Training Neural Networks to Accurately Determine Energies of Structures Outside of the Training Set Using Agglomerative Clustering

Carlos A. Barragan
Department of Chemistry and Biochemistry
California State University, Fullerton
Fullerton, CA
carlosb@csu.fullerton.edu

Michael N. Groves
Department of Chemistry and Biochemistry
California State University, Fullerton
Fullerton, CA
mgroves@fullerton.edu

ABSTRACT
Machine learning has accounted for solving a cascade of data in an efficient and timely manner including as an alternative molecular calculator to replace more expensive *ab initio* techniques. Neural networks (NN) are the most predictive for new cases that are similar to examples in their training sets; however, it is sometimes necessary for the NN to accurately evaluate structures not in its training set. In this project, we quantify how clustering a training set into groups with similar geometric motifs can be used to train a NN so that it can accurately determine the energies of structures not in the training set. This was accomplished by generating over 800 C_8H_7N structures, relaxing them using DFTB+, and grouping them using agglomerative clustering. Some of these groups were assigned to the training group and used to train a NN using the pre-existing Atomistic Machine-learning Package (AMP) [10]. The remaining groups were evaluated using the trained NN and compared to the DFTB+ energy. These two energies were plotted and fitted to a straight line where higher R^2 values correspond to the NN more accurately predicting the energies of structures not in its training set. This process was repeated systematically with a different number of nodes and hidden layers. It was found that for limited NN architectures, the NN did a poor job predicting structures outside of its training set. This was improved by adding hidden layers and nodes as well as increasing the size of the training set.

Categories and Subject Descriptors
Computing methodologies - *Machine learning, Machine learning approaches, Neural networks*

General Terms
Algorithms, Measurement, Reliability

Keywords
Atomistic Machine-learning Package, neural network, genetic algorithm, agglomerative hierarchical clustering, Density Functional Tight Binding.

DOI: https://doi.org/10.22369/issn.2153-4136/12/1/5

1. INTRODUCTION
Machine learning programs are becoming increasingly popular and are a form of widely-accepted method for calculating properties. Such techniques are readily available in any field to help solve problems that would be otherwise difficult to solve or envision. An example of their capabilities is when a research group known as Laser Interferometer Gravitational-wave Observatory (LIGO) witnessed the phenomenon of gravitational waves in outer space. They imposed a technique known as Deep Learning which can learn from immense raw data using artificial neurons or neural networks [5]. Machine learning techniques can work to closely resemble atomistic calculators.

The common approach that we observe when preparing data to train and test a neural network (NN) — such as images or atomic descriptions — is to randomly assign data to the train and test sets. This strategy is appropriate when it is not expected that the NN will need to make predictions on test candidates that are very different from what it was trained on. One example of the random strategy is when researchers prepared tens of thousands of randomly plausible molecules to understand the relationship between light-harvesting systems and excitation energy transfer times such as those found in the pigments of plants [6]. The excitation transfer time refers to how pigments can transfer energy over long distances in the presence of a light-harvesting system, such as light from the sun, to produce energy. In this simulation, machine learning techniques were used to reduce computational cost and to discover which chromophoric molecule (or excitation system) had the most efficient transfer time. Another example of this strategy is using machine learning to discover drug designs in the field of medical science [14]. However, even in the light-harvesting system and excitation study, an improved method is preferred, because the random method is not evenly-sampled and could have redundant information.

We discuss the widely used random approach to introduce an alternative: clustering the data and training the NN with some of the clusters and testing the NN with the remaining clusters. In this case, clustering the molecules organizes them into groups of similar motifs [17]. By training the NN with a group of clusters and then testing it on another group of clusters, the ability for the NN to predict structures outside of its training set can be quantified. This paper will show that when the clustering method is implemented, the predictive ability of the trained NN will dramatically decrease relative to the random approach, but as the NN architecture grows, the NN becomes better at accurately predicting the potential energy of the trial system. This demonstrates that the clustering method can help to define how robust a trained NN is to predict properties of structures not in its training set.

In this work, we present that NN architectures and the training set size are both vital components when applying a NN to structures that are unique from the training set. By clustering the training set, the robustness of the NN can be quantified. When compared to randomly selecting the training and testing groups, it is clear how clustering can help guide designing the structure of a NN so that it can be used for a wider array of applications. In the Methods section, we present how NN architectures and the training sets were assembled using clusters. NNs are trained systematically with three, four, and five hidden layers, each with 10–35 nodes. In the Results section, we present plots of R^2 values from the linear fit between the relaxed DFTB+ energies and the NN calculated energies. Finally, we remark on how clustering training data can help to ensure NNs are robust and guide the architecture of an effective NN.

