

Sparse graph attention networks as efficient ionic liquid potentials

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Stefan Hödl, BSc

Matrikelnummer 01452750

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Allan Hanbury

Mitwirkung: Dr. Jesús Carrete Montaña, Theoretical Materials Chemistry

Univ.Prof. Georg Madsen, Theoretical Materials Chemistry

Wien, 30. September 2022

Stefan Hödl

Allan Hanbury

Sparse graph attention networks as efficient ionic liquid potentials

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Stefan Hödl, BSc

Registration Number 01452750

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Allan Hanbury

Assistance: Dr. Jesús Carrete Montaña, Theoretical Materials Chemistry
Univ.Prof. Georg Madsen, Theoretical Materials Chemistry

Vienna, 30th September, 2022

Stefan Hödl

Allan Hanbury

Declaration of Authorship

Stefan Hödl, BSc

I hereby declare that I have written this master thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 30th September, 2022

Stefan Hödl

Abstract

This work applies a graph neural network to the prediction of the potential energy surface of ionic liquids and organic molecules. Representing atomistic systems as molecular graphs enables the use of message passing neural networks to propagate information using the graph connectivity structure, which has the potential to replace the need for computationally expensive atomistic descriptors.

Graph Attention Networks are adapted to operate on sparse molecular graphs and perform message passing to propagate information using *attention* as an update rule. After multiple message passing layers, a hierarchical regression readout head transforms the obtained features into atomic contributions. The sum of these contributions is the model's prediction of the potential energy, while the atomic forces are calculated as its derivatives.

An efficient JAX implementation leveraging a GPU and just-in-time compilation enables molecular dynamics simulations using the forces, while sparsity avoids the quadratic computational complexity of self-attention and enables the model to efficiently scale to larger systems.

This work adapts the implementation from *NeuralM*, a descriptor-based architecture using spherical Bessel descriptors and a neural network, which serves as a baseline against which to evaluate the model's predictive accuracy and speed. The model achieves chemical accuracy when predicting the forces of an ionic liquid, while achieving a significantly faster runtime with a $4\times$ speedup during training. Evaluation on the very large ANI-1 dataset shows the model's ability to efficiently predict the energy of small organic molecules with chemical accuracy and describe displaced molecular states such as during dissociation or vibration.

Kurzfassung

Diese Masterarbeit verwendet Graphen-basierte neuronale Netzwerke zur Vorhersage der potentiellen Energieoberfläche von organischen Molekülen und ionischen Flüssigkeiten. Die Repräsentation von atomistischen Systemen durch Graphen ermöglicht die Verwendung sogenannter *message passing neural networks*, um Information über die Graphenstruktur an benachbarte Atome zu verbreiten. Dies ermöglicht es, die herkömmlich verwendeten und rechnerisch sehr teuren *Fingerabdrücke* zu vermeiden.

Graph attention networks werden adaptiert, um einen dünnbesetzten molekularen Graphen zu generieren und durch den *Attention*-Mechanismus gewichtete Nachrichten zu versenden. Nach mehreren solchen Ebenen werden die Energiebeiträge durch eine hierarchischen Regressions-Kopf ausgelesen. Die Summe dieser atomistischen Beiträge ist die Vorhersage der potentiellen Energie des Moleküls. Die auf die Atome wirkenden Kräfte werden als Ableitung des Modells evaluiert.

Eine effiziente Implementierung in JAX macht Nutzen von einer Grafikkarte sowie *just-in-time* Kompilierung und ermöglicht dadurch die Simulation der temporalen Entwicklung molekularer Systeme durch Integration der Kräfte. Die quadratische Komplexität des Attention-Mechanismus wird durch Berechnung auf dem dünnbesetzten Graphen vermindert, wodurch Skalierbarkeit zu größeren Systemen ermöglicht wird.

Diese Arbeit adaptiert die Implementation von *NeuralIL*, einer Fingerabdruck-basierten Architektur zur Berechnung der Energie und Kräfte von ionischen Flüssigkeiten. Die Methode nutzt kugelförmige Bessel-Fingerabdrücke sowie ein neuronales Netzwerk und dient als Referenzmethode zum Vergleich der Genauigkeit und Geschwindigkeit des Graphen-basierten Modells.

Das Modell erreicht chemische Genauigkeit bei der Vorhersage der Energie und Kräfte von ionischen Flüssigkeiten bei einem $4\times$ Geschwindigkeitsvorteil beim Trainieren. Evaluierung auf der sehr großen Datenbank von organischen Molekülen ANI-1 zeigt die Fähigkeit des Modells, effizient die Energie von Molekülen in dissoziierten Zuständen und während der molekularen Vibrationen mit chemischer Genauigkeit vorherzusagen.

Contents

Abstract	vii
Kurzfassung	ix
Contents	xi
1 Introduction	1
1.1 Computational chemistry	1
1.2 Molecular dynamics & screening	2
1.3 Systems & states	3
1.4 Aim of the work	3
2 Literature Review	5
2.1 Computational chemistry	5
2.1.1 Atoms and molecules	5
2.1.2 Approximating the Schrödinger equation	6
2.1.3 Potential energy surface	7
2.1.4 Molecular dynamics	8
2.1.5 Displaced states	8
2.1.6 Organic molecules	9
2.1.7 Ionic liquids	10
2.2 Neural network potentials	11
2.2.1 Computational approaches in chemistry	11
2.2.2 High-dimensional neural network potentials	12
2.2.3 Energy & forces	12
2.2.4 Physical invariances	13
2.2.5 Atom-centered symmetry functions	14
2.2.6 Third- and fourth-generation potentials	15
2.2.7 Baseline potentials	16
2.3 Message passing neural networks on molecular graphs	17
2.3.1 Molecular graphs	17
2.3.2 Replacing fingerprints	19
2.3.3 Message passing neural networks	20
	xi

2.3.4	Graph Attention Networks	23
2.3.5	State-of-the-art models	26
2.4	Summary	31
3	Methods	33
3.1	JAT Architecture	33
3.1.1	Graph generator	35
3.1.2	JAT layer	37
3.1.3	Linear self-attention	38
3.1.4	JAT model	41
3.2	JAT Implementation	43
3.2.1	GPUs and parallel processing	43
3.2.2	Just-in-time compilation	44
3.2.3	Atom and edge masks	45
3.3	Hyperparameters & training	45
3.4	Summary	47
4	Results	49
4.1	ANI-1 results	49
4.2	EAN results	59
4.3	Discussion of ANI-1 results	67
4.4	Discussion of EAN results	68
4.5	Outlook and future work	70
5	Conclusion	71
	List of Figures	73
	List of Tables	75
	Bibliography	77

Introduction

A brief overview highlights the most important concepts of the domain and formulates the primary challenges as well as the goal of the thesis. Concepts such as the potential energy surface and molecular states are introduced, which are addressed in detail in the next chapter.

1.1 Computational chemistry

An accurate description of an atomistic system, such as those studied by chemistry and materials science, requires the use of quantum mechanical methods. Although the Schrödinger equation accurately describes the dynamics, an analytical solution is only possible for the simplest systems, and for non-trivial numbers of atoms, the computational and storage requirements of a direct approach to the quantum-mechanical equations are intractable, a problem known as the exponential barrier. [Koh99]

Approximation methods such as those based on density functional theory (DFT) achieve chemical accuracy at reasonable computational cost but do not scale nearly as well as would be needed to routinely deal with thousands of particles. [Koh99] On the other hand, classical force fields cost little to run, but are inherently limited in their accuracy and lack transferability beyond the configurations or dataset the model was parametrized for. [vLB20]

Using neural network force fields (NNFFs) to tackle the computational complexity and scaling has been proven a promising approach for prediction of the potential energy surface (PES) or other properties of interest. [BP07, Beh21] Such potentials are trained in a supervised learning context using a dataset of atomistic configurations to predict the regression targets, obtained from costly DFT-based ab-initio calculations. The availability

and size of high-quality datasets is still a limiting factor in training such models, in particular for molecules which are not in their ground states. [vLB20]

Predictive models need to fulfil the physically relevant invariances, which are commonly captured by the use of atom-centered fingerprints and an appropriate *ansatz*. [BP07] *Translation* and *rotation* do not affect the potential energies or scalar properties, and hence predictions need to be invariant to such transformations of the input coordinates. Since two atoms of the same species are chemically identical, exchanging atoms in the input (a *permutation*) or a change in indexing should not change the model's output.

In recent years, graph-based deep learning approaches have emerged as an alternative approach. They leverage neural message passing to learn representations of the molecular graph instead of the conventionally used atomistic fingerprints capturing the local chemical environment around each atom. This end-to-end learning approach thus avoids the expensive descriptor calculation and achieves favourable scaling for larger systems while respecting invariances and retaining chemical accuracy. [GSR⁺17, GY21, FZJS21]

1.2 Molecular dynamics & screening

In molecular dynamics (MD) the evolution of atomistic systems in time is simulated by numerically integrating the classical equations of motion for the nuclei using the forces acting on each atom for a small time step. [UCS⁺21] Calculating the atomic forces is comparable in computational cost to only calculating the energy, however the relevant timescales necessitate millions of evaluations to simulate a few nanoseconds of a system. Inference using trained NNFFs is many orders of magnitude faster than DFT-based methods and comparable to classical force fields in runtime, while still achieving chemical accuracy. [GSR⁺17]

Another important use case of such potentials is to facilitate the exploration of the vast space of chemically viable configurations in search of new and promising candidates for a given application or domain. Exhaustively searching this space for organic molecules is infeasible, but more generally only a small set of configurations can practically be evaluated using DFT-based approximations given their computational cost. Instead, efficient machine learning potentials serve as the primary filter for *in silico screening* to discover novel and promising compounds, the best of which are subsequently evaluated using highly accurate ab-initio methods. [VEL⁺19]

A similar application is in the domain of materials science, where approximation of the structural and electronic properties of atomic structures guides the discovery of desirable materials. Many properties of interest can be derived from the potential energy, but DFT-based methods are not prescriptive and give no guidance on how to do so. Instead, machine

learning approaches can be used to construct latent representations from the atomic configurations, from which relevant properties of bulk or surface structures, crystals, liquids or organic molecules can be predicted directly using conventional classification or regression heads. Although efficiency is still important to screen the vast space of possible configurations, the model's inference time is far less critical in this domain compared to MD simulations. [FZJS21]

1.3 Systems & states

Beyond the multitude of molecules and systems of interest depending on the domain, the state of the molecule or system plays an important role for the accuracy of predictive models. Many datasets evaluate a molecule's energy and properties in the local neighbourhood of their *ground state*, after *relaxation* of the atomic positions until all atomic forces reach zero. Characterizing a single basin of attraction of the PES requires sampling of the molecule outside of its ground state, which a model trained exclusively on ground state configurations cannot accurately reproduce. Many aspects of chemical and practical relevance however involve transitions between different attraction basins of the PES.

In particular, the dissociation of an atom from the molecule is one such aspect where current approaches commonly fall short, both for the lack of such interactions in benchmark datasets as well as due to some architectures making dissociation impossible by design, such as when treating molecules as static units. Dissociation however is a frequently occurring process during chemical reactions and forms the basis of technologies such as *electrospray propulsion*. [KMHF⁺15]

1.4 Aim of the work

By adapting improved graph attention networks (GATv2) [VCC⁺17, BAY21] to the task of predicting the PES of ionic liquids and organic molecules, this work aims to improve upon the status quo and address shortcomings of current NNFFs as well as explore the applicability of graph-based approaches for the study of molecular systems. Of special interest are accurate and transferable prediction of systems outside their ground states, such as during dissociation and vibration, which are of practical relevance for chemical reactions and MD simulations but where current methods fall short of being both accurate and fast.

Graph-based potentials have already achieved state-of-the-art results on many tasks and shown promise to address shortcomings of current NNFFs as well as dispensing with the need for fingerprints. Their application to various tasks and systems of practical relevance makes graph neural networks an especially interesting architecture which might be able to achieve competitive performance using a general architecture, rather than

through the use of hand-crafted domain-specific extensions to NNFFs which trade off marginal performance gains for drastically more complex models. [FZJS21]

The message passing framework (MPNN) [GSR⁺17] generalizes most graph-based approaches so far, which differ primarily in how information from a node's neighbourhood is updated and aggregated. Graph attention networks (GAT) [VCC⁺17] use an attention-based update and can be categorized within the MPNN regime but have not seen much adoption in the domain of materials science and chemistry.

To this end, the design space of MPNNs using an attentional update rule is explored on an ionic liquid dataset as well as the very large ANI-1 dataset of organic molecules [SIR17a] to obtain comparable results to related works. Many considerations are taken in the name of computational efficiency and scalability, such as leveraging a sparse graph implementation, automatic differentiation and compute accelerators (such as a GPU) in JAX [BFH⁺20] to achieve efficient training and inference suitable for MD simulations.

Literature Review

This chapter introduces the chemical and computational background relevant to the problem, and thereafter discusses conventional as well as modern approaches to the task of predicting the potential energy of molecules. The central concepts of the potential energy surface (Sec. 2.1.3) and molecular states (Sec. 2.1.5) are discussed in detail, which motivates the application of deep learning approaches for molecular dynamics simulations. Physical invariances (Sec. 2.2.4) and the additive *ansatz* (Sec. 2.2) are considered, which apply to descriptor-based as well as modern architectures. A discussion of the message passing neural network framework (Sec. 2.3.3), the attention mechanism and the graph attention network architecture (Sec. 2.3.4) follows. The chapter concludes with an extensive review of relevant architectures, baseline approaches and state-of-the-art models (Sec. 2.3.5).

2.1 Computational chemistry

2.1.1 Atoms and molecules

In chemistry, the smallest unit of matter are atoms that belong to chemical elements, which are made up of a nucleus surrounded by one or more electrons. The simplest atom is hydrogen (H), a single negatively charged electron which is bound to the positively charged nucleus by the electromagnetic force. Multiple atoms can form bonds between each other to create stable structures; the simplest molecule is H₂. It is made up of two H atoms forming a *covalent* bond, by which each H atoms shares its single electron to achieve a more stable, lower-energy state.

An incredible diversity of molecules can be formed from the basic elements hydrogen (H), carbon (C), oxygen (O) and nitrogen (N). A molecule's 3-dimensional spatial structure is determined by the electromagnetic interactions between multiple nuclei and many

electrons with each other, and for a given set of atoms many configurations are possible. The lowest-energy configuration, the *ground state*, is the most stable and therefore likely 3D structure, whereas higher-energy states are occupied with exponentially decreasing probabilities corresponding to the discrete increases in energy of those states. For a molecular ensemble at constant temperature, the Boltzmann distribution describes this probability of observing a molecule in a given state.

2.1.2 Approximating the Schrödinger equation

The accurate description of atoms and molecules necessitates modeling the quantum interactions of electrons using the *wave function* and associated atomic energies, which are described by the Schrödinger equation. However, solving it analytically is computationally intractable and approximations are necessary for even for the smallest systems, such as H₂. [Koh99]

The main complexity arises from contributions of higher-order electron-electron interactions, in particular from the exchange-correlation functional contribution to the electronic energy. Molecular quantum mechanics uses computational methods to calculate these contribution efficiently, which operate within the *Born-Oppenheimer* (BO) approximation to model the *electronic structure* and obtain the wave functions and associated energies of the electrons.

By observing the difference in timescales of motion due to the large difference in mass of the nuclei and electrons, the BO approximation assumes the terms can be separated. The *adiabatic* theorem states that due to sufficiently slow timescales of the movement of the nuclei, the electrons experience gradual and continuous perturbations and thus can adapt their functional electronic form. [ZSH05] This allows simplification of the electrodynamic interactions to electrostatics, in which the electrons are only affected by the electrostatic field created by stationary nuclei.

The total *Hamiltonian* operator acting on the molecules' wave function then becomes the sum of the separately evaluated kinetic energy of the nuclei, the Coulomb repulsion of the nuclei as well as the contribution of the electrons. The *electronic* energy depends only on the relative position of the nuclei but not their kinetic terms, thus the electronic Schrödinger equation can be solved to obtain the electronic wave function, from which the total wave function can be calculated. The electronic Hamiltonian consists of three independent contributions, respectively the electrons' kinetic energy, their mutual repulsion as well as electron–nucleus attractions. [ZSH05]

2.1.3 Potential energy surface

Solving the electronic Schrödinger equation with the Hamiltonian of the system thus yields the wave function and energy of the system, which uniquely defines the geometry and stability of the molecule. A change in the position of one of the nuclei alters the energy of the molecule, which again requires solving the aforementioned equations with the new positions of the nuclei to obtain the energy of the new state. Considering a continuous change of one parameter, such as the inter-atomic distance r_{ij} between the two H atoms forming a bond in a H₂ molecule, gives rise to a curve relating the potential energy to the reaction coordinate, such as the bond distance or angle.

The *potential energy surface* (PES) analogously describes the high-dimensional energy landscape for many degrees of freedom, usually for the 3-dimensional atomic position of each of the N nuclei. The PES is invariant to rotation and translation of the system and is therefore defined by $3N - 6$ degrees of freedom for non-linear systems. It is thus possible to define the system using *internal coordinates*, as the absolute position and orientation in space does not change the potential energy.

Stationary points in the PES are of particular relevance, such as local and global minima which characterize stable states of the molecule. Saddle points characterize transitions between stable configurations, which follow the optimal path of minimal energy required to overcome the saddle point's energy barrier. Knowledge of the PES thus fully characterizes not only a molecule's low-energy stable states, but also the most likely transition between states as well as many other properties of interest. Full knowledge of the PES is thus highly desirable; however, obtaining it is intractable especially for larger systems due to the many continuous degrees of freedom.

Of great practical interest is thus an accurate yet fast computational method to approximate the potential energy for a given molecular configuration. Constructing this functional mapping from the atomic positions to the energy of the molecule is thus the main task of this thesis, and possession of such a mapping allows for the construction of the full PES for every possible configuration.

In practice, low-energy regions of the configurational space are frequently occupied, while other regions are unlikely to ever be observed during the natural vibrations and transitions of molecules. Fully evaluating the PES is therefore not necessary, and a representative sampling of the PES can capture the complex interactions and accurately reproduce its dynamics.

However, this means generalization to sparsely sampled regions of the PES will occasionally be necessary and for such regions the accuracy of the mapping will likely diminish. This issue is described by the *curse of dimensionality*, which formalizes the difficulty of

densely sampling a volume of space when the dimensionality increases. It suggests that only very few samples in the training dataset will be reasonably close and similar to a new configuration, and even those samples will be different in important ways.

2.1.4 Molecular dynamics

With an adequate method to approximate the energy of a configuration, the *forces* acting on each of the atoms can be evaluated as the model's gradient with respect to the atomic positions. Importantly, a force field obtained this way is *conservative*, satisfying the property that work done by the forces only depends on its endpoints, but not its trajectory.

With the forces, Newton's equations of motions can be numerically integrated for a small time step Δt , yielding new positions for every atom. This method is known as *molecular dynamics* (MD), and multiple integrators are available for this purpose, such as the *Verlet* algorithm or *Euler's method*. Repeating this procedure for many time steps yields the temporal evolution of the molecular system.

At the atomic scale, the necessary time step is on the order of femto- (fs, 10^{-15} s) to picoseconds (ps, 10^{-12} s). Millions of time steps and therefore evaluations of the position-forces mapping are necessary to simulate a short segment of the temporal evolution of the system, and the computational demands for speed and accuracy are therefore very high. Beyond that, errors accumulate throughout the MD simulation, and even small inaccuracies can lead to divergent or unphysical behaviour, especially if they are systematic and not random errors. With MD simulations, the diverse range of molecular states, vibrations and transitions can be inspected to gain insight into the properties, transition energies or characteristic distributions of molecules.

2.1.5 Displaced states

Many datasets describe a molecule's properties in their ground state, obtained by relaxing the system in time until all forces are equal to zero. However, in reality a molecule only ever occupies its true ground state at 0 K, and *vibrates* within a stable state's attraction basin at characteristic frequencies depending on the size and energy of the molecule as well as the temperature. A non-linear molecule with N atoms has $3N - 6$ normal modes of vibration, termed its *characteristic frequencies*.

Capturing the behaviour of the system outside of its ground state is thus of relevance, in particular considering the molecule is never actually observed to remain there. An important aspect is *dissociation*, such as the removal of a H^+ proton from a molecule, which happens on a regular basis and is chemically important but not accurately captured with common approximations.

Larger molecular systems harbor further obstacles to a good approximations of the PES, where long-range and higher-order interactions might drastically influence the properties and dynamics of a material. The temperature of a system also influences which transitions a molecule undergoes. A higher temperature changes the occupation probabilities of the PES and thus makes easier to overcome the energy barrier in a transition path.

2.1.6 Organic molecules

Individual organic molecules by themselves show incredible diversity and function, and searching for useful candidates for downstream applications necessitates exploration of the vast space of chemical compounds.

Increasing the number of atoms forming a molecule yields many more relevant structures due to increases from three distinct sources. First, an *isomer* is a molecule with the same composition of atoms. More heavy (non-hydrogen) atoms thus make possible many more sensible combinations to form a molecule, in addition to more hydrogen atoms. For a given isomers, many *molecules* are possible, each of which has many *conformations*.

A *molecule* refers to a particular arrangement of atoms with a fixed set of bonds. Transformation of one molecule to another requires breaking and forming bonds to alter which atoms are bound to what, which requires overcoming a significant energy barrier. Since all atoms of the same species are identical, exchanging two atoms of the same element does not yield a new molecule.

Finally, a *conformation* is a snapshot of a molecule's position in space, which continuously changes during the molecule's vibrations. Larger molecules have many normal modes of vibration, and hence many samples are necessary to exhaustively describe the states of a molecule.

Even small molecules with ~ 10 heavy atoms can have thousands of chemically viable configurations with different properties. The QM9 [RDRVL14] dataset includes chemically relevant small organic molecules with up to 9 heavy atoms in their ground state, relating the 3-dimensional structure to the energy as well as other properties of interest. The molecule $C_7O_2H_{10}$ is one of the largest in the QM9 dataset and has ~ 5950 *conformers*, different structural configurations made of the same atoms. The energy and targets of interest are calculated using DFT, the current go-to computational method due to its good scalability and reasonable accuracy.

