universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Extending equivariant message passing neural networks for excited states“

verfasst von / submitted by

Sascha Mausenberger, BSc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 862
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Chemie
Betreut von / Supervisor:	Priv.-Doz. Dr. Philipp Marquetand

Acknowledgements

I want to thank Philipp Marquetand for his great machine learning lecture, which was the reason for my decision to write my thesis in this field of research, and for supporting me throughout the time. I also want to thank Julia Westermayr for helping me understand how SchNarc works, and Michael Gastegger for his explanations of the SchNetPack 2.0 internals. Next, I want to thank the ViRAPID team (Brigitta, Madlen and Max) for the welcoming atmosphere in the office. Further, I want to thank Leticia González for her bachelors lecture in theoretical chemistry, that initially got me hooked, and Markus Oppel for always finding the time to explain to me how the IT infrastructure of the institute works. And last but not least, I want to thank my girlfriend Gabriele and my family for supporting me during my study.

Abstract

It has been shown that excited-state molecular dynamics simulations can be accelerated with reasonable accuracy by predicting molecular properties with an invariant message passing neural network (MPNN) instead of expensive quantum chemical computations. Even though this method produces accurate results for the selected test systems, equivariant properties like the non-adiabatic couplings (NACs), which determine the hopping probability between different states, are predicted poorly compared to invariant properties like the energies. This is an intrinsic problem which can not be solved, because invariant representations cannot directly predict equivariant properties. In this work, the invariant MPNN was replaced with an equivariant MPNN which is able to predict all molecular properties with significantly improved accuracy. The previous method, called SchNarc, combines the invariant MPNN SchNet included in the machine learning framework SchNetPack and the trajectory surface hopping molecular dynamics package SHARC (Surface Hopping including Arbitrary Couplings). For the new approach, the equivariant MPNN PaiNN (Polarizable Atom Interaction Neural Network), included in SchNetPack 2.0, was combined with SHARC and is called SPaiNN.

Kurzfassung

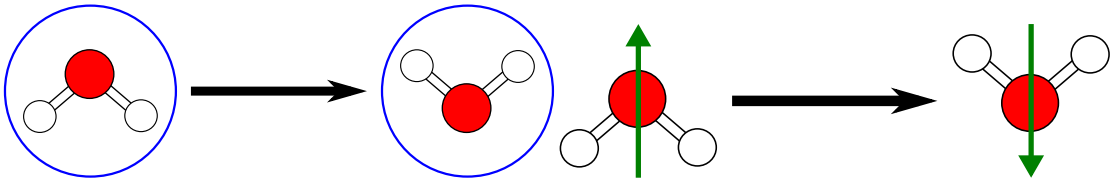
Es wurde gezeigt, dass molekulardynamische Simulationen für angeregte Zustände mit hinreichender Genauigkeit beschleunigt werden können, indem man molekulare Eigenschaften mit invarianten neuronalen Netzen vorhersagt, anstatt teure quantenmechanische Berechnungen zu verwenden. Obwohl diese Methode für ausgewählte Testsysteme gute Ergebnisse erzielt, können equivariante Eigenschaften wie die nicht adiabatischen Kopplungen, welche für die Ermittlung der Übergangswahrscheinlichkeit zwischen zwei Zuständen gebraucht werden, nur sehr schlecht vorhergesagt werden verglichen mit invarianten Eigenschaften wie den Energien. Dies stellt ein intrinsisches Problem dar, welches nicht gelöst werden kann, da invariante neuronale Netze equivariante Eigenschaften nicht direkt vorhersagen können. In dieser Arbeit wurde das invariante neuronale Netzwerk durch ein equivariantes neuronales Netzwerk ersetzt, welches in der Lage ist jegliche molekulare Eigenschaft mit viel höherer Genauigkeit vorherzusagen. Der frühere Ansatz, genannt SchNarc, kombiniert das invariante neuronale Netz SchNet aus dem machine learning framework SchNetPack mit dem trajectory surface hopping Paket SHARC (Surface Hopping including Arbitrary Couplings). Im neuen Ansatz, genannt SPaiNN, wird das equivariante neuronale Netz PaiNN (Polarizable Atom Interaction Neural Network) aus dem machine learning framework SchNetPack 2.0 mit SHARC kombiniert.

Contents

1 Introduction

In the last few years, machine learning has made huge progress. Whether it is translation and generation of texts [1], image generation [2], face recognition [3], prediction of chemical properties [4], generation of new molecular structures with desired properties [5] or folding proteins [6], machine learning has made it into everyday life. In some areas machine learning models are already superior to humans, and achieved things that were once believed to be impossible for machines [7, 8]. Advances have also been made in theoretical chemistry, where machine learning models are able to predict spectra [9], thermodynamic [10] and quantum mechanical properties [11] in a fraction of a second.

This work focuses on accelerating excited-state dynamic simulations using a neural network to predict molecular properties instead of using expensive quantum chemical calculations. Excited-state dynamic simulations are used to theoretically describe photochemical processes in molecules, like charge transfer, energy transfer and excition dissotiation [12]. Understanding this processes can help for example with the development of new light-emitting or photovoltaic materials. Molecular dynamics particularly benefits from fast machine learning predictions since a single trajectory cannot be parallelized because the geometries of each time-step are unknown at the beginning of the simulation. Thus, longer simulation times are possible in a reasonable amount of time. Using machine learning to simulate non-adiabatic excited-state dynamics is a relatively new field of research but great successes were achieved in the past few years [13]. Many different approaches with kernel ridge regression or neural networks have been proposed to predict the needed properties (energies, forces, NACs, ...) either direct or indirect by predicting wave functions or electronic densities [13]. One promising solution, SchNarc [14], which combines the machine learning framework SchNetPack 1.0 [15] and the molecular dynamics program SHARC (Surface Hopping including Arbitrary Couplings) [16], was used as a blueprint for this work. SchNarc [14] uses the invariant representation SchNet [17] to predict energies, forces, non-adiabatic couplings (NACs), spin orbit couplings (SOCs) and (transition) dipole moments for a target configuration of a molecule. These properties are then passed to SHARC, which handles the rest of the simulation. In this work, SchNarc was build from scratch with up-to date python libraries and SchNetPack 2.0 [4], which includes the equivariant representation PaiNN (Polarizable Atom Interaction Neural Network) [18]. Rewriting the code instead of adapting the existing was unavoidable since SchNetPack 2.0 [4] is a major rewrite of SchNetPack 1.0 [15] without backward compatibility.



(a) energy (invariant) (b) dipole moment (equivariant)

Figure 1.1: Illustration of equivariant and invariant molecular properties.

Using PaiNN [18] instead of SchNet [17] improves the prediction accuracy of equivariant properties like NACs and (transition) dipole moments. An illustration of the difference between equivariant and invariant properties is shown in fig. 1.1. The energy as an example for an invariant property, shown on the left side is represented as blue circle. If the molecule is turned upside down the amount of energy stays the same and it stays the same no matter how the molecule is rotated in space. In contrast an equivariant property like the dipole moment, represented as green arrow on the right side, changes with rotation.

