Summer 2023

Extending the Convolution in Graph Neural Networks to Solve Materials Science and Node Classification Problems

Steph-Yves Mike Louis

EXTENDING THE CONVOLUTION IN GRAPH NEURAL NETWORKS TO SOLVE MATERIALS SCIENCE AND NODE CLASSIFICATION PROBLEMS

by

Steph-Yves Mike Louis

Bachelor of Science
Belmont Abbey College, 2015

Master of Science of Public Health
University of South Carolina, 2019

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2023

Accepted by:

Jianjun Hu, Major Professor

Tong Yan, Committee Member

Ming Hu, Committee Member

Forest Agostinelli, Committee Member

Qi Zhang, Committee Member

Ann Vail, Dean of the Graduate School

ACKNOWLEDGEMENTS

I would like to foremost dedicate this degree to my God Yahweh, who has abundantly blessed me throughout my life. I am the most grateful and thankful for advisor Dr. Jianjun Hu. From the time that Dr. Hu first taught me Python, he has been an amazing mentor with advice and feedback to I intend to carry with me for the rest of my scientific career. Without his constant encouragement and push, I would have certainly not come to this point.

I would like to also extend my gratitude to my dissertation committee members, Dr. Yan Tong, Dr. Ming Hu, Dr. Qi Zhang, and Dr. Forest Agostinelli for their time, advice, and support in this research. I feel very honored to have them as committee members.

Furthermore, I take to the occasion to also thank all my closest research collaborators and graduate colleagues Dilanga, Yong, Ali, Yuqi, Sadman, Ruxin, and Yuxin. It has been a thrill to have them as companions during this adventure. Last, I want to also thank my immediate and extended family for their unconditional love and support during this exciting adventure.

ABSTRACT

The usage of graph to represent one's data in machine learning has grown in popularity in both academia and the industry due to its inherent benefits. With its flexible nature and immediate translation to real life observed objects, graph representation had a considerable contribution in advancing the state-of-the-art performance of machine learning in materials.

In this dissertation proposal, we discuss how machines can learn from graph encoded data and provide excellent results through graph neural networks (GNN). Notably, we focus our adaptation of graph neural networks on three tasks: predicting crystal materials properties, nullifying the negative impact of inferior graph node points when learning, and generating crystal structures from material formula.

In the first topic, we propose and evaluate a molecule-appropriate adaptation of the original graph-attention (GAT) model for materials property prediction. With the changes of including the encoded bonds formed by atomic elements and adding a final global-attention layer, our experiments show that our approach (GATGNN) achieves great performance and provides interpretable explanation of each atom.

For the second topic, we analyze the learning process of various well-known GNNs and identify a common issue of propagating noisy information. Aiming to reduce the spread of particularly harmful information, we propose a simple, memory-efficient, and highly scalable method called NODE-SELECT. Our results demonstrate that the combination of hard attention coefficients, binary learnable selection parameter, and

parallel arrangement of the layers significantly reduce the negative impact of noise data propagation within a GNN.

In the third topic, we extend the development of our GATGNN method and apply it to simulate electrodes reaction for predicting voltages.

Finally, we propose a generative approach for generating crystal structures. In this approach, we employ both GNN, variational autoencoder (VAE), and transformers to learn 3-dimentional space positioning of the elements within a unit-vector. This technique shows promising results based on the technique's proof of concept and early results.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In the past few years, Graph Neural Networks (GNN) have gained a lot of traction in the field of machine learning due to their flexible nature and effectiveness at reaching state-of-the-art performance. These recent adaptations of GNNs range across various fields and applications such as Physics, Chemistry, Biology, Computer Vision, Natural Language Processing, Recommendation Systems, and Social or Computer Networks[1, 2]. Notable studies examples from the aforementioned fields include particles tracking and reconstruction [3, 4], predicting binding affinities [5], semantic role labelling [6], and Pinterest' recommendation system [7]. As graph neural networks grew in popularity, other innovative and effective techniques have in parallel also been developed and adapted to further push the state-of-the-art performance in artificial Intelligence.

Besides the obvious benefits that come from using graph neural networks on graph encoded data and those data's relationships, GNN have demonstrated to be very powerful tools that quite often achieve state-of-the-art results for various tasks. Graph neural networks implementations are done with the end goal of either node prediction, edge prediction, graph generation, relationship extraction, or graph coarsening [1, 2, 8, 9]. Notably, the success of graph neural networks can be attributed to their flexible nature for handling custom data and adopting techniques of deep learning innovations or other architectures. Examples of such techniques include recurrent operators adopted from NLP, attention, parse-trees, GAN, VAE, Diffusion, dropout, skip connections, etc [10, 11].

With their wide net of applications and their extensibility, graph neural networks now figure among the top Deep Learning techniques to consider for learning from relational data structures.

1.2 SCOPE OF THE PROPOSED RESEARCH

In this dissertation, we focus on the following three aspects of using graph convolutional neural networks:

1- The first aspect of work involves the task of graph classification / graph prediction. Materials structure prediction is one of the most important topics in materials science. The subject of properties prediction through Deep Learning has widely been studied, but only few works had successfully adapted GNN [12, 13]. In contrast to the prior adaptations of graph convolutional networks (GCN), we investigate the application of graph-attention network technique into predicting 7 physical properties of crystal materials [14]. We propose a method called GATGNN, based on the prior work of graph attention network, to predict the graph encoded materials and provide interpretable insights of each materials' atoms importance to the property of interest.

2- The second aspect of our work focuses on the topic of node classification. In this project, we develop a graph neural network that can cancel specific noisy neighbor's messages during a graph convolution. We apply this framework onto various transductive graph datasets to demonstrate its effectiveness at carrying out its selection mechanism. Our experiments show that such mechanism also offer inherent benefits of scalability and memory efficient meanwhile being resistant to noise propagation.

3- In the future work, we propose to use a variational auto-encoder (VAE) approach in combination with graph convolutional networks to generate graphs. In this method, we take on the difficult challenge of adapting graph convolutional networks to create Physically valid materials structures given a materials formulation, such as $TbGa_3$. From our first preliminary experiments, we have found that our generative model outputs promising results.

1.3 STRUCTURE OF THE DISSERTATION

In chapter 2, we present a method to predict the structural properties of crystal materials and discuss our experimental results. In chapter 3, we report on the development of a new graph convolutional network variant. The highlight of this new variant is its capability to selectively aggregate useful feature information from neighbors. Our future work is presented in chapter 4, and we conclude our work in chapter 5. We list our proposed future work schedule in chapter 6.

CHAPTER 2

GRAPH CONVOLUTIONAL NEURAL NETWORKS WITH GLOBAL ATTENTION FOR IMPROVED MATERIALS PROPERTY PREDICTION

2.1 INTRODUCTION

Machine learning and deep learning [15-17] have been increasingly used in the field of materials science on various applications [18-20] such as rechargeable alkali-Ion batteries, photovoltaics, catalysts, thermoelectrics, superhard materials [21], and superconductors [22]. The two key components of a machine learning model for materials property prediction are the set of features and the particular algorithm. Currently, there are two main categories of features that are widely used: the composition based features [23, 24] and structure based features [12, 13, 25-27]. The former has the benefit of being able to be applied to discover new hypothetical materials while the latter has higher prediction performance but only applicable to materials with characterized structure information either experimentally or by computational crystal structure prediction software such as USPEX [28], which, however, can only predict the structure for relatively simple compositions.

Numerous structural descriptors have been proposed to represent materials such as atom-centered symmetry functions, Coulomb matrix, smooth overlap of atomic positions, deep tensor neural networks, many-body tensor representation, and Voronoi tessellation [19]. Nevertheless, each one of these descriptors have their limitations such as: not being size-invariant by construction, or basing their representation of infinite crystals on local neighborhoods of atoms in the material [19]. The definition and exploitation of local atomic environment is also important in developing neural network potential models [29]. Extensive discussions over these structure descriptors and the characteristics of desired structural descriptors such as invariance to translation, rotation, and permutation of homonuclear atoms can be found in recent reviews [19, 30].

Recently, graph neural networks have gained a lot of attention in materials property prediction [12, 13, 31] due to high representation learning capabilities and their ability to achieve state-of-the-art results for various problems of classifying graph entities or graph nodes [32]. Xie et al. [13] figured among the first researchers to apply graph neural networks to materials property prediction. The former authors achieved impressive results based on their algorithm and their crystal representation as graph. Notably, the encoding consists of representing the unit cell of the crystal material as a graph such that nodes represent the atoms and connecting edges represent the bonds shared amongst the atoms. A direct benefit of representing the crystal material as a CGCNN-converted graph is the naturally derived vector characterization of the atoms and edges. Chen et al. [12] improved the CGCNN model by introducing a global state input including temperature, pressure and entropy. They also found that the element embeddings in their MEGNet models encode periodic chemical trends and can be used for transfer learning based training of models for band gaps and elastic moduli prediction which have limited training data using the embedding learned by models training for formation energy prediction. In another effort to improve CGCNN, Park et al.[31] proposed an approach to incorporate information of the Voronoi tessellated crystal structure, explicit 3-body correlations of neighboring constituent atoms, and an optimized chemical representation of interatomic bonds in the crystal graphs.

While previous graph neural network models for crystal materials property prediction only emphasized on capturing local atomic environment, we propose a deep graph neural network named GATGNN, that is based on a global attention mechanism. The attention mechanism was first introduced into neural networks for natural language

processing (NLP) [20, 33]. Basically, attention is simply a vector which can be used to learn the contribution of different context vector components. It has been used to replace the recurrent neural networks and has achieved significant successes in NLP [34-37]. In this model, we first use local attention layers to capture properties of local atomic environments and then a global attention layer is used to make weighted aggregation of all these atom environment vectors to create the global representation of the whole crystal structure. This allows our model to better capture the fact that different atoms in the crystal have different contributions to the global material property. In materials science, Coley et. al. [25] firstly proposed a global attention mechanism with graph neural network for chemical reactivity prediction. In their model, the global attention coefficient of each atom is calculated by learning a reaction probability of that atom to all other possible (physically possible) matching. In our work, the coefficient of an atom is calculated by learning either one of 1) its importance based on its location in the graph or 2) its energy contribution to the crystallization of the material.

The contributions of our work include:

- We propose a global attention mechanism with graph neural network for material's property prediction.
- Benchmark studies have shown the state-of-the-art performance of our GATNN algorithm in a variety of materials property prediction problems.
- Ablation experiments have been conducted to demonstrate the advantage of GATGNN.
- We have extracted physical insights by examining the learned weights of the GATGNN.

2.2 METHODS

2.2.1 DATA COLLECTION & ENCODING

The Materials Project database is used to collect the properties of all the crystal materials originally used in the works of CGCNN [13] and MEGNET [12]. For the 46,743 CGCNN materials, we apply the same encoding that the authors Xie et. al. used to transform the materials into graphs. Notably, we connect each atom to the 12 nearest atoms in the material, transform the bond distance between each two atoms to a 41-dimensional feature vector, and apply a 16-dimensional feature vector to encode each atom feature. On the other hand, we modified the graph constructing parameters for the MEGNET inorganic materials. Particularly, for the 69,240 materials used by MEGNET, we connect each atom to the nearest 16 atoms, transform the bond distance between each two atoms to a 9-dimensional feature vector, and use the same 16-dimensional feature vector encoding as for CGCNN.

2.2.2 AUGMENTED GRAPH-ATTENTION LAYER

First, we define a graph as the tuple $(V, E, \boldsymbol{A})$ where V defines a set of nodes, E a set of edges, and $\boldsymbol{A}$ the graph's adjacency matrix. Letting $v \in V$ and $e \in E$ be the feature vectors of atoms and edges, we explain the core of our proposed architecture that is based on the Graph Attention (GAT) Network proposed in 2018 by Velickovice et al. [14]. Introduced as a flexible way to adapt the attention mechanism in graph neural networks, GAT Networks proposed a viable solution for dealing with unequal importance of neighboring nodes in graph networks. Making use of the attention heads and context vectors, the GAT architecture makes it possible to learn the importance of the

information obtained from neighboring nodes. Notably, the GAT network allows each node to weigh the information from each neighboring node and the learning of these weights can be described by the following operation:

$$\alpha_{ij}^{(k)} = softmax\left(g\left(a_{ij}^{(k)}\right)\right) = softmax\left(g\left(\boldsymbol{a}^T\left[\boldsymbol{W}^{(k)}\mathbf{v}_i^{(k-1)} || \boldsymbol{W}^{(k)}\mathbf{v}_j^{(k-1)}\right]\right)\right) \quad (2.1)$$

where $\boldsymbol{a}$ and $\boldsymbol{W}$ define the weight vector and weight matrix of layer k, while $\mathbf{v}$ and function $g(\cdot)$ define a node feature vector and a non-linear function. For two connected nodes i and j, a weight α_{ij} is calculated after applying the softmax function to the attention coefficient a_{ij} of their connection. The attention coefficient a_{ij} is learned via linear transformation of the newly transformed feature vectors of nodes i and j combined by $\boldsymbol{a}$. Once the attention weights are learned, the updated node feature vector is calculated by the following convolution,

$$\mathbf{v}_i^{(k)} = \sigma\left(\sum_{j \in N(i) \cup j} \alpha_{ij}^{(k)} \boldsymbol{W}^{(k)} \mathbf{v}_i^{(k-1)}\right) \quad (2.2)$$

, where $N(i)$ defines the neighborhood of node i and σ the sigmoid function. While GAT networks use the information characterizing neighboring nodes, the original architecture has the limitation of not using any available edge information when learning the attention weights. As a solution to the latter shortcoming, we augmented the original GAT layer. Particularly, we improved the learning capability of the GAT layer by adding Batch-Normalization layers and augmenting the node features vectors with the information from their connecting edge as seen below,

$$\mathbf{v}_i^{*(k)} = \mathbf{v}_i^{(k)} || \, \boldsymbol{e}_{ij} \tag{2.3}$$

where $\mathbf{v}_i^{*(k)}$ defines the augmented feature vector of node i found by combining $\mathbf{v}_i$ and $\boldsymbol{e}_{ij}$, the feature vector of the edge connecting node i and its neighbor j. Henceforth, our adaptation of the GAT network for our properties prediction comprises multiple augmented-GAT (AGAT) layers, a global feature pooling layer, and 2 hidden layers. Compared to the properties prediction obtained from applying the original layer, the results of our AGAT layer improved the prediction performance scores by an average of 20%. By using the batch-normalization, softplus non-linear activation function, and the average of 4 attention heads, we further optimized the learning capability of the AGAT layer. Nonetheless, just the application of the AGAT layers to extract the node features resulted in much poorer predictive performance than previous models from CGCNN and MEGNET works. Certainly, graph neural networks provide strong performance when the main goal is the relationship extraction in a local neighborhood. However, graph neural networks do not use the position of the nodes within the entire graph. Corresponding to the theoretical work, the location of an atom with respect to the structure of the material is critically important and not just to the atoms within the vicinity of its site.

Inspired by the works of Coley et. al. in which different weights are assigned to varying reactions [25], we propose a global attention layer, before the pooling layer as seen in Figure 2.1, that is adapted to particularly translate the information directly learned from a local neighborhood to information at the graph level with meaningful interpretability.

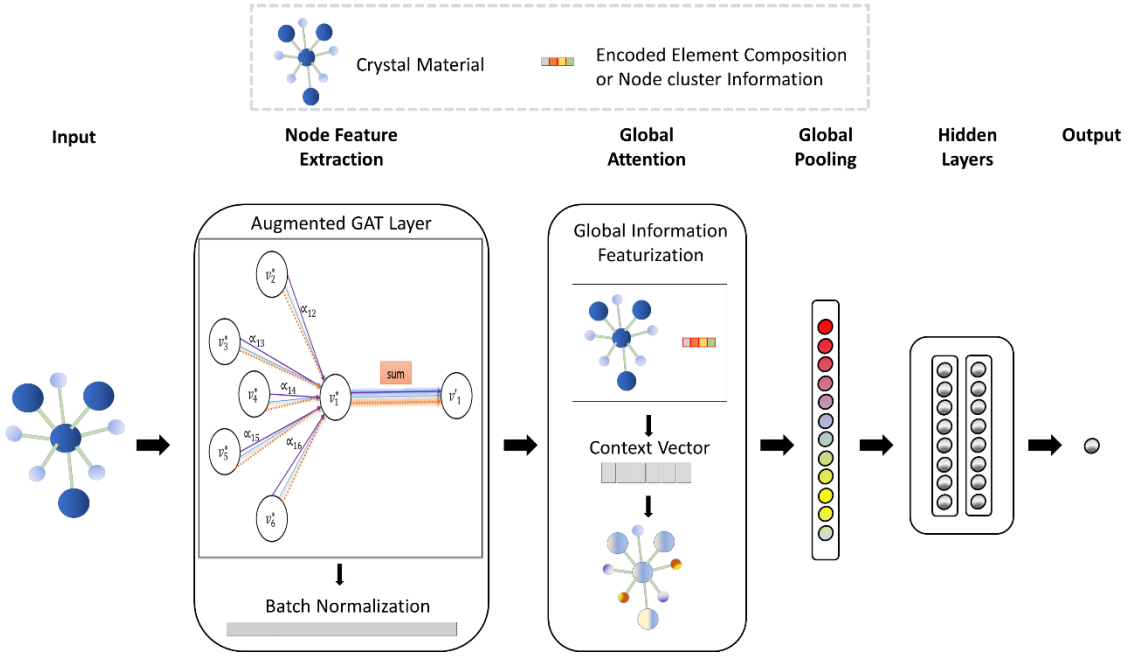

Figure 2.1 Architecture of our global attention graph CNN model GATGNN. Each model is composed of either 5 or 7 of our AGAT layers with 64 neurons. After extracting the node features, a global attention layer is placed before the global pooling of the node features. Finally, the weighted sum of the crystal feature vector is afterwards fed to two hidden fully-connected layers before a final fully-connected layer outputting the predicted property.

2.2.3 CRYSTAL GLOBAL ATTENTION MECHANISM

In this work, two types of global information (GI) are used as additional inputs to construct the Global Attention Layer: a feature vector characterizing the entire crystal graph and a feature vector denoting the crystal node's location in the graph. To represent the feature vector characterizing the entire graph, we used the crystal's elemental composition E (a vector that maps the ratio amount of each element). On the other hand, we represent node i's location in the graph by its cluster a: C_a^i, obtained by feeding its coordinates either the Spectral-Clustering or the K-Means algorithm. In our experiments, only three clusters were used, which shows the best performance in our experiments.

Then, the global information is propagated to the node feature vectors $\mathbf{v}_i^{(p)}$output by the last AGAT layer as defined in table 1. Depending on the choice of the global information, the GI method of propagation differs on either concatenation or deconvolution.

Table 2.1 Description of the global information used and the different propagation method throughout the nodes.

Global Information (GI)	**GI Propagation**	**Formulation**	**Method Identifier**
Graph's elemental Composition E	Concatenation with node feature	$\mathbf{v}_i^{(p)} = \mathbf{v}_i^{(p)} \mid\mid E$	GI M-1
Node i's Cluster	Fixed Cluster Unpooling	$\mathbf{v}_i^{(p)} = \sum_{j \in C_a^i} \mathbf{v}_j^{(p)}$	GI M-2
	Random Cluster Unpooling	$\mathbf{v}_i^{(p)} = \sum_{j \notin C_b} v_j^{(p)} \ni C_b \neq C_a^i$	GI M-3
	Concatenation of graph's pooled feature vector G, node features, and one-hot encoding of node clusters	$\mathbf{v}_i^{(p)} = G \mid\mid \mathbf{v}_i^{(p)} \mid\mid C_a^i$	GI M-4

For the choice of the crystal's elemental composition E, the GI propagation method consists of concatenating E to the feature vector of each node $\mathbf{v}_i^{(p)}$. Figure 2.2 provides an illustration of the schema of the propagation method of the elemental composition. For using the node's location as global information, we propose two new strategies. The first strategy defines the concatenation of the pooled feature vector of the crystal graph G (eq. (2.4)), the feature vector of each node $\mathbf{v}_i^{(p)}$ and the one-hot encoded cluster information of each node C_a^i. The combination of the nodes' feature vector and cluster information with G, is then fed to a single feed-forward layer which outputs a new feature vector encoding the node's location.

$$G = \sum_{i=1}^{N} \mathbf{v}_i^{(p)} \tag{2.4}$$

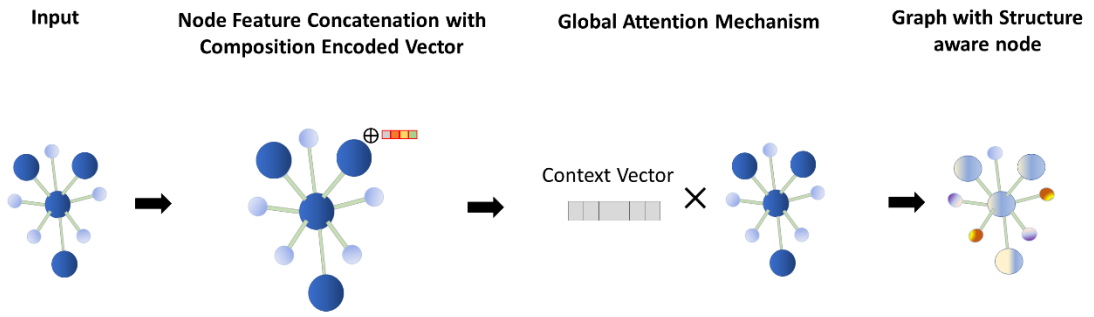

Figure 2.2 Illustration of the global information propagation for the material's elemental composition through the nodes of the crystal. Each crystal node's feature vector is concatenated with the composition encoded vector, which outputs a context vector after forwarded to two fully connected layers.