2.1.7 Ionic liquids

Ions are molecules with a net positive (cation) or negative (anion) charge. Compared to electrically neutral molecules, ions tend to interact with ions of the opposite charge to achieve an overall neutral and more stable state. This makes ions more difficult to model, but their charge generally endows them with functional properties.

A *salt* is an assembly of cations and anions with an overall neutral charge, which is referred to as an *ionic liquid* (IL) in its liquid state. ILs generally have a high *melting point*, the temperature at which the state of the substance transitions from the solid to the liquid phase. ILs exist in their liquid state at room temperature and since they conduct electricity, it makes them very interesting for many potential applications ranging from catalysis to energy storage and green solvents. [TTOK10, MBL04, ES00, JKLR06]

Solvents are critical to many chemical processes, and IL have promising properties as *green solvents* in which reactions can be carried out. Since ILs have negligible *vapor pressure*, almost no compounds *vaporize* and escape, which both reduces the cost and environmental harm as the IL can be recycled and reused. [ES00]. Understanding the influence of the choice of base ions on the properties is of fundamental importance to guide the design of such *designer solvents*. It is possible to train a computational model to directly predict the properties of interest or even generate compounds with such attributes directly. [MDM⁺20]

The first discovered example of an IL is ethylammonium nitrate (EAN, $C_2H_5NH_3NO_3$). [Wal14] It consists of the nitrate anion NO_3^- and the ethylammonium cation $C_2H_5NH_3^+$. EAN has a very low melting point of 12°C and is therefore in its liquid state well below room temperature. [MBL04]

Exchanging the anion or cation in the IL gives rise to different properties, among them the melting point, solubility, polarity and density. Considering the large range of base anion and cation choices and their combinatorial nature, ILs with desirable properties can be designed specifically with an application in mind. [MBL04]

Around 1 billion RTILs are estimated to be possible [MBL04, ES00], requiring a computational approaches for the efficient *screening* of the vast space of possible compounds in the search for desirable molecules. Promising candidates from the screening process are further modeled with highly accurate computational methods and only a few of the best candidates can eventually be validated experimentally. [VEL⁺19]

2.2 Neural network potentials

2.2.1 Computational approaches in chemistry

Computational approaches are required for the approximation of the energy of molecules and compounds since analytical solutions of the Schrödinger equations are intractable for even the simplest systems. Accurate and inexpensive predictions are therefore of great relevance, yet one cannot obtain both but has to choose among one of many methods with their respective assumptions, simplifications and limitations. A rich literature of methods exist all along the spectrum of the accuracy-speed trade-off, ranging from rapid force fields to highly accurate quantum-mechanical ab-initio calculations.

Many methods can be then summarized as using an empirical or theoretical parametrization for each of the individual terms contributing to the total energy, with the optional addition of more elaborate correction terms. The crux of all methods is the electron-electron interaction term, which contributes only a fraction of the total energy yet demands the vast majority of computational efforts. Since an electron is coupled with all other electrons, the *Coulomb* and *exchange* interaction terms cannot be further isolated or simplified, but exhibit many-body effects and result in the so-called *exponential barrier* due to the added degrees of freedom from each additional electron.

Some of the most prominent methods are those based on *density-functional theory* (DFT), a set of reasonably accurate ab-initio methods with good scalability, which is based on modeling the ground-state density of a system of interacting electrons. In theory, all relevant properties can be derived from the ground-state density, and thus explicitly solving the electronic Schrödinger equation can be avoided. [Koh99]

However, DFT-based methods are descriptive rather than prescriptive, and therefore it is generally unknown *how* to derive them. While DFT-based methods are able to scale up to larger systems of thousands of atoms, even small compounds require hours of computation on powerful compute clusters. Despite this, DFT is one of the workhorses of computational chemistry and the go-to method for the calculation of the energy and properties for larger molecular systems. [Beh21]

For MD simulations or screening however, hours of computation for every step is orders of magnitude too slow to be of practical interest. Empirical potentials such as atomistic force fields, among them OPLS-AA [JMTR96], are used when millions of evaluation steps are routinely required. OPLS-AA operates on a timescale of (fractions of) a second per evaluation, which allows for the simulation of much larger systems and longer timescales. However, such methods achieve great speed at the expense of accuracy, in part due to the many simplifications and assumptions required to achieve favourable speed and scaling.

Machine learning (ML) methods are promising candidates, which learn the functional mapping between the atomic positions and their potential energy directly. Neural networks (or generally, any ML approximator) are trained in a supervised context on a reference dataset of positions and energies, which are calculated using DFT-based methods. Once trained, evaluating a new configuration is rapid and comparable in speed to force fields, but approaches the accuracy and generalization of ab-initio methods when given sufficient data to train on. ML-based methods are thus able to strike a favourable balance on the accuracy-speed trade-off. [Beh21]

2.2.2 High-dimensional neural network potentials

Although neural networks have been used to parametrize potentials for very small systems for many years, the introduction of *high-dimensional* neural network potentials (HDNNPs) in 2007 addressed major limitations of earlier approaches and allowed ML potentials to achieve good accuracy. The ideas introduced by Behler and Parrinello are discussed in the following sections, which highlight some key insights prevalent in current state-of-the-art approaches and motivate the introduction of novel architectures to overcome their limitations. [BP07]

Second-generation HDNNPs represent the total potential energy E_{pot} as a sum of atomic contributions (Sec. 2.2.3) and propose the use of separate dense layers and parameters for each atomic species. Put together, these two adjustments ensure the network computes permutation invariant predictions, which previous ML-based approaches struggled with.

Central to their approach was the introduction of a new type of *symmetry function*, which fulfils desirable qualities. These descriptors map the set of Cartesian coordinates to a feature vector for each atom, which describes the local chemical environment of its neighbourhood and determines the molecule's energy. Crucially, these functions are designed to yield descriptors which are invariant to rotation and translation (Sec. 2.2.4) and whose size is independent of the number of atoms in the neighbourhood. These descriptors of static shape then serve as the input to the neural networks, which maps an atom's local environment to its contribution to the total energy. [BP07]

2.2.3 Energy & forces

Additive atomic contributions

Central to the prediction of the molecular PES using current ML potentials is their approach of combining the individual energy contributions from each atom to obtain the total energy, which can be viewed as a global pooling layer. Ω (the neural network) maps the input features, the position p_i and atomic species e_i , to a latent space where permutation invariance is subsequently enforced by the sum over atoms. These latent features are transformed by μ into the prediction of the potential energy. Therefore, the

total energy can be decomposed into additive atomic contributions approximated by Ω as long as μ is known.

$$E_{\text{pot}}(\{p_i, e_i\}) = \mu \left[\sum_{i=1}^{n_{\text{atoms}}} \Omega(p_i, e_i) \right] \quad (2.1)$$

Formally, a justification for this approach can be found in the *deep sets* theorem [ZKR⁺17], which proves the existence of μ . Although Eq. 2.1 is a very general approach, to achieve the theoretical properties the latent space needs to be very high-dimensional. Furthermore, multiple configurations with a varying number of atoms are a poor fit for μ , which needs to consider an appropriate normalization and scale for systems of different size.

Practical considerations for training neural networks thus suggest restricting Ω to a scalar output and μ to the identity function. This *additive ansatz* transfers the property of additivity from the latent space to the potential energy, where it becomes an approximation. Although this ansatz has been used empirically for many years, it strikes the balance between generality of the function and practical considerations of training NN potentials. [MCCB⁺21] The Behler-Parrinello architecture assumes $\mu \equiv 1$. [Beh15]

$$E_{\text{pot}}(\{p_i, e_i\}) = \sum_{i=1}^{n_{\text{atoms}}} \Omega(p_i, e_i) \quad (2.2)$$

Atomic forces

Once the potential energy of the molecule has been calculated, the forces acting on each atom can be obtained as the derivative of the energy with respect to the atomic positions.

$$\mathbf{f}_i = \frac{\partial E_{\text{pot}}(\{p_\alpha, e_\alpha\})}{\partial r_i} = - \sum_{\alpha} \frac{\partial E_{\text{pot}}}{\partial p_\alpha} \cdot \frac{\partial p_\alpha}{\partial r_i} \quad (2.3)$$

If the entire network and descriptor generation is algorithmically differentiable, calculating this derivative is computationally inexpensive, with only a linear increase in overall computational complexity. [G⁺89] Using the forces is thus a great way to better leverage each expensive DFT calculation, this way the $3N$ atomic forces can be used in the loss function, possibly in combination with the scalar E_{pot} . Multiple regression targets allows the models to learn more efficiently from fewer samples during the training process. [SKSF⁺17, COJ⁺17]

2.2.4 Physical invariances

Rotational & translational invariance

Special attention is afforded to the issue of implicit group invariances when representing molecules using Cartesian coordinates in free or Euclidean space $E(3)$. Concretely, the

potential energy does not depend on their orientation or position in space, but is *invariant* to translation and rotation (Eq. 2.4). Mathematically, transformations exactly preserving the internal coordinates can be expressed as follows for a group action $\rho \in \mathcal{G}$ and function f acting on the atomic coordinates R . [BBCV21]

$$E_{\text{pot}} = f(\rho(R)) = f(R) \quad (2.4)$$

Similarly, the forces acting on a molecule do depend on orientation, they are *equivariant* and the vector of atomic forces changes exactly with the rotation of the molecule. [BBCV21]

$$\mathbf{f} = \rho(f(R)) = f(\rho(R)) \quad (2.5)$$

Permutation invariance

Since all atoms of the same species are chemically identical, a permutation or exchange of two or more atoms of the same species does not change the energy and properties of a molecule. Similarly, a change in the indexing order of the atomic positions and species doesn't change the molecule. Any model predicting the PES or its properties should therefore perform equivalent computations to obtain identical features and outputs for a permuted input. Naively however, this requirement does not hold for a multi-layer perceptron (MLP) or other ML model, which does depend on the order of its input.

Exclusively using internal coordinates such as the inter-atomic distance instead of Cartesian coordinates makes the architecture permutation invariant in combination with the sum over atomic contributions. Since the edges generated for the molecular graph only depend on the pairwise distance, permuting the input order of the coordinates would permute the indexing of the atoms and respective edges, however the generated graph would be isomorphic to the original graph. [BBCV21] Since the potential energy obtained is invariant, the calculation of the forces (Eq. 2.3) generates equivariant results for a permuted, translated and rotated set of input coordinates.

2.2.5 Atom-centered symmetry functions

Following the introduction of novel symmetry functions [BP07], a large number of methods based on variations and improvements of the atomic descriptors have been proposed, which build upon the general template of second-generation HDNNPs. Atom-centered neural networks take as input the feature vectors produced by one of many hand-crafted fingerprint methods and transform them into the energy contributions. An extensive review of the numerous proposed fingerprint methods is given by Bartók et. al. [BKC13], while Faber et. al. investigate how the choice of descriptor impacts the models' performance [FHH⁺17].

Despite the significant gains in computational speed compared to DFT-based methods, calculating the fingerprints is drastically more expensive than evaluating the neural network and remains the primary bottleneck. However, reducing the complexity of the descriptors by optimizing the hyperparameters or considering smaller neighbourhoods improves the situation only marginally and yields less accurate predictions.

One notable example for atomistic fingerprints are *spherical Bessel descriptors*, which focus on minimizing redundant information in the descriptors by choosing a minimal basis set based on spherical harmonics. Beyond respecting invariances and being twice-differentiable, these descriptors are *complete*, they allow for the atomic environment to be reconstructed up to symmetry. The local neighbourhood is described by projecting the neighbour density function onto a set of orthonormal basis functions, which are based on spherical harmonics and radial basis functions. [KME20]

Compared to the commonly used *Smooth Overlap of Atomic Position* (SOAP) [BKC13] descriptors, the Bessel descriptors uses different radial basis functions which are much faster to compute. The Bessel descriptors have desirable qualities such as enabling the efficient calculation of atomic forces and avoiding redundant information. Reasonable hyperparameter choices for the number of basis functions and order of approximation yield, for example, 150 features per atom, which represent the local chemical environment within a spherical neighbourhood of 3.5 Å (Angstrom). [MCCB⁺21]

2.2.6 Third- and fourth-generation potentials

Behler’s reviews [Beh15, Beh21] offer a detailed account of the history and evolution of fingerprint-based HDNNPs, provide suggestions for their construction and discuss their limitations. Behler classifies neural approaches proposed since as *third-* and *fourth-generation* HDNNPs, which respectively attempt to include long-range as well as non-local interactions in their architectures. Beyond being highly task-specific, recent approaches tackle these corrections terms with complex architectures using additional independent neural networks trained on different regression targets. While this can indeed increase their performance, it comes with a significant loss in transferability of such methods to related tasks, such as when datasets include separate regression targets for the long-range interaction terms necessary to fit a secondary neural network.

Since such extensions are often specific to the dataset or method, they generally are neither transferable nor available for comparison on standard benchmarks. Thus, adapting such methods to a new task, dataset or class of compounds is cumbersome at best, and improvements in performance compared to simpler approaches are far from guaranteed. A niche exists for specialized methods which offer highly optimized methods specific to a subdomain or class of compound, such as ReaxFF [SHI⁺16] which requires

reparametrization from system to system. General, fast and robust neural approaches which are applicable to most domains "out-of-the-box" are still not readily available.

2.2.7 Baseline potentials

ANI

Smith et. al. discuss the shortcomings of potentials using Behler's symmetry functions, which all fail to achieve transferability between complex chemical environments. They attribute these issues to the lack of a functional form to create distinguishing features and their limited ability to differentiate between multiple chemical species. Their proposed ANI potential instead uses an updated version of the original symmetry functions to construct *atomic environment vectors* better suited for probing the local angular environment of complex chemical systems. [SIR17b]

The authors then focus on training their potential with high-quality configurations by drawing representative samples, for which they propose *normal mode sampling*. They train their model on organic molecules drawn from the GDB-11 database containing up to 8 heavy atoms and further make their generated ANI-1 dataset available to the community. [SIR17a] Beyond achieving excellent accuracy at great speed, they demonstrate the transferability of their ANI potential through 4 case studies, which exclusively consider molecules larger than used for training with at least 9 heavy atoms. [SIR17b]

NeuralIL

Montes-Campos et. al. propose their *NeuralIL* potential for ionic liquids, which leverages second-generation Bessel descriptors in combination with a MLP. The authors parametrize the descriptors for each chemical species individually and allow the NN to mix the features without restrictions, rather than the conventionally used predefined weights to combine the generated features.

Their model is trained on an IL dataset of 15 anode-cathode pairs of EAN ($C_2H_8N_2O_3$) and designed for the application in MD simulations. Efficient differentiation facilitates the rapid evaluation of the forces as the gradient of the energy, which the authors use to train their potential on the $3N_{atoms}$ forces instead of the scalar energy. [MCCB⁺21]

Many of the authors' considerations are taken in the name of optimization of the architecture, which is implemented in JAX as to be fully automatically differentiable and enable the use of just-in-time compilation. Since the implementation of this thesis builds upon the codebase of NeuralIL, central considerations and implementation details shared are discussed in Sec. 3.2. The NeuralIL architecture further serves as a baseline for comparison for both the accuracy and speed of the graph neural network on the IL dataset.

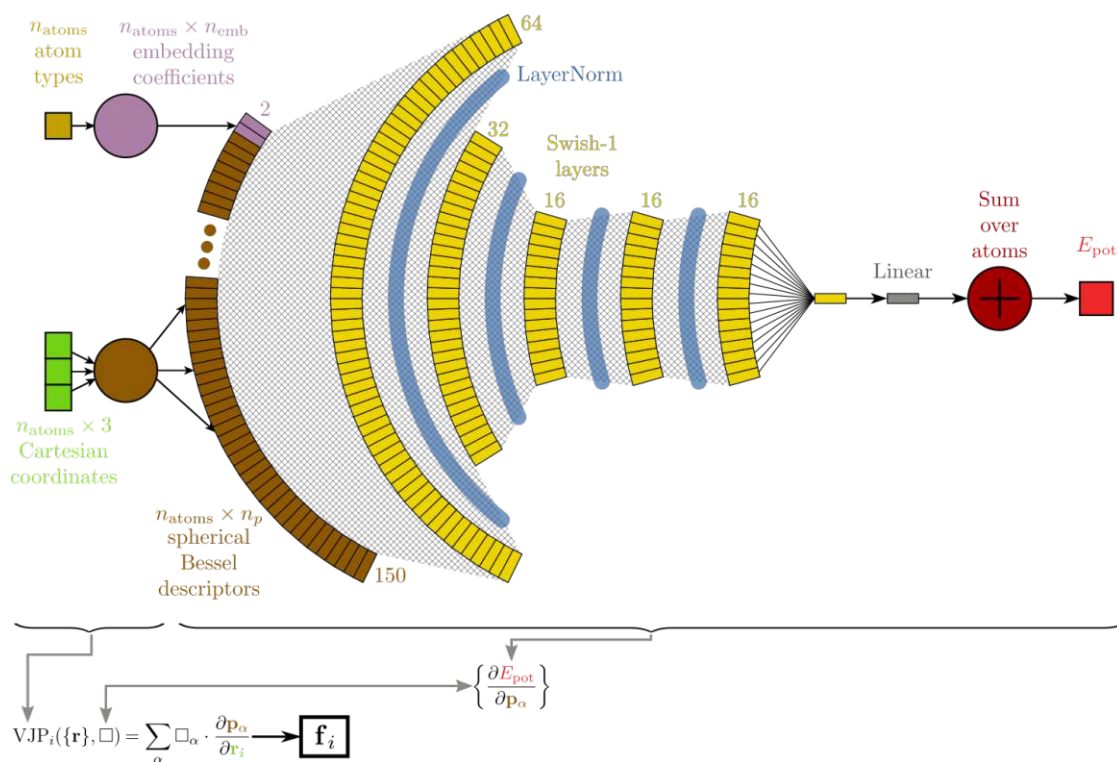


Figure 2.1: Visualization of the reference NeurallL architecture using spherical Bessel descriptors to generate features vectors, which are transformed by projection, nonlinearity and normalization layers into the prediction of the potential energy. Figure reproduced with permission from the authors. [MCCB⁺21]

2.3 Message passing neural networks on molecular graphs

2.3.1 Molecular graphs

Representing a molecule as a graph is a natural way to capture the chemical relations between the atoms and their neighbours. A molecular graph G has $|V|$ vertices (nodes), one for every atom, and $|E|$ (undirected) edges between nodes. In the chemical context, there are four possible types of C–C bonds and thus edge types between two carbon atoms: single, double, triple, or aromatic. Although chemists (or chemistry libraries) can determine the type of bond of a molecule in a configuration, without including a priori knowledge only the scalar inter-atomic distance or the 3-dimensional offset vector can be used as an edge feature.

Data from many domains of interest can be represented as a graph; however, the number of possible edges in a graph scales with $|V| \cdot (|V| - 1)$, which puts a limit on the size of the graphs that can reasonably be modeled computationally. A fully connected graph is thus

only feasible for small systems, and when considering a reasonably sized neighbourhood most nodes only have a few neighbours, while only a couple of nodes are highly connected outliers. Such a structure is often found in real-world graphs such as from social networks, where cliques, clusters and the small-world property define the structure of the network.

In contrast, for a molecular graph the connectivity is determined exclusively by the physical structure of the molecule, and most properties found in large networks do not hold for molecular graphs. Rather, the distribution of the number of neighbours within a small sphere is similar for most nodes, although some atoms of an isolated molecule might have fewer neighbours. Crucially, no node is expected to have a drastically higher number of neighbours than any other node. The edges further represent physically quantifiable relations, defined by the bond type or absolute distance separating the neighbours.

The graph of a molecular configuration defined by Cartesian coordinates is generated by calculating the Euclidean distance for each pair of atoms in the molecule. Only those pairs of nodes which are less than the cutoff distance apart are connected by an edge in the graph. Optionally, further edge types with bond information can be included. The radial cutoff distance is a hyperparameter to be chosen, which is usually selected in the 6 - 10 Å range.

Periodic boundary conditions

When considering a system of molecules, it is frequently simulated in a unit box under the assumption of *periodic boundary conditions* (PBCs), where an exact replica of the unit box repeats in the direction of all three linearly independent axes. Since the surfaces of the simulation box would otherwise necessitate special considerations and a couple of molecules in isolation would rarely be encountered in practice, the use of PBCs comes with practical advantages. It allows the system to be inspected as if it were part of a larger system rather than in isolation, although generalization of the simulation box dynamics to a larger system should be taken with a grain of salt, especially when considering non-crystalline systems such as ionic liquids.

The resulting graph is thus a *periodic graph*, which implies that during graph generation, a node on the perimeter of the box is impacted by the replica atoms located on the other side of the simulation box, and hence they need to be considered when calculating the edge distances for the neighbourhood structure. The use of PBCs also introduces a new constraint, namely that the model's output should be independent of the number of replicas considered, as they are by definition identical and infinite in all directions. For example, an atom might be impacted by the long-range interactions of distant replicas, possibly apart by more than one unit cell. However, predictions should not be influenced by whether the architecture explicitly models two, three or more replicas, as there are infinitely many in all direction anyways.

Type embeddings

Whereas the edge features capture the molecular structure, information about the atomic species is also available. Different atomic species contain a different number of protons Z in the nucleus, which impacts the charge exerted by the atom. Including the atom *type* as a node feature such as with an one-hot encoding is thus essential when more than one species is present in a molecule. A more expressive approach is the use of an *embedding* layer, which maps the discrete set of species to a high-dimensional continuous representation using learned parameters. These embeddings need to be initialized specifically to all species in a dataset (such as {H, C, N, O}), which makes inference on datasets with unseen atomic species impossible without retraining the model.

