DOI: https://doi.org/10.46354/i3m.2019.emss.031

FAST APPROXIMATIONS BY MACHINE LEARNING: PREDICTING THE ENERGY OF DIMERS USING CONVOLUTIONAL NEURAL NETWORKS

Dylan Hennessey[a], Mariusz Klobukowski[b], Paul Lu[c]

[a]Department of Medicine
[b]Department of Chemistry
[c]Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada

[a]dhenness@ualberta.ca, [b]mariusz.klobukowski@ualberta.ca, [c]paullu@ualberta.ca

ABSTRACT

We introduce *fast approximations by machine learning (FAML)* to compute the energy of molecular systems. FAML can be six times faster than a traditional quantum chemistry approach for molecular geometry optimisation, at least for a simple dimer. Hardware accelerators for machine learning (ML) can further improve FAML's performance. Since the quantum chemistry calculations show poor algorithmic scaling, faster methods that produce a similar level of accuracy to the more rigorous level of quantum theory are important.

As a FAML proof-of-concept, we use a convolutional neural network (CNN) to make energy predictions on the F_2 molecular dimer system. Training data for the CNN is computed using a quantum chemistry application (i.e., GAMESS) and represented as an image. Using five-fold cross-validation, we find that the predictions made by the CNN provide a good prediction to the theoretical calculations in a fraction of the time.

Keywords: fast approximation, machine learning, molecular geometry optimisation, quantum chemistry

1. INTRODUCTION

Using the mathematical apparatus of quantum physics, quantum chemistry allows for the simulation of molecules, their properties, and chemical reactions in which they participate. From drug design to protein folding, the possibilities are vast. However, it comes with a price tag which is computing time: the Schrödinger equation, which is at the core of quantum chemistry, cannot be solved analytically for any system containing more than one electron, so we must resort to using approximations. To make matters worse, these approximate methods are iterative, with no guarantee of them converging, and even when they do converge there is no guarantee that they have converged to the global minimum of the problem. By far the biggest roadblock is that even the least accurate of a truly quantum theory has a scaling rate of n^4 (n can be loosely thought of as the size of a system, but it might perhaps be better thought of as how accurately we are describing a system). The most accurate level of theory, full configuration interaction, would take an infinite amount of time to compute the first iteration of its solution for just a single helium atom. Modern processors and graphics processing units (GPUs) have helped reduce computing time substantially, but we are still far away from describing large, asymmetrical systems like proteins at the quantum level in any reasonable amount of time. However, there might be a workaround.

Machine-learning (ML) algorithms have been successful in classifying images with high accuracy (Krizhevsky, Sutskever, and Hinton 2012), learning the parameters of data network protocols (Sivaraman *et al.* 2014), and other tasks. The desire for ML performance has spawned specialized hardware (e.g., tensor processing units (TPUs) (Joupii *et al.* 2017)), representing another enticing technology curve for the high-performance computing (HPC) community. As such, it could be beneficial for quantum chemistry if it can make use of ML algorithms.

Most computational quantum chemists are not interested in any arbitrary molecular geometry. They are usually only interested in regions around the most *stable* geometry, which is the one that minimizes the total energy (equal to the sum potential and kinetic energies) of the system. Once the most stable geometry has been found, work can begin on calculating other properties of interest. Algorithm 1 shows the pseudocode for this geometry optimizing process.

We consider an atypical use of ML as part of this larger, generalized molecular geometry optimisation. Instead of using HPC to improve the ML implementation, we consider how ML can improve an HPC computation via *fast approximations by machine learning* (FAML). For example, computing the total energy of a molecule (i.e., lines 3 and 8, Algorithm 1) is both computationally expensive (Schmidt *et al.* 1993) and central to each iteration of the optimisation algorithm. Multiple cores

and general-purpose graphics processing units (GPGPU) (Schmidt *et al.* 1993; Abadi *et al.* 2015) can be used to accelerate the energy calculation, but we consider the use of neural networks as a different kind of accelerator.