Our last strategy for propagating each node's position throughout the entire crystal, illustrated in Figure 2.3, is based on our operations of pooling and unpooling the cluster features of the graph nodes. Upon separating the crystal nodes into clusters, we apply our cluster-pooling operation, which is the summation of the feature vector of all nodes belonging to that specific cluster. The next operation, the cluster unpooling or deconvolution, consists of replacing the feature vectors of each node by cluster-pooled feature vectors. Each node's feature vector can be replaced either randomly or non-randomly (fixed). In the fixed unpooling method, each node's feature vector is replaced by the summed feature vector of the cluster that node belongs to. On the other hand, the random unpooling method defines the method in which each node's feature vector is replaced by the summed feature vector of any cluster that the node did not belong to. Essentially, our proposed global attention layer relies on the distribution of a meaningful and universal information though all the nodes in the crystal graph. Once all of the graph nodes' feature vectors are updated, they are fed to a feed-forward neural network with

two fully connected layers and a softmax layer outputting a coefficient vector $\boldsymbol{c}$. Notably, that coefficient vector $\boldsymbol{c}$ contains all the corresponding coefficient c_i for each node i in the graph. Finally, each c_i is then multiplied by the corresponding feature vector $\mathbf{v}_i^{(p)}$ of the matching node.

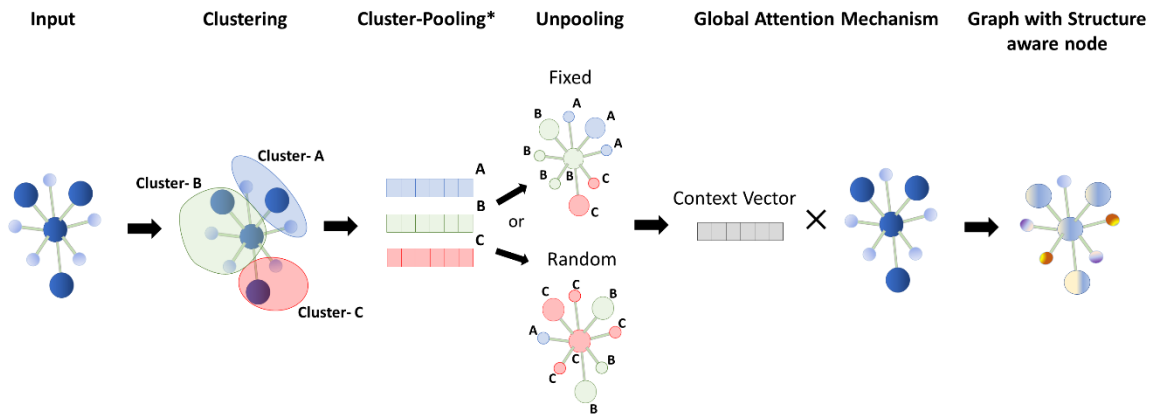

Figure 2.3 Illustration of two propagation methods for the node's cluster. The Cluster-Pooling * defines the aggregation of the feature vectors of all nodes that belong to each of the three clusters. The clusters feature vectors are redistributed through the all the nodes in the unpooling method, which is either fixed or random. After feeding the crystal graph with updated node feature vectors, a context vector is output. The context vector, which contains the weight relating to each node's location, is then multiplied with the input crystal node feature vectors to provide a graph with Structure aware nodes.

2.2.4 MODEL CONSTRUCTION AND TRAINING

To construct our models, we used the open-source library of Pytorch [38] and its geometric deep learning extension of Pytorch-geometric [39]. A model for each designated property was trained for a total of 500 epochs using early-stopping with a patience parameter 150. The additional model hyper-parameters describe a learning rate initiated at $5 \cdot 10^{-3}$ which later reduces to $5 \cdot 10^{-4}$ and then to $5 \cdot 10^{-5}$, a batch-size of

256, 5 or 7 of our AGAT layers with 64 neurons, 4 attention-heads, the Glorot uniform weight initialization, the SmoothL1loss as loss function, and the AdamW optimizer.

The separation of the data into training, testing, and validation sets were carried as described in the original works [12, 13]. A split of 60:20:20 of the whole datasets in CGCNN study was used to train and evaluate our models while a split of 80:10:10 was used for training and evaluating our models as is done in MEGNET study. Given the large enough size of the materials dataset, note that the specification of the exact seed doesn't significantly affect the reported results. All of the models were trained on Nvidia GeForce RTX 2080 Ti GPUs. During the training process, the model parameters leading to the lowest validation error are saved to construct the final model.

Our dataset and code can be found at --- https://github.com/superlouis/GATGNN

2.3. RESULTS

2.3.1 PREDICTION PERFORMANCE IMPROVEMENTS OVER MEGNET AND CGCNN

Table 2.2 provides a summary of the comparison between our models' performance and the models of CGCNN. From the reported mean absolute error (MAE) scores, our models outperform the CGCNN models in nearly all properties prediction problems. Except for the formation-energy prediction problem for which the scores matched, the absolute amount in performance improvement over the previous CGCNN results range from 2.3% to 33%. Material properties with 10% score improvement were observed including Absolute-Energy, Band-Gap, and Bulk-Moduli as shown in the table.

Table 2.2 Performance (MAE) comparison over seven materials property prediction problems of our model compared to CGCNN. The number of training samples used for each model is indicated in parentheses.

Properties	CGCNN	GATGNN (this work)	Units
Formation energy	0.039 (28,046)	0.039 (28,046)	eV/atom
Absolute energy	0.072 (28,046)	**0.048** (28,046)	eV/atom
Fermi energy	0.363 (28,046)	**0.33** (21,885)	eV/atom
Band gap	0.388 (16,485)	**0.322** (16,485)	eV
Bulk-moduli	0.054 (2,041)	**0.047** (2,041)	log(GPa)
Shear -Moduli	0.087 (2,041)	**0.085** (2,041)	log(GPa)
Poisson-Ratio	0.030 (2,041)	**0.029** (2,041)	--

Consistent with the performance comparison with CGCNN, our GATGNN models likewise outperform the MEGNET models in the predictions of three out of four properties as shown in Table 2.3 which displays the side-by-side comparison of the MEGNET results compared to our models. While our models improve the prediction results reported by MEGNET in predicting the band-gap, bulk-moduli, and shear-moduli properties, our models underperformed in the prediction of the formation energy property. The lower performance of our model for the formation energy prediction compared to those of both CGCNN and MEGNET suggests one potential limitation of our models.

Table 2.3 MAE comparison of four materials properties prediction of our GATGNN model compared to MEGNET. The number of training samples used for each model is indicated in parentheses.

Properties	MEGNET	GATGNN (this work)	Units
Formation energy	0.028 ± 0.000 (60,000)	0.039 (28,046)	eV/atom
Band gap	0.33 ± 0.01 (36,720)	**0.31** (36,720)	eV
Bulk-moduli	0.050 ± 0.002 (4,664)	**0.045** (4,664)	log(GPa)
Shear -Moduli	0.079 ± 0.003 (4,664)	**0.075** (4,664)	log(GPa)

2.3.1 ABLATION EXPERIMENTS

In Table 2.4, we provide the MAE comparison for the prediction of the Band-Gap, Bulk-Moduli, and Shear-Moduli properties for all the attention models investigated in this work. Without our minor improvements and the usage of edge information, the original GAT demonstrated a very poor predictive ability for the three tested properties. While the AGAT model recorded comparable results to CGCNN, all of our global information propagation solutions improved its performance with the added attention coefficient vector. For each one of the three properties predictions, the four solutions provided very similar results. The reported descriptive statistics for the Band-Gap, Bulk-Moduli, and Shear-Moduli properties listed as 0.33 ± 0.005, 0.0495 ± 0.002, and 0.0868 ± 0.0015 indicate that the application of the weight coefficients from any GI-propagation method would yield better prediction than without its usage. Even though the third GI-Method of Random Unpooling didn't yield the lowest MAE for either one of the properties, it should be noted that this method showed the most consistency of improving the AGAT performance. In two out of the three properties, the usage of the crystal elemental composition provided the lowest prediction errors. In two out the three properties prediction problems, the learning of the node position through our GI M-4 method of properties yielded the worst result. Figure 4 displays the performance of our model for the GI M-4 method. For none of the properties did the learning of the position with our GI M-4 method of propagation outperform either the fixed or the random unpooling propagation.

Table 2.4 Performance (MAE) comparison over three properties prediction for graph networks with original GAT layer, our AGAT layer, and our four proposed methods of GI propagation.

Properties	GAT	AGAT	GI M-1	GI M-2	GI M-3	GI M-4	Units
Band gap	0.466	0.345	0.329	**0.322**	0.332	0.337	eV
Bulk-moduli	0.081	0.054	**0.047**	0.051	0.048	0.052	log(GPa)
Shear -Moduli	0.121	0.094	**0.085**	0.089	0.086	0.087	log(GPa)

2.4. DISCUSSION

The architecture of our graph neural network offers the inherent benefits of improved accuracy and interepretability. With the use of either one of our proposed global attention layers for properties prediction, we observed a minimum reduction of prediction error of 7% for all property prediction problems evaluated in this study including band gap, bulk modulus and Shear modulus as shown in Table 2.4. Even though our models were trained using a larger number of parameters than CGCNN and a comparable number of parameters to MEGNET, our models have the advantage of a much faster convergence than the latter. Mainly, this improved convergence can be attributed to the implementation of the Batch-normalization within our AGAT layers. For most of the models, about 160 epochs are enough for our GATGNN models to reach comparable results to CGCNN and MEGNET which usually take 500 to 2000 epochs or more with the same batch sizes. Using the bulk-moduli property as an example, Figure 4 displays the resulting training and parity plots from our models. Figure 4 a) provides the details of the predictive performance from our GI M-1 attention model. Our model recorded its lowest validation loss at epoch 252 at which point its parameters were saved. Then, the latter model, with its saved parameters from epoch 252, is later evaluated on the testing set to display the parity plot shown in Figure 4 b) for a MAE of 0.048. While our reported results come from the performance of the model with five AGAT layers for

our CGCNN comparison models and seven for our MEGNET models, using a minimum of three layers with four attention heads would also provide a similar behavior of faster convergence and similar state-of-the-art results.

To obtain additional insight from our GATGNN model, Table 5 provides a detailed analysis of the extracted context vector from the prediction of the material mp-20452 which is displayed in Figure 5. The table lists the interpretable coefficient corresponding to each atom in the unit cell by comparing the results of our proposed model without global attention to the results of the four global attention mechanisms discussed in this work. The last row in the table corresponds to the predicted bulk-moduli property for the material with formula ($Ba(MgPb)_2$). Except for the model trained without the global attention layer, each global attention method provides a meaningful weight to each node with respect to the global information used.

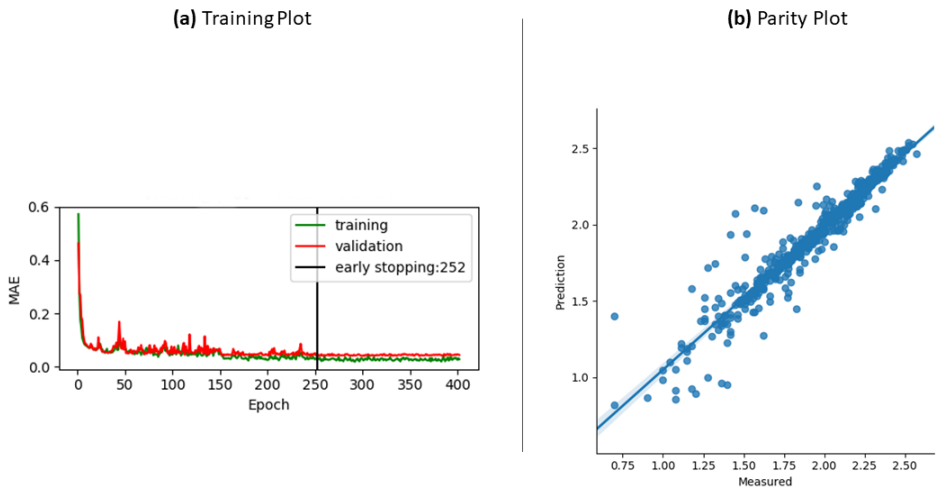

Figure 2.4 Training plot (a) and parity plot (b) for the CGCNN Bulk-Moduli trained model; with property on the log scale.

With all of its weights fixed at 1.0, the AGAT model trained without our global attention had the largest error of 0.09. On the other hand, each one of the other global attention methods returned a mixture of weights per node which offer a different level of information regarding the GI and GI propagation method. The GI M-1 weighted by their types. Barium (Ba) accounted for about 70% of the sum of the context vector while Lead (Pb) and Magnesium (Mg) had a similar weight totaling to 0.146. Particularly, we note that for the four Mg and four Pb atoms, two of each type have half of the weight assigned to the remaining two. The three other global attention methods weight assigned the weights to the nodes based on their location in the graph.

Table 2.5 Context vectors obtained in predicting bulk modulus K_{VRH} of material mp-20452 (with $K_{VRH} = 1.5051$ obtained from each method discussed in our work. Each row lists the rounded weight contribution of an atom in the unit cell for each proposed method. The last row: Predicted K_{VRH} lists the obtained predicted property from each method.

Atom	AGAT	GI M-1	GI M-2	GI M-3	GI M-4
Ba	1.0	0.353	0.05	0.1	0.185
Ba	1.0	0.353	0.185	0.1	0.185
Mg	1.0	0.056	0.05	0.098	0.12
Mg	1.0	0.056	0.185	0.1	0.12
Mg	1.0	0.017	0.037	0.1	0.033
Mg	1.0	0.017	0.037	0.1	0.033
Pb	1.0	0.06	0.05	0.098	0.124
Pb	1.0	0.06	0.037	0.1	0.124
Pb	1.0	0.013	0.185	0.1	0.037
Pb	1.0	0.013	0.185	0.1	0.037
Predicted K_{VRH}	1.5977	1.5024	1.5528	1.5085	1.5204

The GI M-4 had a similar distribution of weights as the GI M-1 method. The global attention layer values the sites of Ba sites as the most critical locations attributing

a total weight of 0.37 to its atoms. The Lead and Magnesium had weights totaling to 0.322 and 0.304. As in GI M-1, half of the Lead and half of the Magnesium had much more significant weights than the remaining. The GI M-2 and GI M-3 methods weighted the nodes based on the node clusters of the graph. For the GI M-2 method, weights are attributed to the nodes based on their corresponding clusters. For the three clusters formed for the mp-20542 material, their corresponding weights correspond to 0.185, 0.05, and 0.037. The cluster that contains the second Ba, the first Mg, and the last two Lead atoms has a cumulative weight of 0.74. The next most important cluster, containing the first Ba, Mg, and Pb, has a cumulative weight of 0.15 while the last cluster totals a weight of 0.11. Lastly, the GI M-3 method provides weights on the basis that the nodes belonged to a different site. With the exception of the first Mg and Pb, all of the are attributed the same weight of 0.1.

With the application of our graph attention layers from which we can extract the nodes' weights, we are able to derive a more effective representation of the crystal feature vector. After that, our global attention layer transforms each node's feature vector to the graph scale by multiplying it with the learned weight to extract the pooled crystal feature vector. In a set of experiments that we conducted with the Random Forest (RF) algorithm, we noted that the extracted pooled crystal feature could allow RF to achieve lower prediction error than the well-known Magpie features [40].

Essentially, our graph network model allows us to extract the pooled crystal feature as a good representative set of features for that crystal, which can be used for property prediction, classification, or other downstream tasks.

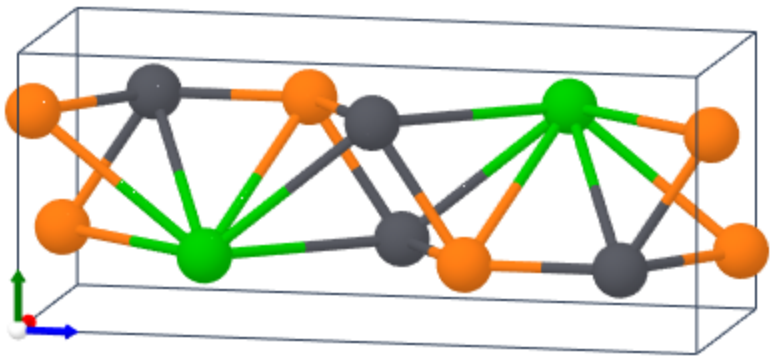

Figure 2.5 $Ba(MgPb)_2$ Crystal structure. (Material-Project id: mp-20452). Orange nodes represent the Mg sites, green the Ba sites, and grey the Pb sites.

Furthermore, like MEGNET [31], our GATGNN model also provides the ability to transfer the learned node relationships and weights from a previously built model trained with large amount of labelled data to training models with limited property data.

Ultimately, our proposed method applies the concept of atoms' importance meanwhile shortening the training time. Our framework first learns the complex relationship amongst the atoms through its AGAT layers. To learn that association, an AGAT layer proceeds to weigh the connection between each atom and its neighboring atoms. However, such weight is only a relative weight; a weight that is assigned to an atom based on its relation to its neighbors. Based on the assumption that the measuring of some properties may also rely on the absolute importance of each atom, our GATGNN applies a global-attention layer to extract this absolute weight. In fact, this global attention layer allows our network to particularly assess the value of each atom with respect to the entire crystal material; hence generating a global weight. We demonstrate in our research that this absolute weight can be obtained from either the atom's type or

the atom's location in the crystal. As seen in the results, our framework reached new SOTA results for properties prediction on nearly all properties, except for formation-energy. Essentially, we associate the success of our framework at better predicting the properties than the baseline models with the three following facts. First, our GATGNN is efficiently implemented. Second, the properties our model predicted well may in fact depend on the absolute ranking of the atoms. Third, the atoms' type and atoms' location may indeed have an association with the intrinsic value of the property. Henceforth, we blame our GATGNN's poorer predictive performance for formation energy on the simple fact that atoms' global weight is irrelevant to a material's property. In summary, we suggest to apply our proposed method in cases where atoms' additional information are available and that there also exists an association between those atomic information and the predictable property.

2.5. CONCLUSION

We proposed GATGNN, a novel graph convolutional neural network model with global attention mechanisms for accurate materials property prediction. Evaluations on standard materials projects dataset over multiple materials properties such as band gap, bulk-moduli, shear moduli, etc show that our proposed graph convolutional network models achieve better predictive performance than renowned CGCNN and MEGNET. The success of our GATGNN can be attributed to its local attention mechanism and global attention layer. Particularly, our global attention technique implements the assumption that some atoms may be more influential to the measured value of the crystal property. Despite the important predicting improvement of our GATGNN over the referred baselines, conducting additional investigations with other local or global atomic

context information would be very important in validating the functioning and importance of our global attention layer and further improve the performance in materials property prediction.

CHAPTER 3

NODE-SELECT: A GRAPH NEURAL NETWORK BASED ON A SELECTIVE PROPAGATION TECHNIQUE

3.1 INTRODUCTION

The use of deep learning techniques for graph analysis has become a very popular research topic in recent years [2]. Commonly referred to as graph neural networks (GNN), these deep learning techniques now figure amongst the most used methods for learning from relational data [1, 2]. Just as in the functioning of convolutional neural networks (CNN), multiple convolution operations can also be applied to learn from non-Euclidean data [1, 2, 17]. Various adaptations of GNNs have been proposed over the years for the purpose of node classification [1, 2]. These GNN adaptations mainly differ in regards to their node embedding techniques, their algorithms' propagation or aggregation methods, and their model's scalability [1, 2]. Examples of important GNN variants include GCN [41], GAT [14], GraphSAGE [42], DeepGCN [43], and Pairnorm as one of the most recent works [44].

Prior GNNs have made use of regularization, Chebyshev polynomials, feature normalization, attention, residual blocks, random sampling, and many other techniques which have further pushed state-of-the-art for GNN [1, 11, 14, 41-43, 45, 46]. Nevertheless, there still remain many factors that still present challenges to GNN [1, 11, 41]. Two important factors that mostly get referenced in the literature are over-smoothing and overfitting. Over-smoothing, defined as excessive similarity of node representation, is a direct consequence of deeply stacking graph convolutional layers [11, 47, 48]. Particularly, the over-smoothing issue has been justified to result from the fact that more noise gets shared than useful information during the convolution operations [47]. On the other hand, overfitting is another issue which happens with adding more parameters and increasing the complexity of the model [46]. Besides the over-smoothing and over-fitting

issues, GNN also suffer from a noticeable conceptual limitation. Few GNN variants provide an implementation that fully mimic the relational rules observed in the networks of real world [14, 44, 46, 49-51]. Nonetheless, the number of GNN variants with mechanisms that easily translate to real-world networks is minimal and remains to be further exploited; particularly, scenarios in which there are consequences to letting any nodes propagate information [52-55].