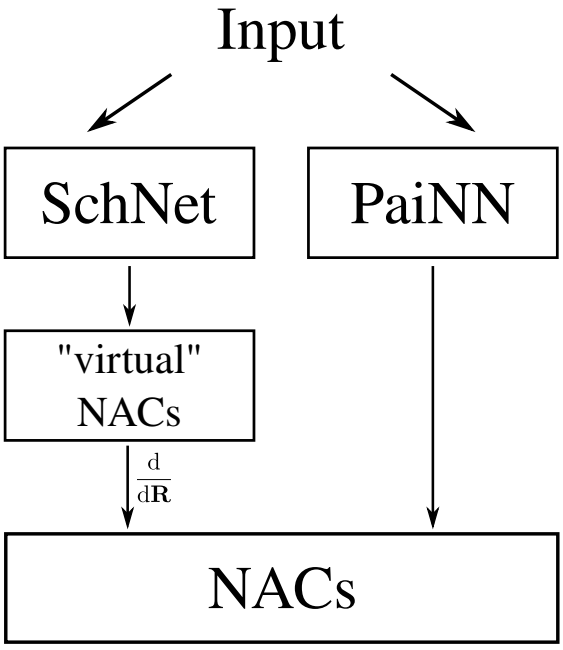


Figure 1.2: Illustration of the differences in predicting NACs with equivariant and invariant representations.

The major advantage of equivariant representations is that equivariant properties can be predicted directly, whereas invariant representations can only predict invariant properties. To overcome this limitation the prediction of the invariant representation has to be post-processed to become equivariant. An example of how this is done in SchNarc [14] for the NACs is shown in fig. 1.2. A so-called "virtual" property is predicted which

is then post-processed by taking the derivative with respect to the nuclear coordinates **R**. Besides the additional computational effort another drawback of taking derivatives of predictions is that small errors in the prediction lead to larger errors in the derivative.

The focus of this work will be the recreation of SchNarc [14] with the equivariant representation PaiNN [18] and the investigation of the effect on excited-state dynamics. As test systems the methlyenimmonium cation $CH_2NH_2^+$ and ethylene were chosen because of their small size, their fast excited-state dynamics and because the training data already existed. The training data for $CH_2NH_2^+$ was generated with MR-CISD(6,4)/aug-cc-pVDZ level of theory (from ref. [14]) and the ethylene data with SA(3)-CASSCF(2/2)/6-31G* level of theory (from ref. [19]).

2 Theory

2.1 Quantum Chemistry for Electronic Excited States

2.1.1 Multi-Reference Configuration Interaction

MRCI is a method that can treat electron correlation by constructing reference configurations and exciting electrons in these configurations. The electronic Schrödinger equation is then solved by variationally minimizing the energy in the configuration space [20].

$$|\Psi^\alpha\rangle = \sum_{k=1}^{N_{\mathrm{CI}}} c_k^\alpha \, |\Phi_k\rangle \tag{2.1}$$

The general concept of CI is to compute the many-electron wave function of an electronic state as linear combination of basis functions. $|\Phi_k\rangle$ are the basis functions representing electronic configurations, $|\Psi^\alpha\rangle$ is the wave function of state α and c_k^α are the CI coefficients [20].

$$\hat{H} \, |\Psi^\alpha\rangle = E^\alpha \, |\Psi^\alpha\rangle \tag{2.2}$$

Starting from the Schrödinger equation (eq. 2.2), with $\hat{H}$ as the electronic Hamiltonian operator and E^α as the energy of state α, inserting eq. 2.1 into eq. 2.2 leads to

$$\sum_{k=1}^{N_{\mathrm{CI}}} c_k^\alpha \hat{H} \, |\Phi_k\rangle = \sum_{k=1}^{N_{\mathrm{CI}}} c_k^\alpha E^\alpha \, |\Phi_k\rangle \tag{2.3}$$

After left multiplication with $\langle\Phi_l|$, the following equation is obtained.

$$\sum_{k=1}^{N_{\mathrm{CI}}} c_k^\alpha \, \langle\Phi_l|\hat{H}|\Phi_k\rangle = \sum_{k=1}^{N_{\mathrm{CI}}} c_k^\alpha E^\alpha \, \langle\Phi_l|\Phi_k\rangle \tag{2.4}$$

Assuming the many-electron basis functions are orthonormal, $\langle\Phi_l|\Phi_k\rangle = \delta_{lk}$ and define a CI-matrix $\mathbf{H}$ with elements $H_{lk} = \langle\Phi_l|\hat{H}|\Phi_k\rangle$ leads to

$$\sum_{k=1}^{N_{\mathrm{CI}}} H_{lk} c_k^\alpha = E^\alpha c_l^\alpha \tag{2.5}$$

Eq. 2.5 can also be written in matrix form and is the main equation of CI [20].

$$\mathbf{H}\mathbf{c}^\alpha = E^\alpha \mathbf{c}^\alpha \tag{2.6}$$

With eq. 2.6 it is also possible to calculate excited states, where the ground state is the lowest eigenvalue of $\mathbf{H}$ and the higher eigenvalues correspond to excited states [20].

2.1.2 Phase of the Wave Function

The arbitrariness of the phase of wave functions is an important effect that has to be considered in molecular dynamics if the nuclear motion is also treated quantum mechanically. This also affects the training of quantum mechanical properties in machine learning because changes in the phase convention lead to jumps in the training data that are hard to fit. There are two independent factors that are responsible for the change of the wave function phase, the Berry phase, also called geometric phase, and the dynamic phase. When the Hamiltonian $\hat{H}(\mathbf{R}(\mathbf{t}))$ of a quantum mechanical system is varied by the parameter $\mathbf{R}(\mathbf{t})$ from $t = 0$ to $t = T$ on a round closed path C, so that $\mathbf{R}(T) = \mathbf{R}(0)$, then the system returns to the initial state, apart from a phase factor [21]. The temporal evolution of a quantum mechanical system can be described by the Schrödinger equation

$$\hat{H}(\mathbf{R}(t))\,|\psi(t)\rangle = i\hbar\frac{\mathrm{d}}{\mathrm{d}t}\,|\psi(t)\rangle \tag{2.7}$$

At any time t, the system consists of the eigenstates $|n(\mathbf{R})\rangle$, with the energies $E_n(\mathbf{R})$ $(\mathbf{R}(\mathbf{t}) = \mathbf{R})$

$$\hat{H}(\mathbf{R})\,|n(\mathbf{R})\rangle = E_n(\mathbf{R})\,|n(\mathbf{R})\rangle \tag{2.8}$$

This equation defines no relationship between the phases of $|n(\mathbf{R})\rangle$ at different $\mathbf{R}$ and the phase can be chosen arbitrarily [21].

A system in one of the states $|n(\mathbf{R}(0))\rangle$ is adiabatically evolved with $\hat{H}$ to the state $|n(\mathbf{R}(t))\rangle$ at time t. Then $|\psi\rangle$ can be written as

$$|\psi(t)\rangle = \exp\left(\frac{-i}{\hbar}\right)\int_0^t \mathrm{d}t' E_n(\mathbf{R}(t'))\exp\left(i\gamma_n(t)\right)|n(\mathbf{R}(t))\rangle \tag{2.9}$$

The first exponential is the dynamical phase factor and $\gamma_n(t)$ is the non-integrable geometric phase factor. The geometric phase factor is not a function of $\mathbf{R}$ and is not single-valued around a closed circuit, $\gamma_n(T) \neq \gamma_n(0)$ [21]. In general, the arbitrariness of the wave function phases can be mitigated by tracking the wave function overlaps between the time steps and adjusting the phases in a way that the overlap is maximized [22].