Beyond the atom species, in principle any set of features can be used as the initial node features, which the neural network can leverage to better approximate the target properties or energy. This usually includes features about the local atomic environment obtained by fingerprints or known properties about the atomic species such as formal charge and hybridization. Cheminformatic tools such as RDKit are frequently used to extract a range of features to augment the initial input features, however this does not necessarily improve the models' performance while increasing the computational cost of both training and inference.

Considering the significant computational cost of atomistic descriptors, careful considerations should be taken to balance the trade-off between speed and accuracy. Using fingerprints can easily be justified when they drastically improve the accuracy or generalization of the model, however if the expensive descriptor calculation can be avoided with a much faster approach, one might consider the trade-off worthwhile even if the model's accuracy decreases slightly.

2.3.2 Replacing fingerprints

A promising avenue is replacing atomistic fingerprints with an end-to-end deep learning approach and relying on the model to construct expressive latent features as part of the training process instead of using expensive hand-crafted functions. By enabling the network to uncover relevant relations through backpropagation, such approaches are more flexible and offer greater representational capacity than conventional descriptors. [KMB⁺16]

Duvenaud et. al. proposed neural graph fingerprints as a differentiable analog to circular fingerprints, which are used to generate fixed-size input vectors using hash-based functions. By making the entire architecture differentiable, they allow for gradient-based optimization for diverse tasks, although their simpler architecture uses only a few hidden layers. [DMI⁺15]

Kearnes et. al. introduce their convolutional *Weaver* architecture to replace fingerprints using atom and pair layers to construct molecule-level representation, which are then fed through fully connected layers and pyramidal task-specific readout *heads* to obtain predictions. The authors discuss many design choices prevalent in current approaches to achieve the desired invariances and use task-specific readout heads instead of training a separate model for each task. They evaluate their approach on solubility, drug efficacy and organic photovoltaic efficiency datasets against other ML models and similar approaches. The authors find that only minimal input features (atom type, bond type and graph distance) are necessary. Their simple model achieves the same performance as their full model augmented with many atom features, among them chirality, formal charge and hybridization. [KMB⁺16]

2.3.3 Message passing neural networks

Gilmer et. al. compare the situation in the chemical domain in 2017 to the advent of deep learning in visual computing, where convolutional neural networks (CNNs) displaced SVMs on top of hand-engineered features for a range of computer vision problems after their success in the ImageNet challenge. [KSH12, GSR⁺17] The authors generalize graph-based approaches in the quantum chemistry literature under their *message passing neural networks* (MPNN) framework to facilitate guided exploration and further refinement of graph neural networks as well as comparability between results.

Since the architecture explored in this work falls within the message passing framework, a detailed exploration of the MPNN components is justified to shed light both on guiding principles and essential components the presented architecture shares with current approaches, as well as highlighting in which aspects Graph Attention Networks (GATs) differ. GATs have seen little usage in the domain of chemistry, despite the success of attention and attention-based architectures (Transformers) in the domain of natural language processing [VSP⁺17]. In essence, the graph neural network applied in this work follows many design principles found in MPNNs but uses an attention-based update rule, which was first introduced for graphs by Veličković et. al. [VCC⁺17] and recently an improved version has been proposed. [BAY21]

Message passing

Gilmer et. al. [GSR⁺17] identify at least 8 notable examples of a model implementing a variation or combination of neural networks operating on an undirected graph G with node features x_v and edge features e_{vw} . The forward pass consists of a message passing phase and a readout phase, similar to spirit to the descriptor feature generation and dense readout layers found in classical approaches.

Through the message function M_t , each node receives messages from its neighbourhood $w \in N(v)$, which are combined by the update function U_t into the next latent node vector

h_v^{t+1} . To achieve sufficient expressive power with such a network, the node features are projected to a high-dimensional latent space before aggregation at each time step.

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \quad (2.6)$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \quad (2.7)$$

This message passing is repeated T times, after which the latent vectors for each node h_v^T are transformed into a feature vector for the whole graph by a readout function R .

$$\hat{y} = R(\{h_v^T | v \in G\}) \quad (2.8)$$

The message function M_t , update function U_t and readout function R are all learned differentiable functions parameterized by neural networks. A detailed discussion of the different choices for those functions is given by the authors of the MPNN framework [GSR⁺17], which make up most notable variants proposed in the literature. Among them are the Weaver architecture introduced earlier [KMB⁺16], Graph Convolutional Networks [WK16], Graph Attention Networks [VCC⁺17], Gated Graph Neural Networks [LTBZ15] and Interaction Networks [BPL⁺16].

Beyond the learnable functions, the models differ in their choice of atom input features and edge representation, such as binning of inter-atomic distances or usage of directed edges, as well as selected hyperparameters and training procedures. Gilmer et. al. further explore promising combinations of previously proposed ideas and highlight difficulties with current approaches, some of which are addressed by this attention-based implementation.

Message function M_t

For a given node x_v , a *message* is sent from every node in its neighbourhood $w \in N(v)$ according to the graph connectivity structure. For every node connected to x_v by an edge, the latent features h_w^t from the previous step are transformed through the message function M_t and subsequently combined by the update function U_t into h_v^{t+1} .

The message function has seen the most variety of approaches, where every aforementioned implementation has come up with their own combination in their choices of nonlinearities, mixing strategies for node and edge vectors (such as concatenation), joint or separate embeddings for sender and receiver nodes, ResNet-inspired skip connections [HZRS16] and their order of application.

All of them are *convolutional* in that they apply the same operation locally everywhere, and combine information in a global pooling (readout) step. [DMI⁺15] By sharing weights

between every atom at each individual layer (or even between all layers), the number of trainable parameters is significantly smaller, but more importantly this allows the models to scale well because the number of parameters in the model is not tied to the size of the graph. [DRA21] Furthermore, local convolutional operations ensure invariant processing of the graph, since a permutation of atoms only changes the neighbours' indexing but not its actual neighbours, and thus the overall computation remains invariant. [BBCV21]

Update function U_t

Since the number of neighbours is generally different for every node, the update function needs to *reduce* all incoming messages to obtain h_v^{t+1} of static dimensionality. Addition or averaging of all projected neighbours' latent vectors is an obvious approach to reduce the incoming messages, which can optionally be concatenated with the receiver's state h_v^t and projected using a MLP to obtain the next latent node states.

A more elaborate approach first leveraged by Gated Graph Neural Networks [LTBZ15] is to use a *gated recurrent unit* (GRU) [CVMBB14], an encoder-decoder architecture for sequences which employs two gates to retain its context while incorporating the incoming messages at every message passing step.

Readout function R

The readout function R must be invariant to permutations of the input, which is usually achieved by a permutation-invariant function such as the sum or average. One of the simplest choices is thus a summation over all scalar atomic contributions from each atom, based on the assumption that the total energy can be decomposed into additive atomic contributions (Sec. 2.2.3).

Another choice for the readout function is the *set2set* approach, which first projects the set of final latent nodes and then applies multiple update steps using an attention-based mechanisms and an LSTM (long short-term memory) model to process the retrieved queries. It produces a graph-level embedding, which is invariant to the order of the inputs and should have more expressive power than simply summing the final node states. [VBK15, GSR⁺17]

Edge features

The graph G can also have features associated to each edge e_{vw} which captures the relation between a pair of nodes. For molecular graphs, this is usually the type of bond or the inter-atomic distance. For most models discussed, the adjacency matrix entries are edge vectors of one-hot encoded bond types as well as binned inter-atomic distances. [GSR⁺17] Alternatively, the pairwise distances can be encoded using radial basis functions (RBF) or similar approaches.

Global features

A further augmentation is the use of a master node capturing global information of the graph, which is connected to every other node with a special edge type. Alternatively, all non-existent edges can instead be replaced by a special virtual edge type. Either modification allows information to propagate throughout the graph much faster than conventionally possible, as the receptive field is otherwise limited by the number of message passing steps (or depth) of the model. Gilmer et. al. highlight the importance of allowing long range interactions between nodes in the graph with either the master node or the set2set output. [GSR⁺17]

2.3.4 Graph Attention Networks

Attention in natural language processing

Attention and attention-based architectures (Transformers) have overtaken most competing approaches in the field on natural language processing (NLP) since their introduction in 2017 [VSP⁺17], and have more recently been adapted to other fields such as computer vision using Vision Transformers [DBK⁺20]. NLP is in essence a sequence modeling task, where sentences of varying length need to be processed. A broad literature of approaches to handle sequences exist, among them recurrent neural networks (RNNs), LSTMs, GRUs and many encoder-decoder architectures. [SVL14]

Most approaches before attention however share the trait that they *auto-regressively* encode the sequence one token (word) at a time while generating a context vector representative of the entire sequence, which is subsequently unrolled one-by-one by the decoder to generate the output tokens. [SVL14] While this works well for short sequences, interactions between distant tokens are quickly lost in longer sentences. More importantly however, the computational complexity scales directly with the sequence length, which severely limits the maximum sequence length and prevents processing a sentence in parallel since the previous tokens' state is required.

Transformers address these issues by computing pair-wise attention weights which capture the relative importance between every pair of tokens in the sentence. This results in $\mathcal{O}(N^2)$ complexity for a sequence of up to N tokens, which still limits the maximum sequence length but is far superior to the scaling behaviour of other approaches. Since the sequence no longer needs to be unrolled but can be processed in its entirety, the computation of the attention weights and their gradients can be performed in parallel. This allows for much more efficient usage of compute accelerators such as GPUs and TPUs, which enables transformers to train on vastly larger datasets.

Attention on graphs

Graph Attention Networks were proposed by Veličković et. al., [VCC⁺17] which apply an attention-based update rule to graph-structured data. However, unlike the Transformer's

attention mechanism which computes the weights as the dot product of query and key vectors, GAT uses a linear attention variant introduced earlier by Bahdanau et. al. [BCB14]

Both methods compute attention weights α_{ij} for each node pair according to the relative importance of node j 's features to node i , which is referred to as self-attention. In NLP tasks, self-attention is computed for every pair of tokens, however in the graph context additional structural information can be leveraged. Instead, *masked* self-attention is computed only for nodes in the neighbourhood $j \in N(i)$. [VCC⁺17]

Beyond injecting the structural information of the graph, this drastically reduces the number of pairs that need to be considered from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|E|)$ for a molecule or system of $|V|$ nodes and $|E|$ edges in the generated graph. For small graphs this difference is insignificant, however this allows GATs to scale efficiently to larger systems.

Similar to the Transformer approach, which replaced RNNs and exclusively used attention [VSP⁺17], the GAT architecture as originally proposed is constructed exclusively with multiple GAT layers and a readout function. The approach however is equally applicable to different attention variants and augmentations inspired by MPNNs and previously proposed ideas.

The GAT layer

A GAT layer takes as input the latent feature vectors $\mathbf{h} = \{h_1, h_2, \dots, h_N\}$, $h_i \in R^F$ for each node in the graph and outputs new node feature vectors of potentially different dimensionality $R^{F'}$.

The node vectors \mathbf{h} are linearly projected by a shared learnable weight matrix $W \in R^{F' \times F}$ and transformed by a nonlinearity σ such as the ReLU function to obtain sufficient expressive power. The attention mechanism $a : R^{F'} \times R^{F'} \rightarrow R$ then computes attention coefficients e_{ij} for each pair of connected nodes. $h_i || h_j$ denotes concatenation of feature vectors.

$$e(h_i, h_j) = a^T \sigma(W[h_i || h_j]) \quad (2.9)$$

The coefficients e_{ij} are normalized across all choices of j using the softmax function to obtain α_{ij} , the normalized attention weights.

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \quad (2.10)$$

The output node vectors are then obtained by computing a weighted sum of the neighbours' projected feature vectors to obtain \mathbf{h}^{t+1} , after optionally applying another nonlinearity σ or node update function.

$$h_i^{t+1} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W h_j^t\right) \quad (2.11)$$

Dynamic attention

Recently, an improved version for the GAT architecture has been published (GATv2), which highlights a significant limitation of the linear attention mechanism and proposes an improved variant. In particular, the authors show the original variant's attention weights are *static*, or unconditioned on the query node. They propose GATv2 which reorders the projection, nonlinearity and attention function, prove it instead computes *dynamic* attention and show the limited expressivity of static attention theoretically and on real-world datasets. [BAY21]

The primary change proposed in GATv2 (which is already reflected in Eq. 2.12) is to apply the nonlinearity σ *between* a and W rather than thereafter, as the matrices could otherwise be collapsed into a single projection layer when applying them consecutively. They further consider dot-product attention as used in Transformers [VSP⁺17] but prove it to be strictly weaker than the proposed dynamic attention. [BAY21]

Computing dynamic attention thus allows for more efficient use of the message channel by allowing α_{ij} to depend on the latent state of both the sender (query or source node) and receiver (key or target node). This desirable property was also used in interaction networks [BPL⁺16] and is discussed in the MPNN framework as *pair messages* [GSR⁺17].

For a graph of $|V|$ nodes and $|E|$ edges, the complexity per GAT layer is $\mathcal{O}(|V|dd' + |E|d')$, where d and d' respectively denote the input and output dimensionality. The computational complexity of all 3 attention mechanisms as proposed is identical, although the original GAT variant could be improved by merging a and W into a single layer. [BAY21]

A further extension to the attention mechanism is to include the edge features (such as the inter-atomic distance) d_{ij} in the attention calculation by concatenation before applying a^T .

$$e(h_i, h_j, d_{ij}) = a^T\left(\sigma(W \cdot [h_i || h_j]) || d_{ij}\right) \quad (2.12)$$

Multi-head attention

For all proposed GAT variants, an extension to use *multi-head* attention has been explored, which splits the mechanism into a few distinct channels. This resembles in spirit multiple

color channels of CNNs and the *towers* approach discussed in MPNNs [GSR⁺17]. To keep computational complexity comparable to the single-headed variant, the overall feature dimensionality is usually split evenly among the 3 to 8 heads. Multiple heads enable a more stable learning process and allow for different attention heads to focus on different aspects of the problem. Since the computations for every head are fully independent, multi-headed self-attention too can be efficiently parallelized. [VCC⁺17]

2.3.5 State-of-the-art models

Architectural principles are shared by many state-of-the-art models in computational chemistry generally and specifically in the context of predicting the PES. Models which can be described by the MPNN framework still differ in their respective choice of domain, task and datasets. Between those, the architectures further diverge in their choices of predicting the PES or molecular properties, configurations in- or outside of their ground states and system of study, such as ionic liquids, organic molecules or crystals.

A distinction needs to be made between models designed for the prediction of the PES and efficient evaluation of the forces, and models which aim to directly predict molecular properties of interest instead. Machine learning approaches are well-suited for both tasks, however the overall architectures are usually designed with one of the task in mind and should therefore be regarded as two distinct classes of models. While it is possible that the latent representations might share similarities, a model designed and trained to predict molecular properties cannot be used to evaluate the atomic forces, and vice versa.

Although many current models are evaluated on standard datasets such as QM9, differences in computational resources, single- or multi-task models and evaluated variants make a comprehensive comparison difficult, especially considering not many attention-based MPNNs have been applied on chemistry tasks.

Brockschmidt [Bro20] extensively compares many state-of-the-art GNN models on standard graph benchmarks, among them GAT evaluated on QM9 [RDRVL14], and finds the

"differences between state of the art models are much smaller than reported in the literature and well-known simple baselines that are often not compared to perform better than recently proposed GNN variants". [Bro20]

Despite this, a closer inspection of the range of different approaches taken in the literature to tackle different tasks and datasets highlights both the ideas implemented with (or without) success and the architectural changes which might prove beneficial on a given task. A detailed exploration of the literature is thus essential in guiding the design of the attentional architecture and implementation thereof.

SchNet and displaced configurations

A focal point in this work are molecules and systems which are not in their ground states, such as during MD simulations, vibrations or dissociation. Models that explicitly consider the forces and train on datasets containing displaced structures are thus of particular interest, of which the *SchNet* architecture [SKSF⁺17] is the most notable and relevant work in the literature.

Schütt et. al. propose *continuous-filter convolution* layers as a replacement for discrete filters and introduce the SchNet architecture to model quantum interactions. They evaluate their approach on QM9, MD17 and their newly introduced ISO-17 dataset of MD trajectories with chemical and structural changes and achieve state-of-the-art results by training on the energy and forces.

The central block uses filter-generating networks to obtain filters based on the RBF-expanded inter-atomic distance, which in combination with atom-wise dense layers, skip connections and nonlinearities constitutes one interaction blocks. An embedding layer followed by three such interaction blocks builds up representations, which are subsequently read out by a standard pyramidal head to obtain atomic contributions. Since the model only uses the inter-atomic distance, it produces invariant energy and equivariant force predictions.

The authors formulate the loss function as a combination of the energy and normalized forces mean squared error (MSE), and find that a combination of both drastically improves the sample efficiency of the model. When comparing their models on the MD17 dataset, they observe the GDML model [CTS⁺17] achieves accurate energy predictions when trained only on the forces, however the model cannot produce accurate force fields when trained exclusively on the energy. Although they do not evaluate SchNet when trained exclusively on the forces, the combined loss function drastically improves the accuracy and reduces the number of samples required to achieve state-of-the-art results and enables better generalization across the space of chemical compounds. [SKSF⁺17]

GNNs in materials chemistry and drug discovery

Fung et. al. [FZJS21] perform a review of GNNs in materials chemistry and propose a workflow and testing platform to evaluate new models. They consider many modern architectures and perform hyperparameter optimization on representative datasets of computational materials for bulk, surface, 2D, cluster and metal-organic frameworks configurations. Their findings confirm that GNNs thrive when ample training data is available, while descriptor-based methods or transfer learning should be considered when using less than $10^3 - 10^4$ samples. They further find that most state-of-the-art models achieve very similar results once properly optimized, mirroring the sentiment from Brockschmidt (2.3.5).

St. John et. al. [SJKP⁺19] investigate the impact of a model’s accuracy when operating on optimized 3D geometry structures compared to 2D geometries without Cartesian coordinates. Since it is not trivial to obtain optimized (relaxed) configurations, the authors use as input only SMILES strings, which algorithmically encode the structure of molecules using short ASCII strings. [Wei88] They train an off-the-shelf MPNN on inputs generated from SMILES strings and compare the trained model to one whose inputs are instead based on 3D geometries. Their 2D model achieves comparable results to models trained on the 3D input, even though some molecules share identical SMILES strings with different 3D geometries. They also find that using a poorly optimized 3D geometry (such as from an inexpensive force field) is worse than excluding the 3D coordinates entirely. Finally, the authors explore transfer learning approaches for problems where only small datasets are available.

Xiong et. al. introduce *attentive fingerprints* in the context of drug discovery, following the MPNN framework and using an attentional message function. Interestingly, their chosen attention function mirrors none of the introduced GAT versions exactly, since it lacks the a^T vector multiplication present in GAT (before) and GATv2 (after) the nonlinearity. They continuously update a global context vector to allow for global propagation of information irrespective of depth as well as a GRU as their update and readout function. Although they discuss the capability of GNNs to extract relevant information without introducing a priori knowledge as initial features, they use the full range of available atom and bond features extracted as one-hot encodings or embeddings. They leverage a virtual fully connected master node to bridge the gap between the input, atom and molecule layers. They evaluate their approach on QM9 and a variety of drug discovery tasks, achieve state-of-the-art results and provide visualizations of the atom vector similarity for Iprodione in the name of interpretability. [XWL⁺19]

Maziarka et. al. propose the *Molecule Attention Transformer* which adapts the Transformer architecture and aims to offer great out-of-the-box performance on a diverse set of molecular property prediction tasks but do not consider the calculation of the potential energy and atomic forces. Their specialized attention mechanism for molecules uses the inter-atomic distance and adjacency matrix in combination with dot product attention as the update rule in every layer, alongside a feed-forward block like conventional in Transformers. Multiple such blocks are stacked between the embedding layer and final pooling and dense readout classification head. The authors use standard RDKit features as input as well as layer normalization and incorporate a *dummy atom* node, which turns out to be central to the architecture. The authors perform an extensive hyperparameter search and comparisons to baseline models and explore pretraining the architecture on a self-supervised atom masking task. Inspection of the attention heads reveals that heads in the initial layers learn simple and easily interpretable chemical patterns, while subsequent layers combine them into more complex features. [MDM⁺20]

For ionic liquids, the most related work in the literature comes from a group of students, who predict the melting point and viscosity of ionic liquids. Their architecture follows the MPNN framework, using pairs of SMILES strings corresponding to the anion and cation as inputs. They use a shared embedding but independent dense layer to allow for cross-interactions between anion-cation properties, four message passing steps and the GRU update function. Two independent readout heads for melting point and viscosity are used, while the remaining MPNN architecture is trained jointly. [QRT20]

Geometric Deep Learning

Significant research efforts have recently focused on building equivariance directly into the model, rather than simply ensuring the model respects the required invariances. This line of research falls under the umbrella of *geometric deep learning* and uses explicitly designed architectural components which model group equivariances by building relevant inductive priors into the architecture.