With the motivation to mainly tackle this existing conceptual limitation, we propose a new kind of graph neural network named: NODE-SELECT. For this conceptual limitation, we introduce an efficient selection mechanism that prevents nodes with representation of poorer quality from propagating information. Beyond the selection process, we also learn a global weight coefficient for these propagating nodes and also combine our memory-efficient layers in parallel as in the ensemble concept. To demonstrate the effectiveness and importance of our proposed method, we evaluate it on standard benchmark datasets with and without noise data [55-57]. Overall, our proposed NODE-SELECT considerably outperforms popular GNN frameworks on graphs augmented with noise vertices and marginally surpasses them on graph without noise vertices.

We summarize our contribution as four main points. **(1)** We implement a very important concept of node selection to graph neural networks. (**2)** We adapt the ensemble concept by stacking our graph convolutional layers in parallel and demonstrate how it benefits our framework. **(3)** We demonstrate through extensive experiments how current GNN can be considerably affected by the presence of harmful vertices representations during the message-passing, but our NODE-SELECT is not affected by them. **(4)** We

demonstrate that our proposed method is extremely scalable with the increase of graph sizes.

3.2 RELATED-WORK

Researchers have used various techniques to do their convolution operation on the graph vertices. Examples of such techniques include the adaptation of gated recurrent units (GRU), Chebyshev polynomials, attention mechanisms, etc... GGNN was the first framework to apply gated recurrent units to sequentially update the feature vectors of the graph nodes [58, 59]. While ChebConv first used Chebyshev polynomials to do the node convolutions, the adaptation by GCN proved to be more effective thanks to its feature aggregation restriction and normalization trick [41, 60]. The technical concept of sampling was introduced by the works of GraphSAGE and FastGCN [42, 45]. Another important technical concept: attention, for graph neural networks, was first adapted in the framework of GAT [14].

In addition to the aforementioned methods, there exist many GNN variants that have further incrementally introduced other important techniques into the field of graph neural networks. Such architectures include DropEdge and DNA-Conv [45, 46]. DropEdge proposes a regularization mechanism to address over-fitting and over-smoothing by randomly removing connecting edges [46]. On the other hand, inspired by the concepts of Jumping Knowledge [61], Fey proposed a dynamic neighborhood aggregation mechanism to offer to their learning model a bigger range of feature information [45]. Nevertheless, there still remains conceptual limitations that still need to be addressed to further advance the field.

The main limitation we aim to address is the need for *a straight-forward adaptation of the natural message-passing mechanism* often found in real-world graphs. In real graphs such as social networks, computer networks, brains, or molecules; we frequently observe an orderly communication between the units. Our goal is to implement this communication of real-world graphs in which only a subset of the vertices actively exchange information simultaneously. Works in the fields of Sociology and Neuroscience have studied examples of this orderly communication. In Sociology, numerous publications have studied the relatable topic that only the best suitable people should lead tasks within a social network [53, 54, 62]. This restriction of propagating vertices in a social setting is often paraphrased as “Too many cooks spoil the broth”.

In Neuroscience there has also been works that studied the topic that only a subset of neurons fire simultaneously in brain networks [63, 64]. Just as there is a clear limitation in propagating vertices in some real-world graphs, we also implement in our NODE-SELECT a similar mechanism that grants our model the flexibility to adapt this restriction.

Besides the need of adapting this conceptual selection mechanism, we also assumed that the implementation of such mechanism would also technically benefit the network. LP-GNN is an example of how adapting a propagation-restricting technique improves a GNN model performance [65]. By constraining the propagation, authors Tiezzi et. al. demonstrate that LP-GNN offered benefits of simplicity and efficiency [65]. Likewise, our adapted selection mechanism could act as a regularization in the network while preventing the least sharing-fit nodes from propagating their embeddings [66, 67]. This restriction would mainly reduce the amount of noise coming from particular nodes.

With the cancellation of *inappropriate nodes'* propagation, the network would also benefit in efficiency having to arguably do less node convolutions [11, 60]. With the selection implemented within each layer, we also presumed that ensembling our layers could be more beneficial than sequentially them. The layers' sequential stacking could lead to poorer performance if prior selections were sub-optimal. Also, ensembling the layers could result in a diverse generation of embeddings which would likely increase accuracy performance of the model [68].

3.3 PROPOSED METHOD

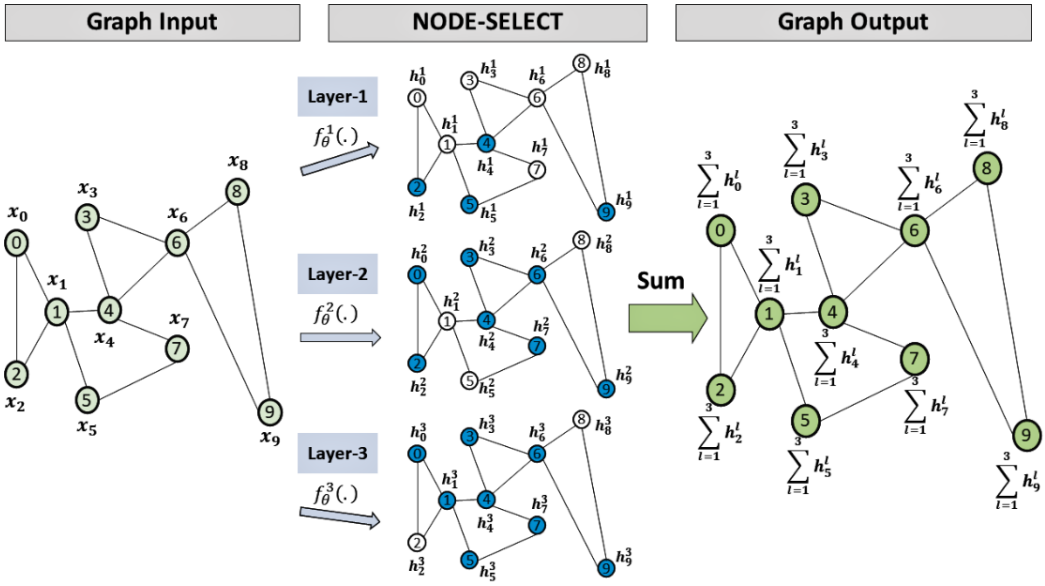

Figure 3. 1 Architecture of the NODE-SELECT graph neural network. Provided a graph, a number of independent layers (3 in this figure) are applied in parallel, where each provides a different state embedding based on their selection of propagating nodes (in blue). Finally, the output of all layers are summed to create the nodes' final hidden state.

We begin by formulating an input graph G as the set $G = (V, E)$ where V and E respectively define a set of nodes v_i and a set of edges e_{ij} connecting 2 nodes: v_i and v_j. Also, we denote as $X = \{x_1, \dots, x_n\}$ or $\{h_1^0, \dots, h_n^0\} \in \mathbb{R}^{N \times F}$ the node features. For the

task of node classification, a graph neural network needs to learn an embedding $h_i^{(l)}$ based on a prior embedding or feature vector $h_i^{(l-1)}$. The operation done by each layer can be expressed by a function $f_\theta^{(l)}$:

$$h_1^{(l)} = f_\theta^{(l)}\left(h_1^{(l-1)}, \Gamma\left\{h_j^{(l-1)}\right\}_{j \in N(i)}\right) \tag{3.1}$$

where θ denotes the trainable parameters of layer l, Γ any function aggregating localized neighborhood information, and $N(i)$ the neighborhood of node v_i .

To update node v_i's embedding, these layers generally utilize information from all neighboring nodes and need to be arranged in sequence within the network. However, such layout favors the potential issue of oversmoothing and lacks adequate techniques to prevent the propagation of noise from specific nodes. In contrast, we propose to use a selection mechanism $S(\cdot)$ to limit noise sharing between nodes and the combination of embeddings from independent layers to reduce noise propagation between layers.

Algorithm 3.1 summarizes the NODE-SELECT framework. At first, the resulting embedding is initialized as a zero vector. Afterwards, a series of 4 ensuing operations take place per each layer. First, the input feature vectors are transformed. These transformed features are used to compute a selectivity score which is then compared with a threshold to determine the nodes selection. Then, the transformed features of only the selected neighbors are aggregated. To get the layer's output node embeddings, the nodes' original transformed feature vector and the selected neighbors' aggregated feature vectors are combined. Lastly, each layer's output embedding is added to the resulting embedding.

3.3.1 NODE-SELECTION

The first step in our method consists of linearly transforming the initial feature vector x_i using a matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$. The transformed feature vector $\mathbf{W}x_i$, with reduced dimensionality, has the same cardinality as the embedding output. Afterwards, we estimate a random selectivity value $\hat{p}_i$ as a measure to classify nodes with potentially harmful or useless embedding information. The goal is to use this $\hat{p}_i$ value so the network can detect nodes whose omission of information improves the training. In this research, we only utilize the sum of the neighbors embedding $N(i)$ to learn this value, but other feature information could also be used. We define our selection technique as:

$$S(v_i) = \begin{cases} 1, & \hat{p}_i \geq T \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Where $\hat{p}_i = \sigma\left(\mathbf{W}_0 \left(\sum_{j \in N_{(i)}} \mathbf{W}x_j\right)\right)$, $\mathbf{W}_0 \in \mathbb{R}^{1 \times F'}$ is a weight matrix, T a determined threshold, and σ a non-linear transformation. Thus, $\hat{p}_i$ can be interpreted as a normalized signal, ranging between 0 and 1, allowing the model to make its selection. This learnable selection of a node will impact how the messages propagate throughout the entire graph. Namely, the embedding information of a non-selected node will have reduced impact on the learning of other nodes' embeddings.

It is important to note that the selection $S(v_i)$ is done irrespective of the size of $N_{(i)}$. In this research, we found that there is a very little - to no correlation between the Boolean selection and a node's neighborhood size (Pearson correlation: 0.03 - 0.3). Figures 13 and 14 in the Appendix provide correlation illustrations for the Cora and Pubmed datasets for the two variables. Instead, what really happens is that the model favors more expressive or more informative neighborhoods [69]. Notably, a

neighborhood with more important nodes will be scaled much higher than a neighborhood with less important nodes, regardless of the size [69].

Algorithm 3.1: Framework for NODE-SELECT

Input: graph $G = (V, E)$; node's neighborhood $N(i)$; node v_i's input features x_i; Layer $l \in [1, L]$; threshold value T

Output: NODE-SELECT embedding $\boldsymbol{h_i'}$ for a node i

$\boldsymbol{h_i'} \leftarrow \vec{0}$

for $l \leftarrow 1$ **to** L **do**

  $S(v_i) \leftarrow 0$; // v_i is not selected

  $\boldsymbol{h_i^l} \leftarrow x_i$; // compute embedding for node i

  $\hat{p}_i \leftarrow$ Compute v_i's selectivity score; // using N_i

  if $\hat{p}_i \geq T$ **then**

    $S(v_i) \leftarrow 1$; // v_i is selected

  end

  $\boldsymbol{A_i} \leftarrow$ Aggregate message from selected neighboring nodes;

  $\boldsymbol{h_i^l} \leftarrow \boldsymbol{h_i'} + \boldsymbol{A_i}$;

  $\boldsymbol{h_i'} \leftarrow \boldsymbol{h_i'} + \boldsymbol{h_i^l}$;

end

$h_i' \in \mathbb{R}^F$

3.3.2 SELECTIVE AGGREGATION

Once a node's selection v_i is computed, we allow the layer to aggregate the embedding information only from its selected neighbors. We propose two types of aggregation: a) a simple message-aggregation from only the nearest 1-hop neighbors and b) a sequential message-aggregation from any q-hop neighbors. In cases where one can assume that learning from farther neighbors provides very little additional information, the simple message-aggregation should be applied. Otherwise, the sequential message-aggregation from any q-hop neighbors is preferable. In fact, SSE and LP-GNN are examples of GNN techniques that restrict the GNN learning to 1-hop neighborhood [65, 70]. Nonetheless, in this research, the difference in terms of performance between the two was very minor ($\leq 0.5\%$).

3.3.2.1 1-HOP NEIGHBORHOOD AGGREGATION

The simplified form of this selectively aggregated information is expressed below:

$$A_i = \sum_{j \in N_{(i)}} \alpha_j \cdot S(v_j) \cdot \mathbf{W} x_j \;\; h_1^{(l)} \tag{3.3}$$

where α_i defines the propagation weight for each selected node. This propagation weight is calculated by linearly transforming the concatenated summed and selectively aggregated neighborhood embeddings; such that $\alpha_i = \sigma\left(\mathbf{W_1}\left(\sum_{j \in N_{(i)}} S(v_j) \cdot \mathbf{W} x_j || \;\; \sum_{j \in N_{(i)}} \mathbf{W} x_j\right)\right)$ and $W_1 \in \; \mathbb{R}^{1 \times 2 \cdot F'}$ is a weight matrix. The concatenation of $\left(\sum_{j \in N_{(i)}} S(v_j) \cdot \mathbf{W} x_j\right)$ and $\left(\sum_{j \in N_{(i)}} \mathbf{W} x_j\right)$ is motivated by the fact that we want the NODE-SELECT layer to learn α_i as a conditional weight. Particularly, this weight must be conditioned on a) whether the source node was originally selected $\left(\sum_{j \in N_{(i)}} \mathbf{W} x_j\right)$ and b) on the fact that messages will be aggregated selectively $\left(\sum_{j \in N_{(i)}} S(v_j) \cdot \mathbf{W} x_j\right)$.

Considering that the learned selection $S(v_j)$ uniformly impacts all of its neighbors' message aggregation, α_i thereby corresponds to the hard attention for a selected node v_i. Our attention approach is rather different than the attention used in other works, i.e. the soft-attention α_{ij} in GAT. Our attention implies a fixed α_{ij} weight $\forall j \; \in N_{(i)} \; (\alpha_{i1} = \alpha_{i2} = \cdots = \alpha_{in})$ whereas the attention imply the contrary. In our experiments, applying this global weight α_i resulted in higher ($\geq 3\%$) accuracy performance than adapting a weight relative to each node's neighbor (i.e. α_{ij} as in GAT).

3.3.2.2 ANY Q-HOP NEIGHBORHOOD AGGREGATION

NODE-SELECT is based on the assumption that lower q-hop neighbors are the most important to the embedding learning. While there are benefits to restricting the aggregation to closest 1-hop neighbors [70] (simpler and quicker computations; reduced complexity; and avoiding noise message passing), they come at the possible cost of reduced expressiveness of the embedding (missing additional important neighbors' information) [71, 72]. To alleviate this potential issue, our NODE-SELECT adapts a method similar to depth-first search approach for capturing information from longer range neighbors. Our approach to sequentially aggregate neighbors' embedding information is described in Algorithm 3.2 and illustrated in Figure 3.12. Essentially, this sequential message-aggregation consists of three main operations that take place at each q-hop neighborhood (at each depth). First, an attention score is computed for each node at that depth. Then, both this weight and the selection are used to propagate embedding information throughout the graph. Last, the selection is itself passed on to all neighbors.

3.3.2.3 FEATURE UPDATE

Following the computation of the selective embedding aggregation, we then use it to compute layer l embedding. This final layer takes the form a function $f_\theta^{(l)}$:

$$f_\theta^{(l)}(x_i) = \sigma(\mathbf{W}x_i) + \sigma(A_i) \tag{3.4}$$

where $\mathbf{W}x_i$ is the originally transformed feature vector and A_i the pooled embedding information from just the selected nodes.

Algorithm 3.2: Sequential message-propagation of NODE-SELECT

Input: graph $G = (V, E)$; node's neighborhood $N(i)$; node v_i's input features h_i^l; depth $D \in \mathbb{Z}^+$; a depth feature Matrix $D \in \mathbb{R}^{D \times D}$; a learnable matrix $\boldsymbol{W_2} \in \mathbb{R}^{1 \times (F' + D)}$; non-linearity σ ; initial selection $S(v_i)$

Output: Sequential aggregated embedding A_i for a node i

$\boldsymbol{A_i} \leftarrow \boldsymbol{h_i'}$

for $q \leftarrow 0$ **to** $D - 1$ **do**

  $\mathbf{d} \leftarrow D[q]$; // get a feature representation of the depth

  $\alpha_i \leftarrow \sigma\big(W_2(A_i || \mathbf{d})\big)$; // compute a depth-selection weight coefficient

  $A_i \leftarrow A_i + \sum_{j \in N_{(i)}} \big(\alpha_j \cdot A_j \cdot S(v_j)\big)$; // selective message aggregation

  if $S(v_i) = 1$ **then**

    $S(v_i) \leftarrow 1; \forall j \in N_{(i)}$; // propagate selection

  end

  $S(v_i) \leftarrow 0$;

end

$A_i \in \mathbb{R}^{F'}$

3.3.2.4 PARALLEL STACKING

The last step consists of simply summing the embedding information from all layers. Given *L* layers, the final embedding output for a given is the summation of all *L* independently learned embeddings $f_\theta^{(l)}(x_i)$. An example of this parallel arrangement with 3 NS layers is depicted in Figure 3.1, within the middle column. This summation is described below.

$$\text{output}_i = \sum_{l=1}^{L} f_\theta^{(l)}(x_i) \tag{3.5}$$

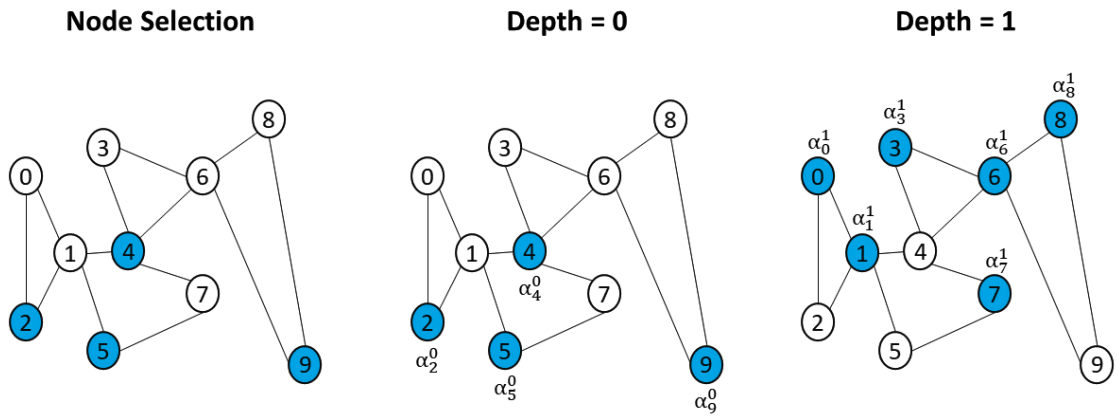

Figure 3.2 Sequential message propagation. First, the layer selects a subset of propagating nodes represented in blue: $V_S^l = \{2,4,5,9\}$. At depth q=0, only the selected nodes $\{2, 4, 5, 9\}$ are allowed to share a proportion $\alpha_{(\cdot)}^{(0)}$ of their embedding (message-passing in 1-hop neighborhood). At depth $q = 1$, only the 1-hop neighbors of the initially selected nodes are allowed to share another proportion $\alpha_{(\cdot)}^{(1)}$ of their updated contribute to the message-passing (message-passing in 2-hop neighborhood).

3.4 EXPERIMENTS

3.4.1 DATASETS

To assess the performance of our proposed model, we conduct two sets of experiments using a total of 8 benchmark datasets. In the first experiment, we utilize all 8 datasets: Cora, CiteSeer, PubMed, Cora Full, Coauthor CS, Coauthor Physics, Amazon Computers, and Amazon Photo. Cora, CiteSeer, and PubMed contain relational data on academic papers [56, 57]. Datasets Coauthor CS (Co-CS) and Coauthor Physics (Co-P) are co-authorship datasets from the Microsoft Academic Graph [55]. Lastly, Amazon Computers (Amz-C) and Amazon Photo (Amz-P) are graph datasets defining segments of the Amazon product categories graphs. For each dataset, we randomly split the nodes so that the training, validation, and testing sets follow a ratio of 20-20-60 percent. This split is repeated for 10 randomly chosen seeds which are used in each model experiment.

In the second experiment, we only use Cora, CiteSeer, and the PubMed datasets which we modify by adding noise data into their graphs. We increase the size of each graph by 10 and 25% using pseudo vertices. We attribute each pseudo vertex a feature vector from a standard normal distribution, a random label, and random neighbors from the original graph. We follow the same 20-20-60 splitting ratio as in the first experiment but afterwards remove the pseudo vertices from the testing set. This split is repeated for 5 randomly chosen seeds. Details of the Datasets are provided in Table 3.1.

Table 3.1 Statistics of transductive Datasets used in this paper.