Algorithm 1: Generalized Molecular Geometry Optimisation, Pseudocode

```
1.  thresh = minimum change in total energy from one geometry
        state to another
2.  current_geo = guess of optimal geometry
3.  current_energy = total energy of current_geo
4.  old_energy = 0
5.
6.      old_energy = current_energy
7.      update current_geo based on derivative of the total energy
            with respect to the geometry
8.      current_energy = total energy of current_geo
9.  end while
10. return current_energy, current_geo
```

Attempts at this have already been made. von Lilienfeld *et al.* (Montavon *et al.* 2013) made a simple neural network composed of four hidden fully connected (FC) layers and used what they call a "Coulomb matrix" as its input. The Coulomb matrix is given by

$$M_{i,j} = \begin{cases} \frac{Z_i^{2.4}}{2} & \text{if } i = j \\ \frac{Z_i Z_j}{|R_i - R_j|} & \text{if } i \neq j \end{cases} \quad (1)$$

where i and j refer to nuclei, Z_i is the nuclear charge of nuclei i, and $|R_i - R_j|$ is the distance between nuclei i and j. Their network makes predictions on 14 different properties of the system in question. They trained on 5,000 different molecules and tested on 2,211 molecules. The network was able to obtain an accuracy good enough for the predictions to be usable, and that is commendable. However it is worth pointing out that a network made only of FC layers is very simple, and a more complicated type of network could perform much better.

As a proof-of-concept, we show that convolutional neural networks (CNNs) can be used to accurately predict the energy of a molecule with a specific geometry. The CNN-based FAML is six times faster than using a traditional quantum chemistry approach, at least for fluorine dimers (F_2). Quantum chemistry algorithms are still used offline to create the training data for the CNN, training is offline, but when implemented in an actual geometry optimisation calculation, only the pre-trained CNN would need to be used online.

Currently, GPGPUs are used to both train and evaluate the CNNs. For future work, we plan to extend the current results to different dimers, and to larger molecules. Also, in theory, FAML makes it feasible to use specialized ML hardware (e.g., TPUs (Joupii *et al.* 2017)) for higher performance, especially since it is not known how quantum chemistry algorithms can be directly adapted to non-GPGPU accelerators.

1.1. Convolutional Neural Networks (CNNs)

CNNs (LeCun *et al.* 1989) are typically composed of three different layers: convolution layers, pooling layers, and FC layers. How these work on a more technical level can be a bit complex, so we will go over each layer individually, starting with the FC layers.

1) FC Layers: An FC layer is really just a single layer from a multilayer perceptron (Zupan and Gasteiger 1993) [8]. The vector of activations h for the FC hidden layer l is given by the following equation

$$h^l = f\left(\Theta^l h^{l-1} + b^l\right) \quad (2)$$

where Θ^l is a matrix of weights, b^l is a vector of biases, and f is a non-linear function. While the sigmoid function $f(x) = 1/(1 + e^{-x})$ was used extensively in the past, the rectified linear unit (ReLU) function is much more commonly used today. It is given in equation 3.

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (3)$$

The first derivative of a ReLU is 1 if $x \geq 0$ and 0 otherwise. This is far simpler than the first derivative of the sigmoid function, which helps greatly during the training process, especially with large CNNs.

2) Convolutional Layers: The convolutional layers are what allows a CNN to become so accurate. Instead of every neuron using all of the data from the previous layer, a neuron in a convolutional layer uses only a subset of the information from the previous layer. This is given mathematically by

$$h_{i,j}^{l,f} = f\left(\left[\sum_{q=1}^{o}\sum_{c=0}^{m-1}\sum_{d=0}^{n-1}\Theta_{c,d}^{l,f,q} h_{x,y}^{l-1,q}\right] + b^{l,f}\right) \quad (4)$$

where i and j are the coordinates of a neuron, m and n are the dimensions of the filter matrix, $x = i-(m-1)/2+c$, $y = j-(n-1)/2+d$ (here we are assuming that m and n are odd), o is the number of features of the previous layer $l-1$, and f is the feature of the current layer l that we are solving for.