2.2 Surface Hopping

Surface hopping is a mixed quantum-classical method to study molecular dynamics that separates the adiabatic and nonadiabatic processes. The adiabatic dynamics of the nuclei are treated classically and the nonadiabatic effects of branching electronic population are introduced through a stochastic algorithm, resulting in semiclassical trajectories that can exchange electronic states. The statistical nature of the wave packet propagation is captured by preparing an ensemble of these trajectories.[23]

Assuming the nuclei move along a trajectory $\mathbf{R}^c(t)$, the time-dependent wave function for the electrons can be written as

$$\varphi(\mathbf{r}, \mathbf{R}^c, t) = \sum_j c_j(t)\Phi_j(\mathbf{r}; \mathbf{R}^c(t)) \tag{2.10}$$

with $\mathbf{r}$ being the electronic coordinates, the superscript c indicates that the nuclei positions are fixed $\mathbf{R}^c(t)$ at time t.

$$\left(i\hbar\frac{\partial}{\partial t} - H_e\right)\varphi(\mathbf{r},\mathbf{R},t) = 0 \tag{2.11}$$

Substituting eq. 2.10 into the time-dependent Schrödinger equation (eq. 2.11) leads to the following set of equations

$$i\hbar\frac{\mathrm{d}c_k}{\mathrm{d}t} + \sum_j(-H^c_{kj} + i\hbar\mathbf{F}^c_{kj}\cdot\mathbf{v}^c)c_j = 0 \tag{2.12}$$

with $\mathbf{F}^c_{kj}$ being the nonadiabatic coupling vector between states j and k, $H^c_{kj} = \langle\Phi_k|H_e|\Phi_j\rangle_{\mathbf{r}}$ and $\mathbf{v}^c$ is the nuclear velocity [23]. The hopping probability between state j and k is given by [24]

$$P_{j\to k} = \left(1 - \frac{|c_j(t+\Delta t)|^2}{|c_j(t)|^2}\right)\frac{\mathcal{R}\left[c_k(t+\Delta t)e^{-i(H^c_{kj}+\mathbf{F^c_{kj}})\Delta t}c^*_j(t)\right]}{|c_j(t)|^2 - \mathcal{R}\left[c_j(t+\Delta t)e^{-i(H^c_{jj}+\mathbf{F^c_{jj}})\Delta t}c^*_j(t)\right]} \tag{2.13}$$

where $\mathcal{R}$ is the real part.

2.3 Machine Learning in Quantum Chemistry

2.3.1 Neural Networks

Neural networks are a computational model inspired by biology. Artificial neurons are connected with coefficients, which form the structure of the neural network. The smallest unit of a neural network, the neuron, can perform simple information processing, but the computational power of neural networks comes from many neurons connected to large networks [25].

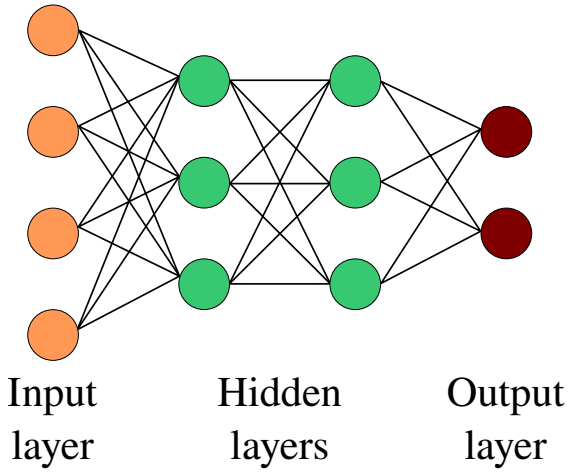


Figure 2.1: Illustration of a feed-forward neural network with an input layer, two hidden layers and an output layer. Neurons are represented as colored circles, the layers are connected by weights represented as black lines.

A typical neural network consists of an input layer, one or more hidden layer(s) and an output layer (see fig. 2.1). Each layer is built with a variable number of neurons, which are not connected to each other. The number of neurons in the input and output layers specify the number of input values that need to be fed into the network or how many output values are generated respectively. Each neuron of a layer is connected to each neuron of the adjacent layers with a numerical value that is being optimized in the training process.

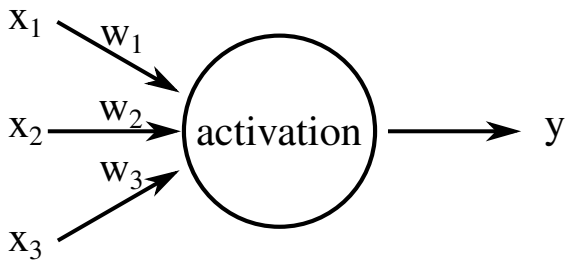


Figure 2.2: Illustration of a single neuron. Inputs x_i are multiplied with the corresponding weights w_i and are passed with another trainable parameter to an activation function which determines the output value y.

$$y = f\left(b + \sum_i x_i \cdot w_i\right) \tag{2.14}$$

In fig. 2.2 is shown how a single neuron produces an output based on the arriving inputs. The inputs x_i multiplied by the corresponding weights w_i are summed up and get passed

with another trainable parameter, the bias b, to some nonlinear activation function f (see eq. 2.14). The weights and biases of all nodes are iteratively updated in the training process by minimizing the loss between the predicted outputs and the expected outputs until the parameters converged to a minimum of the loss function [13, 25].

2.3.2 Message Passing Neural Networks

Graph neural networks (GNNs) are a powerful way for describing molecules, since every molecule can be represented as a graph [26]. Message Passing Neural Networks are a type of GNNs that have a forward pass that consists of two phases. In the first phase, the message passing phase, the nodes within the graph repeatedly exchange messages followed by updates. In the second phase, the readout phase, an aggregation function collects the node states and transforms them into an embedding [18, 26].

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{2.15}$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{2.16}$$

In the message passing phase, hidden states h_v^t (eq. 2.16) at each node v are updated based on the messages m_v^{t+1} (eq. 2.15). Where $N(v)$ represents the neighbours of v and e_{vw} is the edge that connects the nodes v and w in the graph G. The message passing phase is repeated T times.

$$\hat{y} = R(\{h_v^T | v \in G\}) \tag{2.17}$$

In the readout phase the feature vector $\hat{y}$ (eq. 2.17) of the graph is computed with some readout function R that aggregates the initial and final states and transforms them into a graph embedding [27].

2.3.2.1 Equivariant MPNN

To obtain better representations for predicting vectorial properties of local environments, neurons can be geometric objects like vectors and tensors. Such a message pass can be written as

$$\vec{m}_v^{t+1} = \sum \vec{M}_t(\vec{h}_v^t, \vec{h}_w^t, \vec{e}_{vw}) \tag{2.18}$$

$$R\vec{f}(\vec{x}) = \vec{f}(R\vec{x}) \tag{2.19}$$

The message function $\vec{M}_t$ and the update function $\vec{U}_t$ have to fulfill eq. 2.19 for any rotation matrix R to be equivariant.[18]

3 Algorithmic Development

3.1 Phase-less Loss

Besides correcting the phases in the data set with tracking the overlaps of the references, another option to overcome the phase problem (see section 2.1.2) is using a phase-less loss. The basic idea of phase-less loss is to generate all possible phase combinations of the prediction and find the one that has the lowest error compared to the target. Since the first phase element can be chosen to be either 1 or -1, the number of possible phase combinations is 2^{n-1} with n couplings. To illustrate the previously established approach [14], an example of the phase-less loss with three couplings is shown below.

