MACHINE LEARNING FOR PREDICTING CHEMICAL ENERGIES

Alexandra McIsaac
amcisaac@mit.edu

May 10, 2019

ABSTRACT

This paper explores the use of machine learning methods for predicting energies of molecules. We seek to predict energies at room temperature, which are most important for comparing to experiment, rather than energies at absolute zero which is more common in the literature. Models are developed to predict the internal energy at absolute zero, the internal energy at room temperature, the enthalpy at room temperature, and the Gibbs' Free Energy at room temperature, using only the molecular geometry as an input (represented by the eigenvalues of the Coulomb matrix). We use two machine learning methods to build these models: kernel ridge regression and a neural network. We find that the kernel ridge regression is more successful than the neural network.

1 Introduction

One of the primary tasks of computational chemistry is to calculate the energies of atoms and molecules. These energies allow us to study many interesting properties, such as reactivity, as well as gain microscopic insight into a problem that can be challenging to probe experimentally. It also allows for computational screening before synthesis or experiment, so that valuable physical resources can be saved for compounds that are likely to be successful for a given task. Many methods have been developed to calculate molecular energies from first principles, using various approximations to solve the time-independent Schrödinger equation. These methods have had much success, but with the cheapest method scaling as $\mathcal{O}(N^4)$, some researchers have begun looking for more computationally tractable solutions.

A few groups have applied machine learning to molecular energy calculations. Some of these works have used kernel ridge regression or neural networks to map the molecular geometry (a list of all the atoms in the molecule, along with the Cartesian coordinates and nuclear charges of those atoms) to various properties,[1–4] most commonly atomization energy,[4–7] which is the energy of the reaction where a molecule decomposes into free atoms. Thermodynamic properties at room temperature, such as internal energy, enthalpy, or free energy, are the most important for comparing to experiment; however, there have been few attempts to machine learn these properties. Some work has been done developing a machine learning-based correction to conventional computational methods, in order to bring the conventional theory closer to the experimental answer.[6,8] However, it seems that little work has been done to predict these properties directly from machine learning. This project aims to address this issue by investigating whether machine learning techniques can be used to predict thermodynamic properties of molecules.

2 Computational Details

2.1 Model inputs and outputs

To train all models, I used data from the QM9 database,[9] which has molecular geometries and thermodynamic properties for 133,885 small organic molecules.

As input features, I wanted to use the same input as is used for conventional molecular energy calculations: the molecular geometry. In order to transform a molecular geometry into something that could be used as a feature, I followed previous work in this field, which represents a molecule using its Coulomb matrix:[7]

$$M_{ij} = \begin{cases} \frac{Z_i Z_j}{R_{ij}} & i \neq j \\ \frac{1}{2} Z_i^{2.4} & i = j \end{cases} \tag{1}$$

Where Z_i and Z_j are the atomic nuclear charge for atoms i and j, and R_{ij} is the distance between atoms i and j. The off-diagonal elements of this matrix represent the Coulomb repulsion energy between two atoms, and the diagonal elements represent an approximation to the potential energy of the atom. This matrix is unique to a given molecule, however, the order of the rows/columns in the matrix depends on the order that the atoms are listed in the geometry, which is arbitrary and should not affect the energy. Therefore, several Coulomb matrices will correspond to the same molecule, which is not ideal. In order to work around this issue, I used the sorted eigenvalues of the Coulomb matrix as my features, as suggested by Rupp et. al.[7] These should also be unique to a given molecule, but should have a one-to-one correspondence (only one set of eigenvalues should correspond to a specific molecule). Because the molecules in the dataset have different numbers of atoms, they will have Coulomb matrices with different dimensions. Therefore, to make all input vectors the same dimension, I have padded the Coulomb matrix of smaller molecules with zeros such that it matches the dimension of the largest molecule. The largest molecule in the dataset has 29 atoms, so all input vectors have 29 features.

As outputs to predict, I considered four thermodynamic properties: the internal energy at absolute zero (U0), the internal energy at room temperature (U298), the enthalpy at room temperature (H298), and the free energy at room temperature (G298).