3) Pooling: If we do not stride through the $l-1$ layer, layer l will have the exact same dimensions as layer $l-1$. Typically, we compress the l layer in some way. The most common way of doing this is through *pooling*. This is where we look at one particular neuron and its neighbours and apply some function to them. If we took only the maximum value for a feature of one neuron and its neighbours, this would be *max pooling*. Another common way of doing this is *average pooling*. An example of pooling is as follows. After applying Equation 4, we are left with output that has dimensions of $28 \times 28 \times 3$. For a pooling window of 2×2 and for the

first feature, we would look at the data in $h_{0,0}^{l,1}$, $h_{1,0}^{l,1}$, $h_{0,1}^{l,1}$, and $h_{1,1}^{l,1}$ (notice how pooling is done with respect to a single feature, so we would have to do three different pooling operations for each where *i* and *j* coordinates). In max pooling, we would take the maximum value for these data points, and this would become the value of $h_{0,0}^{l_p,1}$ where l_p is the pooled layer of layer *l*. We would stride through the un-pooled layer in steps equal to the dimension of the pooling window (so $h_{0,1}^{l_p,1}$ would be the max of $h_{0,2}^{l,1}$, $h_{1,2}^{l,1}$, $h_{0,3}^{l,1}$, and $h_{1,3}^{l,1}$ and so on) until we have gone through the whole of layer *l*. The output of this pooling operation would compress the input by a factor of four, so the output would have dimensions of 14 × 14 × 3.

Pooling can be thought of as "summarizing" the features of each group of neurons. This allows for layers to specialize the kind of features they look for, and also allows for deeper layers to look for more complex features without needing to greatly expand the window they look at. For instance, the first layer of a CNN might only look for curves and straight lines, the next layer might look for basic shapes like circles, squares, and triangles, and the deepest layers would look for things like faces and other objects.

2. METHODS

The overall goal of this work is to make a CNN that can accurately predict properties of a chemical system. As a first step, we designed a CNN that can predict the total energy of the system, the kinetic energy of the electrons, the potential energy of attraction between electrons and nuclei, the potential energy of repulsion between electrons, and the potential energy of repulsion between nuclei. We chose these because optimizing the geometry of a chemical system requires finding a molecular geometry (i.e., coordinates of atoms in a molecule) where the total energy is at a minimum. This adds a further layer of iterative calculations to an already computationally difficult task and it would be good if there was a method of placing atoms in sensible locations beforehand. Because CNNs work best at image recognition, we need some way to turn a molecule into something that resembles an image. We achieve this in the following way.

2.1. Digitizing Chemical Space

First, we begin by choosing the size of the dimensions of the image to be 512 × 512 × 3. Each pixel of the image represents a $(0.01\text{Å})^2$ area of space. Note that this restricts us to two-dimensional chemical systems. An area of $(0.01\text{Å})^2$ was chosen because changes in molecular geometry smaller than this do not have a strong impact on the properties we want to predict. An image size of 512 × 512 was chosen because this gives us enough room to dissociate dimers, and also to optimize performance on GPUs.

At a simple level, the energy of a molecular system is a function of the nuclear charges present, and the coordinates in space that the nuclei occupy. Everything else can be derived from just these inputs. Because of this, the features of a pixel should at least include the nuclear charge of the nuclei occupying that pixel. For a dimer system, this would produce an image that is completely black except for two white pixels. Even though in theory this should be all that is needed, we are not giving the CNN much to work with. Therefore, we decided to include some additional information to help point the CNN in the right direction.

We thought that it might be helpful to include some information about the attractive and repulsive forces. We did this by including a simplified version of the bonding orbital that would form between the 1s electrons in the system, as well the anti-bonding orbital that would form. These were calculated from Gaussian type orbitals, given by the following equations

$$I_{i,j,2} = \sum_{k=1}^{n} N_k e^{-Z_k r_{ij}^2} \tag{5}$$

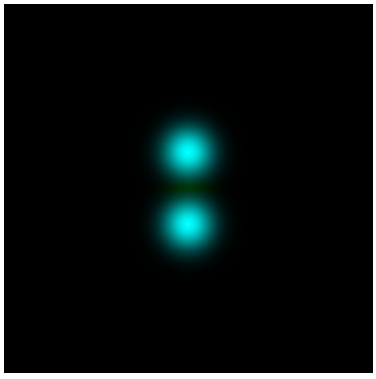
$$I_{i,j,3} = \frac{1}{2} \sum_{k=1}^{n} \sum_{l=1}^{n} |N_k e^{-Z_k r_{ij}^2} - N_l e^{-Z_l r_{ij}^2}| \tag{6}$$

where $I_{i,j,x}$ is the x^{th} channel of the pixel at *i,j*, *n* is the number of nuclei in the system, r_{ij} is the distance of the pixel *i,j*, from nuclei *k*, and N_k is a normalization constant that satisfies

$$N_k^2 \int_0^\infty r^2 |e^{-Z_k r^2}|^2 dr = 1 \tag{7}$$

An example of what these images look like is shown in Fig. 1.