Note that while many simpler architectures already achieve invariant energy and equivariant force predictions, allowing the model to use latent vectors (rather than scalars) should enable models to be more expressive, even when the output is invariant in both cases. This drastically limits the space of possible functions the model can fit to those that fulfil the relevant symmetries and thus helps the model generalize better using fewer samples. Bronstein et. al. discuss symmetries, representations and invariances on five domains, of which graphs are most widely considered, and propose a blueprint for building such geometric deep learning architectures. [BBCV21]

Among this research, Transformer variants have been proposed for groups such as the *SE(3)-Transformer* which produces equivariant predictions under continuous 3D rotations. Although the authors evaluate their model on QM9 alongside other standard benchmarks, no other chemical tasks are considered. [FWFW20] Similarly, *Tensor Field Networks* build their filters using spherical harmonics and thus obtain equivariant latent representations throughout the network, which the authors evaluate on a missing point prediction task. [TSK⁺18]

A closely related architecture from the chemical domain is *Cormorant*, whose authors build a *rotationally covariant* model for MD simulations and property predictions. Their work leverages Clebsch-Gordan nonlinearities, which are covariant to rotation and invariant to translation. Since each component of the architecture receives and outputs scalars or covariant vectors, it achieves overall rotational covariance. Translational invariance is obtained through the use of internal coordinates. The model is evaluated on QM9 and MD17 and achieves state-of-the-art performance on the MD17 task without training on the forces explicitly, which the authors suggest as one natural extension to their work. [AHK19]

2. LITERATURE REVIEW

Covariant compositional networks (CCN) explore a similar approach to achieve covariance with respect to permutation by defining covariant update and activation functions for the MPNN. Rotational invariance is however only given as a motivating example and not actually considered in their architecture, which is evaluated on QM9 and another dataset. [HTP⁺18]

Batzner et. al. propose *NequIP*, following a similar approach of designing equivariant convolutions in a graph neural network framework, which enables the model's internal representations to leverage equivariant vectors and higher-order tensors instead of invariant scalars. The authors place special emphasis on enabling MD simulations and design their architecture accordingly, which further allows them to train on the atomic forces instead of the energy. The authors achieve state-of-the-art results and highlight the model's excellent sample efficiency, which they attribute and verify to stem from their equivariant augmentations. [BMS⁺21]

2.4 Summary

A computational approach is necessary to approximate of the intractable Schrödinger equation. A balance between many desirable attributes needs to be achieved, where predicting the energy and atomic forces of a molecular system should be computationally cheap and achieve chemical accuracy. It should generalize well enough to sparsely sampled regions of the PES by learning to reproduce the complex interactions of the electronic structure. The method should further be able to accommodate a varying number of atoms or molecules and periodic boundary conditions without needing re-adjustments. Finally, the method should be able to work for diverse molecular structures and states of matter, in the ground state and outside thereof, and capture dynamic interactions such as molecular vibration and dissociation. Such a model can be applied either for MD simulations using the atomic forces or screening for molecules with desirable properties, which places the emphasis on different model attributes.

Current architectures approximate the potential energy as a sum of atomic contributions, which are predicted by neural networks. Conventional approaches leverage one of many atom-centered fingerprints (descriptors), which are computationally expensive but very descriptive and invariant to rotation and translation. These features are the input to a MLP with a regression head, which predicts each atom's contribution to the total energy.

Message passing neural networks (MPNNs) instead use an end-to-end differentiable deep learning architecture, which propagates information throughout an atom's neighbourhood using the graph connectivity structure. After an embedding layer for the atomic species, multiple message passing layers incrementally build expressive latent features, which are similarly transformed by a regression readout head into atomic contributions.

Modern approaches in the literature primarily differ in their choice of message function, which are either convolutional or attentional. Transformers and attention-based architectures quantify the relative importance of each node pair to generate weighted messages, whereas other models generate messages with convolutional filters. The update function reduces these messages by aggregating the variable number of messages into a new feature vector of fixed dimensionality. The MPNN framework [GSR⁺17] generalizes most of these architectures, which in combination with task-specific optimizations and equivariant augmentations make up most state-of-the-art approaches.

Methods

The **J**raph **A**ttention **N**e**T**work (JAT) architecture of this work incorporates many elements and design choices of previous works in the literature, which are discussed in the previous section. This chapter introduces the architectural components of the JAT model, which consists of multiple JAT layers performing message passing stacked in between the graph generation and embedding layer and the pyramidal readout head.

Visualizations of the message passing layer (Fig. 3.3), attention mechanism (Fig. 3.4) and graph generator (Fig. 3.2) further illustrate the central components of the overall architecture (Fig. 3.1). Section 3.2 focuses on implementation details of the JAT model in JAX and explores central considerations to achieve computational efficiency and inference time suitable for MD simulations.

3.1 JAT Architecture

Multiple viable options are available for many architecture components, making an exhaustive study of all combinations infeasible considering the combinatorial nature of components in ML models. Rather, the architecture evolved to its final state based on a combination of studying which approaches were applied with (or without) success for similar models and datasets in the literature, as well as guided by intuition and empirical experimentation on datasets of varying size and difficulty.

Generally speaking, the design space of such an architecture is continuous, implying that an exchange of one component for another, such as using a different activation function, will only slightly change the models' characteristics. Functionally similar if not identical components are thus expected to differ only marginally, and the exact impact of a choice between two components is not easy to determine definitively. In such instances, Occam's

3. METHODS

razor suggests favoring simplicity over complex methods if no clear advantage can be observed.

The same holds true for the implementation of the architecture, where implementations can be functionally equivalent, but harbor drastic differences in their computational complexity or numerical stability. Significant emphasis and effort has gone into making the implementation efficient, which further influences the design and implementation of the JAT model.

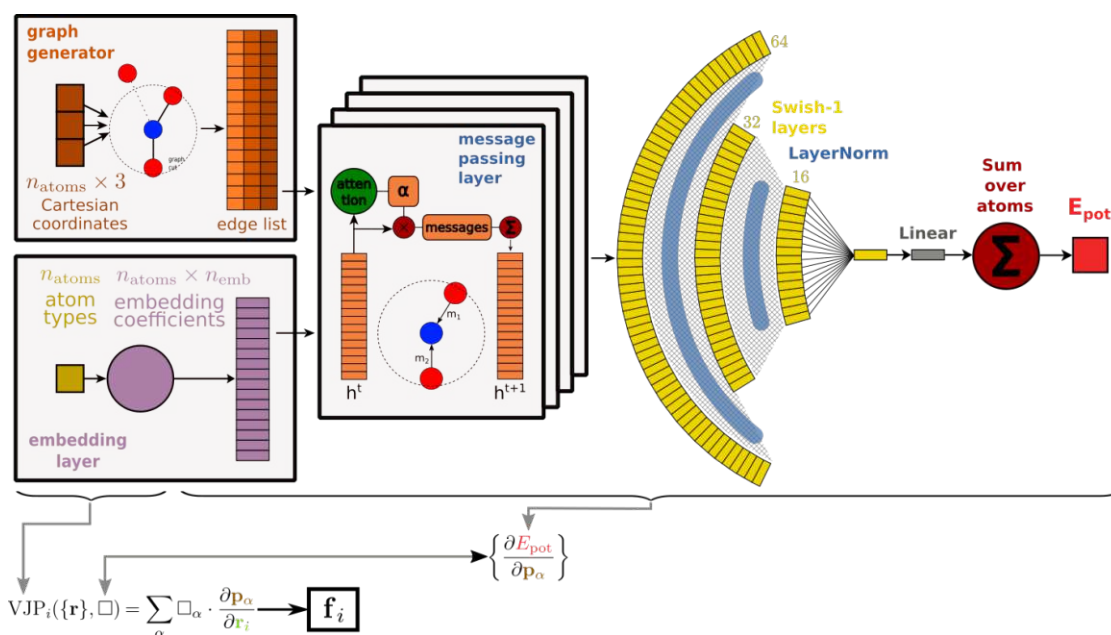


Figure 3.1: Visualization of the entire JAT architecture. Using the positions and species (types) as input, the graph generator (Fig. 3.2) generates a sparse edge list, with which T message passing steps (here 4) using an attentional update function (Fig. 3.4) are performed. The features h^0 at $t = 0$ are obtained from the type embedding layer, for $t = 1 - T$ the output features of the last message passing steps serve as input for the next JAT layer. (Fig. 3.3)

The readout head (here visualized truncated) transform the features h^T using projection, nonlinearity and normalization layers into the energy contribution for each individual atom. The JAT model's prediction for the potential energy E_{pot} is obtained as the sum over all atoms' contributions, while the forces \mathbf{f} are obtained as the model's derivative with respect to the atomic positions.

3.1.1 Graph generator

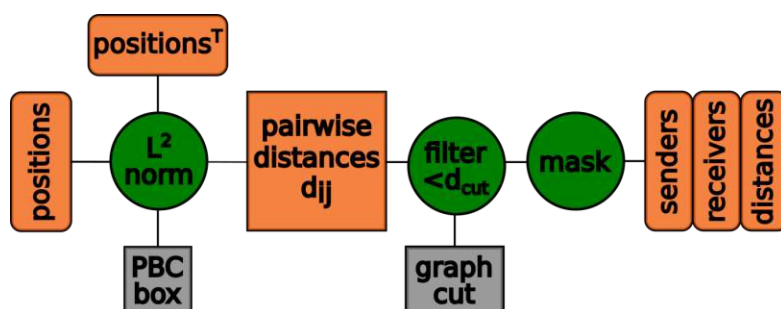


Figure 3.2: Visualization of the graph generator component of the JAT architecture. Using the Cartesian coordinates (positions) of all atoms, the pairwise distance matrix is calculated using the Euclidean L^2 norm under consideration of periodic boundary conditions. The distances are filtered using the graph cut parameter to only include pairs within close proximity, which are connected by an edge. This generated molecular graph is represented as an *edge list* in a sparse format using three arrays, respectively the triplets of sender, receiver and distance. Since the number of edges depends on the positions, the edge list is padded with masked edges up to a static maximum.

To perform message passing on a graph, the system as described by Cartesian coordinates first needs to be transformed into a molecular graph as discussed in Sec. 2.3.1. An essential aspect is the use of a *sparse* graph representation for the JAT model, which facilitates drastic computational gains by avoiding unnecessary calculations for irrelevant node pairs by focusing only on those edges present in the graph. Representing a graph in a sparse manner boils down to using an *edge list* to represent the graph adjacency matrix efficiently, such as the CSR (*compressed sparse row*) format. Three one-dimensional arrays respectively denote the *row index*, *column index* and *value* of every *non-zero* entry. In the generated molecular graph, the three arrays corresponds to the *sender*, *receiver* and *Euclidean distance* between the two nodes.

The inclusion of the pairwise Euclidean distance as an edge feature used by the attention mechanism is central to the model’s performance. Without including these internal coordinates in the architecture, the model lacks information about the molecules’ physical structure and thus cannot distinguish between isomers. Such a model does not learn beyond the first epoch and performs abysmally, comparable to using the randomly initialized model without training to perform inference.

The graph generator thus calculates the Euclidean distance for every node pair, and filters them using the radial cutoff hyperparameter to retain only those edges which are in the considered neighbourhood of a node. If applicable, periodic boundary conditions need to be taken into account, which are calculated as the minimum distance between every node and any of its replicas instead.

This approach implies undirected edges in the graph, however in practice the separation between sender and receiver in message passing requires every edge to appear twice, where each node is once the sender and once the receiver. The approach could be reformulated to treat each edge only once, however this comes at a loss of expressiveness since then the interactions necessarily need to be identical in either direction, which requires using the same projection matrices $W_s = W_r$.

Considering the significant impact of the cutoff parameter on the necessary computation and accuracy of the model, some discussion is justified on what constitutes an appropriate choice. For $d_{\text{cut}} \rightarrow \infty$, the generated graph is fully connected, hence an attention weight α_{ij} is calculated for every edge, which results in the general case of a dense MPNN calculation for every entry in the adjacency matrix (excluding self-edges). For datasets with configurations of individual molecules with a couple of heavy atoms, a graph cutoff of 3 Å suffices for nearly fully connected molecular graphs. When considering systems with multiple molecules, the maximum diameter is drastically larger and a 5 Å cutoff still results in a sparse molecular graph. Hence, the gains in efficiency become especially significant for larger systems with very favorable scaling, while small molecules barely benefit from the sparse implementation.

Directly related is the *receptive field* of the JAT model, which increases proportionally with increases of the graph cutoff and additional MPNN layers (*depth*). Similar to how deeper layers in CNNs construct increasingly complex features with every layer, the region of pixels influencing a given neurons' output (its receptive field) increases with each consecutive layer. A large receptive field is advantageous but does not come for free, since increasing either the cutoff or depth increases the computational cost.

Edge mask

The sparse graph representation allows calculating *masked self-attention* for all relevant edges instead of every possible edge but introduces a new difficulty. Since the number of neighbours in the graph depends on the atomic positions, the number of edges is expected to be different for every configuration. However, just-in-time (JIT) compilation necessitates the matrices to be of static shape, independent of the input data (see also Sec. 3.2.2). Therefore, an *edge mask* is necessary, which pads the variable number of edges up to a static maximum. This necessarily introduces an overhead of wasted calculations since any padded edge must not contribute to the result. In spite of this overhead, masked self-attention is vastly more efficient by reducing the number of necessary edge calculations to $n_{\text{atoms}} \times \max(n_{\text{neighbours}})$.

Node mask

Since the number of atoms in a molecule varies in some datasets, the graph generation needs to treat such *empty* atoms accordingly. Atoms are padded up to the maximum number of atoms in the dataset, and the padded atom's species is denoted with -1 instead of the atomic number, such that the embedding and readout layers can treat the padded atoms accordingly.

Embedding

An embedding layer is used to transform the discrete atomic species of each atom to a continuous feature vector h_i^0 (*embedding*), which serves as input to the first message passing layer. This is much more expressive than the simplest alternative of one-hot encoding the atomic species. The embedding matrix is learned through backpropagation as part of the end-to-end training procedure, which allows the architecture to independently discover relevant features for each species. Choosing an insufficient dimensionality for the embedding vector is detrimental to the performance of the model.

3.1.2 JAT layer

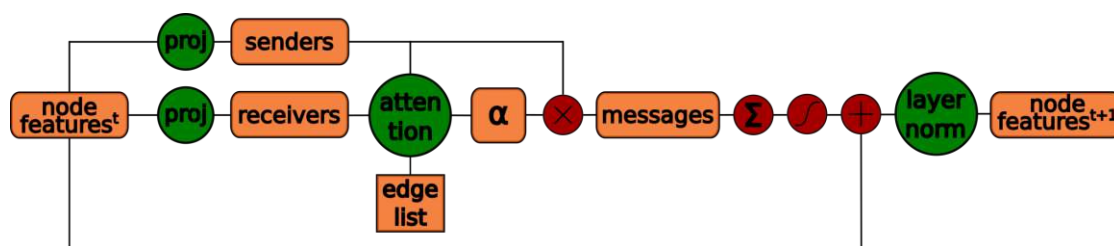


Figure 3.3: Visualization of a single JAT layer, which performs a single round of message passing to update the node feature vectors. The features \mathbf{h}^t at step t are projected into senders and receivers, and for every pair in the edge list a weight α is calculated using the attention mechanism. The messages are calculated as the element-wise multiplication of the sender features and attention weights α . These messages are aggregated to each receiver using a segment sum and transformed with a nonlinearity, skip connection and layer normalization to obtain the updated node features \mathbf{h}^{t+1} . These are fed into the next JAT layer to repeat the message passing procedure for multiple rounds.

Every JAT layer performs one round of message passing, which consists of updating the feature vectors using two linear projections, calculating the attention weights α_{ij} for every relevant edge, performing a segment sum over the weighted messages from a node's neighbourhood and finally applying a node update function to combine and transform the aggregated messages into the new feature vectors.

Node projection

Each layer takes as input the previous hidden state $\mathbf{h} = \{h_1, h_2, \dots, h_N\}$, $h_i \in R^F$ and outputs a feature vector of potentially different dimensionality $h'_i \in R^{F'}$. The linear

projection is parametrized by a shared weight matrix $W \in R^{F' \times F}$, which is applied to every nodes' latent vector. This learnable transformation is important to give the model sufficient expressive power, since the attention weights are not used to transform the features themselves but are applied during the weighted aggregation of the neighbourhood features into each node.

This transformation can also be performed using two different sets of parameters W_s and W_r , respectively for the sender and receiver nodes, allowing the model to learn separate features relevant for both the calculation of attention weights and the node features to be aggregated. While separate projection weights enable representing more complex functional relationships, this naturally increases both the inference time and trainable parameters, and thus the increase in expressive power must be weighted against the computational trade-offs.

The dot-product attention function used in Transformers uses three projections, referred to as the *query*, *key* and *value* vectors. The dot product of the query and key vector measures the similarity between two nodes, while the value vector is then *retrieved* to be used. [VSP⁺17]

3.1.3 Linear self-attention

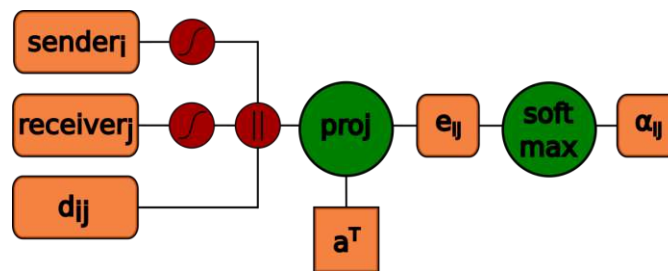


Figure 3.4: Visualization of the attention mechanism of the JAT architecture. For every edge_{*ij*} in the edge list, the features of sender_{*i*}, receiver_{*j*} and *d_{ij}* are *lifted* and with a projection parametrized by a^T transformed into e_{ij} . These weights are normalized over all received messages with a segment softmax function to obtain α_{ij} .

Then, *masked self-attention* is computed for every node pair using the *linear attention* function discussed in Sec. 2.3.4 to obtain the attention weights e_{ij} (Eq. 2.9), which are normalized using the softmax function to obtain α_{ij} (Eq. 2.10). Since the attention weights are only used for the neighbourhood aggregation, calculating α_{ij} for every pair of nodes in the graph would be highly inefficient. Instead, the attention function is calculated based on a sparse graph data structure (the edge list), which stores triplets of the sender index, receiver index and their edge features.

Naively, one could project the concatenated sender and receiver nodes with a corresponding weight matrix $W'[h_i||h_j]$, $W' \in R^{2F' \times 2F}$ for every node pair in the edge list. This however would again be highly inefficient, since the same weight matrix is applied many times unnecessarily. Instead, the projections are split into sender and receiver $W_s h_j$ and $W_r h_i$, possibly using the same weights. [VJ20] Thus, each projection is applied only once per node, and subsequently the relevant sender and receiver vectors are retrieved (*lifted*) for the attention function based on the edge list.

$$e(h_i, h_j) = a^T \sigma(W'[h_i||h_j]) = a^T \sigma(W_r h_i + W_s h_j) \quad (3.1)$$

Nonlinearity and order of operation

With the improved GATv2 proposed by [BAY21] to compute *dynamic attention*, the nonlinearity σ is applied after the node projection W but before instead of after the multiplication with a^T . This ensures α_{ij} is conditioned on both the sender and receiver node state and makes a drastic difference in expressive power of the architecture. This stands in somewhat of a contrast with a continuous design space assumption introduced earlier, where a small change in the order of operation should only influence the results marginally.

Interestingly, the choice of nonlinearity has a rather small impact. GAT applied the LeakyReLU activation, which mirrors the ReLU function but further has a small negative slope defined by a parameter to allow for the gradients to *leak* through. The GATv2 authors retain the LeakyReLU function for consistency but emphasize their findings are independent of the choice of nonlinearity.

The activation used for the JAT model is the smooth Swish-1 function, which was discovered using automated ML search as a superior alternative for deeper models. [RZL17] The parameter β is chosen as 1, where different choices yield other familiar functions, such as approaching ReLU for $\beta \rightarrow \infty$. When predicting the atomic forces, the gradient of the model is effectively being evaluated, for which Swish is advantageous since it is continuously differentiable everywhere.

$$s_\beta(x) = x \cdot \text{sigmoid}(\beta x) = \frac{x}{1 + e^{-\beta x}} \quad (3.2)$$

Multi-headed attention

Using multiple attention heads effectively means passing multiple independent messages in parallel. This implies the use of separate sets of weights for each attention head, defined by the linear projection and attention matrices W_s , W_r and a^T . The messages are aggregated and processed independently per head, which lends itself to parallel processing and thus does not drastically decrease the speed of the model. To facilitate comparability, the full dimensionality of a single headed implementation is usually split evenly among the heads for the multi-headed context. When using multiple attention

heads, a transformed edge feature such as the second power of the distance can instead be appended for each head.

Layer normalization

The use of a normalization scheme contributes to a more rapid and stable learning process and helps to avoid vanishing and exploding gradients, which makes it an essential ingredient to train deeper neural networks. A common choice is batch normalization, which centers and scales intermediate activations using the mean and variance of the batch. A superior alternative used in the JAT model is layer normalization, which instead normalizes each samples' activations using their own mean and variance. [BKH16] This eliminates the need for keeping track of batch metrics during training, which are otherwise required for inference on the validation and test sets. This not only reduces the overall complexity, but further yields well-defined behaviour when taking derivatives.

Skip connections

Inspired by ResNet [HZRS16], skip connections are used to carry forward the nodes' previous state explicitly, rather than forcing the model to learn the identity function in addition to parametrizing the projection. The skip connection is an element-wise addition of the node state from the beginning of the layer, which *skips* the projection, message passing and aggregation and amounts to a simple element-wise addition of each nodes' *before* and *after* feature vector.

$$h_i^{t+1} = h_i^t + \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W_s h_j^t\right) \quad (3.3)$$

Since the shape of both tensors needs to be identical for the element-wise addition, a projection of h_i^t is necessary if the dimensionality of the node features is modified during the message passing step. This would be the case when a different output dimensionality is used for the projection matrix $W \in R^{F' \times F}$, $F' \neq F$ or when multi-headed attention is used. Since this introduces another matrix of learnable parameters, it is usually beneficial to avoid changing the feature dimensionality frequently between multiple message passing rounds. Matching the embedding layer dimensionality with the first message passing layer's output avoids the need for another projection.

In convolutional architectures a pyramidal layer structure is commonly found; however, an increase in the dimensionality is usually accompanied by a corresponding decrease in spatial resolution to ensure the required computation per layer remains approximately constant. A similar pooling technique called *coarse-graining* can be applied for very large molecular systems to reduce the number of nodes to fewer *beads* in the network. [WOW⁺19] Since the number of atoms is small in this work, no pooling (aside from the readout layer) is applied and all JAT layers share the same feature dimensionality.

Gated Recurrent Unit

As an alternative to skip connections, the GRU gate can be applied as an update function to combine the aggregated messages with the receiver’s latent vector. This replaces the skip connection since it explicitly incorporates the previous state, which is combined with the newly generated feature vector from the message passing step. To achieve this, the GRU uses a gate function and an activation function to retain the context, for which the default choices of *sigmoid* and *tanh* perform well. [CVMBB14]

3.1.4 JAT model

Pyramidal readout head

Akin to the pyramidal dense layers used to combine the fingerprint features into a regression target, a similar stack of dense layers, nonlinearities and layer normalization is used to transform the latent vector \mathbf{h}^T from the final message passing layer into the prediction of the atomic energy contributions. As long as this MLP is of sufficiently high dimensionality and depth, it should be able to accurately reproduce the regression target using the latent features generated from the message passing steps, rather than using those obtained from fingerprint methods.