2.2 Kernel Ridge Regression

2.2.1 Background

The first method used was Kernel Ridge Regression, which had shown good success for internal energy at absolute zero, and thus should generalize well to energies at room temperature. Kernel Ridge Regression uses the following form for its objective function:[5]

$$J(\mathbf{X}, \mathbf{Y}; \alpha, \gamma) = \sum_i (y_i - \sum_j c_j k(x_j, x_i; \gamma))^2 + \alpha \sum_{i,j} c_i k(x_i, x_j; \gamma) c_j \tag{2}$$

Where $\mathbf{X}$ is the input features, $\mathbf{Y}$ is the reference values (here, the actual energy), α is a regularization hyperparameter, γ is a kernel hyperparameter (equal to $1/\sigma^2$ for the Gaussian and Laplacian kernels), c_i and c_j are the regression weights, and $k(x_i, x_j)$ is the kernel function, which introduces non-linearity into the regression problem. It has previously been shown that the Laplacian kernel gives good results for molecular properties.[5] I tested linear, quadratic, cubic, Gaussian, and Laplacian kernels, and found that the Gaussian kernel gave the best results for this data set. All regression training was performed using the Python package SciKitLearn.

I trained regression models for each of the four thermodynamic properties: the internal energy at absolute zero (U0), the internal energy at room temperature (U298), the enthalpy at room temperature (H298), and the free energy at room temperature (G298).

2.2.2 Model development

To evaluate the best kernel and hyperparameters for this problem, I conducted several sweeps over hyperparameters for a variety of kernels (linear, quadratic, cubic, Laplacian, and Gaussian). I conducted all sweeps on the U0 data set of internal energies at zero Kelvin, because this property has been successfully modeled with Kernel Ridge Regression already, albeit using a different data set.[7] I expect the model performance to be similar for all thermodynamic properties, and thus the optimal kernel for the U0 data should generalize to the U298, H298, and G298 data.

First, to identify the most promising kernels, I did a preliminary sweep over hyperparameters using a small training data set of 2,500 molecules, with no cross-validation. All errors reported are mean absolute errors. From this preliminary sweep, I identified the quadratic and Gaussian kernels as giving the smallest testing error at the optimal hyperparameters.

I then conducted a more robust sweep of hyperparameters for the quadratic and Gaussian kernels, using 5-fold cross-validation on a data set of 10,000 total molecules, yielding a training set of 8,000 molecules for each round and a testing set of 2,000 molecules. From this cross-validation sweep I identified the best hyperparameters to be $\alpha = 1.5 \times 10^{-5}$ for quadratic (Fig. 1) and $\alpha = 1 \times 10^{-11}$, $\gamma = 1/\sigma^2 = 5 \times 10^{-6}$ for Gaussian (Fig. 2).

Using these optimal hyperparameters and kernels, I assessed the effect of training set size on the 5-fold cross-validation testing error, shown in Fig. 3. I used training set sizes ranging from 4,000-32,000 molecules. The error in both kernels plateaus around 28,000 to 32,000 training molecules, but the Gaussian error with optimal α and γ (34 kcal/mol) is much lower than the quadratic at optimal α (40 kcal/mol). From this final sweep, I decided to use the Gaussian kernel with a training set size of 30,000 molecules for all properties.

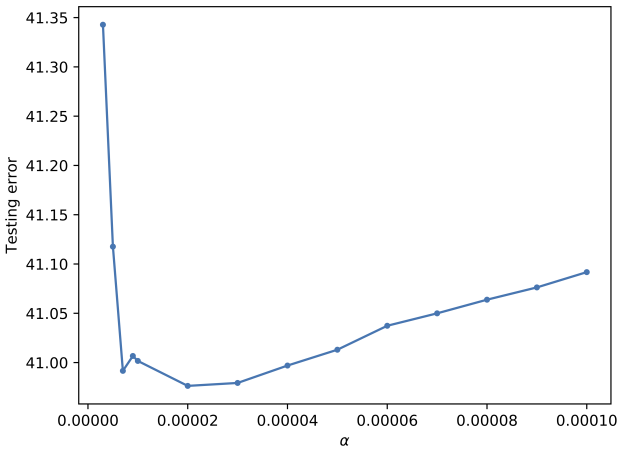

Figure 1: 5-fold cross-validation hyperparameter sweep for the quadratic kernel. Testing error is in kcal/mol. Larger values of α were tested, but gave large errors. Only the optimal range of α is shown.