Dataset	Nodes	Edges	Classes	Features
CiteSeer	3,327	4,552	6	3,703
Cora	2,708	5,278	7	1,433
PubMed	19,117	44,324	3	500
Co-P	34,493	247,962	5	8,415
Co-CS	18,833	81,894	15	6,805
Cora Full	19,793	63,421	70	8,710
Amz-P	7,650	245,861	8	8,415
Amz-C	13,752	119,081	10	767

3.4.2 EXPERIMENTAL SETUP

We compare our proposed method to 6 GNN variants selected for either their robust performance, contrasting node sampling method, or both. These baselines include DropEdge, FastGCN, GAT, GCN, GraphSAGE, and Node2vec [14, 41, 42, 46, 73, 74]. Given that each framework performs differently under various training dynamics, we perform random hyper-parameter and only report results from the best performing

models with respect to the validation set. We apply a fixed dropout rate of 0.5 after the GNN layers and use Adam as optimizer [75, 76]. We implement all the models using Pytorch and the library of Pytorch-Geometric [38, 77]. The hyper-parameters from the best performing models are provided in Table 5 and 6 of the Appendix B.

3.4.3 RESULTS

Table 3.2 displays the average accuracies over the 10 random splits from the first experiment. As seen, NODE-SELECT consistently matches or outperforms the performance by the baselines by up to 1.4 percentage points. Table 3.3 lists the average classification accuracy of 5 random splits for the second experiment with noise introduced to the training. Once a GNN model is introduced to a considerably amount of noise information, the results demonstrate that the accuracy significantly drops [66]. Nevertheless, our NODE-SELECT is only marginally affected by the presence of noise information whereas the baselines considerably are. As shown in the results, our proposed method particularly stands out in these noise experiments by outperforming the other baselines by up to 20 percentage points. Simply put, the resilient ability of our network to be affected by noise information is due to the used selection mechanism which allows the network a direct control of blocking nodes propagating them.

Table 3.2 Results from the first experiment with the 8 standard benchmarks. Average testing accuracy (%) and standard deviation from 10 random splits are listed.

Dataset	DropEdge-GCN	FastGCN	GAT	GCN	GraphSAGE	Node2vec	NODE-SELECT
CiteSeer	56.2 ± 1.6	74.0 ± 1.0	$\mathbf{74.2 \pm 0.8}$	74.0 ± 0.7	73.7 ± 0.7	55.3 ± 0.7	74.1 ± 1.1
Cora	83.5 ± 2.3	82.1 ± 2.6	$\mathbf{86.0 \pm 0.7}$	85.0 ± 0.7	$\mathbf{86.0 \pm 0.7}$	78.1 ± 0.8	$\mathbf{86.0 \pm 0.7}$
PubMed	87.1 ± 0.5	87.6 ± 0.5	86.4 ± 0.3	87.2 ± 0.3	86.2 ± 0.3	80.2 ± 0.4	$\mathbf{88.1 \pm 0.3}$
Co-P	95.9 ± 0.1	95.5 ± 0.3	95.7 ± 0.1	95.9 ± 0.1	95.4 ± 0.2	93.0 ± 0.1	$\mathbf{96.5 \pm 0.1}$
Co-CS	92.8 ± 0.6	92.2 ± 0.4	92.2 ± 0.2	93.1 ± 0.2	93.4 ± 0.2	87.7 ± 0.3	$\mathbf{94.8 \pm 0.1}$
Cora Full	57.4 ± 1.8	60.8 ± 1.0	64.8 ± 0.5	$\mathbf{67.3 \pm 0.5}$	64.9 ± 0.3	58.8 ± 0.3	$\mathbf{67.3 \pm 0.6}$
Amz-C	84.9 ± 3.4	83.5 ± 2.2	90.0 ± 0.7	89.4 ± 0.5	$\mathbf{90.2 \pm 0.5}$	87.2 ± 0.4	89.6 ± 0.4
Amz-P	89.2 ± 4.1	91.0 ± 0.9	93.7 ± 0.6	935 ± 0.2	$\mathbf{94.4 \pm 0.5}$	91.0 ± 0.3	$\mathbf{94.4 \pm 0.4}$

Table 3.3 Results from the second experiment with the Cora, Citeseer, and Pubmed benchmarks. Graphs sizes from the datasets are increased by 10 and 25% from the addition of noise data. Average testing accuracy (%) and standard deviation from 5 random splits are listed.

Variant	CiteSeer		Cora		Pubmed	
	+10%	+25%	+10%	+25%	+10%	+25%
DropEdge-GCN	42.0 ± 2.3	35.6 ± 1.4	38.1 ± 3.4	34.4 ± 2.9	74.4 ± 12.0	46.2 ± 12.3
FastGCN	33.7 ± 0.9	29.9 ± 1.5	40.8 ± 2.1	30.3 ± 1.2	57.3 ± 0.9	47.9 ± 1.3
GAT	34.1 ± 1.2	34.1 ± 1.8	61.7 ± 2.0	58.0 ± 1.4	55.0 ± 4.8	52.5 ± 7.5
GCN	56.0 ± 0.8	49.3 ± 1.1	74.3 ± 1.7	65.2 ± 1.3	58.4 ± 0.6	54.8 ± 0.6
GraphSAGE	35.4 ± 1.7	33.9 ± 0.7	52.8 ± 1.3	51.9 ± 1.7	45.4 ± 0.5	42.7 ± 1.1
Node2vec	38.7 ± 0.8	35.4 ± 0.8	58.4 ± 1.4	51.1 ± 1.3	62.9 ± 0.4	58.5 ± 0.3
NODE-SELECT	$\mathbf{68.6 \pm 0.4}$	$\mathbf{64.8 \pm 2.0}$	$\mathbf{80.9 \pm 3.8}$	$\mathbf{78.4 \pm 1.8}$	$\mathbf{83.6 \pm 0.9}$	$\mathbf{79.7 \pm 0.8}$

3.5 DISCUSSION

3.5.1 PARALLEL VS SEQUENTIAL STACKING

Compared to the traditional sequential stacking of GNN layers, NODE-SELECT adopts the approach of stacking its layers in parallel. Based on our experiments, we have found that the parallel stacking of these selective layers yielded better and more stable results. Figure 3.3 illustrates a comparative study of these two stacking options. As demonstrated, the parallel setting proves to be more beneficial with its results reaching higher accuracy and lower variance. The sequential layout forces a layer l to depend on the set of selected nodes $V_s^{(l-1)}$ of a previous layer $l-1$. In the rare cases that the composition of $V_s^{(l-1)}$ is not completely suitable to the weights of layer l, the model may result in a much lower performance; thus observing a higher variance and reduced accuracy. The parallel layout removes this dependence issue by stacking its layers side-by-side and having them learn independently as in the ensemble method [78].

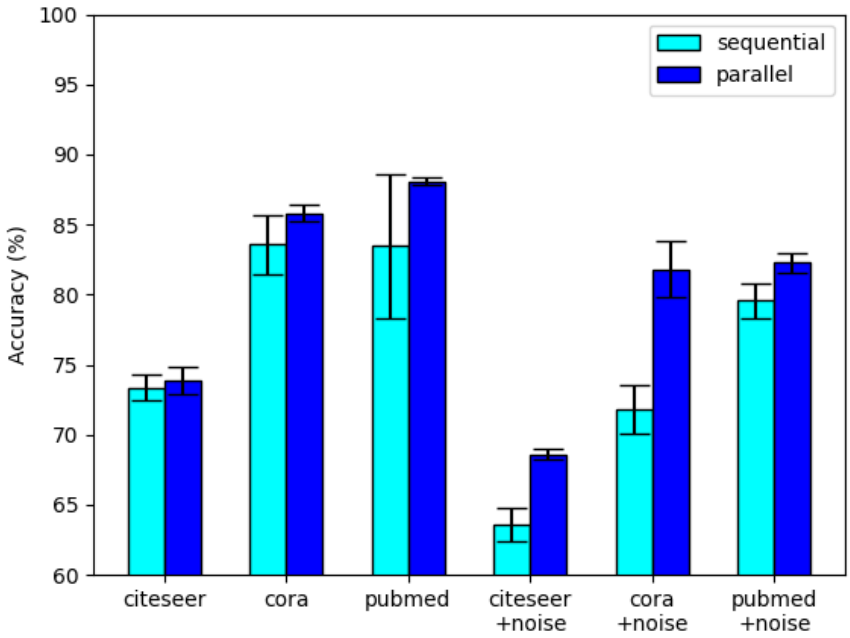

Figure 3. 3 Comparison of sequential vs. parallel layers stacking

3.5.2 LAYERS OPERATION

Because of the parallel configuration, any given NODE-SELECT layer operates independently. Each layer separately makes a node selection that yields to a node embedding. Figures 3.4 and 3.5 contrast these layers' differences in terms of their predicted sensitivity value, accuracy score, and proportion of selected nodes for experiments on the Cora and Pubmed Dataset. In Figure 3.4, the $\hat{p}_i$ values of 3 nodes from the training set are predicted by 3 parallel layers on a Cora experiment. Because of the layers' independence, a node's $\hat{p}_i$ values from distinct layers are also independent. The independence is illustrated with Layer-1 learning to output high values above the threshold for node 1728 but conservative values node 1601, yet Layer-3 does the opposite. In Figure 3.5, the accuracy results ranging from 75 to 85% for the 10 layers

used in a trained model are displayed with the selection percentage ranging between 22 and 100 % (in blue). For instance, layer 6 reached the highest accuracy of 85% with a selection percentage of 84% while layer 3 had the third lowest accuracy score of 80% with 100\% selection proportion. As demonstrated in the figure, a layer's accuracy performance is not correlated to its proportion in size of selected nodes. However, based on our experiments, we found that a layer that more effectively *filters* the most noise-propagating nodes is more likely to reach a higher accuracy. We also see the reported accuracy (in green) of the final output, obtained by summing the embeddings of the 10 layers, being much higher than any layer's individual accuracy score.

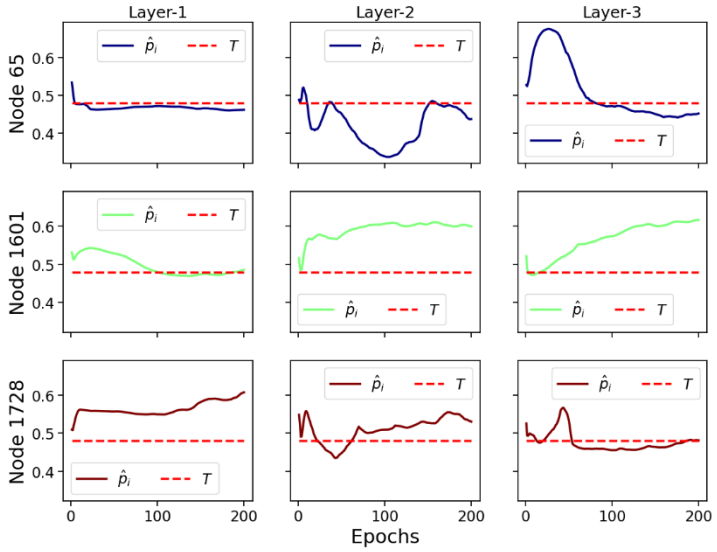

Figure 3.4 Learning plot of $\hat{p}_i$ during a NODE-SELECT model training. Red dashed line --- represents the threshold parameter T. For each node (65, 1601, 1728), an independent layer learns its selection (allowing it to share its embedding information). A given node v_i is selected when its $\hat{p}_i$ signal is at least as high as the threshold T.

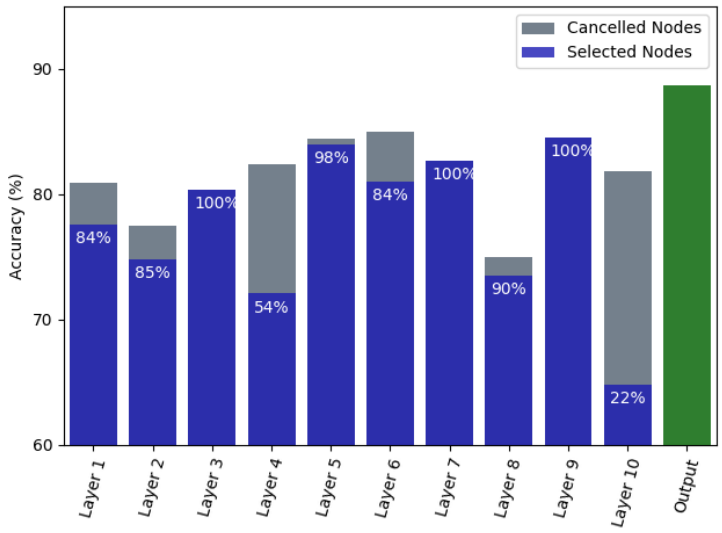

Figure 3.5 Accuracy score by layer and output for model trained on Pubmed. Blue color displays the percentage of selected nodes $V_s^{(l)}$. The green color displays the reported accuracy of the final output, obtained by summing the embeddings from the 10 layers.

3.5.3 SCALABILITY

A direct benefit of stacking our layers in parallel is that our method is very scalable. Because our method is composed of ensembled 1-layer GNN, its memory usage is very effective. For example, the number of trainable parameters can be estimated with $L\big(F'(3+F)\big)$; with L describing the number of layers, F' the output dimension, and F the input dimension. Figure 3.6 contrasts how NODE-SELECT scales with larger graphs when compared to some baseline frameworks. As demonstrated, NODE-SELECT scales to larger graphs comparably to GCN and much better than GAT, GraphSAGE, and DropEdge. As the graph gets larger, the amount of memory required for the learning to take place also increases and the challenge of being scalable affects many current GNN [1, 11]. Nonetheless, NODE-SELECT adapts very well to larger graphs.

Also, our proposed framework demonstrates very good efficiency. The time complexity of our variant, which depends on both hyperparameters of depth Q and layers L, can be expressed as $O(QL|\mathcal{E}|)$ for the any q-hop neighborhood aggregation technique and $O(L|\mathcal{E}|)$ for the 1-hop neighborhood aggregation technique. In our experiments, we found that Q and K are never simultaneously large. For smaller graphs, the maximum effective size for L was 3 with Q set at 2. On the other hand, larger graphs needed at most a L of 25 with a smaller Q fixed at 1.

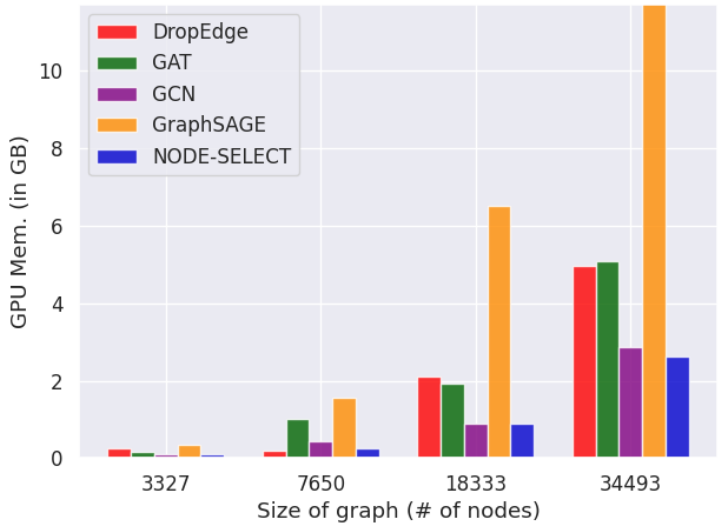

Figure 3.6 Scalability of NODE-SELECT compared to other methods when graph size increases.

3.5.4 EFFECT OF PARAMETERS L AND T

In contrast to other GNNs, NODE-SELECT only has two configurable parameters that affect the model's performance. The parameter L guides the model's fitting behavior while L guides its selection mechanism. In our study, we found that any arbitrary number of layer leads to a good performance. However, depending on the size and properties of the graph, too few layers may cause the model to under-fit while too many to over-fit.

Table 3.4 provides a simple illustration of the effect of increasing the parameter L. Using only 1 layer causes the model to under-fit with an accuracy of 94%. Using 20 or more (100) layers causes the model to over-fit with accuracy scores that are below the well-fit models trained with 5 or 10 layers. Particularly, as a NODE-SELECT model uses more layers (past the optimal number), each individual layer becomes weaker. This decrease in performance results from the fact that each layer learns the patterns pertaining to a specific region in a graph and thus generalizes poorly.

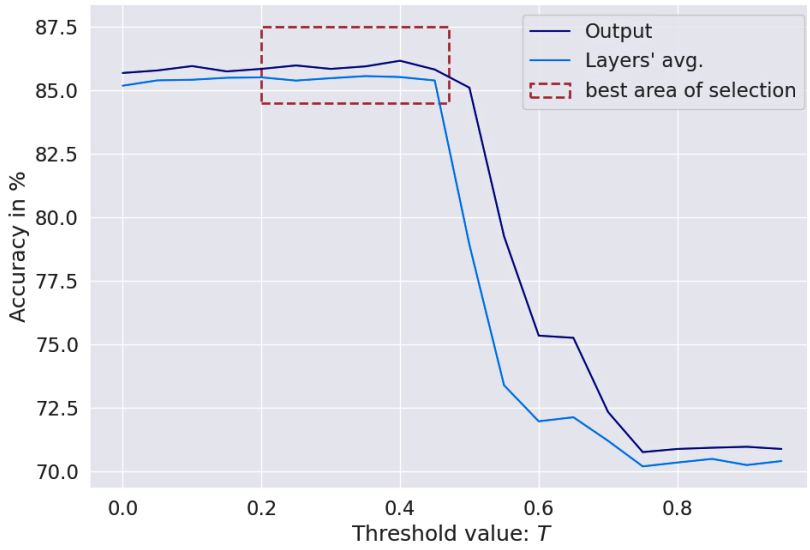

Figure 3.7 Results of using variable value of the threshold parameter T from Cora experiments. Red rectangle contours area with high accuracy where selection is most *diverse* (great variance of selection).

A NODE-SELECT model will adjust its weights to retain as much embedding information as possible during training. Therefore, a NODE-SELECT only begins to cancel nodes when the threshold parameter is not too small; for instance, above T value of 0.3 However, using low values for the threshold results in performance that is only marginally lower than with moderate threshold. Also, the selection mechanism is a lot more effective when the graph is not small. In our experiments with smaller graphs (Cora and Citeseer), an effective layer only cancelled a minimal number of nodes (i.e. between

0 and 10%). Figure 3.7 displays the effect of changing T on the model accuracy. Using the parameter T allows the model to conservatively remove subset of nodes that are not needed to lower its loss. Using a T value in a range of $0.3 \leq T \leq 0.49$ gives the best results in terms of both accuracies and node cancellations. However, the application of a large T value leads the model to cancel too many nodes, thereby losing crucial information from potentially important nodes.

Table 3.4 Results of using variable number of layers. Rows in blue are for under-fit models and rows in red for over-fit models.

# of Layers	Co-CS		
	Layers Mean accuracy	Model Accuracy	Layers avg. size of selection
1	94.0	94.0	78%
5	93.8	95.0	51%
10	93.0	94.8	52%
20	85.5	94.8	64%
100	62.3	93.5	66%

3.5 CONCLUSION

We introduced NSGNN, a novel graph neural network for node-classification, which learns node embeddings by summing correlated embeddings learned by its layers. Inspired by the functioning of real-world graphs, our NODE-SELECT addresses the conceptual limitation of selective propagation based on the nodes' global importance. As opposed to other frameworks which sequentially convolve the embeddings, thus removing key information in the embeddings, our NODE-SELECT relies on various complementary convolutions to enhance those key information. Besides reaching state-of-the-art performance in experiments which introduced noise propagation, its scalability

to larger graphs is much more effective than other baselines. With a simple selection mechanism that allows our model to effectively adapt to the problem of noise-propagating nodes, we expect that our proposed method can adapt to real world problems where such mechanism can be very important such as in Botnet detection or cancellation of particular instances within a graph. In this research, we combined all layers' embedding only with summation operator. As demonstrated in the literature, the sum readout/operator is one of the most frequently used and effective pooling method used in GNN [1, 11]. Nonetheless, further research may also be done to additionally improve our method's performance by testing other ways to combine the independent layers' embedding or how other ways to ensemble the separate layers (i.e. boosting).

CHAPTER 4

ACCURATE PREDICTION OF VOLTAGE OF BATTERY ELECTRODE MATERIALS USING ATTENTION BASED GRAPH NEURAL NETWORKS

4.1 INTRODUCTION

Batteries are the dominant source of energy for diverse applications and main work-horse for portable electronics [79, 80]. Common examples where batteries are increasingly adopted are electric vehicles and grid energy storage [81, 82]. Besides their wide use, there are still big interests in improving these batteries' performance for more improved reliability in devices demanding large energy density. But to develop next-generation batteries, accurate and efficient exploration of large chemical space is necessary and predicting their performance represents the first step towards this goal. Traditionally, the exploration of these batteries or electrodes' properties was done using time-consuming physics based simulations [83] and/or by using resource-intensive experiments [84]. Mainly, examining each material in the large chemical space while searching for robust electrodes imposes great difficulties with these traditional methods. Hence, machine learning has been used as an alternative for their impressive performance and are increasingly adopted in the battery community for predicting the performance metrics of battery components including intercalation potentials or voltage [79, 85-94].