2. Literature Review

Atomistic calculators have become a beneficial tool for calculating properties at the atomic scale. Calculations start with descriptions of the physical interactions of the atoms, which then yields information about their collective behavior [16]. The Density Functional based Tight Binding (DFTB+) is an example of such atomistic calculator. Through the DFTB+ package, we collected potential energy values of readily-formed molecules. Atomistic calculators have helped perform calculations when studying computational techniques, including the genetic algorithm (GA).

A GA is a computational heuristic that can be used to solve for the global atomic minimum of chemical structures. The GA is efficient at searching through many different molecular motifs in the potential energy surface by utilizing the principles of natural selection. In nature, a species must increase its fitness if it is going to learn to adapt to its environment [18]. The GA is designed to work in a similar manner, because it will iteratively improve upon the immediate population [8]. This is because in a real-world situation the GA does not know the answer. Neither the configuration space nor the global minimum are known prior to the search [8]. The evaluation step — the aspect of the GA where the lowest conformation energy is being searched — will be the most time-consuming for the optimization of offspring structures. Typically, structures are relaxed using an expensive *ab initio* method.

The GA begins with two parent molecules in a starting population. The cut-and-splice operator cuts the parent molecules in two, resulting in four fragments. Typically, a fragment from each is selected at random to be spliced together to form a new child molecule [1]. The cut-and-splice apparatus is illustrated on the upper right-hand corner of Figure 1, where the box-like figures are meant to represent molecular sites. Two parent molecules are shown with the same number of boxes to represent their similar chemical stoichiometry. Upon performing the cut, the parent molecules are now color-coded to signify their fragments (parent molecule 1 is shown in blue and green while parent molecule 2 is shown in red and yellow). Finally, two fragments are collected from each parent molecule and spliced together to form a new structure. This new structure is the child structure (shown in blue and yellow) and is one plausible combination of bringing in genetic information from both parent molecules. Given that the program will read a diverse molecular population, the outcome of the cut-and-splice operation will be different combinations of diverse offspring structures.

The next step of the GA heuristic is the evaluation process. The energy of the child structure is evaluated, and if the structure is fit enough to be in the population, it is added. The search for the lowest conformation energy then continues with a new parent pair. The structure with the local minimum is in the bottom-center of Figure 1. In the iterative process of the GA, potential energies of new offspring structures are constantly being compared to the energy of the population [9]. Molecular configurations that are better in terms of minimum potential value replace configurations with a higher potential energy. If the energy of the offspring structure is lower than the energy of the population, then it is added to the population. Conversely, if the potential energy of the child structure is higher than the energy of the population, then that offspring molecule is deleted. The program thus continues the evolutionary-driven perspective of constantly searching for the lowest conformation energy while eliminating unfit offspring structures with higher energies. Given enough generations, the population will eventually get trapped in either a local minimum or find the global minimum.

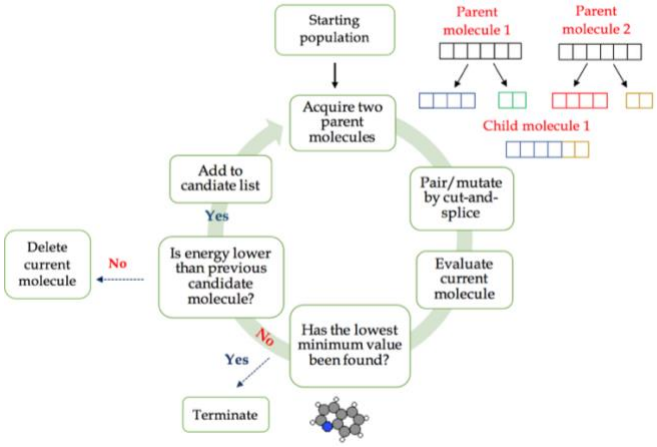

Figure 1. A flowchart outlining the steps of the GA from start to end. The program begins with a starting population, which proceeds in a loop where two molecules are selected and a new candidate structure is created using a cut-and-splice operation. The candidate is then evaluated and potentially replaces the least favorable member of the population. The GA terminates upon locating the lowest conformation structure.