Figure 1: The image produced by the Equations 5, 6, and 7. The atoms are F. The green channel is assigned Equation 5, the blue channel is assigned Equation 6, and the red channel is assigned the Z of the nuclei. Note that the image has been normalized such that the maximum value of any channel is 255 and the minimum is 0.



As for the actual training data, we use the results of GAMESS (Schmidt *et al.* 1993) calculations at the CASSCF (Siegbahn *et al.* 1981; Roos, Taylor, and Siegbahn 1980; Siegbahn *et al.* 1980) level of theory and the 6-311+G(2d,2p) basis set (Krishnan *et al.* 1980; Clark *et al.* 1983; Frisch, Pople, and Binkley 1984) on F_2 dimers. The dimers have randomly generated coordinates, so we can be reasonably sure that there are no exact duplicates.

2.2. CNN Architecture

As mentioned, the input layer for our CNN has dimensions of 512 × 512 × 3. We chose a filter with dimensions of 5 × 5 for each hidden layer. The first hidden layer takes in the 3 channels from the input layer, and outputs 32 new features. The value of 32 was chosen for optimisation of the calculations on GPUs. The first hidden layer will have 5×5×3×32 weights and 32 biases. The output of this layer goes through an average pooling operation with a 2 × 2 window and a stride of 2. The output after pooling has dimensions of 256×256×32. Because spatial information is so important to these calculations, each subsequent layer pulls out twice the number of features than the layer before it, in the hopes that this will preserve enough of the information that would get compressed by pooling. In total, there are five convolutional layers and five average pooling operations. The output of the last layer has dimensions of 16 × 16 × 512. This then gets reshaped into a vector with 131072 dimensions, and then goes through the FC layers. There are two FC layers, the first has a weight matrix with dimensions of 131072 × 2048, and the second has dimensions of 2048 × 2048.

In a preliminary study (unpublished), we found that the predictions of total energy were more accurate if we also trained the CNN to predict the electron kinetic energy, the electron-electron repulsion potential energy, the electron-nuclei attraction potential energy, and the nucleus-nucleus repulsion potential energy at the same time. Therefore, the output layer has a weight matrix with dimension of 2048 × 5 and the outputs are the five energy types just described. ReLUs are used for all layers except the output layer, which uses no activation function. A dropout of 50% is also included between the two FC layers, and the output.

2.3. Initialization and Training

The weights and biases are initialized with random numbers from a normal distribution with zero mean and a standard deviation of 0.1. We use the mean absolute error (MAE) as the error function, and we optimize the error using the Adam optimisation algorithm (Kingma and Ba 2014) and a batch size of 32. We performed some initial calculations using 80% of the data to train on and the remaining 20% for testing. We used 5 fold cross-validation for more in-depth experiments. Each fold was constructed by sorting the data by distance between dimers, and then distributing them to each fold round-robin style.

3. EMPIRICAL RESULTS AND DISCUSSION

We generated 100,000 systems ranging in distance from 1.00Å−7.24Å. We found that the energy was skewed in favour of the lower energies, so we added approximately 90,000 more examples that ranged from 1.00Å−1.50Å. We then normalized the energies with the following equation

$$e' = \frac{e - e_{\min}}{e_{\max} - e_{\min}} \tag{8}$$

where e′ is the normalized value of energy e, and e_{max} and e_{min} are the maximum and minimum values for the energies of the same type as e. This uniformly scales all energies of the same type between zero and one. This helps to reduce the training time, as the CNN does not have to spend as much time trying to find the range of the data. Even so, it can take as long as approximately 40 hours to complete 14 epochs for the F_2 CNN. The results of training for over 14 epochs are given in Table 1.

We can see that the results between folds are all similar. The largest deviations for each energy type is given in Table 2. Scatter plots showing how the predictions line up with their actual values, as well as how the predictions match up to the energy curves are in Figures 2 and 3.