$$\mathbf{\Phi} = \begin{pmatrix} C_{01} \\ C_{12} \\ C_{02} \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} C_{01} & C_{01} & C_{01} & C_{01} \\ C_{12} & C_{12} & -C_{12} & -C_{12} \\ C_{02} & -C_{02} & C_{02} & -C_{02} \end{pmatrix} \tag{3.1}$$

$$\mathcal{L}^{\text{phaseless}} = \min\left(\mathcal{L}(\mathbf{\Phi}_{*,j}, C^{\text{target}})\right) \tag{3.2}$$

In SchNarc [14] the phase-less loss is implemented like in (eq. 3.1). First all possible phase vectors are generated ($2^2 = 4$ for three couplings), then the prediction C is multiplied element-wise with the phase matrix. Then, the loss $\mathcal{L}$ (e.g. MSE, MAE, ...) of each column from the resulting matrix to the target is calculated and the lowest value is the minimal phase-less Loss $\mathcal{L}^{\text{phaseless}}$ (eq. 3.2). The latter is minimized as it is commonly done in ML with a variant of stochastic gradient descent (AdamW [28]) to yield the corrected coupling vectors. With this method, the data set gets smoothed since it removes sudden jumps that would make it difficult for a machine learning model to fit the data. It is important to note that the phase conventions of two models can differ, which plays a role if two or more models are used for making predictions. This algorithm scales exponentially $\mathcal{O}(2^n)$ with the number of couplings and the memory consumption scales with $\mathcal{O}(n \cdot m)$, with n being the number of couplings and m being the number of possible phase vectors.

Since this loss function gets called multiple times per epoch exponential scaling is highly unfavorable and restricts this method to a low number of couplings. Looking at the resulting phase matrix in (eq.3.1) there are quite a few redundant calculations that could be avoided. The first coupling C_{01} gets multiplied four times with 1 while the other two couplings C_{12} and C_{02} get multiplied two times with 1 and two times with -1.

$$\mathcal{L}_{01}^{\text{phaseless}} = \min(\mathcal{L}(C_{01}, C_{01}^{\text{target}}), \mathcal{L}(-C_{01}, C_{01}^{\text{target}}))$$
$$\mathcal{L}_{12}^{\text{phaseless}} = \min(\mathcal{L}(C_{12}, C_{12}^{\text{target}}), \mathcal{L}(-C_{12}, C_{12}^{\text{target}}))$$
$$\mathcal{L}_{02}^{\text{phaseless}} = \min(\mathcal{L}(C_{02}, C_{02}^{\text{target}}), \mathcal{L}(-C_{02}, C_{02}^{\text{target}})) \tag{3.3}$$
$$\mathcal{L}^{\text{phaseless}} = \sum_i \mathcal{L}_i^{\text{phaseless}} \tag{3.4}$$

To avoid these redundancies a new approach was developed. Instead of trying to find the correct phase convention for all couplings at once with the corresponding 2^{n-1} combinations, each coupling is treated individually like in (eq. 3.3). First, each coupling is multiplied with 1 and -1 then the lowest loss $\mathcal{L}$ to the corresponding coupling from the target gets returned. The sum of the errors of each coupling (eq. 3.4) results in the same loss as in (eq. 3.2). This algorithm scales linearly with the number of couplings $\mathcal{O}(n)$ and the memory consumption scales with $\mathcal{O}(2 \cdot n)$. Further improvement would be possible with using the fact that the phase of the first coupling can be chosen freely. Since the extra code needed would make the function less readable and would add some overhead in form of reshaping and slicing the input and output tensors, an implementation of this feature was not attempted.

3.2 Smooth NACs

Learning NACs directly is quite a hard task for machine learning models due to the shape of the couplings that is hard to fit with all the containing singularities and cusps. One way of smoothing these sharp peaks is to use the relation between NACs and the energy difference between the two corresponding states (see fig. 3.1).

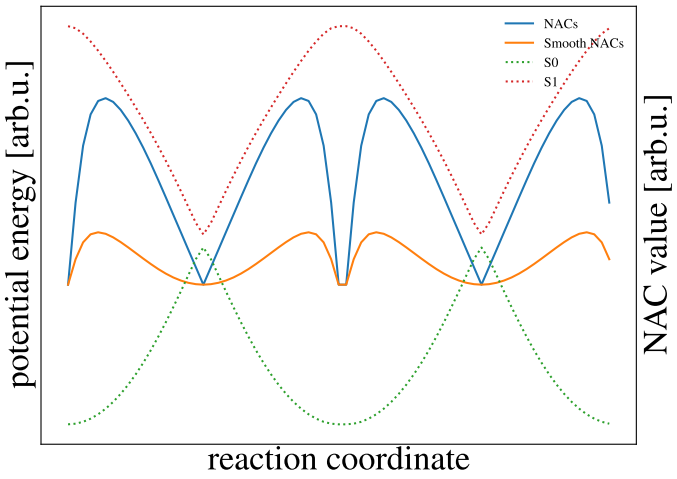


Figure 3.1: Illustration of the distribution of NAC values (norm of the summed atom contributions), normal and smoothed, compared to corresponding energies in a two state system along some arbitrary reaction coordinate.

$$C_{ij}^{\text{NAC}} \approx \langle \Psi_i | \frac{\partial}{\partial \mathbf{R}} | \Psi_j \rangle = \frac{1}{E_i - E_j} \tilde{C}_{ij} \text{ for } i \neq j \tag{3.5}$$

$$\tilde{C}_{ij} = \langle \Psi_i | \frac{\partial \hat{H}_{\text{el}}}{\partial \mathbf{R}} | \Psi_j \rangle \tag{3.6}$$

In SchNarc [14], the NACs are smoothed by training $\tilde{C}_{ij}$ (eq. 3.6) which can be obtained by multiplying the NACs with the corresponding energy gap.

$$\tilde{C}_{ij} = C_{ij}^{\text{NAC}} \cdot \Delta E_{ij} \tag{3.7}$$

The same approach was used in this project with some minor differences. Instead of calculating $\tilde{C}_{ij}$ on the fly during training, this property was precalculated and added to the database used for training. This approach avoids doing the same calculations over and over again at the cost of slightly increased disk usage for the database if the smooth NACs are stored in addition the the NACs. However, the original NAC data could be omitted since it is not needed for training smooth NACs anyway. While SchNarc [14] uses the NAC data without preprocessing, in this project the NAC data was normalized before training. Normalizing data ensures that all features of the data have similar scaling which helps the model to make better predictions.

3.3 Switching from SchNetPack 1.0 to SchNetPack 2.0

Using the latest SchNetPack version is crucial since the equivariant representation PaiNN was first implemented in SchNetPack 2.0 [4]. Although it would be possible to implement

PaiNN in SchNetPack 1.0 [15], which would ensure compatibility with SchNarc [14], this was not attempted. As the latest version uses up-to date libraries, improved algorithms, new features and active development, building SchNarc [14] from scratch was favored.