The GA can be integrated with a NN to improve its ability to search the configuration space for the global minimum [12]. Machine learning techniques have revolutionized data analysis in most things we use today, such as travel booking, navigation, media recommendation, image recognition, and competitive board games. These systems are driven by the computational power, significant amount of data, and training ability of the neural network [2]. The neural network is comprised of hidden layers and nodes. Figure 2 depicts a NN containing an input layer, two hidden layers with four nodes each, and an output layer. Each node passes information forward in the form of entities known as weights and constant biases which are calculated via an activation function [10]. The process of the NN works like the human brain. The NN relies on the structural connection of nodes, where information can be obtained and sent between hidden layers, like how the brain relies on the connection of neurons [11].

Figure 2 showcases a fast feed forward NN. The connection of the eight nodes and two hidden layers is represented by the gray-colored arrows. The input layer passes a weight value representing connection strength to the first, second, third, and fourth nodes of the first hidden layer. If we focus on the first node of the first hidden layer, we can witness how that information is then passed on to the

first, second, third, and fourth node of the second hidden layer. This process repeats in the second node of the first hidden layer to all nodes of the second hidden layer, and so forth, until all nodes are inter-connected. The NN can only advance in one direction (from left to right) and terminates with an output value with no loops or turns [15].

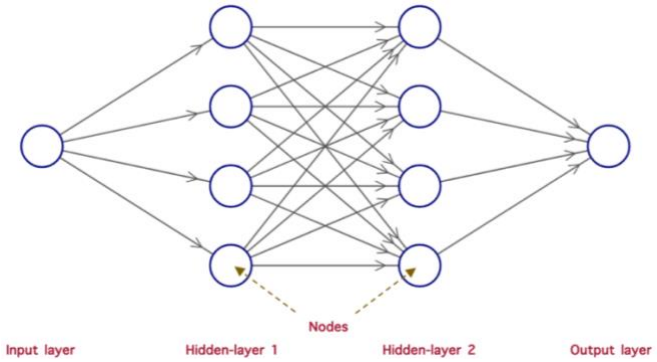

Figure 2. A visualization of a neural network containing an input layer, two hidden layers with four nodes each, and an output layer.

Numerous cluster programs exist, but the clustering method utilized for this project is known as agglomerative hierarchical clustering (AHC). Clusters created by AHC have been used efficiently on molecular populations to improve GA efficiency [9]. In general, molecules of the same chemical stoichiometry exist in different chemical configurations, such as lines, rings, or a combination of both. The AHC recognizes this diversity of structures and attempts to group them according to their close molecular similarity. This is achieved through the similarity threshold, which is the cutoff that groups structures together. What makes the AHC unique from other clustering programs is its bottom-up approach, where each data point starts in its own cluster and then proceeds by successfully merging similar classes of clusters together, which forms a hierarchy [7].

Figure 3 showcases this process through the visualization of a simple dendrogram. At first, each structure lives in its own cluster. When the AHC recognizes similar structures, it groups those two clusters (which were separated before) to form one cluster. At the end, the AHC has created three clusters (presented in blue, red, and green). Although seven molecules are shown in Figure 3, the AHC program can read in a variety of readily-formed chemical structures to form more sophisticated groups.

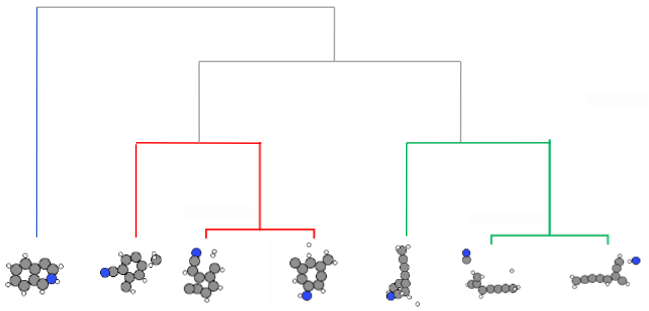

Figure 3. A segment of a dendrogram showing seven C_9H_7N molecules being grouped into three clusters (blue, red and green) based on a similarity index.

3. METHODS

To test how well the NN performs, we compared its values to those determined from DFTB+. DFTB+ combines the accuracy given from the DFT method with the efficiency of the Tight-Binding (TB) method [3]. This atomistic calculator was chosen because it is a fast, empirical method that allows us to perform many simulations with an appropriate level of accuracy. All structures are composed of C_9H_7N, whose global minimum is quinoline. This molecule is chosen because it is complex enough to have many local minima, but not so many that it becomes difficult to easily categorize them manually.

The NN was implemented using the Atomistic Machine-learning Package (AMP), an open-source, Python-based (accelerated by Fortran) code that was built to interface seamlessly with the Atomic Simulation Environment (ASE) [13]. ASE is an open-source, Python-based common front end that is capable of supporting many molecular calculators. AMP is capable of using several descriptors and activation functions for its NN; however, for this project, the default Gaussian descriptor and hyperbolic tangent activation function were used.