Based on the assumption that combining the features obtained from the MPNN is no more difficult than those obtained by fingerprints, JAT uses a pyramidal head of similar design as employed by NeuralIL. A stack of dense layers of dimensionality 64 : 32 : 16 : 16 is interleaved with Swish-1 nonlinearities and layer normalization. Since the output should be scalar (see Sec. 2.2.3), a dense layer with a single output is applied before the sum over all atomic contributions to account for the characteristic range and a possible offset of the potential energy. [MCCB⁺21] Using the additive ansatz, the sum over each atoms’ rescaled atomic contribution while considering masked atoms is taken to obtain the prediction of the total potential energy.

Atomic forces

The atomic forces are obtained from the gradient of the potential energy with respect to the atomic positions (Eq. 2.3). Since both the architecture and the JAX framework are designed for efficient backpropagation, calculation of the atomic forces benefits equally and can be obtained at little additional cost. Compared to only calculating the potential energy, additionally evaluating the forces results only in a linear increase in computational cost.

To obtain stable MD simulations, the calculated forces must be continuous, which requires the JAT model to be differentiable with respect to both the positions and parameters. This requirement prevents the use of an ordinary loss and activation function, and necessitates replacing them with continuously differentiable (*smooth*) alternatives. [GGG19]

Loss function

The loss function used for training is the log-cosh loss of the atomic forces, which can be viewed as a smooth approximation of the mean squared error (MSE). The hyperparameter α shifts the relative weight between the mean absolute error (MAE) and MSE by determining the transition between the linear and quadratic regimes. If the prediction error is smaller than the characteristic scale argument $\alpha = 0.1 \text{ eV}/\text{\AA}$, the contribution to the loss approximates the MSE, while for errors larger than α the loss contribution scales linearly. [MCCB⁺21]

Increasing the α parameter to 1 or 10 eV/ \AA decreases the loss contribution of poorly predicted molecules, which reduces the impact of outliers during training but allows the model to largely ignore very difficult molecules. Such a behaviour was observed, indicated by a drastic difference between the MAE and root mean squared error (RMSE). $\langle \cdot \rangle$ denotes an average over the configurations in a training batch in Eq. 3.4 and 3.5.

$$\mathcal{L} = \left\langle \frac{\alpha}{3n_{atoms}} \sum_{i=1}^{n_{atoms}} \sum_{C \in \{x,y,z\}} \log \left[\cosh \left(\frac{\hat{f}_i^{(C)} - f_i^{(C)}}{\alpha} \right) \right] \right\rangle \quad (3.4)$$

For datasets of relaxed molecules, all forces are zero and thus they cannot be used in the loss function. For datasets without forces, the difference between the predicted and reference potential energy is used instead, for which only the scalar energy instead of $3n_{atoms}$ force contributions can be used.

$$\mathcal{L}_{E_{\text{pot}}} = \left\langle \log \left[\cosh \left(\frac{\hat{E} - E}{\alpha} \right) \right] \right\rangle \quad (3.5)$$

However, training on the scalar energy alone is a poor choice in comparison to training with the $3N_{atoms}$ forces, which are much more expressive than the energy. Training models with the forces is thus easier and significantly more accurate as long as reference forces have also been calculated and are available in the dataset. Although the potential energy is not considered explicitly in the forces loss function, models trained exclusively on the atomic forces can also reproduce the potential energy. The GDML model achieves this by integrating over the directly predicted forces to obtain the energy [CTS⁺17], while approaches using the additive ansatz can reproduce the energy up to an additive constant since the origin of the energies is arbitrary.

Alternatively, the loss function can be formulated to use both the energy as well as the force prediction errors, using a hyperparameter ρ to determine the relative importance of the energy and average force error, such as used by SchNet or DimeNet. [SKSF⁺17, GGG19]

To avoid numerical stability issues due to the hyperbolic cosine, the contribution to the loss is calculated with the equivalent but more stable expression based on the JAX implementation of $\text{SoftPlus}x = \log(1 + e^x)$. [MCCB⁺21]

$$\frac{\log [\cosh (\alpha x)]}{\alpha} = \frac{\text{SoftPlus}(2\alpha x) - \log 2}{\alpha} - x \quad (3.6)$$

3.2 JAT Implementation

The central motivation of replacing fingerprints is to avoid their computational bottleneck and achieve an accurate yet computationally efficient model with good complexity scaling to enable MD simulations of larger systems. Some central aspects to achieve this goal have already been discussed briefly, and although the entire architecture is designed in this light, drastic gains in efficiency can be achieved through the use of JAX [BFH⁺20] with just-in-time (JIT) compilation, automatic differentiation and a sparse implementation. This section details all those aspects crucial to the efficiency of training and inference on GPUs and the decisions taken to achieve such an implementation of the JAT architecture. Two implementations of the GAT architecture were instrumental towards achieving a first working implementation of the JAT architecture, specifically the GAT model of the *jraph* library and the annotated GATv2 implementation by LabML. [GKB⁺, VJ20]

3.2.1 GPUs and parallel processing

Parallelization

Parallel processing of multiple configurations in batches drastically reduces the total compute time required per evaluation through the use of *SIMD* (single instruction multiple data) operations. Furthermore, this is important for gradient descent, which operates on the aggregate gradients of the batch to update the model parameters. While this is not possible when using the model for inference in molecular dynamics, in other cases such as screening and especially during training it is important to parallelize the processing of multiple configurations. However, MD simulation engines enable splitting large simulation boxes into independent subdomains to be evaluated independently, which necessitates locality but is crucial to enable efficient simulations of very large systems.

Parallelization is further facilitated by the *vmap* function in JAX, which maps a function defined on a single instance across a batch of such instances. Since this is subsequently compiled and optimized to machine code, this comes without any loss of performance. It is thus easy to define a function to operate on a single configuration and then *vmap* the function across the batch, which abstracts away much of the complexity. Furthermore, the same methods are usable for training and inference, further simplifying the implementation to make it easier to understand and debug.

While these gains in computational efficiency apply to CPU processing, they become especially relevant when leveraging compute accelerators such as GPUs and TPUs, which are designed and heavily optimized for parallel processing. TPUs are designed exclusively for training very large machine learning models and are only available for industry applications. However, a modern GPU is readily available and can drastically accelerate the design process, training and inference of reasonably-sized modern ML architectures.

CUDA

Nvidia's CUDA API provides implementations for many operations frequently used in ML, such as the *matrix-multiply and addition* or *convolution* which constitute the building blocks of modern ML architectures. These CUDA operations are optimized at the hardware level and translated from python code by JAX via XLA into CUDA calls and thus do not require separate implementations. This does however introduce additional requirements and difficulties to set up the computational environment, as a CUDA-enabled environment necessitates a range of additional dependencies. It can thus be cumbersome to achieve interoperability and harmony between all components, ranging from GPU-specific CUDA version requirements to incompatibilities between stable versions and new features introduced in recently published JAX & jaxlib libraries.

A GPU-accelerated environment however massively reduces the time required to implement, debug and iterate the model architecture by enabling rapid training, where issues during the early phase of training become apparent within minutes. Using small subsets of the training data further accelerates this prototyping phase to ensure the model trains and learns properly during the critical initial epochs. Large training sets are then used to observe the models' learning trajectory during training as it approaches production quality.

3.2.2 Just-in-time compilation

To recoup the downsides of an interpreted programming languages such as python, just-in-time compilation of the important function calls, such as of the model's forward prediction and gradient evaluation calls, can be *compiled* at runtime (*just-in-time*). This reduces many of the overheads, such as from function *overloading* which allows for the usage of a function despite slightly different input arguments. The compiler can fuse operations and avoid expensive memory allocation or read and write operations, which results in speedups of three or more orders of magnitude compared to the evaluation of a regular function.

A *tracer*, an object to determine the shapes of inputs as they are processed by the various functions triggered within the *jitted* function, is passed to evaluate the *stack trace* of the call without actually performing any calculations. This information is then used to compile the function to highly-optimized machine code, in particular for the case

of compute accelerators and GPUs which can best leverage CUDA operations for expensive calculations such as the matrix-multiplication.

Jit-compilation necessitates the matrices to be of *static* shape, which means the shape of the output and intermediary objects cannot change dependent on the input to the function. For example, the output of a function which returns a vector for every neighbour of a node is *dynamic*, while the scalar output of a function which returns the number of neighbours is a *static* integer irrespective of the input graph.

3.2.3 Atom and edge masks

This can be an issue in the case of molecules of different sizes, such as for the ANI-1 database of organic molecules made up of 2 - 29 atoms in total. On the other hand, in molecular dynamics the number and species of molecules in a system is predetermined throughout the entire simulation, where the static requirement does not pose an issue. Otherwise, a *mask* is required, where the tensor shape is chosen to accommodate the largest molecules in the dataset and a mask is applied for smaller ones. This introduces a new overhead which leads to a part of the computation being *wasted*, as masked atoms cannot affect the result of the calculation itself.

Furthermore, a mask is necessary for the calculation of the attention weights between all pairs of nodes connected by an edge in the sparse molecular graph. The number of edges however depends on relative positions of the atoms at a particular time step, which is not known in advance and is expected to change at every step of the MD simulation. Therefore, an edge mask is required to exclude messages from all masked edges from the aggregation, which are technically not present in the graph. The mask must accommodate a maximum number of possible edges, an upper bound to this is $n_{atoms} \times \max(n_{neighbours})$. This is still drastically smaller than the maximum number of possible edges for a fully connected graph (n_{atoms}^2), in particular for larger systems.

3.3 Hyperparameters & training

Beyond the parameters of the neural network architecture, a few hyperparameters control the training process of the network, which can have a drastic impact in the accuracy and transferability of the final model and its capability to generalize to novel samples. Hyperparameter tuning describes this difficult task of finding an optimal combination of all hyperparameters to achieve great generalization, while keeping the required amount of computational power to train the architecture within reasonable bounds.

Each parameter has an impact on different aspects of the training process, which exhibit nonlinear dependencies. For example, the number of samples processed in parallel during training (the *batch size*) is not only limited by the available hardware and model com-

plexity, considering the entire batch and model need to fit within (GPU) memory, but directly affects the number of update steps as well as the time required to complete one full pass of the training set (one *epoch*). Increasing the batch size requires fewer update steps in total, but each such accumulated gradient update contains more information and thus an update step with a larger learning rate (LR) can confidently be taken. Yet a large batch size can negatively impact the convergence of the model, such as when the model parameters oscillate around the minima with large steps. In contrast, by using many small update steps the model parameters are more likely to reach a minima; however, this also makes it easier for to get trapped within a local minima and never leave its basin of attraction.

Learning rate schedule

A LR schedule drastically accelerates the training and convergence behaviour of the architecture. Rather than continuously decaying the LR or dropping it drastically after a large number of epochs to then fine-tune the weights, a cyclic LR schedule is applied. In this *one-cycle* schedule introduced by Smith, the LR linearly increases for the first 45% of batches in the epoch, peaks and is similarly decreased back down for the following 45%. Hyperparameters define the LR *minimum* and *maximum*, as well as the *final* LR applied for updates in the remaining 10% of the training epoch. [Smi18] The LR is multiplied with the negative gradient at every step to update the model parameters using gradient descent, which is easily facilitated within JAX without needing to re-compile the model.

This LR schedule mirrors the one employed by NeuralIL, where the authors find it drastically decreases the number of epochs necessary to achieve chemical accuracy from approximately 3000 to 500 epochs. [MCCB⁺21] Such *super-convergence* may be attributed to instilling *momentum* in the learning process, whereby the parameters of the model can more easily leave local minima as the LR increases, which facilitates more rapid exploration of the loss landscape in search for optimal weights. [Smi18]

Overfitting

Overfitting is a common issue when training neural networks, whereby the model *memorizes* the training data, which leads to a continuous decrease in the training loss. However, this coincides with an increase in the validation and test error and comes at the cost of poor generalization of the model. The optimal model, which best generalizes to new data, is obtained from the delicate balance of under- and overfitting and does not necessarily score best in terms of training loss.

Early stopping is one technique to combat overfitting, whereby the training process is interrupted when the validation loss starts increasing again; however, ideally one wants to achieve natural convergence of the model as indicated by a horizontal test loss curve. [Smi18]. This is significantly influenced by all hyperparameters, including the LR as well as *weight decay* and *regularization*.

Regularization

Some form of regularization is present in almost all neural network architectures, ranging from simple L^2 -norm penalties on the network parameter magnitude to elaborate terms in the loss function. Dropout acts as another form of regularization by reducing the dependence on individual weights, however dropout is usually applied in a classification context. The frequently used ADAM [KB14] optimizer uses an estimate of the *momentum* and *RMSprop* to achieve quicker convergence, possibly in addition to using *weight decay* of the parameters. Finally, the batch size too act as a form of regularization, and larger batches generally allow for larger update steps and hence a larger LR without divergence.

Incremental improvements

In practice, the vast number of possible configurations of the model architecture as well as hyperparameters makes it infeasible to exhaustively explore multiple configurations, especially during the iterative process of designing and implementing the model architecture. Whether a change in implementation has a measurable effect is difficult to definitively determine without training multiple models, yet it is impossible to make any progress while evaluating every minor change. This however might lead to local minima in the design of the architecture, where incremental changes keep improving the model performance, yet ultimately a superior yet elusive implementation exists. While a hyperparameter search for the final architecture should yield optimal results for the given implementation, this does not imply the implementation is optimal.

3.4 Summary

The JAT architecture is built around multiple JAT layers, each of which performs one message passing step by aggregating messages from the graph neighbourhood, which are weighted with the linear self-attention mechanism. The graph generator first constructs an edge list from the molecules' Cartesian coordinates, while an embedding layer transforms the atomic species into the initial latent vector for every atom. This latent vector is incrementally refined with 4 - 7 message passing steps, which the pyramidal regression readout head uses to predict the energy contribution for each atom.

Sparsity is essential and leveraged throughout the architecture, from the sparse graph representation (edge list) to masked self-attention, to avoid the quadratic scaling of large fully connected graphs. Layer normalization, skip connections, a cyclic learning rate schedule, the smooth Swish-1 nonlinearity and the smooth log-cosh loss function contribute to efficient training of the deep learning architecture. The JAX implementation leverages parallelization, a GPU and just-in-time compilation to achieve an efficient model suitable for molecular dynamics simulations.

Results

This chapter presents the results of the JAT architecture through many figures and tables, which illustrate how the best architecture parameters are discovered, how the JAT model performs and which parameters have a significant influence on the model. The results on the large ANI-1 and small EAN dataset are presented first, which is followed by a discussion thereof.

4.1 ANI-1 results

The ANI-1 dataset is a very large collection of small organic molecules with calculated DFT energies for over 20 million displaced conformations. The dataset is divided into 8 files, respectively constituent of all molecules with 1 - 8 heavy (non-hydrogen) atoms. A data loader is provided with the data files, which sum up to 5.7 GB of uncompressed data. Models are trained on small parts of the total dataset and evaluated on holdout samples of the same subsets. For example, the subset 4 contains a total of 650K samples, while subset 5 has over 1.8M conformations. Only a fraction of the configurations are drawn from each subset to keep the total training set of reasonable size, such as when training on all subsets with up to 7 heavy atoms.

In particular, during development of the JAT architecture the subsets {3} with 121K or {1, 2, 3} with 202K conformations were used. The dataset for model architecture comparison runs uses 10% of a molecules' conformations and up to 50K conformations from each of the subsets {1, 2, 3, 4, 5, 6, 7}, which yields a total of 199K conformations as the training set and 21K each for the validation and test set.

The production runs select 5% of a molecules' conformations and up to 200K from each of the subsets {1, 2, 3, 4, 5, 6, 7} unless otherwise specified, which yields a total of 4910 molecules with 355K conformations in the training and 39K in the validation set, setting aside 555 molecules with 43K conformations as the test set. All isomers of $C_2H_6N_2$ and $C_4H_4N_4$ were excluded to evaluate generalization to out-of-sample conformers.

To find a good architecture and explore whether new features improve or degrade the model, hundreds of training runs have been performed throughout the development of the architecture and code to train it. Discovery of what generally works and what doesn't is enabled by tracking all experiment runs and their configurations, settings and hyperparameters with the *weights and biases* [Bie20] library. This generally enables reconstruction of which settings achieved the best performance after trial and error throughout many experiments. With this approach an architecture with the following generally well-performing parameters for the ANI-1 dataset is incrementally established.

- 4 - 6 JAT layers of dimensionality 48
- 4 attention heads
- Inter-atomic distance as edge features instead of smoothed inverse distance
- 3 Å graph cutoff
- Skip connections
- Log-cosh loss parameter $\alpha = 0.1$
- Learning rate schedule: min 5×10^{-5} , max 4×10^{-4} , end 5×10^{-6}
- ~ 30 epochs with batch size 32
- ~ 2500 conformations/sec runtime on Nvidia RTX 2080 GPU

Choosing a larger graph generation cutoff parameter is important for subsets of larger molecules, which determines the receptive field of the model in combination with the model's depth. For subsets {1, 2, 3} a cutoff of 1.9 Å is sufficient, while for larger molecules in the subsets {1, 2, 3, 4, 5, 6, 7} 3 Å are necessary to achieve good performance.

To verify the general architecture is nearly optimal, 7 production runs are performed with an exactly identical configuration up to a single variation, which is changed from the reference configuration of 4 message passing layers of dimensionality 48 with 4 attention heads and skip connections. Figure 4.1 shows the energy MAE for the validation and test set of those 7 architecture variations trained on 561K conformations with up to 7 heavy atoms, while table 4.1 documents the validation and test set energy MAE and time required to train these architectures.

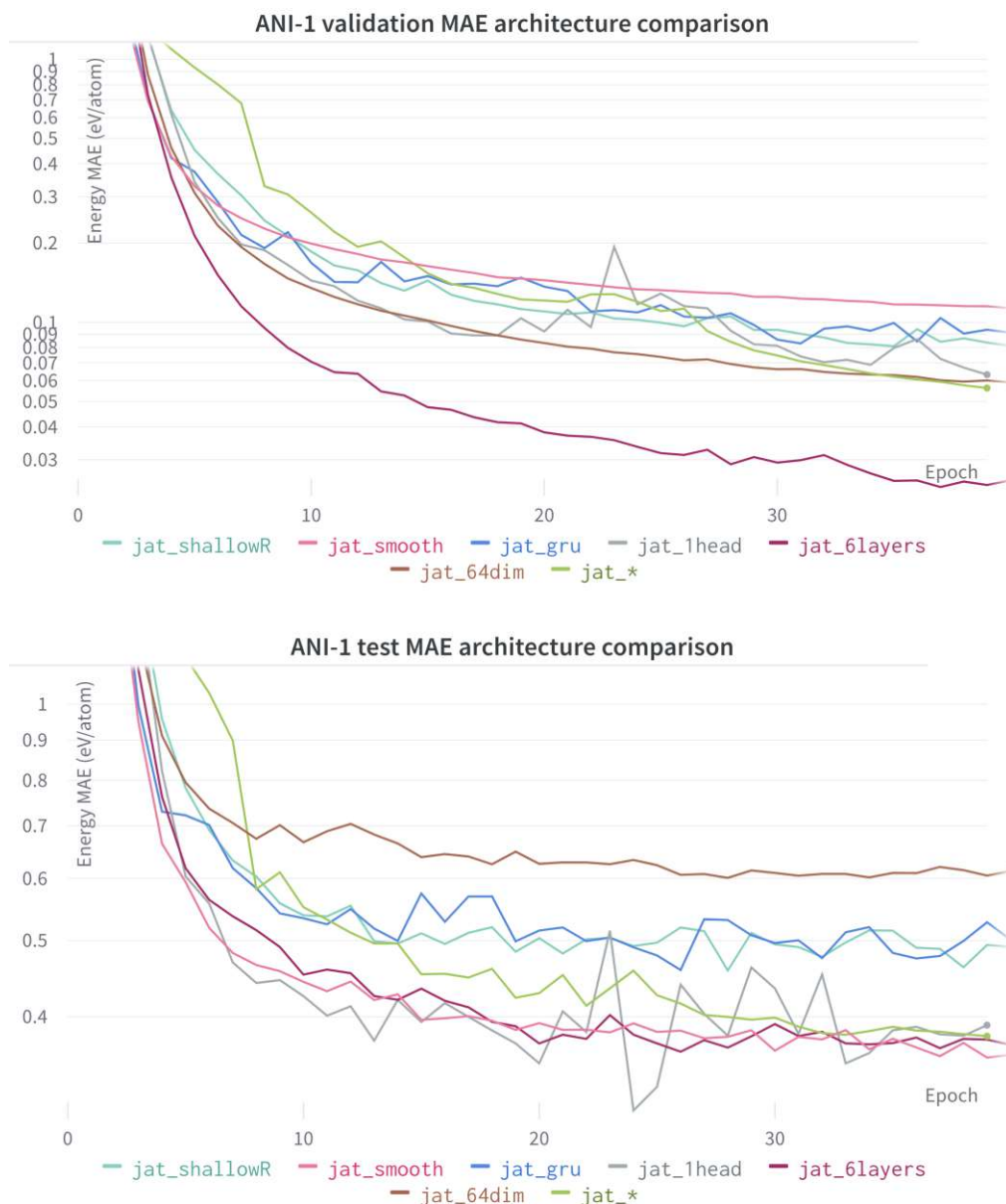


Figure 4.1: Comparison of the energy MAE (eV/atom) on the validation (top panel) and test set (bottom panel) for variations in the JAT architecture.

