Building potential energy surfaces of heteroaromatic compounds using neural networks

Second Research Project

November 5, 2021

Author:

Joël Benninga

Supervisors:

prof. dr. Shirin Faraji

Albert Thie, MSc

Zernike Institute for Advanced Materials

Theoretical and Computational Chemistry group

ABSTRACT

Machine learning is on its way to revolutionize the field of computational chemistry. In recent years it has made remarkable progress in many areas due to advances in the development of machine learning algorithms and improved hardware resources. Applications of machine learning in the field of computational chemistry include molecular dynamics, Monte Carlo simulations, and more specifically in drug design and material screening. The goal of this research project is to explore the application of machine learning for the building of potential energy surfaces of heteroaromatic compounds, as prerequisite for excited-state quantum dynamics. The potential energy surface is the most basic quantity to describe a chemical system, and determines all of its properties. Although quantum mechanical methods exist to accurately calculate potential energy surfaces, these are computationally expensive. Therefore, neural networks – a class of machine learning algorithms – have recently emerged as a promising alternative for the construction of potential energy surfaces.

In this project, three neural networks are optimized by grid search, trained on surface hopping dynamics simulations, and finally applied to predict potential energy surfaces of pyrazine, pyrrole and furan. Grid search allows the selection of optimized neural networks, which are subsequently trained with the goal of obtaining a training loss below 0.1 eV. This goal is only achieved for the pyrazine neural network, although the pyrrole and furan neural networks are not far off. Finally, the performance of the trained neural networks is assessed using test datasets, *i.e.* surface hopping simulations that are not evaluated in training. Overall, the three models have a test error, *i.e.* a mean prediction error on the testing dataset, below 1 eV. The pyrazine model is the most accurate, and the pyrrole model the least, which has a significant bias towards underprediction.

This research can be continued by improving the accuracy of the neural networks. The most progress would likely be made with hyperparameter tuning. Another approach would be to increase the size of the datasets of both training and testing data. Neural network training on more data should lead to an increase in prediction accuracy. A validation step during the training phase could prove beneficial as well. The research can also be expanded upon by investigating other heteroaromatic molecules, such as thiophene or indole.

In conclusion, machine learning appears to be a very useful and accurate tool for the building of potential energy surfaces of heteroaromatic compounds. The application of neural networks for electronic energy calculations would save a significant amount of computational time, meaning that research could be conducted much faster, at lower cost.

PREFACE

This research project was carried out at the University of Groningen within the Theoretical and Computational Chemistry group between May and October 2021. I wish to thank prof. dr. Shirin Faraji for her supervision and for giving me the opportunity to conduct this research project in her group. I also want to thank Albert Thie for guiding me throughout this project, and Max Menger for helping with PySurf. Lastly, I want to thank all group members for the interesting talks and discussions.

Hardenberg, 18th of October, 2021

Joël Benninga

Contents

ABSTRACT ..... **I**

PREFACE ..... **II**

1 INTRODUCTION **1**

1.1 Neural networks 1

1.2 Construction of potential energy surfaces with neural networks 4

1.3 Goal 6

2 COMPUTATIONAL DETAILS **8**

2.1 Electronic structure calculations 8

2.2 Surface hopping simulations 8

2.3 Neural networks 8

3 RESULTS AND DISCUSSION **10**

3.1 Grid search 10

3.2 Model training 12

3.3 Model testing 14

4 CONCLUSION AND OUTLOOK **17**

5 REFERENCES **19**

6 SUPPORTING INFORMATION **23**

6.1 Neural network 23

6.2 Grid search results 24

1 INTRODUCTION

Machine learning is emerging as the next big leap in computational chemistry. In recent years it has made remarkable progress in many areas due to advances in the development of machine learning algorithms and improved hardware resources.[1] Applications of machine learning in the field of computational chemistry include molecular dynamics[2], Monte Carlo simulations[3], and more specifically in drug design and material screening.[4]

Under certain conditions, such as sufficient training data, machine learning algorithms can predict physicochemical properties many orders of magnitude faster than traditional computational chemistry methods, without loss of accuracy.[1] These properties include Potential Energy Surfaces (PES)[5], excited states[6], forces[7], spectroscopic quantities[8], and reaction pathways.[9] The PES is the most basic quantity to describe a chemical system, and determines all its properties. It is defined as a function that yields the potential energy of a molecular system with respect to its atomic coordinates. Information about the PES can be obtained by solving the Schrödinger equation. Many Quantum Mechanical (QM) methods exist to very accurately solve this equation. However, due to the high computational costs only the most efficient methods, such as Density Functional Theory (DFT)[10] can be used. Therefore, Neural Networks (NN), a class of machine learning algorithms, have recently emerged as a promising alternative for the construction of PESs.[11]

1.1 Neural networks

Although inspired by neurons in the brain, NNs in machine learning are far from the reality of how the brain works. NNs consist of layers of computational functions called neurons that are connected to other neurons in other layers. Starting from an initial input layer that receives the data, each layer's output is the subsequent layer's input.[12]

One type of NN that is relevant for the construction of PESs is a feed-forward NN (Figure 1). In this example the NN is composed of four layers. Between the input and output layers are two additional ones, often referred to as hidden layers. The input layer consists of four neurons, each associated with an atomic coordinate. These neurons are all connected to the neurons in the first hidden layer by weight parameters, portrayed as arrows. Weight parameters, or weights, are values which represent the stored knowledge of the NN. Weights are used to calculate the values associated with neurons in the hidden layers and output layer. First, the output of the first hidden layer is calculated, which then serves as input for the second. Finally, the output of the second hidden layer is transformed to potential energy in the neuron of the output layer. The potential energy obtained from the NN can then be compared with the corresponding energy from QM calculations. The goal of the so-called training process is to minimize this difference.[11,12]

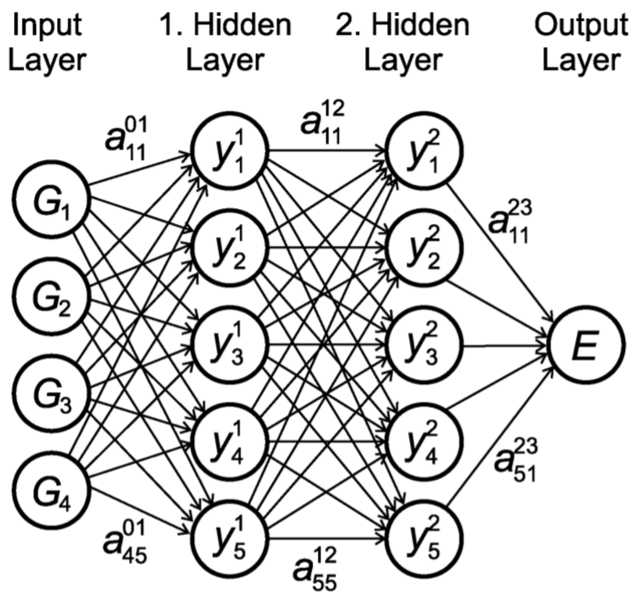


Figure 1: An example of a feed-forward neural network transforming the four atomic coordinates (G_1 to G_4) to energy (E) by passing through two hidden layers. The neurons in each layer are connected to neurons in adjacent layers by weight parameters (a). Figure reproduced from reference 11.