Recently multiple ML-based approaches to predict the voltage of electrodes materials have been used [90-92]. Such approaches include models based on L-potentials and other simple deep neural networks, which are the most accurate. ML-potentials based models are trained to learn the potential energy surface of solids using data from physics based simulations [90]. Though promising, the accuracy of these ML-potentials is limited to particular types of materials with specific atomic compositions and have weak transferability. Another important limitation to these ML-potentials is that generating the dataset for their model from density functional theory (DFT), for each composition space,

is extremely challenging. For instance, Viswanathan et al. adapted a ML-potentials to predict voltage, only for NCM (Ni,Mn, and Co)-based electrodes, using a relatively small DFT based dataset [90]. While the accuracy of Viswanathan's model compares to that of DFT simulations for Li-graphite based electrodes, its transferability to Na-graphite or K-graphite is impractical. This impracticability results from the fact that the Li-graphite specific ML-potentials model performs poorly on other electrodes type. Since such data does not exist for each material for other metal-ion battery, ML models that will work well with diverse metal-ion batteries and transfers equally well is necessary.

For other models including deep neural network, ML-models are trained on nearly 5000 electrodes materials from materials project database with density functional theory calculated voltage as target [91, 92]. Nonetheless, these models have poorer predictive ability than other ML-models used for properties of solids [12]. This poor performance can be attributed to both the small size of the dataset and chemical diversity within such small dataset for the different metal-ion batteries. Also, more advanced ML methods, such as graph-based ML approaches had not been applied yet in the domain of electrode materials. The majority of existing literature mostly employs pre-defined calculated features as input information for the materials. Solely relying on these pre-calculated atomic composition features implies that the model ignores any 3D-structural environment information within the materials and thus misses the key attributes used in reference physics based simulations [91, 92]. However, we anticipate that combining both chemical composition along with some 3D-structural information inside of a more complex deep learning method may lead to a significantly more accurate voltage prediction; as it has been shown for other crystals' properties predictions [12, 13, 91]. In

this work, we adapt a graph-convolutional neural network that learns the robust representation of electrodes materials from the atom types and corresponding 3D-coordinates only. We show that our method provides gradual improvement upon existing work. Our proposed method shows great transferability to new metal-ion battery chemistry as it outperforms all related published works on voltage prediction. Our voltage-prediction solution includes two techniques. The first consists of a model which learns the chemical reaction of input electrodes and outputs their average voltage. The other technique involves using a trained model that predicts E_{form} of individual electrodes. These electrodes' E_{form} are subsequently used within our derived formula to output the voltage-prediction. Comparing our performance to those found in the literature, we show that calculating the voltage from formation energy of electrodes is ideal way to predict voltage in the scenario where there is limited data to train ML models for intercalation reactions.

4.2 METHODS

4.2.1 DATA

DFT computed voltages and structures of electrodes materials for 4402 battery systems were collected from the Material Project (MP) database using Pymatgen Materials Genomes (pymatgen) [95]. The distribution of the data set, which consists of 10 different metal ion (Cs, Y, Zn, Ca, Li, Mg, Na, K, Al, and Rb) batteries, is shown in Figure 1 [96]. Because of its high popularity as a charge carrier, Li (2291) has the highest number of battery systems. The other battery frameworks include Ca-based systems (484), Mg (393), Na (328), Zn (385), Rb (50), and Cs (39)-based electrodes. These electrode distributions are displayed in Figure 4.1.

To develop the two proposed methods below described, we use the previously mentioned data set of 4402 electrodes for the chemical reaction-based model and another data set of about 60000 materials downloaded from the MP database for E_{form}-based model [97]. The two data sets are mutually exclusive of their materials.

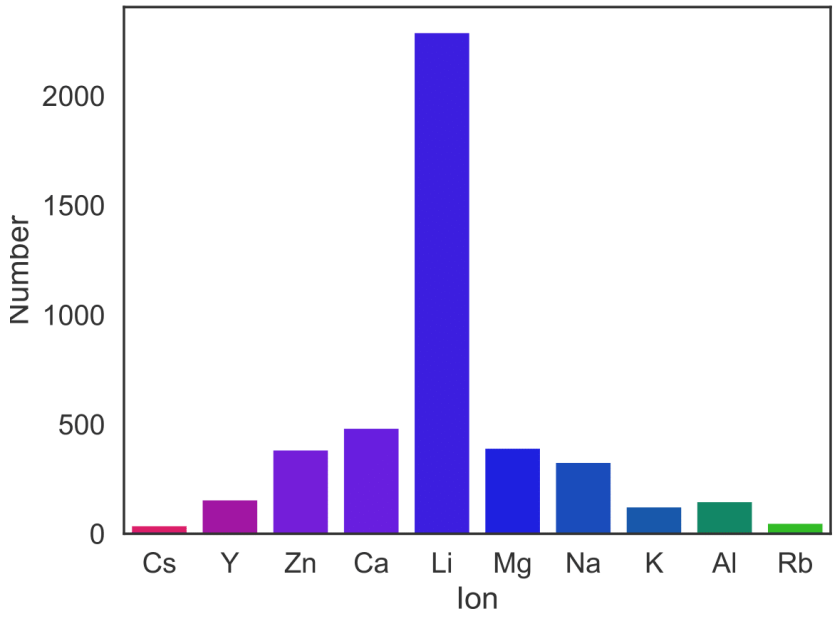

Figure 4.1 Distribution of the numbers of battery frameworks collected from the MP database for each metal ion.

4.2.2 VOLTAGE

Understanding the chemical reaction of an intercalation battery framework is essential to learn the voltage of the corresponding system. As an example, the intercalation reaction of the hypothetical layered material $A_{x_1}B_\alpha C_\beta D\gamma$ and A ions to form $A_{x_2}B_\alpha C_\beta D\gamma$ can be represented as follows:

$$A_{x_1}B_\alpha C_\beta D\gamma + (\mathrm{x}_2 - \mathrm{x}_1)\mathrm{A}^{z+} + z(x_2 - x_1)e^- \leftrightarrow A_{x_2}B_\alpha C_\beta D\gamma \quad (4.1)$$

The electrode on the left-hand side, which reacts with the cation, exhibits a higher potential than the electrode on the right-hand side. Thus, we label the left-hand and right-hand side electrodes as highpotential and low-potential electrodes, respectively. To calculate the voltage in DFT, we estimate the Gibbs free energy of individual electrodes defined as $G = \Delta E + P\Delta V' - T\Delta S$, where ΔE is the internal energy change, P is the pressure, $\Delta V'$ is the volume change, T is the temperature, and ΔS is the entropy difference of the system. However $P\Delta V' \approx 10^{-5}$ eV and $T\Delta S \approx 25$ meV at room temperature. Therefore, by neglecting those two terms, we can calculate the voltage (V) by only considering the internal energy change as shown in eq 4.2. Here, the terms $E[\eta_{x_i}]$ $(i = 1, 2)$ xi are the total energy of the chemical formula (η) with x_1 and x_2 contents of the ion (A), and z is the valency of the intercalating metal ion. For instance, $\eta_{x_1} = A_{x_1}B_\alpha C_\beta D\gamma$ and $\eta_{x_2} = A_{x_2}B_\alpha C_\beta D\gamma$ for the chemical reaction of eq 4.1. For the ions mentioned in Figure 1, z = 1 for Li, Na, K, Rb, and Cs, z = 2 for Ca, Mg, and Zn, and z = 3 for Al and Y.

$$V \approx \frac{1}{z(x_2 - x_1)e}\left(E[\eta_{x_1}] - E[\eta_{x_2}] + (x_2 - x_1)E(A)\right) \tag{4.2}$$

$$E_{form} = E[\eta_{x_1}] - x_1E(A) - \propto E(B) - \beta E(C) - \gamma E(D) \tag{4.3}$$

The Eform per unit formula of $A_{x_1}B_\alpha C_\beta D\gamma$ electrode is given by eq 4.3, where E(A), E(B), E(C), and E(D) represent the energy of each atom in their bulk phase. By computing the difference between formation energies of high and low electrodes, we can show that voltage can be determined using eq 4.4.

$$V \approx \frac{1}{z(x_2 - x_1)e}\left(E_{form_{x_1}} - E_{form_{x_2}}\right) \tag{4.4}$$

However, we can calculate the voltage in two different ways with ML: (I) by training the ML models directly to learn the voltage and (II) by training ML models to learn the formation energy of involved highand low-potential electrodes. We compare the performance of the model trained on each case. In the first method, the structures corresponding to high- and low-potential electrodes for an intercalation reaction are used simultaneously (hence labeled reaction-based model). In contrast, for the second method, the structure of each compound is input separately to first learn the formation energy (labeled E_{form}-based model). Once the formation energy is predicted, the corresponding voltage can then be calculated by using eq 4.4.

4.2.3 GRAPH NEURAL NETWORK ARCHITECTURE

Graph neural networks (GNN) are deep neural networks that have been applied to effectively learn latent features from network or graph data [1]. Our models are based on a subset of these GNNs that adapt the technique of the attention mechanism to GNN [14, 98]. Particularly, we adapt the technique of GATGNN introduced by Louis et al. [50] for this study.

The input to our GATGNN adapted models is materials encoded as a graph with nodes representing atoms and edges representing those nodes connections. We encode a material as a graph where each node (atom) connects with the 16 nearest nodes (atoms). Each atom type is then attributed a 92-dimensional vector, and each edge's distance is also encoded as a 41-dimensional vector [50].

Compared to other GNN applied in the domain of materials, GATGNN learns each atom's contribution both locally (within a local atomic space) and globally (with respect to all atoms in the material) [12, 13, 99, 100]. The overall architecture of the

GATGNN model is shown in Figure 4.2. GATGNN first efficiently captures the atoms' local importance through its augmented graph attention layers (AGAT) and then a global attention.

In our research, we adapt the proposed GATGNN to construct our models for learning voltage from materials. For the local attention, our model consist of four AGAT layers of four attention heads, each consisting of 64 neurons. The local soft-attention αi,j between a node i and a neighbor j can be represented as

$$\alpha_{i,j} = \frac{\exp\left(a_{i,j}\right)}{\sum_{k \in N_i} \exp\left(a_{i,k}\right)} \tag{4.5}$$

where N_i denotes the neighborhood of node i and $a_{i,j}$ is the parametrized weight coefficient between nodes i and j, which denotes the importance of node j to node i in equation 4.5.

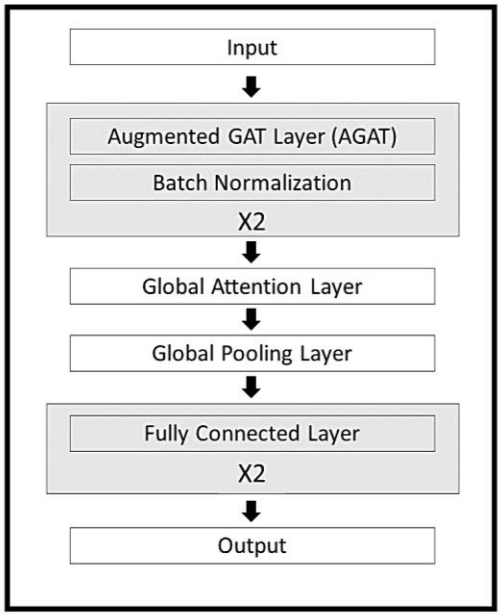

Figure 4.2 An overview of the GATGNN architecture

Upon learning the local importance of the atoms, we subsequently use a single fully connected layer to learn this global attention value. In our study, the two inputs needed for the global attention layer are an atom's embedding and a material's compositional vector [50]. The global attention g_i can be described as follows:

$$g_i = \frac{(\boldsymbol{x}_i || \boldsymbol{E}) \cdot \boldsymbol{W}}{\sum_{x_c \in \boldsymbol{X}} (\boldsymbol{x}_c || \boldsymbol{E}) \cdot \boldsymbol{W}} \tag{4.6}$$

$\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{F}}$ denotes a learned embedding, $\boldsymbol{E}$ denotes a compositional vector of the crystal, $\boldsymbol{W} \in \mathbb{R}^{1 \times (F + |E|)}$ denotes a parametrized matrix, and x_c denotes the learned embedding of any atom c within the crystal in eq 4.6.

We implement all the components in our proposed approach using deep learning libraries of Pytorch and the library of PytorchGeometric [77, 101]. The same SmoothL1 loss function is used to train both models [102].

4.2.4 CHEMICAL-REACTION BASED VOLTAGE PREDICTOR

Because of their recent advances, ML models have been increasingly adopted for learning the properties of chemical reactions of molecules [103, 104]. These models, however, have not been used for reactions involving crystals. Hence, we developed a GNN model that considers the chemical reaction of electrodes and metal ions as the input. Our proposed model is based on the previously existing GATGNN method.

Mainly, our reaction-based model consists of two modified GATGNN ($GATGNN^R$) modules arranged in parallel, which are both followed by a series of hidden fully connected layers. Figure 4.3 illustrates the framework of our proposed model. To start, a low and a high electrode are input into the low/high dedicated $GATGNN^R$ module which learn from their corresponding electrodes. Following the graph convolutions, the

output of both blocks or modules is then concatenated into a 128-dimensional vector to be fed to two fully connected layers.

The final predicted average voltage is calculated by learning the chemical interaction of the two electrodes. Hyperparameters were optimized for all the models used in this work. We train the model for 500 epochs with early stopping using a learning rate of 1×10^{-3} , a weight decay of 5×10^{-3} , and a batch size of 128.

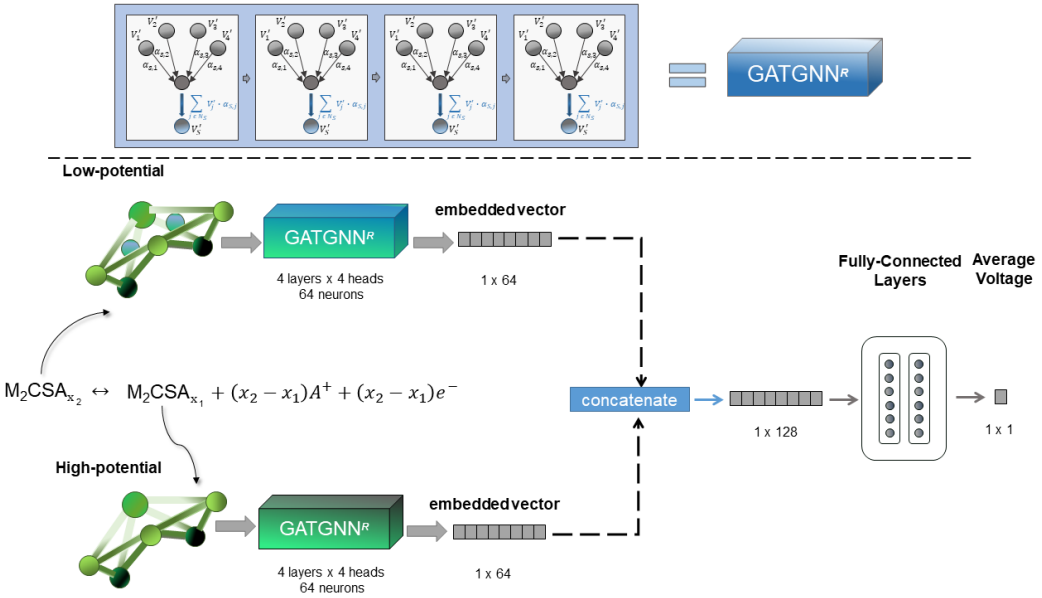

Figure 4.3 Architecture of our reaction-based average voltage model. The top panel shows the underlying GATGNN modules used in the work.

4.2.5 FORMATION ENERGY BASED VOLTAGE PREDICTOR

Our proposed E_{form} -based model is based on an optimized pretrained model of a GATGNN of four layers 128 neurons with four attention heads to model crystals to their Eform. As illustrated in Figure 4, the proposed method consists of one GATGNN which independently outputs the E_{form} for the low- and high-potential electrodes. This optimized E_{form} -based GATGNN model was trained for 300 epochs early stopping by using a learning rate of 1×10^{-3} , a weight decay of 5×10^{-3} , and a batch size of 64. For

the data set, we split the data into three sets: with 85% used for training, 7.5% for testing, and 7.5% for validation. Following the prediction of the electrodes' E_{form}, their average voltage is subsequently obtained by using eq 4.4.

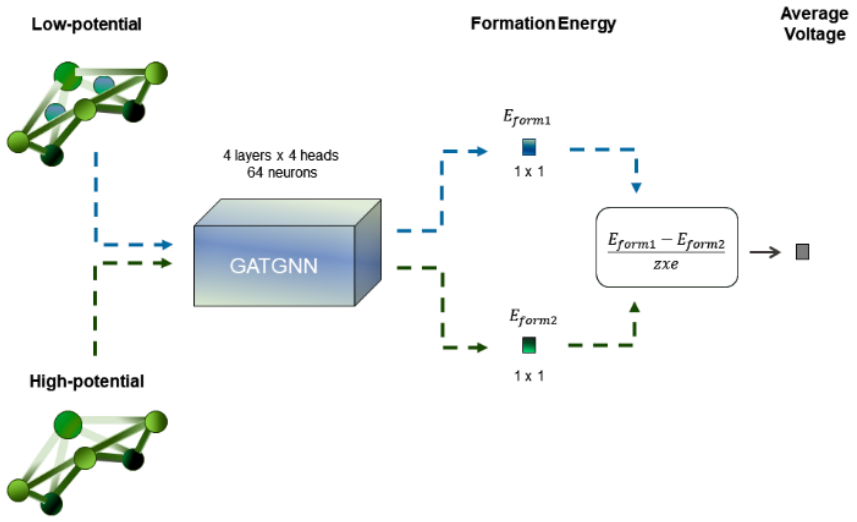

Figure 4.4 Architecture of E_{form}-based average voltage model.

4.2.6 EXPERIMENTS

To assess the validity of our proposed method, we conducted two different experiments. In the first experiment, we do a 10-fold cross-validation as done in the works by Moses et al.16 In the second experiment, we apply the holdout test with a data split of 85% used for training, 7.5% for testing, and 7.5% for validation. Notably, we use the first experiment to compare the performance of our proposed method to other voltage works that only use composition. In contrast, we use the second experiment to compare the performance of using the reaction-based model to the Eform-based model.

4.3 RESULTS AND DISCUSSION

To evaluate the performance of the models, we used the metric of mean absolute error (MAE) defined in the equation

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |V_i^{DFT} - V_i^{ML}| \tag{4.7}$$

where V_i^{DFT} represents the voltage computed from DFT, V_i^{DFT} the machine learning predicted voltage, *i* a given battery sample, and N the total number of samples in a data set. Table 1 displays the cross-validation results obtained from our first experiment and the reported results from the composition-based experiments.16 Whereas the data splits may be different, the folds were obtained from the same electrodes data set. Based on the listed MAE values, our proposed reaction-based model outperforms the composition-based model by more than 13.6%. Our reaction-based model achieved an overall average MAE of 0.34 with 0.02 standard deviation whereas the composition-based mode did 0.39 with the variance. From the results, we can conclude that our proposed method considerably benefits from the additional structural information and the learned reaction of the two electrodes.

In our second experiment, the testing set consisted of 329 battery systems comprised of 177 Li, 35 Ca, 3 Cs, 7 Rb, 11 K, 10 Y, 20 Na, 11 Al, 29 Zn, and 26 Mg battery systems. To prevent data leakage, we made sure to remove all electrodes from the 60,000-size data set used to pretrain the E_{form} -based model. The same testing set is used for evaluating our E_{form} -based approach. Figure 4.5 reports the corresponding MAE and the parity plot comparing the DFT and ML voltage for both the reaction-based and the E_{form} -based models from the second experiment. As can be seen, both reaction-based (in red) and E_{form} -based (in blue) models achieve good performance with a MAE of 0.34

and 0.31. Even though the models used in our two different proposed methods are trained by using significantly different training sets, both in terms of size (60000 vs 4402) and data (electrodes vs non-electrodes), it is noteworthy to observe that they both achieve significantly lower MAE performance than the initially proposed composition-based model [92]. The holdout test experiment resulted in a MAE value of 0.315 for the E_{form} -based model and 0.344 for reaction-based model.

Table 4.1 MAE Results for the 10-Fold Cross-Validation Study Comparing Our Proposed Reaction-Based Model to the Other Method's Composition-Based Model (Average MAE and Std Listed in the Last Column)

Model	1	2	3	4	5	6	7	8	9	10	Avg & std
Reaction-based	0.34	0.34	0.35	0.32	0.35	0.36	0.39	0.33	0.34	0.31	0.34±0.02
Composition-based	0.42	0.39	0.38	0.38	0.43	0.39	0.38	0.39	0.39	0.37	0.39±0.02

While the MAE obtained from E_{form} -based model is about 8% lower than the reaction-based approach, this improvement comes at the cost of using more than 13 times more training samples (see Figure 4.5). Nonetheless, the E_{form} -based model does not learn directly from the battery electrodes. Therefore, we suggest that neither method is superior to the other, but instead they are alternatives. In situations where there is a lack of battery electrodes, the E_{form} -based method can be a good way to study the voltages of intercalation reactions.