*jat_** denotes the reference configuration while the legend names respectively indicate the changes thereof. *jat_** uses 4 message passing layers, each of dimensionality 48 with 4 attention heads and skip connections.

gru uses the GRU update instead of skip connections, *1head* uses only a single attention head, *64dim* increases the dimensionality of each layer to 64 and *6layers* uses 6 instead of 4 message passing layers. *Smooth* uses a smoothed inverse inter-atomic distance function for the edge features, while *shallowR* uses a shallow regression readout head of dimensionality 64:24:1 instead of the regular 64:32:16:16:1.

	Validation MAE (eV/atom)	Test MAE (eV/atom)	Time to train 40 epochs (minutes)	Configurations per second (est. it/s)
jat_gru	0.0944	0.4472	58	2500
jat_1head	0.0632	0.3902	40	3400
jat_64dim	0.0534	0.5815	58	2400
jat_6layers	0.0194	0.3536	65	2200
jat_smooth	0.1071	0.3555	44	3300
jat_*	0.0562	0.3778	53	2700
jat_shallowR	0.0646	0.4973	44	3300

Table 4.1: Comparison of the energy MAE (eV/atom), throughput and total time required to train multiple JAT model variations for 40 epochs on the ANI-1 dataset with up to 7 heavy molecules. The architectures correspond to the ones shown in Fig. 4.1. 10% of every molecule and up to 50K configurations per heavy subset were selected to yield a total of 199K configurations in the training set and 21K configurations each in the validation and test set.

An estimate of the average throughput of the architecture is provided but varies significantly during training. This is partially due to using the GPU as both the host systems’ display driver and model training, introducing varying loads throughout the training epochs. An epoch takes about 1 to 2 minutes to cycle through 199K training configurations and evaluate the validation and test set once.

Training set scaling laws

The impact of the size of the training set is evaluated by training identical architectures on training sets of varying size, while the size of the validation and test set is held constant. Fig. 4.2 shows the test set energy MAE after 30 epochs of training on 76K, 152K, 304K and 560K samples containing configurations with up to 8 heavy atoms, while table 4.2 documents the results. The validation and test set are each composed of 76K samples and their energy MAE is indistinguishable, hence only the test set plot is shown. Note that this comparison was performed with a log-cosh parameter $\alpha = 10$.

	Validation MAE (eV/atom)	Test MAE (eV/atom)	Time to train (minutes)
jat_76K	0.3897	0.3873	30
jat_152K	0.1066	0.1064	56
jat_304K	0.0542	0.0547	91
jat_560K	0.0343	0.0346	147

Table 4.2: Training set scaling laws on the ANI-1 dataset. Model architecture as well as validation and test set size are held constant at 76K, while the JAT model is trained on increasingly large training sets. Validation and test set results are almost identical, while the time required to train the models scales with their performance.

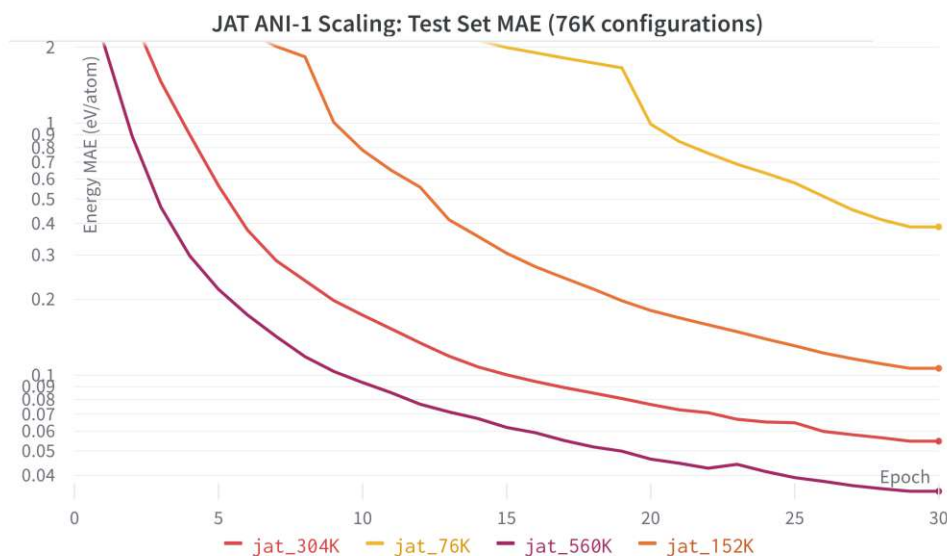


Figure 4.2: Training set scaling laws on the ANI-1 dataset. Model architecture as well as validation and test set size are held constant at 76K, while the JAT model is trained on increasingly large training sets. The impact of additional samples on the performance can clearly be seen, showing the importance of additional data over model capacity.

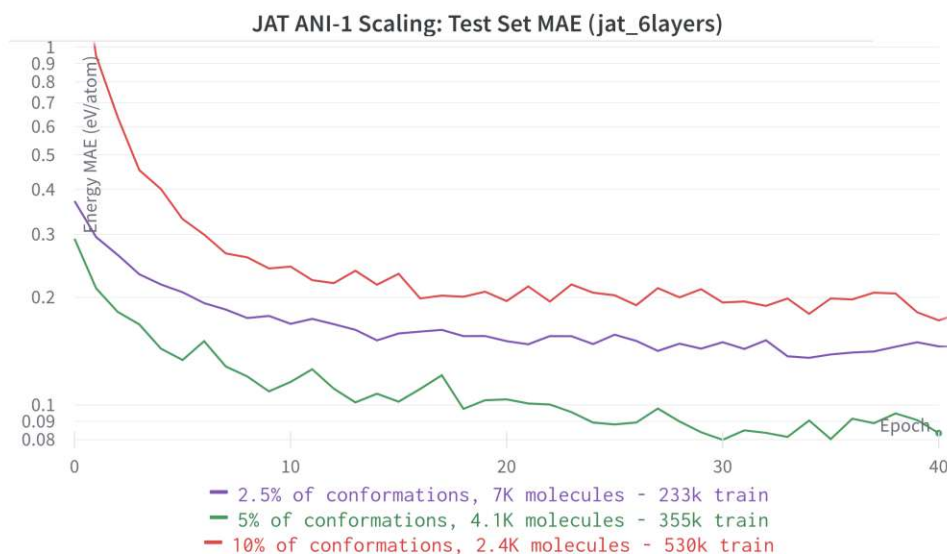


Figure 4.3: JAT scaling laws with respect to dataset selection. The reference JAT model with 6 layers is trained on three different datasets drawn by selecting 2.5%, 5% or 10% of conformations per molecule with a limit of 200K conformations per heavy subset. A smaller fraction yields more molecules and fewer conformations in total, but despite a smaller absolute training set size the model is able to generalize better.

Sampling fraction of conformations and molecules

Similarly, how the conformations and molecules are selected impacts the difficulty of the training, validation and test set as well as the model’s performance. While larger datasets lead to superior results, training on more molecules with fewer conformations exposes the model to more diversity and thus lead to better generalization on the test set, shown in Fig. 4.3. Rather than selecting all conformations of a few molecules, it is very beneficial to randomly draw 2.5 - 10% of conformations for every molecule. This drastically increases the number of molecules in the dataset without increasing the total size of the training data. Despite being smaller in absolute size, drawing only 5% of conformations allows faster training of the model with better generalization to the test set.

	Train conf	Molecules	Validation MAE (eV/atom)	Test MAE (eV/atom)	Time to train (minutes)
2.5%	233K	7K	0.0388	0.1368	74
5%	355K	4.1K	0.0220	0.0834	104
10%	530K	2.4K	0.0187	0.1843	143
5% {1:8}	892K	40.8K	0.0165	0.0296	278

Table 4.3: JAT scaling laws with respect to the sampling fraction of conformations and molecules, measured as the MAE (eV/atom) on the validation and test set after training for 40 epochs with 6 message passing layers.

The final run labeled {1:8} selects 5% of molecules for all subsets including 8 heavy atoms. This model uses 5 layers of dimensionality 48 and specifications as the reference architecture and is the best model run performed. The results are shown at epoch 37, after which the model starts overfitting to the validation set and the test set error starts diverging, as can be seen in the figure of the full run around epoch 50 (Fig. 4.5).

Number of message passing layers

To determine the optimal depth for the JAT model, three more runs were performed on the large 5% dataset with up to 7 heavy molecules. Fig. 4.4 visualizes the validation and test set MAE for 5, 6 and 7 JAT layers. Table 4.4 documents the corresponding results after 40 training epochs for all three models, as well as after 80 epochs for the best-performing architecture with 5 layers.

Production run on all ANI-1 subsets

A final run applies the best recipe and includes subset {8}, sampling 5% of configurations of 40764 molecules with a 500K limit from each subset for a total of 892K training and 99K validation conformations, almost 5% of the entirety of ANI-1. The model is evaluated on 109K conformations of 4590 molecules exclusive to the test set. The full run is shown in Fig. 4.5 and table 4.3, which trains 9 hours for 80 epochs on a GPU.

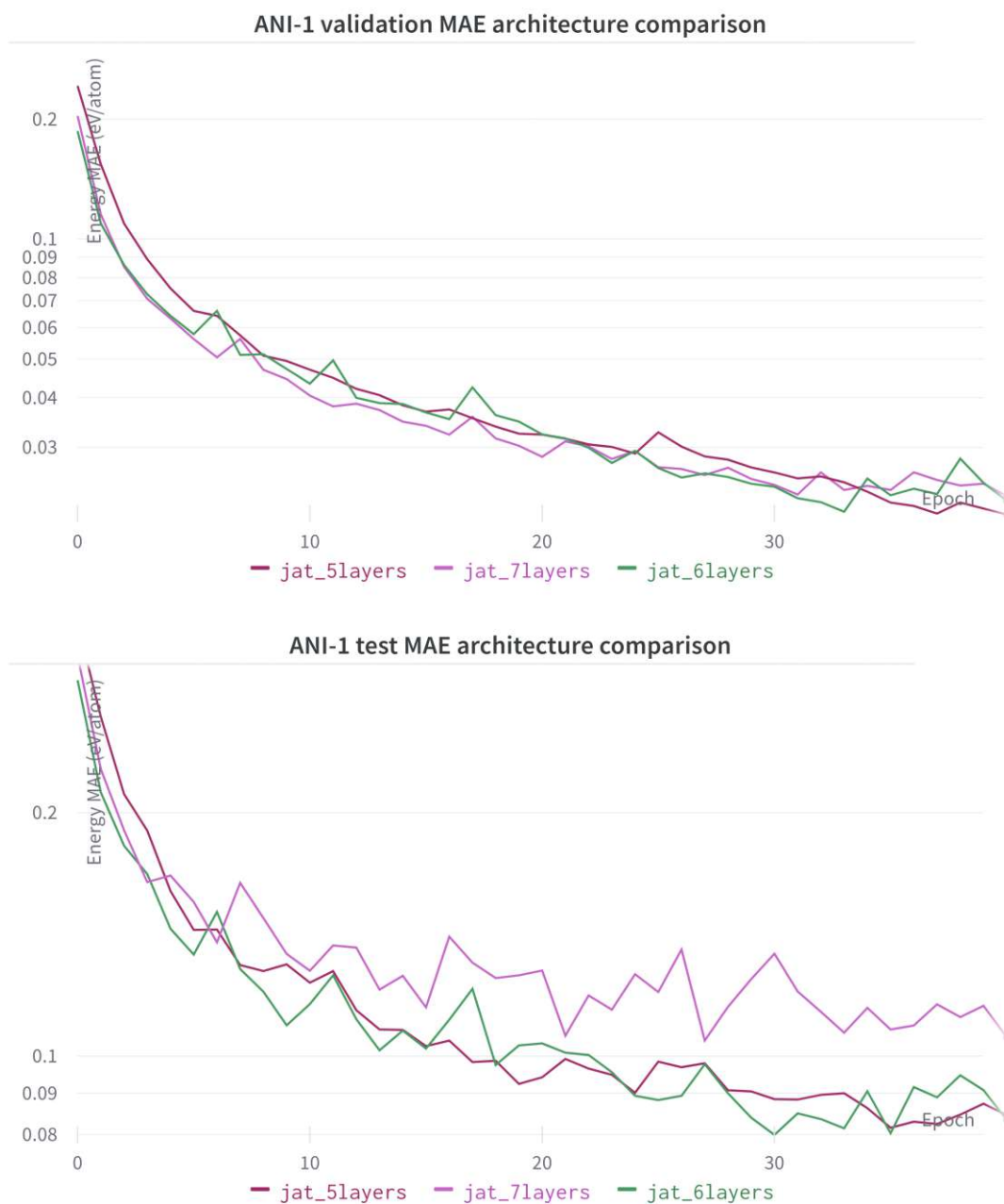


Figure 4.4: JAT architecture comparison between 5, 6 and 7 message passing layers. Here the energy MAE (eV/atom) for the validation and test set is shown, highlighting the divergence on the test set. All models were trained on 5% of conformations of up to 7 heavy atoms with 4100 molecules.

4. RESULTS

	Validation MAE (eV/atom)	Test MAE (eV/atom)	Time to train (minutes)
5 layers	0.0211	0.0873	94
6 layers	0.0245	0.0908	104
7 layers	0.0243	0.1154	119
5 layers 80 epochs	0.0136	0.0700	190

Table 4.4: JAT architecture comparison between 5, 6 and 7 message passing layers, measured as the MAE (eV/atom) on the validation and test set after training for 40 epochs. 5% of conformations per molecule with up to 7 heavy atoms were used as the training data. Additionally, the MAE of the 5-layer architecture after 80 training epochs is documented. It should be noted that for this dataset, further training epochs significantly decrease the models' validation statistics, while the test set statistics improve by a smaller margin.

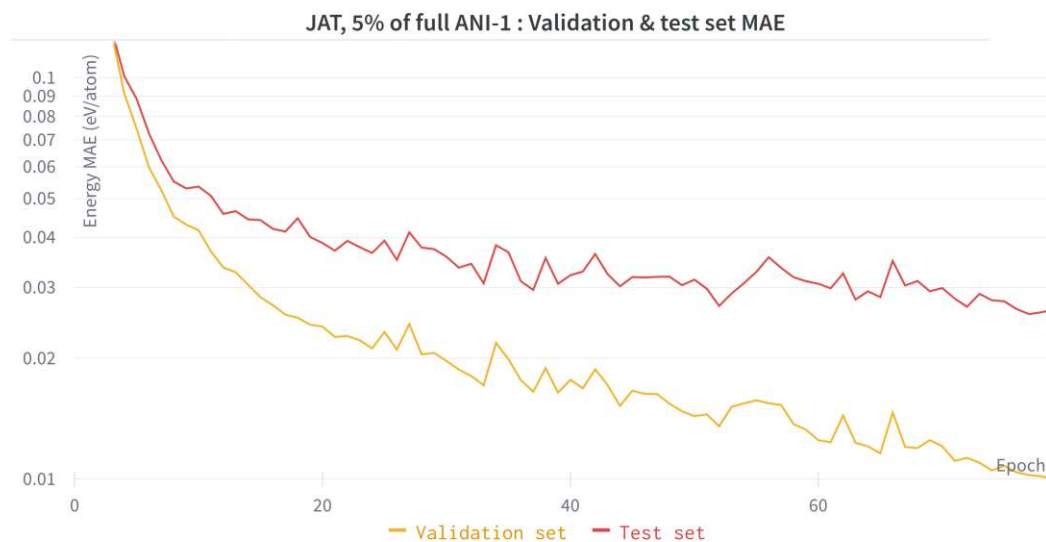


Figure 4.5: Production run of the best JAT architecture training on the largest subset of ANI-1 used. 5% of ANI-1, using all heavy subsets with a limit of 500K each, yield a total of 892K training and 99K validation conformations of 40.8K molecules. The test set has 109K conformations of 4590 molecules. The plot shows the MAE (eV/atom) for both the validation and test set as training progresses for 80 epochs.

Stratified analysis of heavy subsets

To analyse the model’s predictions for different sizes of molecules, molecules are stratified into subsets of N heavy atoms and evaluated separately. An increase in the number of heavy atoms results in many more relevant data points, due to increases from three distinct sources as elaborated in Sec. 2.1.6. More combinations of N heavy atoms (isomers) are possible, each of which has many molecules. For a given molecule, many conformations are sampled using the molecules’ normal modes of vibration, of which there are $3N - 6$ for non-linear molecules due to its degrees of freedom.

To illustrate this, table 4.5 lists statistics about the distribution of molecules in the ANI-1 dataset. Subset 1 to 4 make up a total of 97 molecules, while subset 7 and 8 constitute the large majority of the dataset. Of note is subset 8, where some molecules only have 7 conformations (both C_8H_{10} and $C_4H_4N_4$), while the largest (C_8H_{18}) has just 432 conformations. From subset 6 and 7, $C_4H_6N_2$ with 56 and C_7H_8 with 36 conformations are noticeable. C_4H_{10} is the molecule with the by far most conformations of a single molecule in the dataset (17280).

The maximum denotes the number of conformations for a single molecule, but since there are multiple conformers more conformations than indicated by the maximum in table 4.5 are contained for a given isomer. For example, the isomer $C_2H_6N_2$ has 4 conformations with ~ 11520 conformations each for a total of 46078. The isomer $C_4H_4N_4$ has a total of 119 configurations with 7 to 180 conformations each, for a total of 19848 in the ANI-1 dataset.

Heavy subset	Number of molecules	Total number of conformations	Conformations of smallest molecule	Conformations of largest molecule
1	3	10800	1800	5400
2	13	50962	480	8469
3	20	151200	1440	12960
4	61	651936	2880	17280
5	267	1813151	2880	10800
6	1406	1682245	56	1944
7	7760	6460162	36	1512
8	47932	11236918	7	432

Table 4.5: ANI-1 dataset statistics about molecules and their conformations. For each subset of N heavy atoms, the total number of conformations as well as the minimum and maximum conformations of a single molecule are listed.

Heavy subset	Val MAE (eV/atom)	Val RMSE (eV/atom)	Val conf	Test MAE (eV/atom)	Test RMSE (eV/atom)	Test conf
1	1.0025	1.0826	49	-	-	0
2	0.0480	0.1585	221	0.1258	0.2174	311
3	0.0177	0.0286	682	0.0550	0.0643	650
4	0.0122	0.0174	2848	0.3309	0.8166	2453
5	0.0125	0.0176	8166	0.0559	0.1856	10209
6	0.0196	0.0493	7770	0.0328	0.0970	7092
7	0.0165	0.0260	29618	0.0219	0.0842	31937
8	0.0144	0.0401	49784	0.0169	0.0785	56499

Table 4.6: JAT model prediction MAE and RMSE for stratified splits of the validation and test subset. *Heavy subset* refers to the number of heavy atoms constituting the subset split. Validation and test subsets are identical to the ones used during training. The trained production run model shown in Fig. 4.5 after 40 training epochs is used to predict the energy of these conformations.

Dissociation during reactions

The reaction $2\text{C}_2\text{H}_6\text{N}_2 + 4\text{H}_2 \rightleftharpoons \text{C}_4\text{H}_4\text{N}_4$ offers an example for the dissociation of molecules into constituent parts or their reverse reaction. All conformations of all $\text{C}_2\text{H}_6\text{N}_2$ and $\text{C}_4\text{H}_4\text{N}_4$ isomers are excluded for training, thus all conformations are outside of the training set to allow for the inspection of the model’s behaviour on these molecules. The model used for this analysis is the production run model trained for 40 epochs on the largest subset including molecules with 8 heavy atoms, shown in Fig. 4.5.

The energy of the 46078 conformations of $\text{C}_2\text{H}_6\text{N}_2$ are predicted with an energy MAE of 0.04177 eV/atom and RMSE of 0.0457 eV/atom, while the 19848 $\text{C}_4\text{H}_4\text{N}_4$ conformations prediction errors are 0.0125 eV/atom MAE and 0.0177 eV/atom RMSE. The model is able to accurately model both main constituent molecules in this reaction, and while reproducing the reaction itself requires MD simulations, this is a good indicator that the model is able to accurately predict dissociation as well as the transition between states.

4.2 EAN results

The second dataset is of the ionic liquid ethylammonium nitrate (EAN). The dataset was generated by performing a MD simulation using OPLS-AA with periodic boundary conditions and sampling 741 configurations of the trajectory of 15 anion-cation EAN pairs. These snapshots of the MD trajectory are then evaluated using DFT to obtain accurate estimates of the potential energy and atomic forces, against which the model's predictions are evaluated to obtain the loss used to train the JAT architecture.

This is the same dataset used to train Neurall, which serves as a baseline against which to evaluate the JAT model. [MCCB⁺21] While developing the JAT architecture, a smaller dataset with only 6 EAN pairs was used to accelerate the training process. All models are trained on the atomic forces, for which the following hyperparameters have been found to perform best.

- 5 - 9 JAT layers of dimensionality 32 - 64
- 1 attention head (multiple heads do not improve performance)
- Inter-atomic distance edge features instead of smoothed inverse distance
- 5 Å graph cutoff
- Skip connections
- Log-cosh loss parameter $\alpha = 10$
- Learning rate schedule: min 5×10^{-4} , max 3×10^{-3} , end 5×10^{-6}
- ~500 epochs with batch size 8
- ~100 configurations/sec runtime on Nvidia RTX 2080 GPU

In a similar fashion to the architecture comparison on ANI-1, a comparison for the EAN dataset is performed to discover an optimal combination of the JAT architecture. Eight variations of the reference architecture are compared, which differ from *jat_** with 5 JAT layers of dimensionality 32 with one attention head, skip connections and a shallow readout head.

gru uses the GRU update instead of skip connections, *64dim* doubles the dimensionality of each layer to 64 and *Nlayers* respectively use N message passing layers. *Smooth* uses a smoothed inverse distance function for the edge features (inter-atomic distance). Combinations, such as *7layers_48dim* indicate combinations of the above modifications. Fig. 4.6 visualizes the forces MAE (eV/Å) for each architecture variation, while table 4.7 documents the corresponding results.