The NN training process is based on solving an optimization problem in which the NN model weights are constantly modified to find a solution that is in good accordance with the training data. First, the NN ‘predicts’ the output in a forward pass. Then the error, *i.e.* the difference between prediction and observation, propagates back through the NN in a backward pass, where the NN weights are updated to minimize this error. After training, the NN can be used as a black box that takes molecular coordinates as input, and provides potential energy as output.[12]

The training process is iterative and a stochastic gradient descent algorithm, such as Adam[13] is typically used. First, the total error, *E*, is computed with

$$E = \frac{1}{2}\sum_{p}\sum_{j}\left(y_j - t_j\right)^2 \tag{1}$$

where *p* is an index over input-output pairs, *j* is an index over output neurons, *y* is the predicted output and *t* is the target output. To minimize *E* by gradient descent, the partial derivative of *E* with respect to each weight in the NN is required. This is simply the sum of the partial derivatives for each of the input-output pairs. For a given pair, the partial derivatives of the error with respect to each weight is computed in the backward pass. The backward pass starts by calculating *∂E / ∂y* for each output neuron. Differentiating equation (1) for a given input-output pair gives

$$\frac{\partial E}{\partial y_j} = y_j - t_j \tag{2}$$

Then the chain rule can be applied to calculate *∂E / ∂x*

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dx_j} \tag{3}$$

Here x_j reflects the total input value of the neurons in the penultimate layer to an output neuron, and d*y* / d*x* depends on the type of activation function[14] used in the NN. Equation (3) can be used to determine how a change in the neurons in this layer will affect the error. Since the total input, x_j, is a linear function of the weights per

$$x_j = \sum_i y_i w_{ji} \tag{4}$$

where w_{ji} is a weight from x_j to y_i, it is possible to compute how the error will be affected by changing the weights, with

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}}$$

$$= \frac{\partial E}{\partial x_j} \cdot y_i \tag{5}$$

For the output of the *i*th neuron, the contribution to $\partial E / \partial y_i$ resulting from the effect of *i* on *j* is

$$\frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial y_i} = \frac{\partial E}{\partial x_j} \cdot w_{ji} \tag{6}$$

Taking into account all connections between output neuron y_i and input neurons x_j, computing $\partial E / \partial y_i$ is possible with

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} w_{ji} \tag{7}$$

Now *∂E / ∂y* can be computed for any neuron in the penultimate layer when given *∂E / ∂y* for all output neurons. This procedure can be repeated to compute the term for earlier layers, while simultaneously computing ∂E / ∂w. Finally, gradient descent can be used to change each weight by an amount proportional to ∂E / ∂w[15]

$$\Delta w = -\alpha \frac{\partial E}{\partial w} \tag{8}$$

where the learning rate, α, is typically a value between 0 and 1.

A commonly used algorithm for stochastic gradient descent is Adam. Adam is an adaptive gradient descent algorithm, meaning that it computes individual learning rates for each weight of the NN. Adam uses momentum by taking into consideration exponentially weighted averages of gradients to converge towards minima more rapidly and avoid local minima.[13]

1.2 Construction of potential energy surfaces with neural networks

In literature there have been quite some reports on utilising NNs for the construction of PESs. For example, Smith *et al.* demonstrated how a deep NN trained on DFT calculations can learn an accurate and transferable potential for organic molecules. Their NN, which was named ANAKIN-ME (Accurate NeurAl networK engINe for Molecular Energies) or ANI-1 for short, was trained on a data set of small organic molecules of up to eight heavy atoms. They then showed its applicability to much larger systems of 10-24 heavy atoms, including well known drug molecules (Figure 2). The ANI-1 potential correctly predicts the stability of these structures and captures the large conformational changes. Additionally, the NN accurately produced the shape and the smoothness of the PESs corresponding to bond stretching, angle bending, and dihedral rotations. Furthermore, the ANI-1 proved to be more accurate against the reference DFT level of theory than DFTB and PM6, two of the most widely used semi-empirical QM methods.[5]

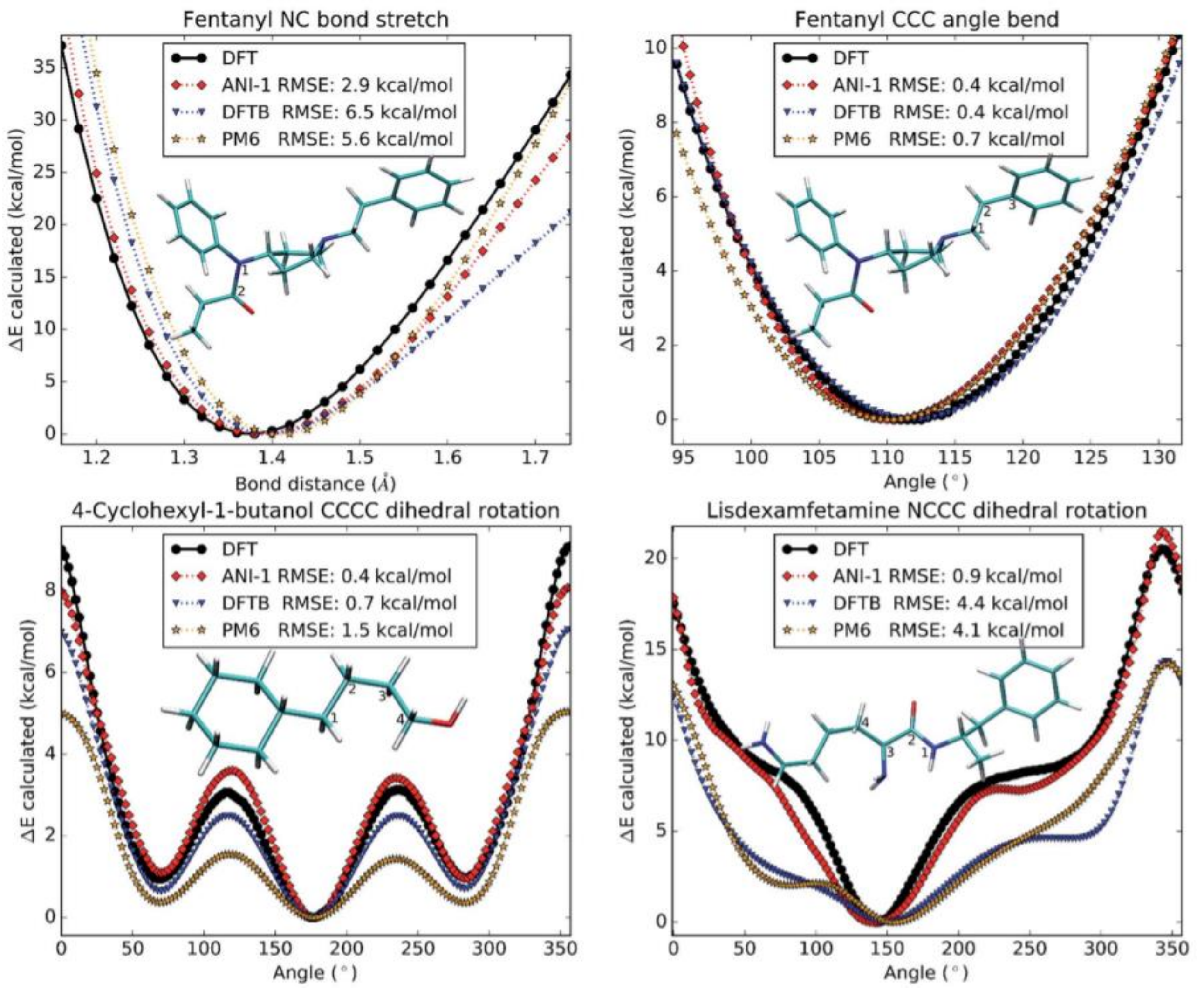


Figure 2: One-dimensional potential surface scans generated from DFT, the ANI-1 potential, and two popular semi-empirical methods, DFTB and PM6. The atoms used to produce the scan coordinate are labelled in the images of the molecules in every subplot. Each figure also lists the RMSE, in the legend, for each method compared to the DFT potential surface. Figure reproduced from reference 5.

