Polymorphic Modelling the Potential Energy Surface of Carbon Systems using Graph Neural Networks

Gina Chun

May 5, 2022

Abstract

We present a machine learning method to model the potential energy surface of carbon structures with the following properties: polymorphic and physically-plausible. The model is polymorphic in that it models carbon structures regardless of the number of carbon atoms in the molecule. The model is physically-plausible because it is not dependent on the specific Cartesian coordinates of every single atom, making it invariant to translation, rotation, and permutation.

The design of this method starts with graph representation for every molecule with the edges holding the distances between an atom's nearest neighbors so that only the local environment is captured. It also uses a message passing graph neural network (MPGNN) which allows nodes to learn the data from neighboring nodes. This approach comes from how molecules are composed of structural units that are approximately constant, thus representing a molecule as a network of local "neighborhoods" produces local features that can be aggregated into producing higher-level features of the entire molecule.

The results show that the MPGNN is indeed learning the potential energy surface and is consistently improving as the number of message passing increases. A comparison between the MPGNN and a dummy regressor and an untrained MPGNN proves that the model is functioning as it should and is properly learning, but a comparison with the QUIP software shows that the MPGNN model still has room for improvement. The present work demonstrates a less costly approach to model the potential energy surface of carbon structures with a MPGNN, considering that current frameworks still have a high computational cost. The findings also show the potential for further improvement on the model and better future performance through even more fine tuning.

Contents

1 Introduction **4**

1.1 Why Carbon is Important 4

1.2 Why Use Machine Learning 4

1.3 Using Machine Learning on Carbon Molecules 5

2 Background **6**

2.1 Chemistry 6

2.1.1 Geometry and Structure 6

2.1.2 Quantum Mechanics 7

2.1.3 QM Software 9

2.2 Machine Learning 11

2.2.1 Graph Neural Networks 11

3 Method **13**

3.0.1 Model 13

4 Experimental Evaluation **15**

4.0.1 Training Dataset 15

4.0.2 Test Dataset 17

4.0.3 Results 18

4.0.4 Suggestions 21

1 Introduction

1.1 Why Carbon is Important

Carbon is one of the most important and widely investigated elements. Carbon is all around us in a variety of forms and is an area of study which has huge significance across many scientific fields and applications. The study of carbon would help the understanding of fossil fuels, aid the challenges of energy-related issues, and work towards solutions for climate change. Carbon materials are also intertwined in modern industries and technologies such as in steel, clothing, synthetic materials, and even in the silicon microchips in smartphones. A better understanding of carbon will lead to advancement in technology and more innovation. This can in turn lead to general increase in the quality of living, not just in the US but also in other countries as well, which is important for people and communities in need for more easily accessible technology. Not only will these technologies be cheaper, their advancement will also make them more sustainable, eco-friendly, and ethical as well.

To better understand carbon, studying the energy of different carbon molecules is very important. Knowing the potential energy of a carbon structure will allow us to derive other important properties of that carbon molecule. Being able to easily predict the potential energy can help unlock much more information on carbon, but is currently difficult because of all the different configurations carbon can be in. Carbon has such vast structural diversity which makes it particularly arduous to study but also that more important to study as well.

1.2 Why Use Machine Learning

As carbon is an important element, its versatility makes it a very complex subject of study. Carbon manifests in various physical forms, called allotropes, each with very different properties. Because of this, studying carbon can be difficult and very expensive with

conventional methods. Machine learning can be much more cost and time efficient than traditional scientific methods of experimentation. It also opens up a new space of methods for experiments that scientists cannot physically replicate with traditional techniques. For example, machine learning can be used for experiments in very small spatial scales or for large scale simulations all while maintaining high performance accuracy. It also allows researchers to take advantage of similarities between certain physical systems and learning systems, such as neural networks, to be able to model these systems and to also just learn more about areas that are difficult to model otherwise such as quantum mechanics. Current theories and equations in quantum mechanics can calculate energy for a singular atom but it becomes increasingly difficult for complex molecules. Thus the application of machine learning and neural networks can greatly help the complex and costly study of carbon with all of its various geometries.

1.3 Using Machine Learning on Carbon Molecules

Currently, there are machine learning models that do predict the potential energy of carbon molecules. However, they are still very computationally expensive at an $O(N^3)$ and require a large, comprehensive amount of data. We propose a lower cost approach to learning the potential energy surface of carbon molecules with it being both polymorphic and physically-plausible. This means the model for the potential energy surface does not depend on the number of carbon atoms or the specific Cartesian coordinates of those atoms. Our method is to represent carbon molecules as graphs and have a graph neural network with message passing learn and predict on this graph-represented data. Our results show that this neural network is able to learn the potential energy of carbon structures and consistently improves as the number of message passing increases and other parameters are adjusted. Our model still needs improvement to match current chemistry softwares that predict potential energy but our experiments show the potential for even more promising performance and ways to improve our model for future research.