The comparison scheme comparing energies calculated by NN to those calculated by DFTB+ is illustrated in Figure 4. C_9H_7N molecules were evaluated with both NN and DFTB+ calculators. These energies were compared to determine how well a NN accurately calculated its energy. Once the testing set was fully evaluated, the NN and DFTB+ energies were plotted and fit to a straight line. The R^2 value from this fit was used to quantify how close the NN can represent the DFTB+ calculated value. The R^2 value determined from a data set where the train and test sets were identical was 0.9999, indicating a nearly perfect match. The lower the R^2 value, the less predictive the NN is on average at determining the energy calculated by DFTB+.

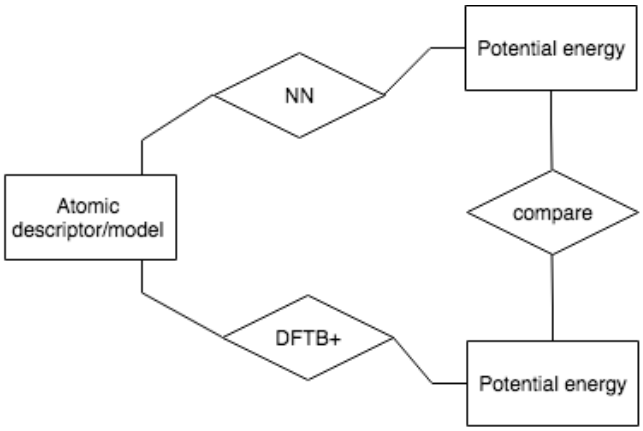

Figure 4. An overview of the comparison scheme in which NN and DFTB+ are used to evaluate the predictive nature of the NN. Both the NN and DFTB+ are used to evaluate the potential energy of a candidate, and the two values are compared.

A starting population consisted of 813 molecules generated from an evolutionary algorithm using the C_9H_7N stoichiometry. These were clustered into 21 clusters using AHC. The similarity index was arbitrarily set to generate 21 clusters, so that each cluster was sufficiently small, to have some flexibility in grouping them into five equally-sized groups with roughly 20% of the total structures in each group. Regardless of this effort, group 1 was composed of one very large cluster (288 molecules), while the remaining groups

were composed of roughly equal numbers of structures (133, 129, 125, and 138 molecules). Figure 5 illustrates a schematic showing group formation separated by arrows from left to right. The first step shows a sample of the population set. The second step shows that all similar molecules were clustered using the AHC method. The third and final step presents Groups 1–5 (represented by circles) which contain approximately 16% of the total population from the 813 molecules (Group 1 has 35% of the structures).

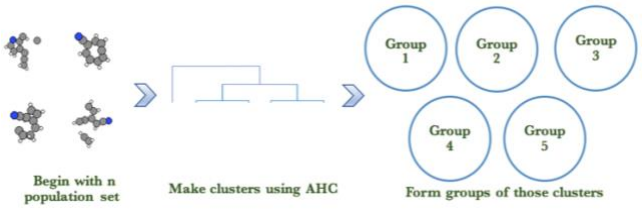

Figure 5. A schematic displaying how groups were formed from the results of the agglomerative clustering of initially generated structures. Each group consists of approximately 20% of the atomic molecules from the population.

The train/test cycle presented in Figure 6 outlines how the five train/test groups were created. When training a NN, a large training set is ideal. Sometimes this is not possible, so we tested the effect of 20% and 80% training set sizes to quantify the effect of a limited training set on a NN versus having a much larger training set. Figure 6 outlines the case of a 20% training set size. To start, one of the five groups was designated the training group, while the other four groups were combined to make the test set. For example, Test 1 included 525 molecules (the combined total of Groups 2–5) while Train 1 included 288 molecules. A NN was trained using the training set, and then all the DFTB+ energies of the structures in the test set were compared to those calculated using the newly-trained NN (as in Figure 4). This process was repeated four more times, where each of the five groups had a turn being the training set. The purpose of the train/test cycle was to average out any structure-related issues in any of the training sets. For each of the five sets, a scatter plot of the NN energy versus the DFTB+ energy was fit to a straight line, and the five R^2 values were averaged. To calculate the results from 80% of the structures being in the training, the process in Figure 6 was repeated with the training and testing groups reversed.