Several NNs have been developed to improve upon ANI-1. The ANI-1x potential outperformed ANI-1 on multiple benchmarks, while using only a quarter of the initial data. Additionally, ANI-1ccx predicted a variety of organic properties for isolated organic molecules more accurately than DFT.[16] Their most recent NN, ANI-2x, was an extension of ANI-1x incorporating three additional atomic elements: S, F, and Cl.[17]

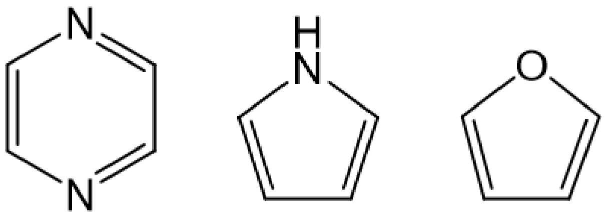
NNs have also been used for the construction of PESs for chemical reactions. For instance, Liu *et al.* used the Permutation Invariant Polynomial-NN (PIP-NN)[18] method to produce a PES for the reaction $OH + HO_2 \rightarrow H_2O + O_2$. The PES was fitted with a two-hidden layer PIP-NN to 108,000 CCSD(T)[19] energies with a Root Mean Square Error (rmse) of 12.6 meV.[20]

Zhang and co-workers used the fundamental invariant (FI)[21] NN method to construct a PES for the reactions $H + H_2O_2 \rightarrow H_2 + HO_2$ and $H + H_2O_2 \rightarrow OH + H_2O$. The PES was fitted with a two-hidden layer FI-NN to roughly 114,000 CCSD(T) energies with an rmse of 5.7 meV, using 26 FIs as input vector.[22]

Prediction of spectroscopic properties with NNs has been done as well. Allouche *et al*. developed NNs to reproduce the PESs and dipole mapping of 11 polycyclic aromatic hydrocarbons. Three different NNs composed of two hidden layers of 15, 20, and 30 neurons per layer were trained by using a database including around 8,900 B3LYP[23]/N07D[24] energies with an rmse of 0.4-0.7 meV. Infrared (IR) frequencies and intensities were predicted using the trained NNs. Overall, they found that the NN predictions lead to good agreement with IR fundamental frequencies both simulated by DFT and obtained in experiments.[25]

1.3 Goal

Heteroaromatic compounds, which share structural similarities with polycyclic aromatic hydrocarbons, are invaluable building blocks for pharmaceutical and synthetic chemistry and materials science.[26] These compounds contain heteroatoms (*e.g.* O, N) within the aromatic ring structure. The goal of this research project is to explore the application of NNs for the building of PESs of three heteroaromatic compounds: pyrazine, pyrrole, and furan (Figure 3). The main reasons for selecting these molecules, from a computational point of view, are 1) the PESs of these molecules are already available using highly accurate electronic structure methods, and 2) full quantum excited-state dynamics are available that make the benchmarking and validation aspects of the NNs feasible.



1 2 3

*Figure 3: Chemical structures of heteroaromatic compounds, **1**: pyrazine, **2**: pyrrole, **3**: furan.*

In the next chapter the computational details are described. In Chapter 3 the results can be found followed by the conclusion, discussion and outlook in Chapter 4. The references are listed in Chapter 5, and the report ends with the supporting information in Chapter 6.

2 COMPUTATIONAL DETAILS

All computations were performed on the Nieuwpoort or Theochem high-performance computing clusters on compute nodes equipped with 2 CPUs with 14 cores each, 1 TB disk and 128 GB RAM.

2.1 Electronic structure calculations

Optimized electronic structures of the heteroaromatic compounds were calculated in order to perform trajectory surface hopping simulations[27] with the PySurf framework.[28] The equilibrium geometry of pyrazine, its frequencies and normal mode displacements were obtained by applying DFT for the ground state and time-dependent DFT[29] for the excited states using the B3LYP functional[23] and Pople's 6-31G* basis set[30], as implemented in the Q-Chem program package.[31] For pyrrole and furan the MP2 functional[32] and aug-cc-pVDZ basis set[33] were used to perform the same calculations. For these compounds the surface hopping simulations failed to converge more often at the B3LYP/6-31G* level of theory.

2.2 Surface hopping simulations

The PySurf framework[26] was used to simulate standard non-adiabatic surface hopping dynamics. This was done by propagating 100 trajectories – whose initial geometry and velocity were based on a Wigner sampling algorithm[34] – for 100 fs with a time step of 0.5 fs. The second excited adiabatic state of the models was chosen as initial state for the trajectories. At each point along the trajectories the energies and gradients of each state were stored in the database, along with their respective geometries.

2.3 Neural networks

The NNs were constructed using the PyTorch framework[35]. As an example, the architecture of the NN used for furan is presented in the supporting information (Chapter 6.1, Figure 10). The NN is essentially a class consisting of several modules. It is defined by subclassing 'nn.Module', which is the base class for all NN modules. The NN layers are initialized in '__init__', which sets up the structure of the NN. The operations performed on the data are implemented in the 'forward' method. The 'nn.Flatten' layer converts input into a contiguous array. Thereafter, the data is passed through the modules within ‘nn.Sequential’ sequentially. The ‘nn.Linear’ module applies a linear transformation on the input, whereas ‘nn.ReLU’ applies the rectified linear unit function, which introduces a cut-off function at 0. The NN in this example contains three hidden layers of 80 neurons, as seen by the ‘nn.Linear(80, 80)’ line, repeated thrice.

The models were trained using a mean squared error loss function, along with the Adam[13] optimization algorithm. The training function can be found in the supporting information (Chapter 6.1, Figure 11). A loop was used to call the training function over a specified number of iterations (epochs). In a single epoch, the model made predictions on the full training dataset in batches of 64, and the error was propagated back to adjust the model’s parameters.

3 RESULTS AND DISCUSSION

In this chapter the results concerning the grid search, training, and testing of the models are reported. Finally, the prediction accuracy of these models is described.

The first step of the project was to simulate surface hopping dynamics by propagating trajectories along PESs. Figure 4 shows the PESs of pyrazine as functions of time of a single trajectory. For each molecule, *i.e.* pyrazine, pyrrole and furan, 100 trajectories were propagated, with the goal of using 90 for training the NNs, and the remaining 10 for testing. However, due to convergence failures in the dynamics simulations of pyrrole and furan, 44 training trajectories and five testing trajectories were produced for the former, while this was 77:9 for the latter.