The most important difference between SchNetPack 1.0 [15] and SchNetPack 2.0 [4] is the new data pipeline. The new data pipeline is now fully sparse, thus, padding atomic environments with a varying number of neighbours with zeros is no longer needed [4].

$$T_{1.0} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & 0 & 0 \end{bmatrix} \tag{3.8}$$

$$T_{2.0} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & b_1 & b_2 \end{bmatrix} \tag{3.9}$$

To illustrate this, an example with two systems A and B batched together into a tensor T is shown. System A includes four entries $\{a_1, a_2, a_3, a_4\}$ and system B includes two entries $\{b_1, b_2\}$. In SchNetPack 1.0 [15] both systems are concatenated, adding an additional dimension to the tensor (eq. 3.8). If one system is larger than the other, the size difference has to be padded with zeros. SchNetPack 2.0 [4] avoids padding with concatenating the systems by extending the first dimension of the tensor instead of adding additional dimensions (eq. 3.9).

With the new pipeline some changes in the data set structure (see tab. 3.1) have to be done to make use of the improved pipeline.

Table 3.1: Structure of the tensors for each property in comparison between SchNarc [14] and SPaiNN. **R** are the x,y,z components of the vectors.

Property	SchNarc	SPaiNN
Energy	[Energy]	[1,Energy]
Forces	[State,Atom,**R**]	[Atom,State,**R**]
NACs	[Coupling,Atom,**R**]	[Atom,Coupling,**R**]
Dipoles	[Coupling,**R**]	[1,Coupling,**R**]

The structures of the properties in SchNarc [14] were not chosen arbitrarily, they are in the shape as SHARC prints them in the output file of the quantum mechanical calculation. And this structure has to be reestablished when passing predictions to SHARC, which requires reshaping the tensors back to the SchNarc [14] format (see tab. 3.1).

4 Computational Details

4.1 Training Parameters

Most of the parameters used for training were the same as used in SchNarc[14] to train the methylenimmonium cation (see tab. 4.1). No hyperparameters were optimized, since the goal of this project was to compare the performance of different representations. Only the weightings of the loss from the trained properties were adapted since the algorithmic optimizations changed the value range of the NAC loss several orders of magnitude. Also there were no weightings for the (transition)dipoles since they were not included in the SchNarc[14] paper (see tab. 4.2). The splittings of the data sets into test, training and validation data is described in tab. 4.3.

Table 4.1: Hyperparameters used in all trainings.

Parameter	Value
Batch size	50
Features	256
Interactions	6
Smooth NACs	Yes
LR	0.0001
LR decay	0.8
Minimum LR	1e-6
LR patience	50
Epochs	3000
Number of Layers	3
Number of Gaussians	50
Cutoff	20
Precision	FP64

Table 4.2: Used loss function for each trained property and contributions to the total loss.

Property	Loss function	Weight
Energy	MSE	1.0
Forces	MSE	1.0
NACs	Phase-less MSE	1.0
Dipoles	Phase-less MSE	0.1

Table 4.3: Size of the splits used in training for each data set.

Data set	train data	validation data	test data
$CH_2NH_2^+$	3000	300	700
ethylene	4000	400	689

4.2 Surface Hopping Parameters

The initial geometries, velocities and parameters for all 3846 $CH_2NH_2^+$ trajectories were used "as is" from the SchNarc[14] paper (see tab.4.4). The initial geometries, velocities and parameters for all 1000 ethylene trajectories were used as is from "A molecular perspective on Tully models for nonadiabatic dynamics"[19] (see tab.4.5).

Table 4.4: Parameters used in the input files of all $CH_2NH_2^+$ trajectories. Not included, rngseed which is different for each file.

Parameter	Value
veloc	external
nstates	3 0 0
actstates	3 0 0
state	3 mch
coeff	auto
ezero	-94.7095344465
tmax	100
stepsize	0.5
nsubsteps	25
surf	diagonal
coupling	nacdr
gradcorrect	True
ekincorrect	parallel_nac
reflect_frustrated	none
decoherence_scheme	edc
decoherence_param	0.1
hopping_procedure	sharc
grad_all	True
nac_all	True
nospinorbit	True

Table 4.5: Parameters used in the input files of all ethylene trajectories. Not included, rngseed which is different for each file.

Parameter	Value
veloc	external
nstates	2
actstates	2
state	3 mch
coeff	auto
ezero	0
tmax	80
stepsize	0.5
nsubsteps	25
surf	diagonal
coupling	nacdr
gradcorrect	True
ekincorrect	parallel_vel
reflect_frustrated	none
decoherence_scheme	edc
decoherence_param	0.1
hopping_procedure	sharc
grad_select	True
nac_all	True
eselect	0.001
nospinorbit	True

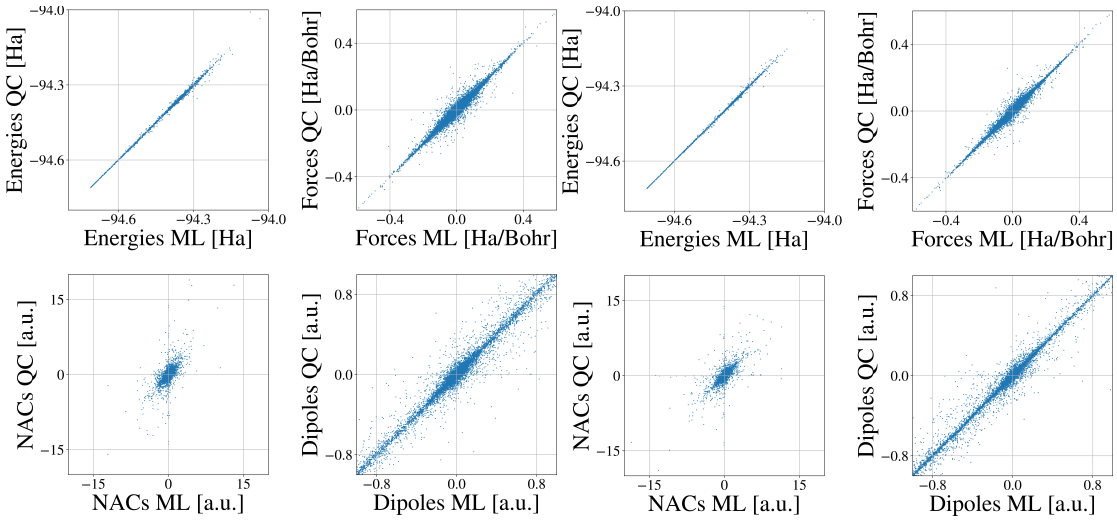
5 Results and Discussion

In this project, SchNarc [14] was recreated from scratch with improved algorithms and improved usability. This was necessary since migrating from SchNetPack 1.0 [15] to SchNetPack 2.0 [4], in order to use the equivariant representation PaiNN, required fundamental adaptions to the original code base. Throughout the rest of the thesis the recreated version of SchNarc [14] will be referenced as SchNarc 2.0, when using the invariant representation SchNet. If the equivariant representation PaiNN was used, it will be referenced as SPaiNN.

SPaiNN has huge advantages compared to SchNarc 2.0 not only in performance but also in predicting equivariant properties like NACs, SOCs and (transition) dipole moments. SchNarc 2.0 itself cannot predict these vectorial properties since it uses an invariant representation of the target geometry. A computationally expensive trick is used in SchNarc 2.0 to circumvent this obstacle for the NACs: First a "virtual" property is predicted then the derivative is taken to make the result equivariant.

In order to compare SchNarc 2.0 and SPaiNN, corresponding ML models were trained for two molecules. The first molecule, methylenimmonium cation was chosen because of its small size and fast excited-state dynamics. The small size makes it possible to generate a large data set with highly accurate quantum chemical reference calculations and the ultrafast transitions are challenging to reproduce with ML models [29]. The second molecule, ethylene was chosen for similar reasons and because with simply switching the nitrogen atom from $CH_2NH_2^+$ with a carbon atom makes it possible to reuse the geometries from the methylenimmonium cation [30].