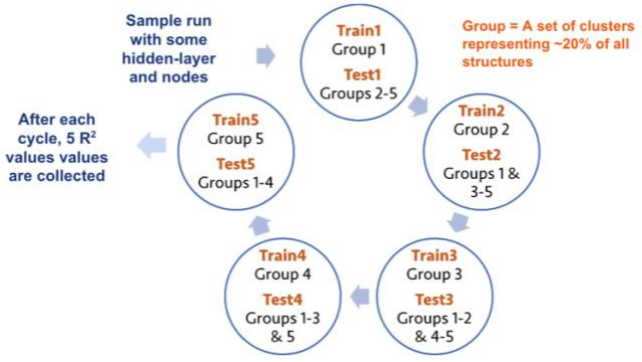

Figure 6. The representation of the train/test cycle that results in determining an R^2 value for each of the five assignments of the groups.

Finally, the effect of the NN architecture was tested by varying the value of hidden layers and nodes for each cycle. For this study, we examined three, four, and five hidden layers, each containing 5, 10, 15, 20, 25, 30, and 35 nodes.

4. RESULTS

Figures 7 and 8 plot the results of the random and clustered grouping of the training set. Each figure displays the average R^2 value — each collected from one complete round of the train/test cycle — for each hidden layer/node combination. The 20%/80% sized training sets are indicated by the blue and red lines, respectively.

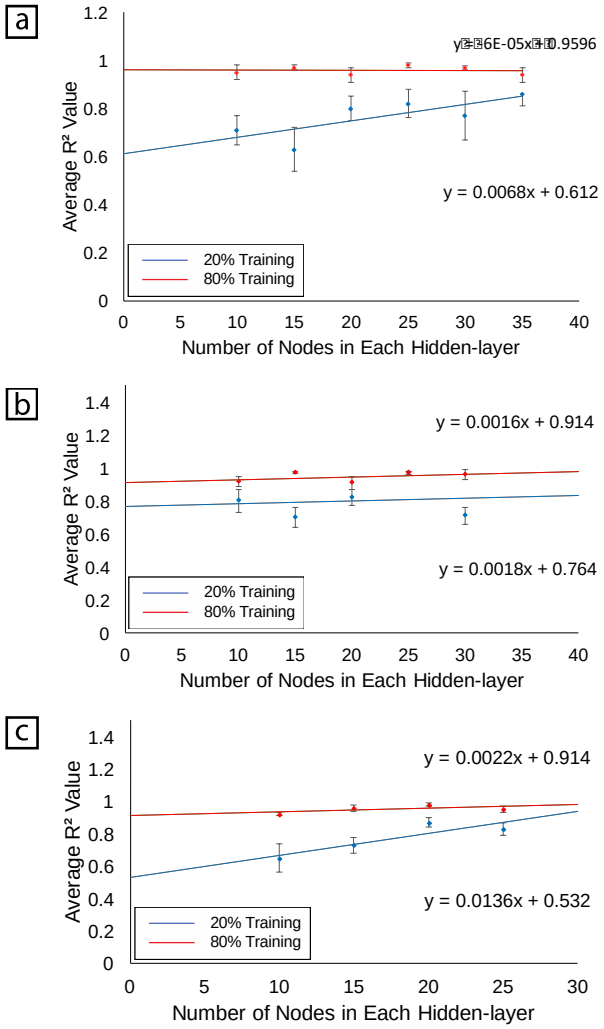

Figure 7. Three linear plots depicting averaged R^2 values and their uncertainties, where the NN is trained with three (in a), four (in b), and five (in c) hidden layers. The red line shows the 80% training set, and the blue line shows the 20% set.

Figures 7a, 7b, 7c, are the analysis for the randomly arranged groups. Figure 7a has three hidden layers, Figure 7b has four hidden layers, and Figure 7c has five hidden layers. Each figure displays averaged R^2 values as the number of nodes increased. The 80% training sets in Figures 7a, 7b, and 7c, have much smaller slopes and higher y-intercepts compared to the clustered data. Conversely, the 20% training sets have greater slopes and lower y-intercepts. There is less uncertainty in each of the averaged R^2 values in the 80% sets than in the 20% sets, as demonstrated by the smaller error bars. In general, the 80% training sets have higher R^2 values compared to the 20% training sets.

Figures 8a, 8b, 8c, show the results for the clustered groups. Figure 8a has three hidden layers, Figure 8b has four hidden layers, and Figure 8c has five hidden layers. In general, the 80% training sets in the clustered data have greater slopes and higher y-intercepts. The 20% training sets have smaller slopes and lower y-intercepts. The uncertainties of the 80% and 20% training sets appear to be similar. The 80% training sets have higher R^2 values, while the 20% training sets have lower R^2 values. In both instances (regarding random and clustered populations), the training set with more molecules resulted in an increase in R^2 values. These data are summarized in Table 1.