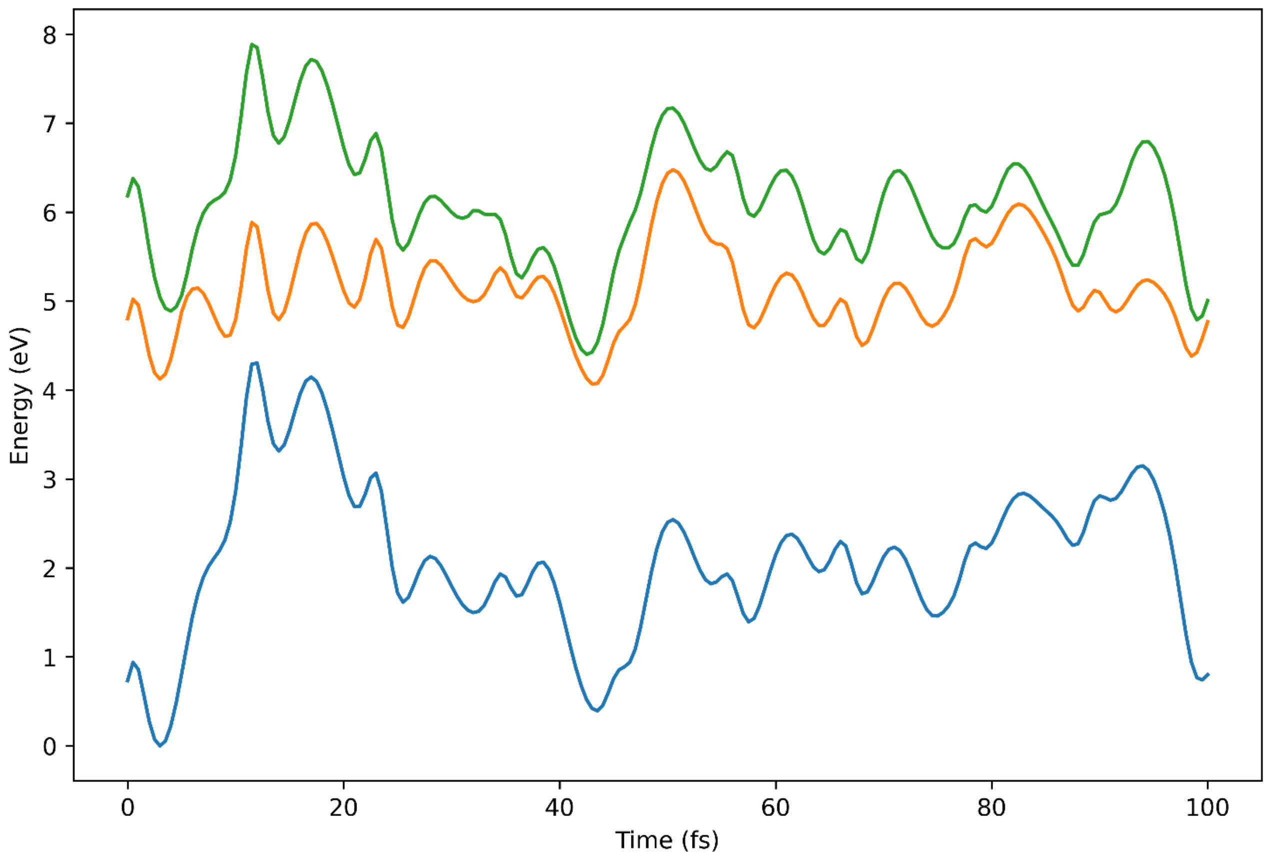


Figure 4: Potential energy surfaces of pyrazine along a training trajectory. S_0 (blue); S_1 (orange); S_2 (green).

3.1 Grid search

For this project, grid search was deemed to be the most appropriate method of model parameter optimization. Therefore, for each molecule a grid search was performed consisting of 48 variations of an NN. The grid searches were executed using the training trajectories of each molecule separately. The parameters along the axes of the grid were the amount of hidden layers (1-6) and the number of neurons (10-80) that made up the NN. The fixed parameters of the grid search were the number of epochs (10,000) and the learning rate (10^{-4}). Figure 5 presents the results of each grid search, in which the loss per hidden layer and neurons is shown. The bar charts can be hard to read for some grid points. Therefore, the precise loss can be found in the supporting information (Chapter 6.2). The parameters corresponding with the lowest loss were selected and subsequently used in training. The selected model parameters are portrayed in Table 1.

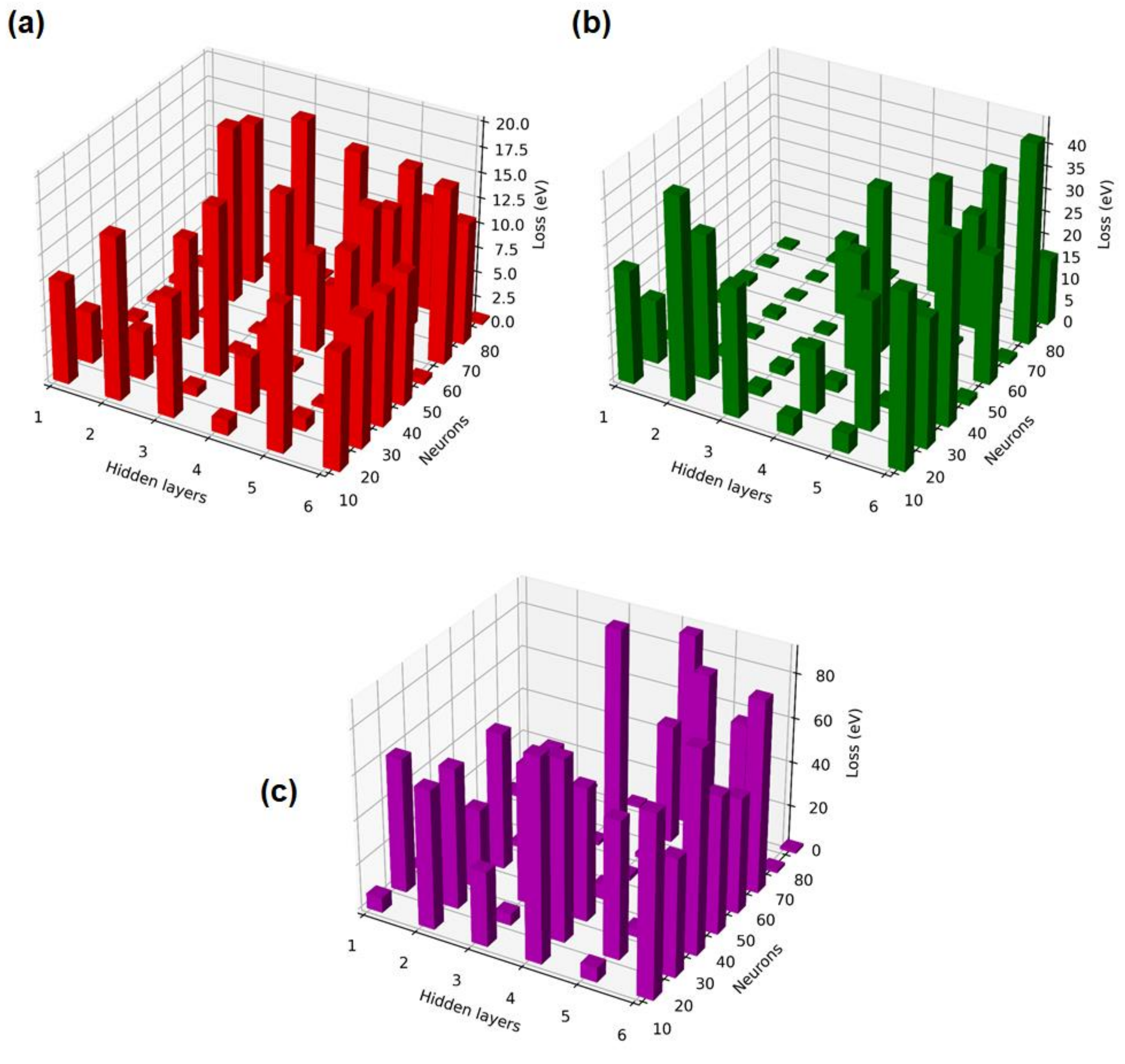


Figure 5: Grid search results per model after 10,000 epochs at a learning rate of 10^{-4}. (a) Pyrazine. (b) Pyrrole. (c) Furan.

Table 1: Model parameters as a result of the grid searches

Model	Hidden layers	Neurons
Pyrazine	4	80
Pyrrole	3	80
Furan	3	80

3.2 Model training

The selected models were trained with the goal of obtaining a training loss of 0.1 eV. For pyrazine (Figure 6) this was achieved after training for 7,500 epochs. The training was completed after 100,000 epochs, resulting in a final loss of 0.09 eV.