5.1 Training with $CH_2NH_2^+$



(a) SchNarc 2.0 (b) SPaiNN

Figure 5.1: Scatter plots of two models, with different representations, trained with the $CH_2NH_2^+$ data base from ref. [14] with three singlet states. The same hyperparameters and split files were used for both models.

As a first step, the scatter plots for SchNarc 2.0 and SPaiNN models trained with the $CH_2NH_2^+$ data set, for energies, forces, NACs and (transition) dipole moments were compared (fig. 5.1). In these plots no differentiation between states, coordinates and/or atoms was made. Each point represents a comparison between a single element of the predicted and reference tensor over the whole test set.

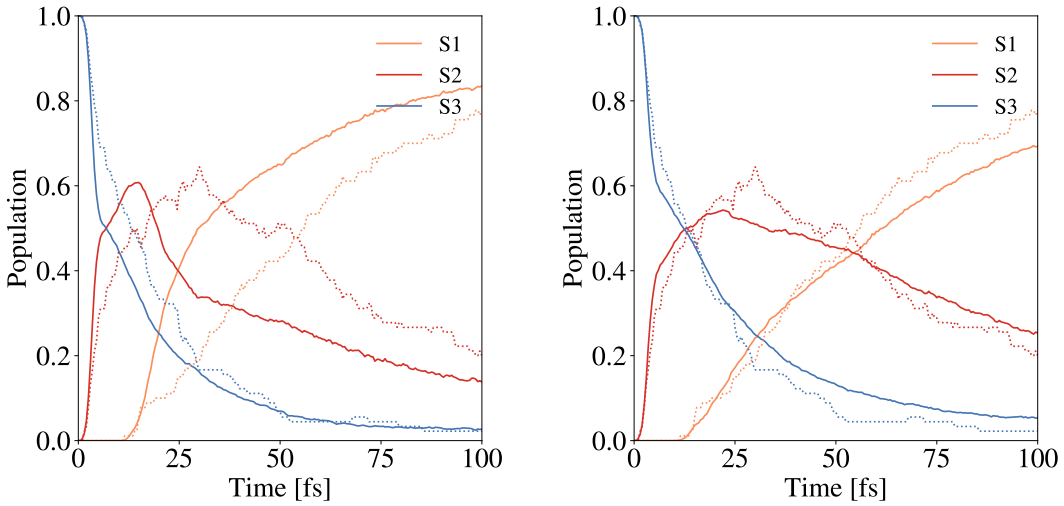
From the visual perspective, the differences between SchNarc 2.0 and SPaiNN are hardly noticeable when looking at energies and forces, which is not surprising since these properties do not directly benefit from an equivariant representation. However, the difference becomes more apparent when comparing dipole moments and NACs. While the distinction may not be drastic, there is still a noticeable difference between the two when viewed alongside each other. When putting an imaginary regression line through the NAC plots, it seems that the slope of the model trained with SPaiNN is close to 1, while the SchNarc 2.0 model is slightly steeper. For the dipole moments, the imaginary lines of both models are approximately 1, which is ideal, but the points in the SchNarc 2.0 model are more spread out around the ideal line than with the SPaiNN model.

Table 5.1: MAE of both $CH_2NH_2^+$ models for each property. Lower errors indicated in bold.

Property	MAE	
	SchNarc 2.0	SPaiNN
Energy	0.06721	**0.06710**
Forces	0.00640	**0.00387**
NACs	0.14553	**0.10819**
Dipoles	0.03919	**0.02604**

Like in the scatter plots, no differentiation between states, coordinates and/or atoms was made. The MAE was calculated between every single element of the prediction and the reference tensor over the whole test set. SPaiNN outperforms SchNarc 2.0 in all properties when evaluated using the MAEs of the predicted test set (tab. 5.1). This suggests that SPaiNN is more accurate and reliable for predicting these properties.

5.1.1 Excited-State Dynamics



(a) SchNarc 2.0

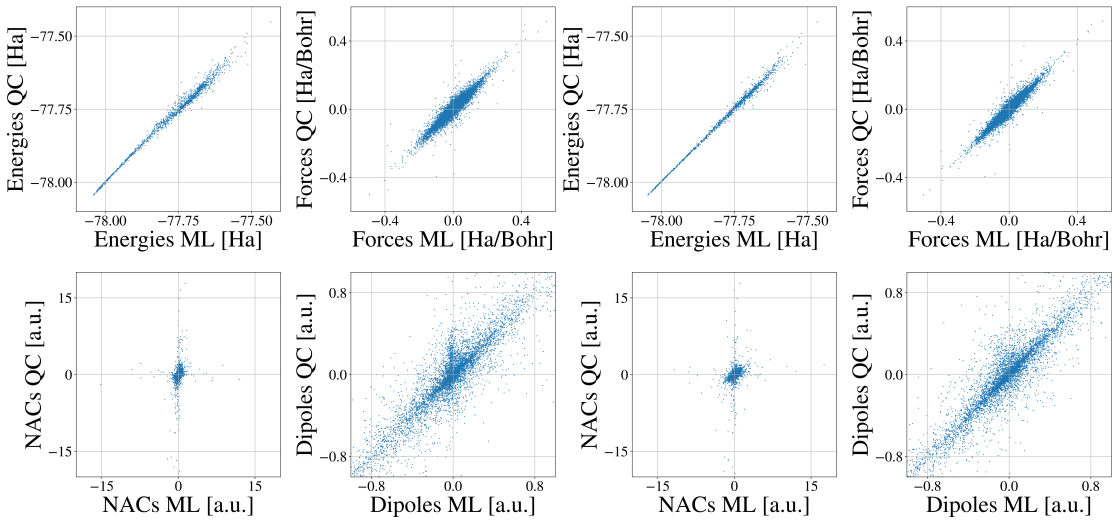
(b) SPaiNN

Figure 5.2: Population curves from dynamic simulations of $CH_2NH_2^+$ with two different ML models. Solid lines are the populations obtained from 3846 $CH_2NH_2^+$ trajectories from the ML models. Dotted lines indicate results from 90 $CH_2NH_2^+$ trajectories, calculated with QC (MR-CISD(6,4)/aug-cc-pVDZ) trajectories [14].

The trend continues in the dynamics (fig. 5.2), where the simulations with SPaiNN are significantly better than the simulations with SchNarc 2.0, resulting in more accurate

population transfers over the whole time scale when compared to the reference. Looking at the SchNarc 2.0 dynamics, only the decrease of the population from state 3 agrees well with the reference data, the population of state 2 increases faster than in the reference at the beginning and then drops fast until around 30 fs and then continues decreasing almost linearly coming closer to the reference at the end of the simulation. The population of state 1 starts increasing about the same time, but steeper as the reference, at the end of the simulation the population is coming closer to the reference population. On the contrary all populations of the three states simulated with SPaiNN are well reproduced compared to the reference.

5.2 Training with ethylene



(a) SchNarc 2.0 (b) SPaiNN

Figure 5.3: Scatter plots of two models, with different representations, trained with the ethylene data base [19] with two singlet states. The same hyperparameters and split files were used for both models.