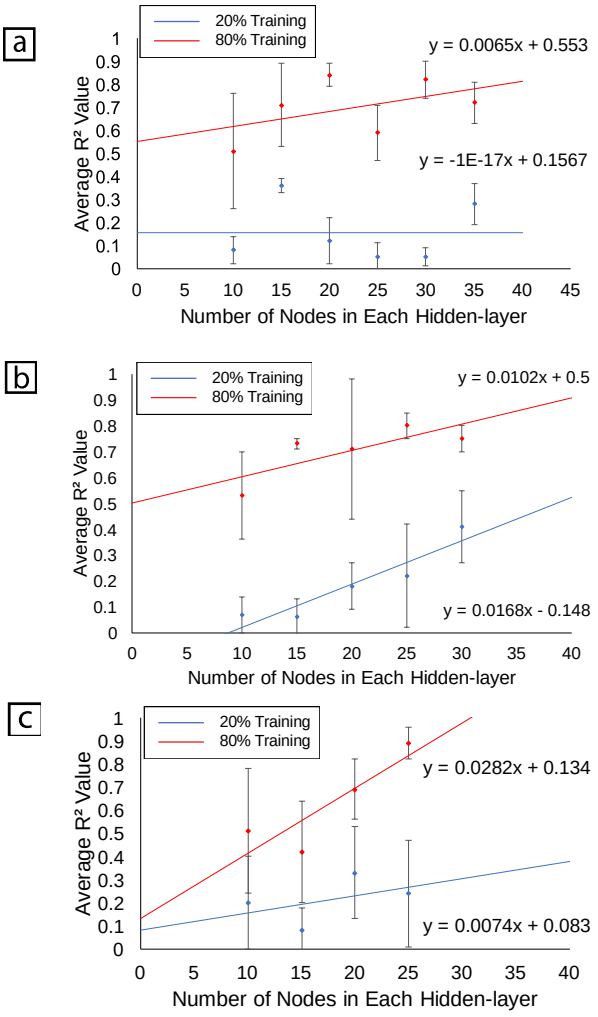

Figure 8. Three linear plots depicting averaged R^2 values and their uncertainties where the NN is trained with three (in a), four (in b), and five (in c) hidden layers. The red line shows the 80% training set, and the blue line shows the 20% set.

5. DISCUSSION

In the random data, the slopes for the 20% training sets are larger compared to the slopes for the 80% training sets. Even though the 20% trained data have much lower y-intercepts (meaning that they are not as accurate as the 80% data for small number of nodes), the slopes mean they eventually make up the difference as the number of nodes increases. One reason for the small slopes in the 80% data is that the R^2 values are already close to the maximum value of 1. This indicates that with a large amount of data, this NN is able to reliably calculate the DFTB+ energy for structures that are similar to those in its training set, regardless of the NN architecture. If a limited amount of data is available for the training set, reliable predictions can still be made with a NN, so long as the NN architecture is large.

This is different for the clustered data. The slopes (in general) for the 20% trained data are much smaller than the slopes for the 80% trained data. This means that smaller training data sets will require a very large number of nodes if they are going to catch up to the 80% data, if at all. Furthermore, contrary to the random data, for the larger training sets, there appears to be a much stronger correlation between NN architecture and the number of nodes in each hidden layer. This probably stems from the NN needing more extensive architectures to help predict structures that are not in the training set.

Table 1: Summary of the slopes and y-intercepts from Figures 7 and 8.

	20% Training		80% Training	
	Slope	y-intercept	Slope	y-intercept
Random Grouping				
3 Hidden Layers	0.0068	0.612	-6.00E-05	0.9596
4 Hidden Layers	0.0018	0.764	0.0016	0.914
5 Hidden Layers	0.0136	0.532	0.0022	0.914
Clustered Grouping				
3 Hidden Layers	-1.00E-17	0.1567	0.0065	0.553
4 Hidden Layers	0.0168	-0.148	0.0102	0.5
5 Hidden Layers	0.0074	0.083	0.0282	0.134

The larger y-intercepts for the 80% cases shows that they are much better at predicting structures outside of the training set for small NN architectures. It follows that when more examples are shown to the NN, it translates to better accuracy. The fact that the y-intercept drops significantly between the random and clustered data suggests that it matters to train the NN in a specific way if it needs to accommodate structures that it has never seen before. This is due to the fact that the random training set is comprised of all types of structures, while the clustered case attempts to segregate molecule types. and this is maintained in forming the training and testing groups. As a result, a false sense of accuracy exists for structures that have never been seen before when training with a random starting population.