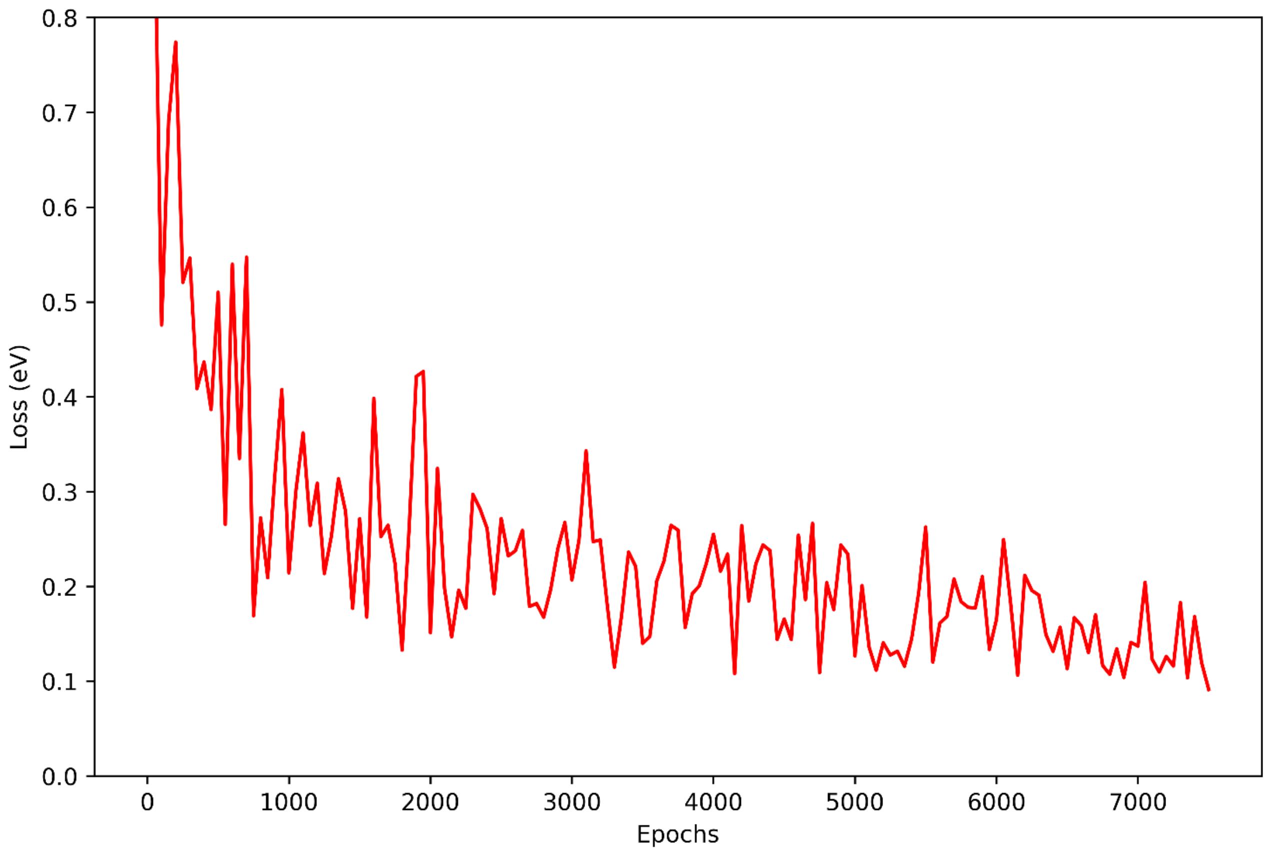


Figure 6: Loss (eV) over 7,500 epochs corresponding to the training of the pyrazine model containing 4 hidden layers and 80 neurons.

For pyrrole (Figure 7) the 0.1 eV goal was almost achieved: after 100,000 epochs a final loss of 0.15 eV was obtained.

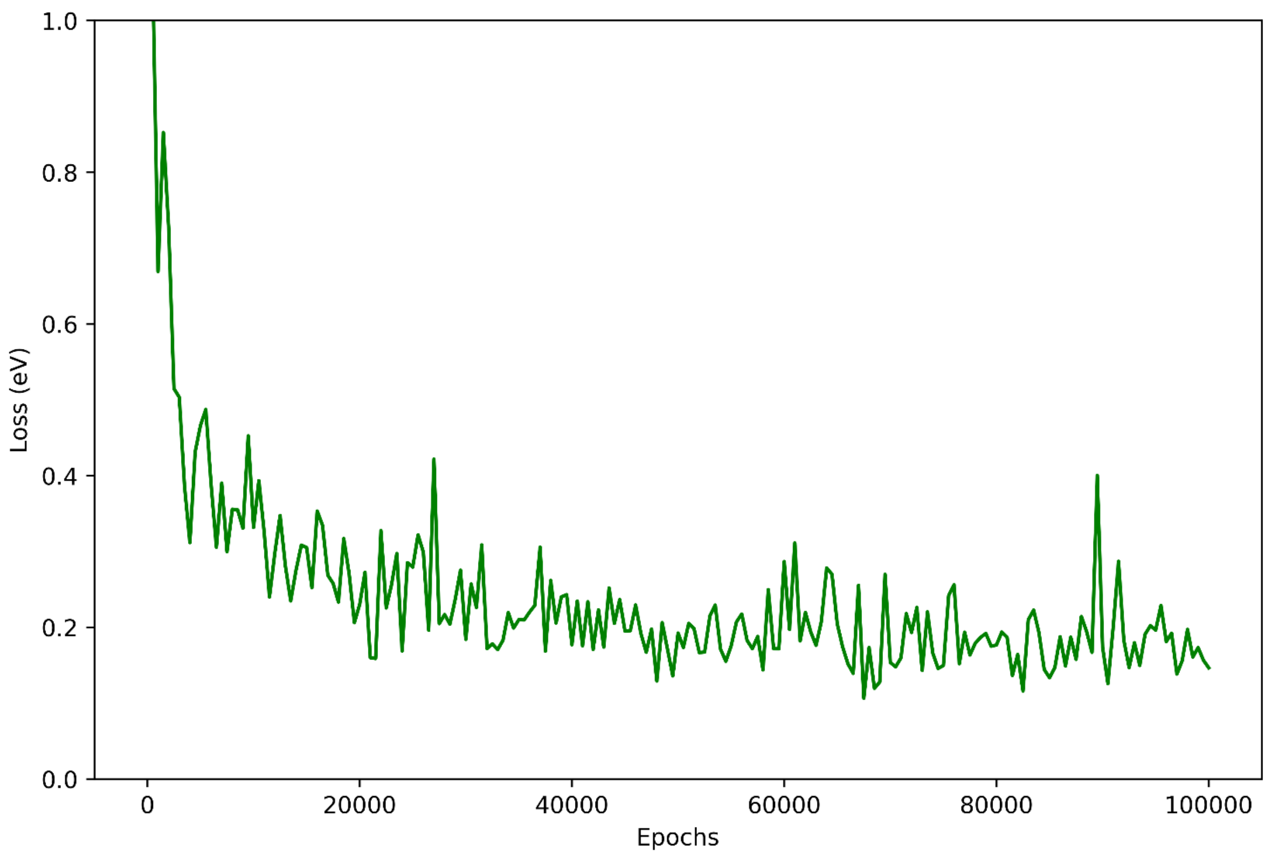


Figure 7: Loss (eV) over 100,000 epochs corresponding to the training of the pyrrole model containing 3 hidden layers and 80 neurons.