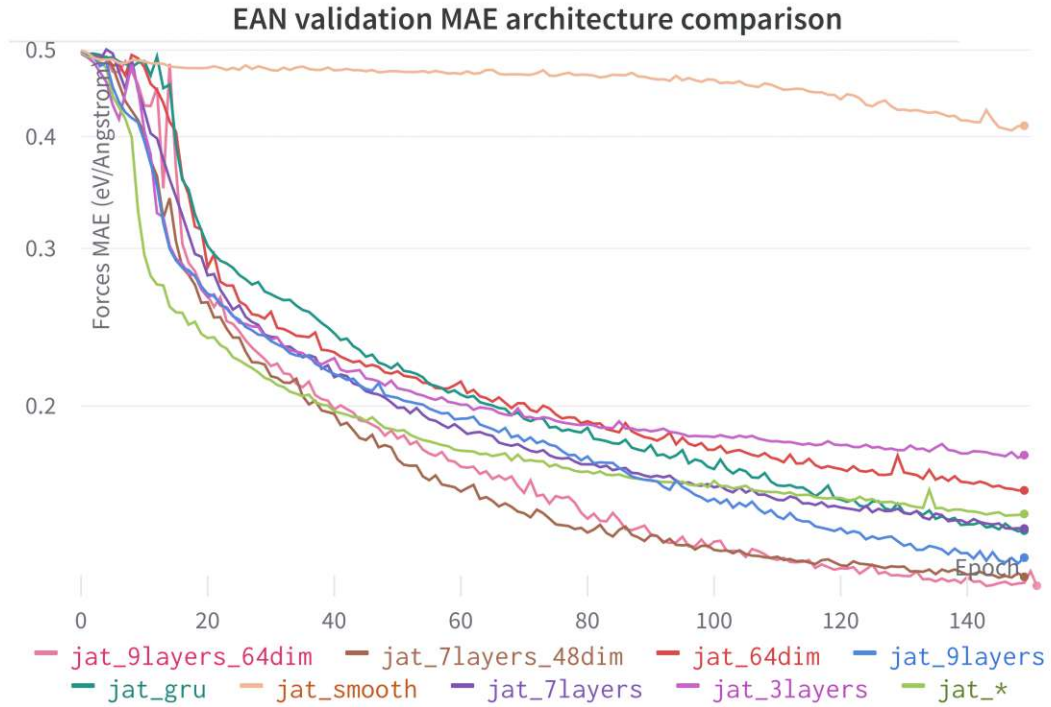


Figure 4.6: Comparison of validation set forces MAE ($\text{eV}/\text{\AA}$) for variations in the JAT architecture on the EAN IL dataset. *jat_** denotes the reference configuration while the legend names respectively indicate the variations thereof.

Model Variant	Validation MAE ($\text{eV}/\text{\AA}$)	Test MAE ($\text{eV}/\text{\AA}$)	Validation RMSE ($\text{eV}/\text{\AA}$)	Test RMSE ($\text{eV}/\text{\AA}$)
jat_gru	0.1451	0.1368	0.2660	0.2755
jat_smooth	0.4114	0.4227	0.6733	0.7403
jat_64dim	0.1609	0.1518	0.3001	0.2947
jat_3layers	0.1762	0.1677	0.3103	0.3136
jat_*	0.1514	0.1399	0.2734	0.2655
jat_7layers	0.1458	0.1361	0.2606	0.2554
jat_9layers	0.1353	0.1275	0.2485	0.2528
jat_7layers_48dim	0.1288	0.1204	0.2332	0.2397
jat_9layers_64dim	0.1307	0.1230	0.2384	0.2474

Table 4.7: Comparison of validation and test set MAE and RMSE of the forces ($\text{eV}/\text{\AA}$) for variations in the JAT architecture on the EAN IL dataset. *jat_** denotes the reference configuration, while the suffixes correspond to Fig. 4.6.

Increasing the message passing layers and dimensionality increases the model’s capacity and generally improves results, however only up to a point. Beyond 7 layers with dimensionality 48, additional layers make training much more unstable (see also Fig. 4.11) and can lead to quicker convergence, which however does not ultimately improve the accuracy compared to smaller architecture variants.

Influence of randomness

To quantify the influence of randomness on the model’s performance, five identical JAT models were trained on the EAN dataset with different random seeds. All hyperparameters except the random seed are held constant, which influences both the model initialization and dataset shuffling before the split into the training, validation and test set.

To isolate the influence of the random initialization of the model, five more models were trained with identical dataset splits to ensure *only* the models’ initialization differs between runs. By keeping the seed constant and only ensuring a different initialization, the influence of the random seed on data shuffling is removed, ensuring identical training, validation and test splits.

For both experiments, the validation and test set MAE are plotted in Fig. 4.7 as the mean (line) and its standard error (shaded area) for the forces MAE, while table 4.8 documents the corresponding results.

	Validation MAE (eV/Å)	Test MAE (eV/Å)	Validation RMSE (eV/Å)	Test RMSE (eV/Å)
best	0.1498	0.1399	0.2734	0.2655
worst	0.1876	0.1786	0.3426	0.3674
committee	0.1673 σ 0.0178	0.1620 σ 0.1606	0.3068 σ 0.0289	0.2982 σ 0.0404
best (init only)	0.1514	0.1399	0.2723	0.2655
worst (init only)	0.1757	0.169	0.3196	0.3181
committee (init only)	0.1620 σ 0.0106	0.1531 σ 0.0120	0.2963 σ 0.0225	0.2932 σ 0.0230

Table 4.8: Comparison of two sets of five identical runs of the JAT model trained on the EAN dataset to quantify uncertainty, corresponding to Fig. 4.7. The top three entries correspond to the left column of plots where randomness influences both initialization and dataset shuffling, while the bottom three (labeled *init only*) hold constant the dataset shuffling and splits to isolate the randomness to initialization.

4. RESULTS

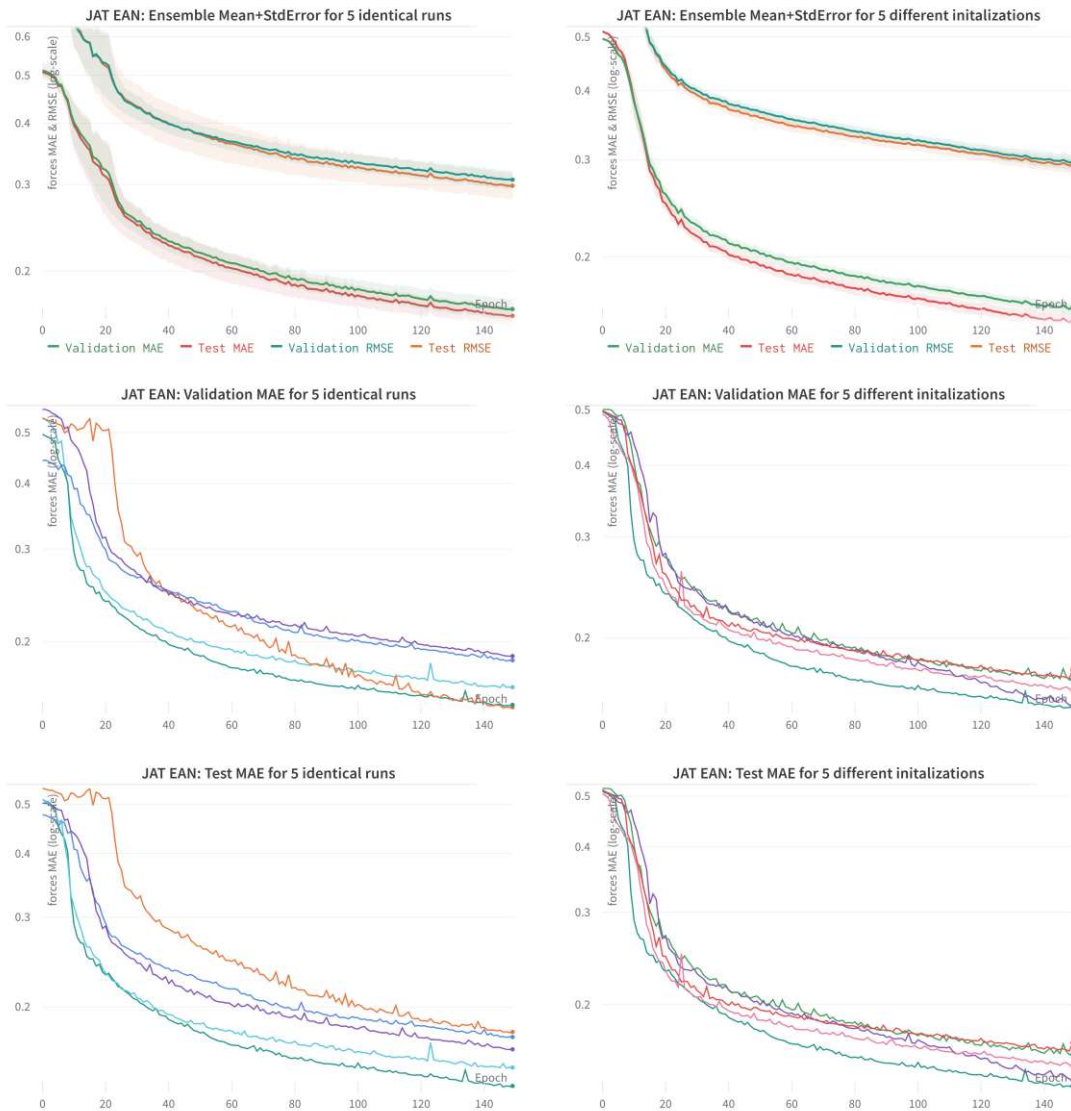


Figure 4.7: Comparison of two sets of five identical runs of the JAT model trained on the EAN dataset to quantify uncertainty. In the three figures on the left, hyperparameters are identical for all five runs except for the random seed which influences both initialization and dataset shuffling. The three panels on the right are identical to the left ones, except that the dataset split is held constant, thus isolating the impact of randomness from model parameter initialization.

The top figure shows the committee mean (line) and standard error (shaded area) for the validation set (green, turquoise) and test set (red, orange). The middle and bottom rows respectively show the validation and test set MAE for the five individual runs. All Y-axes are log-scaled for visibility.

While some variance remains, the error bars (standard error) are much tighter for the right panels. This suggests that different dataset splits have a larger impact on the final models' variance in performance than does initialization.

Baseline comparison

Next, the JAT model is compared to the baseline architecture NeuralIL to evaluate the accuracy and runtime of both methods. The NeuralIL model is trained with the reference hyperparameters provided, which uses Bessel descriptors with a 3.5 Å radial cutoff with 4 basis functions and 2 embedding dimensions for the atomic species. The model uses a learning rate schedule with a minimum, maximum and final rate of 1×10^{-3} , 1×10^{-2} and 1×10^{-5} and trains for 500 epochs.

The best JAT model discovered during the architecture comparison is selected, which uses 7 message passing layers with dimensionality 48, one attention head, skip connections and a 5 Å graph cutoff. It is trained for 3000 epochs using a log-cosh parameter of 10 and a learning rate schedule with a minimum, maximum and final rate of 5×10^{-4} , 3×10^{-3} and 5×10^{-6} .

To ensure comparability, both models were trained from scratch on the same system, ensuring equal compute resources. The JAT architecture trains approximately $4\times$ faster, and therefore Table 4.9 also compares the models after training JAT for a similar time frame as required to train NeuralIL for 500 epochs, even though the JAT model retains the speed advantage during inference. Fig. 4.8 visualizes the training against the NeuralIL baseline, while Fig. 4.9 shows the forces MAE and RMSE of the validation and test set for 3000 training epochs.

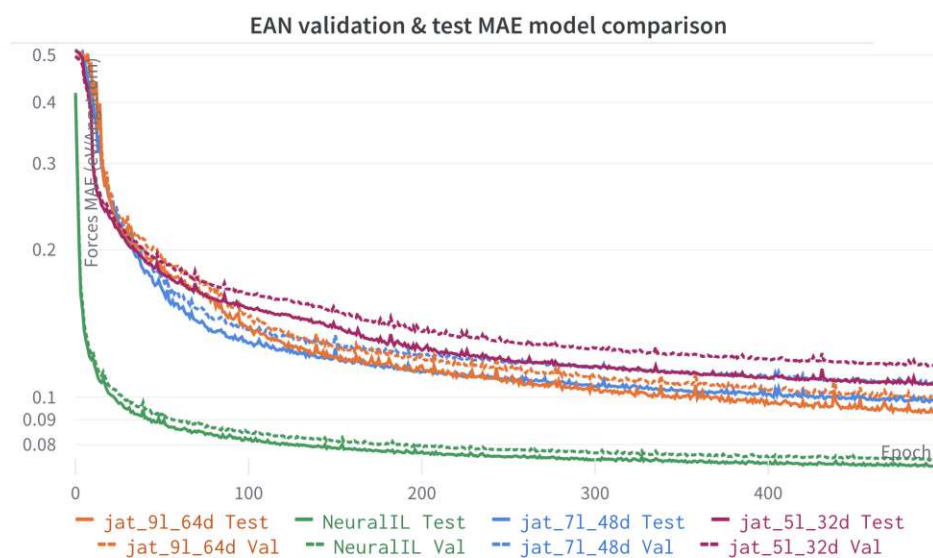


Figure 4.8: Comparison of the best JAT architectures against the NeuralIL baseline. Validation (dots) and test set (lines) MAE are shown for three variants of the JAT model as well as the reference NeuralIL architecture.

4. RESULTS

	Validation MAE (eV/Å)	Test MAE (eV/Å)	Validation RMSE (eV/Å)	Test RMSE (eV/Å)	Time to train (minutes)
NeuralIL	0.0747	0.0724	0.1233	0.1446	156
jat_9layers_64dim	0.0992	0.0935	0.1682	0.1858	63
jat_7layers_48dim	0.1063	0.0976	0.1881	0.1958	39
jat_5layers_32dim	0.1157	0.1061	0.2075	0.2023	22
jat_7layers_48dim after 1775 epochs	0.0877	0.0832	0.1519	0.1617	~156
jat_7layers_48dim after 3000 epochs	0.0818	0.0793	0.1386	0.1699	263

Table 4.9: Comparison of the best JAT architectures against the NeuralIL baseline after 500 training epochs. Validation and test set MAE and RMSE are documented for three variants of the JAT model as well as the reference NeuralIL architecture. Additionally, the results after 1775 and 3000 epochs are listed for comparison using a similar time to train the models.

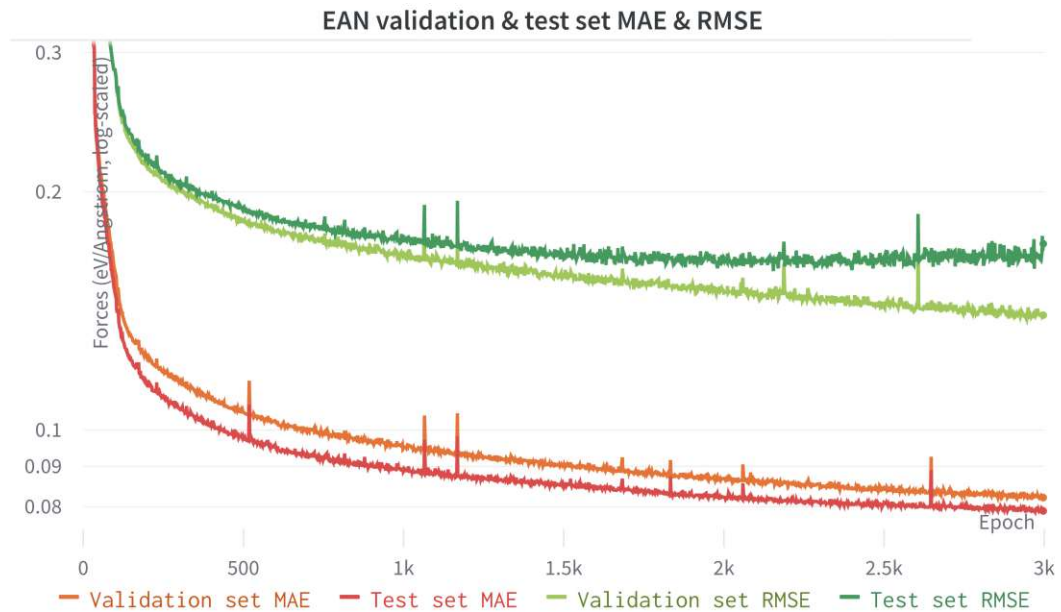


Figure 4.9: Validation and test set MAE and RMSE of the best JAT model with 7 layers and dimensionality 48, trained on the EAN dataset for 3000 epochs. Of note is the early plateau of the RMSE on the test set, whereas the other errors continue decreasing as training progresses.

Dissociation of a N–O bond

To study the model's ability to predict dissociation, a N–O bond is *pulled apart* and the JAT model's projected predictions of the forces are plotted compared to the DFT reference.

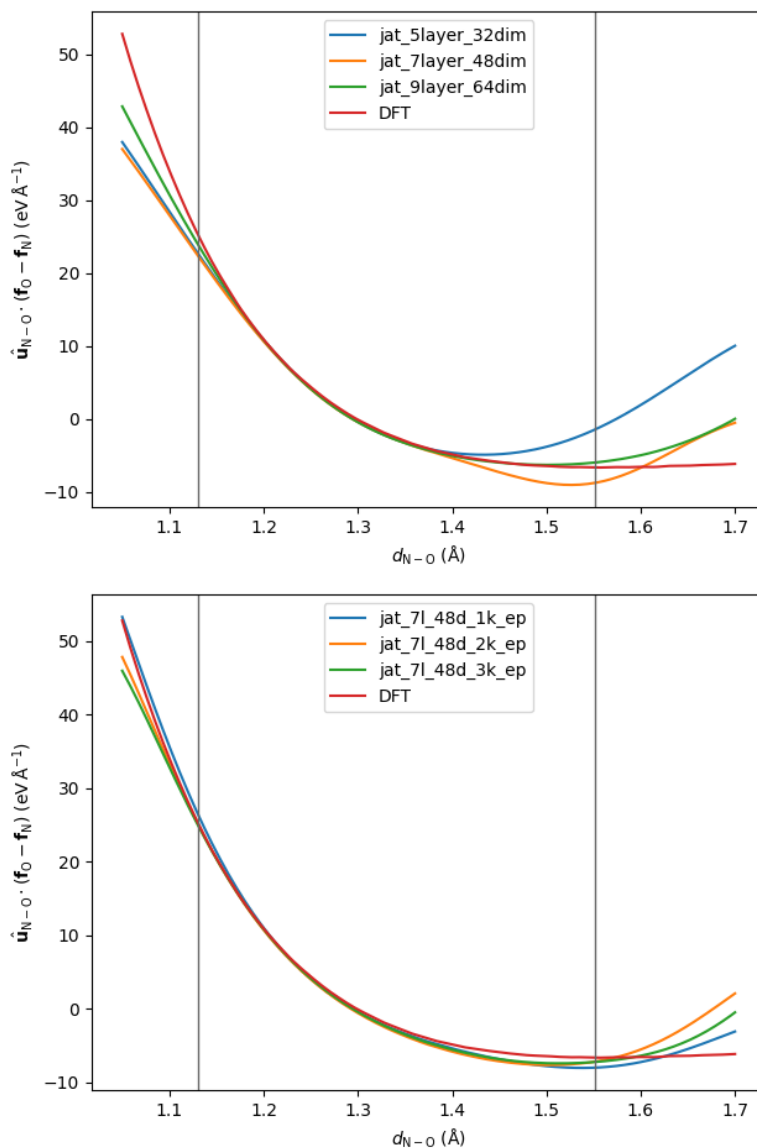


Figure 4.10: JAT prediction of the projected forces during dissociation of a N–O bond of an EAN molecule. The top panel compares multiple JAT models after 500 training epochs, while the bottom panel shows the same model during different stages of training, respectively after 1000, 2000 and 3000 epochs. The vertical bars indicate the boundaries of the training data, which is centred at 1.267 \AA where the force is zero.

Collapse of very large JAT architectures

Finally, Fig. 4.11 showcases the collapse of some initializations of the largest JAT variant, where the error and training loss temporarily increase drastically. This behaviour was only observed for the largest variant of the model with 9 layer of dimensionality 64 and is likely caused by the learning rate schedule.

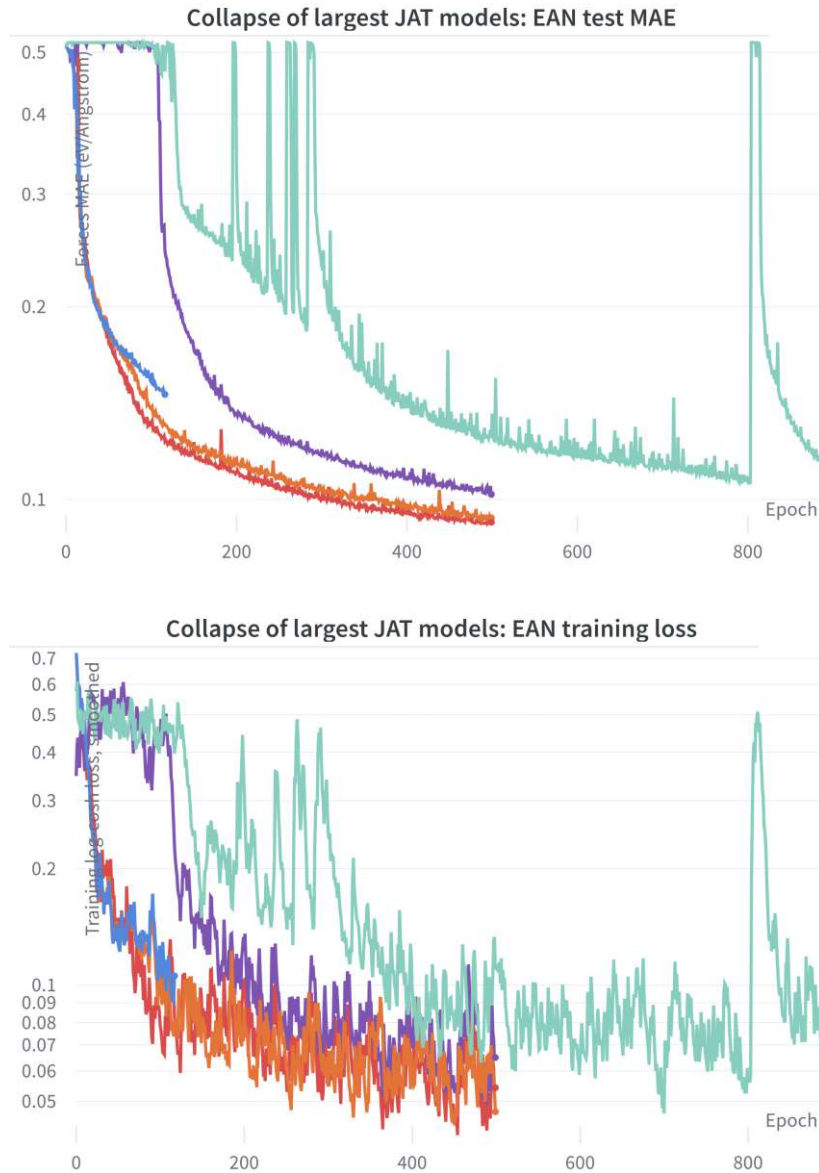


Figure 4.11: Collapse of some initializations of the largest JAT variant presented, which fail to learn for over 100 epochs (purple, turquoise). The turquoise model seems to occasionally collapse when training for 1000 epochs.