2 Background

2.1 Chemistry

Using machine learning to study any kind of material, not just carbon, is just one method under a wider category of techniques called material modeling. Material modeling in chemistry is the method of using computer simulations to predict physical and chemical properties of a material in replace of real-world experiments. This process of predicting the properties of a given material is called a *forward problem*. Through existing tools such as statistical mechanics and the Schrödinger equation, which will be talked about later, an exact unique solution exists for the forward problem of predicting material properties and is the reason why many opt to use this procedure. Material modeling also uses this approach because the inverse problem, constructing a material through a set of pre-determined properties, is much more difficult and costly to do.

2.1.1 Geometry and Structure

"Structure determines properties" is a very important and central concept in chemistry. This structure-property relationship explains that since all materials are made of atoms, identifying and describing a material comes down to simply understanding its atomic structure. For studying structures, there is never a need to look at the entire material as well. That would be too many atoms to study at once and atoms of most solid materials, like crystalline carbon, are arranged in an organized repetitive pattern so it is sufficient enough to look at just a patch of that pattern. Studying the properties of a microscopic structure, in general, translates well to its macroscopic structure which allows it to pair well with the material modeling approach.

An atom can be described as having a heavy nucleus surrounded by a cloud of light electrons and since the mass of an atom is concentrated on its nucleus, the atom's position is associated with the position of the nucleus. Thus moving atoms can be described as

moving nuclei being followed by their clouds of electrons.

The Born–Oppenheimer approximation is the mathematical formulation of this concept and is the foundation of material modeling. Molecular and material properties can be roughly divided into two categories: electronic and thermodynamic properties. Electronic properties are essentially the properties of that electron cloud that is following the nucleus and is best described through quantum mechanics, which will be covered in more detail later. To give a general overview, the electronic cloud can be in certain states such as its lowest-energy state called the *ground state*. Some ground state properties include the atomization energy, which is the energy released when a molecule is formed, and the dipole moment and polarizability, which describes the shape and responsiveness of the cloud. Since thermodynamic properties can be more easily derived through knowing the electronic properties first, we will be predicting ground-state electronic properties so that is what the background will be focused on.

2.1.2 Quantum Mechanics

Quantum mechanics is the set of laws that describe objects on an atomic level. One of the core principles is that an object can be in multiple states. An object like an electron can be in any combination of these two states. Mathematically, this can be expressed as a linear combination of two basis vectors that act as the two position states. The state of this object can be expressed as a *wave function*, which generalizes the two position vectors to the infinite number of positions in the three-dimensional space. This fundamental idea is why electrons are commonly referred to as clouds.

The laws of quantum mechanics also states that each observable, quantifiable, physical property that an atomic object has is associated with a linear operator. The actual value of the quantity is given by the eignevalues of the operator. One of the most important operators is the energy operator, called the Hamiltonian, and its associated eigenvalue equation is called a Schrödinger equation. The energy of a particular eigenstate as a

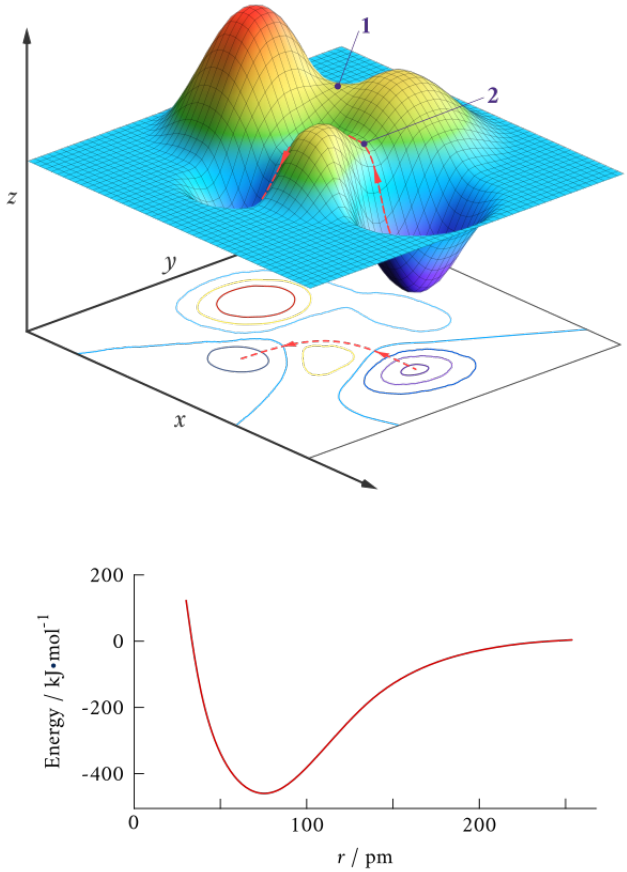

Figure 1: Example of a 3-D and 2-D PES. The top graph shows the z-axis representing the potential energy, red being the highest and violet being the lowest potential energy. The lower graph has the distance between two atoms as the x-axis and potential energy on the y-axis. Image source from LibreTexts [8]

function of the nuclear positions is called a *potential energy surface* and it completely determines the dynamics of the motion of the atoms. The potential energy surface is shown visually in figure 1. The solution to this equation allows many electronic properties of both ground and excited states to be determined. The Schrödinger equation can be mathematically solved for singular atoms but it gets more difficult for more complex molecules. There are methods such as the density functional theory and the Hartree-Fock that try to solve this issue by manipulating the Hamiltonian or the vector space itself, but their computational costs make these methods practically inefficient. Once the Schrödinger equation is solved, evaluating properties such as atomization energy or dipole moment can be done through the eigenvalues and eigenstates.