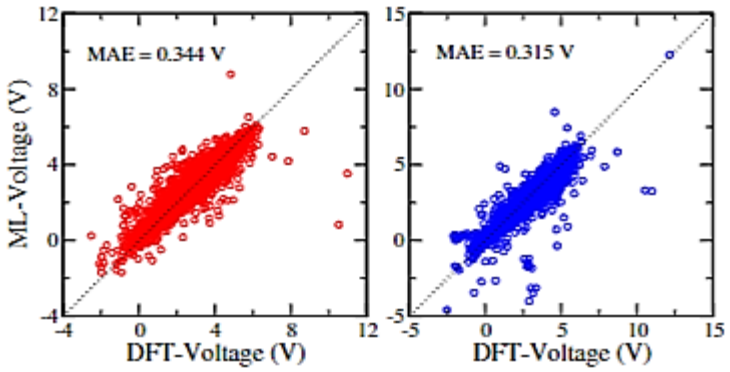

Figure 4.5 Parity Plot comparing the DFT (DFT-Voltage) to the ML (ML-Voltage) Voltage for (a) – left: the reaction-based and (b): right E_{form}-based models.

To examine the transferability of our models, we evaluated our model's performance at predicting the voltage in Na- and K-ion-based electrodes. For this, we replaced the Li ions in known Li electrodes structures from the MP database with Na and K ions. For such structures, geometry optimization was not performed to calculate the voltage from our ML models. This approach allows us to examine the effect of geometry optimization on the performance of our models while predicting the voltage. Our approach is motivated by benchmark ML models used for molecules, which have achieved chemical accuracy with empirically obtained 3D coordinates from SMILES strings of molecules [105]. From the MP database, we extracted only the DFT voltage of those Na- and K-ion-based electrodes, which share the same structure (symmetry, space groups) as the corresponding Li-based electrodes. As listed in Table 4.2, the Na ion shares a relatively large amount of Li electrode structures (127), while K-ion electrodes have only 32. We replaced Li in those Li-based electrodes with Na and K ions and calculated the voltage using our ML models. We compared such voltage for Na- and K-based electrodes with the corresponding DFT voltage taken from the MP database (see Figure 4.6).

Table 4.2 Number of Na and K Battery Frameworks That Share the Same Structures of Li Frameworks (Test Set Size) and the MAE Values in V for Predicting Voltage of the Battery Systems, Where Li Ions Were Substituted by Those Alkali Ions

		MAE GATGNN	
Alkali ion	Test set size	Reaction-base	E_{form} -based
Na	127	0.56	0.47
K	32	0.87	0.70

From these examinations, we obtained a somewhat large MAE of 0.56 V from the reaction-based model. For the K-ion battery electrodes, we obtained moderately large errors from both reaction-based and formation-energy-based models. We found that formation-energy-based models are more accurate at predicting voltage for a K-ion battery with a MAE of 0.70 V. The atomic radius difference between K and Li atoms is larger than that between Na and Li atoms. Therefore, the electrodes where Li was replaced by K exhibit more significant error compared to that from Na counterparts, when we predict the voltage without optimizing the atomic positions and the geometry as above. Although first-principles calculations always require geometry and cell parameter optimization before calculating voltage, our proposed E_{form} -based approach can predict the voltage of Na battery frameworks without the optimization procedure.

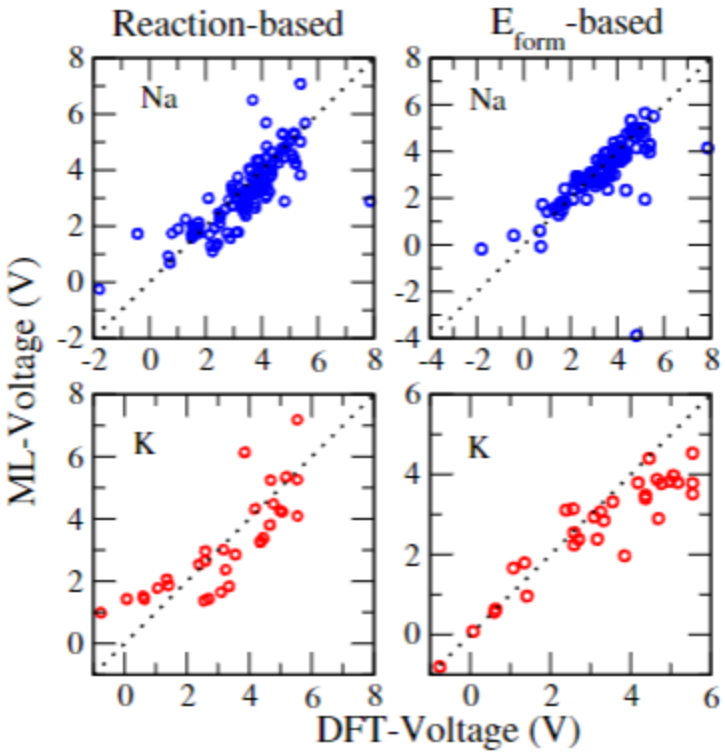

Figure 4.6 Performance of reaction-based and E_{form}-based models to predict the voltage of Na and K-ion based electrodes.

To further study the accuracy in predicting the voltage of the battery frameworks where Na substituted Li, we compared the predicted voltage from our Eform-based model (VGATGNN) to the DFT calculations (VDFT) reported by Moses et al., as shown in Table 3. In works done by Moses et al., the authors screened the MP database to find Na-ion battery electrodes with high voltage and minimal volume change during the charging and discharging using only the composition of electrode materials [92]. We find that 18 battery frameworks of their data set are in our list of new Na electrodes and listed in Table 2. We also compared our predictions with the voltage predicted with simple deep neural network from their research referenced as VDNN [92]. In Table 4.3, |ΔVGATGNN| is the absolute difference between VDFT and VGATGNN, while |ΔVDNN| is that between VDFT and VDNN. It is clear that the error from our model

(|ΔVGATGNN|, 0.34 V) is considerably smaller than |ΔVDNN| (0.89 V). We also found that the composition-based model (VDNN) predicts a large positive voltage of 2.85 V for the NaV2O4 and Na2VO2 electrode pair, whereas our model was able to correctly predict the negative voltage of −0.21 V, which is close to the DFT predicted value of −0.35 V. This shows that our approach can more accurately represent the energetics of electrode materials compared to traditional composition-only-based models.

Table 4.3 Comparison of the Voltages Calculated Using DFT (V.DFT) and That Predicted by the $Eform$-Based Model (V.GATGNN). The DFT calculations were performed by Moses et al. after optimizing the electrodes where Li ions were replaced by Na. Our VGATGNN values were predicted without relaxing the structures. We also provide the voltages for respective electrodes predicted by the composition-based DNN model (V.DNN). |ΔV.GATGNN| is the absolute difference between V.DFT and V.GATGNN, while |ΔVDNN| is the absolute difference between V.DFT and V.DNN. The voltage values and absolute voltage differences were calculated in V.

Formula-high	Formula-low	V.DFT	V.GATGNN	V.DNN	|ΔV.GATGNN|	|ΔV.DNN|
$NaMn_3OF_8$	$Na_4Mn_3OF_8$	3.15	3.42	3.20	0.27	0.05
$NaCuF_4$	Na_2CuF_4	4.31	4.56	4.01	0.25	0.30
Na_2CrO_4	Na_4CuF_4	1.46	1.27	3.07	0.19	1.61
$TiCrO_4$	Na_2TiCrO_4	1.54	2.28	3.31	0.74	1.77
$Mn_3P_6WO_{24}$	$Na_6Mn_3P_6WO_{24}$	3.47	3.54	3.50	0.07	0.03
$CrWO_6$	Na_2CrWO_6	3.75	3.41	3.10	0.34	0.65
$NaVTe(WO_6)_2$	$Na_4VTe(WO_6)_2$	2.90	2.78	3.20	0.12	0.30

Another data set that we considered was published by Ong et al. to study the voltages of Li- and Na-ion intercalation materials listed in Table 4 [106]. In ref [106], DFT calculations of NaMPO4 (M = Fe, Mn, Co, and Ni) materials with Olivine and Maricite structures were performed. Those two classes of materials have Pnma space group symmetry. Even though both have similar structures, Na and M atoms switch their sites in the two material types (i.e., in maricite structures, Na is at M's site and M is at Na's site of olivine structure). However, the DFT calculations in Table. 4.4 show that

both structures exhibit similar theoretical voltages for a given compound. The E_{form}-based model predicted voltages provide a good accuracy (MAE = 0.23 V).

Table 4.4 Comparison of the Voltages Calculated by Ong et al. (V.DFT) and That Predicted by the $Eform$-Based Model (V.GATGNN). |ΔVGATGNN| is the absolute difference between VDFT and VGATGNN. Here, the DFT calculations were performed after optimizing the electrodes where Li ions were replaced by Na. Our VGATGNN values were predicted without relaxing the structures. The voltage values and absolute voltage differences were calculated in V.

Material	Structure	V.DFT	V.GATGNN	ΔV.GATGNN
$NaFePO_4$	olivine	3.08	3.36	0.28
$NaMnPO_4$	olivine	3.59	3.85	0.26
$NaCoPO_4$	olivine	4.19	3.71	0.48
$NaFePO_4$	maricite	3.13	3.04	0.09
$NaMnPO_4$	maricite	3.48	3.71	0.23
$NaCoPO_4$	maricite	4.09	4.05	0.04

Finally, we predicted the voltages of new Na electrode materials with the $(PO4)^{3-}$ polyanion group. These are an interesting class of materials as inductive effects of $(PO4)^{3-}$, $(P2O7)^{3-}$, and $(SO4)^{3-}$ polyanions offer high operating voltage. As an example, DFT calculations of $NaxMnM'(PO_4)_3$ (M′ = Cr, Ti, and Zr) show that an average voltage around 4 V can be obtained. Na3V2(PO4)2F3 fluorophosphate has a theoretical voltage of 3.9 V. Thus, $(PO4)^{3-}$-based electrodes are popular as high-density cathode materials for Na-ion batteries. $Na_3(MPO_4)_2F_3$ (M = Al, V, Fe, Cr, and Ga) and AVPO4F (A = Na and Li) have been widely studied as the cathode materials for Li and Na batteries [107, 108]. Table 4.5 contains predicted voltages for 7 fluorophosphate materials, which are not included in the MP database. The $NaTiPO_4F$ and $TiPO_4F$ pair of electrodes provides a

considerably lower voltage than that of the same family of electrodes with the other transition metal atoms. Ti-based compounds have been widely investigated as anode materials for Na-battery electrodes due to low operating voltages and stability [109]. It is reported that NaTiO2 (1.37 V) electrode has a low theoretical voltage, even though Ni- and Co-based counterparts have voltages higher than 3.3 V [106]. The battery framework with the NaCuPO4F and CuPO4F pair of electrodes shows the highest voltage, which is 5.52 V, in Table 4.5. All the other battery systems in Table 4.5 exhibit theoretical voltages between 3.1 and 4.5 V.

Table 4.5 Voltages Predicted by Eform-Based Model for 10 Fluorophosphate-Based Battery Frameworks, Which Are Not Included in the MP Database for Na Batteries (Voltage Values Calculated in V)

Formula-low	Formula-high	V.GATGNN
$NaTiPO_4F$	$TiPO_4F$	1.32
$NaCrPO_4F$	$CrPO_4F$	3.48
$NaFePO_4F$	$FePO_4F$	3.22
$NaCuPO_4F$	$CuPO_4F$	5.52
$NaMnPO_4F$	$MnPO_4F$	4.48
Na_2CoPO_4F	$NaCoPO_4F$	3.48
Na_2MnPO_4F	$NaMnPO_4F$	3.42

4.4 CONCLUSION

In summary, we developed two attention-based graph neural networks that combine the chemical compositions with spatial information to predict the voltage of the battery electrode materials. The first method predicts the voltage by considering the

chemical reaction between a high-potential electrode and the metal ions to form a low-potential electrode. The second model predicts the E_{form} of individual electrodes before being used to compute the voltage. Results were compared with the latest composition-based model [92] from the literature for predicting voltage. Our structure-based models are much more accurate than this benchmark work. Our E_{form}-based model consistently provides lower MAE compared to that from reaction-based model. And, we show that relative to known models in the literature, our E_{form} based model demonstrates high transferability of performance when applied to Na electrodes. Furthermore, we predicted the average voltages of 10 fluorophosphate-based battery frameworks, which are not included in the MP database for Na batteries. Those fluorophosphates have the NaxMPO4F (M = Ti, Cr, Fe, Cu, Mn, Co, and Ni) general chemical formula. We could show that it can expect average voltages greater than 3.1 V from those Na battery frameworks except from the NaTiPO4F and TiPO4F pair of electrodes, where it exhibits an average voltage of 1.32 V.

CHAPTER 5

LAST WORK: ADAPTING GRAPH NEURAL NETWORK WITH STABLE DIFFUSION FOR THE GENERATION OF CRYSTAL STRUCTURES

5.1 MOTIVATION

Generating a valid structure for stable materials is a very important and challenging problem. This task proves to be very difficult because of the large number of atoms to be mapped, the infinite number of atomic arrangements in space, and the physical requirement that these atomic positions lie in local energy minimum per quantum mechanics [110-112]. Generating Physiochemically acceptable coordinates for materials has the potential to lead to a lot of technological breakthroughs since solid state materials are the foundation of important technologies on which the world depends [113]. Hence addressing this problem by using Graph Neural Network coupled with a robust generative technique represents a nice opportunity to make an important contribution to the scientific realm of materials.

5.2 INTRODUCTION

The task of molecule generation using deep learning is a very difficult problem. Specifically, generating stable inorganic materials is of the utmost difficulty. Compared to the actual generational process that takes place in the lab, the application of a deep learning generative method would significantly improve the discovery of physiochemically acceptable novel crystal structures. Requiring a lot less resources meanwhile taking less time, the implementation of the right generative crystal model has the potential to lead to a lot of technological [110].

Addressing the crystal structure generation problem with deep learning involves adapting an appropriate machine learning (ML) architecture and an effective generative technique [114]. Given the chemical bonds shared by the atoms in the molecules, graph

neural networks (GNN) represent an ideal class of ML algorithms to learn from the crystal materials [114]. GNNs are a class of deep learning models designed to operate on graphs, which are highly effective at representing complex relationships and dependencies between data entities [1, 8, 32]. In the context of molecules, GNNs can be used to learn from the graph representation of molecular structures, which includes atoms as nodes and chemical bonds as edges. GNNs have already been applied to various tasks in inorganic chemistry, such as predicting materials physical properties, predicting unit cell parameters, or even discovering new materials [50, 110, 115]. Solving the problem of generating materials also requires the selection of an effective generative technique to couple with GNNs. Other than VAE and GANs which are great options, Stable diffusion also stands as an alternative generative technique for addressing our structure generation problem.

Stable diffusion is a promising generative method that has been recently introduced as a powerful alternative to create high quality data [116]. Considered as a specialized class of VAE models, stable diffusion adapts a probabilistic approach to learn the reversed Markov chain that progressively destroyed the original input data [116]. In the context of molecules, adopting a stable diffusion model means training a model that learns how to sequentially optimize atomic positions. Therefore, stable diffusion models can learn the underlying physiochemical rules that bind the atoms interaction in our studied molecules. This approach can provide a more accurate description of the underlying physics, leading to more realistic and stable structures.

In this work, we propose a method called Structural-Diffusion that adapts stable diffusion and GNNs to generate crystal structures from an input formula. Structural-

Diffusion fits in the class of Conditional generative models. Structural-Diffusion receives as input a compositional formula and outputs corresponding crystal structures. To our best knowledge, our method is the first adaptation of GNNs and Stable Diffusion that operates on the original data input. Our main contributions include:

- We implemented a robust generative approach for generating inorganic materials structures from formula input.
- Our model operates on the original form of the input data. From a technical perspective, our model is the first true adaptation of stable diffusion with deep learning for how it directly learns on un-transformed data.
- Compared to other methods, our StructR-Diffusion requires a lot less resources and training data to achieve high performance.
- Our model achieves proves that it can generate diverse structures that are optimized even prior to DFT relaxations.

5.3 RELATED WORK

Generative models have shown promising results in the discovery of new inorganic material structures [117, 118]. Traditional methods for discovering new materials structures involve trial-and-error experimentation or computationally expensive simulations [117, 118]. Generative models, on the other hand, can generate novel structures based on existing data and can accelerate the discovery process by suggesting potential candidates for further investigation. These models use machine learning algorithms to learn patterns and relationships in large datasets of existing materials

structures and can generate new structures that satisfy specific criteria, such as stability and functionality.

Researchers have adapted various variants of GAN and VAE techniques to generate new structures. Zhao et. al. developed a GAN model that encoded the material's information as 2D matrices generate materials from the cubic crystal system. Zhao's method, named CubicGAN, achieved great generative performance but was limited to only generating cubic materials from only 3 space-groups [111]. Xie et. al. recently proposed a method called CDVAE in which the authors combine VAE and Stable Diffusion to generate new inorganic materials. Even though Xie et. al. employed stable diffusion in their models, the authors only partially applied it on the latent variable of the crystal input rather than the original data input [110]. Hoogeboom et. al. also published a similar work in which they implemented an equivariant model that learns to generate new organic molecules by substituting candidate atoms and optimizing the coordinates [119]. While Hoogeboom's EDM closely resembles our method, the domain knowledge of organic molecules is a lot simpler than our targeted domain of inorganic chemistry [110]. GeoDiff and Torsional Diffusion are two other similar methods that succumb to the same limitation as EDM [119, 120]. To the best of our knowledge, our method is the first approach that combines GNNs and stable diffusion on the domain of materials science to generate a new structure for an input formula.

5.4 PROPOSED METHOD

5.4.1 DATASET AND ENCODING

The Materials Project database is used to collect a total of 52,548 crystal materials along with their stability information. In our experiment design, we decided to only use

stable materials containing between 3 and 8 atoms, hence reducing our dataset to 11,501 data samples. We split our dataset using ratio of 80:10:10 for the training, validation, and testing sets. We provide the distribution of the crystal system distribution and atomic distribution in Figure 5.1. On the left, we illustrate the atomic distribution as a heatmap of the periodic table while on the right we display the bar-plot of our dataset's crystal system distribution.

To encode the materials, we used the graph defining scheme that is similar to the one used in CDVAE [110]. We begin by formulating an input graph G as the set $G = (\boldsymbol{V}, \boldsymbol{P}, \boldsymbol{C}, L)$ where $\boldsymbol{V} = \{v_0, v_1, \dots, v_N\}$ defines the set of nodes that represent the material's atoms and N the number of atoms, $\boldsymbol{P} = \{p_0, p_1, \dots, p_N\}$ the numerical index of the atoms in the crystal structure, $\boldsymbol{C} = \{c_0, c_1, \dots, c_N\}$ the atomic coordinates, and $L = \{a, b, c, \alpha, \beta, \gamma\}$ the lattice parameters of the crystal material.

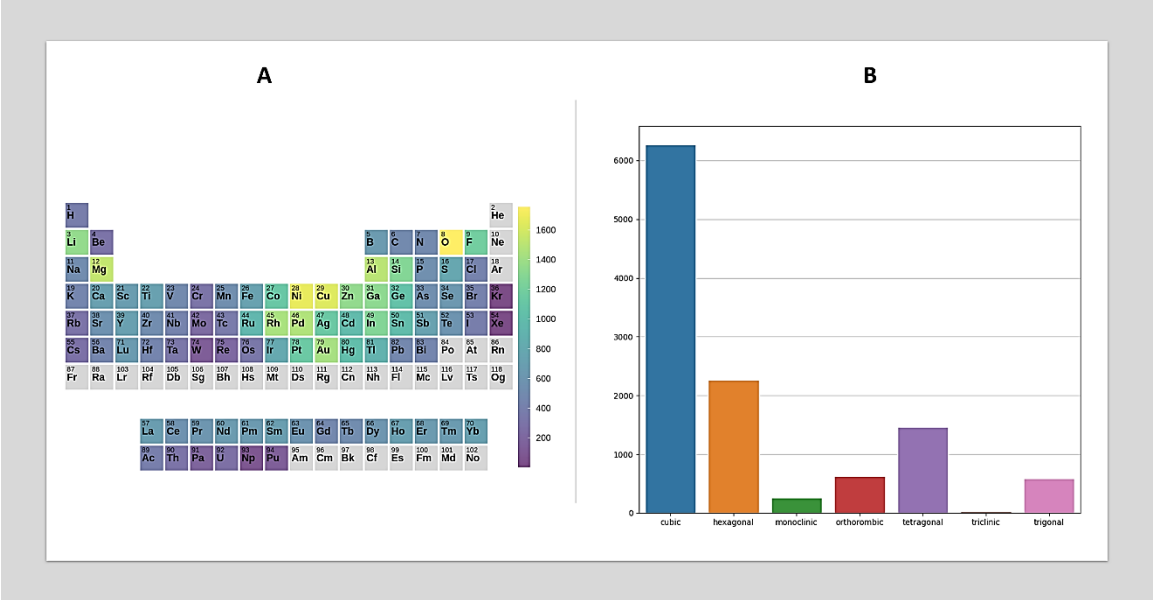

Figure 5.1 Distribution of the atom frequency in the dataset on the left in A. Distribution of the crystal systems counts in the dataset in B.