Table 1: The MAE when predicting the total energy (TE), the kinetic energy of electrons (EKE), the potential energy of attraction between electrons and nuclei (ENPE), the potential energy of repulsion between electrons (EEPE), and the potential energy of repulsion between nuclei (NNPE) of a F_2 dimer. Energy is given in Hartree units. Folds 0 to 2 had 37,896 images. Folds 3 and 4 had 37,895 images.

Fold	TE	EKE	ENPE	EEPE	NNPE
0	0.00177	0.0118	0.449	0.207	0.213
1	0.00141	0.0093	0.287	0.163	0.178
2	0.00138	0.0097	0.345	0.219	0.220
3	0.00143	0.0100	0.283	0.209	0.200
4	0.00171	0.0116	0.438	0.195	0.220

Table 2: Largest absolute deviations (LAD) for the predictions of the total energy (TE), the kinetic energy of electrons (EKE), the potential energy of attraction between electrons and nuclei (ENPE), the potential energy of repulsion between electrons (EEPE), and the potential energy of repulsion between nuclei (NNPE) of a F_2 dimer. Energy is given in Hartree units. Distances are given in Å.

Energy Type	LAD	Actual Energy of LAD	Distance of LAD
TE	0.0621	-198.534	1.0008
EKE	0.3181	200.631	1.0024
ENPE	3.2617	-560.249	1.0309
EEPE	2.0255	120.571	1.0066
NNPE	2.2541	42.581	1.0066

Finally, we discuss how the CNN is six times faster than GAMESS in predicting the energy of the dimer molecule. As there is a reduction in the accuracy, there would be little point in using this method if the predictions took longer than using the actual GAMESS program. To make a prediction, the image must first be generated, then the image must be converted into a form TensorFlow can use, and then the data must flow through the CNN.

For the nearly 190,000 molecules used in our experiment, the average time for generating the images is 0.068s per image, the average time for converting the data for TensorFlow is 0.117s per image, and the average time for running the data through the CNN is 0.028s per image. This gives a total time of 0.213s per image. The average time per F_2 calculation using GAMESS is 1.227s. This means that the CNN is almost six times faster. It is also worth pointing out that for anything that will fit into the dimensions of the image we use should require approximately the same amount of time for the CNN to predict, while the time need to calculate larger systems will increase dramatically.

4. CONCLUDING REMARKS

The HPC, ML, and computational science communities are interacting in a variety of interesting ways. First, well-known HPC techniques such as GPGPUs and specialized hardware are improving the performance of ML and scientific computations. Second, ML techniques such as the fast approximations by machine learning (FAML) strategy introduced here, can potentially replace an expensive computation (e.g., quantum chemistry) with a faster ML computation, and stay within acceptable limits of accuracy and error. Using FAML for molecular geometry optimisation (Algorithm 1), the expensive quantum chemistry calculation can be done offline, the ML training can be done offline, so that the cheaper CNN evaluation is all that is done online. Third, FAML provides (in theory) a feasible way to take advantage of a technology trend towards specialized hardware for ML (e.g., TPUs (Joupii *et al.* 2017)) that is less obvious for, say, traditional computational quantum chemistry algorithms. Opportunities to exploit these interactions may provide significant performance benefits, such as the six-fold increase in performance of the CNN-based FAML, for a dimer.

While this study is mostly a proof-of-concept, its findings are promising. We have shown that CNNs offer a viable method of shortening the time needed to predict the energy of a simple molecular system. The predictions made by our CNN have an average of only 0.001 MAE with respect to calculations performed by GAMESS, and are also much faster to obtain. We have shown that we can increase the accuracy of predictions by simultaneously making predictions of related data. We have also shown that generating the training data by randomly generating geometries of molecular systems is valid, so long as the level of theory can dynamically describe the breaking of electron pairs.

In a future study, we will design a CNN that is able to make predictions about dimers with different atoms. It would also be interesting to see if the CNN can make accurate predictions about molecules it has never seen before. For instance, we might train the CNN on the molecules H_2, F_2, Cl_2, FCl, and HF, and then see if the CNN can accurately make predictions on HCl, which is outside of the training set. Finally, we plan on empirically evaluating a CNN-based FAML in an actual, end-to-end, molecular geometry optimisation computation.