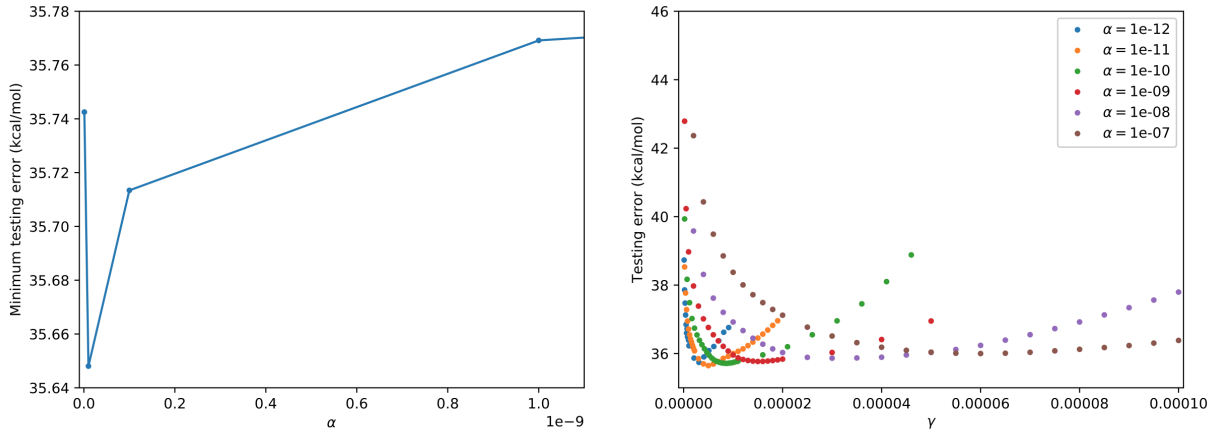

Figure 2: 5-fold cross-validation hyperparameter sweep for the Gaussian kernel. Note that α and γ values up to 0.01 were tested, but gave very large minimum errors. Only the near-optimal values are shown. Left: Minimum testing error as a function of α. “Minimum testing error" refers to the testing error at the optimal value of γ for that α. Right: Testing error as a function of γ, for several values of α.

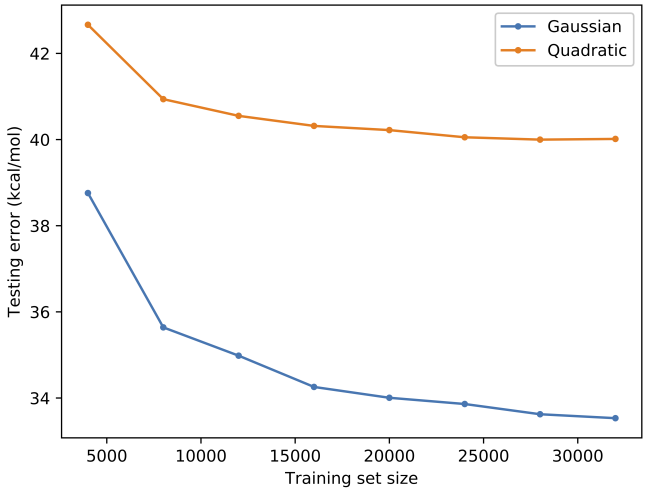

Figure 3: Effect of training data set size on 5-fold cross-validation testing error.

After performing the above tests for the U0 data set, I performed just the 5-fold cross-validation test on 10,000 molecules for the U298, H298, and G298 data, to find the best hyperparameters for the Gaussian kernel for these data. I found that the same optimal hyperparameters that were identified for U0 were also the optimal parameters for these data.