Unfortunately, the same held for the training of furan (Figure 8): a final loss of 0.39 eV was found after training for 100,000 epochs. The training process of the three models was quite similar. At first, a high loss is obtained, after which the loss rapidly decreases and becomes somewhat stable. In hindsight, training with far fewer epochs would have decreased computational cost and achieved a comparable result.

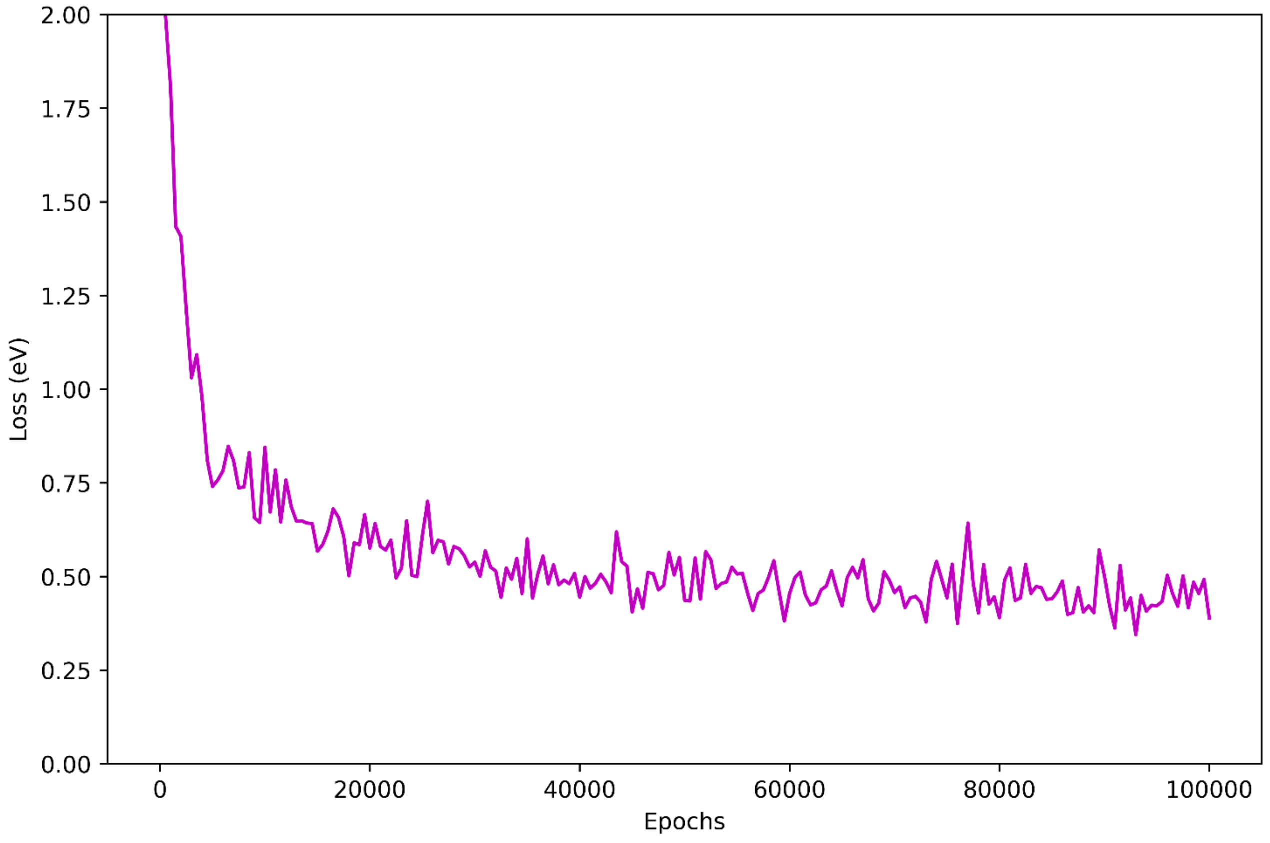


Figure 8: Loss (eV) over 100,000 epochs corresponding to the training of the furan model containing 3 hidden layers and 80 neurons.

3.3 Model testing

The trained models were finally used to predict the PESs of the molecular systems. While the training was done on training trajectories obtained with the PySurf framework, the testing was done on trajectories which the models have not 'seen', since the models are possibly overfitted to the training data.[36] The geometries of these testing trajectories were used to predict their corresponding energies. Then, these values were compared to the observed values, and a test error and bias was determined (Table 2).

Table 2: Test error and bias obtained by comparing observed testing data to data predicted by the models

Model	Test error (eV)	Bias (eV)
Pyrazine	0.23 ± 0.01	-0.01 ± 0.01
Pyrrole	0.65 ± 0.03	-0.19 ± 0.05
Furan	0.57 ± 0.02	0.03 ± 0.04

Overall, the three models had a test error below 1 eV. The test error was calculated by averaging the absolute difference between the observed energy and the energy predicted by the model at all data points of the testing trajectories. The bias was calculated in the same way as the test error, but without using absolute values. The bias allows the quantification of overprediction (positive bias) and underprediction (negative bias). The pyrazine model was most accurate. It had the smallest test error and bias. This was not unexpected, as the model's training produced the smallest loss. Least accurate was the pyrrole model, which had a significant bias towards underprediction. Although the test error of the furan model was similar to that of the pyrrole model, its bias was much closer to zero. Still, the bias of the furan model revealed a slight overprediction.

In order to illustrate model performance, a comparison between observed PESs and PESs predicted by the model is made in Figure 9. It can be seen that along the pyrazine trajectory (Figure 9a), the NN was quite capable of producing an almost identical PES. Although it seems that the model tends to overpredict, *e.g.* between 50-55 fs, this was not necessarily the case for other trajectories. This is evident from the small, negative bias associated with the model.

The pyrrole (Figure 9b) and furan (Figure 9c) predictions produced a higher loss. For pyrrole a general trend was underprediction. This is very evident for all electronic states between 65-75 fs and 93-100 fs, where relatively large deviations (~1.5 eV) from the dynamics simulations are seen. Similar deviations appear for the furan NN, which fluctuated between overprediction and underprediction. Most striking, perhaps, is the ground state (S_0) between 70-95 fs. But also the first excited state (S_1) at 30 fs, and all states at 85 fs. Regardless, the NNs show considerable results in reproducing PESs of surface hopping dynamics simulations.