2.1.3 QM Software

Currently, there are programs used in computational chemistry to implement the methods mentioned above to calculate properties. One software, called Vienna Ab initio Simulation Package or VASP, performs ab initio quantum mechnical calculations and uses DFT as its base methodology. Another software package, called QUantum Mechanics and Interatomic Potentials or QUIP [1], has a collection of tools to conduct molecular dynamics simulations and works with other external packages such as LAMMPS or ASE. An example run may look like something in Figure 2 where the QUIP command is executed on the command line using a referenced XYZ file. The XYZ file must in a format where the first line is the number of atoms and the second line is a series of key and value pairs which must have the Lattice key and Properties key. The typical output of a QUIP command can also be seen in Figure 2 where package information is printed out first and then the energy that QUIP predicts is returned. We will be using the QUIP software as reference calculations to experimental results. The Carbon GAP 20 potential [4] will be used as the parameters in the QUIP calculations.

A)

```
quip atoms_filename=example.xyz param_filename=../../Carbon_GAP_20/
Carbon_GAP_20_potential/Carbon_GAP_20.xml E
```

B)

```
64
Lattice="7.10842262195238 0 0 0 7.10842262195238 0 0 0 7.10842262195238 " PBC="1 1 1" Properties=species:S:1:p
os:R:3:force:R:3 Energy="-515.36256422"
C     0.000000000000000e+00   0.000000000000000e+00   0.000000000000000e+00 -1.16500000e-05  -5.03000000e-06
 2.76000000e-06
C     1.777105655000000e+00   1.777105655000000e+00   0.000000000000000e+00  4.13600000e-05   3.56500000e-05
 2.78000000e-06
C     0.000000000000000e+00   1.777105655000000e+00   1.777105655000000e+00 -5.47300000e-05   4.65100000e-05
 8.11300000e-05
C     1.777105655000000e+00   0.000000000000000e+00   1.777105655000000e+00  9.90800000e-05  -8.46900000e-05
 8.27300000e-05
C     8.885528275000000e-01   8.885528275000000e-01   8.885528275000000e-01  9.00000000e-08  -5.15000000e-06
 9.20000000e-07
C     2.665658482500000e+00   2.665658482500000e+00   8.885528275000000e-01  1.67240000e-04   1.47770000e-04
 7.38000000e-06
C     8.885528275000000e-01   2.665658482500000e+00   2.665658482500000e+00 -4.75500000e-05   1.60520000e-04
 2.08880000e-04
C     2.665658482500000e+00   8.885528275000000e-01   2.665658482500000e+00  2.09440000e-04  -8.55000000e-05
 1.98740000e-04
C     3.554211310000000e+00   0.000000000000000e+00   0.000000000000000e+00  2.07300000e-04  -1.87340000e-04
-1.79940000e-04
C     5.331316965000000e+00   1.777105655000000e+00   0.000000000000000e+00 -4.10800000e-05   1.15550000e-04
-1.75760000e-04
C     3.554211310000000e+00   1.777105655000000e+00   1.777105655000000e+00  1.68510000e-04   1.10720000e-04
 1.66290000e-04
C     5.331316965000000e+00   0.000000000000000e+00   1.777105655000000e+00 -8.50000000e-06  -2.81910000e-04
 1.55010000e-04
```

