCN layers.

It is also worth noticing that in usual convolutional layers samples are naturally downgraded while in graph convolution layers only the signal strength is modified. It is then necessary to apply pooling on the graph signal which requires graph clustering. Several methods exist [14] and may help in boosting the performance of the GCN layers.

At the moment there is no native routine in MagmaDNN that can satisfy our need in implementing the layer with batched sparse-dense matrix multiplication with strides. We expect to see a performance boost with such routine.

6 Acknowledgments

Research was done with support from the National Science Foundation, Oak Ridge National Laboratory, Joint Institute of Computational Sciences, University of Tennessee, Knoxville, the Chinese University of Hong Kong, and Hong Kong University of Science and Technology. This work would not be possible without assistance and mentorship provided by supervisors Dr. Kwai Wong, Dr. Stanimire Tomov, Dr. Ying Wai Li, Dr. Markus Eisenbach, Dr. Massimiliano Lupo Pasini, and Dr. Junqi Yin.

References

[1] M. Loebl. Some discrete tools in statistical physics. *Physics and Theoretical Computer Science*, pages 317–332, 2006.

[2] R. Segev W. Bialek E. Schneidman, M. J. Berry II. Weak pairwise correlations imply strongly correlated network states in a neural population. *arXiv:q-bio/0512013 [q-bio.NC]*, 2005. arXiv:q-bio/0512013.

[3] G. Bresler. Efficiently learning ising models on arbitrary graphs. In *STOC '15 Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 771–782, Portland, Oregon, USA, June 14-17 2015. ACM New York, NY, USA.

[4] Y. LeCun M. Henaff, J. Bruna. Deep convolutional networks on graph-structured data. *arXiv:1506.05163 [cs.LG]*, 2015. arXiv:1506.05163.

[5] K. Cho M. Chen, Z. Lin. Graph convolutional networks for classification with a structured label space. *arXiv:1710.04908 [cs.*, 2017. arXiv:1710.04908.

[6] Vladimir Golkov, Marcin J Skwark, Antonij Golkov, Alexey Dosovitskiy, Thomas Brox, Jens Meiler, and Daniel Cremers. Protein contact prediction from amino acid co-evolution using convolutional networks for graph-valued images. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4222–4230. Curran Associates, Inc., 2016.

[7] P. Frossard A. Ortega P. Vandergheynst D. I Shuman, S. K. Narang. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv:1211.0053 [cs.DM]*, 2012. arXiv:1211.0053.

[8] M. Welling Th. N. Kipf. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907 [cs.LG]*, 2016. arXiv:1609.02907.

[9] Magmadnn site, 2019. `https://bitbucket.org/icl/magmadnn`.

[10] N-S. Tomov F. Betancourt S. Tomov K. Wong Nichols, D. and J. Dongarra. Magmadnn: Towards high-performance data analytics and machine learning for data-driven scientific computing. Frankfurt, Germany, June 2019. Springer International Publishing.

[11] Api reference guide for cublas, 2019. https://docs.nvidia.com/cuda/cublas/index.html.

[12] Stanimire Tomov, Jack Dongarra, and Marc Baboulin. Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Computing*, 36(5-6):232–240, June 2010.

[13] D. P. Landau Fugao Wang. An efficient, multiple range random walk algorithm to calculate the density of states. *arXiv:cond-mat/0011174 [cond-mat.stat-mech]*, 2000. arXiv:cond-mat/0011174.

[14] Pytorch geometric documentation, 2019. https://pytorch-geometric.readthedocs.io.