The scatter plots of the models trained with the ethylene data set (fig. 5.3) were created in the same manner as for $CH_2NH_2^+$ (fig. 5.1). SPaiNN consistently outperforms SchNarc 2.0 in the prediction of properties. A visual comparison of the results for ethylene (fig. 5.3) shows that SPaiNN produces more accurate and reliable predictions. Unlike with $CH_2NH_2^+$ (see section 5.1) there is a noticeable difference in the energy plots between the two representations even though an equivariant representation should not improve the prediction of this property. Also the difference in the NAC and transition dipole plots are immediately noticeable and even larger than between the $CH_2NH_2^+$ models.

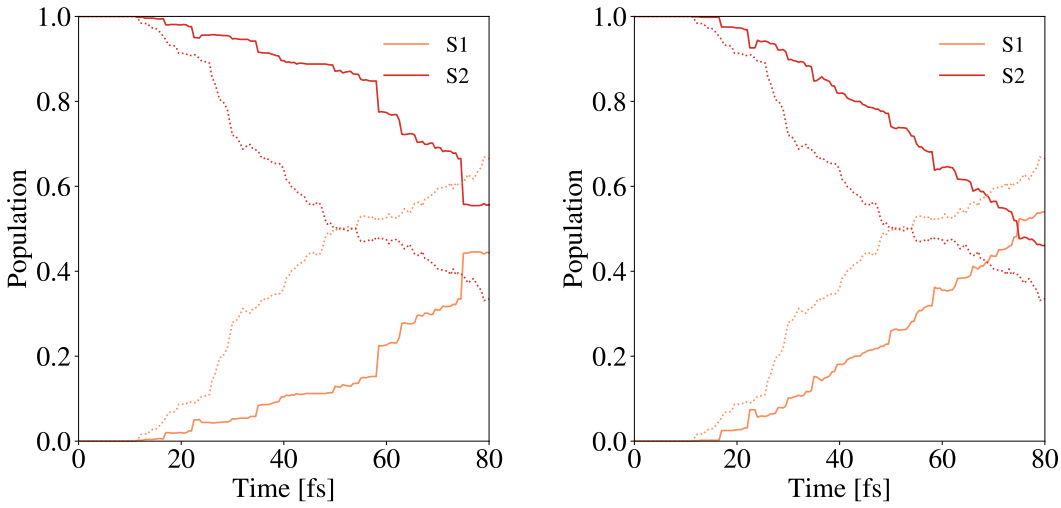
Table 5.2: MAE of both ethylene models for each property. Lower errors indicated in bold.

Property	MAE	
	SchNarc 2.0	SPaiNN
Energy	**0.08775**	0.08858
Forces	0.01012	**0.00814**
NACs	0.23886	**0.21439**
Dipoles	0.11870	**0.10133**

The MAE was calculated in the same manner as with the $CH_2NH_2^+$ trained models (tab. 5.1), no differentiation between states, coordinates and/or atoms was made. Although the energies (fig. 5.3) appear to be better predicted by SPaiNN, the MAE (tab. 5.2) is slightly lower with SchNarc 2.0. It is surprising that the forces have a lower MAE with SPaiNN since these are the derivatives of the energies with respect to the atomic positions.

The reason for this could be an artefact in one of the states produced by inconsistencies in the quantum chemical calculations, which is not reproduced with ML [14] (see ref. [14], supporting information, fig. S5).

5.2.1 Excited-State Dynamics



(a) SchNarc 2.0

(b) SPaiNN

Figure 5.4: Population curves from dynamic simulations of ethylene with two different ML models. Solid lines are the populations obtained from 1000 ethylene trajectories [19] from the ML models. Dotted lines, 1000 ethylene trajectories, calculated with QC (SA(3)-CASSCF(2/2)/6-31G*) trajectories [14].

For both models the agreement with the reference simulation based on QC potentials (fig. 5.4) could be better. Trying different hyper parameters and extending the data set would probably improve the dynamics, but that is not pursued here. It can be shown that SPaiNN is consistently better than SchNarc 2.0. The point where the populations of both states cross, is somewhere around 60 fs in the reference, the crossing of the populations with SPaiNN is at around 75 fs and beyond the maximal simulation time of 80 fs with SchNarc 2.0.

5.3 Performance of Phase-less Loss Implementations

To test the performance of both phase-less loss (phase-less MSE) implementations in SchNarc [14] and SchNarc 2.0/SPaiNN (see section 3.1), the execution time for two random tensors in the shape of [couplings*6*3] with various number of couplings was measured. In order to get more reliable numbers, the execution time was measured for 10.000 repetitions. This was done three times for both implementations and the result was averaged over the three measurements.

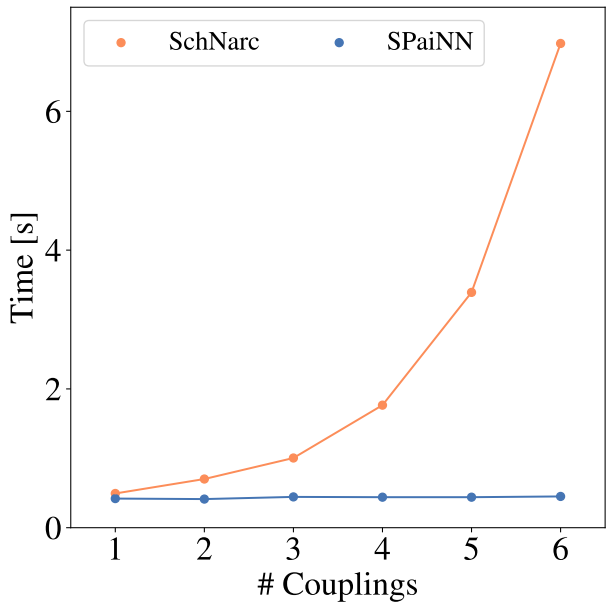


Figure 5.5: Comparison of the time needed to calculate the phase-less loss of n couplings with random data.

Like it was predicted in section 3.1, the phase-less loss implementation in SchNarc [14] scales exponentially with the number of couplings (see fig.5.5). The new phase-less loss implementation in SchNarc 2.0 and SPaiNN clearly outperforms the previous implementation even with a small number of couplings.

Table 5.3: Averaged execution times for both phase-less loss implementations with various number of couplings.

Number of couplings	Time [s]	
	SchNarc	SPaiNN
1	0.493	0.420
2	0.702	0.413
3	1.005	0.444
4	1.766	0.440
5	3.391	0.440
6	6.978	0.451

The scaling of the improved algorithm could not be determined from the measurements due to the fast execution time and high standard deviation from sampling just three measurements (see tab.5.3). Increasing the number of repetitions per measurement by some orders of magnitude would give a clearer picture of the true scaling of the algorithm,

but this was not attempted since the difference between the two implementations is large enough to leave no doubt about performance.

6 Conclusion and Outlook

In this work, the excited-state neural network package SchNarc [14] was recreated from scratch with the latest python libraries and SchNetPack 2.0 [4], which includes the equivariant representation PaiNN [18]. The new implementation is named SchNarc 2.0 or SPaiNN, depending on which representation is used to predict molecular properties. Using an equivariant representation to directly predict equivariant properties like NACs and (transition) dipole moments has significantly improved the prediction accuracy compared to the invariant representation SchNet [17] and it even improved the prediction accuracy for invariant properties like the energies. With the improved prediction accuracy from PaiNN it was possible to run excited-state molecular dynamics with two test systems (ethylene, $CH_2NH_2^+$), that resulted in a qualitatively better description of the population transfers that agree well with reference data. Beside the improvement of predictions, there were also algorithmic improvements that significantly speed up the training with SchNarc 2.0 and SPaiNN. The most notable improvement was made with a new implementation of the phase-less loss, which reduced the scaling from exponential to linear. Although directly predicting the equivariant NACs with PaiNN has been proven to be the better method, there is still a lot of room for improvement. The arbitrariness of the wave function phase and the singularities and cusps of the NACs still makes them hard to learn for machine learning models. A promising approach to make learning the NACs more reliable was recently proposed by Richardson [31] and will be implemented and tested in future versions of SchNarc 2.0/SPaiNN.