C)

```
libAtoms::Hello World: 26/04/2022   07:14:30
libAtoms::Hello World: git version  https://github.com/libAtoms/QUIP,704e49f-dirty
libAtoms::Hello World: QUIP_ARCH    linux_x86_64_gfortran_openmp
libAtoms::Hello World: compiled on  Nov 18 2021 at 19:23:00
libAtoms::Hello World: OpenMP parallelisation with 24 threads
WARNING: libAtoms::Hello World: environment variable OMP_STACKSIZE not set explicitly. The default value - system and compiler depe
ndent - may be too small for some applications.
libAtoms::Hello World: Random Seed = 26070578
libAtoms::Hello World: global verbosity = 0

Calls to system_timer will do nothing by default

Using calc args:
Using pre-relax calc args:
Using param_filename: ../../Carbon_GAP_20/Carbon_GAP_20_potential/Carbon_GAP_20.xml
Using init args:
WARNING: Potential_initialise using default init_args "Potential xml_label=GAP_2020_4_27_60_2_50_5_436"
Potential_Sum:

Potential 1:
Potential containing potential
IP : 17
IPModel_Glue : Glue Potential
IPModel_Glue : n_types = 1 cutoff = 10.000000000000000
IPModel_Glue : type 1 atomic_num 6

Potential 2:
Potential containing potential
IP : 6
IPModel_GAP : Gaussian Approximation Potential
IPModel_GAP : label = GAP_2020_4_27_60_2_50_5_436
IPModel_GAP : cutoff = 4.5000000000000000
IPModel_GAP : E_scale = 1.0000000000000000
IPModel_GAP : command_line =  at_file=General_Carbon_V10_2_4000_All_GAP_17_Unique_LD_Iteration_1.xyz gap={distance_2b n_sparse=15 t
heta_uniform=1.0 sparse_method=uniform covariance_type=ard_se cutoff=4.5 delta=2.0:angle_3b n_sparse=200 theta_uniform=1.0 sparse_m
ethod=uniform covariance_type=ard_se cutoff=2.5 delta=0.05:soap n_max=12 l_max=4 atom_sigma=0.5 zeta=4.0 cutoff=4.5 cutoff_transiti
on_width=1.0 central_weight=1.0 n_sparse=9000 delta=0.2 covariance_type=dot_product sparse_method=cur_points radial_decay=-0.5} def
ault_sigma={0.001 0.01 0.05 0.0} energy_parameter_name=energy force_parameter_name=force virial_parameter_name=virial do_copy_at_fi
le=F sparse_jitter=1.0e-8 gp_file=Carbon_GAP_V10_2_LDIter1_grr6_2b_3b_soap_n12l4_sp9k_delta-2-0.05-0.2.xml core_ip_args={IP Glue} c
ore_param_file=r6_innercut.xml config_type_sigma={Liquid:0.050:0.5:0.5:0.0:Liquid_Interface:0.050:0.5:0.5:0.0:Amorphous_Bulk:0.005:
0.2:0.2:0.0:Amorphous_Surfaces:0.005:0.2:0.2:0.0:Surfaces:0.002:0.1:0.2:0.0:Dimer:0.002:0.1:0.2:0.0:Fullerenes:0.002:0.1:0.2:0.0:De
fects:0.001:0.01:0.05:0.0:Crystalline_Bulk:0.001:0.01:0.05:0.0:Nanotubes:0.001:0.01:0.05:0.0:Graphite:0.001:0.01:0.05:0.0:Diamond:0
.001:0.01:0.05:0.0:Graphene:0.001:0.01:0.05:0.0:Graphite_Layer_Sep:0.001:0.01:0.05:0.0:Single_Atom:0.0001:0.001:0.05:0.0}
IPModel_GAP : log likelihood =  -0.78945463690449493E+007 -0.81117104372598842E+008 -0.50553368261851919E+009

Energy=-505.92208003630520
Cell Volume: 359.18626474930375 A^3
AT 64
AT PBC="T T T" Energy=-505.92208003630520 cutoff=10.00000000 nneightol=1.20000000 Lattice="7.10842262       0.00000000       0.0000
0000       0.00000000       7.10842262       0.00000000       0.00000000       0.00000000       7.10842262" Properties=species:S:1:
pos:R:3:force:R:3:Z:I:1:map_shift:I:3:n_neighb:I:1
AT C              0.00000000      0.00000000      0.00000000     -0.00001165     -0.00000503      0.00000276       6       0
 0       0     884
AT C              1.77710565      1.77710565      0.00000000      0.00004136      0.00003565      0.00000278       6       0
```

Figure 2: A) is the QUIP command line. B) is the example.xyz file used in the A) command line with the required Lattice and Properties keyd. C) is the output of the QUIP command with the predicted energy highlighted.

2.2 Machine Learning

In a molecule, different pairs of atoms and bonds have different distances so it is easy to describe these molecules as a graph. The atoms are the nodes and the bonds are the undirected edges. In general, there are three types of prediction tasks that are done on graphs: graph-level, node-level, and edge-level. Graph-level approaches predict a single property for a whole graph while a node-level approach predicts a property for each node in the graph. Edge-level approaches predict the property or existence of edges in a particular graph. We will be focusing on the graph-level task of predicting the energy of carbon molecules.

Integrating these graphs into machine learning models will depend on how they are represented. An obvious representation method would be the adjacency matrix since it captures the graph's connectivity. However, the more nodes there are, the more space-inefficient the adjacency matrix becomes and in addition, the adjacency matrix of a molecule is not unique. There are many different ways to express the same connectivity with these matrices which causes the model to be permutation variant since it will see these matrices as different molecules when they are actually the same. A better memory-efficient representation is adjacency lists where each item in this list describes the connection between just two nodes. This cuts out a lot of the parts of the graph that are disconnected that would be needlessly recorded in an adjacency matrix.