2.3 Neural Networks

2.3.1 Background

As an alternative to kernel ridge regression, I also developed a neural network model in hopes that the more complex hypothesis class could perform better than a regression. For the overall structure, I selected a fully connected feed-forward neural network, as it seemed best suited for a regression-type problem. A feed-forward neural network is made up of layers, each of which contains a specified number of units. In the first layer, the input features (X) are multiplied by weights (defined for each unit), and then summed to form a new quantity, Z:

$$Z = W^T X + W_0 \tag{3}$$

Z is the transformed by the activation function for that layer to yield a:

$$A = f(Z) \tag{4}$$

A is then fed into the next layer, and the process is repeated, but replacing the original input features, X, with the output of the previous layer, A. We define a loss function for the final layer, then optimize the weights by minimizing the final loss of the network with respect to the weight parameters at each layer using error back-propogation. I used the least squared error as a loss function, in order to be consistent with the Kernel Ridge Regression model. All Neural Network development was performed with the Python packages Keras and TensorFlow.

2.3.2 Model development

There are several choices to be made when developing a Neural Network model. For the output layer, I chose a single unit with a linear activation function, in order to output a numerical prediction of the energy. For the activation function of the hidden layers, I explored several options for the activation function at each layer: ReLU, hyperbolic tangent, and sigmoid functions. However, I found in preliminary tests that only ReLU gave reasonable results (<100 kcal/mol average error) and thus I proceeded using ReLU for all hidden layers.

To evaluate the best architecture for the Neural Network, I explored the effect of using different numbers of hidden layers and different numbers of units per layer. I tested models with 1-4 hidden layers, and for each number of layers, I conducted a sweep of different number of units (35, 25, 15, and 5 units) per input or hidden layer. I also tested a model with 5 hidden layers, but due to computational cost, instead of conducting a full sweep for the number of units in each layer, I took the best two architectures from the 4 hidden layer model and explored the effect of adding an

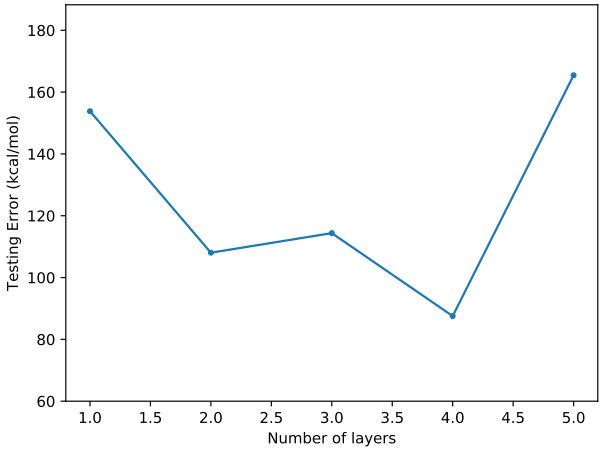

Figure 4: 5-fold cross-validation testing error, reported for the optimal architecture for each number of hidden layers. Larger numbers of hidden layers were tested (ranging from 10-50), but gave significantly larger errors and are thus not shown.

extra hidden layer with 35, 25, 15, and 5 units either before the first hidden layer or after the last hidden layer. In addition, I tried a few models with 10-50 hidden layers, however they gave significantly larger errors than the smaller numbers of layers mentioned above and so I did not pursue any more robust testing. For all initial tests, the model was trained for 20 epochs with a truncated dataset containing 40,000 training points and 10,000 testing points, all for the U0 dataset. I examined a plot of the training and testing error vs number of epochs for each architecture considered to ensure appropriate convergence. In addition, I checked that increasing the complexity of the network was not leading to overfitting by comparing the training and testing losses, which remained comparable in every case.

From these tests, I used the testing error to identify the optimal architecture for each number of hidden layers, which is reported in Table 1. I then performed 5-fold cross-validation on the optimal architecture for each number of layers to identify the overall best model. The 5-fold cross-validation testing error for each optimal architecture (mean absolute error) is plotted in Fig 4. From these results, I chose to use the optimal neural network with 4 hidden layers. I used 25 input units, 15 units in the first hidden layer, 5 units in the second and third hidden layers, and 15 units in the fourth hidden layer. Because the best hyperparameters were the same across all four data sets (U0, U298, H298, G298) for the kernel ridge regression, I assumed that would extend to this model and used this architecture to fit all four models. Because training the Neural Network was much less computationally demanding than the regression, I used the entire dataset to train each model, using 80% of the data for training and 20% for testing. All neural network models were optimized with the Adam method.