ACKNOWLEDGMENTS

This research was funded by the University of Alberta and Natural Sciences and Engineering Research Council (NSERC) of Canada. The NSERC Discovery Grants to Mariusz Klobukowski and Paul Lu are gratefully acknowledged. This research was enabled in part by support provided by Compute Canada (http://www.computecanada.ca).

REFERENCES

Abadi M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

Clark T., J. Chandrasekhar, G. Spitznagel, and P.V. Schleyer, "Efficient diffuse function-augmented basis sets for anion calculations. III. The 3-21+G basis set for first-row elements, Li–F," *J. Comp. Chem.*, vol. 4, pp. 294–301, 1983.

Frisch M.J., J.A. Pople, and J. Binkley, "Self-consistent molecular orbital methods 25. Supplementary functions for Gaussian basis sets," *J. Chem. Phys.*, vol. 80, pp. 3265–3269, 1984.

Joupii N., C. Young, N. Patil, and D. Patterson, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA 2017, Toronto, ON, Canada, June 24-28, 2017*, 2017, pp. 1–12.

Kingma D.P. and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

Krishnan R., J. Binkley, R. Seeger, and J.A. Pople, "Self-consistent molecular orbital methods. XX. A basis set for correlated wave functions," *J. Chem. Phys.*, vol. 72, pp. 650–654, 1980.

Krizhevsky A., I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

LeCun Y., B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989. [Online]. Available: http://dx.doi.org/10.1162/neco.1989.1.4.541

Montavon G., M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Moller, and O.A. von Lilienfeld, "Machine learning of molecular electronic properties in chemical compound space," *New Journal of Physics*, vol. 15, no. 9, p. 095003, 2013. [Online]. Available: http://stacks.iop.org/1367-2630/15/i=9/a=095003

Roos B.O., P. Taylor, and P.E.M. Siegbahn, "A complete active space SCF method (CASSCF) using a density matrix formulated super-CI approach," *Chem. Phys.*, vol. 48, pp. 157–173, 1980.

Schmidt M.W., K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, and J.A. Montgomery, "General atomic and molecular electronic structure system," *J. Comput. Chem.*, vol. 14, no. 11, pp. 1347–1363, 1993.

Siegbahn P.E.M., J. Almlof, A. Heiberg, and B.O. Roos, "The complete active space SCF (CASSCF) method in a Newton-Raphson formulation with application to the HNO molecule," *J. Chem. Phys.*, vol. 74, no. 4, pp. 2384–2396, 1981.

Siegbahn P.E.M., A. Heiberg, B.O. Roos, and B. Levy, "A comparison of the super-CI and the Newton-Raphson scheme in the complete active space SCF method," *Phys. Scr.*, vol. 21, pp. 323–327, 1980.

Sivaraman A., K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 479–490.

Zupan J. and J. Gasteiger, *Neural Networks for Chemists: An Introduction*. New York, NY, USA: John Wiley & Sons, Inc., 1993.

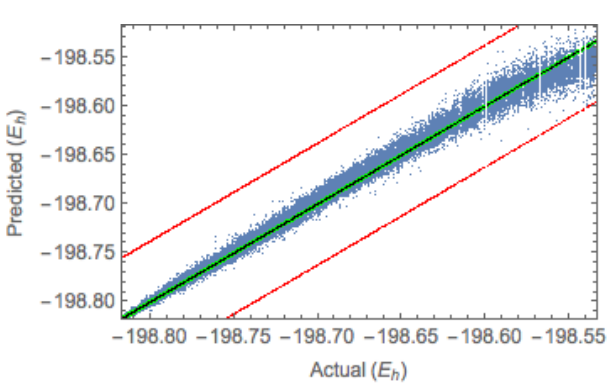
AUTHORS BIOGRAPHY

Dylan Hennessey completed an M.Sc. in Chemistry at the University of Alberta (2018), under the supervision of Mariusz Klobukowski. He is currently engaged in bioinformatics research in the Department of Medicine.