2.2.1 Graph Neural Networks

A graph neural network (GNN) consists of transformations on all attributes of the given graph that preserves its symmetries and thus is permutation invariant. The simplest kind of GNN has each layer apply a multilayer perceptron (MLP) on each attribute of a graph which includes the nodes, edges, and global-context. Since predictions are made on the nodes of the graph, *pooling* is a way to gather information about the edges and give them to the nodes. This is done by gathering the embeddings of each item that

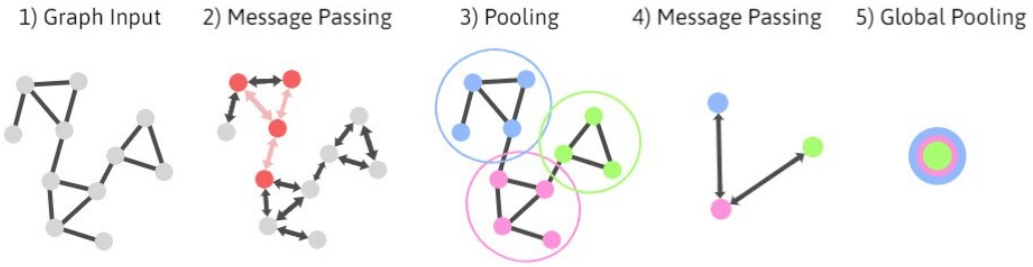

Figure 3: Simple example of message passing in a graph. Image is author's own image.

will be pooled into a matrix and then aggregate them, usually through a summation. More complex predictions can be made by using pooling in each of the GNN layers which lets information exchange influence updated embeddings. This allows the embeddings to be aware of the graph's connectivity and is called *message passing*. The forward pass of a message passing neural network (MPNN) consists of two steps: message passing and readout. In message passing, each node has a hidden state or feature vector and a function of neighboring hidden states are aggregated for each node. The hidden state of a node is updated using the obtained message and the previous hidden state of that node. Conceptually this allows each node to understand and use previous history to influence their future update. The readout step computes a feature vector for the whole graph.

A specific way to carry out message passing is by using graph convolutions. Just like in convolutional neural networks, the input is multiplied by a set of weights known as *kernels*. It acts like a filter that scans across the input graph and allows the model to learn features from neighboring nodes, as shown in figure 3. Convolutions collect and condense information about local environments within a graph to learn about its connectivity. Through this kernel, neighboring information can be aggregated with the information the node is already holding and thus performs message passing throughout the convolutions.

3 Method

Our goal is to model the potential energy surface of carbon molecules through machine learning. The challenge is to have the model not be limited to the number of atoms each molecule has and that it is invariant to translation, rotation, and permutation. Our approach takes advantage of the fact that a molecule is made up of consistent structural units to create a network made up of all the local "neighborhoods" of the molecule. Our graph representation allows the energy of the whole molecule to be found through local environment information, which will be described in more detail in section 4.0.1.

Several current research work [3–5] of similar domains tend to use a Gaussian Approximation Potential (GAP) to model molecules. The GAP framework allows training with first-principles calculations at a significantly reduced cost with similar predictive quality. Even though these GAP frameworks are transferable and show good performance, its downside lies in its $O(N^3)$ computational cost, even though it is better than other previous methods of molecular simulations. In addition, not all kinds of configurations can be represented by GAP so data that does not display a Gaussian uncertainty can be difficult to predict. We propose a message passing graph neural network as a more cost effective technique to emulate what GAPs are doing to model the potential energy surface of carbon structures.

3.0.1 Model

The model uses the python package DGL-LifeSci [2] which allows an easy integration of tasks in chemistry and graph neural networks. The implemented model is split into three parts: message passing, readout, and linear layers. The message passing graph neural network (MPNNGNN) from DGL-LifeSci first starts with a linear and ReLU layer. Then there is a cycle that consists of a graph convolution layer, another ReLU layer, and a multi-layer gated recurrent unit (GRU). This cycle is repeated for the number of messsage passing set in the beginning and through this repetition, node representations are being

updated and message passing is being performed. The second part of the model is a MLP-based readout where it updates node representations with a MLP and then computes graph representations out of those node features. The third part of the model is a simple combination of linear, dropout, ReLU, linear layers.

An important property of this model is that it is permutation invariant. This means that the output of the network does not depend on the order of the input. To put it another way, say that there is a singular water molecule with its atoms labeled "oxygen", "hydrogen 1", and "hydrogen 2". The two hydrogen atoms are not conceptually different in this molecule so it should not matter if "hydrogen 2" gets inputted before "hydrogen 1" in the model. This model is permutation invariant because nodes are communicating between each other and are learning from local graph information through message passing. Message passing allows the nodes not to be dependent on a particular input and its order.

4 Experimental Evaluation

4.0.1 Training Dataset

The data that is used comes from a database of configurations called Carbon Data Set Total [4]. For the model to be accurate and transferable, the database represents a reduced subset of all available configurations from the thermally accessible chemical space. The molecular properties were computed using the same level of tightly converged plane-wave DFT including dispersion corrections. The VASP plane-wave DFT code was used to perform spin-polarized calculations with the optB88-vdW dispersion inclusive exchange–correlation functional, a plane-wave cutoff of 600 eV, and a projector augmented wave pseudopotential. In addition, dense reciprocal space Monkhorst-Pack grids were used, along with a Gaussian smearing of 0.1 eV, applied to the energy levels. Some *ab initio* and iteratively improved GAP driven molecular dynamics simulations have been performed for all of these structures at a number of temperatures.