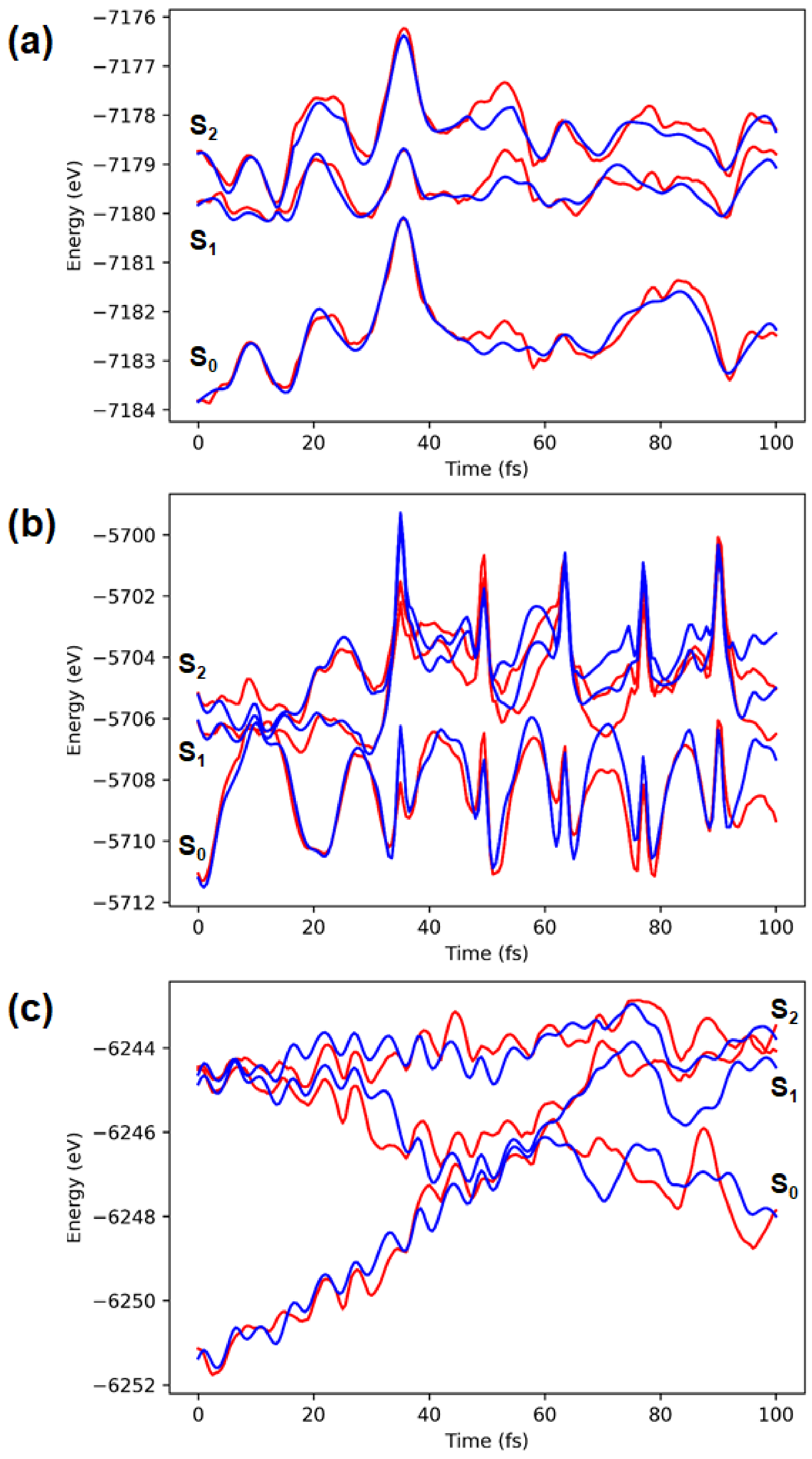


Figure 9: Potential energy surfaces of pyrazine (a), pyrrole (b), and furan (c) along testing trajectories. The blue states were produced in surface hopping simulations, whereas the red states were produced using the previously described neural networks.

4 CONCLUSION AND OUTLOOK

The goal of this research project is to explore the application of machine learning for the building of PESs of heteroaromatic compounds. Specifically, three NN models were optimized by grid search, trained on surface hopping dynamics simulations, and finally applied to predict PESs of pyrazine, pyrrole and furan, respectively. The surface hopping dynamics were simulated using the PySurf software package, while the NNs were trained using the PyTorch framework.

The grid search consisted of varying two parameters – the amount of hidden layers (1-6) and the number of neurons (10-80) – and choosing the model associated with the smallest loss. Naturally, grid search will not lead to the optimal model, since not all conditions are considered. For example, the optimal model might consist of seven hidden layers and 33 neurons. Moreover, the grid search was limited to two parameters, whereas including variations on the learning rate, for instance, would likely yield more accurate models.[37] Nevertheless, a compromise had to be made between model accuracy and computational time, and grid search was very useful given the timescale of the project. An extensive hyperparameter optimization using a more sophisticated approach, such as simulated annealing[38], could be done in a follow-up study.

Grid search allowed the selection of optimized NNs, which were subsequently trained with the goal of obtaining a training loss below 0.1 eV. This goal was only achieved for the pyrazine NN, although the pyrrole and furan NNs were not far off (0.15 eV and 0.39 eV, respectively). Better results might be achieved by increasing the amount of epochs, but most importantly by more extensive hyperparameter optimization.

Finally, the performance of the trained NNs was assessed using testing datasets, *i.e.* surface hopping simulations that were not evaluated in training. Overall, the three models had a test error below 1 eV. The pyrazine model had the smallest test error (0.23 ± 0.01 eV) and bias (-0.01 ± 0.01 eV), and was therefore the most accurate model. The pyrrole and furan models had a test error similar to one another (0.65 ± 0.03 eV and 0.57 ± 0.02 eV, respectively), but the pyrrole NN had a significant bias towards underprediction (-0.19 ± 0.05 eV). Interestingly, the test error of the pyrrole NN was much higher than its training loss when compared to the other models. This might be related to the relatively low amount of testing trajectories for pyrrole. Although there is room for improvement, the NNs show considerable results in reproducing PESs of surface hopping dynamics simulations.

This research can be continued by improving the accuracy of the NNs. As mentioned, the most progress would likely be made with hyperparameter tuning. Another approach would be to increase the size of the datasets of both training and testing data. Training on more data should lead to an increase in prediction accuracy.[39] A validation step during the training phase could prove beneficial as well. The research can also be expanded upon by investigating other heteroaromatic molecules, such as thiophene or indole.

In conclusion, machine learning appears to be a very useful and accurate tool for the building of PESs of heteroaromatic compounds. The application of NNs for electronic energy calculations would save a significant amount of computational time, meaning that research could be conducted much faster, at lower cost. Machine learning, in general, has become instrumental for recent progress in computational chemistry, and is on its way to completely revolutionize the field.

5 REFERENCES

[1] P.O. Dral, Quantum chemistry in the age of machine learning, *J. Phys. Chem. Lett.* **2020**, 11, 2336.

[2] Q. Liu, D. Lu and M. Chen, Structure and dynamics of warm dense aluminum: a molecular dynamics study with density functional theory and deep potential, *J. Phys.: Condens. Matter* **2020**, 32, 144002.

[3] A. Pihlajamäki, J. Hämäläinen, J. Linja, P. Nieminen, S. Malola, T. Kärkkäinen and H. Häkkinen, Monte Carlo simulations of $Au_{38}(SCH_3)_{24}$ nanocluster using distance-based machine learning methods, *J. Phys. Chem. A* **2020**, 124, 4827.

[4] K.T. Butler, D.W. Davies, H. Cartwright, O. Isayev and A. Walsh, Machine learning for molecular and materials science, *Nature* **2018**, 559, 547.

[5] J.S. Smith, O. Isayev and A.E. Roitberg, ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost, *Chem. Sci.* **2017**, 8, 3192.

[6] J. Westermayr, M. Gastegger and P. Marquetand, Combining SchNet and SHARC: the SchNarc machine learning approach for excited-state dynamics, *J. Phys. Chem. Lett.* **2020**, 11, 3828.

[7] O.T. Unke and M. Meuwly, PhysNet: a neural network for predicting energies, forces, dipole moments, and partial charges, *J. Chem. Theory Comput.* **2019**, 15, 3678.

[8] F.M. Paruzzo, A. Hofstetter, F. Musil, S. De, M. Ceriotti and L. Emsley, Chemical shifts in molecular solids by machine learning, *Nat. Commun.* **2018**, 9, 4501.

[9] S. Stocker, G. Csányi, K. Reuter and J.T. Margraf, Machine learning in chemical reaction space, *Nat. Commun.* **2020**, 11, 5505.

[10] R.G. Parr, Density functional theory of atoms and molecules, *Horizons of Quantum Chemistry* **1980**, 3, 5.

[11] J. Behler, Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations, *Phys. Chem. Chem. Phys.* **2011**, 13, 17930.