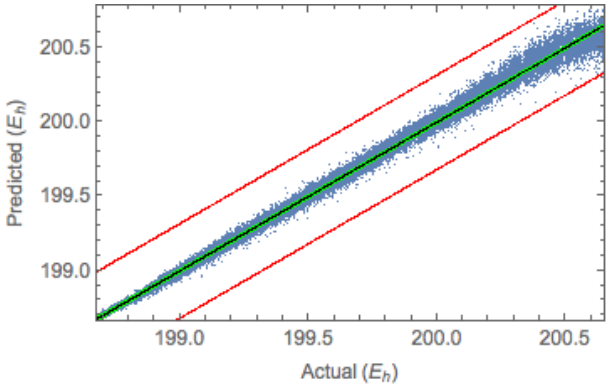
Mariusz Klobukowski is a Professor of Chemistry, Department of Chemistry, University of Alberta. His research program is focused on the development and applications of methods for accurate computational studies of electronic structure, geometry, vibrational spectra, reaction mechanisms, and one-electron properties of organometallic molecules, molecular ions, and molecular clusters in their ground and excited electronic states.

Paul Lu is a Professor of Computing Science, Department of Computing Science, University of Alberta. His research interests are in the area of software systems, including parallel and distributed systems, operating systems, file systems, high-performance computing, and network protocols. He is also one of the co-developers of the "Problem Solving, Python Programming, and Video Games" (PVG) (2018) massive open online course (MOOC) on the Coursera platform.

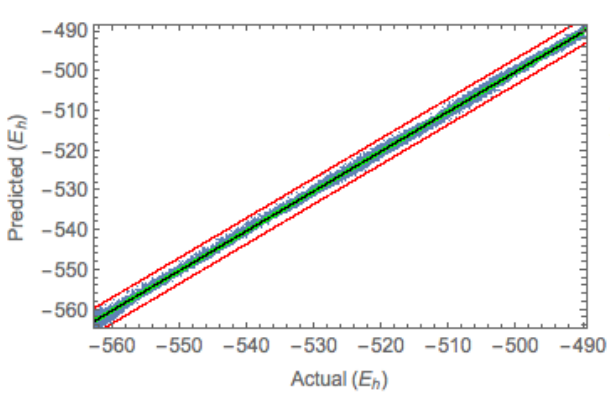
Figure 2: Scatter plots made from the predictions of the total energy (a), kinetic energy (b), electron-nuclei potential energy (c), electron-electron potential energy (d), and nuclei-nuclei potential energy (e). Results from all folds for the F_2 CNN are shown. Energies are given in Hartree units. The black diagonal line in each plot shows where perfect predictions lie. The green line shows the range of predictions that fall within the MAE. The red line shows the range of predictions that fall within the largest error.



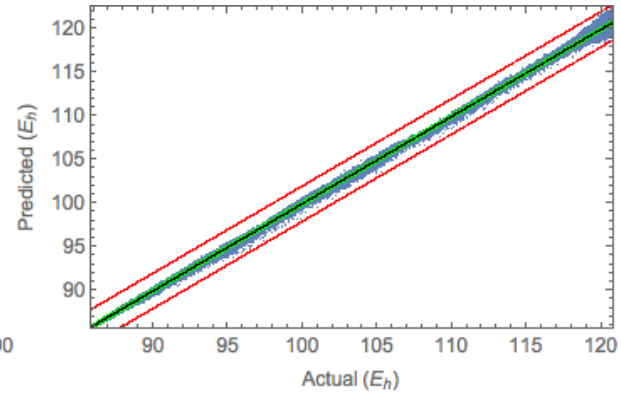
(a)



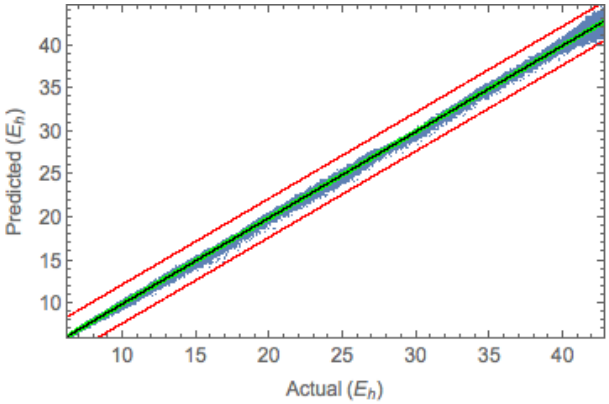
(b)



(c)