We encode the atomic attributes using the method provided by Goodall et. al. in their work of representational learning on materials, thus resulting in $\boldsymbol{V} \in \mathbb{R}^{N\times 200}$ [121]. We apply the positional encoding, used in both Transformers and Stable-Diffusion original works', to encode the numerical index of the atoms thereby resulting in $\boldsymbol{P} \in \mathbb{R}^{N\times 32}$ [34, 116, 122]. The dimensionality of $\boldsymbol{C}$ and L are $N \times 3$ and 1×6, or $C \in \mathbb{R}^{N\times 3}$ and $L \in \mathbb{R}^{1\times 6}$. In the context of our generative problem, we only aim to generate new coordinates and lattice parameters. We formulate our generative problem as a function $f: f(\boldsymbol{X}, \boldsymbol{P}) = (\boldsymbol{C}, L)$, that generates a new spatial arrangement (coordinates) and unit cell shape-size combination for an input formula.

5.4.2 STABLE DIFFUSION

As in the prior works that used GAN and VAE to generate crystal structures, our Stable Diffusion model also learns the structural data distribution along with crucial underlying physicochemical laws of crystallography. Figure 4.2 displays the 3 distinct distributions that our model targets to learn. In our experimental design, we treat the coordinates data, the abc parameters, and the $\alpha\beta\gamma$ parameters as variables from three different distributions. The validation of this approach is based on various statistical tests, i.e. 2-Sample Kolmogorov-Smirnov test that were conducted during the early stages of this work. To achieve the objective of learning the 3 distributions, our diffusion model learns the reverse of the diffusion process.

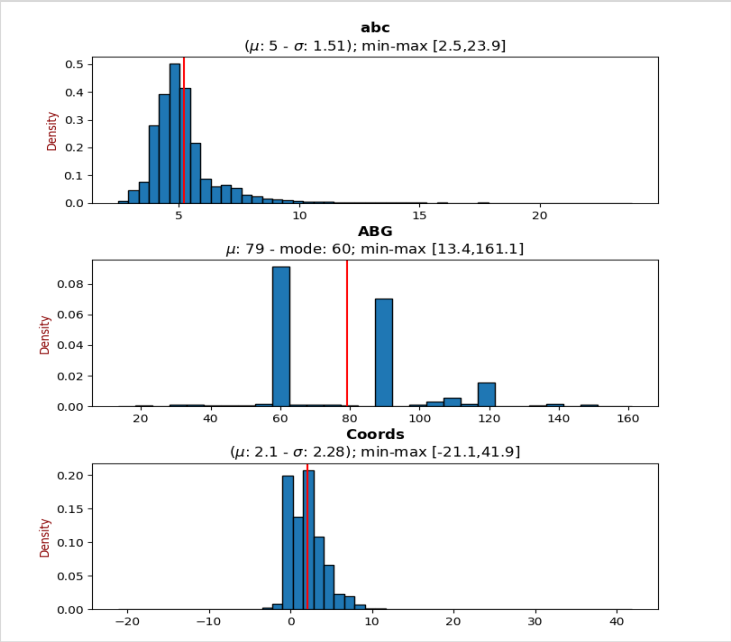

Figure 5.2 Illustration of the 3 data distribution that make up the structure of crystal materials.

Diffusion defines the process during which some small noise amount $z \in \mathcal{N}(0,1)$ is sequentially added to the graph data G for a total of T timesteps. With G_0 as the ground truth crystal structure, G_t for $t = 1, \dots, T$ defines the disturbed graph structure at the t diffusion step. As t increases, G_0 will be gradually diffused until it eventually becomes comparable to a white noise. Also referred to as forward process, the diffusion can also be formulated as a fixed posterior distribution q such that:

$$q(G_t|G_{t-1}) = \mathcal{N}\left(G_t; \sqrt{1-\mathrm{B_t}} \cdot G_{t-1}, \mathrm{B}_t\right) \tag{5.1}$$

where $\mathcal{N}$ defines a conditional Gaussian distribution whose mean depends on the previous graph G_{t-1} and B_t: $\mathrm{B}_1, \dots, \mathrm{B}_T$ a fixed variance schedule. In our experiment, we use a total of $T = 10$ timesteps and linearly scheduled values for $\mathrm{B}_t = [10^{-4}, 0.2]$. The forward process is illustrated in Figure 5.3 as the Diffusion Process with the flow going rightward.

As timestamps increase, the structure of the original graph is perturbed until the graph structure converges into a white noise distribution.

In the reverse process, the goal is to construct a plausible graph crystal structure from G_0 starting from G_T. This learnable denoising procedure can be formulated as a parametrizable p_θ such that:

$$p_\theta(G_0, G_T) = \prod_{t=T}^{1} p_\theta(G_{t-1}, G_t) \tag{5.2}$$

with θdenoting the parameters of a neural network. Figure 4.3 illustrates the denoising procedure as the sequence of operations from left to right. Simply put, the aim is to have a neural network learn to iteratively improve a graph structure G_t to a refined structure form G_{t-1}.

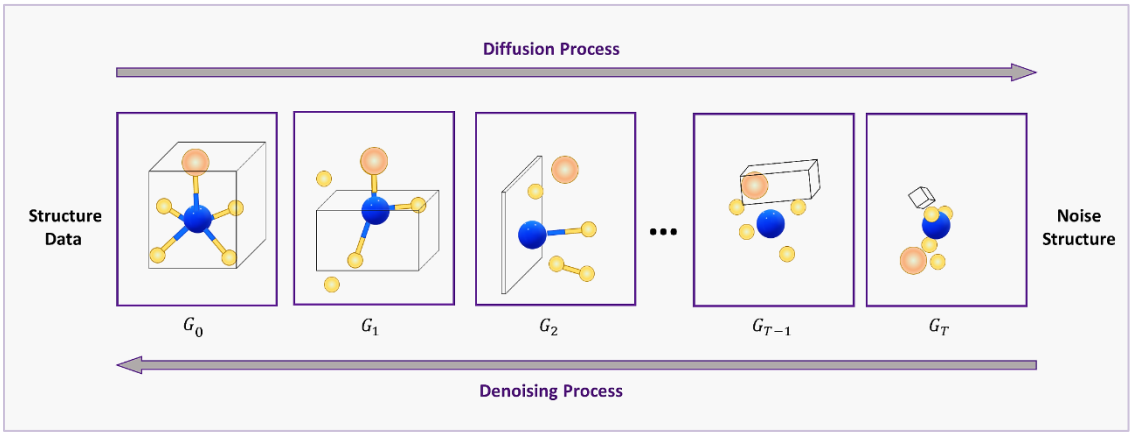

Figure 5.3 High level explanation of the forward and reverse process involved in the stable diffusion technique. The operations moving left to right read the diffusion process in which noise is sequentially added to the original data. The operations moving right to left define the learnable denoising process in which a noisy structure is generatively refined.

4.4.3 MODEL

Our approach to conditionally generate a crystal structure for an input formula involves training 3 neural network models corresponding to the 3 distributions that we

want to learn: the lattice *abc* parameters, the lattice $\alpha\beta\gamma$ parameters, and the coordinates. The 2 lattice models are identical and share the same architecture. Each lattice model is composed of blocks of Fully Connected, Batch normalization, Positional Embedding, and Dropout layers [38]. The coordinates model is composed of various graph-transformer, Group Normalization, Dropout, and Positional Embedding layers [38, 39, 123, 124] . The codes of our proposed method of StructR-Diffusion are in the process of being released as an API. If one wants to access our software codes prior to the release they can submit a request form using the url https://forms.gle/316Ar7h73qmREeBu6 and henceforth access the private GitHub repository https://github.com/superlouis/generative-research.

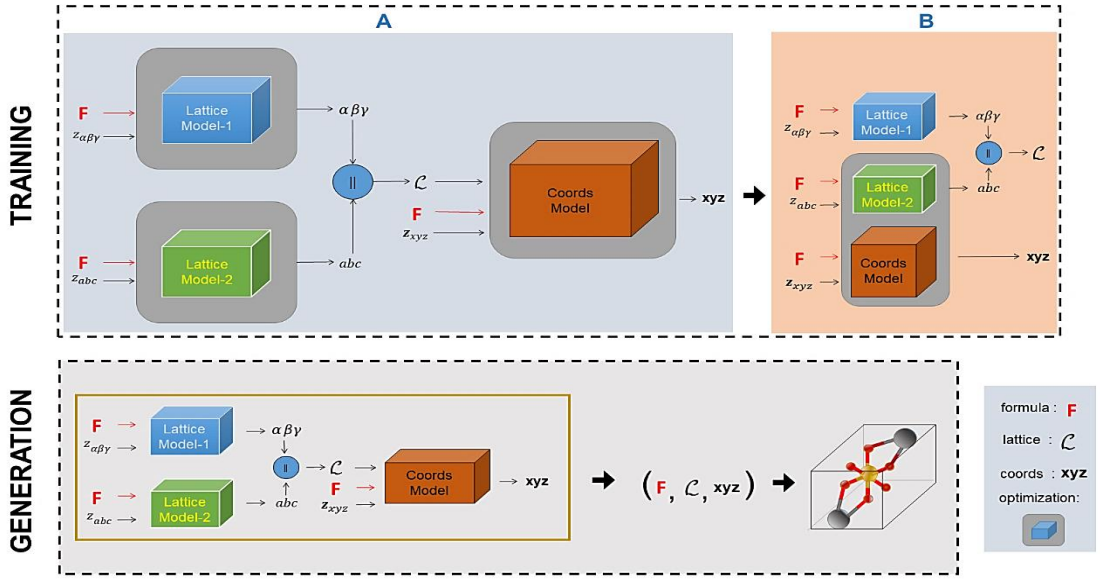

Figure 5.4 Architecture of our proposed Method. The training scheme involves two parts listed as **A** and **B**. In **A**, the individual models are independently optimized to output the lattice parameters and atomic coordinates. In **B**, the abc-model and coordinate model are together optimized to further enhance the quality of the generated output. For generating new structures, random noise is fed as input to each individual model component and the outputs are then used to generate a new structure.

Figure 5.4. illustrates the general overview of both the training and generation strategy used in StructR-Diffusion. The top illustration displays the two stages involved in training the model. In the first stage: A, the 3 models are independently optimized to

predict a refined data d_{t-1} from a diffused input d_t and a formula F. Our introduced variable of data d_{t-1} corresponds to abc_{t-1} for the abc model, $\alpha\beta\gamma_{t-1}$ for the $\alpha\beta\gamma$ model, and C_t for the coordinates. In the second stage of the training: B, the abc and coordinates are simultaneously optimized while the $\alpha\beta\gamma$ model is frozen. This is done to improve the quality of the generated structures by enforcing the network to capture the relationship between the coordinates and lattice parameters. The bottom illustration displays how our trained StructR-Diffusion model generates a new structure. Starting with a chemical formula and a random noise sampled from standard normal distribution, our model iteratively denoises the provided input until timestep $t = 0$.

4.4 RESULTS & DISCUSSION

To evaluate the performance of our proposed generative method, we conducted a series of statistical measurements on the generated structures for a total of 560 chemical formula sampled from our training, validation, and testing sets. We first aim to measure how well did our model learn from the target distributions: distribution of abc parameters, distribution of $\alpha\beta\gamma$ parameters, and that of the coordinates. Table 4.1 below reports the results for the 2-Sample Kolmogorov Smirnov test, coefficient of overlap, and the Jensen-Shannon distance measures.

To obtain the distributions of the generated structures, an average of 9 structures were generated for each studied formula. The 2-Sample Kolmogorov-Smirnov test was first conducted with null-hypothesis that the distribution of existing structures was identical to the distribution of the generated structures. We used the alternative that they were different. In all 3 tests, the p-value was smaller than 10^{-30}, thus we rejected that the hypothesis that the generated and existing structure distributions were the same. This

conclusion supports our initially stated objective for the generative network to model the targeted distribution meanwhile not overfitting. Notably, we want the model-generated distribution to be very similar to the existing structures distribution, but not the same.

Table 5.1 Results comparing distributions of generated structures and real structures. The columns list the computed values for the 2-Sample Kolmogorov Smirnov test, coefficient of overlap, and the Jensen-Shannon divergence distance. A higher OVL is preferred and lower JS-dist. is better.

	2KS p-value	**OVL** $^{\uparrow}$	**JS-dist.** $^{\downarrow}$
abc	0.000	0.681	0.169
$\alpha\beta\gamma$	0.000	0.327	0.237
coordinates	0.000	0.595	0.128

We measure the similarity between the model-generated and the existing structures distributions by calculating the coefficient of overlapping (OVL) [125] and the Jensen-Shannon divergence distance [126]. Our model-generated distribution is very similar to the existing structures distribution for the abc parameters and the coordinates. The $\alpha\beta\gamma$ parameters distributions are not very similar. The OVL of the abc distributions comparison is 0.681 while the coordinates' is 0.595, which suggests that the generated and existing structures distributions are indeed close. By the same token, the JS-distance values are both low and indicate that the model-generated and existing structures are similar. Figure 4.5 plots histograms of the two types of distributions for the abc, $\alpha\beta\gamma$, and the coordinates which also support our previous conclusion.

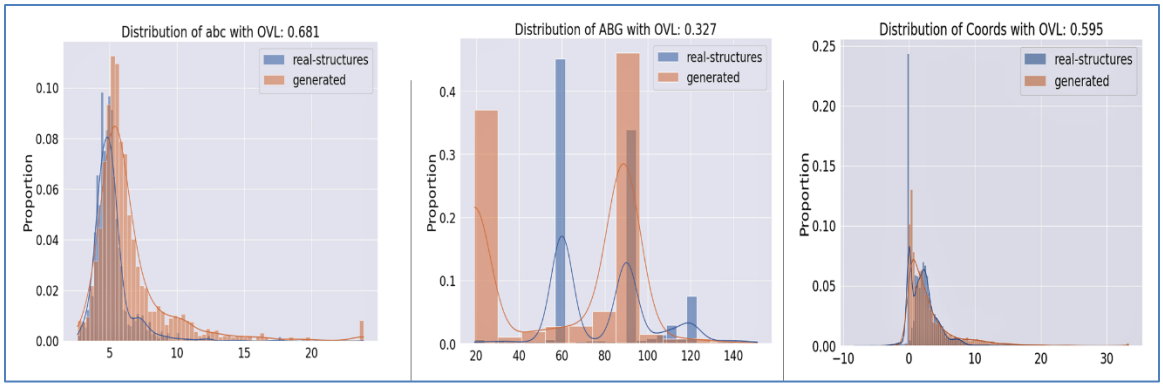

Figure 5.5 Illustration of the distributions between the model-generated distributions and the existing structures distribution. The coordinates and abc parameters are very close, but the $\alpha\beta\gamma$ are not very similar.

Table 5.2 List of 3 tests carried to evaluate the model's ability at generating valid structures. 45% of our generated structures have low formation-energy, 56% of the structures have volumes that fall within the observed volume limits of the dataset, and about 2% of the generated matched structures from the dataset.

	Proportion
$eform < 0.0$	45.541%
$25^* \leq volume \leq 383^*$	56.288%
Reconstructed structures	1.745%

We conducted additional measurements to evaluate the model's ability to generate valid structures. The tests included predicting the formation-energy of the generated structures, comparing the measured volumes to the observed measurements from our dataset, and evaluating the model's performance at reconstructing the structures seen from the dataset. Table 4.2 below reports the results from the prior tests. About 45% of our generated structures have low formation-energy, 56% of the structures have volumes

that fall within the observed volume limits of the dataset, and about 2% of the generated matched structures from the dataset. Figure 4.6 also compares predictions of formation energy between our model-generated structures and their matched structures found in our dataset. As previously discussed, a little under half of our generated structures have low formation energy, which is one of the numerous factors indicating of a crystal structure validity and stability.

We also visually inspected the generated crystal structures by viewing their cif-files in the Vesta software [127]. Figure 5.7 presents an illustration of 6 visually interesting structures from formulas $Er\ He\ Se$, $Na\ Ca\ As$, $Li_2\ S$, $Rbi_2\ Se$, $Y\ Ni\ As$, and $Ti_2\ Co_4$. Some generated structures appear like ones that could have come from our dataset.

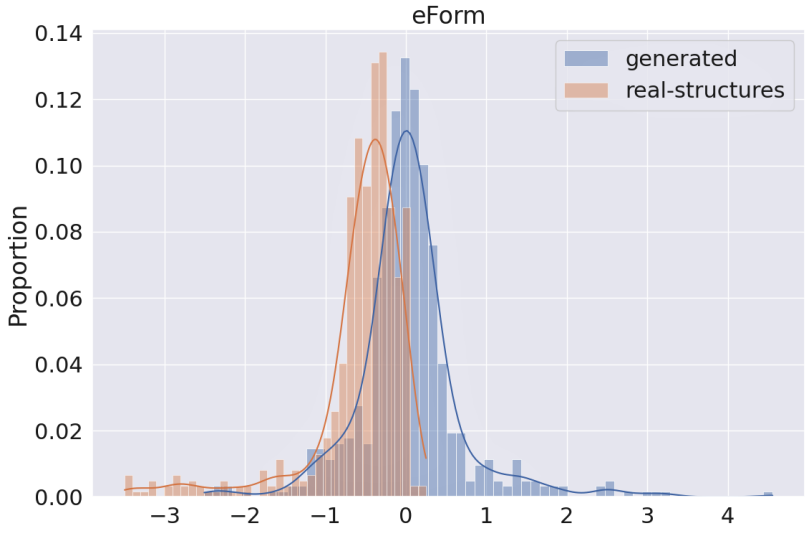

Figure 5.6 Illustration showing how generated structures' formation energy compares to the dataset's structures.

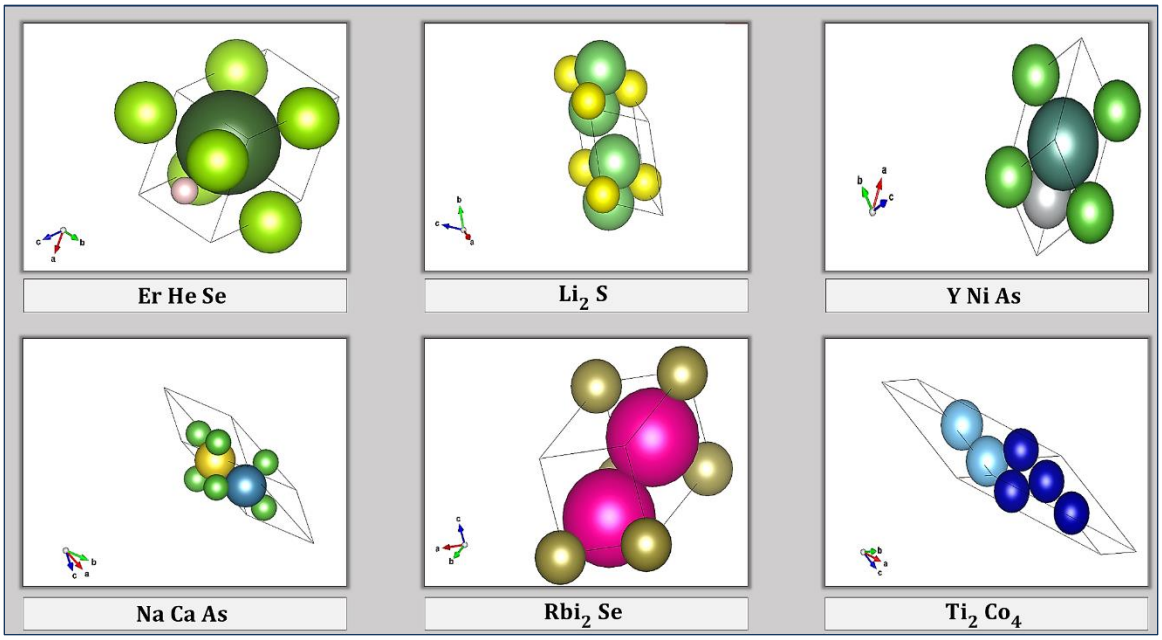

Figure 5.7 Visualization of 6 generated structures in the Vesta Software before DFT relaxation.

Even though our proposed approach of StructR-Diffusion can generate structures, it still faces a few limitations. Our approach's main drawbacks include a) many generated structures are too small (volume < 3 A^3), a bias to generate cubic structures, many generated structures have atoms that are very close to each other (distance < 0.3 A). Nonetheless, since our model generates structures by filtering noise information from the structures, our proposed StructR-Diffusion could endlessly improve a prior output. It's also important to note that our model can generate structures from formulas containing less than 3 or more atoms than 8 (numbers of atoms supported in our dataset).

4.5 CONCLUSION

We proposed StructR-Diffusion, a stable-diffusion generative model that uses graph convolutional neural network to conditionally generate crystal structures based on formulas. Various statistical tests, physical attribute predictions, and visual inspections show that our proposed graph convolutional network model has a good generative

capability. Even though our proposed approach has limitations of bias of cubic crystal systems, tendency to generate smaller (volume) structures, and frequent fault of placing atoms too close, StructR-Diffusion provides flexibility of use and endless ability to improve its generated output. Moving forward, a key sub-problem could be to improve the $\alpha\beta\gamma$ model and pair its independent optimization to the other two (abc and coordinates).