Number of HL	# input units	# units in HL1	# units in HL2	# units in HL3	# units in HL4	# units in HL5
1	15	25				
2	25	5	15			
3	35	15	5	15		
4	25	15	5	5	15	
5	35	5	35	5	25	25

Table 1: Optimal unit arrangements for each number of hidden layers. "# units in HL n" refers to the number of units in the nth hidden layer, e.g. "# units in HL1" means number of units in the first hidden layer.

3 Results and Discussion

3.1 Kernel Ridge Regression

Separate regression models were trained for U0, U298, H298, and G298, using 30,000 molecules as a training set and the remaining 103,886 molecules as a testing set. All models showed similar errors, which are reported in Table 2. The

error in the conventional method used to generate the data set is around 7 kcal/mol, so while none of my models manage to obtain the same accuracy as the original method, the regression models for all of the thermodynamic properties have errors that are on average below 0.02% of the total energy being predicted. In addition, the training and testing errors are very similar, indicating that no overfitting is taking place.

Energy type	Training Error (kcal/mol)	Testing Error (kcal/mol)
U0	32.12	33.85
U298	32.13	33.87
H298	32.11	33.84
G298	32.11	33.83

Table 2: Training and testing errors for the internal energy at absolute zero (U0), internal energy at room temperature (U298), enthalpy at room temperature (H298), and free energy at room temperature (G298).

3.2 Neural Network

Separate neural networks were trained for U0, U298, H298, and G298, using 107,108 molecules for training and the remaining 26,778 molecules for testing. All models were trained for 20 epochs, and convergence was checked by examining the plot of error vs number of epochs, shown in Fig. 5.