List of Tables

List of Figures

Bibliography

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners (2020), arXiv:2005.1465.

[2] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, Zero-Shot Text-to-Image Generation (2021), arXiv:2102.12092.

[3] M. M. Kasar, D. Bhattacharyya, T. hoon Kim, Face Recognition Using Neural Network: A Review, *International Journal of Security and Its Applications*, **10**, 81–100 (2016).

[4] K. T. Schütt, S. S. P. Hessmann, N. W. A. Gebauer, J. Lederer, M. Gastegger, SchNetPack 2.0: A neural network toolbox for atomistic machine learning (2022), arXiv:2212.05517.

[5] M. Goel, S. Raghunathan, S. Laghuvarapu, U. D. Priyakumar, MoleGuLAR: Molecule Generation Using Reinforcement Learning with Alternating Rewards, *Journal of Chemical Information and Modeling*, **61**, 5815–5826 (2021).

[6] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, D. Hassabis, Highly accurate protein structure prediction with AlphaFold, *Nature*, **596**, 583–589 (2021).

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, *Nature*, **529**, 484–489 (2016).

[8] OpenAI, :, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, S. Zhang, Dota 2 with Large Scale Deep Reinforcement Learning (2019), arXiv:1912.06680.

[9] C. D. Rankine, M. M. M. Madkhali, T. J. Penfold, A Deep Neural Network for the Rapid Prediction of X-ray Absorption Spectra, *The Journal of Physical Chemistry A*, **124**, 4263–4270 (2020).

[10] A. Chouai, S. Laugier, D. Richon, Modeling of Thermodynamic Properties Using Neural Networks: Application to Refrigerants, *Fluid Phase Equilibria*, **199**, 53–62 (2002).

[11] K. T. Schütt, S. S. P. Hessmann, N. W. A. Gebauer, J. Lederer, M. Gastegger, SchNetPack 2.0: A neural network toolbox for atomistic machine learning (2022), arXiv:2212.05517.

[12] T. R. Nelson, A. J. White, J. A. Bjorgaard, A. E. Sifain, Y. Zhang, B. Nebgen, S. Fernandez-Alberti, D. Mozyrsky, A. E. Roitberg, S. Tretiak, Non-adiabatic Excited-State Molecular Dynamics: Theory and Applications for Modeling Photophysics in Extended Molecular Materials, *Chemical Reviews*, **120**, 2215–2287 (2020).

[13] B. Bachmair, M. M. Reiner, M. X. Tiefenbacher, P. Marquetand, Recent advances in machine learning for electronic excited state molecular dynamics simulations, in: *Chemical Modelling: Volume 17*, volume 17, 178–200, The Royal Society of Chemistry (2023).

[14] J. Westermayr, M. Gastegger, P. Marquetand, Combining SchNet and SHARC: The SchNarc Machine Learning Approach for Excited-State Dynamics, *The Journal of Physical Chemistry Letters*, **11**, 3828–3834 (2020).

[15] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, K.-R. Müller, SchNetPack: A Deep Learning Toolbox For Atomistic Systems, *Journal of chemical theory and computation*, **15**, 448–455 (2019).

[16] S. Mai, M. Richter, M. Heindl, M. F. S. J. Menger, A. Atkins, M. Ruckenbauer, F. Plasser, L. M. Ibele, S. Kropf, M. Oppel, P. Marquetand, L. González, SHARC2.1: Surface Hopping Including Arbitrary Couplings — Program Package for Non-Adiabatic Dynamics, sharc-md.org (2019).

[17] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, K.-R. Müller, SchNet: A continuous-filter convolutional neural network for modeling quantum interactions (2017), arXiv:1706.08566.

[18] K. T. Schütt, O. T. Unke, M. Gastegger, Equivariant message passing for the prediction of tensorial properties and molecular spectra (2021), arXiv:2102.03150.

[19] L. M. Ibele, B. F. E. Curchod, A molecular perspective on Tully models for nonadiabatic dynamics, *Physical Chemistry Chemical Physics*, **22**, 15183–15196 (2020).

[20] F. Plasser, H. Lischka, *Multi-Reference Configuration Interaction*, chapter 9, 277–297, John Wiley & Sons, Ltd (2020).

[21] M. V. Berry, Quantal phase factors accompanying adiabatic changes, *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, **392**, 45–57 (1984).

[22] J. Westermayr, M. Gastegger, M. F. S. J. Menger, S. Mai, L. González, P. Marquetand, Machine learning enables long time scale molecular photodynamics simulations, *Chem. Sci.*, **10**, 8100–8107 (2019).

[23] M. Barbatti, Nonadiabatic dynamics with trajectory surface hopping method, *WIREs Computational Molecular Science*, **1**, 620–633 (2011).

[24] S. Mai, P. Marquetand, L. González, Nonadiabatic dynamics: The SHARC approach, *WIREs Computational Molecular Science*, **8** (2018).

[25] S. Agatonovic-Kustrin, R. Beresford, Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research, *Journal of Pharmaceutical and Biomedical Analysis*, **22**, 717–727 (2000).

[26] R. Mercado, T. Rastemo, E. Lindelöf, G. Klambauer, O. Engkvist, H. Chen, E. Bjerrum, Graph Networks for Molecular Design, *Machine Learning: Science and Technology*, **2** (2020).

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural Message Passing for Quantum Chemistry (2017), arXiv:1704.01212.

[28] I. Loshchilov, F. Hutter, Decoupled Weight Decay Regularization (2017), arXiv:1711.05101.

[29] J. M. Westermayr, Machine learning for excited-state molecular dynamics simulations (2020), doi:10.25365/THESIS.63947.

[30] J. Westermayr, P. Marquetand, Deep learning for UV absorption spectra with SchNarc: First steps toward transferability in chemical compound space, *The Journal of Chemical Physics*, **153**, 154112 (2020).

[31] J. O. Richardson, Machine learning of double-valued nonadiabatic coupling vectors around conical intersections, *The Journal of Chemical Physics*, **158**, 011102 (2023).

Acronyms

CI Configuration Interaction. 5

GNN Graph Neural Network. 9

LR Learning Rate. 15

MAE Mean Absolute Error. 11, 21, 23, 29

ML Machine Learning. 11, 19, 21, 23, 24, 31

MPNN Message Passing Neural Network. iii, vii, 9

MRCI Multi-Reference Configuration Interaction. 5

MSE Mean Squared Error. 11, 15, 24

NAC Non-Adiabatic Coupling. iii, vii, 1, 2, 12–15, 19–23, 27, 31

PaiNN Polarizable Atom Interaction Neural Network. iii, v, 1–3, 13, 14, 19, 27

SHARC Surface Hopping including Arbitrary Couplings. iii, v, 1, 14

SOC Spin Orbit Coupling. 1, 19

SPaiNN A combination of SHARC and PaiNN. iii, v, 14, 19–25, 27, 29