This comprehensive database contains approximately 17,000 configurations, which is too large to conduct training at a reasonable computational cost. A subset of this database, called Carbon GAP 20 Training set, was chosen to use in training through the farthest point sampling method. After that, data saturation of this sampling was checked to ensure that this new training dataset was as comprehensive as the full database. The training data contains 6,088 configurations of mostly crystalline carbon and some liquid and amorphous carbon with their respective properties. As shown in Figure 4, the full database is very comprehensive for all possible crystalline phases of carbon at moderate temperatures and pressures with the additional inclusion of more exotic allotropes.

This dataset of geometries is then turned into a graph dataset. For each atom in each molecule, four nearest neighbors are found and the distances are calculated to that atom. Once the the distances for the four nearest neighbors of each atom is found, it is used to create a DGL [6] graph that represents the whole molecule where the edge features of

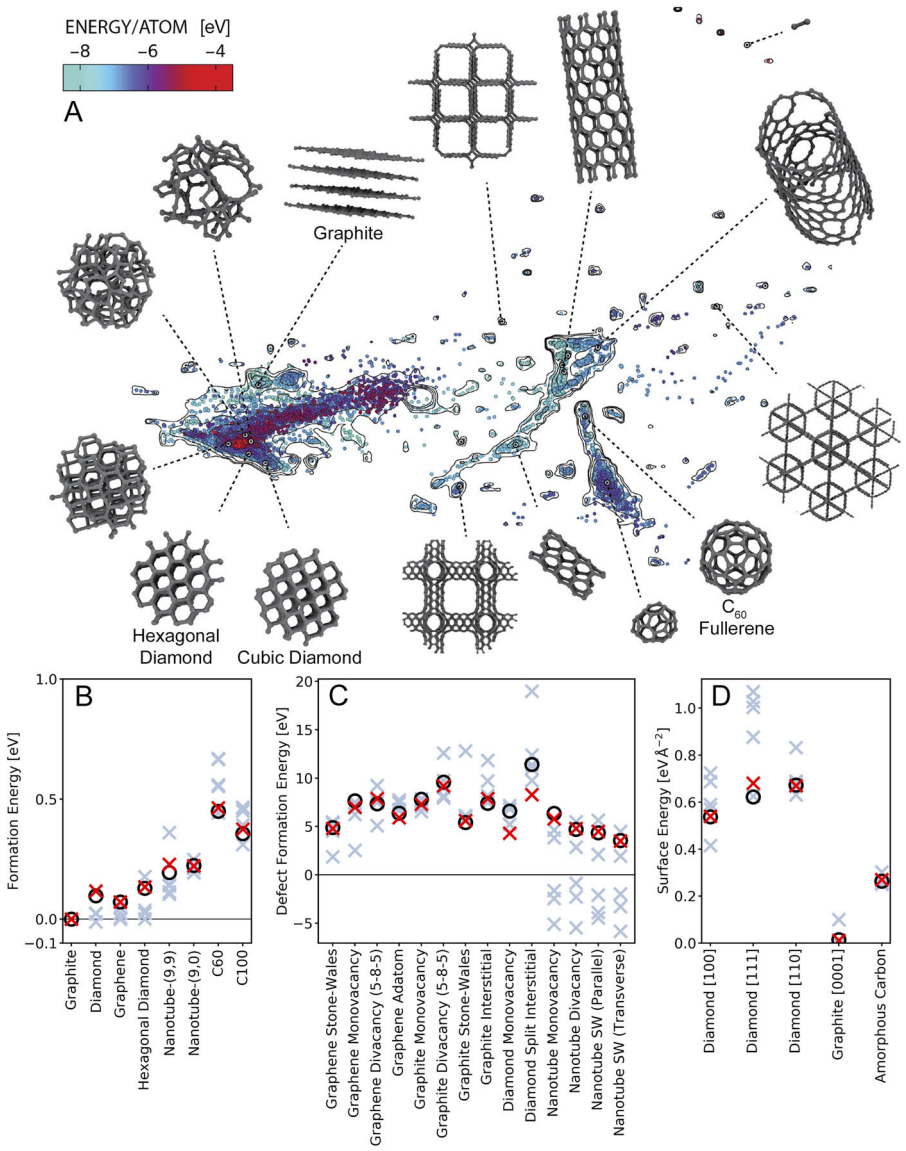

Figure 4: Map of the training data with some of the key structures (top). Summary of predicted formation energies, defect formation energies, and surface energies of crystalline carbon (bottom). Image source from GAP20 [4]

the graph contain the distance information. This method of representation uses the local environment of each atom to describe a whole molecule and determine its energy. This representation is both rotation and translation invariant. This is because the distance between these atoms do not change if rotated or translated, thus it does not depend on its specific Cartesian location.