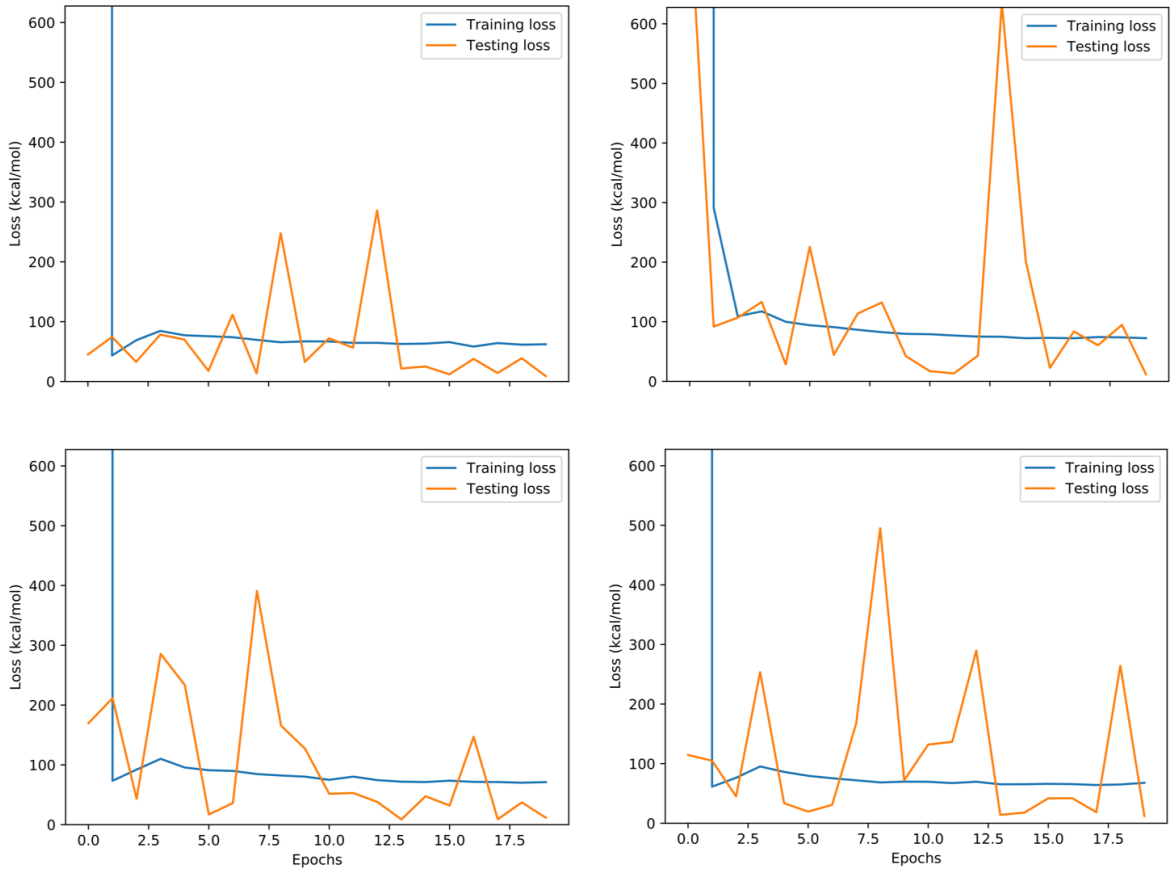

Figure 5: Convergence of training and testing error with respect to epoch for all four NN models. Top right: U0, top left: U298, bottom right: H298, bottom left: G298.

As was seen in the regression method, all models showed similar errors, reported in Table 3. The training and testing errors are similar, indicating that no overfitting is taking place. Unfortunately, all of the neural network models produced higher errors than the regression model, but they are still on average only about 0.025% of the total energies.

Energy type	Training Error (kcal/mol)	Testing Error (kcal/mol)
U0	52.67	53.13
U298	59.82	60.57
H298	61.48	61.04
G298	58.13	59.11

Table 3: Training and testing errors for the internal energy at absolute zero (U0), internal energy at room temperature (U298), enthalpy at room temperature (H298), and free energy at room temperature (G298).

4 Discussion

From examining the results of the regression and neural network, it appears that both have relatively low percentage errors. Surprisingly, the neural network performs noticeably worse than the regression. This could be due to the kernel in the regression, which expands the size of the input space and provides a measure of similarity between the molecules, which could make extrapolation easier. Another explanation would be the number of parameters in the models. The regression model has a weight for each point in the training set, yielding 30,000 parameters in the case presented here. The 4-layer neural network, however, has only about 1,400 parameters, which is determined by the number of units in each layer. Perhaps a deeper neural network with more hidden layers could be more competitive with the kernel regression; however, all of the models with >10 layers tested in the preliminary model selection process showed errors >10 times as large as those reported with the 4-hidden-layer model.

Unfortunately, while the percent error is very low for all of the thermodynamic properties, it is still larger than the difference between any of the room-temperature energies and the internal energy at absolute zero. The mean average difference between U298 and U0 is around 5.3 kcal/mol, the difference between H298 and U0 is around 5.9 kcal/mol, and the difference between G298 and U0 is around 21 kcal/mol. This means that neither the regression nor the neural network models are accurate enough to capture the subtle effect of temperature on the energies.

For the regression, there was little difference in predictions between the four models; molecules that were well predicted (or poorly predicted) in one model were also well predicted (or poorly predicted) by the others. There were many molecules with large errors; over 1500 molecules had a larger than 120 kcal/mol error, though most of the energies are large enough that even this large error is below 1%.

For the neural network, there was more variation in which molecules were well (or poorly) predicted between the four different models. This could be due to the stochastic nature of the optimizer; although the four thermodynamic energies have relatively similar values, because they are optimized using a stochastic algorithm, the different models may learn different weights that still give equally good predictions. There were more molecules with greater than 120 kcal/mol error, about 1800 molecules. Note that the test set for the neural network was also smaller than that of the regression model, so a larger percentage of the molecules have high error. This is not surprising given the higher mean absolute error for the neural network in comparison

There did not seem to be any noticeable trends among molecules that were poorly predicted. However, for both the neural network and the regression there was a strong trend among molecules that were extremely well predicted (MAE < 0.6 kcal/mol): they are not chemically stable. The vast majority of such molecules feature two highly unstable motifs: a three- or four-membered ring. While it is somewhat surprising that such unstable compounds were included in the dataset, they do not stand out from the other data points; they do not have particularly high or low energies, and are not clustered in the training data set. However, these highly strained structures probably all have very similar features in their Coulomb matrices; since the atoms in a three- or four-membered ring are closer together than is typical, the off-diagonal elements should be larger than a standard single bond.