Neural network architectures and training set size are important when trying to apply the NN calculator to structures that were not in the training set. This is illustrated by comparing the random training sets to the clustered ones. For the random training sets, regardless of the NN arrangement, it appears that we get at least a reasonable, if not very close, match to the DFTB+ energy value. Conversely, in the clustered data, we get a dramatically-reduced ability to predict the energy of the testing set structures, especially for architectures with a smaller number of hidden layers. This indicates that clustering should be used as a strategy to train NNs when it is expected that the NN will predict structures that are unique from the training set.

There are other examples of effective pairing of clustering data with other machine learning techniques. One study showed that the GA efficiently located the global minimum because it took advantage of the clustered configuration space [1]. Because of their potential for locating the global minimum, clusters and the GA are often found working simultaneously in areas of computational research. Another example showing the effectiveness of combining the GA

and clusters is found in a study where an approach known as the distributed hierarchical genetic algorithm (DHGA) was used for optimization, pattern matching, and space search exploration [4]. One study showed that when starting populations were clustered, the chances of finding the local minimum increased within a margin of error opposed to populations that were not subjected to the clustered approach [9]. The combination of the GA and clusters is constructive due to the ability of the GA to recognize a pattern when searching through the clustered configuration space. The GA can act upon this information and continue the search for the local minimum [1].

This project shows how clustering can work to illustrate how robust a trained NN is at accurately calculating properties of structures not in its training set. Traditional training of NN's with randomly chosen structures might provide a false sense of accuracy. The common starting conditions for NN are typically based on a random set, such as those found in the neural network potential (NNP) simulations that aimed to learn transferable potentials for organic molecules [15]. We have shown that certain NN architectures do not appear to do well when given a previously unseen structure. Clustering should be considered to determine a NN architecture that is robust enough to recognize structures that it has not seen before, since it can be quantified how accurate the NN is as a part of the training/testing sets. When randomly selected training/testing sets are used, the NN will perform well, but that is expected, since the test set is composed of structures similar to those on which it was trained.

We believe clusters should help improve the search scheme when deciding to create improved sophisticated programs of chemically relevant molecules in the potential energy surface. By learning to make the NN more robust in this project, we can discuss training a NN to generate starting populations for GAs. We envision a strategy where clustering is used to group together previous search results, and new starting populations can be quickly evaluated using a NN and compared to the clustered data to find new structural motifs and start new GA searches in unseen areas of the potential energy surface. This work informs how the NN should be constructed for this eventual application of a thorough and complete search for the global minimum. Ultimately, we have showed how training a NN requires some analysis to determine an architecture that will be robust enough to predict results from trial cases that it has not seen before using clusters.

6. CONCLUSION

In this paper, we demonstrate the importance of training NNs for calculating the energy of molecular structures using clustering to group together the training and testing data if it is expected that the NN will evaluate new structures that are outside of the training set. Using a random grouping in the training and testing sets resulted in the NN being able to almost exactly reproduce the correct energies of all tested structures. However, when the NN was trained with a group of clustered structures and tested on a separate group of clustered structures, the NN performed much worse. Enlarging the training set and increasing the number of hidden layers and nodes dramatically improves the ability of the NN to predict molecular energies of structures that are not similar to those in the training set. This approach provides a framework to determine the proper architecture to train more robust NNs with a quantifiable metric.

7. REFLECTION

I (CB) am excited to have been a part of the Blue Waters Student Internship Program. The financial support replaced having to work a part-time job, which provided the time necessary to complete the work for this publication. Furthermore, the two-week long computational science institute at the University of Illinois at Urbana-Champaign, hosted by the Shodor foundation, was a great experience. I really enjoyed learning about the many applications of high-performance computing, and the whole experience allowed me to build upon my computer science and chemistry backgrounds. This whole process really opened my mind to the opportunities in this field, and I hope to pursue these opportunities in the future. I would like to thank everyone involved with this program for a genuinely great experience.