4.0.2 Test Dataset

Two datasets are used to test the performance of the model. One dataset contains carbon allotropes of monolayer graphene, bilayer graphene, graphite, and diamond with their DFT energies calculated using VASP. For simplicity, this dataset will be called the DUNN dataset for it was originally used to develop a Dropout Uncertainty Neural Network potential [7]. The second test set comes from the source of the training set, Carbon Data Set Total. Since the training data is a subset, it was removed from the total dataset to create a Modified Total dataset to test on. The graph dataset versions of these test sets are created the same way the training graph dataset was made, referenced in section 4.0.1.

Three other measurements were made to additionally evaluate the model performance. The first measurement uses the dummy regressor from Scikit-learn to make predictions and calculate its mean squared error (MSE) on both the DUNN and Modified Total test set. The second measurement uses the untrained version of the model to directly calculate predictions and MSE for both test sets to see how the model performs without any learning. The third measurement uses the QUIP software and the GAP 20 potential [4] to calculate energies and MSE on both test sets once again. This measurement will allow the direct comparison of the cited research's results with our results.

Parameters				
Model	Learning Rate	Message Passing	Batch Size	Nearest Neighbor
1	0.0005	5	16	4
2	0.0005	7	32	4
3	0.0001	5	32	4
Untrained (DUNN)	0.0005	5		4
4	0.0005	5	16	4
5	0.0005	7	32	4
6	0.0001	5	32	4
Untrained (Modified Total)	0.0005	5		4

Table 1: Table of the differences in parameters between the top six models

4.0.3 Results

Experiments were started with 1 and 3 rounds of message passing with a batch size of 16, learning rate of 0.001, and 4 nearest neighbors. For those runs, the model was overfitting the training data because the train loss was gradually decrease while the test loss was unstable. We then just increased the number of message passing to 5 but those as well had unstable test losses even when the train losses were very stable. Because of this, we kept the number of message passing at 5 but lowered the learning rate and varied the batch size. Once both train and test losses seemed to gradually decrease, we also ran some experiments with the message passing set to 7. The specifications of the experiments done with message passing of 5 and 7 can be seen in Table 1 and the graphs of their MSE can be seen in Figure 5.

First, the results clearly show that the model is able to train. All the experiments produce an MSE lower than the untrained model for both test sets which shows that the model is working. All the runs also have an MSE lower than the dummy regressor baseline which proves it is learning better than a predictor with a simple strategy. The model is definitely able to learn the potential energy surface because its train and test losses are consistently improving and becoming more stable as the number of message passing

DUNN Dataset	
Model	MSE (ev)
QUIP	45.256
1	1,655.831
2	2,516.364
3	18,836.808
Dummy	25,620.847
Untrained	228,335.155

Table 2: Table of the MSE of the models done with the DUNN dataset.

Modified Total Dataset	
Model	MSE (ev)
QUIP	16.989
4	3,388.47
5	7,583.152
6	8,438.149
Dummy	191,405.849
Untrained	418,529.850

Table 3: Table of the MSE of the models done with the Modified Total dataset.

increases. The runs with the same number of message passing also improve compared to each other as other parameters are adjusted because the MSE is lowering at several magnitudes between each other as seen in Table 2 and Table 3.

Even though our test datasets come from different sources, the model seems to improve and learn those datasets in a similar way which can be seen by comparing Table 2, Table 3, and Table 1. This demonstrates the transferable, polymorphic, and physically-plausible properties of the model. However, the MSE of the QUIP software is still considerably lower than any of the experiments with the MPGNN model which shows that there is still room for improvement to the model. The decreasing losses of the MPGNN model experiments show the possibility for further improvement and better future performance through even more fine tuning of the model. Since the MPGNN model was able to learn the potential energy surface of carbon molecules, it can still be proposed as a much less costly approach to polymophic modelling compared to current GAP frameworks.