Examining the regression kernel confirms this hypothesis. One can input a molecule to the kernel and it will report what other molecules in the dataset have the most similar input features. For these three- and four-member rings, the kernel outputs very similar looking molecules, indicating that the Coulomb matrix eigenvalues are very similar among this class of molecules. However, for molecules without these features, the molecules identified by the kernel as most similar are actually not similar at all. This suggests that the major source of error in these models is an insufficient input feature choice: while the Coulomb matrix eigenvalues can be good predictors of chemical structure (and thus energy), they often are too simplified and lose important information that was stored in the full Coulomb matrix.

5 Conclusion

I have used two machine learning methods, kernel ridge regression and neural networks, to train models that predict the internal energy at absolute zero (U0), the internal energy at room temperature (U298), the enthalpy at room temperature (H298), and the Gibbs' Free Energy at room temperature (G298). The regression performed better than the neural network, with errors about half as large as those of the neural network. However, both models produced errors below 0.03% on average, making them quite accurate. While the percent error is low, the neural network was unable to improve upon the regression, and the absolute errors are still too large to distinguish between the room-temperature properties and the internal energy at absolute zero.

The errors in this model seem to stem from the use of the Coulomb matrix eigenvalues as input features. Using only the eigenvalues of the Coulomb matrix results in the loss of valuable information about molecular structure, and thus makes it of limited utility in predicting energies. For certain classes of molecules these eigenvalues are good predictors, but for the majority of molecules, the similarity in Coulomb matrix eigenvalues does not indicate a similarity in molecular structure. However, because the order of atoms in a molecular geometry is arbitrary, using the full Coulomb matrix is not an option as one molecule could have several Coulomb matrices associated with it. Perhaps a standardized ordering of atoms in a molecule could be devised, leading to a well-defined way of using the whole Coulomb matrix.

Potential future work could also be done to improve the performance of the neural network. Because it seems the regression model's performance comes from the kernel's measure of similarity between the molecules, one could feed this kernel into the neural network instead of the bare inputs. However, this would greatly increase the number of input features, which would increase the computational cost and introduce potential for overfitting. Another way to increase the representative power of the neural network would be to increase the number of hidden layers or units per layer beyond what is considered here, however, as these quantities increase, more computational power must be used to determine the best architecture and train the network.

References

[1] Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the Kohn-Sham equations with machine learning. *Nature Communications*, 8(1):872, 2017.

[2] Raghunathan Ramakrishnan, Mia Hartmann, Enrico Tapavicza, and O. Anatole von Lilienfeld. Electronic spectra from TDDFT and machine learning in chemical space. *The Journal of Chemical Physics*, 143(8):084111, 2015.

[3] Matthias Rupp, Raghunathan Ramakrishnan, and O. Anatole von Lilienfeld. Machine Learning for Quantum Mechanical Properties of Atoms in Molecules. *The Journal of Physical Chemistry Letters*, 6(16):3309–3313, 2015.

[4] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, 2013.

[5] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O. Anatole von Lilienfeld, Alexandre Tkatchenko, and Klaus-Robert Müller. Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419, 2013.

[6] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Big Data Meets Quantum Chemistry Approximations: The δ-Machine Learning Approach. *Journal of Chemical Theory and Computation*, 11(5):2087–2096, 2015.

[7] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Physical Review Letters*, 108(5):058301, 2012.

[8] Xue-Mei Duan, Zhen-Hua Li, Guo-Liang Song, Wen-Ning Wang, Guan-Hua Chen, and Kang-Nian Fan. Neural network correction for heats of formation with a larger experimental training set and new descriptors. *Chemical Physics Letters*, 410(1):125–130, 2005.

[9] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014.