[12] H. Gokcan and O. Isayev, Learning molecular potentials with neural networks, *Comput. Mol. Sci.* **2021**, 1564.

[13] D.P. Kingma and J.L Ba, Adam: a method for stochastic optimization, **2015**, Published as a conference paper at ICLR.

[14] S. Sharma, S. Sharma and A. Athaiya, Activation functions in neural networks, *IJEAST* **2020**, 4, 310.

[15] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, *Nature* **1986**, 323, 533.

[16] J.S. Smith *et al.* The ANI-1ccx and ANI-1x data sets, coupled-cluster and density functional theory properties for molecules, *Sci. Data* **2020**, 7, 134.

[17] C. Devereux *et al.* Extending the applicability of the ANI deep learning molecular potential to sulfur and halogens, *J. Chem. Theory Comput.* **2020**, 16, 4192.

[18] B. Jiang and H. Guo, Permutation invariant polynomial neural network approach to fitting potential energy surfaces, *J. Chem. Phys.* **2013**, 139, 054112.

[19] R.J. Bartlett and M. Musial, Coupled-cluster theory in quantum chemistry, *Rev. Mod. Phys.* **2007**, 79, 291.

[20] Y. Liu, M. Bai, H. Song, D. Xie and J. Li, Anomalous kinetics of the reaction between OH and HO_2 on an accurate triplet state potential energy surface, *Phys. Chem. Chem. Phys.* **2019**, 21, 12667.

[21] K. Shao, J. Chen, Z. Zhao and D.H. Zhang, Fitting potential energy surfaces with fundamental invariant neural network, *J. Chem. Phys.* **2016**, 145, 071101.

[22] X. Lu, K. Shao, B. Fu, X. Wang and D.H. Zhang, An accurate full-dimensional potential energy surface and quasiclassical trajectory dynamics of the H + H_2O_2 two-channel reaction, *Phys. Chem. Chem. Phys.* **2018**, 20, 23095.

[23] C. Lee, W. Yang and R.G. Parr, Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density, *Phys. Rev. B* **1988**, 37, 785.

[24] V. Barone, P. Cimino and E. Stendardo, Development and validation of the B3LYP/N07D computational model for structural parameter and magnetic tensors of large free radicals, *J. Chem. Theory and Comput.* **2008**, 4, 751.

[25] G. Laurens, M. Rabary, J. Lam, D. Peláez and A.R. Allouche, Infrared spectra of neutral polycyclic aromatic hydrocarbons based on machine learning potential energy surface and dipole mapping, *Theor. Chem. Acc.* **2021**, 140, 66.

[26] I.V. Seregin and V. Gevorgyan, Direct transition metal-catalyzed functionalization of heteroaromatic compounds, *Chem. Soc. Rev.* **2007**, 36, 1173.

[27] M. Barbatti, Nonadiabatic dynamics with trajectory surface hopping method, *Comput. Mol. Sci.* **2011**, 1, 620.

[28] M.F.S.J. Menger, J. Ehrmaier and S. Faraji, PySurf: a framework for database accelerated dynamics, *J. Chem. Theory Comput.* **2020**, 16, 7681.

[29] E.K.U. Gross and W. Kohn, Time-dependent density-functional theory, *Adv. Quant. Chem.* **1990**, 21, 255.

[30] W.J. Hehre, R. Ditchfield and J.A. Pople, Self-consistent molecular orbital methods. XII. Further extensions of Gaussian-type basis sets for use in molecular orbital studies of organic molecules, *J. Chem. Phys.* **1972**, 56, 2257.

[31] Y. Shao *et al.* Advances in molecular quantum chemistry contained in the Q-Chem 4 program package, *Mol. Phys.* **2015**, 113, 184.

[32] J.S. Binkley and J.A. Pople, Møller-Plesset theory for atomic ground state energies, *Int. J. Quantum Chem.* **1975**, 9, 229.

[33] T.H. Dunning, Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen, *J. Chem. Phys.* **1989**, 90, 1007.

[34] L. Sun and W.L. Hase, Comparisons of classical and Wigner sampling of transition state energy levels for quasiclassical trajectory chemical dynamics simulations, *J. Chem. Phys.* **2010**, 133, 044313.

[35] A. Paszke *et al*. Automatic differentiation in PyTorch, **2017**, Published as a conference paper at NIPS.

[36] S. Manzhos and T. Carrington, Neural network potential energy surfaces for small molecules and reactions, *Chem. Rev.* **2021**, 121, 10187.

[37] J. Bergstra and Y. Bengio, Random search for hyper-parameter optimization, *J. Mach. Learn. Res.* **2012**, 13, 281.

[38] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by simulated annealing, *Science* **1983,** 220, 671.

[39] T. Kavzoglu, Increasing the accuracy of neural network classification using refined training data, *Environ. Model. Softw.* **2009**, 24, 850.

6 SUPPORTING INFORMATION

6.1 Neural network

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(27, 80),
            nn.ReLU(),
            nn.Linear(80, 80),
            nn.ReLU(),
            nn.Linear(80, 80),
            nn.ReLU(),
            nn.Linear(80, 80),
            nn.ReLU(),
            nn.Linear(80, 3),
            nn.ReLU()
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

Figure 10: Architecture for a neural network containing three hidden layers of 80 neurons.

```
def train_loop(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    for batch, (X, y) in enumerate(dataloader):
        # Compute prediction and loss
        pred = model(X)
        loss = loss_fn(pred, y)


        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Figure 11: Neural network training function.

6.2 Grid search results

Table 3: Grid search loss (eV) of pyrazine per model after 10,000 epochs at a learning rate of 10^{-4}

		Neurons							
		10	20	30	40	50	60	70	80
Hidden layers	1	10.2	5.2	0.9	0.6	0.4	0.6	0.3	0.4
	2	16.1	4.9	0.8	10.2	0.4	17.6	16.3	0.2
	3	11.9	0.9	16.8	0.3	0.3	12.5	18.0	0.2
	4	1.5	5.6	6.4	0.6	9.8	4.6	16.3	0.2
	5	14.4	1.2	0.4	13.6	15.9	13.9	16.1	10.9
	6	11.5	12.8	13.1	13.1	0.5	17.4	12.2	0.2

Table 4: Grid search loss (eV) of pyrrole per model after 10,000 epochs at a learning rate of 10^{-4}

		Neurons							
		10	20	30	40	50	60	70	80
Hidden layers	1	25.1	14.1	1.9	1.8	1.5	1.4	0.9	0.8
	2	44.7	32.2	2.1	1.4	1.4	0.7	0.6	0.6
	3	28.5	1.9	1.8	1.6	0.9	17.4	0.7	0.5
	4	3.9	14.4	2.4	26.4	36.5	0.9	0.9	25.8
	5	4.3	28.5	1.6	1.3	1.1	0.9	25.7	31.0
	6	38.5	28.5	41.3	1.2	28.5	1.0	44.7	14.9

Table 5: Grid search loss (eV) of furan per model after 10,000 epochs at a learning rate of 10^{-4}

		Neurons							
		10	**20**	**30**	**40**	**50**	**60**	**70**	**80**
Hidden layers	**1**	6.8	60.2	4.5	3.8	2.3	2.2	2.0	1.6
	2	62.1	62.6	35.5	61.5	2.6	1.7	1.5	1.4
	3	33.3	5.2	62.1	60.8	2.3	1.8	90.2	1.2
	4	90.2	80.7	59.6	6.4	2.3	2.1	53.0	86.0
	5	6.6	61.5	3.8	2.7	2.3	90.2	2.0	52.3
	6	80.7	52.3	90.2	61.5	51.6	86.0	1.5	1.3