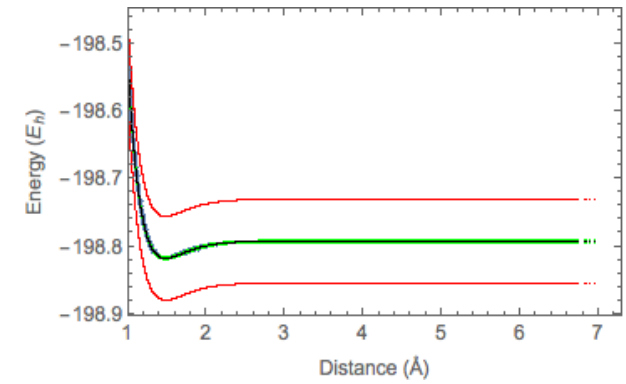


(d)

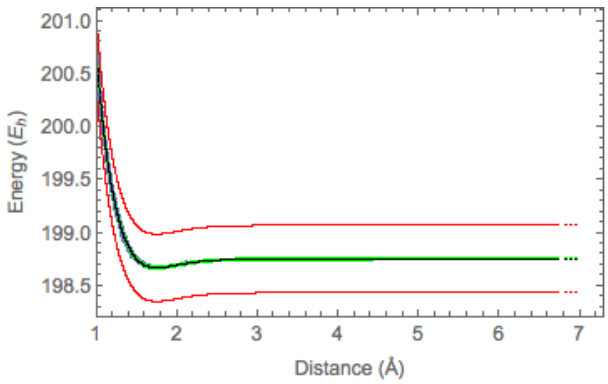


(e)

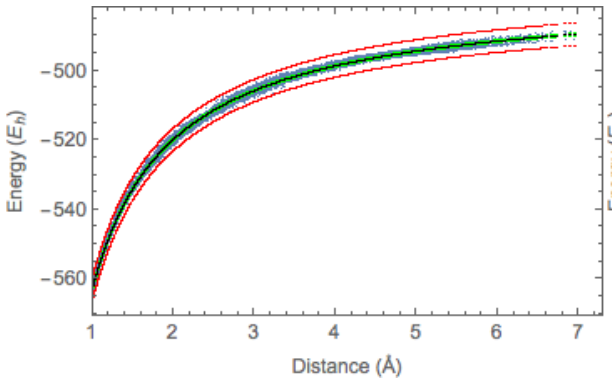
Figure 3: The predictions of the energy curves as a function of the distance between dimers for the total energy (a), kinetic energy (b), electron-nuclei potential energy (c), electron-electron potential energy (d), and nuclei-nuclei potential energy (e). Results from all folds for the F_2 CNN are shown. Energies are given in Hartree units and distances are given in Å. The black points in each plot show where perfect predictions lie. The green points show the range of predictions that fall within the MAE. The red points show the range of predictions that fall within the largest error.



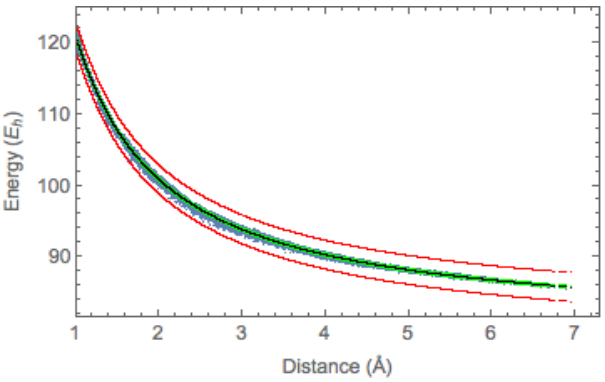
(a)



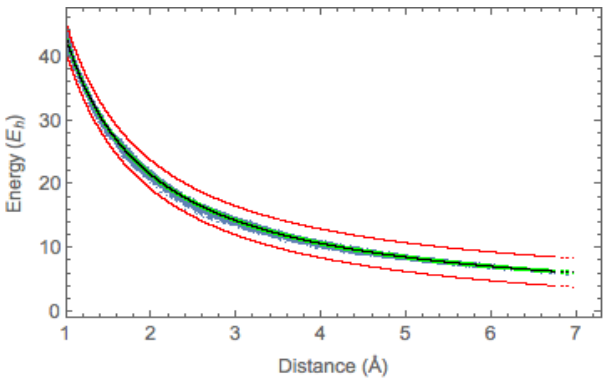
(b)



(c)



(d)



(e)