4.3 Discussion of ANI-1 results

The JAT architecture is able to reach chemical accuracy when predicting the energy of small organic molecules while using only a small fraction of ANI-1 as the training set to keep the total time and necessary computational resources within reasonable scope. It achieves a MAE of 16 to 30 meV/atom, respectively the average over the validation and test set. All experiments which explore scaling laws suggest using the entire dataset would significantly improve the model’s validation, but especially the test set results.

The stratified analysis shows excellent accuracy across most heavy subsets, where the outlier for the 1 heavy atom split can be explained with only 3 molecules in the dataset, leaving no molecule to the test set. When measuring the RMSE on stratified splits into heavy atom subsets, the difference between the validation and test set is larger than for the MAE but of excellent accuracy nonetheless. On subsets of 6 - 8 heavy atoms the RMSE falls in the 25 - 80 meV/atom range on the validation and 65 - 95 meV/atom on the test set. The test set RMSE shows outliers for the subsets 2, 4 and 5, which might be alleviated by including more than 5% of conformations for those small molecules.

Architecture

On ANI-1, the JAT architecture performed best with 4-5 message passing layers and 4 attention heads with a dimensionality of 48 to 64. Any parameter change leading to increased model capacity is generally beneficial, however adding more than 6 layers is generally not advantageous considering the significant increase in runtime. This might be due to the rather small diameter of larger molecules in the dataset, for which the receptive field is large enough with a graph cutoff of 3 Å. Further increasing the dimensionality, depth and capacity of the model would likely be beneficial when training on the full dataset, although diminishing returns are to be expected. Most attempted variations of architectural components performed poorly, such as the GRU update which is not justified due to its significant increase in complexity with no performance gains, or the smoothed inverse distance as edge features which performed significantly worse.

Dataset scaling

The first set of production runs used a rather large characteristic scale argument for the log-cosh loss of $\alpha = 10$ eV, which led to a divergence between the MAE and RMSE prediction errors. Changes in how conformations are sampled for the training, validation and test set increased the diversity of molecules and thus made the validation and test set significantly more challenging. Including most molecules but only ~5-10% of its conformations and using $\alpha = 0.1$ eV led to much better training behaviour with sufficiently large training sets and ultimately improved validation and test set performance as well as the model’s ability to generalize to novel molecules, in particular for the RMSE.

Further experiments were performed by using different fractions of conformations for each molecule. It shows that sampling 5% of 4.1K molecules yields an overall smaller dataset, on which the model significantly outperforms the 10% variant despite its significantly smaller size (355K rather than 530K conformations in the training set). The final run uses the best architecture and largest dataset with 891K conformations which includes molecules with 8 heavy atoms and trains on 5% of ANI-1.

Training stability and convergence

No noticeable differences due to different allocation of molecules to the training, validation and test set due to randomness were observed, which can be explained by the size of the dataset making it highly unlikely for mostly difficult molecules to be allocated to one but not the other set. Training on ANI-1 is generally very stable, especially when training on most subsets of heavy atom with many conformations. A collapse of the model was not observed and generally, most models achieve at least good accuracy when training sufficiently long.

Generalization to the test set is severely limited by a smaller training set. When using only ~ 220 K conformations to train on, the model reaches a similar validation set MAE but hits a barrier on the test set. Molecules with fewer heavy atoms constitute a larger share of this small training set, but this does not account for the poor test set generalization since this ratio is approximately the same for the test set. By a significant margin, the best test set RMSE results were obtained from the final run which uses the largest dataset.

The predictive accuracy for all conformations of the $C_2H_6N_2$ and $C_4H_4N_4$ isomers suggests the model is able to model non-equilibrium molecules and its conformations with chemical accuracy, which in ANI-1 are sampled from the molecules' normal modes of vibration. This suggests the model would be able to reproduce the aforementioned reaction, transitions between states as well as dissociation. Whether this holds in practice should be further evaluated using MD simulations.

4.4 Discussion of EAN results

For the ionic liquid dataset, the JAT model is able to reach chemical accuracy at a excellent speed. Training the best JAT model for 500 epochs takes one fourth of the time required to train the baseline model NeuralIL on the 15-pair EAN dataset but does not outperform the baseline in terms of accuracy. Instead, ~ 1800 training epochs can be performed in the time required to train NeuralIL for 500 epochs, in which the JAT model reaches a forces validation MAE of 88 meV/Å and RMSE of 152 meV/Å compared to 75 meV/Å MAE and 123 meV/Å RMSE achieved by NeuralIL.

Considering the significant difference in runtime of $4\times$, using the JAT model for MD simulations might be the superior choice since both models reach chemical accuracy. This difference in runtime would further increase in favor of JAT when scaling to larger systems due to its sparse implementation. In comparison, the Bessel descriptors are the most expensive computational component and need to be evaluated separately for additional atoms of larger systems.

Architectures

The best architecture variant found uses 7 message passing layers of dimensionality 48 with a single attention head. 9 layers of dimensionality 64 were also evaluated and do marginally increase performance at the cost of increased runtime and model instability which can occasionally lead to model collapse. It is particularly interesting that 7 to 9 or even more message passing layers are beneficial and not detrimental to the model's performance. The generally accepted consensus advocates for the use of no more than 3 to 4 layers, due to the effects of averaging or over-smoothing of the latent vectors with too many update steps. This is the case for the ANI-1 dataset but not for EAN, where only using 4 layers significantly hinders performance. Possible explanations are the rather large system of 225 atoms or the periodic boundary conditions present in the 15-pair EAN dataset, where increases in the receptive field with depth allow for better modeling of long-range interactions between replicas.

Influence of randomness

Two experiments are performed to evaluate the influence of randomness on the models' training behaviour and accuracy. The first set of five runs includes randomness from both dataset shuffling and model initialization, the second set isolates the source of randomness to initialization. Based on these results, different dataset splits have a larger impact on the final models' variance in performance than does initialization. The small size of the dataset with 741 configurations in total accounts for this difference, since the difficulty of the 50 randomly selected samples respectively used as the validation and test set can vary.

Dissociation

The model's ability to predict dissociated configurations is explored by evaluating the projected forces when pulling apart a N–O bond, which is plotted against the DFT baseline. The models closely fit the DFT reference when the model operates within the boundaries of the training data distribution, while outside thereof the forces are overestimated in the distribution's tail. For the larger models a better and closer fit can be observed as training progresses.

4.5 Outlook and future work

One avenue for further study is to examine whether the predicted scaling behaviour to larger systems and larger datasets holds in practice. Attention-based architectures can effectively scale to very large datasets and tend to show the biggest advantages for the largest datasets compared to other architectures. The scaling behaviour has been explored extensively on ANI-1, but the entire dataset was not used due to limitations in computational resources.

Similarly, the sparse implementation allows the JAT model to scale well to larger systems when compared to regular self-attention on a fully connected graph, but especially when compared to descriptor-based architectures. A limiting aspect is the availability of DFT reference data for large systems. Evaluating the extent of the benefit over other approaches when scaling the JAT model to much larger molecules and simulation boxes is left for future work.

The study of dissociation necessitates both a dataset with representative conformations as well as conformations suitable to evaluate dissociation. This is hindered by the rare natural occurrence of such phenomena and thus necessitates generating suitable reference data to explicitly explore dissociation. Generating such a dataset and evaluating the JAT architecture is another promising avenue for future work, both to validate whether the presented model in its current form is able to describe dissociation and to further optimize the architecture for this task.

On the architecture side, many ideas might be worth exploring. Inclusion of a high-dimensional projection layer similar to the one used in Transformers might significantly increase the model performance. Concretely, this means projecting the latent features to a very high-dimensional space, applying a nonlinearity and projecting the features back to its original dimensionality directly after aggregation in every message passing layer. Furthermore, using a global master node connected to all nodes might enable messages to circumvent the receptive field, which is usually limited by the depth and graph cutoff radius. However, this prevents dividing a large simulation box into independent computations for parallelization, which is crucial for the integration into MD simulation engines.

Performing an exhaustive hyperparameter search can yield a superior combination of parameters and might lead to minor gains in accuracy or efficiency. This is however limited in scope given the size of the ANI-1 dataset and has to be weighted against simply increasing the size of the training dataset, which is likely to offer larger benefits compared to fine-tuning hyperparameters. Finally, inspection of the attention layers' weights might yield insights into the architecture and guide the design of future architectures.

Conclusion

This work explores the applicability of attention-based deep learning architectures by adapting Graph Attention Networks to the prediction of the potential energy surface of molecules. To achieve this, the JAT model generates a sparse molecular graph from the atomic positions and embeds the atomic species. The feature vectors describing each atom are incrementally refined with multiple attention-based message passing layers. A pyramidal readout head transforms the final latent representation into the regression target, each atoms' contribution to the total energy. Their sum is the model's prediction of the potential energy of the molecule or system, from which the atomic forces can be obtained as its gradient.

This approach avoids the computationally expensive evaluation of atomistic descriptors and is therefore significantly faster than the NeuralIL baseline. The JAT model approaches the chemical accuracy of the baseline when trained with similar computational resources while retaining the 4× speed advantage for inference.

Since efficiency is a crucial aspect for the model's application in MD simulations, the JAT architecture leverages sparsity in the graph representation, message passing and attention mechanism to achieve favorable scaling to larger systems. The architecture is implemented in JAX, enabling the efficient evaluation of the forces as well as significant performance gains through the use of a GPU and just-in-time compilation.

The JAT model is trained and evaluated on a very large organic molecule dataset (ANI-1) and a small ionic liquid dataset (EAN) and achieves chemical accuracy at excellent speed. Only the energy but not the atomic forces are available in ANI-1, and achieving chemical accuracy is significantly more difficult when training only on the energies.

5. CONCLUSION

The final production run trains on 5% of the entire ANI-1 dataset with 892K conformations and best generalizes to the test set. It reaches a MAE of $\sim 16 - 30$ meV/atom when predicting the energy of validation and out-of-sample test molecules in ANI-1 after training for 40 epochs. The predictions are similarly accurate for smaller and larger molecules, both for the energy MAE and RMSE. The model shows excellent behaviour when scaling to large datasets, where increasing the number of molecules and total size of the training set drastically improves the generalization to the test set.

For the EAN dataset, JAT predicts the forces with an MAE of $\sim 79 - 82$ meV/Å, slightly worse but $4\times$ faster than the NeuralIL baseline. Particularly interesting is the benefit of 7 - 9 message passing layers, without which the model performs noticeably worse. It is able to accurately predict dissociated configurations in both datasets, such as the N–O bond stretch of an EAN molecule or the vibrations of out-of-sample $C_2H_6N_2$ and $C_4H_4N_4$ isomers.

List of Figures

2.1	Visualization of the reference NeuralIL architecture	17
3.1	Visualization of the entire JAT architecture	34
3.2	Visualization of the graph generator component	35
3.3	Visualization of a single JAT layer which performs a message passing round	37
3.4	Visualization of the attention mechanism	38
4.1	JAT architecture comparison: ANI-1 validation & test set MAE for variations in the architecture	51
4.2	JAT training set scaling laws on ANI-1	53
4.3	JAT ANI-1 scaling comparison of sampling fraction for conformations and molecules	53
4.4	JAT ANI-1 architecture comparison: message passing layers MAE	55
4.5	JAT ANI-1 results for production run using 5% of ANI-1	56
4.6	JAT architecture comparison: EAN validation MAE for variations in the architecture	60
4.7	JAT EAN analysis: impact of randomness from data shuffling and initialization	62
4.8	JAT EAN comparison: validation and test set MAE against NeuralIL baseline	63
4.9	EAN results: Validation and test set MAE and RMSE of the best JAT model	64
4.10	JAT EAN analysis: dissociation of a N-O bond	65
4.11	JAT EAN analysis: training collapse of very large models	66

List of Tables

4.1	JAT architecture comparison: ANI-1 energy MAE, runtime and throughput	52
4.2	JAT training set scaling laws on ANI-1	52
4.3	JAT ANI-1 scaling comparison of sampling fraction for conformations and molecules	54
4.4	JAT ANI-1 architecture comparison: 5, 6 & 7 layers	56
4.5	JAT ANI-1 analysis: stratified heavy molecule dataset statistics	57
4.6	JAT ANI-1 analysis: stratified heavy molecule MAE & RMSE	58
4.7	JAT Architecture comparison: EAN validation and test set MAE & RMSE for variations in the architecture	60
4.8	JAT EAN analysis: impact of randomness from data shuffling and initialization	61
4.9	JAT EAN architecture comparison: validation and test set MAE & RMSE against NeuralIL	64

Bibliography

- [AHK19] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.
- [BAY21] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.
- [BBCV21] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.
- [BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [Beh15] Jörg Behler. Constructing high-dimensional neural network potentials: A tutorial review. *International Journal of Quantum Chemistry*, 115(16):1032–1050, 2015.
- [Beh21] Jorg Behler. Four generations of high-dimensional neural network potentials. *Chemical Reviews*, 2021.
- [BFH⁺20] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. Jax: composable transformations of python+ numpy programs, 2018. <http://github.com/google/jax>, 4:16, 2020.
- [Bie20] Lukas Biewald. Experiment tracking with weights and biases. Software available from <https://wandb.ai>, 2020.
- [BKC13] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

- [BMS⁺21] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. Se (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *arXiv preprint arXiv:2101.03164*, 2021.
- [BP07] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.
- [BPL⁺16] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.
- [Bro20] Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. In *International Conference on Machine Learning*, pages 1144–1152. PMLR, 2020.
- [COJ⁺17] Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.
- [CTS⁺17] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017.
- [CVMBB14] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [DBK⁺20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [DMI⁺15] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.
- [DRA21] Ameya Daigavane, Balaraman Ravindran, and Gaurav Aggarwal. Understanding convolutions on graphs. *Distill*, 2021. <https://distill.pub/2021/understanding-gnns>.
- [ES00] Martyn J Earle and Kenneth R Seddon. Ionic liquids. green solvents for the future. *Pure and applied chemistry*, 72(7):1391–1398, 2000.

- [FHH⁺17] Felix A. Faber, Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S. Schoenholz, George E. Dahl, Oriol Vinyals, Steven Kearnes, Patrick F. Riley, and O. Anatole von Lilienfeld. Prediction errors of molecular machine learning models lower than hybrid dft error. *Journal of Chemical Theory and Computation*, 13(11):5255–5264, Oct 2017.
- [FWFW20] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.
- [FZJS21] Victor Fung, Jiaxin Zhang, Eric Juarez, and Bobby G Sumpter. Benchmarking graph neural networks for materials chemistry. *npj Computational Materials*, 7(1):1–8, 2021.
- [G⁺89] Andreas Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.
- [GGG19] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2019.
- [GKB⁺] Jonathan Godwin, Thomas Keck, Peter Battaglia, Victor Bapst, Thomas Kipf, Yujia Li, Kimberly Stachenfeld, Petar Velickovic, and Alvaro Sanchez-Gonzalez. Jraph: A library for graph neural networks in jax., 2020. <http://github.com/deepmind/jraph>.
- [GSR⁺17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
- [GY21] Weiyi Gong and Qimin Yan. Graph-based deep learning frameworks for molecules and solid-state materials. *Computational Materials Science*, page 110332, 2021.
- [HTP⁺18] Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M Anderson, and Risi Kondor. Predicting molecular properties with covariant compositional networks. *The Journal of chemical physics*, 148(24):241745, 2018.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [JKLR06] Anna Jarosik, Sebastian R Krajewski, Andrzej Lewandowski, and Przemysław Radzinski. Conductivity of ionic liquids in mixtures. *Journal of Molecular Liquids*, 123(1):43–50, 2006.

- [JMTR96] William L Jorgensen, David S Maxwell, and Julian Tirado-Rives. Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids. *Journal of the American Chemical Society*, 118(45):11225–11236, 1996.
- [KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [KMB⁺16] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.
- [KME20] Emir Kocer, Jeremy K Mason, and Hakan Erturk. Continuous and optimally complete description of chemical environments using spherical bessel descriptors. *AIP Advances*, 10(1):015021, 2020.
- [KMHF⁺15] David Krejci, Fernando Mier-Hicks, Corey Fucetola, Paulo Lozano, Andrea Hsu Schouten, and Francois Martel. Design and characterization of a scalable ion electrospray propulsion system. 2015.
- [Koh99] W. Kohn. Nobel lecture: Electronic structure of matter—wave functions and density functionals. *Reviews of Modern Physics*, 71:1253–1266, 1999.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- [LTBZ15] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
- [MBL04] Kenneth N Marsh, John A Boxall, and Ruediger Lichtenthaler. Room temperature ionic liquids and their mixtures—a review. *Fluid phase equilibria*, 219(1):93–98, 2004.
- [MCCB⁺21] Hadrián Montes-Campos, Jesús Carrete, Sebastian Bichelmaier, Luis M Varela, and Georg KH Madsen. A differentiable neural-network force field for ionic liquids. *Journal of chemical information and modeling*, 62(1):88–101, 2021.
- [MDM⁺20] Łukasz Maziarka, Tomasz Danel, Sławomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanisław Jastrzębski. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*, 2020.
- [QRT20] Tyler Quill, Shayta Roy, and Yaakov Tuchman. Predicting ionic liquid materials properties from chemical structure. http://cs230.stanford.edu/projects_winter_2020/reports/32496282.pdf, 2020. Massachusetts Institute of Technology CS230 Coursework.

- [RDRV14] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.
- [RZL17] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
- [SHI⁺16] Thomas P Senftle, Sungwook Hong, Md Mahbubul Islam, Sudhir B Kylasa, Yuanxia Zheng, Yun Kyung Shin, Chad Junkermeier, Roman Engel-Herbert, Michael J Janik, Hasan Metin Aktulga, et al. The reaxff reactive force-field: development, applications and future directions. *npj Computational Materials*, 2(1):1–14, 2016.
- [SIR17a] Justin S. Smith, Olexandr Isayev, and Adrian E. Roitberg. Ani-1, a data set of 20 million calculated off-equilibrium conformations for organic molecules. *Scientific Data*, 4(1), Dec 2017.
- [SIR17b] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.
- [SJPK⁺19] Peter C St. John, Caleb Phillips, Travis W Kemper, A Nolan Wilson, Yanfei Guan, Michael F Crowley, Mark R Nimlos, and Ross E Larsen. Message-passing neural networks for high-throughput polymer screening. *The Journal of chemical physics*, 150(23):234111, 2019.
- [SKSF⁺17] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Saucedo Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.
- [Smi18] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1—learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.
- [SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.
- [TSK⁺18] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.
- [TTOK10] Tsukasa Torimoto, Tetsuya Tsuda, Ken-ichi Okazaki, and Susumu Kuwabata. New frontiers in materials science opened by ionic liquids. *Advanced Materials*, 22(11):1196–1221, 2010.

- [UCS⁺21] Oliver T Unke, Stefan Chmiela, Huziel E Saucedo, Michael Gastegger, Igor Poltavsky, Kristof T Schutt, Alexandre Tkatchenko, and Klaus-Robert Muller. Machine learning force fields. *Chemical Reviews*, 2021.
- [VBK15] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.
- [VCC⁺17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017.
- [VEL⁺19] Vishwesh Venkatraman, Sigvart Evjen, Kallidanthiyil Chellappan Lethesh, Jaganathan Joshua Raj, Hanna K Knuutila, and Anne Fiksdahl. Rapid, comprehensive screening of ionic liquids towards sustainable applications. *Sustainable Energy & Fuels*, 3(10):2798–2808, 2019.
- [VJ20] Nipun Wijerathne Varuna Jayasiri. labml.ai annotated paper implementations. <https://nn.labml.ai/graphs/gatv2>, 2020.
- [vLB20] O Anatole von Lilienfeld and Kieron Burke. Retrospective on a decade of machine learning for chemical discovery. *Nature communications*, 11(1):1–4, 2020.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [Wal14] PI Walden. Ueber die molekulargrosse und elektrische leitfähigkeit einiger geschmolzenen salze. *Известия Российской академии наук. Серия математическая*, 8(6):405–422, 1914.
- [Wei88] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.
- [WK16] Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.
- [WOW⁺19] Jiang Wang, Simon Olsson, Christoph Wehmeyer, Adrià Pérez, Nicholas E Charron, Gianni De Fabritiis, Frank Noé, and Cecilia Clementi. Machine learning of coarse-grained molecular dynamics force fields. *ACS central science*, 5(5):755–767, 2019.
- [XWL⁺19] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery

with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019.

- [ZHSH05] Theresa Julia Zielinski, Erica Harvey, Robert Sweeney, and David M Hanson. Quantum states of atoms and molecules. *Journal of Chemical Education*, 82(12):1880, 2005.
- [ZKR⁺17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.