CHAPTER 6

CONCLUSION

In our dissertation proposal, we had presented our work on graph-based properties prediction, node classification of static graphs, and our future work on graph generation using a generative model based on VAE. In our first project, we applied the technique of Attention onto inorganic molecules to predict their physical properties. Our adaptation of the multi-head attention learns from the molecule atoms and connections meanwhile taking the neighboring atoms proximity (distance) in consideration. Beyond the local attention, we further adapted a global attention mechanism that captures the importance of the atoms to one another. We showed that the application of attention from both local and global perspective leads to more accurate properties prediction and interpretable results that are physically insightful. Our third project is an extension of this first project work. We adapted the same architecture and retrained a new model to predict electrodes voltages.

For our second project, we classified nodes of some graph-represented data objects. The list of these graph data includes co-authorship datasets, relational data on academic publication, and related items from the Amazon product categories graph. We classified the nodes of the graphs using our proposed method of NODE-SELECT that learns to aggregate information ONLY from relevant neighbors. We showed that this method is very efficient and highly effective. Compared to other widely known Graph Convolutional Networks, NODE-SELECT is highly resistant to noisy data nodes and their feature propagation.

For the future work, we had planned to continue developing an innovative generative VAE-based method that can create a structure when provided a formula and a space group. However, we switched direction and instead opted to use stable diffusion

method instead due to the lack of its implementation in the field of Materials Science. In our research, we developed a new ML approach called StructR-Diffusion that learns to filter noise information from noisy input structures. In our research, we successfully addressed our prior goals which were to a) implement a generative model that could implicitly learn real Physicochemical laws of molecules interaction, b) address prior limitations that our initial VAE faced, c) solve the flaws that prevent general GNN models from generating larger structures, and d) extend the model to also include a diffusion module. According to our knowledge, our work is the first adaptation of stable diffusion on crystal materials that operate a) directly on the raw forms of structure data, b) conditionally generate structures, and c) offers a high-level flexibility.

BIBLIOGRAPHY

1. Wu, Z., et al., *A comprehensive survey on graph neural networks.* IEEE transactions on neural networks and learning systems, 2020. **32**(1): p. 4-24.
2. Zhou, J., et al., *Graph neural networks: A review of methods and applications.* AI Open, 2020. **1**: p. 57-81.
3. Drielsma, F., et al., *Scalable, end-to-end, deep-learning-based data reconstruction chain for particle imaging detectors.* arXiv preprint arXiv:2102.01033, 2021.
4. Ju, X., et al., *Graph neural networks for particle reconstruction in high energy physics detectors*. arXiv preprint arXiv:2003.11603, 2020.
5. Zhang, Z., et al., *Graph neural network approaches for drug-target interactions.* Current Opinion in Structural Biology, 2022. **73**: p. 102327.
6. Marcheggiani, D. and I. Titov, *Encoding sentences with graph convolutional networks for semantic role labeling.* arXiv preprint arXiv:1703.04826, 2017.
7. Yang, C., et al. *Multisage: Empowering gcn with contextualized multi-embeddings on web-scale multipartite networks*. in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020.
8. Sato, R., *A survey on the expressive power of graph neural networks.* arXiv preprint arXiv:2003.04078, 2020.
9. Wu, L., et al., *Graph neural networks for natural language processing: A survey.* arXiv preprint arXiv:2106.06090, 2021.
10. Jabbar, A., X. Li, and B. Omar, *A survey on generative adversarial networks: Variants, applications, and training.* ACM Computing Surveys (CSUR), 2021. **54**(8): p. 1-49.
11. Zhou, J., et al., *Graph neural networks: A review of methods and applications.* arXiv preprint arXiv:1812. 08434, 2018.
12. Chen, C., et al., *Graph networks as a universal machine learning framework for molecules and crystals.* Chemistry of Materials, 2019. **31**(9): p. 3564-3572.
13. Xie, T. and J.C. Grossman, *Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties.* Physical review letters, 2018. **120**(14): p. 145301.
14. Veličković, P., et al., *Graph attention networks.* arXiv preprint arXiv:1710.10903, 2017.
15. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks,* in *Advances in neural information processing systems*. 2012. p. 1097-1105.
16. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks.* Communications of the ACM, 2017. **60**(6): p. 84-90.
17. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning.* nature, 2015. **521**(7553): p. 436-444.
18. Agrawal, A. and A. Choudhary, *Deep materials informatics: Applications of deep learning in materials science.* Mrs Communications, 2019. **9**(3): p. 779-792.
19. Chen, C., et al., *A critical review of machine learning of energy materials.* Advanced Energy Materials, 2020. **10**(8): p. 1903242.

20. Vasudevan, R.K., et al., *Materials science in the artificial intelligence age: high-throughput library generation, machine learning, and a pathway from correlations to the underpinning physics.* MRS communications, 2019. **9**(3): p. 821-838.
21. Mansouri Tehrani, A., et al., *Machine learning directed search for ultraincompressible, superhard materials.* Journal of the American Chemical Society, 2018. **140**(31): p. 9844-9853.
22. Li, S., et al., *Critical temperature prediction of superconductors based on atomic vectors and deep learning.* Symmetry, 2020. **12**(2): p. 262.
23. Cao, Z., et al., *Convolutional Neural Networks for Crystal Material Property Prediction Using Hybrid Orbital-Field Matrix and Magpie Descriptors.* Crystals, 2019. **9**(4): p. 191.
24. Ward, L., et al., *Matminer: An open source toolkit for materials data mining.* Computational Materials Science, 2018. **152**: p. 60-69.
25. Coley, C.W., et al., *A graph-convolutional neural network model for the prediction of chemical reactivity.* Chemical science, 2019. **10**(2): p. 370-377.
26. Kajita, S., et al., *A universal 3D voxel descriptor for solid-state material informatics with deep convolutional neural networks.* Scientific reports, 2017. **7**(1): p. 16991.
27. Schütt, K.T., et al., *Schnet–a deep learning architecture for molecules and materials.* The Journal of Chemical Physics, 2018. **148**(24): p. 241722.
28. Oganov, A.R., et al., *Structure prediction drives materials discovery*. Nature Reviews Materials, 2019. **4**(5): p. 331-348.
29. Smith, J.S., O. Isayev, and A.E. Roitberg, *ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost.* Chemical science, 2017. **8**(4): p. 3192-3203.
30. Schmidt, J., et al., *Recent advances and applications of machine learning in solid-state materials science.* npj Computational Materials, 2019. **5**(1): p. 1-36.
31. Park, C.W. and C. Wolverton, *Developing an improved Crystal Graph Convolutional Neural Network framework for accelerated materials discovery.* arXiv preprint arXiv:1906. 05267, 2019.
32. Wu, Z., et al., *A comprehensive survey on graph neural networks.* arXiv preprint arXiv:1901. 00596, 2019.
33. Zhang, H., et al., *Self-attention generative adversarial networks.* arXiv preprint arXiv:1805. 08318, 2018.
34. Devlin, J., et al., *Bert: Pre-training of deep bidirectional transformers for language understanding.* arXiv preprint arXiv:1810.04805, 2018.
35. Jin, J., et al., *Attention mechanism-based deep learning pan-specific model for interpretable MHC-I peptide binding prediction.* bioRxiv, 2019: p. 830737.
36. Liu, Z., et al., *DeepSeqPanII: an interpretable recurrent neural network model with attention mechanism for peptide-HLA class II binding prediction.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2021.
37. Shazeer, N., et al., *Talking-heads attention.* arXiv preprint arXiv:2003.02436, 2020.
38. Paszke, A., et al., *Automatic differentiation in pytorch.* NIPS-W, 2017.
39. Fey, M. and J.E. Lenssen, *Fast Graph Representation Learning with PyTorch Geometric*, in *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.
40. Ward, L., et al., *A general-purpose machine learning framework for predicting properties of inorganic materials.* npj Comput. Mater., 2016. **2**: p. 16028.
41. Kipf, T.N. and M. Welling, *Semi-supervised classification with graph convolutional networks.* arXiv preprint arXiv:1609.02907, 2016.
42. Hamilton, W., Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, in *Advances in neural information processing systems*. 2017. p. 1024-1034.
43. Li, G., et al., *Deepgcns: Can gcns go as deep as cnns?*, in *Proceedings of the IEEE International Conference on Computer Vision*. 2019. p. 9267-9276.

44. Zhao, L. and L. Akoglu, *PairNorm: Tackling Oversmoothing in GNNs*, in *International Conference on Learning Representations*. 2019.
45. Fey, M., *Just jump: Dynamic neighborhood aggregation in graph neural networks.* arXiv preprint arXiv:1904.04849, 2019.
46. Rong, Y., et al., *Dropedge: Towards deep graph convolutional networks on node classification*, in *International Conference on Learning Representations*. 2019.
47. Chen, D., et al., *Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View*.
48. Li, Q., Z. Han, and X.-M. Wu, *Deeper insights into graph convolutional networks for semi-supervised learning*, in *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
49. Liu, M., H. Gao, and S. Ji, *Towards deeper graph neural networks*, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020. p. 338-348.
50. Louis, S.-Y., et al., *Graph convolutional neural networks with global attention for improved materials property prediction.* Physical Chemistry Chemical Physics, 2020. **22**(32): p. 18141-18148.
51. Zhang, J., et al., *Gaan: Gated attention networks for learning on large and spatiotemporal graphs.* arXiv preprint arXiv:1803. 07294, 2018.
52. Feily, M., A. Shahrestani, and S. Ramadass, *A survey of botnet and botnet detection*, in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. 2009. p. 268-273.
53. Leo, F.M., et al., *How many leaders does it take to lead a sports team? The relationship between the number of leaders and the effectiveness of professional sports teams.* PloS one, 2019. **14**(6): p. e0218167.
54. Rese, A., H.-G. Gemünden, and D. Baier, *'Too Many Cooks Spoil The Broth': Key Persons and their Roles in Inter-Organizational Innovations.* Creativity and Innovation Management, 2013. **22**(4): p. 390-407.
55. Shchur, O., et al., *Pitfalls of graph neural network evaluation.* arXiv preprint arXiv:1811. 05868, 2018.
56. Bojchevski, A. and S. Günnemann, *Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking.* arXiv preprint arXiv:1707. 03815, 2017.
57. Sen, P., et al., *Collective classification in network data.* AI magazine, 2008. **29**(3): p. 93-93.
58. Cho, K., et al., *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* arXiv preprint arXiv:1406. 1078, 2014.
59. Li, Y., et al., *Gated graph sequence neural networks.* arXiv preprint arXiv:1511. 05493, 2015.
60. Defferrard, M., X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering*, in *Advances in neural information processing systems*. 2016. p. 3844-3852.
61. Xu, K., et al., *Representation learning on graphs with jumping knowledge networks.* arXiv preprint arXiv:1806. 03536, 2018.
62. Groysberg, B., J.T. Polzer, and H.A. Elfenbein, *Too many cooks spoil the broth: How high-status individuals decrease group effectiveness.* Organization Science, 2011. **22**(3): p. 722-737.
63. Havenith, M.N., et al., *Synchrony makes neurons fire in sequence, and stimulus properties determine who is ahead.* Journal of neuroscience, 2011. **31**(23): p. 8570-8584.
64. Sasaki, T., N. Matsuki, and Y. Ikegaya, *Interneuron firing precedes sequential activation of neuronal ensembles in hippocampal slices.* European Journal of Neuroscience, 2014. **39**(12): p. 2027-2036.

65. Tiezzi, M., et al., *Deep Constraint-based Propagation in Graph Neural Networks.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.
66. Fox, J. and S. Rajamanickam, *How Robust Are Graph Neural Networks to Structural Noise?* arXiv preprint arXiv:1912. 10206, 2019.
67. Oymak, S., *Learning compact neural networks with regularization*, in *International Conference on Machine Learning*. 2018. p. 3966-3975.
68. Jan, Z.M., B. Verma, and S. Fletcher, *Optimizing clustering to promote data diversity when generating an ensemble classifier*, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2018. p. 1402-1409.
69. Dong, L., et al., *Improving graph neural network via complex-network-based anchor structure.* Knowledge-Based Systems, 2021. **233**: p. 107528.
70. Dai, H., et al., *Learning steady-states of iterative algorithms over graphs*, in *International conference on machine learning*. 2018. p. 1106-1114.
71. LeClair, A., et al., *Improved code summarization via a graph neural network*, in *Proceedings of the 28th International Conference on Program Comprehension*. 2020. p. 184-195.
72. Zeng, A., et al., *Hop-Aware Dimension Optimization for Graph Neural Networks.* arXiv preprint arXiv:2105. 14490, 2021.
73. Chen, J., T. Ma, and C. Xiao, *Fastgcn: fast learning with graph convolutional networks via importance sampling.* arXiv preprint arXiv:1801. 10247, 2018.
74. Grover, A. and J. Leskovec, *node2vec: Scalable feature learning for networks*, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016. p. 855-864.
75. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412. 6980, 2014.
76. Srivastava, N., et al., *Dropout: a simple way to prevent neural networks from overfitting.* The journal of machine learning research, 2014. **15**(1): p. 1929-1958.
77. Fey, M. and J.E. Lenssen, *Fast graph representation learning with PyTorch Geometric.* arXiv preprint arXiv:1903.02428, 2019.
78. Dietterich, T.G., *Ensemble methods in machine learning*, in *International workshop on multiple classifier systems*. 2000. p. 1-15.
79. Guo, H., et al., *Accelerated atomistic modeling of solid-state battery materials with machine learning.* Frontiers in Energy Research, 2021. **9**: p. 695902.
80. Li, M., et al., *30 years of lithium-ion batteries.* Advanced Materials, 2018. **30**(33): p. 1800561.
81. Goel, S., R. Sharma, and A.K. Rathore, *A review on barrier and challenges of electric vehicle in India and vehicle to grid optimisation.* Transportation engineering, 2021. **4**: p. 100057.
82. Deng, J., et al., *Electric vehicles batteries: requirements and challenges.* Joule, 2020. **4**(3): p. 511-515.
83. Jain, A., Y. Shin, and K.A. Persson, *Computational predictions of energy materials using density functional theory.* Nature Reviews Materials, 2016. **1**(1): p. 1-13.
84. Gandomi, Y.A., et al., *Critical review—experimental diagnostics and material characterization techniques used on redox flow batteries.* Journal of The Electrochemical Society, 2018. **165**(5): p. A970.
85. Allam, O., et al., *Molecular structure–redox potential relationship for organic electrode materials: density functional theory–Machine learning approach.* Materials Today Energy, 2020. **17**: p. 100482.
86. Artrith, N., A. Urban, and G. Ceder, *Constructing first-principles phase diagrams of amorphous Li x Si using machine-learning-assisted sampling with an evolutionary algorithm.* The Journal of chemical physics, 2018. **148**(24): p. 241711.

87. Cubuk, E.D., A.D. Sendek, and E.J. Reed, *Screening billions of candidates for solid lithium-ion conductors: A transfer learning approach for small data.* The Journal of chemical physics, 2019. **150**(21): p. 214701.
88. Cunha, R.P., et al., *Artificial intelligence investigation of NMC cathode manufacturing parameters interdependencies.* Batteries & Supercaps, 2020. **3**(1): p. 60-67.
89. Dixit, M.B., et al., *Synchrotron imaging of pore formation in Li metal solid-state batteries aided by machine learning.* ACS Applied Energy Materials, 2020. **3**(10): p. 9534-9542.
90. Houchins, G. and V. Viswanathan, *An accurate machine-learning calculator for optimization of Li-ion battery cathodes.* The Journal of Chemical Physics, 2020. **153**(5): p. 054124.
91. Joshi, R.P., et al., *Machine learning the voltage of electrode materials in metal-ion batteries.* ACS applied materials & interfaces, 2019. **11**(20): p. 18494-18503.
92. Moses, I.A., et al., *Machine learning screening of metal-ion battery electrode materials.* ACS Applied Materials & Interfaces, 2021. **13**(45): p. 53355-53362.
93. Natarajan, A.R. and A. Van der Ven, *Machine-learning the configurational energy of multicomponent crystalline solids.* npj Computational Materials, 2018. **4**(1): p. 56.
94. Shandiz, M.A. and R. Gauvin, *Application of machine learning methods for the prediction of crystal system of cathode materials in lithium-ion batteries.* Computational Materials Science, 2016. **117**: p. 270-278.
95. Ong, S.P., et al., *Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis.* Computational Materials Science, 2013. **68**: p. 314-319.
96. Zhou, F., et al., *First-principles prediction of redox potentials in transition-metal compounds with LDA+ U.* Physical Review B, 2004. **70**(23): p. 235121.
97. Jain, A., et al., *Commentary: The Materials Project: A materials genome approach to accelerating materials innovation.* APL materials, 2013. **1**(1): p. 011002.
98. Louis, S.-Y., et al., *Node-select: a graph neural network based on a selective propagation technique.* Neurocomputing, 2022. **494**: p. 396-408.
99. Rosen, A.S., et al., *Machine learning the quantum-chemical properties of metal–organic frameworks for accelerated materials discovery.* Matter, 2021. **4**(5): p. 1578-1597.
100. Omprakash, P., et al., *Graph representational learning for bandgap prediction in varied perovskite crystals.* Computational Materials Science, 2021. **196**: p. 110530.
101. Paszke, A., et al., *Automatic differentiation in pytorch.* 2017.
102. Girshick, R. *Fast r-cnn*. in *Proceedings of the IEEE international conference on computer vision*. 2015.
103. Gallarati, S., et al., *Reaction-based machine learning representations for predicting the enantioselectivity of organocatalysts.* Chemical Science, 2021. **12**(20): p. 6879-6889.
104. Coley, C.W., et al., *Prediction of organic reaction outcomes using machine learning.* ACS central science, 2017. **3**(5): p. 434-443.
105. Hirohara, M., et al., *Convolutional neural network based on SMILES representation of compounds for detecting chemical motif.* BMC bioinformatics, 2018. **19**: p. 83-94.
106. Ong, S.P., et al., *Voltage, stability and diffusion barrier differences between sodium-ion and lithium-ion intercalation materials.* Energy & Environmental Science, 2011. **4**(9): p. 3680-3688.
107. Barker, J., M. Saidi, and J. Swoyer, *A sodium-ion cell based on the fluorophosphate compound NaVPO4 F.* Electrochemical and Solid-State Letters, 2002. **6**(1): p. A1.
108. Zhao, J., et al., *A phase-transfer assisted solvo-thermal strategy for low-temperature synthesis of Na 3 (VO 1− x PO 4) 2 F 1+ 2x cathodes for sodium-ion batteries.* Chemical Communications, 2015. **51**(33): p. 7160-7163.
109. Fang, Y., et al., *Recent advances in sodium-ion battery materials.* Electrochemical Energy Reviews, 2018. **1**: p. 294-323.

110. Xie, T., et al., *Crystal diffusion variational autoencoder for periodic material generation.* arXiv preprint arXiv:2110.06197, 2021.
111. Zhao, Y., et al., *High-throughput discovery of novel cubic crystal materials using deep generative neural networks.* Advanced Science, 2021. **8**(20): p. 2100566.
112. Zhao, Y., et al., *Physics guided deep learning for generative design of crystal materials with symmetry constraints.* npj Computational Materials, 2023. **9**(1): p. 38.
113. Butler, K.T., et al., *Machine learning for molecular and materials science.* Nature, 2018. **559**(7715): p. 547-555.
114. Guo, X. and L. Zhao, *A systematic survey on deep generative models for graph generation.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.
115. Omee, S.S., et al., *Scalable deeper graph neural networks for high-performance materials property prediction.* Patterns, 2022. **3**(5): p. 100491.
116. Croitoru, F.-A., et al., *Diffusion models in vision: A survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.
117. Fuhr, A.S. and B.G. Sumpter, *Deep generative models for materials discovery and machine learning-accelerated innovation.* Frontiers in Materials, 2022: p. 182.
118. Schwalbe-Koda, D. and R. Gómez-Bombarelli, *Generative models for automatic chemical design.* Machine Learning Meets Quantum Physics, 2020: p. 445-467.
119. Hoogeboom, E., et al. *Equivariant diffusion for molecule generation in 3d*. in *International Conference on Machine Learning*. 2022. PMLR.
120. Jing, B., et al., *Torsional diffusion for molecular conformer generation.* arXiv preprint arXiv:2206.01729, 2022.
121. Goodall, R.E. and A.A. Lee, *Predicting materials properties without crystal structure: Deep representation learning from stoichiometry.* Nature communications, 2020. **11**(1): p. 6280.
122. Dufter, P., M. Schmitt, and H. Schütze, *Position information in transformers: An overview.* Computational Linguistics, 2022. **48**(3): p. 733-763.
123. Zhou, K., et al., *Towards deeper graph neural networks with differentiable group normalization.* Advances in neural information processing systems, 2020. **33**: p. 4917-4928.
124. Shi, Y., et al., *Masked label prediction: Unified message passing model for semi-supervised classification.* arXiv preprint arXiv:2009.03509, 2020.
125. Franco-Pereira, A.M., et al., *Inference on the overlap coefficient: The binormal approach and alternatives.* Statistical methods in medical research, 2021. **30**(12): p. 2672-2684.
126. Nielsen, F., *On the Jensen–Shannon symmetrization of distances relying on abstract means.* Entropy, 2019. **21**(5): p. 485.
127. Momma, K. and F. Izumi, *VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data.* Journal of applied crystallography, 2011. **44**(6): p. 1272-1276.