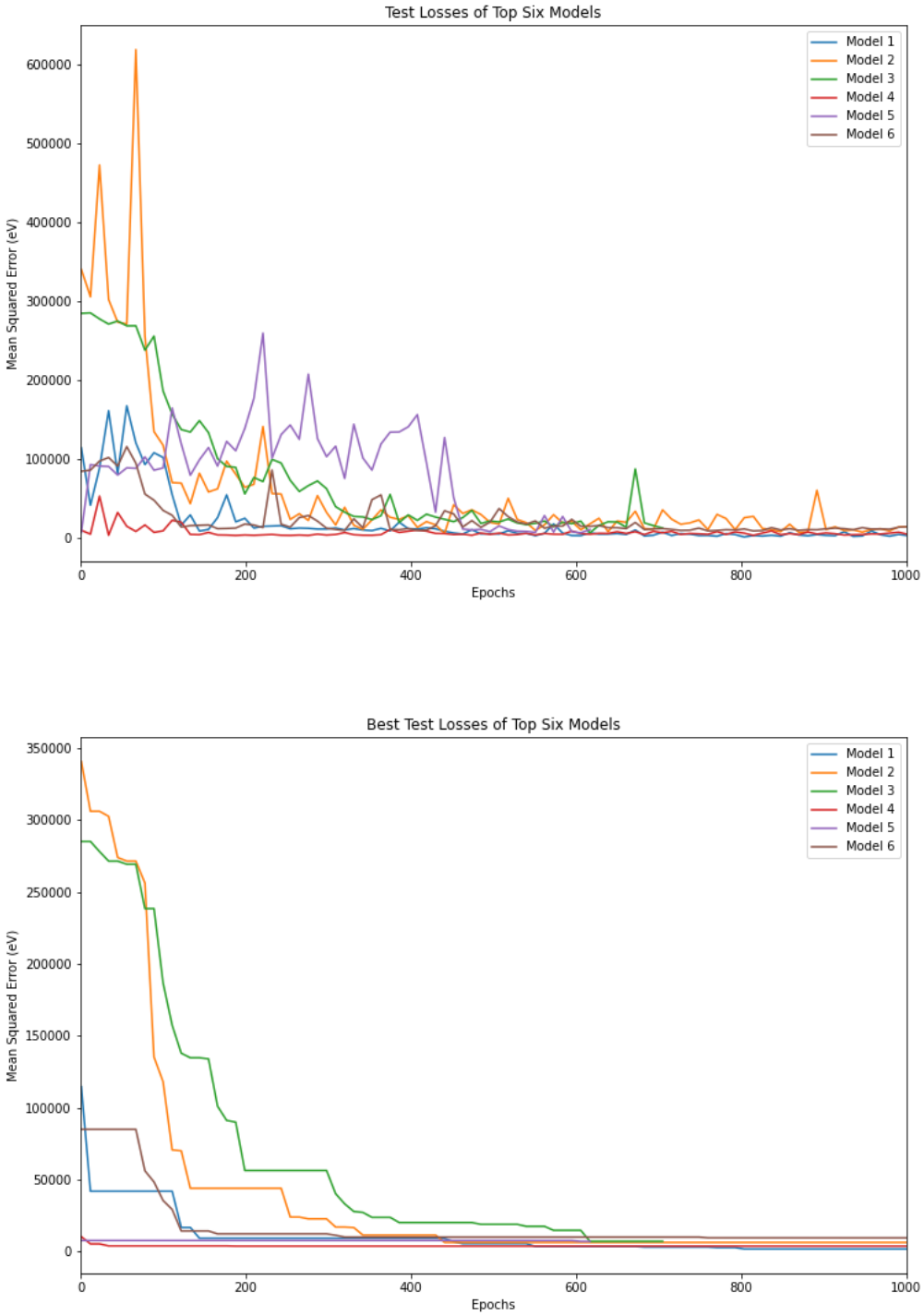

Figure 5: Both legends refer to the runs in Table 1. Upper graph shows the MSE test losses for each of the six models. Lower graph shows the best MSE losses for each of the six models.

4.0.4 Suggestions

Our findings open up the potential for more improvement in future work and experimentation. Adjustments to hyperparameters, trying different activation functions in the model, and using even higher numbers of message passing would be ways to further continue the study of this MPGNN model. Another suggestion could be to change the graph representation to a hypergraph representation where an edge can join any number of vertices unlike in a traditional graph where each edge connects exactly two vertices. If the ternary edges contained angle information between atoms, the model would have even more information, distances and angles of nearest neighbors, to learn the potential energy surface. The message passing portion of the model would be able to learn about surrounding nodes even better with the additional information they hold. The many possibilities for future research demonstrate the potential capability of the MPGNN model and further validate it as an inexpensive viable approach to modelling carbon structures.

References

[1] Gábor Csányi, Steven Winfield, J R Kermode, A De Vita, Alessio Comisso, Noam Bernstein, and Michael C Payne. Expressive programming for computational physics in fortran 95+. *IoP Comput. Phys. Newsletter*, page Spring 2007, 2007.

[2] Mufei Li, Jinjing Zhou, Jiajing Hu, Wenxuan Fan, Yangkang Zhang, Yaxin Gu, and George Karypis. Dgl-lifesci: An open-source toolkit for deep learning on graphs in life science. *ACS Omega*, 2021.

[3] Yuan-Bin Liu, Jia-Yue Yang, Gong-Ming Xin, Lin-Hua Liu, Gábor Csányi, and Bing-Yang Cao. Machine learning interatomic potential developed for molecular simulations on thermal properties of -ga2o3. *The Journal of Chemical Physics*, 153(14):144501, 2020.

[4] Patrick Rowe, Volker L. Deringer, Piero Gasparotto, Gábor Csányi, and Angelos Michaelides. An accurate and transferable machine learning potential for carbon. *The Journal of Chemical Physics*, 153(3):034702, 2020.

[5] Christoph Schran, Fabien Brieuc, and Dominik Marx. Transferability of machine learning potentials: Protonated water neural network potential applied to the protonated water hexamer. *The Journal of Chemical Physics*, 154(5):051101, 2021.

[6] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. 2020.

[7] Mingjian Wen and Ellad B. Tadmor. Uncertainty quantification in molecular simulations with dropout neural network potentials. *npj Computational Materials*, 2020.

[8] Ümit Kaya via LibreTexts. CC BY-NC.