8. REFERENCES

[1] Ulrich Bodenhofer. 2004. *Genetic Algorithms: Theory and Applications*. Fuzzy Logic Laboratorium Linz-Hagenberg. Johannes Kepler University, Linz, Austria. Retrieved from https://www.flll.jku.at/div/teaching/Ga/GA-Notes.pdf

[2] Yudong Cao, Gian G. Guerreschi, and Alán Aspuru-Guzik. 2017. Quantum Neuron: An elementary building block for machine learning on quantum computers. arXiv: 1711.11240. Retrieved from https://arxiv.org/abs/1711.11240

[3] Ye Chen, Meng Liu, Jianhua Chen, Yuqiong Li, Cuihua Zhao, and Xiao Mu. 2018. A density functional based tight binding (DFTB+) study on the sulfidization-amine flotation mechanism of smithsonite. *Appl. Surf. Sci.* 458 (Nov. 2018), 454-463. DOI: https://doi.org/10.1016/j.apsusc.2018.07.014

[4] Gautam Garai and B.B. Chaudhuri. 2007. A distributed hierarchical genetic algorithm for efficient optimization and pattern matching. *Pattern Recogn.* 40, 1 (Jan. 2007), 212-228. DOI: https://doi.org/10.1016/j.patcog.2006.04.023

[5] Daniel George and E.A. Huerta. 2018. Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data. *Phys. Lett. B.* 778 (Mar. 2018), 64-70. DOI: https://doi.org/10.1016/j.physletb.2017.12.053

[6] Florian Häse, Christoph Kreisbeck, and Alán Aspuru-Guzik. 2017. Machine learning for quantum dynamics: deep learning of excitation energy transfer properties. *Chem. Sci.* 8, 12 (Dec. 2017), 8419-8426. DOI: https://doi.org/10.1039/c7sc03542j

[7] Anil K. Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recogn. Lett.* 31, 8 (Jun. 2010), 651–666. DOI: https://doi.org/10.1016/j.patrec.2009.09.011

[8] Mathias S. Jørgensen, Michael N. Groves, and Bjørk Hammer. 2017. Combining Evolutionary Algorithms with Clustering toward Rational Global Structure Optimization at the Atomic Scale. *J. Chem. Theory Comput.* 13, 3 (Feb. 2017), 1486-1493. DOI: https://doi.org/10.1021/acs.jctc.6b01119

[9] Nicholas Kellas and Michael N. Groves. 2020. Improvement of the Evolutionary Algorithm on the Atomic Simulation Environment Though Intuitive Starting Population Creation and Clustering. *Journal of Computational Science Education* 11, 2 (Apr. 2020), 29-35. DOI: https://doi.org/10.22369/issn.2153-4136/11/2/5

[10] Alireza Khorshidi and Andrew A. Peterson. 2016. Amp: A modular approach to machine learning in atomistic simulations. *Comput. Phys. Commun.* 207, (Oct. 2016), 310-324. DOI: https://doi.org/10.1016/j.cpc.2016.05.010

[11] Philipp Koehn. 1994. *Combining Genetic Algorithms and Neural Networks: The Encoding Problem*. Master's thesis. The University of Tennessee, Knoxville, Knoxville, TN.

[12] Esben L. Kolsbjerg, Andrew A. Peterson, and Bjørk Hammer. 2018. Neural-network-enhanced evolutionary algorithm applied to supported metal nanoparticles. *Phys. Rev. B.* 97, 19 (May 2018), 195424. DOI: https://doi.org/10.1103/PhysRevB.97.195424

[13] Ask Hjorth Larsen, et al. 2017. The atomic simulation environment—a Python library for working with atoms. *J. Phys.-Condens. Mat.* 29, 27, (Jun. 2017), 273002. DOI: https://doi.org/10.1088/1361-648X/aa680e

[14] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. 2018. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.* 4, 1 (Dec. 2017), 120-131. DOI: https://doi.org/10.1021/acscentsci.7b00512

[15] Justin S. Smith, Olexandr Isayev, and Adrian E. Roitberg. 2017. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* 8, 4 (Feb. 2017), 3192-3203. DOI: https://doi.org/10.1039/c6sc05720a

[16] Logan Ward and Chris Wolverton. 2017. Atomistic calculations and materials informatics: A review. *Curr. Opin. Solid St. M.* 21, 3 (Jun. 2017), 167-176. DOI: https://doi.org/10.1016/j.cossms.2016.07.002

[17] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. 2010. Image Clustering Using Local Discriminant Models and Global Integration. *IEEE T. Image Process.* 19, 10 (Apr. 2010), 2761-2773. DOI: https://doi.org/10.1109/TIP.2010.2049235

[18] Xinjie Yu and Mitsuo Gen. 2010. *Introduction to Evolutionary Algorithms.* Springer-Verlag London. DOI: https://doi.org/10.1007/978-1-84996-129-5